Université de Montréal

From Specialists to Generalists: Inductive Biases of Deep Learning for Higher Level Cognition

par

Anirudh Goyal

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée en vue de l'obtention du grade de
Philosophiæ Doctor (Ph.D.)
en Discipline

October 17, 2022

© Anirudh Goyal, 2022

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée

From Specialists to Generalists: Inductive Biases of Deep Learning for Higher Level Cognition

présentée par

Anirudh Goyal

a été évaluée par un jury composé des personnes suivantes :

Aaron Courville
(président-rapporteur)

Yoshua Bengio
(directeur de recherche)

Hugo Larochelle
(membre du jury)

Jay McClelland
(examinateur externe)

Ian Charest
(représentant du doyen de la FESP)

Sommaire

Les réseaux de neurones actuels obtiennent des résultats de pointe dans une gamme de domaines problématiques difficiles. Avec suffisamment de données et de calculs, les réseaux de neurones actuels peuvent obtenir des résultats de niveau humain sur presque toutes les tâches. En ce sens, nous avons pu former des *spécialistes* capables d'effectuer très bien une tâche particulière, que ce soit le jeu de Go, jouer à des jeux Atari, manipuler le cube Rubik, mettre des légendes sur des images ou dessiner des images avec des légendes. Le prochain défi pour l'IA est de concevoir des méthodes pour former des *généralistes* qui, lorsqu'ils sont exposés à plusieurs tâches pendant l'entraînement, peuvent s'adapter rapidement à de nouvelles tâches inconnues. Sans aucune hypothèse sur la distribution génératrice de données, il peut ne pas être possible d'obtenir une meilleure généralisation et une meilleure adaptation à de nouvelles tâches (inconnues).

Les réseaux de neurones actuels obtiennent des résultats de pointe dans une gamme de domaines problématiques difficiles. Une possibilité fascinante est que l'intelligence humaine et animale puisse être expliquée par quelques principes, plutôt qu'une encyclopédie de faits. Si tel était le cas, nous pourrions plus facilement à la fois comprendre notre propre intelligence et construire des machines intelligentes. Tout comme en physique, les principes eux-mêmes ne suffiraient pas à prédire le comportement de systèmes complexes comme le cerveau, et des calculs importants pourraient être nécessaires pour simuler l'intelligence humaine. De plus, nous savons que les vrais cerveaux intègrent des connaissances a priori détaillées spécifiques à une tâche qui ne pourraient pas tenir dans une courte liste de principes simples. Nous pensons donc que cette courte liste explique plutôt la capacité des cerveaux à apprendre et à s'adapter efficacement à de nouveaux environnements, ce qui est une grande partie de ce dont nous avons besoin pour l'IA. Si cette hypothèse de simplicité des principes était correcte, cela suggérerait que l'étude du type de biais inductifs (une autre façon de penser aux principes de conception et aux a priori, dans le cas des systèmes d'apprentissage) que les humains et les animaux exploitent pourrait aider à la fois à clarifier ces principes et à fournir source d'inspiration pour la recherche en IA.

L'apprentissage en profondeur exploite déjà plusieurs biais inductifs clés, et mon travail envisage une liste plus large, en se concentrant sur ceux qui concernent principalement le traitement cognitif de niveau supérieur. Mon travail se concentre sur la conception de tels modèles en y incorporant des hypothèses fortes mais générales (biais inductifs) qui permettent un raisonnement de haut niveau sur la structure du monde. Ce programme de recherche est à la fois ambitieux et pratique, produisant des algorithmes concrets ainsi qu'une vision cohérente pour une recherche à long terme vers la généralisation dans un monde complexe et changeant. Mon travail explore les thèmes de:

- **IA cognitivement informée :** Incorporer des informations sur la façon dont les humains traitent les informations visuelles et exploitent la structure du monde dans la conception des architectures d'IA et des méthodes d'apprentissage automatique.
- **Apprentissage de la représentation causale :** Intégration des idées de *causalité* dans des réseaux profonds pour apporter des améliorations à la généralisation hors distribution plus théoriquement fondées et mathématiquement rigoureuses.

Pour passer de spécialistes à généralistes, il est important de réfléchir à la façon dont un agent peut réutiliser, recomposer et recombiner les informations entre les tâches. Un paradigme dominant dans l'apprentissage par renforcement (RL) moderne consiste à apprendre des règles de comportement à usage général à partir des expériences passées de l'agent. Ces règles sont généralement représentées dans les poids synaptiques d'un modèle de réseau de neurones calculant une politique paramétrique ou une fonction de valeur. On peut réaliser un transfert à des tâches nouvelles via la représentation des compétences, via des modèles de la dynamique du système et via des données brutes. Ma thèse explore ces différentes manières de réaliser le transfert en incorporant des biais inductifs généraux pour exploiter la structure du monde.

Dans le premier article, nous montrons comment la décomposition des connaissances en morceaux interchangeables promet un avantage de généralisation lorsqu'il y a des changements dans la distribution. Un agent apprenant interagissant avec son environnement est susceptible d'être confronté à des situations nécessitant de nouvelles combinaisons de connaissances existantes. Nous émettons l'hypothèse qu'une telle décomposition des connaissances est particulièrement pertinente pour pouvoir généraliser de manière systématique aux changements hors distribution. Dans le deuxième article, nous avons proposé une architecture qui synchronise les connaissances entre ces différents modules par rapport à l'utilisation d'interactions par paires dominantes dans les architectures d'apprentissage automatique.

Dans le troisième article, nous proposons un objectif théorique de l'information qui permet d'apprendre des politiques modulaires de manière complètement décentralisée.

Dans le quatrième article, nous explorons un paradigme alternatif dans lequel nous formons un réseau pour mapper un ensemble de données d'expériences passées à un comportement optimal. Plus précisément, nous augmentons un agent RL avec un processus de récupération (paramétré comme un réseau de neurones) qui a un accès direct à un ensemble de données d'expériences. Cet ensemble de données peut provenir des expériences passées de l'agent, de démonstrations d'experts ou de toute autre source pertinente. Le processus de récupération est formé pour récupérer des informations de l'ensemble de données qui peuvent être utiles dans le contexte actuel, pour aider l'agent à atteindre son objectif plus rapidement et plus efficacement.

Mots-clés: Apprentissage en profondeur, traitement du langage naturel, apprentissage des représentations, modèles génératifs, modélisation du langage

Summary

Current neural networks achieve state-of-the-art results across a range of challenging problem domains. Given enough data, and computation, current neural networks can achieve human-level results on mostly any task. In the sense, that we have been able to train *specialists* that can perform a particular task really well whether it's the game of GO, playing Atari games, Rubik's cube manipulation, image caption or drawing images given captions. The next challenge for AI is to devise methods to train *generalists* that when exposed to multiple tasks during training can quickly adapt to new unknown tasks. Without any assumptions about the data generating distribution it may not be possible to achieve better generalization and adaption to new (unknown) tasks.

A fascinating possibility is that human and animal intelligence could be explained by a few principles (rather than an encyclopedia). If that was the case, we could more easily both understand our own intelligence and build intelligent machines. Just like in physics, the principles themselves would not be sufficient to predict the behavior of complex systems like brains, and substantial computation might be needed to simulate human intelligence. In addition, we know that real brains incorporate some detailed task-specific a priori knowledge which could not fit in a short list of simple principles. So we think of that short list rather as explaining the ability of brains to learn and adapt efficiently to new environments, which is a great part of what we need for AI. If that simplicity of principles hypothesis was correct it would suggest that studying the kind of inductive biases (another way to think about principles of design and priors, in the case of learning systems) that humans and animals exploit could help both clarify these principles and provide inspiration for AI research.

Deep learning already exploits several key inductive biases, and my work considers a larger list, focusing on those which concern mostly higher-level cognitive processing. My work focuses on designing such models by incorporating in them strong but general assumptions (inductive biases) that enable high-level reasoning about the structure of the world. This research program is both ambitious and practical, yielding concrete algorithms as well as a cohesive vision for long-term research towards generalization in a complex and changing world. My work explores the topics of:

- **Cognitively Informed AI:** Incorporating insights as to how humans process visual information and exploit the structure of the world into the design of AI architectures and machine learning methods.
- **Causal Representation Learning:** Integrating ideas from *causality* into deep networks to make improvements in out-of-distribution generalization more theoretically grounded and mathematically rigorous.

In order to make a transition from specialists to generalists, it's important to think about how an agent reuse, recompose and recombine information across tasks. A dominant paradigm

in modern reinforcement learning (RL) is to learn general purpose behaviour rules from the agent's past experiences. These rules are typically represented in the weights of a parametric policy or value function network model. One can achieve transfer via representation of skills, through models of the system dynamics and through raw data. My thesis explore these different ways to achieve transfer by incorporating general inductive biases for exploiting the structure of the world.

In the first article, we show how decomposing knowledge into interchangeable pieces promises a generalization advantage when there are changes in distribution. A learning agent interacting with its environment is likely to be faced with situations requiring novel combinations of existing pieces of knowledge. We hypothesize that such a decomposition of knowledge is particularly relevant for being able to generalize in a systematic manner to out-of-distribution changes. In the second article, we proposed an architecture which synchronizes knowledge among these different modules as compared to using pair-wise interactions dominant in machine learning architectures.

In the third article, we propose an information theoretic objective that allows one to learn modular policies in a completely decentralized fashion.

In the fourth article, we explore an alternative paradigm in which we train a network to map a dataset of past experiences to optimal behavior. Specifically, we augment an RL agent with a retrieval process (parameterized as a neural network) that has direct access to a dataset of experiences. This dataset can come from the agent's past experiences, expert demonstrations, or any other relevant source. The retrieval process is trained to retrieve information from the dataset that may be useful in the current context, to help the agent achieve its goal faster and more efficiently.

Keywords: Deep Learning, Natural Language Processing, Representation Learning, Generative Models, Language Modeling

Contents

List of tables

List of figures

List of Acronyms and Abbreviations

BPTT	Backpropagation Through Time
CNN	Convolutional Neural Network
ERM	Empirical Risk Minimization
GAN	Generative Adversarial Network
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
MLE	Maximum Likelihood Estimation
MLP	Multi-Layer Perceptron
NLP	Natural Language Processing
NMT	Neural Machine Translation
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
BPTT	Backpropagation Through Time

VAE	Variational Autoencoder

To my grandparents!

Acknowledgements

Although the title page of this document may identify me as the author, I cannot help but consider it a blatant misrepresentation of where the credit is actually due.

I'm grateful to Prof. Yoshua Bengio for accepting me as a graduate student. At first, I was selected as a visiting intern at Mila during Spring 2016. At that time, it was not obvious to me why he selected me as I did not know the answer to any of the questions he asked me during my interview. I was really just happy that I got the chance. I am glad that I started my journey at Mila in 2016, it would have been very difficult to get to know Prof. Yoshua Bengio and learn from him if I had started later. From the day 1, he encouraged me to have the right intuitions, and approaching the problem from different perspectives. His guidance has been very important in illuminating the importance of random exploration for research, and in my understanding of how brains do credit assignment through time. He always knew what are the good research questions to think about. His ability to give meaningful feedback on a wide range of research topics is impressive. He has helped me not only to become a better researcher but also a better human-being. His ability to stay calm in any situation is a superpower, and something which I have improved a lot over the years. I have realized the importance of being both reliable and reflective from Yoshua Bengio. I hope to strengthen these two qualities for many more years to come. I would consider myself successful if in 30 years from now, my future self retains the same enthusiasm and excitement towards things my future self may like as present in Prof. Yoshua Bengio.

I would like to thank Nan Rosemary Ke and Alex Lamb, the subset of my friends and collaborators with whom I have worked the most closely in recent years. None of this work would have been even a remote possibility if not for their contributions to our joint research, as well as their continual support and guidance. We have fought over all the possible things one could think of, and I am grateful that we chose to support and uplift each other. Alex's excitement and capability to have a discussion on a wide range of topics is intriguing. Rosemary's enthusiasm of always improving herself is inspiring. It's interesting how I met Alex and Rosemary. I was sitting in the lab alone, and Alex Lamb asked me if I wanted to brainstorm about research. This is how I wrote my first paper (Professor Forcing). During Christmas holidays in Dec 2015, I was sitting in the lab (alone), and Nan Rosemary Ke asked me if I want to join her and others for Christmas dinner. I hope you both achieve success and happiness in life.

I'm grateful to Prof. Mike Mozer. Much of the research presented in this thesis would not have been possible without our continued interactions. I have learned many life and science lessons from him. He has helped me to express my thoughts in a better way. I am thankful to him for the trust and time he invested in me.

I got the chance to visit Prof. Bernhard Scholkopf in Tuebingen. I really enjoyed our interactions, as well as late night movie sessions. I was intrigued by his ICML 2017 talk which led me to explore knowledge factorization and its connections to causality. I'm grateful to his time during my visit. Thanks to Nasim Rahaman for making my stay pleasant.

I'm thankful to Prof. Sergey Levine for hosting me as a visiting student at UC Berkeley in 2018. I've learned a lot from him. His ability to provide meaningful feedback on any topic is astounding. I learned from him the importance of thinking important research questions, as well as how to pitch your research in a way which resonates with the readers. Collaborations with him provided me much needed confidence as well as breakthoughs in terms of my attitude towards research.

I'd like to thank Hugo Larochelle for all the things he does. I've "learned" much of the Deep Learning concepts from his YouTube lectures. I'm also grateful for his time and for his assistance in various different projects. I admire his quality of being able to uplift junior members of the research community.

I'd like to thank David Silver for taking a chance on me during my time at DeepMind. Interactions with him were the highlight of my week. During our interactions, his ability to "summarize" my thoughts better than myself was a learning experience. His excitement (and research work) towards making progress on some of the most difficult problems is very inspiring. In future, I hope to make progress on the problems he cares about.

I would like to say thanks to Charles Blundell for making my stay at DeepMind possible and helping me deal with difficult situations. He gave me the freedom to focus on important research problems, and assisted me in making important life decisions. I appreciate his help and time.

I would like to thank Nicolas Heess for continued discussions and collaborations. Our research interests are closely aligned. I appreciate his feedback and advice.

I am grateful for help and support from Stefan Bauer. Collaborations with him helped me figure out interesting research questions, and seek answers to those questions. I learned to appreciate the art of making coffee from him.

During my time at Mila, I've got ample opportunities to *assist* many undergrads, masters and junior PhD students. I've had the pleasure to *assist* Aniket Didolkar, Vedant Shah, Nikhil Barhate, Frederik Träuble, Mihir Prabhudesai, Soumye Singhal, Samarth Sinha, Sarthak Mittal, Phanideep Gampa, Shagun Sodhani, Ossama Ahmed, Sumukh Aithal, Kartikeya Badola, Kanika Madan, Yashas Annadani, Kshitij Gupta, Riashat Islam, Rithesh Kumar. I would like to say thanks to Dianbo Liu for engaging in various collaborations with me.

To my many friends, co-authors, co-workers, and colleagues (random order): Aaron C. Courville, Abhimanyu Dubey, Ahmed Touati, Alex Pentland, Alexander Neitz, Alexandre Lacoste, Adriana Romero, Amanpreet Singh, Amir Hosein Khasahmadi, Augustus Odena, Alessandro Sordoni, Amjad Almahairi, Asja Fischer, Anna Huang, Ahmed Touati, Akram Erraqabi, Arnaud Bergeron, Bart van Merrienboer, Ben Poole, Benjamin Scellier, Bhargav Kanuparthi, Caglar Gulcehre, Chen Sun, Chien-Feng Liao, Christopher Joseph Pal, Colin Raffel, Cesar Laurent, Daniel Jiwoong Im, Devi Parikh, Dhaval Adjodah, Dhruv Batra, Di He, DJ Strouse, David Krueger, David Warde-Farley, Devon Hjelm, Dmitriy Serdyuk, Dzmitry Bahdanau, Elliot Creager, Esteban Moro, Florian Shkurti, Francesco Locatello,

Faruk Ahmed, Francesco Visin, Frederic Bastien, Gabriel Huang, Giancarlo Kerg, Gaetan Marceau Caron, Guillaume Alain, Guillaume Dumas, Guillaume Lajoie, Han Zhang, Homanga Bharadhwaj, Harm de Vries, Hongyu Zang, Ioannis Mitliagkas, Ishmael Belghazi, Iulian Vlad Serban, Joao Felipe Santos, Jonathan Binas, Jian Tang, Joelle Pineau, Jonas Rothfuss, Jorg Bornschein, Jordan Hoffmann, János Kramár, Jose Sotelo, Junyoung Chung, Joseph Paul Cohen, Krishna Kumar Singh, Kelvin Xu, Kenji Kawaguchi, Kristy Choi, Kyle Goyette, Laurent Charlin, Manuel Wuthrich, Muhammad Waleed Gondal, Marius Leordeanu, Marzyeh Ghassemi, Marcin Moczulski, Mohak Sukhwani, Mathieu Germain, Mehdi Mirza, Mohammad Pezeshski, Myriam Cote, Nasim Rahaman, Nicolas Ballas, Negar Rostamzadeh, Olexa Bilaniuk, Oussama Boussif, Pascal Lamblin, Pierre-Luc Carrier, Philemon Brakel, Philippe Beaudoin, Pritish Mohapatra, Remi Tachet des Combes, Romain Laroche, Ryan Lowe, Saeid Asgari, Saizheng Zhang, Samuele Bolotta, Shawn Tan, Sandeep Subramanian, Silvia Chiappa, Sebastien Lachapelle, Simon Lefrancois, Suriya Singh, Sina Honari, Surya Ganguli, Soroush Mehri, Sungjin Ahn, Tim Cooijmans, Tegan Maharaj, Tianmin Shu, Tikeng Notsawo Pascal, Tristan Deleu, Vedant Shah, Vincent Dumoulin, William Fedus, Xin Li, Xu Ji, Xue Bin Peng, Yash Sharma, Yashas Annadani, Guillaume Dumas and Zafarali Ahmed; Thank you for everything you do.

I would like to thanks to Timothy Lillicrap for our discussions throughout my stay at DeepMind. I'm grateful that I got the chance to collaborate with Ivo Danihelka. Work at DeepMind would not have been possible without his help. He patiently listened to my thoughts and answered all my questions. He helped me in improving my coding skills by giving thorough feedback throughout my stay at DeepMind. I would also like to say thanks to Theophane Weber, Abe Friesen, Andrea Banino, Adria Puigdomenech Badia, Arthur Guez, Mehdi Mirza, Peter C. Humphreys, Ksenia Konyushkova, Laurent Sifre, Michal Valko, Simon Osindero and Murray Shanahan for engaging in collaboration with me which paved a path towards the paper which plays a crucial part in this thesis. I would also like to say thanks to Daan Wiestra, Oriol Vinyals, Ivo Danihelka, Danilo Jimenez Rezende, Hado Van Hasselt, Matthew Botvinick, Thore Graepal, Martin Riedmiller, Felix Hill, Pablo Sprechman, Alexander Lerchner, Jane Wang, Rui Zhu, Doina Precup, Fabio Viola, Laurent Sifre, Dan Horgan, Jessica Hamrick, Max Jaderberg, Pushmeet Kohli and Andre Barreto for useful discussions.

Thanks to Celine Begin for all her patience during the thesis submission process and for her organizational help during my studies. Thanks to Julie Mongeau for taking time and arranging my meetings with Prof. Yoshua Bengio. Thanks to Linda Peinthiere for helping me throughout my time at Mila.

I would like to thank my undergrad research advisor Prof. C.V Jawahar, for accepting me as a research assitant, and teaching me how to do research. Thanks to Dr. Shailesh Kumar, for advising me during my final year of the undergrad. He encouraged me to think more, thanks to all his brainstorming sessions. I would like to thank Dr. Rahul Sukthankar, for showing faith in me when I was most uncertain about my future. Without him, I would not have been able to come to Mila. Thanks to Dr. Ray Kurzweil for giving me a chance to intern in his team. He fueled up my curiosity to understand how the brain works. I would also like to say thanks to all my friends in undergrad at IIIT Hyderabad namely Anubhav

Gupta, Kshitij Kansal, Mayank Gupta, Shubham Sangal, Venkatesh Jamula, Nehal J Wani, Ankush Jain, Deepesh Jain, Ashutosh Borkar, Ashutosh Singla, Ayush Lodha, Parth Kapadia, Taradheesh Bali, Rishabh Raj, Soumya Jain, Tushant Jha, Monam Agarwal, Surya Singh, Mohak Sukhwani, Rohit Girdhar, Rama Krishna Vedantam, Krishna Kumar Singh, Smit Hinsu, Nadeem Moidu. Thanks to all of you for acting as a stimulant to pursue graduate life.

There are lots of researchers whom I've not collaborated with, but seeing them achieve success in their academic life has inspired me to do better. I would like to thank Abhishek Gupta, Kevin Ellis, Pulkit Agrawal, Deepak Pathak, Chelsea Finn, Simon Lacoste Julien, Ben Poole, Pieter Abbeel, Jascha Sohl Dickstein, Animesh Garg, Klauss Greff, Julian Schrittwieser, Ilya Sutskever, Ian Goodfellow, Durk Kingma, Jonathan Ho, Katerina Fragkiadaki, Danijar Hafner, Taco Cohen, Behnam Neyshabur, Michael Chang, Shane Gu, Chris Maddison, Danilo Rezende and Jacob Andreas.

I'd also like to thank John Langford, Michael Rabbat, Jonathan Shlens, Hugo Larochelle, Animesh Garg, Katerina Fragkiadaki, Deepak Pathak, Pulkit Agrawal, Murray Shanahan, Charles Blundell, Abhishek Gupta, recruiters and department chairs for helping me navigate through the transition between finishing my Ph.D and what to do after it.

I would like to thank various different coffee shops and restaurants in Montreal namely AuntDai, Allo Mon Coco, Nouilles de Lan Zhou, Faberge, Ferlucci, L'Avenue, Banh Mi Saigon, Cafe Noble, Regina Cafe, Cafe Italia, Cafe Vito, Cafe Pista, Baristello, Cafelleni, Oui Mais Non, Le Toasteur, Nosthes, Leméac, Darbar for providing great atmosphere which acted as a stimulant for my thesis. Special mention to all the bubble tea places near University of Berkeley Campus.

The work reported in this thesis would not have been possible without the financial support from: Ubisoft, Google, Samsung, IBM, NSERC, Calcul Quebec, Compute Canada, the Canada Research Chairs and CIFAR.

Thanks to many different (unknown to me) reviewers for taking time and reviewing the work present in this thesis. I appreciate your feedback and help.

Finally, I would like to thank my grandparents, parents, brother and other members of my family. This thesis is dedicated to my grandparents.

Chapter 1

Introduction

1.1. Introduction

Is 100% accuracy on the test set enough? Many machine learning systems have achieved excellent accuracy across a variety of tasks (Deng et al., 2009; Mnih et al., 2013; Schrittwieser et al., 2019), yet the question of whether their reasoning or judgement is correct has come under question, and answers seem to be wildly inconsistent, depending on the task, architecture, training data, and interestingly, the extent to which test conditions match the training distribution. Have the main principles required for deep learning to achieve human-level performance been discovered, with the main remaining obstacle being to scale up? Or do we need to follow a completely different research direction not built on the principles discovered with deep learning, in order to achieve the kind of cognitive competence displayed by humans? Our goal here is to better understand the gap between current deep learning and human cognitive abilities so as to help answer these questions and suggest research directions for deep learning with the aim of bridging the gap towards human-level AI. Our main hypothesis is that deep learning succeeded in part because of a set of inductive biases (preferences, priors or assumptions), but that additional ones should be included in order to go from good in-distribution generalization in highly supervised learning tasks (or where strong and dense rewards are available), such as object recognition in images, to strong out-of-distribution generalization and transfer learning to new tasks with low sample complexity (few examples needed to generalize well). To make that concrete, we consider some of the inductive biases humans may exploit in conscious thought using highly sequential cognition operating at the level of conscious processing, and review some early work exploring these "high-level cognitive inductive priors" in deep learning. We use the term *high-level* to talk about variables that are manipulated at the conscious level of processing and are thus generally verbalizable. However, humans can consciously focus attention on low-level or intermediate-level features, e.g., by describing an odd-coloured pixel, not just very abstract concepts like objects or social situations. We argue that the deep learning progression from MLPs to convnets to transformers has in many ways been an (incomplete) progression towards the original goals of deep learning, i.e., to enable the discovery of a hierarchy of representations, with the most abstract ones, often associated with language, at the top. Note however, that although language may give us a view on system 2, these abilities are likely to pre-exist language as there is evidence of surprisingly strong forms of on-the-fly reasoning in some non-human animals, like corvids (Taylor et al., 2009). Our arguments suggest that while deep learning brought remarkable progress, it needs to be extended in qualitative and not just quantitative

ways: larger and more diverse datasets and more computing resources (Brown et al., 2020) are important but insufficient without additional inductive biases (Vaswani et al., 2017; He et al., 2016; Gilmer et al., 2017; Shazeer et al., 2017; Fedus et al., 2021a; Hinton, 2021; Welling, 2019; Dosovitskiy et al., 2020; Battaglia et al., 2018). We make the case that evolutionary forces, the interactions between multiple agents, the non-stationary and competition systems put pressure on the learning machinery to achieve the kind of flexibility, robustness and ability to adapt quickly which humans seem to have when they are faced with new environments (Bansal et al., 2017; Liu et al., 2019; Baker et al., 2019; Leibo et al., 2019) but needs to be improved with deep learning. The sought-after inductive biases should thus especially help AI to progress on these fronts. In addition to thinking about the learning and sample complexity advantage of these inductive biases, this paper links them with knowledge representation in neural networks, with the idea that by decomposing knowledge into its stable parts (like causal mechanisms) and volatile parts (random variables), and factorizing knowledge in small and somewhat independent pieces that can be recomposed dynamically as needed (to reason, imagine or explain at an *explicit* and verbalizable level), one may achieve the kind of systematic generalization which humans enjoy and is common in natural language (Marcus, 1998, 2019; Lake & Baroni, 2017; Bahdanau et al., 2018; McClelland et al., 1987).

This thesis is based on the following publications:

Knowledge Decomposition and Modularity

- Inductive biases od deep learning for higher level cognition: **Anirudh Goyal**, Yoshua Bengio, arXiv:2011.15091.
- Recurrent Independent Mechanisms: **Anirudh Goyal**, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Bernhard Schölkopf, Yoshua Bengio, arXiv:1909.10893 **(ICLR'21)**.
- Object Files and Schemata: Factorizing Declarative and Procedural Knowledge in Dynamical Systems: **Anirudh Goyal**, Alex Lamb, Phanideep Gampa, Philippe Beaudoin, Sergey Levine, Charles Blundell, Yoshua Bengio, Michael Mozer, arXiv:2006.16225 **(ICLR'21)**.
- Learning to Combine Top-Down and Bottom-Up Signals in Recurrent Neural Networks with Attention over Modules: Sarthak Mittal, Alex Lamb, **Anirudh Goyal**, Vikram Voleti, Murray Shanahan, Guillaume Lajoie, Michael Mozer, Yoshua Bengio, **(ICML'20)**.
- Neural Production Systems: Learning Rule-Governed Visual Dynamics: **Anirudh Goyal**, Aniket Didolkar, Nan Rosemary Ke, Charles Blundell, Philippe Beaudoin, Nicolas Heess, Michael Mozer, Yoshua Bengio, arXiv:2103.01937 **(NeurIPS'21)**.
- Coordination Among Neural Modules Through a Shared Global Workspace: **Anirudh Goyal**, Aniket Didolkar, Alex Lamb, Kartikeya Badola, Nan Rosemary Ke, Nasim Rahaman, Jonathan Binas, Charles Blundell, Michael Mozer, Yoshua Bengio, arXiv:2103.01197 **(ICLR'22)**.
- Discrete Valued Neural Communication: Dianbo Liu, Alex Lamb, Kenji Kawaguchi, **Anirudh Goyal**, Chen Sun, Michael Curtis Mozer, Yoshua Bengio, arXiv:2107.02367 **(NeurIPS'21)**.
- Fast and slow learning of Recurrent Independent Mechanisms: Kanika Madan, Nan Rosemary Ke, **Anirudh Goyal**, Bernhard Schölkopf, Yoshua Bengio, arXiv:2105.08710 **(ICLR'21)**.

- Sparse Attentive Backtracking: Temporal Credit Assignment Through Reminding: Nan Rosemary Ke, **Anirudh Goyal**, Olexa Bilaniuk, Jonathan Binas, Michael C. Mozer, Chris Pal, Yoshua Bengio, arXiv:1809.03702 **(NeurIPS'17)**.

Deep Learning and Causality

- A Meta-Transfer Objective for Learning to Disentangle Causal Mechanisms: Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Rosemary Ke, Sébastien Lachapelle, Olexa Bilaniuk, **Anirudh Goyal**, Christopher Pal, arXiv: 1901.10912 **(ICLR'20)**.
- Learning Neural Causal Models from Unknown Interventions: Nan Rosemary Ke, Olexa Bilaniuk, **Anirudh Goyal**, Stefan Bauer, Hugo Larochelle, Chris Pal, Yoshua Bengio, arXiv:1910.01075.
- Systematic Evaluation of Causal Discovery in Visual Model Based Reinforcement Learning: Nan Rosemary Ke, Aniket Didolkar, Sarthak Mittal, **Anirudh Goyal**, Guillaume Lajoie, Stefan Bauer, Danilo Rezende, Yoshua Bengio, Michael Mozer, Christopher Pal, arXiv:2107.00848 **(NeurIPS'21)**.
- CausalWorld: A Robotic Manipulation Benchmark for Causal Structure and Transfer Learning: Ossama Ahmed, Frederik Träuble, **Anirudh Goyal**, Alexander Neitz, Yoshua Bengio, Bernhard Schölkopf, Manuel Wüthrich, Stefan Bauer, arXiv:2010.04296 **(ICLR'21)**.

Information bottleneck and Reinforcement Learning

- InfoBot: Transfer and Exploration via the Information Bottleneck: **Anirudh Goyal**, Riashat Islam, Daniel Strouse, Zafarali Ahmed, Matthew Botvinick, Hugo Larochelle, Sergey Levine, Yoshua Bengio, arXiv:1901.10902 **(ICLR'19)**.
- The Variational Bandwidth Bottleneck: Stochastic Evaluation on an Information Budget: **Anirudh Goyal**, Yoshua Bengio, Matthew Botvinick, Sergey Levine, arXiv: 2004.11935 **(ICLR'20)**.
- Reinforcement Learning with Competitive Ensembles of Information-Constrained Primitives: **Anirudh Goyal**, Shagun Sodhani, Jonathan Binas, Xue Bin Peng, Sergey Levine, Yoshua Bengio, arXiv:1906.10667 **(ICLR'20)**.
- Retrieval-Augmented Reinforcement Learning: **Anirudh Goyal**, Abram L. Friesen, Andrea Banino, Theophane Weber, Nan Rosemary Ke, Adria Puigdomenech Badia, Arthur Guez, Mehdi Mirza, Peter C. Humphreys, Ksenia Konyushkova, Laurent Sifre, Michal Valko, Simon Osindero, Timothy Lillicrap, Nicolas Heess, Charles Blundell, arXiv:2202.08417 **(ICML'22)**.

1.2. Thesis Overview

The thesis contains 4 representative articles from the above list that focus on knowledge factorization and generalization in deep learning and deep reinforcement learning. This thesis is organized in the following unusual way, due to the large number of papers to be covered. The first substantive part of this document (chapter 2) contains a review of my proposed research area (inductive biases for deep learning of higher level cognition), an overview of the main contributions of the above papers, as well as a vision of proposed future research. It is written as a position paper which could stand on its own and a version of it is posted on arXiv (co-authored by Yoshua Bengio and myself) and accepted for publication. The

next four chapters are four of the above papers whose contributions we believe to be more significant and central to the thesis, going in more details in some of the key ideas and have the more usual format of a machine learning conference paper.

Chapter 5 presents an architectural bias for promoting systematic generalization by factorizing knowledge into variables and rules, such that one can reuse information about rules across variables as long as type of variables matches the type of input a particular rule expects.

Chapter 9 explores the use of information bottleneck for learning policy primitives in a completely decentralized fashion.

Chapter 11 presents an architecture where RL agent is augmented with the retrieval process parameterized as a neural network. The goal of the retrieval process is to provide contextual information relevant to the RL agent from a large database.

Chapter 12 presents an overall conclusion of the contributions in this thesis.

1.3. Probabilistic Machine Learning

Machine learning systems are able to make predictions or produce actions by automatically learning from data and experience. Mitchell et al. (1997) define a learning machine as an algorithm that improves with experience on a particular task, given a certain performance measure to evaluate the system. In this thesis, we are interested in questions of how machine learning systems can generalize either when trained with limited data or limited task diversity.

We often assume the ability to independently sample a certain finite number of datapoints from the same unknown data distribution. Some of that data is partitioned for our systems to learn on and the rest is reserved for testing how it generalizes on unseen examples. Generalizing to data drawn from the same distribution is a fairly common evaluation setup used in most machine learning systems. Most theoretical guarantees on the generalization properties of machine learning systems have required this assumption. This is however being questioned more as our systems do not generalize systematically and perform poorly when dealing with out-of-distribution data. For the sake of this introduction however, we will focus on the case where our data is independent and identically distributed (iid), which means that both training data and test cases are assumed to come from the same distribution.

1.3.1. Supervised Learning

Supervised Learning serves as the fundamental framework for most of the work in this thesis and much of machine learning research as a whole. It consists of learning to predict output targets from inputs given a labeled dataset of input/output pairs, where all pairs are assumed to be sampled uniformly from an unknown distribution. An appealing aspect of supervised learning is that the learned model's accuracy on held-out samples is an objective measure of success. Additionally, the correct outputs for the model are provided in the dataset, which makes credit assignment relatively straightforward. Compare this to playing a video game, where the correct actions are never given to the player.

Some examples of supervised learning include image classification (where the image is the input and the class label is the output) and speech recognition (where the speech signal is the input and the corresponding text is the output).

The goal of supervised learning is to learn a function f_θ which maps an example's input to its output. These functions f_θ map a d-dimensional, typically real-valued vector input $\mathbf{x} \in \mathbb{R}^d$ from the space of possible inputs X to a discrete scalar output label y from the space of k

possible outputs Y. We assume our data comes from an unknown joint probability distribution over inputs and outputs $p(\mathbf{x}, y)$ where $\mathbf{x} \in X$ and $y \in Y$ and we would like to learn a function $f_\theta : \mathbb{R}^d \to \mathbb{R}^k$ that maps X to an estimate of some statistic of $P(Y|X)$, like the probability itself or $E[Y|X]$. We observe examples from this distribution $((\mathbf{x_1}, y_1) \dots (\mathbf{x_N}, y_N))$. The subscript θ is used to indicate that our function is parameterized. This doesn't always have to be the case, and there exist "non-parameteric" models in machine learning, but in the context of this thesis that focuses on neural networks, we are interested in learning parametric functions. There are many possible functions that can map from X to a distribution over Y, but we are often interested in learning one that makes as few mistakes as possible. We define a loss function $L(\hat{y}, y)$ that measures the penalty incurred when classifying y as $\hat{y}$ and we try to optimize f_θ to minimize this loss given a finite amount of data. We can use this loss to define the empirical risk associated with a particular function as the expected loss over the entire training set $R_{emp}(X, Y, f_\theta) = \frac{1}{N} \sum_{i=1}^{N} L(f_\theta(\mathbf{x_i}), y_i)$. Empirical Risk Minimization (ERM) tries to find $\arg\min_{f \in F} R_{emp}(f)$. However, for a given dataset, there may be many functions $f \in F$ in the hypothesis space, that achieve the same or even 0 empirical risk on the training data subset, but generalize poorly to unseen examples. In such cases, we want to bias our learners to certain solutions that may have better generalization properties such as those with low parameter norms with L1/L2 regularization, with max-margin methods (**?**), or by using dropout (Srivastava et al., 2014).

In the case where we want to do probabilistic classification, $f_\theta(\mathbf{x})$ outputs a probability distribution over possible class labels and the loss function used is typically the negative log-likelihood of the correct class $-\log q_\theta(Y = y|\mathbf{x}) = -\log f_{\theta,y}((x))$, where $f_{\theta,y}()$ is the y-th output element of the vector $f_\theta(\mathbf{x})$ containing all the output probabilities. Intuitively, we wish to maximize the probability the model assigns to the correct class, and the use of the log causes a very high loss when a small probability is assigned to the correct class. The frequentist approach to parameter estimation termed Maximum Likelihood Estimation (MLE) seeks a setting of parameters that maximizes the likelihood of the observed data. This decomposes into a product over the examples in the case where their conditional distributions are independent and identically distributed (the i.i.d. assumption): $q_\theta(Y|X) = \prod_{i=1}^{N} q_\theta(y_i|\mathbf{x_i})$. Since the logarithm is a monotonically increasing function, we can maximize the $\log q_\theta(y_1, y_2, \dots, y_N|\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$ and break the product into a bunch of sums $\log q_\theta(y_1, y_2, \dots, y_N|\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) = \sum_{i=1}^{N} \log q_\theta(y_i|\mathbf{x_i})$. To turn it into a minimization problem, we can minimize the negative log-likelihood of the data, showing that empirical risk minimization (ERM) with the negative log-likelihood loss corresponds to maximum-likelihood estimation of the parameters of the model. At inference/test time, if we want to choose one of the labels, the model would output $\arg\max_{\tilde{y}} q_\theta(\tilde{y}|\mathbf{x}_i)$. However, the output probability distribution also gives us other information, such as the uncertainty in the output, which can be represented by the entropy of $q_\theta(y_i|\mathbf{x_i})$.

1.3.2. Distributions and Likelihood

So far, we have discussed generative modeling in qualitative terms. We want models which can simulate from the dynamics of the world. We want models that can synthesize realistic looking data. However, before going further it is useful to understand the probabilistic

interpretation of generative models, which gives a formal framework for studying generative models. The essential idea is that we treat observations from the world as samples from a distribution $x, y \sim p(x, y)$. For example, we could consider the distribution over all human faces which can occur in reality to be $p(x)$ and consider each face as a sample with an associated label $y \sim p(y|x)$. If we have access to a recorded dataset (for example a set of faces), we may also choose to treat these points as a finite collection of samples from this distribution.

At the same time, we can interpret our generative model as an estimating distribution $q_\theta(y|x)$, which is described by a set of parameters θ. Then we can frame generative modeling as trying to ensure that $p(y|x)$ and $q_\theta(y|x)$ become as similar as possible. Statistical divergences give a natural mathematical framework for this. A divergence is a function $D(p||q) : S \times S \to R$ taking two distributions p and q over a space of distributions S as inputs, with the properties:

$$D(p||q) \geq 0.0 \tag{1.1}$$

$$D(p||q) = 0.0 \iff p = q \tag{1.2}$$

Notably, there is no symmetry assumption, so in general $D(p||q) \neq D(q||p)$. The probabilistic approach to generative modeling frames learning as an optimization problem where the loss corresponds to a given divergence.

$$\mathcal{L}(\theta) = \operatorname*{argmin}_\theta D(p(y|x)||q_\theta(y|x)). \tag{1.3}$$

1.3.3. Maximum Likelihood and KL-Divergence

We will now present the classical result showing how minimizing KL-divergence can be applied to the supervised learning setting introduced previously. In particular, we assume the ability to sample from $p(x, y)$ and wish to learn $q_\theta(y|x)$.

What is the right algorithm for finding a distribution $q_\theta(y|x)$ which minimizes a divergence between itself and $p(y|x)$? Before selecting the type of divergence to minimize, a natural question is to consider what types of expressions we are capable of optimizing, and work backwards to find a suitable divergence. In general, we only have access to samples from the distribution $p(x, y)$ and not any additional information about the distribution. At the same time, $q_\theta(y|x)$ is a model that we control, so it's reasonable to believe that we'll be able to design it so that it has a density that we can compute as well as the ability to draw samples.

The KL-divergence can be rewritten as an expression in which the only term that depends on the parameters is an expectation involving $q_\theta(y|x)$ over samples from $p(x, y)$. Beginning with two distributions $p(x, y)$ (the empirical distribution) and $q_\theta(y|x)$ (the model distribution), we write the KL-divergence:

$$D_{KL}(p(y|x)||q_\theta(y|x)) = \mathop{\mathbb{E}}_{x\sim p(x)}\left[\int p(y|x)\log p(y|x)dy - \int p(y|x)\log q_\theta(y|x)dy\right] \tag{1.4}$$

$$= \mathop{\mathbb{E}}_{x\sim p(x)}\int p(y|x)\log\frac{p(y|x)}{q_\theta(y|x)}dy \tag{1.5}$$

$$= \mathop{\mathbb{E}}_{x,y\sim p(x,y)}\left[\log\frac{p(y|x)}{q_\theta(y|x)}\right] \tag{1.6}$$

$$= \mathop{\mathbb{E}}_{x,y\sim p(x,y)}[\log p(y|x) - \log q_\theta(y|x)] \tag{1.7}$$

$$= H_p(y|x) - \mathop{\mathbb{E}}_{x,y\sim p(x,y)}[\log q_\theta(y|x)] \tag{1.8}$$

Then we can show the maximum likelihood estimation for a set of N data points.

$$\theta^* = \arg\max_\theta \prod_{i=1}^{N} q_\theta(y_i|x_i) \tag{1.9}$$

$$= \arg\max_\theta \sum_{i=1}^{N} \log q_\theta(y_i|x_i) \tag{1.10}$$

$$= \arg\min_\theta \frac{1}{N}\sum_{i=1}^{N} -\log q_\theta(y_i|x_i) \tag{1.11}$$

$$\sim= \arg\min_\theta \mathop{\mathbb{E}}_{x_i,y_i\sim p(x,y)}[-\log q_\theta(y|x)] \tag{1.12}$$

The objective for maximum likelihood is to maximize the log-density $\log(q_\theta(y|x))$ over real data points sampled from the distribution $p(x, y)$. It can be written as:

$\mathrm{D}_{KL}(p(y|x)||q(y|x)) = \mathbb{E}_{x\sim p(x)}\left[\int p(y|x)\log p(y|x)dy - \int p(y|x)\log q_\theta(y|x)dy\right]$
$= -H(p(y|x)) + \mathrm{CE}(p(y|x), q_\theta(y|x))$

Thus we can see that the KL-divergence decomposes into two terms in Eq. 2.3.3: a cross-entropy term (likelihood) and a term for the entropy of the true data distribution. Because the entropy of the true data distribution doesn't depend on the estimator, the KL-divergence can be minimized by maximizing likelihood. Another useful consequence of this is that the entropy of the true data distribution can be estimated by such a generative model if it maximizes likelihood among all possible distributions, i.e., $q_\theta = p$, the KL-divergence is 0 and the cross-entropy equals the entropy.

1.3.4. ERM and probabilistic framing of the learning problem

The empirical risk minimization objective corresponds to minimizing a loss function of interest over a set of examples x with labels y sampled from a generally unknown distribution $p(x, y)$

$$\theta^* = \arg\min_{\theta} \sum_{x,y \sim p(x,y)} L(y, f(x)) \tag{1.13}$$

in the hope of obtaining a function with low expected loss on examples from the same distribution. A special case of empirical risk minimization is the maximum likelihood objective. Essentially, it can be shown that maximizing a model's likelihood on some data distribution is equivalent to minimizing an expectation of loss over samples from the empirical data distribution (which is non-zero only on the data points), which can then be empirically estimated with a sum over examples. The objective for maximum likelihood is maximizing the log-density $log(q_\theta(x))$ over real data points sampled from the distribution $p(x)$. The use of maximum likelihood training for generative models (such as sequence models) is an unconditional generalization of the use of conditional maximum likelihood for supervised learning.

1.3.5. Teacher Forcing

A specific variant of the maximum likelihood principle can be used for training sequence models. This form of training is known as teacher forcing (Williams & Zipser, 1989), due to the use of the ground-truth samples y_t being fed back into the model to be conditioned on for the prediction of later outputs. These fed back samples force the next-step-ahead predictions to stay close to the specific ground-truth sequence being trained on. The teacher forcing procedure can be formally justified by using the chain rule of probability. For example, in the case of three variables, this factorization is: $p(y_1, y_2, y_3) = p(y_3|y_1, y_2)p(y_2|y_1)p(y_1)$, where we see that we take the ground-truth samples (coming from the "teacher") as input, rather than the predictions generated by the model itself.

When performing prediction, the ground-truth sequence is not available for conditioning predictions and we sample from the joint distribution over the sequence by sampling each y_t from its conditional distribution given the previously generated samples. This procedure can result in poor generation over long sequences as small prediction errors compound over many steps of generation. This can lead to poor prediction performance as the sequence of previously generated samples diverges from observed sequences seen during training.

1.4. Neural Architectures

1.4.1. Deep Neural Networks

The simplest deep neural networks consist of multiple alternating layers which each consist of a learnable linear projection followed by a (generally) fixed non-linearity:

$$\mathbf{h}_j^l = f\Big(\sum_{i=1}^{d_{in}^l} \mathbf{W}_{ij}^l \mathbf{h}_i^{l-1} + \mathbf{b}_i^l\Big) \tag{1.14}$$

If we use matrix notation, we can remove the subscripts and simplify this expression to:

$$\mathbf{h}^l = f(\mathbf{W}^l \mathbf{h}^{l-1} + \mathbf{b}^l) \tag{1.15}$$

In the above equation, the non-linearity is referred to as $f()$. Many different non-linearities have been shown to work well, but the ReLU non-linearity is simple and remains in wide usage: $f(a) = \max(a, 0)$.

The necessity of the non-linearity is intriguing. Initially, we observe that any product of multiple weight matrices can be equivalently written as a single weight matrix: $\mathbf{W} = \mathbf{W}_1\mathbf{W}_2...\mathbf{W}_n$. Thus a deep network without the use of a non-linearity is no more expressive than a linear model.

Neural networks with a single wide enough hidden layer are universal function approximators, but the width of the hidden layer may need to be very large to handle complex problems. Htad (1987) showed limited expressiveness for shallow networks.

1.4.2. Training Neural Networks with Backpropagation

In the simplest case, we want to train a neural network to minimize error on an i.i.d dataset of examples $((\mathbf{x}_1, y_1) \dots (\mathbf{x}_N, y_N))$. When using the negative log-likelihood loss function, ERM corresponds to maximum-likelihood estimation of the parameters of the model. In this section, we will discuss by far the most popular approach to training deep neural networks, the backpropagation algorithm (Rumelhart et al., 1986a; Werbos, 1974).

This technique involves computing gradients of the network's loss with respect to the parameters $\frac{\partial \mathcal{L}}{\partial \theta}$, which can then be used to incrementally update θ in the direction which locally reduces the loss:

$$\theta_t = \theta_{t-1} - \epsilon \frac{\partial \mathcal{L}}{\partial \theta}. \tag{1.16}$$

This gradient descent algorithm is provably justified for convex optimization problems, such as optimizing linear models with mean square error as the loss function. While neural networks are generally non-convex, it has been found empirically that gradient descent can still work well for a wide range of neural network architectures. However, this success depends on the details of the architecture and the initialization scheme and is far from guaranteed. For example, if a neural network is initialized with all parameters set to zero, the gradient is zero, and training stagnates - so it is generally essential to initialize the network with random-valued parameters.

A following question is how gradients can be computed for gradient descent. By far the most popular algorithm for this is backpropagation, which is a special case of reverse-mode automatic differentiation (for a review of automatic differentiation techniques we refer the reader to Margossian (2019)). The backpropagation algorithm is the application of the chain rule of calculus to neural networks. In its first stage, the gradient with respect to an intermediate hidden layer is computed as: $\frac{\partial \mathcal{L}}{\partial h_i} = \frac{\partial \mathcal{L}}{\partial h_{i+1}} \frac{\partial h_{i+1}}{\partial h_i}$, which can further be reduced to multiplying by the transposed weight matrix and multiplying by the point-wise derivative of the activation function. The gradient with respect to the weight matrices (and hence parameters) can then be computed based on the hidden states and the gradients with respect to the hidden states.

1.4.3. Stochastic Gradient Descent

On small datasets, it is often reasonable to compute the gradient across all the examples in the dataset and use this full gradient for each update. However, for large datasets, this is clearly sub-optimal, as the gradient on a small subset of the dataset may be very similar to

the gradient on the full dataset. As a result, the number of updates done for a fixed amount of computation would scale poorly in the size of the dataset. To illustrate this, we could imagine that with our computation budget we are able to do N full gradient descent updates. If we were to modify our dataset by duplicating every example K times, then we would only be able to do $\frac{N}{K}$ updates, which eventually would shrink to be less than 1 (meaning that we'd never do a single update). We can instead use a random subset of the examples to compute an unbiased estimate of the gradient (referred to as a *stochastic gradient*). The model takes many stochastic gradient steps to converge around a point estimate of the parameters that achieves low error. The SGD update rule at step t for a parameter $\mathbf{W}^l$, after seeing example $(\mathbf{x}, \mathbf{y})$, can be written as

$$\mathbf{W}_t^l = \mathbf{W}_{t-1}^l - \alpha \frac{\partial \mathcal{L}(\mathbf{x}, \mathbf{y}; \theta_{\mathbf{t-1}})}{\partial \mathbf{W}_{t-1}^l} \tag{1.17}$$

Where we use a learning rate α to control how large of a step to take in the direction of the gradient. α can be fixed over the course of training or changed adaptively as a function of t. Often we want to decay α near the end of training (to approach the minimum closer) and we also may want α to be small near the beginning of training to improve stability.

1.4.4. Recurrent Neural Networks

Another form of structure that can be added to deep neural networks involves applying the same function to a hidden state multiple times. Such an architecture is called a *Recurrent Neural Network* and the most basic variant can be written as: $\mathbf{h}_t = F(\mathbf{h}_{t-1}, x, \theta)$ where F is our function parameterized by θ (Rumelhart et al., 1986a). This kind of repeated computation could be significantly more parameter-efficient and could be useful in problems which require multiple functionally similar processing steps.

We can generalize this further to consider an architecture in which a sequence of inputs $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is transformed it into a sequence of vector representations $\mathbf{h} = (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n)$. We use subscripts to denote the position of an element within the sequence. This more general form of recurrent neural network can be written as: $\mathbf{h}_t = F(\mathbf{h}_{t-1}, \mathbf{x}_t, \theta)$. In this setup, each step in the repeated computation is provided with a single position in an input sequence. This is a widely used architecture for processing time series or other sequences such as text and audio.

A simple choice for the recurrent function F is to use a learned linear function of the input and the hidden state followed by an elementwise non-linearity. We can then write this as: $\mathbf{h}_t = \tanh(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t + \mathbf{b})$. This is parameterized by a recurrent weight matrix $\mathbf{W}$ and input weight matrix $\mathbf{U}$ and a bias term b that are shared across time. If the inputs $\mathbf{x}_i \in \mathbb{R}^{d_{in}}$ are continuous vector valued inputs, then $\mathbf{U} \in \mathbb{R}^{d_{in} \times d_{hid}}$ and $\mathbf{W} \in \mathbb{R}^{d_{hid} \times d_{hid}}$. The hidden state of the recurrent network may then be processed by a linear layer or an MLP to produce an output from the network, such as: $y_t = \mathbf{W}_{o2} \tanh(\mathbf{W}_{o1}\mathbf{h}_t)$

Recurrent networks can be trained using backpropagation, in much the same way as feedforward networks like MLPs are trained. These RNNs can be unrolled in time into a deep feedforward model with shared weights and the same principles used to compute gradients in Section 2.4.2 can be used while adding gradients across multiple positions to account for weight sharing. This algorithm is commonly referred to as Backpropagation Through Time (BPTT) (Rumelhart et al., 1986a; Werbos, 1990) due to its original usage for time-series, but the concept applies to any type of sequential data.

While exact gradients can be computed via backpropagation through time, the algorithm has a significant challenge in practice. If we backpropagate gradients for T steps, then in the absence of an activation function (the linear-RNN case), the backward gradient contains a product with the term W occuring T times, which approximately scales as $||W||^T$. If the norm of W is larger than 1, the resulting gradient tends to be very large for even moderately long time horizons. This is referred to colloquially as *Exploding Gradients*. If the norm of $||W||$ is smaller than 1, the resulting gradient will tend to be small, which is referred to as *Vanishing Gradients* (Hochreiter, 1991b; Bengio et al., 1994b; Pascanu et al., 2013a). To illustrate this problem more concretely, if we consider 200 time steps, then if the weight norm is 1.05, the resulting $||W||^T$ is about 15000. If the weight norm is 0.95, the resulting $||W||^T$ is about 10^{-5}. So even if the weight norm is only slightly deviated from 1, the resulting gradients are very badly scaled. Adding an activation function which has very small or zero gradients at many points can help address the exploding gradient problem at the expense of making the vanishing gradient problem more severe.

The simplest solution to this is *Truncated Backpropagation through Time* which simply stops computing gradients after a certain number of steps K (even if we compute the hidden state of the recurrent neural network for more than K steps). This clearly solves the gradient scaling issue yet also means that the model fails to systematically learn *Long Range Dependencies* (Bengio et al., 1994b; Pascanu et al., 2013a).

A simple way of reducing the impact of *exploding gradients* is gradient clipping, where the magnitude of the gradients with respect to every parameter is clipped to a maximum value (Pascanu et al., 2012). This is a poor solution in principle since it changes the expected value of the gradients. However, if the gradients only occasionally become very large, this gradient clipping strategy can prevent these very large gradients from ruining the progress of the training algorithm.

A more practical and more widely used solution for addressing vanishing and exploding gradients is to construct a recurrent neural network in which information is added into the hidden state on each recurrent time step after passing through a *Gating Function*. Gated recurrent networks were first presented in the form of the Long Short-Term Memory (LSTM) architecture (Hochreiter & Schmidhuber, 1997a), which was simplified to the Gated-Recurrent Unit (GRU) architecture (Cho et al., 2014).

The GRU introduces two gates $z_t = \sigma(W_z x_t + U_z h_{t-1})$ and $r_t = \sigma(W_r x_t + U_r h_{t-1})$, which both output a score from 0 to 1 for each unit. A pre-gated update value for the recurrent state is computed as: $\tilde{h}_t = \tanh(W x_t + U(r_t \odot h_t))$. The update for the recurrent state is: $h_t = (1 - z_t)h_{t-1} + z_t \tilde{h}_t$. Intuitively, z_t controls whether each unit keeps its value from the previous time-step or takes an updated value. r_t controls which units are used to compute that updated value for the recurrent hidden state.

We can gain more insights into the training dynamics of the GRU when z_t and r_t are saturated. If we always have $z_t = 1$ and $r_t = 1$, then the GRU becomes identical to a vanilla RNN, and it has the exact same training dynamics and resulting instabilities. On the other hand, when $z_t = 0$, then $h_t = h_{t-1}$ and the gradient is passed backward without modification. When $r_t = 0$, we can make updates to the hidden state but only using the input and an additive dependence on the previous hidden state (weighted by z_t). For example if $z_t = 0.5$ and $r_t = 0$, the GRU behaves as an additive integrator of a non-linear function of the input value: $\tanh(W x_t)$.

The resulting parameter gradients from a GRU network can still be an exponential of $||W||$, but it is no longer simply $||W||^T$. Rather, it depends on the value of z_t and r_t, with

lower values for these gates making the gradient less sensitive to the value of $||W||$. Intuitively, if a unit is only relevant on a small number of time steps and the computation between units is well-separated (the latter is unlikely to exactly hold in practice), then the gradient for that unit may effectively only go through a few applications of the recurrent weight matrix. For this reason, these gated recurrent networks tend to be fairly stable to train on sequences of substantial length (thousands of time steps), although if all the gates are saturated to 1 then it can still suffer from vanishing or exploding gradients, so gradient clipping can help if this saturated condition occurs infrequently.

1.4.5. Attention

Attention (as it is most widely used in deep learning) was introduced for the purpose of improving the capture of long-range dependencies and reducing the information bottleneck in sequence models (Bahdanau et al., 2014). The essential idea is to learn an input-dependent weighting to directly control how information is shared between pairs of positions. This can improve long-range dependencies by allowing information to directly flow from a distant position, rather than being preserved across many steps by a gated recurrent network. It can reduce the bottleneck by allowing a position to depend on many other positions, rather than the single most recent recurrent hidden state.

The most general interface for attention considers three matrices (each consisting of multiple positions each with its own representation vector) as input: queries $Q \in \mathrm{R}^{N_q \times d_q}$, keys $K \in \mathrm{R}^{N_k \times d_q}$, and values $V \in \mathrm{R}^{N_k \times d_v}$. The basic concept of attention (Bahdanau et al., 2014) is that by taking the dot product of all combinations of position-vectors in Q and K will yield a matrix of affinities $\alpha \in \mathrm{R}^{N_q \times N_k}$ between the queries and keys. This is then normalized such that every querying position $1, 2, ..., N_q$ will have an affinity over the keys which sums to 1, which is accomplished using a softmax function. This normalized affinity score is then multiplied by V (effectively weighting over positions in V) to yield an output matrix $A \in \mathrm{N_q} \times \mathrm{d_v}$.

Attention can be used to complement a gated recurrent network (Bahdanau et al., 2014), by allowing information to flow between distant positions. Vaswani et al. (2017) found that attention can be used in the absence of any recurrent architecture in the *Transformer*. This has a significant scaling advantage since it removes the sequential processing between positions required for recurrent networks. At the same time, a single layer of attention has much weaker processing capabilities compared to a single recurrent layer, so Transformers typically require many attention layers to achieve good performance. Additionally, the attention operation is permutation invariant, so ordering information needs to be provided through position encoding (either sinusoids with different periods or separate learned parameters per position).

Chapter 2

Introduction

2.1. Introduction

Is 100% accuracy on the test set enough? Many machine learning systems have achieved excellent accuracy across a variety of tasks (Deng et al., 2009; Mnih et al., 2013; Schrittwieser et al., 2019), yet the question of whether their reasoning or judgement is correct has come under question, and answers seem to be wildly inconsistent, depending on the task, architecture, training data, and interestingly, the extent to which test conditions match the training distribution. Have the main principles required for deep learning to achieve human-level performance been discovered, with the main remaining obstacle being to scale up? Or do we need to follow a completely different research direction not built on the principles discovered with deep learning, in order to achieve the kind of cognitive competence displayed by humans? Our goal here is to better understand the gap between current deep learning and human cognitive abilities so as to help answer these questions and suggest research directions for deep learning with the aim of bridging the gap towards human-level AI. Our main hypothesis is that deep learning succeeded in part because of a set of inductive biases (preferences, priors or assumptions), but that additional ones should be included in order to go from good in-distribution generalization in highly supervised learning tasks (or where strong and dense rewards are available), such as object recognition in images, to strong out-of-distribution generalization and transfer learning to new tasks with low sample complexity (few examples needed to generalize well). To make that concrete, we consider some of the inductive biases humans may exploit in conscious thought using highly sequential cognition operating at the level of conscious processing, and review some early work exploring these "high-level cognitive inductive priors" in deep learning. We use the term *high-level* to talk about variables that are manipulated at the conscious level of processing and are thus generally verbalizable. However, humans can consciously focus attention on low-level or intermediate-level features, e.g., by describing an odd-coloured pixel, not just very abstract concepts like objects or social situations. We argue that the deep learning progression from MLPs to convnets to transformers has in many ways been an (incomplete) progression towards the original goals of deep learning, i.e., to enable the discovery of a hierarchy of representations, with the most abstract ones, often associated with language, at the top. Note however, that although language may give us a view on system 2, these abilities are likely to pre-exist language as there is evidence of surprisingly strong forms of on-the-fly reasoning in some non-human animals, like corvids (Taylor et al., 2009). Our arguments suggest that while deep learning brought remarkable progress, it needs to be extended in qualitative and not just quantitative

ways: larger and more diverse datasets and more computing resources (Brown et al., 2020) are important but insufficient without additional inductive biases (Vaswani et al., 2017; He et al., 2016; Gilmer et al., 2017; Shazeer et al., 2017; Fedus et al., 2021a; Hinton, 2021; Welling, 2019; Dosovitskiy et al., 2020; Battaglia et al., 2018). We make the case that evolutionary forces, the interactions between multiple agents, the non-stationary and competition systems put pressure on the learning machinery to achieve the kind of flexibility, robustness and ability to adapt quickly which humans seem to have when they are faced with new environments (Bansal et al., 2017; Liu et al., 2019; Baker et al., 2019; Leibo et al., 2019) but needs to be improved with deep learning. The sought-after inductive biases should thus especially help AI to progress on these fronts. In addition to thinking about the learning and sample complexity advantage of these inductive biases, this paper links them with knowledge representation in neural networks, with the idea that by decomposing knowledge into its stable parts (like causal mechanisms) and volatile parts (random variables), and factorizing knowledge in small and somewhat independent pieces that can be recomposed dynamically as needed (to reason, imagine or explain at an *explicit* and verbalizable level), one may achieve the kind of systematic generalization which humans enjoy and is common in natural language (Marcus, 1998, 2019; Lake & Baroni, 2017; Bahdanau et al., 2018; McClelland et al., 1987).

This thesis is based on the following publications:

Knowledge Decomposition and Modularity

- Inductive biases od deep learning for higher level cognition: **Anirudh Goyal**, Yoshua Bengio, arXiv:2011.15091.
- Recurrent Independent Mechanisms: **Anirudh Goyal**, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Bernhard Schölkopf, Yoshua Bengio, arXiv:1909.10893 **(ICLR'21)**.
- Object Files and Schemata: Factorizing Declarative and Procedural Knowledge in Dynamical Systems: **Anirudh Goyal**, Alex Lamb, Phanideep Gampa, Philippe Beaudoin, Sergey Levine, Charles Blundell, Yoshua Bengio, Michael Mozer, arXiv:2006.16225 **(ICLR'21)**.
- Learning to Combine Top-Down and Bottom-Up Signals in Recurrent Neural Networks with Attention over Modules: Sarthak Mittal, Alex Lamb, **Anirudh Goyal**, Vikram Voleti, Murray Shanahan, Guillaume Lajoie, Michael Mozer, Yoshua Bengio, **(ICML'20)**.
- Neural Production Systems: Learning Rule-Governed Visual Dynamics: **Anirudh Goyal**, Aniket Didolkar, Nan Rosemary Ke, Charles Blundell, Philippe Beaudoin, Nicolas Heess, Michael Mozer, Yoshua Bengio, arXiv:2103.01937 **(NeurIPS'21)**.
- Coordination Among Neural Modules Through a Shared Global Workspace: **Anirudh Goyal**, Aniket Didolkar, Alex Lamb, Kartikeya Badola, Nan Rosemary Ke, Nasim Rahaman, Jonathan Binas, Charles Blundell, Michael Mozer, Yoshua Bengio, arXiv:2103.01197 **(ICLR'22)**.
- Discrete Valued Neural Communication: Dianbo Liu, Alex Lamb, Kenji Kawaguchi, **Anirudh Goyal**, Chen Sun, Michael Curtis Mozer, Yoshua Bengio, arXiv:2107.02367 **(NeurIPS'21)**.
- Fast and slow learning of Recurrent Independent Mechanisms: Kanika Madan, Nan Rosemary Ke, **Anirudh Goyal**, Bernhard Schölkopf, Yoshua Bengio, arXiv:2105.08710 **(ICLR'21)**.

- Sparse Attentive Backtracking: Temporal Credit Assignment Through Reminding: Nan Rosemary Ke, **Anirudh Goyal**, Olexa Bilaniuk, Jonathan Binas, Michael C. Mozer, Chris Pal, Yoshua Bengio, arXiv:1809.03702 **(NeurIPS'17)**.

Deep Learning and Causality

- A Meta-Transfer Objective for Learning to Disentangle Causal Mechanisms: Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Rosemary Ke, Sébastien Lachapelle, Olexa Bilaniuk, **Anirudh Goyal**, Christopher Pal, arXiv: 1901.10912 **(ICLR'20)**.
- Learning Neural Causal Models from Unknown Interventions: Nan Rosemary Ke, Olexa Bilaniuk, **Anirudh Goyal**, Stefan Bauer, Hugo Larochelle, Chris Pal, Yoshua Bengio, arXiv:1910.01075.
- Systematic Evaluation of Causal Discovery in Visual Model Based Reinforcement Learning: Nan Rosemary Ke, Aniket Didolkar, Sarthak Mittal, **Anirudh Goyal**, Guillaume Lajoie, Stefan Bauer, Danilo Rezende, Yoshua Bengio, Michael Mozer, Christopher Pal, arXiv:2107.00848 **(NeurIPS'21)**.
- CausalWorld: A Robotic Manipulation Benchmark for Causal Structure and Transfer Learning: Ossama Ahmed, Frederik Träuble, **Anirudh Goyal**, Alexander Neitz, Yoshua Bengio, Bernhard Schölkopf, Manuel Wüthrich, Stefan Bauer, arXiv:2010.04296 **(ICLR'21)**.

Information bottleneck and Reinforcement Learning

- InfoBot: Transfer and Exploration via the Information Bottleneck: **Anirudh Goyal**, Riashat Islam, Daniel Strouse, Zafarali Ahmed, Matthew Botvinick, Hugo Larochelle, Sergey Levine, Yoshua Bengio, arXiv:1901.10902 **(ICLR'19)**.
- The Variational Bandwidth Bottleneck: Stochastic Evaluation on an Information Budget: **Anirudh Goyal**, Yoshua Bengio, Matthew Botvinick, Sergey Levine, arXiv: 2004.11935 **(ICLR'20)**.
- Reinforcement Learning with Competitive Ensembles of Information-Constrained Primitives: **Anirudh Goyal**, Shagun Sodhani, Jonathan Binas, Xue Bin Peng, Sergey Levine, Yoshua Bengio, arXiv:1906.10667 **(ICLR'20)**.
- Retrieval-Augmented Reinforcement Learning: **Anirudh Goyal**, Abram L. Friesen, Andrea Banino, Theophane Weber, Nan Rosemary Ke, Adria Puigdomenech Badia, Arthur Guez, Mehdi Mirza, Peter C. Humphreys, Ksenia Konyushkova, Laurent Sifre, Michal Valko, Simon Osindero, Timothy Lillicrap, Nicolas Heess, Charles Blundell, arXiv:2202.08417 **(ICML'22)**.

2.2. Thesis Overview

The thesis contains 4 representative articles from the above list that focus on knowledge factorization and generalization in deep learning and deep reinforcement learning. This thesis is organized in the following unusual way, due to the large number of papers to be covered. The first substantive part of this document (chapter 2) contains a review of my proposed research area (inductive biases for deep learning of higher level cognition), an overview of the main contributions of the above papers, as well as a vision of proposed future research. It is written as a position paper which could stand on its own and a version of it is posted on arXiv (co-authored by Yoshua Bengio and myself) and accepted for publication. The

next four chapters are four of the above papers whose contributions we believe to be more significant and central to the thesis, going in more details in some of the key ideas and have the more usual format of a machine learning conference paper.

Chapter 5 presents an architectural bias for promoting systematic generalization by factorizing knowledge into variables and rules, such that one can reuse information about rules across variables as long as type of variables matches the type of input a particular rule expects.

Chapter 9 explores the use of information bottleneck for learning policy primitives in a completely decentralized fashion.

Chapter 11 presents an architecture where RL agent is augmented with the retrieval process parameterized as a neural network. The goal of the retrieval process is to provide contextual information relevant to the RL agent from a large database.

Chapter 12 presents an overall conclusion of the contributions in this thesis.

2.3. Probabilistic Machine Learning

Machine learning systems are able to make predictions or produce actions by automatically learning from data and experience. Mitchell et al. (1997) define a learning machine as an algorithm that improves with experience on a particular task, given a certain performance measure to evaluate the system. In this thesis, we are interested in questions of how machine learning systems can generalize either when trained with limited data or limited task diversity.

We often assume the ability to independently sample a certain finite number of datapoints from the same unknown data distribution. Some of that data is partitioned for our systems to learn on and the rest is reserved for testing how it generalizes on unseen examples. Generalizing to data drawn from the same distribution is a fairly common evaluation setup used in most machine learning systems. Most theoretical guarantees on the generalization properties of machine learning systems have required this assumption. This is however being questioned more as our systems do not generalize systematically and perform poorly when dealing with out-of-distribution data. For the sake of this introduction however, we will focus on the case where our data is independent and identically distributed (iid), which means that both training data and test cases are assumed to come from the same distribution.

2.3.1. Supervised Learning

Supervised Learning serves as the fundamental framework for most of the work in this thesis and much of machine learning research as a whole. It consists of learning to predict output targets from inputs given a labeled dataset of input/output pairs, where all pairs are assumed to be sampled uniformly from an unknown distribution. An appealing aspect of supervised learning is that the learned model's accuracy on held-out samples is an objective measure of success. Additionally, the correct outputs for the model are provided in the dataset, which makes credit assignment relatively straightforward. Compare this to playing a video game, where the correct actions are never given to the player.

Some examples of supervised learning include image classification (where the image is the input and the class label is the output) and speech recognition (where the speech signal is the input and the corresponding text is the output).

The goal of supervised learning is to learn a function f_θ which maps an example's input to its output. These functions f_θ map a d-dimensional, typically real-valued vector input $\mathbf{x} \in \mathbb{R}^d$ from the space of possible inputs X to a discrete scalar output label y from the space of k

possible outputs Y. We assume our data comes from an unknown joint probability distribution over inputs and outputs $p(\mathbf{x}, y)$ where $\mathbf{x} \in X$ and $y \in Y$ and we would like to learn a function $f_\theta : \mathbb{R}^d \to \mathbb{R}^k$ that maps X to an estimate of some statistic of $P(Y|X)$, like the probability itself or $E[Y|X]$. We observe examples from this distribution $((\mathbf{x_1}, y_1) \dots (\mathbf{x_N}, y_N))$. The subscript θ is used to indicate that our function is parameterized. This doesn't always have to be the case, and there exist "non-parameteric" models in machine learning, but in the context of this thesis that focuses on neural networks, we are interested in learning parametric functions. There are many possible functions that can map from X to a distribution over Y, but we are often interested in learning one that makes as few mistakes as possible. We define a loss function $L(\hat{y}, y)$ that measures the penalty incurred when classifying y as $\hat{y}$ and we try to optimize f_θ to minimize this loss given a finite amount of data. We can use this loss to define the empirical risk associated with a particular function as the expected loss over the entire training set $R_{emp}(X, Y, f_\theta) = \frac{1}{N} \sum_{i=1}^{N} L(f_\theta(\mathbf{x_i}), y_i)$. Empirical Risk Minimization (ERM) tries to find $\arg\min_{f \in F} R_{emp}(f)$. However, for a given dataset, there may be many functions $f \in F$ in the hypothesis space, that achieve the same or even 0 empirical risk on the training data subset, but generalize poorly to unseen examples. In such cases, we want to bias our learners to certain solutions that may have better generalization properties such as those with low parameter norms with L1/L2 regularization, with max-margin methods (**?**), or by using dropout (Srivastava et al., 2014).

In the case where we want to do probabilistic classification, $f_\theta(\mathbf{x})$ outputs a probability distribution over possible class labels and the loss function used is typically the negative log-likelihood of the correct class $-\log q_\theta(Y = y|\mathbf{x}) = -\log f_{\theta,y}((x))$, where $f_{\theta,y}()$ is the y-th output element of the vector $f_\theta(\mathbf{x})$ containing all the output probabilities. Intuitively, we wish to maximize the probability the model assigns to the correct class, and the use of the log causes a very high loss when a small probability is assigned to the correct class. The frequentist approach to parameter estimation termed Maximum Likelihood Estimation (MLE) seeks a setting of parameters that maximizes the likelihood of the observed data. This decomposes into a product over the examples in the case where their conditional distributions are independent and identically distributed (the i.i.d. assumption): $q_\theta(Y|X) = \prod_{i=1}^{N} q_\theta(y_i|\mathbf{x_i})$. Since the logarithm is a monotonically increasing function, we can maximize the $\log q_\theta(y_1, y_2, \dots, y_N|\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$ and break the product into a bunch of sums $\log q_\theta(y_1, y_2, \dots, y_N|\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N) = \sum_{i=1}^{N} \log q_\theta(y_i|\mathbf{x_i})$. To turn it into a minimization problem, we can minimize the negative log-likelihood of the data, showing that empirical risk minimization (ERM) with the negative log-likelihood loss corresponds to maximum-likelihood estimation of the parameters of the model. At inference/test time, if we want to choose one of the labels, the model would output $\arg\max_{\tilde{y}} q_\theta(\tilde{y}|\mathbf{x}_i)$. However, the output probability distribution also gives us other information, such as the uncertainty in the output, which can be represented by the entropy of $q_\theta(y_i|\mathbf{x_i})$.

2.3.2. Distributions and Likelihood

So far, we have discussed generative modeling in qualitative terms. We want models which can simulate from the dynamics of the world. We want models that can synthesize realistic looking data. However, before going further it is useful to understand the probabilistic

interpretation of generative models, which gives a formal framework for studying generative models. The essential idea is that we treat observations from the world as samples from a distribution $x, y \sim p(x, y)$. For example, we could consider the distribution over all human faces which can occur in reality to be $p(x)$ and consider each face as a sample with an associated label $y \sim p(y|x)$. If we have access to a recorded dataset (for example a set of faces), we may also choose to treat these points as a finite collection of samples from this distribution.

At the same time, we can interpret our generative model as an estimating distribution $q_\theta(y|x)$, which is described by a set of parameters θ. Then we can frame generative modeling as trying to ensure that $p(y|x)$ and $q_\theta(y|x)$ become as similar as possible. Statistical divergences give a natural mathematical framework for this. A divergence is a function $D(p||q) : S \times S \to R$ taking two distributions p and q over a space of distributions S as inputs, with the properties:

$$D(p||q) \geq 0.0 \tag{2.1}$$

$$D(p||q) = 0.0 \iff p = q \tag{2.2}$$

Notably, there is no symmetry assumption, so in general $D(p||q) \neq D(q||p)$. The probabilistic approach to generative modeling frames learning as an optimization problem where the loss corresponds to a given divergence.

$$\mathcal{L}(\theta) = \underset{\theta}{\operatorname{argmin}}\, D(p(y|x)||q_\theta(y|x)). \tag{2.3}$$

2.3.3. Maximum Likelihood and KL-Divergence

We will now present the classical result showing how minimizing KL-divergence can be applied to the supervised learning setting introduced previously. In particular, we assume the ability to sample from $p(x, y)$ and wish to learn $q_\theta(y|x)$.

What is the right algorithm for finding a distribution $q_\theta(y|x)$ which minimizes a divergence between itself and $p(y|x)$? Before selecting the type of divergence to minimize, a natural question is to consider what types of expressions we are capable of optimizing, and work backwards to find a suitable divergence. In general, we only have access to samples from the distribution $p(x, y)$ and not any additional information about the distribution. At the same time, $q_\theta(y|x)$ is a model that we control, so it's reasonable to believe that we'll be able to design it so that it has a density that we can compute as well as the ability to draw samples.

The KL-divergence can be rewritten as an expression in which the only term that depends on the parameters is an expectation involving $q_\theta(y|x)$ over samples from $p(x, y)$. Beginning with two distributions $p(x, y)$ (the empirical distribution) and $q_\theta(y|x)$ (the model distribution), we write the KL-divergence:

$$D_{KL}(p(y|x)||q_\theta(y|x)) = \mathop{\mathbb{E}}_{x\sim p(x)}\left[\int p(y|x)\log p(y|x)dy - \int p(y|x)\log q_\theta(y|x)dy\right] \tag{2.4}$$

$$= \mathop{\mathbb{E}}_{x\sim p(x)}\int p(y|x)\log\frac{p(y|x)}{q_\theta(y|x)}dy \tag{2.5}$$

$$= \mathop{\mathbb{E}}_{x,y\sim p(x,y)}\left[\log\frac{p(y|x)}{q_\theta(y|x)}\right] \tag{2.6}$$

$$= \mathop{\mathbb{E}}_{x,y\sim p(x,y)}[\log p(y|x) - \log q_\theta(y|x)] \tag{2.7}$$

$$= H_p(y|x) - \mathop{\mathbb{E}}_{x,y\sim p(x,y)}[\log q_\theta(y|x)] \tag{2.8}$$

Then we can show the maximum likelihood estimation for a set of N data points.

$$\theta^* = \arg\max_\theta \prod_{i=1}^{N} q_\theta(y_i|x_i) \tag{2.9}$$

$$= \arg\max_\theta \sum_{i=1}^{N} \log q_\theta(y_i|x_i) \tag{2.10}$$

$$= \arg\min_\theta \frac{1}{N}\sum_{i=1}^{N} -\log q_\theta(y_i|x_i) \tag{2.11}$$

$$\sim= \arg\min_\theta \mathop{\mathbb{E}}_{x_i,y_i\sim p(x,y)}[-\log q_\theta(y|x)] \tag{2.12}$$

The objective for maximum likelihood is to maximize the log-density $\log(q_\theta(y|x))$ over real data points sampled from the distribution $p(x, y)$. It can be written as:

$\mathrm{D}_{KL}(p(y|x)||q(y|x)) = \mathbb{E}_{x\sim p(x)}\left[\int p(y|x)\log p(y|x)dy - \int p(y|x)\log q_\theta(y|x)dy\right]$
$= -H(p(y|x)) + \mathrm{CE}(p(y|x), q_\theta(y|x))$

Thus we can see that the KL-divergence decomposes into two terms in Eq. 2.3.3: a cross-entropy term (likelihood) and a term for the entropy of the true data distribution. Because the entropy of the true data distribution doesn't depend on the estimator, the KL-divergence can be minimized by maximizing likelihood. Another useful consequence of this is that the entropy of the true data distribution can be estimated by such a generative model if it maximizes likelihood among all possible distributions, i.e., $q_\theta = p$, the KL-divergence is 0 and the cross-entropy equals the entropy.

2.3.4. ERM and probabilistic framing of the learning problem

The empirical risk minimization objective corresponds to minimizing a loss function of interest over a set of examples x with labels y sampled from a generally unknown distribution $p(x, y)$

$$\theta^* = \arg\min_{\theta} \sum_{x,y \sim p(x,y)} L(y, f(x)) \tag{2.13}$$

in the hope of obtaining a function with low expected loss on examples from the same distribution. A special case of empirical risk minimization is the maximum likelihood objective. Essentially, it can be shown that maximizing a model's likelihood on some data distribution is equivalent to minimizing an expectation of loss over samples from the empirical data distribution (which is non-zero only on the data points), which can then be empirically estimated with a sum over examples. The objective for maximum likelihood is maximizing the log-density $log(q_\theta(x))$ over real data points sampled from the distribution $p(x)$. The use of maximum likelihood training for generative models (such as sequence models) is an unconditional generalization of the use of conditional maximum likelihood for supervised learning.

2.3.5. Teacher Forcing

A specific variant of the maximum likelihood principle can be used for training sequence models. This form of training is known as teacher forcing (Williams & Zipser, 1989), due to the use of the ground-truth samples y_t being fed back into the model to be conditioned on for the prediction of later outputs. These fed back samples force the next-step-ahead predictions to stay close to the specific ground-truth sequence being trained on. The teacher forcing procedure can be formally justified by using the chain rule of probability. For example, in the case of three variables, this factorization is: $p(y_1, y_2, y_3) = p(y_3|y_1, y_2)p(y_2|y_1)p(y_1)$, where we see that we take the ground-truth samples (coming from the "teacher") as input, rather than the predictions generated by the model itself.

When performing prediction, the ground-truth sequence is not available for conditioning predictions and we sample from the joint distribution over the sequence by sampling each y_t from its conditional distribution given the previously generated samples. This procedure can result in poor generation over long sequences as small prediction errors compound over many steps of generation. This can lead to poor prediction performance as the sequence of previously generated samples diverges from observed sequences seen during training.

2.4. Neural Architectures

2.4.1. Deep Neural Networks

The simplest deep neural networks consist of multiple alternating layers which each consist of a learnable linear projection followed by a (generally) fixed non-linearity:

$$\mathbf{h}_j^l = f\Big(\sum_{i=1}^{d_{in}^l} \mathbf{W}_{ij}^l \mathbf{h}_i^{l-1} + \mathbf{b}_i^l\Big) \tag{2.14}$$

If we use matrix notation, we can remove the subscripts and simplify this expression to:

$$\mathbf{h}^l = f(\mathbf{W}^l \mathbf{h}^{l-1} + \mathbf{b}^l) \tag{2.15}$$

In the above equation, the non-linearity is referred to as $f()$. Many different non-linearities have been shown to work well, but the ReLU non-linearity is simple and remains in wide usage: $f(a) = \max(a, 0)$.

The necessity of the non-linearity is intriguing. Initially, we observe that any product of multiple weight matrices can be equivalently written as a single weight matrix: $\mathbf{W} = \mathbf{W}_1\mathbf{W}_2...\mathbf{W}_n$. Thus a deep network without the use of a non-linearity is no more expressive than a linear model.

Neural networks with a single wide enough hidden layer are universal function approximators, but the width of the hidden layer may need to be very large to handle complex problems. Htad (1987) showed limited expressiveness for shallow networks.

2.4.2. Training Neural Networks with Backpropagation

In the simplest case, we want to train a neural network to minimize error on an i.i.d dataset of examples $((\mathbf{x}_1, y_1) \dots (\mathbf{x}_N, y_N))$. When using the negative log-likelihood loss function, ERM corresponds to maximum-likelihood estimation of the parameters of the model. In this section, we will discuss by far the most popular approach to training deep neural networks, the backpropagation algorithm (Rumelhart et al., 1986a; Werbos, 1974).

This technique involves computing gradients of the network's loss with respect to the parameters $\frac{\partial \mathcal{L}}{\partial \theta}$, which can then be used to incrementally update θ in the direction which locally reduces the loss:

$$\theta_t = \theta_{t-1} - \epsilon \frac{\partial \mathcal{L}}{\partial \theta}. \tag{2.16}$$

This gradient descent algorithm is provably justified for convex optimization problems, such as optimizing linear models with mean square error as the loss function. While neural networks are generally non-convex, it has been found empirically that gradient descent can still work well for a wide range of neural network architectures. However, this success depends on the details of the architecture and the initialization scheme and is far from guaranteed. For example, if a neural network is initialized with all parameters set to zero, the gradient is zero, and training stagnates - so it is generally essential to initialize the network with random-valued parameters.

A following question is how gradients can be computed for gradient descent. By far the most popular algorithm for this is backpropagation, which is a special case of reverse-mode automatic differentiation (for a review of automatic differentiation techniques we refer the reader to Margossian (2019)). The backpropagation algorithm is the application of the chain rule of calculus to neural networks. In its first stage, the gradient with respect to an intermediate hidden layer is computed as: $\frac{\partial \mathcal{L}}{\partial h_i} = \frac{\partial \mathcal{L}}{\partial h_{i+1}} \frac{\partial h_{i+1}}{\partial h_i}$, which can further be reduced to multiplying by the transposed weight matrix and multiplying by the point-wise derivative of the activation function. The gradient with respect to the weight matrices (and hence parameters) can then be computed based on the hidden states and the gradients with respect to the hidden states.

2.4.3. Stochastic Gradient Descent

On small datasets, it is often reasonable to compute the gradient across all the examples in the dataset and use this full gradient for each update. However, for large datasets, this is clearly sub-optimal, as the gradient on a small subset of the dataset may be very similar to

the gradient on the full dataset. As a result, the number of updates done for a fixed amount of computation would scale poorly in the size of the dataset. To illustrate this, we could imagine that with our computation budget we are able to do N full gradient descent updates. If we were to modify our dataset by duplicating every example K times, then we would only be able to do $\frac{N}{K}$ updates, which eventually would shrink to be less than 1 (meaning that we'd never do a single update). We can instead use a random subset of the examples to compute an unbiased estimate of the gradient (referred to as a *stochastic gradient*). The model takes many stochastic gradient steps to converge around a point estimate of the parameters that achieves low error. The SGD update rule at step t for a parameter $\mathbf{W}^l$, after seeing example $(\mathbf{x}, \mathbf{y})$, can be written as

$$\mathbf{W}_t^l = \mathbf{W}_{t-1}^l - \alpha \frac{\partial \mathcal{L}(\mathbf{x}, \mathbf{y}; \theta_{\mathbf{t-1}})}{\partial \mathbf{W}_{t-1}^l} \tag{2.17}$$

Where we use a learning rate α to control how large of a step to take in the direction of the gradient. α can be fixed over the course of training or changed adaptively as a function of t. Often we want to decay α near the end of training (to approach the minimum closer) and we also may want α to be small near the beginning of training to improve stability.

2.4.4. Recurrent Neural Networks

Another form of structure that can be added to deep neural networks involves applying the same function to a hidden state multiple times. Such an architecture is called a *Recurrent Neural Network* and the most basic variant can be written as: $\mathbf{h}_t = F(\mathbf{h}_{t-1}, x, \theta)$ where F is our function parameterized by θ (Rumelhart et al., 1986a). This kind of repeated computation could be significantly more parameter-efficient and could be useful in problems which require multiple functionally similar processing steps.

We can generalize this further to consider an architecture in which a sequence of inputs $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is transformed it into a sequence of vector representations $\mathbf{h} = (\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n)$. We use subscripts to denote the position of an element within the sequence. This more general form of recurrent neural network can be written as: $\mathbf{h}_t = F(\mathbf{h}_{t-1}, \mathbf{x}_t, \theta)$. In this setup, each step in the repeated computation is provided with a single position in an input sequence. This is a widely used architecture for processing time series or other sequences such as text and audio.

A simple choice for the recurrent function F is to use a learned linear function of the input and the hidden state followed by an elementwise non-linearity. We can then write this as: $\mathbf{h}_t = \tanh(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{x}_t + \mathbf{b})$. This is parameterized by a recurrent weight matrix $\mathbf{W}$ and input weight matrix $\mathbf{U}$ and a bias term b that are shared across time. If the inputs $\mathbf{x}_i \in \mathbb{R}^{d_{in}}$ are continuous vector valued inputs, then $\mathbf{U} \in \mathbb{R}^{d_{in} \times d_{hid}}$ and $\mathbf{W} \in \mathbb{R}^{d_{hid} \times d_{hid}}$. The hidden state of the recurrent network may then be processed by a linear layer or an MLP to produce an output from the network, such as: $y_t = \mathbf{W}_{o2} \tanh(\mathbf{W}_{o1}\mathbf{h}_t)$

Recurrent networks can be trained using backpropagation, in much the same way as feedforward networks like MLPs are trained. These RNNs can be unrolled in time into a deep feedforward model with shared weights and the same principles used to compute gradients in Section 2.4.2 can be used while adding gradients across multiple positions to account for weight sharing. This algorithm is commonly referred to as Backpropagation Through Time (BPTT) (Rumelhart et al., 1986a; Werbos, 1990) due to its original usage for time-series, but the concept applies to any type of sequential data.

While exact gradients can be computed via backpropagation through time, the algorithm has a significant challenge in practice. If we backpropagate gradients for T steps, then in the absence of an activation function (the linear-RNN case), the backward gradient contains a product with the term W occuring T times, which approximately scales as $||W||^T$. If the norm of W is larger than 1, the resulting gradient tends to be very large for even moderately long time horizons. This is referred to colloquially as *Exploding Gradients*. If the norm of $||W||$ is smaller than 1, the resulting gradient will tend to be small, which is referred to as *Vanishing Gradients* (Hochreiter, 1991b; Bengio et al., 1994b; Pascanu et al., 2013a). To illustrate this problem more concretely, if we consider 200 time steps, then if the weight norm is 1.05, the resulting $||W||^T$ is about 15000. If the weight norm is 0.95, the resulting $||W||^T$ is about 10^{-5}. So even if the weight norm is only slightly deviated from 1, the resulting gradients are very badly scaled. Adding an activation function which has very small or zero gradients at many points can help address the exploding gradient problem at the expense of making the vanishing gradient problem more severe.

The simplest solution to this is *Truncated Backpropagation through Time* which simply stops computing gradients after a certain number of steps K (even if we compute the hidden state of the recurrent neural network for more than K steps). This clearly solves the gradient scaling issue yet also means that the model fails to systematically learn *Long Range Dependencies* (Bengio et al., 1994b; Pascanu et al., 2013a).

A simple way of reducing the impact of *exploding gradients* is gradient clipping, where the magnitude of the gradients with respect to every parameter is clipped to a maximum value (Pascanu et al., 2012). This is a poor solution in principle since it changes the expected value of the gradients. However, if the gradients only occasionally become very large, this gradient clipping strategy can prevent these very large gradients from ruining the progress of the training algorithm.

A more practical and more widely used solution for addressing vanishing and exploding gradients is to construct a recurrent neural network in which information is added into the hidden state on each recurrent time step after passing through a *Gating Function*. Gated recurrent networks were first presented in the form of the Long Short-Term Memory (LSTM) architecture (Hochreiter & Schmidhuber, 1997a), which was simplified to the Gated-Recurrent Unit (GRU) architecture (Cho et al., 2014).

The GRU introduces two gates $z_t = \sigma(W_z x_t + U_z h_{t-1})$ and $r_t = \sigma(W_r x_t + U_r h_{t-1})$, which both output a score from 0 to 1 for each unit. A pre-gated update value for the recurrent state is computed as: $\tilde{h}_t = \tanh(W x_t + U(r_t \odot h_t))$. The update for the recurrent state is: $h_t = (1 - z_t)h_{t-1} + z_t \tilde{h}_t$. Intuitively, z_t controls whether each unit keeps its value from the previous time-step or takes an updated value. r_t controls which units are used to compute that updated value for the recurrent hidden state.

We can gain more insights into the training dynamics of the GRU when z_t and r_t are saturated. If we always have $z_t = 1$ and $r_t = 1$, then the GRU becomes identical to a vanilla RNN, and it has the exact same training dynamics and resulting instabilities. On the other hand, when $z_t = 0$, then $h_t = h_{t-1}$ and the gradient is passed backward without modification. When $r_t = 0$, we can make updates to the hidden state but only using the input and an additive dependence on the previous hidden state (weighted by z_t). For example if $z_t = 0.5$ and $r_t = 0$, the GRU behaves as an additive integrator of a non-linear function of the input value: $\tanh(W x_t)$.

The resulting parameter gradients from a GRU network can still be an exponential of $||W||$, but it is no longer simply $||W||^T$. Rather, it depends on the value of z_t and r_t, with

lower values for these gates making the gradient less sensitive to the value of $||W||$. Intuitively, if a unit is only relevant on a small number of time steps and the computation between units is well-separated (the latter is unlikely to exactly hold in practice), then the gradient for that unit may effectively only go through a few applications of the recurrent weight matrix. For this reason, these gated recurrent networks tend to be fairly stable to train on sequences of substantial length (thousands of time steps), although if all the gates are saturated to 1 then it can still suffer from vanishing or exploding gradients, so gradient clipping can help if this saturated condition occurs infrequently.

2.4.5. Attention

Attention (as it is most widely used in deep learning) was introduced for the purpose of improving the capture of long-range dependencies and reducing the information bottleneck in sequence models (Bahdanau et al., 2014). The essential idea is to learn an input-dependent weighting to directly control how information is shared between pairs of positions. This can improve long-range dependencies by allowing information to directly flow from a distant position, rather than being preserved across many steps by a gated recurrent network. It can reduce the bottleneck by allowing a position to depend on many other positions, rather than the single most recent recurrent hidden state.

The most general interface for attention considers three matrices (each consisting of multiple positions each with its own representation vector) as input: queries $Q \in \mathrm{R}^{N_q \times d_q}$, keys $K \in \mathrm{R}^{N_k \times d_q}$, and values $V \in \mathrm{R}^{N_k \times d_v}$. The basic concept of attention (Bahdanau et al., 2014) is that by taking the dot product of all combinations of position-vectors in Q and K will yield a matrix of affinities $\alpha \in \mathrm{R}^{N_q \times N_k}$ between the queries and keys. This is then normalized such that every querying position $1, 2, ..., N_q$ will have an affinity over the keys which sums to 1, which is accomplished using a softmax function. This normalized affinity score is then multiplied by V (effectively weighting over positions in V) to yield an output matrix $A \in \mathrm{N_q} \times \mathrm{d_v}$.

Attention can be used to complement a gated recurrent network (Bahdanau et al., 2014), by allowing information to flow between distant positions. Vaswani et al. (2017) found that attention can be used in the absence of any recurrent architecture in the *Transformer*. This has a significant scaling advantage since it removes the sequential processing between positions required for recurrent networks. At the same time, a single layer of attention has much weaker processing capabilities compared to a single recurrent layer, so Transformers typically require many attention layers to achieve good performance. Additionally, the attention operation is permutation invariant, so ordering information needs to be provided through position encoding (either sinusoids with different periods or separate learned parameters per position).

Chapter 3

Inductive Biases of Deep Learning of Higher Level Cognition

3.1. Data, Statistical Models and Causality

Our current state-of-the-art machine learning systems sometimes achieve good performance on a specific and narrow task, using very large quantities of labeled data, either by supervised learning or reinforcement learning (RL) with strong and frequent rewards. Instead, humans are able to understand their environment in a more unified way (rather than with a separate set of parameters for each task) which allows them to quickly generalize (from few examples) on a new task, thanks to their ability to reuse previously acquired knowledge. Instead, current systems are generally not robust to changes in distribution (Peters et al., 2017b; Geirhos et al., 2020; Hendrycks et al., 2021; Koh et al., 2021; Schneider et al., 2020), adversarial examples (Goodfellow et al., 2014; Kurakin et al., 2016), spurious correlations (Krueger et al., 2021; Beery et al., 2018; Arjovsky et al., 2019) etc.

One possibility studied in the machine learning literature is that we should train our models with multiple datasets, each providing a different view of the underlying model of the world shared by humans (Baxter, 2000). Whereas multi-task learning usually just pools the different datasets (Caruana, 1997; Collobert & Weston, 2008; Ruder, 2017), we believe that there is something more to consider: we want our learner to perform well on a completely new task or distribution, either immediately (with zero-shot out-of-distribution generalization), or with a few examples (i.e. with efficient transfer learning) (Ravi & Larochelle, 2016; Wang et al., 2016; Finn et al., 2017; Cabi et al., 2019; Jang et al., 2022; Reed et al., 2022; Ahn et al., 2022; Brown et al., 2020; Alayrac et al., 2022; Borgeaud et al., 2022; Chowdhery et al., 2022; Sanh et al., 2021; Lu et al., 2022; Raffel et al., 2020).

This raises the question of changes in distribution or task. Whereas the traditional train-test scenario and learning theory assumes that test examples come from the same distribution as the training data, just dropping that assumption means that we cannot say anything about generalization to a modified distribution. Hence new assumptions are required about how the different tasks or the different distributions encountered by a learning agent are related to each other.

We use the term *structural-mechanistic* (Schölkopf, 2015) to characterize models which follow an underlying mechanistic understanding of reality. They are closely related to the structural causal models used to capture causal structure (Pearl, 2009). The key property of such models is that they will make correct predictions over a variety of data distributions

which are drawn from the same underlying causal system, rather than being specific to a particular distribution. To give a concrete example, the equation $E = MC^2$ relates mass and energy in a way which we expect to hold regardless of other properties in the world. On the other hand, an equation like "$\text{GDP}_t = 1.05\,\text{GDP}_{t-1}$+ noise" may be correct under a particular data distribution (for example a country with some growth pattern) but will fail to hold when some aspects of the world are changed, even in ways which did not happen or could not happen, i.e., in a counterfactual.

However, humans do not represent all of their knowledge in such a neat verbalizable way as Newton's equations. Most humans understand physics first of all at an *intuitive* level and in solving practical problems we typically combine such implicit knowledge with explicit verbalizable knowledge (McCloskey, 1983; Baillargeon et al., 1985; Spelke et al., 1992; Battaglia et al., 2013). We can name high-level variables like position and velocity but may find it difficult to explain the intuitively known mechanisms which relate them to each other, in everyday life (by opposition to a physicist running a simulation of Newton's equations).

Implicit and Explicit Knowledge

An important question for us is how knowledge can be represented in these two forms, the implicit – intuitive and difficult to verbalize – and the explicit – which allows humans to share part of their thinking process through natural language.

Causal understanding hinges on capturing the effect of interventions as changes in distribution. Humans frequently explain their perception (at the explicit level) and reason in terms of causal structure, and causal structure is really about how a joint distribution between causal random variables can change under interventions, i.e., actions. This suggests that one possible direction that deep learning needs to incorporate includes more notions about agency, reasoning and causality, even when the application only involves single inputs like an image and not actually learning a policy. For this purpose we need to examine how to go beyond the statistical learning framework that has dominated deep learning and machine learning in recent decades. Instead of thinking of data as a set of examples drawn independently from the same distribution, we should probably reflect on the origin of the data through a real-world non-stationary process. We claim that this perspective would help learning agents, such as babies or robots to succeed in the changing environments. This paper mostly discusses inductive biases inspired by higher-level cognition and aimed at facing these generalization challenges, pointing to existing work to implement some of them. However, for the most part, how to efficiently implement and combine these inductive biases in a single system remains an open question.

3.2. About Inductive Biases

The no-free-lunch theorem for machine learning (Wolpert et al., 1995; Baxter, 2000) basically says that some set of preferences (or inductive bias) over the space of all functions is necessary to obtain generalization, that there is no completely general-purpose learning algorithm, that any learning algorithm will generalize better on some distributions and worse on others. Typically, given a particular dataset and loss function, there are many possible solutions (e.g. parameter assignments) to the learning problem that exhibit equally "good" performance on the training points. Given a finite training set, the only way to generalize to

Inductive Bias	Corresponding property
Distributed representations	Inputs mapped to patterns of features
Convolution	group equivariance (usually over space)
Deep architectures	Complicated functions = composition of simpler ones
Graph Neural Networks	equivariance over entities and relations
Recurrent Nets	equivariance over time
Soft attention	equivariance over permutations
Self-supervised pre-training	$P(X)$ is informative about $P(Y\|X)$

Table 1. Examples of current inductive biases in deep learning. Many have to do with the architecture while the last one influences the training framework and objective.

new input configurations is then to rely on some assumptions or preferences about the solution we are looking for. An important question for AI research aiming at human-level performance then is to identify inductive biases that are most relevant to the human perspective on the world around us. Inductive biases, broadly speaking, encourage the learning algorithm to prioritise solutions with certain properties. Table 1 lists some of the inductive biases already used in various neural networks, and the corresponding properties. Although they are often expressed in terms of a neural architecture, they can also be about how the networks are trained, e.g., unsupervised pre-training, self-supervised learning and semi-supervised training, which all have to do with the input distribution $P(X)$ being informative about future tasks $P(Y|X)$. Other relevant elements which are not directly about inductive biases (and not discussed further in this paper) include for example the ability of a learning agent to actively seek knowledge (e.g. in active learning or reinforcement learning) or to obtain information from other agents (e.g., social learning, multi-agent learning).

From Inductive Biases to Algorithms. There are many ways to encode such biases—e.g. explicit regularisation objectives (Bishop et al., 1995; Bishop, 1995; Srivastava et al., 2014; Kukačka et al., 2017; Zhang et al., 2017), architectural constraints (Yu & Koltun, 2015; Long et al., 2015; Dumoulin & Visin, 2016; He et al., 2016; Huang et al., 2017), parameter sharing (Hochreiter & Schmidhuber, 1997b; Pham et al., 2018), implicit effects of the choice of the optimization method (Jastrzębski et al., 2017; Smith & Le, 2017; Chaudhari & Soatto, 2018), self-supervised learning or self-supervised pre-training (Hinton et al., 2006; Erhan et al., 2010; Devlin et al., 2018b; Chen et al., 2020b,d; Grill et al., 2020), invariance or equivariance to known transformations (Bruna et al., 2013; Defferrard et al., 2016; Ravanbakhsh et al., 2017; Thomas et al., 2018; Finzi et al., 2020; Satorras et al., 2021) or choices of prior distributions in a Bayesian model (Jeffreys, 1946; Berger & Bernardo, 1992; Gelman, 1996; Fortuin, 2022). For example, one can build translation invariance of a neural network output by replacing matrix multiplication by convolutions (LeCun et al., 1995) and pooling (Krizhevsky et al., 2012), or by averaging the network predictions over transformations of the input (feature averaging) (Zhang et al., 2017), or by training on a dataset augmented with these transformations (data augmentation) (Krizhevsky et al., 2012). Whereas some inductive biases can easily be encoded into the learning algorithm (e.g. with convolutions), the preference over functions is sometimes implicit and not intended by the designer of the learning system, and it is sometimes not obvious how to turn an inductive bias into a machine learning method, this conversion often being the core contribution of machine learning papers.

Inductive Bias	Corresponding Property
High-level variables play a **causal** role	Learning representations of latent entities/attributes
Changes in distribution are due to causal **interventions**	The source of changes in distribution is sparse and localized in the appropriate semantic space
Knowledge is generic, defined over abstract variables, and can be applied on different instances	Factorizing knowledge in terms of abstract variables and reusable functions that encapsulate how these variables interact with each other
Sparsity of dependencies	Learned functions operate on a sparse set of variables (like arguments in typed-programming languages)
Short causal chains	Causal chains used to perform learning or inference (to obtain explanations or plans for achieving some goal) are broken down into short causal chains of events that may be far in time from each other
Context-dependent processing involving goals, top-down influence, and bottom-up competition	Top-down contextual information is dynamically combined with bottom-up sensory signals at every level of the hierarchy of computations relating low-level and high-level representations

Table 2. Proposed additional inductive biases for deep learning: much progress has been made in learning representation of high level variables (entities or objects). Much more progress is needed on other inductive biases such as the ones listed above. It would also be useful to integrate these inductive biases into a unified architecture.

Inductive Biases as Data. We can think of inductive biases or priors and built-in structure as "training data in disguise", and one can compensate lack of sufficiently powerful priors by more data (Welling, 2019). Interestingly, different inductive biases may be equivalent to more or less data (even possibly exponentially more data): we suspect that inductive biases based on a form of compositionality (like distributed representations (Pascanu et al., 2013b), depth (Montufar et al., 2014) and attention (Bahdanau et al., 2014; Vaswani et al., 2017)) can potentially also provide a larger advantage (to the extent that they apply well to the function to be learned). In general, priors can be imperfect and this shows most with large datasets. Even for good priors, the advantage of inductive biases may be smaller on very large datasets, which suggests that transfer settings (where only few examples are available for the new distribution) are interesting to evaluate the advantage of inductive biases and of their implementation.

Agency, Sequential Decision Making and Non-Stationary Data Streams. The classical framework for machine learning is based on the assumption of identically and independently distributed data (i.i.d.), i.e test data has the same distribution as the training data. This is a very important assumption, because if we did not have that assumption, then we would not be able to say anything about generalization to new examples from the same distribution. Unfortunately, this assumption is too strong, and reality is not like this, especially for agents taking decisions one at a time in an environment from which they also get observations. The distribution of observations seen by an agent may change for many reasons: the agent acts (intervenes) in the environment, other agents intervene in the environment, or

simply our agent is learning and exploring, visiting different parts of the state-space as it does so, discovering new parts of it along the way, thus experiencing non-stationarities along the way. Although sequential decision-making is ubiquitous in real life, there are scenarios where thinking about these non-stationarities may seem unnecessary (like object recognition in static images). However, if we want to build learning systems which are robust to changes in distribution, it may be necessary to train them in settings where the distribution changes! And then of course there are applications of machine learning where the data is sequential and non-stationary (like historical records of anything) or even more so, where the learner is also an agent or is an agent interacting with other agents (like in robotics, autonomous driving or dialogue systems). That means we may need to go away from large curated datasets typical of supervised learning frameworks and instead construct non-stationary controllable environments as the training grounds and benchmarks for our learners. This complicates the task of evaluating and comparing learning algorithms but is necessary and we believe, feasible, e.g. see (Yu et al., 2017; Packer et al., 2018; Chevalier-Boisvert et al., 2018a; Dulac-Arnold et al., 2020; Ahmed et al., 2020).

Transfer Learning and Continual Learning. Instead of a fixed data distribution and searching for an inductive bias which works well with this distribution, we are thus interested in transfer learning (Pratt et al., 1991; Pratt, 1993) and continual learning (Ring, 1998) scenarios, with a potentially infinite stream of tasks, and where the learner must extract information from past experiences and tasks to improve its learning speed (i.e., sample complexity, which is different from asymptotic performance which is currently the standard) on future and yet unseen tasks. Suppose the learner faces a sequence of tasks, A, B, C and then we want the learner to perform well on a new task D. Short of any assumptions it is nearly impossible to expect the learner to perform well on D. However if there is some shared structure, between the transfer task (i.e task D) and source tasks (i.e tasks A, B and C), then it is possible to generalize or transfer knowledge from the source task to the target task. Hence, if we want to talk meaningfully about knowledge transfer, it is important to talk about the assumptions on the kind of data distribution that the learner is going to face, i.e., (a) what they may have in common, what is stable and stationary across the environments experienced and (b) how they differ or how changes occur from one to the next in case we consider a sequential decision-making scenario. This division should be reminiscent of the work on *meta-learning* (Bengio et al., 1990; Schmidhuber, 1987; Finn et al., 2017; Ravi & Larochelle, 2016), which we can understand as dividing learning into slow learning (of stable and stationary aspects of the world) and fast learning (of task-specific aspects of the world). This involves two time scales of learning, with an outer loop for meta-learning of meta-parameters and an inner loop for regular learning of regular parameters. In fact we could have more than two time scales (Clune, 2019): think about the outer loop of evolution, the slightly faster loop of cultural learning (Bengio, 2014) which is somewhat stable across generations, the faster learning of individual humans, the even faster learning of specific tasks and new environments within a lifetime, and the even faster inner loops of motor control and planning which adapt policies to the specifics of an immediate objective like reaching for a fruit. Ideally, we want to build an understanding of the world which shifts as much of the learning to the slower and more stable parts so that the inner learning loops can succeed faster, requiring less data for adaptation.

Systematic Generalization and Out-of-Distribution Generalization. In this paper, we focus on the objective of out-of-distribution (OOD) generalization, i.e., generalizing outside of the specific distribution(s) from which training observations were drawn. A more

general way to conceive of OOD generalization is with the concept of sample complexity in the face of new tasks or changed distributions. One extreme is zero-shot OOD generalization while the more general case, often studied in meta-learning setups, involves k-shot generalization (from k examples of the new distribution).

Whereas the notions of OOD generalization and OOD sample complexity tell us what we want to achieve (and hint at how we might measure it) they say nothing about how to achieve it. This is where the notion of *systematic generalization* becomes interesting (Smolensky, 1988; Fodor & Pylyshyn, 1988; Marcus, 1998; McClelland et al., 1987). Systematic generalization is a phenomenon which was first studied in linguistics (Lake & Baroni, 2017; Bahdanau et al., 2018) because it is a core property of language: the meaning for a novel composition of existing concepts (e.g. words) can be derived systematically from the meaning of the composed concepts. This very clearly exists in language, but humans benefit from it in other settings, e.g., understanding a new object by combining properties of different parts which compose it. Systematic generalization even makes it possible to generalize to new combinations that have zero probability under the training distribution: it is not just that they did not occur in the training data, but that even if we had seen an infinite amount of training data from our training distribution, we would not have any sample showing this particular combination. For example, when you read a science fiction scenario for the first time, that scenario could be impossible in your life, or even in the aggregate experiences of billions of humans living today, but you can still imagine it and make sense of it (e.g., predict the end of the scenario from the beginning). Empirical studies of systematic generalization were performed by (Bahdanau et al., 2018, 2019), where particular forms of combinations of linguistic concepts were present in the training distribution but not in the test distribution, and current methods take a hit in performance, whereas humans would be able to answer such questions easily.

Humans use inductive biases providing forms of compositionality, making it possible to generalize from a finite set of combinations to a larger set of combinations of concepts. Deep learning already benefits from a form of compositional advantage with distributed representations (Hinton, 1984; Bengio & Bengio, 2000; Bengio et al., 2001), which are at the heart of why neural networks work so well. There are theoretical arguments about why distributed representations can bring a potentially exponential advantage (Pascanu et al., 2013b), if this matches properties of the underlying data distribution. Another advantageous form of compositionality in deep nets arises from the depth itself, i.e., the composition of functions, with provable up to exponential advantages under the appropriate assumptions (Montufar et al., 2014). However, a form of compositionality that we propose here and should be better incorporated in deep learning is the form called systematicity (Lake & Baroni, 2018) defined by linguists, and more recently systematic generalization in machine learning papers (Bahdanau et al., 2018; Ruis et al., 2020; Akyürek et al., 2020).

Current deep learning methods tend to overfit the training *distribution*. This would not be visible by looking at a test set from the same distribution as the training set, so we need to change our ways of evaluating the success of learning because we would like our learning agents to generalize in a systematic way, out-of-distribution. This only makes sense if the new environment has enough shared components or structure with previously seen environments, which corresponds to certain assumptions on distributional changes, bringing back the need for appropriate inductive biases, about distributions (e.g., shared components) as well as about how they change (e.g., via agents' interventions).

The structure of next two sections is as follows: In section 3.3, we motivate some of the system-2 inductive biases inspired by human cognition. We think endowing machine learning systems with an efficient implementation of these inductive biases could improve (there's an ample evidence already but much more progress needs to be made to achieve human-level AI) the generalization and adaptation performance of the machine learning models. In section 3.4, we open a parenthesis to review material on causal dependencies to deepen some of the discussion made in section 3.3 on inductive biases linked to the causal nature of high-level semantic variables.

3.3. Inductive biases based on higher-level cognition as a path towards systems that generalize better OOD

Synergy between AI research and cognitive neuroscience. Our aim is to take inspiration from (and further develop) research into the cognitive science of conscious processing, to deliver greatly enhanced AI, with abilities observed in humans thanks to high-level reasoning leading among other things to greater abilities to face unusual or novel situations by reasoning, compositionally reusing existing knowledge and being able to communicate about that. At the same time, new AI models could drive new insights into the neural mechanisms underlying conscious processing, instantiating a virtuous circle. Machine learning procedures have the advantage that they can be tested for their effective learning abilities, and in our case in terms of out-of-distribution abilities or in the context of causal environments changing due to interventions, e.g., as in (Ahmed et al., 2020). Because they have to be very formal, AI models can also suggest hypotheses for how brains might implement an equivalent strategy with biological machinery. Testing these hypotheses could in turn provide more understanding about how brains solve the same problems and help to refine the deep learning systems.

3.3.1. Conscious vs Unconscious Processing in Brains

Imagine that you are driving a car from your office, back home. You do not need to pay a lot of attention to the road and you can talk to the passenger. Now imagine encountering a road block due to construction: you have to pay more attention, you have to be on lookout for new information, if the passengers starts talking to you, then you may have to tell the person, "please let me drive". It is interesting to consider that when humans are confronted with a new situation, very commonly they require their *conscious* attention (Carlson & Dulany, 1985; Newman et al., 1997). In the driving example, when there is a road block you need to pay attention in order to think through what to do next, and you probably don't want to be disturbed, because your conscious attention can only focus on one thing at a time.

There is something in the way humans process information which seems to be different – both functionally and in terms of the neural signature in the brain – when we deal with conscious processing and novel situations (changes in the distribution) which require our conscious attention, compared to our habitual routines. In those novel situations, we generally have to *think*, *focus* and *attend* to specific elements of our perception, actions or memories and sometimes inhibit our reactions based on context (e.g., facing new traffic rules or a road block). Why would humans have evolved to deal with such an ability with changes in distribution? Maybe simply because life experience is highly non-stationary.

System 1 and System 2. Cognitive scientists distinguish (Schneider et al., 1982; Redgrave et al., 2010; Botvinick et al., 2001a,b; Mozer et al., 2001; Bargh, 1984) *habitual* versus *controlled* processing, where the former correspond to default behaviors, whereas the

latter require attention and *mental effort*. Daniel Kahneman introduced the framework of fast and slow thinking (Kahneman, 2011), and describes the *system 1* and *system 2* styles of processing in our brain. Some tasks can be achieved using only system 1 abilities whereas others also require system 2 and conscious processing. There are also notions of explicit (verbalizable) knowledge and explicit processing (which roughly correspond to system 2) and implicit (intuitive) knowledge and corresponding system 1 neural computations. The default (or unconscious) processing of system 1 can take place very rapidly (as fast as about 100ms) and mobilize many areas of the brain in parallel. On the other hand, controlled (or conscious) processing involves a sequence of thoughts, usually verbalizable, typically requiring seconds to achieve. Whereas we can act in fast and precise habitual ways without having to think consciously, the reverse is not true: controlled processing (i.e., system 2 cognition) generally requires unconscious processing to perform much of its work. It is as if the conscious part of the computation was just the top-level program and the tip of the iceberg. Yet, it seems to be a very powerful one, which makes it possible for us to solve new problems creatively by recombining old pieces of knowledge, to reason, to imagine explanations and future outcomes, to plan and to apply or discover causal dependencies. It is also at that level that we interface with other humans through natural language. And when a word refers to a complex concept for which we do not have a clear verbalizable and precise explanation (like how we manage to drive our bike), we can still name it and reason about how it relates with other pieces of knowledge, etc. Even imagination and planning (which are hallmarks of system 2 abilities) require system 1 computations to sample candidate solutions to a problem (from a possibly astronomical number, which we never have to explicitly examine).

Our brain seems to thus harbour two very different types of knowledge: the kind we can explicitly reason about and communicate verbally (system 2 knowledge) and the kind that is intuitive and implicit (system 1 knowledge). When we learn something new, it typically starts being represented explicitly, and then as we practice it more, it may migrate to a different, implicit form. When you learn the grammar of a new language, you may be given some set of rules, which you try to apply on the fly, but that requires a lot of effort and is done painfully slowly. As you practice this skill, it can gradually migrate to a habitual form, you make less mistakes (for the common cases), you can read / translate / write more fluently, and you may even eventually forget the original rules. When a new rule is introduced, you may have to move back some of that processing to system 2 computation to avoid inconsistencies. It looks as if one of the key roles of conscious processing is to integrate different sources of knowledge (from perception and memory) in a coherent way.

The Global Worskpace Theory. The above division of labour is at the heart of the cognitive neuroscience Global Workspace Theory (or GWT) from Baars (Baars, 1993, 1997) and its extension, the Global Neuronal Workspace model (Shanahan, 2006, 2010, 2012; Dehaene & Changeux, 2011; Dehaene et al., 2017; Dehaene, 2020). The GWT suggests an architecture allowing specialist components to interact. The key claim of the GWT is the existence of a shared representation—sometimes called a blackboard (McClelland, 1986), sometimes a workspace—that can be modified by any selected specialist and whose content is broadcast to all specialists. That selection is based on a form of attention and can correspond to dynamically selecting (based on the input) a module or a few modules in a modular neural net that are most appropriate for a particular context and task. The basic idea of deep learning frameworks inspired by the GWT is to explore a similar communication and coordination scheme for a neural net comprising of distinct modules (Shanahan, 2006, 2005). The GWT theory posits that conscious processing revolves around a communication

bottleneck between selected parts of the brain which are called upon when addressing a current task. There is a threshold of relevance beyond which information which was previously handled unconsciously gains access to this bottleneck, instantiated in a working memory. When that happens, that information is broadcast to the whole brain, allowing the different relevant parts of it to synchronize, forcing each module to learn to exchange with other modules in a way that allows swapping one module for another as source or destination of communicated content, i.e., with a shared "language". These shared representations can be interpreted by many other modules. This gives rise to semantic representations that are not tied to a particular modality but can be triggered by any of the sensory channels. As we argue throughout this paper, this makes it possible to flexibly obtain new combinations of pieces of knowledge, enabling a compositional advantage aligned with the needs of systematic generalization out-of-distribution.

3.3.2. Attention as dynamic information flow.

The GWT suggests a fleeting memory capacity in which only one consistent content can be dominant at any given moment, which suggests a sharper form of attention than the soft attention currently dominant in deep learning and described below. Attention is about sequentially selecting what computation to perform on what quantities. Let us consider a machine translation task from English to French. To obtain a good translation generating the next French word, we normally focus especially on the "right" few words in the source English sentence that may be relevant to do the translation. This is the motivation that stimulated our work on content-based soft self-attention (Bahdanau et al., 2014) but may also be at the heart of conscious processing in humans as well as in future deep learning systems with both system 1 and system 2 abilities.

Content-Based Soft Attention. Soft attention forms a soft selection of one element (or multiple elements) from a set of elements at the previous level of computations, i.e we are taking a convex combination of the values of the elements at the previous level. These convex weights are coming from a softmax that is conditioned on how each of the elements' key vector matches some query vector. In a way, attention is parallel, because computing these attention weights considers all the possible elements in some set, yielding a score for each of them, to decide which of them are going to receive the most attention. With stochastic hard-attention (Xu et al., 2015) one samples from a distribution over elements to choose the attended content, whereas with soft attention (Bahdanau et al., 2014) one mixes these contents with different positive convex weights. Content-based attention also introduces a non-local inductive bias into neural network processing, allowing it to infer long-range dependencies that might be difficult to discern if computations are biased by local proximity. Attention is at the heart of the current state-of-the art NLP systems (Devlin et al., 2018a; Brown et al., 2020) and is the standard tool for memory-augmented neural networks (Graves et al., 2014a; Sukhbaatar et al., 2015; Gulcehre et al., 2016; Santoro et al., 2018). Attention and memory can also help address the problem of credit assignment through long-term dependencies (Ke et al., 2018; Kerg et al., 2020) by creating dynamic skip connections through time (i.e., a memory access) which unlock the problems of vanishing gradients and learning long-term dependencies (Hochreiter, 1991a; Bengio et al., 1994a). Attention also transforms neural networks from machines that are processing vectors (e.g., each layer of a deep net), to machines that are processing sets, more particularly sets of key/value pairs, as with Transformers (Vaswani et al., 2017; Santoro et al., 2018). Soft attention uses the

product of a *query* (or *read key*) represented as a matrix Q of dimensionality $N_r \times d$, with d the dimension of each key, with a set of N_o objects each associated with a *key* (or *write-key*) as a row in matrix K^T ($N_o \times d$), and after normalization with a softmax yields outputs in the convex hull of the *values* (or *write-values*) V_i (row i of matrix V). The result is

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V,$$

where the softmax is applied to each row of its argument matrix, yielding a set of convex weights. With soft attention, one obtains a convex combination of the values in the rows of V, whereas stochastic hard attention would sample one of the value vectors with probability equal to that weight. If the soft attention is focused on one element for a particular row (i.e., the softmax is saturated), we get deterministic hard attention: only one of the objects is selected and its value copied to row j of the result. Note that the d dimensions in the key can be split into *heads* which then have their attention matrix and write values computed separately. Note that hard attention is more biologically plausible (we only see one interpretation of the Necker cube (Cohen, 1959) at once, and have one thought at a time) but soft attention enables end-to-end training and has been the most commonly used in deep learning architectures up to now,. e.g., with transformers (Vaswani et al., 2017). However, there is recent evidence (Liu et al., 2021) that if the communication bottleneck is discretized, better OOD generalization is observed, maybe because the resulting simpler lingua franca would make it easier to swap one module for another in the attention-controlled communication between modules.

Attention as dynamic connections. We can think of attention as a way to create a dynamic connection between different blocks of computation, whereas in the traditional neural net setting, connections are fixed. On the receiving end (downstream module) of an attention-selected input, it is difficult to tell from the selected value vector from where it comes (among the selected upstream modules which competed for attention). To resolve this, it would make sense that the information being propagated along with the selected value includes a notion of key or type or name, i.e., of where the information comes from, hence creating a form of indirection (a reference to where the information came from, which can be passed to downstream computations).

Attention implements variable binding. When the inputs and outputs of each of the modules are a set of objects or entities (each associated with a key and value vector), we have a generic object-processing machine which can operate on "variables" in a sense analogous to variables in a programming language: as interchangeable arguments of functions. Because each object has a key embedding (which one can understand both as a name and as a type), the same computation can be applied to any variable which fits an expected "distributed type" (specified by a query vector). Each attention head then corresponds to a typed argument of the function computed by the factor. When the key of an object matches the query of head k, it can be used as the k-th input vector argument for the desired computation. Whereas in regular neural networks (without attention) neurons operate on fixed input variables (the neurons which are feeding them from the previous layer), the key-value attention mechanisms make it possible to select on the fly which variable instance (i.e. which entity or object) is going to be used as input for each of the arguments of some computation, with a different set of query embeddings for each argument head. The computations performed on the selected inputs can be seen as *functions with typed arguments*, and attention is used to *bind* their formal argument to the selected input, albeit in a soft differentiable way (that mixes multiple possibilities) in the case of soft attention. Type constraints have already been found useful in

identification for causal discovery (Brouillard et al., 2022). Current attention-based neural network already implement key-value-query soft attention mechanism (as above). What is missing is an ability to handle discrete types, hard (but possibly stochastic) choices of arguments, and more powerful inference machinery that uses not just type matching but is also able to reason about which modules and variables should be composed in a given context.

3.3.3. Blend of Serial and Parallel Computations.

From a computational perspective, one hypothesis about the dynamics of communication between different modules is that different modules generally act in parallel and receive inputs from other modules. However, when they do need to communicate information with another *arbitrary* module, the information has to go through a routing bottleneck (the global workspace) controlled by an attention mechanism. Because so few elements can be put in coherence at each step of the GWT selection, the inference process generally requires several such steps, leading to the highly sequential nature of system 2 computation (compared with the highly parallel nature of system 1 computation). The contents which have thus been selected are essentially the only ones which can be committed to memory, starting with short-term memory. Working memory refers to the ability of the brain to operate on a few recently accessed elements (i.e., those in short-term memory) (Baddeley, 1992; Cowan, 1999). These elements can be remembered and have a heavy influence on the next thought, action or perception, as well as on what learning focuses on, possibly playing a role similar to desired outputs, goals or targets in supervised learning for system 1 computations.

Partial State. From an RL perspective, it is interesting to note that if the GWT holds an important part of the state (including imagined future states, when planning), it does not describe all the aspects of the environment, only a handful of them, as already explored in the RL litterature (Zhao et al., 2021). This is different from standard RL approaches where the input (or the sequence of past inputs) is mapped to a fixed-size (estimated and latent) state vector. The GWT suggests instead that, besides long-term memory content (which mostly does not change), the rapidly changing state should be seen as a very small set of entities (e.g., objects or particular attributes of objects, and their relation), with an information content similar to that of a single sentence. This suggests neural net architectures in which very few modules and specific (variable, value) pairs are selected at every inference step, based on those that were recently selected, the current sensory input and the current contents of memory (which can also compete for write-access to the workspace). Only the selected modules would be under pressure to adapt when the result of the combination needs to be tuned, leading to selective adaptation similar to that explored by (Bengio et al., 2019) (see Section 3.4.3 above) where just a few relevant modules need to adapt to a change in distribution.

System 2 to System 1 Consolidation. As an agent, a human being is facing frequent changes because of their actions or the actions of other agents in the environment. Most of the time, humans follow their habitual policy, but tend to use system 2 cognition when having to deal with unfamiliar settings. It allows humans to generalize out-of-distribution in surprisingly powerful ways, and understanding this style of processing would help us build these abilities in AI as well. This is illustrated with our early example of driving in an area with unfamiliar traffic regulations, which requires full conscious attention (Section 3.3.1). This observation suggests that system 2 cognition is crucial in order to achieve the kind of flexibility and robustness to changes in distribution required in the natural world (Shenhav

et al., 2017; Kool & Botvinick, 2018). It looks like current deep learning systems are fairly good at perception and system 1 tasks. They can rapidly produce an answer (if you have parallel computing like that of GPUs) through a complex calculation which is difficult (or impossible) to dissect into the application of a few simple verbalizable operations. They require a lot of practice to learn and can become razor sharp good at the kinds of data they are trained on. On the other hand, humans enjoy system 2 abilities which permit fast learning (I can tell you a new rule in one sentence and you do not have to practice it in order to be able to apply it, albeit awkwardly and slowly at first) and systematic generalization, both of which should be important characteristics of the next generation of deep learning systems.

Between-Modules Interlingua and Communication Topology. If the brain is composed of different modules, it is interesting to think about what code or lingua franca is used to communicate between them, such that it can lead to interchangeable pieces of knowledge being dynamically selected and combined to solve a new problem. The GWT bottleneck may thus also play a role in forcing the emergence of such a lingua franca (Baars, 1997; Koch, 2004; Shanahan, 2006): the same information received by module A (e.g. "there is a fire") can come from any other module (say B, which detected a fire by smell, or C which detected a fire by sight). Hence B and C need to use a compatible representation which is broadcast via the GWT bottleneck for A's use. Again, we see the crucial importance of attention mechanisms to force the emergence of shared representations and indirect references exchanged between the modules via the conscious bottleneck. However, the GWT bottleneck is by far not the only way for modules to communicate with each other. Regarding the topology of the communication channels between modules, it is known that modules in the brain satisfy some spatial topology such that the computation is not all-to-all between all the modules. It is plausible that the brain uses both fixed local or spatially nearby connections as well as the global broadcasting system with top-down influence. We also know that there are hierarchical communication routes in the visual cortex (on the path from pixels to object recognition), and we know how successful that has been in computer vision with convnets. Combining these different kinds of inter-module communication modalities in deep network thus seems well advised as well (Watts & Strogatz, 1998; Latora & Marchiori, 2001; Rahaman et al., 2020): (1) Modules which are near each other in the brain layout can probably communicate directly without the need to clog the global broadcast channel (and this would not be reportable consciously). (2) Modules which are arbitrarily far from each other in the spatial layout of the brain can exchange information via the global workspace, following the theatre analogy of Baars' GTW. The other advantage of this communication route is of course the exchangeability of the sources of information being broadcast, which we hypothesize leads to better systematic generalization. The role of working memory in the GWT is not just as a communication buffer. It also serves as a blackboard (or analogously the "registers" in CPUs) where operations can be done locally to improve coherence. This enables a coherence-seeking mechanism: the different modules (especially the active ones) should adopt a configuration of their internal variables (and especially the more abstract entities they communicate to other modules) which is consistent with what other active modules "believe". It is possible, that a large part of the functional role of conscious processing is for that purpose, which is consistent with the view of the working memory as a central element of the inference machinery seeking to obtain coherent configurations of the variables interacting according to some piece of knowledge (such as a factor of the factor graph, a causal dependency).

System 2 inductive biases

We are proposing to take inspiration from cognition and build machines which integrate two very different kinds of representations and computations corresponding to the system 1 / implicit / unconscious vs system 2 / explicit / conscious divide.

This paper is about inductive biases not yet sufficiently integrated in state-of-the-art deep learning systems but which could help us achieve these system 2 abilities. In the next subsection, we summarize some of these system 2 inductive biases.

3.3.4. Semantic Representations Describing Verbalizable Concepts

Conscious content is revealed by reporting it, often with language (Colagrosso & Mozer, 2004). This suggests that high-level variables manipulated consciously are closely related with their verbal forms (like words and phrases). This yields maybe the most influential inductive bias we want to consider in this paper: that *high-level variables (manipulated consciously) are generally verbalizable*. To put it in simple terms, we can imagine the high-level semantic variables captured at this top level of a representation to be associated with single words (although we can also use words to identify some lower-level variables). In practice, the notion of word is not always the same across different languages, and the same semantic concept may be represented by a single word or by a phrase. There may also be more subtlety in the mental representations (such as accounting for uncertainty, concept representation and continuous-valued properties) which is not always or not easily well reflected in their verbal rendering. Much of what our brains know actually cannot be easily translated in natural language and forms the content of system 1 knowledge. This means that system 2 (verbalizable) knowledge is incomplete: words are mostly pointers to knowledge which belongs to system 1 and thus is in great part not consciously accessible. The system 2 inductive biases do not need to cover all the aspects of our internal model of the world (they couldn't), only those aspects of our knowledge which we are able to communicate with language. The rest would have to be represented in pure system 1 (non system 2) machinery, such as in an encoder-decoder that could relate low-level actions and low-level perception to semantic variables that can be operated on at the system-2 level. If there is some set of properties that apply well to some aspects of the world, then it would be advantageous for a learner to have a subsystem that takes advantage of these properties (the inductive priors described here) and a subsystem which models the other aspects. These inductive priors then allow faster learning and potentially other advantages like systematic generalization, at least concerning these aspects of the world which are consistent with these assumptions (system 2 knowledge, in our case).

High-level representations describe verbalizable concepts

There is a simple lossy mapping from semantic representations going through the GWT bottleneck to natural language expressions. This is an inductive bias which could be exploited in grounded language learning scenarios (Winograd, 1972; Hermann et al., 2017; Chevalier-Boisvert et al., 2018a; Hill et al., 2019) where we couple language data with observations and actions by an agent.

This suggests that natural language understanding systems should be trained in a way that couples natural language with what it refers to. This is the idea of *grounded language learning*. It would put pressure on the top-level representation so that it captures the kinds of concepts expressed with language. One can view this as a form of weak supervision, where we don't force the top-level GWT representations to be human-specified labels, only that there is a simple relationship between these representations and utterances which humans would often associate with the corresponding meaning. Our discussion about causality should also suggest that passive observation may be insufficient: in order to capture the causal structure understood by humans, it may be necessary for learning agents to be embedded in an environment in which they can act and thus discover its causal structure (Binz & Schulz, 2022; Kosoy et al., 2022). Studying this kind of setup was the motivation for our work on the Baby AI environment (Chevalier-Boisvert et al., 2018a).

3.3.5. Semantic Variables Play a Causal Role and Knowledge about them is Modular

Biological phenomena such as bird flocks have inspired the design of several distributed multi-agent systems, for example, swarm robotic systems, sensor networks, and modular robots. Despite this, most machine learning models employ the opposite inductive bias, i.e., with all elements (e.g., artificial neurons) interacting all the time. The GWT (Baars, 1997; Dehaene, 2020) also posits that the brain is composed in a modular way, with a set of expert modules which need to communicate but only do so sparingly and via a bottleneck through which only a few selected bits of information can be squeezed at any time. If we believe that theory, these selected elements are the concepts present to our mind at any moment, and a few of them are called upon and joined in working memory in order to reconcile the interpretations made by different modular experts across the brain. The decomposition of knowledge into recomposable pieces, a hallmark of classical AI based on rules (Russell, 2010) also makes sense as a requirement for obtaining systematic generalization (Bahdanau et al., 2018): conscious attention would then select which expert and which concepts (which we can think of as variables with different attributes and values) interact with which pieces of knowledge (which could be verbalizable rules or non-verbalizable intuitive knowledge about these variables) stored in the modular experts. On the other hand, the modules which are not brought to bear in this conscious processing may continue working in the background in a form of default or habitual computation (which would be the form of most of perception). For example, consider the task of predicting from pixel-level information the motion of balls sometimes colliding against each other as well as the walls. It is interesting to note that all the balls follow their default dynamics, and only when balls collide do we need to intersect information from several bouncing balls in order to make an inference about their future states. Saying that the brain modularizes knowledge is not sufficient, since there could be a huge number of ways of factorizing knowledge in a modular way. We need to think about the desired properties of modular decompositions of the acquired knowledge, and we propose here to take inspiration from the causal perspective on understanding how the world works, to help us define both the right set of variables and their relationship.

Semantic variables are often also causal variables

We hypothesize that semantic variables are often also causal variables. Words in natural language often refer to agents (subjects, which cause things to happen), objects (which are controlled by agents), actions (often through verbs) and modalities or properties of agents, objects and actions (for example we can talk about future actions, as intentions, or we can talk about time and space where events happen, or properties of objects or of actions). However, note that we can also name many low-level (like pixels) and intermediate features (like L-shaped edges). It is thus plausible to assume that causal reasoning of the kind we can verbalize involves as variables of interest those semantic variables which we can name, and that they can be at any level of the processing hierarchy in the brain, including at the highest levels of abstraction, where signals from all modalities join, such as pre-frontal cortex (Cohen et al., 2000), and where concepts can be manipulated in a way that is not specific to a single modality.

The connection between causal representations and modularity is profound: an assumption which is commonly associated with structural causal models is that it should break down knowledge about the causal influences into *independent mechanisms* (Peters et al., 2017b). As explained in Section 3.4.1, each such mechanism relates direct causes to their direct effect and knowledge of one such mechanism should not tell us anything about another mechanism (otherwise we should restructure our representations and decomposition of knowledge to satisfy this information-theoretic independence property). This is not about statistical independence of the corresponding random variables but about the algorithmic mutual information between the descriptions of these mechanisms. What it means practically and importantly for out-of-distribution adaptation is that if a mechanism changes (e.g. because of an intervention), the representation of that mechanism (e.g. the parameters used to capture a corresponding conditional distribution) may need to be adapted but that of the others do not need to be tuned to account for that change (Bengio et al., 2019).

These mechanisms may be organized in the form of a causal graph which scientists attempts to identify. The sparsity of the change in the joint distribution between the semantic variables (discussed more in Section 3.3.6) is different but related to a property of such high-level structural causal model: the sparsity of the graph capturing the joint distribution itself (discussed in Section 3.3.8). In addition, the causal structure, the causal mechanisms and the definition of the high-level causal variables tend to be stable across changes in distribution, as discussed in Section 3.3.7.

3.3.6. Local Changes in Distribution in Semantic Space

Consider a learning agent, like a learning robot or a learning child. What are the sources of non-stationarity for the distribution of observations seen by such an agent, assuming the environment is in some (generally unobserved) state at any particular moment? Two main sources are (1) the non-stationarity due to the environmental dynamics (including the learner's actions and policy) not having converged to an equilibrium distribution (or equivalently the mixing time of the environment's stochastic dynamics is longer than the lifetime of the learning agent) and (2) causal interventions by agents (either the learner of

interest or some other agents). The first type of change includes for example the case of a person moving to a different country, or a videogame player learning to play a new game or a never-seen level of an existing game. That first type also includes the non-stationarity due to changes in the agent's policy arising from learning. The second case includes the effect of actions such as locking some doors in a labyrinth (which may have a drastic effect on the optimal policy). The two types can intersect, as the actions of agents (including those of the learner, like moving from one place to another) contribute to the first type of non-stationarity.

Changes in distribution are localized in the appropriate semantic space

Let us consider how humans describe these changes with language. For many of these changes, they are able to explain the source of change with a few words (a single sentence, often). This is a very strong clue for our proposal to include as an inductive bias the assumption that *the source of most changes in distribution are localized in the appropriate semantic space*: only one or a few variables or mechanisms need to be modified to account for the change.

Note how humans will even create new words when they are not able to explain a change with a few existing words, with the new words corresponding to new latent variables, which when introduced, make the changes explainable "easily" (assuming one understand the definition of these variables and of the mechanisms relating them to other variables).

For system-2 distributional changes (due to interventions), we automatically get locality of the source of changes (which start at one or a few nodes of the causal graph). This is a plausible assumption since, by virtue of being localized in time and space, actions can only directly affect very few high-level variables, with other effects (on downstream variables) being consequences of the initial intervention. This sparsity of sources of change is a strong assumption which can put pressure on the learning process to discover high-level representations which have that property. Here, we are assuming that the learner has to jointly discover these high-level representations (i.e. how they relate to low-level observations and low-level actions) as well as how the high-level variables relate to each other via causal mechanisms.

3.3.7. Stable Properties of the World

Above, we have talked about changes in distribution due to non-stationarities, but there are aspects of the world that are stationary, which means that learning about them would eventually converge. In an ideal scenario, our learner has an infinite lifetime and the chance to learn everything about the world (a world where there are no other agents) and build a perfect model of it, at which point nothing is new and all of the above sources of non-stationarity are gone. In practice, only a small part of the world will be understood by the learning agent, and interactions between agents (especially if they are learning) will perpetually keep the world out of equilibrium. If we divide the knowledge about the world captured by the agent into the stationary aspects (which should converge) and the non-stationary aspects (which would generally keep changing), we would like to have as much knowledge as possible in the stationary category. The stationary part of the model might require many observations for it to converge, which is fine because learning these parts can be amortized over the whole lifetime (or even multiple lifetimes in the case of multiple cooperating cultural agents, e.g.,

in human societies). On the other hand, the learner should be able to quickly learn the non-stationary parts (or those the learner has not yet realized can be incorporated in the stationary parts), ideally because very few of these parts need to change, if knowledge is well structured. Hence we see the need for at least two speeds of learning, similar to the division found in meta-learning of learnable coefficients into meta-parameters on one hand (for the stable, slowly learned aspects) and parameters on the other hand (for the non-stationary, fast to learn aspects), as already discussed above in Section 3.2.

Stable v/s Unstable properties of the world

There should be several speeds of learning, with more stable aspects learned more slowly and more non-stationary or novel ones learned faster, and pressure to discover stable aspects among the quickly changing ones. This pressure would mean that more aspects of the agent's represented knowledge of the world become stable and thus less needs to be adapted when there are changes in distribution.

For example, consider scientific laws, which are most powerful when they are universal. At another level, consider the mapping between the perceptual input, low level actions, and the high-level semantic variables. An encoder that would implement this mapping should ideally be highly stable, or else downstream computations would need to track those changes (and indeed the low-level visual cortex seems to compute features that are very stable across life, contrary to high-level concepts like new visual categories). Causal interventions are taking place at a higher level than the encoder, changing the value of an unobserved high-level variable or changing one of the mechanisms. If a new concept is needed, it can be added without having to disturb other represented knowledge, especially if it can be learned as a composition of existing high-level features and concepts. We know from observing humans and their brain that new concepts which are not obtained from a combination of old concepts (like a new skill or a completely new object category not obtained by composing existing features) take more time to learn, while new high-level concepts which can be readily defined from other high-level concepts can be learned very quickly (as fast as with a single example or definition).

Another example arising from the analysis of causal systems is that causal interventions (which are in the non-stationary, quickly inferred or quickly learned category) may temporarily modify the causal graph structure (which specifies which variable is a direct cause of which) by breaking causal links (when we set a variable we break the causal link from its direct causes) but that most of the causal graph is a stable property of the environment. Hence, we need neural architectures which make it easy to quickly adapt the relationship between existing concepts, or to define new concepts from existing ones.

3.3.8. Sparse Factor Graph in the Space of Semantic Variables

Sparsity as to how variables and factors interact with each other

Our next inductive bias for high-level variables can be stated simply: the joint distribution between high-level concepts can be represented by a sparse factor graph.

Any joint distribution can be expressed as a factor graph (Kschischang et al., 2001; Frey, 2012; Kok & Domingos, 2005), but we claim that the ones which can be conveniently described with natural language have the property that they should be sparse. A factor graph is a particular factorization of the joint distribution. A factor graph is bipartite, with variable nodes on one hand and factor nodes on the other. Factor nodes represent dependencies between the variables to which they are connected. To illustrate the sparsity of verbalizable knowledge, consider knowledge graphs and other relational systems, in which relations between variables often involve only two arguments (i.e., two variables). In practice, we may want factors with more than two arguments, but probably not a lot more. A factor may capture a causal mechanism between its argument variables, and thus we should introduce an additional semantic element to these factors: each argument of a causal factor should either play the role of cause or of effect, making the bipartite graph directed.

It is easy to see that linguistically expressed knowledge satisfies this sparsity property by noting that statements about the world can be expressed with a sentence and each sentence typically has only a few words, and thus relates very few concepts. When we write "If I drop the ball, it will fall on the ground", the sentence clearly involves very few variables, and yet it can make a very strong prediction about the position of the ball. A factor in a factor graph involving a subset S of variables is simply stating a probabilistic constraint among these variables. It allows one to predict the value of one variable given the others (if we ignore other constraints or factors), or more generally it allows us to describe a preference for joint sets of values for a subset of S. The fact that natural language allows us to make such strong predictions conditioned on so few variables should be seen as surprising: it only works because the variables are semantic ones. If we consider the space of pixel values in images, it is very difficult to find such strongly predictive rules, e.g., to predict the value of one pixel given the value of three other pixels. What this means is that pixel space does not satisfy the sparsity prior associated with the proposed inductive bias.

We claim that the proposed inductive bias is closely related to the bottleneck of the GWT of conscious processing. Our interpretation of this restriction on write access in the GWT by a very small number of specialists selected on the fly by an attention mechanism is that it stems from an assumption on the form of the joint distribution between high-level variables whose values are broadcast. If the joint distribution factor graph is sparse, then only a few variables (those involved in one factor or a few connected factors) need to be synchronized at each step of an inference process, e.g., consider loopy belief propagation (Frey et al., 2001; Murphy et al., 2013). By constraining the size of the working memory, evolution may have thus enforced the sparsity of the factor graph. The GWT also makes a claim that the workspace is associated with the conscious contents of cognition, which can be reported verbally. One can also make links with the original von Neumann architecture of computers. In both the GWT and the von Neumann architecture, we have a communication bottleneck with in the former the working memory and in the latter the CPU registers where operations are performed. The communication bottleneck only allows a few variables to be brought to the nexus (working memory in brains, registers in the CPU) . In addition, the operations on these variables are extremely sparse, in the sense that they take very few variables at a time as arguments (no more than the handful in working memory, in the case of brains, and generally no more than two or three in typical assembly languages). This sparsity constraint is consistent with a decomposition of computation in small chunks, each involving only a few elements. In the case of the sparse factor graph assumption we only consider that sparsity

constraint for declarative knowledge (verbalizing "how the world works", its dynamics and statistical or causal structure).

This assumption about the joint distribution between the high-level variables at the top of our deep learning hierarchy is different from the assumption commonly found in many papers on disentangling factors of variation (Higgins et al., 2016; Burgess et al., 2018; Chen et al., 2018; Kim & Mnih, 2018; Locatello et al., 2019), where the high-level variables are assumed to be marginally independent of each other, i.e., their joint distribution factorizes into independent marginals. We think this deviates from the original goals of deep learning to learn abstract high-level representations which capture the underlying explanations for the data. Note that one can easily transform one representation (with a factorized joint) into another (with a non-factorized joint) by some transformation (think about the independent noise variables in a structural causal model, Section 3.4). However, we would then lose the properties introduced up to now (that each variable is causal and corresponds to a word or phrase, that the factor graph is sparse, and that changes in distribution can be originated to one or very few variables or factors).

Instead of thinking about the high-level variables as completely independent, we propose to see them as having a very structured joint distribution, with a sparse factor graph and other characteristics (such as dependencies which can be instantiated on particular variables from generic schemas or rules, described below). We argue that if these high-level variables have to capture semantic variables expressible with natural language, then the joint distribution of these high-level semantic variables must have sparse dependencies rather than being independent. For example, high-level concepts such as "table" and "chair" are not statistically independent, instead they come in very powerful and strong but sparse relationships. Instead of imposing a very strong prior of complete independence at the highest level of representation, we can have this slightly weaker but very structured prior, that the joint is represented by a sparse factor graph. Interestingly, recent studies confirm that the top-level variables in generative adversarial networks (GANs), which are independent by construction, generally do not have a semantic interpretation (as a word or short phrase), whereas many units in slightly lower layers do have a semantic interpretation (Bau et al., 2018).

Why not represent the causal structure with a directed graphical model? In these models, which are the basis of standard representations of causal structure (e.g., in structural causal models, described below), knowledge to be learned is stored in the conditional distribution of each variable (given its direct causal parents). However, it is not clear that this is consistent with the requirements of independent mechanisms. For example, typical verbally expressed rules have the property that many rules could apply to the same variable. Insisting that the independent units of knowledge are conditionals would then necessarily lump the corresponding factors in the same conditional. This issue becomes even more severe if we think of the rules as generic pieces of knowledge which can be reused to be applied to many different tuples of instances, as elaborated in the next subsection. Another reason for a formulation that is not constrained to an acyclic graph is that humans also reason about relations between variables at equilibrium (such as voltage and current), which can mutually be causes of each other (i.e., arrows can go both ways).

3.3.9. Variables, Instances and Reusable Knowledge Pieces

A standard graphical model is static, with a separate set of parameters for each conditional distribution (in a directed acyclic graph) or factor (in a factor graph). There are extensions

which allow parameter sharing, e.g. through time with dynamic Bayes nets (Spirtes et al., 2000), or in undirected graphical models such as Markov Networks (Kok & Domingos, 2005) which allow one to "instantiate" general "patterns" into multiple factors of the factor graph. Markov Networks can for example implement forms of recursively applied probabilistic rules. But they do not take advantage of distributed representations and other inductive biases of deep learning.

The inductive bias we are presenting here is that instead of separately defining specific factors in the factor graph (maybe each with a piece of neural network), each having its separate set of parameters, we would define generic factors, "schemas" or "factor templates". A schema, or generic factor is a reusable probabilistic relation, i.e., with argument variables which can be bound to instances (also discussed in (Rumelhart et al., 1986b)). A static instantiated rule is a thing like 'if John is hungry then he looks for food'. Instead, a more general rule is a thing like, 'for all X, if X is a human and X is hungry, then X looks for food' (with some probability). X can be bound to specific instances (or to other variables which may involve more constraints on the acceptable set). In classical symbolic AI, we have unification mechanisms to match together variables, instances or expressions involving variables and instances, and thus keep track of how variables can ultimately be 'bound' to instances (or to variables with more constraints on their attributes), when exploring whether some schema can be applied to some objects (instances or more generic objects) with properties (constituting a database of entities).

The proposed inductive bias is also inspired by the presence of such a structure in the semantics of natural language and the way we tend to organize knowledge according to relations, e.g., in knowledge graphs (Sowa, 1987). Natural language allows us to state rules involving variables and is not limited to making statements about specific instances.

Knowledge is generic and can be instantiated on different instances.

The independent mechanisms (with separate parameters) which specify dependencies between variables are generic, i.e., they can be instantiated in many possible ways to specific sets of arguments with the appropriate types or constraints.

What this means in practice is that we do not need to hold in memory the full instantiated graph with all possible instances and all possible mechanisms relating them (or worse, all the generic factor instantiations that are compatible with the data, in a Bayesian posterior). Instead, inference involves generating the needed pieces of the graph and even performing reasoning (i.e. deduction) at an abstract level, where nodes in the graph (random variables) stand not for instances but for sets of instances belonging to some category or satisfying some constraints. Whereas one can unfold a recurrent neural network or a Bayesian network to obtain the fully instantiated graph, in the case we are talking about, similarly to a Markov network, it is generally not feasible to do that. It means that inference procedures always look at a small piece of the (partially) unfolded graph at a time and they can reason about how to combine these generic schemas without having to fully instantiate them with concrete instances or concrete objects in the world. One way to think about this, inspired by how we do programming, is that we have functions with generic and possibly typed variables as arguments and we have instances on which a program is going to be applied. At any time (as you would have in Prolog), an inference engine must match the rules with the current

instances (so the types and other constraints between arguments are respected) as well as other elements (such as what we are trying to achieve with this computation) in order to combine the appropriate computations. It would make sense to think of such a computation controller, as an internal policy with attention and memory access as actions, to select which pieces of knowledge and which pieces of the short-term (and occasionally long-term) memory need to be combined in order to push new values in working memory (Shanahan & Baars, 2005; Shanahan, 2006; Baars, 1993, 1997).

An interesting outcome of such a representation is that one can apply the same knowledge (i.e knowledge specified by a schema which links multiple abstract entities together) to different instances (i.e different "object files" in cognitive psychology (Noles et al., 2005; Gordon & Irwin, 1996; Kahneman et al., 1992)). For example, you can apply the same laws of physics to two different balls that are visually different (and maybe have different colors and masses). This is also related to notions of arguments and indirection in programming. The power of such relational reasoning resides in its capacity to generate inferences and generalizations that are constrained by the roles that elements play, and the roles they can play may depend on the properties of these elements, but these schemas specify how entities can be related to each other in systematic (and possibly novel) ways. In the limit, relational reasoning yields universal inductive generalization from a finite and often very small set of observed cases to a potentially infinite set of novel instances, so long as those instances can be described by attributes (specifying types) allowing to bound them to appropriate schemas.

There are two forms of knowledge representation we have discussed: declarative knowledge or hypotheses, i.e., that can be verbalized (e.g. of facts, hypotheses, explicit causal dependencies, etc), and inference machinery used to reason with these pieces of knowledge. Standard graphical models only represent the declarative knowledge and typically require expensive but generic iterative computations (such as Monte-Carlo Markov chains) to perform approximate inference (Cowles & Carlin, 1996; Gilks et al., 1995). However, brains need fast inference (Gigerenzer & Goldstein, 1996), and most of the advances made with deep learning concern such learned fast inference computations. Doing inference using only the declarative knowledge (the graphical model) is very flexible (any question of the form "predict some variables given other variables or imagined interventions" can be answered) but also very slow. In general, searching for a good configuration of the values of top-level variables which is consistent with the given context is computationally intractable. However, different approximations can be made which trade-off computational cost for quality of the solutions found. This difference could also be an important ingredient of the difference between system 1 (fast and parallel approximate and inflexible inference) and system 2 (slower and sequential but more flexible inference). We also know that after system 2 has been called upon to deal with novel situations repeatedly, the brain tends to bake these patterns of response in habitual system 1 circuits which can do the inference job faster and more accurately but have lost some flexibility. When a new rule is introduced, the system 2 is flexible enough to handle it and slow inference needs to be called upon again. Neuroscientists have also accumulated evidence that the hippocampus is involved in replaying sequences (from memory or imagination) for consolidation into cortex (Alvarez & Squire, 1994; Hassabis et al., 2007) so that they can be presumably committed to cortical long-term memory and fast inference.

3.3.10. Relevant causal chains (for learning or inference) can be approximated as very short chains

In a *clock-based segmentation*, the boundaries between discrete time steps are spaced equally (Hihi & Bengio, 1995; Chung et al., 2016; Koutnik et al., 2014). In an *event-based segmentation*, the boundaries depend on the state of the environment, resulting in dynamic duration of intervals (Mozer & Miller, 1997). Our brains seem to segment streams of sensory inputs into meaningful representations of variable-length episodes and *events* (Suddendorf & Corballis, 2007; Ciaramelli et al., 2008; Berntsen et al., 2013; Dreyfus, 1985; Richmond & Zacks, 2017).

The detection of a relevant event in the temporal stream triggers information processing of the event. The psychological reality of event-based segmentation can be illustrated through a familiar phenomenon. Consider the experience of traveling from one location to another, such as from home to office. If the route is unfamiliar, as when one first starts a new job, the trip is confusing and lengthy, but as one gains more experience following the route, one has the sense that the trip becomes shorter. One explanation for this phenomenon is as follows. On an unfamiliar route, the orienting mechanism that detects novel events is triggered for a large number of such events over the course of the trip. In contrast, few novel events occur on a familiar route. If our perception of time is event-based, meaning that higher centers of cognition count the number of events occurring in a temporal window, not the number of milliseconds, then one will have the sense that a familiar trip is shorter than an unfamiliar trip.

Event segmentation allows functional representations that support temporal reasoning, an ability that arguably relies on neural circuits to encode and retrieve information to and from memory (Zacks et al., 2007; Radvansky & Zacks, 2017; Baldassano et al., 2017). Indeed, faced with a task, our brains appear to easily and *selectively* pluck context-relevant past information from memory, enabling both powerful multi-scale associations as well as flexible computations to relate temporally distant events. As we argue here, the ability of the brain to efficiently segment sensory inputs into events, and the ability to *selectively* recall information from the distant past based on the current context helps to efficiently propagate information (such as credit assignment or causal dependencies) over long time spans. Both at the cognitive and at the physiological levels, there is evidence of information "routing" mechanisms that enable this efficient propagation of information, although they are far from being sufficiently understood (Stocco et al., 2010; Ben-Yakov & Henson, 2018; Bonasia et al., 2018).

Relevant Causal Chains tend to be sparse.

Our next inductive bias is almost a consequence of the biases on causal variables and the bias on the sparsity of the factor graph for the joint distribution between high-level variables. Causal chains used to perform learning (to imagine counterfactuals and to propagate and assign credit) or inference (to obtain explanations or plans for achieving some goal) are broken down into short causal chains of events which may be far in time but linked by the top-level factor graph over semantic variables.

At least at a conscious level, humans are not able to reason about many such events at a time, due to the limitations on short-term memory and the bottleneck of conscious

processing Baars (1997). Hence it is plausible that humans would exploit an assumption on temporal dependencies in the data: that the most relevant ones only involve short dependency chains, or a small-depth graph of direct dependencies. Depth here refers to the longest path in the relevant graph of dependencies between events. What we showed earlier (Ke et al., 2018; Kerg et al., 2020) is that this prior assumption is the strongest ingredient to mitigate the issue of vanishing gradients that occurs when trying to learn long-term dependencies (Bengio et al., 1994a).

3.3.11. Context-dependent processing involving goals, top-down influence, and bottom-up competition

Successful perception in humans clearly relies on both top-down and bottom-up signals (Buschman & Miller, 2007; Beck & Kastner, 2009; McMains & Kastner, 2011; Kinchla & Wolfe, 1979; Rauss & Pourtois, 2013; McClelland & Rumelhart, 1981). Top-down information encodes relevant context, priors and preconceptions about the current scene: for example, what we might expect to see when we enter a familiar place. Bottom-up signals consist of what is literally observed through sensation. The best way to combine top-down and bottom-up signals remains an open question, but it is clear that these signals need to be combined in a way which is dynamic and depends on context - in particular top-down signals are especially important when stimuli are noisy or hard to interpret by themselves (for example walking into a dark room). Additionally, which top-down signals are relevant also changes depending on the context. It is possible that combining specific top-down and bottom-up signals that can be weighted dynamically (for example using attention) could improve robustness to distractions and noisy data.

In addition to the general requirement of dynamically combining top-down and bottom-up signals, it makes sense to do so at every level of the processing hierarchy to make the best use of both sources of information at every stage of that computation, as is observed in the visual cortex (with very rich top-down signals influencing the activity at every level).

Dynamic Integration of Bottom-up and Top-Down Information

In favour of architectures in which top-down contextual information is dynamically combined with bottom-up sensory signals at every level of the hierarchy of computations relating low-level and high-level representations.

3.4. Declarative Knowledge of Causal Structure

Whereas a statistical model captures a single joint distribution, a causal model captures a large family of joint distributions, each corresponding to a different intervention (or set of interventions), which modifies the unperturbed or default distribution (e.g., by removing parents of a node and setting a value for that node). Whereas the joint distribution $P(A, B)$ can be factored either as $P(A)P(B|A)$ or $P(B)P(A|B)$ (where in general both graph structures can fit the data equally well), only one of the graphs corresponds to the correct causal structure and can thus consistently predict the effect of interventions. The asymmetry is best illustrated by an example: if A is altitude and B is average temperature, we can see that intervening on A will change B but not vice-versa.

Preliminaries Given a set of random variables X_i, a Bayesian network is commonly used to describe the dependency structure of both probabilistic and causal models via a Directed Acyclic Graph (DAG). In this graph structure, a variable (represented by a particular node) is independent of all the other variables, given all the direct neighbors of a variable. The edge direction identify a specific factorization of the joint distribution of the graph's variables:

$$p(X_1, \ldots, X_n) = \prod_{i=1}^{m} p(X_i \mid \mathbf{PA}_i). \tag{3.1}$$

Structural causal models (SCMs). A Structural Causal Model (SCM) (Peters et al., 2017b) over a finite number M of random variables X_i given a set of *observables* $X_1, \ldots, X_M$ (modelled as random variables) associated with the vertices of a DAG G, is a set of structural assignments

$$X_i := f_i(X_{pa(i,C)}, N_i) \ , \quad \forall i \in \{1, \ldots, M\} \tag{3.2}$$

where f_i is a deterministic function, the set of noises $N_1, \ldots, N_m$ are assumed to be *jointly independent*, and $pa(i, C)$ is the set of parents (direct causes) of variable i under configuration C of the SCM directed acyclic graph, i.e., $C \in \{0, 1\}^{M \times M}$, with $c_{ij} = 1$ if node i has node j as a parent (equivalently, $X_j \in X_{pa(i,C)}$; i.e. X_j is a direct cause of X_i). Causal structure learning is the recovery of the ground-truth C from observational and interventional data, possibly yielding a posterior distribution over causal structures compatible with the data, and a neural network can be trained to generate graphs from that posterior (Deleu et al., 2022).

Interventions. Without experiments, or interventions i.e., in a purely-observational setting, it is known that causal graphs can be distinguished only up to a Markov equivalence class, i.e., the set of graphs compatible with the observed dependencies. In order to identify the true causal graph, the learner needs to perform interventions or experiments i.e., interventional data is generally needed (Eberhardt et al., 2012).

3.4.1. Independent Causal Mechanisms.

A powerful assumption about how the world works which arises from research in causality (Peters et al., 2017b) and briefly introduced earlier is that the causal structure of the world can be described via the composition of independent causal mechanisms.

Independent Causal Mechanisms (ICM) Principle. A complex generative model, temporal or not, can be thought of as composed of independent mechanisms that do not inform or influence each other. In the probabilistic case, this means a particular mechanism should not inform (in the information theory sense) or influence the other mechanisms.

This principle subsumes several notions important to causality, including separate intervenability of causal variables, modularity and autonomy of subsystems, and invariance (Pearl, 2009; Peters et al., 2017a).

This principle applied to the factorization in equation. 3.1, tells us that the different factors should be independent in the sense that (a) performing an intervention on one of the mechanisms $p(X_i|\mathbf{PA}_i)$ does not change any of the other mechanisms $p(X_j|\mathbf{PA}_j)$ $(i \neq j)$, (b) knowing some other mechanisms $p(X_i|\mathbf{PA}_i)$ $(i \neq j)$ does not give us information about any another mechanism $p(X_j|\mathbf{PA}_j)$.

3.4.2. Exploit changes in distribution due to causal interventions

Nature doesn't shuffle examples. Real data arrives to us in a form which is not iid, and so in practice what many practitioners of data science or researchers do when they

collect data is to *shuffle* it to make it iid. "Nature doesn't shuffle data, and we should not" Bottou (2019). When we shuffle the data, we destroy useful information about those changes in distribution that are inherent in the data we collect and contain information about causal structure. Instead of destroying that information about non-stationarities, we should use it, in order to learn how the world changes.

3.4.3. Relation between meta-learning, causality, OOD generalization and fast transfer learning

To illustrate the link between meta-learning, causality, OOD generalization and fast transfer learning, consider the example from (Bengio et al., 2019). We consider two discrete random variables A and B, each taking N possible values. We assume that A and B are correlated, without any hidden confounder. The goal is to determine whether the underlying causal graph is $A \to B$ (A *causes* B), or $B \to A$. Note that this underlying causal graph cannot be identified from observational data from a single (training) distribution p only, since both graphs are Markov equivalent for p, i.e. consistent with observational data of any size. In order to disambiguate between these two hypotheses, (Bengio et al., 2019) use samples from some transfer distribution $\tilde{p}$ in addition to our original samples from the training distribution p.

Without loss of generality, they fix the true causal graph to be $A \to B$, which is unknown to the learner. Moreover, to make the case stronger, they consider a setting called *covariate shift*, where they assume that the change (again, whose nature is unknown to the learner) between the training and transfer distributions occurs after an intervention on the cause A. In other words, the marginal of A changes, while the conditional $p(B \mid A)$ does not, i.e. $p(B \mid A) = \tilde{p}(B \mid A)$. Changes on the cause will be most informative, since they will have direct effects on B. (Bengio et al., 2019) find experimentally that this is sufficient to identify the causal graph, while (Priol et al., 2020) justify this with theoretical arguments in the case where the intervention is on the cause.

In order to demonstrate the advantage of choosing the causal model $A \to B$ over the anti-causal $B \to A$, (Bengio et al., 2019) compare how fast the two models can adapt to samples from the transfer distribution $\tilde{p}$. They quantify the speed of adaptation as the log-likelihood after multiple steps of fine-tuning via (stochastic) gradient ascent on the example wise log-likelihood, starting with both models trained on a large amount of data from the training distribution. They show via simulations that the model corresponding to the underlying causal structure adapts faster. Moreover, the difference between the quality of the predictions made by the causal and anti-causal models as they see more post-intervention examples is more significant when adapting on a small amount of data, of the order of 10 to 30 samples from the transfer distribution. Indeed, asymptotically, both models recover from the intervention perfectly and are not distinguishable. This is interesting because it shows that generalization from few examples (after a change in distribution) actually contains more signal about the causal structure than generalization from a lot of examples (whereas in machine learning we tend to think that more data is always better). (Bengio et al., 2019) make use of this property (the difference in performance between the two models) as a noisy signal to infer the direction of causality, which here is equivalent to choosing how to modularize the joint distribution. The connection to meta-learning is that in the inner loop of meta-learning we adapt to changes in the distribution, whereas in the outer loop we slowly converge towards a good model of the causal structure (which describes what is shared across environments and

interventions). Here the meta-parameters capture the belief about the causal graph structure and the default (unperturbed) conditional dependencies, while the inner loop parameters are those which capture the change in the graph due to the intervention.

(Ke et al., 2019) further expanded this idea to deal with more than two variables. To model causal relations and out-of-distribution generalization one can view real-world distributions as arising from the composition of causal mechanisms. Any change in distribution (e.g., when moving from one setting/domain to a related one) is attributed to changes in as few as possible (but at least one) of those mechanisms (Goyal et al., 2019b; Bengio et al., 2019; Ke et al., 2019). A do-intervention or hard intervention would set the value of a variable to some value irrespective of the causal parents of that variable, thus disconnecting that node from its parents in the causal graph. By inferring this graph surgery, an intelligent agent should be able to recognize and make sense of these sparse changes and quickly adapt their pre-existing knowledge to this new domain. A current hypothesis is that a causal graphical model defined on the appropriate causal variables would be more efficiently learned than one defined on the wrong representation. Preliminary work based on meta-learning (Ke et al., 2019; Dasgupta et al., 2019; Bengio et al., 2019) suggests that, parameterizing the correct variables and causal structures, the parameters of the graphical model capturing the (joint) observational distribution can be adapted faster to changes in distribution due to interventions. This comes as a consequence of the fact that fewer parameters need to be adapted to account for the intervention (Priol et al., 2020). In this sense, learning causal representations may bring immediate benefits to machine learning models in terms of reduced sample complexity.

3.4.4. Actions and affordances as part of the causal model

Understanding causes and effects is a crucial component of the human cognitive experience. Humans are agents and their actions change the world (sometimes only in little ways), and those actions can inform them about the causal structure in the world. Understanding that causal structure is important in order to plan further actions in order to achieve desired consequences, or to attribute credit to one's or others' actions, i.e., to understand and cope with changes in distribution occuring in the world. However, in realistic settings such as those experienced by a child or a robot, the agent typically does not have full knowledge of what abstract action was performed and needs to perform inference over that. The agent would thus have a causal model of latent causal variables (how they influence each other and relate to each other), an intervention model relating low-level actions with interventions (or intentions to change specific high-level variables), as well as an observation model (relating high-level causal variables and sensory observations). In addition to these models, it would have inference machinery associated with them, including a high-level policy generating goals (i.e. intentions to intervene in a particular way).

A human-centric version of this viewpoint is the psychological theory of affordances (Gibson, 1977; Cisek, 2007; Pezzulo & Cisek, 2016) that can be linked to predictive state representations in reinforcement learning: what can we do with an object? What are the consequences of these actions? Learning affordances as representations of how agents can cause changes in their environment by controlling objects and influencing other agents is more powerful than learning a data distribution. It would not only allow us to predict the consequences of actions we may not have observed at all, but it also allows us to envision which potentialities would result from a different mix of interacting objects and agents. This line of thinking is directly related to the work in machine learning and reinforcement learning on controllability of aspects of the environment (Bengio et al., 2017; Thomas et al., 2017). A clue about a

good way to define causal variables is precisely that there exist actions or skills to control one causal variable while not directly influencing most others (i.e., except as an effect of the causal variable which is being controlled). A learner thus needs to discover an intervention model (what actions give rise to what interventions), but the locality of interventions in the causal graph can also help the learner figure out a good representation space for causal variables.

3.5. Conclusions

To be able to handle dynamic, changing conditions, we want to move from deep statistical models which are able to perform system 1 tasks to deep structural models also able to perform system 2 tasks by taking advantage of the computational workhorse of system 1 abilities. Today's deep networks may benefit from additional structure and inductive biases to do substantially better on system 2 tasks, natural language understanding, out-of-distribution systematic generalization and efficient transfer learning. We have tried to clarify what some of these inductive biases may be, but much work needs to be done to improve that understanding and find appropriate ways to incorporate these priors in neural architectures and training frameworks. We have motivated these inductive biases in terms of expected (and observed in recent work) gains in terms of out-of-distribution generalization and fast adaptation in transfer settings rather than the standard test set from the same distribution as the training set. The general insight here is that the proposed inductive biases should help organize knowledge into the stable reusable parts that are likely to be useful in new settings and tasks (such as causal mechanisms), separating them from the more volatile pieces of information (the values of variables) that can be changed by agents (through causal intervention) or those that are affected by these changes and may vary across environments or tasks. Some of the more salient inductive biases we propose deserve especially more attention in deep learning research include (a) the fairly direct connection between high-level variables and natural language or more generally how humans communicate knowledge among them, i.e., we can verbalize our thoughts to a large extent and this can provide rich insights about underlying inductive biases such as these: (b) the modular decomposition of knowledge into independent reusable pieces that can be composed on the fly to address new contexts, (c) the causal interpretation of actions by agents and of changes in distribution, with agents generally intending to affect a single or very few (generally latent) variables, and (d) the sparsity of dependencies between high-level variables (and thus the small number of variables that are linked by causal mechanisms imagined by humans to explain their environment). Finally, we would also like to mention that inductive biases are not the only way to bridge the gap to high-level human cognition: we may gain by improving our optimization algorithms, by scaling up neural networks (Sutton, 2019) and by moving to other frameworks that better capture uncertainty about the world (e.g., by learning a Bayesian posterior over neural network models as compared to learning a point estimates). It would also be intriguing to think of ways to combine all these different elements together.

Chapter 4

Prologue to the first article

4.1. Article Details

Neural Production Systems (NPS). Anirudh Goyal, Aniket Didolkar, Nan Rosemary Ke, Charles Blundell, Philieppe Beaudoin, Nicolas Heess, Michael Mozer, Yoshua Bengio. The paper was accepted for presentation at NeurIPS'21.
Personal Contribution. This work builds upon RIMs (Goyal et al., 2019c) which proposes the idea of decomposing information into an ensemble of modules where each module is responsible for modelling the state of an entity, as well as dynamics of the entity. Yoshua Bengio pointed out the limitations of RIMs and provided the direction as to how to address those limitations i.e., ideally we want to factorize knowledge into set of reusable pieces which can be dynamically instantiated to change the state of modules. I had the idea of structuring the architecture as a set of modules, and reusable rules such that rules can be used to encapsulate interactions between different modules and hence change the state of the slots in a dynamic and context dynamic fashion. I implemented the model on experiments for learning a world model. Aniket Didolkar ran the toy experiments. Aniket and myself both ran the bouncing ball experiments. Nan Rosemary Ke ran physical reasoning experiments. All the authors contributed in writing the paper.

4.2. Context

Visual environments are structured, consisting of distinct objects or entities. These entities have properties – both visible and latent – that determine the manner in which they interact with one another. To partition images into entities, deep learning researchers have proposed structural inductive biases such as slot-based architectures. Learning modular structures which reflect the dynamics of the environment can lead to better generalization and robustness to changes which only affect a few of the underlying causes. In our previous work, we have proposed Recurrent Independent Mechanisms (RIMs) (ICLR'21) (Goyal et al., 2019c), a recurrent architecture in which multiple groups of recurrent cells operate with nearly independent transition dynamics, communicate only sparingly through the bottleneck of attention, and are only updated at time steps where they are most relevant. In RIMs, each module was responsible for modelling the state of an individual entity as well as modelling the dynamics of an entity. To model interactions among entities, equivariant graph neural nets (GNNs) or self-attention is used, but these are not particularly well suited to the task for two reasons. First, GNNs do not predispose interactions to be sparse, as relationships

among independent entities are likely to be. Second, GNNs do not factorize knowledge about interactions in an entity-conditional manner. In this work, we proposed an alternative such that interactions between different entities are sparse and modelled in a dynamic and context dependent fashion.

4.3. Contributions

We take inspiration from cognitive science and resurrect a classic approach, production systems, which consist of a set of rule templates that are applied by binding placeholder variables in the rules to specific entities. Rules are scored on their match to entities, and the best fitting rules are applied to update entity properties. In a series of experiments, we demonstrate that this architecture achieves a flexible, dynamic flow of control and serves to factorize entity-specific and rule-based information. This disentangling of knowledge achieves robust future-state prediction in rich visual environments, outperforming state-of-the-art methods using GNNs, and allows for the extrapolation from simple (few object) environments to more complex environments. We are not the first to propose a neural instantiation of a production system architecture. Touretzky & Hinton (1988) gave a proof of principle that neural net hardware could be hardwired to implement a production system for symbolic reasoning; our work fundamentally differs from theirs in that (1) we focus on perceptual inference problems and (2) we use the architecture as an inductive bias for learning.

4.4. Research Impact

In learning to model a visual environment, many solutions could be found which fit the data well but will not generalize in a systematic way to different settings. To achieve systematic generalization, the knowledge about individual entities (in the form of slots) and sparse rules should be factored from each other. This will allow any rule to be applied to multiple slots or many rules can be applied to the same slot when appropriate. Although the above factorization may seem naturally desirable, it is not built into the neural network architectures: they lump together (for each artificial neuron) both the specification of a rule (via the synaptic weights of the neuron) and the values of the attributes of entities (via the neural activity). We conjecture – and experiments with NPS show – that attention mechanisms offer even more flexibility in factorizing sparse rules and slots and combine them in arbitrary ways demanded by *context and the attributes of entities*. The idea of factorizing knowledge in terms of entities, as well as rules has been useful for influencing the thoughts of my colleagues as well as the general research community. Even though the idea is relatively recent, there have been few follow-ups already using it for model-based RL (Ke et al., 2021) and in the context of continual learning where one can add more rules in a completely decentralized fashion. This is the only work which I am aware of where the interactions among the different entities (as well as the representation of these entities) are sparse and dynamic and learned in a context dependent fashion.

Chapter 5

Neural Production Systems

5.1. Introduction

Despite never having taken a physics course, every child beyond a young age appreciates that pushing a plate off the dining table will cause the plate to break. The laws of physics accurately characterize the dynamics of our natural world, and although explicit knowledge of these laws is not necessary to reason, we can reason explicitly about objects interacting through these laws. Humans can verbalize knowledge in propositional expressions such as "If a plate drops from table height, it will break," and "If a video-game opponent approaches from behind and they are carrying a weapon, they are likely to attack you." Expressing propositional knowledge is not a strength of current deep learning methods for several reasons. First, propositions are discrete and independent from one another. Second, propositions must be quantified in the manner of first-order logic; for example, the video-game proposition applies to any X for which X is an opponent and has a weapon. Incorporating the ability to express and reason about propositions should improve generalization in deep learning methods because this knowledge is modular— propositions can be formulated independently of each other— and can therefore be acquired incrementally. Propositions can also be composed with each other and applied consistently to all entities that match, yielding a powerful form of *systematic generalization*.

The classical AI literature from the 1980s can offer deep learning researchers a valuable perspective. In this era, reasoning, planning, and prediction were handled by architectures that performed propositional inference on symbolic knowledge representations. A simple example of such an architecture is the *production system* (Laird et al., 1986; Anderson, 1987), which expresses knowledge by *condition-action rules*. The rules operate on a *working memory*: rule conditions are matched to entities in working memory inspired by cognitive science, and such a match can trigger computational actions that update working memory or external actions that operate on the outside world.

Production systems were typically used to model high-level cognition, e.g., mathematical problem solving or procedure following; perception was not the focus of these models. It was assumed that the results of perception were placed into working memory in a symbolic form that could be operated on with the rules. In this article, we revisit production systems but from a deep learning perspective which naturally integrates perceptual processing and subsequent inference for visual reasoning problems. We describe an end-to-end deep learning model that constructs object-centric representations of entities in videos, and then operates on these entities with differentiable—and thus learnable—production rules. The essence of

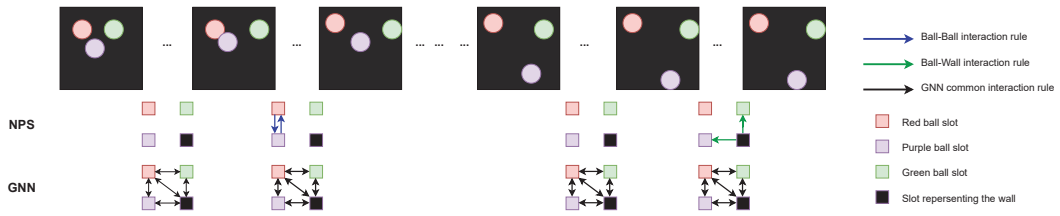


Fig. 1. In this figure we show a visual comparison between NPS and dense architectures like GNNs. In NPS, a rule is only applied when an interaction takes place and it is applied only to the slots affected by the interaction. NPS also uses different rules for different kinds of interactions, while in GNN a common rule is applied to all slots irrespective of whether an interaction takes place or not (because of parameter sharing). Note the dynamic nature of the interaction graph in NPS, while in GNN, the graph is static.

these rules, carried over from traditional symbolic system, is that they operate on variables that are *bound*, or linked, to the entities in the world. In the deep learning implementation, each production rule is represented by a distinct MLP with query-key attention mechanisms to specify the rule-entity binding and to determine when the rule should be triggered for a given entity.

5.1.1. Variables and entities

What makes a rule general-purpose is that it incorporates placeholder *variables* that can be bound to arbitrary *values* or—the term we prefer in this article—*entities*. This notion of binding is familiar in functional programming languages, where these variables are called arguments. Analogously, the use of variables in the production rules we describe enable a model to reason about any set of entities that satisfy the selection criteria of the rule.

Consider a simple function in C like `int add(int a, int b)`. This function binds its two integer operands to variables a and b. The function does not apply if the operands are, say, character strings. The use of variables enables a programmer to reuse the same function to add any two integer values

In order for rules to operate on entities, these entities must be represented explicitly. That is, the visual world needs to be parsed in a task-relevant manner, e.g., distinguishing the sprites in a video game or the vehicles and pedestrians approaching an autonomous vehicle. Only in the past few years have deep learning vision researchers developed methods for object-centric representation (Le Roux et al., 2011; Eslami et al., 2016; Greff et al., 2016; Raposo et al., 2017; Van Steenkiste et al., 2018; Kosiorek et al., 2018; Engelcke et al., 2019; Burgess et al., 2019; Greff et al., 2019; Locatello et al., 2020; Ahmed et al., 2020; Goyal et al., 2019c; Zablotskaia et al., 2020; Rahaman et al., 2020; Du et al., 2020; Ding et al., 2020; Goyal et al., 2020; Ke et al., 2021). These methods differ in details but share the notion of a fixed number of *slots* (see Figure 1 for example), also known as *object files*, each encapsulating information about a single object. Importantly, the slots are interchangeable, meaning that it doesn't matter if a scene with an apple and an orange encodes the apple in slot 1 and orange in slot 2 or vice-versa.

A model of visual reasoning must not only be able to represent entities but must also express knowledge about entity dynamics and interactions. To ensure *systematic* predictions, a model must be capable of applying knowledge to an entity regardless of the slot it is in and must be capable of applying the same knowledge to multiple instances of an entity. Several distinct approaches exist in the literature. The predominant approach uses graph neural networks to model slot-to-slot interactions (Scarselli et al., 2008; Bronstein et al., 2017;

Watters et al., 2017; Van Steenkiste et al., 2018; Kipf et al., 2018; Battaglia et al., 2018; Tacchetti et al., 2018). To ensure systematicity, the GNN must share parameters among the edges. In a recent article, Goyal et al. (2020) developed a more general framework in which parameters are shared but slots can dynamically select which parameters to use in a state-dependent manner. Each set of parameters is referred to as a *schema*, and slots use a query-key attention mechanism to select which schema to apply at each time step. Multiple slots can select the same schema. In both GNNs and SCOFF, modeling dynamics involves each slot interacting with each other slot. In the work we describe in this article, we replace the direct slot-to-slot interactions with rules, which mediate sparse interactions among slots (See arrows in Figure 1).

Thus our main contribution is that we introduce NPS, which offers a way to model dynamic and sparse interactions among the variables in a graph and also allows dynamic sharing of multiple sets of parameters among these interactions. Most architectures used for modelling interactions in the current literature use statically instantiated graph which model all possible interactions for a given variable at each step i.e. dense interactions. Also such dense architectures share a single set of parameters across all interactions which maybe quite restrictive in terms of representational capacity. A visual comparison between these two kinds of architectures is shown in Figure 1. Through our experiments we show the advantage of modeling interactions in the proposed manner using NPS in visually rich physical environments. We also show that our method results in an intuitive factorization of rules and entities.

5.2. Production System

Formally, our notion of a production system consists of a set of entities and a set of rules, along with a mechanism for selecting rules to apply on subsets of the entities. Implicit in a rule is a specification of the properties of relevant entities, e.g., a rule might apply to one type of sprite in a video game but not another. The control flow of a production system dynamically selects rules as well as bindings between rules and entities, allowing different rules to be chosen and different entities to be manipulated at each point in time.

The neural production system we describe shares essential properties with traditional production system, particularly with regard to the compositionality and generality of the knowledge they embody. Lovett & Anderson (2005) describe four desirable properties commonly attributed to symbolic systems that apply to our work as well.

Production rules are modular. Each production rule represents a unit of knowledge and are *atomic* such that any production rule can be intervened (added, modified or deleted) independently of other production rules in the system.

Production rules are abstract. Production rules allow for generalization because their conditions may be represented as high-level abstract knowledge that match to a wide range of patterns. These conditions specify the attributes of relationship(s) between entities without specifying the entities themselves. The ability to represent abstract knowledge allows for the transfer of learning across different environments as long as they fit within the conditions of the given production rule.

Production rules are sparse. In order that production rules have broad applicability, they involve only a subset of entities. This assumption imposes a strong prior that dependencies among entities are sparse. In the context of visual reasoning, we conjecture that this prior is superior to what has often been assumed in the past, particularly in the disentanglement literature—independence among entities Higgins et al. (2016); Chen et al. (2018).

Production rules represent causal knowledge and are thus asymmetric. Each rule can be decomposed into a {condition, action} pair, where the action reflects a state change that is a causal consequence of the conditions being met.

These four properties are sufficient conditions for knowledge to be expressed in production rule form. These properties specify *how* knowledge is represented, but not *what* knowledge is represented. The latter is inferred by learning mechanisms under the inductive bias provided by the form of production rules.

5.3. Neural Production System: Slots and Sparse Rules

The Neural Production System (NPS), illustrated in Figure 2, provides an architectural backbone that supports the detection and inference of entity (object) representations in an input sequence, and the underlying rules which govern the interactions between these entities in time and space. The input sequence indexed by time step t, $\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^t, \ldots, \boldsymbol{x}^T\}$, for instance the frames in a video, are processed by a neural encoder (Greff et al., 2019; Goyal et al., 2019c, 2020) applied to each $\boldsymbol{x}^t$, to obtain a set of M entity representations $\{\boldsymbol{V}_1^t, \ldots, \ldots, \boldsymbol{V}_M^t\}$, one for each of the M slots. These representations describe an entity and are updated based on both the previous state, $\boldsymbol{V}^{t-1}$ and the current input, $\boldsymbol{x}^t$.

NPS consists of N separately encoded rules, $\{\boldsymbol{R}_1, \boldsymbol{R}_2, .., \boldsymbol{R}_N\}$. Each rule consists of two components, $\boldsymbol{R}_i = (\vec{\boldsymbol{R}}_i, MLP_i)$, where $\vec{\boldsymbol{R}}_i$ is a learned rule embedding vector, which can be thought of as a template defining the condition for when a rule applies; and MLP_i, which determines the action taken by a rule. Both $\vec{\boldsymbol{R}}_i$ and the parameters of MLP_i are learned along with the other parameters of the model using back-propagation on an objective optimized end-to-end.

In the general form of the model, each slot selects a rule that will be applied to it to change its state. This can potentially be performed several times, with possibly different rules applied at each step. Rule selection is done using an attention mechanism described in detail below. Each rule specifies conditions and actions on a pair of slots. Therefore, while modifying the state of a slot using a rule, it can take the state of another slot into account. The slot which is being modified is called the primary slot and other is called the contextual slot. The contextual slot is also selected using an attention mechanism described in detail below.

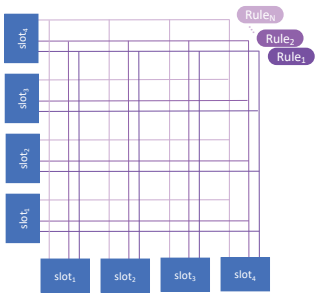


Fig. 2. Rule and slot combinatorics. Condition-action rules specify how entities interact. Slots maintain the time-varying state of an entity. Every rule is matched to every pair of slots. Through key-value attention, a goodness of match is determined, and a rule is selected along with its binding to slots.

5.3.1. Computational Steps in NPS

In this section, we give a detailed description of the rule selection and application procedure for the slots. First, we will formalize the definitions of a few terms that we will use to explain our method. We use the term **primary slot** to refer to slot $\boldsymbol{V}_p$ whose state gets modified by a rule $\boldsymbol{R}_r$. We use the term **contextual slot** to refer to the slot $\boldsymbol{V}_c$ that the rule $\boldsymbol{R}_r$ takes into account while modifying the state of the primary slot $\boldsymbol{V}_p$.

Algorithm 1 Sequential Neural Production System model

Input: Input sequence $\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^t, \ldots, \boldsymbol{x}^T\}$, set of embeddings describing the rules $\vec{\boldsymbol{R}}_i$, and set of MLPs (MLP_i) corresponding to each rule $\boldsymbol{R}_{1\ldots N}$. Hyper-parameters specific to are the number of stages K, the number of slots M, and the number of rules N. W^k, W^q, $\widetilde{W}^k$, and $\widetilde{W}^q$ are learnable weights.

for *each input element* $\boldsymbol{x}^t$ *with* $t \leftarrow 1$ *to* T **do**

Step 1: *Update or infer the entity state in each slot* j, $\boldsymbol{V}_j^{t,0}$, *from the previous state,* $\boldsymbol{V}_j^{t-1,K}$ *and the current input* $\boldsymbol{x}_t$.

for *each stage* $h \leftarrow 0$ *to* $K-1$ **do**

Step 2: *Select {rule, primary slot} pair*

- $\boldsymbol{k}_i = \vec{\boldsymbol{R}}_i \boldsymbol{W}^k \quad \forall i \in \{1, \ldots, N\}$
- $\boldsymbol{q}_j = \boldsymbol{V}_j^{t,h} \boldsymbol{W}^q \quad \forall j \in \{1, \ldots, M\}$
- $r, p = \text{argmax}_{i,j} \left(\boldsymbol{q}_j \boldsymbol{k}_i + \gamma\right)$
  where $\gamma \sim \text{Gumbel}(0, 1)$

Step 3: *Select contextual slot*

- $\boldsymbol{q}_{r,p} = V_p^{t,h} \widetilde{\boldsymbol{W}}^q$
- $\boldsymbol{k}_j = \boldsymbol{V}_j^{t,h} \widetilde{\boldsymbol{W}}^k \quad \forall j \in \{1, \ldots, M\}$
- $c = \text{argmax}_j \left(\boldsymbol{q}_{r,p} \boldsymbol{k}_j + \gamma\right)$
  where $\gamma \sim \text{Gumbel}(0, 1)$

Step 4: *Apply selected rule to primary slot conditioned on contextual slot*

- $\widetilde{\boldsymbol{R}} = \text{MLP}_r(\text{Concatenate}([\boldsymbol{V}_p^{t,h}, \boldsymbol{V}_c^{t,h}]))$
- $\boldsymbol{V}_p^{t,h+1} = \boldsymbol{V}_p^{t,h} + \widetilde{\boldsymbol{R}}$

end

end

Notation. We consider a set of $\boldsymbol{N}$ rules $\{\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_N\}$ and a set of $\boldsymbol{T}$ input frames $\{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^T\}$. Each frame $\boldsymbol{x}^t$ is encoded into a set of $\boldsymbol{M}$ slots $\{\boldsymbol{V}_1^t, \boldsymbol{V}_2^t, \ldots, \boldsymbol{V}_M^t\}$. In the following discussion, we omit the index over $\boldsymbol{t}$ for simplicity.

Step 1. is external to NPS and involves parsing an input image, $\boldsymbol{x}^t$, into slot-based entities conditioned on the previous state of the slot-based entities. Any of the methods proposed in the literature to obtain a slot-wise representation of entities can be used (Greff et al., 2019; Goyal et al., 2019c, 2020). The next three steps constitute the rule selection and application procedure.

Step 2. For each primary slot $\boldsymbol{V}_p$, we attend to a rule $\boldsymbol{R}_r$ to be applied. Here, the queries come from the primary slot: $\boldsymbol{q}_p = \boldsymbol{V}_p W^q$, and the keys come from the rules: $\boldsymbol{k}_i = \vec{R}_i W^k \quad \forall \boldsymbol{i} \in \{1, \ldots, \boldsymbol{N}\}$. The rule is selected using a straight-through Gumbel softmax (Jang et al., 2016) to achieve a learnable hard decision: $\boldsymbol{r} = \text{argmax}_i(\boldsymbol{q}_p \boldsymbol{k}_i + \gamma)$, where $\gamma \sim \text{Gumbel}(0, 1)$. This competition is a noisy version of rule matching and prioritization in traditional production systems.

Step 3. For a given primary slot $\boldsymbol{V}_p$ and selected rule $\boldsymbol{R}_r$, a contextual slot $\boldsymbol{V}_c$ is selected using another attention mechanism. In this case the query comes from the primary slot: $\boldsymbol{q}_p = \boldsymbol{V}_p W^q$, and the keys from all the slots: $\boldsymbol{k}_j = \boldsymbol{V}_j W^q \quad \forall \boldsymbol{j} \in \{1, \ldots, \boldsymbol{M}\}$. The selection takes place using a straight-through Gumbel softmax similar to step 2: $\boldsymbol{c} = \text{argmax}_j(\boldsymbol{q}_p \boldsymbol{k}_j + \gamma)$, where $\gamma \sim \text{Gumbel}(0, 1)$. Note that each rule application is sparse since

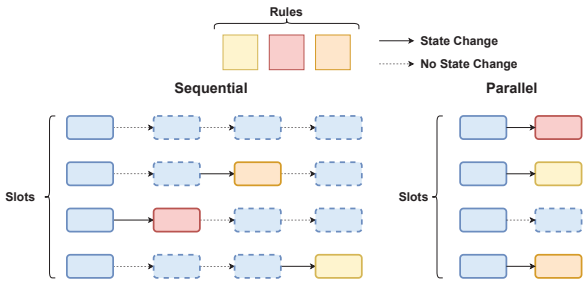


Fig. 3. This figure demonstrates the sequential and parallel rule application.

it takes into account only 1 contextual slot for modifying a primary slot, while other methods like GNNs take into account all slots for modifying a primary slot.

Step 4. Rule Application: the selected rule $\boldsymbol{R}_r$ is applied to the primary slot $\boldsymbol{V}_p$ based on the rule and the current contents of the primary and contextual slots. The rule-specific MLP_r, takes as input the concatenated representation of the state of the primary and contextual slots, $\boldsymbol{V}_p$ and $\boldsymbol{V}_c$, and produces an output, which is then used to change the state of the primary slot $\boldsymbol{V}_p$ by residual addition.

5.3.2. Rule Application: Sequential vs Parallel Rule Application

In the previous section, we have described how each rule application only considers another contextual slot for the given primary slot i.e., **contextual sparsity**. We can also consider **application sparsity**, wherein we use the rules to update the states of only a subset of the slots. In this scenario, only the selected slots would be *primary slots*. This setting will be helpful when there is an entity in an environment that is stationary, or it is following its own default dynamics unaffected by other entities. Therefore, it does not need to consider other entities to update its state. We explore two scenarios for enabling application sparsity.

Parallel Rule Application. Each of the M slots selects a rule to potentially change its state. To enable sparse changes, we provide an extra **Null Rule** in addition to the available N rules. If a slot picks the null rule in step 2 of the above procedure, we do not update its state.

Sequential Rule Application. In this setting, only one slot gets updated in each rule application step. Therefore, only one slot is selected as the primary slot. This can be facilitated by modifying step 2 above to select one {primary slot, rule} pair among $\boldsymbol{NM}$ {rule, slot} pairs. The queries come from each slot: $\boldsymbol{q}_j = \boldsymbol{V}_j W^q \quad \forall \boldsymbol{j} \in \{1, \ldots, \boldsymbol{M}\}$, the keys come from the rules: $\boldsymbol{k}_i = R_i W^k \quad \forall \boldsymbol{i} \in \{1, \ldots, \boldsymbol{N}\}$. The straight-through Gumbel softmax selects one (primary slot, rule) pair: $\boldsymbol{p}, \boldsymbol{r} = \text{argmax}_{i,j}(\boldsymbol{q}_p \boldsymbol{k}_i + \gamma)$, where $\gamma \sim \text{Gumbel}(0, 1)$. In the sequential regime, we allow the rule application procedure (step 2, 3, 4 above) to be performed multiple times iteratively in K rule application stages for each time-step t.

A pictorial demonstration of both rule application regimes can be found in Figure 3. We provide detailed algorithms for the sequential and parallel regimes in Appendix.

5.4. Experiments

We demonstrate the effectiveness of NPS on multiple tasks and compare to a comprehensive set of baselines. To show that NPS can learn intuitive rules from the data generating distribution, we design a couple of simple toy experiments with well-defined discrete operations. Results show that NPS can accurately recover each operation defined by the data

and learn to represent each operation using a separate rule. We then move to a much more complicated and visually rich setting with abstract physical rules and show that factorization of knowledge into rules as offered by NPS does scale up to such settings. We study and compare the parallel and sequential rule application procedures and try to understand the settings which favour each. We then evaluate the benefits of reusable, dynamic and sparse interactions as offered by NPS in a wide variety of physical environments by comparing it against various baselines. We conduct ablation studies to assess the contribution of different components of NPS. Here we briefly outline the tasks considered and direct the reader to the Appendix for full details on each task and details on hyperparameter settings.

Discussion of baselines. NPS is an interaction network, therefore we use other widely used interaction networks such as multihead attention and graph neural networks (Goyal et al. (2019c), Goyal et al. (2020), Veerapaneni et al. (2019), Kipf et al. (2019)) for comparison. Goyal et al. (2019c) and Goyal et al. (2020) use an attention based interaction network to capture interactions between the slots, while Veerapaneni et al. (2019) and Kipf et al. (2019) use a GNN based interaction network. We also consider the recently introduced convolutional interaction network (CIN) (Qi et al., 2021) which captures dense pairwise interactions like GNN but uses a convolutional network instead of MLPs to better utilize spatial information. The proposed method, similar to other interaction networks, is agnostic to the encoder backbone used to encode the input image into slots, therefore we compare NPS to other interaction networks across a wide-variety of encoder backbones.

Table 1. This table shows the segregation of rules for the MNIST Transformation task. Each cell indicates the number of times the corresponding rule is used for the given operation. We can see that NPS automatically and perfectly learns a separate rule for each operation.

	Rule 1	Rule 2	Rule 3	Rule 4
Translate Down	5039	0	0	0
Translate Up	0	4950	0	0
Rotate Right	0	0	5030	0
Rotate Left	0	0	0	4981

5.4.1. Learning intuitive rules with NPS: Toy Simulations

We designed a couple of simple tasks with well-defined discrete rules to show that NPS can learn intuitive and interpretable rules. We also show the efficiency and effectiveness of the selection procedure (step 2 and step 3 in section 5.3.1) by comparing against a baseline with many more parameters. Both tasks require a single modification of only one of the available entities, therefore the use of sequential or parallel rule application would not make a difference here since parallel rule application in which all-but-one slots select the null rule is similar to sequential rule application with 1 rule application step. To simplify the presentation, we describe the setup for both tasks using the sequential rule application procedure.

MNIST Transformation. We test whether NPS can learn simple rules for performing transformations on MNIST digits. We generate data with four transformations: {Translate Up, Translate Down, Rotate Right, Rotate Left}. We feed the input image ($\boldsymbol{X}$) and the transformation ($\boldsymbol{o}$) to be performed as a one-hot vector to the model. The detailed setup is described in Appendix. For this task, we evaluate whether NPS can learn to use a unique rule for each transformation.

We use 4 rules corresponding to the 4 transformations with the hope that the correct transformations are recovered. Indeed, we observe that **NPS successfully learns to represent each transformation using a separate rule** as shown in Table 1. Our model achieves an MSE of 0.02.

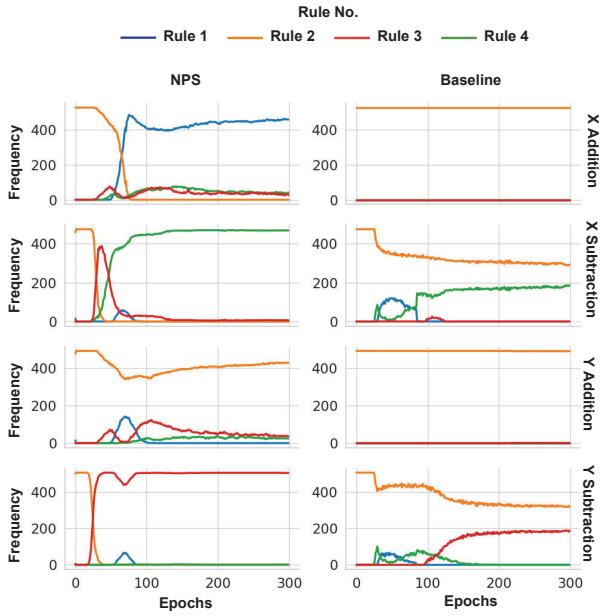


Fig. 4. Coordinate Arithmetic Task. Here, we compare NPS to the baseline model in terms of segregation of rules as the training progresses. X-axis shows the epochs and Y-axis shows the frequency with which Rule i is used for the given operation. We can see that NPS disentangles the operations perfectly as training progresses with a unique rule specializing to every operation while the baseline model fails to do so.

Coordinate Arithmetic Task. The model is tasked with performing arithmetic operations on 2D coordinates. Given (X_0, Y_0) and (X_1, Y_1), we can apply the following operations: {**X Addition**: $(X_r, Y_r) = (X_0 + X_1, Y_0)$, **X Subtraction**: $(X_r, Y_r) = (X_0 - X_1, Y_0)$, **Y Addition**: $(X_r, Y_r) = (X_0, Y_0 + Y_1)$, **Y Subtraction**: $X_r, Y_r = (X_0, Y_0 - Y_1)$}, where (X_r, Y_r) is the resultant coordinate.

In this task, the model is given 2 input coordinates $X = [(x_i, y_i), (x_j, y_j)]$ and the expected output coordinates $Y = [(\hat{x_i}, \hat{y_i}), (\hat{x_j}, \hat{y_j})]$. The model is supposed to infer the correct rule to produce the correct output coordinates. During data collection, the true output is obtained by performing a random transformation on a randomly selected coordinate in X (primary coordinate), taking another randomly selected coordinate from X (contextual coordinate) into account. We use an NPS model with 4 rules for this task. We use the the selection procedure in step 2 and step 3 of algorithm 1 to select the primary coordinate, contextual coordinate, and the rule. For the baseline we replace the selection procedure in NPS (i.e. step 2 and step 3 in algorithm 1) with a routing MLP similar to Fedus et al. (2021b).

Table 2. This table shows segregation of rules when we use NPS with 5 rules but the data generation distributions describes only 4 possible operations. We can see that only 4 rules get majorly utilized thus confirming that NPS successfully recovers all possible operations described by the data.

	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5
X Addition	360	99	45	13	0
X Subtraction	0	482	0	1	0
Y Subtraction	0	39	453	2	0
Y Addition	0	57	15	99	335

This routing MLP has 3 heads (one each for selecting the primary coordinate, contextual coordinate, and the rule). The baseline has 4 times more parameters than NPS. The final output is produced by the rule MLP which does not have access to the true output, hence the model cannot simply copy the true output to produce the actual output. Unlike the

MNIST transformation task, we do not provide the operation to be performed as a one-hot vector input to the model, therefore it needs to infer the available operations from the data demonstrations.

We show the segregation of rules for NPS and the baseline in Figure 4. **We can see that NPS learns to use a unique rule for each operation while the baseline struggles to disentangle the underlying operations properly**. **NPS also outperforms the baseline in terms of MSE** achieving an MSE of $0.01_{\pm 0.001}$ while the baseline achieves an MSE of $0.04_{\pm 0.008}$. To further confirm that NPS learns all the available operations correctly from raw data demonstrations, we use an NPS model with 5 rules. **We expect that in this case NPS should utilize only 4 rules since the data describes only 4 unique operations and indeed we observe that NPS ends up mostly utilizing 4 of the available 5 rules** as shown in Table 2.

5.4.2. Parallel vs Sequential Rule Application

We compare the parallel and sequential rule application procedures, to understand the settings that favour one or the other, over two tasks: (1) Bouncing Balls, (2) Shapes Stack. We use the term PNPS to refer to parallel rule application and SNPS to refer to sequential rule application.

Shapes Stack. We use the shapes stack dataset introduced by Groth et al. (2018). This dataset consists of objects stacked on top of each other as shown in Figure 5. These objects fall under the influence of gravity. For our experiments, We follow the same setup as Qi et al. (2021). In this task, given the first frame, the model is tasked with predicting the object bounding boxes for the next $\boldsymbol{t}$ timesteps. The first frame is encoded using a convolutional network followed by RoIPooling (Girshick (2015)) to extract object-centric visual features. The object-centric features are then passed to the dynamics model to predict object bounding boxes of the next $\boldsymbol{t}$ steps. Qi et al. (2021) propose a Region Proposal Interaction Network (RPIN) to solve this task. The dynamics model in RPIN consists of an Interaction Network proposed in Battaglia et al. (2016b). To better utilize spatial information, Qi et al. (2021) propose an extension of the interaction operators in interaction net to operate on 3D tensors. This is achieved by replacing the MLP operations in the original interaction networks with convolutions. They call this new network Convolutional Interaction Network (CIN). For the proposed model, we replace this CIN in RPIN by NPS. To ensure a fair comparison to CIN, we use CNNs to represent rules in NPS instead of MLPs. CIN captures all pairwise interactions between objects using a convolutional network. In NPS, we capture sparse interactions (contextual sparsity) as compared to dense pairwise interactions captured by CIN. Also, in NPS we update only a few subset of slots per step instead of all slots (application sparsity).

Model Name	Test	Transfer
RPIN (Qi et al. (2021))	$1.254_{\pm 0.008}$	$6.377_{\pm 0.325}$
PNPS	$1.250_{\pm 0.007}$	$5.411_{\pm 0.45}$
SNPS	$1.68_{\pm 0.02}$	$5.80_{\pm 0.15}$

Table 3. Prediction error of the compared models on the shapes stack dataset (lower is better) for the test as well as transfer setting. In the test setting the number of rollout steps $\boldsymbol{t}$ is set to 15 and in the transfer setting it is set to 30. We can see that PNPS outperforms the RPIN baseline in both the test and transfer setting while SNPS fails to do so. Results across 15 seeds.

We consider two evaluation settings. (1) **Test setting**: The number of rollout timesteps is same as that seen during training (i.e. $\boldsymbol{t} = 15$); (2) **Transfer Setting**: The number of rollout timesteps is higher than that seen during training (i.e. $\boldsymbol{t} = 30$).

We present our results on the shapes stack dataset in Table 3. We can see that both PNPS and SNPS outperform the baseline RPIN in the transfer setting, while only PNPS outperforms the baseline in the test setting and SNPS fails to do so. We can see that PNPS outperforms SNPS. We attribute this to the reduced *application sparsity* with PNPS, i.e., it is more likely that the state of a slot gets updated in PNPS as compared to SNPS. For instance, consider an NPS model with $\boldsymbol{N}$ uniformly chosen rules and $\boldsymbol{M}$ slots. The probability that the state of a slot gets updated in PNPS is $P_{PNPS} = \boldsymbol{N} - 1/\boldsymbol{N}$ (since 1 rule is the null rule), while the same probability for SNPS is $P_{SNPS} = 1/\boldsymbol{M}$ (since only 1 slot gets updated per rule application step).

For this task, we run both PNPS and SNPS for $\boldsymbol{N} = \{1, 2, 4, 6\}$ rules and $\boldsymbol{M} = 3$. For any given $\boldsymbol{N}$, we observe that $P_{PNPS} > P_{SNPS}$. Even when we have multiple rule application steps in SNPS, it might end up selecting the same slot to be updated in more than one of these steps. We report the best performance obtained for PNPS and SNPS across all $\boldsymbol{N}$, which is $\boldsymbol{N} = \{2{+}1 \quad \text{Null Rule}\}$ for PNPS and $\boldsymbol{N} = 4$ for SNPS, in Table 3. Shapes stack is a dataset that would prefer a model with less application sparsity since all the objects are tightly bound to each other (objects are placed on top of each other), therefore all objects spend the majority of their time interacting with the objects directly above or below them. We attribute the higher performance of PNPS compared to RPIN to the higher contextual sparsity of PNPS. Each example in the shapes stack task consists of 3 objects. Even though the blocks are tightly bound to each other, each block is only affected by the objects it is in direct contact with. For example, the top-most object is only affected by the object directly below it. The contextual sparsity offered by PNPS is a strong inductive bias to model such sparse interactions while RPIN models all pairwise interactions between the objects. Figure 5 shows an intuitive illustration of the PNPS model for the shapes stack dataset. In the figure, *Rule 2* actually refers to the Null Rule, while *Rule 1* refers to all the other non-null rules. The bottom-most block picks the Null Rule most times, as the bottom-most block generally does not move.

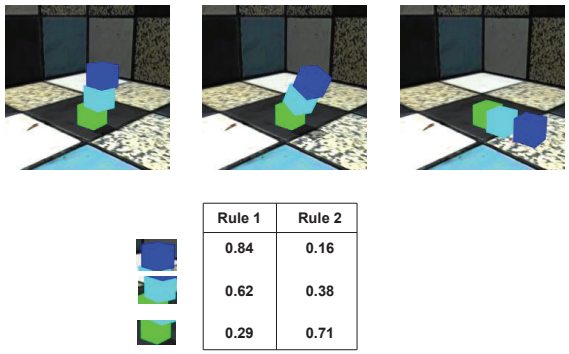


Fig. 5. Here we show the rule selection statistics from the proposed model for all entities in the shapes stack dataset across all examples. Each example contains 3 entities as shown above. Each cell in the table shows the probability with which the given rule is triggered for the corresponding entity. We can see that the bottom-most entity triggers rule 2 most of the time while the other 2 entities trigger rule 1 most often. This is quite intuitive as, for most examples, the bottom-most entity remains static and does not move at all while the upper entities fall. Therefore, rule 2 captures information which is relevant to static entities, while rule 1 captures physical rules relevant to the interactions and motion of the upper entities.

Bouncing Balls. We consider a bouncing-balls environment in which multiple balls move with billiard-ball dynamics. We validate our model on a colored version of this dataset. This is a next-step prediction task in which the model is tasked with predicting the final binary mask of each ball. We compare the following methods: (a) SCOFF (Goyal et al., 2020): factorization of knowledge in terms of slots (object properties) and schemata, the latter capturing object dynamics; (b) SCOFF++: we extend SCOFF by using the idea of iterative competition as proposed in slot attention (SA) (Locatello et al., 2020); SCOFF + PNPS/SNPS: We replace pairwise slot-to-slot interaction in SCOFF++ with parallel or sequential rule application.

For comparing different methods, we use the Adjusted Rand Index or ARI (Rand, 1971). To investigate how the factorization in the form of rules allows for extrapolating knowledge from fewer to more objects, we increase the number of objects from 4 during training to 6-8 during testing.

We present the results of our experiments in Table 4. Contrary to the shapes stack task, we see that SNPS outperforms PNPS for the bouncing balls task. The balls are not tightly bound together into a single tower as in the shapes stack. Most of the time, a single ball follows its own dynamics, only occasionally interacting with another ball. Rules in NPS capture interaction dynamics between entities, hence they would only be required to change the state of an entity when it interacts with another entity. In the case of bouncing balls, this interaction takes place through a collision between multiple balls. Since for a single ball, such collisions are rare, SNPS, which has higher application sparsity (less probability of modifying the state of an entity), performs better as compared to PNPS (lower application sparsity). Also note that, SNPS has the ability to compose multiple rules together by virtue of having multiple rule application stages.

Given the analysis in this section, we can conclude that PNPS is expected to work better when interactions among entities are more frequent while SNPS is expected to work better when interactions are rare and most of the time, each entity follows its own dynamics. Note that, for both SNPS and PNPS, the rule application considers only 1 other entity as context. Therefore, both approaches have equal *contextual sparsity* while the baselines that we consider (SCOFF and RPIN) capture dense pairwise interactions. We discuss the benefits of *contextual sparsity* in more detail in the next section.

5.4.3. Benefits of Sparse Interactions Offered by NPS

In NPS, one can view the computational graph as a dynamically constructed GNN resulting from applying dynamically selected rules, where the states of the slots are represented on the different nodes of the graph, and different rules dynamically instantiate an hyper-edge between a set of slots (the primary slot and the contextual slot). It is important to emphasize that the topology of the graph induced in NPS is dynamic and sparse (only a few nodes affected), while in most GNNs the topology is fixed and dense (all nodes affected). In this section, through a thorough set of experiments, we show that learning sparse and dynamic interactions using NPS indeed works better for the problems we consider than learning dense interactions using GNNs. We consider two types of tasks: (1) Learning Action Conditioned World Models (2) Physical Reasoning. We use SNPS for all these experiments since in the environments that we consider here, interactions among entities are rare.

Model Name	Test	Transfer
SCOFF	0.28	0.15
SCOFF++	0.8437	0.2632
PNPS (10 Rules+1 Null Rule)	0.7813	0.1997
SNPS (10 Rules)	0.8518	0.3553

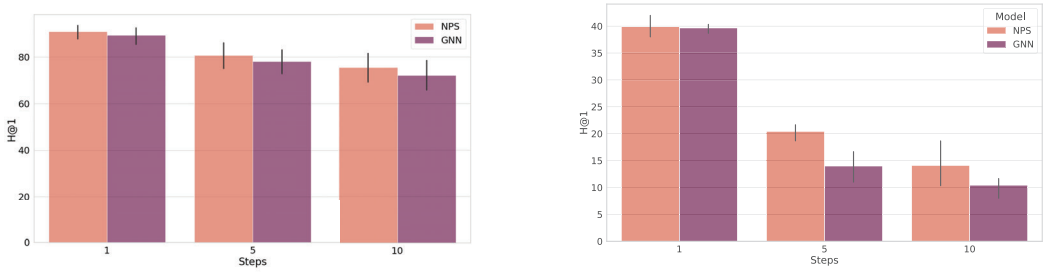
Table 4. Here we show the ARI achieved by the models on the bouncing balls dataset (higher is better). We can see that SNPS outperforms SCOFF and SCOFF++ while PNPS has a poor performance in this task. Results average across 2 seeds.

Learning Action-Conditioned World Models. For learning action-conditioned world models, we follow the same experimental setup as Kipf et al. (2019). Therefore, all the tasks in this section are next-K step ($K = \{1, 5, 10\}$) prediction tasks, given the intermediate actions, and with the predictions being performed in the latent space. We use the Hits at Rank 1 (H@1) metrics described by Kipf et al. (2019) for evaluation. H@1 is 1 for a particular example if

the predicted state representation is nearest to the encoded true observation and 0 otherwise. We report the average of this score over the test set (higher is better).

Physics Environment. The physics environment (Ke et al., 2021) simulates a simple physical world. It consists of blocks of unique but unknown weights. The dynamics for the interaction between blocks is that the movement of heavier blocks pushes lighter blocks on their path. This rule creates an acyclic causal graph between the blocks. For an accurate world model, the learner needs to infer the correct weights through demonstrations. Interactions in this environment are sparse and only involve two blocks at a time, therefore we expect NPS to outperform dense architectures like GNNs. This environment is demonstrated in Appendix Fig **??**.

We follow the same setup as Kipf et al. (2019). We use their C-SWM model as baseline. For the proposed model, we only replace the GNN from C-SWM by NPS. GNNs generally share parameters across edges, but in NPS each rule has separate parameters. For a fair comparison to GNN, we use an NPS model with 1 rule. Note that this setting is still different from GNNs as in GNNs at each step every slot is updated by instantiating edges between all pairs of slots, while in NPS an edge is dynamically instantiated between a single pair of slots and only the state of the selected slot (i.e., primary slot) gets updated.



a **Physics Env** b **Atari Games**

Fig. 6. Action-Conditioned World Models, with number of future steps to be predicted for the world-model on the horizontal axes. (a) Here we show a comparison between GNNs and the proposed NPS on the physics environment using the H@1 metric (higher is better). (b) Comparison of average H@1 scores across 5 Atari games for the proposed model NPS and GNN.

The results of our experiments are presented in Figure 6(a). We can see that NPS outperforms GNNs for all rollouts. Multi-step settings are more difficult to model as errors may get compounded over time steps. The sparsity of NPS (only a single slot affected per step) reduces compounding of errors and enhances symmetry-breaking in the assignment of transformations to rules, while in the case of GNNs, since all entities are affected per step, there is a higher possibility of errors getting compounded. We can see that even with a single rule, we significantly outperform GNNs thus proving the effectiveness of dynamically instantiating edges between entities.

Atari Games. We also test the proposed model in the more complicated setting of Atari. Atari games also have sparse interactions between entities. For instance, in Pong, any interaction involves only 2 entities: (1) paddle and ball or (2) ball and the wall. Therefore, we expect sparse interactions captured by NPS to outperform GNNs here as well.

We follow the same setup as for the physics environment described in the previous section. We present the results for the Atari experiments in Figure 6(b), showing the average H@1 score across 5 games: Pong, Space Invaders, Freeway, Breakout, and QBert. As expected, we

can see that the proposed model achieves a higher score than the GNN-based C-SWM. The results for the Atari experiments reinforce the claim that NPS is especially good at learning sparse interactions.

Learning Rules for Physical Reasoning. To show the effectiveness of the proposed approach for physical reasoning tasks, we evaluate NPS on another dataset: Sprites-MOT (He et al., 2018). The Sprites-MOT dataset was introduced by He et al. (2018). The dataset contains a set of moving objects of various shapes. This dataset aims to test whether a model can handle occlusions correctly. Each frame has consistent bounding boxes which may cause the objects to appear or disappear from the scene. A model which performs well should be able to track the motion of all objects irrespective of whether they are occluded or not. We follow the same setup as Weis et al. (2020). We use the OP3 model (Veerapaneni et al., 2019) as our baseline. To test the proposed model, we replace the GNN-based transition model in OP3 with the proposed NPS.

We use the same evaluation protocol as followed by Weis et al. (2020) which is based on the MOT (Multi-object tracking) challenge (Milan et al., 2016). The results on the MOTA and MOTP metrics for this task are presented in Table 5. We can see that for almost all metrics, NPS outperforms the OP3 baseline. Although this dataset does not contain physical interactions between the objects, sparse rule application should still be useful in dealing with occlusions. At any time step, only a single object is affected by occlusions i.e., it may get occluded due to another object or due to a prespecified bounding box, while the other objects follow their default dynamics. Therefore, a rule should be applied to only the object (or entity) affected (i.e., not visible) due to occlusion and may take into account any other object or entity that is responsible for the occlusion.

Model	MOTA ↑	MOTP ↑
OP3	$89.1_{\pm 5.1}$	$78.4_{\pm 2.4}$
NPS	$90.72_{\pm 5.15}$	$79.91_{\pm 0.3}$

Table 5. Sprites-MOT. Comparison between the proposed NPS and the baseline OP3 using the MOTA and MOTP metrics on the sprites-MOT dataset (↑: higher is better). Average over 3 random seeds.

5.5. Discussion and Conclusion

For AI agents such as robots trying to make sense of their environment, the only observables are low-level variables like pixels in images. To generalize well, an agent must induce high-level entities as well as discover and disentangle the rules that govern how these entities actually interact with each other. Here we have focused on perceptual inference problems and proposed NPS, a neural instantiation of production systems by introducing an important inductive bias in the architecture following the proposals of Marcus (2003); Bengio (2017); Goyal & Bengio (2020); Ke et al. (2021).

Limitations & Looking Forward. Our experiments highlight the advantages brought by the factorization of knowledge into a small set of entities and sparse sequentially applied rules. Immediate future work would investigate how to take advantage of these inductive biases for more complex physical environments (Ahmed et al., 2020) and novel planning methods, which might be more sample efficient than standard ones (Schrittwieser et al., 2020). We also find that Sequential and Parallel NPS have different properties suited towards different domains. Future work should explore how to effectively combine these two approaches.

Chapter 6

Prologue to the second article

6.1. Article Details

Coordination Among Neural Modules Through a Shared Global Workspace. Anirudh Goyal, Aniket Didolkar, Alex Lamb, Kartikeya Badola, Nan Rosemary Ke, Nasim Rahaman, Jonathan Binas, Charles Blundell, Michael Mozer, Yoshua Bengio. This work was accepted for presentation at **International Conference on Learning Representations (ICLR), 2022**.

Personal Contribution. Yoshua Bengio had the original thought of having a bottleneck which can be used for synchronizing information in the human brains, and made the connection to the theories of consciousness (global workspace). I had the idea of how such a bottleneck can be implemented efficiently in commonly used neural networks such as slot based models (RIMs), Transformers and Graph Neural Networks. I implemented the original version and got proof-of-concept results on toy tasks. Aniket Didolkar and Kartikeya Badola ran experiments on bouncing balls and the CIFAR jumbling task. Nasim Rahaman ran the experiments on Starcraft dataset which are not included in the paper. Alex Lamb ran the experiments integrating shared workspace and Transformers with independent mechanisms (TIMs)(Lamb et al., 2021). Nan Rosemary Ke ran the experiments on the physical reasoning benchmark. Jonathan Binas ran MNIST experiments. All the authors contributed in writing of the paper.

6.2. Context

Deep learning has seen a movement away from representing examples with a monolithic hidden state towards a richly structured state. For example, Transformers segment by position, and object-centric architectures decompose images into entities. In all these architectures, interactions between different elements are modeled via pairwise interactions: Transformers make use of self-attention to incorporate information from other positions and object-centric architectures make use of graph neural networks to model interactions among entities. We consider how to improve on pairwise interactions in terms of global coordination and a coherent, integrated representation that can be used for downstream tasks.

6.3. Contributions

In cognitive science, a *global workspace* architecture has been proposed in which functionally specialized components share information through a common, bandwidth-limited

communication channel. We explore the use of such a communication channel in the context of deep learning for modeling the structure of complex environments. The proposed method includes a shared workspace through which communication among different specialist modules takes place but due to limits on the communication bandwidth, specialist modules must compete for access. We show that capacity limitations have a rational basis in that (1) they encourage specialization and compositionality and (2) they facilitate the synchronization of otherwise independent specialists. All communication occurs through key-value attention, which ensures that the specialists are interchangeable, and that any specialist can pass information to the workspace in a form that others can learn to interpret. Experiments on prediction and visual reasoning tasks highlight the advantages brought by the conjunction of modularity and the shared workspace.

6.4. Research Impact

This paper is very recent, and already led to many follow-ups where the authors have scaled the idea to more complex datasets (Jaegle et al., 2021). Even more recently, Lu et al. (2022) used the idea to synchronize information among different modalities such that a single model can perform a large variety of AI tasks spanning classical computer vision tasks, including pose estimation, object detection, depth estimation and image generation, vision-and-language tasks such as region captioning and referring expression comprehension, to natural language processing tasks such as question answering and paraphrasing. They achieve this unification by homogenizing every supported input and output into a bottleneck of discrete vocabulary tokens, and running a transformer on top of the encoded tokens. My work has also gained a lot of attention from researchers thinking about global workspace theories in cognitive neuroscience.

The proposed model can also be seen as integrating out different ideas popular in modular architectures (Andreas et al., 2016; Goyal et al., 2019c), memory networks (Graves et al., 2014b; Santoro et al., 2018) and mixture of experts (Jacobs et al., 1991), and hence combining some of their benefits in a unified architecture. The proposed model is factored as a set of specialists (incorporating modularity). The proposed model achieves coordination among different specialists via the use of a shared workspace (in the Neural Turing machines, there is only a single specialist i.e., without any modularity). Multiple experts can be active at the same time (generally not the case with a mixture of experts).

Chapter 7

Coordination Among Neural Modules Through a Shared Global Workspace

7.1. Introduction

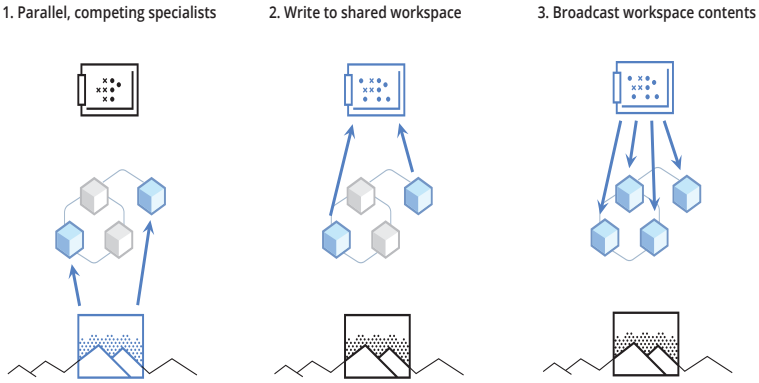


Fig. 1. Step 1: an ensemble of specialist modules doing their own default processing; at a particular computational stage, depending upon the input, a subset of the specialists becomes active. **Step 2:** the active specialists get to write information in a shared global workspace. **Step 3:** the contents of the workspace are broadcast to all specialists.

Deep Learning has seen a movement towards more structured models with cleaner separation between different pieces of information often handled by different components. The induced structure, and separation of knowledge has improved generalization, model-size scaling, and long-range dependencies (Berner et al., 2019; Vinyals et al., 2019; Brown et al., 2020). This opens up questions about how to achieve coherence and coordination between different components in such architectures. Looking back to the 1980s, the focus in AI was much less on learning and more on constructing articulated, multi-component architectures and examining how intelligence might emerge from interactions among this collection of simple, functionally specialized components (Fodor, 1983; Braitenberg, 1986; Minsky, 1988; Brooks, 1991). Each of these specialist modules is on the scale of a typical component of a computer program, like a subroutine that implements a narrow, prespecified function from certain input contents to certain output contents.[1] Through appropriate communication and coordination, a set of specialists can achieve complex, dynamic, and flexible behavior patterns.

[1]In the literature, specialists are sometimes referred to as processes or agents.

As a concrete illustration, consider the task of driving a car in terms of specialists. One specialist might monitor the position of the car with respect to lines on the road, and another specialist might adjust the steering direction based on the perceptual data. In addition, there might be specialists which provide alerts when certain events occur, such as loud sounds, reaching a critical intersection on a route, or coming into close proximity to the car in front. To execute the task of driving the car properly, all these specialists need to interact coherently and broadcast their individual information to each other.

Arguably, modern ML and AI has yet to develop broad architectural frameworks for learning both the specialist modules and how they should *interact*, while the classical view lacks an articulate story about how learning could take place successfully in such frameworks. In this article, we revisit this classical view with modern machine learning tools based on end-to-end learning and differentiable memory and attention mechanisms. Inspired by the Global Workspace Theory (Baars, 1993; Dehaene et al., 1998; Shanahan & Baars, 2005; Shanahan, 2006, 2010, 2012; Dehaene et al., 2017) from cognitive neuroscience, we argue that more flexibility and generalization emerge through an architecture of specialists if their training encourages them to communicate effectively with one another via the bottleneck of a shared workspace (Figure. 1).

Distributed specialist modules. From a computational perspective, articulated multi-component architectures composed of sparsely interacting specialist modules show desirable scaling properties (e.g., more specialists can seamlessly be added), increased robustness (the system can tolerate the removal of or changes in individual specialists), and efficiency (information is processed predominantly locally, reducing the cost of communication between specialists). However, modularization also requires mechanisms to establish sharing of compatible representations across specialists, a form of shared internal language. While portions of a task might be solved by independent specialists, synchronization is critical particularly when there are statistical, functional, or causal dependencies among the specialists.

Coherence through a shared workspace. In cognitive neuroscience, the Global Workspace Theory (GWT) (Baars, 1993; Dehaene et al., 2017) suggests an architecture allowing specialist modules to interact. The key claim of GWT is the existence of a shared representation—sometimes called a blackboard, sometimes a workspace—that can be modified by any specialist and that is broadcast to all specialists, along with the notion that write access is limited to maintain coherence. Our interpretation of this restriction on write access is that it stems from an assumption on the form of the joint distribution between high-level concepts. In this paper, we explore a communication and coordination scheme similar to the one proposed by GWT for modern neural network architectures like Transformers (Vaswani et al., 2017; Dehghani et al., 2018; Parmar et al., 2018; Radford et al., 2019; Brown et al., 2020) and attention-based modular architectures (Goyal et al., 2019c; Rahaman et al., 2020; Mittal et al., 2020a; Goyal et al., 2020; Madan et al., 2021).

In terms of our driving example, the workspace could be used to override default behaviors by giving high priority to specialist modules which provide alerts of various sorts (loud sounds, presence of a child on the street), allowing specialists which respond to such alerts to take control of behavior over default driving routines. This scenario implies that prioritization of signals in a shared workspace is critical.

A shared communication channel necessitates common representations. For a multitude of specialist modules to cooperate, a common language is necessary (Baars, 1997). For example, in the driving scenario, alerts may come from auditory or visual processing specialists, but regardless of the source, a signal for danger must be placed in the workspace

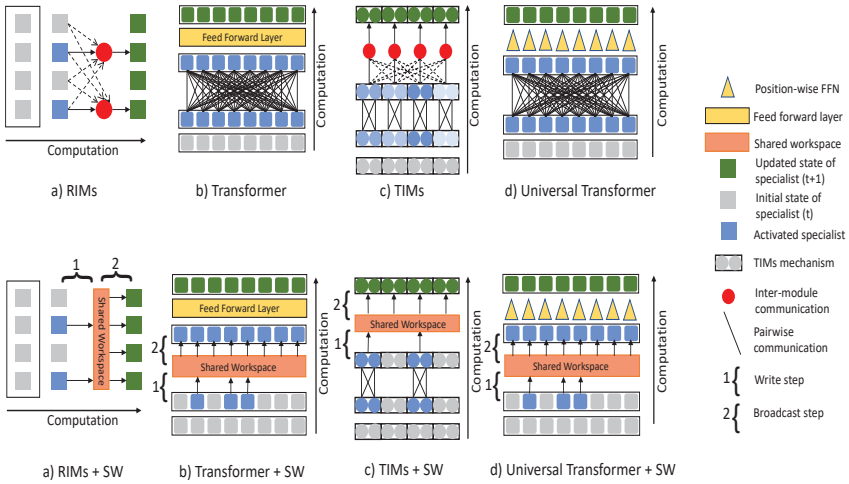


Fig. 2. Using a Shared Workspace for creating global coherence in RIMs, Transformers, TIMs and Universal Transformers (UT). (Top Half) All four of these architectures use pairwise communication (using key-value attention) to establish coherence between individual specialist modules. In the case of RIMs (Goyal et al., 2019c) and TIMs (Lamb et al., 2021), these specialists are independent modules that compete with each other in order to take control over the state update based on a given input. In the case of Transformers (Vaswani et al., 2017) and Universal Transformers (Dehghani et al., 2018), each specialist is associated with a different position. Activated specialists are denoted by a blue shade and the intensity depends on the degree of activation. In the case of Universal Transformers, the state update dynamics for each position is shared across all layers and all positions (denoted by a yellow triangle). (Bottom Half) We replace pairwise communication with a shared workspace to create *global coherence* between different specialists. Communication using the shared workspace is a two-step process (as denoted by 1 and 2 in the figures). In the first step (1), specialists compete for write access to the shared workspace, resulting in a subset of them being activated (in blue), and only the activated specialists perform the write operation on the workspace. In the second step (2), the contents of the shared workspace are broadcast to all the specialists.

to override default behavior, whether that behavior is controlled by a radio-tuning specialist or a steering specialist. Although specialist modules can be pre-wired to have compatible communication interfaces, we will model an architecture in which an ensemble of specialist modules is *trained in coordination*, which should lead to a shared language (Colagrosso & Mozer, 2005). Internally, individual specialists can use whatever form of representations that serves them, but their inputs and outputs require alignment with other specialists in order to synchronize. For example, an unusual event such as a rough thud under the wheels might not have been previously experienced, but the mere signalling of novelty could override default specialists. Without a global communication channel, specialists would have to learn to communicate through pairwise interactions, which might limit coordination of behavior in novel situations: global communication ensures exchangeability of knowledge to achieve systematic generalization.

7.2. Synchronizing neural modules through a shared workspace

We investigate a neural architecture reminiscent of the GW model, where a number of sparsely communicating specialist modules interact via a shared working memory. In particular, we extend the Transformer (Vaswani et al., 2017), attention and slot-based modular

architectures (Goyal et al., 2019c) by adding a shared workspace and allowing modules (each representing an entity) to compete for write access in each computational stage.

Key-value attention. Key-value attention defines the backbone of updates to the hidden states in the proposed model. This form of attention is widely used in self-attention models and performs well on a wide array of tasks (Bahdanau et al., 2014; Vaswani et al., 2017; Santoro et al., 2018). Key-value attention selects an input value based on the match of a query vector to a key vector associated with each value. To allow differentiability and thus easier learnability, selection is soft and computes a convex combination of all the values. Such a mechanism makes it possible to change on-the-fly both the source of input and how the shared workspace is updated. It also makes the outputs of the specialists and the elements of the memory permutation invariant: they should be considered as an *unordered* set of elements to be selected by an attention mechanism from the contents of specialists. More precisely, soft attention uses the product of a *query* (represented as a matrix Q of dimensionality $N_r \times d$, with N_r queries, and d the dimension of each query) with a set of N_o objects each associated with a *key* as a row in matrix K^T ($N_o \times d$). After normalization with a softmax the resulting convex weights are used to combine the *values* V_i (row i of matrix V): where the softmax is applied to each row of its argument matrix, yielding a set of convex weights. For our experiments, we use multihead dot product attention.

Neural modules with pairwise interactions. Our approach to synchronizing neural modules is highly general and mostly agnostic to the task, domain, or specific choice of architecture, with the only requirement being that the model consists of multiple specialist modules which either operate independently or have sparse interactions requiring to only match pairs of modules at a time. Our goal is to explore how introducing a shared workspace can help these modules to become better synchronized and coordinated. We show the utility of the shared workspace for synchronization in (a) Transformers (Vaswani et al., 2017), in which all interactions between positions are performed via attention, and (b) slot-based architectures like Recurrent Independent Mechanisms or RIMs (Goyal et al., 2019c) in which all pairwise interactions between modules are performed via attention. In the context of slot-based architectures, each slot's content is associated with a specialist module, whereas in Transformers different entities each associated with a different position acts as a specialist module (Figure 2).

Both Transformers and RIMs utilize a self-attention mechanism for sharing information between modules, typically implemented in a pairwise manner, i.e., each specialist attends to every other specialist. Instead, we facilitate information sharing among specialist modules through a *limited capacity shared workspace*. In this framework at each computational stage, different specialists compete for write access to the common workspace. The contents of the workspace, in turn, are broadcast to all specialist modules simultaneously.

Notation. The input is processed through a sequence of computational stages indexed by t, and at each stage, n_s entities are operated on (i.e., n_s different modules in slot-based architectures like RIMs or n_s different positions in the case of Transformers). Each of these n_s specialist modules has a distinct internal n_h-dimensional state $\boldsymbol{h}_t^k$, for $k \in \{1, ..., n_s\}$. The specialist modules communicate with each other via a shared workspace divided into n_m memory *slots*, each consisting of a vector of n_l elements, denoted $\boldsymbol{M} = [\boldsymbol{m}_1; \ldots \boldsymbol{m}_j; \ldots \boldsymbol{m}_{n_m}]$. The shared workspace is updated across different computational stages i.e., different time-steps in recurrent architecture and different layers in the case of Transformers. At each computational stage t, different specialists compete for writing in the shared workspace, but all specialists can read from the current state of the workspace. In the case of an autoregressive task, we

can restrict the information sharing to previous positions and keep a separate version of the workspace for each position.

7.2.1. Specifics of the Shared Workspace.

Step 1: Process Input to obtain an entity representation for each specialist. The first step is external to the proposed method, and involves processing the input to form the initial representation vector for each of the different specialists. Different common deep learning architectures can be used to form the representation of different specialists. For example, Transformers start with a matrix $n_s \times n_h$ whose rows are initialized as the n_h-dimensional embeddings of the input at each position of the sequence. Slot-Based Recurrent architectures like RIMs consist of a single-layer recurrent structure where the hidden state $\mathbf{h}_t$ at computational stage t is decomposed into the substates of the n_s specialists, $\mathbf{h}_t^k$ for $k = 1, ...n_s$.

In the proposed scheme, within each computational stage, the updates of the hidden state of different specialists follow a two-step process. First, specialists compete and write to a shared workspace. Second, information from the workspace gets broadcast to all the specialists, as detailed next.

Step 2: Writing Information in the shared workspace. The specialists compete to write into the shared workspace, whose contents need to be updated in the context of new information received from different specialists. This step ensures that only the critically important signals make it to the shared workspace, therefore preventing the workspace from being cluttered. Let matrix $\boldsymbol{R}$ represent the combined state of all the specialists (i.e. $h_t^k \quad \forall k \in \{1, \ldots, n_s\}$ as the rows of $\boldsymbol{R}$). In order to implement the competition between specialists to write into the workspace, we use a key-query-value attention mechanism. In this case, the query is a function of the state of the current workspace memory content, represented by matrix $\boldsymbol{M}$ (with one row per slot of the memory), i.e $\widetilde{\boldsymbol{Q}} = \boldsymbol{M}\widetilde{\boldsymbol{W}}^q$. Keys and values are a function of the information from the specialists i.e., a function of $\boldsymbol{R}$. We apply dot product attention to get the updated memory matrix: $\boldsymbol{M} \leftarrow \text{softmax}\left(\frac{\widetilde{\boldsymbol{Q}}(\boldsymbol{R}\widetilde{\boldsymbol{W}}^e)^{\mathrm{T}}}{\sqrt{d_e}}\right)\boldsymbol{R}\widetilde{\boldsymbol{W}}^v$. The use of a regular softmax to write into $\boldsymbol{M}$ leads to a standard soft competition among different specialists to write in the shared workspace. One can also use a top-k softmax (Ke et al., 2018) to select a fixed number of specialists allowed to write in the shared workspace: based on the pre-softmax values, a fixed number of k specialists which have the highest values are selected, and get access to write in the shared workspace. Selection with a top-k softmax is a hybrid between hard and soft selection. We denote the set of thus selected specialists as $\mathcal{F}_t$. We note that we can apply the attention mechanism multiple times to distill information from different specialists into the shared workspace. Here, the contents of the shared workspace are updated in the gated way as proposed in RMC (Santoro et al., 2018).

Step 3: Broadcast of information from the shared workspace. Each specialist then updates its state using the information broadcast from the shared workspace. We again utilize an attention mechanism to perform this consolidation. All the specialists create queries $\widehat{\boldsymbol{q}}_k = \boldsymbol{h}_t^k\widehat{\boldsymbol{W}}^q$, which are matched with the keys $\widehat{\boldsymbol{\kappa}}_j = (\boldsymbol{m}_j\widehat{\boldsymbol{W}}^e)^{\mathrm{T}} \quad \forall k \in \{1, \ldots, n_s\},\ j \in \{1, \ldots, n_m\}$ from the updated memory slots, forming attention weights $s_{k,j} = \text{softmax}\left(\frac{\widehat{\boldsymbol{q}}_k\widehat{\boldsymbol{\kappa}}_j}{\sqrt{d_e}}\right)$. The memory slot values generated by each slot of the shared workspace and the attention weights are then used to update the state of all the specialists: $\boldsymbol{h}_t^k \leftarrow \boldsymbol{h}_t^k + \sum_j s_{k,j}\widehat{\boldsymbol{v}}_j$ where $\widehat{\boldsymbol{v}}_j = \boldsymbol{m}_j\widehat{\boldsymbol{W}}^v \quad \forall k \in \{1, \ldots, n_s\}$. After receiving the broadcast information from the

workspace, each specialist update their state by applying some dynamics function i.e., one step update of LSTM or GRU units in the case of recurrent architectures, and a feedforward layer in the case of Transformers. This yields the new value $\boldsymbol{h}_{t+1}^k$ for the k-th specialist, from which we start the next stage $(t+1)$.

Replacing pairwise interactions among neural modules with interaction facilitated by the shared workspace allows for the following:

1. Higher-order (HO) interaction among neural modules. The two-step write-read process first allows each memory slot to store a 'filtered summary' of the current input where the 'filter' is determined by the previous state of that slot ('Query' for the write step). Neural modules then summarize the information contained in these slots and update their state. Hence unlike pairwise interaction, messages passed among neural modules in the shared workspace setting also include HO interaction terms; those consisting of more than 2 modules at a time. Naturally, HO interaction require that messages passed among neural modules lie in the same representation space, which is precisely what we aim to achieve by allowing message passing only via a singular global channel.

2. Dynamic filtering due to persistence of memory. With a shared workspace (SW), contents of the memory slot play a key role in filtering and summarizing the information contained in the input at a given time step. Persistence of memory throughout an episode 1) would allow the memory layer to summarize and filter information based on what it has seen thus far 2) should ideally lead to better generalization as the model is able to dynamically modify its filtering machinery for a particular input. In contrast, "inducing points" in Set Transformers (Lee et al., 2019a) are fixed after training and hence the bottleneck cannot adjust itself on the fly for any new input. We present comparisons on several tasks in section 7.4. They show the importance of these properties by comparing performance of SW with 2×Self-Attention (to simulate HO interaction without global communication).

Computational Complexity of using shared workspace for synchronizing different specialists. To encourage a coherent global coordination, Transformers and slot-based recurrent architectures rely on pairwise interactions captured via an attention mechanism. Unfortunately, such attention mechanisms scale quadratically with the number of specialists. Here, we propose a method which uses a shared workspace to create global coherence between different specialists and in the process, replaces the pairwise interactions of conventional dot-product attention. The computational complexity of the proposed method is thus *linear* in the number of specialists. In our experimentation, the number of memory slots is practically constant, which suggests a very favourable scaling behavior, and certainly much less than quadratic. As a point of reference, what would correspond to the number of slots in human working memory (Baars, 1993) is indeed very small (less than 10 slots).

7.3. Related Work

This work taps into a line of reasoning put forward by historical works, such as Minsky (1988); Braitenberg (1986); Fodor (1983), wherein it is argued that in order to be able to deal with a wide spectrum of conditions and tasks, an intelligent system should be comprised of many interacting specialized modules or programs, rather than a single "one-size-fits-all" entity. While modular architectures have been the subject of a number of research directions, (Jacobs et al., 1991; Bottou & Gallinari, 1991; Ronco et al., 1997; Reed & De Freitas, 2015; Andreas et al., 2016; Rosenbaum et al., 2017; Fernando et al., 2017; Shazeer et al., 2017; Rosenbaum et al., 2019a; Goyal & Bengio, 2020), we focus here on a mechanism for achieving

coherence and synchronization between specialist modules via a global workspace shared between all specialists.

Prior works have explored incorporating slot-based memory in the context of recurrent neural networks (Graves et al., 2014b, 2016; Santoro et al., 2018). In the context of transformers, Burtsev & Sapunov (2020) introduce *memory tokens* that are processed in addition to sequence tokens, whereas Dai et al. (2019) (Transformer-XL) propose to partition a long sequence to smaller segments and use the activations of the previous segment in memory while processing the current segment. Building on the latter, Rae et al. (2019) propose to store activations from prior segments in a compressed memory. However, these methods do not restrict memory writes to be sparse and competitive. Recent advances in this direction include the global neuronal workspace (GNW) model (Dehaene & Changeux, 2011), which identifies the global workspace with a large network of excitatory pyramidal neurons with long-range axonal processes connecting prefrontal and parietal cortices. Further, deploying a shared workspace to establish coherence between different specialists as opposed to using all-pair communication has an added benefit, in that it allows us to tackle the $O(n^2)$ complexity of self-attention. This makes our work related to previous work on reducing the computational complexity of dot product attention in Transformers. Lee et al. (2019a) introduce the *ISAB* module, which maps between sets and comprises two dot-product attention layers. In the first layer, a set of trainable parameters are used as queries and the elements of the input set as keys; in the second layer, the output of the first layer is used as keys and the input set as queries. However, unlike in this work, the intermediate states (corresponding to the output of the first layer) are not maintained across layers. Concurrent to our work, (Jaegle et al., 2021) also introduced the idea of using a latent bottleneck for addressing quadratic complexity by learning a bottleneck but there are important differences. For example. in Perceiver the latent bottleneck iteratively queries the information about different positions, and does not maintain the representation of the different specialists. More precisely, in our proposed method different specialists write information in the workspace and then information gets read from the shared workspace. In Perceiver, the latent bottleneck iteratively reads information from the set of positions. We also show the applicability of the proposed idea both for slot based models and Transformers.

7.4. Experiments

Here we briefly outline the tasks on which we applied the idea of the shared workspace and direct the reader to the appendix for some more experiments (Appendix **??**), full details on each task and details on hyperparameter settings for the model. The experiments have the following goals: (a) Demonstrate that the use of the shared workspace can improve results on a wide array of challenging benchmark tasks, with the goal of demonstrating the practical utility and breadth of the technique. (b) Show that the shared workspace addresses coherence between different specialists by achieving improved performance without requiring all pairwise interactions. Finally, to show wide applicability of our model, we integrate SW in TIMs (Lamb et al., 2021), SCOFF (Goyal et al., 2020) and BRIMs (Mittal et al., 2020b) and show improvements over the default communication method used in each.

Making sense of the visual input. Using a shared workspace introduces a bottleneck in sharing of information between specialists. Since the size of the workspace is limited and generally much lower than the number of specialists, there is a limit to the amount of information that can be exchanged among specialists. We hypothesize that mediating communication through

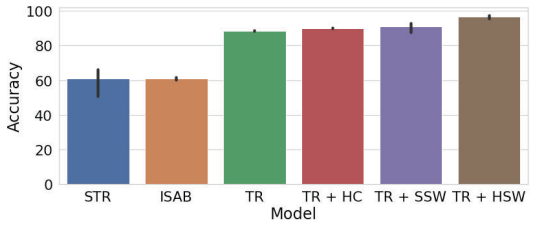


Fig. 3. Detecting Equilateral Triangles. Here, we compare the performance of the Transformers with shared workspace to other Transformer baselines. Here, we plot the test accuracy for each model.

Model	**Top-1** %	**Top-5** %
ISAB	$65.3_{\pm 0.025}$	$83.6_{\pm 0.011}$
STR	$70.6_{\pm 0.08}$	$87.33_{\pm 0.06}$
TR	$70.83_{\pm 0.44}$	$87.8_{\pm 0.08}$
TR + HC	$70.17_{\pm 0.31}$	$88.33_{\pm 0.2}$
TR + HSW (Ours)	$71.07_{\pm 0.04}$	$88.6_{\pm 0.49}$
TR + SSW (Ours)	$71.33_{\pm 0.34}$	$88.3_{\pm 0.05}$

Fig. 4. Comparison on CATER Object Tracking. Here, we compare the Top-1 and Top-5 accuracy of Transformers with shared workspace and Transformers with self-attention. We can see that Transformers with a shared workspace outperform those with pairwise self-attention.

a limited capacity workspace should encourage the model to look at relevant information that is important for the downstream objective. We test this hypothesis on a set of visually challenging benchmarks. For our experiments, we use either Transformers or RIMs as a backbone. We consider variants of Transformers based on different subsets of important properties. *Transformers* [TR]: Self-attention based multi-layer architecture (Vaswani et al., 2017) with shared parameters across layers. *Set transformer* [ISAB]: Transformers where self attention is replaced by ISAB module (Lee et al., 2019a). *Sparse Transformers* [STR]: Transformers with sparse factorizations of the attention matrix (Child et al., 2019). *High Capacity Transformers* [TR+HC]: Same as TR but with different parameters across layers. *Transformers with Shared Workspace with soft-competition* [TR+SSW]: Transformers with different positions competing with each other to write in shared workspace using soft-competition. *Transformers with Shared Workspace with top-k competition* [TR+HSW]: Transformers with different positions competing with each other to write in shared workspace using top-k competition.

Detecting Equilateral Triangles. We first use a simple toy task to test our hypothesis where the model should detect equilateral triangles in images (Ahmad & Omohundro, 2009). Each image is of size 64×64 and contains 3 randomly placed clusters of points. For equilateral triangles, the midpoints of these clusters are equidistant from each other. This is a binary classification task where the model has to predict whether the three given clusters form an equilateral triangle or not. To feed an image into a Transformer, we follow the same methodology as used in vision Transformers (Dosovitskiy et al., 2020). We first divide an image into equal sized 4×4 patches and treat each patch as a different input position of the Transformer.

To solve this task correctly, the model only needs to attend to relevant information i.e., to patches that contain the cluster of points. Therefore, using a limited capacity shared workspace should be useful here. Our results (presented in Figure 3) confirm this hypothesis. We can see that *Transformers with shared workspace attention converge much faster and*

reach higher accuracy as compared to the baseline Transformer. Our method also outperforms Set Transformer by a significant margin.

Multi MNIST Generation. In this task, we train an Image Transformer (Parmar et al., 2018) (pixel-by-pixel, raster-order generative model) for next-pixel prediction on the "MultiMNIST dataset" where each image consists of 4 independently sampled MNIST digits stacked horizontally to form one image (see Figure **??** for demonstration). The main aim of this task is to observe the inductive biases that allow for specialization of mechanisms in TIMs (Lamb et al., 2021). Each image in the MultiMNIST dataset can be broken down into different sets of independent spatial components. Since the digits which make up the image are independently selected, the joint distribution of pixel intensities in any one of the four sections of the image is statistically independent of the pixel intensities in any other section of the image. Moreover each section of the image can be further broken down into independent spatial components: one that pertains to the background and one that pertains to the foreground. One can expect that architectures that are made up of sparsely interacting different mechanisms to naturally capture this statistical independence by dividing labour among different mechanisms. While, for monolithic architectures, a major portion of their training time will be spent in learning these statistical independencies from scratch. We find that replacing the pairwise communication in TIMs with a shared workspace (TIMs + SW) leads to better and more interpretable division of labor among specialists as shown in Figure 6. From the figure, It is clear that the TIMs model is unable to divide labour among specialists with mechanism 2 being activation for all the pixels in the image. On the other hand, we can see that TIMs + SW is able to divide labor among specialists with each mechanism focusing on a different aspect of the image. We can see that mechanism 2 gets activated for the digits which are present towards the centre of each of the 4 columns while mechanisms 3 and 4 cover the background of the digits, with mechanism 3 covering the area between adjacent digits and mechanism 4 covering the area above and below the digits. Thus, we can see that using a shared workspace aids the division of labor among different specialists. We also find that TIMs + SW results in the least cross-entropy loss in the test set when compared to TIMs and Image Transformers (Parmar et al., 2018).

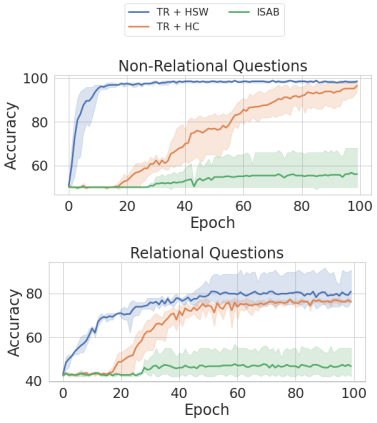
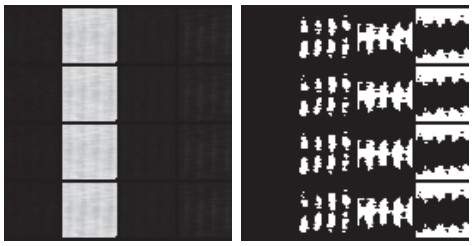


Fig. 5. Comparison on Sort-of-CLEVR relational reasoning. Speed of convergence for relational and non-relational questions in the sort-of-clevr dataset. We can see that the proposed model converges much faster than the baselines in both cases.

CATER: Object Tracking. Cater is a spatio-temporal reasoning video dataset introduced in Girdhar & Ramanan (2019). Each video contains 3D objects organized in a 6×6 grid. Each object affords certain actions that can be performed on them. These actions result in movement of the concerned objects and change in their positions. Some of these actions include: *rotate*, *pick-place*, *slide*, *contain*. Throughout the duration of the video, a number of these actions are performed to get the final state of the grid. Note that only a single object undergoes an action, at any instant. The task that we focus on here is called *localization*. In this task, the goal is to predict the location of the target object in the final frame. In this case the target object is called a snitch. The snitch as well as the other objects move across

the 6×6 grid. In some scenarios, the snitch may be covered by other objects hence hiding it from the view. In such cases, tracking the movement of the snitch across frames becomes essential. Therefore, capturing long-range temporal dependencies is essential to solve this task.

The information exchange limit enforced by the limited capacity of the shared workspace should be useful here as well. For CATER, in some frames the snitch is not visible as it is covered by other objects. Therefore, ideally the model only needs to attend to frames in which the snitch is visible. Additionally, if the snitch is visible throughout the video in all frames, then to accurately predict the final position of the snitch, the model only needs to attend to the final frame of the video and can completely ignore the initial frames. The results for this task are presented in Table 4. We also experimented with both soft competition TR+SSW and hard competition TR+HSW, with only $k = 5$ specialists writing into the shared workspace. We can see that models with a shared workspace outperform those with pairwise multihead attention thus confirming our hypothesis about the benefits of a shared workspace for this task. As shown in Table 4 proposed method convincingly outperforms the Set Transformer.



a Mechanism Activation Maps for TIMs

b Mechanism Activation Maps for TIMs+SW

Fig. 6. This figure shows the mechanism activation map for all 4 mechanims used in the multi-mnist generation task for both TIMs and TIMs + SW. Both the images in the figure correspond to the activation maps from 4 different examples. Each activation map contains 4 mechanisms shown from left to right in a single row. Each mechanism is shown using a 32 x 32 image, a particular pixel in a mechanism activation map is shown in white if that mechanism was used during the generation of that pixel while generating the image.

Relational Reasoning : Sort-of-CLEVR. In relational reasoning, the model is tasked with answering questions about certain properties of various objects and their relations with other objects. The model is presented with an image and a question for that image. This task has a clear sparse structure as in order to answer the questions correctly, it needs to only reason about a specific subset of objects that the question mentions. For this task, we use the Sort-of-CLEVR dataset (Santoro et al., 2017).

Model	Num. Slots	ARI ↑	MSE ↓
SCOFF	-	$0.276_{\pm 0.001}$	$0.083_{\pm 0.0}$
SCOFF + SW	2	$0.154_{\pm 0.007}$	$0.135_{\pm 0.002}$
SCOFF + SW	4	$0.487_{\pm 0.085}$	$0.059_{\pm 0.0}$
SCOFF + SW	5	$0.915_{\pm 0.0}$	$0.035_{\pm 0.0}$
SCOFF + SW	8	$0.891_{\pm 0.001}$	$0.039_{\pm 0.0}$
SCOFF + SW	10	$0.351_{\pm 0.001}$	$0.08_{\pm 0.0}$

Table 1. Here we show the performance of SCOFF augmented with shared workspace attention on the bouncing balls task. We also analyse the effect of varying number of slots in the shared workspace. This also shows that by increasing the number of slots performance decreases hence validating claims regarding bandwidth limited communication channel via shared workspace.

Each image in Sort-of-CLEVR is of size 75×75 and contains 6 randomly placed geometrical shapes of 6 possible colors and 2 possible shapes. Each image comes with 10 relational questions and 10 non-relational questions. Non-relational questions only consider properties of individual objects. On the other hand, relational questions consider relations among multiple objects. The input to the model consists of the image and the corresponding question. We first obtain a sequence of equal-sized patches for the image as in vision Transformers (Dosovitskiy et al.,

2020). We concatenate the resulting patch sequence with the representation of the question and pass the combined sequence through the Transformer. Sort-of-CLEVR has a finite number of possible answers, hence this task is setup as a classification task.

We present the results for this task in Figure 5. We observe that *the Transformers with the shared workspace converge faster and outperform the baselines for relational as well as non-relational questions*. The superior performance with shared memory can be attributed to the inherent sparsity of this task. For instance, in non-relational questions, the model only needs to attend to a single object referenced in the question to answer it correctly, while relational questions only consider a small subset of objects in the image, thus sparsity is helpful for both these types of questions. Therefore, the limited capacity of the shared workspace forces the model to attend to only relevant information.

Shared Workspace for Physical Reasoning. In this task, we consider a set of bouncing balls and the model is tasked with predicting the trajectory of the balls at each step. In order to solve this task, a coherent picture of where and which objects will collide needs to be established by the learner. We use the bouncing-ball dataset from Van Steenkiste et al. (2018). We train the model for next-step prediction. We compare the proposed approach against SCOFF (Goyal et al., 2020). The results of our comparison are shown in Table 1. We use the ARI and MSE metric for comparison. ARI measures how well the different balls are segregated into different slots, higher ARI means better segregation. We can see that using a shared workspace results in higher ARI as compared to pairwise communication in SCOFF. Thus, using a shared workspace results in better division of labor among specialists.

Shared Workspace for Atari Video Games. We start by training RIMs, RIMs + shared workspace (SW) on three "source" games (Pong, River Raid, and Seaquest) and test if the learned features transfer to a different subset of randomly selected "target" games (Alien, Asterix, Boxing, Centipede, Gopher, Hero, James Bond, Krull, Robotank, Road Runner, Star Gunner, and Wizard of Wor). We take a sufficient number of specialists in RIMs (10). We train on source games for 10M steps, and then fine-tune on transfer games for 10M more steps. We choose these games as they were also used in the original RIMs paper (Goyal et al., 2019c). Using a suite of 36 game pairs, we find that RIMs + SW outperforms RIMs on both game A (a median performance ratio of 1.13; mean of 1.16) and game B (a median performance ratio of 1.11; mean of 1.15). The improved performance with RIMs + SW is due to better forward transfer (knowledge acquired for game A facilitates the learning of game B) and reduced backward interference (knowledge acquired for game B does not disrupt knowledge acquired for game A), presumably thanks to a more appropriate modularization of knowledge.

7.5. Conclusion

Inspired by cognitive neuroscience global workspace theories, we have proposed a shared workspace model for establishing coherence among modular neural specialists while exchanging information in a systematic way. We show that using a limited capacity shared workspace as a bottleneck for mediating communication among specialists results in better performance across a wide range of visual reasoning benchmarks as compared to the pairwise interactions typically used in self-attention schemes.

The proposed approach combines several key properties: knowledge and expertise is divided among specialists, they compete to post new contents to the workspace, and after being updated, the shared workspace is accessible to all specialists for their own updates.

Chapter 8

Prologue to the third article

8.1. Article Details

Reinforcement Learning with Competitive Ensembles of Information-Constrained Primitives Anirudh Goyal, Shagun Sodhani, Jonathan Binas, Xue Bin Peng, Sergey Levine, Yoshua Bengio. This work was accepted for presentation at *International Conference on Learning Representations (ICLR), 2020.*

Personal Contribution. This work was conducted during my visit to Prof. Sergey Levine's lab at U.C Berkeley (Fall 2018). When thinking about factorizing knowledge into set of policies, Yoshua Bengio used to mention to train a *central controller* which learns to select which policy to activate. I did not resonate with the thought of learning such a central controller to decide which policy to activate. I was thinking there should be a decentralized approach where each policy can decide in which state to act. Thus I had the initial idea of learning policies in completely decentralized fashion. I ran experiments for continuous control experiments. Shagun Sodhani ran experiments for Four Rooms and BabyAI. Xue Bin Peng ran imitation learning experiments. Jonathan Binas and myself wrote majority of the paper. Sergey Levine and Yoshua Bengio assisted in general high level advising of the project.

8.2. Context

A central challenge in reinforcement learning is discovering effective policies for tasks where rewards are sparsely distributed. In our previous work (Goyal et al., 2019a), we postulate that in the absence of useful reward signals, an effective exploration strategy should seek out *decision states*. These states lie at critical junctions in the state space from where the agent can transition to new, potentially unexplored regions. We propose to learn about decision states from prior experience. By training a goal-conditioned policy with an information bottleneck, we can identify decision states by examining where the model actually leverages the goal state. We find that this simple mechanism effectively identifies decision states, even in partially observed settings. In effect, the model learns the sensory cues that correlate with potential subgoals. In new environments, this model can then identify novel subgoals for further exploration, guiding the agent through a sequence of potential decision states and through new regions of the state space. In this work, we extended the idea of training a policy with an information bottleneck to policy primitives such that different policies "activate" in different parts of the state space.

8.3. Contributions

Reinforcement learning agents that operate in diverse and complex environments can benefit from the structured decomposition of their behavior. Often, this is addressed in the context of hierarchical reinforcement learning, where the aim is to decompose a policy into lower-level primitives or options, and a higher-level meta-policy that triggers the appropriate behaviors for a given situation. However, the meta-policy must still produce appropriate decisions in all states. In this work, we propose a policy design that decomposes into primitives, similarly to hierarchical reinforcement learning, but without a high-level meta-policy. Instead, each primitive can decide for themselves whether they wish to act in the current state. We use an information-theoretic mechanism for enabling this decentralized decision: each primitive chooses how much information it needs about the current state to make a decision and the primitive that requests the most information about the current state acts in the world. The primitives are regularized to use as little information as possible, which leads to natural competition and specialization. We experimentally demonstrate that this policy architecture improves over both flat and hierarchical policies in terms of generalization.

8.4. Research Impact

Most of the work in hierarchical reinforcement learning focused on learning policy primitives and a meta-policy such that the meta-policy decides which policy primitive should activate for decision making. In this work, we propose a *decentralized* method where each primitive can decide for themselves whether they wish to act in the current state or not. The general idea of making decisions in decentralized fashion has changed my entire research trajectory (as summarized in chapter. 5 as well as influenced the broader research community). There have been a few follow-ups using the similar idea (Tirumala et al., 2019, 2020) including a review paper. This work was a followup to my previous work InfoBot (Goyal et al., 2019a) which has been used extensively for training goal conditioned policies with information bottleneck for improving generalization. At the time of writing, this was the only work that can learn different primitives corresponding to different "motions" for continuous control tasks directly from pixel observations.

The work we have presented bears some interesting connections to cognitive science and neuroscience. Both of these fields draw a fundamental distinction between automatic and controlled action selection (Miller et al., 2001). In automatic responses, perceptual inputs directly trigger actions according to a set of habitual stimulus-response associations. In controlled behaviour, automatic responses are overridden in order to align behaviour with a more complete representation of task context, including current goals. As an example, on the drive to work, automatic responding may trigger the appropriate turn at a familiar intersection, but top-down control may be needed to override this response if the same intersection is encountered on the route to a less routine destination. Our proposed architecture contains two pathways that correspond rather directly to the automatic and controlled pathways that have been posited in cognitive neuroscience models. In the neuroscience context, representation of task context and the function of overriding automatic responses has been widely linked with the prefrontal cortex and it is interesting to consider the route within InfoBot from goal to action representations in this light. Notably, recent work has suggested that prefrontal control processes are associated with subjective costs; ceteris paribus, human decision-makers will opt for habitual or automatic routes to behaviour. This correspondence with neuroscience

provides some indirect encouragement for the approach implemented in the present work. In turn, proposed framework provides an indication for why a cost of control may exist in human cognition, namely that this encourages the emergence of useful habits, with payoffs for efficient exploration and transfer.

Chapter 9

Reinforcement Learning with Competitive Ensembles of Information-Constrained Primitives

9.1. Introduction

Learning policies that generalize to new environments or tasks is a fundamental challenge in reinforcement learning. While deep reinforcement learning has enabled training powerful policies, which outperform humans on specific, well-defined tasks (Mnih et al., 2015a), their performance often diminishes when the properties of the environment or the task change to regimes not encountered during training.

This is in stark contrast to how humans learn, plan, and act: humans can seamlessly switch between different aspects of a task, transfer knowledge to new tasks from remotely related but essentially distinct prior experience, and combine primitives (or skills) used for distinct aspects of different tasks in meaningful ways to solve new problems. A hypothesis hinting at the reasons for this discrepancy is that the world is inherently compositional, such that its features can be described by compositions of small sets of primitive mechanisms (Parascandolo et al., 2017). Since humans seem to benefit from learning skills and learning to combine skills, it might be a useful inductive bias for the learning models as well.

This is addressed to some extent by the hierarchical reinforcement learning (HRL) methods, which focus on learning representations at multiple spatial and temporal scales, thus enabling better exploration strategies and improved generalization performance (Dayan & Hinton, 1993; Sutton et al., 1999b; Dietterich, 2000; Kulkarni et al., 2016). However, hierarchical approaches rely on some form of learned high-level controller, which decides when to activate different components in the hierarchy. While low-level sub-policies can specialize to smaller portions of the state space, the top-level controller (or master policy) needs to know how to deal with any given state. That is, it should provide optimal behavior for the entire accessible state space. As the master policy is trained on a particular state distribution, learning it in a way that generalizes to new environments effectively becomes the bottleneck for such approaches (Sasha Vezhnevets et al., 2017; Andreas et al., 2017).

We argue, and empirically show, that in order to achieve better generalization, the interaction between the low-level primitives and the selection thereof should itself be performed without requiring a single centralized network that understands the entire state space. We, therefore, propose a decentralized approach as an alternative to standard HRL, where we

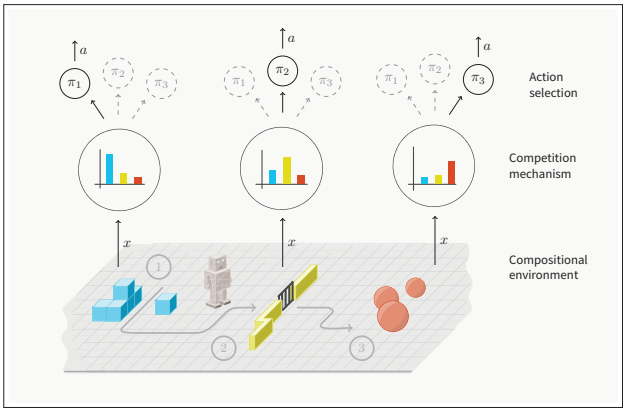


Fig. 1. Illustration of our model (Left): An intrinsic competition mechanism, based on the amount of information each primitive requests, is used to select a primitive to be active for a given input. Each primitive focuses on distinct features of the environment; in this case, one policy focuses on boxes, a second one on gates, and the third one on spheres. Right: The primitive-selection mechanism of our model. The primitive with most information acts in the environment and gets the reward.

only learn a set of low-level primitives without learning an explicit high-level controller. In particular, we construct a factorized representation of the policy by learning simple *primitive* policies, which focus on distinct regions of the state space. Rather than being gated by a single meta-policy, the primitives directly compete with one another to determine which one should be active at any given time, based on the degree to which their state encoders "recognize" the current state input. While, technically, the competition between primitives implicitly realizes a global selection mechanism, we consider our model *decentralized* in the sense that individual primitives can function on their own, and can be combined in new ways, without relying on an explicit high-level controller.

We frame the problem as one of information transfer between the current state and a dynamically selected primitive policy. Each policy can, by itself, decide to request information about the current state, and the amount of information requested is used to determine which primitive acts in the current state. Since the amount of state information that a single primitive can access is limited, each primitive is encouraged to use its resources wisely. Constraining the amount of accessible information in this way naturally leads to a decentralized competition and decision mechanism where individual primitives specialize in smaller regions of the state space. We formalize this information-driven objective based on the variational information bottleneck. The resulting set of competing primitives achieves both a meaningful factorization of the policy and an effective decision mechanism for which primitives to use. Importantly, not relying on a centralized meta-policy enables the individual primitive mechanisms can be recombined in a *plug-and-play* fashion, and the primitives can be transferred seamlessly to new environments.

Contributions: In summary, the contributions of our work are as follows: (1) We propose a method for learning and operating a set of functional primitives in a decentralized way, without requiring an explicit high-level meta-controller to select the active primitives (see Fig. 1 for illustration). (2) We introduce an information-theoretic objective, the effects of which are twofold: a) it leads to the specialization of individual primitives to distinct regions of the state space, and b) it enables a competition mechanism, which is used to select active primitives in a decentralized manner. (3) We demonstrate the superior transfer learning performance of our model, which is due to the flexibility of the proposed framework regarding

the dynamic addition, removal, and recombination of primitives. Decentralized primitives can be successfully transferred to larger or previously unseen environments, and outperform models with an explicit meta-controller for primitive selection.

9.2. Preliminaries

We consider a Markov decision process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where the state space $\mathcal{S}$ and the action space $\mathcal{A}$ may be discrete or continuous. The environment emits a bounded reward $r : \mathcal{S} \times \mathcal{A} \to [r_{min}, r_{max}]$ on each transition and $\gamma \in [0, 1)$ is the discount factor. $\pi(.|s)$ denotes a policy over the actions given the current state s. $R(\pi) = \mathbb{E}_\pi[\sum_t \gamma^t r(s_t)]$ denotes the expected total return when an agent follows the policy π. The standard objective in reinforcement learning is to maximize the expected total return $R(\pi)$. We use the concept of the information bottleneck (Tishby et al., 2000b) to learn compressed representations. The information bottleneck objective is formalized as minimizing the mutual information of a *bottleneck* representation layer with the input while maximizing its mutual information with the corresponding output. This type of input compression has been shown to improve generalization (Achille & Soatto, 2016; Alemi et al., 2016a).

9.3. Information-Theoretic Learning of Distinct Primitives

Our goal is to learn a policy, composed of multiple primitive sub-policies, to maximize the expected reward over T-step interactions for a distribution of tasks. Simple primitives which focus on solving a part of the given task (and not the complete task) should generalize more effectively, as they can be applied to similar aspects of different tasks (subtasks) even if the overall objective of the tasks are drastically different. Learning primitives in this way can also be viewed as learning a factorized representation of a policy, which is composed of several *independent* policies.

Our proposed approach consists of three mechanisms: 1) a mechanism for restricting a particular primitive to a subset of the state space; 2) a competition mechanism between primitives to select the most effective primitive for a given state; 3) a regularization mechanism to improve the generalization performance of the policy as a whole. We consider experiments with both fixed and variable sets of primitives and show that our method allows for primitives to be added or removed during training, or recombined in new ways. Each primitive is represented by a differentiable, parameterized function approximator, such as a neural network.

9.3.1. Primitives with an Information Bottleneck

To encourage each primitive to encode information from a particular part of state space, we limit the amount of information each primitive can access from the state. In particular, each primitive has an information bottleneck with respect to the input state, preventing it from using all the information from the state.

We define the overall policy as a mixture of primitives,

$$\pi(a \mid s) = \textstyle\sum_k c_k \pi^k(a \mid s)\,,$$

where $\pi^k(a \mid s)$ denotes the k^{th} primitive and $c_k = \delta_{kk'}$ for $k' \sim p(k' \mid s)$. We denote the probability of selecting the k^{th} primitive as $\alpha_k(s) := p(k \mid s)$.

Rather than learning an explicit model for $p(k \mid s)$, however, we impose an information-based mechanism for selecting primitives, wherein we limit the amount of information each primitive can contain and select the ones that request the most information about the state. To implement an information bottleneck, we design each of the K primitives to be composed of an encoder $p_{\text{enc}}(z_k \mid s)$ and a decoder $p_{\text{dec}}(a \mid z_k)$, together forming the primitive policy,

$$\pi_\theta^k(a \mid s) = \int_z p_{\text{enc}}(z_k \mid s)\, p_{\text{dec}}(a \mid z_k)\, \mathrm{d}z_k\,.^1$$

The encoder output z_k is meant to represent the information about the current state s that an individual primitive k believes is important to access in order to perform well. The decoder takes this encoded information and produces a distribution over the actions a. Following the variational information bottleneck objective (Alemi et al., 2016a), we penalize the KL divergence of $p_{enc}(z_k|s)$ and a prior $p(z)$,

$$\mathcal{L}_k = \mathrm{D}_{\text{KL}}\left(p_{\text{enc}}(z_k \mid s)||p(z)\right)\,. \tag{9.1}$$

In other words, a primitive pays an "information cost" proportional to $\mathcal{L}_k$ for accessing the information about the current state.

In the experiments below, we fix the prior to be a unit Gaussian. In the general case, we can learn the prior as well and include its parameters in θ. The information bottleneck encourages each primitive to limit its knowledge about the current state, but it will not prevent multiple primitives from specializing to similar parts of the state space. To mitigate this redundancy, and to make individual primitives focus on different regions of the state space, we introduce an information-based competition mechanism to encourage diversity among the primitives.

9.3.2. Competing Information-Constrained Primitives

We can use the information measure from equation 9.1 to define a selection mechanism for the primitives without having to learn a centralized meta-policy. The intuition is that the information content of an individual primitive encodes its effectiveness in a given state s such that the primitive with the highest value $\mathcal{L}_k$ should be activated in that particular state.

In particular, we set $\alpha_k = Z^{-1}\exp(\beta\mathcal{L}_k)$ to obtain a distribution over k as a function of the information content, activating the primitives with the highest information content. Here, $Z = \sum_k \exp(\beta\mathcal{L}_k)$ is a normalization constant. This mechanism enables competition between primitives, leading them to focus on parts of the state space that they "understand" well and letting others act in other parts.

Trading reward and information. To perform proper credit assignment, the environment reward is distributed to primitives according to their participation in the global decision, i.e. the reward r_k given to the k^{th} primitive is weighted by its selection coefficient, such that $r_k = \alpha_k r$, with $r = \sum_k r_k$. Hence, a primitive can potentially get a higher reward when deciding to act, but it also pays a higher price for accessing more information about the current state. The information bottleneck and the competition mechanism, when combined with the overall reward maximization objective, will lead to specialization of individual primitives to distinct regions in the state space. That is, each primitive should specialize in a part of the state space that it can reliably associate rewards with. Since the entire ensemble still needs to understand all of the state space for the given task, different primitives need to encode and focus on different parts of the state space.

[1] In practice, we estimate the marginalization over z using a single sample throughout our experiments.

9.3.3. Regularizing Primitive Selection

The objective described above will optimize the expected return while minimizing the information content of individual primitives. This is not sufficient, however, as it might lead to highly unbalanced outcomes: some primitives might be more active initially and learn to become even more active, completely disabling other primitives.

Thus, in addition to minimizing each primitive's absolute information content, we need to normalize their activity with respect to each other. To do so, we penalize their information content in proportion to their activation by adding a regularization term of the form

$$\mathcal{L}_{\text{reg}} = \textstyle\sum_k \alpha_k \mathcal{L}_k \,. \tag{9.2}$$

Note that this can be rewritten as $\mathcal{L}_{\text{reg}} = -H(\alpha) + \text{LSE}(\mathcal{L}_1, \ldots, \mathcal{L}_K)$, where $H(\alpha)$ is the entropy of α, and LSE is the *LogSumExp* function, $\text{LSE}(x) = \log(\sum_j e^{x_j})$. Thus, minimizing $\mathcal{L}_{\text{reg}}$ increases the entropy of α, leading to a diverse set of primitive selections, in turn, ensuring that different combinations of the primitives are used. Similarly, LSE approximates the maximum of its arguments, $\text{LSE}(x) \approx \max_j x_j$, and, therefore, penalizes the dominating $\mathcal{L}_k$ terms, thus equalizing their magnitudes.

9.3.4. Objective and Algorithm Summary

Our overall objective function consists of 3 terms,
(1) The expected return from the standard RL objective, $R(\pi)$ which is distributed to the primitives according to their participation,
(2) The individual bottleneck terms leading the individual primitives to focus on specific parts of the state space, $\mathcal{L}_k$ for $k = 1, \ldots, K$,
(3) The regularization term applied to the combined model, $\mathcal{L}_{\text{reg}}$.
The overall objective for the k^{th} primitive thus takes the form:

$$J_k(\theta) \equiv \mathbb{E}_{\pi_\theta}[r_k] - \beta_{\text{ind}}\mathcal{L}_k - \beta_{\text{reg}}\mathcal{L}_{\text{reg}} \,, \tag{9.3}$$

where $\mathbb{E}_{\pi_\theta}$ denotes an expectation over the state trajectories generated by the agent's policy, $r_k = \alpha_k r$ is the reward given to the kth primitive, and β_{ind}, β_{reg} are the parameters controlling the impact of the respective terms.

Implementation: In our experiments, the encoders $p_{\text{enc}}(z_k \mid s)$ and decoders $p_{\text{dec}}(a \mid z_k)$ (see. Fig. 1) are represented by neural networks, the parameters of which we denote by θ. Actions are sampled through each primitive every step. While our approach is compatible with any RL method, we maximize $J(\theta)$ computed on-policy from the sampled trajectories using a score function estimator (Williams, 1992; Sutton et al., 1999a) specifically A2C (Mnih et al., 2016) (unless otherwise noted). Every experimental result reported has been averaged over 5 random seeds. Our model introduces 2 extra hyper-parameters β_{ind}, β_{reg}.

9.4. Related Work

There are a wide variety of hierarchical reinforcement learning approaches(Sutton et al., 1998; Dayan & Hinton, 1993; Dietterich, 2000). One of the most widely applied HRL framework is the *Options* framework ((Sutton et al., 1999b)). An option can be thought of as an action that extends over multiple timesteps, thus providing the notion of temporal abstraction or subroutines in an MDP. Each option has its own policy (which is followed if the option is selected) and the termination condition (to stop the execution of that option).

Many strategies are proposed for discovering options using task-specific hierarchies, such as pre-defined sub-goals (Heess et al., 2017), hand-designed features (Florensa et al., 2017), or diversity-promoting priors (Daniel et al., 2012; Eysenbach et al., 2018). These approaches do not generalize well to new tasks. Bacon et al. (2017) proposed an approach to learn options in an end-to-end manner by parameterizing the intra-option policy as well as the policy and termination condition for all the options. Eigen-options (Machado et al., 2017) use the eigenvalues of the Laplacian (for the transition graph induced by the MDP) to derive an intrinsic reward for discovering options as well as learning an intra-option policy.

In this work, we consider a sparse reward setup with high dimensional action spaces. In such a scenario, performing unsupervised pretraining or using auxiliary rewards leads to much better performance (Frans et al., 2017; Florensa et al., 2017; Heess et al., 2017). Auxiliary tasks such as motion imitation have been applied to learn motor primitives that are capable of performing a variety of sophisticated skills (Liu & Hodgins, 2017; Peng et al., 2017; Merel et al., 2019b,a). Our work is also related to the *Neural Module Network* family of architectures (Andreas et al., 2017; Johnson et al., 2017; Rosenbaum et al., 2019b) where the idea is to learn *modules* that can perform some useful computation like solving a subtask and a *controller* that can learn to combine these modules for solving novel tasks. More recently, Wu et al. (2019) proposed a framework for using diverse suboptimal world models to learn primitive policies. The key difference between our approach and all the works mentioned above is that we learn functional primitives without requiring any explicit high-level meta-controller or master policy.

9.5. Experimental Results

In this section, we briefly outline the tasks that we used to evaluate our proposed method and direct the reader to the appendix for the complete details of each task along with the hyperparameters used for the model. We designed experiments to address the following questions: **a) Learning primitives** – Can an ensemble of primitives be learned over a distribution of tasks? **b) Transfer Learning using primitives** – Can the learned primitives be transferred to unseen/unsolvable sparse environments? **c) Comparison to centralized methods** – How does our method compare to approaches where the primitives are trained using an explicit meta-controller, in a centralized way?

Baselines. We compare our proposed method to the following baselines: **a) Option Critic** (Bacon et al., 2017) – We extended the author's implementation[2] of the Option Critic architecture and experimented with multiple variations in terms of hyperparameters and state/goal encoding. None of these yielded reasonable performance in partially observed tasks, so we omit it from the results. **b) MLSH** (Meta-Learning Shared Hierarchy) (Frans et al., 2017) – This method uses meta-learning to learn sub-policies that are shared across tasks along with learning a task-specific high-level master. It also requires a phase-wise training schedule between the master and the sub-policies to stabilize training. We use the MLSH implementation provided by the authors[3]. **c) Transfer A2C:** In this method, we first learn a single policy on the one task and then transfer the policy to another task, followed by fine-tuning in the second task.

[2]https://github.com/jeanharb/option_critic

[3]https://github.com/openai/mlsh

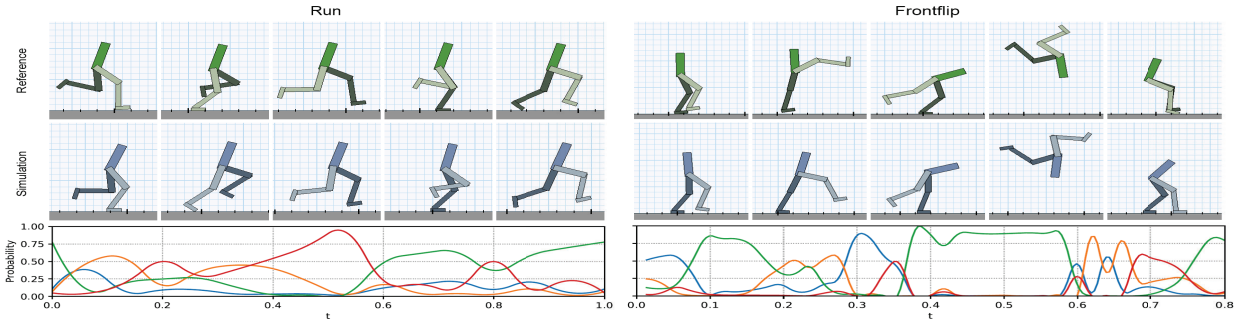


Fig. 2. Snapshots of motions learned by the policy. **Top:** Reference motion clip. **Middle:** Simulated character imitating the reference motion. **Bottom:** Probability of selecting each primitive.

9.5.1. Learning Ensembles of Functional Primitives

We evaluate our approach on a number of RL environments to demonstrate that we can indeed learn sets of primitive policies focusing on different aspects of a task and collectively solving it.

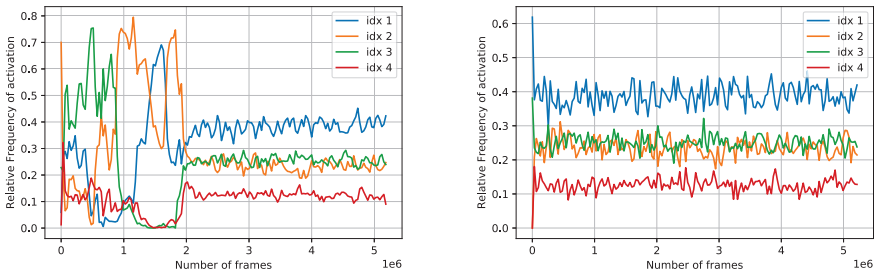


Fig. 3. Convergence of four primitives on Four Room Maze: Left: We trained four primitives on the Four Room Maze task, where the goal was sampled from one of the two fixed goals. We see that the proposed algorithm is able to learn four primitives. Right: We transfer the learned primitives to the scenario where the goal is sampled from one of the four possible goals. The checkpointed model is ran on 100 different episodes (after a fixed number of steps/updates) and the normalized frequency of activation of the different primitives is plotted.

Four Room Maze: We consider the Four-rooms gridworld environment (Sutton et al., 1999c) where the agent has to navigate its way through a grid of four interconnected rooms to reach a goal position within the grid. We consider the scenario where the starting position of the agent is fixed, but the goal is sampled from a discrete set. Fig. 3 shows that the proposed algorithm can learn four primitives.

Motion Imitation. To evaluate the proposed method in terms of scalability, we present a series of tasks from the motion imitation domain, showing that we can use a set of distinct primitives for imitation learning. In these tasks, we train a simulated 2D biped character to perform a variety of highly dynamic skills by imitating motion capture clips recorded from human actors. Each mocap clip is represented by a target state trajectory $\tau^* = \{s_0^*, s_1^*, ..., s_T^*\}$, where s_t^* denotes the target state at timestep t. The input to the policy is augmented with a goal $g_t = \{s_{t+1}^*, s_{t+2}^*\}$, which specifies the the target states for the next two timesteps. Both the state s_t and goal g_t are then processed by the encoder $p_{\text{enc}}(z_t|s_t, g_t)$. The repertoire of skills consists of 8 clips depicting different types of walks, runs, jumps, and flips. The motion imitation approach closely follows Peng et al. (2018). To analyze the specialization of the various primitives, we computed 2D embeddings of states and goals which each primitive is active in, and the actions proposed by the primitives. Fig. 4 illustrates the embeddings

computed with t-SNE (van der Maaten & Hinton, 2008). The embeddings show distinct clusters for the primitives, suggesting a degree of specialization of each primitive to certain states, goals, and actions.

9.5.2. Multi-Task Training

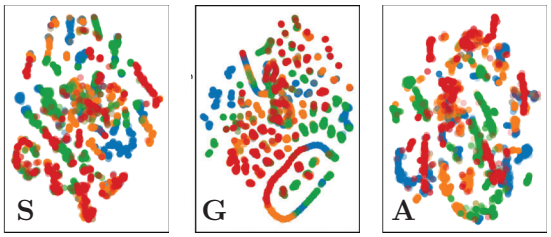


Fig. 4. Embeddings visualizing the states (S) and goals (G) which each primitive is active in, and the actions (A) proposed by the primitives for the motion imitation tasks. A total of four primitives are trained. The primitives produce distinct clusters.

We evaluate our model in a partially-observable 2D multi-task environment called Minigrid, similar to the one introduced in (Chevalier-Boisvert et al., 2018b). The environment is a two-dimensional grid with a single agent, impassable walls, and many objects scattered in the environment. The agent is provided with a natural language string that specifies the task that the agent needs to complete. The setup is partially observable, and the agent only gets the small, egocentric view of the grid (along with the natural language task description). We consider three tasks here: the *Pickup* task (A), where the agent is required to pick up an object specified by the goal string, the *Unlock* task (B) where the agent needs to unlock the door (there could be multiple keys in the environment, and the agent needs to use the key which matches the color of the door) and the *UnlockPickup* task (C), where the agent first needs to unlock a door that leads to another room. In this room, the agent needs to find and pick up the object specified by the goal string.

We train agents with varying numbers of primitives on various tasks – concurrently, as well as in transfer settings. The different experiments are summarized in Figs. 5 and 7. An advantage of the multi-task setting is that it allows for quantitative interpretability as to when and which primitives are being used. The results indicate that a system composed of multiple primitives generalizes more easily to a new task, as compared to a single policy. We further demonstrate that several primitives can be combined dynamically and that the individual primitives respond to stimuli from new environments when trained on related environments.

9.5.3. Do Learned Primitives Help in Transfer Learning?

We evaluate our approach in the settings where the adaptation to the changes in the task is vital. The argument in favor of modularity is that it enables better knowledge transfer between related tasks. Naturally, the transfer is easier when the tasks are closely related, as the model will only need to learn how to compose the already-learned primitives. In general, it is difficult to determine how closely related two tasks are, however, and the inductive bias of modularity could even be harmful if the two tasks are very different. In such cases, we could add new primitives (which would need to be learned) and still obtain a sample-efficient transfer, as some part of the task structure would already have been captured by the pretrained primitives. This approach can be extended towards adding primitives during training, providing a seamless way to combine knowledge about different tasks to solve more complex tasks. We investigate here the transfer properties of a primitive trained in one environment and transferred to a different one. Results are shown in Fig. 5.

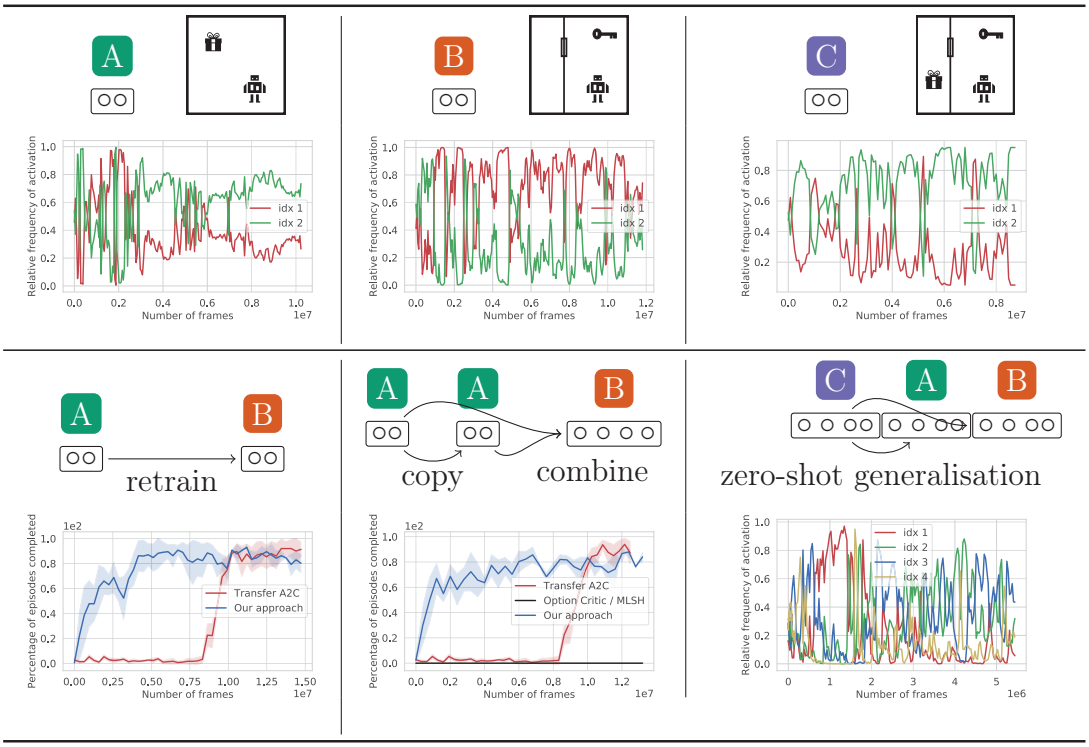


Fig. 5. Multitask training. Each panel corresponds to a different training setup, where different tasks are denoted A, B, C, ..., and a rectangle with n circles corresponds to an agent composed of n primitives trained on the respective tasks. Top row: activation of primitives for agents trained on single tasks. Bottom row: **Retrain:** Two primitives are trained on task A and transferred to task B. The results (success rates) indicate that the multi-primitive model is substantially more sample efficient than the baseline (transfer A2C). **Copy and Combine:** More primitives are added to the model over time in a plug-and-play fashion (two primitives are trained on task A; the model is extended with a copy of the two primitives; the resulting four-primitive model is trained on task B.) This is more sample efficient than other strong baselines, such as (Frans et al., 2017; Bacon et al., 2017). **Zero-Shot Generalization:** A set of primitives is trained on task C, and zero-shot generalization to task A and B is evaluated. The primitives learn a form of spatial decomposition which allows them to be active in both target tasks, A and B. The checkpointed model is ran on 100 different episodes, and the normalized frequency of activation of the different primitives is plotted.

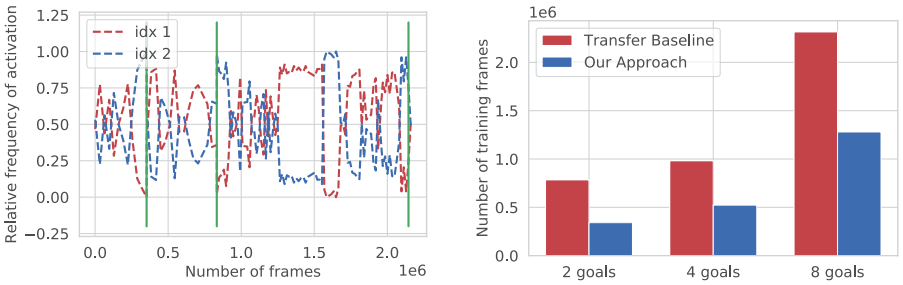
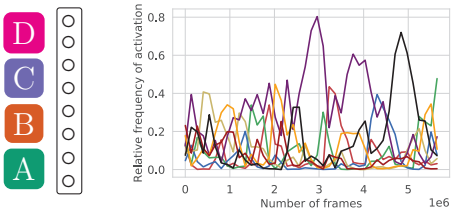


Fig. 6. Continual Learning Scenario: The plot on the left shows that the primitives remain activated. The solid green line shows the boundary between the tasks. The plot on the right shows the number of samples required by our model and the transfer baseline model across different tasks. We observe that the proposed model takes fewer steps than the baseline (an A2C policy trained in a similar way), and the gap in terms of the number of samples keeps increasing as tasks become harder. The checkpointed model is ran on 100 different episodes (after a fixed number of steps/updates) and the normalized frequency of activation of the different primitives is plotted.

Continuous control for ant maze. We evaluate the transfer performance of pretrained primitives on the cross maze environment (Haarnoja et al., 2018). Here, a quadrupedal ant robot must walk to the different goals along the different paths. The goal is randomly chosen from a set



Method	3 goals	10 goals
Flat Policy (PPO)	11 ± 5 %	4 ± 2 %
Option critic	18 ± 10 %	7 ± 3 %
MLSH	32 ± 3 %	5 ± 3 %
Explicit high level policy	21 ± 5 %	11 ± 2 %
Proposed method	68 ± 3%	40 ± 3%

Fig. 7. Left: Multitask setup, where we show that we are able to train eight primitives when training on a mixture of four tasks in the Minigrid environment. Here, the *x-axis* denotes the number of frames (timesteps). Right: Success rates of the different methods on the Ant Maze tasks. Success rate is measured as the number of times the ant is able to reach the goal (based on 500 sampled trajectories).

of available goals at the start of each episode. We pretrain a policy with a motion reward in an environment which does not have any walls (similar to Haarnoja et al. (2018)), and then transfer the policy to the second task where the ant has to navigate to a random goal chosen from one of the 3 (or 10) available goal options. For our model, we make four copies of the pretrained policies and then finetune the model using the pretrained policies as primitives. We compare to both MLSH (Frans et al., 2017) and option-critic (Bacon et al., 2017). All these baselines have been pretrained in the same manner. As evident from Fig. 7, our method outperforms the other approaches. The fact that the initial policies successfully adapt to the transfer environment underlines the flexibility of our approach.

Zero Shot Generalization: The purpose of this experiment is to show that the model consisting of multiple primitives is somewhat able to decompose the task C into its subtasks, A and B. The better this decomposition is the better should the model transfer to the individual subtasks. In order to test this, we trained a set of 4 primitives on task C, and then evaluate them (without finetuning) on tasks A and B. We note that the ensemble is able to solve the transfer tasks, A and B, successfully 72% of the time, while a monolithic policy's success rate is 38%. This further shows that the primitives learn meaningful decompositions.
Continual Learning: 4 Rooms Scneario. We consider a continual learning scenario where we train two primitives for two-goal positions ie the goal position is selected randomly from one of the two positions at the start of the episode. The primitives are then transfer (and finetuned) on four-goal positions then transfer (and finetune) on eight-goal positions. The results are shown in fig. 6. The proposed method achieves better sample efficiency as compared to training a single monolithic policy.

9.6. Summary and Discussion

We present a framework for learning an ensemble of primitive policies that can collectively solve tasks without learning an explicit master policy. Rather than relying on a centralized, learned meta-controller, the selection of active primitives is implemented through an information-theoretic mechanism. The learned primitives can be flexibly recombined to solve more complex tasks. Our experiments show that, on a partially observed "Minigrid" task and a continuous control "Ant Maze" walking task, our method can enable better transfer than flat policies and hierarchical RL baselines, including the Meta-learning Shared Hierarchies model and the Option-Critic framework. On Minigrid, we show how primitives trained with

our method can transfer much more successfully to new tasks. On the Ant Maze, we show that primitives initialized from a pretrained walking control can learn to walk to different goals in a stochastic, multi-modal environment with nearly twice the success rate of a more conventional hierarchical RL approach, which uses the same pretraining but a centralized high-level policy. The proposed framework could be very attractive for continual learning settings, where one could add more primitive policies over time. Thereby, the already learned primitives would keep their focus on particular aspects of the task, and newly added ones could specialize on novel aspects.

Chapter 10

Prologue to the fourth article

10.1. Article Details

Retrieval Augmented Reinforcement Learning. Anirudh Goyal, Abram L. Friesen, Theophane Weber, Andrea Banino, Nan Rosemary Ke, Adria Puigdomenech Badia, Arthur Guez, Mehdi Mirza, Peter C. Humphreys, Ksenia Konyushkova, Laurent Sifre, Michal Valko, Simon Osindero, Timothy Lillicrap, Nicolas Heess, Charles Blundell. This work is accepted for presentation at **International Conference on Machine Learning (ICML), 2022**.

Personal Contribution. This work was conducted during an internship at DeepMind. The original idea of augmenting an RL agent with a retrieval process which has access to past experiences came in a discussion with Timothy Lillicrap and Nicolas Heess. Adria Puigdomenech Badia and Andrea Banino helped me to implement the first version of the model using the DeepMind infrastructure. Abram L. Friesen helped me setup the baseline for GridRoboman experiments. Rosemary Nan Ke helped me to run baselines for Atari problems. Theophane Weber helped in general brainstorming. Arthur Guez, Peter Humphreys and Mehdi Mirza helped me to run experiments on GO (which are not included in this paper). Ksenia Konyushkova constructed the GridRoboman benchmark used in this paper. The paper was written by Abram L. Friesen, Theophane Weber, Andrea Banino and myself. Timothy Lillicrap, Michal Valko and Simon Osindero helped in structuring the paper. Laurent Sifre participated in group discussions and provided general feedback. Charles Blundell proof read the paper. David Silver advised the project.

10.2. Context

Learning long-term dependencies in extended temporal sequences requires credit assignment to events far back in the past. The most common method for training recurrent neural networks, back-propagation through time (BPTT), requires credit information to be propagated backwards through every single step of the forward computation, potentially over thousands or millions of time steps. This becomes computationally expensive or even infeasible when used with long sequences. Importantly, biological brains are unlikely to perform such detailed reverse replay over very long sequences of internal states (consider days, months, or years.) However, humans are often reminded of past memories or mental states which are associated with the current mental state. In our previous work, Sparse Attentive Backtracking (Ke et al., 2018) we considered the hypothesis that such memory associations between past and present could be used for credit assignment through arbitrarily

long sequences, propagating the credit assigned to the current state to the associated past state. Based on this principle, we proposed an algorithm which only back-propagates through a few of these temporal skip connections, realized by a learned attention mechanism that associates current states with relevant past states. The work presented in this thesis can be seen as an generalizing the idea of Sparse Attentive Backtracking for reinforcement learning problems.

Most deep reinforcement learning (RL) algorithms distill experience into parametric behavior policies or value functions via gradient updates. While effective, this approach has several disadvantages: (1) it is computationally expensive, (2) it can take many updates to integrate experiences into the parametric model, (3) experiences that are not fully integrated do not appropriately influence the agent's behavior, and (4) behavior is limited by the capacity of the model. In this paper we explore an alternative paradigm in which we train a network to map a dataset of past experiences to optimal behavior.

10.3. Contributions

In this work, we augment an RL agent with a retrieval process (parameterized as a neural network) that has direct access to a dataset of experiences. This dataset can come from the agent's past experiences, expert demonstrations, or any other relevant source. The retrieval process is trained to retrieve information from the dataset that may be useful in the current context, to help the agent achieve its goal faster and more efficiently. The proposed method facilitates learning agents that at test-time can condition their behavior on the entire dataset and not only the current state, or current trajectory. We integrate our method into two different RL agents: an offline DQN agent and an online R2D2 agent. In offline multi-task problems, we show that the retrieval-augmented DQN agent avoids task interference and learns faster than the baseline DQN agent. On Atari, we show that retrieval-augmented R2D2 learns significantly faster than the baseline R2D2 agent and achieves higher scores. We run extensive ablations to measure the contributions of the components of our proposed method.

10.4. Research Impact

This paper is very recent, and already led to a follow-up where the authors scale the idea where one can query a database which consists of millions of past experiences as compared to few thousands as used in this work (Humphreys et al., 2022). The proposed work also validates the conjecture put forward in (Logan et al., 2021): *Perceiving and remembering pose the same computational problems: desired information must be extracted from complex multidimensional structures. The conjecture is that the extraction process is selective attention. Turned outward, it retrieves information from perception. Turned inward, it retrieves information from memory.*

In the future, I hope to see many more papers using a paradigm in which we train a network to map a dataset of past experiences to optimal behavior.

Chapter 11

Retrieval Augmented Reinforcement Learning

11.1. Introduction

A host is preparing a holiday meal for friends. They remember that the last time they went to the grocery store during the holiday season, all of the fresh produce was sold out. Thinking back to this past experience, they decide to go early! The hypothetical host is employing *case-based reasoning* (e.g., Kolodner, 1992; Leake, 1996). Here, an agent *recalls* a situation similar to the current one and uses information from the previous experience to solve the current task. This may involve adapting old solutions to meet new demands, or using previous experiences to make sense of new situations.

In contrast, a dominant paradigm in modern reinforcement learning (RL) is to learn general purpose behaviour rules from the agent's past experience. These rules are typically represented in the weights of a parametric policy or value function network model. Most deep RL algorithms integrate information *across trajectories* by iteratively updating network parameters using gradients that are computed along *individual trajectories* (collected online or stored in an experience replay dataset, Lin, 1992). For example, many off-policy algorithms reuse past experience by "replaying" trajectory snippets in order to compute weight updates for a value function represented by a deep network (Ernst et al., 2005; Riedmiller, 2005; Mnih et al., 2015a; Heess et al., 2015; Lillicrap et al., 2015).

This paradigm has clear advantages but at least two interrelated limitations: First, after learning, an agent's past experiences no longer play a direct role in the agent's behavior, even if they are relevant to the current situation. This occurs because detailed information in the agent's past experience is lost due to practical constraints on network capacity. Second, since the information provided by individual trajectories first needs to be distilled into a general purpose parametric rule, an agent may not be able to exploit the specific guidance that a handful of individual past experiences could provide, nor rapidly incorporate novel experience that becomes available—it may take many replays through related traces in the past experiences for this to occur (Weisz et al., 2021).

In this work, we develop an algorithm that overcomes these limitations by *augmenting* a standard reinforcement learning agent with a *retrieval process* (parameterized via a neural network). The purpose of the retrieval process is to help the agent achieve its objective by providing relevant contextual information. To this end, the retrieval process uses a learned attention mechanism to dynamically access a large pool of past trajectories stored in a dataset (e.g., a replay buffer), with the aim of integrating information across these. The proposed method shown in Figure 1, enables an agent to retrieve information from a dataset of

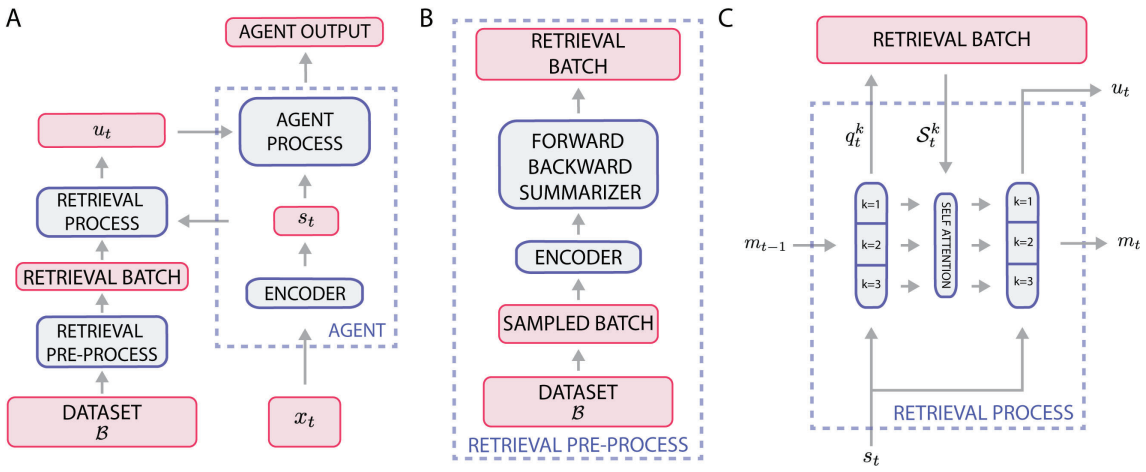


Fig. 1. Retrieval-augmented agent (R2A) architecture: (A) R2A augments the agent with a retrieval process. The retrieval process and the agent maintain separate internal states, $\boldsymbol{m}_t$ and $\boldsymbol{s}_t$, respectively. The retrieval process retrieves information relevant to the agent's current internal state $\boldsymbol{s}_t$ from the retrieval batch, which is a pre-processed sample from the retrieval dataset $\mathcal{B}$. The retrieved information $\boldsymbol{u}_t$ is used by the agent process to inform its output (e.g., a policy or value function). (B) A batch of raw trajectories is sampled from the retrieval dataset $\mathcal{B}$ and encoded (using the same encoder as the agent). Each encoded trajectory is then summarized via forward and a backward summarization functions (section 11.2.2) and sent to the retrieval process. (C) The retrieval process is parameterized as a recurrent model and the internal state $\boldsymbol{m}_t$ is partitioned into slots. Each slot independently retrieves information from the retrieval batch, which is used to update the slot's representation and sent to the agent process in $\boldsymbol{u}_t$. Slots also interact with each other via self-attention. See section 11.2.3 for more details.

trajectories. The high-level idea is to have two different processes. First, the *retrieval process* makes a "query" for relevant contextual information in the dataset. Second, the *agent process* performs inference and learning based on the information provided by the retrieval process. These two processes have different internal states but interact to shape the representations and predictions of each other: the agent process provides the relevant context, and the retrieval process uses the context and its own internal state to generate a query and retrieve relevant information, which is in turn used by the agent process to shape the representation of its policy and value function (see Fig. 1A). Our proposed retrieval-augmented RL paradigm could take several forms. Here, we focus on one particular instantiation applied to multiple different RL agents and environments to validate our hypothesis that learning a retrieval process can help an RL agent achieve its objective.

Summary of experimental results. We first show that the performance and sample efficiency of R2D2 (Kapturowski et al., 2018), a state-of-the-art off-policy RL algorithm, on Atari games can be improved by retrieval augmentation. In this setting, we run a series of ablations to demonstrate the benefits of our design decisions and to show how our approach compares with related work. In online Atari, the agent retrieves from its own experiences on the same game; however, retrieval can also query external data from other agents or other tasks. We thus evaluated on three separate multi-task offline RL environments (gridroboman, BabyAI (Chevalier-Boisvert et al., 2018a), CausalWorld (Ahmed et al., 2020)(a continuous control benchmark), where the retrieved data is first from a different agent in the same task and then from different agents and includes data from other tasks. In all cases, the retrieval-augmented agent learns faster and achieves higher reward.

11.2. Retrieval-Augmented Agents

We now present our method for augmenting an RL agent with a retrieval process, thereby reducing the agent's dependence on its model capacity, and enabling fast and flexible use of past experiences. A retrieval-augmented agent (R2A) consists of two main components: (1) the retrieval process, which takes in the current state of the agent, combines this with its own internal state, and retrieves relevant information from an external dataset of experiences; and (2) a standard reward-maximizing RL agent, which uses the retrieved information to improve its value or policy estimates. See Figure 1 for an overview. The retrieval process is trained to retrieve information that the agent can use to improve its performance, without explicit knowledge of the agent's policy. Importantly, the retrieval process has its own internal state, which enables it to integrate and combine information across retrievals. In the following, we focus on value-based methods, such as DQN (Mnih et al., 2015b) and R2D2 (Kapturowski et al., 2018), but our approach is equally applicable to policy-based methods.

11.2.1. Retrieval-augmented agent

Formally, the agent receives an input $\boldsymbol{x}_t$ at each timestep t. Each input is processed by a neural encoder (e.g., a resnet if the input is an image) to obtain an abstract internal state for the agent $\boldsymbol{s}_t = f_\theta^{\text{enc}}(\boldsymbol{x}_t)$. For clarity, we focus here on the case of a single vector input, however, each input could also include the history of past observations, actions, and rewards, as is the case when f_θ^{enc} is a recurrent network. These embeddings are used by the agent and retrieval processes. The retrieval process operates on a dataset $\mathcal{B} = \{((\boldsymbol{x}_t, a_t, r_t), \ldots, (\boldsymbol{x}_{t+l}, a_{t+l}, r_{t+l}))\}$ of l-step trajectories, for $l \geq 1$ (r_t refers to the reward at time-step t, if available). This dataset could come from other agents or experts, as in offline RL or imitation learning, or consist of the growing set of the agent's own experiences. Then, a retrieval-augmented agent (R2A) consists of the retrieval process and the agent process, parameterized by $\theta = \{\theta^{\text{enc}}, \theta^{\text{retr}}, \theta^{\text{agent}}\}$,

$$\textbf{Retrieval process } f_{\theta,\mathcal{B}}^{\text{retr}} : \boldsymbol{m}_{t-1}, \boldsymbol{s}_t \mapsto \boldsymbol{m}_t, \boldsymbol{u}_t$$
$$\textbf{Agent process } f_\theta^{\text{agent}} : \boldsymbol{s}_t, \boldsymbol{u}_t \mapsto Q_\theta(\boldsymbol{s}_t, \boldsymbol{u}_t, a)$$

Retrieval Process. The retrieval process is parameterized as a neural network and has an internal state $\boldsymbol{m}_t$. The retrieval process takes in the current abstract state of the agent process $\boldsymbol{s}_t$ and its own previous internal state $\boldsymbol{m}_{t-1}$ and uses these to retrieve relevant information from the dataset $\mathcal{B}$, which it then summarizes in a vector $\boldsymbol{u}_t$, and also updates its internal state $\boldsymbol{m}_t$.

Agent Process. The state of the agent $\boldsymbol{s}_t$ and the information from the retrieval process $\boldsymbol{u}_t$ are then passed to the action-value function, itself used to select external actions.

The above defines a parameterization for a retrieval-augmented agent. For retrieval to be effective, the retrieval process needs to: (1) be able to efficiently query a large dataset of trajectories, (2) learn and employ a similarity function to find relevant trajectories, and (3) encode and summarize the trajectories in a manner that allows efficient discovery of relevant past and future information.

Below, we explain how we achieve these desiderata. At a high-level, to reduce computational complexity given a experience dataset of hundreds of thousands of trajectories, R2A operates on samples from the dataset. R2A then encodes and summarizes the sampled trajectories using auxiliary losses and bi-directional sequence models to enable efficient retrieval of temporal information. Finally, R2A uses attention to select semantically relevant trajectories.

11.2.2. Retrieval batch sampling and pre-processing.

Sampling a retrieval batch from the retrieval dataset. To reduce computational complexity, R2A uniformly samples a large batch of past experiences from the retrieval dataset and then retrieves from the sampled batch. We denote the sampled batch as the "retrieval batch" and the number of trajectories in the retrieval batch as $n_{\text{retrieval}}$.

Encoding and forward-backward summarization of the retrieval dataset and corresponding auxiliary losses. Since the agent's internal state extracts information from observations which relate to the task at hand, we choose to re-encode the raw experiences in the "retrieval batch" using the agent encoder module (i.e., f_θ^{enc}). However, this representation is a function only of past observations (i.e., it's a causal representation) and may not be fully compatible with the needs of the retrieval operation. For that reason, we propose to further encode the retrieved batch of information by a learned *summarization* function, applied on the output of the encoder module, which captures information about the past and the future within a particular trajectory by using a bi-directional model (e.g., parameterized as a bi-directional RNN or a transformer).

$$\textbf{Forward Summarizer } f_\theta^{\text{fwd}} : (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_t) \mapsto \boldsymbol{h}_t$$
$$\textbf{Backward Summarizer } f_\theta^{\text{bwd}} : (\boldsymbol{s}_l, \ldots, \boldsymbol{s}_t) \mapsto \boldsymbol{b}_t$$

For each trajectory in the retrieval batch, we represent each time-step within a trajectory by a set of two vectors $\boldsymbol{h}_{i,t}$ and $\boldsymbol{b}_{i,t}$ where $\boldsymbol{h}_{i,t}$ summarizes the past (i.e., from $t' = 0$ to $t' = t$ time-steps of the i^{th} trajectory) while $\boldsymbol{b}_{i,t}$ summarizes the future (i.e., from $t' = t$ to $t' = l$ time-steps) within the i^{th} trajectory. In addition, taking inspiration from (Jaderberg et al., 2016; Ke et al., 2019; Devlin et al., 2018b; Mazoure et al., 2020), we use auxiliary losses to improve modeling of long term dependencies when training the parameters of our forward and backward summarizers. The goal of these losses is to force the representation $(\boldsymbol{h}_{i,t}, \boldsymbol{b}_{i,t})_{i,t\geq 0}$ to capture meaningful information for the unknown downstream task. For our experiments, we use supervised losses where we have access to actions or rewards in the retrieval batch. For ablations we also experiment with self-supervised losses. For supervised auxiliary losses, we use policy, value, and reward prediction (Silver et al., 2017; Schrittwieser et al., 2019), and for self-supervised losses, we use a BERT-style masking loss (Devlin et al., 2018b).

11.2.3. Retrieving contextual information.

In this section, we explain how the retrieval process, when provided with relevant contextual information represented by the agent's current state $\boldsymbol{s}_t$, interacts with the summarized information in the retrieval batch to select information $\boldsymbol{u}_t$ to provide to the agent in return.

Retrieval process state parameterization. We parameterize the process that retrieves information from past experience as a structured parametric model with multiple separate memory slots (or sub-units). The state of the retrieval process is a set of n_f memory slots denoted by $\boldsymbol{m}_t = \{\boldsymbol{m}_t^k |\ k \in \{1, \ldots, n_f\}\}$ (indexed by the agent time-step t). Slots are initialized randomly at the beginning of the episode. Each slot independently queries and retrieves relevant information from the pool of data. The slots then update their values independently based on the retrieved information, followed by an integration step during

Algorithm 2 One timestep of a retrieval-augmented agent (R2A).

Input: Current input $\boldsymbol{x}_t$, previous retrieval process state $\boldsymbol{m}_{t-1} = \{\boldsymbol{m}_{t-1,k} |\ k \in \{1, \ldots, n_f\}\}$, dataset of l-step trajectories $\mathcal{B} = \{((\boldsymbol{x}_t^i, \boldsymbol{h}_t^i, \boldsymbol{b}_t^i, a_t^i, r_t^i), \ldots, (\boldsymbol{x}_{t+l}^i, \boldsymbol{h}_{t+l}^i, \boldsymbol{b}_{t+l}^i, a_{t+l}^i, r_{t+l}^i))\}$ for $l \geq 1$ and $1 \leq i \leq n_{\text{traj}}$, where $\boldsymbol{h}$ and $\boldsymbol{b}$ are the outputs of the forward & backward summarizers. We first encode the current input at time-step t using the encoder $\boldsymbol{s}_t = f_\theta^{\text{enc}}(\boldsymbol{x}_t)$.

Step 1: Compute the query. For all $1 \leq\ k \leq n_f$, compute
$\widehat{\boldsymbol{m}}_{t-1}^k = \text{GRU}_{\boldsymbol{\theta}}\left(\boldsymbol{s}_t, \boldsymbol{m}_{t-1}^k\right)$
$\boldsymbol{q}_t^k = f_{\text{query}}(\widehat{\boldsymbol{m}}_{t-1}^k)$

Step 2: Identify the most relevant trajectories. For all $1 \leq k \leq n_f, 1 \leq\ j \leq l$ and $1 \leq\ i \leq n_{\text{traj}}$,
$\boldsymbol{\kappa}_{i,j} = (\boldsymbol{h}_j^i \boldsymbol{W}_{\text{ret}}^{\text{e}})^{\text{T}}$
$\ell_{i,j}^k = \left(\frac{\boldsymbol{q}_t^k \boldsymbol{\kappa}_{i,j}}{\sqrt{d_e}}\right)$
$\alpha_{i,j}^k = \text{softmax}\left(\ell_{i,j}^k\right)$.
Given scores α, the top-k_{traj} trajectories (resp. top-k_{states} states) are selected and denoted by $\mathcal{T}_t^k$ (resp. $\mathcal{S}_t^k$).

Step 3: Retrieve information from the most relevant trajectories and states.
$\alpha_{i,j}^k = \text{softmax}\left(\ell_{i,j}^k\right),\ i \in \mathcal{T}_t^k, j \in \mathcal{S}_t^k$.
$\boldsymbol{g}_t^k = \sum_{i,j} \alpha_{i,j}^k \boldsymbol{v}_{i,j}$ where $\boldsymbol{v}_{i,j} = \boldsymbol{b}_{i,j} \boldsymbol{W}_{\text{ret}}^{\text{v}}$

Step 4: Regularize the retrieved information by using information bottleneck.
$\boldsymbol{z}_t^k \sim p(z|\boldsymbol{g}_t^k)$

Step 5: Update the states of the slots.
Slotwise update using retrieved information:
$\widetilde{\boldsymbol{m}}_t^k \leftarrow \widehat{\boldsymbol{m}}_{t-1}^k + \boldsymbol{z}_t^k \quad \forall k \in \{1, \ldots, n_f\}$
Joint slot update through self-attention:
$\boldsymbol{c}_t^k = \widehat{\boldsymbol{m}}_{t-1}^k \boldsymbol{W}_{\text{SA}}^q \quad \forall k \{1, \ldots, n_f\}$
$\beta_{k,k'} = \text{softmax}_{k'}\left(\frac{\boldsymbol{c}_t^k \boldsymbol{\kappa}_t^{k'}}{\sqrt{d_e}}\right)$ where $\boldsymbol{\kappa}_t^{k'} = (\widetilde{\boldsymbol{m}}_t^{k'} \boldsymbol{W}_{\text{SA}}^e)^{\text{T}} \quad \forall\ k,\ k' \in \{1, \ldots, n_f\}$
$\boldsymbol{m}_t^k \leftarrow \widetilde{\boldsymbol{m}}_t^k + \sum_{k'} \beta_{k,k'} \boldsymbol{v}_{k'}$ where $\boldsymbol{v}_{k'} = \widetilde{\boldsymbol{m}}_t^k \boldsymbol{W}_{\text{SA}}^v \quad \forall k \in \{1, \ldots, n_f\}$

Step 6: Update the agent state using the retrieved information.
$\boldsymbol{d}_t = \boldsymbol{s}_t \boldsymbol{W}_{\text{ag}}^q$
$\boldsymbol{\kappa}^k = (\boldsymbol{z}_t^k \boldsymbol{W}_{\text{ag}}^e)^{\text{T}} \quad \forall k \in \{1, \ldots, n_f\}$
$\gamma_k = \text{softmax}_k\left(\frac{\boldsymbol{d}_t \boldsymbol{\kappa}^k}{\sqrt{d_e}}\right)$
$\boldsymbol{u}_t \leftarrow \sum_k \gamma_k \boldsymbol{v}_k$ where $\boldsymbol{v}_k = \boldsymbol{z}_t^k \boldsymbol{W}_{\text{ag}}^v \quad \forall k \in \{1, \ldots, n_f\}$.
$\widetilde{\boldsymbol{s}}_t \leftarrow \boldsymbol{s}_t + \boldsymbol{u}_t$

which information is shared between slots. Algorithm 2 specifies the six steps of R2A, which we explain in detail below.

Step 1: Query computation. Each slot independently computes its prestate using a GRU on the contextual information from the agent: $\widehat{\boldsymbol{m}}_{t-1}^k = \text{GRU}_{\boldsymbol{\theta}}\left(\boldsymbol{s}_t, \boldsymbol{m}_{t-1}^k\right) \quad \forall k \in \{1, \ldots, n_f\}$. Then, each slot independently computes a *retrieval query* which will be matched against information in the retrieval batch: $\boldsymbol{q}_t^k = f_{\text{query}}(\widehat{\boldsymbol{m}}_{t-1}^k) |\ k \in \{1, \ldots, n_f\}$[1] where $\boldsymbol{q}_t^k$ is the query generated by the k^{th} slot at timestep t.

[1] f_{query} is parameterized as a neural network.

Step 2: Identification of most relevant trajectories and states for each slot. The retrieval mechanism process uses an attention mechanism to match a query produced by the retrieval state associated with each slot m_t^k to keys computed on each time step of each trajectory of the retrieval batch. Formally, for each time step and each trajectory in the buffer, we compute a key $\boldsymbol{\kappa}_{i,j}$ by using a linear projection with matrix $\boldsymbol{W}_{\text{ret}}^{\text{e}}$ on the forward summaries h: $\boldsymbol{\kappa}_{i,j} = (\boldsymbol{h}_j^i \boldsymbol{W}_{\text{ret}}^{\text{e}})^{\text{T}}$. Each query $\boldsymbol{q}_t^k$ is then matched with the set of all keys $\boldsymbol{\kappa}_{i,j}$, forming attention logits[2] $\ell_{i,j}^k = \left(\frac{\boldsymbol{q}_t^k \boldsymbol{\kappa}_{i,j}}{\sqrt{d_e}}\right)$ and corresponding attention weights $\alpha_{i,j}^k = \text{softmax}\left(\ell_{i,j}^k\right)$ for $i \leq n_{\text{traj}}, 0 \leq j \leq T$.

Intuitively, $\alpha_{i,j}^k$ captures the extent to which the j^{th} timestep of the i^{th} trajectory in the buffer will be relevant to memory m_t^k through the query q_t^k. It follows that $\sum_j \alpha_{i,j}^k$ is a measure of how relevant the i^{th} trajectory is as a whole for q_t^k. Following previous work (Ke et al., 2018; Goyal et al., 2019c), matching only on the most relevant trajectories will increase the robustness of the retrieval mechanism. We therefore select, for each query, the set $\mathcal{T}_t^k$ of k_{traj} trajectories with highest aggregated score $\sum_j \alpha_{i,j}^k$. Note that typically the queries corresponding to different slots will select different top-k_{traj} trajectories from the retrieval batch. Following the selection of relevant trajectories, we renormalize the weights α, and use another top-k mechanism, this time to choose the set of most relevant states $\mathcal{S}_t^k$ (i.e. which maximizes $\sum_{i \in \mathcal{T}_t^k} \alpha_{i,j}$).

Step 3: Information retrieval from the most relevant trajectories and states. The next step of the retrieval mechanism consists in computing the renormalized weights α on the subsets $\mathcal{T}_t^k$ and $\mathcal{S}_t^k$ ($\alpha_{i,j}^k = \text{softmax}\left(\ell_{i,j}^k\right)$, $i \in \mathcal{T}_t^k$, $j \in \mathcal{S}_t^k$) and using those weights to compute the final retrieved information. The value retrieved from the buffer for query q_t^k is computed as the α-weighted average of a linear function of the *backward* state summaries: $\boldsymbol{g}_t^k = \sum_{i,j} \alpha_{i,j}^k \boldsymbol{v}_{i,j}$ where $\boldsymbol{v}_{i,j} = \boldsymbol{b}_{i,j} \boldsymbol{W}_{\text{ret}}^{\text{v}}$.

Step 4: Regularization of the retrieved information via an information bottleneck. We regularize the retrieved information $\boldsymbol{g}_t^k$ via the use of an information bottleneck (Tishby et al., 2000a; Alemi et al., 2016b). Intuitively, each query pays a price to exploit information from the retrieval batch. Formally, we parametrize two Gaussian distributions $p(Z|g_t^k)$ (which has access to the retrieved information) and $r(Z|m_{t-1})$ (which only has access to the memory units). We define $\boldsymbol{z}_t^k$ as a single sample from $p(Z|g_t^k)$ via the reparameterization trick to ensure differentiability (Kingma & Welling, 2013; Rezende et al., 2014), and ensure that $\boldsymbol{z}_t^k$ does not contain too much information by adding an additional loss $D_{\text{KL}}(p||r)$ to the overall agent loss. We provide more details in the appendix.

Step 5: Slot update. The representation of each slot is first additively updated as a function of the retrieved information $\widetilde{\boldsymbol{m}}_t^k \leftarrow \widehat{\boldsymbol{m}}_{t-1}^k + \boldsymbol{z}_t^k$. The final update $\boldsymbol{m}_t^k$ consists of an update in which all slots interact through self-attention (as normally done in transformers; see Algorithm 2 for details).

Step 6: Updating the agent state using retrieved information. The primary goal of the retrieval process is to extract information which may be useful for the agent process. Here, we use the retrieved information to change the state of the agent s_t. In the previous step, the retrieved information is used to change the state of the slots. For this step, we use a similar attention mechanism. Here, the query is a function of the state of the agent process $\boldsymbol{d}_t = \boldsymbol{s}_t \boldsymbol{W}_{\text{ag}}^q$, which are matched with the keys $\boldsymbol{\kappa}^k = (\boldsymbol{z}_t^k \boldsymbol{W}_{\text{ag}}^e)^{\text{T}} \quad \forall k \in \{1, \ldots, n_f\}$, as a result of

[2]We drop time indexing from attention-related quantities to simplify notation.

retrieving information, forming attention weights $\gamma_k = \text{softmax}\left(\frac{\boldsymbol{d}_t \boldsymbol{\kappa}^k}{\sqrt{d_e}}\right)$. The values generated as a result of retrieved information by different slots and the attention weights are then used to update the state of the learning agent : $\boldsymbol{u}_t \leftarrow \sum_k \gamma_k \boldsymbol{v}_k$ where $\boldsymbol{v}_k = \boldsymbol{z}_t^k \boldsymbol{W}_{\text{ag}}^v \quad \forall k \in \{1, \ldots, n_f\}$. $\boldsymbol{u}_t$ is the result of the attention over the retrieved information which is then used to change the representation of the agent process : $\widetilde{\boldsymbol{s}}_t \leftarrow \boldsymbol{s}_t + \boldsymbol{u}_t$. We also shape the representation of the action-value function $Q(s_t, \boldsymbol{z}_t^k, a_t)$ by conditioning the value function on the retrieved information $\boldsymbol{z}_t^k$ (again via a similar attention mechanism).

11.3. Experimental Results

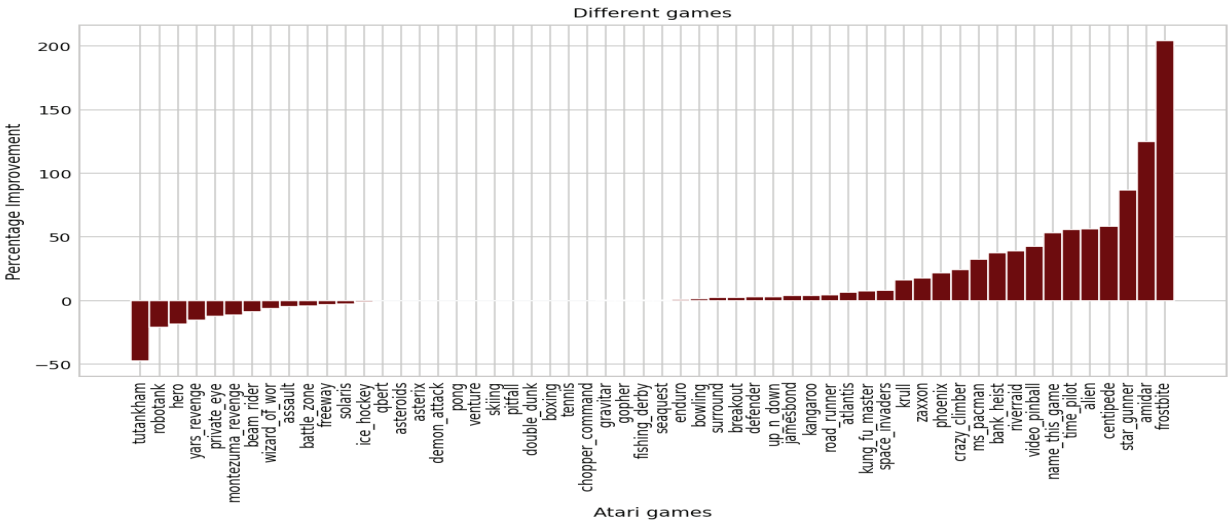
To evaluate R2A, we analyze its performance in three different settings. First, to test whether an agent equipped with retrieval augmentation can achieve better performance (i.e., higher rewards) and scale with more, we test the proposed method on the Atari arcade learning environment (ALE) (Bellemare et al., 2013), a single-task off-policy setting where the retrieval process extracts relevant information from the agent's current replay buffer. We then run a series of ablations of R2A to better understand the roles and effects of its components. Second, to test whether an agent equipped with retrieval augmentation can compensate for lack in capacity when training a single agent on multiple tasks, we evaluate on a multi-task, offline environment that we created, called *gridroboman*. In this environment, a single network is trained on data from all tasks and then, at evaluation time, the retrieval process queries a retrieval dataset containing only data from the task being evaluated. Third, to test if the retrieval augmentation can also benefit when data from the other tasks is present in the retrieval dataset, we evaluate R2A in a multi-task offline version of the BabyAI environment (Chevalier-Boisvert et al., 2018a), and a continuous control manipulation benchmark (Ahmed et al., 2020). Again, a single network is trained on all tasks but now the retrieval process queries a retrieval dataset containing data from all tasks.

In our experiments, the retrieval process selects the top $k_{\text{traj}} = 10$ most relevant trajectories (step 2, section 11.2.3), and then retrieves relevant information from the selected trajectories (step 3, section 11.2.3) using the top $k_{\text{states}} = 10$ most relevant states. To summarize the experiences in the retrieval batch we use a forward and backward GRU with 512 hidden units. To train the representation of these, we use auxiliary losses in the form of action, reward, and value prediction (section 11.2.2).

11.3.1. Atari: Single-task off-policy RL

In this experiment, our goal is to evaluate whether retrieval augmentation improves the performance and sample efficiency of a strong, recurrent baseline agent on a challenging, visually-complex environment—the Atari 2600 videogame suite (Bellemare et al., 2013). We use recurrent replay distributed DQN (R2D2, Kapturowski et al. (2018)) as the baseline agent and compare retrieval-augmented R2D2 (RA-R2D2) to vanilla R2D2. The retrieval dataset is the agent's current replay buffer. The agent process is parameterized as a GRU (Hochreiter & Schmidhuber, 1997b), and the retrieval process is parameterized as the slot-based recurrent architecture described in Section 2, using 8 slots. Retrieval batches consist of 256 trajectories from the retrieval dataset.

Overall, we observe an increase of $11.32 \pm 1.2\%$ in the mean human normalized score relative to the R2D2 baseline over 2 billion environment steps, demonstrating that retrieval augmentation is quite beneficial in Atari and that the agent's own replay buffer is a useful



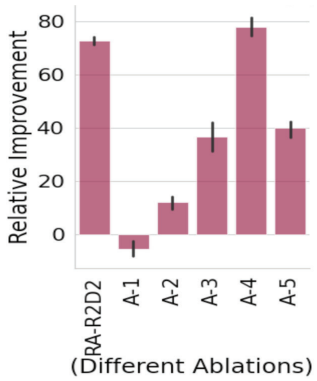
a Per-game relative performance of retrieval-augmented R2D2.

Fig. 2. Relative percentage improvement in mean human normalized score of retrieval-augmented R2D2 vs vanilla R2D2 on different Atari games, measured by human normalized score. We report the average score from 3 seeds per method and per game. Black lines show standard deviations from 3 seeds.

source for retrieval. Raw scores and training curves are presented in Appendix A.??. Figure 11.2a shows the relative improvement of RA-R2D2 versus the R2D2 baseline. Empirically, retrieval augmentation helps the most in the case of Frostbite, which requires temporally extended planning strategies (Lake et al., 2017). For more discussion, refer to Appendix **??**.

11.3.1.1. Ablations and analysis. To understand the benefit of different components of retrieval augmentation, we ablate RA-R2D2 on the 10 Atari games it performs best relative to R2D2. The ablations are as follows, and Figure 11.3a shows the performance of RA-R2D2 and each ablation relative to the R2D2 baseline.

(A-1) Importance of a separate retrieval process. In R2A, the retrieval process and the agent process are parameterized separately, i.e., they have their own internal states. Here we examine what happens when the agent's state is used to query the retrieval batch instead of using the retrieval state $\boldsymbol{m}_t$. To implement this we modify Step 1 of Algorithm 2 to make the query a direct function of the state of the agent, $\boldsymbol{q}_t = f_{\text{query}}(\boldsymbol{s}_t)$. The resulting query is used in the same way as above. The resulting ablated model is akin to the episodic control baseline of Pritzel et al. (2017).



a Relative performance of ablated RA-R2D2.

Fig. 3. Relative percentage improvement of ablated RA-R2D2 versus baseline R2D2 for 5 ablations on 10 Atari games. Black lines show standard deviations from 3 seeds

Conclusion: It is crucial to parameterize the agent process and retrieval process separately, as using the agent state does no better than the baseline. This also shows the benefit of our retrieval formulation as compared to

episodic control. Further, Pritzel et al. (2017) observed that direct access to the replay buffer improves performance given low data, but the advantage disappears with more data. Here our experiments show that the agent equipped with retrieval augmentation achieves better results even in the large data regime (2B time-steps).

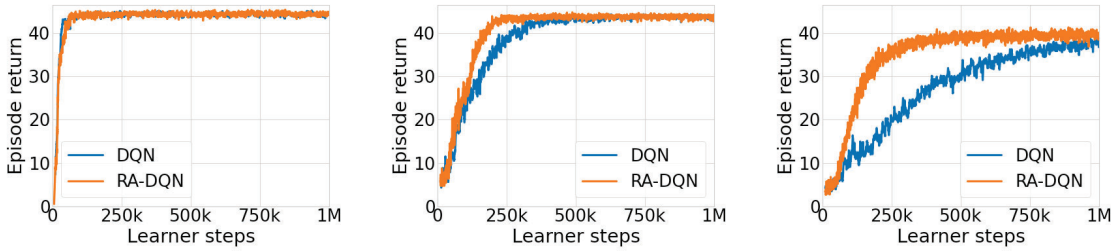
(A-2) Importance of retrieving information. We examine what happens when the retrieval process does not have access to the retrieval dataset and hence no information is retrieved, keeping all else the same. This ablation thus validates that R2A benefits from retrieval, not from an increase in computation and parameters. Specifically, the retrieval process updates the state of the slots using a transformer (i.e., in Step 1 we replace GRU with a transformer), and the updated state of the transformer is used by the agent process to shape the representation of its value function.
Conclusion: R2A retrieves information that is useful to the agent and its performance gains are not simply due to an increase in model capacity and computation.

(A-3) Shorter retrieved trajectories. We decrease the length of the trajectories that are summarized during retrieval pre-processing, thus reducing the amount of past and future information the retrieval process can retrieve. By default, the trajectories in the retrieval dataset are of length 80. To perform this ablation, we decrease the length of the effective context to only include information from 5 timesteps.
Conclusion: Decreasing the length of the context in the retrieval dataset results in worse performance, thus showing the importance of incorporating contextual information using forward and backward summarization.

(A-4, A-5) Importance of auxiliary losses to summarize retrieval batch. Here we study the use of self-supervised BERT style masking losses in addition to using action, reward and value prediction. We use these auxiliary losses on top of the representation learned by the forward and the backward dynamics model. To implement these losses, we randomly mask 15% of the hidden states in a trajectory, and then, using the representation of hidden states at other time-steps, we predict the representation of masked hidden states. In *A-5*, we study using *only* self-supervised BERT style masking losses for summarizing the trajectories.
Conclusion: Ablation A-4 demonstrates that the performance of R2A can further be improved by incorporating BERT style auxiliary losses but that only using BERT style auxiliary losses results in worse performance (but still better than baseline R2D2).

11.3.2. Gridroboman: Multi-task offline RL with a task-specific retrieval dataset

Beyond querying the agent's own experiences, retrieval can provide helpful information from other sources of experiences, including experts or other agents, such as in offline RL where the agent must learn from a fixed dataset of experiences generated by other agents without interacting with the environment during training. A major challenge in offline RL is distributional shift—the mismatch between the distribution of states in the training data and those visited by the agent when acting— which makes it difficult to learn an accurate value function for states and actions rarely seen during training. We hypothesize that the retrieval process can improve performance in the offline setting by retrieving trajectories (including states, actions, and rewards) relevant to the agent's current state, particularly for states and actions that are rare in the offline dataset. We test this hypothesis on a multi-task offline RL setup where a single agent is trained on multiple tasks simultaneously but at evaluation time the retrieval dataset contains only trajectories from the evaluated task.



a Training and testing on 10 tasks.

b Training and testing on 20 tasks.

c Training and testing on 30 tasks.

Fig. 4. Gridroboman: Multi-task offline RL with a task-specific retrieval dataset. Average episode return vs. learner steps for the multi-task gridroboman environment when training and evaluating on 10, 20, and 30 tasks. With fewer tasks (a), the baseline DQN agent (blue) and the retrieval-augmented DQN agent (orange) perform identically; however, when the number of tasks increases (b, c), the retrieval-augmented agent learns much more effectively than the baseline DQN agent. Results are the average of 3 seeds for each method.

Table 1. BabyAI: Multi-task offline RL with a multi-task retrieval dataset. Mean success rate of retrieval-augmented recurrent DQN (RA-RDQN) versus a recurrent DQN (RDQN) baseline on the 40 BabyAI levels, as a function of the amount of training data. RA-RDQN is run twice, once with only the current task being evaluated in the retrieval dataset and once with all tasks in it. Results are the average of 3 random seeds with standard errors.

Method	Success Rate (50K)	Success Rate (200K)
RDQN	32% ± 4%	45% ± 6%
RA-RDQN (single-task retrieval buffer)	48% ± 4%	64% ± 5%
RA-RDQN (multi-task retrieval buffer, without IB)	47% ± 3%	59% ± 6%
RA-RDQN (multi-task retrieval buffer)	55% ± 5%	74% ± 3%

Table 2. CausalWorld: Multi-task offline RL with a multi-task retrieval dataset. Mean success rate of retrieval-augmented behaviour cloning on continuous control task (RA-RDQN) as compared to vanilla behaviour cloning baseline on the 5 tasks. Results are the average of 3 random seeds with standard errors.

Method	Success Rate (50K)
BC (behavior cloning)	61% ± 10%
RA-BC (single-task retrieval buffer)	71% ± 7%
RA-BC (multi-task retrieval buffer)	82% ± 5%

For this experiment, we created a minimalistic grid-world-based robotic manipulation environment (gridroboman) with 30 tasks related to the three objects (red, green, and blue) on the board. Gridroboman is built on the pycolab game engine (Stepleton et al.). The environment is inspired by the challenges of robotic manipulation, and includes tasks such as "go to object X" and "put object X on object Y". Here, we incorporate retrieval augmentation into a vanilla DQN agent as agent-state-recurrence is not needed for this task. Figure 4 shows the results of training retrieval-augmented DQN (RA-DQN, orange) and DQN (blue) on increasing numbers of tasks. With fewer tasks, RA-DQN and DQN perform identically; however, when the number of training tasks increases the retrieval-augmented agent is able to learn much more effectively than the baseline agent. Training on more tasks requires either additional model capacity or the ability to extract information from fewer relevant samples for each task. By directly querying task-relevant experiences in the offline dataset, retrieval augmentation improves sample efficiency. Note that while the retrieval process does afford extra model capacity to the agent directly, ablation A-2 in section 11.3.1.1 shows that the retrieved information is what is crucial to performance, not the increased capacity.

11.3.3. BabyAI: Multi-task offline RL with a multi-task retrieval dataset

Here we evaluate the benefit of retrieval augmentation when data from other tasks is present in the retrieval dataset. Multi-task retrieval data can be either harmful if the retrieved information misguides the agent or beneficial if information from the other tasks is relevant to the current task. Due to the use of attention in the retrieval process, we hypothesize that R2A will be able to retrieve relevant information (and ignore irrelevant information) from other tasks.

To test our hypothesis, we use the BabyAI environment (Chevalier-Boisvert et al., 2018a), a partially observable multi-room grid world in which harder tasks are composed of simpler tasks and are formulated using subsets of a synthetic language. At the start of each episode, the agent is placed in a random room and must navigate to a randomly located goal. Due to the partial observability, we use a recurrent DQN (RDQN) agent as the baseline and compare its performance to a retrieval-augmented RDQN (RA-RDQN) agent.

As is common in this environment, we measure the success rate of each agent, defined as the ratio of tasks the agent was able to accomplish given a fixed number of steps for each task. Table 1 shows the performance of RA-RDQN with a multi-task replay, RA-RDQN with a replay specific to the current task, and the baseline for varying amounts of offline training data (50K trajectories per task versus 200K trajectories per task). As expected from the previous experiment, retrieval augmentation improves performance over the baseline when using a single-task replay. Performance further improves when using a multi-task replay. We believe that this is due to the compositional nature of tasks in BabyAI, where information about a subtask can be more informative than information about the overall task.

Analysis of retrieved information. To understand this effect better, we analyzed the properties of the retrieved information in the multi-task setting in BabyAI. Out of the 40 BabyAI tasks, 15 are compositional—i.e., solving them requires composing information from 2 or more other tasks (e.g., going to the door, fetching a key, etc.). We looked at how often the agent retrieves information from other tasks when solving each task. For the compositional tasks, the agent retrieves information from other tasks 54% of the time, whereas this number is only 21% for the non-compositional tasks. This suggests that the retrieval-augmented agent is retrieving information from other tasks when the current task is compositional and using this information retrieved from relevant sub-tasks to improve its performance.

Information bottleneck ablation. We ran an ablation to validate the use of the information bottleneck (RA-RDQN (multi-task retrieval buffer, without IB)). Table 1 shows the agent performs worse without the information bottleneck (but better than baseline). Such an information bottleneck has been shown to improve generalization (Teh et al., 2017; Goyal et al., 2019a; Galashov et al., 2019).

11.3.4. CausalWorld: Multi-task offline continuous control

We also evaluate the performance of the R2A on a suite of 5 continuous control object manipulation tasks from the CausalWorld benchmark Ahmed et al. (2020). We use the same setup for retrieval pre-processing as in BabyAI but use behaviour cloning (BC) as the underlying algorithm, which has been shown to be a strong baseline for offline RL (Gulcehre et al., 2020).

We compare the performance of BC to retrieval augmented BC (RA-BC). Table 2 shows the performance of the RA-BC with both a multi-task retrieval buffer and a single-task retrieval buffer. Retrieval augmentation improves the performance of BC in both cases.

11.4. Related Work

Episodic control. The idea of allowing deep RL agents to adapt based on past experiences using a non-parametric memory is not new (Blundell et al., 2016; Pritzel et al., 2017; Hansen et al., 2018; Eysenbach et al., 2019; van Hasselt et al., 2019; Fortunato et al., 2019; Zhu et al., 2020). The basic idea is that the agent is equipped with an episodic memory system, which is used to recall past experiences to inform decisions. There are two important differences between R2A and these methods. (1) In these methods, a local action-value function is constructed by using information about the nearest neighbors in the replay buffer, and then the agent makes a decision about which action to execute based on both the local value function as well as the global value function. However, in the proposed work, we employ a parameterized network (the retrieval process), which has access to the information in the replay buffer, and the agent process uses the retrieved information to shape the predictions of its value function in a fully differentiable way (using attention). (2) In these episodic control methods, there is only one process (the agent), which has direct access to the replay buffer. However, in R2A, the agent has indirect access to the replay buffer via the retrieval process.

Retrieval in language models. Retrieval-based methods have recently been developed for question answering, controllable generation, and machine translation (Guu et al., 2020; Lee et al., 2019b; Lewis et al., 2020; Sun et al., 2021; Borgeaud et al., 2021). The general scheme in such methods is to combine a parametric model (like a BERT-style masked language model or a pre-trained seq2seq model) with a non-parametric retrieval system. These methods share some similarities with our proposed model, since they all involve a retrieval component, but focus on different domains.

Model-based RL. Their are different ways to integrate knowledge across past experiences. One of the most common way is by learning a model of the world, and using the predictions from the model to improve the policy and the value function (Sutton, 1991; Silver et al., 2008; Silver, 2009; Allen & Koomen, 1983; Silver et al., 2016; Pascanu et al., 2017; Racanière et al., 2017; Silver et al., 2018; Springenberg et al., 2020; Schrittwieser et al., 2020). To integrate information across different episodes (potentially separated by many time-steps), a model may needed be unrolled for many time-steps leading to compounding errors. In , the agent has direct access to the information in the retrieval dataset, and querying across multiple trajectories (in parallel) in the retrieval dataset potentially separated by hundreds of time-steps.

Structural Inductive Biases. Deep learning have proposed structural inductive biases such as Transformers (Vaswani et al., 2017; Dehghani et al., 2018; Radford et al., 2019; Chen et al., 2020a,c; Dosovitskiy et al., 2020) or slot based recurrent architectures (Battaglia et al., 2016a; Zambaldi et al., 2018; Battaglia et al., 2018; Goyal et al., 2019c; Watters et al., 2019; Goyal et al., 2020; Veerapaneni et al., 2020) where the induced structure has improved generalization, model-size scaling, and longrange dependencies.

Reinforcement Learning with Offline Datasets. Recent work in RL has tried exploiting large datasets collected across many tasks to improve the sample efficiency of RL algorithms (Vecerik et al., 2017; Pertsch et al., 2020; Nair et al., 2020; Siegel et al., 2020). An advantage of such large datasets is that they can be collected cheaply, and can then be

reused for learning many downstream tasks. A general scheme for exploiting information about such task-agnostic datasets is either using them to directly improve the value function, or by extracting a set of skills or options and learning new tasks by recombining them. In our work, we try to use information in the replay buffer by querying and searching for the relevant information across multiple trajectories which otherwise would take many replays through coincidentally relevant information for this to occur.

Separation of concerns. In Hierarchical RL (HRL) (Heess et al., 2016; Frans et al., 2017; Vezhnevets et al., 2017; Florensa et al., 2017; Hausman et al., 2018; Goyal et al., 2019d), there's separation of concerns among different policies, each policy focuses on a different aspect of the task, e.g., giving task relevant information to the high level policy only such that low level policy learns behaviours that are task agnostic. In these methods, the high level policy shapes the behaviour of low level policy by either influencing representations or by influencing rewards.

It is possible to view our work through an analogous lens: wherein the "retrieval process" is the higher level policy (and has access to the all the information in the replay buffer) and is influencing the representation of the agent process that is interacting with the environment. However, there are also notable differences in our work—for instance, the agent process also directly shapes the representation of the retrieval process, which is generally not the case in HRL (e.g., in Vezhnevets et al. (2017) the manager directly influences the worker, but the worker does not directly influence the manager).

Efficient Credit assignment. Learning long term dependencies requires assigning credit to time-steps far back in the past. Common methods for assigning credit in dynamics model like backpropagation through time requires information to be propagated backwards through every single step in the past. This could become computationally expensive when used with very long sequences. Methods which try to get around this problem only back-propagate information through a selected time-steps in the past, realized by a learned mechanism that associates current state with relevant past states (Ke et al., 2018; Wayne et al., 2018; Arjona-Medina et al., 2019; Goyal et al., 2018; Fortunato et al., 2019). Most of these works consider assigning credit to states within the same trajectory, whereas the proposed model , searches for the relevant information in the replay buffer which includes information from other trajectories also.

Memory retrieval as attention turned inward. The proposed work also validates the conjecture put forward in (Logan et al., 2021): *Perceiving and remembering pose the same computational problems: desired information must be extracted from complex multidimensional structures. The conjecture is that the extraction process is selective attention. Turned outward, it retrieves information from perception. Turned inward, it retrieves information from memory.*

11.5. Conclusion.

In this work, we developed R2A, an algorithm that augments an RL agent with a retrieval process. The retrieval process and the agent have separate states and shape the representation and predictions of each other via attention. The goal of the retrieval process is to retrieve useful information from a dataset of experiences to help the agent achieve its objective more efficiently and effectively. We show that R2A improves sample efficiency over R2D2, a strong off-policy agent, and compensates for insufficient capacity when training in multi-task offline RL environments. Multiple ablations show the importance of the different components of R2A, including retrieving information from past experiences and parameterizing the agent

and retrieval process separately instead of giving the agent process direct access to the replay buffer.

Limitations and Future Work. It would be useful to investigate and extend the proposed idea in these different ways: (a) First, investigate training of the retrieval process and the agent process using different objectives as compared to training them in an end-to-end fashion, (b) Second, scaling R2A to more complex multi-agent problems like in Starcraft (Vinyals et al., 2019), where the retrieval process may be shared between different agents. In R2A, we only query a subset of the retrieval dataset, which limits the generality of the method. (c) Third, even more intriguing would be the possibility of learning an abstract model with abstract internal actions, and rewards, rather than learning a model which queries for information from the retrieval dataset, and hence avoiding the need for Monte Carlo tree search common in the state of the art planning methods (Schrittwieser et al., 2020). (d) Fourth, we don't evaluate the R2A in a few-shot learning setting, where we first pre-train an encoding function of the trajectories in the reply dataset and during test time the agent is exposed to a new task, and needs to use the helper process to adapt faster. We aim to formulate these problems and seek answers in the future work.

Chapter 12

Conclusion

The articles presented as a part of this thesis explore how one can use inductive biases in models of system dynamics, policies and raw data for achieving transfer from one task to another task.

(1) **Neural Production Systems** (Chapter 5): This work factorizes the visual world in terms of objects and rules that captures causal interactions between entities. This allows to reuse the knowledge of rules across different object as long as the type of object matches the type of input that rules expect.

(2) **Coordination among Neural Modules Using a Shared Workspace** (Chapter 7): This work explores the use of a communication channel in which functionally specialized components share information through a common, bandwidth-limited communication channel. The proposed method includes a shared workspace through which communication among different specialist modules takes place but due to limits on the communication bandwidth, specialist modules must compete for access. We show that capacity limitations have a rational basis in that (1) they encourage specialization and compositionality and (2) they facilitate the synchronization of otherwise independent specialists.

(3) **RL with Competitive Ensembles of Information Constrained Primitives** (Chapter 9): This work proposes an information-theoretic training objective for learning modular policies in a decentralized fashion. Rather than relying on a centralized, learned meta-controller, the selection of active primitives is implemented through an information-theoretic mechanism. The learned primitives can be flexibly recombined to solve more complex tasks.

(4) **Retrieval Augmented Reinforcement Learning** (Chapter 11): This work augments an agent with the retrieval process which is parameterized as a neural network. There is information asymmetry between the agent and the retrieval process. The agent has access to the information about its current state or current trajectory, whereas the retrieval process has access to all the past experiences (i.e., raw visual observations). The goal of the retrieval process is to provide relevant contextual information that may be relevant to the agent in its current context.

12.1. Projects Looking Forward

The ideas presented in this paper are still in early stages of maturation, with only a few papers beginning the necessary work of ironing out the devil in the detail. Many open questions and paths remain and we highlight a few here.

- One of the big remaining challenges in line with the ideas discussed here remains to jointly learn a large-scale encoder (mapping low-level pixels to high-level causal variables) and a large-scale causal model of these high-level variables. An ideal scenario for this would be that of model-based reinforcement learning, where the causal model would learn the stochastic system dynamics. We have done this (Bengio et al., 2019) only at a small scale (with two causal variables) and using an encoder guaranteed to have singular values 1 for its Jacobian, which avoids a potential collapse of the encoder. In order to avoid collapse, one possibility is to use a contrastive loss at the high level (e.g. as in Deep Infomax (Hjelm et al., 2018) for example).
- Another major challenge is to unify in a single architecture both the declarative knowledge representation (like a structural causal model) and the inference mechanism (maybe implemented with attention and modularity, like with RIMs and their variants). There is a lot of data from human cognition about the consolidation of rule-based behavior into fast habitual skills which could serve as inspiration, e.g., maybe using replay from the hippocampus to train cortical modules to be consistent with the declarative knowledge. Existing work on variational auto-encoders can serve as inspiration as well (with the encoder being the inference machine and the decoder being the causal model, in that case). See Section 3.3.9 for a relevant discussion.
- Most current deep learning models use fixed parameter sharing and fixed regular memory access patterns which are well tailored to modern computing hardware (such as GPUs and TPUs) relying on SIMD parallelism. However, the form of attention-driven computation described in this thesis may require dynamic, irregular and sparse memory access and parameter sharing which does not fit well with GPUs and makes it difficult to parallelize computation across examples of a minibatch. Tackling this may require innovation in neural architectures, low-level programming and hardware design. In terms of model-induced parallelism, the SCOFF approach (Goyal et al., 2020) shows some promise in this direction, where most of the computation is decentralized in each of the experts, and the conscious processing is just the tip of the iceberg in terms of computational cost.
- The way humans plan is very different from the approach currently used in model-based RL (or hybrids such as AlphaZero based on MCTS and value functions). Humans seem to exploit the inductive bias about the sparsity of the causal factor graph and the fact that reasoning sequences in the abstract space can be very short. It means that when humans plan, they do not build trajectories of the full state but instead partial state trajectories where only some aspects (variables) of the state are considered. In addition, they do not unfold future trajectories for every discrete time step but directly learn how to relate temporally distant events, similar to how credit assignment is considered by Ke et al. (2018). It would be interesting to explore these inductive biases in novel planning methods, which might be much more efficient than standard ones. When we plan, we can consider the possibility of novel situations, and if a model misses important aspects of the causal structure, it may not generalize well to

these novel changes, and the plans may radically overestimate or underestimate some novel possibilities.

- Scaling to large number of modules: the brain is probably composed of a very large number of independent modules, whereas most of the current work in modular deep learning deals with much smaller numbers of modules, like 20. It would be interesting to consider new algorithms, architectures that can help in extending to a very large number of modules.
- Macro and Micro Modules: the kinds of modules that are usually considered in the GWT are pretty high-level, e.g., face recognition, gait recognition, object recognition, visual routines, auditory speech perception, auditory object recognition, tactile object recognition. These are *macro-modules* rather than modules that carve up the visual input into single objects i.e *micro-modules.* Most of the work we have done focuses on micro-modules. How should a modular hierarchy be structured which accounts for both these large-scale and fine-scale ways of modularizing knowledge and computation?

12.2. Looking Backward: Relation to Good Old-Fashioned Symbolic AI

How are the approaches proposed here for system 2 computation different from and related to classical symbolic AI (GOFAI [1]) approaches? Let us first review some of the issues with these classical approaches which have motivated solutions building on top of deep learning instead.

(1) We want efficient large scale learning, e.g., brought by variants of stochastic gradient descent and end-to-end learning behind the state-of-the-art in modern deep learning. It is challenging to learn to perform pure symbol manipulation on a large scale because of the discreteness of the operations.

(2) We want semantic grounding of the higher-level concepts in terms of the lower-level observations and lower-level actions (which is done by system-1 computation in the brain). This is important because some of the understanding of the world (maybe a large part) is not represented at the conscious system-2 level, and is completely lacking when representing knowledge purely in symbolic terms.

(3) We want distributed representations of higher-level concepts (to benefit from the remarkable advantages this brings in terms of generalization): whereas pure symbolic representations put every symbol at the same distance of every other symbol, distributed representations represent symbols through a vector of attributes, with related symbols having overlapping representations.

(4) We want efficient search and inference. A computational bottleneck of GOFAI is search, which in probabilistic terms is equivalent to the problem of inference and is generally intractable and needs to be approximated. Variational auto-encoders have shown how this computational cost can be amortized by training the inference machinery. This the only way we currently know how to do this in a general enough way, and it is consistent with cognitive neuroscience of transfer from system 2 to system 1 of habitual skills.

(5) We want to handle uncertainty, which most machine learning approaches are meant to handle.

[1] Good Old-Fashioned Artificial Intelligence

We already have these capabilities with the current deep learning toolbox. What is missing is to integrate to that toolbox what will be required to achieve the kind of systematic generalization and decomposition of knowledge into small exchangeable pieces which is typically associated with GOFAI. We believe that it will not be sufficient to slap GOFAI methods on top of representations produced by neural nets, for the above reasons, but especially points 1, 3 and 4.

References

Alessandro Achille and Stefano Soatto. Information dropout: learning optimal representations through noise. *CoRR*, abs/1611.01353, 2016. URL `http://arxiv.org/abs/1611.01353`.

S. Ahmad and S. Omohundro. Equilateral triangles: A challenge for connectionist vision. 2009.

Ossama Ahmed, Frederik Träuble, Anirudh Goyal, Alexander Neitz, Manuel Wüthrich, Yoshua Bengio, Bernhard Schölkopf, and Stefan Bauer. Causalworld: A robotic manipulation benchmark for causal structure and transfer learning. *arXiv preprint arXiv:2010.04296*, 2020.

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Ekin Akyürek, Afra Feyza Akyürek, and Jacob Andreas. Learning to recombine and resample data for compositional generalization. *arXiv preprint arXiv:2010.03706*, 2020.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022.

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, and Kevin Murphy. Deep variational information bottleneck. *CoRR*, abs/1612.00410, 2016a. URL `http://arxiv.org/abs/1612.00410`.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. In *International Conference on Learning Representations*, 2016b.

James F Allen and Johannes A Koomen. Planning using a temporal world model. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pp. 741–747, 1983.

Pablo Alvarez and Larry R Squire. Memory consolidation and the medial temporal lobe: a simple network model. *Proceedings of the national academy of sciences*, 91(15):7041–7045, 1994.

John R Anderson. Skill acquisition: Compilation of weak-method problem situations. *Psychological review*, 94(2):192, 1987.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 39–48, 2016.

Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 166–175. JMLR. org, 2017.

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. RUDDER: Return decomposition for delayed rewards. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Bernard J Baars. *A cognitive theory of consciousness*. Cambridge University Press, 1993.

Bernard J Baars. In the theatre of consciousness. global workspace theory, a rigorous scientific theory of consciousness. *Journal of Consciousness Studies*, 4(4):292–309, 1997.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *AAAI*, pp. 1726–1734, 2017.

Alan Baddeley. Working memory. *Science*, 255(5044):556–559, 1992.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv e-prints*, art. arXiv:1409.0473, Sep 2014.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. Systematic generalization: what is required and can it be learned? *arXiv preprint arXiv:1811.12889*, 2018.

Dzmitry Bahdanau, Harm de Vries, Timothy J O'Donnell, Shikhar Murty, Philippe Beaudoin, Yoshua Bengio, and Aaron Courville. Closure: Assessing systematic generalization of clevr models. *arXiv preprint arXiv:1912.05783*, 2019.

Renee Baillargeon, Elizabeth S Spelke, and Stanley Wasserman. Object permanence in five-month-old infants. *Cognition*, 20(3):191–208, 1985.

Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula, 2019.

Christopher Baldassano, Janice Chen, Asieh Zadbood, Jonathan W Pillow, Uri Hasson, and Kenneth A Norman. Discovering event structure in continuous narrative perception and memory. *Neuron*, 95(3):709–721, 2017.

Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

John A Bargh. Automatic and conscious processing of social information. In *American Psychological Association convention, 1982, Washington, DC, US; Portions of the research discussed in this chapter were presented at the aforementioned conference, and at the 1982 meetings of the Society for Experimental Social Psychology in Nashville, Indiana.* Lawrence Erlbaum Associates Publishers, 1984.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems*, pp. 4502–4510, 2016a.

Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110(45): 18327–18332, 2013.

Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *CoRR*, abs/1612.00222, 2016b. URL `http://arxiv.org/abs/1612.00222`.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv*

preprint arXiv:1806.01261, 2018.

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B Tenenbaum, William T Freeman, and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. *arXiv:1811.10597, ICLR'2019*, 2018.

Jonathan Baxter. A model of inductive bias learning. *Journal of artificial intelligence research*, 12:149–198, 2000.

Diane M Beck and Sabine Kastner. Top-down and bottom-up mechanisms in biasing competition in the human brain. *Vision research*, 49(10):1154–1165, 2009.

Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 456–473, 2018.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Aya Ben-Yakov and Richard N Henson. The hippocampal film editor: sensitivity and specificity to event boundaries in continuous experience. *Journal of Neuroscience*, 38(47): 10057–10068, 2018.

Emmanuel Bengio, Valentin Thomas, Joelle Pineau, Doina Precup, and Yoshua Bengio. Independently controllable features. *CoRR*, abs/1703.07718, 2017. URL `http://arxiv.org/abs/1703.07718`.

Samy Bengio and Yoshua Bengio. Taking on the curse of dimensionality in joint distributions using neural networks. *IEEE Transactions on Neural Networks*, 11(3):550–557, 2000.

Y Bengio, P Simard, and P Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994a.

Yoshua Bengio. Evolving culture versus local minima. In *Growing Adaptive Machines*, pp. 109–138. Springer, 2014.

Yoshua Bengio. The consciousness prior. *arXiv preprint arXiv:1709.08568*, 2017.

Yoshua Bengio, Samy Bengio, and Jocelyn Cloutier. *Learning a synaptic learning rule*. Citeseer, 1990.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994b.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. In *Advances in Neural Information Processing Systems*, pp. 932–938, 2001.

Yoshua Bengio, Tristan Deleu, Nasim Rahaman, Rosemary Ke, Sébastien Lachapelle, Olexa Bilaniuk, Anirudh Goyal, and Christopher Pal. A meta-transfer objective for learning to disentangle causal mechanisms. *arXiv:1901.10912*, 2019.

James O Berger and José M Bernardo. On the development of the reference prior method. *Bayesian statistics*, 4(4):35–60, 1992.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Dorthe Berntsen, Søren Risløv Staugaard, and Louise Maria Torp Sørensen. Why am i remembering this now? predicting the occurrence of involuntary (spontaneous) episodic memories. *Journal of Experimental Psychology: General*, 142(2):426, 2013.

Marcel Binz and Eric Schulz. Using cognitive psychology to understand gpt-3. *arXiv preprint arXiv:2206.14576*, 2022.

Chris M Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *arXiv preprint arXiv:1606.04460*, 2016.

Kyra Bonasia, Melanie J Sekeres, Asaf Gilboa, Cheryl L Grady, Gordon Winocur, and Morris Moscovitch. Prior knowledge modulates the neural substrates of encoding and retrieving naturalistic events at short and long delays. *Neurobiology of Learning and Memory*, 153: 26–39, 2018.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. *arXiv preprint arXiv:2112.04426*, 2021.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*, pp. 2206–2240. PMLR, 2022.

Leon Bottou. Learning representations using causal invariance. *ICLR Keynote Talk*, 2019.

Léon Bottou and Patrick Gallinari. A framework for the cooperation of learning algorithms. In *Advances in neural information processing systems*, pp. 781–788, 1991.

Matthew M Botvinick, Todd S Braver, Deanna M Barch, Cameron S Carter, and Jonathan D Cohen. Conflict monitoring and cognitive control. *Psychological review*, 108(3):624, 2001a.

Matthew M Botvinick, Todd S Braver, CS Carter, DM Barch, and JD Cohen. Evaluating the demand for control: Anterior cingulate cortex and crosstalk monitoring. *Psychological Review*, 108:624–652, 2001b.

Valentino Braitenberg. *Vehicles: Experiments in synthetic psychology*. MIT press, 1986.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Rodney A Brooks. Intelligence without representation. *Artificial intelligence*, 47(1-3):139–159, 1991.

Philippe Brouillard, Perouz Taslakian, Alexandre Lacoste, Sébastien Lachapelle, and Alexandre Drouin. Typing assumptions improve identification in causal discovery. In *Conference on Causal Learning and Reasoning*, pp. 162–177. PMLR, 2022.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in \beta-vae. *arXiv preprint arXiv:1804.03599*, 2018.

Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

Mikhail S Burtsev and Grigory V Sapunov. Memory transformer. *arXiv preprint arXiv:2006.11527*, 2020.

Timothy J Buschman and Earl K Miller. Top-down versus bottom-up control of attention in the prefrontal and posterior parietal cortices. *science*, 315(5820):1860–1862, 2007.

Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, et al. Scaling data-driven robotics with reward sketching and batch reinforcement learning. *arXiv preprint arXiv:1909.12200*, 2019.

Richard A Carlson and Don E Dulany. Conscious attention and abstraction in concept learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 11(1):45, 1985.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Pratik Chaudhari and Stefano Soatto. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–10. IEEE, 2018.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International Conference on Machine Learning*, pp. 1691–1703. PMLR, 2020a.

Ricky TQ Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, pp. 2610–2620, 2018.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020b.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020c.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020d.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: First steps towards grounded language learning with a human in the loop. *arXiv preprint arXiv:1810.08272*, 2018a.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`, 2018b.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*, 2016.

Elisa Ciaramelli, Cheryl L Grady, and Morris Moscovitch. Top-down and bottom-up attention to memory: a hypothesis (atom) on the role of the posterior parietal cortex in memory retrieval. *Neuropsychologia*, 46(7):1828–1851, 2008.

Paul Cisek. Cortical mechanisms of action selection: the affordance competition hypothesis. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1485):1585–1599, 2007.

Jeff Clune. Ai-gas: Ai-generating algorithms, an alternate paradigm for producing general artificial intelligence. *arXiv preprint arXiv:1905.10985*, 2019.

Jonathan D Cohen, Matthew Botvinick, and Cameron S Carter. Anterior cingulate and prefrontal cortex: who's in control? *Nature neuroscience*, 3(5):421–423, 2000.

Leon Cohen. Rate of apparent change of a necker cube as a function of prior stimulation. *The American Journal of Psychology*, 72(3):327–344, 1959.

Michael Colagrosso and Michael C Mozer. Theories of access consciousness. *Advances in neural information processing systems*, 17, 2004.

Michael D. Colagrosso and Michael C Mozer. Theories of access consciousness. In L. K. Saul, Y. Weiss, and L. Bottou (eds.), *Advances in Neural Information Processing Systems 17*, pp. 289–296. MIT Press, 2005. URL `http://papers.nips.cc/paper/2715-theories-of-access-consciousness.pdf`.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pp. 160–167, 2008.

Nelson Cowan. An embedded-processes model of working memory. 1999.

Mary Kathryn Cowles and Bradley P Carlin. Markov chain monte carlo convergence diagnostics: a comparative review. *Journal of the American Statistical Association*, 91(434): 883–904, 1996.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Christian Daniel, Gerhard Neumann, and Jan Peters. Hierarchical relative entropy policy search. In *Artificial Intelligence and Statistics*, pp. 273–281, 2012.

Ishita Dasgupta, Jane Wang, Silvia Chiappa, Jovana Mitrovic, Pedro Ortega, David Raposo, Edward Hughes, Peter Battaglia, Matthew Botvinick, and Zeb Kurth-Nelson. Causal reasoning from meta-reinforcement learning. *arXiv preprint arXiv:1901.08162*, 2019.

Peter Dayan and Geoffrey E Hinton. Feudal reinforcement learning. In *Advances in neural information processing systems*, pp. 271–278, 1993.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

S. Dehaene, H. Lau, and S. Kouider. What is consciousness, and could machines have it? *Science*, 358(6362):486–492, 2017.

Stanislas Dehaene. *How We Learn: Why Brains Learn Better Than Any Machine... for Now*. Penguin, 2020.

Stanislas Dehaene and Jean-Pierre Changeux. Experimental and theoretical approaches to conscious processing. *Neuron*, 70(2):200–227, 2011.

Stanislas Dehaene, Michel Kerszberg, and Jean-Pierre Changeux. A neuronal model of a global workspace in effortful cognitive tasks. *Proceedings of the national Academy of Sciences*, 95(24):14529–14534, 1998.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

Tristan Deleu, António Góis, Chris Chinenye Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. Bayesian structure learning with generative flow networks. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018a. URL `http://arxiv.org/abs/1810.04805`.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018b.

Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

David Ding, Felix Hill, Adam Santoro, and Matt Botvinick. Object-based attention for spatio-temporal reasoning: Outperforming neuro-symbolic models with flexible distributed architectures. *arXiv preprint arXiv:2012.08508*, 2020.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Hubert L Dreyfus. From socrates to expert systems: The limits and dangers of calculative rationality. 1985.

Yilun Du, Kevin Smith, Tomer Ulman, Joshua Tenenbaum, and Jiajun Wu. Unsupervised discovery of 3d physical objects from video. *arXiv preprint arXiv:2007.12348*, 2020.

Gabriel Dulac-Arnold, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. An empirical investigation of the challenges of real-world reinforcement learning. *arXiv preprint arXiv:2003.11881*, 2020.

Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

Frederick Eberhardt, Clark Glymour, and Richard Scheines. On the number of experiments sufficient and in the worst case necessary to identify all causal relations among n variables. *arXiv preprint arXiv:1207.1389*, 2012.

Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. Genesis: Generative scene inference and sampling with object-centric latent representations. *arXiv preprint arXiv:1907.13052*, 2019.

Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 201–208. JMLR Workshop and Conference Proceedings, 2010.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

SM Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Koray Kavukcuoglu, and Geoffrey E Hinton. Attend, infer, repeat: Fast scene understanding with generative models. *arXiv preprint arXiv:1603.08575*, 2016.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

Benjamin Eysenbach, Ruslan Salakhutdinov, and Sergey Levine. Search on the replay buffer: Bridging planning and reinforcement learning. *arXiv preprint arXiv:1906.05253*, 2019.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2021a.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *CoRR*, abs/2101.03961, 2021b. URL `https://arxiv.org/abs/2101.03961`.

Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

Marc Finzi, Samuel Stanton, Pavel Izmailov, and Andrew Gordon Wilson. Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data. In *International Conference on Machine Learning*, pp. 3165–3176. PMLR, 2020.

Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

Jerry A Fodor. *The modularity of mind*. MIT press, 1983.

Jerry A Fodor and Zenon W Pylyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1-2):3–71, 1988.

Vincent Fortuin. Priors in bayesian deep learning: A review. *International Statistical Review*, 2022.

Meire Fortunato, Melissa Tan, Ryan Faulkner, Steven Hansen, Adrià Puigdomènech Badia, Gavin Buttimore, Charlie Deck, Joel Z Leibo, and Charles Blundell. Generalization of reinforcement learners with working and episodic memory. *arXiv preprint arXiv:1910.13406*, 2019.

K. Frans, J. Ho, X. Chen, P. Abbeel, and J. Schulman. Meta Learning Shared Hierarchies. *arXiv e-prints*, October 2017.

Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. *arXiv preprint arXiv:1710.09767*, 2017.

Brendan J Frey. Extending factor graphs so as to unify directed and undirected graphical models. *arXiv preprint arXiv:1212.2486*, 2012.

Brendan J Frey, Ralf Koetter, and Nemanja Petrovic. Very loopy belief propagation for unwrapping phase images. *Advances in Neural Information Processing Systems*, 14, 2001.

Alexandre Galashov, Siddhant M Jayakumar, Leonard Hasenclever, Dhruva Tirumala, Jonathan Schwarz, Guillaume Desjardins, Wojciech M Czarnecki, Yee Whye Teh, Razvan Pascanu, and Nicolas Heess. Information asymmetry in KL-regularized RL. *arXiv preprint arXiv:1905.01240*, 2019.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

Andrew Gelman. Bayesian model-building by pure thought: some principles and examples. *Statistica Sinica*, pp. 215–232, 1996.

James J Gibson. The theory of affordances. *Hilldale, USA*, 1(2), 1977.

Gerd Gigerenzer and Daniel G Goldstein. Reasoning the fast and frugal way: models of bounded rationality. *Psychological review*, 103(4):650, 1996.

Walter R Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice*. CRC press, 1995.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1263–1272. JMLR. org, 2017.

Rohit Girdhar and Deva Ramanan. CATER: A diagnostic dataset for compositional actions and temporal reasoning. *CoRR*, abs/1910.04744, 2019. URL `http://arxiv.org/abs/1910.04744`.

Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. URL `http://arxiv.org/abs/1504.08083`.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Robert D Gordon and David E Irwin. What's in an object file? evidence from priming studies. *Perception & Psychophysics*, 58(8):1260–1277, 1996.

Anirudh Goyal and Yoshua Bengio. Inductive biases for deep learning of higher-level cognition. *arXiv preprint arXiv:2011.15091*, 2020.

Anirudh Goyal, Philemon Brakel, William Fedus, Soumye Singhal, Timothy Lillicrap, Sergey Levine, Hugo Larochelle, and Yoshua Bengio. Recall traces: Backtracking models for efficient reinforcement learning. *arXiv preprint arXiv:1804.00379*, 2018.

Anirudh Goyal, Riashat Islam, Daniel Strouse, Zafarali Ahmed, Matthew Botvinick, Hugo Larochelle, Sergey Levine, and Yoshua Bengio. Infobot: Transfer and exploration via the information bottleneck. *arXiv preprint arXiv:1901.10902*, 2019a.

Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms, 2019b. URL `arXiv:1909.10893`.

Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms, 2019c.

Anirudh Goyal, Shagun Sodhani, Jonathan Binas, Xue Bin Peng, Sergey Levine, and Yoshua Bengio. Reinforcement learning with competitive ensembles of information-constrained primitives. *arXiv preprint arXiv:1906.10667*, 2019d.

Anirudh Goyal, Alex Lamb, Phanideep Gampa, Philippe Beaudoin, Sergey Levine, Charles Blundell, Yoshua Bengio, and Michael Mozer. Object files and schemata: Factorizing declarative and procedural knowledge in dynamical systems. *arXiv preprint arXiv:2006.16225*, 2020.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014a.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014b. URL `http://arxiv.org/abs/1410.5401`.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471, 2016.

Klaus Greff, Antti Rasmus, Mathias Berglund, Tele Hotloo Hao, Jürgen Schmidhuber, and Harri Valpola. Tagger: Deep unsupervised perceptual grouping. *arXiv preprint arXiv:1606.06724*, 2016.

Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object

representation learning with iterative variational inference. In *International Conference on Machine Learning*, pp. 2424–2433, 2019.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Oliver Groth, Fabian Fuchs, Ingmar Posner, and Andrea Vedaldi. Shapestacks: Learning vision-based physical intuition for generalised object stacking. *CoRR*, abs/1804.08018, 2018. URL `http://arxiv.org/abs/1804.08018`.

Caglar Gulcehre, Sarath Chandar, Kyunghyun Cho, and Yoshua Bengio. Dynamic neural turing machine with soft and hard addressing schemes. *arXiv preprint arXiv:1607.00036*, 2016.

Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gomez Colmenarejo, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, et al. Rl unplugged: A suite of benchmarks for offline reinforcement learning. *arXiv preprint arXiv:2006.13888*, 2020.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*, 2020.

Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. *arXiv preprint arXiv:1804.02808*, 2018.

Johan Hstad. *Computational Limitations of Small-Depth Circuits*. Mit Press, 1987.

Steven Hansen, Pablo Sprechmann, Alexander Pritzel, André Barreto, and Charles Blundell. Fast deep reinforcement learning using online adjustments from the past. *arXiv preprint arXiv:1810.08163*, 2018.

Demis Hassabis, Dharshan Kumaran, and Eleanor A Maguire. Using imagination to understand the neural basis of episodic memory. *Journal of neuroscience*, 27(52):14365–14374, 2007.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Zhen He, Jian Li, Daxue Liu, Hangen He, and David Barber. Tracking by animation: Unsupervised learning of multi-object attentive trackers. *CoRR*, abs/1809.03137, 2018. URL `http://arxiv.org/abs/1809.03137`.

Nicolas Heess, Greg Wayne, David Silver, Timothy Lillicrap, Yuval Tassa, and Tom Erez. Learning continuous control policies by stochastic value gradients, 2015.

Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

Nicolas Heess, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, Ali Eslami, Martin Riedmiller, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF*

International Conference on Computer Vision, pp. 8340–8349, 2021.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, et al. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*, 2017.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

Salah Hihi and Yoshua Bengio. Hierarchical recurrent neural networks for long-term dependencies. *Advances in neural information processing systems*, 8, 1995.

Felix Hill, Andrew Lampinen, Rosalia Schneider, Stephen Clark, Matthew Botvinick, James L McClelland, and Adam Santoro. Environmental drivers of systematicity and generalization in a situated agent. *arXiv preprint arXiv:1910.00571*, 2019.

Geoffrey Hinton. How to represent part-whole hierarchies in a neural network. *arXiv preprint arXiv:2102.12627*, 2021.

Geoffrey E Hinton. Distributed representations. 1984.

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen [in german] diploma thesis. *TU Münich*, 1991a.

Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 91(1), 1991b.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9 (8):1735–1780, November 1997a. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `http://dx.doi.org/10.1162/neco.1997.9.8.1735`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997b.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Peter C Humphreys, Arthur Guez, Olivier Tieleman, Laurent Sifre, Théophane Weber, and Timothy Lillicrap. Large-scale retrieval for reinforcement learning. *arXiv preprint arXiv:2206.05314*, 2022.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, Geoffrey E Hinton, et al. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention. *arXiv preprint arXiv:2103.03206*, 2021.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pp. 991–1002. PMLR, 2022.

Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Harold Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 186(1007): 453–461, 1946.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2989–2998, 2017.

Daniel Kahneman. *Thinking, fast and slow*. Macmillan, 2011.

Daniel Kahneman, Anne Treisman, and Brian J Gibbs. The reviewing of object files: Object-specific integration of information. *Cognitive psychology*, 24(2):175–219, 1992.

Steven Kapturowski, Georg Ostrovski, John Quan, Remi Munos, and Will Dabney. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2018.

Nan Rosemary Ke, Anirudh Goyal ALIAS PARTH GOYAL, Olexa Bilaniuk, Jonathan Binas, Michael C Mozer, Chris Pal, and Yoshua Bengio. Sparse attentive backtracking: Temporal credit assignment through reminding. In *Advances in neural information processing systems*, pp. 7640–7651, 2018.

Nan Rosemary Ke, Olexa Bilaniuk, Anirudh Goyal, Stefan Bauer, Hugo Larochelle, Chris Pal, and Yoshua Bengio. Learning neural causal models from unknown interventions. *arXiv preprint arXiv:1910.01075*, 2019.

Nan Rosemary Ke, Aniket Didolkar, Sarthak Mittal, Anirudh Goyal, Guillaume Lajoie, Stefan Bauer, Danilo Rezende, Yoshua Bengio, Michael Mozer, and Christopher Pal. Systematic evaluation of causal discovery in visual model based reinforcement learning. *arXiv preprint arXiv:2107.00848*, 2021.

Giancarlo Kerg, Bhargav Kanuparthi, Anirudh Goyal, Kyle Goyette, Yoshua Bengio, and Guillaume Lajoie. Untangling tradeoffs between recurrence and self-attention in neural networks. *arXiv preprint arXiv:2006.09471*, 2020.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

Ronald A Kinchla and Jeremy M Wolfe. The order of visual processing:"top-down,"˜"bottom-up," or "middle-out". *Perception & psychophysics*, 25(3):225–231, 1979.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. *arXiv preprint arXiv:1802.04687*, 2018.

Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. *arXiv preprint arXiv:1911.12247*, 2019.

Christof Koch. The quest for consciousness a neurobiological approach. 2004.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on*

Machine Learning, pp. 5637–5664. PMLR, 2021.

Stanley Kok and Pedro Domingos. Learning the structure of markov logic networks. In *Proceedings of the 22nd international conference on Machine learning*, pp. 441–448, 2005.

Janet L Kolodner. An introduction to case-based reasoning. *Artificial intelligence review*, 6 (1):3–34, 1992.

Wouter Kool and Matthew Botvinick. Mental labour. *Nature human behaviour*, 2(12):899–908, 2018.

Adam Kosiorek, Hyunjik Kim, Yee Whye Teh, and Ingmar Posner. Sequential attend, infer, repeat: Generative modelling of moving objects. *Advances in Neural Information Processing Systems*, 31:8606–8616, 2018.

Eliza Kosoy, David M Chan, Adrian Liu, Jasmine Collins, Bryanna Kaufmann, Sandy Han Huang, Jessica B Hamrick, John Canny, Nan Rosemary Ke, and Alison Gopnik. Towards understanding how machines can learn causal overhypotheses. *arXiv preprint arXiv:2206.08353*, 2022.

Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. A clockwork rnn. In *International Conference on Machine Learning*, pp. 1863–1871. PMLR, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pp. 5815–5826. PMLR, 2021.

Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

Jan Kukačka, Vladimir Golkov, and Daniel Cremers. Regularization for deep learning: A taxonomy. *arXiv preprint arXiv:1710.10686*, 2017.

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pp. 3675–3683, 2016.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

John E Laird, Paul S Rosenbloom, and Allen Newell. Chunking in soar: The anatomy of a general learning mechanism. *Machine learning*, 1(1):11–46, 1986.

Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, pp. 2879–2888, 2018.

Brenden M Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. *arXiv preprint arXiv:1711.00350*, 2017.

Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40, 2017.

Alex Lamb, Di He, Anirudh Goyal, Guolin Ke, Chien-Feng Liao, Mirco Ravanelli, and Yoshua Bengio. Transformers with competitive ensembles of independent mechanisms, 2021. URL `https://openreview.net/forum?id=1TIrbngpW0x`.

Vito Latora and Massimo Marchiori. Efficient behavior of small-world networks. *Physical review letters*, 87(19):198701, 2001.

Nicolas Le Roux, Nicolas Heess, Jamie Shotton, and John Winn. Learning a generative model of images by factoring appearance and shape. *Neural Computation*, 23(3):593–650, 2011.

David B Leake. Case-based reasoning: experiences, lessons, and future directions. 1996.

Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pp. 3744–3753, 2019a.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. *arXiv preprint arXiv:1906.00300*, 2019b.

Joel Z Leibo, Edward Hughes, Marc Lanctot, and Thore Graepel. Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research. *arXiv preprint arXiv:1903.00742*, 2019.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *arXiv preprint arXiv:2005.11401*, 2020.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992.

Dianbo Liu, Alex M Lamb, Kenji Kawaguchi, Anirudh Goyal ALIAS PARTH GOYAL, Chen Sun, Michael C Mozer, and Yoshua Bengio. Discrete-valued neural communication. *Advances in Neural Information Processing Systems*, 34:2109–2121, 2021.

Libin Liu and Jessica Hodgins. Learning to schedule control fragments for physics-based characters using deep q-learning. *ACM Transactions on Graphics*, 36(3), 2017.

Siqi Liu, Guy Lever, Josh Merel, Saran Tunyasuvunakool, Nicolas Heess, and Thore Graepel. Emergent coordination through competition. *arXiv preprint arXiv:1902.07151*, 2019.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pp. 4114–4124, 2019.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *arXiv preprint arXiv:2006.15055*, 2020.

Gordon D Logan, Gregory E Cox, Jeffrey Annis, and Dakota RB Lindsey. The episodic flanker effect: Memory retrieval as attention turned inward. *Psychological Review*, 128(3): 397, 2021.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

Marsha C Lovett and John R Anderson. Thinking as a production system. *The Cambridge handbook of thinking and reasoning*, pp. 401–429, 2005.

Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi. Unified-io: A unified model for vision, language, and multi-modal tasks. *arXiv preprint arXiv:2206.08916*, 2022.

Marlos C Machado, Marc G Bellemare, and Michael Bowling. A laplacian framework for option discovery in reinforcement learning. *arXiv preprint arXiv:1703.00956*, 2017.

Kanika Madan, Nan Rosemary Ke, Anirudh Goyal, Bernhard Schölkopf, and Yoshua Bengio. Meta attention networks: Meta-learning attention to modulate information between recurrent independent mechanisms. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Lc28QAB4ypz.

Gary F Marcus. Rethinking eliminative connectionism. *Cognitive psychology*, 37(3):243–282, 1998.

Gary F Marcus. *The algebraic mind: Integrating connectionism and cognitive science*. MIT press, 2003.

Gary F Marcus. *The algebraic mind: Integrating connectionism and cognitive science*. MIT press, 2019.

Charles C Margossian. A review of automatic differentiation and its efficient implementation. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(4):e1305, 2019.

Bogdan Mazoure, Remi Tachet des Combes, Thang Doan, Philip Bachman, and R Devon Hjelm. Deep reinforcement and infomax learning. *arXiv preprint arXiv:2006.07217*, 2020.

James L McClelland. The programmable blackboard model of reading. *Parallel distributed processing: Explorations in the microstructure of cognition*, 2:122–169, 1986.

James L McClelland and David E Rumelhart. An interactive activation model of context effects in letter perception: I. an account of basic findings. *Psychological review*, 88(5):375, 1981.

James L McClelland, David E Rumelhart, PDP Research Group, et al. *Parallel Distributed Processing, Volume 2: Explorations in the Microstructure of Cognition: Psychological and Biological Models*, volume 2. MIT press, 1987.

Michael McCloskey. Intuitive physics. *Scientific american*, 248(4):122–131, 1983.

Stephanie McMains and Sabine Kastner. Interactions of top-down and bottom-up mechanisms in human visual cortex. *Journal of Neuroscience*, 31(2):587–597, 2011.

Josh Merel, Arun Ahuja, Vu Pham, Saran Tunyasuvunakool, Siqi Liu, Dhruva Tirumala, Nicolas Heess, and Greg Wayne. Hierarchical visuomotor control of humanoids. In *International Conference on Learning Representations*, 2019a. URL https://openreview.net/forum?id=BJfYvo09Y7.

Josh Merel, Leonard Hasenclever, Alexandre Galashov, Arun Ahuja, Vu Pham, Greg Wayne, Yee Whye Teh, and Nicolas Heess. Neural probabilistic motor primitives for humanoid control. In *International Conference on Learning Representations*, 2019b. URL https://openreview.net/forum?id=BJl6TjRcY7.

Anton Milan, Laura Leal-Taixé, Ian D. Reid, Stefan Roth, and Konrad Schindler. MOT16: A benchmark for multi-object tracking. *CoRR*, abs/1603.00831, 2016. URL http://arxiv.org/abs/1603.00831.

Earl K Miller, Jonathan D Cohen, et al. An integrative theory of prefrontal cortex function. *Annual review of neuroscience*, 24(1):167–202, 2001.

Marvin Minsky. *Society of mind*. Simon and Schuster, 1988.

Tom M Mitchell et al. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, 45(37):870–877, 1997.

Sarthak Mittal, Alex Lamb, Anirudh Goyal, Vikram Voleti, Murray Shanahan, Guillaume Lajoie, Michael Mozer, and Yoshua Bengio. Learning to combine top-down and bottom-up signals in recurrent neural networks with attention over modules. In *International Conference on Machine Learning*, pp. 6972–6986. PMLR, 2020a.

Sarthak Mittal, Alex Lamb, Anirudh Goyal, Vikram Voleti, Murray Shanahan, Guillaume Lajoie, Michael Mozer, and Yoshua Bengio. Learning to combine top-down and bottom-up signals in recurrent neural networks with attention over modules. *arXiv preprint arXiv:2006.16981*, 2020b.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015a.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015b.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937, 2016.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, pp. 2924–2932, 2014.

Michael C Mozer and Debra Miller. Parsing the stream of time: The value of event-based segmentation in a complex real-world control problem. *International School on Neural Networks, Initiated by IIASS and EMFCSC*, pp. 370–388, 1997.

Michael C Mozer, Michael Colagrosso, and David Huber. A rational analysis of cognitive control in a speeded discrimination task. *Advances in neural information processing systems*, 14, 2001.

Kevin Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference: An empirical study. *arXiv preprint arXiv:1301.6725*, 2013.

Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. AWAC: Accelerating online reinforcement learning with offline datasets. 2020.

James Newman, Bernard J Baars, and Sung-Bae Cho. A neural global workspace model for conscious attention. *Neural Networks*, 10(7):1195–1206, 1997.

Nicholaus S Noles, Brian J Scholl, and Stephen R Mitroff. The persistence of object file representations. *Perception & Psychophysics*, 67(2):324–334, 2005.

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint arXiv:1810.12282*, 2018.

Giambattista Parascandolo, Niki Kilbertus, Mateo Rojas-Carulla, and Bernhard Schölkopf. Learning independent causal mechanisms. *arXiv preprint arXiv:1712.00961*, 2017.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer, 2018.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR, abs/1211.5063*, 2(417):1, 2012.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. pp. 1310–1318, 2013a.

Razvan Pascanu, Guido Montufar, and Yoshua Bengio. On the number of inference regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098, ICLR'2014*, 2013b.

Razvan Pascanu, Yujia Li, Oriol Vinyals, Nicolas Heess, Lars Buesing, Sebastien Racanière, David Reichert, Théophane Weber, Daan Wierstra, and Peter Battaglia. Learning model-based planning from scratch. *arXiv preprint arXiv:1707.06170*, 2017.

Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, New York, NY, 2nd edition, 2009.

Xue Bin Peng, Glen Berseth, Kangkang Yin, and Michiel Van De Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Trans. Graph.*, 36(4):41:1–41:13, July 2017. ISSN 0730-0301. doi: 10.1145/3072959.3073602. URL `http://doi.acm.org/10.1145/3072959.3073602`.

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4):143:1–143:14, July 2018. ISSN 0730-0301. doi: 10.1145/3197517.3201311. URL `http://doi.acm.org/10.1145/3197517.3201311`.

Karl Pertsch, Youngwoon Lee, and Joseph J Lim. Accelerating reinforcement learning with learned skill priors. *arXiv preprint arXiv:2010.11944*, 2020.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of Causal Inference - Foundations and Learning Algorithms*. MIT Press, Cambridge, MA, USA, 2017a. ISBN 978-0-262-03731-0.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. MIT press, 2017b.

Giovanni Pezzulo and Paul Cisek. Navigating the affordance landscape: feedback control as a process model of behavior and cognition. *Trends in cognitive sciences*, 20(6):414–424, 2016.

Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018.

Lorien Y Pratt. Discriminability-based transfer between neural networks. In *Advances in neural information processing systems*, pp. 204–211, 1993.

Lorien Y Pratt, Jack Mostow, Candace A Kamm, and Ace A Kamm. Direct transfer of learned information among neural networks. In *Aaai*, volume 91, pp. 584–589, 1991.

Rémi Le Priol, Reza Babanezhad Harikandeh, Yoshua Bengio, and Simon Lacoste-Julien. An analysis of the adaptation speed of causal models. *arXiv preprint arXiv:2005.09136*, 2020.

Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adria Puigdomenech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *International Conference on Machine Learning*, pp. 2827–2836. PMLR, 2017.

Haozhi Qi, Xiaolong Wang, Deepak Pathak, Yi Ma, and Jitendra Malik. Learning long-term visual dynamics with region proposal interaction networks. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=_X_4Akcd8Re`.

Sébastien Racanière, Théophane Weber, David P Reichert, Lars Buesing, Arthur Guez, Danilo Rezende, Adria Puigdomenech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 5694–5705, 2017.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

Gabriel A Radvansky and Jeffrey M Zacks. Event boundaries in memory and cognition. *Current opinion in behavioral sciences*, 17:133–140, 2017.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Nasim Rahaman, Anirudh Goyal, Muhammad Waleed Gondal, Manuel Wuthrich, Stefan Bauer, Yash Sharma, Yoshua Bengio, and Bernhard Schölkopf. S2rms: Spatially structured recurrent modules. *arXiv preprint arXiv:2007.06533*, 2020.

William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

David Raposo, Adam Santoro, David Barrett, Razvan Pascanu, Timothy Lillicrap, and Peter Battaglia. Discovering objects and their relations from entangled scene representations. *arXiv preprint arXiv:1702.05068*, 2017.

Karsten Rauss and Gilles Pourtois. What is bottom-up and what is top-down in predictive coding? *Frontiers in psychology*, 4:276, 2013.

Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. Equivariance through parameter-sharing. In *International conference on machine learning*, pp. 2892–2901. PMLR, 2017.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

Peter Redgrave, Manuel Rodriguez, Yoland Smith, Maria C Rodriguez-Oroz, Stephane Lehericy, Hagai Bergman, Yves Agid, Mahlon R DeLong, and Jose A Obeso. Goal-directed and habitual control in the basal ganglia: implications for parkinson's disease. *Nature Reviews Neuroscience*, 11(11):760–772, 2010.

Scott Reed and Nando De Freitas. Neural programmer-interpreters. *arXiv preprint arXiv:1511.06279*, 2015.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of The 31st International Conference on Machine Learning*, pp. 1278–1286, 2014.

Lauren L Richmond and Jeffrey M Zacks. Constructing experience: Event models from perception to action. *Trends in cognitive sciences*, 21(12):962–980, 2017.

Martin Riedmiller. Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In *European conference on machine learning*, pp. 317–328. Springer, 2005.

Mark B Ring. Child: A first step towards continual learning. In *Learning to learn*, pp. 261–292. Springer, 1998.

Eric Ronco, Henrik Gollee, and Peter J Gawthrop. Modular neural networks and self-decomposition. *Technical Report CSC-96012*, 1997.

Clemens Rosenbaum, Tim Klinger, and Matthew Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *arXiv preprint arXiv:1711.01239*, 2017.

Clemens Rosenbaum, Ignacio Cases, Matthew Riemer, and Tim Klinger. Routing networks and the challenges of modular and compositional computation. *arXiv preprint arXiv:1904.12774*, 2019a.

Clemens Rosenbaum, Ignacio Cases, Matthew Riemer, and Tim Klinger. Routing networks and the challenges of modular and compositional computation. *arXiv preprint arXiv:1904.12774*, 2019b.

Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M Lake. A benchmark for systematic generalization in grounded language understanding. *Advances in neural information processing systems*, 33:19861–19872, 2020.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986a.

David E Rumelhart, Paul Smolensky, James L McClelland, and G Hinton. Sequential thought processes in pdp models. *Parallel distributed processing: explorations in the microstructures of cognition*, 2:3–57, 1986b.

Peter Norvig Russell. Artificial intelligence: A modern approach by stuart. *Russell and Peter Norvig contributing writers, Ernest Davis...[et al.]*, 2010.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.

Adam Santoro, David Raposo, David GT Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *arXiv preprint arXiv:1706.01427*, 2017.

Adam Santoro, Ryan Faulkner, David Raposo, Jack W. Rae, Mike Chrzanowski, Theophane Weber, Daan Wierstra, Oriol Vinyals, Razvan Pascanu, and Timothy P. Lillicrap. Relational recurrent neural networks. *CoRR*, abs/1806.01822, 2018. URL `http://arxiv.org/abs/1806.01822`.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1703.01161*, 2017.

Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pp. 9323–9332. PMLR, 2021.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

Steffen Schneider, Evgenia Rusak, Luisa Eck, Oliver Bringmann, Wieland Brendel, and Matthias Bethge. Improving robustness against common corruptions by covariate shift adaptation. *Advances in Neural Information Processing Systems*, 33:11539–11551, 2020.

Walter Schneider, Sue T Dumais, and Richard M Shiffrin. Automatic/control processing and attention. Technical report, Illinois Univ Champaign Human Attention Research Lab, 1982.

B. Schölkopf. Artificial intelligence: Learning to see and act. *Nature*, 518(7540):486–487, 2015.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *arXiv preprint arXiv:1911.08265*, 2019.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.

Murray Shanahan. Consciousness, emotion, and imagination. *Approaches to Machine Consciousness*, pp. 26, 2005.

Murray Shanahan. A cognitive architecture that combines internal simulation with a global workspace. *Consciousness and cognition*, 15(2):433–449, 2006.

Murray Shanahan. *Embodiment and the inner life: Cognition and Consciousness in the Space of Possible Minds*. Oxford University Press, USA, 2010.

Murray Shanahan. The brain's connective core and its role in animal cognition. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1603):2704–2714, 2012.

Murray Shanahan and Bernard Baars. Applying global workspace theory to the frame problem. *Cognition*, 98(2):157–176, 2005.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Amitai Shenhav, Sebastian Musslick, Falk Lieder, Wouter Kool, Thomas L Griffiths, Jonathan D Cohen, Matthew M Botvinick, et al. Toward a rational and mechanistic account of mental effort. *Annual review of neuroscience*, 40(1):99–124, 2017.

Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020.

David Silver. Reinforcement learning and simulation-based search in computer go. 2009.

David Silver, Richard S Sutton, and Martin Müller. Sample-based learning and search with permanent and transient memories. In *Proceedings of the 25th international conference on Machine learning*, pp. 968–975, 2008.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

David Silver, Hado Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, and Thomas Degris. The predictron: End-to-end learning and planning. In *International Conference on Machine Learning*, pp. 3191–3199. PMLR, 2017.

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

Samuel L Smith and Quoc V Le. A bayesian perspective on generalization and stochastic gradient descent. *arXiv preprint arXiv:1710.06451*, 2017.

Paul Smolensky. On the proper treatment of connectionism. *Behavioral and brain sciences*, 11(1):1–23, 1988.

John F Sowa. Semantic networks. 1987.

Elizabeth S Spelke, Karen Breinlinger, Janet Macomber, and Kristen Jacobson. Origins of knowledge. *Psychological review*, 99(4):605, 1992.

Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. *Causation, prediction, and search*. MIT press, 2000.

Jost Tobias Springenberg, Nicolas Heess, Daniel Mankowitz, Josh Merel, Arunkumar Byravan, Abbas Abdolmaleki, Jackie Kay, Jonas Degrave, Julian Schrittwieser, Yuval Tassa, et al. Local search for policy iteration in continuous control. *arXiv preprint arXiv:2010.05545*, 2020.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Tom Stepleton, Olivier Delalleau, Georg Ostrovski, and Sam Wenke. URL `https://github.com/deepmind/pycolab`.

Andrea Stocco, Christian Lebiere, and John R Anderson. Conditional routing of information to the cortex: A model of the basal ganglia's role in cognitive coordination. *Psychological review*, 117(2):541, 2010.

Thomas Suddendorf and Michael C Corballis. The evolution of foresight: What is mental time travel, and is it unique to humans? *Behavioral and brain sciences*, 30(3):299–313, 2007.

Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 28*, pp. 2440–2448. Curran Associates, Inc., 2015. URL `http://papers.nips.cc/paper/5846-end-to-end-memory-networks.pdf`.

Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, et al. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *arXiv preprint arXiv:2107.02137*, 2021.

Richard Sutton. The bitter lesson. *Incomplete Ideas (blog)*, 13:12, 2019.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press, 1998.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pp. 1057–1063, Cambridge, MA, USA, 1999a. MIT Press. URL `http://dl.acm.org/citation.cfm?id=3009657.3009806`.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112 (1-2):181–211, 1999b.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112 (1-2):181–211, 1999c.

Andrea Tacchetti, H Francis Song, Pedro AM Mediano, Vinicius Zambaldi, Neil C Rabinowitz, Thore Graepel, Matthew Botvinick, and Peter W Battaglia. Relational forward models for multi-agent learning. *arXiv preprint arXiv:1809.11044*, 2018.

Alex H Taylor, Gavin R Hunt, Felipe S Medina, and Russell D Gray. Do new caledonian crows solve physical problems through causal reasoning? *Proceedings of the Royal Society*

B: Biological Sciences, 276(1655):247–254, 2009.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4496–4506, 2017.

Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

Valentin Thomas, Jules Pondard, Emmanuel Bengio, Marc Sarfati, Philippe Beaudoin, Marie-Jean Meurs, Joelle Pineau, Doina Precup, and Yoshua Bengio. Independently controllable features. *arXiv preprint arXiv:1708.01289*, 2017.

Dhruva Tirumala, Hyeonwoo Noh, Alexandre Galashov, Leonard Hasenclever, Arun Ahuja, Greg Wayne, Razvan Pascanu, Yee Whye Teh, and Nicolas Heess. Exploiting hierarchy for learning and transfer in kl-regularized rl. *arXiv preprint arXiv:1903.07438*, 2019.

Dhruva Tirumala, Alexandre Galashov, Hyeonwoo Noh, Leonard Hasenclever, Razvan Pascanu, Jonathan Schwarz, Guillaume Desjardins, Wojciech Marian Czarnecki, Arun Ahuja, Yee Whye Teh, et al. Behavior priors for efficient reinforcement learning. *arXiv preprint arXiv:2010.14274*, 2020.

Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000a.

Naftali Tishby, Fernando C. N. Pereira, and William Bialek. The information bottleneck method. *CoRR*, physics/0004057, 2000b. URL `http://arxiv.org/abs/physics/0004057`.

David S Touretzky and Geoffrey E Hinton. A distributed connectionist production system. *Cognitive science*, 12(3):423–466, 1988.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL `http://www.jmlr.org/papers/v9/vandermaaten08a.html`.

Hado P van Hasselt, Matteo Hessel, and John Aslanides. When to use parametric models in reinforcement learning? *Advances in Neural Information Processing Systems*, 32: 14322–14333, 2019.

Sjoerd Van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *arXiv preprint arXiv:1802.10353*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2017.

Rishi Veerapaneni, John D. Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua B. Tenenbaum, and Sergey Levine. Entity abstraction in visual model-based reinforcement learning. *CoRR*, abs/1910.12827, 2019. URL `http://arxiv.org/abs/1910.12827`.

Rishi Veerapaneni, John D Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua Tenenbaum, and Sergey Levine. Entity abstraction in visual model-based reinforcement learning. In *Conference on Robot Learning*, pp. 1439–1456. PMLR, 2020.

Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pp. 3540–3549. PMLR, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Nicholas Watters, Daniel Zoran, Theophane Weber, Peter Battaglia, Razvan Pascanu, and Andrea Tacchetti. Visual interaction networks: Learning a physics simulator from video. In *Advances in neural information processing systems*, pp. 4539–4547, 2017.

Nicholas Watters, Loic Matthey, Matko Bosnjak, Christopher P Burgess, and Alexander Lerchner. Cobra: Data-efficient model-based rl through unsupervised object discovery and curiosity-driven exploration. *arXiv preprint arXiv:1905.09275*, 2019.

Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

Greg Wayne, Chia-Chun Hung, David Amos, Mehdi Mirza, Arun Ahuja, Agnieszka Grabska-Barwinska, Jack Rae, Piotr Mirowski, Joel Z Leibo, Adam Santoro, et al. Unsupervised predictive memory in a goal-directed agent. *arXiv preprint arXiv:1803.10760*, 2018.

Marissa A. Weis, Kashyap Chitta, Yash Sharma, Wieland Brendel, Matthias Bethge, Andreas Geiger, and Alexander S. Ecker. Unmasking the inductive biases of unsupervised object representations for video sequences, 2020.

Gellért Weisz, Philip Amortila, and Csaba Szepesvári. Exponential lower bounds for planning in mdps with linearly-realizable optimal action-value functions. In *Algorithmic Learning Theory*, pp. 1237–1264. PMLR, 2021.

Max Welling. Do we still need models or just more data and compute? *University of Amsterdam, April*, 20, 2019.

Paul Werbos. Beyond regression:" new tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*, 1974.

Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

Ronald J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3-4):229–256, 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL `https://doi.org/10.1007/BF00992696`.

Ronald J. Williams and David Zipser. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Computation*, 1(2):270–280, 06 1989. ISSN 0899-7667. doi: 10.1162/neco.1989.1.2.270. URL `https://doi.org/10.1162/neco.1989.1.2.270`.

Terry Winograd. Understanding natural language. *Cognitive psychology*, 3(1):1–191, 1972.

David H Wolpert, William G Macready, et al. No free lunch theorems for search. Technical report, Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

Bohan Wu, Jayesh K Gupta, and Mykel J Kochenderfer. Model primitive hierarchical lifelong reinforcement learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 34–42. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pp. 2048–2057, 2015.

Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017.

Polina Zablotskaia, Edoardo A Dominici, Leonid Sigal, and Andreas M Lehrmann. Unsupervised video decomposition using spatio-temporal iterative inference. *arXiv preprint arXiv:2006.14727*, 2020.

Jeffrey M Zacks, Nicole K Speer, Khena M Swallow, Todd S Braver, and Jeremy R Reynolds. Event perception: a mind-brain perspective. *Psychological bulletin*, 133(2):273, 2007.

Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Deep reinforcement learning with relational inductive biases. In *International Conference on Learning Representations*, 2018.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Mingde Zhao, Zhen Liu, Sitao Luan, Shuyuan Zhang, Doina Precup, and Yoshua Bengio. A consciousness-inspired planning agent for model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 34:1569–1581, 2021.

Guangxiang Zhu, Zichuan Lin, Guangwen Yang, and Chongjie Zhang. Episodic reinforcement learning with associative memory. 2020.