Université de Montréal

Inductive Biases for Efficient Information Transfer in Artificial Networks

par

Giancarlo Kerg

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

September 1, 2022

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée

Inductive Biases for Efficient Information Transfer in Artificial Networks

présentée par

Giancarlo Kerg

a été évaluée par un jury composé des personnes suivantes :

Irina Rish

(président-rapporteur)

Guillaume Lajoie

(directeur de recherche)

Yoshua Bengio

(codirecteur)

Pierre-Luc Bacon

(membre du jury)

Jonathan D. Cohen

(examinateur externe)

(représentant du doyen de la FESP)

Résumé

Malgré des progrès remarquables dans une grande variété de sujets, les réseaux de neurones éprouvent toujours des difficultés à exécuter certaines tâches simples pour lesquelles les humains excellent. Comme indiqué dans des travaux récents, nous émettons l'hypothèse que l'écart qualitatif entre l'apprentissage en profondeur actuel et l'intelligence humaine est le résultat de biais inductifs essentiels manquants. En d'autres termes, en identifiant certains de ces biais inductifs essentiels, nous améliorerons le transfert d'informations dans les réseaux artificiels, ainsi que certaines de leurs limitations actuelles les plus importantes sur un grand ensemble de tâches. Les limites sur lesquelles nous nous concentrerons dans cette thèse sont la généralisation systématique hors distribution et la capacité d'apprendre sur des échelles de temps extrêmement longues. Dans le premier article, nous nous concentrerons sur l'extension des réseaux de neurones récurrents (RNN) à contraintes spectrales et proposerons une nouvelle structure de connectivité basée sur la décomposition de Schur, en conservant les avantages de stabilité et la vitesse d'entraînement des RNN orthogonaux tout en améliorant l'expressivité pour les calculs complexes à court terme par des dynamiques transientes. Cela sert de première étape pour atténuer le problème du "exploding vanishing gradient" (EVGP). Dans le deuxième article, nous nous concentrerons sur les RNN avec une mémoire externe et un mécanisme d'auto-attention comme un moyen alternatif de résoudre le problème du EVGP. Ici, la contribution principale sera une analyse formelle sur la stabilité asymptotique du gradient, et nous identifierons la pertinence d'événements comme un ingrédient clé pour mettre à l'échelle les systèmes d'attention. Nous exploitons ensuite ces résultats théoriques pour fournir un nouveau mécanisme de dépistage de la pertinence, qui permet de concentrer l'auto-attention ainsi que de la mettre à l'échelle, tout en maintenant une bonne propagation du gradient sur de longues séquences. Enfin, dans le troisième article, nous distillons un ensemble minimal de biais inductifs pour les tâches cognitives purement relationnelles et identifions que la séparation des informations relationnelles des entrées sensorielles est un ingrédient inductif clé pour la généralisation OoD sur des entrées invisibles. Nous discutons en outre des extensions aux relations non-vues ainsi que des entrées avec des signaux parasites.

Mots clés: réseaux de neurones, apprentissage automatique, apprentissage de représentations profondes, apprentissage de représentations, réseaux de neurones récurrents, généralisation hors distribution, généralisation systématique, biais inductifs.

Abstract

Despite remarkable advances in a wide variety of subjects, neural networks are still struggling on simple tasks humans excel at. As outlined in recent work, we hypothesize that the qualitative gap between current deep learning and human-level artificial intelligence is the result of missing essential inductive biases. In other words, by identifying some of these key inductive biases, we will improve information transfer in artificial networks, as well as improve on some of their current most important limitations on a wide range of tasks. The limitations we will focus on in this thesis are out-of-distribution systematic generalization and the ability to learn over extremely long-time scales. In the First Article, we will focus on extending spectrally constrained Recurrent Neural Networks (RNNs), and propose a novel connectivity structure based on the Schur decomposition, retaining the stability advantages and training speed of orthogonal RNNs while enhancing expressivity for short-term complex computations via transient dynamics. This serves as a first step in mitigating the Exploding Vanishing Gradient Problem (EVGP). In the Second Article, we will focus on memory augmented self-attention RNNs as an alternative way to tackling the Exploding Vanishing Gradient Problem (EVGP). Here the main contribution will be a formal analysis on asymptotic gradient stability, and we will identify event relevancy as a key ingredient to scale attention systems. We then leverage these theoretical results to provide a novel relevancy screening mechanism, which makes self-attention sparse and scalable, while maintaining good gradient propagation over long sequences. Finally, in the Third Article, we distill a minimal set of inductive biases for purely relational cognitive tasks, and identify that separating relational information from sensory input is a key inductive ingredient for OoD generalization on unseen inputs. We further discuss extensions to unseen relations as well as settings with spurious features.

Keywords: neural networks, machine learning, deep learning, representation learning, recurrent neural networks, out-of-distribution generalization, systematic generalization, inductive biases.

Contents

List of tables

List of figures

List of acronyms and abbreviations

AI	Artificial Intelligence
ML	Machine Learning
RL	Reinforcement Learning
SM	Supplemental Material
SGD	Stocastic Gradient Descent
OoD	out-of-distribution
GPU	Graphics Processing Unit
NLP	Natural Language Processing
MLP	Multilayer Perceptron
MHA	Multi-Head Attention
e.g.	*exempli gratia* [for instance]
i.e.	*id est* [that is]
PTB	Penn Tree Bank Corpus task (as defined in Marcus et al. (1993))
BPC	Bit per Character
sMNIST	sequential MNIST task
pMNIST, psMNIST	permuted sequential MNIST task (as defined in Le et al. (2015))
LSTM	Long Short-Term Memory network (as defined in Hochreiter and Schmidhuber (1997))
RNN	Recurrent Neural Network
expRNN	architecture introduced in Lezcano-Casado and Martínez-Rubio (2019)
GRU	Gated Recurrent Unit (as defined in Cho et al. (2014b))
GORU	Gated Orthogonal Recurrent Unit (as defined in Jing et al. (2019))
EVGP	Exploding Vanishing Gradient Problem
FMC	Fisher Memory Curve
SAB	Sparse Attentive Backtracking (as defined in Ke et al. (2018))
ESBN	Emergent Symbol Binding Network (as defined in Webb et al. (2021))
TCN	Temporal context normalization (as defined in Webb et al. (2020))
Predi-Net	architecture introduced in Shanahan et al. (2019)
RN	Relation Net (as defined in Santoro et al. (2017))

Acknowledgment

First, I would like to express my deep gratitude for having had the opportunity to do my PhD at Mila, which I experienced as a very enriching, stimulating and vibrant research environment, and which has offered me a lot of freedom and flexibility to pursue my interests and ideas.

To my mother and my father, Anna and Carlo, to whom this thesis is dedicated, for their unconditional love, support and for raising me to value education. To Stephanie Santeliz, for her invaluable support during my most difficult times in my PhD, and teaching me about the importance of self-love. To Ben Kuechen, for teaching me the power of the body-mind connection and how to stay grounded in stressful situations. To Julia Zasada, for her ongoing loving and emotional support.

To Yoshua Bengio, for taking me under his wings and giving me the opportunity to spend these last 5 years at Mila. Being advised by one of the most brilliant AI researchers of our time is a real privilege. His contagious passion, commitment to understanding intelligence and wealth of research ideas have been an incredible source of inspiration for me.

To Guillaume Lajoie, for his unwavering guidance, encouragement, positivity and very consistent support throughout my PhD, not only on high-level research ideas but also on low-level technical details. None of this thesis would have existed without him.

To Sarthak Mittal, for having been one of the most important people supporting me in my PhD, for constantly challenging my ideas and for having had some of the most insightful research discussions during my PhD. He tremendously supported me in producing working code. I just recall how many times he has saved my life!

To Kyle Goyette, for countless study sessions at La Récréation and collaborating on pretty much all homework assignments for my coursework as well as for very stimulating research discussions. He has been a great help and motivation for me to improve my coding skills.

To Bhargav Kanuparthi, for having been an amazing roommate and for many interesting discussions around AI research as well as personal development.

To Olexa Bilaniuk, for his ongoing and invaluable technical support. Whenever I was stuck with a technical problem, he was always helping me out, for which I am truly grateful.

To Victor Geadah, for having led some very interesting research projects in computational neuroscience, which I could be part of.

To Blake Richards, for his enthusiasm, invaluable support, interesting discussions in my last article, as well as for pointing me towards the ESBN paper.

To Céline Bégin, for always being very supportive and helpful in any administrative matters. She was always very quick to reply, and helped me out a great deal!

I would love to thank all remaining collaborators David Rolnick, Stefan Horoi, Guy Wolf, Tristan Deleu, David Kanaa, Leo Feng, Pierre-Luc Bacon, Anirudh Goyal, Maximilian Puelma Touzel, Gauthier Gidel, Eugene Vorontsov, Devansh Arpit, Rosemary Ke, Ioannis Mitliagkas, Stanislaw Jastrzebski, Oliver Astrand, Huan Wang, Caiming Xiong, Richard Socher, Kyunghyun Cho, Krzysztof Geras. It has been a privilege to collaborate with so many brilliant minds.

Further, I would love to thank Matt Jones, Mike Mozer, Johnathan Cohen, Simon Segert, Taylor Webb, Doina Precup, Veronica Chelu, Guillaume Rabusseau, Alison Gopnik, Tim Cooijmans, Sarath Chandar, Johnathan Binas, César Laurent, Tianyu Li, Naz Sepah, Yassine Yaakoubi, Rim Assouel and Chen Sun, for interesting discussions and feedback on some of the presented articles.

Finally, I would love to extend my appreciation (in no particular order) to Samuel Lavoie, Dora Jambor, Rémi Le Priol, Chin-Wei Huang, Ahmed Touati, Benjamin Scellier, Akram Erraqabi, Salem Lahlou, Léonard Boussioux, Mohammed Pezeshki, Nasim Rahaman, Florian Bordes, Sébastien Lachapelle, Amine Natik, Ankesh Anand, Christos Tsirigotis, David Venuto, Harm de Vries, Paul Bertin, Thomas George, Tom Bosc, Simon Guiroy, Vikram Voleti, Xavier Bouthillier, Kyle Kastner, Lucas and Massimo Caccia.

I would love to thank UNIQUE for the PhD Excellence Scholarship as well as Calcul Québec and Compute Canada for providing me with computing ressources.

Chapter 1

Introduction

In recent years, neural networks have produced some remarkable advances in a wide variety of subjects such as computer vision, natural language processing as well as computational biology. Besides the ongoing efforts to improve model architectures, these advances are due to the increasing amount of available data and substantial computational resources (Sejnowski, 2018).

Meanwhile, there is a substantial qualitative gap between current deep learning and human cognitive abilities. This includes: out-of-distribution systematic generalization, transfer learning with low sample complexity, ability to reason over extremely long time-scales, and the ability to discover a hierarchy of robust representations that can be flexibly recombined and repurposed in a variety of tasks. Building on the hypothesis that human intelligence is the result of a few principles or key inductive biases (Goyal and Bengio, 2020), we argue that studying the inductive biases humans use, as well as the inductive biases providing substantial improvement on the mentioned limitations, will bring us a step closer human level AI.

Intuitively, inductive biases are a set of preferences for the learning algorithm to prioritize solutions with certain properties over others. Common examples are for instance: the use of convolutions to prioritize solutions with space equivariance, or the use of recurrence to prioritize solutions with equivariance over time. More generally, we would like to prioritize solutions that generalize well on the kind of data that is relevant to the human perspective on the world around us. This is motivated by the *no-free-lunch theorem* (Wolpert and Macready, 1995; Baxter, 2000), stating that there is no completely general-purpose learning algorithm generalizing better on all distributions, and that some inductive biases or preferences over the set of all functions is necessary for generalization.

1.1. Contributions

In this thesis we will look more closely at (1) specific sequential tasks that require learning over extremely long time scales and (2) special cases for out-of-distribution generalization, where current traditional deep learning methods fail. We will then propose and study a set of inductive biases to circumvent these limitations enhancing information propagation. Here, we will look at information propagation from multiple angles: for (1), we will look at information propagation by examining gradient stability/propagation, while for (2), we will consider out-of-distribution generalization in various purely relational tasks as a way to measure how well relational information between sensory input is being encoded and propagated. The goal of this thesis is then to study a new set of inductive biases and their implications on information propagation in settings (1) and (2).

In order to tackle (1), we will look at recurrent neural networks as it is an important and extremely popular family of neural network models designed to process sequential data of variable length. One of the main challenges in training recurrent neural networks on longer input sequences is the so-called exploding and vanishing gradient problem (EVGP), which makes it difficult to capture long-term dependencies. More specifically, we will see in subsection §2.2.3 that one cannot robustly store bits of information in an RNN's hidden states such that small perturbations in the input do not affect the asymptotic behaviour of the hidden state trajectory in the state space, without having to face the vanishing gradient problem. One of the first proposed strategies to mitigating this problem has been to design new RNN architecture with additional gates such as the LSTM or GRU (Hochreiter and Schmidhuber, 1997; Cho et al., 2014a), while later approaches have explored spectrally constraining the connectivity matrix in vanilla RNNs either at initialization (Le et al., 2015; Henaff et al., 2016) or throughout the entire training period (Arjovsky et al., 2016; Wisdom et al., 2016; Hyland and Rätsch, 2017; Mhammedi et al., 2017; Jing et al., 2017; Vorontsov et al., 2017; Lezcano-Casado and Martínez-Rubio, 2019). Finally there is substantial empirical evidence that the use of attention systems in memory augmented recurrent neural networks can help to bypass this problem (Vaswani et al., 2017; Ke et al., 2018; Graves et al., 2014), but it comes at the expense of increased computational complexity.

In this thesis, we will first focus on spectrally constrained approaches throughout the entire training period. We are proposing an extension of orthogonal RNNs offering flexible trade-off between long-term memorization and short complex computation. More specifically, we are leveraging the Schur decomposition of the connectivity matrix to increase the expressivity of orthogonal RNNs by including a non-normal connectivity structure (and hence allowing eigenbases to be non-orthogonal) while retaining control of the eigenvalues' norms. We will see that these non-normal dynamics allow for transient expansion and compression, affording additional expressivity to better encode complex inputs while retaining control of gradient propagation. This will be the topic of the **First Article** entitled *"Non-normal Recurrent Neural Network (nnRNN): learning long time dependencies while improving expressivity with transient dynamics"* (Kerg et al., 2019), in Chapter 3.

Further, we will look at self-attention systems, and provide a formal analysis of gradient propagation for a whole family of memory augmented recurrent neural networks. We derive key quantities governing gradient propagation and leverage those findings to propose a novel relevancy screening mechanism offering a trade-off between computational complexity and good long-term gradient propagation. More specifically, we will identify "event relevancy" as a key concept to efficiently scale attention systems to very long sequences. We will experimentally compare our relevancy screening mechanism with other memory augmented networks, various orthogonal RNNs as well as gated architectures. This will be the topic of the **Second Article** entitled *"Untangling trade-offs between recurrence and self-attention in neural networks"* (Kerg et al., 2020), in Chapter 4.

In order to tackle (2), we are departing from purely looking at gradient propagation as an indication for information propagation. We will now be concerned by "associative information" capturing the relational information between sensory input, and we use out-of-distribution generalization in various purely relational tasks as a way to measure information propagation. Our goal is to distill a minimal set of inductive biases sufficient to solve these purely tasks. Among others, we identify the separation of relational information from the sensory input to be an essential inductive bias for OoD generalization on unseen inputs. Leveraging our findings, we are proposing a novel architecture merely based on relational information, and compare its performance with other existing methods on tasks testing for OoD generalization not only involving unseen inputs but also unseen relations. We will then look at limitations and possible extensions for the more challenging cases involving spurious features. This will be the topic of the **Third Article** entitled *"On Neural Architecture Inductive Biases for Relational Tasks"* (Kerg et al., 2022), in Chapter 5.

1.2. List of Excluded Contributions

During my PhD, I made several other contributions, not included in this thesis, on various machine learning topics such as Computational Neuroscience, Optimization and Meta-Learning.

- **Computational Neuroscience:**
  - Goal-driven optimization of single-neuron properties in artificial networks reveals regularization role of neural diversity and adaptation. V. Geadah, S. Horoi, **G. Kerg**, G. Wolf, G. Lajoie. Under review. (Geadah et al., 2022a)
  - Top-down optimization recovers biological coding principles of single-neuron adaptation in RNNs. V. Geadah, **G. Kerg**, S. Horoi, G. Wolf, G. Lajoie. Accepted at *Computational and Systems Neuroscience (COSYNE), 2022.* (Geadah et al., 2022b)
  - Network-level computational advantages of single-neuron adaptation. V. Geadah, G. Lajoie, **G. Kerg**, S. Horoi and G. Wolf. Accepted at *Computational and Systems Neuroscience (COSYNE), 2021.* (Geadah et al., 2021)
  - Advantages of biologically-inspired adaptive neural activation in RNNs during learning. V. Geadah, **G. Kerg**, S. Horoi, G. Wolf and G. Lajoie (Geadah et al., 2020)
- **Optimization:**

    - Catastrophic Fisher Explosion: Early Phase Fisher Matrix Impacts Generalization. S. Jastrzebski, D. Arpit, O. Astrand, **G. Kerg**, H. Wang, C. Xiong, R. Socher, K. Cho and K. Geras. Accepted at the *Thirty-eighth International Conference on Machine Learning (IMCL), 2021.* (Jastrzebski et al., 2021)
    - h-detach: Modifying the LSTM gradient towards better optimization. B. Kanuparthi*, D. Arpit*, **G. Kerg**, R. Ke, I. Mitliagkas and Y. Bengio. Accepted at the *Seventh International Conference on Learning Representations (ICLR), 2019.* (Kanuparthi et al., 2019), *indicates first authors.
- **Meta-Learning:**
    - Continuous-Time Meta-Learning with Forward Mode Differentiation. T. Deleu, D. Kanaa, L. Feng, **G. Kerg**, Y. Bengio, G. Lajoie, P.-L. Bacon. Accepted at the *Tenth International Conference on Learning Representations (ICLR), 2022.* (Deleu et al., 2022)

Before presenting the main articles, we will go over relevant background in Chapter 2: we will first cover the basics of neural networks and gradient-based learning in Section 2.1, which we will then use to further expand on recurrent neural networks and their gradient stability issues in Section 2.2. We will then give a brief introduction to the Schur decomposition and normal matrices in Section 2.3, as well as a short literature review for unitary RNNs in Section 2.4. An introduction to Fisher information in Section 2.5 will serve as a pre-requisite to study the short-term memory trace in linear RNNs in Section 2.6. This will be a useful background for the First Article in Chapter 3 as well as the Second Article in Chapter 4. Finally, we will cover the basics of self-attention and multi-head attention in Section 2.7, which will serve as background for the Second Article in Chapter 4 and the Third Article in Chapter 5.

Chapter 2

Background

2.1. Introduction to neural networks

2.1.1. Artificial neurons

An **artificial neuron** is a data processing system, biologically inspired by the neurons of the human brain, which attempts to approximate a function f mapping d input scalars to a single output scalar.

More formally, consider d input scalars $\{x_1,x_2,\ldots,x_d\}$, then an artificial neuron $h(x)$ is defined by:

$$h(x) = \phi\left(\sum_{i=1}^{d} w_i x_i + b\right)$$

where $w_i \in \mathbb{R}$ are the *weights*, $b \in \mathbb{R}$ the *bias*, and ϕ is a non-linear function called the *activation function*.

The term $\sum_{i=1}^{d} w_i x_i + b$ can be more compactly rewritten as $\mathbf{w}^T \cdot \mathbf{x} + b$ and is referred to as the *pre-activation*.

Popular choices for the activation function ϕ are:

- the **sigmoid** activation function,

$$\sigma : \mathbb{R} \to]0,1[: x \mapsto \frac{1}{1+e^{-x}}$$

  which is often interpreted as a probability of activation.

- the **tanh** or **hyperbolic tangent** activation function,

$$\tanh : \mathbb{R} \to]-1,1[: x \mapsto \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

- the **Rectified Linear Unit (ReLU)** activation function,

$$\text{ReLU} : \mathbb{R} \to \mathbb{R}^+ : x \mapsto \max(0,x) = x \cdot \mathbf{1}_{x \geq 0} = \begin{cases} 0 & x < 0, \\ x & x \geq 0. \end{cases}$$

  One sometimes uses a variant called **leaky ReLU** defined via a small positive quantity $\epsilon > 0$ called the *leak*,

$$\text{leaky ReLU} : \mathbb{R} \to \mathbb{R} : x \mapsto \max(0,x) + \min(0,\epsilon x) = \begin{cases} \epsilon x & x < 0, \\ x & x \geq 0. \end{cases}$$

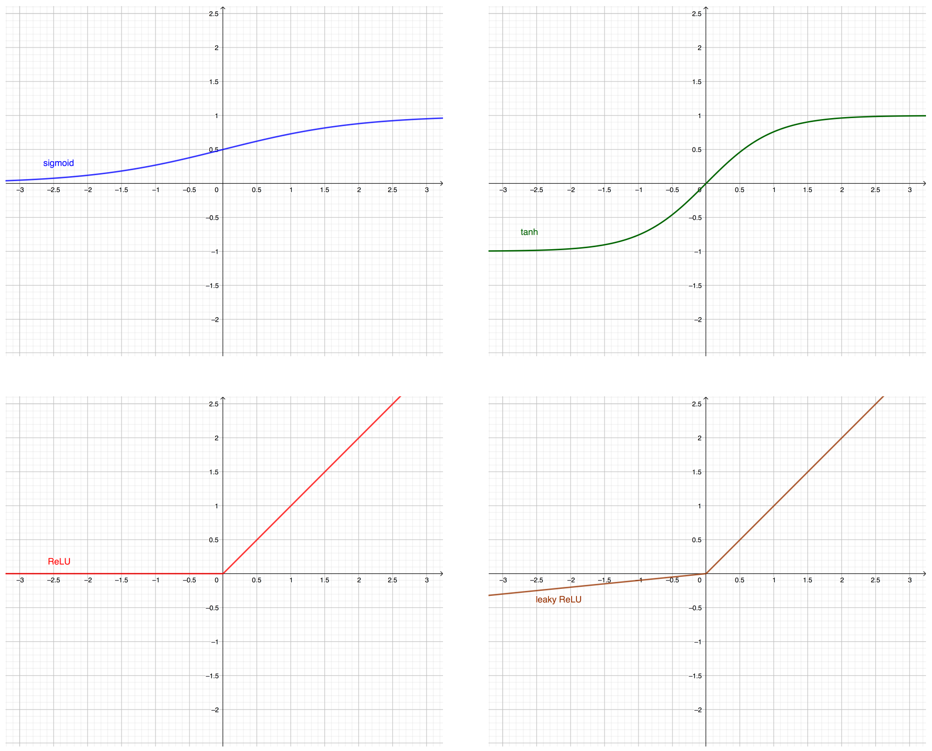


Fig. 1. Graphs of the most popular activation functions

How the set of parameters $\theta = \{w_1,w_2,\ldots,w_d,b\}$ is *trained* as to approximate f is being discussed in 2.1.3

2.1.2. Deep feedforward networks

Deep feedforward networks, or **feedforward neural networks**, or **multilayer perceptrons** (MLPs) is a data processing system, composed of artificial neurons organized in a layer-wise structure as to approximate a function mapping d inputs to a single output.

More formally, let us assume we have L layers, and in each layer $l = 1, \dots, L$, we have n_l artificial neurons, indexed $j = 1, \dots, n_l$, defined by

$$h_j^{(l)}(x) = \phi_l \left((\mathbf{w}_j^{(l)})^T x + b_j^{(l)} \right)$$

where ϕ_l is the activation function of layer l and $(\mathbf{w}_j^{(l)}, b_j^{(l)})$ are the weights and biases respectively. Note that x needs to be of dimension n_{l-1}, the number of neurons of the previous layer $l-1$.

We can write the previous equation more succinctly, by defining the matrix $\mathbf{W}^{(l)}$ to be composed of the row vectors $(\mathbf{w}_j^{(l)})^T$, the column vector $\mathbf{b}^{(l)}$ to be composed of the biases $b_j^{(l)}$ and the column vector $\mathbf{h}^{(l)}(x)$ to be composed of the outputs $h_j^{(l)}(x)$, as follows

$$\mathbf{h}^{(l)}(x) = \phi_l \left(\mathbf{W}^{(l)} x + \mathbf{b}^{(l)} \right)$$

where ϕ_l is applied element-wise, $\mathbf{h}^{(l)} : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$, $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$, $\mathbf{b}^{(l)} \in \mathbb{R}^{n_l}$, $n_0 = d$ the number of inputs, and $n_L = 1$. Then we can define the deep feed-forward network as

$$F(x) = (\mathbf{h}^{(L)} \circ \mathbf{h}^{(L-1)} \circ \ldots \circ \mathbf{h}^{(2)} \circ \mathbf{h}^{(1)})(x)$$

where the layers $l = 1, \dots, L-1$ are called the *hidden layers*, $l = 0$ the *input layer*, and $l = L$ the *output layer*.

The universal approximation theorem (Hornik et al., 1989) states that a feedforward network with one hidden layer can uniformly approximate any continuous function on compact sets of $\mathbb{R}^n$ to any desired degree of accuracy, given enough neurons in the hidden layer. The theorem does not provide any practical steps to find the weights and biases, but at least it gives us a guarantee that feedforward networks can express most functions that we wish to express or approximate for practical purposes. The drawback, however, is that the hidden layer might need to be too large to be useful and potentially fail to learn and generalize correctly. The challenges faced when attempting to find the appropriate weights and biases will be discussed in 2.1.3

2.1.3. Training neural networks

The process of continuously adapting the weights and biases of a feedforward network as to optimize a given performance measure P is called *training*.

Given a data set $D = \{(\mathbf{x}^{(1)},y_1), (\mathbf{x}^{(2)},y_2), \dots, (\mathbf{x}^{(n)},y_n)\}$, we usually partition $D = D_{\text{train}} \cup D_{\text{test}}$ into a training set D_{train} and test set D_{test}, where during training the model only uses D_{train} to adapt weights an biases. Ideally, we would like our performance measure P to be solely defined on the test set D_{test}, but this is intractable and hence we can only optimize P indirectly.

More formally, let

- θ be the set of parameters (i.e. the set of weights and biases)
- p_{data} the underlying *data-generating distribution* of D,
- $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ the per-example loss function,
- $f(\mathbf{x};\theta)$ the predicted output when the input is $\mathbf{x}$, and
- y the target output.

We would then like to minimize the objective function, also called *risk*,

$$J^*(\theta) = \mathbb{E}_{(\mathbf{x},y)\sim p_{\text{data}}} L(f(\mathbf{x};\theta),y)$$

However we don't have access to p_{data} directly, but only to the empirical distribution $\hat{p}_{\text{data}}$ via the training set D_{train}. We could therefore attempt to minimize the *empirical risk*

$$\hat{J}(\theta) = \mathbb{E}_{(\mathbf{x},y)\sim \hat{p}_{\text{data}}} L(f(\mathbf{x};\theta),y) = \frac{1}{|D_{\text{train}}|} \sum_{(\mathbf{x}^{(i)},y_i)\in D_{\text{train}}} L(f(\mathbf{x}^{(i)};\theta),y_i)$$

This procedure is called *empirical risk minimization*, and we hope that by reducing the *empirical risk*, we would reduce the *risk* as well. Unluckily, this approach might not work either as we are prone to overfitting: as we have seen with the universal approximation theorem, feedforward networks can potentially have very large capacity and thus might risk to simply memorize the dataset, in which case the empirical risk would be small but the actual risk would be high.

Instead, we take out a validation set D_{val} from the training set D_{train}, leaving us with a remaining training set $D'_{\text{train}} = D_{\text{train}} \setminus D_{\text{val}}$, we then minimize the empirical risk on D'_{train} while keeping track of the empirical risk or some other chosen performance measure on the held-out validation set D_{val}. We then choose to stop the empirical risk minimization procedure on D'_{train}, when the empirical risk on D_{val} (or some other chosen performance measure) starts to go up. This procedure is a regularization technique called *early stopping*.

2.1.4. Gradient-based learning

The empirical risk minimization on D'_{train} is usually done with a gradient based-optimization procedure such as as Gradient Descent or one of its variants. Gradient Descent is an iterative procedure that takes small steps in the steepest descent direction attempting to find a local minimum, following the update rule:

$$\theta_{t+1} \leftarrow \theta_t - \eta \nabla \hat{J}(\theta_t)$$

where η is called the learning rate, which can amplify or reduce the step size. Note that the algorithm stops whenever we are at a critical point θ^* (i.e. $\nabla \hat{J}(\theta^*) = 0$), or slows down whenever we are close to a critical point. Critical points are not necessarily local minima (or maxima), but can also be saddle points. Regions around saddle points are often hard to escape and are an active area of research in deep learning optimization.

In practice, we typically use *Stochastic Gradient Descent* (SGD) rather than plain Gradient Descent (also called '*batch Gradient Descent*', implying that we use the full training set), where instead of calculating $\hat{J}(\theta)$ over the entire data set D'_{train}, we use an approximation $\hat{J}_{\mathcal{S}}(\theta)$ calculated via a small randomly selected subset $\mathcal{S}$ of D'_{train} at each iteration. Since the small subset $\mathcal{S}$ is randomly selected, we have

$$\mathbb{E}_{\mathcal{S}} \nabla_\theta \hat{J}_{\mathcal{S}}(\theta) = \nabla_\theta \hat{J}(\theta)$$

and $\text{Var}_{\mathcal{S}} \hat{J}_{\mathcal{S}}(\theta)$ is a decreasing function in $|\mathcal{S}|$.

In fact one can prove that the standard error $\text{SE}((\nabla_\theta \hat{J}_{\mathcal{S}}(\theta))_i) = \frac{\sigma_i}{\sqrt{n}}$, where σ_i is the standard deviation of $(\nabla_\theta L(f(\mathbf{x},\theta),y))_i$ across $\mathcal{S}$, and $n = |\mathcal{S}|$. Thus we can see that we have less than linear returns to using more examples to estimate the gradient, while the computational costs grow linearly.

Also, there might potentially be a lot of redundancies in the training set, either identical copies or examples having negligible influence on the final gradient.

Further, small batches can offer a regularizing effect (Wilson and Martinez, 2003): the smaller the batch, the more noise we add to the gradient, and the stronger the regularization becomes.

Finally, in order to efficiently calculate gradients we would like the per-example loss function to be differentiable.

Commonly chosen per-example loss functions are:

- the L^2**-loss** defined by $L(x,y) = (x - y)^2$, mostly used in regression tasks.
- the **cross-entropy loss** defined by $L(x,y) = \text{CE}(x,y) = -x \log y$, mostly used in classification tasks.

2.1.5. Back-propagation algorithm

The Back-propagation algorithm is an iterative algorithm that efficiently computes gradients of the empirical risk. In the case of deep feedforward networks, the algorithm recursively applies the chain rule from layer to layer in order to compute the gradient in such a way as to avoid the exponential explosion of repeated subexpressions.

At each iteration, the algorithm contains one forward pass and one backward pass.

In the forward pass, it computes the pre-activations and activations, layer by layer starting from the input layer all the way to the output layer, and finally computing the empirical risk.

In the backward pass, it computes the error on the activations and pre-activations starting from the output layer all the way to the input layer.

Now we will derive two lemmas that are useful for the calculation of gradients in deep feedforward networks:

Lemma 2.1.1. *Let L be a real valued loss function, that depends on some vector $v = f(w)$, where $f : \mathbb{R}^m \to \mathbb{R}^n$. Then*

$$\nabla_w L = J_f^T \cdot \nabla_v L$$

where J_f is the Jacobian matrix of f.

PROOF. For each $i \in \{1,2,\ldots,m\}$, we have

$$\frac{\partial L}{\partial w_i} = \sum_j \frac{\partial v_j}{\partial w_i} \frac{\partial L}{\partial v_j} = \sum_j \left(J_f^T\right)_{ij} \cdot \frac{\partial L}{\partial v_j}$$

In matrix form this rewrites as,

$$\nabla_w L = J_f^T \cdot \nabla_v L$$

□

Lemma 2.1.2. *Let L be a real valued loss function, that depends on some vector $v = Aw + b$, where $A \in \mathbb{R}^{n\times m}$, $v,b \in \mathbb{R}^n$ and $w \in \mathbb{R}^m$. Then*

$$\nabla_A L = \nabla_v L \cdot w^T$$

and

$$\nabla_b L = \nabla_v L$$

PROOF. For each pair $(i,j) \in \{1,2,\ldots,n\} \times \{1,2,\ldots,m\}$, only v_i is a function of A_{ij}, and thus

$$\frac{\partial L}{\partial A_{ij}} = \frac{\partial v_i}{\partial A_{ij}} \cdot \frac{\partial L}{\partial v_i} = w_j \cdot \frac{\partial L}{\partial v_i}$$

Thus we see that $\nabla_A L$ is the exterior product between $\nabla_v L$ and w, and thus

$$\nabla_A L = \nabla_v L \otimes w = \nabla_v L \cdot w^T$$

Finally, it is trivial that for each $i \in \{1,2,\ldots,n\}$, we have

$$\frac{\partial L}{\partial b_i} = \frac{\partial L}{\partial v_i}$$

and thus

$$\nabla_b L = \nabla_v L$$

□

As in §2.1.2, let us rewrite a deep feedforward network as

$$F(x,\theta) = (h^{(L)} \circ h^{(L-1)} \circ \ldots \circ h^{(1)})(x)$$

and define $f^{(l)}(x) = (h^{(l)} \circ h^{(l-1)} \circ \ldots \circ h^{(1)})(x)$ and $a^{(l)}(x) = W^{(l)} f^{(l-1)}(x) + b^{(l)}$ for each $l = 1, \ldots, L$.

Then using Lemma 2.1.1 repeatedly, we get

$$\nabla_{f^{(l)}} L = J^T_{h^{(l+1)}} \cdot \nabla_{f^{(l+1)}} L \tag{2.1.1}$$

$$= J^T_{h^{(l+1)}} \cdot \ldots \cdot J^T_{h^{(L)}} \cdot \nabla_{f^{(L)}} L \tag{2.1.2}$$

$$= \left(\prod_{k=l+1}^{L} J^T_{h^{(k)}} \right) \cdot \nabla_F L \tag{2.1.3}$$

Using Lemma 2.1.1 again, we get

$$\nabla_{a^{(l)}} L = J^T_{\phi_l} \cdot \nabla_{f^{(l)}} L \tag{2.1.4}$$

$$= \text{diag}(\phi_l'(a^{(l)})) \cdot \nabla_{f^{(l)}} L \tag{2.1.5}$$

$$= \text{diag}(\phi_l'(a^{(l)})) \cdot \left(\prod_{k=l+1}^{L} J^T_{h^{(k)}} \right) \cdot \nabla_F L \tag{2.1.6}$$

Then using Lemma 2.1.2, we get

$$\nabla_{W^{(l)}} L = \nabla_{a^{(l)}} L \cdot f^{(l-1)}(x)^T \tag{2.1.7}$$

$$= \text{diag}(\phi_l'(a^{(l)})) \cdot \left(\prod_{k=l+1}^{L} J^T_{h^{(k)}} \right) \cdot \nabla_F L \cdot f^{(l-1)}(x)^T \tag{2.1.8}$$

and

$$\nabla_{b^{(l)}} L = \nabla_{a^{(l)}} L \tag{2.1.9}$$

$$= \text{diag}(\phi_l'(a^{(l)})) \cdot \left(\prod_{k=l+1}^{L} J^T_{h^{(k)}} \right) \cdot \nabla_F L \tag{2.1.10}$$

Note that the key quantity $\left(\prod_{k=l+1}^{L} J^T_{h^{(k)}}\right) \cdot \nabla_F L$ can be efficiently computed from left to right (i.e. starting with $\nabla_F L$ and left-multiplying with the Jacobian matrices) as opposed to from right to left. Once $\nabla_{f^{(l+1)}} L$ has been computed we store it and compute $\nabla_{f^{(l)}} L$ by left-multiplying with the Jacobian $J^T_{h^{(l+1)}}$.

2.2. Recurrent neural networks

2.2.1. Definition

Recurrent neural networks (or RNNs in short), introduced by Rumelhart et al. (1986), are a type of neural networks designed to process (varying length) sequential input data, outputting either a scalar or an output sequence. RNNs exploit the topology of sequential data via the so-called *parameter sharing* idea: the same parameters are used across different time steps making it possible to generalize to sequence lengths not seen during training.

Let us take an example of a recurrent network that maps an input sequence to an output sequence of the same length. More, formally let $x_1,x_2,\dots,x_\tau$ with $x_i \in \mathbb{R}^n$ be an input sequence , and $y_1,y_2,\dots,y_\tau$ with $y_i \in \mathbb{R}^k$ the target output sequence, then at each time step from $t = 1$ to $t = \tau$, we apply the equations:

$$h_t = \phi(\underbrace{Ux_t + Vh_{t-1} + b}_{a_t}) \tag{2.2.1}$$

$$o_t = Wh_t + c \tag{2.2.2}$$

$$\hat{y}_t = \text{softmax}(o_t) \tag{2.2.3}$$

where h_0 is a specified initial state, $h_t \in \mathbb{R}^m$, $V \in \mathbb{R}^{m\times m}$, $U \in \mathbb{R}^{m\times n}$, $W \in \mathbb{R}^{m\times k}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^k$ and ϕ is an activation function $\mathbb{R} \to \mathbb{R}$ inducing a function $\mathbb{R}^m \to \mathbb{R}^m$ via element-wise application. We refer to o_t as the output, h_t the hidden state, $\hat{y}_t$ the predicted output and x_t the input at time t respectively.

Let us recall that **softmax** is a function $\mathbb{R}^k \to \mathbb{R}^k : (z_1,z_2,\dots,z_k) \mapsto g(z)$, where for $j = 1,2,\dots,k$, we have

$$g(z)_j = \frac{e^{z_j}}{\sum_{i=1}^k e^{z_i}}$$

This can be seen as smoothed argmax function, where the maximum of $(z_1,...,z_k)$ is replaced by a value close to 1, and all the other values are being pushed close to 0. Since the sum of output vector sums to 1, we can view it as a vector of probabilities over the output. Also note that for $k = 2$ this boils down to using the sigmoid function.

We typically take the cross-entropy loss between y_t and $\hat{y}_t$ for each time step t, i.e. our total loss L writes as $L = \sum_t L_t$, where

$$\begin{aligned} L_t &= \text{CE}(y_t, \hat{y}_t) && (2.2.4) \\ &= -\sum_i (y_t)_i \cdot \log((\hat{y}_t)_i) && (2.2.5) \\ &= -\sum_i (y_t)_i \cdot \log(\text{softmax}(o_t)_i) && (2.2.6) \\ &= \log\Big(e^{(o_t)_1} + \ldots + e^{(o_t)_n}\Big) - \sum_i (y_t)_i (o_t)_i && (2.2.7) \end{aligned}$$

2.2.2. Gradient computation in recurrent neural networks

Now let us think how we would compute the gradient of this loss function with respect to the parameters (i.e. weights and biases) of the recurrent network. The gradient computation needs to perform a forward pass moving from $t = 1$ to $t = \tau$, followed by a backward pass from $t = \tau$ to $t = 1$, which is a runtime complexity of $O(\tau)$. Note that we cannot speed up the computation via parallelization since in the forward pass the state at each time step needs to be computed after the other, and needs to be stored until being used during the backward pass, thus the memory cost being $O(\tau)$. This back-propagation algorithm is called *back-propagation through time (BPTT)*.

Let us remark that there is a way to achieve parallelization of computation of the gradient in recurrent networks by omitting hidden-to-hidden recurrent connections, and having connections from the output of one time step to the hidden state of the next time step instead. These models can be trained with what is called *teacher forcing*, but are less powerful because the output units need to capture all the information about the past, which is unlikely to happen unless the user finds a way to engrain the full state information of the system into the target output. For more information see Goodfellow et al. (2016).

Let $V^{(t)}, U^{(t)}, W^{(t)}$ be the matrices V,U,W multiplying h_{t-1}, x_t, h_t respectively, and let $b^{(t)}, c^{(t)}$ be the vectors b,c in a_t, o_t respectively.

By first applying Lemma 2.1.2 and then Lemma 2.1.1, we get:

- $\nabla_V L = \sum_t \nabla_{V^{(t)}} L = \sum_t \nabla_{a_t} L \cdot h_{t-1}^T = \sum_t \text{diag}(\phi'(a_t)) \cdot \nabla_{h_t} L \cdot h_{t-1}^T$
- $\nabla_U L = \sum_t \nabla_{U^{(t)}} L = \sum_t \nabla_{a_t} L \cdot x_t^T = \sum_t \text{diag}(\phi'(a_t)) \cdot \nabla_{h_t} L \cdot x_t^T$
- $\nabla_W L = \sum_t \nabla_{W^{(t)}} L = \sum_t \nabla_{o_t} L \cdot h_t^T$
- $\nabla_b L = \sum_t \nabla_{b^{(t)}} L = \sum_t \nabla_{a_t} L = \sum_t \text{diag}(\phi'(a_t)) \cdot \nabla_{h_t} L$

- $\nabla_c L = \sum_t \nabla_{c^{(t)}} L = \sum_t \nabla_{o_t} L$

We are thus left with two key terms: $\nabla_{o_t} L$ and $\nabla_{h_t} L$

$\boxed{\text{For } \nabla_{o_t} L}$: Let us recall from equation 2.2.7 that our total loss L writes as $L = \sum_t L_t$, where

$$L_t = \log\Big(e^{(o_t)_1} + \ldots + e^{(o_t)_n}\Big) - \sum_i (y_t)_i (o_t)_i$$

thus

$$\left(\nabla_{o_t} L\right)_j = \left(\nabla_{o_t} L_t\right)_j = \frac{\partial L_t}{\partial (o_t)_j} = \text{softmax}(o_t)_j - (y_t)_j = (\hat{y}_t)_j - (y_t)_j$$

or equivalently in vector form,

$$\nabla_{o_t} L = \hat{y}_t - y_t$$

$\boxed{\text{For } \nabla_{h_t} L}$: let us distinguish between the cases $t = \tau$ and $t < \tau$, where τ is the last time step.

- If $t = \tau$, we use Lemma 2.1.1 to get

$$\nabla_{h_\tau} L = \left(\frac{\partial o_\tau}{\partial h_\tau}\right)^T \cdot \nabla_{o_\tau} L = W^T \cdot \nabla_{o_\tau} L$$

- If $t < \tau$, we notice that the loss L is impacted by h_t through o_t and h_{t+1}, thus

$$\nabla_{h_t} L = \left(\frac{\partial o_t}{\partial h_t}\right)^T \cdot \nabla_{o_t} L + \left(\frac{\partial h_{t+1}}{\partial h_t}\right)^T \cdot \nabla_{h_{t+1}} L = W^T \cdot \nabla_{o_t} L + J_{t+1}^T \cdot \nabla_{h_{t+1}} L$$

where

$$J_{t+1} = \frac{\partial h_{t+1}}{\partial h_t} = \frac{\partial h_{t+1}}{\partial a_{t+1}} \frac{\partial a_{t+1}}{\partial h_t} = \underbrace{\text{diag}(\phi'(a_{t+1}))}_{=:D_{t+1}} \cdot V$$

Unrolling the recursive expression until timestep τ, we get

$$\nabla_{h_t} L = W^T \cdot \nabla_{o_t} L + \ldots + J_{t+1}^T \cdot \ldots \cdot J_\tau^T \cdot W^T \cdot \nabla_{o_\tau} L \tag{2.2.8}$$

$$= \sum_{l=t}^{\tau} \left(\prod_{s=t+1}^{l} J_s^T\right) \cdot W^T \cdot \nabla_{o_l} L \tag{2.2.9}$$

$$= \sum_{l=t}^{\tau} \left(\prod_{s=t+1}^{l} J_s^T\right) \cdot W^T \cdot (\hat{y}_l - y_l) \tag{2.2.10}$$

2.2.3. Exploding vanishing gradient problem (EVGP)

Note that in the terms $\nabla_U L$, $\nabla_V L$ and $\nabla_b L$, we can spot the term $\nabla_{h_t} L$, which contains the products

$$\prod_{s=t+1}^{l} J_s^T = \prod_{s=t+1}^{l} (V^T \cdot D_s^T)$$

for $l = t, \ldots, \tau$.

If for one moment, we would imagine to be in the case of a linear recurrent neural network, where $D_s = \mathrm{Id}$, and we would assume the connectivity matrix V to admit an eigendecomposition of the form

$$V = Q\Lambda Q^T$$

then for $l \geq t+1$, we would get

$$\prod_{s=t+1}^{l} J_s^T = Q\Lambda^{l-t}Q^T$$

where the $(l-t)$-th power of the eigenvalues less than one would decay to zero, and $(l-t)$-th power of the eigenvalues greater than one would explode, if $(l-t)$ is sufficiently large.

Thus, unless the eigenvalues of V are all on the complex unit circle, the product $\prod_{s=t+1}^{l} J_s^T$ either explodes or vanishes given enough time steps $(l-t)$. This problem is called the *exploding and vanishing gradient problem (EVGP)* and was independently discovered by Bengio et al. (1993), Bengio et al. (1994) and Hochreiter (1991).

In fact, the results in Bengio et al. (1993) and Bengio et al. (1994) show that in order to for the hidden state to store bits of information in a way that is resistant to noise (i.e. small perturbation in the input do not affect the asymptotic behaviour of the hidden state trajectory in the state space), we need to have the spectral radii of the Jacobian matrices J_s to be smaller than one, leading to gradient vanishing. In other words we cannot robustly store information for an extended period of time in an RNN, without having to face the vanishing gradient problem.

One way to mitigate the vanishing gradient problem is to use "gated recurrent neural networks", most popular of which are *long-short term memory* (LSTM) networks (Hochreiter and Schmidhuber, 1997), and *gated recurrent unit* (GRU) networks (Cho et al., 2014a). Gates are additional paths through time that have been added such that the gradient can flow for longer durations, without vanishing or exploding as quickly as in the vanilla RNN case.

Another way has been to control the spectrum of the connectivity matrix, either at initialization or during the entire training period (Le et al., 2015; Henaff et al., 2016; Arjovsky et al., 2016; Wisdom et al., 2016; Lezcano-Casado and Martínez-Rubio, 2019). For a more detailed overview see Subsection 2.4 and for a very detailed discussion see Appendix §D.4.

2.3. Schur decomposition

In this subsection, we are going to present the Schur decomposition, which is a matrix decomposition we will make use of later on in chapter 3, in order to control the eigenspectrum and the non-normality of the eigenbasis of the RNN connectivity matrix.

Theorem. Let $A \in \mathbb{C}^{n\times n}$, then there exists a unitary matrix P and an upper-triangular matrix U (called the *Schur form* of A) such that

$$A = PUP^*$$

where P^* is the conjugate transpose of P. This decomposition is called the *Schur decomposition*.

Note that the diagonal elements of U are the eigenvalues of A. In fact,

$$\begin{aligned}\det\{(A-\lambda\mathrm{Id})\} &= \det\{(PUP^* - \lambda PP^*)\}\\ &= \det\{[P(U-\lambda\mathrm{Id})P^*]\}\\ &= \det\{P\}\cdot\det\{(U-\lambda\mathrm{Id})\}\cdot\det\{P^*\}\\ &= \det\{(U-\lambda\mathrm{Id})\}\end{aligned}$$

2.3.1. Normal matrices

Definition. A matrix $A \in \mathbb{C}^{n\times n}$ is normal if $A^*A = AA^*$, where A^* is the conjugate transpose of A.

(Note that all unitary matrices are normal matrices.)

Lemma 2.3.1. *$A \in \mathbb{C}^{n\times n}$ is normal if and only if its Schur form is normal.*

PROOF. Consider the Schur decomposition $A = PUP^*$, where U is the Schur form of A. Then,

$$A^*A = (PU^*P^*)(PUP^*) = PU^*UP^*$$

$$AA^* = (PUP^*)(PU^*P^*) = PUU^*P^*$$

And finally,

$$\begin{aligned}A^*A = AA^* &\Leftrightarrow PU^*UP^* = PUU^*P^*\\ &\Leftrightarrow U^*U = UU^*\\ &\Leftrightarrow U \text{ is normal}\end{aligned}$$

□

Lemma 2.3.2. *Let $U \in \mathbb{C}^{n\times n}$ be an upper-triangular matrix. Then U is normal if and only if U is diagonal.*

PROOF. Let us prove this claim by induction on n. The case $n = 1$ is trivial.
Let us now assume the claim to be true for $n \geq 1$, and consider an upper-triangular matrix $U \in \mathbb{C}^{(n+1)\times(n+1)}$ that is also normal. Then,

$$(UU^*)_{1,1} = \sum_{k=1}^{n+1} |U_{1,k}|^2$$

while

$$(U^*U)_{1,1} = |U_{1,1}|^2$$

Hence $U_{1,k} = 0$ for all $k > 1$.

Thus if V is the $n \times n$ sub-matrix of U, obtained by deleting the first row and first column of U, then we need to have $V^*V = VV^*$, which by induction hypothesis is diagonal. Thus U is diagonal.

Conversely, every diagonal matrix is trivially normal. $\square$

Definition. A matrix $A \in \mathbb{C}^{n\times n}$ is said to be *diagonalizable* if there exists an invertible matrix $P \in \mathbb{C}^{n\times n}$ and a diagonal matrix $D \in \mathbb{C}^{n\times n}$ such that

$$A = PDP^{-1}$$

(Note that here P defines an eigenbasis that is not necessarily orthogonal) **Defintion.** A matrix $A \in \mathbb{C}^{n\times n}$ has an *orthonormal eigenbasis* if there exists a unitary matrix $P \in \mathbb{C}^{n\times n}$ and a diagonal matrix $D \in \mathbb{C}^{n\times n}$ such that,

$$A = PDP^*$$

(Note that here $P^{-1} = P^*$)

Corollary 2.3.3. *A matrix $A \in \mathbb{C}^{n\times n}$ is normal if and only if it has an orthonormal eigenbasis.*

PROOF. Let us assume the matrix $A \in \mathbb{C}^{n\times n}$ is normal, and consider its Schur decomposition $A = PUP^*$, then by Lemma 2.3.1, we know that U is normal, and by Lemma 2.3.2, we know that U is diagonal.

Conversely, every matrix which has an orthonormal eigenbasis is trivially normal. $\square$

Intuition. Hence, one way of understanding normal vs non-normal matrices, is that normal matrices are exactly those matrices which have a diagonal Schur form and thus an orthonormal eigenbasis.

This view might allows to quantitatively describe "how non-normal" a matrix is, by looking "how far" its Schur form is from being diagonal. One example of such a measure has been introduced by Henrici (1962) and is given by

$$d : \mathbb{R}^{n\times n} \to \mathbb{R}^+ : A \mapsto \sqrt{\|A\|_F^2 - \sum_i |\lambda_i|^2}$$

where $\|.\|_F$ denotes the Frobenius norm and λ_i is the i-th eigenvalue of A. If we now consider the Schur decomposition of $A = PUP^*$, then

$$\|A\|_F^2 = \text{tr}(AA^*) = \text{tr}(PUU^*P^*) = \text{tr}(UU^*) \geq \sum_i |\lambda_i|^2$$

thus $d(W) = 0$ if and only if W is normal, and $d(W) > 0$ for all W non-normal.

2.3.2. The real Schur decomposition

Theorem. Let $A \in \mathbb{R}^{n\times n}$, then there exists an orthogonal matrix $Q \in \mathbb{R}^{n\times n}$ and a block upper-triangular matrix T of order 2 (i.e. T has diagonal blocks of size at most 2), such that

$$A = QTQ^T$$

where Q^T is the transpose of Q. We call T *the real Schur form* of A, and the decomposition is called *the real Schur decomposition*.

Before starting to derive this results, let us first consider some basic facts about linear algebra.
Lemma 2.3.4. *Let $A \in \mathbb{R}^{n\times n}$. The eigenvalues of A are either real or come in pairs of complex conjugates.*

PROOF. One way to prove this, goes by considering the Schur decomposition of $A = PUP^*$, and taking the complex conjugate:

$$PUP^* = A = \bar{A} = \bar{P}\bar{U}\bar{P}^*$$

(where $\bar{A}$ denotes conjugate of A and $\bar{P}^*$ denotes the conjugate of the conjugate transpose of P, thus simply the transpose of P) and noting that the set of eigenvalues of A, are the diagonal elements of U as well as of $\bar{U}$.

Another way to prove this, goes by consider the characteristic polynomial

$$p(\lambda) = \det\{(A - \lambda \mathrm{Id})\}$$

and noting that if $p(\lambda) = 0$, then $p(\bar{\lambda}) = \overline{p(\lambda)} = \bar{0} = 0$, since $p \in \mathbb{R}[X]$. □

In other words, if we consider the Schur decomposition $A = PUP^*$, then the diagonal elements of U are either real or have pairs of complex conjugates.
Let us recall that each complex number z can be written as $z = re^{i\theta}$, where $r = |z|$ and $e^{i\theta} = \cos\theta + i\sin\theta$. Further note that $\bar{z} = re^{-i\theta}$, and let us consider the identity,

$$\underbrace{\begin{pmatrix} r\cos\theta & -r\sin\theta \\ r\sin\theta & r\cos\theta \end{pmatrix}}_{=:\mathcal{R}(r,\theta)} = \underbrace{\begin{pmatrix} \frac{1}{\sqrt{2}} & -\frac{i}{\sqrt{2}} \\ -\frac{i}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}}_{=:W} \begin{pmatrix} re^{i\theta} & 0 \\ 0 & re^{-i\theta} \end{pmatrix} \underbrace{\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix}}_{W^*}$$

where $WW^* = W^*W = \mathrm{Id}$ is a unitary matrix, and $\mathcal{R}(r,\theta)$ is called a *rotation block*.

Now in order to outline the derivation of the real Schur decomposition from the Schur decomposition, let us construct a unitary matrix W_n composed of W sub-matrices (one for each pair of complex conjugates on the diagonal of U) such that PW_n^* is orthogonal and $W_nUW_n^*$ is block upper-triangular of order 2, where the order 2 blocks on the diagonal come from the corresponding rotation blocks. In other words, we have

$$A = \underbrace{(PW_n^*)}_{=Q} \underbrace{(W_n U W_n^*)}_{=T} \underbrace{(PW_n^*)^T}_{=Q^T}$$

2.4. Unitary RNN Literature Review

As already outlined in subsection §2.2.3, recurrent neural networks suffer from the exploding and vanishing gradient problem. One approach to mitigate this problem has been to use gates in order to have the gradient flow more efficiently along longer time horizons and therefore better capture long-term dependencies (Hochreiter and Schmidhuber, 1997; Cho et al., 2014a). Another approach is to stick with the vanilla RNN and constrain the spectrum of the connectivity matrix either at initialization (Le et al., 2015; Henaff et al., 2016) or throughout the entire training period (Arjovsky et al., 2016; Wisdom et al., 2016; Hyland and Rätsch, 2017; Mhammedi et al., 2017; Jing et al., 2017; Vorontsov et al., 2017; Lezcano-Casado and Martínez-Rubio, 2019).

Using orthogonal or identity connectivity matrices at initialization has been discussed in Saxe et al. (2014) and Le et al. (2015) (IRNN), which then became a stepping-stone for the seminal work of Arjovsky et al. (2016), called *unitary evolution RNN* (uRNN), outlining the use of unitary connectivity matrices in vanilla RNNs throughout the entire training procedure. In Arjovsky et al. (2016), unitary weight matrices have been parametrized as a product of several structured matrices, whose parameters are learned during training. However, the main drawback is that this parametrization does not have full capacity over the full unitary matrix group. For a more detailed discussion see Appendix §D.1.

To address this problem, Wisdom et al. (2016) provides necessary condition for a family of parametrized $n \times n$ unitary matrices to satisfy in order to contain all $n \times n$ unitary matrices, using Sard's theorem (Sard, 1942), and then leverages the theory of Riemannian gradient descent in order to show how to directly optimize a full-capacity unitary matrix along the Stiefel manifold (i.e. the manifold consisting of all $n \times n$ unitary matrices). Around the same time, another interesting and somewhat similar approach has been suggested in Hyland and Rätsch (2017) which parametrizes $U(n)$ directly in the corresponding Lie algebra $\mathfrak{u}(n)$ of skew- Hermitian matrices via the exponential matrix map.

Even though both methods Wisdom et al. (2016) and Hyland and Rätsch (2017) provide full-capacity parametrizations, it comes at the expense of increased computational cost $O(n^3)$.

Meanwhile Mhammedi et al. (2017) and Jing et al. (2017) both propose parametrizations where the user can decide how much of the matrix space should be covered by the parametrization, by tuning the number m of matrices involved in the parametrization product, leading to computational complexity $O(mn)$ if n is the number of hidden units. This is of particular use since for a lot of tasks a small subspace of the unitary matrix space is sufficient to solve the task. For more detailed discussions see Appendix D.3.

Building on the works of Wisdom et al. (2016) and Hyland and Rätsch (2017), Vorontsov et al. (2017) and Helfrich et al. (2018) both propose a real-valued full-capacity orthogonal matrix parametrization, where the connectivity matrix is updated using an additive update rule avoiding the compounded roundoff errors that would otherwise come in with a multiplicative update rule.

Finally, in the same spirit as Hyland and Rätsch (2017), Lezcano-Casado and Martínez-Rubio (2019) propose to parametrize the special orthogonal matrix group $SO(n)$ (or the unitary matrix group $U(n)$ for the complex case) via the exponential matrix map. Here, however, the exponential matrix map is approximated up to machine precision by combining the scaling-square trick with the Padé approximation as shown in Al-Mohy and Higham (2009), leading to efficient gradient calculations. It is further shown that, using the exponential map, a (constraint) minimization problem on connected and compact Lie groups (such as $SO(n)$ and $U(n)$) reduces to an equivalent (unconstraint) minimization problem on the respective Lie algebras which are isomorphic to a Euclidean vector space. The mentioned method is called expRNN and outperforms all other mentioned models across the board. For more detailed discussions see Appendix §D.4.

2.5. Fisher information theory

This section is provided in order to give some background for the tools used in subsection 2.6.

Let us consider an observable random variable X whose probability distribution $p(X|\theta)$ depends on some unknown parameter $\theta \in \mathbb{R}$. In frequentist statistics, one typically picks θ as to maximize the log-likelihood of $p(x|\theta)$ for some observed x's. In order to assess how good the estimate for θ is one defines the *score function*

$$s(x,\theta) = \frac{\partial}{\partial\theta}\log p(x|\theta)$$

which, if θ is the true underlying parameter, gives zero in expectation:

$$\begin{aligned}\mathbb{E}_{p(X|\theta)}\left[s(X,\theta)\right] &= \mathbb{E}_{p(X|\theta)}\left[\frac{\partial}{\partial\theta}\log p(X|\theta)\right] \\ &= \int \frac{\frac{\partial}{\partial\theta}p(x|\theta)}{p(x|\theta)}p(x|\theta)dx \\ &= \frac{\partial}{\partial\theta}\int p(x|\theta)dx \\ &= \frac{\partial}{\partial\theta}1 = 0\end{aligned}$$

Intuitively, if p is sharply peaked around its optimal value for θ given the observed x's, then we would need much less observation to give a good estimate for θ than in the case where the shape p

as a function of θ would be rather flat. In the former case, we would think of X as carrying a lot of information about the parameter θ, while in the latter case we would think of X as carrying not so much information about θ. In order to measure how much information X carries about θ, our first reflex would be to think of second derivatives or some sort of variance.

Formally we define the *Fisher information*

$$\mathcal{I}(\theta) = \mathrm{Var}_{p(X|\theta)}\left[s(X,\theta)\right] = \mathbb{E}_{p(X|\theta)}\left[s(X,\theta)^2\right]$$

Intuitively, the Fisher information measures how much information X carries about the parameter θ, by measuring the amount of variance we get in our assessment via s. The larger $\mathcal{I}(\theta)$ is, the more information the assessment s provides us about our estimate, the more information X carries about θ and the less data we need to be sufficiently confident about our estimates for θ.

To link it back to our intuition of second derivatives, we can observe that

$$\begin{aligned}
\frac{\partial^2}{\partial\theta^2}\log p(X|\theta) &= \frac{\partial}{\partial\theta}\frac{\frac{\partial}{\partial\theta}p(X|\theta)}{p(X|\theta)} \\
&= \frac{\left(\frac{\partial^2}{\partial\theta^2}p(X|\theta)\right)\cdot p(X|\theta) - \left(\frac{\partial}{\partial\theta}p(X|\theta)\right)^2}{p(X|\theta)^2} \\
&= \frac{\frac{\partial^2}{\partial\theta^2}p(X|\theta)}{p(X|\theta)} - \left(\frac{\frac{\partial}{\partial\theta}p(X|\theta)}{p(X|\theta)}\right)^2 \\
&= \frac{\frac{\partial^2}{\partial\theta^2}p(X|\theta)}{p(X|\theta)} - s(X,\theta)^2
\end{aligned}$$

taking the expectation both sides, gives

$$\begin{aligned}
\mathbb{E}_{p(X|\theta)}\left[\frac{\partial^2}{\partial\theta^2}\log p(X|\theta)\right] &= \int \frac{\frac{\partial^2}{\partial\theta^2}p(x|\theta)}{p(x|\theta)}p(x|\theta)dx - \mathcal{I}(\theta) \\
&= \frac{\partial^2}{\partial\theta^2}\int p(X|\theta)dx - \mathcal{I}(\theta) \\
&= \frac{\partial^2}{\partial\theta^2}1 - \mathcal{I}(\theta) \\
&= -\mathcal{I}(\theta)
\end{aligned}$$

and thus

$$\mathcal{I}(\theta) = -\mathbb{E}_{p(X|\theta)}\left[\frac{\partial^2}{\partial\theta^2}\log p(X|\theta)\right]$$

which measures the expected amount of curvature or the expected amount of "spread-out-ness" of the log-likelihood function $\log p(X|\theta)$. Near the maximum log-likelihood, large Fisher information indicates a sharp maximum, and thus we don't need as many observations in order to be confident about our estimate for θ, while low Fisher information indicates low curvature around the maximum, hence many nearby values have similar log-likelihood than the maximum and a lot more observations are needed.

Note that above derivation is only correct if $\log p(X|\theta)$ is twice differentiable with respect to θ, and if some further regularity conditions on p are satisfied so that the Leibniz rule can be applied (which is usually the case in most practical situations).

If we now consider the higher dimensional case where $\theta = [\theta_1, \dots, \theta_n]^T$, we analogously define the score function as the gradient

$$s(x,\theta) = \nabla_\theta \log p(x|\theta)$$

where $\mathbb{E}_{p(X|\theta)}[s(X,\theta)] = 0$ and the *Fisher information matrix* (FIM) as

$$\begin{aligned}
[\mathcal{I}(\theta)]_{i,j} &= \mathrm{Cov}_{p(X|\theta)}[s(X,\theta)_i, s(X,\theta)_j] \\
&= \mathbb{E}_{p(X|\theta)}[s(X,\theta)_i s(X,\theta)_j] \\
&= \mathbb{E}_{p(X|\theta)}\left[\left(\frac{\partial}{\partial\theta_i}\log p(x|\theta)\right)\left(\frac{\partial}{\partial\theta_j}\log p(x|\theta)\right)\right]
\end{aligned}$$

Similarly as in the one-dimensional case, one can prove that the FIM is equal to the negative Hessian matrix of the log-likelihood function,

$$[\mathcal{I}(\theta)]_{i,j} = -\mathrm{E}_{p(X|\theta)}\left[\frac{\partial^2}{\partial\theta_i\partial\theta_j}\log p(X|\theta)\right]$$

under the analogous regularity assumptions for the higher dimensional case.

2.6. Non-normal dynamics in linear RNNs

A lot of the theory about non-normal dynamics and temporal memory capacity in neural networks have been developed in the computational neuroscience literature (Ganguli et al., 2008; Hennequin et al., 2012; L White et al., 2004), which highlights how attempting to understand the human brain can lead to a useful development of artificial neural networks.

In this section we are going to discuss the work of Ganguli et al. (2008), where Fisher information theory is applied to study the short-term memory trace of the recurrent network with update equation

$$h_t = \phi(Wh_{t-1} + vs_t + z_t)$$

where ϕ is an activation function, $W \in \mathbb{R}^{N\times N}$, $v \in \mathbb{R}^N$ with $\|v\| = 1$, s_t is a scalar input signal, and $z_t \sim \mathcal{N}(0,\epsilon \cdot \mathrm{Id}_N)$ is a multivariate Gaussian noise of dimension N.

We would like to study the limits of these short-term memory traces as well as the properties of the recurrent networks necessary to achieve these limits. For this purpose we introduce the Fisher memory matrix (FMM) given by

$$\mathbf{J}_{kl}\left(s_{\leq t}\right) = \mathbb{E}_{p\left(\mathbf{h}_t|s_{\leq t}\right)}\left[-\frac{\partial^2}{\partial s_{t-k}\partial s_{t-l}}\log p\left(\mathbf{h}_t|s_{\leq t}\right)\right]$$

where $s_{\leq t} = \{s_{t-k}|k \geq 0\}$ denotes the history of past input signals. Note that the FMM can be viewed as a Fisher information matrix (seen in §2.5) where the parameters are the past signals $s_{\leq t}$, and (similar to the derivation in §2.5) one can rewrite the FMM as

$$\mathbf{J}_{kl}\left(s_{\leq t}\right) = \mathbb{E}_{p\left(\mathbf{h}_t|s_{\leq t}\right)}\left[\left(\frac{\partial}{\partial s_{t-k}}\log p\left(\mathbf{h}_t|s_{\leq t}\right)\right)\left(\frac{\partial}{\partial s_{t-l}}\log p\left(\mathbf{h}_t|s_{\leq t}\right)\right)\right]$$

Similarly to the FIM, the FMM measures how much information the current state h_t carries about the past signals $s_{\leq t}$. The diagonal element $J_{kk}(s_{\leq t})$ of the FMM is precisely Fisher information h_t retains about the input signal s_{t-k}, while the off-diagonal element $J_{kl}(s_{\leq t})$ measures the interference between two past signals s_{t-k} and s_{t-l}.

In order to gain a better intuition of the above definitions, one considers the case where ϕ is the identity. We then get

$$\boxed{h_t = \sum_{k=0}^{\infty} W^k v \cdot s_{t-k} + \sum_{k=0}^{\infty} W^k z_{t-k}}$$

where $h_t|s_{\leq t}$ is a multivariate Gaussian of mean

$$\mu_t := \mathbb{E}[h_t|s_{\leq t}] = \sum_{k=0}^{\infty} W^k v \cdot s_{t-k}$$

and covariance matrix

$$C := \mathrm{Var}[h_t|s_{\leq t}] = \epsilon \cdot \sum_{k=0}^{\infty}(W^k)(W^k)^T$$

Hence a simple calculation shows that

$$J_{kl}(s_{\leq t}) = (W^k v)^T C^{-1}(W^l v)$$

which is an expression independent of the signal history $s_{\leq t}$, thus we will simply write J_{kl} from now onwards. Note that the elements J_{kk} are proportional to $\frac{1}{\epsilon}$, which intuitively means that, the larger the magnitude ϵ of the noise we inject at each time step, the smaller the Fisher information, and the less information h_t seems to carry about the past input signals $s_{\leq t}$.

The signal-to-noise ratio (SNR) of the input vector $vs_n + z_n$ at any given time n is defined as the Fisher information

$$\text{Var}\left[\frac{\partial}{\partial s_n}\log p(vs_n + z_n|s_n)\right] = v^T \cdot (\epsilon\text{Id})^{-1} \cdot v = \frac{1}{\epsilon}$$

and viewing J_{kk} as a multiple of $\frac{1}{\epsilon}$, one can think of J_{kk} as the fraction of the input SNR at time $t-k$ that is remaining in the system at time t. Since all input SNRs are equal to $\frac{1}{\epsilon}$, the following sum can be interpreted as the total SNR embedded in h_t relative to the input SNRs $\frac{1}{\epsilon}$:

$$\begin{aligned} J_{tot} &= \sum_{k=0}^{\infty} J_{kk} \\ &= \sum_{k=0}^{\infty}(W^k v)^T \cdot C^{-1} \cdot (W^k v) \\ &= v^T \cdot \underbrace{\left[\sum_{k=0}^{\infty}(W^k)^T \cdot C^{-1} \cdot W^k\right]}_{=:J^{(s)}} \cdot v \end{aligned}$$

where $J^{(s)}$ is called the spatial Fisher information matrix.

If W is normal, a simple calculation using the matrix eigendecomposition shows that $J^{(s)} = \frac{1}{\epsilon}\text{Id}$, and thus $J_{tot} = \frac{1}{\epsilon}$, which is independent of the direction v by which the input signals are fed into the system. This implies that the total SNR has to stay constant regardless of how one picks v and W, as long as W is normal. Either one chooses W and v to enhance memory of the inputs from the recent past, while sacrificing memory from the remote past, or the other way around.

However when W is non-normal, the expression $J_{tot} = v^T \cdot J^{(s)} \cdot v$ does not simplify and thus stays dependent on the direction v. Hence J_{tot} is maximized when taking v as the eigenvector of the largest eigenvalue of $J^{(s)}$, and minimized when taking v as the eigenvector of the smallest eigenvalue of $J^{(s)}$.

Now note that $\text{Tr}(J^{(s)}) = \frac{N}{\epsilon}$, and thus we get the bound $J_{tot} \leq \frac{N}{\epsilon}$, and it turns out that there are choices for W and v such that the inequality becomes an equality. Let us consider the non-normal matrix $W_{ij} = \sqrt{\alpha}\delta_{j,i-1} + \sqrt{\beta}\delta_{j,i+1}$ and the vector $v_i = \delta_{i,1}$, where we can spot the feed-forward connections $\sqrt{\alpha}$ (giving rise to a delay line) as well as the feedback connections $\sqrt{\beta}$ (giving rise to a feedback delay line).

For $\beta = 0$ and $\alpha > 1$, we get $J_{tot} \approx \frac{N}{\epsilon}(1 - \frac{1}{\epsilon})$, where the stronger the delay line, the closer we get to optimal memory in the network. Here the signal comes in at the source and while propagating form layer to layer, gets amplified exponentially until the last layer where the signal then dies out after a time of order of N. Note that the larger N becomes, the bigger the extensive memory difference to normal matrices becomes.

An alternative would be to consider a delay line with feedback connections, where $\beta > 0$ and $\alpha < 1$, with the additional condition $1 - \sqrt{\alpha} < \sqrt{\beta} < \frac{1}{4\sqrt{\alpha}}$. Here again the signal enters at the source and getting exponentially amplified from layer to layer, while at each layer having part of the signal being fed backwards into the previous layer. This allows the amplification to last longer than order of N time steps, at the expense of having less aggressive amplification.

Let us fix k, and consider the sequence of signal amplification $A_m = \|W^m v\|^2$, for $0 \leq m \leq k$, then it is shown that

$$J_{kk} \leq \frac{1}{\epsilon \sum_{m=0}^{k} A_m^{-1}}$$

with equality if and only if $W_{ij} = \sqrt{\frac{A_{i-1}}{A_{i-2}}}\delta_{j,i-1}$ for $i = 2, \ldots, k+1$ and $v_i = \delta_{i,1}$. Hence for a given amount of signal amplification, the delay line is the unique network that achieves minimal noise amplification.

Finally note that when performing a change of basis (i.e. replacing W by PWP^T, with P being an orthogonal matrix) in the expression of J_{kl}, we would only need to pick a new vector $v' = Pv$ to get the same J_{kl} and thus the same J_{tot}. Hence one can view the matrices that are unitarily equivalent to simple delay lines as opposites to normal matrices, and extreme examples of non-normal matrices when comparing their ability to propagate signal in the above recurrent network. One can further prove that matrices that are unitarily equivalent to simple delay lines have a diagonal Fisher memory matrix, and thus they do not give rise to any interference between signals injected at different time steps. For more details, see Ganguli et al. (2008).

2.7. Attention

An attention function Φ maps a query $q \in \mathbb{R}^d$ and a set of key-value pairs $\{(k_1,v_1), \ldots, (k_n,v_n)\}$ where $k_i \in \mathbb{R}^d$ and $v_i \in \mathbb{R}^m$ to an output $o \in \mathbb{R}^m$. More precisely,

$$\Phi(q,\{(k_1,v_1), \ldots, (k_n,v_n)\}) = \alpha_1 v_1 + \ldots + \alpha_n v_n = o \in \mathbb{R}^m$$

where $\bar{\alpha} = (\alpha_1, \ldots, \alpha_n) = \text{softmax}[a(q,k_1), \ldots, a(q,k_n)]$ and $a : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ is the *attention alignment function*. Note that the output is a weighted sum or convex combination of the value vectors v_i, where the weight α_i assigned to each value v_i quantifies how well the query q "aligns" with the corresponding key vector k_i, according to the attention alignment function a. We will now look at two common attention functions used in the literature which distinguish themselves merely by the choice of their alignment function.

Additive Attention. The first attention function proposed in the literature by Bahdanau et al. (2014) is often referred to "Additive Attention". It uses the following "additive" attention alignment function:

$$a(q,k) = v_a^T \cdot \tanh\left(W_a \cdot q + U_a \cdot k\right)$$

where $v_a \in \mathbb{R}^n$, and $W_a, U_a \in \mathbb{R}^{n \times d}$.

Scaled Dot-Product Attention. Later on the seminal work of Vaswani et al. (2017) proposes the "Scaled Dot-Product Attention" defined via the following attention alignment function:

$$a(q,k) = \frac{q^\top \cdot k}{\sqrt{d}}$$

One sometimes uses the simple "Dot-Product Attention" where one leaves out the scaling factor $\sqrt{d}$, but for large values of d, one prefers to use the "Scaled Dot-Product Attention" as otherwise the dot-product $q^\top k$ can grow large in magnitude, which after application of the softmax function might result in extremely small gradients. For small values of d, both dot-products perform similarly, while additive attention outperforms the dot-product attention without scaling factor (Britz et al., 2017). While both additive and dot-product attention have similar theoretical complexity, dot-product attention is faster and more memory-efficient in practice as it can be implemented with a highly optimized matrix multiplication code.

Multi-Head Attention. Instead of using one attention function, one can use multiple attention functions Φ_i in parallel to enable the model to jointly attend to different positions and hence increase representational capacity. The outputs $o^{(i)}$ of each attention head are concatenated and projected to give the final output. More precisely, for each of the h attention heads Φ_i, we define separate alignment functions

$$a_i(q,k) = \frac{f_i(q)^\top \cdot g_i(k)}{\sqrt{n}}$$

where the functions $f_i, g_i : \mathbb{R}^d \to \mathbb{R}^n$ are projecting q and k onto a common space $\mathbb{R}^n$. Hence

$$\Phi_i(q, \{(k_1,v_1), \ldots, (k_n,v_n)\}) = \alpha_1^{(i)} v_1 + \ldots + \alpha_n^{(i)} v_n = o^{(i)} \in \mathbb{R}^m$$

where $\bar{\alpha}^{(i)} = (\alpha_1^{(i)}, \ldots, \alpha_n^{(i)}) = \text{softmax}[a_i(q,k_1), \ldots, a_i(q,k_n)]$.

One commonly chooses f_i and g_i to be linear maps, in which case one has weight matrices $W_q^{(i)} \in \mathbb{R}^{n \times d}$ and $W_k^{(i)} \in \mathbb{R}^{n \times d}$ such that

$$a_i(q,k) = \frac{(W_q^{(i)} \cdot q)^\top \cdot (W_k^{(i)} \cdot k)}{\sqrt{n}}$$

To get the final result, one concatenates the outputs $o^{(i)}$ and performs an additional linear projection to get the final output:

$$o = W_o[o^{(i)}, \ldots, o^{(h)}]$$

where $W_o \in \mathbb{R}^{d_o \times hm}$.

2.7.1. Self-Attention in RNNs

Let $x_t \in \mathbb{R}^m$ be the input and $h_t \in \mathbb{R}^n$ be the RNN hidden state at time step t, satisfying the update equation for all $t \geq 1$,

$$h_{t+1} = \phi(Vs_t + Ux_{t+1} + b) \tag{2.7.1}$$

$$s_t = f(h_t, c_t) \tag{2.7.2}$$

$$c_t = \Phi(s_{t-1}, \{(h_1, h_1), \ldots, (h_n, h_n)\}) \tag{2.7.3}$$

$$= \alpha_1^{(t)} h_1 + \ldots + \alpha_t^{(t)} h_t \tag{2.7.4}$$

where $\bar{\alpha}^{(t)} = (\alpha_1^{(t)}, \ldots, \alpha_t^{(t)}) = \text{softmax}[a(s_{t-1}, h_1), \ldots, a(s_{t-1}, h_t)]$, ϕ is a non-linearity, $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^p$, $V \in \mathbb{R}^{n \times p}$, $U \in \mathbb{R}^{n \times m}$, $b \in \mathbb{R}^n$ and where $a : \mathbb{R}^p \times \mathbb{R}^n \to \mathbb{R}^n$ is the attention *alignment function*.

In the original paper Bahdanau et al. (2014), one used $f(h_t, c_t) = h_t + c_t$ in which case $p = n$, but concatenation $f(h_t, c_t) = (h_t, c_t)$ would be more general and tends to improve performance in a lot of cases. One can then either use "additive attention" (as in Bahdanau et al. (2014)),

$$a(s_{t-1}, h_j) = v_a^\top \cdot \tanh\left(W_a \cdot s_{t-1} + U_a \cdot h_j\right)$$

or one can use "scaled dot-product attention" (as suggested in Vaswani et al. (2017)),

$$a(s_{t-1}, h_j) = \frac{(W_a \cdot s_{t-1})^\top \cdot (U_a \cdot h_j)}{\sqrt{n}}$$

where $v_a \in \mathbb{R}^n$, $W_a \in \mathbb{R}^{n \times p}$ and $U_a \in \mathbb{R}^{n \times n}$.

Chapter 3

Non-normal Recurrent Neural Network (nnRNN): learning long time dependencies while improving expressivity with transient dynamics

This chapter is based on a NeurIPS 2019 paper (Kerg et al., 2019).

3.1. Prologue to the First Article

3.1.1. Article Details

Title: Non-normal Recurrent Neural Network (nnRNN): learning long time dependencies while improving expressivity with transient dynamics

Authors: Giancarlo Kerg[1,2,*], Kyle Goyette[1,2,3,*], Maximilian Puelma Touzel[1,4], Gauthier Gidel[1,2], Eugene Vorontsov[1,5], Yoshua Bengio[1,2,6], Guillaume Lajoie[1,7]

[*] Indicates first authors. Ordering determined by coin flip.

Affiliations:

1: Mila - Quebec AI Institute, Canada

2: Université de Montréal, DIRO, Montreal, Canada

3: Université de Montréal, CIRRELT, Montreal, Canada

4: IVADO post-doctoral fellow

5: Ecole Polytechnique de Montréal, Montreal,Canada

6: CIFAR senior fellow

7: Université de Montréal, Département de Mathématiques et Statistiques, Montreal, Canada

3.1.2. Contributions

Giancarlo Kerg had the main idea of using the Schur decomposition to train non-normal spectrally constrained matrices. Giancarlo Kerg did most of the smaller scale toy experiments supporting the design and directions of the larger scale experiments while Kyle Goyette implemented all of the larger scale experiments, and did all the hyperparameter tuning. Giancarlo Kerg derived most theorems and led the writing of the article.

3.1.3. Paper Abstract

A recent strategy to circumvent the exploding and vanishing gradient problem in RNNs, and to allow the stable propagation of signals over long time scales, is to constrain recurrent connectivity matrices to be orthogonal or unitary. This ensures eigenvalues with unit norm and thus stable dynamics and training. However this comes at the cost of reduced expressivity due to the limited variety of orthogonal transformations. We propose a novel connectivity structure based on the Schur decomposition and a splitting of the Schur form into normal and non-normal parts. This allows to parametrize matrices with unit-norm eigenspectra without orthogonality constraints on eigenbases. The resulting architecture ensures access to a larger space of spectrally constrained matrices, of which orthogonal matrices are a subset. This crucial difference retains the stability advantages and training speed of orthogonal RNNs while enhancing expressivity, especially on tasks that require computations over ongoing input sequences.

3.2. Introduction

Training recurrent neural networks (RNN) to process temporal inputs over long timescales is notoriously difficult. A central factor is the exploding and vanishing gradient problem (EVGP) (Hochreiter and Schmidhuber, 1997; Bengio et al., 1994; Pascanu et al., 2013a), which stems from the compounding effects of propagating signals over many iterates of recurrent interactions. Several approaches have been developed to mitigate this issue, including the introduction of gating mechanisms (e.g. (Jing et al., 2019; Hochreiter and Schmidhuber, 1997)), purposely using non-saturating activation functions (Chandar et al., 2019), and manipulating the propagation path of gradients (Kanuparthi et al., 2019). Another way is to constrain connectivity matrices to be orthogonal (and more generally, unitary) leading to a class of models we refer to as *orthogonal RNNs* (Mhammedi et al., 2017; Maduranga et al., 2019; Lezcano-Casado and Martínez-Rubio, 2019; Wisdom et al., 2016; Jing et al., 2017; Vorontsov et al., 2017; Helfrich et al., 2018; Arjovsky et al., 2016). Orthogonal RNNs have eigenspectra with unit norm, therefore helping to prevent exponential growth or decay in long products of Jacobians associated with EVGP. They perform exceptionally well on tasks requiring memorization of inputs over long time-scales (Henaff et al., 2016) (outperforming gated networks) but struggle on tasks involving continued computations across timescales. A contributing factor to this limitation is the mutually orthogonal nature of connectivity eigendirections which substantially limits the space of solutions available to orthogonal RNNs.

In this paper, we propose a first step toward a solution to this expressivity problem in orthogonal RNNs by allowing non-orthogonal eigenbases while retaining control of eigenvalues' norms. We achieve this by leveraging the *Schur* decomposition of the connectivity matrix, providing a separation into "diagonal" and "feed-forward" parts, with their own optimization constraints. Mathematically, this amounts to adding "non-normal" connectivity, and we call our novel architecture *non-normal RNN* (nnRNN). In linear algebra, a matrix is called *normal* if its eigenbasis is orthogonal, and *non-normal* if not. Orthogonal matrices are normal, with eigenvalues with norm one (i.e. on the unit circle). In recurrent networks, normal connectivity produces dynamics solely characterized by the eigenspectrum while non-normal connectivity allows transient expansion and compression. Transient dynamics have known computational advantages (Hennequin et al., 2012; Ganguli et al., 2008), but orthogonal RNNs cannot produce them. The added flexiblity in nnRNN allows such transients, and we show analytically how they afford additional expressivity to better encode complex inputs, while at the same time retaining efficient signal propagation to learn long-term dependencies. Through a series of numerical experiments, we show that the nnRNN provides two main advantages:

(1) On tasks well suited for orthogonal RNNs, nnRNN learns orthogonal (normal) connectivity and matches state-of-the art performance while training as fast as orthogonal RNNs.
(2) On tasks requiring additional expressivity, non-normal connectivity emerges from training and nnRNN outperforms orthogonal RNNs.

From a parametric standpoint, this advantage can be attributed to the fact that the nnRNN has access to *all* matrices with unit-norm eigenspectra, of which orthogonal ones are only a subset.

3.3. Background

3.3.1. Unitary RNNs and constrained optimization

First outlined in Arjovsky et al. (2016) and inspired by Saxe et al. (2014); Yang et al. (2015); Le et al. (2015), RNNs whose recurrent connectivity is determined by an orthogonal, or unitary matrix are a direct answer to the EVGP since their eigenspectra exactly lie on the complex unit circle. The same mechanism was invoked in a series of theoretical studies for deep and recurrent networks in the large size limit, showing that ideal regimes for effective network performance are those initialized with such spectral attributes (Raghu et al., 2017; Pennington et al., 2017; Chen et al., 2018). By construction, orthogonal matrices and their complex-valued counterparts, unitary matrices, are isometric operators and do not expand or contract space, which helps to mitigate the EVGP. A central challenge to train unitary RNNs is to ensure that parameter updates are restricted to the manifolds satisfying orthogonality constraints known as *Stiefel* manifolds (see review in Jiang and Dai (2015)). This is an active area of optimization research, and several techniques have been used for orthogonal or unitary RNN training. In Arjovsky et al. (2016), the authors construct connectivity matrices with long products of rotation matrices leveraging fast Fourier transforms.

In Wisdom et al. (2016); Vorontsov et al. (2017); Helfrich et al. (2018), the Cayley transform is used, which parametrizes weight matrices using skew-symmetric matrices that need to be inverted (c.f. Chang et al. (2019) for an RNN implementation directly using skew-symmetric matrices). Another approach uses Householder reflections (Mhammedi et al., 2017). Recent studies also adapt some of these methods to the quaternion domain (Parcollet et al., 2019). The methods listed above have their advantages by either being fast, or memory efficient, but suffer from only parametrizing a subset of all orthogonal (unitary) matrices. A novel approach considering the group of unitary matrices as a Lie group and leveraging a parametrization via the exponential map applied to its Lie algebra, addresses this problem and currently outperforms the rest on many tasks (Lezcano-Casado and Martínez-Rubio, 2019). Still, of all matrices with unit-norm eigenvalues, unitary matrices are only a small subset and remain limited in their expressivity since they are restricted to isometric transformations (Henaff et al., 2016). This is why orthogonal RNNs, while performing better than a conventional RNN or LSTM at some tasks (e.g. copy task (Hochreiter and Schmidhuber, 1997), or sequential MNIST (Le et al., 2015)), struggle at more complex tasks requiring computations across multiple timescales.

3.3.2. Non-normal connectivity

Any diagonalizable matrix V can be expressed as $V = P\Theta P^{-1}$ where P's columns are V's eigenvectors and Θ is a diagonal matrix containing its eigenvalues. V is said to be *normal* if its eigenbasis is orthogonal and thus, $P^{-1} = P^{\top}$ and $V = P\Theta P^{\top}$. Orthogonal matrices are normal matrices with eigenvalues on the unit circle. When a matrix is *non-normal*, it is diagonalized with a non-orthogonal basis. However, it is still possible to express it using an orthogonal basis at the cost of adding (lower) triangular structure to Θ. This is known as the *Schur* decomposition: for any matrix V, we have $V = P(\Lambda + T)P^{\top}$ with P an orthogonal matrix, Λ a diagonal matrix containing the eigenvalues, and T a strictly lower-triangular matrix.[1] In short, T contains the interactions between the orthogonal column vectors of P (called *Schur modes*). P and T are obtained from orthogonalizing the non-orthogonal eigenbasis of V, and do not affect the eigenspectrum. As a recurrent matrix, T represents purely feed-forward structure that produces strictly transient dynamics impossible to produce in normal (orthogonal) matrices. In other words, if a normal and non-normal matrix share exactly the same eigenspectrum, their long-term dynamics will be equivalent, but their short-term activity can differ greatly. We revisit this distinction in §3.4. It was exploited by Goldman (2009); Hennequin et al. (2012) to analyze the decomposition of the activity of recurrent networks (in continuous time) into a normal part responsible for slow fluctuations, and a non-normal part producing fast, transient ones. How this mechanism propagates information was studied in Ganguli et al. (2008) for stochastic linear dynamics. The authors show analytically that

[1]When eigenvalues and eigenvectors are complex, P is unitary and $P^{\top}$ corresponds to conjugate transposition. However for any real V, it is possible to find an orthogonal (real) P with Λ being block-diagonal with 2×2 blocks instead of complex-conjugate eigenvalues.

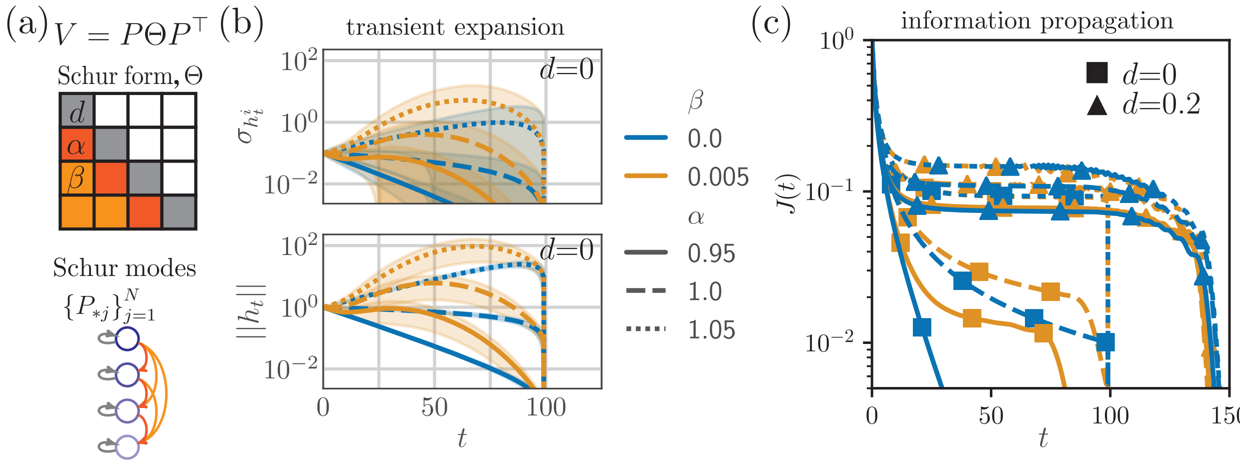


Fig. 1. *Benefits of non-normal dynamics.* (a) The Schur decomposition provides the lower-triangular Schur form (top). A feed-forward interaction coupling among Schur modes underlies non-normal dynamics (bottom). (b) Lower triangle generates stronger transients. Trajectories of standard deviation across hidden units (top) and norm of hidden state vector (bottom) obtained from the dynamics of Eq. equation 3.4.1. Lines and shading are average and standard deviation, respectively, over 10^3 initial conditions uniformly distributed on the unit hypersphere. Parameters: $d = 0$. (c) Fisher memory curves across α and β (see legend in (b)) as computed by Eq. A.2.2. Parameters: $d = 0$ (■), $d = 0.2$ (▲). $N = 100$ for (b) and (c).

non-normal dynamics can lead to extensive memory traces, as measured by the Fisher information of the trajectory ensemble parametrized by the input signal. To the best of our knowledge, an explicit demonstration and explanation of the benefits of non-normal dynamics for learning in RNNs is lacking, though see Orhan and Pitkow (2019) for similar ideas used for initialization.

3.4. Non-normal matrices are more expressive and propagate information more robustly than orthogonal matrices

We now outline the role of non-normal dynamics exploited here. To provide mathematically-grounded intuition for the benefit it provides to learning, we first present a generic RNN dynamics,

$$\begin{aligned} h_{t+1} &= \phi(V h_t + U x_{t+1} + b) \\ V &= P\Theta P^\top, \; \Theta = \Lambda + T \end{aligned} \tag{3.4.1}$$

where $h_t \in \mathbb{R}^N$ is the time-varying hidden state vector, ϕ is a nonlinear function, x_t is the input sequence projected into the dynamics via matrix U, and b is a bias (we omit the output for brevity). V is the matrix of recurrent weights, which in line with Section 3.3.2 we decompose into its lower-triangular Schur form Θ in Eq. equation 3.4.1, with P orthogonal and Θ lower triangular. Θ has two parts: a (block) diagonal part Λ, and a strictly lower triangular part T.[2]

The Schur decomposition maps the hard problem of controlling the directions of a non-orthogonal basis to the easier problem of specifying interactions between fixed orthogonal modes. It is important to highlight the fact that an orthonormalization of the eigenbasis is just a change

[2]We use the real Schur decomposition but a similar treatment can be derived for the complex case.

in representation and thus has no effect on the spectrum of V, which still lies on the diagonal of Λ. The triangular part T can thus be modified independently from the constraint (employed in orthogonal RNN approaches) that the spectrum have values equal or near 1. The ability to encode complex signals and then selectively recall past inputs is a basic requirement needed to solve many sequence-based tasks. Intuitively, the two features that allow systems to perform well in such tasks are:

(1) High dimensional activity to better encode complex input.

(2) Efficient signal propagation, to better learn long-term dependencies.

To illustrate how non-normal dynamics controlled by the entries in the lower triangle of T contribute to these two features, we consider a simplified linear case where $\phi(h_t) = h_t$ and Θ is parametrized as follows and illustrated in Fig. 1(a)):

$$(\Theta)_{i,j} = d\delta_{i,j} + \alpha\delta_{i,j+1} + \beta \sum_{2\leq k\leq i} \delta_{i,j+k} \,. \tag{3.4.2}$$

Here, diagonal entries are set to d, sub-diagonal entries to α, and the remaining entries in the lower triangle to β. By varying α and β we will show how the lower triangle in T enhances expressivity and information propagation.

3.4.1. Non-normality drives expressive transients

RNNs can be made more expressive with stronger fluctuations of hidden state dynamics. The dependence of hidden state variance on the values of Θ was studied in depth in Hennequin et al. (2012). Here, we present experiments where the RNN parametrized by Eqs. equation 3.4.1, equation 3.4.2 exemplifies some of those results. We numerically compute a set of trajectories over a sampled ensemble of inputs with $x_t > 0$ for $t = 0$ and 0 otherwise. Without loss of generality we assume a form of U and distribution of x_0 that leads to input-dependent initial conditions on the unit hypersphere in the space of h_t. For $\alpha = 0.95$,1.0,1.05, $\beta = 0, 0.005$ and $d = 0$, we see that trajectories of single units exhibit increasing large transients with increasing α and β, that abruptly end at $t = N$ (Fig. 1(b)). The latter is a result of the nilpotent property of a strictly triangular matrix: each iteration removes the top entries in each column until $\Theta^N = 0$. Computing ensemble statistics, we find that α contributes significantly to the strength of the exponential amplification, while β structures the shape of the transient. This ability of T to both exhibit amplification, and to control its shape, is what endows the Schur form Θ with expressivity (see Section 3.6.3 for empirical evidence in trained nnRNNs).

3.4.2. Non-normality allows for efficient information propagation

Propagation of information in a network requires feed-forward interactions. Perhaps the most simple example of a feed-forward structure is the local feed-forward chain (also called *delay-line* (Ganguli et al., 2008)), where each mode feeds its signal only to the next mode in the chain

($\alpha > 0$, $\beta = 0$, $d = 0$; see Fig. 1(a)). In this case, we denote Θ by Θ_{delay}. As a consequence, signals feeding the first entry of Θ_{delay} propagate down the chain and are amplified or attenuated according to the values of these non-zero entries. Moreover, inputs from different time steps do not interact with each other thanks to this ordered propagation down the line. In contrast, the signal is not propagated across modes for dynamics given by a purely (block) diagonal Θ. It instead simply decays within the mode into which it was injected on the timescale intrinsic to that mode, which can be much less than the $O(N)$ timescale of the chain.

To quantify the efficiency with which a RNN can store inputs, we follow and extend the approach of Ganguli et al. (2008). For a given scalar-valued input sequence, $x_t = s_t + \xi_t$, $t \in \mathbb{N}$, composed of signal s_t and injected noise ξ_t, the noise ensemble induces the conditional distribution, $P(h_{:t}|s_{:t})$, over trajectories of hidden states, $h_{:t}$, given the received input, $s_{:t}$, where $:t$ subscript is short hand for $(k: \ k \leq t)$. Taking the signal sequence $s_{:t}$ as a set of parameters of a model, and $P(h_{:t}|s_{:t})$ as this model's likelihood, the corresponding Fisher information matrix that captures how $P(h_{:t}|s_{:t})$ changes with the input $s_{:t}$ is,

$$\mathbf{J}_{k,l}(s_{:t}) = \left\langle -\frac{\partial^2}{\partial s_k \partial s_l} \log P(h_{:t}|s_{:t}) \right\rangle_{P(h_{:t}|s_{:t})} \qquad k,l \leq t\,. \tag{3.4.3}$$

The diagonal of this matrix, $J(t) := \mathbf{J}_{t,t}$ is called the Fisher memory curve (FMC) and has a simple interpretation: if a single signal s_0 is injected into the network at time 0, then $J(t)$ is the Fisher information that h_t retains about this single signal.

Ganguli et al. (2008) proved that the delay line Θ_{delay} achieves the highest possible values for the FMC when $k \leq N$: $J(k) = \alpha^k \frac{\alpha-1}{\alpha^{k+1}-1}$. However, we show (proof in §A.2) that any strictly lower-triangular matrix may approach the performance of a delay line:

Proposition 3.4.1. *Let $\Theta \in \mathbb{R}^{N\times N}$ be any strictly lower-triangular matrix with $\sqrt{\alpha}$ on the lower diagonal and let $T_{Gram} \in \mathbb{R}^{N\times N}$ be the triangular matrix associated with the Gram–Schmidt orthogonalization process of the columns of Θ (thus with only 1 on the diagonal). Then,*

$$J(k) \geq \frac{\alpha^k}{\sigma_{\max}^{2(N-1)}} \frac{\alpha - 1}{\alpha^{k+1} - 1}\,, \tag{3.4.4}$$

where $\sigma_{\max}$ is the maximum singular value of T_{Gram}.

Note that $\sigma_{\max} \geq 1$ and is equal to 1 for a delay line and close to 1 when Θ is close to a delay line. In Fig. 1(a) we present a class of matrices providing feed-forward interaction and compute the FMC of some matrices of this class in fig. 1(c). The delay line from Ganguli et al. (2008) with $\alpha > 1$ (shown to be optimal for $t \leq N$) retains the most Fisher information across time up to time step N, when the nilpotency of Θ erases all information. As expected from Prop.3.4.1, non-zero β, which endows the dynamics with expressivity (Fig. 1(b)), does not significantly degrade the information propagation of the delay line. Interestingly, the addition of diagonal terms ($d > 0$), i.e. Λ non-zero, helps to maintain almost optimal values of the FMC for $t < N$, while extending the memory beyond $t = N$, and thus outperforming the delay line with regards to the area under the

FMC (see Table 5 in the supplemental materials (SM)).

Together with the last section, these results demonstrate that non-normal dynamics, as parametrized through the entries in the lower triangle of Θ, provide significant benefits to expressivity and information propagation. What remains to show is how these benefits translate into enhanced performance of our nnRNN on actual tasks.

3.4.3. Non-normal matrix spectra and gradient propagation

While eigenvalues control the exponential growth and decay of matrix iterates, the spectral norm of these iterates may behave differently (Bengio et al., 1994). This norm is dominated by the modulus of the largest singular value of the matrix, and can thus differ from the eigenvalues' moduli. This is a subtle difference influencing gradient growth rates, and is explicitly revealed by different spectral constraints on RNNs. For comparison, a singular value decomposition (SVD) is presented in Zhang et al. (2018) with the same motivation as our Schur-decomposition: to maintain expressivity, whilst controlling a sprectrum (both using regularization). First note that, while constraining the eigenspectrum to the unit circle, non-normality implies having the largest singular value (and thus the spectral norm of the Jacobian) greater than 1. Hence, our approach mitigates gradient vanishing, but not necessarily gradient explosion. In this case however, gradients explode polynomially in time rather than exponentially (Pascanu et al., 2013b; Arjovsky et al., 2016). We provide a theorem (proof in SM§A.3) to establish this for triangular matrices.

Proposition 3.4.2. *Let $A \in \mathbb{R}^{n\times n}$ be a matrix such that $A_{ii} = 1$, $A_{ij} = x$ for $i < j$, and $A_{ij} = 0$ otherwise. Then for all integer $t \geq 1$ and $j > i$, we have $(A^t)_{ij} = p^{(t)}_{j-i}(x)$ is polynomial in x of degree at most $j - i$, where the coefficient of x^0 is zero and the coefficient of x^l is $O(\binom{t}{l})$ for $l = 1, 2, \ldots, j - i$ (which is polynomial in t of degree at most l).*

This reveals that gradient explosion in nnRNN with unit-norm eigenspectrum, if present, is polynomial and thus not as severe as the case where eigenvalues are larger than one (in which case the gradient explosion is exponential). In §3.6.3, we illustrate that relaxing unit-norm requirements for eigenvalues using regularization allows the optimizer to find a task-dependent trade-off, thus balancing control over exponential vanishing and polynomial exploding gradients respectively. See also SM§A.6 for gradient propagation measurements.

3.5. Implementing a non-normal RNN

The nnRNN is a standard RNN model where we parametrize the recurrent connectivity matrix V using its real Schur decomposition[3] as in Eq. equation 3.4.1, yielding the form:

$$V = P\left(\begin{bmatrix} \mathcal{R}_1 & 0 & \dots & 0 \\ 0 & \mathcal{R}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathcal{R}_{N/2} \end{bmatrix} + \begin{bmatrix} 0 & 0 & \dots & 0 \\ t_{2,1} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ t_{N,1} & t_{N,2} & \dots & 0 \end{bmatrix}\right) P^\top \tag{3.5.1}$$

with

$$\mathcal{R}_i(\gamma_i,\theta_i) \overset{\text{def}}{=} \gamma_i \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix},$$

where P is constrained to be an $N \times N$ orthogonal matrix. Each parameter above (including entries in P) is subject to optimization, as well as specific constraints outlined below. We note that although this parametrizattion uses the Schur form, we never explicitly compute Schur decompositions, which would be expensive and has stability issues.[4] Note that Eq. 3.5.1 can express any matrix V with a set of complex-conjugate pairs of eigenvalues.

During training, the orthogonal matrix P is optimized using the expRNN algorithm (Lezcano-Casado and Martínez-Rubio, 2019), a Riemannian gradient descent-like algorithm operating inside the Stiefel manifold of orthogonal matrices. We note that other suitable orthogonality-preserving algorithms could be used here (see §3.3) but we found expRNN to be the fastest and most stable. Instead of rigidly enforcing that eigenvalues be of unit norm, we found relaxing this constraint to be helpful. We therefore allow γ_i to be optimized but add a strong L2 regularization constraint $\delta\|\|1-\gamma_i\|\|_2^2$ to encourage them to be to close to 1. The hyperparameter δ is tuned differently for each task (see SM§A.1) but remains high overall, indicating only mild departure from unit-norm eigenvalues. Both θ_i and t_{ij} are freely optimized via automatic differentiation. The non-linearity we use is *modReLU*, as defined in Arjovsky et al. (2016); Helfrich et al. (2018). We initialize P as in Lezcano-Casado and Martínez-Rubio (2019) using Henaff or Cayley initialization scheme (Henaff et al., 2016), θ_i from a uniform distribution between 0 and 2π, and γ_i's are initialized at 1.

We reiterate that the set of orthogonal matrices is a subset of all the connectivity matrices covered by nnRNN, by setting all γ's to 1, and $T = 0$. Consequently the connectivity matrix in nnRNN has more parameters than an orthogonal matrix: $N(N-1)/2$ for T, and $N/2$ γ_i's, which in total gives roughly $N^2/2$ more parameters than orthogonal RNNs.

The forward pass of the nnRNN has the same complexity as that of a vanilla RNN, that is $O(Tn^2)$, for a hidden state of size n and a sequence of length T. The backward pass is similarly $O(Tn^2)$ plus the update cost of P, in addition to a once-per-update cost of $O(n^3)$ to combine the Schur parametrization via matrix multiplication. Importantly, the nnRNN leverages

[3]See discussion for more details about complex-valued implementations.

[4]See §A.4 for a discussion.

any orthogonal/unitary optimizer for P which have complexities ranging from $O(n \log n)$ to $O(n^3)$ at each update, with their own advantages and caveats (see §3.3.1). We chose the expRNN scheme, which is $O(n^3)$ in the worst case, but has fast run-time in practice.

3.6. Numerical experiments

In this section, we test the performance of our nnRNN on various sequential processing tasks. We have two goals:

(1) Establish the nnRNN's ability to perform as well as orthogonal RNNs on tasks with pathologically long term dependencies: the copy task and the permuted sequential MNIST task.
(2) Demonstrate improved performance over orthogonal RNNs on a more realistic task requiring ongoing computation and output: the Penn Tree Bank character-level benchmark.

We compare our nnRNN model to the following architectures: vanilla RNN (RNN), the orthogonally initialized RNN (RNN-orth) (Henaff et al., 2016), the Efficient Unitary RNN (EURNN) (Mhammedi et al., 2017), and the Exponential RNN (expRNN) (Lezcano-Casado and Martínez-Rubio, 2019). Our goal is to establish performance for non-gated models, but we include LSTM (Hochreiter and Schmidhuber, 1997) for reference. For comparison, models are separately matched in the number of hidden units and number of parameters. Every training run was tuned with a thorough optimization hyper-parameter search. Model training and task setup are detailed in the SM§A.1.

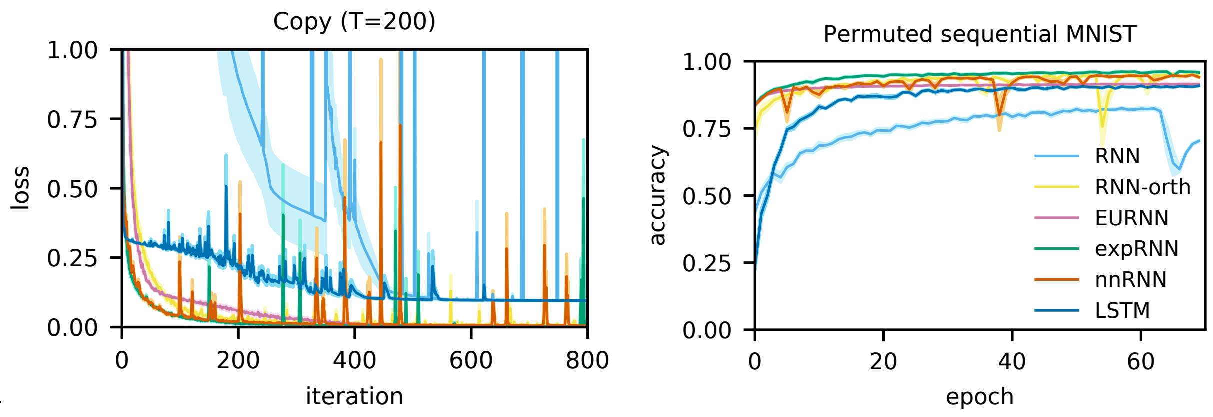


Fig. 2. Holding the number N of hidden units constant, model performance is plotted for the copy task (T=200, left; cross-entropy loss; $N \sim 128$) and for the permuted sequential MNIST task (right; accuracy; $N \sim 512$). Shading indicates one standard error of the mean.

3.6.1. Copy task & Permuted sequential MNIST

The copy task, introduced in Hochreiter and Schmidhuber (1997), requires that a model read a sequence of inputs, wait for some delay T (here we use $T = 200$), and then output the same sequence. Fig. 2 shows the cross entropy of each tested with $N = 128$ hidden units. We see little difference if we match the number of parameters with $\sim$ 18.9K (see Fig. 1 in SM§A.1.2). For reference, a model that simply predicts a constant set of output tokens for every input sequence is expected to achieve a *baseline* loss of 0.095. As shown in Henaff et al. (2016), an orthogonal RNN is an optimal solution for the copy task. Indeed, the LSTM struggled to solve the task and RNN failed completely, unlike all orthogonal RNNs who learn to solve at very high performance very quickly. The proposed nnRNN matched the performance of orthogonal RNNs, as well as best training timescales.

Sequential MNIST (Le et al., 2015) requires a model to classify an MNIST digit after reading the digit image one pixel at a time. The pixels are permuted in order to increase the time delay between inter-dependent pixels, making the task harder. Fig. 2 shows mean validation accuracy of each tested model with with $N = 512$ hidden units (see Fig. 1 in SM§A.1.2 for parameter match). As with the copy task, the nnRNN matches orthogonal RNNs in performance, whereas RNN and LSTM show lesser performances.

3.6.2. Penn Tree Bank (PTB) character-level prediction

Character level language modelling with the Penn Treebank Corpus (PTB) (Marcus et al., 1993) consists of predicting the next character at each character in a sequence of text (see SM§A.1.3 for test accuracy). We compare the performance of different models on this task in Table 1 in terms of test mean bits per character (BPC), where lower BPC indicates better performance. We compare truncated backpropagation through time over 150 time steps and over 300 time steps.

	Test Bit per Character (BPC)			
	Fixed # params (∼1.32M)		Fixed # hidden units ($N = 1024$)	
Model	$T_{PTB} = 150$	$T_{PTB} = 300$	$T_{PTB} = 150$	$T_{PTB} = 300$
RNN	2.89 ± 0.002	2.90 ± 0.0016	2.89 ± 0.002	2.90 ± 0.002
RNN-orth	1.62 ± 0.004	1.66 ± 0.006	1.62 ± 0.004	1.66 ± 0.006
EURNN	1.61 ± 0.001	1.62 ± 0.001	1.69 ± 0.001	1.68 ± 0.001
expRNN	1.49 ± 0.008	1.52 ± 0.001	1.51 ± 0.005	1.55 ± 0.001
nnRNN	**1.47 ± 0.003**	**1.49 ± 0.002**	**1.47 ± 0.003**	**1.49 ± 0.002**

Table 1. PTB test performance: Bit per Character (BPC), for sequence lengths $T_{PTB} = 150, 300$. Two comparisons across models shown: fixed number of parameters (left), and fixed number of hidden units (right). Error range indicates standard error of the mean.

In contrast to the copy and psMNIST tasks (see §3.6.1), the PTB task requires online computation across several inputs received in the past. Furthermore, it is a task that demands an output from the network at each time step, as opposed to a prompted one. These ingredients are not particularly well-suited for orthogonal transformations since it is not enough to simply keep inputs in memory or integrate input paths to a classification outcome, the network must transform past inputs to compute a probability distribution. Gated networks are well-suited for such tasks, and we could get an LSTM with $N = 1024$ hidden units to achieve 1.37 ± 0.003 BPC (see §3.7 for a discussion).

Importantly, without the use of gating mechanisms, our nnRNN outperformed all other models we tested. To our knowledge, it also surpasses all reported performances for other non-gated models. While the performance gap to expRNN (the state-of-the-art orthogonal RNN) is modest for equal number of parameters and shorter time scale ($T_{PTB} = 150$), it appreciatively improves for $T_{PTB} = 300$. Where the nnRNN shines is for equal numbers of hidden units, where the performance gap to expRNN is much greater. This suggests two things (i) the nnRNN improves propagation of *meaningful* signals over longer time scales, and (ii) it's connectivity structure provides superior expressivity for a fixed number of neurons, a desirable feature for efficient model deployment. In the next section, we explore the structure of trained nnRNN weights to illustrate that the mechanisms responsible for this performance gain are consistent with the arguments presented in §3.4.

3.6.3. Analysis of learned connectivity structure

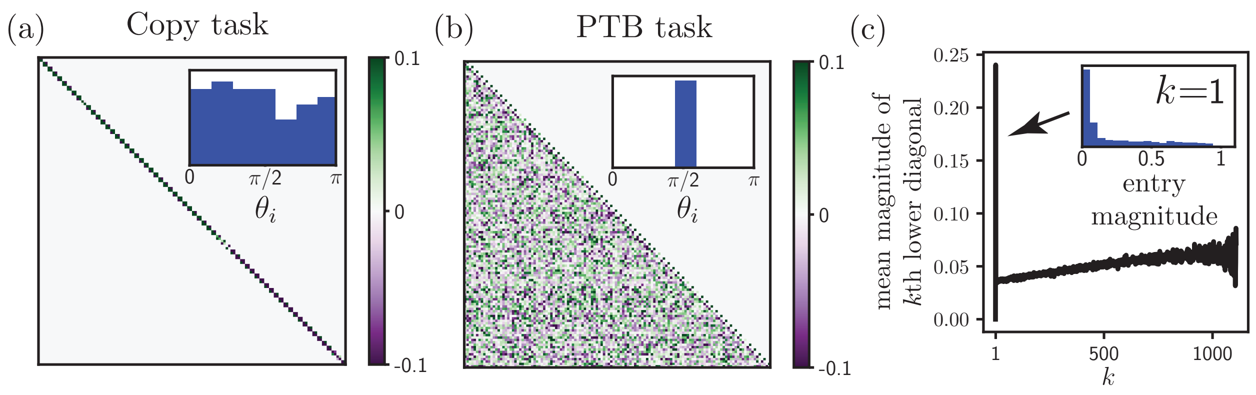


Fig. 3. *Learned Θs show decomposition into Λ and T.* Elements of learned Θ matrix entries for copy task (a) are concentrated on the diagonal, and distributed in the lower triangle for the PTB task (b). Insets in (a) and (b) show the distribution of eigenvalues angles θ_i (c.f. Eq. 3.5). (c) The mean magnitude of entries along the kth sub-diagonal of the lower triangle in (b) shows both a delay-line and lower triangle component. Inset: the distribution of entry magnitudes along the delay line is bimodal from its two contributions: the cosine of uniformly distributed angles, and the relatively small, but significant pure delay line entries.

To validate the theoretical arguments in favor of non-normal dynamics presented in §3.4, we take a look at the connectivity structure that emerges from our training procedure (see §3.5). Fig. 3 (and 2 in SM§A.5) shows the triangular Schur form $\Theta = \Lambda + T$ of the recurrent connectivity matrix $V = P\Theta P^{\top}$, at the end of training. For the copy task, Θ is practically composed of 2×2 rotation blocks along its diagonal (i.e. $T = 0$). This indicates that the learned dynamics are normal, and orthogonal. In contrast, for the PTB task we find that the lower triangular part T shows a lot of structure, indicating that non-normal transient dynamics are used to solve the prediction task.

The distributions of elements of T away from the diagonal highlights the nature of the tasks. The network distributes the angles roughly uniformly in the case of the copy task, consistent with the explicit optimal solution that involves such a distribution of rotations (Henaff et al., 2016). For the PTB task however, the angles strongly align, promoting the delay-line motif in Θ, shown in §3.4 to be optimal for the information propagation useful for character prediction. This is more clearly demonstrated by the mean absolute value of entries away from the diagonal, shown in Fig. 3. The rest of the triangle also shows structure, consistent with our proof that the lower triangle and delay line can jointly contribute to information propagation.

In summary, these findings indicate that when tasks are well-suited for isometric transformations (e.g. storing things in memory for later recall) the nnRNN easily learns to eliminate non-normal dynamics and restricts itself to the set of orthogonal matrices. Moreover, it does so without any penalty on learning speed, as shown in Fig. 2. However, when tasks require online computations, non-normal dynamics come into play and enable transient activity to be shaped for computations.

Lastly, as already discussed in §3.4.3, the expressivity afforded by non-normality must come with a trade-off between maintaining the eigen and singular spectra "close" to the unit circle, balancing control over exponential vanishing and polynomial exploding gradients respectively. This fact remains true for any parametrization of non-normal matrices, including the SVD used in spectral RNN Zhang et al. (2018). The nnRNN is naturally suited to target this balance by explicitly allowing regularization over normal and non-normal parts of a matrix, and enabling the optimizer to find that trade-off. This explains why we find that allowing eigenvalues to deviate slightly from the unit circle throughout training (regularization on γ), along with weight decay for the non-normal part, yields the best results with most stable training. Further evidence of this balancing mechanism is found in trained matrices (see Fig. 3). For the PTB task, non-normal structure emerges and the mean eigenvalue norm is balanced at $\bar{\gamma} \sim 0.958$. In contrast for the copy task, matrices remain normal and $\bar{\gamma} \sim 1$. See SM§A.6 for additional experiments with fixed γ further outlining their role in this trade-off.

3.7. Discussion

With the nnRNN, we showed that augmenting orthogonal recurrent connectivity matrices with non-normal terms increases the flexibility of a recurrent network. We compared the nnRNN's performance to several other recurrent models on distinct tasks; some that are well suited for orthogonal RNNs, and another that targets their limitations. We find that non-normal structure affords two distinct improvements for nnRNNs:

(1) *Preservation of advantages from purely orthogonal RNNs* (long-term gradient propagation; fast learning on tasks involving long-term memory)

(2) *Compared to orthogonal RNNs, increased expressivity on tasks requiring online computations thanks to transient dynamics.*

To better understand why this is, we derived analytical expressions that outline the role of non-normal dynamics that were corroborated by an analysis of nnRNN connectivity structure after training. Importantly, the nnRNN leverages existing optimization algorithms for orthogonal matrices with increased scope, all the while retaining learning speed.

The principal contribution of this paper is not to report major gains in performance as measured by tests, but rather to convincingly outline a promising novel direction for spectrally constrained RNNs. This spans the expressivity and ability to handle long-term dependencies of orthogonal RNNs on one hand, and completely unconstrained RNNs on the other. The nnRNN is a first step toward a trainable RNN parametrization where regularization over the eigenspectrum is readily available while conserving the flexibility of arbitrary eigenbases. This allows explicit control over quantities with direct impact on gradient propagation and expressivity, providing a promising RNN toolbox. Unlike the orthogonal RNNs present in our tests, which have benefited over the years from a series of algorithmic improvements, our nnRNN is basic in its implementation, and presents a number of areas for direct improvement. These include (i) using a complex-valued parametrization as in Arjovsky et al. (2016), (ii) exploring better initializations, and (iii) identifying helpful regularization schemes for the non-normal part. Beyond these, we should mention that the Schur decomposition presents implicit instabilities which can jeopardize training when eigenbases become degenerate (see SM§A.4). Simple perturbation schemes to prevent this should greatly improve performance.

Finally, we acknowledge that on a number of time-dependent tasks, gated recurrent networks such as the LSTM or the GRU (Jing et al., 2019) have clear advantages (see also Tallec and Ollivier (2018) for a derivation of gated dynamics from first principles). Building on these, there is promising evidence that combining orthogonal connectivity with gates can greatly help learning (Jing et al., 2019). This further motivates the development of spectrally constrained recurrent architectures to be combined with gating, thereby optimizing the efficiency of gradient propagation and expressivity with both explicit mechanisms, and implicit structure. Ongoing work in this direction is under way, leveraging our nnRNN findings.

Chapter 4

Untangling tradeoffs between recurrence and self-attention in neural networks

This chapter is based on a NeurIPS 2020 paper. (Kerg et al., 2020)

4.1. Prologue to the Second Article

4.1.1. Article Details

Title: Untangling tradeoffs between recurrence and self-attention in neural networks

Authors: Giancarlo Kerg[1,2,], Bhargav Kanuparthi [1,2,*], Anirudh Goyal [1,2], Kyle Goyette [1,2,3], Yoshua Bengio[1,2,4], Guillaume Lajoie[1,2,5]

[*] Indicates first authors. Ordering determined by coin flip.

Affiliations:

1: Mila - Quebec AI Institute, Canada

2: Université de Montréal, Département d'Informatique et Recherche Opérationelle, Montreal, Canada

3: Université de Montréal, CIRRELT, Montreal, Canada

4: CIFAR senior fellow

5: Université de Montréal, Département de Mathématiques et Statistiques, Montreal, Canada

4.1.2. Contributions

Giancarlo Kerg derived the entire formal analysis including all theorems, propositions and proofs. Further Giancarlo Kerg also came up with the relevancy screening mechanism, after plotting the attention heat diagrams. Bhargav Kanuparthi ran all large scale experiments, did all hyperparameter tuning and produced the majority of the code. Giancarlo Kerg led the writing of the article.

4.1.3. Paper Abstract

Attention and self-attention mechanisms, are now central to state-of-the-art deep learning on sequential tasks. However, most recent progress hinges on heuristic approaches with limited understanding of attention's role in model optimization and computation, and rely on considerable memory and computational resources that scale poorly. In this work, we present a formal analysis of how self-attention affects gradient propagation in recurrent networks, and prove that it mitigates the problem of vanishing gradients when trying to capture long-term dependencies by establishing concrete bounds for gradient norms. Building on these results, we propose a relevancy screening mechanism, inspired by the cognitive process of memory consolidation, that allows for a scalable use of sparse self-attention with recurrence. While providing guarantees to avoid vanishing gradients, we use simple numerical experiments to demonstrate the tradeoffs in performance and computational resources by efficiently balancing attention and recurrence. Based on our results, we propose a concrete direction of research to improve scalability of attentive networks.

4.2. Introduction

We live in a world where most of the information takes a sequential form, largely because it is delivered over time. Performing computations on streams of sequential inputs requires extracting relevant temporal dependencies and learning to recognize patterns across several timescales. Humans can effortlessly make associations relating events stored in memory which are far from each other in time and thus, capture long-term dependencies.

Historically, recurrent neural networks (RNNs) have been the deep network architecture of choice for this type of task since, just like neural circuits in the brain, they enable *dynamics* that can be shaped to interact with input streams. However, RNNs (including gated RNNs Schmidhuber and Hochreiter (1997); Cho et al. (2014b)) still struggle with large timescales as their iterative nature leads to unstable information propagation Bengio et al. (1994); Pascanu et al. (2013a); Schmidhuber and Hochreiter (1997); Hochreiter (1991).This is because most standard RNNs rely on their current state h_t, a vector of fixed dimension, to represent a summary of relevant past information. Indeed, Bengio et al. (1994) showed that without making additional assumptions, storing information in a fixed-size state vector in a stable way necessarily leads to vanishing gradients when back-propagating through time (see also (Hochreiter, 1991)). Several attempts have been made to augment RNN dynamics with external memory to mitigate these issues Sukhbaatar et al. (2015); Graves et al. (2014); Santoro et al. (2018); Graves et al. (2016), but it is only recently that access to externally stored information has become effective with the introduction of *attention*, and more particularly *soft attention* mechanisms Bahdanau et al. (2014). Attention provides a way by which a system can dynamically access past states and inputs across several timescales, bypassing the need of sequential propagation and ignoring irrelevant information (or distractor information). There is substantial empirical evidence that attention, especially *self-attention* (Vaswani et al. (2017); Ke et al. (2018)), is very helpful to improve learning and computations over long-term dependencies. However, to the best of our knowledge, there is currently limited understanding of gradient scaling properties in the presence of attention. Moreover, attending over long sequences requires to hold inputs and/or past states in memory, a process that typically scales quadratically with sequence length.

Much like work from the '90s established formal results for gradient exploding/vanishing in deep/recurrent networks Bengio et al. (1994), we believe it is crucial to establish similar theoretical tools for attention mechanisms, as these methods are under intense development where scalability and complexity are important issues. In this paper, we contribute to this direction with a *formal analysis of gradient propagation in self-attentive systems which precisely quantify trade-offs between recurrence and attention*, offering valuable guarantees for attention mechanism development. Concretely exploiting these theorems, we propose a simple family of screening mechanisms to *maximally reduce computational complexity and memory usage, while simultaneously maintaining good gradient propagation over large time scales*. Using simple tasks for their ease of interpretation, and their variety of computational demands, we illustrate the efficacy of this approach in numerical experiments.

The remainder of this paper is as follows. In Section §4.3, we give a brief outline of related cognitive processes and neural network mechanisms. In §4.4, we present our central results: asymptotic guarantees for gradient propagation in self-attentive recurrent networks. To illustrate how to exploit these guarantees, in §4.5, we showcase a simple *relevancy screening mechanism* that aims to efficiently consolidate relevant memory, reducing the size of the computational graph from quadratic to linear in sequence length. Finally, in §4.6, we compare various recurrent and attention models with our proposed relevancy screening mechanism on a series of simple numerical experiments, while, in §4.7, we analyze their gradient propagation properties together with their GPU usage.

4.3. Background

To perform complex tasks, our brains rely on mechanisms to encode and retrieve information to and from memory (Zacks et al., 2007; Radvansky and Zacks, 2017).

In contrast, standard RNNs follow rigid sequential dynamics as they are parametric i.e with a fixed-size state vector. Self-attention methods can overcome this limitation by giving access to previous past states for computing the next state. For the sake of the discussion, we call such RNNs, which are augmented by the memory of the past states as *semi-parametric* RNNs. The use of soft-attention (Bahdanau et al., 2014) in such models has improved performance on many tasks such as reading comprehension, abstractive summarization, textual entailment and learning task-independent sentence representations (Parikh et al., 2016; Lin et al., 2017; Paulus et al., 2017; Yang et al., 2019) as well as in the self-supervised training of extremely large language models (Devlin et al., 2018; Radford et al., 2019) due to their ability to handle long-term dependencies.

Intriguingly, the most notable advances in the use of attention is in purely attention-based systems such as the Transformer Vaswani et al. (2017), which completely foregoes recurrence and inspired some of the work listed above. While the performance of these systems is impressive, their memory and computation requirements grows quadratically with the total sequence length.

To address this issue, many variants that aim to "sparsify" the attention matrix have been proposed. Notably, Ke et al. (2018) developed the Sparse Attentive Backtracking model (SAB), a self-attentive Long Short-Term Memory network (LSTM) Hochreiter and Schmidhuber (1997) that leverages sparsity by selecting only the top-k states in memory based on an attention score, propagating gradients only to those chosen hidden states. Recently, Zhao et al. (2019) propose to use a similar top-k attention, and Child et al. (2019) introduce sparse masks which attends to roughly $\sqrt{n}$ locations in memory, implementing explicit selection methods for Transformers. Reformer models (Kitaev et al., 2020) replace the dot-product attention by locality-sensitive hashing, changing its complexity from $O(T^2)$ to $O(T)$, where T is the sequence length. Finally, TransformerXL (Dai et al., 2019) enables learning dependencies beyond a fixed length without disrupting temporal coherence and has resulted in state of the art performance in language models.

Still, most of these approaches naively sub-sample input streams for memory storage. Our brains on the other hand, seem to select relevant information from the recent past to commit to long term memory based on their relevancy, a process often referred to as memory consolidation Alberini et al. (2013). Attempts at mimicking this sparse temporal selectivity process has shown great promise in a variety of contexts (Graves et al., 2014; Munkhdalai et al., 2019; Harutyunyan et al., 2019; Goyal et al., 2019), and our work aims to formalize this idea for self-attentive recurrent networks.

4.4. Theoretical analysis of gradient propagation

In this section, we analyze the influence of self-attention onto gradient propagation in recurrent networks with self-attention. In order to do so let us first recall the equations of a recurrent neural network with self-attention. We note that even though we are using "vanilla RNNs" in the formulations of our results, any recurrent network can take its place (see §4.6 where we use LSTMs in the experiments). Let $x_t \in \mathbb{R}^m$ be the input and $h_t \in \mathbb{R}^n$ be the hidden state at time step t, satisfying the update equation for all $t \geq 1$,

$$h_{t+1} = \phi(V s_t + U x_{t+1} + b) \tag{4.4.1}$$

$$s_t = f(h_t, c_t) \tag{4.4.2}$$

where ϕ is a non-linearity, $f : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$, $V \in \mathbb{R}^{n\times n}$, $U \in \mathbb{R}^{n\times m}$, $b \in \mathbb{R}^n$ and $c_t = \alpha_{1,t}h_1 + \alpha_{2,t}h_2 + \ldots + \alpha_{t,t}h_t$ with $\alpha_{i,t} := \frac{\exp\{(e_{i,t})\}}{\sum_{j=1}^{t}\exp\{(e_{j,t})\}}$ and $e_{i,t} := a(s_{t-1}, h_i)$, where $a : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ is the attention *alignment function*. Throughout, we assume training is done via gradient descent of a cost function L using back-propagation.

Oftentimes, one uses $s_t = f(h_t,c_t) = h_t + c_t$ (but concatenation would be more general), and for all $t > 1$ and $1 \leq j \leq t$, $a(s_{t-1},h_j) = v_a^T \cdot \tanh(W_a \cdot s_{t-1} + U_a \cdot h_j)$, where $v_a \in \mathbb{R}^n$, and $W_a,U_a \in \mathbb{R}^{n\times n}$. The latter choice for alignment function is sometimes referred to as "additive self-attention" and was used in the original paper Bahdanau et al. (2014). We emphasize that the results presented in this section hold independently of the choice of the alignment function as, we will discuss later in this section. Lastly, while results presented below are relatively succinct, their derivations are involved and we refer the interested reader to the Appendix for detailed proofs.

4.4.1. Preliminaries

Our goal in this section is to establish formal propagation rules for a system where multiple paths of signal propagation are possible. We would like to understand the relationship between skip connections (those coming from self-attention) and recurrent connections, as well as how the interplay between the two leads to good gradient propagation. In order to achieve this, we seek to analyze the asymptotic behaviour of $\|\nabla_{h_t} L\| = \| \left(\frac{ds_T}{dh_t}\right)^T \nabla_{s_T} L\|$, as $T \to \infty$. We accomplish this by decomposing $\nabla_{h_t} L$ with respect to all possible gradient backpropagation paths, or in other words, by decomposing $\frac{ds_T}{dh_t}$ into sums of products of Jacobian matrices corresponding to those gradient paths, using Proposition B.1.7.

Proposition 4.4.1. *For all $t \geq 1$, $k \geq j \geq 0$, $k' \geq 0$, let $E_{k'}^{(t)} = \frac{\partial s_{t+k'}}{\partial h_t}$, and $F_{k+1,j}^{(t)} = \frac{\partial s_{t+k+1}}{\partial h_{t+j+1}} \cdot J_{t+j} + 1_{j=k} \cdot \frac{\partial s_{t+k+1}}{\partial s_{t+k}}$, with J_{t+j} the Jacobian matrix $\frac{dh_{t+j+1}}{ds_{t+j}}$. Then, we have*

$$\frac{ds_{t+k}}{dh_t} = \sum_{s=0}^{k} \bar{\xi}_{0:k}^{(t)}(s) \tag{4.4.3}$$

where for all $s \geq 1$, $\bar{\xi}_{0:k}^{(t)}(s) = \sum_{0 \leq i_1 < \ldots < i_s < k} F_{k,i_s}^{(t)} \cdot F_{i_s,i_{s-1}}^{(t)} \cdot \ldots \cdot F_{i_2,i_1}^{(t)} \cdot E_{i_1}^{(t)}$ and where $\bar{\xi}_{0:k}^{(t)}(0) = E_k^{(t)}$.

(Proof in Appendix B.1.2, Proposition B.1.7)

Here, each term $F_{k,i_s}^{(t)} \cdot F_{i_s,i_{s-1}}^{(t)} \cdot \ldots \cdot F_{i_2,i_1}^{(t)} \cdot E_{i_1}^{(t)}$ corresponds to exactly one gradient path involving exactly $s+1$ skip connections going from t to $t+k$, via the s hidden states $h_{t+i_s+1}, \ldots, h_{t+i_1+1}$. In particular, $\bar{\xi}_{0:k}^{(t)}(s)$ is the sum of all terms containing exactly s Jacobian matrices J, and thus the larger s is, the more $\bar{\xi}_{0:k}^{(t)}(s)$ is prone to vanishing.

Intuition: In order to find paths that are not vanishing as $T \to \infty$, we want to find gradient paths with: **(i)** a bounded path length s so that the number of Jacobian matrices involved in the product is limited. **(ii)** attention scores that are sufficiently bounded away from 0, so that the resulting product of attention scores is sufficiently bounded away from 0 as well. In order to see how exactly the attention weights come into play via matrices E and F, we refer to Proposition B.1.12 from Appendix B.1.2.

Defintions: Let us fix an integer $t \geq 1$, an integer $s \in \{1,2,\ldots,T-t\}$, and an ordered set of indices $i_1,i_2,\ldots,i_s \in \{0,1,\ldots,T-t-1\}$, verifying $i_1 \leq i_2 \leq \ldots \leq i_s$.

- For sequences $\{g(T)\}_{T\geq 1}$ and $\{f(T)\}_{T\geq 1}$, we say that $\underline{f(T) = \Omega(g(T))}$ if there exists positive constants c and T_0 such that $f(T) \geq c \cdot g(T)$ for all $T \geq T_0$.
- At time t, we call a past hidden state h_i a *relevant event* if the attention weight $\alpha_{i,t}$ is sufficiently bounded away from zero.
- We call the s-tuple $(i_1,i_2,\dots,i_s)$ a *dependency chain γ of depth s*, as it induces a gradient backpropagation path going via the s hidden states $h_{t+i_s+1},\dots,h_{t+i_1+1}$.
- We call *dependency depth* the *smallest* depth among all dependency chains where the product of the corresponding attention scores is $\Omega(1)$ as $T \to \infty$.

The central message is that *if the dependency depth is bounded from above and sufficiently small, then we mitigate gradient vanishing*. As we see below, task structure introduces different ways in which this may happen. We now present a formal treatment for specific cases, and lay the groundwork to take advantage of this structure during learning.

4.4.2. Uniform relevance case

Suppose each state has equal relevance in some task. What can be said about gradient propagation? This translates to having each attention weight $\alpha_{i,t} = 1/t$ for all $t \geq i \geq 1$. We then have dependency chains of depth 1 but with vanishing rate $\Omega(1/T)$, as formalized in the following theorem (cf. §B.1.3)

Theorem 4.4.2. *Let h_t be the hidden state at time t of a vanilla RNN with uniform attention, under mild assumptions on the connectivity matrix V, and trained with respect to a loss L, then if T is the total sequence length, we have*

$$\|\nabla_{h_t} L\| = \Omega(1/T) \tag{4.4.4}$$

as $T \to \infty$. (proof in Appendix §B.1.3, Theorem B.1.34)

This corresponds to the case where all past events contribute equally error signals. We also note that this result is independent of the choice of the alignment function a (cf. Remark B.1.19 in the Appendix §B.1.3).

Intuition: As a "worst case scenario" Theorem 4.4.2 reveals the true trade-off of early self-attentive recurrent networks Bahdanau et al. (2014). On one hand, the lower bound obtained on gradient norm is substantially better than in a vanilla RNN without attention, where vanishing happens at an exponential rate, as opposed to a polynomial one here. This situation does not lend itself to sparse memory approaches as all events need to be held in memory, thus conserving quadratically scaling complexity. In contrast many inputs and tasks do not call for uniform attention and naturally lend themselves to sparse dependency paths for computation. The next case treats this situation. Nevertheless, this uniform attention bound is applicable in practice for two reasons: (1) typically, attention weights are initialized uniformly, and early training may result in gradients best described by this regime. (2) We experimentally verified that gradient propagation remains stable throughout training for a fully self-attentive RNN, where this bound is relevant, see Fig 2 (§4.7).

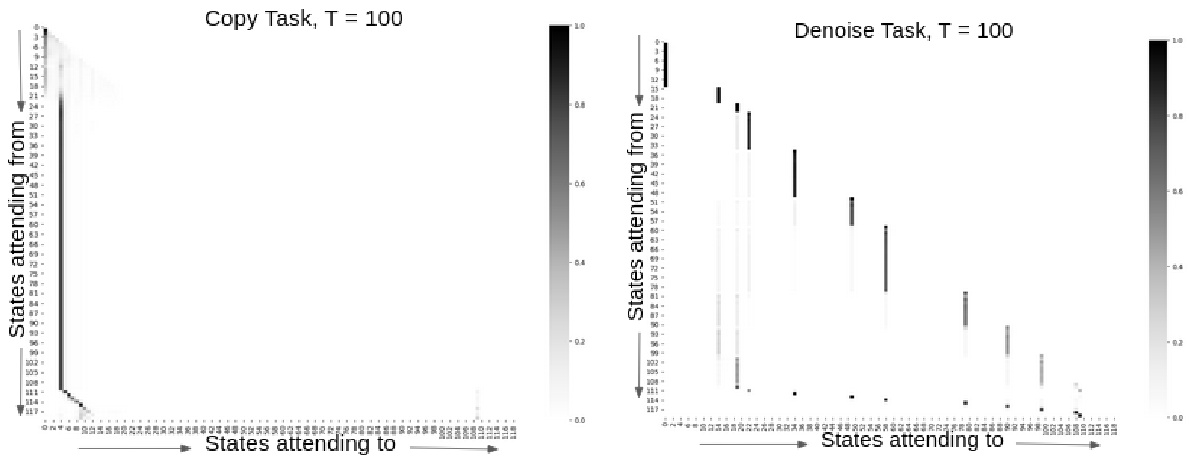


Fig. 1. *Magnitude of attention weights between states in a trained, fully recurrent and fully attentive model* (Bahdanau et al. (2014)). Each pixel in the lower triangle corresponds to the attention weight of the skip connection departing from the the state marked on the y-axis to the state marked on the x-axis. Left shows Copy task, right shows Denoise task. Task details in § 4.6

4.4.3. Sparse relevance case with bounded dependency depth

Now let us look at a more realistic case where only a sparse subset of past states are relevant for the task at hand, and the gradient needs to access those states efficiently for good learning. Figure 1 illustrates this scenario by showing the attention scores for two input examples computed by a simple self-attentive model Bahdanau et al. (2014), trained on Copy and Denoise tasks respectively (see § 4.6). This structure introduces the possibility to impose sparsity in the computational graph, and to limit memory use. With these constraints in mind, the goal is to engineer dependency chains that enable best gradient propagation between these relevant events.

Notation: We consider a κ-sparse attention mechanism of dependency depth d.

- *Sparsity coefficient*: $\kappa \geq 1$. Borrowing from the SAB model Ke et al. (2018), at each time step, attention is allowed at most κ relevant events from the past. That is, for any t there are at most κ indices i such that $\alpha_{i,t} \neq 0$, which gives rise to a sparse temporal segmentation via the most relevant events.
- *Maximal dependency depth*: d. This is the maximal dependency depth across all time steps t.

Theorem 4.4.3. *Let h_t be the hidden state at time t of a vanilla RNN with κ-sparse uniform attention mechanism of maximal dependency depth d, and under mild assumptions on the connectivity matrix V, then*

$$\|\nabla_{h_t} L\| = \Omega(1/\kappa^d) \tag{4.4.5}$$

as $T \to \infty$. (proof in Appendix §B.1.4, Theorem B.1.39)

Similarly to uniform case, Theorem 4.4.3 is independent of the choice of the alignment function a (cf. remark B.1.37 in the appendix).

Intuition: Notice the dependency depth d affects the lower bound exponentially, while κ affects it polynomially. In other words, the number of relevant events attended to at each time step contributes far less to gradient vanishing than the number of events in the longest dependency chain. Theorem 4.4.3 outlines the tradeoff between computational complexity as $T \to \infty$ and gradient propagation when balancing attention and recurrence. Attending directly to many relevant past events reduces d and ensures good gradients at the expense of the complexity cost associated with storing past events and computing attention scores (the strategy employed by Transformers Vaswani et al. (2017)). On the other hand, enforcing small sparsity coefficient κ helps keep computational complexity low ($O(\kappa T)$), but forces the error gradient through recurrent paths, thereby augmenting the dependency depth d and degrading gradient signal strength. Importantly, κ and d co-vary in ways that depend on the task's underlying relevancy structure, a point that is explained in detail in Appendix §B.3 (See Fig 1). In the extreme case where κ and d are assumed to be bounded, we have $\Omega(1/\kappa^d) = \Omega(1)$, and thus we mitigate gradient vanishing. In other situations where κ and d scale in other ways, an explicit sparsification strategy can be derived by exploiting Theorem 4.4.3, as we illustrate in the next section.

4.5. Relevancy screening mechanism

Equipped with the results from the previous section, we wish to refine heuristics that strike a balance between good gradient propagation and computational/memory complexity. Building on the SAB model Ke et al. (2018), we remark that although sparse attention attends to the top-κ events at any point in time, attention scores must be computed on all events stored in memory to extract the κ best ones. Thus, the resource bottleneck is not controlled by κ, but rather by the number of stored events in memory. In SAB, there is a naive attempt to control this number by only recording network states at each 10 time steps. However, this reduces the size of the computational graph only by a constant factor, but retains $O(T^2)$ complexity. In contrast, Theorem 4.4.3 tells us that the only important events to conserve for good gradient propagation are the *relevant* ones (also see Remark B.1.17 in Appendix §B.1.2). Thus, we propose to reduce complexity while maintaining good gradient propagation by selectively storing events that are predicted to be relevant in the future, using a *relevancy screening mechanism*.

The idea is simple: devise a screening function $C(i)$ which estimates the future relevance of h_i, and store selected events in a *relevant set* $R_t = \{h_i | i < t \wedge C(i) = True\}$ for future attention. In principle, one can explicitly control how R_t grows with t, thus mitigating the complexity scaling outlined above.

Here, $C(i)$ could take many forms, the best of which depends on task structure. In what follows, we present an example screening mechanism meant to showcase the lessons learned from Theorem 4.4.3, but we refer the interested reader to §4.8 for further possibilities.

We take inspiration from memory consolidation principles in human cognition Alberini et al. (2013), which defines the transfer of events from short-term to long-term memory. We remark that for some tasks such as those depicted in Figure 1, relevance varies very little across time. To implement relevancy screening for such tasks, at every time step t we attend to two subsets of the past hidden states. We call the first subset a *short-term buffer* $S_t = \{h_{t-\nu}, h_{t-\nu+1}, .., h_{t-1}\}$ which consists of the hidden states of the last ν time steps, while the second subset is the relevant set R_t. We compute the *relevance score* at time step i, $\beta(i) = \sum_{j=i}^{i+\nu-1} \alpha_{i,j}$, measuring the integrated attention scores over our short-term buffer S_t. More precisely, $C(i)$ is satisfied if $\beta(i)$ is part of the top ρ relevance scores when compared to all previously observed hidden states, where ρ is a fixed hyper-parameter satisfying $\rho \geq |R_t|$ for all t. The pseudo-code in Algorithm 1 describes the screening mechanisms and the interaction between the short-term buffer S_t and a finite size relevant set R_t. '.replaceWith()' is a function replacing the hidden state with the lowest relevance score by the hidden state in the argument.

Algorithm 1 Relevancy Screening

```
1:  procedure: RelRNN($\mathbf{s}_{t-1}, \mathbf{x}_t$)
      Require: Previous macro-state - $\mathbf{s}_{t-1}$
      Require: Input - $\mathbf{x}_t$, $\nu > 0$, $\rho > 0$
      Require: Short-term buffer $s_{t-1}^{(i)} \in S_{t-1}$
      Require: Relevant set $r_{t-1}^{(i)} \in R_{t-1}$
2:  $h_t \leftarrow \phi(V\mathbf{s}_{t-1} + U\mathbf{x}_t + b)$
3:  $S_t = S_{t-1}$.add($h_t$)
4:  if $t - \nu > 0$ then
5:      $S_t = S_t$.remove($h_{t-\nu}$)
6:  if $t - \rho > 0$ and $C(t-\rho) = True$ then
7:      $R_t = R_{t-1}$.replaceWith($h_{t-\rho}$)
8:  $M_t = [S_t, R_t]$
9:  for all $m^{(i)} \in M_t$ do
10:     $\tilde{z}^{(i)} \leftarrow v_a^T \cdot \tanh(W_a \mathbf{s}_{t-1} + U_a m^{(i)})$
11: $z \leftarrow$ softmax($\tilde{z}$)
12: $\mathbf{s}_t = h_t + \sum_i z^{(i)} m^{(i)}$
13: return $\mathbf{s}_t$
```

To see how the relevancy screening mechanism is grounded in the theory developed in §4.4, note that the sets S_t and R_t give rise to a sparse attention mechanism with sparsity coefficient κ satisfying $\kappa = \nu + \rho \geq |S_t| + |R_t|$. Hence, memory complexity is constant while the $O(T^2)$ bottleneck of computational complexity is replaced by $O((\rho + \nu) \cdot T) = O(T)$. Lastly, applying Theorem 4.4.3, we get the following guarantee for all $t \geq 0$: $\|\nabla_{h_t} L\| = \Omega(1/(\rho + \nu)^d)$ as $T \to \infty$. Thus the choices of ν and ρ not only directly impact computational complexity and gradient propagation, but also indirectly influence gradient propagation via the implicit effect of $\kappa = \nu + \rho$ on d as already discussed in §4.4. Finally, as already mentioned, see Fig 1 in Appendix §B.3, where we perform an experimental trade-off analysis between κ and d by tweaking ρ and ν in the relevancy screening mechanism.

4.6. Experiments

Before describing experiments, we make a few remarks. First, we stress that Relevancy Screening can be applied to any semi-parametric attentive model but we refer to the version presented below, which uses an RNN/LSTM base, as RelRNN/RelLSTM ("*Relevance RNN /LSTM*"). Second, our objective is not to find state-of-the-art performance but to highlight the advantages of event relevancy and selective sparsity. Finally, we note that relevancy-based sparsity does not necessarily improve performance over fully attentive models, but rather allows efficient and scalable usage. As we show below, RelRNN and RelLSTM perform almost identically to other self-attentive recurrent models (e.g. Bahdanau et al. (2014); Ke et al. (2018)) on simple tasks, but use considerably less memory and compute complexity. In what follows, we denote MemRNN/MemLSTM for vanilla self-attention RNN/LSTM as defined in Bahdanau et al. (2014). We also refer to Appendices §B.3, §B.2, §B.4 for additional experimental results and implementation details.

4.6.1. Tasks with sparse dependency chains

A good stereotypical task type that captures sparse sequences of important events are memorization tasks. Here, the network has to memorize a sequence of relevant characters presented among several non-relevant ones, store it for some given time delay and output them in the same order as they were read towards the end of the sequence.

Copy task (Hochreiter and Schmidhuber, 1997): The characters to be copied are presented in the first 10 time steps, then must be outputted after a long delay of T time steps (see full description in Arjovsky et al. (2016)). Thus, all the *relevant events* occur in the first 10 time steps. This can be corroborated by the attention score found in Figure 1 which was generated using full self-attention. Henaff et al. (2016) show that orthogonal RNNs (orth-RNN) provide an optimal solution. We also consider expRNN (Casado and Martínez-Rubio, 2019) which does optimization in the unitary space and is so far the best purely performing recurrent model for large time delays for this task.

Table 2 (Appendix §B.4) reports test performances of orth-RNN, expRNN, MemRNN, SAB, RelRNN and RelLSTM for $T = \{100, 200, 300, 500, 1000, 2000\}$ on the Copy Task. We find that orth-RNN solves this task up to $T = 500$, but that accuracy decays beyond that point, similarly to LSTM. RelRNN, RelLSTM, SAB and expRNN perfectly solve this task with $\mathbf{100}\%$ accuracy for all T, while Fig 2 in Appendix §B.4 shows that RelRNN learn copy and denoise tasks with significantly fewer number of updates as compared to other baselines. MemRNN solves this task until $T = 100$ but overflows memory (OOM) afterwards.

Transfer Copy task: An important advantage of sparse attentive recurrent models such as RelRNN is that of generalization. This is illustrated by the Transfer Copy scores (Hochreiter and Schmidhuber, 1997) where models are trained on Copy task for $T = 100$ and evaluated for $T > 100$. Table 1 shows results for the models listed above, in addition to h-detach (Kanuparthi et al., 2019), an LSTM-based model with improved gradient propagation. Importantly, where purely recurrent networks performed well on the original task, all fail to transfer, with discrepancy growing with T. As expected, MemRNN and SAB keep good performance but RelRNN outperforms them, with almost perfect performance for all T. While both SAB and RelRNN use sparse memory storage

Table 1. Results for Transfer Copy task.

T	100	200	400	2000	5000
orth-RNN	99%	4%	16%	10%	0%
expRNN	100%	86%	73%	58%	50%
MemRNN	99%	**99%**	**99%**	92%	OOM
RelRNN	**100%**	**99%**	**99%**	**99%**	**99%**
LSTM	99%	64%	48%	19%	14%
h-detach	100%	91%	77%	51%	42%
SAB	99%	95%	95%	95%	95%
RelLSTM	**100%**	**99%**	**99%**	**99%**	**99%**

Table 2. Results for Denoise task.

T	100	300	500	1000	2000
orth-RNN	90%	71%	61%	29%	3%
expRNN	34%	25%	20%	16%	11%
MemRNN	99%	**99%**	**99%**	**99%**	OOM
RelRNN	**100%**	**99%**	**99%**	**99%**	**99%**
LSTM	82%	41%	33%	21%	15%
GORU	92%	93%	91%	93%	73%
SAB	99%	**99%**	**99%**	**99%**	**99%**
RelLSTM	**100%**	**99%**	**99%**	**99%**	**99%**

and retrieval, the distinguishing factor is RelRNN's use of relevancy screening, indicating it's importance for transfer. The performance of RelLSTM on Transfer Copy is exactly the same as RelRNN.

Denoise task Jing et al. (2019): This generalizes the Copy task as the symbols that need to be copied are now randomly distributed among the T time steps, requiring the model to selectively pick the inputs that need to be copied. We test our method against all the previously mentioned models in addition to GORU Jing et al. (2019) for various values of T (Table 2). RelLSTM performs exactly as RelRNN and again, we see RelRNN maintain complete performance across all T values, outperforming all purely recurrent models. MemRNN performs as RelRNN/RelLSTM but fails to train due to memory overflow beyong $T = 500$.

4.6.2. Tasks with dense temporal dependencies

In contrast to sparse information found in the tasks above, we now illustrate RelRNN and RelLSTM's performance on tasks with densely distributed information on long sequences.

Here, we perform tests on pMNIST Le et al. (2015), a variant of MNIST LeCun and Cortes (2010) where pixels are fed sequentially in a permuted order to the network, as well as character level Penn Tree Bank corpus (PTB) (Marcus et al., 1993) where the next letter in a text needs to be predicted.

See Table 3 for results. Implementation details and further test data found in Appendix §B.4, including attention heatmaps such as the ones found in Figure 1, showing dense attention for RelRNN in both tasks. We note that gated RNNs such as LSTMs are known to perform well here, and that orthogonal RNNs such as those tested here are also very good. The full attention model (MemRNN) fails to train on the optimization setup used here for both tasks, again due to overflow in memory.

4.7. Analysis

Table 3. PTB and pMNIST results.

	PTB Task		pMNIST
Model	BPC	Accuracy	Accuracy
RNN	1.56	66%	90.4%
orth-RNN	1.53	66%	93.4%
expRNN	1.49	68%	**96.6**%
RelRNN	**1.43**	**69**%	92.8%
LSTM	**1.36**	**73**%	91.1%
h-detach	-	-	92.3%
SAB	1.37	-	94.2%
RelLSTM	**1.36**	**73**%	**94.3**%

In this section we analyze the maximal GPU usage and gradient norm of $\|\nabla_{h_t} L\|$ across time t for the Denoise Task. All the models were run using a NVIDIA TitanXP GPU and their peak usage was recorded in order to quantify the amount of computational resources used for each of them. We varied sequence length T from 200 to 2000 in order to measure the trend in the usage. To measure propagating gradients as a function of t, we trained models on $T = 1000$ and computed $\log \|\nabla_{h_t} L\|$.

As illustrated in Figure 2 (center), we confirm MemRNN scales quadratically with T, same as SAB which shows an improvement but only by a constant factor. We also confirm that RelLSTM scales linearly with T similar to RNN and LSTM. Figure 2 (left) shows that the gradient norms for RNN explode and for LSTM vanish as t increases. The gradient norms of all attention models were stable, as expected from the results of Section §4.4. To better visualize the interplay between gradient norm and GPU usage, Figure 2 (right) shows the final averaged log gradient norm against Max GPU usage for different times $T = \{400, 600, 800\}$. As expected, purely recurrent models (RNN, LSTM) show very little GPU usage differences across distinct T values, while their performance and gradients degrade with increasing t. Note that the RNN's gradients explode while the LSTM's vanish, both exponentially in t. Standard self attentive models (MemRNN, SAB) on the other hand, show opposite trends, with stable gradients but GPU usage quadratically increasing in T. As expected from Theorem 2 (§4.4), RelLSTM shows both stable gradients and stable GPU usage[1].

The optimal trade-off between memory usage and good gradient propagation achieved by RelLSTM highlights the importance of a dynamic memory that attempts to predict relevancy in order to only store exactly those events that help with learning. We note the Denoise task has a small number of relevant events and that not all tasks share this structure. Nevertheless, this experiment

[1]The measurements for both GPU usage and gradient norm are identical for both RelLSTM and RelRNN.

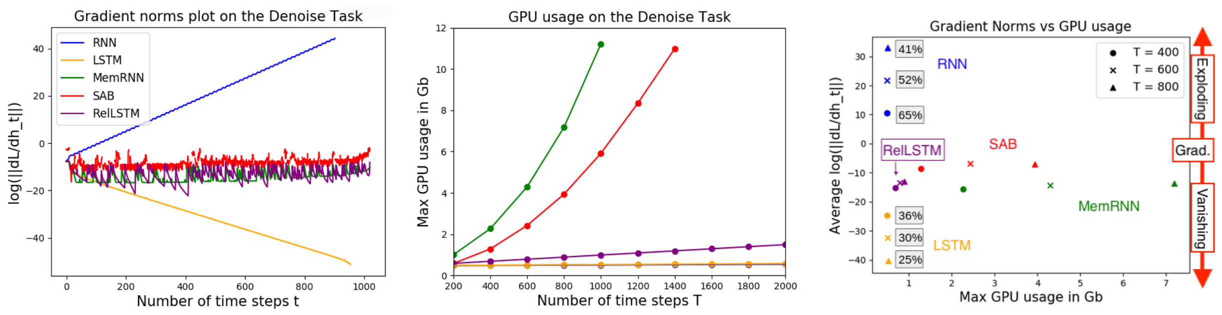


Fig. 2. **(Left)** gradient norm plots of $\|\nabla_{h_t} L\|$ in log scale after training for Denoise Task with t ranging from 0 (latest time step) to 1000 (furthest time step).**(Center)** Maximal GPU usage as a function of total sequence length T.**(Right)** Mean log gradient norm v.s. Max GPU usage for $T = 400$,600,800. Model testing accuracy is 100% unless indicated by marker label (see Table 2).

highlights how important resource gains can be made by shifting efforts from offsetting memory growth by a constant factor, to a relevancy screening method.

4.8. Conclusion & Discussion

Our main contribution is a formal analysis of gradient propagation in self-attention RNNs, from which we derive two quantities that are governing gradient propagation: sparsity and dependency depth. Meanwhile we identify event relevancy as a key concept to efficiently scale attentive systems to very long sequential computations. This is illustrated via a *Relevancy Screening Mechanism*, inspired by the cognitive process of memory consolidation in the brain, that efficiently selects network states, called relevant events, to be committed to long-term memory based on a screening heuristic operating on a fixed-size short-term memory buffer. We showcase the benefits of this mechanism in an attentive RNN and LSTM which we call RelRNN and RelLSTM respectively, using simple but illustrative numerical experiments, and demonstrate the optimal trade-off between memory usage and good gradient propagation it achieves.

As outlined in §4.4 and §4.5, this trade-off is a reflection of the task-specific balance between sparsity and dependency depth parameters. While our proposed relevancy screening mechanism exploits "local" attention scores (measured while events are in short-term memory buffer), we acknowledge other types of relevancy could be formulated with heuristics better suited to distinct environments. For instance, promising directions include leveraging predictive coding principles to select "surprising events", or neural networks could be used to learn the screening function $C(i)$ in an end-to-end fashion.

Broader Impact

We provide a framework for researchers to shape gradient propagation and memory footprint in self-attentive RNNs, which is helpful in tasks requiring ongoing online predictions that cannot be based on future inputs (i.e. in an online sequential setting) and where long-term credit assignment is crucial, such as various RL tasks Harutyunyan et al. (2019); Hung et al. (2019). The added resource gains can save GPU hours and thus have a positive environmental impact. Along this line, we firmly believe that researchers should take environmental impact of model training seriously, and we are hopeful that our work contributes to this direction.

Meanwhile, the theoretical tools provided in the proofs lay the ground for more theoretical work on attentive systems to emerge in the future. More effective RNN models can amplify already existing biases in RNN-based NLP systems through an increased exposure to bias. Finally, we cannot exclude that the cognitive inductive bias we use to build our relevancy screening mechanism may induce prediction quality disparity (e.g. in language modelling) because of the memory tokens it throws away.

Chapter 5

On Neural Architecture Inductive Biases for Relational Tasks

This chapter is based on a NeurIPS 2022 submission (under review). (Kerg et al., 2022)

5.1. Prologue to the Third Article

5.1.1. Article Details

Title: On Neural Architecture Inductive Biases for Relational Tasks

Authors: Giancarlo Kerg[1,2], Sarthak Mittal[1,2], David Rolnick[1,3,5], Yoshua Bengio[1,2,4,5], Blake Richards[1,3,5], Guillaume Lajoie[1,2,5]

Affiliations:

1: Mila - Quebec AI Institute, Canada

2: Université de Montréal

3: McGill University

4: CIFAR Senior Fellow

5: CIFAR AI Chair

5.1.2. Contributions

Giancarlo Kerg produced most of the experiments, code and hyperparameter search. Giancarlo Kerg came up with the full architecture of *CoRelNet* and *CoRelNet-T*, after running a series of ablations on the ESBN architecture. Giancarlo Kerg also designed all newly developed tasks in this paper and led the writing of the article.

5.1.3. Paper Abstract

Current deep learning approaches have shown good in-distribution generalization performance, but struggle with out-of-distribution generalization. This is especially true in the case of tasks involving abstract relations like recognizing rules in sequences, as we find in many intelligence tests. Recent work has explored how forcing relational representations to remain distinct from sensory representations, as it seems to be the case in the brain, can help artificial systems. Building on this work, we further explore and formalize the advantages afforded by "partitioned" representations of relations and sensory details, and how this inductive bias can help recompose learned relational structure in newly encountered settings. We introduce a simple architecture based on similarity scores which we name *Compositional Relational Network (CoRelNet).* Using this model, we investigate a series of inductive biases that ensure abstract relations are learned and represented distinctly from sensory data, and explore their effects on out-of-distribution generalization for a series of relational psychophysics tasks. We find that simple architectural choices can outperform existing models in out-of-distribution generalization. Together, these results show that partitioning relational representations from other information streams may be a simple way to augment existing network architectures' robustness when performing out-of-distribution relational computations.

5.2. Introduction

Current deep learning systems have performed astonishingly well on a multitude of domains, ranging from natural language (Devlin et al., 2018; Child et al., 2019; Dai et al., 2019) and speech recognition (Pratap et al., 2019; Oord et al., 2016) to image classification (Dosovitskiy et al., 2020; He et al., 2015). This progress has been obtained with extensive compute and data (Brown et al., 2020). However, a line of research shows that deep learning models still struggle to perform well in tasks that require abstract relational reasoning, especially in low data regimes and out-of-distribution (OoD) settings (Webb et al., 2021; Kim and Linzen, 2020; Johnson et al., 2017; Yi et al., 2019; Newman et al., 2020; Nogueira et al., 2021; Mittal et al., 2021; Ke et al., 2021).

A working hypothesis we explore here is that most artificial systems do not work well on relational reasoning because they do not encode an explicit notion of relations between different objects, and rather rely too much on representations of object features. There is a body of machine learning approaches exploiting relational structure (Scarselli et al., 2008; Velicković et al., 2017; Kipf et al., 2018; Battaglia et al., 2018; Bengio et al., 2019; Webb et al., 2020; Zhang et al., 2019), but the majority of machine learning systems do not explicitly encode such relations. In contrast, in our brains, relational dependencies across high-level entities form a key ingredient to our understanding of the world and augment our ability to reason in potentially unseen scenarios (Whittington et al., 2019), For example, the relationship between a parent and their child often generalizes to different animal groups and thus explicit modeling of relationships between objects could lead to better out-of-distribution (OoD) generalization, since a number of generic relation types generalize over multiple domains.

Recently, there has been substantial work on attention-based systems (Bahdanau et al., 2014; Luong et al., 2015; Vaswani et al., 2017; Hudson and Manning, 2018) that do compute a notion of similarity between objects, which can be understood as relations between said objects. For example, transformers (Vaswani et al., 2017) compute multiple parallel relational connections between the objects in a scene (Dosovitskiy et al., 2020; Parmar et al., 2018). However, the relational connections computed by transformers are only used as a means to route information between different objects (i.e. via attention scores), and they are not explicitly leveraged for predictions themselves. However, some work does address this shortcoming by explicitly focusing on encoding relational information between objects. One approach is to group objects with similar features, and to attend to these groupings based on context (Santoro et al., 2017; Locatello et al., 2020; Santoro et al., 2018). This has shown promising aptitudes, but suffers from a few issues such as scaling, and may generalize poorly in OoD settings where group membership is not straightforward. Another approach, first outlined by Webb et al. (2021) with the Emergent Symbols through Binding in Memory (ESBN) method, leverages the abstract nature of relations (e.g. parent and child) by explicitly segregating sensory information from relational encoding. While showcasing the importance of separate sensory and relational encodings, ESBN also includes a number of architectural design choices that may or may not be important for OoD generalization in relational tasks.

Here we build on previous relational approaches and explore in depth the inductive bias of separating sensory information from abstract, relational representations. We do so using a model that we call *Compositional Relational Network (CoRelNet)*. CoRelNet uses the simplest possible architecture to distill a minimal set of inductive biases that allow for OoD generalization in relational tasks. Thanks to this simplicity, CoRelNet allows for considerably more robust performance and generalization than several benchmarks, including ESBN. Through a series of numerical experiments involving relational tasks on data with spurious features, scaled input size, and variable contexts, we demonstrate and analyze why the simple act of building representations of similarity relations between objects, irrespective of sensory encoding, is enough to improve a range of widely used model architectures (e.g. multi-layer perceptrons and transformers). Thus, CoRelNet variants confirm the importance of abstract relational encoding and helps to establish minimal inductive biases to enhance other architectures. We find that focusing on these inductive biases to design new and simpler architectures considerably improves OoD generalization across all settings tested, and thus is a promising approach for other machine learning researchers to adopt.

5.3. Motivation and Related Work

Human cognition is heavily dependent on the ability to understand relationships between different entities in the world, regardless of their sensory attributes (Kriete et al., 2013). This ability allows us to excel at a number of tasks that require understanding of abstract rules that govern the relationships between objects in a way that abstracts out the purely perceptual qualities of those objects (see e.g. (Jones and Love, 2007)). For example, people can easily grasp that different institutions can have the same organizational structure, e.g. different high schools will have a principal, teachers, and students with similar relations between these individuals regardless of the specific people who hold those roles.

Ongoing work in neuroscience aims to understand how our brains represent such abstract relationships. Early results identified cognitive maps (Tolman, 1948) and so-called relational memories (Cohen and Eichenbaum, 1993) which seem to encode abstract relational information invariant to some perceptual features (see also Sedda and Scarpina (2012); Goodale and Milner (1992)). While this abstraction of relational knowledge from memory is not entirely disconnected from sensory modality (Barsalou et al., 2003), there is evidence that isolated perceptual features alone are not principally driving relational reasoning, and that complex interdependence of processed features and memory gives rise to abstraction (Goldstone et al., 1989).

Consistent with this neuro-cognitive picture, AI systems which work well on a variety of domains like machine translation (Vaswani et al., 2017; Dehghani et al., 2018; Devlin et al., 2018), image classification (Dosovitskiy et al., 2020), etc. do not perform as well on tasks that explicitly require inference of relational structure between entities (Webb et al., 2021; Johnson et al., 2017; Yi et al., 2019). An example of such a task is Raven's Progressive Matrices (Raven and Court, 1938) which, given a sequence of objects, tests the ability to infer the relations between objects and to use it to select candidate objects that satisfy the underlying relational structure. In response to this shortcoming, a number of AI elements inspired from neuroscience have been proposed to tackle relational abstractions. A fruitful approach has been to endow AI systems with memory and with various mechanisms that enforce abstract representations (Santoro et al., 2018; Graves et al., 2014; Mittal et al., 2020; Pritzel et al., 2017; Hill et al., 2020; Fortunato et al., 2019). Recent work combines this approach with the explicit separation of memory-stored representations and sensory inputs in the ESBN model (Webb et al., 2021) (see also Whittington et al. (2019)), and reveals impressive performance and OoD generalization properties on relational tasks. In this work, we develop CoRelNet, with the goal of identifying a minimal architecture needed to impart an effective inductive bias towards OoD generalizable relational representations.

5.4. Relational Tasks

In this section we describe the collection of tasks used to evaluate relational learning, these range from commonly used tasks in cognitive tests to tasks purposely designed to evaluate artificial systems.

5.4.1. Relational Cognitive Tests

We first describe a set of four relational tasks which are used to assess abstract relational reasoning in cognitive tests. In each of the tasks, object images are provided sequentially such that some specific abstract rule governs the relationship among them. To test for OoD generalization, the objects used for training are different from those used during testing while keeping the abstract rule

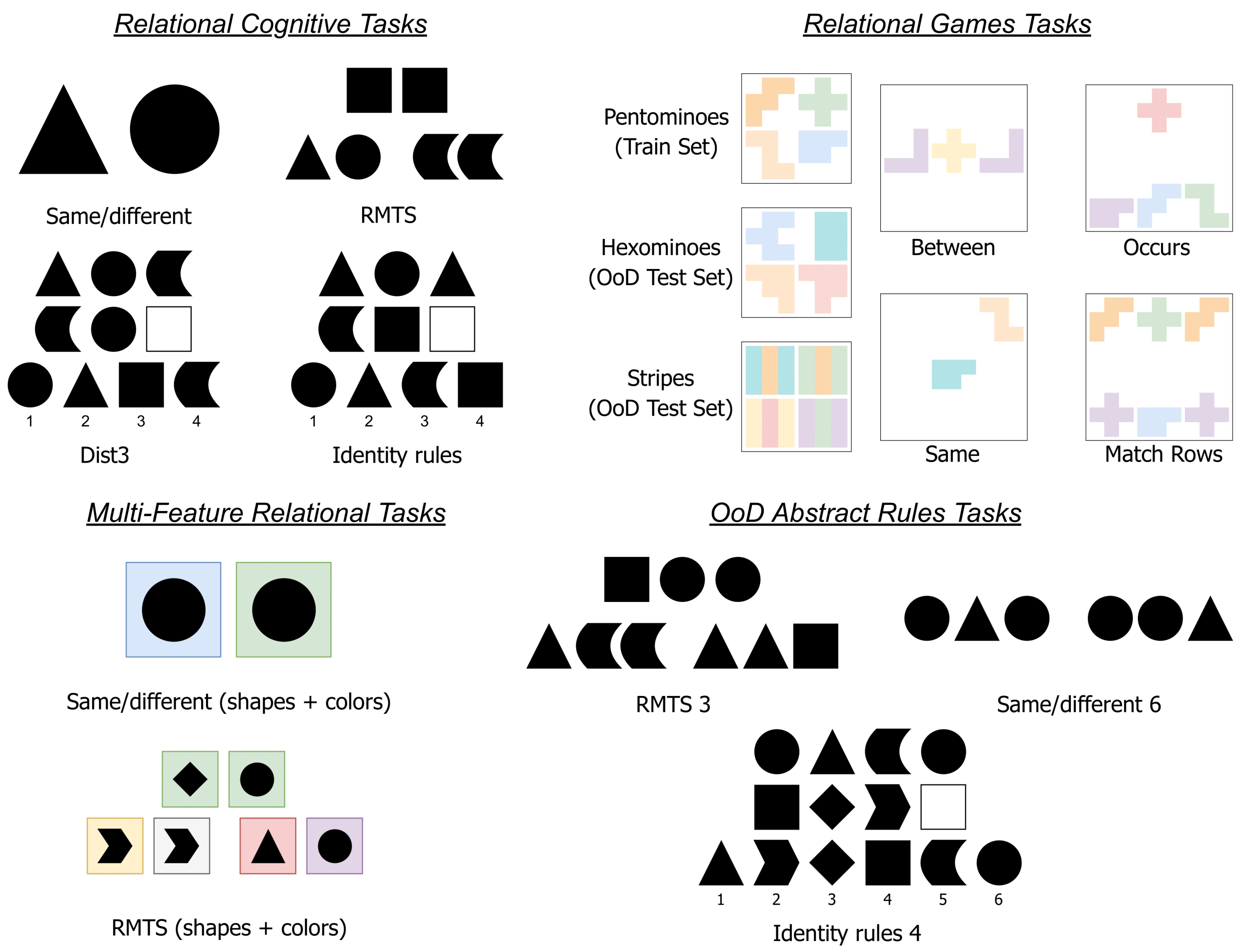


Fig. 1. Top left. Relational cognitive tasks. Same/different discrimination task (answer: "different"). RMTS task (here the source pair (top row) has two identical objects matching the second candidate pair in the bottom row). Dist3 task, choosing a candidate in the bottom row to replace the white square (answer: 2). Identity rules task (ABA pattern, answer: 3). **Top right.** Relational Games tasks. Example objects from the training set ('pentominoes'), two held-out test sets ('hexominoes' and 'stripes'), as well as examples from the tasks 'between' (label: True), 'occurs' (label: False) ,'same' (label: False), 'match rows' (label: True, matching pattern: ABA) **Bottom left.** Examples from relational tasks with a spurious feature (background color). In same/diff. (shapes+colors), the label is 'same' as the two shapes are identical. In RMTS (shapes+colors), the source pair (top row) has two distinct shapes, matching the second candidate pair in the bottom row. **Bottom right.** In RMTS3, the source triplet matches the first target triplet. In same/diff 6, the two triplets are 'different'. In identity rules 4, the answer is 4.

consistent. See Figure 1 for illustrations. Even though these tasks might appear simple at first, it is indeed very challenging for deep neural network architectures to generalize well in OoD settings like this (Kim J, 2018; Webb et al., 2021), including LSTM, Transformers, Relation Networks, Predi-Net, ESBN and others.

Same/different discrimination. Two objects are shown, and the task is to determine whether they are the same or different.

Relational match-to-sample (RMTS). Three pairs of objects are presented: a *source* pair and two *target* pairs. The task consists of identifying which target pair has the same relational structure as the source pair, i.e., if the source pair consists of two identical/different objects, the task is to identify the target pair which contains two identical/different objects.

Distribution-of-three (Dist3). Two rows of three objects followed by a row of four objects are presented. The same set of 3 (distinct) objects that are shown in the first row are in the second row in a permuted order, but with the last object masked. The task is to identify the missing object of the second row from the set of four objects shown in the third row. See P. A. Carpenter (1990) for more details.

Identity rules. Objects are presented in the same manner as in Dist3. However here, the first row contains a sequential pattern of objects (e.g. ABA or ABB or AAA). The task is to pick the object from the last row which completes the second row's sequence such that the sequential pattern of the first row matches the one of the second row. See Marcus et al. (1999) for more details.

5.4.2. Relational Games

This is a set of binary classification tasks first outlined by Shanahan et al. (2019). Each Relational Game is governed by some underlying abstract rule (see list below) and involves the presentation of an image containing objects laid out on a 3×3 grid. An image is labelled as "True" if the objects in the image satisfy the abstract rule, and False otherwise. The training set consists of simple geometric shapes called "pentominoes", and we use two held-out OoD test sets consisting of different shapes called "hexominoes" and "stripes". See Figure 1 for illustrations. Contrary to the implementation by Shanahan et al. (2019), and more in line with the tasks above, here the 3×3 grid is presented sequentially. The rules considered are as follows.

Same/Different. Objects are "same" if they have the same shape, colour and orientation.

Between. This relation holds if the image is composed of three objects displayed in a straight line where the outer objects are "same".

Occurs. This relation holds if the object in the top row is the "same" as one of the objects in the bottom row.

Row Matching. This relation holds that the first row and last row of objects are governed by the same underlying abstract pattern such as AAB, AAA, or ABB.

Xoccurs. This relation holds if the object in the top row is the "same" as one of the objects in the bottom row, while the other two objects in the bottom row are different.

Colour/Shape. This relation has four labels: same-same, same-different, different-same and different-different, depending on whether the two objects shown in the image have the same or different colour/shape.

Left-of. This positional task is slightly different. Only two objects with different luminance are presented somewhere in the 3×3 grid (other places are left blank). The relation holds if the object of lower luminance is to the 'left-of' the other object. Note that this relation is antisymmetric, since interchanging the objects changes the label of the image.

5.4.3. Enhanced Relational Cognitive Tests

We modify tasks from §5.4.1 to probe more difficult and abstract relations in order to further explore OoD generalization and model limitations. Modifications probe two aspects: unseen relations, and spurious correlations/distractors.

Unseen Relations. We construct tasks where we restrict not only the set of objects but also the set of relations seen during training, while testing on unseen objects as well as unseen relations. These include **RMTS 3:** similar to RMTS where for training only triplets of the form (AAA, ABA and BAA) are shown, while testing *only* involves triplets of the form (ABC, AAB). **Same/diff 6:** Similar to Same/Different but comparing 2 triplets instead of single objects. Each triplet consists of at most two distinct objects (thereby of forms AAA, AAB, ABA or BAA). During training, only triplets of the form (AAA, ABA, BAA) are shown, while evaluation is done on examples that have at least one triplet of the form (AAB). **Identity rules 4:** Similar to 'Identity rules' but with rows of 4 objects and multiple choice of 6. Note that for each quadruple in row 1 or 2, we can have at most three distinct objects. During training, only quadruples with at most two distinct objects are shown, while testing relies *only* on quadruples with exactly three distinct objects. **Identity rules 4 Missing:** Identical to Identity rules 4 but the training set does not include quadruples of the form ($ABAA$, $ABAB$). See also Appendix §C.3 where we include another variation called *identity rules 4 [flipped]* where testing and training sets are swapped. **Distribution-of-N:** Same as Dist-3 but with N objects. See also Appendix §C.4 for results.

Spurious Correlations. In this last set of tasks, we add spurious features to objects. We again consider the four relational tasks from § 5.4.1, but this time we provide each object with a (distracting) background color. The tasks consist of identifying the same relational structure but based *solely* on shapes.

5.5. Models

Using tasks described in § 5.4, we compare five models[1]: Transformers, PrediNet, ESBN, and two versions of our model: CoRelNet, and CoRelNet-T. All the models use an encoder q which is sequentially applied to each of the inputs $\{x_t\}_{t=1}^T$ followed by a temporal context normalization (TCN) step, which has shown to significantly improve OoD generalization in relational reasoning tasks (Webb et al., 2020).

$$\{x_t\}_{t=1}^T \longmapsto q(\{x_t\}_{t=1}^T) \longmapsto \text{TCN}(q(\{x_t\}_{t=1}^T)) = \{z_t\}_{t=1}^T$$

The encoded inputs $\{z_t\}_{t=1}^T$ are then being fed to the respective models. **Transformer.** Introduced in Vaswani et al. (2017), transformers rely on multiple parallel attention mechanisms (Multi-Head Attention or MHA) to route information between different input tokens, which can be elements in a set or in a sequence, etc. While these systems do compute multiple similarity matrices in

[1]All models were trained on single RTX8000 GPUs and each experiment took a few hours.

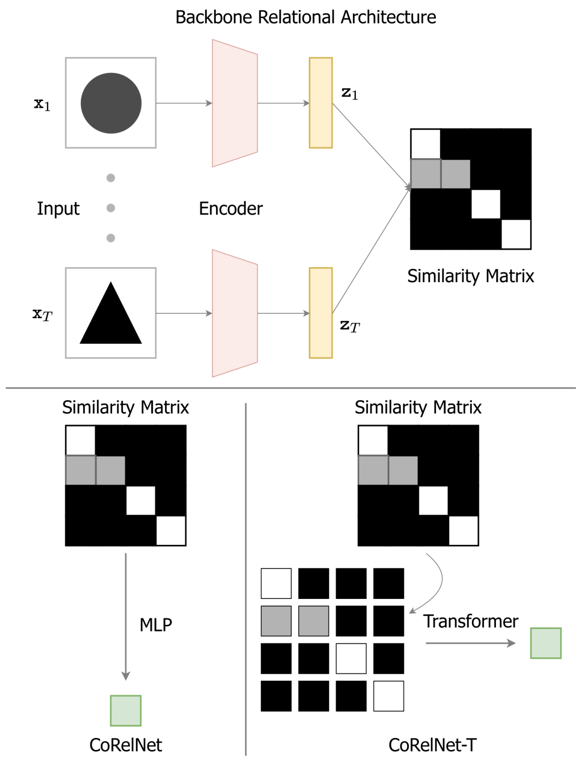


Fig. 2. Illustration of Relational Architecture. (top) The backbone relational architecture describes the common backbone present in all of the relational architectures considered in this work, with a similarity matrix with entries for each pair of input objects. (bottom) CoRelNet considers an MLP to process the similarity matrix obtained at the end of the backbone while CoRelNet-T uses a Transformer.

parallel, these are further combined with sensory information to drive final prediction and hence the dichotomy between explicit relational and sensory structure is absent. Prior work (Webb et al., 2021) hypothesizes that the absence of this dichotomy leads to poor OoD generalization, and we analyze this further through our ablations in §5.6.

PrediNet. Introduced by Shanahan et al. (2019), PrediNet leverages Multi-Head Attention (MHA) to map inputs onto objects, relations and propositions, thereby attempting to extract the underlying relational structure via propositional representations compatible with predicate calculus.
ESBN. Emergent Symbol Binding Network introduced in Webb et al. (2021) constructs an external memory through a recurrent controller and is the first model to make use of the separation of sensory and relational information. This means the model is separated into two distinct information processes. An LSTM is used to process relational data, incurring additional processing.

CoRelNet. In our main model, *CoRelNet*, we compute the self-attention coefficients

$$\{R_{i,\cdot}\}_{i=1}^{T} = \{\text{softmax}(z_i^\top \cdot \mathbf{z})\}_{i=1}^{T}$$

over the encoded inputs $\{z_t\}_{t=1}^{T}$, and feed those *directly* into an MLP decoder to produce the final output o (see Figure 2):

$$R \overset{\text{flatten}}{\longmapsto} \text{flatten}(R) \overset{\text{MLP}}{\longmapsto} o$$

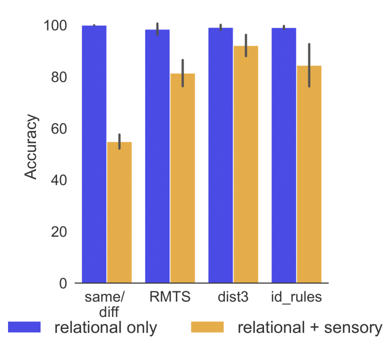
Task	*Test Set*	*CoRelNet*	*CoRelNet-T*	*ESBN*	*PrediNet**	*Transformer*
same	Hex.	$94.7_{\pm 5.4}$	$98.7_{\pm 0.5}$	$60.9_{\pm 18.0}$	$99.1_{\pm 0.5}$	$96.3_{\pm 3.6}$
	Str.	$90.4_{\pm 9.0}$	$98.4_{\pm 0.6}$	$60.5_{\pm 17.0}$	$97.7_{\pm 1.5}$	$93.6_{\pm 4.5}$
between	Hex.	$96.6_{\pm 2.2}$	$91.9_{\pm 3.3}$	$79.5_{\pm 2.7}$	$94.4_{\pm 3.6}$	$90.9_{\pm 4.2}$
	Str.	$93.1_{\pm 5.4}$	$87.0_{\pm 4.7}$	$82.5_{\pm 4.5}$	$92.4_{\pm 5.4}$	$87.5_{\pm 6.4}$
occurs	Hex.	$96.2_{\pm 2.2}$	$91.6_{\pm 4.6}$	$74.3_{\pm 0.9}$	$92.9_{\pm 3.3}$	$95.9_{\pm 2.4}$
	Str.	$88.7_{\pm 5.3}$	$79.3_{\pm 12.1}$	$73.3_{\pm 0.9}$	$87.4_{\pm 6.5}$	$88.7_{\pm 5.4}$
xoccurs	Hex.	$92.2_{\pm 6.4}$	$91.7_{\pm 6.9}$	$63.0_{\pm 0.8}$	$67.2_{\pm 8.7}$	$75.1_{\pm 11.9}$
	Str.	$83.6_{\pm 10.9}$	$85.4_{\pm 6.4}$	$64.7_{\pm 2.1}$	$61.3_{\pm 6.3}$	$69.4_{\pm 13.0}$
row matching	Hex.	$97.7_{\pm 0.8}$	$95.4_{\pm 5.1}$	$81.1_{\pm 2.7}$	$50.3_{\pm 0.5}$	$50.3_{\pm 0.6}$
	Str.	$94.8_{\pm 1.3}$	$90.5_{\pm 5.2}$	$76.8_{\pm 4.3}$	$50.5_{\pm 0.5}$	$50.4_{\pm 0.5}$
col./shape	Hex.	$47.2_{\pm 3.7}$	$49.6_{\pm 0.8}$	$31.2_{\pm 10.1}$	$74.9_{\pm 10.6}$	$89.1_{\pm 2.9}$
left-of	Hex.	$99.2_{\pm 0.7}$	$97.6_{\pm 1.2}$	$49.9_{\pm 0.3}$	$94.9_{\pm 0.9}$	$96.0_{\pm 1.9}$

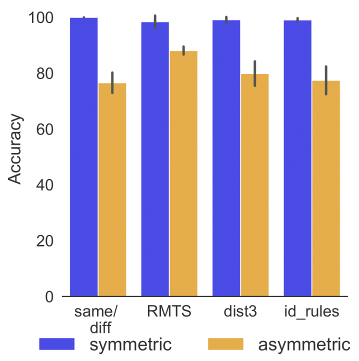
Table 1. Out-Of-Distribution test accuracies for the Relational Game tasks on the two held-out sets "Hexominoes" (Hex.) and "Stripes" (Str.). Results reflect test accuracies averaged over 10 seeds (Details in Appendix §C.2).

First note that self-attention is computed via dot products and hence can be interpreted as giving rise to a (symmetric) similarity measure encoding relational information of the objects in the input sequence. Since the input to the MLP decoder is only composed of self-attention coefficients $R_{i,j}$ (carrying the relational information), and not any sensory information, we make use of the mentioned important inductive bias of *separating sensory details from relational information*. Meanwhile, the MLP applies independent weights to each $R_{i,j}$ which allows for maximal flexibility and enables the model to remain sensitive to positional information when making use of the mentioned relational information provided. This advantage will be of specific use in later experiments (see Figure 5)

CoRelNet-T. This is a variation of the CoRelNet model where the MLP decoder is replaced by a Multi-Head Attention layer (Transformer) with 8 heads to produce the final output. Each row of the similarity matrix is treated as an individual token, and these tokens, combined with learned positional embeddings, are fed to the transformer system to provide the prediction (see Figure 2; *bottom right*).

In both CoRelNet and CoRelNet-T, we also explore different sensory encoding schemes based on either a learned or a random encoder, as part of our analysis which is described in more detail in §5.6. We also refer the reader to Appendix §C.1 for hyperparameters and more architectural details.





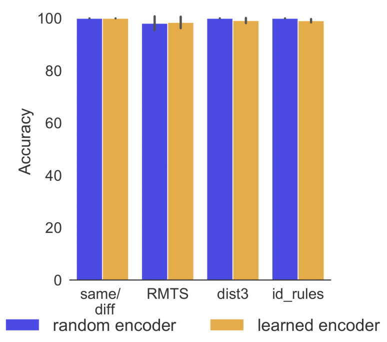


Fig. 4. Test accuracies on the 4 basic relational tasks. Results are displayed for the OOD cases for each task. Results reflect test accuracies averaged over 10 seeds. **Left.** Results showing average accuracy of relational only models (ESBN, CoRelNet and CoRelNet-T) vs average accuracy of relational + sensory input models (here for all three models (ESBN, CoRelNet and CoRelNet-T) we concatenated the encoded input vectors to the input of the decoder.) For detailed performances see Figure 7 in the appendix. **Middle.** Results showing average accuracy of symmetric models (ESBN, CoRelNet, CoRelNet-T) vs average accuracy of their asymmetric counterparts. For detailed performances see Figure 9 in the appendix. **Right.** Results showing average test accuracy of ESBN, CoRelNet and CoRelNet-T with a random encoder vs learned encoder. For the random encoder, the weights of the encoder are randomly initialized, but not updated via backpropagation. For detailed performances see Figure 8 in the appendix.

5.6. Results

We note that given the structure of the relational cognitive tasks, the ground-truth prediction is completely de-coupled from the actual shapes of individual objects, and relies entirely on relations between them. In this section, we analyze the specific inductive biases that allow our models to perform well in OoD settings that respect this assumption. For hyperparameters see Table 1 in Appendix C.1.

All you need is a set of similarity scores. We first evaluate the hypothesis that the relational information between the objects seen in the input sequence is all we need for OoD generalization on these relational cognitive tasks. To this end, we first investigate the importance of disconnection of relational information from the sensory input for OoD generalization. We hypothesize that if a model learns this function and ignores any information regarding the absolute identity of the objects, then it will be able to generalize well in the OoD settings where the only change performed is the identity of objects considered. This is indeed what we find in experiments, where the models that make predictions solely based on similarity scores between objects (*ESBN*, *CoRelNet* and *CoRelNet-T*) do exceptionally well

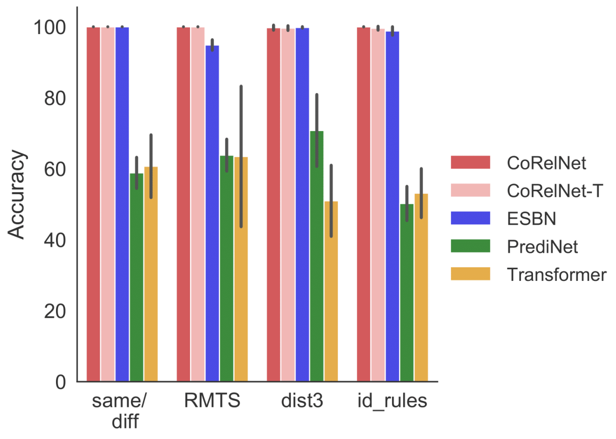


Fig. 3. Test accuracies on the relational tasks. Results are displayed for OoD cases for each task. Results reflect test accuracies averaged over 10 seeds. For details see Figure 6 in the appendix.

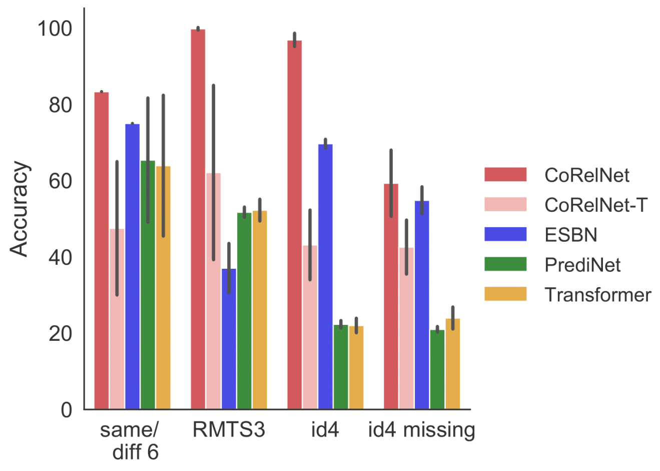
on OoD generalization, as opposed to models that also rely on the sensory details of the objects (*Transformer*, *LSTM*, *RN*, etc). This is illustrated in Figure 3 as well as in the results outlined by Webb et al. (2021). This makes the point that the relational information alone in the form of a $T \times T$ similarity matrix between objects is sufficient to generalize OoD on these tasks, where T is the number of objects in the sequence.

Next, we also show experimentally that a a simple MLP decoder, like in *CoRelNet*, outperforms more complex architecture like *ESBN* or *CoRelNet-T* on OoD generalization (also see Figure 5), assuming the number of objects is fixed. Thus, having stripped away all the additional inductive biases regarding writing keys and recurrence from *ESBN*, we actually see better performance and optimal OoD generalization.

As additional evidence for the hypothesis of this section, we see in Figure 4 (left) that the concatenation of sensory information to the already present relational information in input to the classifier stage degrades the OoD generalization capacity of the models. This is because overfitting on the sensory information of objects cannot lead to generalization in the OoD settings, while memorizing the similarity matrix patterns does generalize on unseen new shapes, since the prediction rule as a function of the similarity matrix remains the same in as well as out-of-distribution.

Symmetry. Having established that learning using the similarity matrix can work well for the tasks at hand, the problem reduces to finding a similarity measure that can decide same-vs-different even for unseen objects. We note that the use of a (symmetric) dot-product leads to an important bias: *it always scores identical objects higher than different objects*. To understand how much the symmetric nature of the dot-product plays a role as a relevant inductive bias, we ablate over an asymmetric version of the dot-product, defined as $(W_1 \cdot z_t)^\top (W_2 \cdot z_t)$, where W_1 and W_2 are learnable matrices, learned via backpropagation. We train *ESBN*, *CoRelNet* and *CoRelNet-T* with symmetric vs asymmetric versions and see that indeed a symmetric dot-product is essential for OoD generalization on these relational tasks (see Figure 4 (center)). Note that the symmetric dot-product can be recovered from the asymmetric one but the model is not able to learn it, indicating that it needs to be explicitly encoded as an inductive bias.

Learning a sensory encoder. From results presented in the last two paragraphs and Figure 4 (left and center), we conclude that if the encoder provides an injective mapping (mapping two distinct objects to two distinct encodings), the symmetric inductive bias of the dot-product allows the model to detect same-vs-different without any encoder training. Given sufficient representation capacity, it becomes hard for a random encoder to lead to a non-injective mapping. This begs the question as to whether the models will maintain their performance even with a randomly initialized and untrained encoder (random encoder)? Figure 4 (right) shows that for the relational cognitive tasks, this is indeed the case, hence, in these tasks, the only part that needs to be trained (besides potentially the parameters of the normalization step) is the decoder (an RNN-based system for ESBN; MLP for CoRelNet, etc.). The job of this decoder is essentially to map patterns in the similarity matrix to final predictions. Note that reducing the data-setting from a sequence of objects



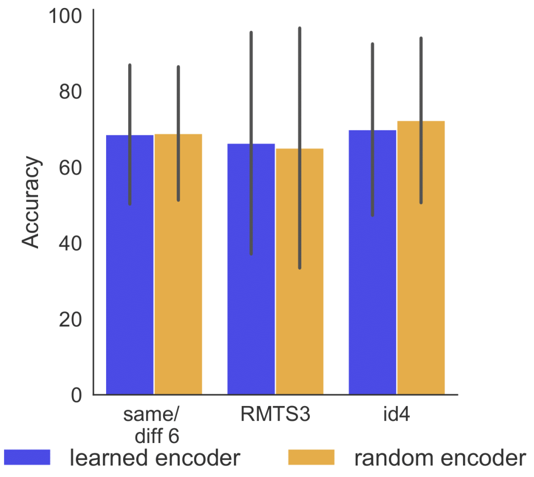


Fig. 5. Test accuracies on the relational tasks with unseen relations. Results are for the OOD cases for each task. Results reflect test accuracies averaged over 10 seeds. **Left.** Overall performance for all 5 models. For full details see Figure 10 in the appendix. **Right.** Results showing average test accuracy of ESBN, CoRelNet and CoRelNet-T with a random encoder vs learned encoder. For the random encoder, the weights of the encoder are randomly initialized, but not updated via backpropagation. For full details see Figure 11 in the appendix.

with various shapes to a similarity matrix reduces the hypothesis space and incentivizes the decoder to learn the true prediction rule for these set of tasks.

Generalization with unseen relations. Next, we consider a scenario where only a subset of relations are shown during training. We see that *CoRelNet* outperforms ESBN and Transformer on all of these tasks (Figure 5), with near perfect accuracy on *RMTS 3* and *identity rules 4*, confirming that it is not only able to generalize to unseen objects but also to unseen relations. We hypothesize that the drop in performance of all models in the *identity rules for 4 elements (with missing variations)* task is due to the fact that some of the “essential base elements" that span the space of all possible abstract relations are left out. We specifically set up this task in order to see a substantial drop in performance. We also note that leaving out either $ABAA$ or $ABAB$ alone is not sufficient for *CoRelNet* to experience a drop in performance, both classes of quadruples need to be removed from the training set.

Generalization with spurious features. Results in Figure 6 show that random encoder performs poorly (averaged over ESBN and CoRelNet models) in the presence of spurious features. In the relational cognitive tasks, the entire input image can be considered to compute the underlying relational structure, and all the encoder had to accomplish is distinguish two distinct images. In the spurious feature tasks, the encoder not only has to learn what features to ignore on seen shapes but also do so on unseen shapes. Hence a random encoder is not sufficient to accomplish this goal. We also included ESBN and *CoRelNet* with a linear L1-regularization layer after the encoder in order to dampen the effect of spurious features. We see that even a trained encoder struggles on ignoring the color, and often requires strong regularization to do well.

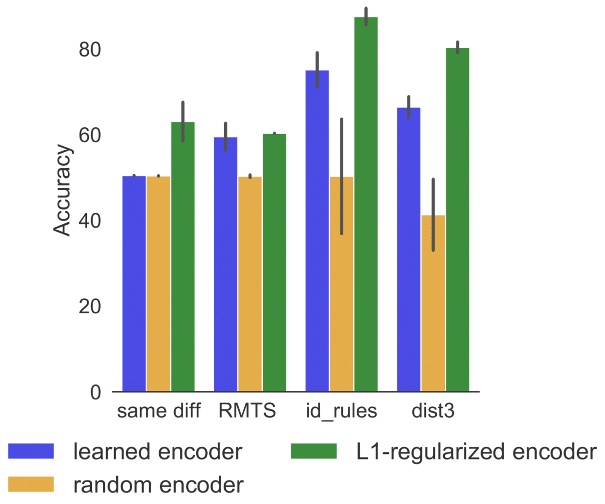


Fig. 6. Test accuracies on the relational tasks with spurious features. Results are displayed for the OOD cases, and reflect test accuracy averaged over 10 seeds. The random encoder model class displays the average accuracies of ESBN and *CoRelNet* with random encoder, while the learned encoder model class displays the average accuracies of ESBN and *CoRelNet* with a trained encoder. The L1-regularized include ESBN and *CoRelNet*, both having L1-regularization on learnt representations (with $\lambda = 5$ for RMTS and $\lambda = 1$ for the other three tasks). For more details see Appendix §C.5.

5.7. Conclusion

We conclude that for purely relational tasks, the relational information between the objects is all that is needed for good OoD generalization. In particular, for the models we considered, disconnecting the said relational information from the sensory input is an essential inductive bias for OoD generalization on unseen inputs (Figure 4 (left)). This is further strengthened by the fact that after removing the effects of encoder-training as well as additional inductive biases in the decoder, we still achieve good OoD generalization as long as final prediction is driven *solely* by relational information.

Further, we see that computing the relational information through a (*symmetric*) dot-product is useful, as it always scores higher on identical objects than on different objects (Figure 4 (center)). We also conclude that as long as the encoder is injective and there are no spurious features, even a random encoder can accomplish good OoD generalization not only on unseen objects but often also on unseen relations (Figure 4 (right) and Figure 5 (right)). In other words, OoD performance on purely relational tasks without spurious features is a direct reflection of how well the decoder translates the relational information encoded in the similarity matrix to the correct output. It turns out that even a simple MLP decoder does the job. We also note that OoD performance on unseen objects and relations can deteriorate if certain crucial classes of training samples are withheld (refer to Identity rules 4 task with missing examples in §5.4.3 and Figure 5 (left)).

In the presence of spurious features, we see that a random encoder no longer does well, as expected since the encoder now needs to selectively encode only a subset of features (also on unseen objects) which are directly relevant for the underlying rule while ignoring the rest. We experimentally show that this is substantially harder as even using a learned encoder is often not enough and requires strong regularizations on the representations (see Figure 6).

Finally, we highlight that the use of network architectures that promote the dichotomy of relational and sensory signals could potentially promote unwanted biases when sub-optimal, as seen in models that were not able to ignore distracting features. The framework and analysis proposed in this work could help better disentangle sensory and relational information to mitigate this potentially problematic impact on society.

5.8. Future work

This work analyses the use of self-attention coefficients or some kind of similarity measure between objects as sole inputs for downstream predictions to implement the inductive bias of separating sensory inputs from relational information. This procedure is easy and flexible to implement in a variety of settings, potentially addressing some limitation of current large scale models that struggle with OoD generalization. We would also like to explore this inductive bias in several architectures with potentially more general similarity measures which not only rely on a notion of "sameness".

We also see that in some OoD tasks, separating sensory inputs from relational information might be beneficial, but sometimes only by a small margin (Figure 4 (left); for instance in the Dist3 task the difference is smallest). Meanwhile, there might be more complex real-world tasks which may require heavier reliance on sensory information. A potential idea would be to explore a flexible task-based competition structure between the use of sensory and relational information, where the use of sensory information for downstream prediction has to undergo a bottleneck due to the competition structure as well as additional task-specific regularization.

Another interesting avenue is to explore settings where multiple abstract rules are at play and need to be individually inferred and combined in a flexible manner, often in the presence of spurious features and out-of-distribution compositions. This is a challenging problem as the encoder has to learn to ignore spurious features for a given rule, which is crucial especially when combined with multiple rules at play. We believe that understanding these settings and distilling a key set of inductive biases that may make artificial systems succeed in these complex domains is an important future work.

Chapter 6

Conclusion

The aim of this thesis was to understand the qualitative gap between current deep learning and human cognitive abilities in regards to out-of-distribution systematic generalization as well as learning over extremely long time scales, by studying a specific set of key inductive biases to circumvent these limitations and thus enhancing information propagation.

- **The exploding and vanishing gradient problem (EVGP)** is one of the main challenges in training recurrent neural networks on longer input sequences, making it difficult to capture long-term dependencies. In chapter 3, we focussed on spectrally constrained approaches throughout the entire training period, extending existing methods to include non-normal dynamics in the connectivity matrix allowing connectivity eigendirections to be non-orthogonal.

  Inspired by previous work on non-normal RNN dynamics in the computational neuroscience literature (Ganguli et al., 2008; Goldman, 2009; Hennequin et al., 2012), we trained directly on the factorized Schur decomposition of the connectivity matrix, hence gaining direct access to the eigenvalues as well as the non-normal parts quantifying the interactions between the different Schur modes. This bypasses the need to compute a costly factorization explicitly at each update of gradient descent. In other words, we can directly control gradient stability by enforcing constraints on the connectivity eigenvalues, while allowing the transient dynamics of the non-normal part to perform additional short term computations. We therefore preserve the advantages from purely orthogonal RNNs (such as long-term gradient propagation and fast learning on tasks involving long-term memorization), while providing increased expressivity for tasks requiring online computations.

  We further show in Proposition 3.4.2 that nnRNN with unit-norm eigenspectra can have at most polynomial gradient explosion. Thus, relaxing the unit-norm constraint on the eigenspectra of the connectivity matrix will lead to a trade-off between exponential vanishing and polynomial exploding gradients, which can be modulated by the L2 regularization hyperparameter δ. This naturally translates into task-specific trade-off between long-term memorization and short-term complex computation. By examining the learned connectivity structure after training, we noted that on some tasks long-term memorization is more important (such as the copy task), while on other tasks (such as PTB) short-term complex computations become more relevant for good performance.

- Motivated by the empirical evidence that memory augmented recurrent networks mitigate the mentioned exploding and vanishing gradient problem (EVGP), we provide a **formal gradient propagation analysis** with asymptotic guarantees for a whole family of memory augmented recurrent networks in chapter 4. We identify the sparsity coefficient κ and maximal dependency depth d as two key quantities governing gradient stability. We saw that modulating the sparsity coefficient κ, implicitly influences maximal dependency depth d and hence gradient stability, which directly translates into a trade-off between computational complexity and good long-term gradient propagation.

  Inspired by the cognitive process of memory consolidation, and in order to leverage the findings of our theoretical analysis, we are proposing a **novel relevancy screening mechanism** allowing explicit control over the sparsity coefficient κ, by modulating the hyperparameters ν and ρ, which are the sizes of the short-term buffer and relevant set respectively.

  After comparing our method to other gated, spectrally constraint or memory augmented recurrent networks, it turns out that the relevancy screening mechanism is the only architecture achieving a linear memory usage and compute complexity, while maintaining good gradient stability on very long sequences. This is reflected in the experimental results, where *RelLSTM* and *RelRNN* outshine all other recurrent models on tasks with very long input sequences and a sparse dependency structure. More specifically, models with quadratic complexity typically run into memory overflow issues, while models with poor gradient stability deteriorate in performance for longer input sequences.

  Finally, it is worth noting that all spectrally constraint methods struggle to solve the Transfer copy and Denoise tasks for very long sequences, hinting towards the fact that, in the context of recurrent networks, sparse self-attention is a much more useful inductive prior when it comes to learning over very long time scales in a robust manner dealing while mitigating the gradient vanishing problem.

- We finally studied a family of various purely relational tasks to distill **a minimal set of inductive biases for out-of-distribution generalization** in chapter 5. We found out that relational information is all you need in tasks where the final prediction is solely driven by an abstract rule, and where inputs are not containing any spurious features. We further noted that adding sensory information to the relational information as being part of the input to the decoder degrades OoD performance on unseen inputs, indicating that the dichotomy of sensory details and relational information is a useful inductive bias in the context of purely relational tasks.

  The OoD performance on unseen inputs does not deteriorate after removing the effects of encoder-training, and thus reflects how well the decoder translates the relational information to the correct output. Meanwhile, in the presence of spurious features, a random encoder no longer gives good OoD performance, and needs to selectively encode a subset of relevant input features even on unseen inputs, which is a challenging problem requiring further regularization.

  Finally, we investigated harder tasks, involving not only unseen inputs but also unseen relations, where specific classes of relational configurations are completely withheld from the training set. It turns out that our proposed model, does well as long as all ’crucial’ classes are present in the training set, and deteriorates substantially if some of those classes are not shown during training.

Chapter 7

Future Work

7.1. Synthesizing sensory input and relational information

Behavioural experiments in (Barsalou et al., 2003) implicate modality-specific systems in the representation and use of conceptual knowledge, highlighting that higher level processing is not easily decoupled from perception. Further, (Goldstone and Barsalou, 1998) and (Ottmar et al., 2015) conclude that perceptual processes provide useful mechanisms that may be co-opted by abstract thought, and that perceptual manipulations have an effect on conceptual processing.

Hence, even though `CoRelNet` might provide a useful inductive bias for purely relational tasks based on a notion of sameness, one might want to investigate how to synthesize computational pathways stemming directly from sensory inputs with pathways using purely relational information, in order to design biologically plausible architectures solving more complex real-world tasks.

7.2. Asymmetric similarity measures

Both (Tversky, 1977) and (Nosofsky, 1991) argue that human similarity judgement is asymmetric. Namely, when asking 'is A similar to B', B becomes the reference/anchor, and one focusses on context-dependent features of B, and compares them to A.

Meanwhile, it is straightforward that the use of a symmetric similarity measure (here a simple dot-product) in purely relational architectures such as `ESBN` and `CoRelNet` limits us to tasks with underlying symmetric relations. In fact, let us consider the 'comparison task' (see Figure 1 for an example), where one gets to see two objects with identical shape, one of which is small and the other one large. The tasks consists in picking the position of the larger object. This relationship is antisymmetric and clearly cannot be learned by architectures with a symmetric similarity measure such as `ESBN` or `CoRelNet`. We will also consider a variation of the 'comparison task', called the 'transitive comparison task', where one gets to see two objects with identical shape, but of different sizes. During training, one gets to see pairs of objects of sizes (medium, large), (large, medium), (small, medium) or (medium, small) respectively. At testing, one gets to see only pairs of objects

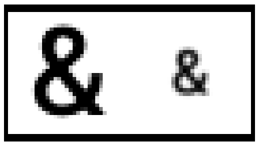
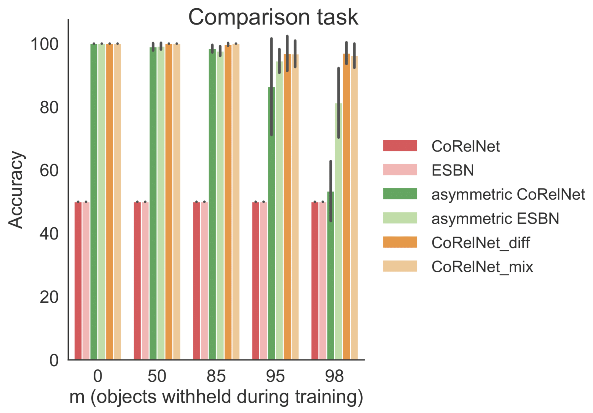


Fig. 1. Example of the 'comparison task'. Here the larger object is at position '0', hence the answer is '0'.



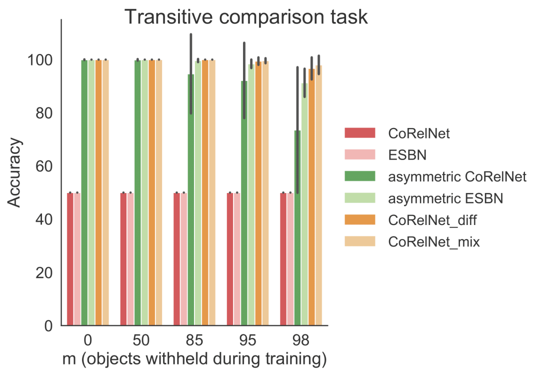


Fig. 2. Results reflect test accuracies averaged over 10 seeds, for 'comparison task' (left) and 'transitive comparison task' (right).

of sizes (small, large) and (large, small). Hence one not only needs to be able to generalize the comparison task to unseen shapes but also be able to do so transitively.

Hence, we would like to revisit the use of a symmetric similarity measure and instead use an asymmetric one. One first step would be to use a modified similarity measure $z_{t'} \cdot W z_t$, where $W := W_1^\top W_2$ is a learned linear transformation between a "reference" and a "comparison" space. We implemented this modified similarity measure within `ESBN` and `CoRelNet`, and labelled it `'asymmetric ESBN'` and `'asymmetric CoRelNet'`. Results in Figure 2 indicate that this modified similarity measure enhances the ability to learn the comparison task compared to `ESBN` and `CoRelNet`. Another idea, is to project the encoded sensory inputs z_t down to a scalar via $z'_t = W \cdot z_t$, where W is a learned linear transformation $\mathbb{R}^{\dim(z_t)} \longmapsto \mathbb{R}$, and one constructs the mutual difference matrix D (instead of the similarity matrix via the dot-product), where $D_{ij} = z'_i - z'_j = W(z_i - z_j)$. flatten$(D)$ is then fed to the final output MLP. Let us denote this model by `'CoRelNet diff'`. It turns out that `'CoRelNet diff'` solves both the 'comparison' and 'transitive comparison' tasks. (See Fig. 2).

7.3. Competition of rules in relational reasoning

As we can see in Figure 2, some similarity measures work well on some tasks, and worse on others. This motivates the idea of using multiple candidate rules (each using a different similarity measure), computed independently within the same model. To get the final output, one uses a competition mechanism between the rules to pick the winning rule, the output of which then constitutes the input to the final output MLP. (The competition mechanisms involves rule signature

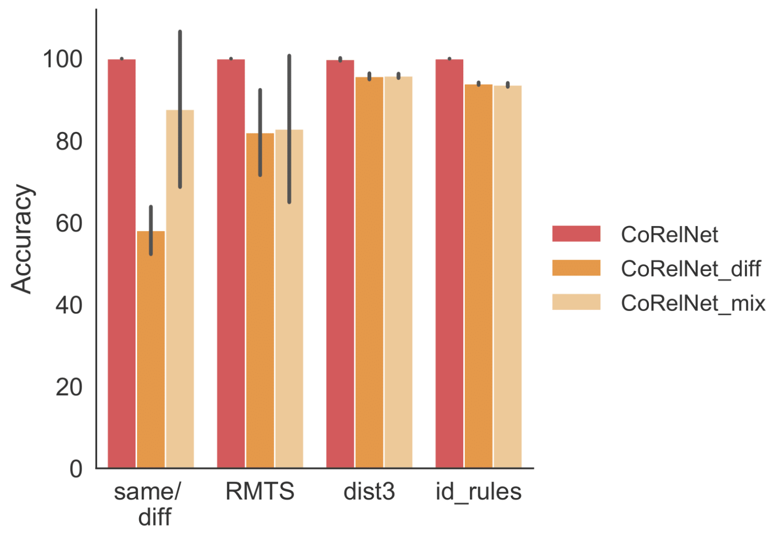


Fig. 3. Results reflect test accuracies averaged over 10 seeds, for relational cognitive tasks for the most extreme OOD cases $m = 98$ for same/diff and $m = 95$ for all other tasks.

embeddings as 'keys' and uses a learned linear transformation of the encoded sensory inputs as 'queries' to perform a key-query dot-product attention mechanism via a gumbel-softmax.) We denote this model by `'CoRelNet mix'` for the case where we consider two rules, one stemming from the difference matrix, and one from the dot-product similarity matrix. `'CoRelNet mix'` performs decent across a variety of tasks (see Figures 2 and 3). The high variance in the 'same/diff' and 'RMTS' tasks is due to the high sensitivity of the competition mechanism to initialization. Improving this competition mechanism constitutes an interesting avenue for future work. `'CoRelNet mix'` might give rise to interesting multi-task experiments involving a variety of tasks and relationship types.

7.4. Handling spurious features in relational reasoning

As already mentioned in Section 7.2, human similarity judgment involves selecting salient features in the reference object, and thus generalization in humans builds on the ability to pick up rules in environments with spurious or distracting features. Meanwhile, (Tran and Pashler, 2017) show that humans struggle to find relevant features in perceptual classification tasks, involving more than two dimensions along which stimuli vary. Moreover, (Goldstone et al., 1989) argue that in human cognition, featural similarity depends on or is influenced by other features being present, hinting towards the hypothesis that humans are instance-based learners doing nearest-neighbor classification, and that the ability to pick up relevant features among irrelevant ones comes from a lot of prior experience. Hence, one potential avenue to explore is using a multi-task set up or a curriculum learning approach to facilitate the encoder to pick up on relevant features in noisy environments, using a flexible top-down context-dependent attention mechanism.

7.5. Extending the notion of relevancy for long-range sequential processing

The relevancy screening mechanism outlined in Chapter 4 uses a hard-coded screening heuristic $C(i)$, which might potentially overemphasize "local" attention scores. A more agnostic approach might involve learning or meta-learning $C(i)$ end-to-end, either by leveraging predictive coding principles emphasizing more on "surprising events", or via RL based methods inspired by the Gating Model (Braver and Cohen, 1999; Todd et al., 2008).

References

Al-Mohy, A. and Higham, N. (2008). Computing the fréchet derivative of the matrix exponential, with an application to condition number estimation. *SIAM J. Matrix Analysis Applications*, 30:1639–1657.

Al-Mohy, A. and Higham, N. (2009). A new scaling and squaring algorithm for the matrix exponential. *SIAM Journal on Matrix Analysis and Applications*, 31.

Alberini, C., Johnson, S., and Ye, X. (2013). *Memory Reconsolidation*, pages 81–117.

Anderson, E., Bai, Z., Bischof, C., Blackford, L. S., Demmel, J., Dongarra, J. J., Du Croz, J., Hammarling, S., Greenbaum, A., McKenney, A., and Sorensen, D. (1999). *LAPACK Users' Guide (Third Ed.)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Arjovsky, M., Shah, A., and Bengio, Y. (2016). Unitary evolution recurrent neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1120–1128. JMLR.org.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Barsalou, L. W., Kyle Simmons, W., Barbey, A. K., and Wilson, C. D. (2003). Grounding conceptual knowledge in modality-specific systems. *Trends in Cognitive Sciences*, 7(2):84–91.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

Baxter, J. (2000). A model of inductive bias learning. *J. Artif. Int. Res.*, 12(1):149–198.

Bengio, Y., Deleu, T., Rahaman, N., Ke, R., Lachapelle, S., Bilaniuk, O., Goyal, A., and Pal, C. (2019). A meta-transfer objective for learning to disentangle causal mechanisms. *arXiv:1901.10912*.

Bengio, Y., Frasconi, P., and Simard, P. (1993). Problem of learning long-term dependencies in recurrent networks. pages 1183 – 1188 vol.3.

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

Braver, T. S. and Cohen, J. D. (1999). Dopamine, cognitive control, and schizophrenia: the gating model. *Progress in brain research*, 121:327–349.

Britz, D., Goldie, A., Luong, M., and Le, Q. V. (2017). Massive exploration of neural machine translation architectures. *CoRR*, abs/1703.03906.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Casado, M. L. and Martínez-Rubio, D. (2019). Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. *CoRR*, abs/1901.08428.

Chandar, S., Sankar, C., Vorontsov, E., Kahou, S. E., and Bengio, Y. (2019). Towards Non-saturating Recurrent Units for Modelling Long-term Dependencies. *AAAI*.

Chang, B., Chen, M., Haber, E., and Chi, E. H. (2019). AntisymmetricRNN: A Dynamical System View on Recurrent Neural Networks. *ICLR*.

Chen, M., Pennington, J., and Schoenholz, S. S. (2018). Dynamical Isometry and a Mean Field Theory of RNNs: Gating Enables Signal Propagation in Recurrent Neural Networks. *ICML*.

Chevalier-Boisvert, M. and Willems, L. (2018). Minimalistic gridworld environment for openai gym. `https://github.com/maximecb/gym-minigrid`.

Child, R., Gray, S., Radford, A., and Sutskever, I. (2019). Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Cho, K., van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*.

Cohen, N. and Eichenbaum, H. (1993). *Memory, amnesia, and the hippocampal system*. The MIT Press.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, L. (2018). Universal transformers. *CoRR*, abs/1807.03819.

Deleu, T., Kanaa, D., Feng, L., Kerg, G., Bengio, Y., Lajoie, G., and Bacon, P.-L. (2022). Continuous-time meta-learning with forward mode differentiation. In *International Conference on Learning Representations*.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Fortunato, M., Tan, M., Faulkner, R., Hansen, S., Badia, A. P., Buttimore, G., Deck, C., Leibo, J. Z., and Blundell, C. (2019). Generalization of reinforcement learners with working and episodic memory. In *Advances in Neural Information Processing Systems*, pages 12469–12478.

Ganguli, S., Huh, D., and Sompolinsky, H. (2008). Memory traces in dynamical systems. *Proceedings of the National Academy of Sciences*, 105(48):18970–18975.

Geadah, V., Horoi, S., Kerg, G., Wolf, G., and Lajoie, G. (2022a). Goal-driven optimization of single-neuron properties in artificial networks reveals regularization role of neural diversity and adaptation. *bioRxiv*.

Geadah, V., Kerg, G., Horoi, S., Wolf, G., and Lajoie, G. (2020). Advantages of biologically-inspired adaptive neural activation in rnns during learning.

Geadah, V., Kerg, G., Horoi, S., Wolf, G., and Lajoie, G. (2022b). Top-down optimization recovers biological coding principles of single-neuron adaptation in rnns. In *Computational and Systems Neuroscience (COSYNE)*.

Geadah, V., Lajoie, G., Kerg, G., Horoi, S., and Wolf, G. (2021). Network-level computational advantages of single-neuron adaptation. In *Computational and Systems Neuroscience (COSYNE)*.

Goldman, M. S. (2009). Memory without Feedback in a Neural Network. *Neuron*, 61(4):621–634.

Goldstone, R. L. and Barsalou, L. W. (1998). Reuniting perception and conception. *Cognition*, 65(2):231–262.

Goldstone, R. L., Gentner, D., and Medin, D. L. (1989). Relations relating relations. In *In Proceedings (tf the Eleventh Annual Conference of the Cognitive Science Society (pp. 131 - 138*.

Goodale, M. A. and Milner, A. D. (1992). Separate visual pathways for perception and action. *Trends in neurosciences*, 15(1):20–25.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Goyal, A. and Bengio, Y. (2020). Inductive biases for deep learning of higher-level cognition.

Goyal, A., Lamb, A., Hoffmann, J., Sodhani, S., Levine, S., Bengio, Y., and Schölkopf, B. (2019). Recurrent independent mechanisms. *arXiv preprint arXiv:1909.10893*.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S. G., Grefenstette, E., Ramalho, T., Agapiou, J., et al. (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476.

Harutyunyan, A., Dabney, W., Mesnard, T., Azar, M. G., Piot, B., Heess, N., van Hasselt, H. P., Wayne, G., Singh, S., Precup, D., and Munos, R. (2019). Hindsight Credit Assignment. pages 12467–12476.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Helfrich, K., Willmott, D., and Ye, Q. (2018). Orthogonal recurrent neural networks with scaled Cayley transform. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1969–1978, Stockholmsmässan, Stockholm Sweden. PMLR.

Henaff, M., Szlam, A., and LeCun, Y. (2016). Recurrent orthogonal networks and long-memory tasks. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2034–2042, New York, New York, USA. PMLR.

Hennequin, G., Vogels, T. P., and Gerstner, W. (2012). Non-normal amplification in random balanced neuronal networks. *Phys. Rev. E*, 86:011909.

Henrici, P. (1962). Bounds for iterates, inverses, spectral variation and fields of values of non-normal matrices. *Numer. Math.*, 4(1):24–40.

Hill, F., Tieleman, O., von Glehn, T., Wong, N., Merzic, H., and Clark, S. (2020). Grounded language learning fast and slow. *arXiv preprint arXiv:2009.01719*.

Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen netzen [in german] diploma thesis. *TU Münich*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359 – 366.

Hudson, D. A. and Manning, C. D. (2018). Compositional attention networks for machine reasoning. *arXiv preprint arXiv:1803.03067*.

Hung, C.-C., Lillicrap, T., Abramson, J., Wu, Y., Mirza, M., Carnevale, F., Ahuja, A., and Wayne, G. (2019). Optimizing agent behavior over long time scales by transporting value. *Nature Communications*, 10(1).

Hyland, S. and Rätsch, G. (2017). Learning unitary operators with help from u(n).

Jastrzebski, S., Arpit, D., Astrand, O., Kerg, G., Wang, H., Xiong, C., Socher, R., Cho, K., and Geras, K. (2021). Catastrophic fisher explosion: Early phase fisher matrix impacts generalization. In *38th International Conference on Machine Learning (ICML)*.

Jiang, B. and Dai, Y.-H. (2015). A framework of constraint preserving update schemes for optimization on Stiefel manifold. *Mathematical Programming*, 153(2):1–41.

Jing, L., Gulcehre, C., Peurifoy, J., Shen, Y., Tegmark, M., Soljacic, M., and Bengio, Y. (2019). Gated orthogonal recurrent units: On learning to forget. *Neural computation*, 31(4):765–783.

Jing, L., Shen, Y., Peurifoy, J., Skirlo, S., LeCun, Y., and Tegmark, M. (2017). Tunable Efficient Unitary Neural Networks (EUNN) and their application to RNNs. *ICML*.

Johnson, J., Hariharan, B., Van Der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R.

(2017). Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910.

Jones, M. and Love, B. C. (2007). Beyond common features: The role of roles in determining similarity. *Cognitive Psychology*, 55(3):196–231.

Kahan, W. (2006). Is there a small skew cayley transform with zero diagonal? *Linear Algebra and its Applications*, 417(2):335 – 341. Special Issue in honor of Friedrich Ludwig Bauer.

Kanuparthi, B., Arpit, D., Kerg, G., Ke, N. R., Mitliagkas, I., and Bengio, Y. (2019). h-detach: Modifying the LSTM gradient towards better optimization. In *International Conference on Learning Representations*.

Ke, N. R., Didolkar, A. R., Mittal, S., Goyal, A., Lajoie, G., Bauer, S., Rezende, D. J., Mozer, M. C., Bengio, Y., and Pal, C. (2021). Systematic evaluation of causal discovery in visual model based reinforcement learning.

Ke, N. R., Goyal, A., Bilaniuk, O., Binas, J., Mozer, M. C., Pal, C., and Bengio, Y. (2018). Sparse attentive backtracking: Temporal credit assignment through reminding. In *Advances in Neural Information Processing Systems*, pages 7640–7651.

Kerg, G., Goyette, K., Touzel, M. P., Gidel, G., Vorontsov, E., Bengio, Y., and Lajoie, G. (2019). Non-normal recurrent neural network (nnrnn): learning long time dependencies while improving expressivity with transient dynamics. *CoRR*, abs/1905.12080.

Kerg, G., Kanuparthi, B., Goyal, A., Goyette, K., Bengio, Y., and Lajoie, G. (2020). Untangling tradeoffs between recurrence and self-attention in artificial neural networks. *Advances in Neural Information Processing Systems*, 33.

Kerg, G., Mittal, S., Rolnick, D., Bengio, Y., Richards, B., and Lajoie, G. (2022). On neural architecture inductive biases for relational tasks.

Kim, N. and Linzen, T. (2020). Cogs: A compositional generalization challenge based on semantic interpretation. *arXiv preprint arXiv:2010.05465*.

Kim J, Ricci M, S. T. (2018). Not-so-clevr: learning same–different relations strains feedforward neural networks. *Royal Society*.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. (2018). Neural relational inference for interacting systems. *arXiv preprint arXiv:1802.04687*.

Kitaev, N., Kaiser, Ł., and Levskaya, A. (2020). Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Kriete, T., Noelle, D. C., Cohen, J. D., and O'Reilly, R. C. (2013). Indirection and symbol-like processing in the prefrontal cortex and basal ganglia. *Proceedings of the National Academy of Sciences*, 110(41):16390–16395.

L White, O., Lee, D., and Sompolinsky, H. (2004). Short-term memory in orthogonal neural networks. *Physical review letters*, 92:148102.

Lax, P. D. (2007). *Linear Algebra and Its Applications*. John Wiley & Sons.

Le, Q., Sarlos, T., and Smola, A. (2013). Fastfood - approximating kernel expansions in loglinear time. In *30th International Conference on Machine Learning (ICML)*.

Le, Q. V., Jaitly, N., and Hinton, G. E. (2015). A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.

Lezcano-Casado, M. and Martínez-Rubio, D. (2019). Cheap Orthogonal Constraints in Neural Networks: A Simple Parametrization of the Orthogonal and Unitary Group. *ICML*.

Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. (2020). Object-centric learning with slot attention. *arXiv preprint arXiv:2006.15055*.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Maduranga, K. D. G., Helfrich, K. E., and Ye, Q. (2019). Complex Unitary Recurrent Neural Networks using Scaled Cayley Transform. *AAAI*.

Marcus, G., Vijayan, S., Rao, S. B., and Vishton, P. (1999). Rule learning by seven-month-old infants. *Science*.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

Merity, S., Keskar, N. S., and Socher, R. (2018). An Analysis of Neural Language Modeling at Multiple Scales. *arXiv preprint arXiv:1803.08240*.

Mhammedi, Z., Hellicar, A., Rahman, A., and Bailey, J. (2017). Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2401–2409, International Convention Centre, Sydney, Australia. PMLR.

Mittal, S., Lamb, A., Goyal, A., Voleti, V., Shanahan, M., Lajoie, G., Mozer, M., and Bengio, Y. (2020). Learning to combine top-down and bottom-up signals in recurrent neural networks with attention over modules. In *International Conference on Machine Learning*, pages 6972–6986. PMLR.

Mittal, S., Raparthy, S. C., Rish, I., Bengio, Y., and Lajoie, G. (2021). Compositional attention: Disentangling search and retrieval. *arXiv preprint arXiv:2110.09419*.

Munkhdalai, T., Sordoni, A., WANG, T., and Trischler, A. (2019). Metalearned Neural Memory. pages 13331–13342.

Newman, B., Hewitt, J., Liang, P., and Manning, C. D. (2020). The eos decision and length extrapolation. *arXiv preprint arXiv:2010.07174*.

Nogueira, R., Jiang, Z., and Lin, J. (2021). Investigating the limitations of transformers with simple arithmetic tasks. *arXiv preprint arXiv:2102.13019*.

Nosofsky, R. M. (1991). Stimulus bias, asymmetric similarity, and classification. *Cognitive Psychology*, 23(1):94–140.

Oord, A. v. d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*.

Orhan, A. E. and Pitkow, X. (2019). Improved memory in recurrent neural networks with sequential non-normal dynamics. *arXiv.org*.

Ottmar, E., Weitnauer, E., Landy, D., and Goldstone, R. (2015). *Graspable Mathematics: Using Perceptual Learning Technology*, page 24.

P. A. Carpenter, M. A. Just, P. S. (1990). What one intelligence test measures: A theoretical account of the processing in the raven progressive matrices test. *Psychological Review*.

Parcollet, T., Ravanelli, M., Morchid, M., Linarès, G., Trabelsi, C., De Mori, R., and Bengio, Y. (2019). Quaternion Recurrent Neural Networks. *ICLR*.

Parikh, A. P., Täckström, O., Das, D., and Uszkoreit, J. (2016). A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., and Ku, A. (2018). Image transformer. *International Conference on Machine Learning*.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013a). On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013b). On the difficulty of training recurrent neural networks. *ICML*.

Paulus, R., Xiong, C., and Socher, R. (2017). A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Pennington, J., Schoenholz, S. S., and Ganguli, S. (2017). Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. *Advances in Neural Information Processing Systems*.

Pratap, V., Hannun, A., Xu, Q., Cai, J., Kahn, J., Synnaeve, G., Liptchinsky, V., and Collobert, R. (2019). Wav2letter++: A fast open-source speech recognition system. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6460–6464. IEEE.

Pritzel, A., Uria, B., Srinivasan, S., Puigdomenech, A., Vinyals, O., Hassabis, D., Wierstra, D., and Blundell, C. (2017). Neural episodic control. *arXiv preprint arXiv:1703.01988*.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Radvansky, G. A. and Zacks, J. M. (2017). Event boundaries in memory and cognition. *Current opinion in behavioral sciences*, 17:133–140.

Raghu, M., Poole, B., Kleinberg, J., Ganguli, S., and Dickstein, J. S. (2017). On the expressive power of deep neural networks. In *ICML*.

Raven, J. C. and Court, J. (1938). *Raven's progressive matrices*. Western Psychological Services Los Angeles, CA.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*.

Santoro, A., Faulkner, R., Raposo, D., Rae, J., Chrzanowski, M., Weber, T., Wierstra, D., Vinyals, O., Pascanu, R., and Lillicrap, T. (2018). Relational recurrent neural networks. In *Advances in Neural Information Processing Systems 31*.

Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. (2017). A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976.

Sard, A. (1942). The measure of the critical values of differentiable maps. *Bull. Amer. Math. Soc.*, 48(12):883–890.

Saxe, A. M., McClelland, J. L., and Ganguli, S. (2014). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *ICLR*.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80.

Schmidhuber, J. and Hochreiter, S. (1997). Long short-term memory. *Neural Computation*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms.

Sedda, A. and Scarpina, F. (2012). Dorsal and ventral streams across sensory modalities. *Neuroscience bulletin*, 28(3):291–300.

Sejnowski, T. J. (2018). *The deep learning revolution*. MIT press.

Shanahan, M., Nikiforou, K., Creswell, A., Kaplanis, C., Barrett, D., and Garnelo, M. (2019). An explicitly relational neural network architecture. *arXiv preprint arXiv:1905.10307*.

Sukhbaatar, S., szlam, a., Weston, J., and Fergus, R. (2015). End-to-end memory networks. In *Advances in Neural Information Processing Systems 28*, pages 2440–2448.

Tagare, H. D. (2011). Notes on optimization on stiefel manifolds. Techincal Report, Yale University.

Tallec, C. and Ollivier, Y. (2018). Can recurrent neural networks warp time? *ICLR*.

Todd, M., Niv, Y., and Cohen, J. D. (2008). Learning to use working memory in partially observable environments through dopaminergic reinforcement. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc.

Tolman, E. (1948). Cognitive maps in rats and men. *Psychological Review*, 55(4):189–208.

Tran, R. and Pashler, H. (2017). Learning to exploit a hidden predictor in skill acquisition: Tight linkage to conscious awareness. *PLOS ONE*, 12(6):1–17.

Tversky, A. (1977). Features of similarity. *Psychological Review*, 84(4):327–352.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Velicković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Vorontsov, E., Trabelsi, C., Kadoury, S., and Pal, C. (2017). On orthogonality and learning rnn with long term dependencies. *ICML*.

Webb, T. W., Dulberg, Z., Frankland, S. M., Petrov, A. A., O'Reilly, R. C., and Cohen, J. D. (2020). Learning representations that support extrapolation. *Proceedings of the 37th International Conference on Machine Learning*.

Webb, T. W., Sinha, I., and Cohen, J. D. (2021). Emergent symbols through binding in external memory. *The International Conference on Learning Representations (ICLR)*.

Whittington, J. C., Muller, T. H., Mark, S., Chen, G., Barry, C., Burgess, N., and Behrens, T. E. (2019). The tolman-eichenbaum machine: Unifying space and relational memory through generalisation in the hippocampal formation. *BioRxiv*, page 770495.

Wilson, D. R. and Martinez, T. R. (2003). The general inefficiency of batch training for gradient descent learning. *Neural Netw.*, 16(10):1429–1451.

Wisdom, S., Powers, T., Hershey, J., Le Roux, J., and Atlas, L. (2016). Full-capacity unitary recurrent neural networks. In *Advances in Neural Information Processing Systems 29*, pages 4880–4888.

Wolpert, D. H. and Macready, W. G. (1995). No free lunch theorems for search. volume Technical Report SFI-TR-95-02-010. Santa Fe Institute.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764.

Yang, Z., Moczulski, M., Denil, M., d. Freitas, N., Smola, A., Song, L., and Wang, Z. (2015). Deep fried convnets. In *ICCV*.

Yi, K., Gan, C., Li, Y., Kohli, P., Wu, J., Torralba, A., and Tenenbaum, J. B. (2019). Clevrer: Collision events for video representation and reasoning. *arXiv preprint arXiv:1910.01442*.

Zacks, J. M., Speer, N. K., Swallow, K. M., Braver, T. S., and Reynolds, J. R. (2007). Event perception: a mind-brain perspective. *Psychological bulletin*, 133(2):273.

Zhang, C., Gao, F., Jia, B., Zhu, Y., and Zhu, S.-C. (2019). Raven: A dataset for relational and analogical visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5317–5327.

Zhang, J., Lei, Q., and Dhillon, I. (2018). Stabilizing Gradients for Deep Neural Networks via Efficient SVD Parameterization. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, pages 5806–5814, Stockholmsmässan, Stockholm Sweden. PMLR.

Zhao, G., Lin, J., Zhang, Z., Ren, X., Su, Q., and Sun, X. (2019). Explicit sparse transformer: Concentrated attention through explicit selection. *arXiv preprint arXiv:1912.11637*.

Appendix A

Appendix for 'Non-normal Recurrent Neural Network (nnRNN): learning long time dependencies while improving expressivity with transient dynamics'

A.1. Task setup and training details

All code freely available at `https://github.com/nnRNN/nnRNN_release`.

A.1.1. Copy task

For the copy task, networks are presented with an input sequence x_t of length $10 + T_c$. For $t = 1, \ldots, 10$, x_t can take one of 8 distinct values $\{a_i\}_{i=1}^{8}$. For the following $T_c - 1$ time steps, x_t takes the same value a_9. At $t = T_c$, a cue symbol $x_t = a_{10}$ prompts the model to recall the first 10 symbols and output them sequentially in the same order they were presented. Models are trained to minimize the average cross entropy loss of symbol recalls. A model that simply predicts a constant set of output tokens for every input sequence would achieve a *baseline* loss of $\frac{10\log(8)}{T+20}$. All models were trained using a mini batch size of 10. All non-gated models except "RNN" were initialized such that the recurrent network was orthogonal. The non-normal RNN had it's orthogonal weight matrix initialized as in expRNN with the log weights initialized using Henaff intialization. Importantly, all non-gated models used the *modReLU* activation function for state-to-state transitions. This is critical for the copy task since a nonlinearity makes the task very difficult to solve (Vorontsov et al., 2017) and *modReLU* acts as identity at initialization. Fig. 1 (left) shows cross entropy loss for all models throughout training when the number of parameters is held constant. Model and training hyperparameters are summarized in Table 1.

A.1.2. Sequential MNIST classification task

The sequential MNIST task (Le et al., 2015) measures the ability of an RNN to model complex long term dependencies. In this task, each pixel is fed into the network one at a time, after

Model	hid	LR	LR orth	α	δ	T decay	V init
nnRNN	128	0.0005	10^{-6}	0.99	0.0001	10^{-6}	Henaff
expRNN	128	0.001	0.0001	0.99			Henaff
expRNN	176	0.001	0.0001	0.99			Henaff
LSTM	128	0.0005		0.99			Glorot Normal
LSTM	63	0.001		0.99			Glorot Normal
RNN Orth	128	0.0002		0.99			Random orth
EURNN	128	0.001		0.5			
EURNN	256	0.001		0.5			
RNN	128	0.001		0.9			Glorot Normal

Table 1. Hyperparameters for the copy task. Here, "hid" is hidden state size, "LR" is learning rate, "LR orth" is the learning rate of the orthogonal transition matrix (its skew symmetric matrix), α is the smoothing parameter of RMSprop, δ is as in equation 3.5.1, T decay is the weight of the L2 penalty applied on T in equation 3.5.1, and "V init" is the initialization scheme for the state transition matrix.

which the network must classify the digit. Permutation increases the difficulty of the problem by applying a fixed permutation to the sequence of the pixels, which creates longer term dependencies between the pixels. We train this task for all networks using mini batch sizes of 100. All non-gated networks except "RNN" were initialized with orthogonal recurrent weight matrices using Cayley initialization(Helfrich et al., 2018). The non-normal RNN has it's orthogonal weight matrix initialized as in Lezcano-Casado and Martínez-Rubio (2019) with the log weights initialized using Cayley initialization. Fig. 1 (right) shows validation accuracy for all models throughout training when the number of parameters is held constant. Model and training hyperparameters are summarized in Table 2.

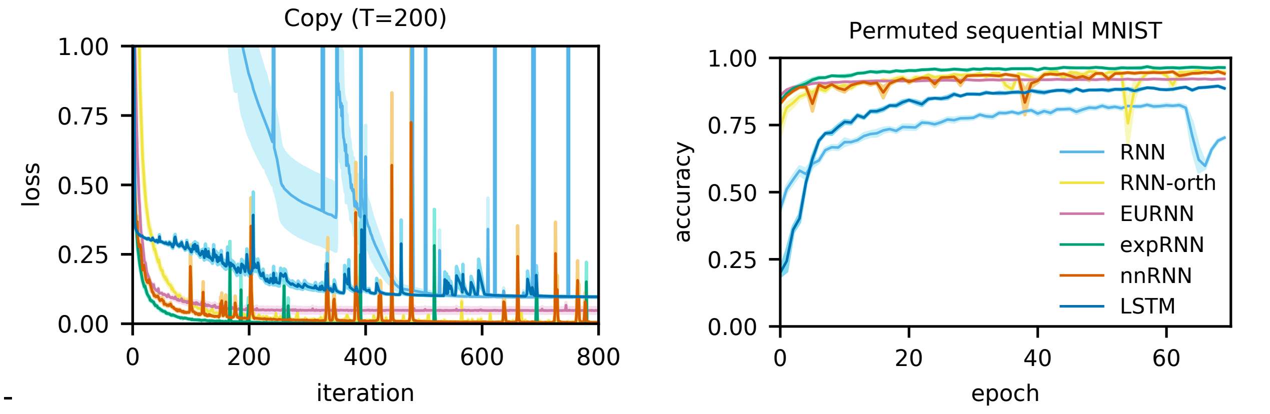


Fig. 1. Holding the number of parameters constant, model performance is plotted for the copy task (T=200, left; cross-entropy loss; 18.9K parameters) and for the permuted sequential MNIST task (right; accuracy; 269K parameters). Shading indicates one standard error of the mean.

Model	hid	LR	LR orth	α	δ	T decay	V init
nnRNN	512	0.00015	$1.5 * 10^{-5}$	0.99	0.15	0.0001	Cayley
expRNN	512	0.0005	$5 * 10^{-5}$	0.99			Cayley
expRNN	722	0.0005	$5 * 10^{-5}$	0.99			Cayley
LSTM	512	0.0005		0.9			Glorot Normal
LSTM	257	0.0005		0.9			Glorot Normal
RNN Orth	512	$5 * 10^{-5}$		0.99			Random orth
EURNN	512	0.0001		0.9			
EURNN	1024	0.0001		0.9			
RNN	512	0.0001		0.9			Glorot Normal

Table 2. Hyperparameters for the permuted sequential mnist task. Here, "hid" is hidden state size, "LR" is learning rate, "LR orth" is the learning rate of the orthogonal transition matrix (its skew symmetric matrix), α is the smoothing parameter of RMSprop, δ is as in equation 3.5.1, T decay is the weight of the L2 penalty applied on T in equation 3.5.1, and "V init" is the initialization scheme for the state transition matrix.

A.1.3. Penn Tree Bank character prediction task

The Penn Tree Bank character prediction task is that of predicting the next character in a text corpus at every character position, given all previous text. Code is adapted from Merity et al. (2018). We trained all models sequentially on the entire corpus, splitting it into sequences of length 150 or 300 for truncated backpropagation through time. Consequently, the initial hidden state for a sequence is the last hidden state produced from its preceding sequence. All models were trained for 100 epochs with a mini batch size of 128. Following training, for each model, the state which yielded the best performance on the validation data was evaluated on the test data. Table 2 reports the same performance for the same model states as in Table 1 in the main text but presents test accuracy instead of BPC. Model and training hyperparameters are summarized in Table 4.

	Test Accuracy			
	Fixed # params ($\sim$1.32M)		Fixed # hidden units ($N = 1024$)	
Model	$T_{PTB} = 150$	$T_{PTB} = 300$	$T_{PTB} = 150$	$T_{PTB} = 300$
RNN	40.01 ± 0.026	39.97 ± 0.025	40.01 ± 0.026	39.97 ± 0.025
RNN-orth	66.29 ± 0.07	65.53 ± 0.09	66.29 ± 0.07	65.53 ± 0.09
EURNN	65.68 ± 0.002	65.55 ± 0.002	64.01 ± 0.002	64.20 ± 0.003
expRNN	68.07 ± 0.15	67.58 ± 0.04	67.51 ± 0.11	66.89 ± 0.024
nnRNN	**68.78 ± 0.0006**	**68.52 ± 0.0004**	**68.78 ± 0.0006**	**68.52 ± 0.0004**

Table 3. PTB test performance: Test Accuracy, for sequence lengths $T_{PTB} = 150, 300$. Two comparisons across models shown: fixed number of parameters (left), and fixed number of hidden units (right). Error range indicates standard error of the mean.

Model	hid	LR	LR orth	α	δ	T decay	V init
Length 150							
nnRNN	1024	0.0008	$8 * 10^{-5}$	0.9	1	0.0001	Cayley
expRNN	1024	0.005	0.0001	0.9			Cayley
expRNN	1386	0.005	0.0001	0.9			Cayley
LSTM	1024	0.008		0.9			Glorot Normal
LSTM	475	0.001		0.99			Glorot Normal
RNN Orth	1024	0.0001		0.9			Random orth
EURNN	1024	0.001		0.9			
EURNN	2048	0.001		0.9			
RNN	1024	10^{-5}		0.9			Glorot Normal
Length 300							
nnRNN	1024	0.0008	$6 * 10^{-5}$	0.9	0.0001	0.0001	Cayley
expRNN	1024	0.005	0.0001	0.9			Cayley
expRNN	1386	0.005	0.0001	0.9			Cayley
LSTM	1024	0.008		0.9			Glorot Normal
LSTM	475	0.003		0.9			Glorot Normal
RNN Orth	1024	0.0001		0.9			Cayley
EURNN	1024	0.001		0.9			
EURNN	2048	0.001		0.9			
RNN	1024	$1 * 10^{-5}$		0.9			Glorot Normal

Table 4. Hyperparameters for the Penn Tree Bank task (at 150 and 300 time step truncation for gradient backpropagation). Here, "hid" is hidden state size, "LR" is learning rate, "LR orth" is the learning rate of the orthogonal transition matrix (its skew symmetric matrix), α is the smoothing parameter of RMSprop, δ is as in equation 3.5.1, T decay is the weight of the L2 penalty applied on T in equation 3.5.1, and "V init" is the initialization scheme for the state transition matrix.

A.1.4. Hyperparameter search

For all models with a state transition matrix that is initialized as orthogonal (nnRNN, expRNN, RNN-orth), three orthogonal initialization schemes were tested: (1) random, (2) Cayley, and (3) Henaff. Random initialization is achieved by sampling a random matrix whose QR decomposition yields an orthogonal matrix with positive determinant 1 and then mapping this orthogonal matrix via a matrix logarithm to the skew symmetric parameter matrix used in expRNN. Cayley and Henaff initializations initialize this skew symmetric matrix as described in Lezcano-Casado and Martínez-Rubio (2019). The vanilla RNN is also tested with a Glorot Normal initialization, with the model then referred to as simply "RNN".

For training, learning rates were searched between 0.01 and 0.0001 in increments of 0.0001, 0.0002 or $10\times$; the learning rate for the orthogonal matrix was always kept near $10\times$ lower; and RMSprop was used as the optimizer with smoothing parameter α as 0.5, 0.9, or 0.99. In equation 3.5.1, δ was searched in 0, 0.0001, 0.001, 0.01, 0.1, 0.15, 1.0, 10; the L2 decay on the strictly lower triangular part of the transition matrix T was searched in 0, 10^{-6}, 10^{-5}, 10^{-4}.

A.2. Fisher Memory Curves for strictly lower-triangular matrices

Let, Θ be a strictly lower triangular matrix such that $[\Theta]_{i+1,i} = \sqrt{\alpha}$ for $1 \leq i \leq N-1$ and A be the associated lower triangular Gram-Schmidt orthogonalization matrix. We have that,

$$\Theta = DA \tag{A.2.1}$$

where D is the delay line, $D_{i+1,i} = \sqrt{\alpha}$ and $A_{i,i} = 1$ for $1 \leq i \leq N$. Let us recall the expression of $J(k)$ for independent Gaussian noise derived by (Ganguli et al., 2008, Eq. 3),

$$J(k) = U^T (\Theta^k)^\top C_n^{-1} \Theta^k U\,, \quad \text{where} \quad C_n = \epsilon \sum_{k=0}^{\infty} \Theta^k (\Theta^k)^\top\,, \tag{A.2.2}$$

and $U = [1,0,\dots,0]$ is the source. We have that for any vector u,

$$u^\top C_n u = \epsilon \sum_{k=0}^{\infty} ((D^k)^\top u)^\top A A^T ((D^k)^\top u) \tag{A.2.3}$$

$$= \epsilon \sum_{k=0}^{N-1} ((D^k)^\top u)^\top A A^T ((D^k)^\top u) \tag{A.2.4}$$

$$\leq \epsilon \sigma_{\max}^{2(N-1)}(A) \sum_{k=0}^{N-1} u^\top D^k (D^k)^\top u \tag{A.2.5}$$

where for the first equality we used the fact that Θ is nilpotent and for the last inequality the fact that $\sigma_{\max}(A) \geq 1$. Recall that for two symmetric matrices we define: $A \succeq B$ if and only if $B - A$ is positive semidefinite. By definition we have,

$$C_n \preceq \epsilon \sigma_{\max}^{2(N-1)}(A) \sum_{k=0}^{\infty} D^k (D^k)^\top = \epsilon \sigma_{\max}^{2}(A) \left(\mathrm{diag}(1, \tfrac{1-\alpha^2}{1-\alpha}, \dots, \tfrac{1-\alpha^N}{1-\alpha}) \right) \tag{A.2.6}$$

where the last equality is due to $[D^k (D^k)^\top]_{i,j} = \alpha^k$ if $i = j \geq k+1$ and 0 otherwise. Thus using (Lax, 2007, Theorem 2 P. 146) we can take the inverse to get,

$$C_n^{-1} \succeq \frac{1}{\epsilon \sigma_{\max}^{2(N-1)}(A)} \left(\mathrm{diag}(1, \tfrac{1-\alpha^2}{1-\alpha}, \dots, \tfrac{1-\alpha^N}{1-\alpha}) \right)^{-1} = \frac{1}{\epsilon \sigma_{\max}^{2(N-1)}(A)} \mathrm{diag}(1, \tfrac{1-\alpha}{1-\alpha^2}, \dots, \tfrac{1-\alpha}{1-\alpha^N})$$

Finally, using that $\Theta^k U = [\underbrace{0,\dots,0}_{k}, \sqrt{\alpha}^k, *, \dots, *]$, we have that for $0 \leq k \leq N-1$,

$$J(k) = U^T (\Theta^k)^\top C_n^{-1} \Theta^k U \tag{A.2.7}$$

$$\geq \frac{1}{\epsilon \sigma_{\max}^{2(N-1)}(A)} \alpha^k \frac{\alpha - 1}{\alpha^{k+1} - 1}\,. \tag{A.2.8}$$

α	β	d	$J_{\text{tot}} = \sum_{t=0}^{\infty} J(t)$
0.95	0.0	0.0	3.03
1.00	0.0	0.0	5.19
1.05	0.0	0.0	12.1
0.95	0.005	0.0	3.18
1.00	0.005	0.0	5.30
1.05	0.005	0.0	12.1
0.95	0.0	0.2	12.0
1.00	0.0	0.2	16.2
1.05	0.0	0.2	20.5
0.95	0.005	0.2	12.1
1.00	0.005	0.2	16.3
1.05	0.005	0.2	20.4

Table 5. Fisher memory curve performance: Shown is the sum of the FMC for the models considered in §3.4.

A.3. Proof of proposition 3.4.2

Let us prove this claim by induction on t. The case $t = 1$ is trivial, so let us assume the claim to be true for $t = r$, then for each $k = 1, \dots n-1$, we expand A^{r+1} as $A^r \cdot A$ and get

$$p_k^{(r+1)}(x) = x \cdot \left[1 + \sum_{s=1}^{k-1} p_s^{(r)}(x)\right] + p_k^{(r)}(x)$$

which again is a polynomial of degree at most k, where the coefficient of x^0 in $p_k^{(r+1)}$ is zero and the coefficient of x^l in $p_k^{(r+1)}$ is $O\left(\binom{r}{l-1}\right) + O\left(\binom{r}{l}\right) = O\left(\binom{r+1}{l}\right)$ for all $l = 1,2,\dots,k$, concluding the induction.

A.4. Numerical instablities of the Schur decomposition

The Schur decomposition is computed via multiple iterations of the QR algorithm. The QR algorithm is known to be *backward stable*, which gives accurate answers as long as the eigenvalues of the matrix at hand are well-conditioned, as is explained in Anderson et al. (1999).

Eigenvalue-sensitivity is measured by the angle formed between the left and right eigenvectors of the same eigenvalues. Normal matrices have coinciding left and right eigenvectors but non-normal matrices do not, and thus certain non-normal matrices such as the Grcar matrix have very high eigenvalue-sensitivity, and thus gives rise to inaccuracies in the Schur decomposition.

This motivates training the connectivity matrix in the Schur decomposition directly instead of applying the Schur decomposition in a separate step.

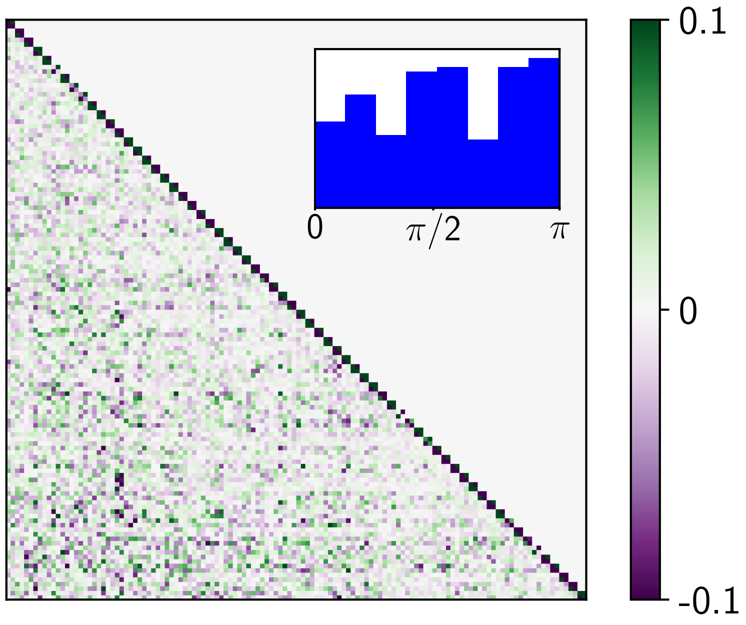


Fig. 2. *.Learned* Θ *on psMNIST task.* Inset: angles θ_i distribution of block diagonal rotations. (c.f. Eq.3.5).

A.5. Learned connectivity structure on psMNIST

For completeness, let us take a look at the Schur matrix after training on psMNIST in Fig. 2. We can see that the distribution of learned angles in the rotation blocks is rather flat, and thus is very different from the distribution learned in the PTB task, as can be seen in Fig. 3. The flatness in distribution comes somewhat close to the flatness of the learned angle distribution in the copy task. In other words, the angle distribution in the PTB task is highly structured, while in the Copy task and psMNIST task, it seems to be close to uniform.

Furthermore, we can also observe that the connectivity structure learned in the lower triangle is significantly weaker in the psMNIST task than in the PTB task, while not being completely absent as in the copy task.

Thus it seems that we can spot a spectrum of connectivity structure:

- the Copy task, with no connectivity structure in the lower triangle, close to uniform angle distribution and the absence of a delay line, on the one end.
- the PTB task, with a lot of connectivity structure in the lower triangle, a very narrow angle distribution and the presence of a delay line, on the other end.

For the psMNIST task, it appears that we are located somewhere in the middle of that spectrum.

A.6. Gradient propagation analysis

In the results from the main text, the parameter γ (scaling factor for eigenvalues) is allowed to be changed by the optimizer, but is heavily regularized to force it to stay close to one. As argued, a mean value of $\bar{\gamma} \sim 0.958$ found for the best solutions on the PTB task is indicative of a trade-off between eigenvalues and singular values to allow stable propagation and good expressivity. To further elucidate the effect, we train nnRNNs with clamped values of γ at 1, and at 0.958.

Results are found in Table 7 and complement those of Table 1 in the main text ($\sim$1.32M params, $N = 1024$ units). As expected for $\gamma = 1$, some run did not converge (asterisks indicate number out of 5) as the emergence of non-normal structure pushes singular values above one. Despite this, on runs that did converge we found the best performance out of all methods (including regularized γ nnRNN), strongly indicating that non-normality does indeed provide more expressivity. For γ clamped at 0.958 the performance was virtually identical to that of nnRNN with regularized γ, indicating non-normal connectivity learning appears robust and independent of γ learning.

Model	$T_{PTB} = 150$	$T_{PTB} = 300$
nnRNN	1.47 $\pm$ 0.003	1.49 $\pm$ 0.002
nnRNN-$\gamma = 1$	1.46 $\pm$ 0.005*	1.49 $\pm$ 0.022**
nnRNN-$\gamma = 0.958$	1.47 $\pm$ 0.005	1.49 $\pm$ 0.008

Table 6. PTB test performance: Bit per Character (BPC), for sequence lengths $T_{PTB} = 150, 300$. Three version of nnRNN shown: nnRNN (reproduced from main text), γ clamped at 1, γ clamped at 0.958. All models have $\sim$1.32M parameters and $N = 1024$. Error range indicates standard error of the mean. Asterisks indicate number of failed runs out of 5.

Fig. 3 shows example gradient norms for nnRNN on PTB task, with eigenvalues clamped or regularized. In this example, all runs converged, and we can observe that gradient norms behave nicely during backpropagation and throughout training. This is indicative that although γ plays an important stabilizing role, gradient explosion leading to diverging training runs appears to be an all-or-nothing event.

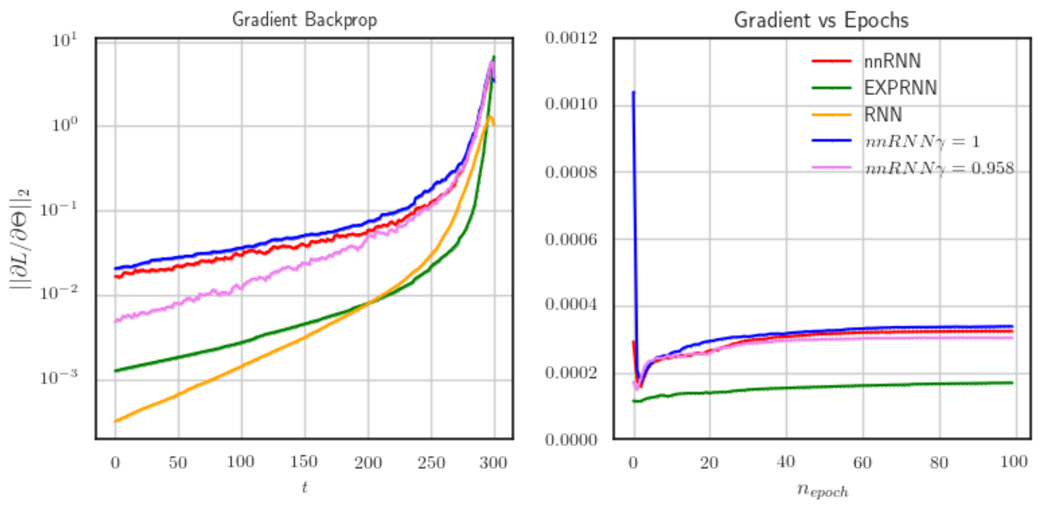


Fig. 3. *Gradient propagation*. The plot on the left shows the gradient magnitude while backpropagating from time step to time step (growing polynomially). The plot on the right shows the gradient magnitude during training

Appendix B

Appendix for 'Untangling tradeoffs between recurrence and self-attention in neural networks'

B.1. Theoretical analysis of gradient propagation

B.1.1. Notational convention

In this paper, we use the notation $\frac{df}{dx}$ to denote the total derivative of f with respect to x, and $\frac{\partial f}{\partial x}$ to denote the partial derivative of f with respect to x.

If we assume $f : \mathbb{R}^n \to \mathbb{R}^m$, and $x \in \mathbb{R}^n$, then $\frac{df}{dx}$ denotes the Jacobian matrix J_f such that

$$(J_f)_{ij} = \frac{df_i}{dx_j} \tag{B.1.1}$$

In particular, with this notation, we have that if a function $L : \mathbb{R}^m \to \mathbb{R}$, and $y \in \mathbb{R}^m$ then $\frac{dL}{dy}$ is a row vector, while the conventional notation for $\nabla_y L$ indicates a column vector. In other words, $(\nabla_y L)^T = \frac{dL}{dy}$. Hence if L is a function of $f(x)$, then

$$\frac{dL}{dx} = \frac{dL}{df} \cdot \frac{df}{dx} \tag{B.1.2}$$

while

$$\nabla_x L = \left(\frac{df}{dx}\right)^T \cdot \nabla_{f(x)} L = J_f^T \cdot \nabla_{f(x)} L \tag{B.1.3}$$

Similarly, we have that $\frac{\partial L}{\partial y}$ is a row vector.

B.1.2. Preliminary results

Let

$$s_t = \psi_t(h_1, h_2, \ldots, h_t, s_{t-1}) \tag{B.1.4}$$

where

$$h_{i+1} = \phi(V s_i + U x_{i+1} + b) \tag{B.1.5}$$

Lemma B.1.1. *For all* $t, k \geq 0$*, we have*

$$\frac{ds_{t+k+1}}{dh_t} = \frac{\partial s_{t+k+1}}{\partial h_t} + \left(\sum_{j=0}^{k} \frac{\partial s_{t+k+1}}{\partial h_{t+j+1}} \frac{dh_{t+j+1}}{dh_t} \right) + \frac{\partial s_{t+k+1}}{\partial s_{t+k}} \frac{ds_{t+k}}{dh_t} \tag{B.1.6}$$

PROOF. Follows directly from the following multivariable chain rule: if

$$g(t) = f(g_1(t), g_2(t), \ldots, g_n(t)) \tag{B.1.7}$$

then

$$\frac{dg}{dt} = \sum_{i=1}^{n} \frac{\partial f}{\partial g_i} \frac{dg_i}{dt} \tag{B.1.8}$$

□

Lemma B.1.2. *If we further denote the Jacobian matrix* $J_k = \frac{\partial s_{k+1}}{\partial h_k}$*, then we get that for all* $t, k \geq 0$*, we have*

$$\frac{ds_{t+k+1}}{dh_t} = \frac{\partial s_{t+k+1}}{\partial h_t} + \sum_{j=0}^{k} \left(\frac{\partial s_{t+k+1}}{\partial h_{t+j+1}} \cdot J_{t+j} + 1_{j=k} \cdot \frac{\partial s_{t+k+1}}{\partial s_{t+k}} \right) \cdot \frac{ds_{t+j}}{dh_t} \tag{B.1.9}$$

PROOF. Follows directly from the observation that

$$\frac{dh_{t+j+1}}{dh_t} = \frac{\partial h_{t+j+1}}{\partial s_{t+j}} \frac{ds_{t+j}}{dh_t} = J_{t+j} \cdot \frac{ds_{t+j}}{dh_t} \tag{B.1.10}$$

□

Remark B.1.3. Let us denote

$$C_{k+1}^{(t)} = \frac{ds_{t+k+1}}{dh_t} \tag{B.1.11}$$

$$E_{k+1}^{(t)} = \frac{\partial s_{t+k+1}}{\partial h_t} \tag{B.1.12}$$

and

$$F_{k+1,j}^{(t)} = \frac{\partial s_{t+k+1}}{\partial h_{t+j+1}} \cdot J_{t+j} + 1_{j=k} \cdot \frac{\partial s_{t+k+1}}{\partial s_{t+k}} \tag{B.1.13}$$

and thus the recursion formula in Lemma B.1.2 rewrites as

$$C_{k+1}^{(t)} = E_{k+1}^{(t)} + \sum_{j=0}^{k} F_{k+1,j}^{(t)} \cdot C_j^{(t)} \tag{B.1.14}$$

The next two results highlight how to solve this recursion.

Lemma B.1.4. *Let C_i,E_i, $F_{i,j} \in \mathbb{R}^{n\times n}$ such that for all $k \geq 0$, we have*

$$C_{k+1} = E_{k+1} + \sum_{j=0}^{k} F_{k+1,j} \cdot C_j \tag{B.1.15}$$

Then for all $k \geq 1$, we have

$$\boxed{C_k = \xi_{0:k} C_0 + \sum_{r=1}^{k} \xi_{r:k} E_r} \tag{B.1.16}$$

where

$$\xi_{r:k} = \sum_{s=1}^{k-r} \xi_{r:k}(s) \tag{B.1.17}$$

with

$$\xi_{r:k}(s) = \sum_{r=i_1<\ldots<i_{s+1}=k} F_{i_{s+1},i_s} \cdot F_{i_{s-1},i_{s-2}} \cdot \ldots \cdot F_{i_2,i_1} \tag{B.1.18}$$

and $\xi_{k:k} = Id$.

PROOF. Let us prove the statement by induction on $k \geq 1$.
For $k = 1$, we have

$$C_1 = E_1 + F_{1,0} C_0 = \xi_{1:1} E_1 + \xi_{0:1} C_0 \tag{B.1.19}$$

Now let us assume the statement to be true for k, then we get

$$C_{k+1} = E_{k+1} + \sum_{j=0}^{k} F_{k+1,j} \cdot \left(\xi_{0:j} C_0 + \sum_{r=1}^{j} \xi_{r:j} E_r \right) \tag{B.1.20}$$

$$= E_{k+1} + \left(\sum_{j=0}^{k} F_{k+1,j} \cdot \xi_{0:j} \right) \cdot C_0 + \sum_{j=0}^{k} \sum_{r=1}^{j} F_{k+1,j} \xi_{r:j} E_r \tag{B.1.21}$$

$$= E_{k+1} + \xi_{0:k+1} C_0 + \sum_{r=1}^{k} \left(\sum_{j=r}^{k} F_{k+1,j} \xi_{r:j} \right) \cdot E_r \tag{B.1.22}$$

$$= \xi_{k+1:k+1} E_{k+1} + \xi_{0:k+1} C_0 + \sum_{r=1}^{k} \xi_{r:k+1} E_r \tag{B.1.23}$$

$$= \xi_{0:k+1} C_0 + \sum_{r=1}^{k+1} \xi_{r:k+1} E_r \tag{B.1.24}$$

$$\tag{B.1.25}$$

□

Lemma B.1.5. *If we further assume that $C_0 = E_0$, then we have for all $k \geq 1$*

$$C_k = E_k + \sum_{s=1}^{k} \sum_{q=s}^{k} \xi_{k-q:k}(s) E_{k-q} \tag{B.1.26}$$

PROOF. Using the previous lemma, we get

$$C_k = E_k + \sum_{s'=1}^{k} \xi_{0:k}(s') C_0 + \sum_{r=1}^{k-1} \sum_{s=1}^{k-r} \xi_{r:k}(s) E_r \tag{B.1.27}$$

Using the assumption $C_0 = E_0$, we get

$$C_k = E_k + \sum_{s'=1}^{k} \xi_{0:k}(s') E_0 + \sum_{r=1}^{k-1} \sum_{s=1}^{k-r} \xi_{r:k}(s) E_r \tag{B.1.28}$$

$$= E_k + \sum_{r=0}^{k-1} \sum_{s=1}^{k-r} \xi_{r:k}(s) E_r \tag{B.1.29}$$

$$\tag{B.1.30}$$

Now let us put $q = k - r$, we get

$$C_k = E_k + \sum_{q=1}^{k} \sum_{s=1}^{q} \xi_{k-q:k}(s) E_{k-q} \tag{B.1.31}$$

$$= E_k + \sum_{s=1}^{k} \sum_{q=s}^{k} \xi_{k-q:k}(s) E_{k-q} \tag{B.1.32}$$

$$\tag{B.1.33}$$

□

Remark B.1.6. First, note that Lemma B.1.5 applies here, since $C_0^{(t)} = E_0^{(t)}$, and thus

$$C_k^{(t)} = E_k^{(t)} + \sum_{s=1}^{k}\sum_{q=s}^{k} \xi_{k-q:k}^{(t)}(s) E_{k-q}^{(t)} \tag{B.1.34}$$

The idea of Lemma B.1.5 was to regroup all terms with the same number of F factors (where each F contains a Jacobian matrix J_k which contains the connectivity matrix V of the recurrent net). One could roughly perceive the term

$$\sum_{q=s}^{k} \xi_{k-q:k}^{(t)}(s) E_{k-q}^{(t)} \tag{B.1.35}$$

as being the term of degree s for $s = 1,2,\dots,k$ and $E_k^{(t)}$ the term of degree 0. This will allow us to consider the terms C roughly as a polynomial in V and we can look the asymptotic behaviour of each of the coefficients of this polynomial individually. This will then give us a very good understanding on how the distribution of the attention weights are affecting the magnitude of total gradient.

Proposition B.1.7. *For all $t \geq 1$, and all $k \geq 0$, we have that*

$$\frac{ds_{t+k}}{dh_t} = \sum_{s=0}^{k} \bar{\xi}_{o:k}^{(t)}(s) \tag{B.1.36}$$

where for all $s \geq 1$,

$$\bar{\xi}_{o:k}^{(t)}(s) = \sum_{0 \leq i_1 < \ldots < i_s < k} F_{k,i_s}^{(t)} \cdot F_{i_s,i_{s-1}}^{(t)} \cdot \ldots \cdot F_{i_2,i_1}^{(t)} \cdot E_{i_1}^{(t)} \tag{B.1.37}$$

and where $\bar{\xi}_{o:k}^{(t)}(0) = E_k^{(t)}$. With for all $k \geq 0$ we have

$$E_k^{(t)} = \frac{\partial s_{t+k}}{\partial h_t} \tag{B.1.38}$$

and for all $k \geq j$ we have

$$F_{k+1,j}^{(t)} = \frac{\partial s_{t+k+1}}{\partial h_{t+j+1}} \cdot J_{t+j} + 1_{j=k} \cdot \frac{\partial s_{t+k+1}}{\partial s_{t+k}} \tag{B.1.39}$$

PROOF. Let $t \geq 1$, and recall that we defined $C_k^{(t)} = \frac{ds_{t+k}}{dh_t}$, for all $k \geq 0$.

As already pointed out, we know that $C_0^{(t)} = E_0^{(t)}$ (thus the claim holds for $k = 0$).

Then by Lemma B.1.5, we know that for all $k \geq 1$ we have

$$\begin{aligned}
C_k^{(t)} &= E_k^{(t)} + \sum_{s=1}^{k}\sum_{q=s}^{k} \xi_{k-q:k}^{(t)}(s) E_{k-q}^{(t)} && \text{(B.1.40)}\\
&= \bar{\xi}_{o:k}^{(t)}(0) + \sum_{s=1}^{k}\sum_{q=s}^{k} \sum_{k-q=i_1<\ldots<i_{s+1}=k} F_{k,i_s}^{(t)} \cdot F_{i_s,i_{s-1}}^{(t)} \cdot \ldots \cdot F_{i_2,i_1}^{(t)} \cdot E_{i_1}^{(t)} && \text{(B.1.41)}\\
&= \bar{\xi}_{o:k}^{(t)}(0) + \sum_{s=1}^{k} \sum_{0\leq i_1<\ldots<i_s<k} F_{k,i_s}^{(t)} \cdot F_{i_s,i_{s-1}}^{(t)} \cdot \ldots \cdot F_{i_2,i_1}^{(t)} \cdot E_{i_1}^{(t)} && \text{(B.1.42)}\\
&= \bar{\xi}_{o:k}^{(t)}(0) + \sum_{s=1}^{k} \bar{\xi}_{o:k}^{(t)}(s) && \text{(B.1.43)}\\
&= \sum_{s=0}^{k} \bar{\xi}_{o:k}^{(t)}(s) && \text{(B.1.44)}
\end{aligned}$$

□

Remark B.1.8. In what follows the main emphasis will be to calculate the $F_{i,j}^{(t)}$ and $E_i^{(t)}$ terms explicitly, since they are the building blocks of the mentioned polynomials in B.1.6.

We will assume that

$$s_t = f(h_t, c_t) \tag{B.1.45}$$

with

$$c_t = \alpha_{1,t}h_1 + \alpha_{2,t}h_2 + \ldots + \alpha_{t,t}h_t \tag{B.1.46}$$

and

$$\alpha_{j,t} = \frac{\exp\{(e_{j,t})\}}{\sum_{i=1}^{t} \exp\{(e_{i,t})\}} \tag{B.1.47}$$

where

$$e_{i,t} = a(s_{t-1}, h_i) \tag{B.1.48}$$

Let us recall that for all $k \geq 0$ we have

$$E_k^{(t)} = \frac{\partial s_{t+k}}{\partial h_t} \tag{B.1.49}$$

and for all $k \geq j$ we have

$$F_{k+1,j}^{(t)} = \frac{\partial s_{t+k+1}}{\partial h_{t+j+1}} \cdot J_{t+j} + 1_{j=k} \cdot \frac{\partial s_{t+k+1}}{\partial s_{t+k}} \tag{B.1.50}$$

Lemma B.1.9. *With the assumption of Remark B.1.8, we have that for all $t \geq 2$*

$$\frac{\partial s_t}{\partial s_{t-1}} = \partial_2 f(h_t,c_t) \cdot \left(\sum_{i=1}^{t} \alpha_{i,t} Y_{i,t} \right) \tag{B.1.51}$$

where $\partial_2 f$ is the the partial derivative of f with respect to the second variable, and where we define

$$Y_{i,t} = h_i \cdot \left(\frac{\partial e_{i,t}}{\partial s_{t-1}} - \sum_{j=1}^{t} \alpha_{j,t} \cdot \frac{\partial e_{j,t}}{\partial s_{t-1}} \right) \tag{B.1.52}$$

PROOF.

$$\begin{aligned}
\frac{\partial s_t}{\partial s_{t-1}} &= \partial_2 f(h_t,c_t) \cdot \frac{\partial c_t}{\partial s_{t-1}} && \text{(B.1.53)}\\
&= \partial_2 f(h_t,c_t) \cdot \left[\sum_{i=1}^{t} h_i \cdot \left(\frac{\partial \alpha_{i,t}}{\partial s_{t-1}} \right) \right] && \text{(B.1.54)}\\
&= \partial_2 f(h_t,c_t) \cdot \left[\sum_{i=1}^{t} h_i \cdot \left(\sum_{j=1}^{t} \frac{\partial \alpha_{i,t}}{\partial e_{j,t}} \cdot \frac{\partial e_{j,t}}{\partial s_{t-1}} \right) \right] && \text{(B.1.55)}\\
&= \partial_2 f(h_t,c_t) \cdot \left[\sum_{i=1}^{t} h_i \cdot \left(\sum_{j=1}^{t} \alpha_{i,t}(1_{i=j} - \alpha_{j,t}) \cdot \frac{\partial e_{j,t}}{\partial s_{t-1}} \right) \right] && \text{(B.1.56)}\\
&= \partial_2 f(h_t,c_t) \cdot \left[\sum_{i=1}^{t} \alpha_{i,t} h_i \left(\frac{\partial e_{i,t}}{\partial s_{t-1}} - \sum_{j=1}^{t} \alpha_{j,t} \frac{\partial e_{j,t}}{\partial s_{t-1}} \right) \right] && \text{(B.1.57)}\\
&= \partial_2 f(h_t,c_t) \cdot \left(\sum_{i=1}^{t} \alpha_{i,t} Y_{i,t} \right) && \text{(B.1.58)}
\end{aligned}$$

□

Lemma B.1.10. *With the assumption of Remark B.1.8, we have that for all $k \geq j$:*

$$\frac{\partial s_k}{\partial h_j} = 1_{k=j} \cdot \partial_1 f(h_k,c_k) + \alpha_{j,k} \partial_2 f(h_k,c_k) \cdot (I + X_{j,k}) \tag{B.1.59}$$

where $\partial_1 f$ and $\partial_2 f$ are the partial derivatives of f with respect to the first and second variable, respectively, and where we define

$$X_{j,k} = \left(h_j - \sum_{i=1}^{k} h_i \alpha_{i,k} \right) \cdot \frac{\partial e_{j,k}}{\partial h_j} \tag{B.1.60}$$

PROOF.

$$\frac{\partial s_k}{\partial h_j} = 1_{k=j} \cdot \partial_1 f(h_k,c_k) \cdot \frac{\partial h_k}{\partial h_k} + \partial_2 f(h_k,c_k) \cdot \frac{\partial c_k}{\partial h_j} \tag{B.1.61}$$

$$= 1_{k=j} \cdot \partial_1 f(h_k,c_k) + \partial_2 f(h_k,c_k) \cdot \left[\alpha_{j,k} \cdot I + \sum_{i=1}^{k} h_i \cdot \frac{\partial \alpha_{i,k}}{\partial h_j}\right] \tag{B.1.62}$$

$$= 1_{k=j} \cdot \partial_1 f(h_k,c_k) + \partial_2 f(h_k,c_k) \cdot \left[\alpha_{j,k} \cdot I + \sum_{i=1}^{k} h_i \cdot \frac{\partial \alpha_{i,k}}{\partial e_{j,k}} \frac{\partial e_{j,k}}{\partial h_j}\right] \tag{B.1.63}$$

$$= 1_{k=j} \cdot \partial_1 f(h_k,c_k) + \partial_2 f(h_k,c_k) \cdot \left[\alpha_{j,k} \cdot I + \sum_{i=1}^{k} h_i \cdot \alpha_{i,k}(1_{i=j} - \alpha_{j,k}) \frac{\partial e_{j,k}}{\partial h_j}\right] \tag{B.1.64}$$

$$= 1_{k=j} \cdot \partial_1 f(h_k,c_k) + \partial_2 f(h_k,c_k) \cdot \left[\alpha_{j,k} \cdot I + \left(h_j \alpha_{j,k} - \alpha_{j,k} \sum_{i=1}^{k} h_i \cdot \alpha_{i,k}\right) \frac{\partial e_{j,k}}{\partial h_j}\right] \tag{B.1.65}$$

$$= 1_{k=j} \cdot \partial_1 f(h_k,c_k) + \alpha_{j,k} \partial_2 f(h_k,c_k) \cdot \left[I + \left(h_j - \sum_{i=1}^{k} h_i \cdot \alpha_{i,k}\right) \frac{\partial e_{j,k}}{\partial h_j}\right] \tag{B.1.66}$$

$$= 1_{k=j} \cdot \partial_1 f(h_k,c_k) + \alpha_{j,k} \partial_2 f(h_k,c_k) \cdot (I + X_{j,k}) \tag{B.1.67}$$

□

Corollary B.1.11. *With the assumption of Remark B.1.8, and the notations of lemma B.1.9 and B.1.10, we have for all $k' \geq 0$,*

$$E_{k'}^{(t)} = 1_{k'=0} \partial_1 f(h_t,c_t) + \alpha_{t,t+k'} \partial_2 f(h_{t+k'},c_{t+k'}) \cdot [I + X_{t,t+k'}] \tag{B.1.68}$$

and for all $k \geq j$,

$$F_{k+1,j}^{(t)} = \alpha_{t+j+1,t+k+1} \cdot \partial_2 f(h_{t+k+1},c_{t+k+1}) \cdot [I + X_{t+j+1,t+k+1}] \cdot J_{t+j} \tag{B.1.69}$$

$$+ 1_{k=j} \cdot \left(\partial_1 f(h_{t+k+1},c_{t+k+1}) J_{t+j} + \partial_2 f(h_{t+k+1},c_{t+k+1}) \cdot \left[\sum_{i=1}^{t+k+1} \alpha_{i,t+k+1} Y_{i,t+k+1}\right]\right) \tag{B.1.70}$$

PROOF. Applying lemma B.1.10, we get that for all $k \geq 0$,

$$E_{k'}^{(t)} = \frac{\partial s_{t+k'}}{\partial h_t} \tag{B.1.71}$$

$$= 1_{k'=0} \cdot \partial_1 f(h_t,c_t) + \alpha_{t,t+k'} \cdot \partial_2 f(h_{t+k'},c_{t+k'}) \cdot [I + X_{t,t+k'}] \tag{B.1.72}$$

$$\tag{B.1.73}$$

and then by applying lemma B.1.9 and B.1.10, we get that for all $k \geq j$,

$$F^{(t)}_{k+1,j} = \frac{\partial s_{t+k+1}}{\partial h_{t+j+1}} \cdot J_{t+j} + 1_{j=k} \cdot \frac{\partial s_{t+k+1}}{\partial s_{t+k}} \tag{B.1.74}$$

$$= [1_{k=j}\partial_1 f(h_{t+k+1}, c_{t+k+1}) \tag{B.1.75}$$

$$+ \alpha_{t+j+1,t+k+1}\partial_2 f(h_{t+k+1}, c_{t+k+1}) \cdot (I + X_{t+j+1,t+k+1})] \cdot J_{t+j} \tag{B.1.76}$$

$$+ 1_{k=j} \cdot \partial_2 f(h_{t+k+1}, c_{t+k+1}) \cdot \left(\sum_{i=1}^{t+k+1} \alpha_{i,t+k+1} Y_{i,t+k+1} \right) \tag{B.1.77}$$

$$= \alpha_{t+j+1,t+k+1} \cdot \partial_2 f(h_{t+k+1}, c_{t+k+1}) \cdot [I + X_{t+j+1,t+k+1}] \cdot J_{t+j} \tag{B.1.78}$$

$$+ 1_{k=j} \cdot \left(\partial_1 f(h_{t+j+1}, c_{t+k+1}) J_{t+j} + \partial_2 f(h_{t+k+1}, c_{t+k+1}) \cdot \left[\sum_{i=1}^{t+k+1} \alpha_{i,t+k+1} Y_{i,t+k+1} \right] \right) \tag{B.1.79}$$

□

Proposition B.1.12. *We can rewrite for all $k' \geq 0$ and all $k \geq j \geq 0$*

$$E^{(t)}_{k'} = \alpha_{t,t+k'} \cdot \tilde{D}^{(t)}_{k',0} + 1_{k'=0} \tilde{R}^{(t)}_0 \tag{B.1.80}$$

$$F^{(t)}_{k+1,j} = \alpha_{t+j+1,t+k+1} \cdot D^{(t)}_{k+1,j} + 1_{k=j} \cdot R^{(t)}_{k+1} \tag{B.1.81}$$

where

$$D^{(t)}_{k+1,j+1} = \partial_2 f(h_{t+k+1}, c_{t+k+1}) \cdot [I + X_{t+j+1,t+k+1}] \cdot J_{t+j} \tag{B.1.82}$$

$$R^{(t)}_{k+1} = \partial_1 f(h_{t+k+1}, c_{t+k+1}) \cdot J_{t+k} + \partial_2 f(h_{t+k+1}, c_{t+k+1}) \cdot [\sum_{i=1}^{t+k+1} \alpha_{i,t+k+1} Y_{i,t+k+1}] \tag{B.1.83}$$

$$\tilde{D}^{(t)}_{k'} = \partial_2 f(h_{t+k'}, c_{t+k'}) \cdot [I + X_{t,t+k'}] \tag{B.1.84}$$

$$\tilde{R}^{(t)}_0 = \partial_1 f(h_t, c_t) \tag{B.1.85}$$

while $X_{i,i'}$ and $Y_{i,i'}$ are defined as in lemma B.1.9 and B.1.10.

PROOF. Follows straight from Corollary B.1.11. □

Remark B.1.13. If we are further assuming that

$$s_t = f(h_t, c_t) = h_t + c_t \tag{B.1.86}$$

then for all $k \geq 0$, we have

$$E^{(t)}_k = 1_{k=0} \cdot I + \alpha_{t,t+k} \cdot [I + X_{t,t+k}] \tag{B.1.87}$$

and for all $k \geq j$, we have

$$F^{(t)}_{k+1,j} = \alpha_{t+j+1,t+k+1} \cdot [I + X_{t+j+1,t+k+1}] \cdot J_{t+j} + 1_{k=j} \cdot \left(J_{t+j} + \left[\sum_{i=1}^{t+k+1} \alpha_{i,t+k+1} Y_{i,t+k+1} \right] \right) \tag{B.1.88}$$

PROOF. This follows directly form corollary B.1.11 and the observation that

$$\partial_1 f(h_t,c_t) = \partial_2 f(h_t,c_t) = I \tag{B.1.89}$$

□

Remark B.1.14. If we are further assuming that

$$e_{j,t} = a(s_{t-1},h_j) = v_a^T \cdot \tanh\left(W_a s_{t-1} + U_a h_j\right) \tag{B.1.90}$$

as done in Bahdanau et al. (2014), we get that

$$\frac{\partial e_{j,t}}{\partial h_j} = v_a^T \cdot \mathrm{diag}[1 - \tanh^2\left(W_a s_{t-1} + U_a h_j\right)] \cdot U_a \tag{B.1.91}$$

and

$$\frac{\partial e_{j,t}}{\partial s_{t-1}} = v_a^T \cdot \mathrm{diag}[1 - \tanh^2\left(W_a s_{t-1} + U_a h_j\right)] \cdot W_a \tag{B.1.92}$$

which we can plug into the definitions of $X_{j,k}$ and $Y_{j,k}$ to get explicit expressions for matrices $E^{(t)}_{k'}$ and $F^{(t)}_{k+1,j}$.

Lemma B.1.15. *If, with the assumptions Remark B.1.8, we assume that for all $i,t \geq 1$, we have $e_{i,t} = a(s_{t-1},h_i,\theta)$ depending on some parameter $\theta \in \mathbb{R}^{N\times M}$, then we have*

$$\frac{dL}{d\theta} = \sum_{j,t} \alpha_{j,t} \cdot \frac{dL}{ds_t} \cdot \partial_2 f(h_t,c_t) \cdot h_j \cdot \left[\sum_i (1_{i=j} - \alpha_{i,t}) \cdot \frac{\partial e_{i,t}}{\partial \theta} \right] \tag{B.1.93}$$

PROOF. If we denote $\theta^{(i,t)}$ to be the parameter for $e_{i,t}$, then we have

$$\frac{dL}{d\theta} = \sum_{i,t} \frac{dL}{d\theta^{(i,t)}} \tag{B.1.94}$$

$$= \sum_{i,j,t} \frac{dL}{d\alpha_{j,t}} \cdot \frac{\partial \alpha_{j,t}}{\partial e_{i,t}} \cdot \frac{\partial e_{i,t}}{\partial \theta^{(i,t)}} \tag{B.1.95}$$

$$= \sum_{i,j,t} \alpha_{j,t}(1_{i=j} - \alpha_{i,t}) \cdot \frac{dL}{d\alpha_{j,t}} \cdot \frac{\partial e_{i,t}}{\partial \theta} \tag{B.1.96}$$

where

$$\frac{dL}{d\alpha_{j,t}} = \frac{dL}{ds_t} \cdot \frac{\partial s_t}{\partial c_t} \cdot \frac{\partial c_t}{\partial \alpha_{j,t}} = \frac{dL}{ds_t} \cdot \partial_2 f(h_t,c_t) \cdot h_j \tag{B.1.97}$$

Hence

$$\frac{dL}{d\theta} = \sum_{i,j,t} \alpha_{j,t}(1_{i=j} - \alpha_{i,t}) \cdot \frac{dL}{ds_t} \cdot \partial_2 f(h_t,c_t) \cdot h_j \cdot \frac{\partial e_{i,t}}{\partial \theta} \tag{B.1.98}$$

$$= \sum_{j,t} \alpha_{j,t} \cdot \frac{dL}{ds_t} \cdot \partial_2 f(h_t,c_t) \cdot h_j \cdot \left[\sum_i (1_{i=j} - \alpha_{i,t}) \cdot \frac{\partial e_{i,t}}{\partial \theta}\right] \tag{B.1.99}$$

□

Lemma B.1.16. *Let us recall that for all $t \geq 0$, we have*

$$h_{t+1} = \phi(\underbrace{Vs_t + Ux_{t+1} + b}_{=a_t}) \tag{B.1.100}$$

where ϕ is a non-linear activation function, $V \in \mathbb{R}^{n\times n}$, $U \in \mathbb{R}^{n\times m}$ and $b \in \mathbb{R}^n$. Then we have that

$$\left[\frac{dL}{dV}, \frac{dL}{dU}, \frac{dL}{db}\right] = \sum_{t=1}^{T} [s_{t-1}, x_t, 1] \cdot \frac{dL}{dh_t} \cdot diag(\phi'(a_t)) \tag{B.1.101}$$

PROOF. Let us denote $V^{(t)}, U^{(t)}, b^{(t)}$ the matrices V,U,b of a_{t-1} respectively, then

$$\left[\frac{dL}{dV}, \frac{dL}{dU}, \frac{dL}{db}\right] = \sum_t \left[\frac{dL}{dV^{(t)}}, \frac{dL}{dU^{(t)}}, \frac{dL}{db^{(t)}}\right] \tag{B.1.102}$$

$$= \sum_t [s_{t-1}, x_t, 1] \cdot \frac{dL}{da_{t-1}} \tag{B.1.103}$$

$$= \sum_t [s_{t-1}, x_t, 1] \cdot \frac{dL}{dh_t} \cdot \frac{dh_t}{da_{t-1}} \tag{B.1.104}$$

$$= \sum_t [s_{t-1}, x_t, 1] \cdot \frac{dL}{dh_t} \cdot \text{diag}(\phi'(a_{t-1})) \tag{B.1.105}$$

□

Remark B.1.17. Combining the fact that $\frac{dL}{dh_t} = \frac{dL}{ds_T}\frac{ds_T}{dh_t}$, the results from propositions B.1.7 and B.1.12, with lemma B.1.16, we see that attention weights $\alpha_{i,t}$ which are very close to 0, do not contribute to the gradient and the learning of V,U and b.

Similarly, it follows directly from lemma B.1.15, that attention weights $\alpha_{i,t}$ which are very close to 0, do not contribute to the gradient and the learning of any parameters θ of the alignment function $e_{i,t} = a(s_{t-1}, h_i, \theta)$. In case we have an alignment function as in remark B.1.14, these parameters are W_a, U_a and v_a.

If we have the case where one state h_i is such that all attention weights $\alpha_{i,t} \approx 0$ for all $t \geq i$, then we can see that h_i does not contribute to the gradient and learning to any parameters be it parameters from the recurrence or the alignment function.

In practice we have observed that in the majority of tasks, most states h_i fall in either of two categories:

- $\alpha_{i,t}$ is sufficiently bounded away from 0 for most $t \geq i$, and thus contributes to learning. This is what we call a "relevant state".
- $\alpha_{i,t} \approx 0$ for almost all $t \geq i$, and thus doesn't contribute much to learning, and the gradient can be approximated by assuming $\alpha_{i,t} = 0$ for all $t \geq i$. This is what he call a "non-relevant state".

This observation is what lead us to the intuition that we can approximate the gradient, by decomposing it via proposition B.1.7, into gradient paths involving only skip connections between "relevant states".

B.1.3. Uniform attention case

Remark B.1.18. In this subsection, we are going to assume:

- no non-linearity in the hidden-to-hidden connection: $J_t = V$ for all t.
- all assumptions from Remark B.1.8.
- uniform attention: $\alpha_{i,t} = 1/t$ for all $t \geq 1$.

B.1.3.1. Overview.

Remark B.1.19. Recalling corollary B.1.11, together the main proposition B.1.7 form last section, we can hope to simplify these expressions using the new assumptions from the previous remark B.1.18. Recalling expression from lemma B.1.9 and B.1.10:

$$X_{j,t} = \left(h_j - \sum_{i=1}^{t} h_i \alpha_{i,t}\right) \cdot \frac{\partial e_{j,t}}{\partial h_j} \tag{B.1.106}$$

$$= \left(h_j - \frac{1}{t}\sum_{i=1}^{t} h_i\right) \cdot \frac{\partial e_{j,t}}{\partial h_j} \tag{B.1.107}$$

Hence, for our calculations we are going to assume that $\left(h_j - \frac{1}{t}\sum_{i=1}^{t} h_i\right) \approx 0$, and thus $X_{j,t} \approx 0$ for all $1 \leq j \leq t$. Similarly,

$$\sum_{i=1}^{t} \alpha_{i,t} Y_{i,t} = \sum_{i=1}^{t} \alpha_{i,t} h_i \cdot \left(\frac{\partial e_{i,t}}{\partial s_{t-1}} - \sum_{j=1}^{t} \alpha_{j,t} \cdot \frac{\partial e_{j,t}}{\partial s_{t-1}} \right) \tag{B.1.108}$$

$$= \frac{1}{t}\sum_{i=1}^{t} h_i \cdot \left(\frac{\partial e_{i,t}}{\partial s_{t-1}} - \sum_{j=1}^{t} \frac{1}{t} \cdot \frac{\partial e_{j,t}}{\partial s_{t-1}} \right) \tag{B.1.109}$$

$$= \frac{1}{t}\sum_{i=1}^{t} h_i \cdot \frac{\partial e_{i,t}}{\partial s_{t-1}} - \frac{1}{t}\sum_{j=1}^{t} \left(\frac{1}{t}\sum_{i=1}^{t} h_i \right) \cdot \frac{\partial e_{j,t}}{\partial s_{t-1}} \tag{B.1.110}$$

$$= \frac{1}{t}\sum_{i=1}^{t} h_i \cdot \frac{\partial e_{i,t}}{\partial s_{t-1}} - \frac{1}{t}\sum_{i=1}^{t} \left(\frac{1}{t}\sum_{j=1}^{t} h_j \right) \cdot \frac{\partial e_{i,t}}{\partial s_{t-1}} \tag{B.1.111}$$

$$= \frac{1}{t}\sum_{i=1}^{t} \left(h_i - \frac{1}{t}\sum_{j=1}^{t} h_j \right) \cdot \frac{\partial e_{i,t}}{\partial s_{t-1}} \tag{B.1.112}$$

$$\approx 0 \tag{B.1.113}$$

Recalling the expression from corollary B.1.11 and that $f(h_t,c_t) = h_t + c_t$ by remark B.1.8, and that $J_t = V$ for all t, this will give for all $k' \geq 0$

$$E_{k'}^{(t)} = \left(\frac{1}{t+k'} + 1_{k'=0} \right) \cdot \mathbf{I} \tag{B.1.114}$$

and for all $k \geq j$, we get

$$F_{k+1,j}^{(t)} = \left(\frac{1}{t+k+1} + 1_{k=j} \right) \cdot V \tag{B.1.115}$$

Hence by recalling proposition B.1.7, the main expression of interest becomes

$$\frac{ds_{t+k}}{dh_t} = \sum_{s=0}^{k} \bar{\xi}_{0:k}^{(t)}(s) = \sum_{s=0}^{k} V^s \cdot \chi_{0:k}^{(t)}(s) \tag{B.1.116}$$

where

$$\chi_{0:k}^{(t)}(s) \overset{\text{def}}{=} \sum_{0 \leq i_1 < \ldots < i_s < k} \left(\frac{1}{t+k} + 1_{k-i_s=1} \right) \cdot \left(\frac{1}{t+i_s} + 1_{i_s - i_{s-1}=1} \right) \cdot \ldots \tag{B.1.117}$$

$$\ldots \cdot \left(\frac{1}{t+i_2} + 1_{i_2-i_1=1} \right) \cdot \left(\frac{1}{t+i_1} + 1_{i_1=0} \right) \tag{B.1.118}$$

Remark B.1.20. The goal is thus to have a good estimation of the terms

$$\chi_{0:k}^{(t)}(s) \tag{B.1.119}$$

in order to then find an asymptotic estimation for

$$\frac{ds_{t+k}}{dh_t} = \sum_{s=0}^{k} V^s \cdot \chi_{0:k}^{(t)}(s) \tag{B.1.120}$$

as $k \to \infty$. In order to do so, we will adopt the following strategy:

Step 1. Estimate the expression

$$\omega_{l:k}^{(t)}(s) \overset{\text{def}}{=} \sum_{l \le i_1 < \ldots < i_s < k} \frac{1}{t+i_s} \cdot \frac{1}{t+i_{s-1}} \cdot \ldots \cdot \frac{1}{t+i_2} \cdot \frac{1}{t+i_1} \tag{B.1.121}$$

for all $s \ge 1$. This will be done in §B.1.3.2.

Step2. Estimate the expression

$$\theta_{l:k}^{(t)}(s) \overset{\text{def}}{=} \sum_{l \le i_1 < \ldots < i_s < k} \left(\frac{1}{t+i_s} + 1_{i_s - i_{s-1} = 1}\right) \cdot \left(\frac{1}{t+i_{s-1}} + 1_{i_{s-1} - i_{s-2} = 1}\right) \cdot \ldots \tag{B.1.122}$$

$$\ldots \cdot \left(\frac{1}{t+i_2} + 1_{i_2 - i_1 = 1}\right) \cdot \left(\frac{1}{t+i_1} + 1_{i_1 = 0}\right) \tag{B.1.123}$$

for all $s \ge 1$, because as we will see the expression $\theta_{l:k}^{(t)}(s)$ can be decomposed into $\omega_{l':k'}^{(t)}(s')$ expressions for $s \ge s' \ge 1$. This will be done in §B.1.3.3.

Step 3. The final step will consist in putting the results from the two previous sub-subsections together, and getting a final asymptotic estimate for $\frac{ds_{t+k}}{dh_t}$ as $k \to \infty$, by noting that

$$\chi_{0:k}^{(t)}(s) = \frac{1}{t+k} \cdot \theta_{0:k}^{(t)}(s) + \frac{1}{t+k-1} \cdot \theta_{0:k-1}^{(t)}(s-1) + \ldots \tag{B.1.124}$$

$$\ldots + \frac{1}{t+k-s+1} \cdot \theta_{0:k-s+1}^{(t)}(1) + \frac{1}{t+k-s} + 1_{k=s} \tag{B.1.125}$$

This will be treated in §B.1.3.4.

B.1.3.2. Estimating ω.

Remark B.1.21. In this sub-subsection we are going to estimate $\omega_{0:k}^{(t)}(s)$, which is a sum of products of s distinct factors. The idea will be to start from the expression

$$\left(\frac{1}{t} + \frac{1}{t+1} + \ldots + \frac{1}{t+k-1}\right)^s \tag{B.1.126}$$

and substract all products containing at least two identical factors, followed by a division by $s!$.

This approach will be similar in spirit to the inclusion-exclusion principle, with the only difference that the desired term will not computed directly, but instead one first establishes a recursive formula using $\omega_{0:k}^{(t)}(s')$ with $s' \leq s$.

Solving this recursive formula will enable us to express $\omega_{0:k}^{(t)}(s)$ only in terms of $(\frac{1}{t} + \frac{1}{t+1} + \ldots + \frac{1}{t+k-1})$. In fact, $\omega_{0:k}^{(t)}(s)$ will be a polynomial of degree s in $(\frac{1}{t} + \frac{1}{t+1} + \ldots + \frac{1}{t+k-1})$.

We adopt this approach, because we have a very good estimate for

$$\frac{1}{t} + \frac{1}{t+1} + \ldots + \frac{1}{t+k-1} \tag{B.1.127}$$

Namely, we know that for all n, we have

$$1 + \frac{1}{2} + \ldots + \frac{1}{n-1} + \frac{1}{n} = \ln n + \gamma + \varepsilon_n \leq \ln n + 1 \tag{B.1.128}$$

where $\gamma > \frac{1}{2}$ is the Euler-Mascheroni constant and ε_n behaves asymptotically as $\frac{1}{2n}$. In other words,

$$\frac{1}{t} + \frac{1}{t+1} + \ldots + \frac{1}{t+k-1} = \ln\left(\frac{t+k-1}{t-1}\right) + \varepsilon_{t+k-1} - \varepsilon_{t-1} \tag{B.1.129}$$

$$= \ln\left[\frac{t+k-1}{t-1} \cdot \exp\{(\varepsilon_{t+k-1} - \varepsilon_{t-1})\}\right] \tag{B.1.130}$$

$$= \ln \beta_{t-1,t+k-1} \tag{B.1.131}$$

where $\beta_{l,l'} \overset{\text{def}}{=} \frac{l'}{l} \cdot \exp\{(\varepsilon_{l'} - \varepsilon_l)\}$. In order to reinforce the intuition here, let us imagine that $T = t + k$, then

$$\ln \beta_{t-1,t+k-1} \sim \ln T \tag{B.1.132}$$

as $T \to \infty$. Hence we should expect $\omega_{0:k}^{(t)}(s)$ to behave asymptotically as a polynomial of degree s in $\ln T$.

Let us emphasize that we would like to express $\omega_{0:k}^{(t)}(s)$ with as much precision as possible (i.e. not omitting the monomials in $\ln T$ of degree less than s), since we would like to later on use this estimate in subsequent steps when summing multiple $\omega_{0:k}^{(t)}(s)$ terms over s.

In order to further ease notation, we will simply write $\omega(s)$ for $\omega_{0:k}^{(t)}(s)$, whenever there is no ambiguity.

Finally, for this sub-subsection only we will use the following notation

$$S_l \overset{\text{def}}{=} \frac{1}{t^l} + \frac{1}{(t+1)^l} + \ldots + \frac{1}{(t+k-1)^l} \tag{B.1.133}$$

for all $l \geq 1$, and keeping in mind that S_l converges as $k \to \infty$, for all $l \geq 2$.

Remark B.1.22. Let us now build a first intuition on how to apply an inclusion-exclusion-like principle in order to calculate $\omega(s)$ for small s.

For $\underline{s = 1}$:

$$\omega(1) = S_1 \tag{B.1.134}$$

For $\underline{s = 2}$:

$$\omega(2) = \frac{1}{2!}\left(S_1^2 - S_2\right) \tag{B.1.135}$$

Here we expand S_1^2, then substract the sum of products of doubles S_2, followed by a division of $2! = 2$ to divide out the number of permutations.

For $\underline{s = 3}$: first we need to substract the sum of products of triples S_3, and then the sum products where exactly two factors are identical $S_2 \cdot \omega(1) - S_3$. The latter appears $\binom{3}{2,1} = \frac{3!}{2!1!} = 3$ times in the expansion of S_1^3. Similarly, we need to divide out the number of permutations $3!$. Hence

$$\omega(3) = \frac{1}{3!}\left[S_1^3 - S_3 - 3 \cdot (S_2 \cdot \omega(1) - S_3)\right] = \frac{S_1^3}{3!} - \frac{1}{2}S_2 \cdot \omega(1) + \frac{1}{3}S_3 \tag{B.1.136}$$

Let us form now on denote (3) for the sum of products of triples, and $(2{,}1)$ the sum of products where exactly two factors are the same.

More generally we would denote

$$(j_1,j_2,\ldots,j_k) \tag{B.1.137}$$

with $j_1 \geq j_2 \geq \ldots \geq j_k \geq 1$, to denote the sum of products where one factor appears exactly j_1 times, another factor (distinct from the previous one!) appears exactly j_2 times, and another factor (distinct from the previous two!) appears exactly j_3 times, etc. This leaves us with exactly k distinct factors each having multiplicity $j_1,j_2,\ldots,j_k$ respectively. This sum appears with

$$\binom{s}{j_1,j_2,\ldots,j_k} = \frac{s!}{j_1! \cdot j_2! \cdot \ldots \cdot j_k!} \tag{B.1.138}$$

repetitions in the expansion of S_1^s, where $s = j_1 + j_2 + \ldots + j_k$.

For $\underline{s = 4}$: when expanding S_1^4, we need to take into account

- $(4) = S_4$ with $\binom{4}{4} = \frac{4!}{4!} = 1$ repetition.

- (3,1) $= S_3 \cdot \omega(1) - S_4$ with $\binom{4}{3,1} = \frac{4!}{3!\cdot 1!} = 4$ repetitions.
- (2,2) $= S_2^2 - S_4$ with $\binom{4}{2,2} = \frac{4!}{2!\cdot 2!} = 6$ repetitions.
- (2,1,1) $= S_2 \cdot \omega(2) - (3,1) = S_2 \cdot \omega(2) - S_3 \cdot \omega(1) + S_4$ with $\binom{4}{2,1,1} = \frac{4!}{2!\cdot 1!\cdot 1!} = 12$ repetitions.

Hence we get

$$\omega(4) = \frac{1}{4!}[S_1^4 - S_4 - 4 \cdot (S_3 \cdot \omega(1) - S_4) - 6 \cdot (S_2^2 - S_4) \tag{B.1.139}$$

$$- 12 \cdot (S_2 \cdot \omega(2) - S_3 \cdot \omega(1) + S_4)] \tag{B.1.140}$$

$$= \frac{1}{4!}[S_1^4 - 4 \cdot S_3 \cdot \omega(1) + 4 \cdot S_4 - 6 \cdot S_2^2 + 6 \cdot S_4 - 12 \cdot S_2\omega(2) \tag{B.1.141}$$

$$+ 12 \cdot S_3 \cdot \omega(1) - 12 \cdot S_4 - S_4] \tag{B.1.142}$$

$$= \frac{1}{4!}\left[S_1^4 - 12 \cdot S_2 \cdot \omega(2) + 8 \cdot S_3 \cdot \omega(1) - 3 \cdot (S_4 + S_2^2)\right] \tag{B.1.143}$$

$$= \frac{S_1^4}{4!} - \frac{S_2}{2}\omega(2) + \frac{S_3}{3}\omega(1) - \frac{(S_4 + 2 \cdot S_2^2)}{8} \tag{B.1.144}$$

Notice how, as we progress with higher values of s, we build a recursive formula in $\omega(s')$ with $s' \leq s$.

Intuition. Note that the coefficient of $\omega(2)$ for $s = 4$, is the same as the coefficient for $\omega(1)$ for $s = 3$, and is the same as the 'constant term' for $s = 2$. Similarly, the coefficient of $\omega(1)$ for $s = 4$ is the same as the 'constant term' for $s = 3$. (By convention here, we don't consider the terms $\frac{S_1^s}{s!}$ to not be part of the 'constant term'.)

Hence, in the recursive formula for $\omega(s)$, we would expect the coefficient of $\omega(s')$ with $s' < s$ to be equal to the 'constant term' in the formula for $\omega(s - s')$.

Notation. For all $s > l \geq 0$, let us denote $\delta_{s,l}$ to be the coefficient of the term $\omega(l)$ in the recursive formula for $\omega(s)$. By convention, we denote $\delta_{s,0}$ for the 'constant term' in the recursive formula for $\omega(s)$. Hence for all $s \geq 1$, we have

$$\omega(s) = \frac{S_1^s}{s!} + \delta_{s,s-1} \cdot \omega(s-1) + \delta_{s,s-2} \cdot \omega(s-2) + \ldots + \delta_{s,1} \cdot \omega(1) + \delta_{s,0} \tag{B.1.145}$$

Hypothesis. The hypothesis will thus rewrite as

$$\delta_{s,l} = \delta_{s-l,0} \tag{B.1.146}$$

for all $s > l \geq 0$, which will prove by induction on s in the next lemma.

Lemma B.1.23. *Let $s \geq 1$. Then*

$$\omega(s) = \frac{S_1^s}{s!} + \delta_{1,0} \cdot \omega(s-1) + \delta_{2,0} \cdot \omega(s-2) + \ldots + \delta_{s-1,0} \cdot \omega(1) + \delta_{s,0} \tag{B.1.147}$$

PROOF. Let us prove by induction on s that for all $s > l \geq 0$, we have

$$\delta_{s,l} = \delta_{s-l,0} \tag{B.1.148}$$

. We already verified the cases $s = 1,2,3,4$ in the previous remark. Thus let us suppose the induction hypothesis is true for s, and consider the mapping

$$\Upsilon : (j_1,j_2,\ldots,j_k) \mapsto (j_1,j_2,\ldots,j_k,1) \tag{B.1.149}$$

where $j_1 \geq j_2 \geq \ldots \geq j_k \geq 1$ and $s = j_1 + j_2 + \ldots + j_k$, mapping a partition of s onto a partition of $s+1$.

If we suppose that $(j_1,j_2,\ldots,j_k)$ consists of exactly r 1's, then we can write

$$(j_1,j_2,\ldots,j_k) = c_r \cdot \omega(r) + c_{r-1} \cdot \omega(r-1) + \ldots + c_1 \cdot \omega(1) + c_0 \tag{B.1.150}$$

for some coefficients $c_r,c_{r-1},\ldots,c_1,c_0$, and with

$$\binom{s}{j_1,j_2,\ldots,j_k} = \frac{s!}{j_1! \cdot j_2 \cdot \ldots \cdot j_k!} \tag{B.1.151}$$

repetitions in the expansion of S_1^s.

The contribution of $(j_1,j_2,\ldots,j_k)$ to the coefficient $\delta_{s,r'}$ of $\omega(r')$ with $r' \leq r < s$, in the final recursive formula of $\omega(s)$ will be

$$\frac{c_{r'}}{j_1! \cdot j_2! \cdot \ldots \cdot j_k!} \tag{B.1.152}$$

(keeping in mind that we are dividing by $s!$ after having done all the substractions from S_1^s).

Meanwhile,

$$(j_1,j_2,\ldots,j_k,1) = c_r \cdot \omega(r+1) + c_{r-1} \cdot \omega(r) + \ldots + c_1 \cdot \omega(2) + c_0 \cdot \omega(1) + \tilde{c_0} \tag{B.1.153}$$

for some coefficient $\tilde{c_0}$, with

$$\binom{s+1}{j_1,j_2,\ldots,j_k,1} = \frac{(s+1)!}{j_1! \cdot j_2 \cdot \ldots \cdot j_k!} \tag{B.1.154}$$

repetitions in the expansion of S_1^{s+1}.

The contribution of $(j_1,j_2,\ldots,j_k,1)$ to the coefficient $\delta_{s+1,r'+1}$ of $\omega(r'+1)$ with $r' \leq r < s$, in the final recursive formula of $\omega(s+1)$ will be

$$\frac{c_{r'}}{j_1! \cdot j_2! \cdot \ldots \cdot j_k!} \tag{B.1.155}$$

(keeping in mind that we are dividing by $(s+1)!$ after having done all the substractions from S_1^{s+1}).

Conversely, the coefficient $\delta_{s+1,r'+1}$ only receives contributions from partitions of $(s+1)$ having at least $(r'+1)$ 1's, which correspond exactly to the contributions from the partitions of s having at least r' 1's. Hence

$$\delta_{s+1,r'+1} = \delta_{s,r'} \tag{B.1.156}$$

Then by the induction hypothesis, we have $\delta_{s,r'} = \delta_{s-r',0}$. In other words

$$\delta_{s+1,r'+1} = \delta_{s-r',0} \tag{B.1.157}$$

which completes the proof by induction. □

Remark B.1.24. Note that all the coefficients $\delta_{s,l}$ consist of linear combination of products with factors equal to S_j with $j \geq 2$, which are known to converge as $T \to \infty$. Thus those can be considered constants when doing an asymptotic analysis in the subsequent sub-subsections. Also note that $\delta_{s,s-1} = \delta_{1,0} = 0$.

Proposition B.1.25. *For all $s \geq 1$, we have*

$$\omega(s) = \sum_{r=0}^{s} \psi_{s-r} \frac{S_1^r}{r!} \tag{B.1.158}$$

where for $l \geq 2$

$$\psi_l \overset{\text{def}}{=} \sum_{k=1}^{l-1} \sum_{(j_1,j_2,\ldots,j_k) \in \Psi_{l,k}} \delta_{j_1,0} \cdot \ldots \cdot \delta_{j_k,0} \tag{B.1.159}$$

with

$$\Psi_{l,k} \overset{\text{def}}{=} \{(j_1,j_2,\ldots,j_k) \text{ with } j_1 \geq \ldots \geq j_k > 1 \text{ and } j_1 + \ldots + j_k = l\} \tag{B.1.160}$$

and where we define $\psi_0 = 1$ and $\psi_1 = 0$.

PROOF. For $l \geq 2$, we have

$$\psi_l = \sum_{k=1}^{l-1} \sum_{(j_1,j_2,\ldots,j_k)\in\Psi_{l,k}} \delta_{j_1,0} \cdot \ldots \cdot \delta_{j_k,0} \tag{B.1.161}$$

$$= \delta_{l,0} + \sum_{k=1}^{l-1} \left(\sum_{j=2}^{l-2} \sum_{(j_2,\ldots,j_k)\in\Psi_{l-j,k-1}} \delta_{j,0} \cdot \delta_{j_2,0} \cdot \ldots \cdot \delta_{j_k,0} \right) \tag{B.1.162}$$

$$= \delta_{l,0} + \sum_{j=2}^{l-2} \left(\sum_{k=1}^{l-j} \sum_{(j_2,\ldots,j_k)\in\Psi_{l-j,k-1}} \delta_{j,0} \cdot \delta_{j_2,0} \cdot \ldots \cdot \delta_{j_k,0} \right) \tag{B.1.163}$$

$$= \delta_{l,0} + \sum_{j=2}^{l-2} \delta_{j,0} \cdot \left(\sum_{k=1}^{l-j} \sum_{(j_2,\ldots,j_k)\in\Psi_{l-j,k-1}} \delta_{j_2,0} \cdot \ldots \cdot \delta_{j_k,0} \right) \tag{B.1.164}$$

$$= \delta_{l,0} + \sum_{j=2}^{l-2} \delta_{j,0} \cdot \psi_{l-j} \tag{B.1.165}$$

$$= \sum_{j=1}^{l} \delta_{j,0} \cdot \psi_{l-j} \tag{B.1.166}$$

(B.1.167)

In other words, we have shown that for all $l \geq 2$,

$$\psi_l = \sum_{j=0}^{l-1} \delta_{l-j} \psi_j \tag{B.1.168}$$

Let us now prove the proposition by induction on s.

The case $\underline{s = 1}$ is trivial by the definition of ψ_0 and ψ_1.

Let us now assume the formula is true for s, and let us prove it for $s+1$. By the previous lemma B.1.25, we know that

$$\omega(s+1) = \frac{S_1^{s+1}}{(s+1)!} + \sum_{l=1}^{s} \delta_{s+1-l,0} \cdot \omega(l) + \delta_{s+1,0} \tag{B.1.169}$$

$$= \frac{S_1^{s+1}}{(s+1)!} + \sum_{l=1}^{s} \delta_{s+1-l,0} \cdot \left(\sum_{r=0}^{l} \psi_{l-r} \cdot \frac{S_1^r}{r!} \right) + \delta_{s+1,0} \tag{B.1.170}$$

$$= \frac{S_1^{s+1}}{(s+1)!} + \sum_{l=1}^{s} \sum_{r=0}^{l} \delta_{s+1-l,0} \cdot \psi_{l-r} \cdot \frac{S_1^r}{r!} + \delta_{s+1,0} \tag{B.1.171}$$

$$= \frac{S_1^{s+1}}{(s+1)!} + \sum_{l=0}^{s} \sum_{r=0}^{l} \delta_{s+1-l,0} \cdot \psi_{l-r} \cdot \frac{S_1^r}{r!} \tag{B.1.172}$$

$$= \frac{S_1^{s+1}}{(s+1)!} + \sum_{r=0}^{s} \sum_{l=r}^{s} \delta_{s+1-l,0} \cdot \psi_{l-r} \cdot \frac{S_1^r}{r!} \tag{B.1.173}$$

$$= \frac{S_1^{s+1}}{(s+1)!} + \sum_{r=0}^{s} \sum_{l'=0}^{s-r} \delta_{s+1-r-l',0} \cdot \psi_{l'} \cdot \frac{S_1^r}{r!} \tag{B.1.174}$$

$$= \frac{S_1^{s+1}}{(s+1)!} + \sum_{r=0}^{s} \psi_{s+1-r} \cdot \frac{S_1^r}{r!} \tag{B.1.175}$$

$$= \sum_{r=0}^{s+1} \psi_{s+1-r} \cdot \frac{S_1^r}{r!} \tag{B.1.176}$$

completing the proof by induction. □

Remark B.1.26. Hence we have shown that for all $s \geq 1$

$$\omega(s) = \sum_{r=0}^{s} \psi_{s-r} \frac{S_1^r}{r!} = \frac{S_1^s}{s!} + \sum_{r=0}^{s-2} \psi_{s-r} \frac{S_1^r}{r!} \tag{B.1.177}$$

or in other words

$$\omega(s) = \frac{(\ln \beta_{t-1,t+k-1})^s}{s!} + \sum_{r=0}^{s-2} \psi_{s-r} \frac{(\ln \beta_{t-1,t+k-1})^r}{r!} \sim \frac{(\ln T)^s}{s!} + \sum_{r=0}^{s-2} \psi_{s-r} \frac{(\ln T)^r}{r!} \tag{B.1.178}$$

as $t+k = T \to \infty$, which is roughly the polynomial in $\ln T$ of degree s we were anticipating.

B.1.3.3. Estimating θ.

Remark B.1.27. Let us now recall the definition for all $s \geq 1$,

$$\theta_{l:k}^{(t)}(s) \overset{\text{def}}{=} \sum_{l \leq i_1 < \ldots < i_s < k} \left(\frac{1}{t+i_s} + 1_{i_s - i_{s-1} = 1} \right) \cdot \left(\frac{1}{t+i_{s-1}} + 1_{i_{s-1} - i_{s-2} = 1} \right) \cdot \ldots \tag{B.1.179}$$

$$\ldots \cdot \left(\frac{1}{t+i_2} + 1_{i_2 - i_1 = 1} \right) \cdot \left(\frac{1}{t+i_1} + 1_{i_1 = 0} \right) \tag{B.1.180}$$

which we would like to estimate using $\omega_{l:k}^{(t)}(s)$.

In order to build a first intuition, let us look at how it plays out for small values for s. Notation. In this subsection we omit the superscript (t) notation because there is no ambiguity. We will also occasionally do the abuse of notation and assume $\omega_{l:k}(0) = 1$ for all $l < k$.

For $\underline{s = 1}$, we get

$$\theta_{0:k}(1) = 1 + \omega_{0:k}(1) \tag{B.1.181}$$

For $\underline{s = 2}$, we get

$$\theta_{0:k}(2) = 1 + \omega_{1:k}(1) + \omega_{0:k-1}(1) + \omega_{0:k}(2) \tag{B.1.182}$$

In what follows, we will use the following recursive formula quite frequently

$$\theta_{0:k}(s+1) = \theta_{0:k-1}(s) + \sum_{j=s}^{k-1} \frac{1}{t+j} \theta_{0:j}(s) \tag{B.1.183}$$

Hence for $\underline{s = 3}$, we get

$$\theta_{0:k}(3) = 1 + \omega_{1:k-1}(1) + \omega_{0:k-2}(1) + \omega_{2:k}(1) + \omega_{0:k-1}(2) \tag{B.1.184}$$

$$+ \omega_{1:k}(2) + \sum_{j=2}^{k-1} \frac{\omega_{0:j-1}(1)}{t+j} + \omega_{0:k}(3) \tag{B.1.185}$$

Now let us further observe that for all $s \geq 1$ and $0 \leq r \leq l$, we have

$$\omega_{l+r:k+r}(s) \leq \omega_{l:k}(s) \leq \omega_{l-r:k-r}(s) \tag{B.1.186}$$

This implies that

$$1 + 2 \cdot \omega_{1:k}(1) + \omega_{0:k}(2) \leq \theta_{0:k}(2) \leq 1 + 2 \cdot \omega_{0:k-1}(1) + \omega_{0:k}(2) \tag{B.1.187}$$

and, similarly,

$$1 + 3 \cdot \omega_{2:k}(1) + 3 \cdot \omega_{1:k}(2) + \omega_{0:k}(3) \leq \theta_{0:k}(3) \leq 1 + 3 \cdot \omega_{0:k-2}(1) + 3 \cdot \omega_{0:k-1}(2) + \omega_{0:k}(3) \tag{B.1.188}$$

Hypothesis. We can thus see the binomial coefficients arising, and we would expect that in general, we have

$$\sum_{r=0}^{s} \binom{s}{r} \cdot \omega_{0:k-s+r}(r) \geq \theta_{0:k}(s) \geq \sum_{r=0}^{s} \binom{s}{r} \cdot \omega_{s-r:k}(r) \tag{B.1.189}$$

Lemma B.1.28. *For all $k \geq s \geq 1$, we have*

$$\sum_{r=0}^{s} \binom{s}{r} \cdot \omega_{0:k-s+r}^{(t)}(r) \geq \theta_{0:k}^{(t)}(s) \geq \sum_{r=0}^{s} \binom{s}{r} \cdot \omega_{s-r:k}^{(t)}(r) \tag{B.1.190}$$

PROOF. Let us prove this lemma by induction on s. The cases $s = 1{,}2{,}3$ have already been treated in the previous remark.

Let us now assume that the claim holds for s, and prove it for $s+1$ using the recursive formula

$$\theta_{0:k}(s+1) = \theta_{0:k-1}(s) + \sum_{j=s}^{k-1} \frac{1}{t+j}\theta_{0:j}(s) \tag{B.1.191}$$

For the lower bound, using the induction hypothesis, we get

$$\begin{aligned}
\theta_{0:k}(s+1) &\geq \sum_{r=0}^{s} \binom{s}{r} \cdot \omega_{s-r:k-1}(r) + \sum_{j=s}^{k-1} \frac{1}{t+j} \sum_{r=0}^{s} \binom{s}{r} \cdot \omega_{s-r:j}(r) && \text{(B.1.192)}\\
&= \sum_{r=0}^{s} \binom{s}{r} \cdot \omega_{s-r:k-1}(r) + \sum_{r=0}^{s} \binom{s}{r} \cdot \sum_{j=s}^{k-1} \frac{1}{t+j} \cdot \omega_{s-r:j}(r) && \text{(B.1.193)}\\
&= \sum_{r=0}^{s} \binom{s}{r} \cdot \omega_{s-r:k-1}(r) + \sum_{r=0}^{s} \binom{s}{r} \cdot \omega_{s-r:k}(r+1) && \text{(B.1.194)}\\
&= \sum_{r=0}^{s} \binom{s}{r} \cdot \omega_{s-r:k-1}(r) + \sum_{r=1}^{s+1} \binom{s}{r-1} \cdot \omega_{s-r+1:k}(r) && \text{(B.1.195)}\\
&= 1 + \omega_{0:k}(s+1) + \sum_{r=1}^{s} \left[\binom{s}{r} + \binom{s}{r-1}\right] \cdot \omega_{s-r+1:k}(r) && \text{(B.1.196)}\\
&= 1 + \omega_{0:k}(s+1) + \sum_{r=1}^{s} \binom{s+1}{r} \cdot \omega_{s-r+1:k}(r) && \text{(B.1.197)}\\
&= \sum_{r=0}^{s+1} \binom{s+1}{r} \cdot \omega_{s-r+1:k}(r) && \text{(B.1.198)}\\
& && \text{(B.1.199)}
\end{aligned}$$

For the upper bound, using the induction hypothesis, we get

$$\theta_{0:k}(s+1) \leq \sum_{r=0}^{s} \binom{s}{r} \cdot \omega_{0:k-1-(s-r)}(r) + \sum_{j=s}^{k-1} \frac{1}{t+j} \sum_{r=0}^{s} \binom{s}{r} \cdot \omega_{0:j-(s-r)}(r) \tag{B.1.200}$$

$$= \sum_{r=0}^{s} \binom{s}{r} \cdot \omega_{0:k-1-(s-r)}(r) + \sum_{r=0}^{s} \binom{s}{r} \cdot \sum_{j=s}^{k-1} \frac{1}{t+j} \cdot \omega_{0:j-(s-r)}(r) \tag{B.1.201}$$

$$\leq \sum_{r=0}^{s} \binom{s}{r} \cdot \omega_{0:k-1-(s-r)}(r) + \sum_{r=0}^{s} \binom{s}{r} \cdot \sum_{j=s}^{k-1} \frac{1}{t+j-(s-r)} \cdot \omega_{0:j-(s-r)}(r) \tag{B.1.202}$$

$$= \sum_{r=0}^{s} \binom{s}{r} \cdot \omega_{0:k-1-(s-r)}(r) + \sum_{r=0}^{s} \binom{s}{r} \cdot \sum_{j'=r}^{k-1-(s-r)} \frac{1}{t+j'} \cdot \omega_{0:j'}(r) \tag{B.1.203}$$

$$= \sum_{r=0}^{s} \binom{s}{r} \cdot \omega_{0:k-1-(s-r)}(r) + \sum_{r=0}^{s} \binom{s}{r} \cdot \omega_{0:k-(s-r)}(r+1) \tag{B.1.204}$$

$$= \sum_{r=0}^{s} \binom{s}{r} \cdot \omega_{0:k-1-(s-r)}(r) + \sum_{r=1}^{s+1} \binom{s}{r-1} \cdot \omega_{0:k-(s+1-r)}(r) \tag{B.1.205}$$

$$= 1 + \omega_{0:k}(s+1) + \sum_{r=1}^{s} \left[\binom{s}{r} + \binom{s}{r-1} \right] \cdot \omega_{0:k-(s+1-r)}(r) \tag{B.1.206}$$

$$= 1 + \omega_{0:k}(s+1) + \sum_{r=1}^{s} \binom{s+1}{r} \cdot \omega_{0:k-(s+1-r)}(r) \tag{B.1.207}$$

$$= \sum_{r=0}^{s+1} \binom{s+1}{r} \cdot \omega_{0:k-(s+1-r)}(r) \tag{B.1.208}$$

$$\tag{B.1.209}$$

completing the proof by induction. □

Remark B.1.29. Let us recall that

$$\omega_{l:k}(r) = \sum_{q=0}^{r} \psi_{r-q} \frac{(\ln \beta_{t+l-1,t+k-1})^q}{q!} \tag{B.1.210}$$

Thus the difference between the upper-bound and the lower-bound becomes

$$\sum_{r=0}^{s} \binom{s}{r} \left[\omega_{o:k-(s-r)}(r) - \omega_{s-r:k}(r) \right] = \sum_{r=0}^{s} \binom{s}{r} \cdot \left[\sum_{q=0}^{r} \psi_{r-q} \frac{(\ln \beta_{t-1,t+k-(s-r)-1})^q - (\ln \beta_{t+s-r-1,t+k-1})^q}{q!} \right] \tag{B.1.211}$$

which converges to zero as $T = t + k \to \infty$.

B.1.3.4. Putting it all together.

Remark B.1.30. Now it is time to turn to $\chi_{0:k}^{(t)}(s)$ and finally put it all together, so that we can finally estimate

$$\frac{ds_{t+k}}{dh_t} = \sum_{s=0}^{k} V^s \cdot \chi_{0:k}^{(t)}(s) \tag{B.1.212}$$

and get the asymptotic estimate when $T = t + k \to \infty$.

Let us recall that

$$\chi_{0:k}^{(t)}(s) = \frac{1}{t+k} \cdot \theta_{0:k}^{(t)}(s) + \frac{1}{t+k-1} \cdot \theta_{0:k-1}^{(t)}(s-1) + \dots \tag{B.1.213}$$

$$\dots + \frac{1}{t+k-s+1} \cdot \theta_{0:k-s+1}^{(t)}(1) + \frac{1}{t+k-s} + 1_{k=s} \tag{B.1.214}$$

Using the abuse of notation $\theta_{l:k}(0) = 1$ for $l < k$, we can rewrite it as follows

$$\chi_{0:k}^{(t)}(s) = 1_{k=s} + \sum_{i=0}^{s} \frac{1}{t+k-i} \cdot \theta_{0:k-i}(s-i) \tag{B.1.215}$$

The idea is to use the inequality from lemma B.1.28, and get a similar result for $\chi_{0:k}^{(t)}(s)$, then show that the lower and upper bound are no more than $\Theta(1/T)$ apart, thus enabling us to eventually get an asymptotic estimate for $\frac{ds_{t+k}}{dh_t}$.

We are also omitting the superscript (t) notation here because of lack of ambiguity.

Lemma B.1.31. *For all $s \geq 0$ and $k \geq 1$, we have*

$$1_{k=s} + \frac{1}{t+k} \cdot \sum_{r=0}^{s} \binom{s+1}{r+1} \cdot \omega_{s-r:k}(r) \leq \chi_{0:k}(s) \leq 1_{k=s} + \frac{1}{t+k-s} \cdot \sum_{r=0}^{s} \binom{s+1}{r+1} \cdot \omega_{0:k-(s-r)}(r) \tag{B.1.216}$$

PROOF. Using the upper-bound of lemma B.1.28, we get

$$\chi_{0:k}(s) = 1_{k=s} + \sum_{i=0}^{s} \frac{1}{t+k-i} \cdot \theta_{0:k-i}(s-i) \tag{B.1.217}$$

$$\leq 1_{k=s} + \sum_{i=0}^{s} \frac{1}{t+k-i} \cdot \sum_{r=0}^{s-i} \binom{s-i}{r} \cdot \omega_{0:k-s+r}(r) \tag{B.1.218}$$

$$\leq 1_{k=s} + \frac{1}{t+k-s} \cdot \sum_{i=0}^{s} \sum_{r=0}^{s-i} \binom{s-i}{r} \cdot \omega_{0:k-s+r}(r) \tag{B.1.219}$$

$$= 1_{k=s} + \frac{1}{t+k-s} \cdot \sum_{r=0}^{s} \left[\sum_{i=0}^{s-r} \binom{s-i}{r} \right] \cdot \omega_{0:k-s+r}(r) \tag{B.1.220}$$

$$= 1_{k=s} + \frac{1}{t+k-s} \cdot \sum_{r=0}^{s} \binom{s+1}{r+1} \cdot \omega_{0:k-s+r}(r) \tag{B.1.221}$$

Similarly, using the lower-bound of lemma B.1.28, we get

$$\chi_{0:k}(s) = 1_{k=s} + \sum_{i=0}^{s} \frac{1}{t+k-i} \cdot \theta_{0:k-i}(s-i) \tag{B.1.222}$$

$$\geq 1_{k=s} + \sum_{i=0}^{s} \frac{1}{t+k-i} \cdot \sum_{r=0}^{s-i} \binom{s-i}{r} \cdot \omega_{s-r:k}(r) \tag{B.1.223}$$

$$\geq 1_{k=s} + \frac{1}{t+k} \cdot \sum_{i=0}^{s} \sum_{r=0}^{s-i} \binom{s-i}{r} \cdot \omega_{s-r:k}(r) \tag{B.1.224}$$

$$= 1_{k=s} + \frac{1}{t+k} \cdot \sum_{r=0}^{s} \left[\sum_{i=0}^{s-r} \binom{s-i}{r} \right] \cdot \omega_{s-r:k}(r) \tag{B.1.225}$$

$$= 1_{k=s} + \frac{1}{t+k} \cdot \sum_{r=0}^{s} \binom{s+1}{r+1} \cdot \omega_{s-r:k}(r) \tag{B.1.226}$$

□

Lemma B.1.32. *For all $s \geq 0$, we have*

$$\chi_{0:k}(s) = 1_{k=s} + \frac{1}{t+k} \left[\sum_{r=0}^{s} \binom{s+1}{r+1} \cdot \omega_{s-r:k}(r) \right] + \Theta\left(\frac{1}{t+k}\right) \tag{B.1.227}$$

for all large enough $k > 1$, and where the implicit constants from the $\Theta(.)$ notation are dependent on s.

PROOF. Building on the previous lemma B.1.31, and substracting the lower bound from the upper bound, we get

$$\sum_{r=0}^{s} \binom{s+1}{r+1} \cdot \left[\frac{\omega_{0:k-s+r}(r)}{t+k-s} - \frac{\omega_{s-r:k}(r)}{t+k} \right] = \sum_{r=0}^{s} \sum_{q=0}^{r} \binom{s+1}{r+1} \frac{\psi_{r-q}}{q!} \cdot \left[\frac{(\ln \beta_{t-1,t+k-s+r-1})^q}{t+k-s} - \frac{(\ln \beta_{t+s-r-1,t+k-1})^q}{t+k} \right. \tag{B.1.228}$$

When assuming that for large k, we have

$$(\ln \beta_{t-1,t+k-s+r-1})^q \approx (\ln \beta_{t+s-r-1,t+k-1})^q \tag{B.1.229}$$

then

$$\frac{(\ln \beta_{t-1,t+k-s+r-1})^q}{t+k-s} - \frac{(\ln \beta_{t+s-r-1,t+k-1})^q}{t+k} \approx \frac{1}{t+k} \cdot \left[\frac{s}{t+k-s} \cdot (\ln \beta_{t-1,t+k-s+r-1})^q \right] \tag{B.1.230}$$

$$\leq \frac{1}{t+k} \cdot \left[\frac{s}{t+k-s} \cdot (\ln \beta_{t-1,t+k-s+r-1})^s \right] \tag{B.1.231}$$

$$\leq \frac{\tau_s}{t+k} \tag{B.1.232}$$

for some $\tau_s > 0$ depending on s, for all sufficiently large k.

In other words, we have

$$\sum_{r=0}^{s} \binom{s+1}{r+1} \cdot \left[\frac{\omega_{0:k-s+r}(r)}{t+k-s} - \frac{\omega_{s-r:k}(r)}{t+k} \right] \leq \frac{\tilde{\tau}_s}{t+k} \tag{B.1.233}$$

for for some $\tilde{\tau}_s > 0$ depending on s, for all sufficiently large k.

Meanwhile, for all large enough k, we have

$$\begin{aligned}
\sum_{r=0}^{s} \binom{s+1}{r+1} \cdot \left[\frac{\omega_{0:k-s+r}(r)}{t+k-s} - \frac{\omega_{s-r:k}(r)}{t+k} \right] &\approx \frac{s}{(t+k)(t+k-s)} \cdot \sum_{r=0}^{s} \sum_{q=0}^{r} \binom{s+1}{r+1} \frac{\psi_{r-q}}{q!} \cdot (\ln \beta_{t-1,t+k-s+r-1})^q && \text{(B.1.234)} \\
&\geq \frac{\tau'_s}{(t+k)^2} \cdot \sum_{r=0}^{s} \sum_{q=0}^{r} \binom{s+1}{r+1} \frac{\psi_{r-q}}{q!} \cdot (\ln \beta_{t-1,t+k-s+r-1})^q && \text{(B.1.235)} \\
&\geq \frac{\tau'_s}{(t+k)^2} \cdot \sum_{r=0}^{s} \sum_{q=0}^{r} \frac{\psi_{r-q}}{q!} \cdot (\ln \beta_{t-1,t+k-s+r-1})^q && \text{(B.1.236)} \\
&= \frac{\tau'_s}{(t+k)^2} \cdot \sum_{q=0}^{s} \sum_{r'=0}^{s-q} \frac{\psi_{r'}}{q!} \cdot (\ln \beta_{t-1,t+k-s+r'+q-1})^q && \text{(B.1.237)} \\
&\approx \frac{\tau'_s}{(t+k)^2} \cdot \sum_{q=0}^{s} \left(\sum_{r'=0}^{s-q} \psi_{r'} \right) \cdot \frac{(\ln (t+k))^q}{q!} && \text{(B.1.238)} \\
&\geq \frac{\tau''_s}{(t+k)^2} \cdot \sum_{q=0}^{s} \frac{(\ln (t+k))^q}{q!} && \text{(B.1.239)} \\
&\approx \frac{\tau''_s \cdot \exp\{[\ln (t+k)]\}}{(t+k)^2} && \text{(B.1.240)} \\
&= \frac{\tau''_s}{t+k} && \text{(B.1.241)}
\end{aligned}$$

for some $\tau'_s, \tau''_s > 0$ depending on s. □

Proposition B.1.33. *If V is a normal matrix with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ of modulus smaller than 1, then*

$$\frac{ds_T}{dh_t} = P \Lambda_T P^* \tag{B.1.242}$$

where P^* is the conjugate transpose of the unitary matrix P (independent of T) and where Λ_T is a diagonal matrix satisfying

$$(\Lambda_T)_{ii} \sim T^{-1} \cdot c + T^{\lambda_i - 1} \cdot c' \tag{B.1.243}$$

for some positive real constants c,c', as $T \to \infty$.

PROOF. Let $V = P\Lambda P^*$ be the Schur decomposition of V, with $\Lambda = \text{diag}(\lambda_1,\lambda_2,\ldots,\lambda_n)$. Note that since we supposed that V is normal, we thus have that the Schur matrix Λ is indeed diagonal and is composed of the eigenvalues on the diagonal.

Based on lemma B.1.32, one can show that there exists a function $g : \mathbb{N} \to \mathbb{R}_0^+$ such that

$$\chi_{0:k}(s) = 1_{k=s} + \frac{1}{t+k}\left[\sum_{r=0}^{s}\binom{s+1}{r+1}\cdot\omega_{s-r:k}(r) + g(s)\right] \tag{B.1.244}$$

Thus

$$\begin{aligned}
\frac{ds_{t+k}}{dh_t} &= \sum_{s=0}^{k} V^s \cdot \chi_{0:k}(s) && \text{(B.1.245)}\\
&= V^k + \frac{1}{t+k}\left[\sum_{s=0}^{k} g(s)\cdot V^s + \sum_{s=0}^{k}\sum_{r=0}^{s}\binom{s+1}{r+1}\cdot\omega_{s-r:k}(r)\cdot V^s\right] && \text{(B.1.246)}\\
&= V^k + \frac{1}{t+k}\left[\sum_{s=0}^{k} g(s)\cdot V^s + \sum_{s=0}^{k}\sum_{r=0}^{s}\sum_{q=0}^{r}\binom{s+1}{r+1}\cdot\psi_{r-q}\frac{(\ln\beta_{t+s-r-1,t+k-1})^q}{q!}\cdot V^s\right] && \text{(B.1.247)}\\
& && \text{(B.1.248)}
\end{aligned}$$

Since the eigenvalues of V are of modulus smaller than 1, we can assume that there exists a constant $d > 0$ (dependent on the choice of eigenvalues of V) such that for all $k > d$ we have $V^k \approx 0$.

Furthermore since $V^m = (P\Lambda P^*)^m = P\Lambda^m P^*$ for all $m \in \mathbb{N}_0$, while keeping in mind that we pick $T = t + k$, we can write

$$\Lambda_T = \frac{1}{T}\left[\sum_{s=0}^{d} g(s)\cdot\Lambda^s + \sum_{s=0}^{d}\sum_{r=0}^{s}\sum_{q=0}^{r}\binom{s+1}{r+1}\cdot\psi_{r-q}\frac{(\ln\beta_{t+s-r-1,T-1})^q}{q!}\cdot\Lambda^s\right] \tag{B.1.249}$$

$$= \frac{1}{T}\left[\sum_{s=0}^{d} g(s)\cdot\Lambda^s + \sum_{s=0}^{d}\sum_{q=0}^{s}\sum_{r=q}^{s}\binom{s+1}{r+1}\cdot\psi_{r-q}\frac{(\ln\beta_{t+s-r-1,T-1})^q}{q!}\cdot\Lambda^s\right] \tag{B.1.250}$$

$$= \frac{1}{T}\left[\sum_{s=0}^{d} g(s)\cdot\Lambda^s + \sum_{q=0}^{d}\sum_{s=q}^{d}\sum_{r=q}^{s}\binom{s+1}{r+1}\cdot\psi_{r-q}\frac{(\ln\beta_{t+s-r-1,T-1})^q}{q!}\cdot\Lambda^s\right] \tag{B.1.251}$$

$$= \frac{1}{T}\left[\sum_{s=0}^{d} g(s)\cdot\Lambda^s + \sum_{q=0}^{d}\sum_{s=q}^{d}\sum_{r=q}^{s}\binom{s+1}{r+1}\cdot\psi_{r-q}\frac{(\Lambda\cdot\ln\beta_{t+s-r-1,T-1})^q}{q!}\cdot\Lambda^{s-q}\right] \tag{B.1.252}$$

$$= \frac{1}{T}\left[\sum_{s=0}^{d} g(s)\cdot\Lambda^s + \sum_{q=0}^{d}\sum_{s'=0}^{d-q}\sum_{r'=0}^{s'}\binom{s'+q+1}{r'+q+1}\cdot\psi_{r'}\frac{(\Lambda\cdot\ln\beta_{t+s'-r'-1,T-1})^q}{q!}\cdot\Lambda^{s'}\right] \tag{B.1.253}$$

$$\sim \frac{1}{T}\left[\sum_{s=0}^{d} g(s)\cdot\Lambda^s + \sum_{q=0}^{d}\sum_{s'=0}^{d-q}\sum_{r'=0}^{s'}\binom{s'+q+1}{r'+q+1}\cdot\psi_{r'}\frac{(\Lambda\cdot\ln T)^q}{q!}\cdot\Lambda^{s'}\right] \tag{B.1.254}$$

$$= \frac{1}{T}\left[\sum_{s=0}^{d} g(s)\cdot\Lambda^s\right] + \frac{1}{T}\left[\sum_{q=0}^{d}\frac{(\Lambda\cdot\ln T)^q}{q!}\cdot\left(\sum_{s'=0}^{d-q}\sum_{r'=0}^{s'}\binom{s'+q+1}{r'+q+1}\cdot\psi_{r'}\cdot\Lambda^{s'}\right)\right] \tag{B.1.255}$$

$$\approx \frac{1}{T}\left[\sum_{s=0}^{d} g(s)\cdot\Lambda^s\right] + \frac{1}{T}\exp\{(\Lambda\cdot\ln T)\}\cdot(c_0 + c_1\cdot\Lambda + \ldots + c_d\cdot\Lambda^d) \tag{B.1.256}$$

$$\sim \frac{c}{T} + \frac{c'}{T}\exp\{(\Lambda\cdot\ln T)\} \tag{B.1.257}$$

for some positive constants c',c,c_0,$c_1,\ldots$,c_d.

Hence

$$(\Lambda_T)_{ii} \sim c\cdot T^{-1} + c'\cdot T^{\lambda_i - 1} \tag{B.1.258}$$

□

Theorem B.1.34. *If V is a normal matrix with eigenvalues of modulus smaller than 1, then*

$$\|\frac{ds_T}{dh_t}\| = \Omega(1/T) \tag{B.1.259}$$

as $T \to \infty$. (here $\|.\|$ is the Frobenius norm.)

PROOF. Let us start off with the observation that

$$T^{-1}\cdot c + T^{\lambda_i - 1}\cdot c' = \Omega\left(T^{-\min(1, 1-\mathfrak{Re}(\lambda_i))}\right) \tag{B.1.260}$$

as $T \to \infty$. And thus, by using proposition B.1.33, we get

$$\|\frac{ds_T}{dh_t}\| = \Omega(T^{-\eta}) \tag{B.1.261}$$

where

$$\eta = \min_{i=1,\ldots,n} \{\min(1,1 - \mathfrak{Re}(\lambda_i))\} \leq 1 \tag{B.1.262}$$

□

Remark B.1.35. Note that V being normal is not a necessary condition for the generality of the theorem to hold. We simply chose V to be normal in order to make the calculations less cumbersome.

In case V is non-normal, its Schur matrix Λ becomes triangular instead of diagonal. In fact, if $t_{i,j}$ are the off-diagonal elements of Schur matrix of V (with $i < j$), then

$$\|V\| = \sqrt{\text{Tr}(V^*V)} = \sqrt{\text{Tr}(\Lambda^*\Lambda)} = \sqrt{\sum_{i=1}^{n} |\lambda_i|^2 + \sum_{i<j} |t_{i,j}|^2} \geq \sqrt{\sum_{i=1}^{n} |\lambda_i|^2} \tag{B.1.263}$$

Thus every lower bound on $\sqrt{\sum_{i=1}^{n} |\lambda_i|^2}$ induces a lower bound on $\|V\|$, and in particular an asymptotic lower bound on the modulus of one of the eigenvalues of $\frac{ds_T}{dh_t}$ induces an asymptotic lower bound on $\|\frac{ds_T}{dh_t}\|$.

B.1.4. Sparse relevance case with bounded dependency depth

Remark B.1.36. Similarly to remark B.1.18, we are going to assume for this subsection:

- no non-linearity in the hidden-to-hidden connection: $J_t = V$ for all t.
- all assumptions from Remark B.1.8.
- κ-sparse attention: for each $t \geq 1$, there are at most $\kappa \leq t$ values for i such that $\alpha_{i,t} \neq 0$. (Let us define $\kappa_t \overset{\text{def}}{=} |\{i \text{ such that } \alpha_{i,t} \neq 0\}|$)
- uniform attention across attended states: for all $t \geq 1$, and all $i \leq t$ such that $\alpha_{i,t} \neq 0$, we have $\alpha_{i,t} = 1/\kappa_t \geq 1/\kappa$.

Remark B.1.37. Similarly to remark B.1.19, let us recall that

$$X_{i,t} = \left(h_i - \sum_{j=1}^{t} \alpha_{j,t} h_j\right) \cdot \frac{\partial e_{i,t}}{\partial h_i} \tag{B.1.264}$$

and that

$$\sum_{i=1}^{t} \alpha_{i,t} Y_{i,t} = \sum_{i=1}^{t} \alpha_{i,t} \cdot h_i \cdot \left(\frac{\partial e_{i,t}}{\partial s_{t-1}} - \sum_{j=1}^{t} \alpha_{j,t} \cdot \frac{\partial e_{j,t}}{\partial s_{t-1}} \right) \tag{B.1.265}$$

$$= \sum_{i=1}^{t} \alpha_{i,t} \cdot h_i \cdot \frac{\partial e_{i,t}}{\partial s_{t-1}} - \sum_{i=1}^{t} \alpha_{i,t} \left(\sum_{j=1}^{t} \alpha_{j,t} \cdot h_j \right) \cdot \frac{\partial e_{i,t}}{\partial s_{t-1}} \tag{B.1.266}$$

$$= \sum_{i=1}^{t} \alpha_{i,t} \cdot \left(h_i - \sum_{j=1}^{t} \alpha_{j,t} h_j \right) \cdot \frac{\partial e_{i,t}}{\partial s_{t-1}} \tag{B.1.267}$$

Thus we can see that both expressions have the common factor $\left(h_i - \sum_{j=1}^{t} \alpha_{j,t} h_j\right)$.

By defining

$$A_t \overset{\text{def}}{=} \{i \text{ such that } \alpha_{i,t} \neq 0\} \tag{B.1.268}$$

we see that

$$h_i - \sum_{j=1}^{t} \alpha_{j,t} h_j = h_i - \frac{1}{\kappa_t} \sum_{j \in A_t} h_j \tag{B.1.269}$$

and we are going to assume for the sake of simplicity that

$$h_i \approx \frac{1}{\kappa_t} \sum_{j \in A_t} h_j \tag{B.1.270}$$

and thus $X_{i,t} \approx 0$ and $\sum_{i=1}^{t} \alpha_{i,t} Y_{i,t} \approx 0$.

Recalling the expression from corollary B.1.11 and that $f(h_t, c_t) = h_t + c_t$ by remark B.1.8, and that $J_t = V$ for all t, this will give for all $k' \geq 0$

$$E_{k'}^{(t)} = \left(\frac{1_{t \in A_{t+k'}}}{\kappa_{t+k'}} + 1_{k'=0} \right) \cdot \mathrm{I} \tag{B.1.271}$$

and for all $k \geq j$, we get

$$F_{k+1,j}^{(t)} = \left(\frac{1_{t+j+1 \in A_{t+k+1}}}{\kappa_{t+k+1}} + 1_{k=j} \right) \cdot V \tag{B.1.272}$$

Hence by recalling proposition B.1.7, the main expression of interest becomes

$$\frac{ds_{t+k}}{dh_t} = \sum_{s=0}^{k} \bar{\xi}_{0:k}^{(t)}(s) = \sum_{s=0}^{k} V^s \cdot \chi_{0:k}^{(t)}(s) \tag{B.1.273}$$

where

$$\chi_{0:k}^{(t)}(s) \overset{\text{def}}{=} \sum_{0 \leq i_1 < \ldots < i_s < k} \left(\frac{1_{t+i_s+1 \in A_{t+k}}}{\kappa_{t+k}} + 1_{k-i_s=1} \right) \cdot \left(\frac{1_{t+i_{s-1}+1 \in A_{t+i_s}}}{\kappa_{t+i_s}} + 1_{i_s - i_{s-1}=1} \right) \cdot \tag{B.1.274}$$

$$\ldots \cdot \left(\frac{1_{t+i_1+1 \in A_{t+i_2}}}{\kappa_{t+i_2}} + 1_{i_2-i_1=1} \right) \cdot \left(\frac{1_{t \in A_{t+i_1}}}{\kappa_{t+i_1}} + 1_{i_1=0} \right) \tag{B.1.275}$$

Remark B.1.38. Let us now have a look at how we could potentially simplify the analysis of $\chi_{0:k}^{(t)}(s)$.

If we further assume V to be normal we can write

$$V = P\Lambda P^* \tag{B.1.276}$$

where $\Lambda = \text{diag}(\lambda_1,\lambda_2,\ldots,\lambda_n)$ is the diagonal matrix consisting of the eigenvalues of V, and P^* is the conjugate transpose of P.

Hence, we can rewrite

$$\frac{ds_{t+k}}{dh_t} = \sum_{s=0}^{k} V^s \cdot \chi_{0:k}^{(t)}(s) = P \cdot \left(\sum_{s=0}^{k} \Lambda^s \cdot \chi_{0:k}^{(t)}(s)\right) \cdot P^* \tag{B.1.277}$$

We can therefore see that the asymptotic behaviour of $\frac{ds_{t+k}}{dh_t}$ depends largely on the asymptotic behaviour of the modulus of the complex-valued polynomial

$$p_{0:k}(\lambda) \overset{\text{def}}{=} \sum_{s=0}^{k} \lambda^s \cdot \chi_{0:k}^{(t)}(s) \tag{B.1.278}$$

and thus

$$\|\frac{ds_{t+k}}{dh_t}\| = \sqrt{\sum_{i=1}^{n} |p_{0:k}(\lambda_i)|^2} \tag{B.1.279}$$

where $\|.\|$ is the Frobenius norm. Hence in order to prove that

$$\|\frac{ds_{t+k}}{dh_t}\| = \Omega(1/\kappa^d) \tag{B.1.280}$$

for all large enough k (note that k and κ here are two different symbols), it would suffice to show that there exists $\lambda \in \{\lambda_1,\ldots,\lambda_n\}$ such that, for all large enough k, we have

$$|p_{0:k}(\lambda)| = \Omega(1/\kappa^d) \tag{B.1.281}$$

For simplicity we are going to assume that for all t, we have $\kappa_t = \kappa$.

Let us further define for all $s \geq 1$,

$$f_{0:k}^{(s)}(i_1,\ldots,i_s) \overset{\text{def}}{=} \left(\frac{1_{t+i_s+1 \in A_{t+k}}}{\kappa_{t+k}} + 1_{k-i_s=1}\right) \cdot \left(\frac{1_{t+i_{s-1}+1 \in A_{t+i_s}}}{\kappa_{t+i_s}} + 1_{i_s-i_{s-1}=1}\right) \cdot \ldots \tag{B.1.282}$$

$$\ldots \cdot \left(\frac{1_{t+i_1+1 \in A_{t+i_2}}}{\kappa_{t+i_2}} + 1_{i_2-i_1=1}\right) \cdot \left(\frac{1_{t \in A_{t+i_1}}}{\kappa_{t+i_1}} + 1_{i_1=0}\right) \tag{B.1.283}$$

whenever $(i_1,\ldots,i_n)$ satisfies $0 \leq i_1 < i_2 < \ldots < i_s < k$, and

$$f_{0:k}^{(s)}(i_1,\ldots,i_s) \overset{\text{def}}{=} 0 \tag{B.1.284}$$

otherwise.

Theorem B.1.39. *Given the κ-sparsity assumption and the dependency depth d, we have that if V is normal and has one positive real eigenvalue, then*

$$\|\frac{ds_{t+k}}{dh_t}\| = \Omega(1/\kappa^d) \tag{B.1.285}$$

for all large enough k.

PROOF. By the hypothesis on the dependency depth d, we know that for each k, there exists $s' \leq d$ and $(i_1,i_2,\dots,i_{s'})$ such that

$$f_{0:k}^{(s')}(i_1,\dots,i_{s'}) \geq \left(\frac{1}{\kappa}\right)^{s'+1} \geq \left(\frac{1}{\kappa}\right)^{d+1} \tag{B.1.286}$$

Hence if λ is real and positive, then for all large enough k, we have

$$|p_{0:k}(\lambda)| = \Omega(1/\kappa^d) \tag{B.1.287}$$

Let us recall that, since V is normal we can write

$$\|\frac{ds_{t+k}}{dh_t}\| = \sqrt{\sum_{i=1}^{n} |p_{0:k}(\lambda_i)|^2} \tag{B.1.288}$$

where $\lambda_1,\dots,\lambda_n$ are the eigenvalues of V.

Hence, if V has at least one positive real eigenvalue then

$$\|\frac{ds_{t+k}}{dh_t}\| = \Omega(1/\kappa^d) \tag{B.1.289}$$

for all large enough k. □

Remark B.1.40. As already mentioned, since κ and d are assumed to be constant, the theorem states that

$$\|\frac{ds_{t+k}}{dh_t}\| = \Omega(1) \tag{B.1.290}$$

The dependence on κ and d was simply given in order to get an intuition on how κ and d are influencing the lower bound, and that d has more leverage on the lower bound than κ.

Regarding the normality of V, the same remark can be made as in remark B.1.35.

Then note that if V is a (real) $n \times n$ matrix, with n odd, then we have at least one real eigenvalue. Thus the restriction of having at least one positive real eigenvalue is not that severe.

Further, one can show that the theorem holds in a slightly more general setting where one might not have at least one positive real eigenvalue.

Let us consider the case where $\kappa = 1$, $|\lambda| < 1$ such that we could consider $\lambda^c \approx 0$ for some large enough positive integer c, and that all states between T and $T - c$ have dependency depth of exactly d (where $T = t + k$), then

$$p_{0:k}(\lambda) = \frac{\lambda^d}{\kappa^d} \cdot (1 + \lambda + \ldots + \lambda^{c-d}) = \frac{\lambda^d}{\kappa^d} \cdot \left(\frac{1 - \lambda^{c-d+1}}{1 - \lambda} \right) \tag{B.1.291}$$

Hence we can see that if we can show that $\left| \frac{1-\lambda^{c-d+1}}{1-\lambda} \right|$ is lower bounded asymptotically by a constant, independent of d and κ, (which it is in this case), then we have

$$|p_{0:k}(\lambda)| = \Omega(1/\kappa^d) \tag{B.1.292}$$

We also see that we would like λ to be sufficiently bounded away from a small set of critical values such as the $(c - d)$-th roots of unity.

In a more general setting, we can rewrite

$$p_{0:k}(\lambda) = \frac{\lambda^d}{\kappa^d} \cdot q_{0:k}(\lambda) \tag{B.1.293}$$

for some polynomial $q_{0:k}$ with positive real coefficients, and we would like λ to be such that $|q_{0:k}(\lambda)| = \Omega(1)$ for all sufficiently large k.

Our hypothesis is that the theorem holds as long as λ is sufficiently bounded away from a small set of critical values in $\mathbb{C} \setminus \mathbb{R}^+$, or in other words, we would need only at least one eigenvalue to satisfy this condition. This set of critical values is a dependent on κ, d and the overall configuration of the attention weights.

B.2. Effects of memory sparsity on basic reinforcement learning tasks

We consider a few tasks from MiniGrid Chevalier-Boisvert and Willems (2018) in the OpenAI gym Brockman et al. (2016) in which an agent must get to certain goal states. We use a partially observed formulation of the task, where the agent only sees a small number of squares ahead of it. Our goal is to compare generalization of the solutions learned by full and sparse memory-augmented models, by training on smaller version of an environment and testing it on a larger version. To do so, we compare the use of MemLSTM (full attention) and RelLSTM (sparse attention). We note that some purely recurrent models can perform well on these tasks where sequence lengths are rather short, but the scope of this experiment is to explicitly compare the effect of different memory densities.

Table 1. Average Train and Test Rewards for MiniGrid Reinforcement Learning task. The models were trained on the smaller version of the environment and tested on the larger version to test to generalization of the solution learned.

Environment	MemLSTM	RelLSTM
	Train	
RedBlueDoors-6x6	**0.97**	**0.97**
GoToObject-6x6	**0.85**	0.84
MemoryS7	0.4	**0.94**
GoToDoor-5x5	0.17	**0.25**
Fetch-5x5	0.42	**0.5**
DoorKey-5x5	**0.94**	0.93
	Test	
RedBlueDoors-8x8	**0.95**	**0.95**
GoToObject-8x8	0.66	**0.74**
MemoryS13	0.24	**0.30**
GoToDoor-8x8	0.11	**0.15**
Fetch-8x8	0.44	**0.45**
DoorKey-16x16	0.31	**0.44**

These tasks are difficult to solve with standard RL algorithms, due to (1) the partial observability of the environment and (2) the sparsity of the reward, given that the agent receives a reward only after reaching the goal. We use Proximal Policy Optimization (PPO, Schulman et al. (2017)) along with MemLSTM, and RelLSTM as the recurrent modules. All models were each trained for 5000000 steps on each environment. The hyperparameters used for RelLSTM are $\nu = 5$ and $\rho = 5$. On the *MiniGrid-DoorKey-5x5-v0* environment the average reward for MemLSTM is 0.94 and RelLSTM is 0.93. On transferring the learned solution to the *16x16* version of that environment the average reward for MemLSTM is 0.31 and RelLSTM is $\mathbf{0.44}$. As illustrated in 1, we find that transfer scores for RelLSTM are much higher than for MemLSTM across several environments.

B.3. Tradeoff analysis between sparsity and gradient propagation

As already discussed in §4.5, the sparsity coefficient κ verifies $\kappa = \nu + \rho \geq |S_t| + |R_t|$ for all time step t, where we denote ν for the size of the short-term buffer, and ρ for the maximal size of the relevant sets R_t. In this section we would like to see how gradient propagation varies when changing sparsity. As already discussed at the end of §4.4 as well as at the end of §4.5, decreasing κ, would increasingly force gradients to backpropagate through the recurrent connections, thus degrading gradient stability. Meanwhile, increasing κ would increase the size of the computational graph. Thus we would like to find the optimal trade-off between sparsity and gradient propagation. This trade-off is clearly task-specific and needs to be determined experimentally. The only way to do so is by either changing ν or changing ρ (or both). Hence we are going to analyze the effects on gradient propagation by separately changing ν and ρ.

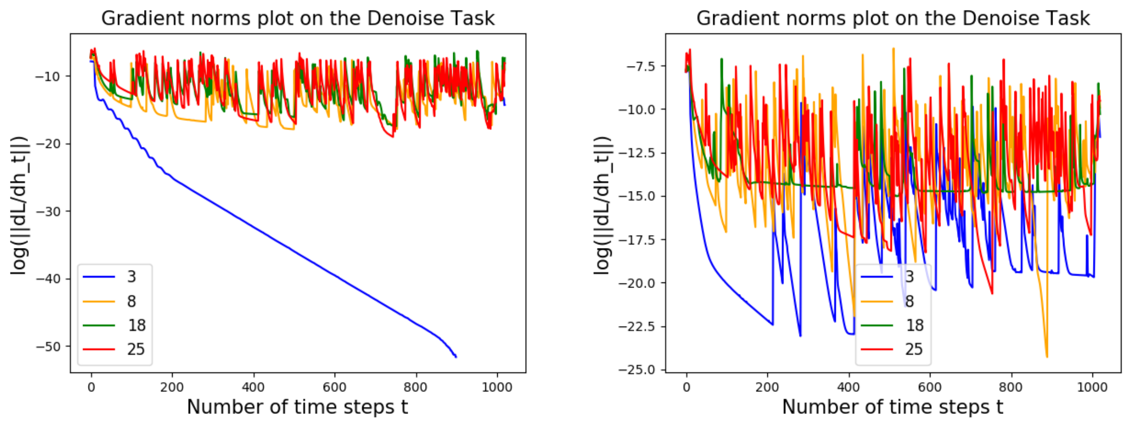


Fig. 1. Both sides show gradient norm plots of $\|\nabla_{h_t} L\|$ in log scale after training for Denoise Task with t ranging from 0 (latest time step) to 1000 (furthest time step). **(Left)** We took four MemLSTM models for $\rho = 3{,}8{,}18{,}25$ while keeping $\nu = 15$ fixed. **(Right)** We took four MemLSTM models for $\nu = 3{,}8{,}18{,}25$ while keeping $\rho = 15$ fixed. (Note that the y-axis of the two plots have different scales, as indicated in the plots.)

For Figure 1 (left), we can see that when choosing ρ too small (here for instance $\rho = 3$), gradient propagation becomes unstable, while larger values for ρ all show stable gradient propagation. This confirms our initial intuition that we can decrease ρ until a task-specific threshold and maintain stable gradient propagation, while decreasing ρ beyond this threshold would cause gradient propagation to become unstable.

For Figure 1 (right), we can see that changing ν has much less leverage on gradient propagation than changing ρ. Gradient propagation stays relatively stable regardless of the values for ν. The only difference is that for the extreme value of $\nu = 3$, we can see that gradient propagation became slightly less stable, because with smaller ν predictions for future relevancy might become less accurate.

B.4. Additional Results

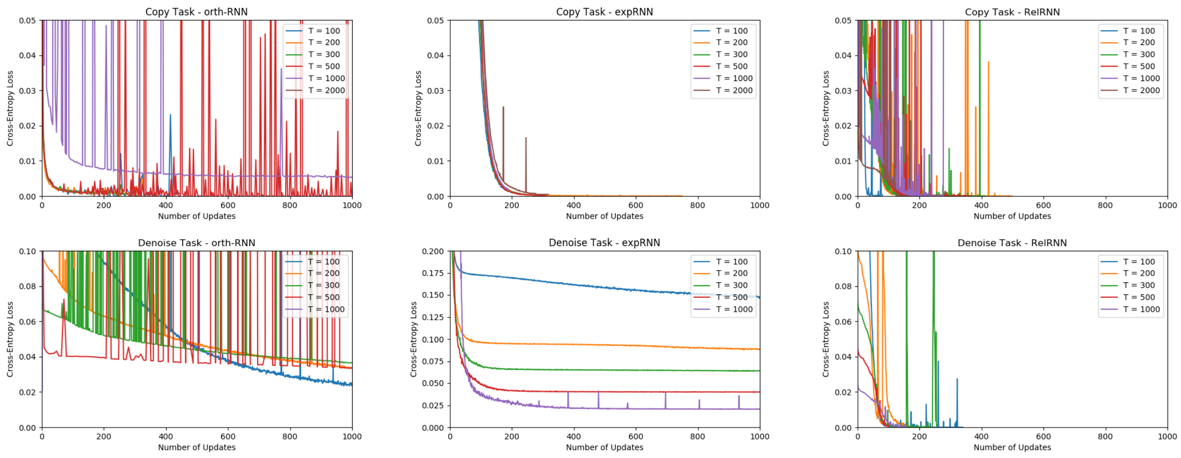


Fig. 2. Cross-entropy vs training updates for Copy (top) and Denoise (bottom) tasks for $T = \{100, 200, 300, 500, 1000, 2000\}$. 1 unit of the x-axis is equal to 100 iterations of training with the exception of expRNN where 1 unit on the x-axis is 10 iterations of training.

Table 2. Results for Copy Task

T	LSTM	orth-RNN	expRNN	MemRNN	SAB	RelRNN	RelLSTM
100	100%	100%	100%	100%	100%	100%	100%
200	100%	100%	100%	100%	100%	100%	100%
300	100%	100%	100%	100%	100%	100%	100%
500	12%	100%	100%	100%	100%	100%	100%
1000	12%	80%	100%	100%	100%	100%	100%
2000	12%	11%	100%	OOM	100%	100%	100%

Table 3. Hyperparameters used for Copy task

Model	lr	optimizer	non-linearity	ν	ρ
orthRNN	0.0002	RMSprop	modrelu	-	-
expRNN	0.0002	RMSprop	modrelu	-	-
LSTM	0.0002	Adam	-	-	-
RelRNN	0.0002	Adam	tanh	10	10

Table 4. Hyperparameters used for Denoise task

Model	lr	optimizer	non-linearity	ν	ρ
orthRNN	0.0002	RMSprop	modrelu	-	-
expRNN	0.0002	RMSprop	modrelu	-	-
LSTM	0.0002	Adam	-	-	-
GORU	0.001	RMSprop	-	-	-
RelRNN	0.0002	RMSprop	modrelu	10	10

Table 5. Hyperparameters used for sequential MNIST

Model	lr (lr orth)	optimizer	non-linearity	ν	ρ
orthRNN	0.0001	Adam	modrelu	-	-
expRNN	0.0001(0.00001)	Adam	modrelu	-	-
LSTM	0.0002		-	-	-
GORU			-	-	
RelRNN	0.0003	Adam	modrelu	10	10

Table 6. Hyperparameters used for PTB

Model	lr (lr orth)	optimizer	non-linearity	ν	ρ
orthRNN	0.001	Adam	tanh	-	-
expRNN	0.003(0.0003)	Adam	tanh	-	-
LSTM	0.0002		-	-	-
GORU			-	-	
RelRNN	0.0003	Adam	tanh	10	5

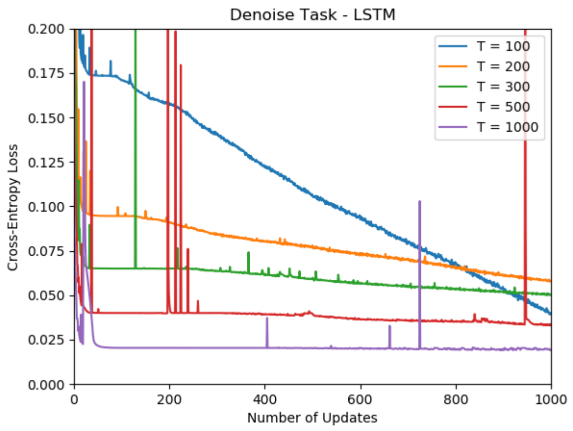


Fig. 3. Training curves for LSTM on Denoise task

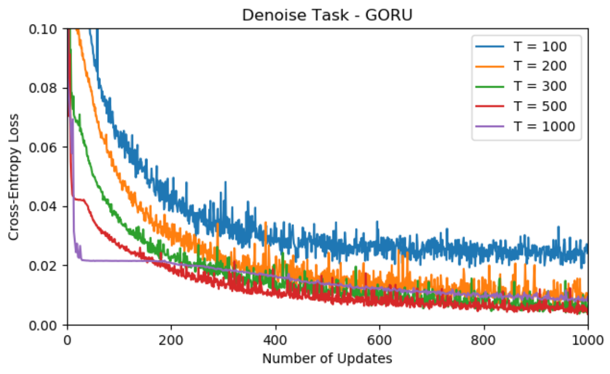


Fig. 4. Training curves for GORU on Denoise task

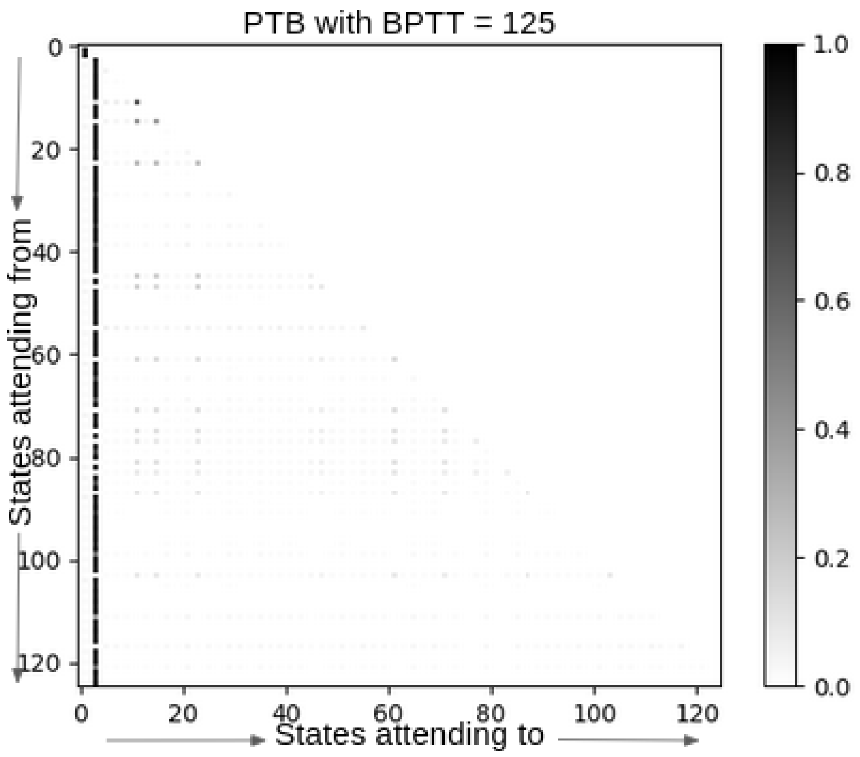


Fig. 5. Heatmap of attention scores on PTB task training with full attention and BPTT of 125

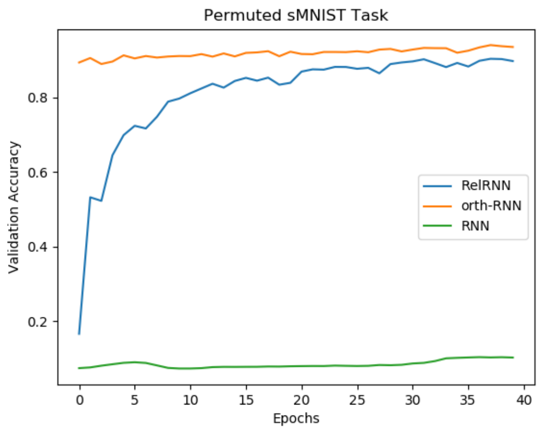


Fig. 6. Validation accuracy curves for pMNIST

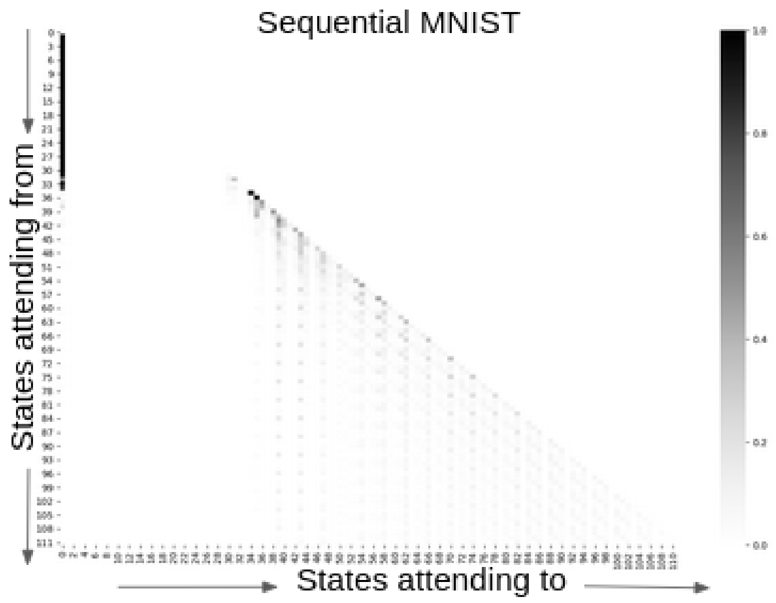


Fig. 7. Heatmap of attention scores on MNIST digit classification. 7 pixels were grouped at each time step to make visualization of heatmap easier.

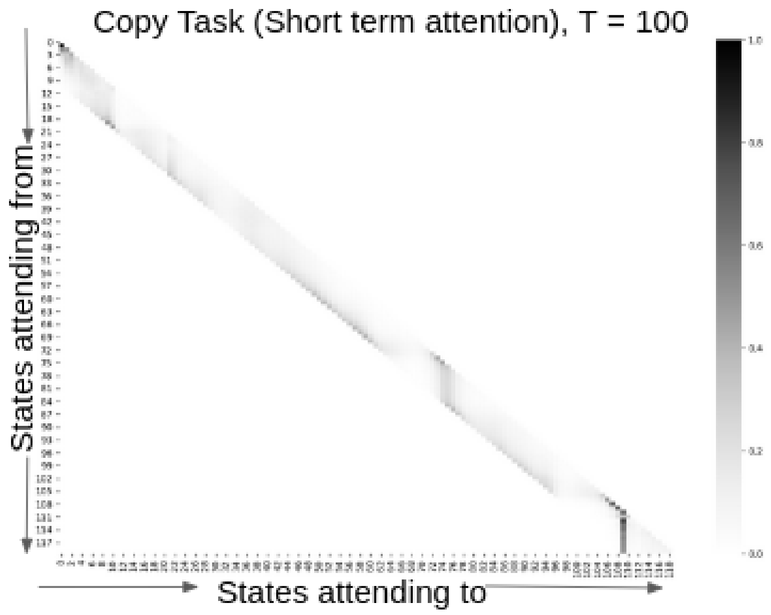


Fig. 8. Heatmap of attention scores on Copy task when only doing attention over the Short Term Buffer.

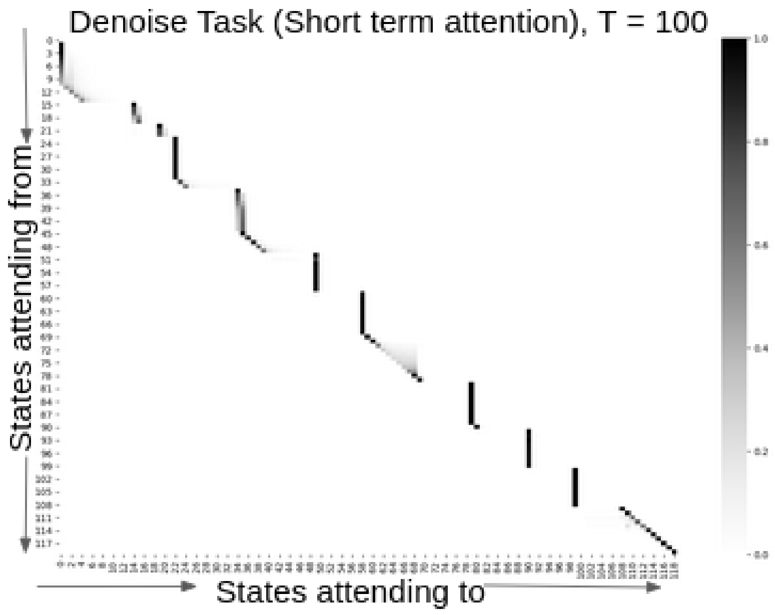


Fig. 9. Heatmap of attention scores on Denoise task when only doing attention over the Short Term Buffer.

Appendix C

Appendix for 'On Neural Architecture Inductive Biases for Relational Tasks'

C.1. Hyperparameters

Table 1 shows the default hyperparameters used for the experiments reported in the main text. All other hyperparamaters, including those of all other models are exactly the same as in Webb et al. (2021).

C.2. Relational Games - additional results

Table 2 displays additional baselines.

C.3. Identity rules 4 (flipped version) results

In the section, we briefly outline a version of the *identity rules 4* task, where training and test set have been swapped, which we call *identity rules 4 [flipped].* In this task, we only train on quadruples with 3 distinct images (only including quadruples of the form $ABCA$, $BACA$ and $BCAA$), while testing only on quadruples with at most two distinct images. We also include a version of this task, with missing variations, where we excluded all quadruples of the form $ABCA$ from the training set. For results, see Figure 1.

C.4. Distribution-of-N results

In this section we are going to look at extensions of the *distribution-of-three* task: we will be looking at distributions of 10 elements, as well as a variation where the set of permutations shown in the training set is restricted and the test set is constructed of permutations not shown during training. We recall that in *distribution-of-three*, the second row in Figure 1 (top left) is a permutation of the first row. The last element of the second row is "removed", and needs to be inferred from the options in the last row, which contains all images shown in the first row and one additional element. In

Table 1. Default hyperparameters

Parameter	Value
Input images size for relational games tasks	$36 \times 36 \times 3$
Iterations for relational games tasks	2500
Input images size for relational cognitive tasks	$32 \times 32 \times 3$
Iterations for relational cognitive tasks	5000
Runs per experiment	10
Optimiser	Gradient descent
Learning rate	$5e-4$
Encoder output dimension	128
Encoder non-linearity	ReLU
Encoder CNN stride	2
Encoder CNN padding	1
Encoder CNN first layer input channels	3
Encoder CNN output channels	32
Encoder CNN kernel size	4
Encoder nb. conv layers	3
Encoder fully connected layer hidden dim	256
CoRelNet decoder	fully connected layer with 1 hidden layer
CoRelNet decoder hidden layer dim	256
CoRelNet normalization type	contextnorm
CoRelNet-T heads	8
CoRelNet-T transformer dim	512
CoRelNet-T query dim	8
CoRelNet-T positional encoding dim	8
CoRelNet-T transformer nb. layers	1
CoRelNet-T normalization type	contextnorm

distribution-of-N, we imagine N distinct images in the first row, and a permutation thereof in the second row. Again, the last element of the second row is "removed" and needs to be inferred from the $N+1$ (the N images shown in the first row and one additional element) options shown in the last row. For the variations of *distribution-of-N*, for which we want to only include unseen relations in the test set, we are going to restrict the training set to only contain permutations (sending the first row to the second row) preserving the set of the first $N-3$ elements, while all the other remaining permutations are used to construct the test set. Hence test and training set have disjoint sets of underlying permutations, and thus relations. For results see Figure 2.

C.5. Details on relational tasks with spurious features

Please see Figure 3.

Task	*Test Set*	*CoRelNet*	*CoRelNet-T*	*MNM*	*LSTM*	*NTM*	*RN*
same	Hex.	$94.7_{\pm 5.4}$	$98.7_{\pm 0.5}$	$97.3_{\pm 1.5}$	$96.4_{\pm 1.6}$	$97.2_{\pm 1.7}$	$92.9_{\pm 2.8}$
	Str.	$90.4_{\pm 9.0}$	$98.4_{\pm 0.6}$	$95.5_{\pm 2.0}$	$90.3_{\pm 6.4}$	$93.4_{\pm 5.3}$	$83.7_{\pm 10.2}$
between	Hex.	$96.6_{\pm 2.2}$	$91.9_{\pm 3.3}$	$93.6_{\pm 2.3}$	$94.9_{\pm 3.3}$	$96.3_{\pm 0.7}$	$73.6_{\pm 8.2}$
	Str.	$93.1_{\pm 5.4}$	$87.0_{\pm 4.7}$	$90.2_{\pm 5.2}$	$91.7_{\pm 4.5}$	$93.5_{\pm 2.5}$	$54.5_{\pm 5.7}$
occurs	Hex.	$96.2_{\pm 2.2}$	$91.6_{\pm 4.6}$	$84.9_{\pm 2.8}$	$92.2_{\pm 2.4}$	$93.5_{\pm 8.1}$	$71.2_{\pm 6.6}$
	Str.	$88.7_{\pm 5.3}$	$79.3_{\pm 12.1}$	$77.0_{\pm 4.5}$	$79.8_{\pm 10.0}$	$87.1_{\pm 11.0}$	$54.6_{\pm 4.0}$
xoccurs	Hex.	$92.2_{\pm 6.4}$	$91.7_{\pm 6.9}$	$73.8_{\pm 6.6}$	$79.1_{\pm 2.8}$	$84.2_{\pm 5.0}$	$65.7_{\pm 5.5}$
	Str.	$83.6_{\pm 10.9}$	$85.4_{\pm 6.4}$	$70.6_{\pm 6.2}$	$77.2_{\pm 5.9}$	$81.7_{\pm 5.8}$	$61.7_{\pm 7.4}$
row matching	Hex.	$97.7_{\pm 0.8}$	$95.4_{\pm 5.1}$	$49.9_{\pm 0.3}$	$50.1_{\pm 0.6}$	$50.2_{\pm 0.5}$	$50.5_{\pm 0.3}$
	Str.	$94.8_{\pm 1.3}$	$90.5_{\pm 5.2}$	$49.8_{\pm 0.6}$	$50.2_{\pm 0.5}$	$49.9_{\pm 0.5}$	$50.4_{\pm 0.4}$
col./shape	Hex.	$47.2_{\pm 3.7}$	$49.6_{\pm 0.8}$	$75.7_{\pm 12.1}$	$80.1_{\pm 7.0}$	$88.0_{\pm 2.4}$	$77.5_{\pm 9.2}$
left-of	Hex.	$99.2_{\pm 0.7}$	$97.6_{\pm 1.2}$	$97.5_{\pm 1.2}$	$97.2_{\pm 1.2}$	$97.5_{\pm 0.9}$	$51.0_{\pm 2.1}$

Table 2. Out-Of-Distribution test accuracies for the Relational Game tasks on the two held-out sets "Hexominoes" (Hex.) and "Stripes" (Str.). Results reflect test accuracies averaged over 10 seeds.

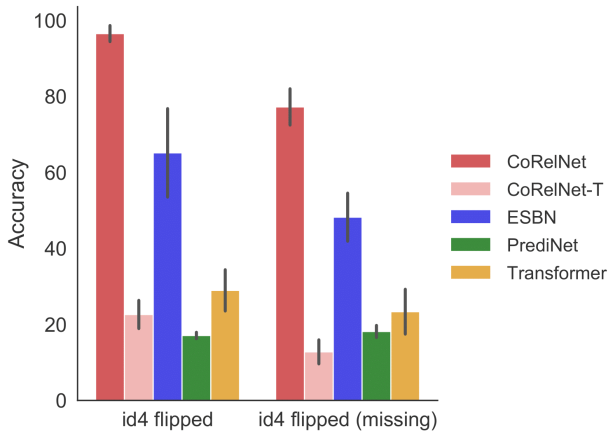


Fig. 1. Results for *identity rules 4 [flipped]* and *identity rules 4 [flipped] (with missing variations)* for $m = 94$, which is the most extreme OOD case. Here m denotes the number of shapes not seen during training. There are a total of 100 shapes, and $m = 94$ here means 6 shapes are shown during training, and testing involves only the 94 unseen shapes. The test result accuracies are averaged over 10 random seeds.

C.6. Spurious features in separated inputs

In this section we investigate a variation of the same different discrimination task with shapes and colours as outlined in Section §5.4.3. Instead of presenting colour and shape in the same image we provide shapes and colours alternatively in different images. Hence we expect that the model

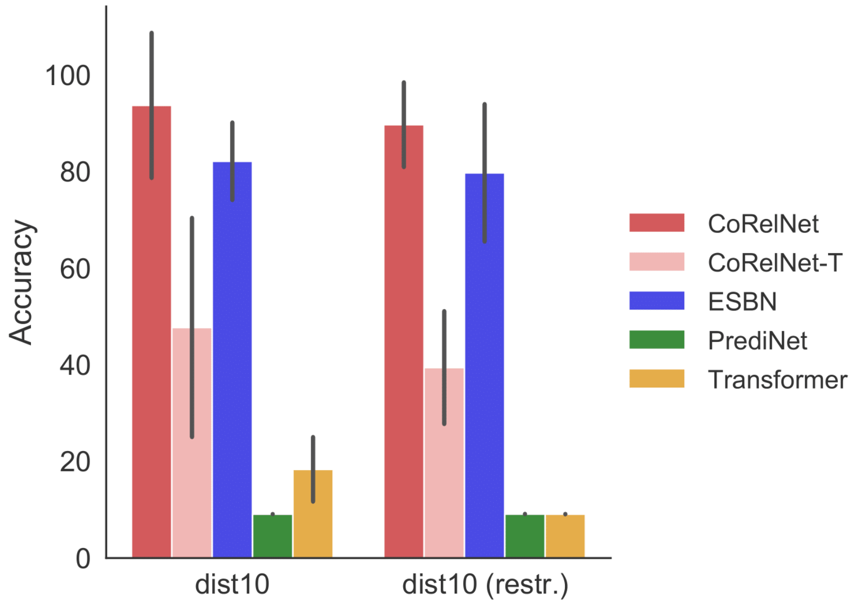


Fig. 2. The test result accuracies for *distribution-of-10* and *distribution-of-10* (restricted - with unseen permutations during testing) for the most extreme OOD case ($m = 89$). Here m denotes the number of shapes not seen during training. There are a total of 100 shapes, and $m = 89$ here means 11 shapes are shown during training, and testing involves only the 89 unseen shapes. The test result accuracies are averaged over 10 random seeds.

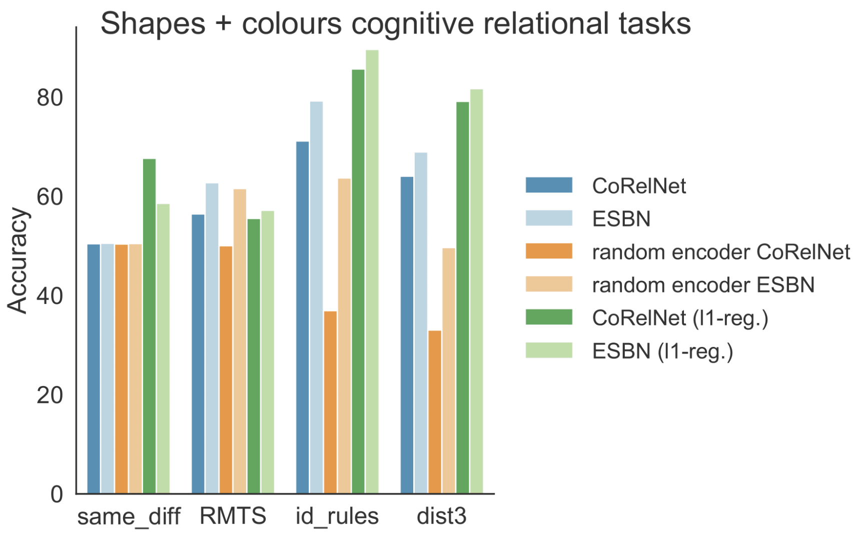
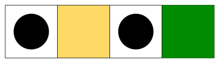
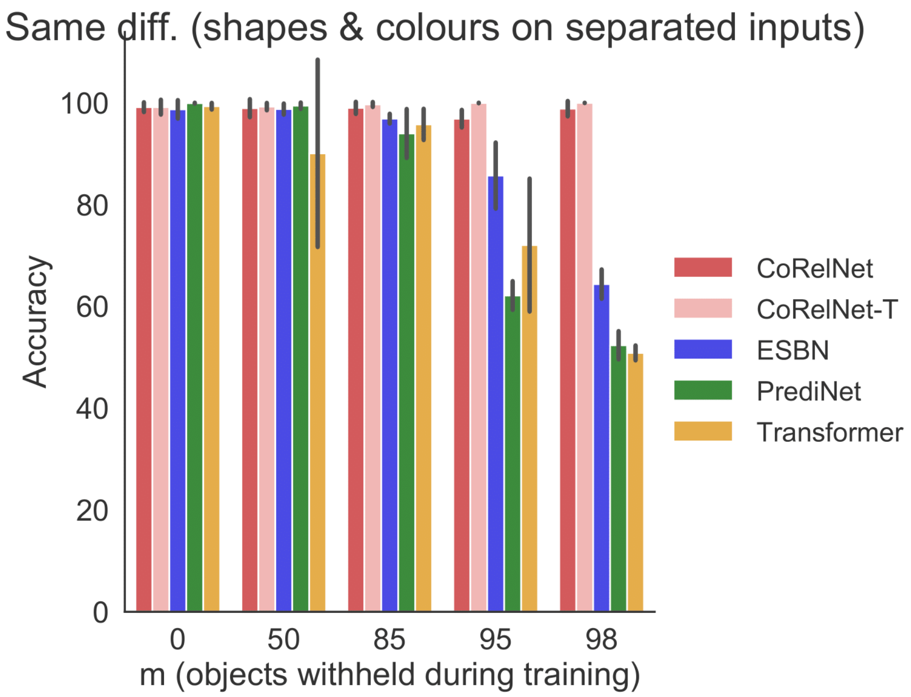


Fig. 3. Figure displaying average test performance across 10 seeds for the respective models displayed in the plot. For RMTS regularization coefficient $\lambda = 5$ was used for both models, otherwise $\lambda = 1$ was used.

can more easily discern the spurious features, as the heavy lifting for ignoring spurious features no longer needs to be done on the level of the encoder but can instead be done on the level of the relational matrix. This is where we believe our model CoRelNet has an advantage over ESBN, as it gets to see the whole similarity matrix all at once, and since spurious features are position dependent, it can decide which positions of the similarity matrix it should consistently ignore. See





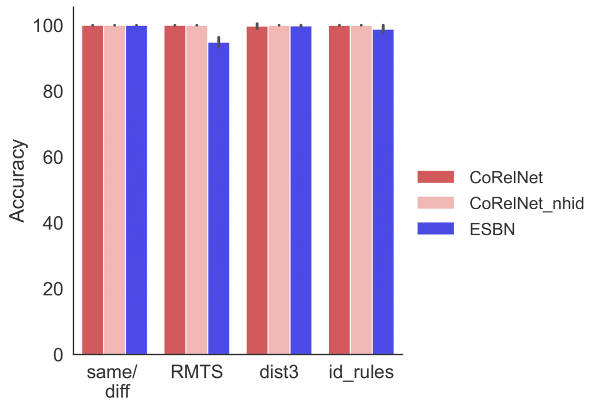
Same diff. (shapes & colours on separated inputs)
Accuracy
0
20
40
60
80
100
0
50
85
95
98
m (objects withheld during training)
CoRelNet
CoRelNet-T
ESBN
PrediNet
Transformer

Fig. 4. Left. Task illustration: example for same different task (shapes+colours) on separated inputs. Inputs are given in sequential order shapes, colour, shapes, colour (4 inputs). The task consists in determining whether the two shapes are the same or different irrespective of the colours shown in the sequence. Here the answer would be "same", since both shapes are identical. **Right.** OoD test accuracy for same different (shapes+colours) on separated inputs. Here the values on the x axis denote the holdout value m, for shapes not shown during training. There are a total of 100 shapes, and $m = 98$ here means 2 shapes are shown during training, and testing involves only the 98 unseen shapes. The set of 100 colours used in training and testing regimes are the same. The case $m = 0$ corresponds to the in-distribution case, where the same 100 shapes are shown at testing and training. The test result accuracies are averaged over 10 random seeds. We can see that for increasing values for the shape holdout parameter m the testing accuracies of ESBN and Transformer are degrading, while CoRelNet maintains close to perfect accuracy.

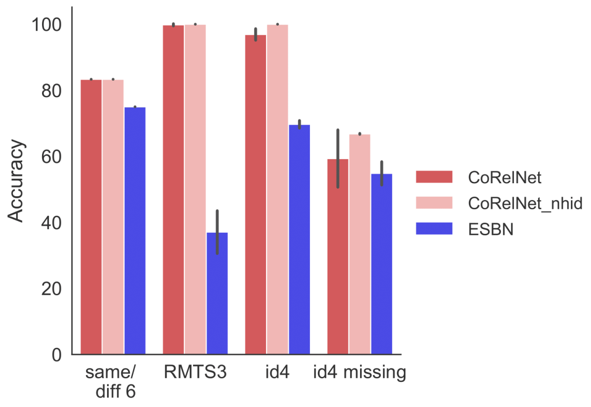
Figure 4 for details.

C.7. Ablation for matching number of parameters in CoRelNet and ESBN

In this section we present an additional ablation where we match the number of parameters of CoRelNet with the number of parameters of ESBN. We call this additional baseline CoRelNet-nhid, and we modify the number of hidden units in the MLP decoder for each task in order to match model size with ESBN. As one can infer from Table 3, CoRelNet has much fewer parameters than ESBN, while showing superior performance. In other words, CoRelNet-nhid as much more parameters than CoRelNet and thus Figure 5 shows that increasing the number of parameters in CoRelNet does not degrade performance.



(a) Basic cognitive relational tasks



(b) Harder cognitive relational tasks

Fig. 5. The additional baseline 'CoRelNet nhid' is being compared to CoRelNet and ESBN. 'CoRelNet nhid' is equivalent to CoRelNet where the hidden size of the decoder MLP is adjusted as to match the number of parameters with ESBN on each of the shown tasks. Results are showing OoD test accuracy averaged across 10 seeds for each task. We took the most extreme OOD cases such as m=98 for same/diff and same/diff6, m= 95 for RMTS, identity rules and dist3, and m=94 for identity rules 4, identity rules 4 missing, RMTS3. The number of parameters is listed in Table 3.

Table 3. Number of parameters

Task	CoRelNet	CoRelNet-nhid	ESBN
Same different	200418	1910760	1910757
RMTS	205026	1910754	1910757
Dist3	211684	1911784	1911783
Identity rules	211684	1911784	1911783
Same diff. 6	205026	1910760	1910757
RMTS 3	211170	1910754	1910757
Identity rules 4	223974	1912808	1912809

C.8. Full plots

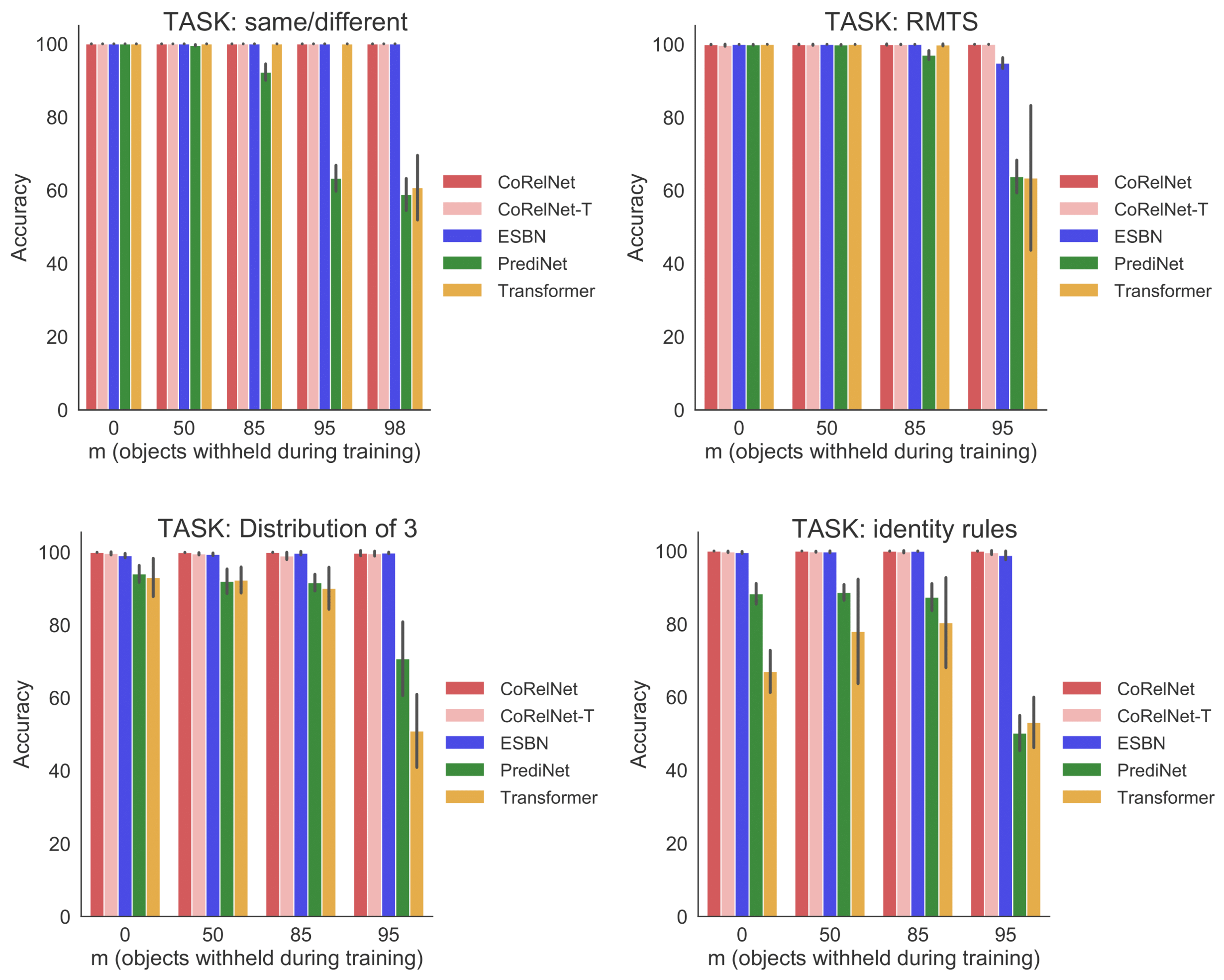


Fig. 6. Full detailed test accuracy results for the four basic relational tasks, across the full range of values for m (the number of heldout shapes during training only shown at testing, displayed on the x-axis). There are total of $n = 100$ shapes, hence $100 - m$ of those are shown during training, and the test set consists only of the other m shapes. The most extreme OoD case corresponds to $m = 98$ for the same different task (98% of the possible combinations of shapes are in the test set and not shown during training), and $m = 95$ for the 3 other tasks. The case $m = 0$ corresponds to the in-distribution case, where the same 100 shapes are shown at testing and training. The test result accuracies are averaged over 10 random seeds.

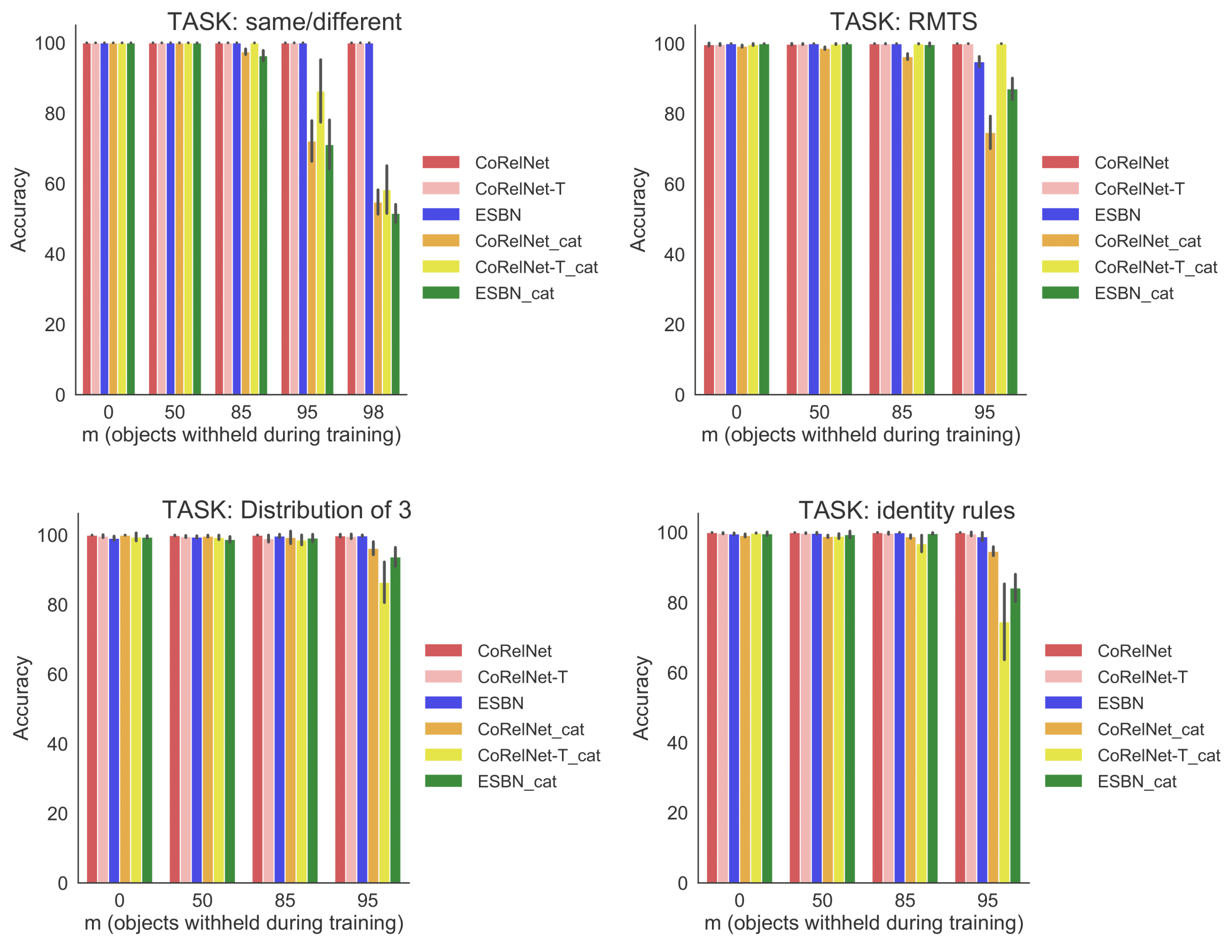


Fig. 7. Full concatenation plots. Full detailed test accuracy results on the four basic relational tasks, across the full range of values for m (the number of heldout shapes during training, displayed on the x-axis). There are total of $n = 100$ shapes, hence $100 - m$ of those are shown during training, and the test set consists only of the other m shapes. The case $m = 0$ corresponds to the in-distribution case, where the same 100 shapes are shown at testing and training. Here the suffix "cat" stands for concatenating the encoded input on top of the input of the decoder. The test result accuracies are averaged over 10 random seeds.

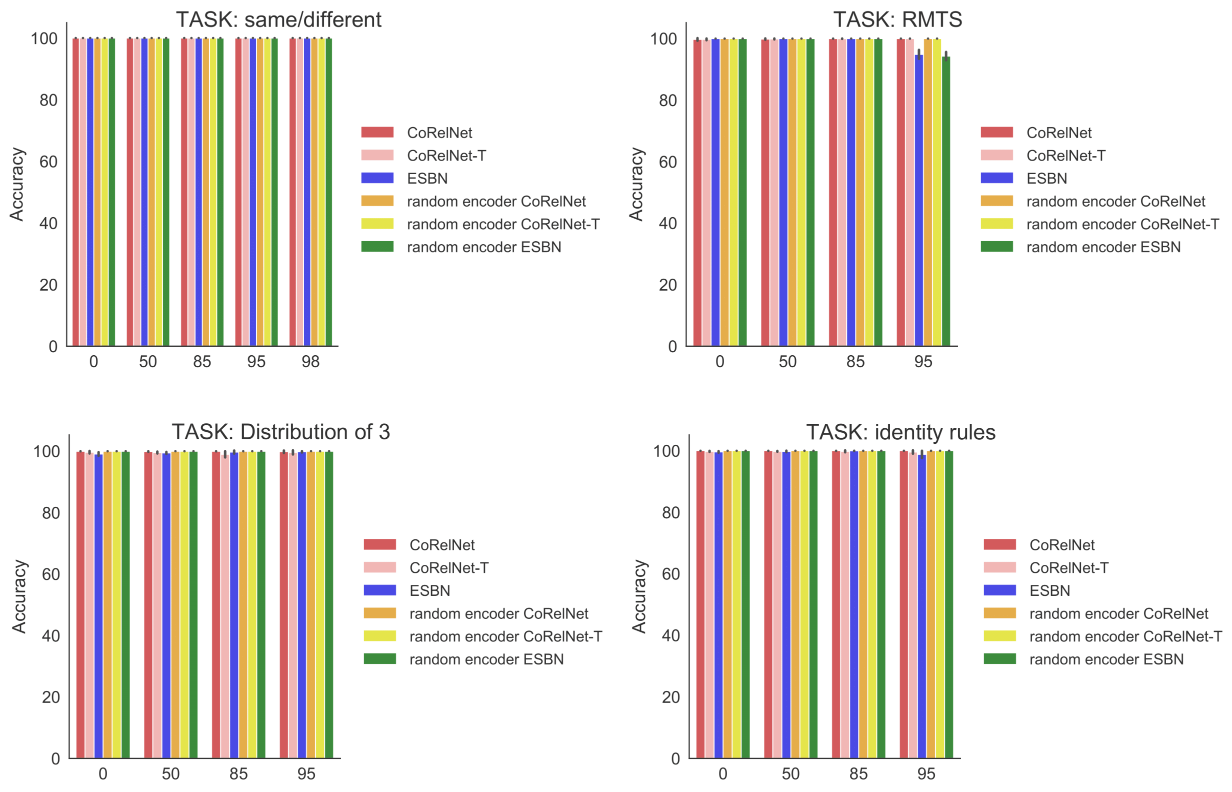


Fig. 8. Full random encoder plots. Full detailed test accuracy results on the four basic relational tasks, across the full range of values for m (the number of heldout shapes during training, displayed on the x-axis). There are total of $n = 100$ shapes, hence $100 - m$ of those are shown during training, and the test set consists only of the other m shapes. The case $m = 0$ corresponds to the in-distribution case, where the same 100 shapes are shown at testing and training. Here the prefix "random encoder" stands for randomly initializing the encoder but not updating it via backpropagation during training. The test result accuracies are averaged over 10 random seeds.

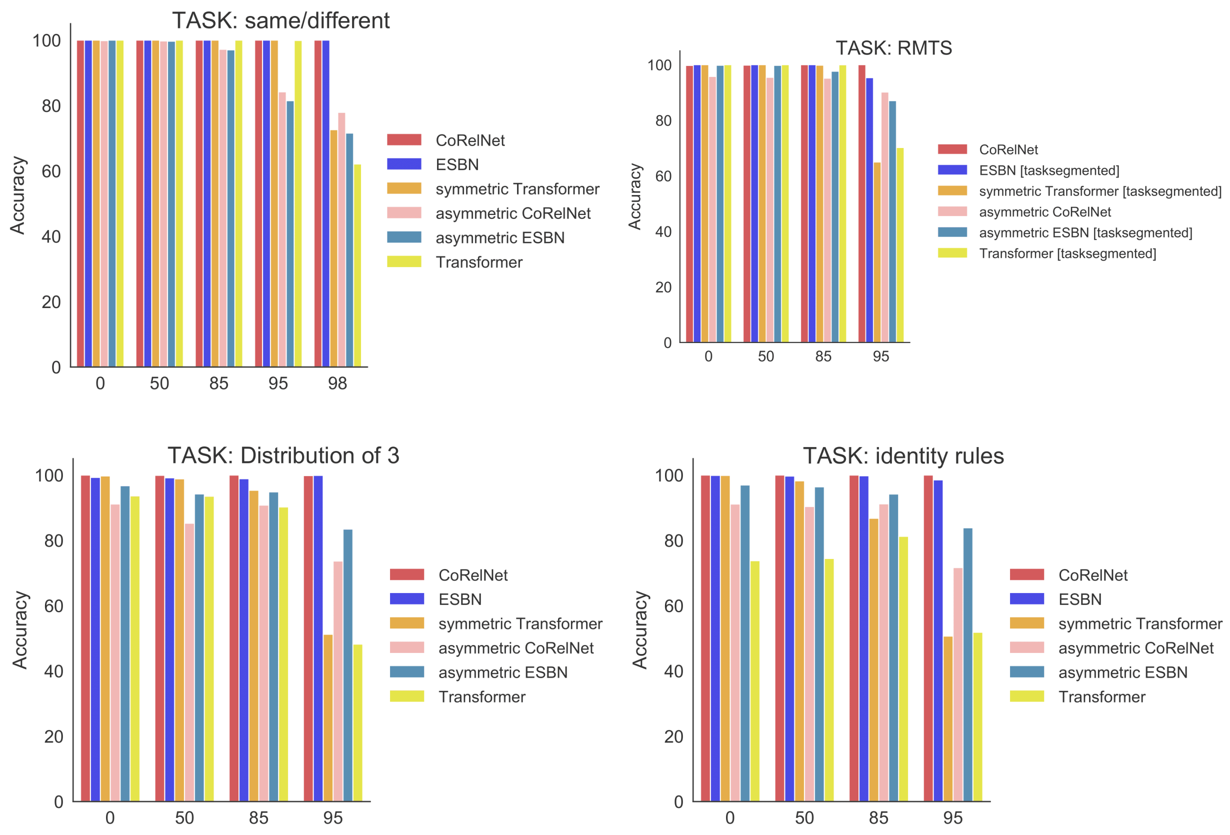


Fig. 9. Full symmetry plots. Full detailed test accuracy results on the four basic relational tasks, across the full range of values for m (the number of heldout shapes during training, displayed on the x-axis). There are total of $n = 100$ shapes, hence $100 - m$ of those are shown during training, and the test set consists only of the other m shapes. The case $m = 0$ corresponds to the in-distribution case, where the same 100 shapes are shown at testing and training. Here the prefix "asymmetric" stands for replacing the symmetric dot-product $z_t^\top z_t$ by the asymmetric $(W_1 \cdot z_t)^\top (W_2 \cdot z_t)$, whenever self-attention is performed. Similarly, the prefix "symmetric" stands for replacing the 'asymmetric' dot-product $(Q \cdot z_t)^\top (K \cdot z_t)$ by the symmetric $(Q \cdot z_t)^\top (Q \cdot z_t)$ counterpart, whenever self-attention is performed. However in our analysis in the main text for symmetric vs asymmetric, we did not include symmetric Transformer and Transformer, since they are not built upon the same inductive bias. The test result accuracies are averaged over 10 random seeds.

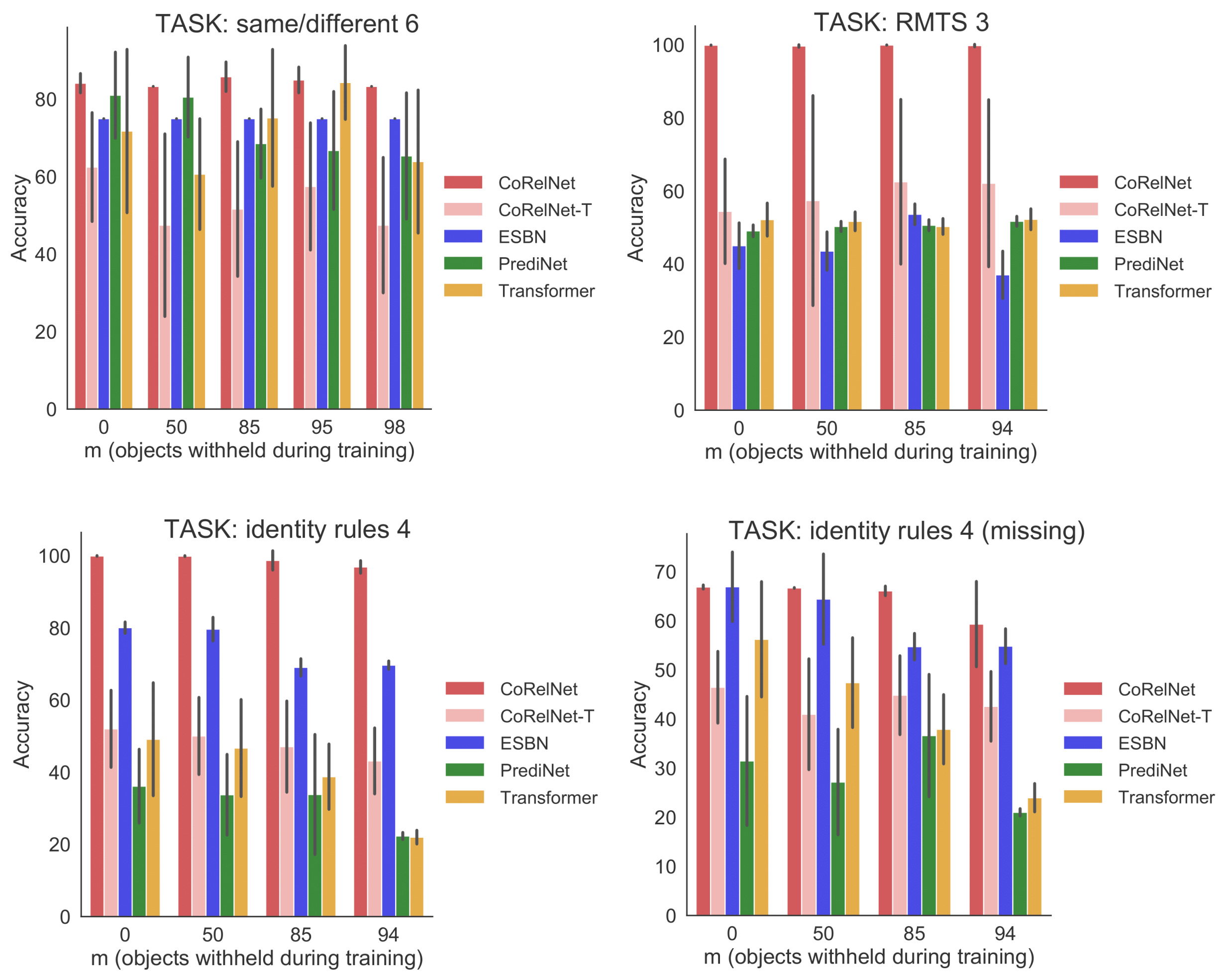
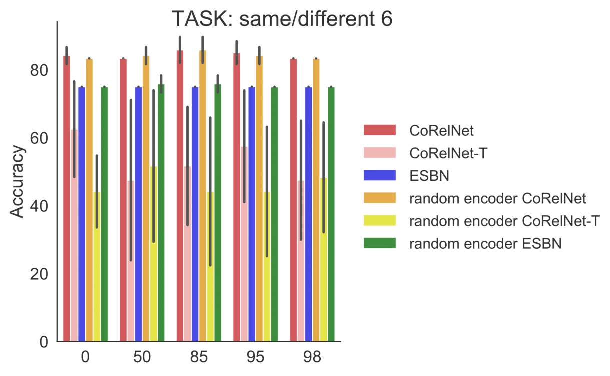
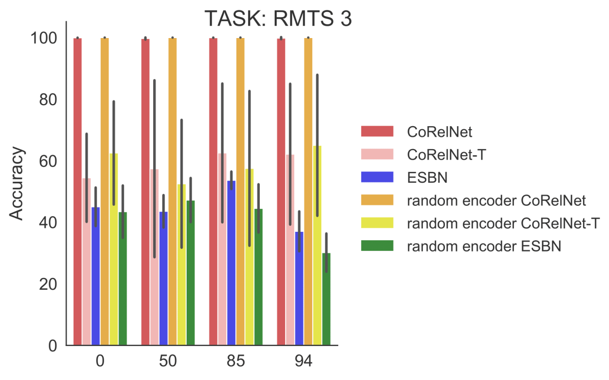


Fig. 10. Full detailed test accuracy results on the harder relational tasks with unseen relations, across the full range of values for m (the number of heldout shapes during training, displayed on the x-axis). There are total of $n = 100$ shapes, hence $100 - m$ of those are shown during training, and the test set consists only of the other m shapes. The case $m = 98$ corresponds to the most extreme OoD case for same/different 6, and $m = 94$ for the other three tasks. The case $m = 0$ corresponds to the in-distribution case, where the same 100 shapes are shown at testing and training. The test result accuracies are averaged over 10 random seeds.





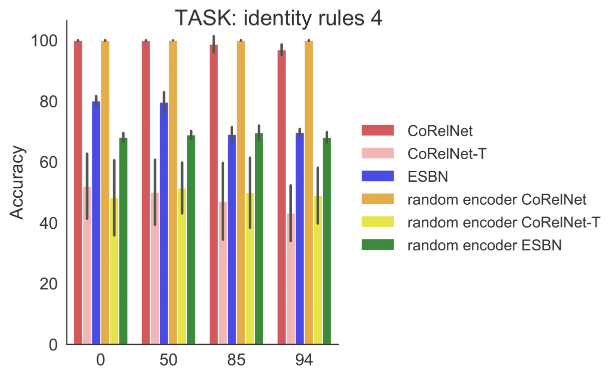


Fig. 11. Full random encoder plots. Full detailed test accuracy results on the harder relational tasks with unseen relations, across the full range of values for m (the number of heldout shapes during training, displayed on the x-axis). There are total of $n = 100$ shapes, hence $100 - m$ of those are shown during training, and the test set consists only of the other m shapes. The case $m = 98$ corresponds to the most extreme OoD case for same/different 6, and $m = 94$ for the other three tasks. The case $m = 0$ corresponds to the in-distribution case, where the same 100 shapes are shown at testing and training. Here the prefix "random encoder" stands for randomly initializing the encoder but not updating it via backpropagation during training. The test result accuracies are averaged over 10 random seeds.

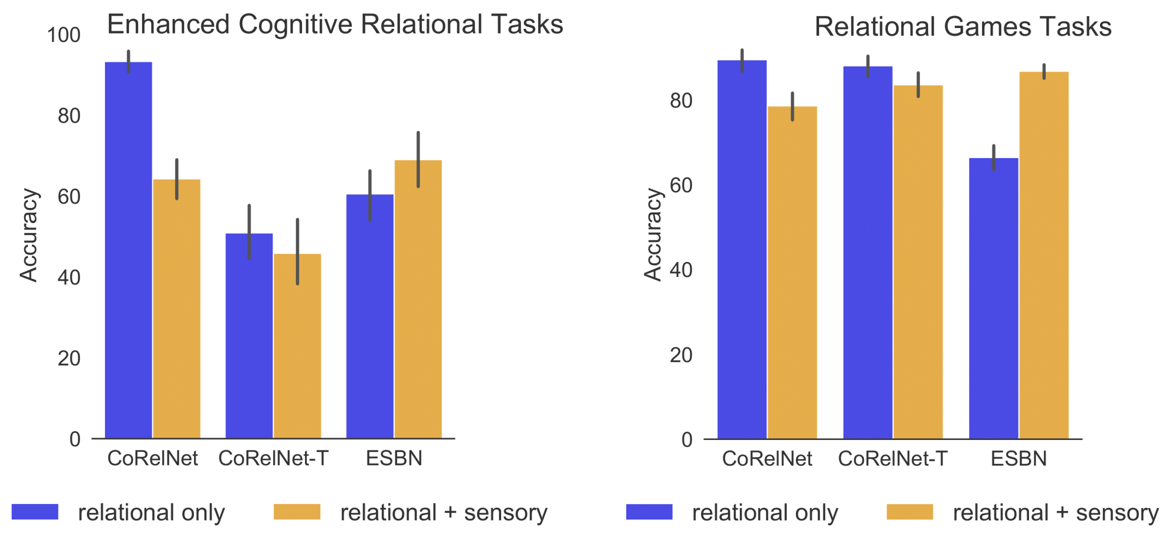


Fig. 12. Comparison relational only vs relational + sensory input models, all models are run on 10 seeds for each task. **Left.** Average OOD test performance across the three enhanced cognitive relational tasks same/diff6 ($m = 98$), RMTS3 ($m = 94$), identity rules 4 ($m = 94$). **Right.** Average OOD test performance across the three enhanced cognitive relational tasks 'same', 'between', 'occurs', 'xoccurs', 'left-of', 'row matching', 'col./shape'. For the tasks involving stripes and hexominoes for OOD testing, both test performances were considered and the mean was computed, otherwise only hexominoes performance was considered. The average OOD test was taken across all tasks.

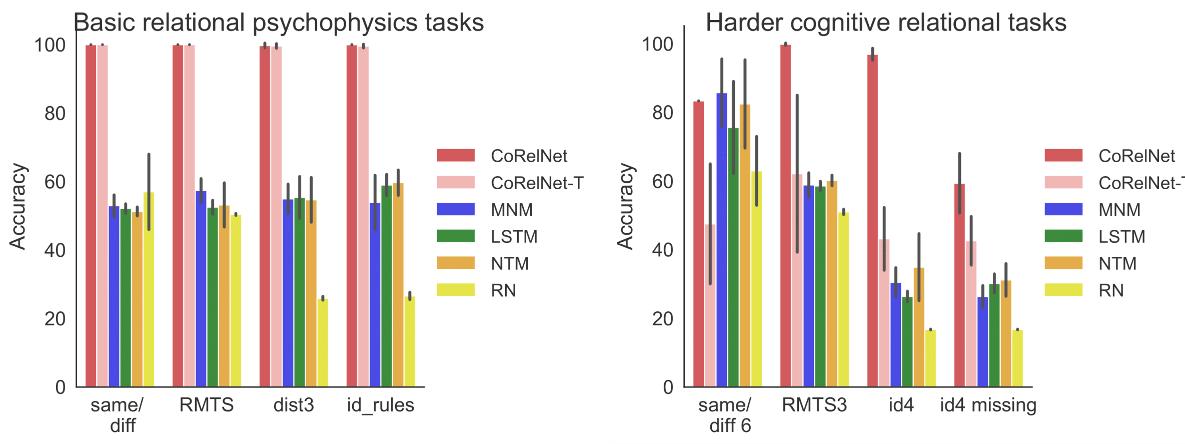


Fig. 13. Additional baselines are being shown, such as MNM, NTM, LSTM and RN. Results are showing OoD test accuracy averaged across 10 seeds for each task. We took the most extreme OOD cases such as $m = 98$ for same/diff and same/diff6, $m = 95$ for RMTS, identity rules and dist3, and $m = 94$ for identity rules 4, identity rules 4 missing, RMTS3.

Appendix D

Unitary RNNs and constraint optimization

As already outlined in subsection §2.2.3, recurrent neural networks suffer from the exploding and vanishing gradient problem. One approach to mitigate this problem has been to use gates in order to have the gradient flow more efficiently along longer time horizons and therefore better capture long-term dependencies (Hochreiter and Schmidhuber, 1997; Cho et al., 2014a). Another approach is to stick with the vanilla RNN and constrain the spectrum of the connectivity matrix either at initialization (Le et al., 2015; Henaff et al., 2016) or throughout the entire training period (Arjovsky et al., 2016; Wisdom et al., 2016; Hyland and Rätsch, 2017; Mhammedi et al., 2017; Jing et al., 2017; Vorontsov et al., 2017; Lezcano-Casado and Martínez-Rubio, 2019).

The literature review of this section is given in chronological order in which the papers appeared on arxiv for the first time.

D.1. Unitary evolution RNN

Using orthogonal or identity connectivity matrices at initialization has been discussed in Saxe et al. (2014) and Le et al. (2015) (IRNN), which then became a stepping-stone for the seminal work of Arjovsky et al. (2016), called *unitary evolution RNN* (uRNN), outlining the use of unitary connectivity matrices in vanilla RNNs throughout the entire training procedure. In Arjovsky et al. (2016), unitary weight matrices have been parametrized as a product of several structured matrices, whose parameters are learned during training. More precisely, the unitary connectivity matrix W is parametrized as follows:

$$W = D_3 R_2 \mathcal{F}^{-1} D_2 \Pi R_1 \mathcal{F} D_1$$

where the matrices D_i are diagonal unitary matrices, the matrices R_i are Householder reflection matrices, Π is a permutation matrix (all prior matrices can be computed in $O(n)$ time, if n is the number of hidden units) and $\mathcal{F}$ is a discrete Fourier transform matrix (which can be computed in $O(n \log n)$ time using the Fast Fourier Transform algorithm). Hence the whole product can be computed in $O(n \log n)$ time. In terms of memory, there are $7n$ parameters (n for each diagonal matrix, $2n$ for each Householder reflection matrix and none for the discrete Fourier transform matrices as well as the permutation matrix). Also note that all mentioned matrices are unitary, and thus the product is unitary. This parametrization was inspired by Yang et al. (2015) and Le et al. (2013).

In Arjovsky et al. (2016), uRNN was evaluated against LSTM, RNN with tanh activation, and IRNN on a handful of tasks especially created to test the model's ability to capture long-term dependencies such as: the copying memory problem, the adding problem as well as pixel-by-pixel MNIST (permuted and non permuted).

Most striking was the copying memory problem, where even when having to recall sequences for more than 500 time steps, uRNN achieves almost perfect performance relatively quickly, while the other models get stuck at the baseline performance of the memoryless strategy.

Another interesting find is that for the non-permuted version of pixel-by-pixel MNIST, the LSTM performs better than uRNN, while for the permuted version, uRNN outperforms LSTM suggesting that the LSTM is better suited to learn local dependencies, while uRNN seems to deal better with complicated dependencies across varying and potentially much longer time scales.

Further, an exploratory experiment where norms of gradients with respect to hidden states were tracked across time (after 100 iterations of training on the adding problem) suggests that the LSTM experiences vanishing gradient much more rapidly than the uRNN.

Finally, the uRNN doesn't suffer from the "hidden state saturation" phenomenon nearly as badly as the LSTM. The "hidden state saturation" phenomenon is a behaviour exhibited in recurrent architectures where the hidden states' flexibility to change and use new information diminishes with sequence length.

The main drawback of this approach is that the parametrization in Arjovsky et al. (2016) does not have full capacity over the full unitary matrix group whenever $n \geq 8$, as pointed out in Wisdom et al. (2016). This problem will be addressed in §D.3

D.2. Henaff initialization

Inspired by the works of Saxe et al. (2014) and Le et al. (2015), Henaff et al. (2016) constructed explicit solutions to the copying memory and adding tasks, using a modified RNN architecture with update equations

$$\begin{aligned} h_t &= \sigma\left(Ux_t + b\right) + Vh_{t-1} \\ y_t &= Wh_t \end{aligned}$$

specifically designed to solve simple long-term memory tasks.

For the copying memory task, an orthogonally initialized $(2d+1) \times (2d+1)$ block matrix is being used as the connectivity matrix

$$Q = \begin{pmatrix} Q_1 & 0 & \cdots & 0 \\ 0 & Q_2 & \cdots & 0 \\ \vdots & & \ddots & 0 \\ 0 & & 0 & Q_d \end{pmatrix}, V = \begin{pmatrix} Q & 0 \\ 0 & 1 \end{pmatrix}$$

consisting of rotation blocks

$$Q_j = \begin{pmatrix} \cos\left(2l_j\pi/(T+S)\right) & \sin\left(2l_j\pi/(T+S)\right) \\ -\sin\left(2l_j\pi/(T+S)\right) & \cos\left(2l_j\pi/(T+S)\right) \end{pmatrix}$$

where l_j's are uniformly sampled from $\{1, \dots, T+S\}$, S is the length of the sequence to be remembered, T the length of time to remember, and d is a fixed positive integer. Note that $V^{T+S} =$ Id, so we can that each Q_j can be considered as a "rotating clock" rotating at an individual speed while aligning with all other "rotating clocks" after $T+S$ time steps. This allows different symbols to surface from the hidden state in the order in which they were seen, while the other symbols remain hidden.

For the adding task, the one-dimensional solution $V = 1$ is being used, while a more redundant higher dimensional solution using V to be an identity matrix could be used as well. Note that this corresponds to case where $l_j = 0$ for all j in the above block matrix Q. Since having l_j's uniformly distributed is an important feature for the solution of the copying memory problem, one can view the two solutions as opposites. It turns out that it is hard to to learn the adding task, when given the initialization from the memory task and vice-versa.

D.3. Full-Capacity Unitary RNNs and subsequent literature

As already pointed out in §D.1, uRNN does not have full capacity. In fact Wisdom et al. (2016) provides necessary condition for a family of parametrized $n \times n$ unitary matrices to satisfy in order to contain all $n \times n$ unitary matrices, using Sard's theorem (Sard, 1942), and then leverages the theory of Riemannian gradient descent in order to show how to directly optimize a full-capacity unitary matrix along the Stiefel manifold (i.e. the manifold consisting of all $n \times n$ unitary matrices).

More precisely, if we denote $U(n)$ the Lie group of $n \times n$ unitary matrices, and making use that the fact that the dimension of a Lie group is equal to the dimension of the Lie algebra, we have that

$$\dim_{\mathbb{R}} U(n) = \dim_{\mathbb{R}} \mathfrak{u}(n) = n^2$$

where $\mathfrak{u}(n)$ is the Lie algebra of $U(n)$ consisting of all skew-Hermitian matrices. Then if Γ is a subset of $U(n)$ with $\dim_{\mathbb{R}}\Gamma = k$, then as a consequence Sard's theorem, Γ cannot contain all of $U(n)$ if $k < n^2$. In the parametrization of Arjovsky et al. (2016), we have $k \leq 7n$, thus failing to contain $U(n)$ for all $n > 7$. More generally, this proves that we cannot explicitly parametrize a full-capacity unitary matrix using only $O(n)$ parameters. For an overview to Lie groups and Lie algebras, see Appendix §E.

In order to directly optimize a full-capacity unitary matrix, Wisdom et al. (2016) considers the Stiefel manifold as a Riemannian manifold where for each unitary matrix W its tangent space T_W is equipped with the inner product defined by

$$\langle Z_1, Z_2 \rangle_W = \mathrm{Tr}(Z_1^*(\mathrm{Id} - \frac{1}{2}WW^*)Z_2)$$

as suggested by Tagare (2011), such that the gradient in the Stiefel manifold of some loss function L with respect to W becomes $A_W \cdot W$, where $A_W = (\nabla_W L)^* W - W^* \nabla_W L$ is skew-Hermitian, and uses the multiplicative update rule at training iteration k,

$$W^{(k+1)} = \mathrm{Cay}\left(\frac{\lambda}{2} A_{W^{(k)}}\right) \cdot W^{(k)}$$

where λ is the learning rate and $\mathrm{Cay}(A) = (\mathrm{Id} + A)^{-1}(\mathrm{Id} - A)$ is the Cayley transformation of A. Note that since $\frac{\lambda}{2} A_{W^{(k)}}$ is skew-Hermitian for each k, $\mathrm{Cay}\left(\frac{\lambda}{2} A_{W^{(k)}}\right)$ is unitary, and thus if $W^{(0)}$ is initialized as a unitary matrix, $W^{(k)}$ will be unitary for all k. Let us keep in mind that, even if $W^{(0)}$ might be unitary up to machine precision, the rounding errors are compounded by the repeated matrix multiplications from multiplicative update rules. Nonetheless, this approach clearly outperforms uRNN on the copy task for longer sequence lengths such as T=1000. In the literature, one refers to Wisdom et al. (2016)'s model as *full-capacity uRNN*, and to Arjovsky et al. (2016)'s model as *restricted-capacity uRNN*.

Around the same time, another interesting and somewhat similar approach has been suggested in Hyland and Rätsch (2017) which parametrizes $U(n)$ directly in the corresponding Lie algebra $\mathfrak{u}(n)$ of skew- Hermitian matrices via the exponential matrix map

$$\exp : \mathfrak{u}(n) \to U(n) : L \mapsto \sum_{j=0}^{\infty} \frac{L^j}{j!}$$

which is surjective in the case of a compact and connected Lie group $U(n)$. Since $\mathfrak{u}(n)$ can be viewed a real vector space of real dimension equal to n^2, one can find a basis $\{T_j\}_{j=1}^{n^2}$, and express every matrix $W \in U(n)$ as

$$W = \exp\left(\sum_{j=1}^{n^2} \nu_j T_j\right)$$

where ν_j are parameters to be learned. Even though, the idea of using gradient descent in vector space sounds appealing, experimentally this approach doesn't provide groundbreaking results and struggles to efficiently compute the matrix exponential.

Even though both methods Wisdom et al. (2016) and Hyland and Rätsch (2017) provide full-capacity parametrizations, it comes at the expense of increased computational cost $O(n^3)$: for the inverse in the Cayley transformation in the case of Wisdom et al. (2016) and for the matrix exponential in Hyland and Rätsch (2017).

Meanwhile Mhammedi et al. (2017) and Jing et al. (2017) both propose parametrizations allowing efficient training of orthogonal/unitary RNNs, using products of Householder reflection matrices (oRNN) and Givens rotation matrices (EURNN) respectively. Here the user can decide how much of the matrix space should be covered by the parametrization, by tuning the number m of matrices involved in the parametrization product, leading to computational complexity $O(mn)$ if n is the number of hidden units. Jing et al. (2017) shows that for some tasks such as the copy task, a small subspace of unitary space is sufficient, while other tasks such as permuted MNIST or the TIMIT speech prediction dataset, only perform well when a larger subspace is covered.

Vorontsov et al. (2017) builds on the approach of Wisdom et al. (2016) using Riemannian gradient descent along the Stiefel manifold, but for real valued martices only, while allowing matrices to step away from the Stiefel manifold using hard and soft constraints.

The soft constraint simply consists in adding the regularization term $\lambda\|W^T W - \mathrm{Id}\|^2$, where $\|.\|$ is the spectral norm, and λ a hyperparameter. For the hard constraint, one parametrizes the connectivity matrix W using the singular value decomposition

$$W = U\Sigma V^T$$

where U and V are optimized via the Riemannian gradient descent algorithm mentioned in Wisdom et al. (2016). The singular values s_i are parametrized with a parameter p_i, and a hyperparameter $m \in [0,1]$ such that

$$s_i = 2m(\sigma(p_i) - 0.5) + 1$$

where p_i is updated freely via stochastic gradient descent. We observe that $s_i \in [1-m, 1+m]$, where $m = 0$ corresponds exactly to W being orthogonal since orthogonal matrices have all their singular values equal to one. Let us further keep in mind that we are performing Cayley transforms in the Riemannian gradient descent algorithms for U and V, thus requiring $O(n^3)$ computational complexity with n being the number of hidden units.

For tasks requiring more long term memory (copy, adding, permuted MNIST), increasing the margin improves convergence rate and performance, while too large margins might lead to instability and worse performance. Even if permitted a larger margin, singular values tend to stay close to one once orthogonally initialized. For PTB, singular values have more of a tendency to drift away. Here, for longer sentences it is more beneficial to use smaller margins, while shorter sentences perform better with larger margins, because they allow more expressivity.

Then the work of Helfrich et al. (2018) introduced another real-valued full-capacity orthogonal matrix parametrization using a scaled version of the Cayley transform. The $n \times n$ connectivity matrix then rewrites as

$$W = \mathrm{Cay}(A)D$$

where A is a $n \times n$ skew-symmetric matrix and D a $n \times n$ diagonal matrix with ± 1 entries, ρ of which are equal to -1 (ρ being a hyperparameter). The intention to scale with D is due the identity

$$\mathrm{Id} - \mathrm{Cay}(A) = (\mathrm{Id} + \mathrm{Cay}(A))A$$

suggesting that $\mathrm{Id} + \mathrm{Cay}(A)$ needs to be invertible (or equivalently W cannot have -1 eigenvalues) for W to have a unique pre-image via the Cayley transform. In particular, having $\mathrm{Cay}(A)$ with eigenvalues very close to -1, requires pretty large entries in A, and thus gradient descent would slow down significantly when learning A. Finally Kahan (2006) showed that every orthogonal matrix W can be written $\mathrm{Cay}(A)D$, with bounded entries $|A_{ij}| \leq 1$ and by finding a suitable matrix D. In other words, instead of hoping that $\mathrm{Id} + W$ is invertible, we now tune ρ hoping that $D + W$ is invertible (locally around the matrix we are attemping to learn) and that the corresponding A doesn't have exploding entries.

Unlike the multiplicative update rule used in Wisdom et al. (2016) and Vorontsov et al. (2017), Helfrich et al. (2018) uses an additive update rule at each iteration k,

$$A^{(k+1)} = A^{(k)} - \lambda \tilde{\nabla}_{A^{(k)}} L$$
$$W^{(k+1)} = \mathrm{Cay}(A^{(k+1)})D$$

where $\tilde{\nabla}_{A^{(k)}} L = \left(V^{(k)}\right)^T - V^{(k)}$ and $V^{(k)} = (\mathrm{Id} + A^{(k)})^{-T} \cdot \nabla_{W^{(k)}} L \cdot \left(D + \left(W^{(k)}\right)^T\right)$. Here, $\nabla_{W^{(k)}} L$ being computed first using standard backpropagation, followed by the additive update rule and the scaled Cayley transform. Inspired by the Henaff initialization (Henaff et al., 2016), they also introduce a new initialization scheme for the $n \times n$ skew-symmetric matrix A, which is referred to as *the Cayley initialization*:

$$A = \begin{bmatrix} B_1 & & \\ & \ddots & \\ & & B_{\lfloor n/2 \rfloor} \end{bmatrix} \quad \text{where} \quad B_j = \begin{bmatrix} 0 & s_j \\ -s_j & 0 \end{bmatrix}$$

with $s_j = \sqrt{\frac{1-\cos(t_j)}{1+\cos(t_j)}}$ and t_j is uniformly sampled from $\left[0, \frac{\pi}{2}\right]$, which ensures that the eigenvalues of $\text{Cay}(A)$ are distributed uniformly on the right unit half-circle, ρ of which will be reflected across the imaginary axis to get the spectrum of $\text{Cay}(A)D$.

This method uses a simpler implementation scheme than previous methods while still having full-capacity orthogonal connectivity matrices, and making use of an additive update rule thus avoiding the compounded roundoff errors that would otherwise come in with a multiplicative update rule. It is reported to have smoother and more stable convergence than previous models while being among the top performers in all tests.

In the follow-up paper Maduranga et al. (2019) it is argued that having to tune ρ individually for every task might critically affect performance, and that it might be better to work in a setting where D would be learnable. They are proposing the complex analogue of the scaled Cayley transform, parametrizing unitary matrices via skew-Hermitian and having the entries of the diagonal matrix D lie on the complex circle thus of the form $e^{i\theta}$, which is differentiable with respect to θ and learnable via gradient descent. However, by doing so, Lezcano-Casado and Martínez-Rubio (2019) argues that the parametrization is not locally unique anymore and therefore can create spurious minima. The method proposed by Lezcano-Casado and Martínez-Rubio (2019) is called (expRNN) will be discussed in the following subsection §D.4.

D.4. expRNN

In the same spirit as Hyland and Rätsch (2017), Lezcano-Casado and Martínez-Rubio (2019) propose to parametrize the special orthogonal matrix group $SO(n)$ (or the unitary matrix group $U(n)$ for the complex case) via the exponential matrix map, with the update equation:

$$h_{t+1} = \sigma\left(\exp(A)h_t + Tx_{t+1}\right)$$

where $A \in \text{Skew}(p)$, $T \in \mathbb{R}^{p\times d}$ and σ is the modReLU activation as used in Arjovsky et al. (2016). Beside the theoretical analysis of the proposed method, the main difference with Hyland and Rätsch (2017) lies in the efficient approximation of the exponential matrix map and in the gradient calculations.

Exponential matrix map approximation. The exponential matrix map is approximated up to machine precision by combining the scaling-square trick with the Padé approximation as shown in

Al-Mohy and Higham (2009): the scaling-square trick makes use of the identity

$$e^A = (e^{A/2^k})^{2^k}$$

(where k is the smallest positive integer such that $\|A\|/2^k \leq \frac{1}{2}$) approximates $e^{A/2^k}$ using the Padé approximants. Padé approximants are approximations of the form

$$\exp(A) \approx p_n(A)q_n(A)^{-1}$$

where p_n and q_n are polynomials of degree n with rational coefficients. For instance, we get the Cayley transform for $n = 1$

Gradient calculation. For gradient calculation one uses the formula:

$$\nabla(f \circ \exp)(A) = B(\mathrm{d}\exp)_{-A}\left(\frac{1}{2}\left(\nabla f(B)^\top B - B^\top \nabla f(B)\right)\right)$$

with $B = exp(A)$, $A \in \mathrm{Skew}(n)$ and $f : \mathbb{R}^{n\times n} \to \mathbb{R}$ any differentiable function.

- B is calculated again via the Padé approximation and the scaling trick as in Al-Mohy and Higham (2009).
- the stochastic gradient $\nabla f(B)$ is calculated via auto-differentiation.
- $(\mathrm{d}\exp)_{-A}$ is approximated to machine-precision by calculating the Fréchet derivative of the expression $e^A = (e^{A/2^k})^{2^k}$ recursively, while using again the Padé approximation and the scaling trick for all the exponential maps emerging in the recursion as shown in Al-Mohy and Higham (2008). This algorithm is about three times as expensive as calculating e^A in the above fashion.

Theoretical analysis. First it is shown that for every connected and compact Lie group G, the (constraint) minimization problem

$$\min_{B\in G} f(B)$$

is equivalent to (unconstraint) minimization problem

$$\min_{A\in\mathfrak{g}} f(\exp(A))$$

where $\mathfrak{g}$ is the Lie algebra of G. In fact, it is shown that the exponential map is surjective whenever G is connected and compact. Thus if $\hat{A} \in \mathfrak{g}$ is minimizer of the second problem, then $\exp\left(\hat{A}\right) \in G$ must be a minimizer of the first problem. Conversely, if $\hat{B} \in G$ is minimizer for the first problem, there exists $\hat{A}' \in \mathfrak{g}$ which is a minimizer for the second problem and verifies $\exp \hat{A}' = \hat{B}$. Finally note that $SO(n)$ and $U(n)$ are both compact and connected Lie groups, with the vector spaces of skew-symmetric $\mathfrak{so}(n)$ and skew-Hermitian matrices $\mathfrak{u}(n)$ as respective Lie algebras, both of which are isomorphic to a Euclidean vector space. For an introduction to Lie Groups and Lie algebras see appendix §E

Then it is argued that since $\exp$ is bi-analytical on an open neighbourhood around the origin, it cannot create spurious minima because it is an immersion, and it cannot create saddle points, because it is a diffeomorpshim. Hence, we have desirable properties to use gradient descent for the optimization problem.

Experiments. expRNN has been tested against LSTM (Hochreiter and Schmidhuber, 1997), the restricted-capacity uRNN (Arjovsky et al., 2016), the full-capacity uRNN (Wisdom et al., 2016), EURNN (Jing et al., 2017), and the scaled Cayley transform orthogonal RNN "scoRNN" (Helfrich et al., 2018), on the copy task, permuted and ordered pixel-by-pixel MNIST, as well as the TIMIT speech dataset.

expRNN outperforms all other models across the board, and is the only architecture to fully converge to the correct answer for the copy task with length $T = 2000$. For both ordered and permuted pixel-by-pixel MNIST tasks, expRNN outperforms scoRNN only slightly and it is further conjectured that using the parametrization of expRNN with an LSTM or GRU would result in even better performance. For the TIMIT dataset, expRNN outperforms all other models by a large margin.

Appendix E

A primer on Lie groups and Lie algebras

E.1. Lie groups

Definition A *Lie group* is a finite dimensional smooth manifold G together with a group structure on G, such that the multiplication $G \times G \to G$ and the attaching of an inverse $g \mapsto g^{-1} : G \to G$ are smooth maps.

A *morphism between two Lie groups* G and H is a map $f : G \to H$, which at the same time is smooth and a group homomorphism. An isomorphism is a bijective map f such that f and f^{-1} are morphisms. A Lie subgroup is a subgroup H in G such that H is also a smooth submanifold of G.

Cartan's Theorem. Any closed subgroup H of a Lie group G, is again a Lie group (in particular, H is an analytic submanifold of G, with the induced analytic structure).

Heine-Borel Theorem. A subset K of $\mathbb{R}^n$ / $\mathbb{C}^n$ (for $n \geq 1$) is compact if and only if K is closed and bounded.

Path-connectedness. A path-connected topological space is also connected.

Proof. Assume X is a path-connected topological space that is not connected, then there exists two disjoint subsets $A, B \subset X$ such that $X = A \cup B$. Now choose $a \in A$ and $b \in B$, and since X is path-connected, there exists a path $\gamma : [0,1] \to X$ in X such that $\gamma(0) = a$ and $\gamma(1) = b$, and thus $[0,1] = \gamma^{-1}(A) \cup \gamma^{-1}(B)$ is not connected. Contradiction.

Compact sets. Every closed subset F of a compact set K, is also compact. Thus every closed subgroup of a compact Lie group is also compact Lie group.

Notation

- Let $t(n)$ be the subgroup of Mat(n,$\mathbb{C}$) that consists of only upper-triangular matrices with all elements on the diagonal having complex norm 1.
- Let $e(n)$ be the maximal subgroup of $t(n)$ consisting only of diagonal matrices (i.e. diagonal matrices with all elements on the diagonal having complex norm 1)

- Let $U(n)$ be the subgroup of Mat(n,$\mathbb{C}$) consisting of unitary matrices (thus $e(n)$ is a subgroup of $U(n)$)
- $O(n)$ be the subgroup of Mat(n,$\mathbb{R}$) consisting of orthogonal matrices.
- $SO(n)$ be the subgroup of $O(n)$ consisting of matrices with determinant 1.
- For a Lie group G, we denote Lie(G) the Lie algebra of G.

Proposition $U(n)$ is a connected and compact Lie Group.

Proof. The map det : $U(n) \to U(1)$ is continuous, and $U(1)$ is a closed subset of $\mathbb{C}$, thus $U(n)$ must be closed seen as the pre-image of a closed set via a continuous map. Since $U(n)$ is a subgroup of the Lie Group Mat(n,$\mathbb{C}$), it is a closed subgroup, and thus by Cartan's theorem $U(n)$ is also a Lie Group.

Since $U(n)$ is also a bounded subset of Mat(n,$\mathbb{C}$), we conclude that $U(n)$ is compact subset by Heine-Borel Theorem.

It can easily been seen that every $P \in U(n)$ is diagonalizable and thus giving rise to a path from the identity to P, proving that $U(n)$ is path-connected and therefore also connected.

Proposition $e(n)$ is a connected and compact Lie Group and subgroup of $U(n)$.

Proof. The map det : $e(n) \to U(1)$ is continuous, and since $U(1)$ is closed, $e(n)$ must be closed as well. $e(n)$ is a closed subgroup of the compact Lie group $U(n)$, and thus is also a compact Lie group. It is obvious to see that $e(n)$ is path-connected and thus also connected.

Proposition $t(n)$ is a connected and locally compact Lie Group.

Proof. Since every element of $t(n)$ can be path-connected to the identity, we have that $t(n)$ is a connected group. The map det : $e(n) \to U(1)$ is continuous, and since $U(1)$ is closed, $e(n)$ must be closed as well, and thus $t(n)$ must be a Lie group by Cartan's theorem.

Let $\xi_n : t(n) \to \mathbb{R}$ be the map computing the sum of the complex norms of each element of a matrix in $t(n)$. We know that ξ_n is continuous. For each $P \in t(n)$, construct an small open ball B_P around $\xi_n(P)$, then $U_P = \xi_n^{-1}(B_P)$ is an open neighbourhood of P, and $\xi_n^{-1}(\overline{B_P}) = \overline{U_P}$ is an closed set containing the open neighbourhood U_P. It is also obvious to prove that $\overline{U_P}$ is bounded and thus by Heine-Borel is compact neighbourhood of P. This shows that $t(n)$ is locally compact.

E.2. Lie algebras

Intuitively, one can think of Lie algebras at tangent planes of a Lie Group at the identity.

Definition A *Lie algebra* is a vector space $\mathfrak{g}$ over some field $\mathbb{F}$ together with a binary operation $[\cdot,\cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ called the *Lie bracket* that satisfies bilinearity, alternativity and the Jacobi identity.

- *Bilinearity*:

$$[ax + by,z] = a[x,z] + b[y,z],$$

$$[z,ax+by] = a[z,x] + b[z,y]$$

for all scalars $a, b \in \mathbb{F}$ and all elements $x, y, z \in \mathfrak{g}$.

- *Alternativity*:

$$[x,x] = 0$$

for all $x \in \mathfrak{g}$.

- *The Jacobi identity*:

$$[x,[y,z]] + [z,[x,y]] + [y,[z,x]] = 0$$

for all $x, y, z \in \mathfrak{g}$.

Remark Note that using bilinearity and alternativity to expand the Lie bracket $[x+y,x+y]$ shows the *anticommutativity* $[x,y] = -[y,x]$, for all elements $x, y \in \mathfrak{g}$

Remark Any matrix Lie group G defines an associated real Lie algebra $\mathfrak{g} = \text{Lie}(G)$, which can be computed as $\mathfrak{g} = \{X \in \text{Mat}(n,\mathbb{C}) \mid (\forall t \in \mathbb{R})(\exp(tX) \in G)\}$.

The Lie bracket of $\mathfrak{g}$ is given by the commutator of matrices $[X,Y] = XY - YX$.

Remark In order to practically find the Lie algebras of matrix Lie group G, a common technique is to construct a paths of matrices $A(t) \in G$ (for $t \in \mathbb{R}$) such that $A(0) = I$, and compute $\dot{A}(0)$. Then we know that $\dot{A}(0)$ is in the Lie algebra of G. If we find a way to parametrize all matrices in G via such paths, then we can find the full Lie algebra.

Proposition The Lie algebra of $e(n)$ consists of all diagonal matrices in Mat$(n,\mathbb{C})$ such that the diagonal elements have zero real part.

Proof. Every matrix $\Lambda \in e(n)$ can be written as diag$(e^{i\theta_1}, \ldots, e^{i\theta_n})$, for some well-chosen $\theta_1, \ldots, \theta_n \in \mathbb{R}$. Thus the path $\Lambda(t) = \text{diag}(e^{it\theta_1}, \ldots, e^{it\theta_n}) \in G$ verifies $\Lambda(0) = I$, and thus $\dot{\Lambda}(0) = \text{diag}(i\theta_1, \ldots, i\theta_n)$ is in the Lie algebra of $e(n)$. Since this path construction can be done for every matrix $\Lambda \in e(n)$, we have a full description of the Lie algebra of $e(n)$.

Proposition The Lie algebra of $t(n)$ consists of all upper-triangular matrices in Mat$(n,\mathbb{C})$ such that the diagonal elements have zero real part.

Proof. Every matrix $M \in t(n)$ can be written as diag$(e^{i\theta_1}, \ldots, e^{i\theta_n}) + T$, for some well-chosen $\theta_1, \ldots, \theta_n \in \mathbb{R}$ and where T is some upper-triangular matrix in Mat$(n,\mathbb{C})$ with zero diagonal elements. Thus the path $M(t) = \text{diag}(e^{it\theta_1}, \ldots, e^{it\theta_n}) + t \cdot T$ is entirely in G and verifies $M(0) = I$. Hence $\dot{M}(0) = \text{diag}(i\theta_1, \ldots, i\theta_n) + T$ is in the Lie algebra of $t(n)$. Since this path construction can be done for every matrix $M \in t(n)$, we have a full description of the Lie algebra of $t(n)$.

Proposition Let G be a Lie Group that is a subgroup of Mat$(n,\mathbb{C})$, then we have

$$\text{Lie}(P \cdot G \cdot P^*) = P \cdot \text{Lie}(G) \cdot P^*$$

for any $P \in U(n)$.

Proof. Follows from the fact that $P \cdot \exp(A) \cdot P^* = \exp(PAP^*)$ for any $A \in$ Mat$(n,\mathbb{C})$

Corollary Lie$(U(n))$ is the set of $n \times n$ skew-Hermitian matrices (oftentimes denoted as $\mathfrak{u}(n)$).

Proof. Using the Schur decomposition one can see that skew-Hermitian matrices are precisely the matrices in Mat$(n,\mathbb{C})$ with all eigenvalues having zero real part. The rest follows from the fact that $U(n) = \{P \cdot e(n) \cdot P^* | P \in U(n)\}$.

Remark Similarly one can prove that Lie$(SO(n))$ is the set of $n \times n$ skew-symmetric matrices (oftentimes denoted $\mathfrak{so}(n)$)

Proposition Let $P \in U(n)$. If G is a connected/compact/locally compact Lie Group, then $P \cdot G \cdot P^*$ is a connected/compact/locally compact Lie Group.

Proof. Follows from the fact that the map $\Psi_P : G \to P \cdot G \cdot P^* : A \mapsto PAP^*$ is smooth group homomorphism.