Université de Montréal

Deep Learning and Reinforcement Learning Methods for Grounded Goal-Oriented Dialogue

par Harm de Vries

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

Thèse présentée à la Faculté des arts et des sciences
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

March, 2020

Résumé

Les systèmes de dialogues sont à même de révolutionner l'interaction entre l'homme et la machine. Pour autant, les efforts pour concevoir des agents conversationnels se sont souvent révélés infructueux, et ceux, malgré les dernières avancées en apprentissage profond et par renforcement. Les systèmes de dialogue palissent de devoir opérer sur de nombreux domaines d'application mais pour lesquels aucune mesure d'évaluation claire n'a été définie. Aussi, cette thèse s'attache à étudier les dialogues débouchant sur un objectif clair (goal-oriented dialogue) permettant de guider l'entrainement, et ceci, dans des environnements multimodaux. Plusieurs raisons expliquent ce choix : (i) cela contraint le périmètre de la conversation, (ii) cela introduit une méthode d'évaluation claire, (iii) enfin, l'aspect multimodal enrichie la représentation linguistique en reliant l'apprentissage du langage avec des expériences sensorielles. En particulier, nous avons développé GuessWhat ?! (Qu-est-ce donc ?!), un jeu imagé coopératif où deux joueurs tentent de retrouver un objet en posant une série de questions. Afin d'apprendre aux agents de répondre aux questions sur les images, nous avons développés une méthode dites de normalisation conditionnée des données (Conditional Batch Nornalization). Ainsi, cette méthode permet d'adapter simplement mais efficacement des noyaux de convolutions visuels en fonction de la question en cours. Enfin, nous avons étudié les tâches de navigation guidée par dialogue, et introduit la tâche Talk the Walk (Racontemoi le Chemin) à cet effet. Dans ce jeu, deux agents, un touriste et un guide, s'accordent afin d'aider le touriste à traverser une reconstruction virtuelle des rues de New-York et atteindre une position prédéfinie.

Keywords: apprentissage profond, apprentissage par renforcement, dialogue, apprentissage du langage avec des expériences sensorielles

Summary

While dialogue systems have the potential to fundamentally change human-machine interaction, developing general chatbots with deep learning and reinforcement learning techniques has proven difficult. One challenging aspect is that these systems are expected to operate in broad application domains for which there is not a clear measure of evaluation. This thesis investigates goal-oriented dialogue tasks in multi-modal environments because it (i) constrains the scope of the conversation, (ii) comes with a better-defined objective, and (iii) enables enriching language representations by grounding them to perceptual experiences. More specifically, we develop GuessWhat, an image-based guessing game in which two agents cooperate to locate an unknown object through asking a sequence of questions. For the subtask of visual question answering, we propose Conditional Batch Normalization layers as a simple but effective conditioning method that adapts the convolutional activations to the specific question at hand. Finally, we investigate the difficulty of dialogue-based navigation by introducing Talk The Walk, a new task where two agents (a "tourist" and a "guide") collaborate to have the tourist navigate to target locations in the virtual streets of New York City.

Keywords: deep learning, reinforcement learning, goal-oriented dialogue, grounded language learning

Contents

List of Figures

List of Tables

List of Abbreviations

AdaM	Adaptative Moment estimation
AI	Artificial Intelligence
BN	Batch Normalization
CBN	Conditional Batch Normalization
CNN	Convolutional Neural Network
DNN	Deep Neural Network
FiLM	Feature-wise Linear Modulation
LSTM	Long-Short Term Memory
MASC	Masked Attention for Spatial Convolutions
MILA	Montréal Institute of Learning Algorithms
MLE	Maximum Likelihood Estimation
MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NLL	Negative Log-Likelihood
NYC	New York City
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SGD	Stochastic Gradient Descent
VQA	Visual Question Answering

Notation

This thesis adopts (where possible) the notation of the deep learning book (Goodfellow et al., 2016). To make this manuscript as self-contained as possible, we include a style sheet with examples below.

Numbers and Arrays

a	A scalar (integer or real)
$\boldsymbol{a}$	A vector
$\boldsymbol{A}$	A matrix
$\mathbf{A}$	A tensor
$\boldsymbol{I}_n$	Identity matrix with n rows and n columns
$\boldsymbol{I}$	Identity matrix with dimensionality implied by context
$\boldsymbol{e}^{(i)}$	Standard basis vector $[0, \dots, 0, 1, 0, \dots, 0]$ with a 1 at position i
$\text{diag}(\boldsymbol{a})$	A square, diagonal matrix with diagonal entries given by $\boldsymbol{a}$
a	A scalar random variable
$\mathbf{a}$	A vector-valued random variable
$\mathbf{A}$	A matrix-valued random variable

Sets and Graphs

$\mathbb{A}$	A set
$\mathbb{R}$	The set of real numbers
$\{0, 1\}$	The set containing 0 and 1
$\{0, 1, \dots, n\}$	The set of all integers between 0 and n
$[a, b]$	The real interval including a and b
$(a, b]$	The real interval excluding a but including b
$\mathbb{A} \backslash \mathbb{B}$	Set subtraction, i.e., the set containing the elements of $\mathbb{A}$ that are not in $\mathbb{B}$
$\mathcal{G}$	A graph
$Pa_{\mathcal{G}}(\mathrm{x}_i)$	The parents of x_i in $\mathcal{G}$

Indexing

a_i	Element i of vector $\boldsymbol{a}$, with indexing starting at 1
a_{-i}	All elements of vector $\boldsymbol{a}$ except for element i
$A_{i,j}$	Element i, j of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{i,:}$	Row i of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{:,i}$	Column i of matrix $\boldsymbol{A}$
$\mathsf{A}_{i,j,k}$	Element (i, j, k) of a 3-D tensor $\mathbf{A}$
$\mathbf{A}_{:,:,i}$	2-D slice of a 3-D tensor
a_i	Element i of the random vector $\mathbf{a}$

Calculus

$\frac{dy}{dx}$	Derivative of y with respect to x
$\frac{\partial y}{\partial x}$	Partial derivative of y with respect to x
$\nabla_{\boldsymbol{x}} y$	Gradient of y with respect to $\boldsymbol{x}$
$\nabla_{\boldsymbol{X}} y$	Matrix derivatives of y with respect to $\boldsymbol{X}$
$\nabla_{\mathbf{X}} y$	Tensor containing derivatives of y with respect to $\mathbf{X}$
$\frac{\partial f}{\partial \boldsymbol{x}}$	Jacobian matrix $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ of $f : \mathbb{R}^n \to \mathbb{R}^m$
$\nabla_{\boldsymbol{x}}^2 f(\boldsymbol{x})$ or $\boldsymbol{H}(f)(\boldsymbol{x})$	The Hessian matrix of f at input point $\boldsymbol{x}$
$\int f(\boldsymbol{x}) d\boldsymbol{x}$	Definite integral over the entire domain of $\boldsymbol{x}$
$\int_{\mathbb{S}} f(\boldsymbol{x}) d\boldsymbol{x}$	Definite integral with respect to $\boldsymbol{x}$ over the set $\mathbb{S}$

Probability and Information Theory

$P(\mathrm{a})$	A probability distribution over a discrete variable
$p(\mathrm{a})$	A probability distribution over a continuous variable, or over a variable whose type has not been specified
$\mathrm{a} \sim P$	Random variable a has distribution P
$\mathbb{E}_{\mathrm{x} \sim P}[f(x)]$ or $\mathbb{E} f(x)$	Expectation of $f(x)$ with respect to $P(\mathrm{x})$
$\mathrm{Var}(f(x))$	Variance of $f(x)$ under $P(\mathrm{x})$
$\mathrm{Cov}(f(x), g(x))$	Covariance of $f(x)$ and $g(x)$ under $P(\mathrm{x})$
$H(\mathrm{x})$	Shannon entropy of the random variable x
$D_{\mathrm{KL}}(P \Vert Q)$	Kullback-Leibler divergence of P and Q
$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Gaussian distribution over $\boldsymbol{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

Functions

$f : \mathbb{A} \to \mathbb{B}$	The function f with domain $\mathbb{A}$ and range $\mathbb{B}$
$f \circ g$	Composition of the functions f and g
$f(\boldsymbol{x}; \boldsymbol{\theta})$	A function of $\boldsymbol{x}$ parametrized by $\boldsymbol{\theta}$. (Sometimes we write $f(\boldsymbol{x})$ and omit the argument $\boldsymbol{\theta}$ to lighten notation)
$\log x$	Natural logarithm of x
$\sigma(x)$	Logistic sigmoid, $\dfrac{1}{1 + \exp(-x)}$
$\|\|\boldsymbol{x}\|\|_p$	L^p norm of $\boldsymbol{x}$
$\|\|\boldsymbol{x}\|\|$	L^2 norm of $\boldsymbol{x}$
$\mathbf{1}_{\text{condition}}$	is 1 if the condition is true, 0 otherwise

Acknowledgement

It goes without saying that this work would not have been possible without the support of many people.

First and foremost, I would like to thank my advisor Prof. Aaron Courville for his invaluable support throughout my doctorate program. His high-level overview of the research field, as well as his deep technical knowledge, has been tremendously helpful in shaping my own research program. Meetings with Aaron were always fun, productive, and often led to exciting new ideas. I admire his enthusiasm for new scientific ideas, his clarity of thought, and his great sense of humility. I feel privileged for having been under his mentorship and for teaching me how to become a better scientist.

I would also like to express my gratitude towards Prof. Yoshua Bengio for accepting me into the PhD program, for his guidance throughout the first year of my studies, and for creating such a stimulating research environment. His enthusiasm and passion for artificial intelligence has been truly inspiring.

I am also thankful to Dr. Roland Memisevic for being a great co-supervisor during my second year.

During my PhD, I had the fortune of going on several research internships. A big thanks goes to Hugo Larochelle for inviting me to his research group at Twitter Cortex, and for being a great collaborator on the projects presented in this thesis. Hugo has the unique ability to quickly understand the depth of your technical problem and to offer new perspectives on them. I'd also like to thank Olivier Pietquin for welcoming me to his group at INRIA Lille, for communicating his enthusiasm for reinforcement learning, and for showing me the French way of living. I'm also thankful to Bilal and Jeremie for being great co-authors. Gratitude is also due to Douwe Kiela for giving me the opportunity to spend time at Facebook AI Research (FAIR). I have enjoyed philosophical perspective on the language grounding problem. At FAIR, I have also benefitted from the many interactions with Jason Weston, Dhruv Batra, and Devi Parikh.

A major role in this work was played by my collaborator and friend Florian Strub. Thank you for bearing with me during all the late night coding and writing. I also enjoyed co-organizing the NeurIPS workshops on Visually Grounded Interaction and Language with you.

I'd like to thank the administrative staff at the University of Montreal for making my life so much easier. In particular, thanks to Celine Begin and Linda Peinthiere for taking care of all the paperwork. At the Montreal Institute for Learn-

ing Algorithms, I was surrounded by many brilliant researchers who contributed to a very unique research environment. I feel honored to have been part of this lab and I am grateful to everyone with whom I had fruitful interactions, including: Adriana Romero, Adrien Ali Taïga, Ahmed Touati, Alexandre De Brébisson, Amar Shah, Amjad Almahairi, Amy Zhang, Asja Fisher, Bart van Merriënboer, Çağlar Gülçehre, César Laurent, Daniel Jiwoong Im, David Krueger, Dzmitry Bahdanau, Ethan Perez, Faruk Ahmed, Felix Hill, Francesco Visin, Frédéric Bastien, Guillaume Alain, Jörg Bornschein, Junyoung Chung, Kelvin Xu, Kyunghyun Cho, Kyle Kastner, Laurent Dinh, Li Yao, Marcin Moczulski, Martin Arjovski, Mathias Berglund, Mathieu Germain, Mehdi Mirza, Mohamed Ishmael Belghazi, Nicolas Ballas, Pascal Lamblin, Pascal Vincent, Saizheng Zhang, Sarath Chandar, Sungjin Ahn, Tegan Maharaj, Tim Cooijmans, Vincent Dumoulin, Yann Dauphin, and Yaroslav Ganin.

I am also thankful for all the stimulating interactions with many members from the research community which I met at conferences or at the research groups I visited: Abhishek Das, Arthur Szlam, Daniele Callandro, Eric Nalisnick, Ilija Bogunovic, Jake Snell, Jasper Snoek, Jeremie Mary, Laurens van der Maaten, Kurt Shuster, Owen Lewis, Matteo Pirotta, Mengye Ren, Michal Valko, Nal Kalchbrenner, Ronan Fruit, Sachin Ravi, and Taco Cohen.

Some special words of gratitude go to my friends who kept me sane during the long working hours. Among them are Alberic, Ayman, Benjamin, Cesar, Daniel, Eduardo, Emile, Fleur, Patrick, and Pieter.

Lastly, I am indebted to my sister in law, Quirine, my brothers, Evert and Jaap, and my parents, Jan and Rineke, for their continuous support, love and care. Even though I was physically far away, they knew I was chasing my dream and their moral support has been invaluable.

Harm de Vries,
March 2019

1 Introduction

While people use natural language in many forms throughout their daily lives, arguably the most natural and fundamental use is dialogue. Every person, whether young or old, can hold a conversation, whereas reading essays or giving presentations are considerably more difficult. The ease with which humans engage in dialogue also manifests itself in how often people use these conversational skills in their day-to-day activities, including when it comes to ordering food, talking about the weather conditions or participating in a political debate.

Inspired by these remarkable human capabilities, researchers in the field of Artificial Intelligence (AI) have long been dreaming of machines that can converse via natural language. The quest for creating conversational agents—also known as dialogue systems or chatbots—is not only driven by academic curiosity but also by their potential economic value. As humans already know how to chat, intelligent machines with true conversational capabilities could be readily deployed in a wide array of domains (e.g. customer support, commerce, healthcare, etc) with minimal adaptation by the end user. By contrast, traditional ways of interacting with computing devices, such as through terminals or graphical user interfaces, are not as natural for humans, leading to a much steeper learning curve. Thus, realizing human-level dialogue systems would genuinely disrupt the field of human-machine interaction and, as a result, accelerate the integration of intelligent machines into our contemporary society.

Early attempts at creating chatbots date back more than 50 years ago with ELIZA, designed by Joseph Weizenbaum in 1966, as the first well-known dialogue system. ELIZA consists of a set of hard-coded rules aiming to simulate a Rogerian psychotherapist by merely rephrasing the user's reply. While convincing to some participants, ELIZA was still easily distinguished from humans by its limited language understanding and lack of common knowledge. Over the last decades, follow-up chatbots have attempted to design better if-then rules, albeit with minimal success, which led many of today's researchers to believe that capturing the

full complexity of dialogue with hand-coded systems is complicated, if not impossible. Chatbot designers have therefore focused on narrow domains like flight or restaurant booking, but even in such setups, it is difficult to account for all possible dialogue scenarios in advance. For rare use cases, this often results in frustrating user experiences. A more viable, scalable, and less human-labor-intensive approach is to build chatbots through the use of data-driven methods. Rather than using hard-coded rules, machine learning methods infer dialogue policies from transcripts of human conversations. Even though such machine learning methods are trained on a limited number of conversations, they have the potential to generalize to use cases beyond the training data.

The last decade has seen an explosion of interest in machine learning, primarily led by a class of learning methods, known as deep learning, that brought breakthroughs in several application domains, including speech recognition, object detection, and machine translation. The success of deep learning was first witnessed in supervised learning tasks, in which one aims to learn an input-output mapping from example pairs. Deep learning methods parameterize this input-output mapping through an artificial neural network, whose operations are loosely inspired by the human brain. Contrary to prior learning methods that extract high-level features from the input data, neural networks often learn directly from low-level input data like RGB pixels or raw audio waves. Deep neural networks have also been incorporated into reinforcement learning setups, where an agent learns to act in an unknown environment in order to maximize some pre-defined reward. Such deep reinforcement learning methods have beaten the world's best player at the game of Go as well as human experts at playing numerous Atari video games.

The rapid progress of deep learning and reinforcement learning has motivated the application of these techniques to the field of dialogue modeling, where they are known under the name of end-to-end approaches or neural models. Neural models have mainly been used in the chitchat setting, where a conversational agent needs to converse with human participants in open-ended domains. Current end-to-end models display rather poor conversational performance, as their generated responses suffer from issues such as (i) being non-specific and (ii) lacking long-term consistency. Because of these limitations, chatbots for commercial applications, such as Amazon's Alexa, Apple's Siri and Google's Assistant, only use neural networks for particular building blocks and utilize hand-coded programs for other

parts.

While chit-chat is considered to be the grand goal for conversational agents, this thesis studies dialogue in a much more constrained context, the so-called goal-oriented dialogue setups. As the name suggests, conversations in such setups are driven by the need to accomplish a specific task, such as booking a flight or reserving a table at a restaurant. Often, these dialogue setups can be phrased as an information-transfer game, where two interlocutors, both having access to some form of private information, converse to resolve their information discrepancy. For example, in the case of restaurant booking, there is an information-seeking agent with particular restaurant preferences and an information-providing agent with access to a list of available restaurants. The goal of the dialogue is then to transfer the information-seeking's value function—i.e. restaurant preferences—to the information-provider, who can then, based on this information, pick a restaurant from the list. Such goal-oriented setup come therefore with a straightforward evaluation measure: whether or not the information-provider chose a restaurant that met the preferences of the information-seeker. Chit-chat settings do not (yet) have an automated evaluation procedure, which makes measuring progress for this class of conversational agents much more cumbersome.

This thesis is concerned with goal-oriented dialogue problems in multi-modal environments, where interlocutors are not only exposed to text-based information but also other input modalities such as images or sounds. The solution to these set of problems requires to combine conversational skills with the ability to perceive, listen, or act in a virtual world. The motivation for studying multi-modal dialogue setups are at least two-fold. First, recent evidence from the field of cognitive science suggests that human language learning is inherently multi-modal (Barsalou, 2008a; Smith and Gasser, 2005). That is, humans run simulations of their perceptual and motor systems when they understand and produce natural language. Given that people acquire language capabilities in a sensory-rich environment, some scholars have argued that a grounded learning environment is essential to fully understand all aspects of language. For example, the fact that a banana is often yellow is much easier to derive from images than from text documents, as these facts are so implicit that no one ever writes them down. Second, as the field of robotics is progressing at an unprecedented rate, we might soon be surrounded by robots doing physical tasks. In this futuristic scenario, robots will perceive and act in

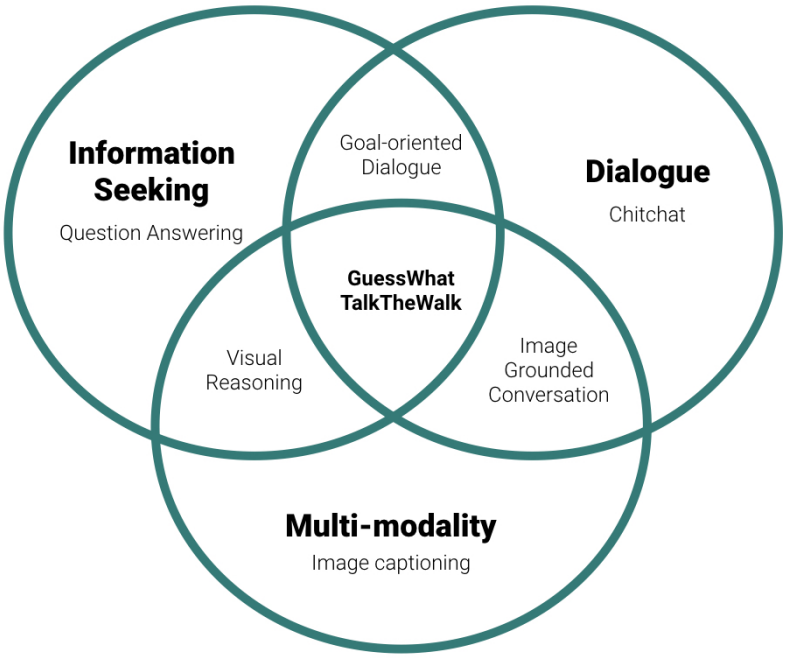


Figure 1.1 – This thesis studies deep and reinforcement learning algorithms for problems on the intersection of the following three pillars: dialogue, information-seeking and multi-modality. This figure illustrates how these setups relate to other popular learning tasks.

the real world, as well as interact with humans via natural language. Multi-modal dialogue is perhaps the closest learning setup to prepare for this scenario.

To summarize, this thesis studies deep learning and reinforcement learning algorithms for problems on the intersection of the following three pillars: dialogue, information-seeking and multi-modality. Other popular learning tasks emerge, such as image captioning, goal-oriented dialogue, and visual question answering, if we only pick other subsets—see Fig. 1.1 for an overview. Concretely, this thesis makes the following contributions to this field:

— We introduce GuessWhat?!, an image guessing game, where one player, the questioner, needs to locate an unknown object by asking yes-no questions to the other player, the oracle. We collect a large scale dataset of more than 150k human-played games and establish supervised deep learning baselines for the three sub-tasks.

— For the task of generating a series of questions, we show that deep reinforcement learning methods achieve higher task accuracy than pure supervised methods. Analysis of the policies reveals that the deep RL method suffers

less from repeating questions, and tailors their questions to strengths of the oracle model.

— We introduce Conditional Batch Normalization (CBN) layers—now better known as Feature-wise Linear Modulation (FiLM) layers in the literature—to modulate the convolutional activations to the question at hand. We insert CBN layers into a pre-trained convolutional network, and show on two visual question answering benchmarks that it is beneficial to modulate early on in the visual processing.

— We introduce Talk the Walk, a new dialogue task where two agents (a "tourist" and a "guide") collaborate to have the tourist navigate to target locations in the virtual streets of New York City. We establish baselines for the full task by training localization models where both agents communicate via emergent or natural language.

2 Background

This chapter covers the basics of machine learning, aiming to provide the reader with a high-level overview of the mathematical concepts underlying the contributions of this thesis. This material is by no means a complete account of machine learning, deep learning, and reinforcement learning, and readers interested in a thorough exposure of these fields are referred to the books of Bishop (2006); Goodfellow et al. (2016); Sutton and Barto (1998), respectively. You are encouraged to skip this chapter if you are already familiar with deep and reinforcement learning.

2.1 Machine Learning

Machine learning is a branch of computer science concerned with studying systems that improve their performance with observations or data. Rather than hand-coding algorithms, machine learning methods infer algorithmic solutions directly from data. The main motivations for the interest in learning systems is that traditional computer science algorithms have fallen short on some real-world tasks. That is, although efficient algorithms have been developed for well-defined mathematical problems like sorting numbers and calculating the shortest path between nodes in a graph, certain tasks in the field of computer vision and natural language processing were too complicated to be governed by a set of hand-written rules. For instance, although humans recognize objects in images effortlessly, it is unclear how they perform such a task, making it incredibly complicated to write down a program for it. Machine learning methods take a fundamentally different approach to solving such problems: by first collecting a set of input-output examples for the objects to recognize, and then searching through a family of functions to find the program that induces the right behavior for these examples. The latter example

falls in the class of supervised learning setups, which we will formalize in the next section.

2.1.1 Supervised Learning

Perhaps the most commonly used form of machine learning is supervised learning. This setup assumes access to a collection of labeled examples, the so-called training set $\mathcal{D} = \{(\boldsymbol{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$. Each data point is characterized by a D-dimensional feature vector $\boldsymbol{x}^{(n)} \in \mathbb{R}^D$, where each dimension, or feature, is a numeric value that represents some quantitative measurement of the object. For example, features of a person could be age, height, and weight, while features of an image could be its pixel values. Each data point also has a label $y^{(n)}$, which represents the desired output. Often, target labels are manually assigned by a human domain expert. If the label belongs to one of K classes, i.e. $y^{(n)} \in \{1, \ldots, K\}$, we speak of a *classification* problem, and if $y^{(n)} \in \mathbb{R}$ is real-valued we call it a *regression* problem. Task involving more complicated label structures, such as sequences or graphs, are referred to as *structured prediction* problems.

The objective of supervised learning tasks is to find a function f that correctly maps the inputs to the desired outputs. To quantify how well function f is fitting the dataset, we specify a per-sample-loss function ℓ that measures the discrepancy between the prediction and target. By summing the per-sample-losses, we obtain the total loss over the dataset:

$$L^{train}(f) = \sum_{n=0}^{N} l(f(\boldsymbol{x}^{(n)}), y^{(n)}). \tag{2.1}$$

For classification problems, a natural loss function is the error, defined as:

$$\ell^{acc}(\hat{y}, y) = \mathbf{1}_{\hat{y}==y}, \tag{2.2}$$

which measures the proportion of examples for which the function predicts the incorrect label. Rather than predicting a single class label, classification models often output a K-dimensional vector representing the model distribution over the class labels, i.e. $\hat{\boldsymbol{y}}_c = P_{\text{model}}(\text{y} = c|\boldsymbol{x})$, and use the negative log likelihood of the correct label as loss:

$$\ell(\hat{\boldsymbol{y}}, y) = -\log \hat{\boldsymbol{y}}_y. \tag{2.3}$$

This particular loss also corresponds to the cross entropy between the data and model distribution (see () for more information on this probabilistic interpretation). For regression problems, we often use the squared error:

$$\ell^{sq}(\hat{y}, y) = (\hat{y} - y)^2, \tag{2.4}$$

or absolute error:

$$\ell^{abs}(\hat{y}, y) = |\hat{y} - y|. \tag{2.5}$$

Note that until now we only considered the loss on the training examples, even though it is not the metric that we care about. Instead, we are interested in the loss on unseen examples, as this indicates how the model would perform when deployed in the real-world. To mathematically characterize generalization behavior, we assume that each example is drawn independently from an unknown probability distribution $p(\mathbf{x}, \mathrm{y})$. This probabilistic assumption allows us to define the expected loss, or risk:

$$R(f) = \int l(f(\boldsymbol{x}), y) dp(\mathbf{x}, \mathrm{y}) \tag{2.6}$$

In practice, we can not compute the risk because we do not have access to the data generating distribution. We therefore estimate this quantity by calculating the loss L^{test} on a separate set of held-out examples, known as the test set $\mathcal{D}_{\text{test}}$.

Now, the critical question that machine learning studies is: given training set $\mathcal{D}$ and loss function l, how do we find a function f that minimizes the test loss L^{test}? The most common principle for finding an approximate solution to this question is *Empirical Risk Minimization*. In this framework, we search for the function f^* in a pre-defined hypothesis space $\mathcal{F}$ that minimizes the training loss L^{train}:

$$f^* = \arg\min_{f \in \mathcal{F}} L^{train}(f), \tag{2.7}$$

in the hope that f^* achieves low loss on the test set. To make this principle more concrete, let us consider a *softmax regression model* for a K-way classification problem. This model first obtains a K-dimensional score vector $\boldsymbol{s}$ by applying a linear transformation the input $\boldsymbol{x}$:

$$\boldsymbol{s} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b} \tag{2.8}$$

which is then transformed into a probability distribution over the class labels via the softmax operation:

$$\hat{\boldsymbol{y}}_c = \text{softmax}(\boldsymbol{s})_c = \frac{\exp(\boldsymbol{s}_c)}{\sum_j \exp(\boldsymbol{s}_j)}. \tag{2.9}$$

Finally, we set the loss to the log probability of outputting the correct label:

$$\ell(\hat{\boldsymbol{y}}, y) = -\log \hat{\boldsymbol{y}}_y. \tag{2.10}$$

In this example, the hypothesis space $\mathcal{F}$ is the set of linear functions, defined by the parameters $\boldsymbol{\theta} = [\text{vec}(\boldsymbol{W}); \boldsymbol{b}]$. Unlike the classification error, the negative log likelihood is differentiable with respect to the parameters $\boldsymbol{\theta}$, meaning that we can apply continuous optimization methods to find optimal parameters $\boldsymbol{\theta}^*$ that minimize the training loss L^{train}. Technically speaking, this optimization problem is *convex* in its parameters and therefore enjoys many favorable properties, including an optimization landscape with one global minimum which is reachable by first-order methods (Boyd and Vandenberghe, 2004). That is, iteratively taking small steps in the direction of the negative gradient:

$$\boldsymbol{\theta}^{t+1} = \boldsymbol{\theta}^t - \alpha \nabla_{\boldsymbol{\theta}} L^{train}(\boldsymbol{\theta}), \tag{2.11}$$

will eventually converge to $\boldsymbol{\theta}^*$ when an appropriately step-size α is chosen.

How well the function f^* generalizes largely depends on the underlying data generating process and the properties of the chosen hypothesis space $\mathcal{F}$. For example, the hypothesis space in the softmax regression problem is restricted to linear functions, so if the underlying data generating process is highly non-linear, we will experience so-called *underfitting*. Because there is no function in the hypothesis space that can fit the data well, this results in high training and test error. The only way then to improve generalization performance is to choose another (more expressive) hypothesis space that better fits the data. The opposite of underfitting can happen as well; if we only have a very small dataset but search in a very large hypothesis space, we are at risk of *overfitting*. Rather than fitting the true underlying function, the models then learns spurious patterns and noise present in the training data. We can easily diagnose overfitting because it results in low training error but high test error. A common way to combat overfitting is—besides

collecting more data—to regularize the search process towards "simpler" functions. Instead of hard-constraining the hypothesis space, regularizers softly prefers "simpler" function in the hypothesis space, often at the expense of higher training error. Usually, a *regularizer* is implemented by adding a term $\Omega(\boldsymbol{\theta})$ to the training loss:

$$L^{train}(\boldsymbol{\theta}) = \sum_{n=0}^{N} l(f(\boldsymbol{x}^{(n)}; \boldsymbol{\theta}), y^{(n)}) + \lambda\Omega(\boldsymbol{\theta}), \tag{2.12}$$

where λ is a *hyperparameter* that determines the strength of the regularization term. Examples of popular regularizers include L1 regularization: $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_1$, and L2 regularization: $\Omega(\boldsymbol{\theta}) = ||\boldsymbol{\theta}||_2$. To determine the amount of regularization λ, as well as the values of other hyperparameters, we often use a third set of examples, the *validation set* $\mathcal{D}_{\text{valid}}$.

2.2 Deep Learning

Deep learning is a branch of machine learning, particularly well-suited to process natural forms of data, such as images, speech, and language. Until very recently, constructing learning models for these application domains required significant engineering effort, as well as considerable domain expertise. For instance, if you want to build a classifier that can recognize cats and dogs from an image, we would first extract hand-crafted features from the picture, which are then fed to a simple (linear) classifier. For such systems, the choice of feature representation is of utmost importance for the success of the machine learning model. Practitioners have therefore devoted much of their time to hand-engineering effective representations by designing preprocessing pipelines for extracting image features. Examples of frequently-used image features are Scale-invariant Feature Transform (SIFT) (Lowe, 2004) or Histograms of Oriented Gradients (HOG) (Dalal and Triggs, 2005).

Rather than hand-designing features, the premise of deep learning is to *learn* representations directly from raw data. While the idea is conceptually simple, representation learning is challenging because of the high-dimensionality of natural data. For example, even a small RGB image of size 64×64 already leads to a

feature space with more than 12k dimensions. Such high-dimensional spaces require exponentially more data points to densely fill the same volume, a phenomenon that is known as the *curse of dimensionality* (Bellman, 1966). So, even for a very large dataset, we are still facing a very sparsely-populated feature space. This is problematic for conventional machine learning methods, such as support vector machines and nearest neighbor methods, that base their predictions on a local neighborhood around the considered data point. Namely, the closest neighbors of a particular data point may be so far apart that their outputs are not semantically related, i.e., we can not leverage the smoothness prior that nearby data points have similar labels. Hence, learning algorithms for this set of problems require one to incorporate priors that go beyond local generalization.

Deep learning models generally assume the existence of multiple factors of variation that are responsible for explaining the underlying data-generating process (Bengio et al., 2013). If we can independently learn about such factors of variation, then it possible to achieve non-local generalization by constructing novel combinations of factors that were not seen during training. Deep learning exploits *distributed representations* (Hinton et al., 1984) to capture the factors of variation. Such representations spread the information across multiple dimensions rather than taking a one-hot value (a vector with one non-zero component), leading to exponentially more efficient representations. For example, to represent a set of N integers, a binary representation only requires $\log_2 N$) values, whereas a one-hot representation needs N values. As the name suggests, the second prior at the essence of deep learning is depth in the representation. Deep learning methods construct such hierarchical representations by composing multiple layers of transformation, aiming to gradually capture more abstract aspects of the data distribution. For certain function classes, deep representations require exponentially fewer parameters than shallow representations, indicating that depth can offer computational and statistical advantages (Montufar et al., 2014; Bengio et al., 2013).

2.2.1 Feed-forward Networks

A feed-forward neural network, or multi-layer perceptron (MLP), is perhaps one of the most prototypical examples of an artificial neural network. These models

are constructed by chaining several sub-functions:

$$f(\boldsymbol{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\boldsymbol{x}))), \tag{2.13}$$

where each $f^{(j)}$ is often referred to as a j^{th} *layer* of the network. Each layer $f^{(j)}$ usually consists of an affine transformation, followed by an *element-wise* activation function g:

$$\boldsymbol{z}^{(j)} = \boldsymbol{W}^{(j)}\boldsymbol{h}^{(j-1)} + \boldsymbol{b}^{(j)}; \tag{2.14}$$

$$\boldsymbol{h}^{(j)} = g^{(j)}(\boldsymbol{z}^{(j)}) \qquad \boldsymbol{h}^{(0)} = \boldsymbol{x} \tag{2.15}$$

where the intermediate representations $\boldsymbol{h}^{(j)} \in \mathbb{R}^{d^{(j)}}$ are usually called the hidden units. The number of hidden units $d^{(j)}$ defines the size of the j^{th} layer. The parameters $\boldsymbol{W}^{(j)} \in \mathbb{R}^{d^{(j)} \times d^{(j-1)}}$ and $\boldsymbol{b} \in \mathbb{R}^{d^{(j)}}$ are called the weights and bias, respectively. When all the elements in $\boldsymbol{W}$ are non-zero, we speak of a *fully-connected* layer. The number of layers define the *depth* of the network. Common choices for the activation function g are:

— the Identity function[1]:

$$g(x) = x. \tag{2.16}$$

— the Sigmoid function:

$$\sigma(x) = \big(1 + \exp(-x)\big)^{-1}. \tag{2.17}$$

— the Hyperbolic Tangent function:

$$\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}; \tag{2.18}$$

— the Rectified Linear Unit:

$$\mathrm{ReLU}(x) = \max(0, x). \tag{2.19}$$

In recent years, ReLUs have become the default choice, as they were shown to outperform the Tanh and Sigmoid activation functions (Glorot et al., 2011; Maas et al.,

1. This choice results in a linear layer. If all layers in the network are linear, then the model can only express linear functions.

2013). For classification problems, the final layer often uses a softmax activation function:

$$\text{softmax}(\boldsymbol{x})_k = \frac{\exp(\boldsymbol{x}_k)}{\sum_{k'} \exp(\boldsymbol{x}_{k'})} \tag{2.20}$$

to output a probability distribution over the class labels.

2.2.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs), or ConvNets, constitute a popular class of neural networks, particularly well-suited to process locally-structured data. While ConvNets have been applied to several data modalities, including speech and natural language, this thesis will only study these models for image data. For that reason, we will establish the notation of convolutional layers in this context. Specifically, we assume that input image $\mathbf{X}$ is a 4-tensor [2], where $\mathbf{X}_{i,c,w,h}$ refers to the i^{th} input sample of the c^{th} channel at location (w, h). For 100 RGB images of size 128×128, this results in $\mathbf{X}$ having dimensions $(100, 3, 128, 128)$.

At the core of a CNN is the convolutional operator $*$, which, in its discrete form, is taking input matrix $\boldsymbol{F}$ and kernel matrix $\boldsymbol{G}$:

$$\boldsymbol{S}_{i,j} = (\boldsymbol{F} * \boldsymbol{G})_{i,j} = \sum_m \sum_n \boldsymbol{F}_{i-m,j-n} \boldsymbol{G}_{m,n}. \tag{2.21}$$

Intuitively, this operation slides the kernel $\boldsymbol{G}$ over the input $\boldsymbol{F}$ to produce matrix $\boldsymbol{S}$, whose output dimensions are determined by the kernel size (W, H), the stride size—i.e. the steps taken when sliding the kernel—and the type of padding (see Dumoulin and Visin (2016) for more details). Now, a convolutional layer produces the c^{th} feature, better known as feature map $\mathbf{H}^{(j)}_{i,c,:,:}$, by summing the result of convolutions with all incoming feature maps:

$$\mathbf{Z}^{(j)}_{i,c,:,:} = \boldsymbol{b}^{(j)}_c + \sum_{c'=0}^{d^{(j-1)}} \mathbf{H}^{(j)}_{i,c',:,:} * \mathbf{K}^{(j)}_{c,c',:,:}, \tag{2.22}$$

where kernel $\mathbf{K}^{(j)} \in \mathbb{R}^{d^{(j)} \times d^{(j-1)} \times W \times H}$ and bias $\boldsymbol{b}^{(j)} \in \mathbb{R}^{d^{(j)}}$ are the learnable parameters. Convolutional layers are designed to take advantage of the structure of

2. Unlike in the section on feed-forward networks, we assume here that $\mathbf{X}$ is batch of data points. This enables us to define the batch normalization operator later in this section.

images, by (i) using kernels that operate on small patches and (ii) sharing the kernel weights across different spatial locations. These architectural restrictions lead to representations that are translation equivariant, i.e. shifting the input image results in an accordingly shifted feature representation. In general, this assumption is useful for images because we are interested in detecting particular features, e.g. a person's eye, no matter where they are located. Compared to fully-connected layers, convolutional layers drastically reduce the number of parameters.

Like fully-connected layers, convolution layers are often followed by a non-linear activation function, such as the ReLU (Eq. 2.19). After this non-linearity, CNNs often add a *pooling layer* with the aim to incorporate some invariance with respect to the precise location of the feature. Pooling layers often compute the average or max over non-overlapping windows of the feature map, thereby reducing its spatial dimensions (often by at least a factor two). As a result, further processing stages require fewer parameters and computation, thereby significantly reducing the risk of overfitting.

Although convolutional neural networks have been used since the early 90s (Lecun et al., 1998), the recent surge of interest started in 2012 by a deep CNN winning the ImageNet classification challenge (Krizhevsky et al., 2012). The winning entry, known as AlexNet, consisted of a 8-layer convolutional network trained on one million images of the ImageNet dataset (Deng et al., 2009), which includes pictures of various dog breeds, foods, automobiles, and so on. Since then, CNNs have been dominating object recognition and detection competitions (Simonyan and Zisserman, 2015; Szegedy et al., 2015; Kaiming et al., 2016), with further architectural modifications leading to even stronger performance. Most notably, more recent convolutional neural networks have smaller kernels (often of size $3x3$) and are significantly deeper. For example, a VGG network (Simonyan and Zisserman, 2015) has 19 layers, while a Residual Network (ResNet) (Kaiming et al., 2016) has up to 152 layers. Deeper networks are increasingly more difficult to train due to vanishing gradients (Bengio et al., 1994; Hochreiter, 1998), so in order to train their 152-layer network, (Kaiming et al., 2016) introduced *residual blocks*:

$$\mathbf{H}^{(j)} = f(\mathbf{H}^{(j-1)}; \boldsymbol{\theta}^{(j)}) + \mathbf{H}^{(j-1)}, \tag{2.23}$$

where the j^{th} feature map is constructed by adding a residual $f(\mathbf{H}^{(j-1)}; \boldsymbol{\theta}^{(j)})$ to the

previous feature map $\mathbf{H}^{(j-1)}$. Adding such skip connections to the network makes it easier to propagate the gradients. Usually, the residual function f consists of two to three convolution layers. The convolutional layers in ResNets make use of another important innovation in training CNNs: *Batch Normalization* (BN) (Ioffe and Szegedy, 2015). This technique speeds up training but simultaneously acts as a regularizer. Given a mini-batch $\mathcal{B} = \{\mathbf{Z}_{i,\cdot,\cdot,\cdot}\}_{i=1}^{N}$ of N feature maps, BN normalizes the activations at training time as follows:

$$BN(\mathbf{Z}_{i,c,h,w}|\gamma_c, \beta_c) = \gamma_c \frac{\mathbf{Z}_{i,c,w,h} - \mathrm{E}_{\mathcal{B}}[F_{\cdot,c,\cdot,\cdot}]}{\sqrt{\mathrm{Var}_{\mathcal{B}}[\mathbf{Z}_{\cdot,c,\cdot,\cdot}] + \epsilon}} + \beta_c, \tag{2.24}$$

where ϵ is a constant damping factor for numerical stability, and γ_c and β_c are trainable scalars introduced to keep the representational power of the original network. Note that for convolutional layers the mean and variance are computed over both the batch and spatial dimensions (such that each location in the feature map is normalized in the same way). At inference time, the batch mean $\mathrm{E}_{\mathcal{B}}$ and variance $\mathrm{Var}_{\mathcal{B}}$ are replaced by the population mean μ and variance σ^2, often estimated by an exponential moving average over batch mean and variance during training.

After training on ImageNet, the intermediate activations of a CNN are surprisingly general image features which can be used to bootstrap other classification tasks for which there is limited data available (Donahue et al., 2014). Usually, this transfer learning procedure extracts activations of the penultimate layer of the CNN, which are then fed as input to a linear classifier or small MLP. In this thesis, we will often use features from VGG (Simonyan and Zisserman, 2015) and ResNet (Kaiming et al., 2016) to obtain such general-purpose image representations.

2.2.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are the architecture of choice for handling sequential data, such as audio and natural language. In this section, we focus on RNNs for natural language processing as this will be the main application in this thesis. Thus, we consider a linguistic sequence $\boldsymbol{x} = (x_1, \ldots, x_K)$ of length K where each word x_k is taken from a predefined vocabulary $\mathbb{V}$. We transform each token into a continuous word-embedding $\boldsymbol{\xi}_k = e(x_k) \in \mathbb{R}^P$ by a learned look-up table

e. The resulting sequence of embeddings $(\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_K)$ is then fed to a recurrent neural network which produces a sequence of RNN state vectors $(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_K)$ by repeatedly applying the transition function f:

$$\boldsymbol{s}_k = f(\boldsymbol{s}_{k-1}, \boldsymbol{\xi}_k). \tag{2.25}$$

Here, each state $\boldsymbol{s}_k$ is a R-dimensional vector representing the information about the elements of the sequence up to index k. Many transition functions are possible, but perhaps the simplest one is:

$$f(\boldsymbol{s}_{k-1}, \boldsymbol{\xi}_k) = \tanh(\boldsymbol{W}[\boldsymbol{s}_{k-1}; \boldsymbol{\xi}_k] + \boldsymbol{b}), \tag{2.26}$$

which leads to the Elman RNN (Elman, 1990). However, these simple RNNs have difficulties learning long-term dependencies as they suffer from the vanishing gradient problem (Bengio et al., 1994; Hochreiter, 1998). For that reason, more complex transition functions were designed, like the Long-Short Term Memory (LSTM) cell (Hochreiter and Schmidhuber, 1997) and the Gated Recurrent Unit (GRU) (Cho et al., 2014), that incorporate gating mechanisms to explicitly control the information flow of the memory cell. LSTMs also introduce a cell memory $\boldsymbol{c}_k$, so that the transition function $f(\boldsymbol{s}_{k-1}, \boldsymbol{c}_{k-1}, \boldsymbol{\xi}_k)$ depends on the previous state $\boldsymbol{s}_k$, :

$$\boldsymbol{i}_k = \sigma\big(\boldsymbol{\xi}_k \boldsymbol{U}^i + \boldsymbol{s}_{k-1} \boldsymbol{W}^i\big) \tag{2.27}$$
$$\boldsymbol{f}_k = \sigma\big(\boldsymbol{\xi}_k \boldsymbol{U}^f + \boldsymbol{s}_{k-1} \boldsymbol{W}^f\big) \tag{2.28}$$
$$\boldsymbol{o}_k = \sigma\big(\boldsymbol{\xi}_k \boldsymbol{U}^o + \boldsymbol{s}_{k-1} \boldsymbol{W}^o\big) \tag{2.29}$$
$$\tilde{\boldsymbol{c}}_k = \tanh\big(\boldsymbol{\xi}_k \boldsymbol{U}^g + \boldsymbol{s}_{k-1} \boldsymbol{W}^g\big) \tag{2.30}$$
$$\boldsymbol{c}_k = \sigma\big(\boldsymbol{f}_k * \boldsymbol{c}_{k-1} + \boldsymbol{i}_t * \tilde{\boldsymbol{c}}_k\big) \tag{2.31}$$
$$\boldsymbol{s}_k = \tanh(\boldsymbol{c}_k) * \boldsymbol{o}_k. \tag{2.32}$$

The input gate $\boldsymbol{i}_k$ and forget gate $\boldsymbol{f}_k$ regulate the information flow into the new cell $\boldsymbol{c}_k$ by weighting the sum over the old cell vector $\boldsymbol{c}_{k-1}$ and new cell vector $\tilde{\boldsymbol{c}_k}$, respectively. To obtain the new state-vector $\boldsymbol{c}_k$, we apply a tanh non-linearity to the cell vector $\boldsymbol{c}_k$ and multiply it element-wise with the output gate $\boldsymbol{o}_k$. Finally, to encode sequence $\boldsymbol{x}$, we take the last hidden state $\boldsymbol{s}_K$.

Besides encoding sequences, RNNs can also be used for generating sequences.

To that end, we take a probabilistic perspective and parameterize the RNN to model the probability for a given sequence $\boldsymbol{y}$, i.e. $P(\boldsymbol{y}) = P(y_1, \ldots, y_L)$. The chain rule enables us to decompose this joint probability into the product of conditional probabilities: $\prod_{l=1}^{L} P(y_l|\mathbf{y}_{1:l-1})$. We let the RNN output the conditional probability $P(y_l|\boldsymbol{y}_{1:l-1})$ through a linear transformation from the hidden state $\boldsymbol{s}_{l-1}$, followed by a softmax:

$$P(y_l|\boldsymbol{y}_{1:l-1}) = \text{softmax}(\boldsymbol{W}\boldsymbol{s}_{l-1} + \boldsymbol{b})_{y_l}. \tag{2.33}$$

The RNN model defines a probability over the sequence $\boldsymbol{y}$, allowing us to optimize its parameters with respect to the negative log likelihood:

$$-\log P(\boldsymbol{y}) = \sum_{l} \log P(y_l|\boldsymbol{y}_{l-1}). \tag{2.34}$$

This way of training is called *teacher forcing*, as we feed in the ground-truth label y_{l-1} of the previous step as input to the current time step k. At evaluation time, we do not have access to ground-truth labels and therefore feed a sample y_{l-1} from its own distribution $P(\mathrm{y}_{l-1}|\mathbf{y}_{1:l-2})$ back as input to the l^{th} time step. Hence, we can generate samples from the sequential data distribution $P(\mathrm{y}_1, \ldots, \mathrm{y}_L)$ by repeatedly sampling words from the conditional model distribution $P(\mathrm{y}_l|\mathbf{y}_{1:l-1})$ till we encounter the special end-of-sequence token.

Sequence-to-sequence models (Cho et al., 2014; Sutskever et al., 2014) combine an RNN encoder and decoder aiming to map the input sequence $\boldsymbol{x} = (x_1, \ldots, x_K)$ to the output sequence $\boldsymbol{y} = (y_1, \ldots, y_L)$. These models first encode the input sequence $\boldsymbol{x}$ via an encoder RNN f^{enc} into a fixed embedding $\boldsymbol{e}^{enc}$. This fixed vector $\boldsymbol{e}^{enc}$ is then concatenated to the input of the decoder RNN f^{dec} at each step, i.e.:

$$\boldsymbol{\xi}_l = [\boldsymbol{\xi}_l; \boldsymbol{e}^{enc}]. \tag{2.35}$$

In this way, we model the conditional distribution $P(\mathrm{y}_1, \ldots, \mathrm{y}_L|\boldsymbol{x})$ rather than $P(\mathrm{y}_1, \ldots, \mathrm{y}_L)$. Encoder-decoder architectures have been successfully applied to machine translation (Cho et al., 2014; Sutskever et al., 2014), where a source language sentence $\boldsymbol{x}$ needs to be translated into target language sentence $\boldsymbol{y}$. Encoder-decoder models also been applied to dialogue (Vinyals and Le, 2015; Serban et al., 2016), where the history of dialogue $\boldsymbol{x}$ is mapped to a dialogue response $\boldsymbol{y}$.

2.2.4 Optimization

Deep learning methods employ continuous optimization methods to search for the parameters $\boldsymbol{\theta}^*$ that minimize the training loss $L(\boldsymbol{\theta}^*)$ [3]. We often use variants of Stochastic Gradient Descent (SGD) to iteratively update the parameters in the direction of steepest descent, as defined by the gradient of the loss L with respect to the parameters $\boldsymbol{\theta}$. To efficiently compute the gradient $\nabla_{\boldsymbol{\theta}} L$ for the neural network parameters, we use a procedure called *backprogation* (Rumelhart et al., 1986). Starting from the output, the backpropagation algorithm walks backwards over the computation graph and computes the partial derivatives with respect to the intermediate nodes via the multi-variate chain rule. More specifically, suppose the loss L is a function of the hidden units $\boldsymbol{h}_1^{(j+1)}, \ldots, \boldsymbol{h}_P^{(j+1)}$, which in turn depend on the hidden unit $\boldsymbol{h}_a^{(j)}$ in the previous layer, then the multi-variate chain rule states that:

$$\frac{\partial L}{\partial \boldsymbol{h}_a^{(j)}} = \sum_p \frac{\partial L}{\partial \boldsymbol{h}_p^{(j+1)}} \frac{\partial \boldsymbol{h}_p^{(j+1)}}{\partial \boldsymbol{h}_a^{(j)}}. \tag{2.36}$$

If we apply this rule to the MLP example in equation 2.14, then the gradient $\nabla_{\boldsymbol{z}^{(j)}} L$ with respect to the pre-activation $\boldsymbol{z}^{(j)}$ is given by following the recursive equation:

$$\nabla_{\boldsymbol{z}^{(j)}} L = \frac{\partial L}{\partial \boldsymbol{z}^{(j)}} = (\boldsymbol{W}^{(j)})^T \frac{\partial L}{\partial \boldsymbol{z}^{(j+1)}} \odot g^{(j)\prime}(\boldsymbol{z}^{(j)}). \tag{2.37}$$

Note that we sometimes write $\nabla_{\boldsymbol{z}^{(j)}} L$ to denote $\frac{\partial L}{\partial \boldsymbol{z}^{(j)}}$. From this expression, it is straightforward to obtain the gradient with respect to parameters $\boldsymbol{W}^{(j)}$ and bias $\boldsymbol{b}^{(j)}$:

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{b}^{(j)}} &= \frac{\partial L}{\partial \boldsymbol{z}^{(j)}} \\ \frac{\partial L}{\partial \boldsymbol{W}^{(j)}} &= \frac{\partial L}{\partial \boldsymbol{z}^{(j)}} (\boldsymbol{h}^{(j-1)})^T \end{aligned}$$

In practice, it only requires one forward and backward pass through the network to obtain the gradient $\nabla_{\boldsymbol{\theta}} L$ with respect to all parameters. In a similar way, the backpropagation algorithm can be applied to CNNs and RNNs, although the latter architecture requires to *backprop through time* (Werbos, 1990): a procedure that "unfolds" the graph over time and copies the parameters over multiple time

3. We omit the "train" superscript for brevity

steps. In recent years, many software packages have been developed that can automatically compute the gradients for any acyclic computation graph. Popular frameworks include Theano (Al-Rfou et al., 2016), PyTorch (Paszke et al., 2017), and TensorFlow (Abadi et al., 2015).

Deep learning methods are especially effective when applied to large datasets. This makes naively applying gradient-based optimization impractical, as calculating the gradient over all examples is often too computationally expensive. A straightforward way to alleviate such a computational bottleneck is to calculate an (unbiased) estimate of the gradient $\nabla_{\boldsymbol{\theta}} L^{\mathcal{B}}$ on a small mini-batch $\mathcal{B} = (\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(M)})$ of M examples. These mini-batch gradients can still provide a good update direction because there is often significant redundancy in the data points. The family of optimization algorithms that use such cheap but noisy parameter updates are known as Stochastic Gradient Descent (SGD) methods. Despite noisy gradients, SGD still converges to a (local) optimum under mild conditions (). In practice, we do not randomly sample mini-batches, but instead go over the dataset in random order. Each sweep is called an *epoch* and processes all examples in the dataset.

Unlike logistic regression, the optimization problem in deep learning is non-convex with respect to its parameters. Non-convex problems are considered to be much harder because they lead to an optimization landscape with many local minima, in which gradient-based methods do not necessarily find the optimal solution. For decades, practitioners believed that the main obstacle for training neural networks was the proliferation of local minima with a much higher error than the global minimum (Dauphin et al., 2014; Choromanska et al., 2015). Recent work, however, has shown that most critical points in high-dimensional landscapes are *saddle points*, which, in principle, are not problematic for local optimization methods. In practice, such saddle points can considerably slow training because of plateaus and ill-conditioned structures. To better handle such pathological curvature and speed up neural network training, stochastic gradient methods often incorporate pre-conditioners (Dauphin et al., 2015) or momentum (Sutskever et al., 2013). In this thesis, we will mainly use AdaM (Kingma and Ba, 2014) as stochastic optimization method, which incorporates diagonal pre-conditioning and momentum—see Algorithm 1 for pseudo-code. It is also important to initialize the neural network parameters in a sensible range so that gradients properly flow through the network. Here, we opt for the popular Glorot initialization scheme (Glorot and Bengio, 2010).

Algorithm 1 ADAM algorithm (Kingma and Ba, 2014)

Require: Initialization of $\boldsymbol{\theta}_0$
Require: learning rate sequence $(\alpha_t)_{t\geq 0}$
Require: decay coefficients $(\beta_1, \beta_2) \in [0, 1]$
Require: damping coefficient $\epsilon > 0$

$\theta \leftarrow \theta_0$
$\mu_1 \leftarrow 0$
$\mu_2 \leftarrow 0$
$t \leftarrow 0$
for epoch $1 \rightarrow$ N **do**
    Shuffle dataset $\mathcal{D}$
    while epoch not done **do**
        Sample mini-batch $\mathcal{B}$
        Calculate mini-batch gradient $\nabla_{\boldsymbol{\theta}} L^{\mathcal{B}}$
        $\boldsymbol{m} \leftarrow \beta_1 \boldsymbol{m} + (1 - \beta_1) \nabla_{\boldsymbol{\theta}} L^{\mathcal{B}}$
        $\boldsymbol{v} \leftarrow \beta_2 \boldsymbol{v} + (1 - \beta_2) \left(\nabla_{\boldsymbol{\theta}_t} L^{\mathcal{B}}\right)^{\odot 2}$
        $\hat{\boldsymbol{g}} \rightarrow \boldsymbol{g} / (1 - \beta_1^{t+1})$
        $\hat{\boldsymbol{v}} \rightarrow \boldsymbol{v} / (1 - \beta_2^{t+1})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_t \frac{\hat{\boldsymbol{g}}}{\sqrt{\hat{\boldsymbol{v}} + \epsilon}}$
        $t \leftarrow t + 1$
return $\boldsymbol{\theta}$

2.3 Reinforcement Learning

Another fundamental topic in machine learning is that of sequential decision-making. Reinforcement Learning (RL) provides a mathematical framework for studying systems that take a sequence of actions. In contrast to supervised learning algorithms, these learning systems do not rely on external human supervision but instead learn from interactions with the environment. The RL framework is quite general and covers problems in diverse application domains, ranging from manipulating robot hands (Andrychowicz et al., 2018) to playing board games (Silver et al., 2017) to learning treatment policies for chronical diseases (Shortreed et al., 2011).

Reinforcement learning formalizes the sequential-decision problem as a Markov Decision Process (MDP) (Sutton and Barto, 1998). At each time step t, an agent observes state $\boldsymbol{x}_t \in \mathbb{S}$ and can take any action $u_t \in \mathbb{A}$ available for the current state. After performing action u_t, the *environment* returns the next state $\boldsymbol{x}_{t+1}$ and gives the agent a reward $r(\boldsymbol{x}_t, u_t)$. The state transition is a random process governed

by $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, u_t)$. If the agent is given access to the state-transition function p of the MDP, we speak of a *planning* problem. For reinforcement learning problems, the agent does not know the state transition probabilities and must learn to take actions from sample trajectories. The agent's behavior is specified by its policy π, which can be either stochastic or deterministic. In this work, we consider a *stochastic policy* $\pi(\mathrm{u}_t|\boldsymbol{x}_t)$ that maps a state $\boldsymbol{x}_t$ to a probability distribution over the action set $\mathbb{U}$. The value function v quantifies the expected amount of future rewards when executing policy π from state $\boldsymbol{x}$:

$$v^{\pi}(\boldsymbol{x}) = E_{u_t \sim \pi(\boldsymbol{x}_t), \boldsymbol{x}_{t+1} \sim p(\mathbf{x}_{t+1}|\boldsymbol{x}_t, u_t)} \left[\sum_{t=1}^{\infty} \gamma^t r(\mathbf{x}_t, u_t) | \mathbf{x}_1 = \boldsymbol{x} \right], \tag{2.38}$$

where $\gamma \in [0, 1]$ is a discount factor that determines how far the agent looks into the future. In other words, the value $v^{\pi}(\boldsymbol{x})$ indicates how good it is to be in state $\boldsymbol{x}$. A related but more useful quantity for optimizing control policies is the action-value $q^{\pi}(\boldsymbol{x}, a)$, which is defined as the expected cumulative rewards for taking action u from state $\boldsymbol{x}$:

$$q^{\pi}(\boldsymbol{x}, u) = E_{u_t \sim \pi(\boldsymbol{x}_t), \boldsymbol{x}_{t+1} \sim p(\mathbf{x}_{t+1}} \left[\sum_{t=1}^{\infty} \gamma^t r(\mathbf{x}_t, u_t) | \mathbf{x}_1 = \boldsymbol{x}, u_1 = u \right] \tag{2.39}$$

A key concept in reinforcement learning is the bellman equation, which let us define the current state value in terms of the values of neighbouring states. More specifically, the expected bellman equation specifies the action value $q^{\pi}(\boldsymbol{x}, a)$ by the following recurrence relation:

$$q^{\pi}(\boldsymbol{x}, u) = r(\boldsymbol{x}, u) + \gamma \sum_{\boldsymbol{x}'} p(\boldsymbol{x}'|\boldsymbol{x}, u) \sum_{u'} \pi(\boldsymbol{x}', u') q^{\pi}(\boldsymbol{x}', u'). \tag{2.40}$$

As we will see later, many RL algorithms exploit this relationship for learning their action policy. Lastly, we define the optimal action-value as $q^*(\boldsymbol{x}, u) = \sup_{\pi} q^{\pi}(\boldsymbol{x}, u)$, which allows us to define the optimal policy by selecting the highest action value for each state:

$$\pi^*(\boldsymbol{x}) = \arg\max_{u'} q^*(\boldsymbol{x}, u'). \tag{2.41}$$

It is important to stress that the reinforcement learning problem does not assume knowledge of the underlying MDP. Instead, it aims to recover the optimal policy

from sample trajectories $\mathcal{T} = \{\tau_n = (\boldsymbol{x}_t^{(n)}, u_t^{(n)}, r_t^{(n)})_{t=1}^T\}_{n=1}^N$. The probability of generating trajectory τ depends on the policy π, the state-transition function p of the MDP, and the distribution over the start state $p(\boldsymbol{x}_1)$:

$$\pi(\tau) = p(\boldsymbol{x}_1) \prod_{t=1}^{T} \pi(u_t|\boldsymbol{x}_t) p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, u_t) \tag{2.42}$$

For policy gradients, we will define the RL objective in terms of an expectation over the trajectory distribution.

There are a few useful categorizations of RL algorithms. The first important division is between model-based and model-free methods. *Model-based algorithms* explicitly learn a model of the environment, which the algorithm utilizes at evaluation time to plan their actions. These model-based planning approaches may reduce the number of samples needed to learn an effective policy. On the other hand, *model-free methods* do not explicitly model their environment and learn a policy directly from example trajectories. Model-free approaches are more effective if the model dynamics are more complicated than learning the actual policy. In this work, we will focus on model-free RL methods.

RL algorithms are also divided into on-policy and off-policy methods. In off-policy methods, the agent learns about the target policy while trajectories are sampled from an exploratory behavior policy. On the contrary, on-policy methods update their policy with samples generated from their own policy. In general, on-policy methods are more stable during training but also less sample-efficient.

The two main approaches to training RL agents are value-based and policy-based methods. We briefly outline the two approaches in the following sub-sections.

2.3.1 Value-based methods

In value-based methods, agents learn a policy π by estimating the action values $q^\pi(\boldsymbol{x}, u)$ via a parameterized function $Q^{\boldsymbol{\theta}} : \mathbb{S} \times \mathbb{U} \to \mathbb{R}$. In *deep reinforcement learning*, the $Q^{\boldsymbol{\theta}}$ function is parameterized by a deep neural network. The most popular way of optimizing Q is through Temporal Difference (TD) learning, which bootstraps the current Q estimate using other action value estimates. That is, the algorithm aims to minimize the discrepancy between the $Q^{\boldsymbol{\theta}}$-estimate and the so-called TD target, which is derived from the expected bellman equation and

provides a 1-step approximation of the expected cumulative reward. For SARSA, an on-policy method, this leads to the following loss for a given state-action-reward-state-action quintuple $\boldsymbol{z} = (\boldsymbol{x}_t, u_t, r_t, \boldsymbol{x}_{t+1}, u_{t+1})$:

$$\ell(\boldsymbol{z}; \boldsymbol{\theta}) = (Q^{\boldsymbol{\theta}}(\boldsymbol{x}_t, u_t) - \overbrace{r_t + \lambda Q^{\boldsymbol{\theta}}(\boldsymbol{x}_{t+1}, u_{t+1})}^{\text{TD target}})^2. \tag{2.43}$$

The total loss is then defined as the expected loss over the distribution of state-action-reward-state-action pairs. We often use stochastic optimization methods, such as Adam, to optimize the parameters $\boldsymbol{\theta}$ of the Q-network. To ensure that the algorithm does not get stuck in sub-optimal behavior and keeps exploring novel action trajectories, SARSA employs an epsilon-greedy policy, which with probability ϵ takes a uniform random action and with probability $1 - \epsilon$ takes a greedy action, i.e. $\arg\max_{u'} Q^{\boldsymbol{\theta}}(\boldsymbol{x}, u')$. We often use a high ϵ at the beginning of training—so as to explore different strategies—and slowly anneal ϵ to zero in order to converge to the optimal policy. While SARSA requires sample trajectories from its own policy, *Q-learning* approximates the action-value function by trajectories generated from a behavior policy. DQN (Mnih et al., 2013) used this off-policy method to train their agent to successfully play a number of Atari games.

2.3.2 Policy-based methods

In contrast to value-based methods, policy-based methods directly parameterize the policy $\pi_{\boldsymbol{\theta}}(\mathrm{u}_t|\boldsymbol{x}_t)$. The goal of policy optimization is to find a policy that maximizes the expected return, also known as the mean value:

$$J(\boldsymbol{\theta}) = E_{\tau \sim \pi^{\boldsymbol{\theta}}(\tau)}[\sum_{t=0}^{T} \gamma^t r(\boldsymbol{x}_t, u_t)], \tag{2.44}$$

where $\gamma \in [0, 1]$ is the discount factor. Note that $\gamma = 1$ is allowed as we consider the episodic scenario (Sutton et al., 1999). To improve the policy, its parameters $\boldsymbol{\theta}$ can be updated in the direction of the gradient of the mean value:

$$\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h + \alpha_h \nabla_\theta J|_{\theta=\theta_h}, \tag{2.45}$$

where h denotes the training time-step and α_h is a learning rate such that $\sum_{h=1}^{\infty} \alpha_h = \infty$ and $\sum_{h=1}^{\infty} \alpha_h^2 < \infty$.

Thanks to the gradient policy theorem (Sutton et al., 1999), the gradient of the mean value can be estimated from a batch of trajectories $\mathcal{T}_h$ sampled from the current policy $\pi_{\boldsymbol{\theta}_h}$ by:

$$\nabla J(\boldsymbol{\theta}_h) = E_{\tau \sim \pi_{\boldsymbol{\theta}}(\tau)} \left[\sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}_h} \log \pi_{\boldsymbol{\theta}_h}(u_t|\boldsymbol{x}_t) q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}_t, u_t) \right], \tag{2.46}$$

where $q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}, u)$ is the state-action value that defines the cumulative expected reward for a given state-action pair. However, these policy gradient estimates often suffer from high variance, thereby significantly slowing down the optimization process. One common trick to obtain unbiased gradient estimates with reduced variance is to subtract a baseline value $b(\boldsymbol{x}_t)$ from the state-action value $q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}_t, u_t)$:

$$\nabla J(\boldsymbol{\theta}_h) = E_{\tau \sim \pi_{\boldsymbol{\theta}}(\tau)} \left[\sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}_h} \log \pi_{\boldsymbol{\theta}_h}(u_t|\boldsymbol{x}_t) (q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}_t, u_t) - b(\boldsymbol{x}_t)) \right]. \tag{2.47}$$

Often, we use the state-value $v^{\pi_{\boldsymbol{\theta}}}(\boldsymbol{x}_t)$ as baseline, so that the policy gradients can be rewritten in terms of the advantage function $A^{\pi_{\boldsymbol{\theta}}}(\boldsymbol{x}, u) = q^{\pi_{\boldsymbol{\theta}}}(\boldsymbol{x}, u) - v^{\pi_{\boldsymbol{\theta}}}(\boldsymbol{x})$. There are different ways to estimate the action-values $q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}, u)$. In REINFORCE (Williams, 1992)), we simply use cumulative discounted rewards of monte-carlo roll-outs, whereas actor-critic methods use temporal difference learning to estimate these values.

3 Prologue to First Article

3.1 Article Details

H. de Vries, F. Strub, S. Chandar, H. Larochelle, O. Pietquin, A. Courville. GuessWhat?! Visual Object Discovery through Multi-modal dialogue. *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.*

Personal contribution In early meetings of the IGLU project (see Context section), Aaron and Olivier came up with the general research direction of goal-oriented and multi-modal dialogue. This idea was further developed into the GuessWhat project—building on top of MS COCO, constraining the oracle, etc—in further discussions with Florian, Sarath, and myself. I've been responsible for developing the website and the data collection on mechanical turk, with help from Florian (website+data collection) and Sarath (data collection). Florian implemented significant parts of the baseline models. Sarath contributed the HRED implementation for the question generator. I've written most parts of the paper, with help from Hugo, Sarath and Florian.

3.2 Context

The GuessWhat project was developed in the context of Interactive Grounded Language Understanding (IGLU), a European-funded project that is part of the CHRIST-ERA program. IGLU is a consortium consisting of research groups from six universities (University of Montreal, University of Lille, University of Mons, University of Zaragoza, University of Sherbrooke, and KTH Royal Institute of Technology), aiming to advance computational methods for interactive and grounded language understanding by bringing together expertise from the fields of machine learn-

ing, speech processing, robotics, human-machine interaction, and neuroscience. The GuessWhat project started as a collaboration between the research groups of Aaron Courville (University of Montreal) and Olivier Pietquin (University of Lille).

At the time, the language-and-vision research community mostly focused on single-turn tasks, such as image captioning or referring expression identification. We found this setting quite unrealistic, as humans often use multiple turns when making predictions. We were interested in incorporating interactive dialogue into such multi-modal tasks.

3.3 Contributions

This papers' main contribution is the development of GuessWhat, an image-based guessing game in which two agents collaborate to discover an hidden object through a series of questions. We collect a large-scale dataset of human-played games, which enables the study of data-hungry deep learning methods for visually grounded dialogue. Concurrent to this work, Das et al. (2017a) developed the Visual Dialog task, which is similar to GuessWhat in many ways, but differs in its (implicit) goal of selecting a target *image* among a set of distractors.

3.4 Recent Developments

Since the release of GuessWhat dataset, several research groups have investigated several sub-tasks of the complete task, see for example (Shekhar et al., 2019; Zhao and Tresp, 2018; Zhang et al., 2018). Even more broadly, there has been a growing interest in the field of grounded language learning and several new tasks and datasets have been introduced, including visually-grounded instruction following (Anderson et al., 2018) and embodied question answering (Das et al., 2018).

4 GuessWhat?! Visual Object Discovery through Multi-Modal Dialogue

People use natural language as the most effective way to communicate, including when it comes to describing the visual world around them. They often need only a few words to refer to a specific object in a rich scene. Whenever such expressions *unambiguously* point to one object, we speak of a referring expression (Krahmer and Deemter, 2012). However, uniquely identifying the referred object is not always possible, as it depends on the listener's state of mind and the context of the scene. Many real life situations, therefore, require multiple exchanges before it is clear what object is referred to:

- Did you see that dog?
* You mean the one in the corner?
- No, the one that's running.
* Yes, what's up with that?

A computer vision system able to hold conversations about what it *sees* would be an important step towards intelligent scene understanding. Such systems would be more transparent and interpretable because humans may naturally interact with them, for example by asking clarifying questions about what it perceives. Still, a fundamental challenge remains: how to create models that understand natural language descriptions and ground them in the visual world.

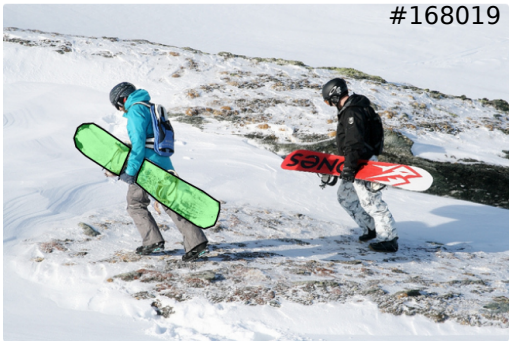
The last few years has seen an increasing interest from the computer vision community in tasks towards this goal. Thanks to advances in training deep neural networks (Goodfellow et al., 2016) and the availability of large-scale classification datasets (Lin et al., 2014; Russakovsky et al., 2015; Zhou et al., 2014), automatic object recognition has now reached human-level performance (LeCun et al., 2015). As a result, attention has been shifted toward tasks involving higher-level image understanding. One prominent example is image captioning (Lin et al., 2014), the task of automatically producing natural language descriptions of an image. Visual Question Answering (VQA) (Antol et al., 2015) is another popular task that involves answering single open-ended questions concerning an image. Closer

Questioner	**Oracle**
Is it a vase?	Yes
Is it partially visible?	No
Is it in the left corner?	No
Is it the turquoise and purple one?	Yes

Figure 4.1 – An example game. After a sequence of four questions, it becomes possible to locate the object (highlighted by a green bounding box).

to our work, the ReferIt game (Kazemzadeh et al., 2014) aims to generate a single expression that refers to one object in the image.

On the other hand, there has been a renewed interest in dialogue systems (O. Lemon and O. Pietquin, 2012; Serban et al., 2015a), inspired by the success of data-driven approaches in other areas of natural language processing (Cho et al., 2014). Traditionally, dialogue systems have been built through heavy engineering and hand-crafted expert knowledge, despite machine learning attempts for almost two decades (Levin and Pieraccini, 1997; Singh et al., 1999a). One of the difficulties comes from the lack of automatic evaluation as – contrary to machine translation – there is no evaluation metric that correlates well with human evaluation (Liu et al., 2016a). A promising alternative is goal-directed dialogue tasks (O. Lemon and O. Pietquin, 2012; Singh et al., 1999a; Weston et al., 2016; Wen et al., 2016) where agents converse to pursue a goal rather than casually chit-chat. The agent's success rate in completing the task can then be used as an automatic evaluation metric. Many tasks have recently been introduced, including the bAbI tasks (Weston et al., 2016) for testing an agent's ability to answer questions about a short story, the movie dialog dataset (Dodge et al., 2016) to assess an agent's capabilities regarding personal movie recommendation and a Wizard-of-Oz framework (Wen et al., 2016) to evaluate an agent's performance for assisting users in finding restaurants.



Is it a person?	*No*
Is it an item being worn or held?	*Yes*
Is it a snowboard?	*Yes*
Is it the red one?	*No*
Is it the one being held by the person in blue?	*Yes*

Is it a cow?	*Yes*
Is it the big cow in the middle?	*No*
Is the cow on the left?	*No*
On the right ?	*Yes*
First cow near us?	*Yes*

Figure 4.2 – Two example games in the dataset. After a sequence of five questions we are able to locate the object (highlighted by a green mask).

In this paper, we bring these two fields together and propose a novel goal-directed task for multi-modal dialogue. The two-player game, called GuessWhat?!, extends the ReferIt game (Kazemzadeh et al., 2014) to a dialogue setting. To succeed, both players must understand the relations between objects and how they are expressed in natural language. From a machine learning point of view, the GuessWhat?! challenge is the following: learn to acquire natural language by interaction on a visual task. Previous attempts in that direction (Unknown, 2016; Wen et al., 2016) do not ground natural language to their immediate environment; instead they rely on an external database through which a conversational agent searches.

The key contribution of this paper is the introduction of the GuessWhat?! dataset that contains 160,745 dialogues composed of 821,889 question/answer pairs on 66,537 images extracted from the MS COCO dataset (Lin et al., 2014). We define three sub-tasks that are based on the GuessWhat?! dataset and prototype deep learning baselines to establish their difficulty. The paper is organized as follows. First, we explain the rules of the GuessWhat?! game in Sec. 4.1. Then, Sec. 4.2 describes how GuessWhat?! relates to previous work. In Sec. 4.3.1 we highlight our design decisions in collecting the dataset, while Sec. 4.3.2 analyses many aspects of

the dataset. Sec. 4.4 introduces the questioner and oracle tasks and their baseline models. Finally, Sec. 4.5 provides a final discussion of the GuessWhat?! game.

4.1 GuessWhat?! game

GuessWhat?! is a cooperative two-player game in which both players see the picture of a rich visual scene with several objects. One player – the **oracle** – is randomly assigned an object (which could be a person) in the scene. This object is not known by the other player – the **questioner** – whose goal it is to locate the hidden object. To do so, the questioner can ask a series of yes-no questions which are answered by the oracle as shown in Fig 4.1 and 4.2. Note that the questioner is not aware of the list of objects, they can only see the whole picture.

Once the questioner has gathered enough evidence to locate the object, they notify the oracle that they are ready to guess the object. We then reveal the list of objects, and if the questioner picks the right object, we consider the game successful. Otherwise, the game ends unsuccessfully. We also include a small penalty for every question to encourage the questioner to ask informative questions. Fig 4.3 and 4.4 display a full game from the perspective of the oracle and questioner, respectively.

The oracle role is a form of visual question answering where the answers are limited to *Yes*, *No* and *N/A* (not applicable). The *N/A* option is included to respond even when the question being asked is ambiguous or an answer simply cannot be determined. For instance, one cannot answer the question "*Is he wearing glasses?*" if the face of the selected person is not visible. The role of the questioner is much harder. They need to generate questions that progressively narrow down the list of possible objects. Ideally, they would like to minimize the number of questions necessary to locate the object. The optimal policy for doing so involves a binary search: eliminate half of the remaining objects with each question. Natural language is often very effective at grouping objects in an image scene. Such strategies depend on the picture, but we distinguish the following types:

Spatial reasoning We group objects spatially within the image scene. One may use absolute spatial information – *Is it on the bottom left of the picture?* – or relative spatial location – *Is it to the left of the blue car?*.

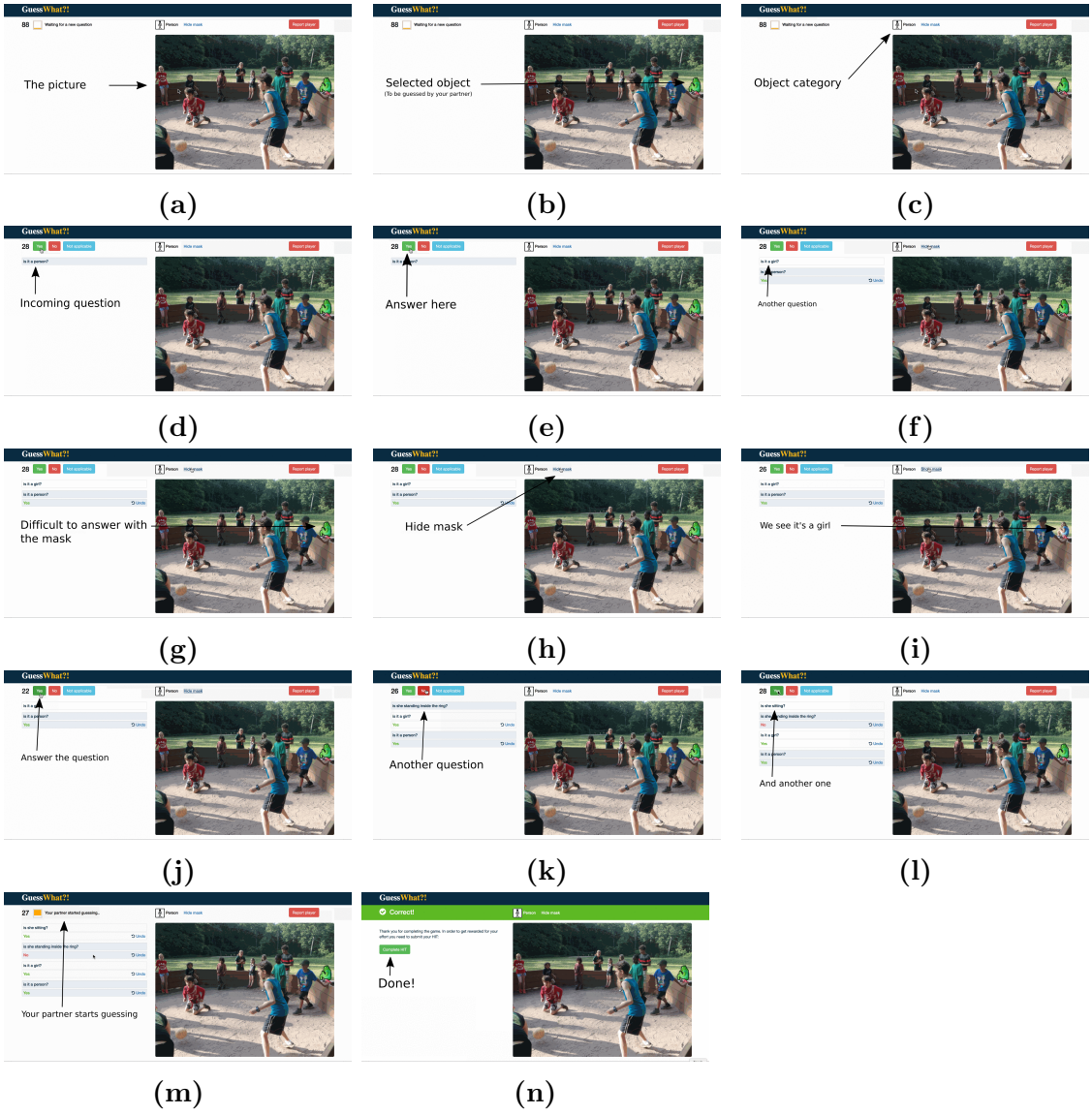


Figure 4.3 – An example game from the perspective of the oracle. Shown from left to right and top to bottom.

Visual properties We group objects by their size – *Is it big?*, shape – *Is it square?* – or color – *Is it blue?*.

Object taxonomy We can use the hierarchical structure of object categories, i.e. taxonomy, to group objects e.g. *Is it a vehicle?* to refer to both cars and trucks.

Interaction We group objects by how we interact with them – *Can you drive*

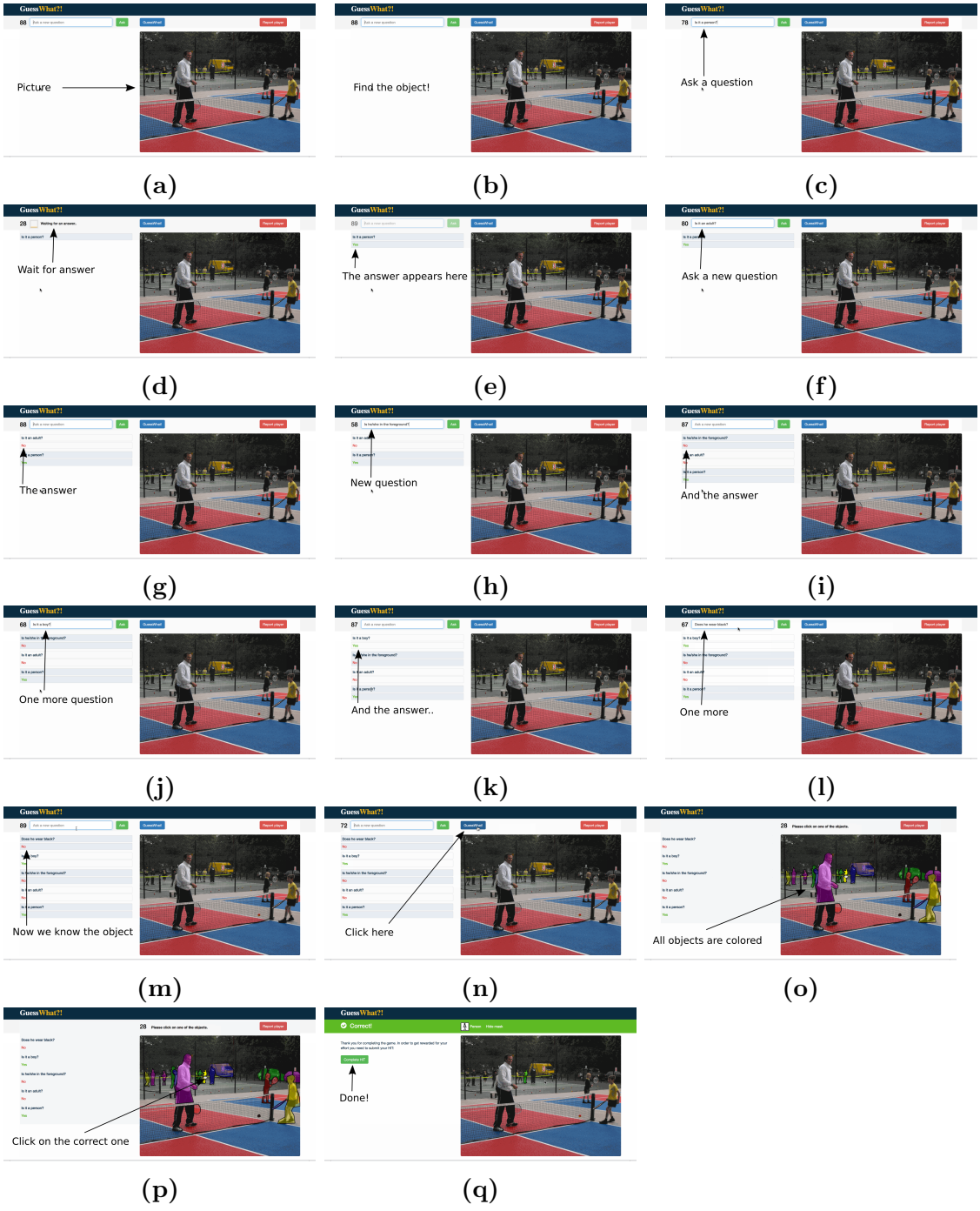


Figure 4.4 – An example game from the perspective of the questioner. Shown from left to right and top to bottom.

it?.

The goal of the GuessWhat?! task is to enable machines to understand natural

descriptions and ground them into the visual world. Note that such higher-level reasoning only occurs when the scene is rich enough i.e. when there are enough objects in the scene. People otherwise tend to fall back to a linear search strategy by simply enumerating objects (often by their category names).

4.2 Related work

The GuessWhat?! game and the data collected from it present opportunities for the extension of current research on image captioning, visual question answering and dialogue systems. In the following, we describe previous work in these areas and relate them to the open challenges offered by GuessWhat?!. We also mention other relevant work on dataset collection.

Image captioning Our work builds on top of the MS COCO dataset (Lin et al., 2014) which consists of 120k images with more than 800k object segmentations. In addition, the dataset provides 5 captions per image which initiated an explosion of interest from the research community into generating natural language descriptions of images. Several methods have been proposed (Karpathy and Fei-Fei, 2015; Vinyals et al., 2015; Xu et al., 2015), all inspired by the encoder-decoder approach Cho et al. (2014); Sutskever et al. (2014) that has proven successful for machine translation. Image captioning research uncovered successful approaches to automatically generate coherent, factual statements about images. Modeling the interactions in GuessWhat?! requires instead to model the process of asking useful questions about images.

VQA datasets Visual Question Answering (VQA) tasks form another well known extension of the captioning task. They instead require answering a question given a picture (*e.g. "How many zebras are there in the picture?", "Is it raining outside?"*). Recently, the VQA challenge (Antol et al., 2015) has provided a new dataset far bigger than previous attempts (Geman et al., 2015; Malinowski and Fritz, 2014) where, much like in GuessWhat?!, questions are free-form. An extensive body of work has followed from this publication, largely building on the image captioning literature (Agrawal et al., 2016; Lu et al., 2016; Shih et al., 2016; Yang et al.,

Figure 4.5 – Samples illustrating the difference between GuessWhat?! and ReferIt games. As both dataset are constructed on top of MS COCO, we picked identical objects (and images).

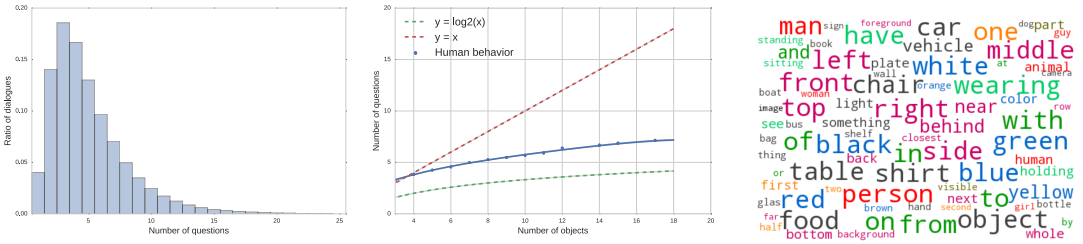
2016a). Unfortunately, many of these advanced methods were shown to marginally improve on simple baselines (Jabri et al., 2016). Recent work (Agrawal et al., 2016) also reports that trained models often report the same answer to a question irrespective of the image, suggesting that they largely exploit predictive correlations between questions and answers present in the dataset. The GuessWhat?! game and dataset attempt to circumvent these issues. Because of the questioner's aim to locate the hidden object, the generated questions are different in nature: they naturally favour spatial understanding of the scene and the attributes of the objects within it, making it more valuable to consult the image. Besides, it only contains binary questions, whose answers we find to be balanced and has twice more questions on average per picture.

Goal-directed dialogue GuessWhat?! is also relevant to the goal-directed dialogue research community. Such systems are aimed at collaboratively achieving a goal with a user, such as retrieving information or solving a problem. Although goal-directed dialogue systems are appealing, they remain hard to design. Thus, they are usually restricted to specific domains such as train ticket sales, tourist information or call routing (Pietquin and Dutoit, 2006; Singh et al., 1999a; Young

et al., 2013). Besides, existing dialogue datasets are either limited to fewer than 100k example dialogues (Dodge et al., 2016), unless they are generated with template formats (Dodge et al., 2016; Wen et al., 2016; Weston et al., 2016) or simulation (Pietquin and Hastie, 2013a; Schatzmann et al., 2006) in which case they don't reflect the free-form of natural conversations. Finally, recent work on end-to-end dialogue systems fail to handle dynamical contexts. For instance, (Wen et al., 2016) intersects a dialogue with an external database to recommend restaurants. In contrast, GuessWhat?! dialogues are heavily grounded by the images. The resulting dialogue is highly contextual and must be based on the content of the current picture rather than an external database. Thus, to the best of our knowledge, the GuessWhat?! dataset marks an important step for dialogue research, as it is the first large scale dataset that is both goal-oriented and multi-modal.

Human computation games GuessWhat?! is in line with Von Ahn's seminal work on human computation games (Ahn and Dabbish, 2004; Ahn et al., 2006) who showed that games are an effective way to gather labeled data. The first ESP game (Ahn and Dabbish, 2004) was developed to collect image tags, and was later extended to Peekaboom (Ahn et al., 2006) to gather object segmentations. These games were developed more than a decade ago, when object recognition was in its infancy and served a different purpose than GuessWhat?!.

ReferIt Probably closest to our work is the ReferIt game (Kazemzadeh et al., 2014; Mao et al., 2015; Yu et al., 2016). In this game, one player observes an annotated object in a scene, for which they need to generate an expression that refers to it (*e.g. ̈the man wearing the white t-shirt¨*). The other player then receives this expression and subsequently clicks on the location of the object within the image. The original dataset (Kazemzadeh et al., 2014) uses the IMAGEClef dataset (Escalante et al., 2010), while three recent extensions (Mao et al., 2015; Yu et al., 2016) were built on top of MS COCO. All three databases select images with only $2-4$ objects of the *same* category. In contrast, GuessWhat?! picks images with $3-20$ objects without further restrictions on the object class, and thus contains three times more images than the ReferIt datasets. To further investigate the difference between ReferIt and GuessWhat?!, we compare three samples for the *same* selected object in Fig 4.5. While ReferIt directly locates the object with a single



(a) **(b)** **(c)**

Figure 4.6 – (a) Number of questions per dialogue (b) Number of questions per dialogue vs the number of objects within the picture (c) Word cloud of GuessWhat?! vocabulary with each word proportional to its frequency. Words are colored based on a hand-crafted clustering. Uninformative words such as *"it"*, *"is"* are manually removed.

expression, GuessWhat?! iteratively narrows down the object by means of positive and *negative* feedback on questions. We also observe that GuessWhat?! dialogues favor more abstract concepts, such as "*Is it edible?*" or "*Is it on oval plate?*" than ReferIt.

I spy The robotics community has explored variants of the "I spy" (Vogel et al., 2010; Parde et al., 2015; Thomason et al., 2016) game for grounded language acquisition. In one of the scenario, the robot is first shown a set of objects that it can pokes thought some predefined actions (grasp, hold, look etc.) while recording several modalities (VGG features, sounds, motor joint positions etc.). As a second step, the human describes one of the object oh his choice and the robot must guess the referenced object. However, the robotic constraints force the game setting to remain limited to small number of objects (4 32), and the training are mainly done in a on-line fashion with no released dataset.

4.3 GuessWhat?! Dataset

4.3.1 Data collection

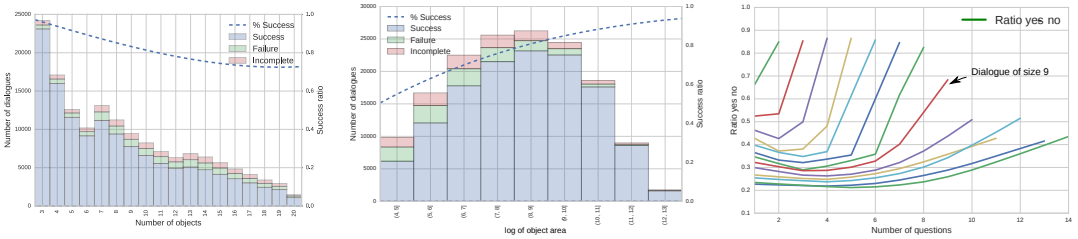
Images We use a subset of the training and validation images and objects of the MS COCO dataset (Lin et al., 2014). We first discard objects that are too small (area $< 500\text{px}^2$) to be decently located by a human observer. Then, we only keep images containing three to twenty objects, to avoid trivial or overly complicated images. In total, we keep 77,973 images with 609,543 objects. We verified that this selection does not significantly alter the original dataset distribution.

Amazon Mechanical Turk The data collection was crowd-sourced on Amazon Mechanical Turk (AMT) (Buhrmester et al., 2011). We created two separate tasks – known as HITs on AMT – for the questioner and oracle roles, and rewarded the questioner slightly more than the oracle. We ensured the quality of the data collection by several means. First, the workers had to go through a qualification round which consisted of successfully completing 10 games while producing fewer than 4 mistakes or disconnects. After qualification, HITs continue to consist of a batch of 10 successful games. We incentivize the worker to produce as many successful dialogues in a row by providing bonuses for making fewer mistakes. Secondly, players could report on each other and players were banned after a certain number of reports. Thus, players were incentivized to cooperate. In the end, we only kept dialogues from qualified people and successful dialogues from the qualification round. In contrast to traditional dataset collection, our game requires an interactive session between two players. Fortunately, we found that the GuessWhat?! game was highly engaging. A total of more than 10K people participated in our HITs, and our top ten participants played over $2,000$ games each.

Since questions were manually typed, they could contain spelling mistakes. Thus, we retrieved all questions containing words that do not occur in an English dictionary and manually corrected the 1000 most common words. For the remaining 30k questions, we created two HITs that to correct the spelling mistakes.

	Full	Finished	Success
# dialogues	160,745	152,000	135,400
# questions	821,889	780,391	672,940
# words	3,985,368	3,788,167	3,254,793
# voc. size	11,464	11,259	10,637
# voc. size (3+)	5,444	5,324	5,013
# images	66,537	66,161	63,642
# segmented objects	535,723	531,847	505,599
# selected objects	134,073	131,415	117,513

Table 4.1 – GuessWhat?! statistics split by dataset types.



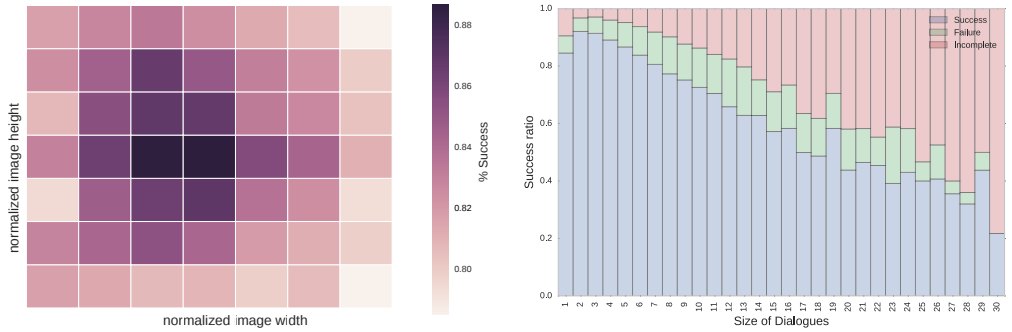
(a) **(b)** **(c)**

Figure 4.7 – (a-b) Histogram of absolute/relative successful dialogues with respect to the number of objects and the size of the objects, respectively. (c) Evolution of answer distribution clustered by the dialogue length

4.3.2 Data analysis

In the following, we explore properties of the data we collected using the GuessWhat?! game. We provide global statistics, examine the vocabulary used by the questioners and highlight the relationship between properties of objects to guess and the odds of having a successful dialogue.

Dataset statistics The raw GuessWhat?! dataset is composed of 160,745 dialogues containing 821,889 question/answer pairs on 66,537 unique images with 1,385,197 objects and 134,073 unique selected objects. The answers are respectively 52.2% *no*, 45.6% *yes* and 2.2% *N/A*. On average, there are 5.2 questions per



(a)

(b)

Figure 4.8 – (a) Heatmap of the success ratio with respect to the spatial location within the picture. (b) Histogram of the success ratio relative to the dialogue length.

dialogue and 2.3 dialogues per image. The dialogues contain 3,985,368 word tokens in total, making up 11,464 different words with at least one occurrence and 5,444 words with at least 3 occurrences. Moreover, 84.2% of the dialogues are successful, 10.3% are unsuccessful and 5.5% are not completed (disconnection, timeout etc.). Thus, different subsets co-exist in the GuessWhat?! dataset, we will refer to the dataset as full, finished and successful when we include all the dialogues, all finished dialogues (successful and unsuccessful) or only successful dialogues, respectively. The previous statistics are broken down into dataset types in Tab 4.1.

Question distributions To get a better understanding of the GuessWhat?! games, we show the number of questions within a dialogue and the average number of questions given the number of objects within a image in Fig 4.6. First, the number of questions within a dialogue decreases exponentially, as players tend to shorten their dialogues to speed up the game (and therefore maximize their gains). More interestingly, we observe that the average number of questions given the number of objects within an image appears to follow a function that grows

at a rate between logarithmically and linearly. A questioning strategy of simply listing objects (e.g. "*is it the chair*", etc.) would imply linear growth in the number of questions, while the optimal binary search strategy would imply logarithmic growth. Thus the human questioners seem to imply a strategy that is somewhere in between. We conjecture three reasons why humans do not achieve the optimal search strategy. First, the questioner does not have access to the ground truth list of objects in the picture, and might, therefore, overestimate the number of objects. Second, some humans tend to favor a linear search strategy. Finally, the questioner may ask additional questions to confirm that he has located the right object. This can be important in the presence of possible oracle errors.

Vocabulary To gain insight into the vocabulary used by the questioner, we compute the frequency of words in the GuessWhat?! corpus and display the most frequent words as a word cloud in Fig 4.6c. Several key words clearly stand out. As explained in Sec. 4.1, some of those key words refer to abstract object properties such as *person* or *object*, spatial locations such as *right/left* or *side* and visual features such as *red/black/white*. Furthermore, prepositions are also heavily used to express relationships between objects. To better understand the sequential aspect of the questions, we study the evolution of the vocabulary at each question round. We observe that questioners use abstract object properties such as *human/object/furniture* only at the beginning of the dialogues, and quickly switch to either spatial or visual terms such as *left/right*, *white/red* or *table,chair*.

Elements of success To investigate whether certain object properties favour success, we compute the success ratio of dialogues relative to: the size of the unknown objects in Fig 4.7b, the number of objects within the images in Fig 4.7a, the object category, the location of objects within the images and the size of the dialogues in Fig 4.9, Fig 4.8a and Fig 4.8b, respectively. As one may expect, the more complex the scene is, the lower the success rate is. When there are only 3 objects, the questioner has 95% success rate, while this ratio drops to around 70% with 20 objects. Similarly, big objects are almost always found while the smallest one are only found 60% of the time. Questioners easily find objects in the middle of the picture but have more difficulties to find them on the border. Finally, objects from categories that are often grouped together, e.g. *bananas* or *books*, have a lower

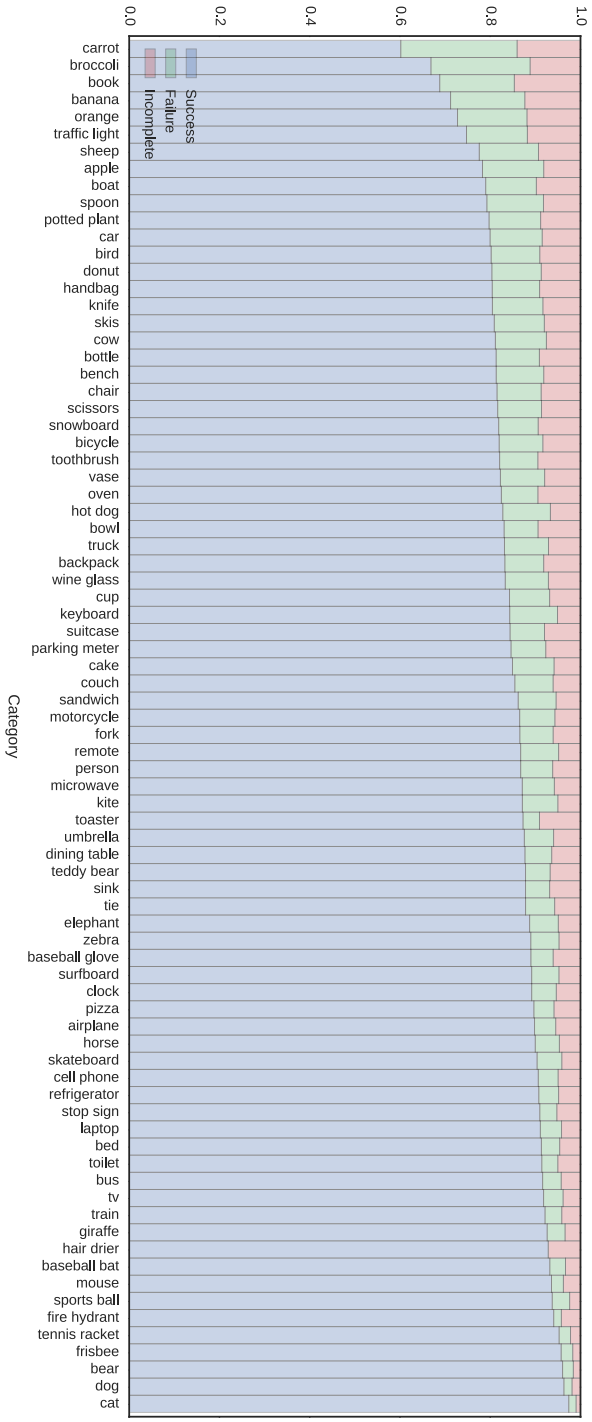


Figure 4.9 – Histogram of success ratio broken down per object category.

success rates.

Miscellaneous In Fig 4.7c we break down the ratio of yes-no answers within the dialogues. While the first yes-no answers are balanced for small dialogues, they often terminate with a final *yes*. In contrast, long dialogues often start with a higher proportion of negative answers which slowly decrease during the exchange.

4.3.3 Dataset release

We split the GuessWhat?! dataset by randomly assigning 70%, 15% and 15% of the *images* and its corresponding dialogues to the training, validation and test set. This way of dividing the data ensures that we evaluate performance on images not seen during training. The GuessWhat?! dataset and the source code is available at https://guesswhat.ai/download.

4.4 Baselines

We now empirically investigate the difficulty of the oracle and questioner tasks. To do so, we trained reasonable baselines for each task and measured their performance.

Formally, a GuessWhat?! game revolves around an image $\mathbf{I} \in \mathbb{R}^{W \times H \times 3}$ containing a set of K segmented objects $\{o_1, \ldots, o_K\}$. Each object o_k is assigned an object category $c_k \in \{1, \ldots, C\}$ and has a pixel-wise segmentation mask $\boldsymbol{S}_k \in \{0, 1\}^{W \times H}$ to specify its location and size. The game further consists of a sequence of questions and answers $D = \{\boldsymbol{q}_1, a_1, \ldots, \boldsymbol{q}_J, a_J\}$, produced by the questioner and oracle. We will use $(\boldsymbol{q}, a)_{1:j}$ to refer to the first j question-answer pairs of a dialogue. Each question $\boldsymbol{q}_j = (w_i^j)_{i=1}^{I_j}$ is a sequence of length I_j with each token w_i^j taken from a predefined vocabulary V. Each answer is either *Yes*, *No* or *N/A*, i.e. $a_j \in \{\text{Yes, No, N/A}\}$. Finally, the oracle has access to the identity of the correct object o^*, and the prediction of the questioner will be denoted as o^{predict}.

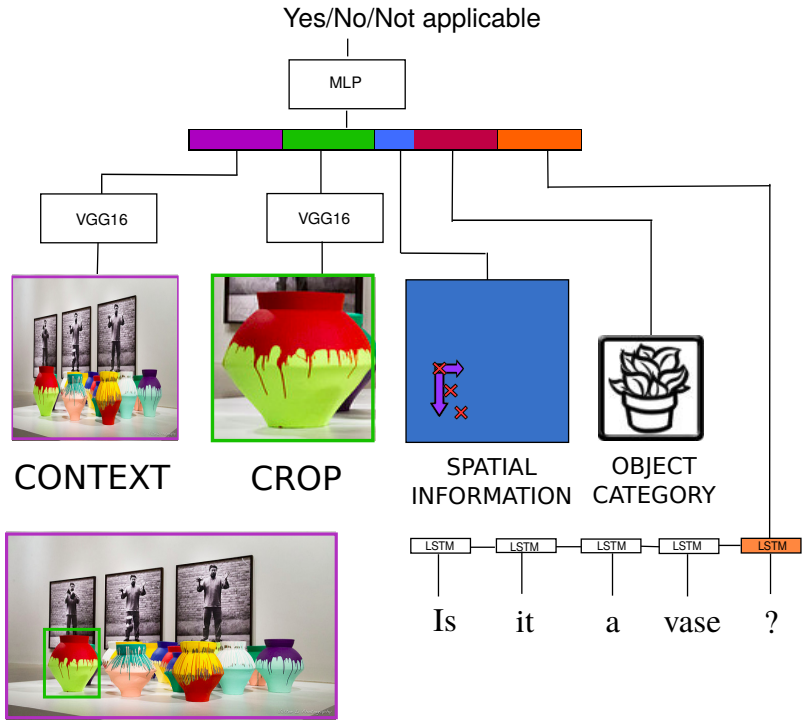


Figure 4.10 – An schematic overview of the "Image + Question + Crop + Spatial + Category" oracle model.

4.4.1 Oracle baselines

The oracle task requires to produce a yes-no answer for any object within a picture given a natural language question. We first introduce our model and then outline its results to get a better understanding of the GuessWhat?! dataset.

Model We propose a simple neural network based approach to this model, illustrated in Fig 6.1. Specifically, we use an appropriate neural network architecture to embed each of the following information: the image $\mathbf{I}$, the cropped object from $\boldsymbol{S}$, its spatial information, its category c and the current question $\boldsymbol{q}$. These embeddings are then concatenated as a single vector and fed as input to a single hidden layer MLP that outputs the final answer distribution using a softmax layer. Finally, we minimize the cross-entropy error during the training and report the classification error at evaluation time.

The details on how we compute the embeddings are as follows.To embed the full image, it is rescaled to a 224 by 224 image and is passed through a pre-trained VGG network to obtain its FC8 features. As for the selected object, it is first cropped

Model	Train err	Val err	Test err
Dominant class (no)	47.4%	46.2%	50.9%
Question	40.2%	41.7%	41.2%
Image	45.7%	46.7%	46.7%
Crop	40.9%	42.7%	43.0%
Question + Crop	22.3%	29.1%	29.2%
Question + Image	37.9%	40.2%	39.8%
Question + Category	23.1%	25.8%	25.7%
Question + Spatial	28.0%	31.2%	31.3%
Question + Category + Spatial	17.2%	21.1%	**21.5%**
Question + Category + Crop	20.4%	24.4%	24.7%
Question + Spatial + Crop	19.4%	26.0%	26.2%
Question + Category + Spatial + Crop	16.1%	21.7%	22.1%
Question + Spatial + Crop + Image	20.7%	27.7%	27.9%
Question + Category + Spatial + Image	19.2%	23.2%	23.5%

Table 4.2 – Classification errors for the oracle baselines on train, valid and test set. The best performing model is "Question + Category + Spatial" and refers to the MLP that takes the question, the selected object class and its spatial features as input.

by finding the smallest rectangle that encapsulates it, based on its segmentation mask. We then rescale the crop to a 224 by 224 square, before obtaining its FC8 features from the pre-trained VGG network. Although we could use the mask to drop out pixels around the selected object, we keep the crop as is since pre-trained VGG networks are exposed to such background noise during their training.

We also embed the spatial information of the crop, to help locate the cropped object within the whole image. To do so, we follow the approach of (Hu et al., 2016; Yu et al., 2016) and extract an 8-dimensional vector of the location of the

bounding box:

$$\boldsymbol{x}_{spatial} = [x_{min}, y_{min}, x_{max}, y_{max}, x_{center}, y_{center}, w_{box}, h_{box}] \tag{4.1}$$

where w_{box} and h_{box} denote the width and height of the bounding box, respectively. We normalize the image height and width such that coordinates range from -1 to 1, and place the origin at the center of the image. As for the object category, we convert its one-hot class vector into a dense category embedding using a learned look-up table. Finally, the embedding of the current natural language question $\boldsymbol{q}$ is computed using an Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) where questions are first tokenized by using the word punct tokenizer from the python nltk toolkit (Bird et al., 2009). For simplicity, we decided to ignore the question-answer pairs history $(\boldsymbol{q}, a)_{1:j-1}$ in our oracle baseline.

Training setting We train all oracle models on the full dataset. During training, we keep the parameters of the VGG network fixed, and optimize the LSTM, object category/word look-up tables and MLP parameters by minimizing the negative log-likelihood of the correct answer. We use ADAM (Kingma and Ba, 2014) for optimization and train for at most 15 epochs. We use early stopping on the validation set, and report the train, valid and test error.

Results We report results for several oracle models using a different set of inputs in Table 4.2. We name the model after the input we feed to it. For instance, (Question+Category+Spatial+Image) refers to the network fed with the question $\boldsymbol{q}$, the object category c, the spatial features $\boldsymbol{x}_{spatial}$ and the full image $\mathbf{I}$.

Because the GuessWhat?! dataset is fairly balanced, simply outputting the most common answer in the training set – No – results in a high 50.8% error rate. Solely providing the image or crop features barely improves upon this result. Only using the question slightly improves the error rate to 41.2%. We speculate that this small bias comes from questioners that refer to objects that are never segmented or over-represented categories. As hoped, we observe that the error rate significantly drops ($< 31\%$) when we finally feed information on the object to guess (crop, spatial or category) to the model. We find that crop and category information are redundant: the (Question+Category) and (Question+Crop) model achieve respectively 29.2% and 25.7% error, while the combined model (Question+Category+Crop) achieves

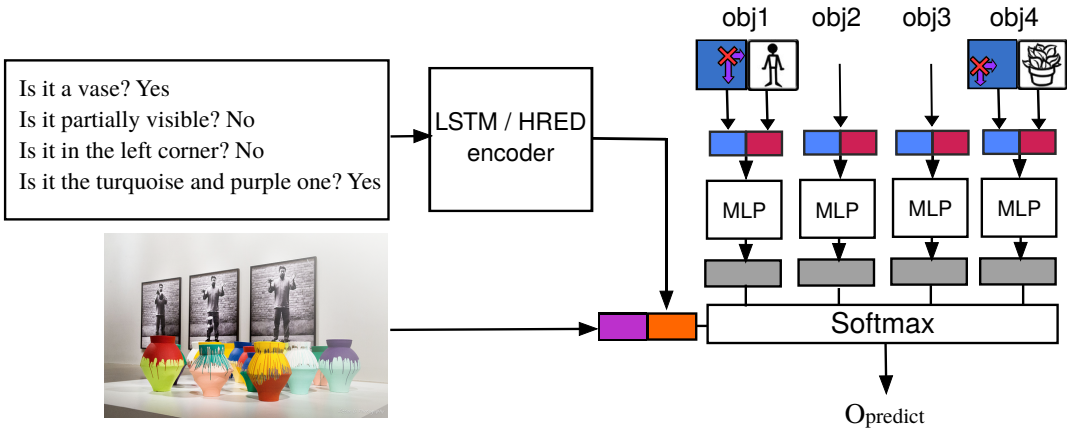


Figure 4.11 – Overview of the guesser model for an image with 4 segmented objects. The weights are shared among the MLPs, this allows for an arbitrary number of objects.

24.7%. In general, we expect the object crop to contain additional information, such as color information, beside the object class. However, we find that the object category outperforms the object crop embedding. This might be partly due to the imperfect feature extraction from the crops. Finally, our best performing model combines object category and its spatial features along with the question.

4.4.2 Questioner baselines

Given an image, the questioner must ask a series of questions and guess the correct object. We separate the questioner task into two different sub-tasks that are trained independently:

Guesser Given an image **I** and a sequence of questions and answers $(\boldsymbol{q}, a)_{1:J}$, the Guesser predict the correct object o^* from the set of all objects O.

Question Generator Given an image **I** and a sequence of j questions and answers $(vq, a)_{1:j}$, the Question Generator produce a new question $\boldsymbol{q}_{j+1}$.

In general, one also needs a module to determine when to start guessing the object (and stop asking questions). In our baseline, we bypass this issue by fixing the number of questions to 5 for the question generator model.

Guesser The role of the guesser model is to predict the correct object. To do so, the guesser has access to the image, the dialogue and the list of objects in the image. We encode the image by extracting its FC8 features from VGG16 network.

Model	Train err	Val err	Test err
Human	10.8%	11.1%	11.1%
Random	82.9%	82.9%	82.9%
LSTM	27.9%	37.9%	**38.7**%
HRED	32.6%	38.2%	39.0%
LSTM+VGG	26.1%	38.5%	39.5%
HRED+VGG	27.4%	38.4%	39.6%

Table 4.3 – Classification errors for the guesser baselines on train, valid and test finished set.

A dialogue of a GuessWhat?! game is a sequence on two different levels: there is a variable number of question-answer pairs where each question in turn consists of a variable-length sequence of tokens. This can be encoded into a fixed size vector by using either an LSTM encoder (Hochreiter and Schmidhuber, 1997) or an HRED encoder (Serban et al., 2015b). While the LSTM encoder considers the dialogue as one flat sequence, HRED explicitly models the hierarchy by two different Recurrent Neural Networks (RNN). First, an encoder RNN creates a fixed-size representation of a question or answer by reading in its tokens and taking the last hidden state of the RNN. This representation is then processed by the context RNN to obtain a representation of the current dialogue *state*. For both models, we concatenate the image and dialogue features and do a dot-product with the embedding for all the objects in the image, followed by a softmax to obtain a prediction distribution over the objects. Given the best performance of the "Question+Category+Spat" oracle model, we represent objects by their category and their spatial features. More precisely, we concatenate the 8-dimensional spatial representation (see Eq. 4.1) and the object category look-up and pass it through an MLP layer to get an embedding for the object. Note that the MLP parameters are shared to handle the variable number of objects in the image. See Fig 4.11 for an overview of the guesser with HRED and LSTM.

Table 4.3 reports the results for the guesser baselines using human-generated dialogues. As a first baseline, we report the performance of a random guesser which does not use the dialogue information. We split the guesser results based on whether they use the VGG features or not. In general, we find that including VGG features does not improve the performance of the LSTM and HRED models. We

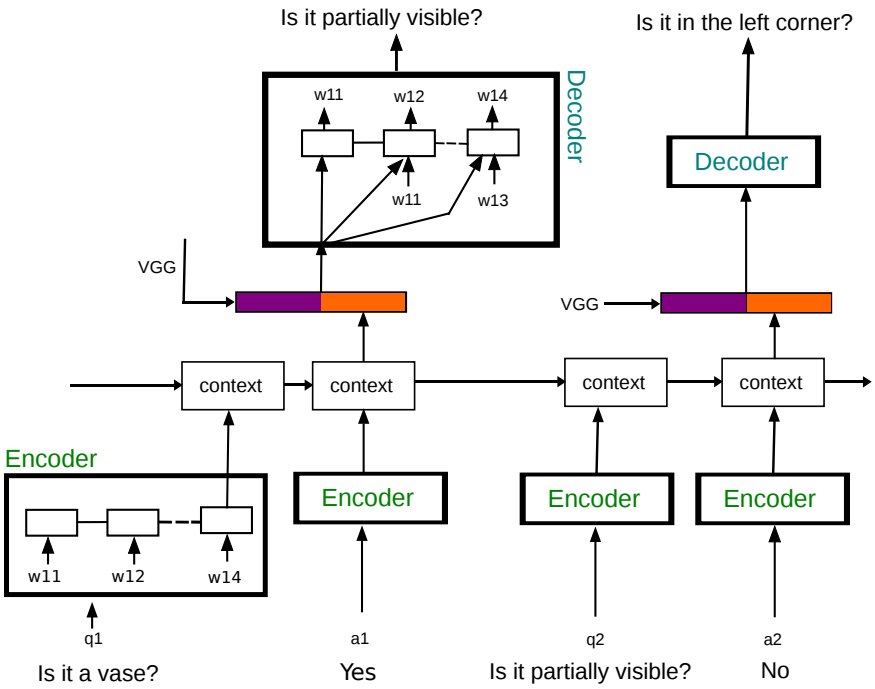


Figure 4.12 – HRED model conditioned on the VGG features of the image. To avoid clutter, we here only show the part of the model that defines a distribution over the third question given the first two questions, its answers and the image $P(\boldsymbol{q}_2|(\boldsymbol{q}, a)_{<2}, I)$. The complete HRED model models the distribution over all questions.

hypothesize that the VGG features are a too coarse representation of the image scene, and that most of the visual information is already encoded in the question and the object features. Surprisingly, we find LSTMs to perform slightly better than the sophisticated HRED models.

Question Generator The question generation task is hard for several reasons. First, it requires high-level visual understanding to ask meaningful questions. Second, the generator should be able to handle long-term context to ask a sequence of relevant questions, which is one of the most challenging problems in dialogue systems. Additionally, we evaluate the question generator using the imperfect oracle and imperfect guesser, which introduces compounding errors. Hierarchical recurrent encoder decoder (HRED) Serban et al. (2015b) is the current state of the art method for natural language generation tasks. We extend this model by conditioning on the VGG features of the image as illustrated in Fig 4.12. Finally, we train

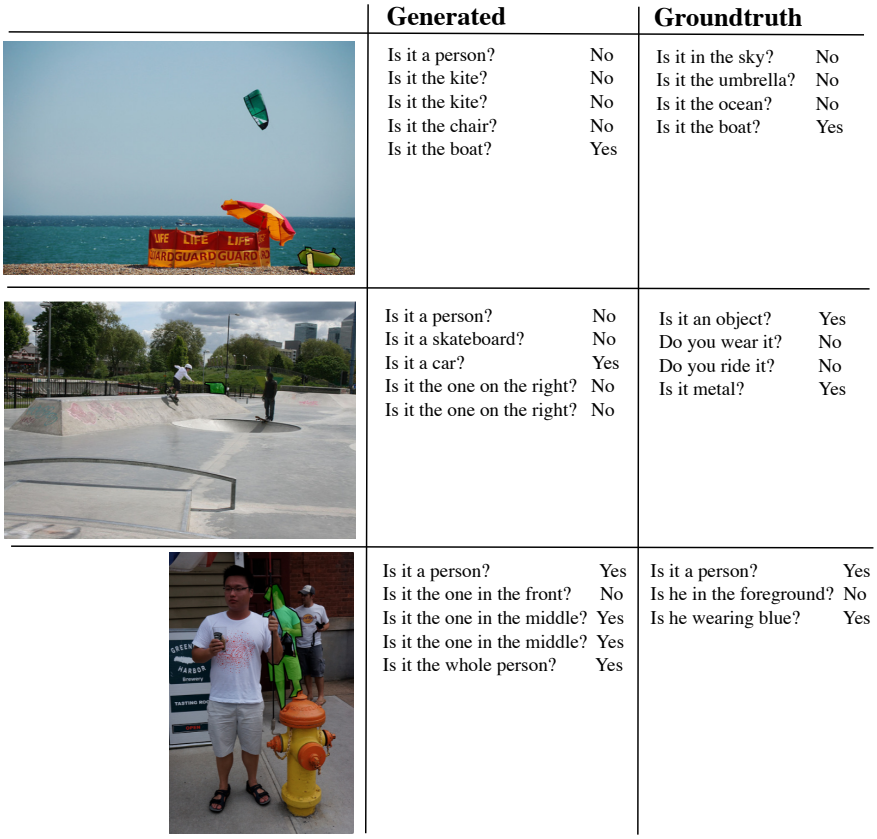


Figure 4.13 – Three samples of QGen+GT model for which the correct object was predicted.

our proposed model by minimizing the conditional negative log-likelihood:

$$
\begin{aligned}
-\log P(\boldsymbol{q}_{1:J}|a_{1:J}, \mathbf{I}) &= -\log \prod_{j=1}^{J} P(\boldsymbol{q}_j|(\boldsymbol{q}, a)_{1:j-1}, \mathbf{I}), \\
&= -\sum_{j=1}^{J}\sum_{i=1}^{I_j} \log P(w_i^j|w_{1:i-1}^j, (\boldsymbol{q}, a)_{1:j-1}, \mathbf{I}).
\end{aligned} \tag{4.2}
$$

with respect to the described parameters. At test time, we use a beam-search to approximately find the most probable question $\boldsymbol{q}_j$. Evaluating the questioner model requires a pre-trained oracle and a pre-trained guesser model. We use our questioner model to first generate a question which is then answered by the oracle model. We repeat this procedure 5 times to obtain a dialogue. We then use the best performing guesser model to predict the object and report its error as the metric for the QGEN model. Since we use ground truth answers during the QGEN

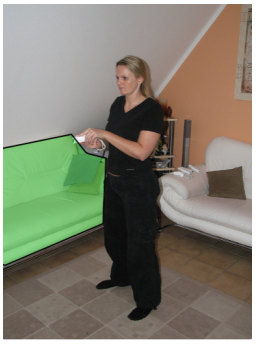
	Generated	Groundtruth
	Is it the cat? No Is it the cat? No Is it the chair? No Is it the book? No Is it the book? No	Is it an animal? No Is it a device? Yes Is it silver in color? Yes
	Is it a person? No Is it a remote? No Is it the chair? Yes Is it the one on the right? No Is it the one next to the right? No	Is it a person? No Is it a couch? Yes Does the couch have two pillows on it? Yes
	Is it a person? Yes Is it the guy in the front? No Is it the guy in the middle? No Is it the guy in the middle? No Is it the guy in the middle? No	Is it a person? Yes Is it in the foreground? No Is it on a screen? Yes

Figure 4.14 – Three dialogue samples of QGen+GT model for which the wrong object was predicted.

training while we use oracle answers at test time, there is a mismatch between the training and testing procedure. This can be avoided by using the oracle answers also during training time. We call these models QGEN+GT and QGEN+ORACLE respectively.

Table 4.4 shows the results. A guesser based on human generated dialogues achieves 38.7% error. The Question Generator models achieve reasonable performance which lies in between the random performance and the performance of the guesser on human dialogues. We observe that using the Oracle's answers while training the Question Generator introduces additional errors which significantly deteriorates performance. Some example dialogues generated by the QGen+GT

Model	Error
Human generated dialogue	38.7%
QGen+GT	53.2%
QGen+ORACLE	66.0%
Random	82.9%

Table 4.4 – Test error for the question generator models (QGEN) based on VGG+HRED(FT) guesser model. We here report the accuracy error of the guesser model fed with the questions from the QGEN model.

model are shown in Fig. 4.13 and 4.14.

4.5 Discussion

We introduced the GuessWhat?! game, a novel framework for multi-modal dialogue. To the best of our knowledge, we present the first large-scale dataset involving images and dialogue. A wide range of challenges may arise from this union as they rely on different fields of machine learning such as natural language understanding, generative models or computer vision. GuessWhat?! turns out to be an engaging game that greatly decreases the cost for collection of a big dataset required for modern algorithms. As a second contribution, we introduced three tasks based on the questioner and oracle role. In each case, we prototyped a neural architecture as a first baseline. We analyzed these results and presented a quantitative description of the GuessWhat?! dataset.

We believe GuessWhat?! could allow for a myriad of other applications that may either be based on the game itself or extending the database to other tasks. For instance, it can be interesting to compute a confidence interval before proceeding to the final guess. Differently, GuessWhat?! could be a test bed for one-shot learning (Fei-Fei et al., 2006) of guessing new object categories, transfer learning on line-drawing images (Castrejon et al., 2016) or using questions from another language. Thus, the GuessWhat?! dataset offers an opportunity to develop original machine learning tasks upon it.

5 Prologue to Second Article

5.1 Article Details

F. Strub, H. de Vries, J. Mary, A. Courville, O. Pietquin. End-to-end Optimization of Visually Grounded Dialogue. *International Joint Conference on Artificial Intelligence (IJCAI), 2017.*

Personal contribution We designed the GuessWhat task with the idea of applying deep reinforcement learning in mind. Hence most credit for this direction should be attributed to Olivier and Aaron. Florian and I both contributed equally to the implementation of the algorithms while sitting next to each other in the SeQuel lab in Lille. The paper was written by Florian and myself, with help from Bilal and Olivier.

5.2 Context

This a natural follow-up on the original GuessWhat paper which established supervised baselines for the three sub-tasks of the full problem. At the time, there was a significant interest in deep reinforcement learning methods, mostly due to their success in solving video games. Here, we investigate how well these learning methods work in the context of visually grounded dialogue.

5.3 Contributions

This paper investigates deep reinforcement learning methods for the questioner task of GuessWhat. The goal-oriented nature of the task allows us to specify a

reasonable reward function training such RL agents: whether or not the generated sequence of questions lead the guesser model to select the correct object. We show that RL agents achieve much better task performance than supervised baselines. However, we find its questions are less linguistically diverse and tailored towards the capabilities of the question-answering model.

5.4 Recent Developments

Concurrent to this work, (Das et al., 2017c) applies deep reinforcement learning to the visual dialog task. For the GuessWhat task, several recent works improved upon the RL algorithms for the questioner agent, most notably (Zhang et al., 2018; Zhao and Tresp, 2018). Other work (Shekhar et al., 2019) has argued that the evaluation of goal-oriented dialogue agents should go beyond their task success and, to that end, propose a simple grammar to analyze the type of questions they generate.

6 End-to-end Optimization of Goal-driven and Visually Grounded Dialogue Systems

Ever since the formulation of the Turing Test, building systems that can meaningfully converse with humans has been a long-standing goal of Artificial Intelligence (AI). Practical dialogue systems have to implement a management strategy that defines the system's behavior, for instance to decide when to provide information or to ask for clarification from the user. Although traditional approaches use linguistically motivated rules (Weizenbaum, 1966), recent methods are data-driven and make use of Reinforcement Learning (RL) (Lemon and Pietquin, 2007). Significant progress in Natural Language Processing via Deep Neural Nets (Bengio et al., 2003) made neural encoder-decoder architectures a promising way to train conversational agents (Vinyals and Le, 2015; Sordoni et al., 2015; Serban et al., 2016). The main advantage of such end-to-end dialogue systems is that they make no assumption about the application domain and are simply trained in a supervised fashion from large text corpora (Lowe et al., 2015).

However, there are many drawbacks to this approach. First, encoder-decoder models cast the dialogue problem into one of supervised learning, predicting the distribution over possible next utterances given the discourse so far. As with machine translation, this may result in inconsistent dialogues and errors that can accumulate over time. As the action space of dialogue systems is vast, and existing datasets cover only a small subset of all trajectories, it is difficult to generalize to unseen scenarios (Mooney, 2006). Second, the supervised learning framework does not account for the intrinsic planning problem that underlies dialogue, *i.e.* the sequential decision making process, which makes dialogue consistent over time. This is especially true when engaging in a task-oriented dialogue. As a consequence, reinforcement learning has been applied to dialogue systems since the late 90s (Levin et al., 1997; Singh et al., 1999b) and dialogue optimization has been generally more studied than dialogue generation. Finally, it is unclear whether encoder-decoder supervised training efficiently integrates external contexts (larger than the history of the dialogue) that is most often used by dialogue participants to interact. This

context can be their physical environment, a common task they try to achieve, a map on which they try to find their way, a database they want to access *etc.* These contexts are all the more important as they are part of the so called *Common Ground*, well studied in the discourse literature (Clark and Schaefer, 1989). Over the last decades, the field of cognitive psychology has also brought empirical evidence that human representations are grounded in perception and motor systems (Barsalou, 2008a). These theories imply that a dialogue system should be grounded in a multi-modal environment in order to obtain human-level language understanding (Kiela et al., 2016).

On the other hand, RL approaches could handle the planning and the non-differentiable metric problems but require online learning (although batch learning is possible but difficult with low amounts of data (Pietquin et al., 2011)). For that reason, user simulation has been proposed to explore dialogue strategies in a RL setting (Eckert et al., 1997; Schatzmann et al., 2006; Pietquin and Hastie, 2013b). It also requires the definition of an evaluation metric which is most often related to task completion and user satisfaction (Walker et al., 1997). Without such a goal-achievement metric, it is difficult to correctly evaluate dialogues (Liu et al., 2016a). In addition, successful applications of the RL framework to dialogue often rely on a predefined structure of the task, such as slot-filling tasks (Williams and Young, 2007) where the task can be casted as filling in a form.

In this paper, we present an architecture for end-to-end RL optimization of a task-oriented question generator of a dialogue system and its application to a multimodal task, grounding the dialogue in a visual context. To do so, we start from a corpus of 150k human-human dialogues collected via the recently introduced GuessWhat?! game (de Vries et al., 2016). The goal of the game is to locate an unknown object in a natural image by asking a series of questions. This task is hard since it requires scene understanding and, more importantly, a dialogue strategy that leads one to rapidly identify the target object. From this data, we first build a supervised agent and a neural training environment. It serves to train a Deep RL agent online which is able to solve the task. We then quantitatively and qualitatively compare the performance of our system to a supervised approach on the same task. In short, our contributions are:

— to propose an original visually grounded goal-directed dialogue system optimized via Deep RL;

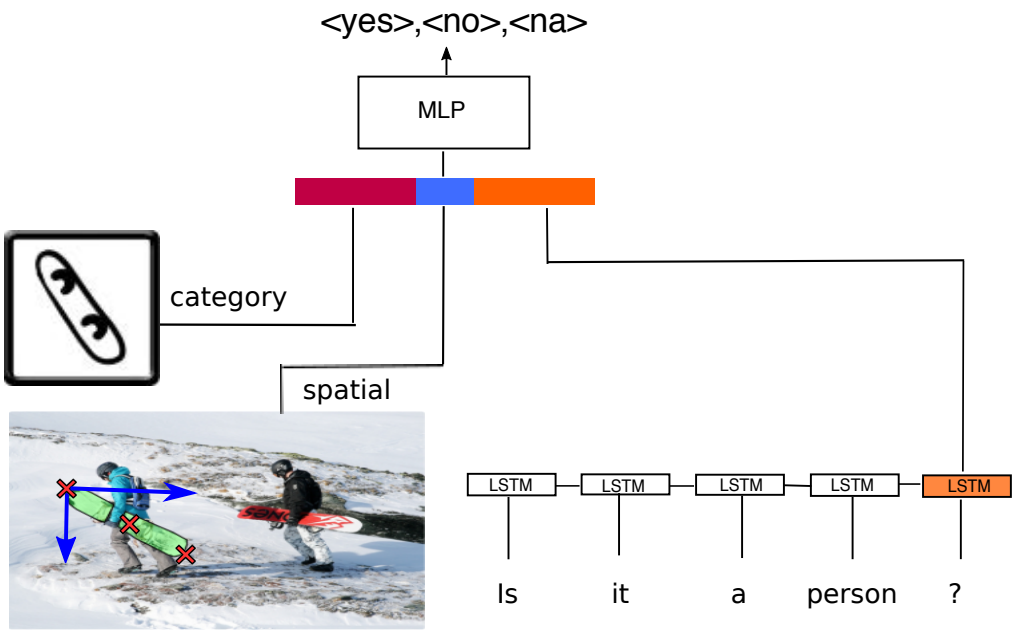


Figure 6.1 – Oracle model.

— to achieve 10% improvement on task completion over a supervised learning baseline.

6.1 GuessWhat?! Game

We briefly explain here the GuessWhat?! game that will serve as a task for our dialogue system, but refer to de Vries et al. (2016) for more details regarding the task and the exact content of the dataset. It is composed of more than 150k human-human dialogues in natural language collected through Mechanical Turk.

6.1.1 Rules

GuessWhat?! is a cooperative two-player game in which both players see the image of a rich visual scene with several objects. One player – the oracle – is randomly assigned an object (which could be a person) in the scene. This object is not known by the other player – the questioner – whose goal is to locate the hidden object. To do so, the questioner can ask a series of yes-no questions which

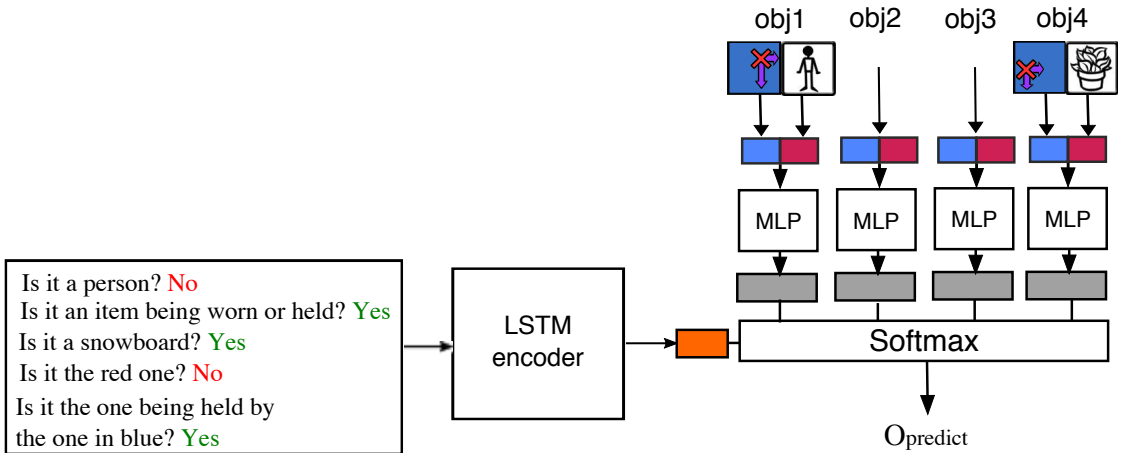


Figure 6.2 – Guesser model.

are answered by the oracle as shown in Fig 4.2. Note that the questioner is not aware of the list of objects and can only see the whole image. Once the questioner has gathered enough evidence to locate the object, he may choose to guess the object. The list of objects is revealed, and if the questioner picks the right object, the game is considered successful.

6.1.2 Notation

Before we proceed, we establish the GuessWhat?! notation that is used throughout the rest of this paper. A game is defined by a tuple $(\mathbf{I}, D, O, o^*)$ where $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$ is a RGB image of height H and width W, D a dialogue with J question-answer pairs $D = (\boldsymbol{q}_j, a_j)_{j=1}^{J}$, O a list of K objects $O = (o_k)_{k=1}^{K}$ and o^* the target object. Moreover, each question $\boldsymbol{q}_j = (w_i^j)_{i=1}^{I_j}$ is a sequence of length I_j with each token w_i^j taken from a predefined vocabulary V. The vocabulary V is composed of a predefined list of words, a question tag <?> that ends a question and a stop token <*stop*> that ends a dialogue. An answer is restricted to be either yes, no or not applicable *i.e.* $a_j \in \{<yes>, <no>, <na>\}$. For each object k, an object category $c_k \in \{1, \ldots, C\}$ and a pixel-wise segmentation mask $\boldsymbol{S}_k \in \{0, 1\}^{H \times W}$ are available.

Finally, to access subsets of a list, we use the following notations. If $l = (l_i^j)_{i=1}^{I,j}$ is a double-subscript list, then $l_{1:i}^j = (l_p^j)_{p=1}^{i,j}$ are the i first elements of the j^{th} list if $1 \leq i \leq I_j$, otherwise $l_{1:p}^j = \emptyset$. Thus, for instance, $w_{1:i}^j$ refers to the first i tokens of

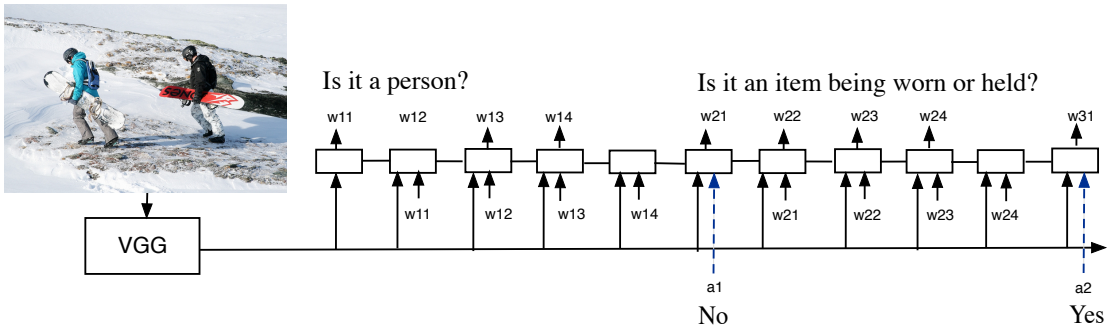


Figure 6.3 – Question generation model.

the j^{th} question and $(\boldsymbol{q}, a)_{1:j}$ refers to the j first question-answer pairs of a dialogue.

6.2 Training Environment

From the GuessWhat?! dataset, we build a training environment that allows RL optimization of the questioner task by creating models for the oracle and guesser tasks. We also describe the supervised learning baseline to which we will compare. This mainly reproduces models introduced in de Vries et al. (2016).

Question generation architecture We split the questioner's job into two different tasks: one for asking the questions and another one for guessing the object. The question generation task requires to produce a new question $\boldsymbol{q}_{j+1}$, given an image $\mathbf{I}$ and a history of j questions and answers $(\boldsymbol{q}, a)_{1:j}$. We model the question generator (QGen) with a recurrent neural network (RNN), which produces a sequence of RNN state vectors $\boldsymbol{s}^j_{1:i}$ for a given input sequence $\boldsymbol{w}^j_{1:i}$ by repeatedly applying the transition function f, i.e. $\boldsymbol{s}^j_{i+1} = f(\boldsymbol{s}^j_i, e(w^j_i))$. The parameterized look-up table e embeds the input token into a continuous vector and for our transition function we use the popular long-short term memory (LSTM) cell (Hochreiter and Schmidhuber, 1997). In order to construct a probabilistic sequence model, we add a softmax function g over the RNN state that computes a distribution over tokens w^j_i from vocabulary V. In the case of GuessWhat?!, the probability of outputting token w^j_i is conditioned on all previous questions and answers tokens as

well as the image $\mathbf{I}$:

$$g(\boldsymbol{s}_i^j)_{w_i^j} = P(w_i^j | w_{1:i-1}^j, (\boldsymbol{q}, a)_{1:j-1}, \mathbf{I}). \tag{6.1}$$

We condition the model on the image by obtaining its VGG16 FC8 features and concatenating it to the input embedding at each step, as illustrated in Fig. 6.3. We train the model by minimizing the conditional negative log-likelihood:

$$\begin{aligned} -\log P(\boldsymbol{q}_{1:J} | a_{1:J}, \mathbf{I}) &= -\log \prod_{j=1}^{J} P(\boldsymbol{q}_j | (\boldsymbol{q}, a)_{1:j-1}, \mathbf{I}), \\ &= -\sum_{j=1}^{J} \sum_{i=1}^{I_j} \log P(w_i^j | w_{1:i-1}^j, (\boldsymbol{q}, a)_{1:j-1}, \mathbf{I}). \end{aligned} \tag{6.2}$$

At test time, we can generate a sample $p(\boldsymbol{q}_j | (\boldsymbol{q}, a)_{1:j-1}, \mathbf{I})$ from the model as follows. Starting from the state $\boldsymbol{s}_1^j$, we sample a new token w_i^j from the output distribution g and feed the embedded token $e(w_i^j)$ back as input to the RNN. We repeat this loop till we encounter an end-of-sequence token. To approximately find the most likely question, $\max_{\boldsymbol{q}_j} \; p(\boldsymbol{q}_j | (\boldsymbol{q}, a)_{1:j-1}, \mathbf{I})$, we use the commonly used beam-search procedure. This heuristics aims to find the most likely sequence of words by exploring a subset of all questions and keeping the K-most promising candidate sequences at each time step.

Oracle The oracle task requires to produce a yes-no answer for any object within an image given a natural language question. We outline here the neural network architecture that achieved the best performance and refer to de Vries et al. (2016) for a thorough investigation of the impact of other object and image information. First, we embed the spatial information of the crop by extracting an 8-dimensional vector of the location of the bounding box

$$\boldsymbol{x}_{spatial} = [x_{min}, y_{min}, x_{max}, y_{max}, x_{center}, y_{center}, w_{box}, h_{box}] \tag{6.3}$$

where w_{box} and h_{box} denote the width and height of the bounding box , respectively. We normalize the image height and width such that coordinates range from -1 to 1, and place the origin at the center of the image. Second, we convert the object category c^* into a dense category embedding using a learned look-up table. Finally,

we use a LSTM to encode the current question $\boldsymbol{q}$. We then concatenate all three embeddings into a single vector and feed it as input to a single hidden layer MLP that outputs the final answer distribution $P(\text{a}|\boldsymbol{q}, c^*, \boldsymbol{x}^*_{spatial})$ using a softmax layer, illustrated in Fig. 6.1.

Guesser The guesser model takes an image $\mathbf{I}$ and a sequence of questions and answers $(\boldsymbol{q}, a)_{1:J}$, and predicts the correct object o^* from the set of all objects. This model considers a dialogue as one flat sequence of question-answer tokens and use the last hidden state of the LSTM encoder as our dialogue representation. We perform a dot-product between this representation and the embedding for all the objects in the image, followed by a softmax to obtain a prediction distribution over the objects. The object embeddings are obtained from the categorical and spatial features. More precisely, we concatenate the 8-dimensional spatial representation and the object category look-up and pass it through an MLP layer to get an embedding for the object. Note that the MLP parameters are shared to handle the variable number of objects in the image. See Fig 6.2 for an overview of the guesser.

6.2.1 Generation of Full Games

With the question generation, oracle and guesser model we have all components to simulate a full game. Given an initial image $\mathbf{I}$, we generate a question $\boldsymbol{q}_1$ by sampling tokens from the question generation model until we reach the question-mark token. Alternatively, we can replace the sampling procedure by a beam-search to approximately find the most likely question according to the generator. The oracle then takes the question $\boldsymbol{q}_1$, the object category c^* and $\boldsymbol{x}^*_{spatial}$ as inputs, and outputs the answer a_1. We append $(\boldsymbol{q}_1, a_1)$ to the dialogue and repeat generating question-answer pairs until the generator emits a stop-dialogue token or the maximum number of question-answers is reached. Finally, the guesser model takes the generated dialogue D and the list of objects O and predicts the correct object.

6.3 GuessWhat?! from RL Perspective

One of the drawbacks of training the QGen in a supervised learning setup is that its sequence of questions is not explicitly optimized to find the correct object. Such training objectives miss the planning aspect underlying (goal-oriented) dialogues. In this paper, we propose to cast the question generation task as a RL task. More specifically, we use the training environment described before and consider the oracle and the guesser as part of the RL agent environment. In the following, we first formalize the GuessWhat?! task as a Markov Decision Process (MDP) so as to apply a policy gradient algorithm to the QGen problem.

6.3.1 GuessWhat?! as a Markov Decision Process

We define the state $\boldsymbol{x_t}$ as the status of the game at step t. Specifically, we define $\boldsymbol{x_t} = ((w_1^j, \ldots, w_i^j), (\boldsymbol{q}, a)_{1:j-1}, \mathbf{I})$ where $t = \sum_{j=1}^{j-1} I_j + i$ corresponds to the number of tokens generated since the beginning of the dialogue. An action u_t corresponds to select a new word w_{i+1}^j in the vocabulary V. The transition to the next state depends on the selected action:

- If $w_{i+1}^j = <stop>$, the full dialogue is terminated.
- If $w_{i+1}^j = <?>$, the ongoing question is terminated and an answer a_j is sampled from the oracle. The next state is $\boldsymbol{x}_{t+1} = ((\boldsymbol{q}, a)_{1:j}, \mathbf{I})$ where $\boldsymbol{q}_j = (w_1^j, \ldots, w_i^j, <?>)$.
- Otherwise the new word is appended to the ongoing question and $\boldsymbol{x}_{t+1} = ((w_1^j, \ldots, w_i^j, w_{i+1}^j), (\boldsymbol{q}, a)_{1:j-1}, \mathbf{I})$.

Questions are automatically terminated after I_{max} words. Similarly, dialogues are terminated after J_{max} questions. Furthermore, a reward $r(\boldsymbol{x}, u)$ is defined for every state-action pair. A trajectory $\tau = (\boldsymbol{x}_t, u_t, r(\boldsymbol{x}_t, u_t))_{1:T}$ is a finite sequence of tuples of length $T \leq J_{max} * I_{max}$ which contains a state, an action, and the reward. Thus, the game falls into the episodic RL scenario as the dialogue terminates after a finite sequence of question-answer pairs. Finally, the QGen output can be viewed as a stochastic policy $\pi_{\boldsymbol{\theta}}(\mathrm{u}|\boldsymbol{x})$ parametrized by $\boldsymbol{\theta}$ which associates a probability distribution over the actions (i.e. words) for each state (*i.e.* intermediate dialogue and image).

6.3.2 Training the QGen with Policy Gradient

While several approaches exist in the RL literature, we opt for policy gradient methods because they are known to scale well to large action spaces (Silver et al., 2016). This is especially important in our case because the vocabulary size is nearly 5k words. The goal of policy optimization is to find a policy $\pi_{\boldsymbol{\theta}}(\mathrm{u}|\boldsymbol{x})$ that maximizes the expected return, also known as the mean value:

$$J(\boldsymbol{\theta}) = E_{\tau\sim\pi^{\boldsymbol{\theta}}(\tau)}[\sum_{t=0}^{T}\gamma^t r(\boldsymbol{x}_t, u_t)], \tag{6.4}$$

where $\gamma \in [0, 1]$ is the discount factor, T the length of the trajectory. Note that $\gamma = 1$ is allowed as we are in the episodic scenario (Sutton et al., 1999). To improve the policy, its parameters can be updated in the direction of the gradient of the mean value:

$$\boldsymbol{\theta}_{h+1} = \boldsymbol{\theta}_h + \alpha_h \nabla_\theta J|_{\theta=\theta_h}, \tag{6.5}$$

where h denotes the training time-step and α_h is a learning rate such that $\sum_{h=1}^{\infty}\alpha_h = \infty$ and $\sum_{h=1}^{\infty}\alpha_h^2 < \infty$.

Thanks to the gradient policy theorem (Sutton et al., 1999), the gradient of the mean value can be estimated from a batch of trajectories $\mathcal{T}_h$ sampled from the current policy $\pi_{\boldsymbol{\theta}_h}$ by:

$$\nabla J(\boldsymbol{\theta}_h) = E_{\tau\sim\pi_{\boldsymbol{\theta}_h}(\tau)}\left[\sum_{t=1}^{T}\nabla_{\boldsymbol{\theta}_h}\log\pi_{\boldsymbol{\theta}_h}(u_t|\boldsymbol{x}_t)(q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}_t, u_t) - b(\boldsymbol{x}_t))\right], \tag{6.6}$$

where $q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}, u)$ is the state-action value and $b(\boldsymbol{x}_t)$ some arbitrarily baseline function which can help reducing the variance of the estimation of the gradient. The state action value $q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}, u)$ quantifies the expected cumulative reward of executing policy $\pi_{\boldsymbol{\theta}_h}$ after taking action u_t in state $\boldsymbol{x}_t$:

$$q^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}_t, u_t) = E_{\tau\sim\pi_{\boldsymbol{\theta}_h}(\tau)}\big[\sum_{t'=t}^{T}\gamma^{t'-t} r(\boldsymbol{x}_{t'}, u_{t'})\big]. \tag{6.7}$$

It is possible to obtain an unbiased estimate of this q-value by using the cumulative reward of the sample trajectory: $\hat{q}^{\pi_{\boldsymbol{\theta}_h}}(\boldsymbol{x}_t, u_t) = \sum_{t'=t}^{T}\gamma^{t'-t} r(\boldsymbol{x}_{t'}, u_{t'})$. This choice leads to the REINFORCE algorithm (Williams, 1992)). Alternatively, we

can estimate the state-action value-function by learning a q-function approximator, resulting in so-called actor-critic methods. Finally, by using the GuessWhat?! game notation for equation 6.6, the policy gradient for the QGen can be written as:

$$\nabla J(\boldsymbol{\theta}_h) = \sum_{j=1}^{J} \sum_{i=1}^{I_j} \nabla_{\boldsymbol{\theta}_h} \log \pi_{\boldsymbol{\theta}_h}(w_i^j | w_{1:i-1}^j, (\boldsymbol{q}, a)_{1:j-1}, \mathbf{I}) \\ (q^{\pi_{\boldsymbol{\theta}_h}}((w_{1:i-1}^j, (\boldsymbol{q}, a)_{1:j-1}, \mathbf{I}), w_i^j) - b(\boldsymbol{x}_t)). \quad (6.8)$$

6.3.3 Reward Function

One tedious aspect of RL is to define a correct and valuable reward function. As the optimal policy is the result of the reward function, one must be careful to design a reward that would not change the expected final optimal policy (Ng et al., 1999). Therefore, we put a minimal amount of prior knowledge into the reward function and construct a zero-one reward depending on the guesser's prediction:

$$r(\boldsymbol{x}_t, u_t) = \begin{cases} 1 & \text{If } \operatorname{argmax}_o[Guesser(\boldsymbol{x}_t)] = o* \text{ and } t = T \\ 0 & \text{Otherwise} \end{cases}. \quad (6.9)$$

So, we give a reward of one if the correct object is found from the generated questions, and zero otherwise.

Note that the reward function requires the target object $o*$ while it is not included in the state $\boldsymbol{x} = ((\boldsymbol{q}, a)_{1:J}, \mathbf{I})$. This breaks the MDP assumption that the reward should be a function of the current state and action. However, policy gradient methods, such as REINFORCE, are still applicable if the MDP is partially observable (Williams, 1992).

6.3.4 Full Training Procedure

For the QGen, oracle and guesser, we use the model architectures outlined in section 6.2. We first independently train the three models with a cross-entropy loss. We then keep the oracle and guesser models fixed, while we train the QGen in the described RL framework. It is important to pretrain the QGen to kick-start training from a reasonable policy. The size of the action space is simply too big to

Algorithm 2 Training of QGen with REINFORCE

Require: Pretrained QGen,Oracle and Guesser
Require: Batch size K
for Each update **do**
  # Generate trajectories $\mathcal{T}_h$
  for $k = 1$ to K **do**
    Pick Image $\mathbf{I}_k$ and the target object $o_k^* \in O_k$
    # Generate question-answer pairs $(\boldsymbol{q}, a)_{1:j}^k$
    for $j = 1$ to J_{max} **do**
      $q_j^k = QGen(\boldsymbol{q}, a)_{1:j-1}^k, \mathbf{I}_k)$
      $a_j^k = Oracle(\boldsymbol{q}_j^k, o_k^*, \mathbf{I}_k)$
      if $<stop> \in \boldsymbol{q}_j^k$ **then**
        delete $(q, a)_j^k$ and break;
    $p(o_k|\cdot) = Guesser((q, a)_{1:j}^k, \mathbf{I}_k, O_k)$
    $r(\boldsymbol{x}_t, u_t) = \begin{cases} 1 & \text{If } \text{argmax}_{o_k} p(o_k|\cdot) = o_k^* \\ 0 & \text{Otherwise} \end{cases}$
  Define $\mathcal{T}_h = ((q, a)_{1:j_k}^k, \mathbf{I}_k, r_k)_{1:K}$
  Evaluate $\nabla J(\boldsymbol{\theta}_h)$ with Eq. equation 6.8 with $\mathcal{T}_h$
  SGD update of QGen parameters $\boldsymbol{\theta}$ using $\nabla J(\boldsymbol{\theta}_h)$
  Evaluate $\nabla L(\boldsymbol{\phi}_h)$ with Eq. equation 6.10 with $\mathcal{T}_h$
  SGD update of baseline parameters using $\nabla L(\boldsymbol{\phi}_h)$

start from a random policy.

In order to reduce the variance of the policy gradient, we implement the baseline $b_{\boldsymbol{\phi}}(\boldsymbol{x}_t)$ as a function of the current state, parameterized by $\boldsymbol{\phi}$. Specifically, we use a one layer MLP which takes the LSTM hidden state of the QGen and predicts the expected reward. We train the baseline function by minimizing the Mean Squared Error (MSE) between the predicted reward and the discounted reward of the trajectory at the current time step:

$$L(\boldsymbol{\phi_h}) = E_{\tau \sim \pi_{\boldsymbol{\theta}_h}(\tau)} \big[b_{\phi_h}(\boldsymbol{x}_t) - \sum_{t'=t}^{T} \gamma^{t'} r_{t'} \big]^2 \tag{6.10}$$

We summarize our training procedure in Algorithm 2.

6.4 Related Work

Outside of the dialogue literature, RL methods have been applied to encoder-decoder architectures in machine translation (Ranzato et al., 2016; Bahdanau et al., 2017) and image captioning (Liu et al., 2016b). In those scenarios, the BLEU score is used as a reward signal to fine-tune a network trained with a cross-entropy loss. However, the BLEU score is a surrogate for human evaluation of naturalness, so directly optimizing this measure does not guarantee improvement in the translation/captioning quality. In contrast, our reward function encodes task completion, and optimizing this metric is exactly what we aim for. Finally, the BLEU score can only be used in a batch setting because it requires the ground-truth labels from the dataset. In GuessWhat?!, the computed reward is independent from the *human* generated dialogue.

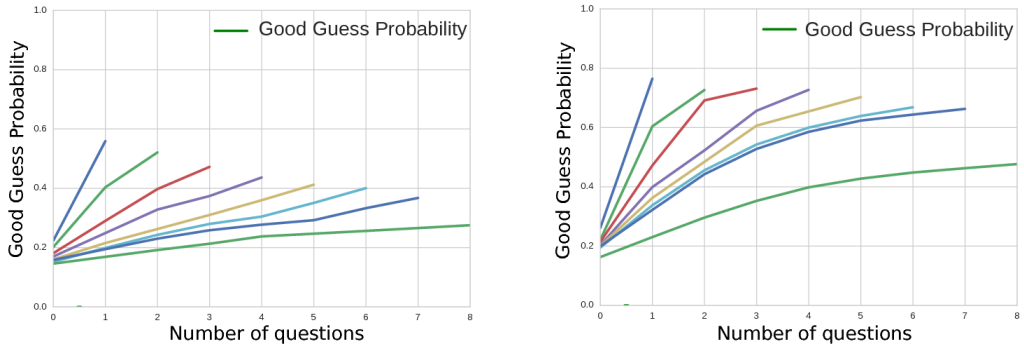
Although visually-grounded language models have been studied for a long time (Roy, 2002), important breakthroughs in both visual and natural language understanding has led to a renewed interest in the field. Especially image captioning (Lin et al., 2014) and visual question answering (Antol et al., 2015) has received much attention over the last few years, and encoder-decoder models (Liu et al., 2016b; Lu et al., 2016) have shown promising results for these tasks. Only very recently the language grounding tasks have been extended to a dialogue setting with the Visual Dialog (Das et al., 2017a,b) and GuessWhat?! (de Vries et al., 2016) datasets. Both games are goal-oriented and can be cast into an RL framework. However, only the GuessWhat?! game requires the question generation task to be visually grounded.

6.5 Experiments

As already said, we used the GuessWhat?! dataset that includes 155,281 dialogues containing 821,955 question/answer pairs composed of 4900 words on 66,537 unique images and 609,543 objects.

Image	Beam Search	REINFORCE
	Is it a person ? no Is it a ball ? no Is it a ball ? no Is it a ball ? no Is it a ball ? no Failure (blue bat)	Is it a person ? no Is a glove ? no Is an umbrella ? no Is in the middle ? no On a person? no is it on on far right? yes Success (red chair)
	Is it a cat ? no Is it a book ? no Is it a book ? no Is it a book ? no Is it a book ? no Failure (person)	Is it a cat ? no Is it on the table ? yes Is it the book ? no Is it fully visible? yes Success (bowl)
	Is it a person ? yes Is it the one in front ? yes Is it the one on the left ? no Is it the one in the middle with the red umbrella ? yes Is it the one to the right of the girl in ? no Failure (umbrella)	Is it a person ? yes Is it girl in white ? yes Success (girl)
	Is it a bag ? yes Is it red ? no Is it the one in the middle ? no Is it the one on the far right ? no Is it the one with the blue bag ? yes Success (most left bag)	Is it a suitcase? yes Is it in the left side ? yes Failure (left bag)

Table 6.1 – Samples extracted from the test set. The blue (resp. purple) box corresponds to the object picked by the guesser for the beam-search (resp. REINFORCE) dialogue. The small verbose description is added to refer to the object picked by the guesser.



(a) CE (sampling) **(b)** REINFORCE (sampling)

Figure 6.4 – (a-b) Each line represents a dialogue of size N and describe the evolution of the average probability of the guesser to find the correct object question after question.

6.5.1 Training Details

We pre-train the networks described in Section 6.2 [1]. After training, the oracle network obtains 21.5% error and the guesser network reports 36.2% error on the test set. Throughout the rest of this section we refer to the pretrained QGen as Cross-Entropy trained model (CE).

We then initialize our environment with the pre-trained models and train the QGen with REINFORCE for 80 epochs with plain stochastic gradient descent (SGD) with a learning rate of 0.001 and a batch size of 64. For each epoch, we sample each training images once, and randomly choose one of its object as the target. We simultaneously optimize the baseline parameters ϕ with SGD with a learning rate of 0.001. Finally, we set the maximum number of questions to 8 and the maximum number of words to 12

6.5.2 Results

Accuracy We report the accuracies of the QGen trained with REINFORCE and CE in Table 6.2. We compare sampling objects from the training set (New Objects) and test set (New Images) i.e. unseen Images. We report the standard deviation over 5 runs in order to account for the sampling stochasticity. On the test set, training with CE obtains 41.6% accuracy, while training with REINFORCE improves to 58.5%. This is also a significant improvement over the beam-search

1. Source code available at: https://guesswhat.ai

		New Objects	New Images
CE	Sampling	41.6% ± 0.2	39.2% ± 0.1
	Greedy	43.5% ± 0.1	40.8%
	BSearch	47.1% ± 0.0	44.6%
REINFORCE	Sampling	58.5% ± 0.3	56.5% ± 0.2
	Greedy	**60.3% ± 0.1**	**58.4%**
	BSearch	60.2% ± 0.1	58.4%
Human		90.1%	
Human with Guesser		63.8%	
Random		18,1%	

Table 6.2 – Guessing accuracy of the QGen with CE and REINFORCE. New objects refers to uniformly sampling objects within the training set, new images refer to sampling objects from the test set.

CE, which achieves 47.1% on the test-set. Our proposed framework thus closes the gap towards human-performance (90.1%) with more than 11%. The beam-search procedure improves over sampling from CE, but lowers the score for REINFORCE.

Samples We qualitatively compare the two methods by analyzing a few generated samples, as shown in Table 10.6. We observe that the beam-search trained with CE keeps repeating the same questions, as can be seen in the two top examples in Tab. 10.6. We noticed this behavior especially on the test set *i.e.* when confronted with unseen images, which may highlight some generalization issues. We also find that the beam-search CE generates longer questions (7.1 tokens on average) compared to REINFORCE (4.0 tokens on average). This qualitative difference is clearly visible in the bottom-left example, which also highlights that CE sometimes generates visually relevant but incoherent sequences of questions. For instance, asking "Is it the one to the right of the girl in?" is not a very logical follow-up of "Is it the one in the middle with the red umbrella?". In contrast, REINFORCE seem to implement a more grounded and relevant strategy.In general, we observe that REINFORCE favors enumerating object categories ("is it a person?") or absolute spatial information ("Is it left?"). Note these are also the type of ques-

tions that the oracle is expected to answer correctly. Differently, REINFORCE is able to efficiently tailor its strategy toward the current dialogue context as shown in Fig 6.4. REINFORCE successfully narrows the space of objects towards the correct one while CE faces more difficulties to output discriminative questions.

Dialogue Length For the REINFORCE trained QGen, we investigate the impact of the dialogue length on the success ratio. Interestingly, REINFORCE learns to stop on average after 4.1 questions, although we did not encode a question penalty into the reward function. This policy may be enforced by the guesser since asking additional but noisy questions greatly lower the prediction accuracy of the guesser as shown in Tab. 10.6. Therefore, the QGen learns to stop asking questions when a dialogue contains enough information to retrieve the target object. However, we observe that the QGen sometimes stops too early, especially when the image contains too many objects of the same category. Interestingly, we also found that the beam-search fails to stop the dialogue. Beam-search uses a length-normalized log-likelihood to score candidate sequences to avoid a bias towards shorter questions. However, questions in GuessWhat?! almost always start with "is it", which increases the average log likelihood of a question significantly. The score of a new question might thus (almost) always be higher than emitting a single $<stop>$ token.

Vocabulary Sampling from the REINFORCE trained model uses 1,2k distinct words while CE (beam-search) vocabulary is reduced to 0.5k unique words. Thus, REINFORCE seems to benefit from exploring the space of words in the training process.

6.6 Conclusion

In this paper, we proposed to build a training environment from supervised deep learning baselines in order to train a Deep RL agent to solve a goal-oriented multimodal dialogue task. We show the promise of this approach on the GuessWhat?! dataset, and observe quantitatively and qualitatively an encouraging improvement over a supervised baseline model. While supervised learning models fail to generate

a coherent dialogue strategy, our method learns when to stop after generating a sequence of relevant questions.

7 Prologue to Third Article

7.1 Article Details

H. de Vries*, F. Strub*, J. Mary, H Larochelle, O. Pietquin, A. Courville. Modulating Early Visual Processing By Language. *Neural Infomation Processing Systems (NIPS), 2017*

Personal contribution I came up with the idea of applying conditional batch normalization to VQA after discussions with Vincent Dumoulin and Aaron. Florian and I implemented the method for a pre-trained ResNet and applied it to the VQA and GuessWhat datasets. Florian and I wrote the paper, with help from Jeremie.

7.2 Context

A few months before this work, Vincent Dumoulin (a PhD student at MILA) successfully applied class-conditional instance normalization in the context of image style transfer. For the GuessWhat project, we were looking to improve the question-answering model, as our previous paper had shown that even the question generation process is limited by the strength of this model. We combined both threads and started investigating the use of conditional normalization methods in the context of visual question answering.

7.3 Contributions

This paper proposes to condition the batch normalization parameters of a pre-trained ResNet on the question at hand. This enables the question to influence the

visual processing early on in the computational pipeline (even though the convolutional weights are frozen). We show that the modulated ResNet architecture performs better than a strong baseline on the VQAv1 and GuessWhat oracle datasets. Through ablation studies we also demonstrate that early modulation of the visual processing pipeline is helpful.

7.4 Recent Developments

An important follow-up work is FiLM (Perez et al., 2018), which extends conditional normalization methods to the domain of visual reasoning. Contrary to the architecture presented in this work, the conditional batch normalization layers are stacked on top of the extracted ResNet representations and convolutional weights are jointly optimized. They also demonstrate that the affine transformation layer can be decoupled from the normalization layer without affecting performance. To better reflect this property, they rename conditional batch-normalization to Feature-wise Linear Modulation (FiLM). These layers have been applied in numerous deep learning applications, such as conditional image generation with generative adversial networks (Brock et al., 2019) and few-shot image classification (Oreshkin et al., 2018).

8 Modulating Early Visual Processing by Language

Human beings combine the processing of language and vision with apparent ease. For example, we can use natural language to describe perceived objects and we are able to imagine a visual scene from a given textual description. Developing intelligent machines with such impressive capabilities remains a long-standing research challenge with many practical applications.

Towards this grand goal, we have witnessed an increased interest in tasks at the intersection of computer vision and natural language processing. In particular, image captioning (Lin et al., 2014), visual question answering (VQA) (Antol et al., 2015; Tejas et al., 2017) and visually grounded dialogue systems (Das et al., 2017a; de Vries et al., 2016) constitute a popular set of example tasks for which large-scale datasets are now available. Developing computational models for language-vision tasks is challenging, especially because of the open question underlying all these tasks: how to fuse/integrate visual and textual representations? To what extent should we process visual and linguistic input separately, and at which stage should we fuse them? And equally important, what fusion mechanism to use?

In this paper, we restrict our attention to the domain of visual question answering which is a natural testbed for fusing language and vision. The VQA task concerns answering open-ended questions about images and has received significant attention from the research community (Antol et al., 2015; Fukui et al., 2016; Malinowski et al., 2015; Tejas et al., 2017). Current state-of-the-art systems often use the following computational pipeline (Ben-Younes et al., 2017; Malinowski et al., 2015; Ren et al., 2015) illustrated in Fig 8.1. They first extract *high-level* image features from an ImageNet pretrained convolutional network (e.g. the activations from a ResNet network (Kaiming et al., 2016)), and obtain a language embedding using a recurrent neural network (RNN) over word-embeddings. These two high-level representations are then fused by concatenation (Malinowski et al., 2015), element-wise product (Lu et al., 2016; Kim et al., 2016, 2017; Malinowski et al., 2015), Tucker decomposition (Ben-Younes et al., 2017) or compact bilinear pool-

ing (Fukui et al., 2016), and further processed for the downstream task at hand. Attention mechanisms (Xu et al., 2015) are often used to have questions attend to specific spatial locations of the extracted higher-level feature maps.

There are two main reasons for why the recent literature has focused on processing each modality independently. First, using a pretrained convnet as feature extractor prevents overfitting; Despite a large training set of a few hundred thousand samples, backpropagating the error of the downstream task into the weights of all layers often leads to overfitting. Second, the approach aligns with the dominant view that language interacts with high-level visual concepts. Words, in this view, can be thought of as "pointers" to high-level conceptual representations. To the best of our knowledge, this work is the first to fuse modalities at the very early stages of the image processing.

In parallel, the neuroscience community has been exploring to what extent the processing of language and vision is coupled (F. Ferreira and M. Tanenhaus, 2007). More and more evidence accumulates that words set visual priors which alter how visual information is processed from the very beginning (Boutonnet and Lupyan, 2015; Kok et al., 2014; Thierry et al., 2009). More precisely, it is observed that P1 signals, which are related to low-level visual features, are modulated while hearing specific words (Boutonnet and Lupyan, 2015). The language cue that people hear ahead of an image activates visual predictions and speed up the image recognition process. These findings suggest that independently processing visual and linguistic features might be suboptimal, and fusing them at the early stage may help the image processing.

In this paper, we introduce a novel approach to have language modulate the *entire* visual processing of a pre-trained convnet. We propose to condition the batch normalization (Ioffe and Szegedy, 2015) parameters on linguistic input (e.g., a question in a VQA task). Our approach, called Conditional Batch Normalization (CBN), is inspired by recent work in style transfer (Dumoulin et al., 2017). The key benefit of CBN is that it scales linearly with the number of feature maps in a convnet, which impacts less than 1% of the parameters, greatly reducing the risk of over-fitting. We apply CBN to a pretrained Residual Network, leading to a novel architecture to which we refer as MODERN. We show significant improvements on two VQA datasets, VQAv1 (Antol et al., 2015) and GuessWhat?! (de Vries et al., 2016), but stress that our approach is a general fusing mechanism that can be

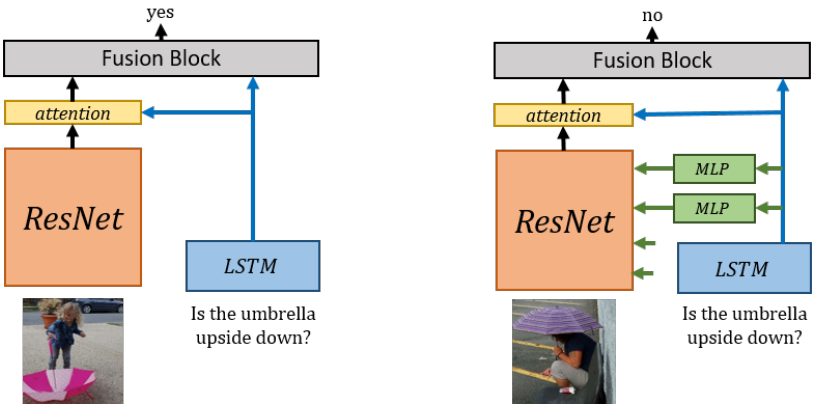


Figure 8.1 – An overview of the classic VQA pipeline (left) vs ours (right). While language and vision modalities are independently processed in the classic pipeline, we propose to directly modulate ResNet processing by language.

applied to other multi-modal tasks.

To summarize, our contributions are three fold:

— We propose conditional batch normalization to modulate the entire visual processing by language from the early processing stages,

— We condition the batch normalization parameters of a pretrained ResNet on linguistic input, leading to a new network architecture: MODERN,

— We demonstrate improvements on state-of-the-art models for two VQA tasks and show the contribution of this modulation on the early stages.

8.1 Background

In this section we provide preliminaries on several components of our proposed VQA model.

8.1.1 Residual networks

We briefly outline residual networks (ResNets) (Kaiming et al., 2016), one of the current top-performing convolutional networks that won the ILSVRC 2015 classification competition. In contrast to precursor convnets (e.g. VGG (Simonyan and Zisserman, 2015)) that constructs a new representation at each layer, ResNet

iteratively refines a representation by adding residuals. This modification enables to train very deep convolutional networks without suffering as much from the vanishing gradient problem. More specifically, ResNets are built from residual blocks:

$$\mathbf{F}^{k+1} = \text{ReLU}(\mathbf{F}^k + R(\mathbf{F}^k)) \tag{8.1}$$

where F^k denotes the outputted feature map. We will refer to $F_{i,c,w,h}$ to denote the i^{th} input sample of the c^{th} feature map at location (w, h). The residual function $R(F^k)$ is composed of three convolutional layers (with a kernel size of 1, 3 and 1, respectively). See Fig. 2 in the original ResNet paper Kaiming et al. (2016) for a detailed overview of a residual block.

A group of blocks is stacked to form a *stage* of computation in which the representation dimensionality stays identical. The general ResNet architecture starts with a single convolutional layer followed by four stages of computation. The transition from one stage to another is achieved through a projection layer that halves the spatial dimensions and doubles the number of feature maps. There are several pretrained ResNets available, including ResNet-50, ResNet-101 and ResNet-152 that differ in the number of residual blocks per stage.

8.1.2 Batch Normalization

The convolutional layers in ResNets make use of Batch Normalization (BN), a technique that was originally designed to accelarate the training of neural networks by reducing the internal co-variate shift (Ioffe and Szegedy, 2015). Given a mini-batch $\mathcal{B} = \{F_{i,\cdot,\cdot,\cdot}\}_{i=1}^{N}$ of N examples, BN normalizes the feature maps at training time as follows:

$$BN(F_{i,c,h,w}; \gamma_c, \beta_c) = \gamma_c \frac{F_{i,c,w,h} - \text{E}_{\mathcal{B}}[F_{\cdot,c,\cdot,\cdot}]}{\sqrt{\text{Var}_{\mathcal{B}}[F_{\cdot,c,\cdot,\cdot}] + \epsilon}} + \beta_c, \tag{8.2}$$

where ϵ is a constant damping factor for numerical stability, and γ_c and β_c are trainable scalars introduced to keep the representational power of the original network. Note that for convolutional layers the mean and variance are computed over both the batch and spatial dimensions (such that each location in the feature map is normalized in the same way). After the BN module, the output is fed to a non-linear activation function. At inference time, the batch mean $\text{E}_{\mathcal{B}}$ and variance

$\text{Var}_{\mathcal{B}}$ are replaced by the population mean μ and variance σ^2, often estimated by an exponential moving average over batch mean and variance during training.

8.1.3 Language embeddings

We briefly recap the most common way to obtain a language embedding from a natural language question. Formally, a question $\boldsymbol{q} = [w_k]_{k=1}^K$ is a sequence of length K with each token w_k taken from a predefined vocabulary V. We transform each token into a dense word-embedding $e(w_k)$ by a learned look-up table. For task with limited linguistic corpora (like VQA), it is common to concatenate pretrained Glove (Pennington et al., 2014) vectors to the word embeddings. The sequence of embeddings $[e(w_k)]_{k=1}^K$ is then fed to a recurrent neural network (RNN), which produces a sequence of RNN state vectors $[\boldsymbol{s}_k]_{k=1}^K$ by repeatedly applying the transition function f:

$$\boldsymbol{s}_{k+1} = f(\boldsymbol{s}_k, e(w_k)). \tag{8.3}$$

Popular transition functions, like a long-short term memory (LSTM) cell (Hochreiter and Schmidhuber, 1997) and a Gated Recurrent Unit (GRU) (Cho et al., 2014), incorporate gating mechanisms to better handle long-term dependencies. In this work, we will use an LSTM cell as our transition function. Finally, we take the last hidden state s_I as the embedding of the question, which we denote as $\boldsymbol{e}_q$ throughout the rest of this paper.

8.2 Modulated Residual Networks

In this section we introduce conditional batch normalization, and show how we can use it to modulate a pretrained ResNet. The key idea is to predict the γ and β of the batch normalization from a language embedding. We first focus on a single convolutional layer with batch normalization module $BN(\mathbf{F}_{i,c,h,w}; \gamma_c, \beta_c)$ for which pretrained scalars γ_c and β_c are available. We would like to directly predict these affine scaling parameters from our language embedding $\boldsymbol{e_q}$. When starting the training procedure, these parameters must be close to the pretrained values to recover the original ResNet model as a poor initialization could significantly deteriorate performance. Unfortunately, it is difficult to initialize a network to output

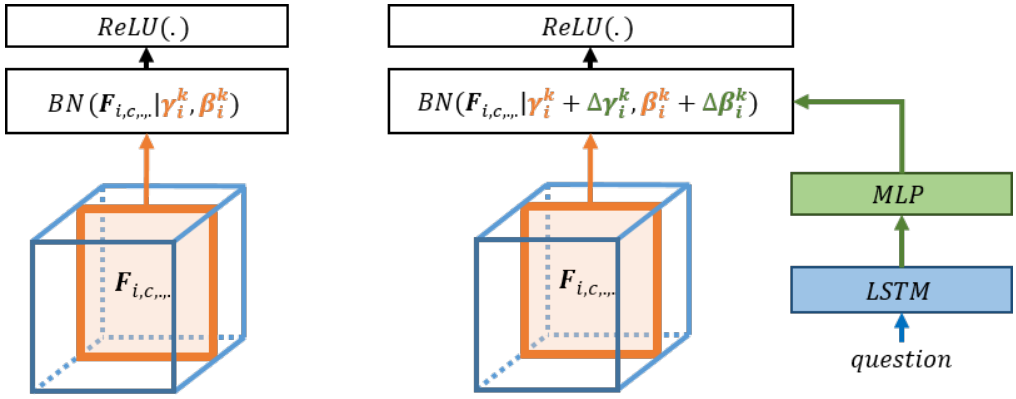


Figure 8.2 – An overview of the computation graph of batch normalization (left) and conditional batch normalization (right). Best viewed in color.

the pretrained γ and β. For these reasons, we propose to predict a change $\Delta\beta_c$ and $\Delta\gamma_c$ on the frozen original scalars, for which it is straightforward to initialize a neural network to produce an output with zero-mean and small variance.

We use a one-hidden-layer MLP to predict these deltas from the question embedding $\boldsymbol{e}_q$ for all feature maps within the layer:

$$\boldsymbol{\Delta\beta} = MLP(\boldsymbol{e}_q) \qquad \boldsymbol{\Delta\gamma} = MLP(\boldsymbol{e}_q) \tag{8.4}$$

So, given a feature map with C channels, these MLPs output a vector of size C. We then add these predictions to the β and γ parameters:

$$\hat{\beta}_c = \beta_c + \Delta\beta_c \qquad \hat{\gamma}_c = \gamma_c + \Delta\gamma_c \tag{8.5}$$

Finally, these updated $\hat{\beta}$ and $\hat{\gamma}$ are used as parameters for the batch normalization: $BN(F_{i,c,h,w}; \hat{\gamma}_c, \hat{\beta}_c))$. We stress that we freeze all ResNet parameters, including $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, during training. In Fig. 8.2, we visualize the difference between the computational flow of the original batch normalization and our proposed modification. As explained in section 8.1.1, a ResNet consists of four stages of computation, each subdivided in several residual blocks. In each block, we apply CBN to the three convolutional layers, as highlighted in Fig. 8.3.

CBN is a computationally efficient and powerful method to modulate neural activations; It enables the linguistic embedding to manipulate entire feature maps by scaling them up or down, negating them, or shutting them off, etc. As there

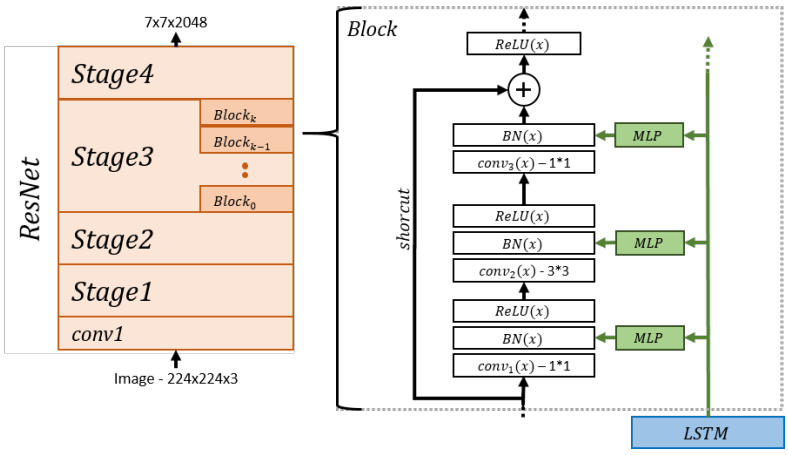


Figure 8.3 – An overview of the MODERN architecture conditioned on the language embedding. MODERN modulates the batch norm parameters in all residual blocks.

only two parameters per feature map, the total number of BN parameters comprise less than 1% of the total number of parameters of a pre-trained ResNet. This makes CBN a very scalable method compared to conditionally predicting the weight matrices (or a low-rank approximation to that).

8.3 Experimental setting

We evaluate the proposed conditional batch normalization on two VQA tasks. In the next section, we outline these tasks and describe the neural architectures we use for our experiments. The source code for our experiments is available at https://github.com/GuessWhatGame. The hyperparameters are also provided in Table 8.1 and 8.2 for GuessWhat?! and VQA, respectively.

8.3.1 VQA

The Visual Question Answering (VQA) task consists of open-ended questions about real images. Answering these questions requires an understanding of vision, language and commonsense knowledge. In this paper, we focus on VQAv1 dataset (Antol et al., 2015), which contains 614K questions on 204K images.

Table 8.1 – GuessWhat?! Oracle hyperparameters

Question	word embedding size	300
	number of LSTM	1
	number of LSTM hidden units	1024
	use Glove	False
Object category	number of categories	90
	category look-up table dimension	512
Crop	crop size	224x224x3
	surrounding factor	1.1
CBN	selected blocks	all
	number of MLP hidden units	512
	ResNet	ResNet-50v1
Fusion block	number of MLP hidden units	512
Optimizer	Name	Adam
	Learning rate	1e-4
	Clip value	3
	number of epoch	10
	batch size	32

Our baseline architecture first obtains a question embedding $\boldsymbol{e_q}$ by an LSTM-network, as further detailed in section 8.1.3. For the image, we extract the feature maps F of the last layer of ResNet-50 (before the pooling layer). For input of size $224x224$ these feature maps are of size $7x7$, and we incorporate a spatial attention mechanism, conditioned on the question embedding $\boldsymbol{e}_q$, to pool over the spatial dimensions. Formally, given a feature maps $\mathbf{F}_{i,\cdot,\cdot,\cdot}$ and question embedding $\boldsymbol{e}_q$, we

Table 8.2 – VQA hyperparameters

Question	word embedding size	300
	number of LSTM	2
	number of LSTM hidden units	1024
	use Glove	True (dim300)
Image	image size	224x224x3
	attention mechanism	spatial
	number of units for attention	512
CBN	selected blocks	all
	number of MLP hidden units	512
	ResNet	ResNet-50v1
Fusion block	fusion embedding size	1024
	number of MLP hidden units	512
	number of answers	2000
Optimizer	Name	Adam
	Learning rate	2e-4
	Clip value	5
	number of epoch	20
	batch size	32

obtain a visual embedding $\boldsymbol{e}_v$ as follows:

$$\boldsymbol{\xi}_{w,h} = MLP([\mathbf{F}_{i,\cdot,w,h}; \boldsymbol{e}_q]) \quad ; \quad \alpha_{w,h} = \frac{\exp(\boldsymbol{\xi}_{w,h})}{\sum_{w,h} \exp(\boldsymbol{\xi}_{w,h})} \quad ; \quad \boldsymbol{e}_v = \sum_{w,h} \alpha_{w,h} \mathbf{F}_{i,\cdot,w,h} \tag{8.6}$$

where $[\mathbf{F}_{i,\cdot,w,h}; \boldsymbol{e}_q]$ denotes concatenating the two vectors. We use an MLP with one hidden layer and ReLU activations whose parameters are shared along the spatial dimensions. The visual and question embedding are then fused by an element-

wise product (Antol et al., 2015; Kim et al., 2016, 2017) as follows:

$$\text{fuse}(\boldsymbol{e}_q, \boldsymbol{e}_v) = \boldsymbol{P}^T \left((\tanh(\boldsymbol{U}^T \boldsymbol{e}_q)) \circ (\tanh(\boldsymbol{V}^T \boldsymbol{e}_v)) \right) + \boldsymbol{b}_P, \tag{8.7}$$

where $\circ$ denotes an element-wise product, and $\boldsymbol{P}$, $\boldsymbol{U}$ and $\boldsymbol{V}$ are trainable weight matrices and $\boldsymbol{b}_P$ is a trainable bias. The linguistic and perceptual representations are first projected to a space of equal dimensionality, after which a tanh non-linearity is applied. A fused vector is then computed by an element-wise product between the two representations. From this joined embedding we finally predict an answer distribution by a linear layer followed by a softmax activation function.

We will use the described architecture to study the impact CBN when using it in several stages of the ResNet. As our approach can be combined with any existing VQA architecture, we also apply MODERN to MLB(Kim et al. (2016, 2017), a state-of-the-art network for VQA More specifically, this network replaces the classic attention mechanism with a more advanced one that included g glimpses over the image features:

$$\boldsymbol{\xi}^g_{w,h} = \boldsymbol{P}^T_{\alpha^g}(\tanh(\boldsymbol{U}'^T \boldsymbol{q}) \circ \tanh(\boldsymbol{V}'^T \boldsymbol{F}^T_{\boldsymbol{i},\cdot,\boldsymbol{w},\boldsymbol{h}}))) \quad ; \quad \alpha^g_{w,h} = \frac{\exp(\boldsymbol{\xi}^g_{w,h})}{\sum_{w,h} \exp(\boldsymbol{\xi}^g_{w,h})} \tag{8.8}$$

$$\boldsymbol{e}_v = \Big|\Big|_g \sum_{w,h} \alpha^g_{w,h} \mathbf{F}_{i,\cdot,w,h} \tag{8.9}$$

where $\boldsymbol{P}_{\alpha^g}$ is a trainable weight matrix defined for each glimpse g, $\boldsymbol{U}'$ and $\boldsymbol{V}'$ are trainable weight matrices shared among the glimpses and $\|$ concatenate vectors over their last dimension.

Noticeably, MODERN modulates the entire visual processing pipeline and therefore backpropagates through all convolutional layers. This requires much more GPU memory than using extracted features. To feasibly run such experiments on today's hardware, we conduct all experiments in this paper with a ResNet-50.

As for our training procedure, we select the 2k most-common answers from the training set, and use a cross-entropy loss over the distribution of provided answers. We train on the training set, do early-stopping on the validation set, and report the accuracies on the test-dev using the evaluation script provided by Antol et al. (2015).

Table 8.3 – VQA accuracies trained with train set and evaluated on test-dev.

	Answer type	Yes/No	Number	Other	**Overall**
224x224	Baseline	79.45%	36.63%	44.62%	58.05%
	Ft Stage 4	78.37%	34.27%	43.72%	56.91%
	Ft BN	80.18%	35.98%	46.07%	58.98%
	MODERN	81.17%	37.79%	48.66%	60.82%
448x448	MLB [1] with ResNet-50	80.20%	37.73%	49.53%	60.84%
	MLB [2] with ResNet-152	80.95%	38.39%	50.59%	61.73%
	MUTAN + MLB [3]	82.29%	37.27%	48.23%	61.02%
	MCB + Attention [4] with ResNet-50	60.46%	38.29%	48.68%	60.46%
	MCB + Attention [5] with ResNet-152	-	-	-	62.50%
	MODERN	81.38%	36.06%	51.64%	62.16%
	MODERN + MLB [6]	82.17%	38.06%	52.29%	**63.01**%

8.3.2 GuessWhat?!

GuessWhat?! is a cooperative two-player game in which both players see the image of a rich visual scene with several objects. One player – the Oracle – is randomly assigned an object in the scene. This object is not known by the other player – the questioner – whose goal it is to locate the hidden object by asking a series of yes-no questions which are answered by the Oracle (de Vries et al., 2016).

The full dataset is composed of 822K binary question/answer pairs on 67K images. Interestingly, the GuessWhat?! game rules naturally leads to a rich variety of visually grounded questions. As opposed to the VQAv1 dataset, the dataset contains very few commonsense questions that can be answered without the image.

In this paper, we focus on the Oracle task, which is a form of visual question

1. (Kim et al., 2017)
2. (Kim et al., 2017)
3. (Ben-Younes et al., 2017)
4. (Fukui et al., 2016)
5. (Fukui et al., 2016)
6. Kim et al. (2017)

Table 8.4 – Ablation study to investigate the impact of leaving out the lower stages of ResNet.

(a) VQA, higher is better

CBN applied to	Val. accuracy
$\emptyset$	56.12%
Stage 4	57.68%
Stages $3-4$	58.29%
Stages $2-4$	58.32%
All	**58.56**%

(b) GuessWhat?!, lower is better

CBN applied to	Test error
$\emptyset$	29.92%
Stage 4	26.42%
Stages $3-4$	25.24%
Stages $2-4$	25.31%
All	**25.06**%

Table 8.5 – GuessWhat?! test errors for the Oracle model with different embeddings. Lower is better.

	Raw features	ft stage4	Ft BN	CBN
Crop	29.92%	27.48%	27.94%	25.06%
Crop + Spatial + Category	22.55%	22.68%	22.42%	**19.52**%
Spatial + Category	21.5%			

answering in which the answers are limited to yes, no and not applicable. Specifically, the oracle may take as an input the incoming question q, the image I and the target object $o*$. This object can be described with its category c, its spatial location and the object crop.

We outline here the neural network architecture that was reported in the original GuessWhat?! paper (de Vries et al., 2016). First, we crop the initial image by using the target object bounding box object and rescale it to a 224 by 224 square. We then extract the activation of the last convolutional layer after the ReLU (stage4) of a pre-trained ResNet-50. We also embed the spatial information of the crop within the image by extracting an 8-dimensional vector of the location of the bounding

box

$$[x_{min}, y_{min}, x_{max}, y_{max}, x_{center}, y_{center}, w_{box}, h_{box}], \tag{8.10}$$

where w_{box} and h_{box} denote the width and height of the bounding box, respectively. We convert the object category c into a dense category embedding using a learned look-up table. Finally, we use an LSTM to encode the current question $\boldsymbol{q}$. We then concatenate all embeddings into a single vector and feed it as input to a single hidden layer MLP that outputs the final answer distribution using a softmax layer.

8.3.3 Baselines

For VQA, we report the results of two state-of-the-art architectures, namely, Multimodal Compact Bilinear pooling network (MCB) (Fukui et al., 2016) (Winner of the VQA challenge 2016) and MUTAN (Ben-Younes et al., 2017). Both approaches employ an (approximate) bilinear pooling mechanism to fuse the language and vision embedding by respectively using a random projection and a tensor decomposition. In addition, we re-implement and run the MLB model described in Section 8.3.1. When benchmarking state-of-the-art models, we train on the training set, proceed early stopping on the validation set and report accuracy on the test set (test-dev in the case of VQA.)

8.3.4 Results

VQA We report the best validation accuracy of the outlined methods on the VQA task in Table8.3. Note that we use input images of size $224x224$ when we compare MODERN against the baselines (as well as for the ablation study presented in Table 8.4a. Our initial baseline achieves 58.05% accuracy, and we find that finetuning the last layers (*Ft Stage 4*) does not improve this performance (56.91%). Interestingly, just finetuning the batch norm parameters (Ft BN) significantly improves the accuracy to 58.98%. We see another significant performance jump when we condition the batch normalization on the question input (*MODERN*), which improves our baseline with almost 2 accuracy points to 60.82%.

Because state-of-the-art models use images of size 448x448, we also include the results of the baseline architecture on these larger images. As seen in Table8.3, this nearly matches the state of the art results with a 62.15%. As MODERN does

not rely on a specific attention mechanism, we then combine our proposed method with MLB (Kim et al., 2016, 2017) architecture, and observe that outperforms the state-of-the-art MCB model (Fukui et al., 2016) by half a point. Please note that we select MLB (Kim et al., 2016, 2017) over MCB (Fukui et al., 2016) as the latter requires fewer weight parameters and is more stable to train.

Note that the presented results use a ResNet-50 while other models rely on extracted image embedding from a ResNet-152. For sake of comparison, we run the baseline models with extracted image embedding from a ResNet-50. Also for the more advanced MLB architecture, we observe performance gains of approximately 2 accuracy points.

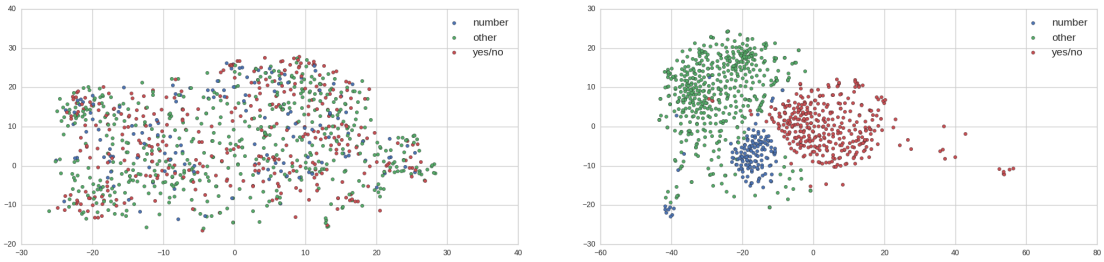
GuessWhat?! We report the best test errors for the outlined method on the Oracle task of GuessWhat?! in Table 8.5. We first compare the results when we only feed the crop of the selected object to the model. We observe the same trend as in VQA. With an error of 25.06%, CBN performs better than than either finetuning the final block (27.48% error) or the batch-norm parameters (27.94% error), which in turn improve over just using the raw features (29.92% error). Note that the relative improvement (5 error points) for CBN is much bigger for GuessWhat?! than for VQA.

We therefore also investigate the performance of the methods when we include the spatial and category information. We observe that finetuning the last layers or BN parameters does not improve the performance, while MODERN improves the best reported test error with 2 points to 19.52% error.

8.3.5 Discussion

By analyzing the results from both VQA and GuessWhat?! experiments, it is possible to have a better insight regarding MODERN capabilities.

MODERN vs Fine tuning In both experiments, MODERN outperforms Ft BN. Both methods update the same ResNet parameters so this demonstrates that it is important to condition on the language representation. MODERN also outperforms Ft Stage 4 on both tasks which shows that the performance gain of MODERN is not due to the increased model capacity.



(a) Feature map projection from raw ResNet

(b) Feature map projection from MODERN

Figure 8.4 – t-SNE projection of feature maps (before attention mechanism) of ResNet and MODERN. Points are colored according to the answer type of VQA. Whilst there are no clusters with raw features, MODERN successfully modulates the image feature towards specific answer types.

Conditional embedding In the provided baselines of the Oracle task of GuessWhat?! (de Vries et al., 2016), the authors observed that the best test error (21.5%) is obtained by only providing the object category and its spatial location. For this model, including the raw features of the object crop actually deteriorates the performance to 22.55% error. This means that this baseline fails to extract relevant information from the images which is not in the handcrafted features. Therefore the Oracle can not answer correctly questions which requires more than the use of spatial information and object category. In the baseline model, the embedding of the crop from a generic ResNet does not help even when we finetune stage 4 or BN. In contrast, applying MODERN helps to better answer questions as the test error drops by 2 points.

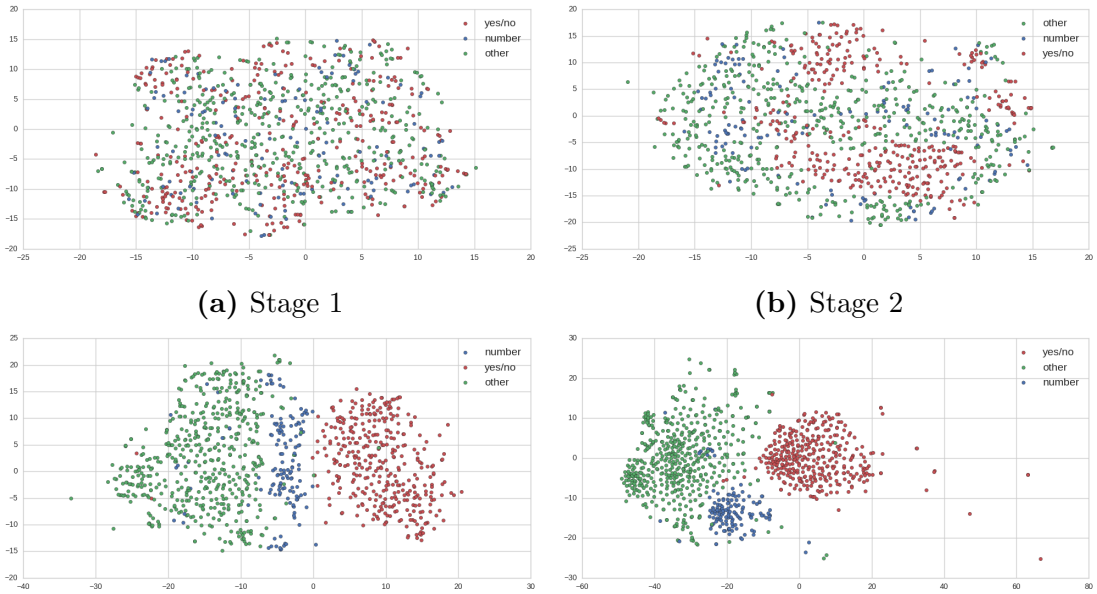
Ablation study We investigate the impact of only modulating the top layers of a ResNet. We report these results in Table 8.4. Interestingly, we observe that the performance slowly decreases when we apply CBN exclusively to later stages. We stress that for best performance it's important to modulate all stages, but if computational resources are limited we recommend to apply it to the two last stages.

Visualizing the representations In order to gain more insight into our proposed fusion mechanism, we compare visualizations of the visual embeddings created by our baseline model and MODERN. We first randomly picked 1000 unique

image/question pairs from the *validation set* of VQA. For the trained MODERN model, we extract image features just before the attention mechanism of MODERN, which we will compare with extracted raw ResNet-50 features and finetune ResNet-50 (Block4 and batchnorm parameters). We first decrease the dimensionality by average pooling over the spatial dimensions of the feature map, and subsequently apply t-SNE (Maaten and Hinton, 2008) to these set of embeddings. We color the points according to the answer type provided by the VQA dataset, and show these visualizations for both models in Fig 8.4 and Fig 8.7. Interestingly, we observe that all answer types are spread out for raw image features and finetuned features. In contrast, the representations of MODERN are cleanly grouped into three answer types. This demonstrates that MODERN successfully disentangles the images representations by answer type which is likely to ease the later fusion process. While finetuning models does cluster features, there is no direct link between those clusters and the answer type. These results indicate that MODERN successfully learns representation that differs from classic finetuning strategies. In Fig. 8.5, we visualize the feature disentangling process stage by stage. It is possible to spot some sub-clusters in the t-SNE representation, as in fact they correspond to image and question pairs which are similar but not explicitly tagged in the VQA dataset. For example, in Fig. 8.6 we highlight pairs where the answer is a color.

8.4 Related work

MODERN is related to a lot of recent work in VQAAntol et al. (2015). The majority of proposed methods use a similar computational pipeline introduced by (Malinowski et al., 2015; Ren et al., 2015). First, extract high-level image features from a ImageNet pretrained convnet, while independently processing the question using RNN. Some work has focused on the top level fusing mechanism of the language and visual vectors. For instance, it was shown that we can improve upon classic concatenation by an element-wise product (Antol et al., 2015; Kim et al., 2016, 2017), Tucker decomposition (Ben-Younes et al., 2017), bilinear pooling (Fukui et al., 2016) or more exotic approaches (Malinowski et al., 2015). Another line of research has investigated the role of attention mechanisms in VQA (Xu and Saenko, 2015; Lu et al., 2016; Yang et al., 2016b). The authors of Lu et al. (2016)



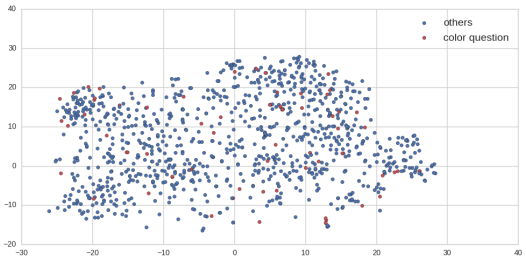
(a) Stage 1 **(b)** Stage 2

(c) Stage 3 **(d)** Stage 4

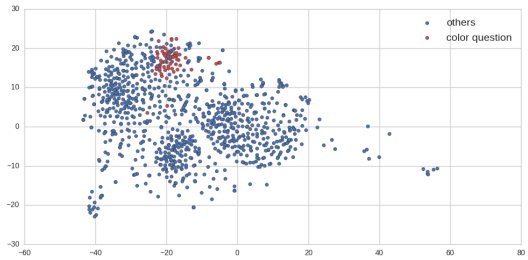
Figure 8.5 – Feature map projection from MODERN for a) stage 1, b) stage 2, c) stage 3, d), stage 4

propose a co-attention model over visual and language embeddings, while Yang et al. (2016b) proposes to stack several spatial attention mechanisms. Although an attention mechanism can be thought of as modulating the visual features by a language, we stress that such mechanism act on the high-level features. In contrast, our work modulates the visual processing from the very start.

MODERN is inspired by conditional instance normalization (CIN) (Dumoulin et al., 2017) that was successfully applied to image style transfer. While previous methods transfered one image style per network, Dumoulin et al. (2017) showed that up to 32 styles could be compressed into a single network by sharing the convolutional filters and learning style-specific normalization parameters. There are notable differences with our work. First, Dumoulin et al. (2017) uses a non-differentiable table lookup for the normalization parameters while we propose a differentiable mapping from the question embedding. Second, we predict a change on the normalization parameters of a pretrained convolutional network while keeping the convolutional filters fixed. In CIN, all parameters, including the transposed convolutional filters, are trained. To the best of our knowledge, this is the first paper to conditionally modulate the vision processing using the normalization pa-



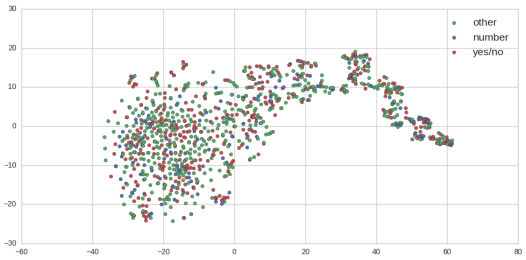
(a) Feature map projection from raw ResNet



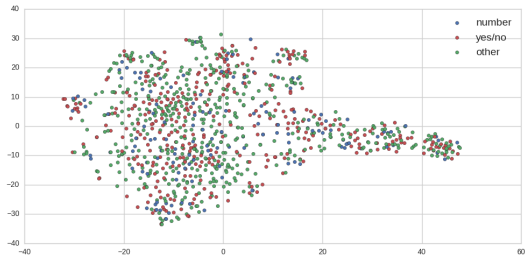
(b) Feature map projection from MODERN

Figure 8.6 – t-SNE projection of feature maps of Reset and MODERN by coloring. Points are colored according to the question type (here, colors) of the image/question pair from the VQA dataset.

rameters.



(a) Feature map projection from ResNet + Block4 Ft



(b) Feature map projection from ResNet + BatchNorm ft

Figure 8.7 – t-SNE projection of feature maps (before attention mechanism) of finetune ResNet. Points are colored according to the answer type of VQA. No answer-type clusters can be observed in both cases.

8.5 Conclusion

In this paper, we introduce Conditional Batch Normalization (CBN) as a novel fusion mechanism to modulate all layers of a visual processing network. Specifically, we applied CBN to a pre-trained ResNet, leading to the proposed MODERN architecture. Our approach is motivated by recent evidence from neuroscience suggesting that language influences the early stages of visual processing. One of the strengths of MODERN is that it can be incorporated into existing architectures,

and our experiments demonstrate that this significantly improves the baseline models. We also found that it is important to modulate the entire visual signal to obtain maximum performance gains.

While this paper focuses on text and images, MODERN can be extended to neural architecture dealing with other modalities such as sound or video. More broadly, CBN can could also be applied to modulate the internal representation of any deep network with respect to any embedding regardless of the underlying task. For instance, signal modulation through batch norm parameters may also be beneficial for reinforcement learning, natural language processing or adversarial training tasks.

9 Prologue to Fourth Article

9.1 Article Details

H. de Vries, K. Shuster, D. Parikh, D. Batra, D. Kiela. Talk the Walk: Navigating New York City Through Grounded Dialogue. *Workshop on Visual Learning and Embodied Agents in Simulation Environments at ECCV, 2018*

Personal contribution I proposed the high-level idea of a dialogue for streetview navigation, which was refined into the Talk the Walk task via discussions with Jason and Douwe. I captured the 360 panoramas (with help from Douwe and Jason), collected the grounded dialogues through Mechanical Turk, and implemented the baselines (with help from Kurt). I also wrote significant parts of the paper, with substantial contributions from Douwe and minor edits from Jason, Dhruv, and Devi.

9.2 Context

In the previous chapters of this thesis, we focused on visually-grounded dialogue tasks in which agents hold a conversation about a static image. The agents in such tasks are passive observers of the environment and therefore do not have the ability to act. The aim of this work was to create an environment in which agents need to work towards a common goal through action, perception and natural language interaction.

9.3 Contributions

This work presents the first large-scale dialogue dataset that is grounded in action and perception. We introduced the "Talk the Walk" task, in which two agents, a tourist and guide, collaborate to have the tourist navigate to a target location in the virtual streets of New York City. By investigating baselines for several sub-tasks, we established the difficulty of the full dialogue problem for current deep and reinforcement learning methods.

9.4 Recent Developments

There has been a growing interest in natural language instruction-following in streetview environments, see e.g. (Chen et al., 2019; Cirik et al., 2018). Other streetview environments have also been open-sourced to facilitate further research in this area (Mirowski et al., 2019).

10 Talk the Walk: Navigating Grids in New York City through Grounded Dialogue

As artificial intelligence plays an ever more prominent role in everyday human lives, it becomes increasingly important to enable machines to communicate via natural language—not only with humans, but also with each other. Learning algorithms for natural language understanding, such as in machine translation and reading comprehension, have progressed at an unprecedented rate in recent years, but still rely on static, large-scale, text-only datasets that lack crucial aspects of how humans understand and produce natural language. Namely, humans develop language capabilities by being embodied in an environment which they can perceive, manipulate and move around in; and by interacting with other humans. Hence, we argue that we should incorporate all three fundamental aspects of human language acquisition—perception, action and interactive communication—and develop a task and dataset to that effect.

We introduce the Talk the Walk dataset, where the aim is for two agents, a "guide" and a "tourist", to interact with each other via natural language in order to achieve a common goal: having the tourist navigate towards the correct location. The guide has access to a map and knows the target location, but does not know where the tourist is; the tourist has a 360-degree view of the world, but knows neither the target location on the map nor the way to it. The agents need to work together through communication in order to successfully solve the task. An example of the task is given in Figure 10.1.

Grounded language learning has (re-)gained traction in the AI community, and much attention is currently devoted to *virtual embodiment*—the development of multi-agent communication tasks in virtual environments—which has been argued to be a viable strategy for acquiring natural language semantics (Kiela et al., 2016). Various related tasks have recently been introduced, but in each case with some limitations. Although visually grounded dialogue tasks (de Vries et al., 2016; Das et al., 2017a) comprise perceptual grounding and multi-agent interaction, their agents are passive observers and do not act in the environment. By contrast,

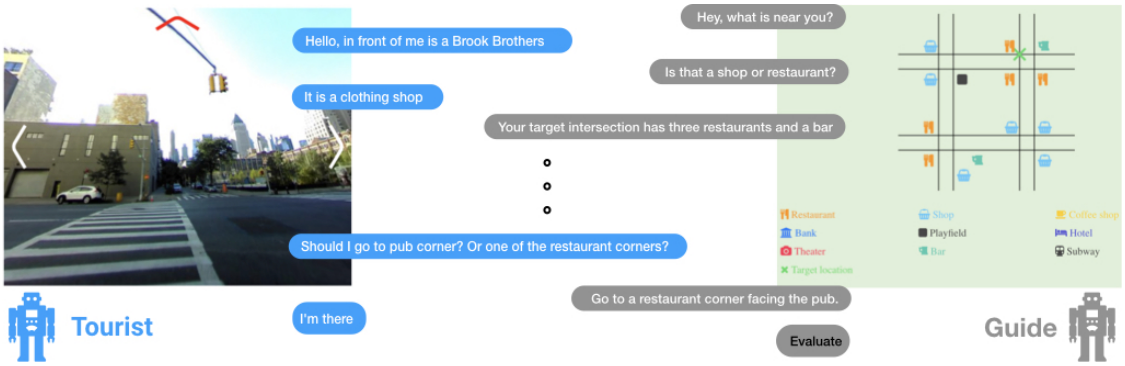


Figure 10.1 – Example of the Talk The Walk task: two agents, a "tourist" and a "guide", interact with each other via natural language in order to have the tourist navigate towards the correct location. The guide has access to a map and knows the target location but not the tourist location, while the tourist does not have a map and is tasked with navigating a 360-degree street view environment.

instruction-following tasks, such as VNL (Anderson et al., 2018), involve action and perception but lack natural language interaction with other agents. Furthermore, some of these works use simulated environments (Das et al., 2018) and/or templated language (Hermann et al., 2017), which arguably oversimplifies real perception or natural language, respectively. See Table 10.1 for a comparison.

Talk The Walk is the first task to bring all three aspects together: *perception* for the tourist observing the world, *action* for the tourist to navigate through the environment, and *interactive dialogue* for the tourist and guide to work towards their common goal. To collect grounded dialogues, we constructed a virtual 2D grid environment by manually capturing 360-views of several neighborhoods in New York City (NYC) [1]. As the main focus of our task is on interactive dialogue, we limit the difficulty of the control problem by having the tourist navigating a 2D grid via discrete actions (turning left, turning right and moving forward). Our street view environment was integrated into ParlAI (Miller et al., 2017) and used to collect a large-scale dataset on Mechanical Turk involving human perception, action and communication.

We argue that for artificial agents to solve this challenging problem, some fundamental architecture designs are missing, and our hope is that this task motivates their innovation. To that end, we focus on the task of localization and develop the novel Masked Attention for Spatial Convolutions (MASC) mechanism. To model the interaction between language and action, this architecture repeatedly conditions

1. We avoided using existing street view resources due to licensing issues.

the spatial dimensions of a convolution on the communicated message sequence.

This work makes the following contributions: 1) We present the first large scale dialogue dataset grounded in action and perception; 2) We introduce the MASC architecture for localization and show it yields improvements for both emergent and natural language; 4) Using localization models, we establish initial baselines on the full task; 5) We show that our best model exceeds human performance under the assumption of "perfect perception" and with a learned emergent communication protocol, and sets a non-trivial baseline with natural language.

10.1 Talk The Walk

We create a perceptual environment by manually capturing several neighborhoods of New York City (NYC) with a 360 camera[2]. Most parts of the city are grid-like and uniform, which makes it well-suited for obtaining a 2D grid. For Talk The Walk, we capture parts of Hell's Kitchen, East Village, the Financial District, Williamsburg and the Upper East Side—see Figure 10.5 for their respective locations within NYC. For each neighborhood, we choose an approximately 5x5 grid and capture a 360 view on all four corners of each intersection, leading to a grid-size of roughly 10x10 per neighborhood.

The tourist's location is given as a tuple (x, y, o), where x, y are the coordinates and o signifies the orientation (north, east, south or west). The tourist can take three actions: turn left, turn right and go forward. For moving forward, we add $(0, 1)$, $(1, 0)$, $(0, -1)$, $(-1, 0)$ to the x, y coordinates for the respective orientations. Upon a turning action, the orientation is updated by $o = (o + d) \mod 4$ where $d = -1$ for left and $d = 1$ for right. If the tourist moves outside the grid, we issue a warning that they cannot go in that direction and do not update the location. Moreover, tourists are shown different types of transitions: a short transition for actions that bring the tourist to a different corner of the *same* intersection; and a longer transition for actions that bring them to a new intersection.

The guide observes a map that corresponds to the tourist's environment. We exploit the fact that urban areas like NYC are full of local businesses, and over-

2. A 360fly 4K camera.

Table 10.1 – Talk The Walk grounds human generated dialogue in (real-life) perception and action.

Project	Perception	Action	Language	Dial.	Size	Acts
Visual Dialog [3]	Real	✗	Human	✓	120k dialogues	20
GuessWhat [4]	Real	✗	Human	✓	131k dialogues	10
VNL [5]	Real	✓	Human	✗	23k instructions	-
Embodied QA [6]	Simulated	✓	Scripted	✗	5k questions	-
TalkTheWalk	Real	✓	Human	✓	10k dialogues	62

lay the map with these landmarks as localization points for our task. Specifically, we manually annotate each corner of the intersection with a set of landmarks $\Lambda^{x,y} = \{l_0, \ldots, l_K\}$, each coming from one of the following categories:

- Bank
- Bar
- Subway
- Hotel
- Playfield
- Coffee Shop
- Shop
- Restaurant
- Theater

The right-side of Figure 10.1 illustrates how the map is presented. Note that within-intersection transitions have a smaller grid distance than transitions to new intersections. To ensure that the localization task is not too easy, we do not include street names in the overhead map and keep the landmark categories coarse. That is, the dialogue is driven by uncertainty in the tourist's current location and the properties of the target location: if the exact location and orientation of the tourist were known, it would suffice to communicate a sequence of actions.

10.1.1 Task

For the Talk The Walk task, we randomly choose one of the five neighborhoods, and subsample a 4x4 grid (one block with four complete intersections) from the entire grid. We specify the boundaries of the grid by the top-left and bottom-right corners $(x_{min}, y_{min}, x_{max}, y_{max})$. Next, we construct the overhead map of the environment, i.e. $\{\Lambda^{x',y'}\}$ with $x_{min} \leq x' \leq x_{max}$ and $y_{min} \leq y' \leq y_{max}$. We subsequently sample a start location and orientation (x, y, o) and a target location

3. Das et al. (2017a)
4. de Vries et al. (2016)
5. Anderson et al. (2018)
6. Das et al. (2018)

$(x, y)_{tgt}$ at random [7].

The shared goal of the two agents is to navigate the tourist to the target location $(x, y)_{tgt}$, which is only known to the guide. The tourist perceives a "street view" planar projection $S_{x,y,o}$ of the 360 image at location (x, y) and can simultaneously chat with the guide and navigate through the environment. The guide's role consists of reading the tourist description of the environment, building a "mental map" of their current position and providing instructions for navigating towards the target location. Whenever the guide believes that the tourist has reached the target location, they instruct the system to evaluate the tourist's location. The task ends when the evaluation is successful—i.e., when $(x, y) = (x, y)_{tgt}$—or otherwise continues until a total of *three* failed attempts. The additional attempts are meant to ease the task for humans, as we found that they otherwise often fail at the task but still end up close to the target location, e.g., at the wrong corner of the correct intersection.

10.1.2 Data Collection

We crowd-sourced the collection of the dataset on Amazon Mechanical Turk (MTurk). We use the MTurk interface of ParlAI (Miller et al., 2017) to render 360 images via WebGL and dynamically display neighborhood maps with an HTML5 canvas. Detailed task instructions, which were also given to our workers before they started their task, are shown in Figure 10.2 and 10.3. We paired Turkers at random and let them alternate between the tourist and guide role across different HITs.

10.1.3 Dataset Statistics

The Talk The Walk dataset consists of over 10k successful dialogues—see Table 10.2 for the dataset statistics split by neighborhood. Turkers successfully completed 76.74% of all finished tasks (we use this statistic as the human success rate). More than six hundred participants successfully completed at least one Talk The Walk HIT. Although the Visual Dialog (Das et al., 2017a) and GuessWhat (de Vries

7. Note that we do not include the orientation in the target, as we found in early experiments that this led to an unnatural task for humans. Similarly, we explored bigger grid sizes but found these to be too difficult for most annotators.

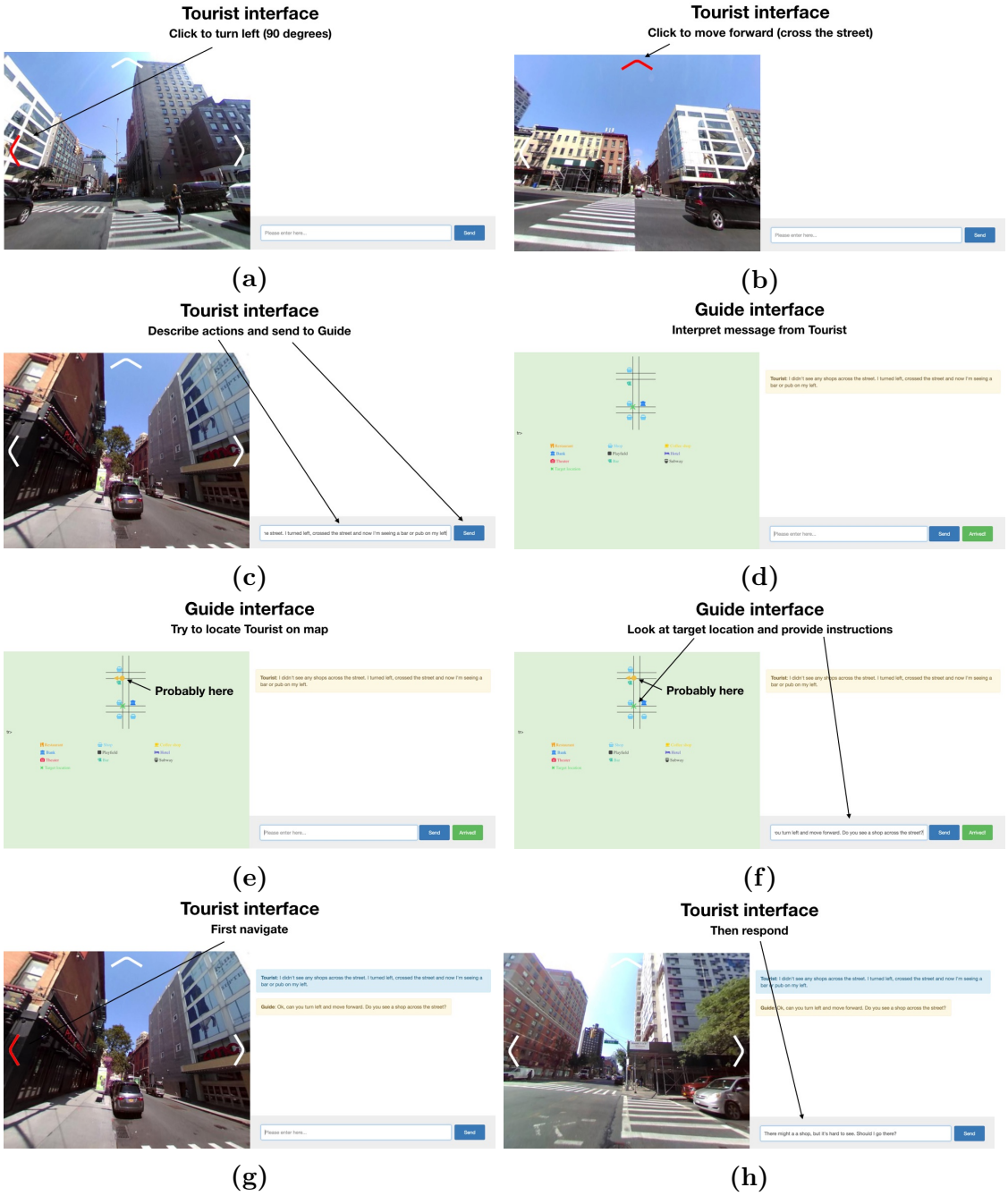


Figure 10.2 – Set of instructions presented to turkers before starting their first task.

et al., 2016) datasets are larger, the collected Talk The Walk dialogs are significantly longer. On average, Turkers needed more than 62 acts (i.e utterances and actions) before they successfully completed the task, whereas Visual Dialog requires

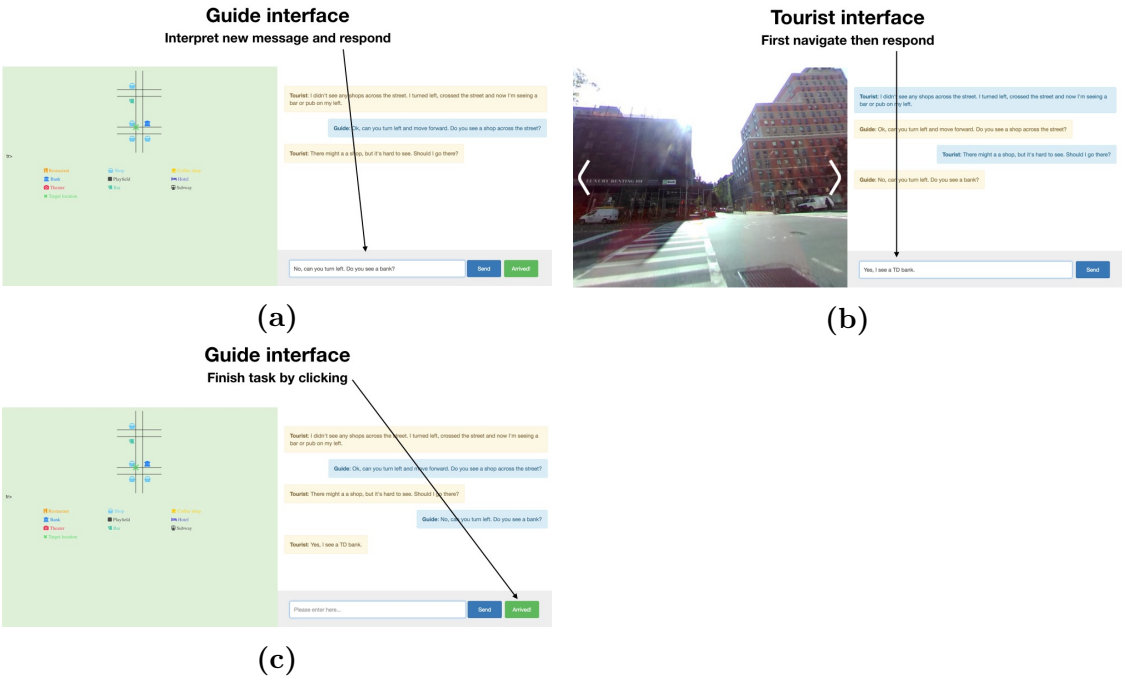


Figure 10.3 – (cont.) Set of instructions presented to turkers before starting their first task.

20 acts. The majority of acts comprise the tourist's actions, with on average more than 44 actions per dialogue. The guide produces roughly 9 utterances per dialogue, slightly more than the tourist's 8 utterances. Turkers use diverse discourse, with a vocabulary size of more than 10K (calculated over all successful dialogues). An example from the dataset is shown in Figure 10.4. The dataset is available at https://github.com/facebookresearch/talkthewalk.

10.2 Experiments

We investigate the difficulty of the proposed task by establishing initial baselines. The final Talk The Walk task is challenging and encompasses several important sub-tasks, ranging from landmark recognition to tourist localization and natural language instruction-giving. Arguably the most important sub-task is localization: without such capabilities the guide can not tell whether the tourist reached the target location. In this work, we establish a minimal baseline for Talk The Walk by utilizing agents trained for localization. Specifically, we let trained

```
Tourist:  ACTION:TURNRIGHT ACTION:TURNRIGHT
Guide:    Hello, what are you near?
Tourist:  ACTION:TURNLEFT ACTION:TURNLEFT ACTION:TURNLEFT
Tourist:  Hello, in front of me is a Brooks Brothers
Tourist:  ACTION:TURNLEFT ACTION:FORWARD ACTION:TURNLEFT ACTION:TURNLEFT
Guide:    Is that a shop or restaurant?
Tourist:  ACTION:TURNLEFT
Tourist:  It is a clothing shop.
Tourist:  ACTION:TURNLEFT
Guide:    You need to go to the intersection in the northwest corner of the map
Tourist:  ACTION:TURNLEFT
Tourist:  There appears to be a bank behind me.
Tourist:  ACTION:TURNLEFT ACTION:TURNLEFT ACTION:TURNRIGHT ACTION:TURNRIGHT
Guide:    Ok, turn left then go straight up that road
Tourist:  ACTION:TURNLEFT ACTION:TURNLEFT ACTION:TURNLEFT ACTION:FORWARD ACTION:TURNRIGHT
          ACTION:FORWARD ACTION:FORWARD ACTION:TURNLEFT ACTION:TURNLEFT ACTION:TURNLEFT
Guide:    There should be shops on two of the corners but you
          need to go to the corner without a shop.
Tourist:  ACTION:FORWARD ACTION:FORWARD ACTION:FORWARD ACTION:TURNLEFT ACTION:TURNLEFT
Guide:    let me know when you get there.
Tourist:  on my left is Radio city Music hall
Tourist:  ACTION:TURNLEFT ACTION:FORWARD ACTION:TURNLEFT ACTION:TURNRIGHT ACTION:TURNRIGHT
Tourist:  I can't go straight any further.
Guide:    ok. turn so that the theater is on your right.
Guide:    then go straight
Tourist:  That would be going back the way I came
Guide:    yeah. I was looking at the wrong bank
Tourist:  I'll notify when I am back at the brooks brothers, and the bank.
Tourist:  ACTION:TURNRIGHT
Guide:    make a right when the bank is on your left
Tourist:  ACTION:FORWARD ACTION:FORWARD ACTION:TURNRIGHT
Tourist:  Making the right at the bank.
Tourist:  ACTION:FORWARD ACTION:FORWARD
Tourist:  I can't go that way.
Tourist:  ACTION:TURNLEFT
Tourist:  Bank is ahead of me on the right
Tourist:  ACTION:FORWARD ACTION:FORWARD ACTION:TURNLEFT
Guide:    turn around on that intersection
Tourist:  I can only go to the left or back the way I just came.
Tourist:  ACTION:TURNLEFT
Guide:    you're in the right place. do you see shops on the corners?
Guide:    If you're on the corner with the bank, cross the street
Tourist:  I'm back where I started by the shop and the bank.
Tourist:  ACTION:TURNRIGHT
Guide:    on the same side of the street?
Tourist:  crossing the street now
Tourist:  ACTION:FORWARD ACTION:FORWARD ACTION:TURNLEFT
Tourist:  there is an I love new york shop across the street on the left from me now
Tourist:  ACTION:TURNRIGHT ACTION:FORWARD
Guide:    ok. I'll see if it's right.
Guide:    EVALUATE_LOCATION
Guide:    It's not right.
Tourist:  What should I be on the look for?
Tourist:  ACTION:TURNRIGHT ACTION:TURNRIGHT ACTION:TURNRIGHT
Guide:    There should be shops on two corners but you need to be on one of the corners
          without the shop.
Guide:    Try the other corner.
Tourist:  this intersection has 2 shop corners and a bank corner
Guide:    yes. that's what I see on the map.
Tourist:  should I go to the bank corner? or one of the shop corners?
          or the blank corner (perhaps a hotel)
Tourist:  ACTION:TURNLEFT ACTION:TURNLEFT ACTION:TURNRIGHT ACTION:TURNRIGHT
Guide:    Go to the one near the hotel. The map says the hotel is a little
          further down but it might be a little off.
Tourist:  It's a big hotel it's possible.
Tourist:  ACTION:FORWARD ACTION:TURNLEFT ACTION:FORWARD ACTION:TURNRIGHT
Tourist:  I'm on the hotel corner
Guide:    EVALUATE_LOCATION
```


Figure 10.4 – Example dialogue from the Talk The Walk dataset.

Figure 10.5 – Map of New York City with red rectangles indicating the captured neighborhoods of the Talk The Walk dataset.

Neighborhood	#success	#failed	#disconnects
Hell's Kitchen	2075	762	867
Williamsburg	2077	683	780
East Village	2035	713	624
Financial District	2042	607	497
Upper East	2081	359	576
Total	10310	3124	3344

Table 10.2 – Dataset statistics split by neighborhood and dialogue status.

tourist models undertake random walks, using the following protocol: at each step, the tourist communicates its observations and actions to the guide, who predicts the tourist's location. If the guide predicts that the tourist is at target, we evaluate its location. If successful, the task ends, otherwise we continue until there have been three wrong evaluations. The protocol is given as pseudo-code in Algorithm 3.

10.2.1 Tourist Localization

The designed navigation protocol relies on a trained localization model that predicts the tourist's location from a communicated message. Before we formalize this localization sub-task in Section 10.2.1, we further introduce two simplifying assumptions—perfect perception and orientation-agnosticism—so as to overcome some of the difficulties we encountered in preliminary experiments.

Perfect Perception Early experiments revealed that perceptual grounding of landmarks is difficult: we set up a landmark classification problem, on which mod-

els with extracted CNN (Kaiming et al., 2016) or text recognition features (Gupta et al., 2016) barely outperform a random baseline—see Section 10.5.3 for full details. This finding implies that localization models from image input are limited by their ability to recognize landmarks, and, as a result, would not generalize to unseen environments. To ensure that perception is not the limiting factor when investigating the landmark-grounding and action-grounding capabilities of localization models, we assume "perfect perception": in lieu of the 360 image view, the tourist is given the landmarks at its current location. More formally, each state observation $S^{x,y,o}$ now equals the set of landmarks at the (x, y)-location, i.e. $S^{x,y,o} = \Lambda^{x,y}$. If the (x, y)-location does not have any visible landmarks, we return a single "empty corner" symbol. We stress that our findings—including a novel architecture for grounding actions into an overhead map, see Section 10.3.2—should carry over to settings without the perfect perception assumption.

Orientation-agnosticism We opt to ignore the tourist's orientation, which simplifies the set of actions to [Left, Right, Up, Down], corresponding to adding [(-1, 0), (1, 0), (0, 1), (0, -1)] to the current (x, y) coordinates, respectively. Note that actions are now coupled to an orientation on the map—e.g. up is equal to going north—and this implicitly assumes that the tourist has access to a compass. This also affects perception, since the tourist now has access to views from all orientations: in conjunction with "perfect perception", implying that only landmarks at the current corner are given, whereas landmarks from different corners (e.g. across the street) are not visible.

Even with these simplifications, the localization-based baseline comes with its own set of challenges. As we show in Section 10.4.1, the task requires communication about a short (random) path—i.e., not only a sequence of observations but also *actions*—in order to achieve high localization accuracy. This means that the guide needs to decode observations from multiple time steps, as well as understand their 2D spatial arrangement as communicated via the sequence of actions. Thus, in order to get to a good understanding of the task, we thoroughly examine whether the agents can learn a communication protocol that simultaneously grounds observations and actions into the guide's map. In doing so, we thoroughly study the role of the communication channel in the localization task, by investigating increasingly constrained forms of communication: from differentiable continuous vectors

to emergent discrete symbols to the full complexity of natural language.

Formalization

The full navigation baseline hinges on a localization model from random trajectories. While we can sample random actions in the emergent communication setup, this is not possible for the natural language setup because the messages are coupled to the trajectories of the human annotators. This leads to slightly different problem setups, as described below.

Emergent language A tourist, starting from a random location, takes $T \geq 0$ *random* actions $A = \{\alpha_0, \ldots, \alpha_{T-1}\}$ to reach target location (x_{tgt}, y_{tgt}). Every location in the environment has a corresponding set of landmarks $\Lambda^{x,y} = \{l_0, \ldots, l_K\}$ for each of the (x, y) coordinates. As the tourist navigates, the agent perceives $T + 1$ state-observations $Z = \{\zeta_0, \ldots, \zeta_T\}$ where each observation ζ_t consists of a set of K landmark symbols $\{l_0^t, \ldots, l_K^t\}$. Given the observations Z and actions A, the tourist generates a message M which is communicated to the other agent. The objective of the guide is to predict the location (x_{tgt}, y_{tgt}) from the tourist's message M.

Natural language In contrast to our emergent communication experiments, we do not take random actions but instead extract actions, observations, and messages from the dataset. Specifically, we consider each tourist utterance (i.e. at any point in the dialogue), obtain the current tourist location as target location $(x, y)_{tgt}$, the utterance itself as message M, and the sequence of observations and actions that took place between the current and previous tourist utterance as Z and A, respectively. Similar to the emergent language setting, the guide's objective is to predict the target location $(x, y)_{tgt}$ models from the tourist message M. We conduct experiments with M taken from the dataset and with M generated from the extracted observations Z and actions A.

10.3 Model

This section outlines the tourist and guide architectures. We first describe how the tourist produces messages for the various communication channels across which the messages are sent. We subsequently describe how these messages are processed by the guide, and introduce the novel Masked Attention for Spatial Convolutions (MASC) mechanism that allows for grounding into the 2D overhead map in order to predict the tourist's location.

10.3.1 The Tourist

For each of the communication channels, we outline the procedure for generating a message M. Given a set of state observations $\{\zeta_0, \ldots, \zeta_T\}$, we represent each observation by summing the L-dimensional embeddings of the observed landmarks, i.e. for $\{\mathbf{o}_0, \ldots, \mathbf{o}_T\}$, $\mathbf{o}_t = \sum_{l \in \zeta_t} E^{\Lambda}(l)$, where E^{Λ} is the landmark embedding lookup table. In addition, we embed action α_t into a L-dimensional embedding $\mathbf{a}_t$ via a look-up table E^A. We experiment with three types of communication channel.

Continuous vectors The tourist has access to observations of several time steps, whose order is important for accurate localization. Because summing embeddings is order-invariant, we introduce a sum over *positionally-gated* embeddings, which, conditioned on time step t, pushes embedding information into the appropriate dimensions. More specifically, we generate an observation message $\mathbf{m}^{obs} = \sum_{t=0}^{T} \text{sigmoid}(\mathbf{g}_t) \odot \mathbf{o}_t$, where $\mathbf{g}_t$ is a learned gating vector for time step t. In a similar fashion, we produce action message $\mathbf{m}^{act}$ and send the concatenated vectors $\mathbf{m} = [\mathbf{m}^{obs}; \mathbf{m}^{act}]$ as message to the guide. We can interpret continuous vector communication as a single, monolithic model because its architecture is end-to-end differentiable, enabling gradient-based optimization for training.

Discrete symbols Like the continuous vector communication model, with discrete communication the tourist also uses separate channels for observations and actions, as well as a sum over positionally gated embeddings to generate observation embedding $\mathbf{h}^{obs}$. We pass this embedding through a sigmoid and generate a message $\mathbf{m}^{obs}$ by sampling from the resulting Bernoulli distributions:

$$\mathbf{h}^{obs} = \sum_{t=0}^{T} \text{sigmoid}(\mathbf{g}_t) \odot \mathbf{o}_t; \qquad \mathbf{m}_i^{obs} \sim \texttt{Bernoulli}(\text{sigmoid}(h_i^{obs}))$$

The action message $\mathbf{m}^{act}$ is produced in the same way, and we obtain the final tourist message $\mathbf{m} = [\mathbf{m}^{obs}; \mathbf{m}^{act}]$ through concatenating the messages.

The communication channel's sampling operation yields the model non-differentiable, so we use policy gradients (Sutton and Barto, 1998; Williams, 1992) to train the parameters θ of the tourist model. That is, we estimate the gradient by

$$\nabla_\theta E_{\mathbf{m}\sim p(\mathbf{h})}[r(\mathbf{m})] = E_{\mathbf{m}}[\nabla_\theta \log p(\mathbf{m})(r(\mathbf{m}) - b)],$$

where the reward function $r(\mathbf{m}) = -\log p(x, y)_{tgt}|\mathbf{m}, \Lambda)$ is the negative guide's loss (see Section 10.3.2) and b a state-value baseline to reduce variance. We use a linear transformation over the concatenated embeddings as baseline prediction, i.e. $b = W^{base}[\mathbf{h}^{obs}; \mathbf{h}^{act}] + \mathbf{b}^{base}$, and train it with a mean squared error loss [8].

Natural Language Because observations and actions are of variable-length, we use an LSTM encoder over the sequence of observations embeddings $[\mathbf{o}_t]_{t=0}^{T+1}$, and extract its last hidden state $\mathbf{h}^{obs}$. We use a separate LSTM encoder for action embeddings $[\mathbf{a}_t]_{t=0}^{T}$, and concatenate both $\mathbf{h}^{obs}$ and $\mathbf{h}^{act}$ to the input of the LSTM decoder at each time step:

$$\begin{aligned} \mathbf{i}_k &= [E^{dec}(w_{k-1}); \mathbf{h}^{obs}; \mathbf{h}^{act}] \qquad \mathbf{h}_k^{dec} = f_{LSTM}(\mathbf{i}_t, \mathbf{h}_{k-1}^{dec}) \\ p(w_k|w_{<k}, A, Z) &= \text{softmax}(W^{out}\mathbf{h}_k^{dec} + \mathbf{b}^{out})_k, \end{aligned} \tag{10.1}$$

where E^{dec} a look-up table, taking input tokens w_k. We train with teacher-forcing, i.e. we optimize the cross-entropy loss: $-\sum_K \log p(w_k|w_{<k}, A, Z)$. At test time, we explore the following decoding strategies: greedy, sampling and a beam-search. We also fine-tune a trained tourist model (starting from a pre-trained model) with policy gradients in order to minimize the guide's prediction loss.

8. This is different from A2C which uses a state-value baseline that is trained by the Bellman residual

10.3.2 The Guide

Given a tourist message M describing their observations and actions, the objective of the guide is to predict the tourist's location on the map. First, we outline the procedure for extracting observation embedding $\mathbf{e}$ and action embeddings $\mathbf{a}_t$ from the message M for each of the types of communication. Next, we discuss the MASC mechanism that takes the observations and actions in order to ground them on the guide's map in order to predict the tourist's location.

Continuous For the continuous communication model, we assign the observation message to the observation embedding, i.e. $\mathbf{e} = \mathbf{m}^{obs}$. To extract the action embedding for time step t, we apply a linear layer to the action message, i.e. $\mathbf{a}_t = W_t^{act}\mathbf{m}^{act} + \mathbf{b}_t^{act}$.

Discrete For discrete communication, we obtain observation $\mathbf{e}$ by applying a linear layer to the observation message, i.e. $\mathbf{e} = W^{obs}\mathbf{m}^{obs} + \mathbf{b}^{obs}$. Similar to the continuous communication model, we use a linear layer over action message $\mathbf{m}^{act}$ to obtain action embedding $\mathbf{a}_t$ for time step t.

Natural Language The message M contains information about observations and actions, so we use a recurrent neural network with attention mechanism to extract the relevant observation and action embeddings. Specifically, we encode the message M, consisting of K tokens w_k taken from vocabulary V, with a bidirectional LSTM:

$$\overrightarrow{\mathbf{h}_k} = f_{LSTM}(\overrightarrow{\mathbf{h}_{k-1}}, E^W(w_k)); \qquad \overleftarrow{\mathbf{h}_k} = f_{LSTM}(\overleftarrow{\mathbf{h}_{k+1}}, E^W(w_k)); \qquad \mathbf{h}_k = [\overrightarrow{\mathbf{h}_k}; \overleftarrow{\mathbf{h}_k}] \tag{10.2}$$

where E^W is the word embedding look-up table. We obtain observation embedding $\mathbf{e}_t$ through an attention mechanism over the hidden states $\mathbf{h}$:

$$s_k = \mathbf{h}_k \cdot \mathbf{c}_t; \quad \mathbf{e}_t = \sum_k \text{softmax}(\mathbf{s})_k \mathbf{h}_k, \tag{10.3}$$

where $\mathbf{c}_0$ is a learned control embedding who is updated through a linear transformation of the previous control and observation embedding: $\mathbf{c}_{t+1} = W^{ctrl}[\mathbf{c}_t; \mathbf{e}_t] +$

$\mathbf{b}^{ctrl}$. We use the same mechanism to extract the action embedding $\mathbf{a}_t$ from the hidden states. For the observation embedding, we obtain the final representation by summing positionally gated embeddings, i.e., $\mathbf{e} = \sum_{t=0}^{T} = \text{sigmoid}(\mathbf{g}_t) \odot \mathbf{e}_t$.

Masked Attention for Spatial Convolutions (MASC)

We represent the guide's map as $U \in \mathbb{R}^{G_1 \times G_2 \times L}$, where in this case $G_1 = G_2 = 4$, where each L-dimensional (x, y) location embedding $\mathbf{u}^{x,y}$ is computed as the sum of the guide's landmark embeddings for that location.

Motivation While the guide's map representation contains only local landmark information, the tourist communicates a trajectory of the map (i.e. actions and observations from multiple locations), implying that directly comparing the tourist's message with the individual landmark embeddings is probably suboptimal. Instead, we want to aggregate landmark information from surrounding locations by imputing trajectories over the map to predict locations. We propose a mechanism for translating landmark embeddings according to state transitions (left, right, up, down), which can be expressed as a 2D convolution over the map embeddings. For simplicity, let us assume that the map embedding U is 1-dimensional, then a left action can be realized through application of the following $3x3$ kernel: $\begin{smallmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{smallmatrix}$, which effectively shifts all values of U one position to the left. We propose to learn such state-transitions from the tourist message through a differentiable attention-mask over the spatial dimensions of a 3x3 convolution.

MASC We linearly project each predicted action embedding $\mathbf{a}_t$ to a 9-dimensional vector $\mathbf{z}_t$, normalize it by a softmax and subsequently reshape the vector into a 3x3 mask Φ_t:

$$\mathbf{z}_t = W^{act}\mathbf{a}_t + b^{act}, \qquad \phi_t = \text{softmax}(\mathbf{z}_t), \qquad \Phi_t = \begin{bmatrix} \phi_t^0 & \phi_t^1 & \phi_t^2 \\ \phi_t^3 & \phi_t^4 & \phi_t^5 \\ \phi_t^6 & \phi_t^7 & \phi_t^8 \end{bmatrix}. \quad (10.4)$$

We learn a 3x3 convolutional kernel $W \in \mathbb{R}^{3 \times 3 \times N \times N}$, with N features, and apply the mask Φ_t to the spatial dimensions of the convolution by first broadcasting its

values along the feature dimensions, i.e. $\hat{\Phi}^{x,y,i,j} = \Phi^{x,y}$, and subsequently taking the Hadamard product: $W_t = \hat{\Phi}_t \odot W$. For each action step t, we then apply a 2D convolution with masked weight W_t to obtain a new map embedding $U_{t+1} = U_t * W_t$, where we zero-pad the input to maintain identical spatial dimensions.

Prediction model We repeat the MASC operation T times (i.e. once for each action), and then aggregate the map embeddings by a sum over positionally-gated embeddings: $\mathbf{u}^{x,y} = \sum_{t=0}^{T} \text{sigmoid}(\mathbf{g}_t) \odot \mathbf{u}_t^{x,y}$. We score locations by taking the dot-product of the observation embedding $\mathbf{e}$, which contains information about the sequence of observed landmarks by the tourist, and the map. We compute a distribution over the locations of the map $p(x, y|M, \Lambda)$ by taking a softmax over the computed scores:

$$s^{x,y} = \mathbf{e} \cdot \mathbf{u}^{x,y}, \qquad p(x, y|M, \Lambda) = \frac{\exp(s^{x,y})}{\sum_{x',y'} \exp(s^{x',y'})}. \tag{10.5}$$

Predicting T While emergent communication models use a fixed length trasjectory T, natural language messages may differ in the number of communicated observations and actions. Hence, we predict T from the communicated message. Specifically, we use a softmax regression layer over the last hidden state $\mathbf{h}_K$ of the RNN, and subsequently sample T from the resulting multinomial distribution:

$$\mathbf{z} = \text{softmax}(W^{tm}\mathbf{h}_K + \mathbf{b}^{tm}); \qquad \hat{T} \sim \texttt{Multinomial}(\mathbf{z}). \tag{10.6}$$

We jointly train the T-prediction model via REINFORCE, with the guide's loss as reward function and a mean-reward baseline.

10.3.3 Comparisons

To better analyze the performance of the models incorporating MASC, we compare against a no-MASC baseline in our experiments, as well as a prediction upper bound.

No MASC We compare the proposed MASC model with a model that does not include this mechanism. Whereas MASC predicts a convolution mask from the tourist message, the "No MASC" model uses W, the ordinary convolutional kernel

Table 10.3 – Accuracy results for tourist localization with emergent language, showing continuous (Cont.) and discrete (Disc.) communication, along with the prediction upper bound. T denotes the length of the path and a ✓ in the "MASC" column indicates that the model is conditioned on the communicated actions.

	MASC	Train			Valid			Test		
		Cont.	Disc.	Upper	Cont.	Disc.	Upper	Cont.	Disc.	Upper
Random		6.25	6.25	6.25	6.25	6.25	6.25	6.25	6.25	6.25
T=0	✗	29.59	28.89	30.23	30.00	30.63	32.50	32.29	33.12	35.00
T=1	✗	39.83	35.40	43.44	35.23	36.56	45.39	35.16	39.53	51.72
	✓	55.64	51.66	62.78	53.12	53.20	65.78	56.09	55.78	72.97
T=2	✗	41.50	40.15	47.84	33.50	37.77	50.29	35.08	41.41	57.15
	✓	67.44	62.24	78.90	64.55	59.34	79.77	66.80	62.15	86.64
T=3	✗	43.48	44.49	45.22	35.40	39.64	48.77	33.11	43.51	55.84
	✓	71.32	71.80	87.92	67.48	65.63	87.45	69.85	69.51	92.41

to convolve the map embedding U_t to obtain U_{t+1}. We also share the weights of this convolution at each time step.

Prediction upper-bound Because we have access to the class-conditional likelihood $p(Z, A|x, y)$, we are able to compute the Bayes error rate (or irreducible error). No model (no matter how expressive) with any amount of data can ever obtain better localization accuracy as there are multiple locations consistent with the observations and actions.

10.4 Results and Discussion

In this section, we describe the findings of various experiments. First, we analyze how much information needs to be communicated for accurate localization in the Talk The Walk environment, and find that a short random path (including actions) is necessary. Next, for emergent language, we show that the MASC architecture can achieve very high localization accuracy, significantly outperforming the baseline

that does not include this mechanism. We then turn our attention to the natural language experiments, and find that localization from human utterances is much harder, reaching an accuracy level that is below communicating a single landmark observation. We show that generated utterances from a conditional language model leads to significantly better localization performance, by successfully grounding the utterance on a single landmark observation (but not yet on multiple observations and actions). Finally, we show performance of the localization baseline on the full task, which can be used for future comparisons to this work.

10.4.1 Analysis of Localization Task

Task is not too easy The upper-bound on localization performance in Table 10.3 suggest that communicating a single landmark observation is not sufficient for accurate localization of the tourist (∼35% accuracy). This is an important result for the full navigation task because the need for two-way communication disappears if localization is too easy; if the guide knows the exact location of the tourist it suffices to communicate a list of instructions, which is then executed by the tourist. The uncertainty in the tourist's location is what drives the dialogue between the two agents.

Importance of actions We observe that the upperbound for only communicating observations plateaus around 57% (even for $T = 3$ actions), whereas it exceeds 90% when we also take actions into account. This implies that, at least for random walks, it is essential to communicate a trajectory, including observations and *actions*, in order to achieve high localization accuracy.

10.4.2 Emergent Language Localization

We first report the results for tourist localization with **emergent language** in Table 10.3.

MASC improves performance The MASC architecture significantly improves performance compared to models that do not include this mechanism. For instance, for $T = 1$ action, MASC already achieves 56.09 % on the test set and this further increases to 69.85% for $T = 3$. On the other hand, no-MASC models hit

Table 10.4 – **Localization** accuracy of tourist communicating in natural language.

Model	Decoding	Train	Valid	Test
Random		6.25	6.25	6.25
Human utterances		23.46	15.56	16.17
Supervised	sampling	17.19	12.23	12.43
	greedy	34.14	29.90	29.05
	beam (size: 4)	26.21	22.53	25.02
Policy Grad.	sampling	29.67	26.93	27.05
	greedy	29.23	27.62	27.30

Table 10.5 – **Full task** evaluation of localization models using protocol of Algorithm 3.

	Train	Valid	Test	#steps
Random	18.75	18.75	18.75	-
Human	76.74	76.74	76.74	15.05
Best Cont.	89.44	86.35	88.33	34.47
Best Disc.	86.23	82.81	87.08	34.83
Best NL	39.65	39.68	50.00	39.14

a plateau at 43%. In Section 10.5.2, we analyze learned MASC values, and show that communicated actions are often mapped to corresponding state-transitions.

Continuous vs discrete We observe similar performance for continuous and discrete emergent communication models, implying that a discrete communication channel is not a limiting factor for localization performance.

10.4.3 Natural Language Localization

We report the results of tourist localization with **natural language** in Table 10.4. We compare accuracy of the guide model (with MASC) trained on utterances

from (i) humans, (ii) a supervised model with various decoding strategies, and (iii) a policy gradient model optimized with respect to the loss of a frozen, pre-trained guide model on *human* utterances.

Human utterances Compared to emergent language, localization from human utterances is much harder, achieving only 16.17% on the test set. Here, we report localization from a single utterance, but in Section 10.5.1 we show that including up to five dialogue utterances only improves performance to 20.33%. We also show that MASC outperform no-MASC models for natural language communication.

Generated utterances We also investigate generated tourist utterances from conditional language models. Interestingly, we observe that the supervised model (with greedy and beam-search decoding) as well as the policy gradient model leads to an improvement of more than 10 accuracy points over the human utterances. However, their level of accuracy is slightly below the baseline of communicating a single observation, indicating that these models only learn to ground utterances in a single landmark observation.

Better grounding of generated utterances We analyze natural language samples in Table 10.6, and confirm that, unlike human utterances, the generated utterances are talking about the observed landmarks. This observation explains why the generated utterances obtain higher localization accuracy. The current language models are most successful when conditioned on a single landmark observation; We show in Section 10.5.1 that performance quickly deteriorates when the model is conditioned on more observations, suggesting that it can not produce natural language utterances about multiple time steps.

10.4.4 Localization-based Baseline

Table 10.5 shows results for the best localization models on the **full task**, evaluated via the random walk protocol defined in Algorithm 3.

Comparison with human annotators Interestingly, our best localization model (continuous communication, with MASC, and $T = 3$) achieves 88.33% on the test set and thus exceed human performance of 76.74% on the full task. While emergent

Algorithm 3 Performance evaluation of location prediction model on full Talk The Walk setup

```
procedure EVALUATE(tourist, guide, T, x_tgt, y_tgt, maxsteps)
    x, y ← randint(0, 3), randint(0, 3)          ▷ initialize with random location
    features, actions ← array(), array()
    features[0] ← features at location (x, y)
    for t = 0; t < T; t++ do                      ▷ create T-sized feature buffer
        action ← uniform sample from action set
        x, y ← update location given action
        features[t + 1] ← features at location (x, y)
        actions[t] ← action

    for i = 0; i < maxsteps; i++ do
        M ← tourist(features, actions)
        p(x, y|·) ← guide(M)
        x_pred, y_pred ← sample from p(x, y|·)
        if x_pred, y_pred == x_tgt, y_tgt then          ▷ target predicted
            if locations[0] == x_tgt, y_tgt then
                return True
            else
                numevaluations ← numevaluations − 1
                if numevaluations ≤ 0 then
                    return False
        features ← features[1 :]
        actions ← actions[1 :]

        x, y ← update location given action          ▷ take new action
        features[t + 1] ← features at location (x, y)
        actions[t] ← action
```

models appear to be stronger localizers, humans might cope with their localization uncertainty through other mechanisms (e.g. better guidance, bias towards taking particular paths, etc). The simplifying assumption of perfect perception also helps.

Number of actions Unsurprisingly, humans take fewer steps (roughly 15) than our best random walk model (roughly 34). Our human annotators likely used some form of guidance to navigate faster to the target.

Method	Decoding	Utterance
Observations		(Bar)
Actions		-
Human		a field of some type
Supervised	greedy	at a bar
	sampling	sec just hard to tell which is a restaurant ?
	beam search	im at a bar
Policy Grad.	greedy	bar from bar from bar and rigth rigth bulding bulding
	sampling	which bar from bar from bar and bar rigth bulding bulding..

Table 10.6 – Samples from the tourist models communicating in natural language. Contrary to the human generated utterance, the supervised model with greedy and beam search decoding produces an utterance containing the current state observation (bar). Also the reinforcement learning model mentions the current observation but has lost linguistic structure. The fact that these localization models are better grounded in observations than human utterances explains why they obtain higher localization accuracy.

10.5 Additional Experiments and Analysis

In addition to the main experiments and results, we present additional experiments and analysis that provide further insights into the Talk The Walk task. In the following subsections, we present the results of additional natural language experiments, visualize the MASC predictions for an emergent communication protocol, and investigate the difficulty of the landmark recognition problem.

10.5.1 Natural Language Experiments

First, we investigate the sensitivity of tourist generation models to the trajectory length, finding that the model conditioned on a single observation (i.e. $T = 0$) achieves best performance. In the next subsection, we further analyze localization models from human utterances by investigating MASC and no-MASC models with increasing dialogue context.

Tourist Generation Models

After training the supervised tourist model (conditioned on observations and action from human expert trajectories), there are two ways to train an accom-

Table 10.7 – Full task performance of localization models trained on human and random trajectories. There are small benefits for training on random trajectories, but the most important hyper-parameter is to condition the tourist utterance on a single observation (i.e. trajectories of size $T = 0$.) at evaluation time.

Trajectories	T	Train	Valid	Test
Random		18.75	18.75	18.75
Human	0	38.21	40.93	40.00
	1	21.82	23.75	25.62
	2	19.77	24.68	23.12
	3	18.95	20.93	20.00
Random	0	39.65	39.68	50.00
	1	28.99	30.93	25.62
	2	27.04	19.06	19.38
	3	20.28	20.93	22.50

panying guide model. We can optimize a location prediction model on either (i) extracted human trajectories (as in the localization setup from human utterances) or (ii) on all random paths of length T (as in the full task evaluation). Here, we investigate the impact of (1) using either human or random trajectories for training the guide model, and (2) the effect of varying the path length T during the full-task evaluation. For random trajectories, guide training uses the same path length T as is used during evaluation. We use a pre-trained tourist model with greedy decoding for generating the tourist utterances. Table 10.7 summarizes the results.

Human vs random trajectories We only observe small improvements for training on random trajectories. Human trajectories are thus diverse enough to generalize to random trajectories.

Effect of path length There is a strong negative correlation between task success and the conditioned trajectory length. We observe that the full task performance quickly deteriorates for both human and random trajectories. This suggests

Table 10.8 – Localization performance using pre-trained tourist (via imitation learning) with beam search decoding of varying beam size. Locations and observations extracted from human trajectories. Larger beam-sizes lead to worse localization performance.

Beam size	Train	Valid	Test
Random	6.25	6.25	6.25
1	34.14	29.90	29.05
2	26.24	23.65	25.10
4	23.59	22.87	21.80
8	20.31	19.24	20.87

that the tourist generation model can not produce natural language utterances that describe multiple observations and actions. Although it is possible that the guide model can not process such utterances, this is not very likely because the MASC architectures handles such messages successfully for emergent communication.

Effect of beam-size

We report localization performance of tourist utterances generated by beam search decoding of *varying beam size* in Table 10.8. We find that performance decreases from 29.05% to 20.87% accuracy on the test set when we increase the beam-size from one to eight.

Localization from Human Utterances

We conduct an ablation study for MASC on natural language with varying dialogue context. Specifically, we compare localization accuracy of MASC and no-MASC models trained on the last [1, 3, 5] utterances of the dialogue (including guide utterances). We report these results in Table 10.9. In all cases, MASC outperforms the no-MASC models by several accuracy points. We also observe that mean predicted $\hat{T}$ (over the test set) increases from 1 to 2 when more dialogue context is included.

#utterances	MASC	Train	Valid	Test	$E[T]$
Random		6.25	6.25	6.25	-
1	✗	23.95	13.91	13.89	0.99
	✓	23.46	15.56	16.17	1.00
3	✗	26.92	16.28	16.62	1.00
	✓	20.88	17.50	18.80	1.79
5	✗	25.75	16.11	16.88	1.98
	✓	30.45	18.41	20.33	1.99

Table 10.9 – Localization given last {1, 3, 5} dialogue utterances (including the guide). We observe that (1) performance increases when more utterances are included; and (2) MASC outperforms no-MASC in all cases; and (3) mean $\hat{T}$ increases when more dialogue context is included.

10.5.2 Visualizing MASC predictions

Figure 10.6 shows the MASC values for a learned model with emergent discrete communications and $T = 3$ actions. Specifically, we look at the predicted MASC values for different action sequences taken by the tourist. We observe that the first action is always mapped to the correct state-transition, but that the second and third MASC values do not always correspond to right state-transitions.

10.5.3 Landmark Classification

While the guide has access to the landmark labels, the tourist needs to recognize these landmarks from raw perceptual information. In this section, we study landmark classification as a supervised learning problem to investigate the difficulty of perceptual grounding in Talk The Walk.

The Talk The Walk dataset contains a total of 307 different landmarks divided among nine classes, see Figure 10.8 for how they are distributed. The class distribution is fairly imbalanced, with shops and restaurants as the most frequent landmarks and relatively few play fields and theaters. We treat landmark recognition as a multi-label classification problem as there can be multiple landmarks on

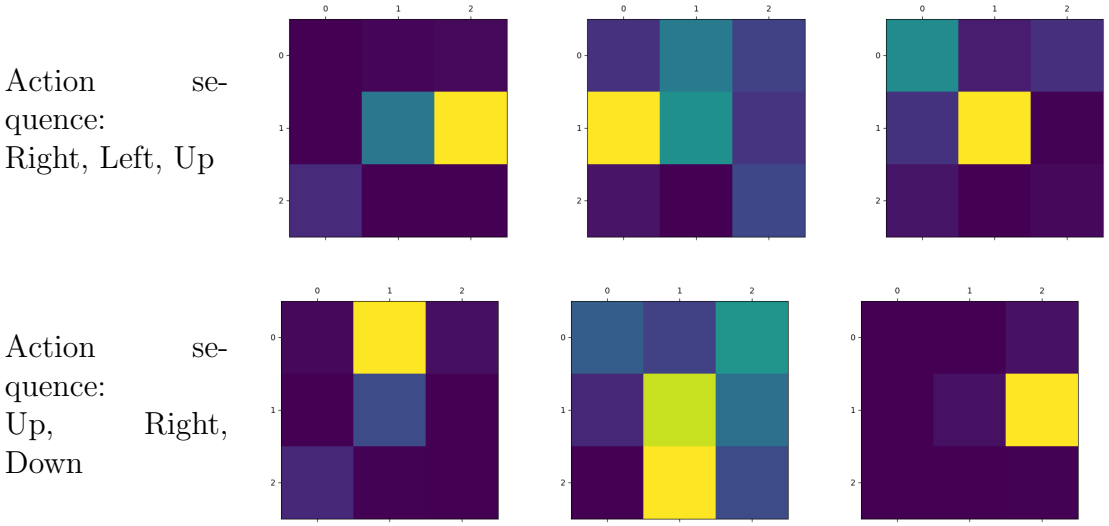


Figure 10.6 – We show MASC values of two action sequences for tourist localization via *discrete* communication with $T = 3$ actions. In general, we observe that the first action always corresponds to the correct state-transition, whereas the second and third are sometimes mixed. For instance, in the top example, the first two actions are correctly predicted but the third action is not (as the MASC corresponds to a "no action"). In the bottom example, the second action appears as the third MASC.

a corner [9].

For the task of landmark classification, we extract the relevant views of the 360 image from which a landmark is visible. Because landmarks are labeled to be on a specific corner of an intersection, we assume that they are visible from one of the orientations facing away from the intersection. For example, for a landmark on the northwest corner of an intersection, we extract views from both the north and west direction. The orientation-specific views are obtained by a planar projection of the full 360-image with a small field of view (60 degrees) to limit distortions. To cover the full field of view, we extract two images per orientation, with their horizontal focus point 30 degrees apart. Hence, we obtain eight images per 360 image with corresponding orientation $v \in \{N1, N2, E1, E2, S1, S2, W1, W2\}$.

We run the following pre-trained feature extractors over the extracted images:

ResNet We resize the extracted view to a 224x224 image and pass it through a ResNet-152 network Kaiming et al. (2016) to obtain a 2048-dimensional

9. Strictly speaking, this is more general than a multi-label setup because a corner might contain multiple landmarks of the same class.

Table 10.10 – Results for **landmark classification**.

Features	Train loss	Valid Loss	Train F1	Valid F1	Valid prec.	Valid recall
All positive	-	-	-	0.39313	0.26128	1
Random (0.5)	-	-	-	0.32013	0.24132	0.25773
Textrecog	0.01462	0.01837	0.31205	0.31684	0.2635	0.50515
Fasttext	0.00992	0.00994	0.24019	0.31548	0.26133	0.47423
Fasttext (100 dim)	0.00721	0.00863	0.32651	0.28672	0.24964	0.4433
ResNet	0.00735	0.00751	0.17085	0.20159	0.13114	0.58763
ResNet (256 dim)	0.0051	0.00748	0.60911	0.31953	0.27733	0.50515

feature vector $S^{resnet}_{x,y,v} \in \mathbb{R}^{2048}$ from the penultimate layer.

Text Recognition We use a pre-trained text-recognition model Gupta et al. (2016) to extract a set of text messages $S^{text}_{x,y,v} = \{R^{text}_{\beta}\}^{B}_{\beta=0}$ from the images. Local businesses often advertise their wares through key phrases on their storefront, and understanding this text might be a good indicator of the type of landmark. In Figure 10.7, we show the results of running the text recognition module on a few extracted images.

For the text recognition model, we use a learned look-up table E^{text} to embed the extracted text features $\mathbf{e}^{\beta}_{x,y,v} = E^{text}(R^{text}_{\beta})$, and fuse all embeddings of four images through a bag of embeddings, i.e., $\mathbf{e}^{fused} = \sum_{v \in \text{relevant views}} \sum_{\beta} e^{\beta}_{x,y,v}$. We use a linear layer followed by a sigmoid to predict the probability for each class, i.e. sigmoid$(W\mathbf{e}^{fused} + b)$. We also experiment with replacing the look-up embeddings with pre-trained FastText embeddings Bojanowski et al. (2016). For the ResNet model, we use a bag of embeddings over the four ResNet features, i.e. $\mathbf{e}^{fused} = \sum_{v \in \text{relevant views}} S^{resnet}_{x,y,v}$, before we pass it through a linear layer to predict the class probabilities: sigmoid$(W\mathbf{e}^{fused} + b)$. We also conduct experiments where we first apply PCA to the extracted ResNet and FastText features before we feed them to the model.

To account for class imbalance, we train all described models with a binary cross entropy loss weighted by the inverted class frequency. We create a 80-20

class-conditional split of the dataset into a training and validation set. We train for 100 epochs and perform early stopping on the validation loss.

The F1 scores for the described methods in Table 10.10. We compare to an "all positive" baseline that always predicts that the landmark class is visible and observe that all presented models struggle to outperform this baseline. Although 256-dimensional ResNet features achieve slightly better precision on the validation set, it results in much worse recall and a lower F1 score. Our results indicate that perceptual grounding is a difficult task, which easily merits a paper of its own right, and so we leave further improvements (e.g. better text recognizers) for future work.

10.6 Related Work

The Talk the Walk task and dataset facilitate future research on various important subfields of artificial intelligence, including grounded language learning, goal-oriented dialogue research and situated navigation. Here, we describe related previous work in these areas.

Related tasks There has been a long line of work involving related tasks. Early work on task-oriented dialogue dates back to the early 90s with the introduction of the Map Task (Anderson et al., 1991) and Maze Game (Garrod and Anderson, 1987) corpora. Recent efforts have led to larger-scale goal-oriented dialogue datasets, for instance to aid research on visually-grounded dialogue (Das et al., 2017a; de Vries et al., 2016), knowledge-base-grounded discourse (He et al., 2017) or negotiation tasks (Lewis et al., 2017). At the same time, there has been a big push to develop environments for embodied AI, many of which involve agents following natural language instructions with respect to an environment(Artzi and Zettlemoyer, 2013; Yu et al., 2017; Hermann et al., 2017; Mei et al., 2016; Chaplot et al., 2018b,a), following-up on early work in this area (MacMahon et al., 2006; Chen and Mooney, 2011). An early example of navigation using neural networks is (Hadsell et al., 2007), who propose an online learning approach for robot navigation. Recently, there has been increased interest in using end-to-end trainable neural networks for learning to navigate indoor scenes(Gupta et al., 2017b,a) or large cities (Brahmbhatt and Hays, 2017; Mirowski et al., 2018), but, unlike our

work, without multi-agent communication. Also the task of localization (without multi-agent communication) has recently been studied (Chaplot et al., 2018a; Vo et al., 2017).

Grounded language learning Grounded language learning is motivated by the observation that humans learn language embodied (grounded) in sensorimotor experience of the physical world (Barsalou, 2008b; Smith and Gasser, 2005). On the one hand, work in multi-modal semantics has shown that grounding can lead to practical improvements on various natural language understanding tasks (see Baroni, 2016; Kiela, 2017, and references therein). In robotics, researchers dissatisfied with purely symbolic accounts of meaning attempted to build robotic systems with the aim of grounding meaning in physical experience of the world (Roy, 2005; Steels and Hild, 2012). Recently, grounding has also been applied to the learning of sentence representations (Kiela et al., 2017), image captioning (Lin et al., 2014; Xu et al., 2015), visual question answering (Antol et al., 2015; de Vries et al., 2017), visual reasoning (Johnson et al., 2017; Perez et al., 2018), and grounded machine translation (Riezler et al., 2014; Elliott et al., 2016). Grounding also plays a crucial role in the emergent research of multi-agent communication, where, agents communicate (in natural language or otherwise) in order to solve a task, with respect to their shared environment (Lazaridou et al., 2016; Das et al., 2017c; Mordatch and Abbeel, 2017; Evtimova et al., 2017; Lewis et al., 2017; Strub et al., 2017; Kottur et al., 2017).

10.7 Conclusion

We introduced the Talk The Walk task and dataset, which consists of crowd-sourced dialogues in which two human annotators collaborate to navigate to target locations in the virtual streets of NYC. For the important localization sub-task, we proposed MASC—a novel grounding mechanism to learn state-transition from the tourist's message—and showed that it improves localization performance for emergent and natural language. We use the localization model to provide baseline numbers on the Talk The Walk task, in order to facilitate future research.

Figure 10.7 – Result of running the text recognizer of (Gupta et al., 2016) on four examples of the Hell's Kitchen neighborhood. **Top row:** two positive examples. **Bottom row:** example of false negative (left) and many false positives (right)

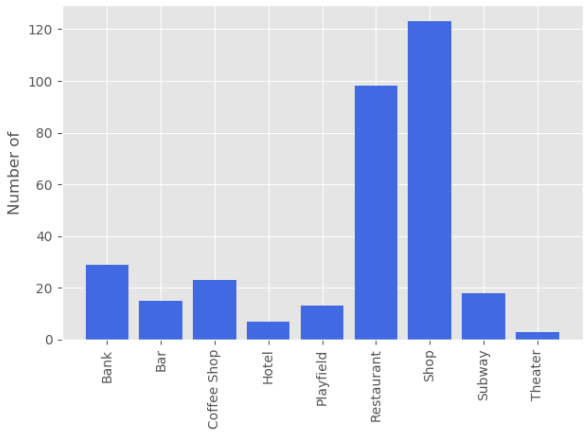


Figure 10.8 – Frequency of landmark classes

11 Conclusion

This thesis investigated deep learning and reinforcement learning methods for multi-modal dialogue problems. As prior work has demonstrated the difficulty of dialogue modelling for open-ended domains, this work focused on this topic in a more constrained setting. Specifically, we argued in favor of goal-oriented tasks because they (i) limit the scope of the conversation and (ii) come with a well-defined objective. Besides these simplifications, we also argued for studying dialogue problems in a multi-modal environment, so as to enable conversational agents to ground their language use into the sensory-motor experience.

Chapter 4 introduced GuessWhat, a two-player guessing game where a so-called questioner agent aims to identify an unknown object via a series of yes-no questions. Our main contribution consisted of the collection of more than 150k human-played games. Using these human-generated dialogues, we prototyped deep learning baselines for the three sub-problems of the GuessWhat game: the oracle task, the questioner task, and the guesser task. The results of these experiments provided insight into the nature of these sub-problems and inspired us to develop the methods presented in Chapter 6 and 8.

In Chapter 6, we pointed out the shortcomings of modeling the questioner agent as a supervised learning task and instead proposed to frame it as a sequential decision problem. To that end, we set up the question generation problem as a reinforcement learning task—with the pretrained oracle and guesser model being part of the environment—and explicitly optimize the agent to find the correct object. Compared to the supervised baseline, the RL agent achieved significantly higher task success and suffered less from repeating questions. On the other hand, we also found its generated questions to be less diverse, as well as tailored to the answering capabilities of the oracle model.

Chapter 8 improved the oracle model by introducing the idea of Conditional Batch Normalization (CBN). Specifically, we proposed to use this conditioning mechanism to let the question modulate the intermediate activations of a pre-

trained CNN. While Chapter 4 found that adding raw image features of either the cropped object or entire scene deteriorated oracle performance, we showed that adding this information improves performance if we apply a modulated CNN. In addition, we showed that the proposed model also improved performance for the Visual Question Answering benchmark. CBN is a very general conditioning mechanism which has been effectively transferred to a wide range of applications, including class-conditional image generation, speech recognition, and few-shot image learning.

Chapter 10 introduced Talk The Walk, a new grounded dialogue task where two agents—a "tourist" and a "guide"—hold a conversation in order to navigate the tourist to a target location. Compared to GuessWhat, Talk The Walk increases the level of complexity along several dimensions. First, both agents in Talk The Walk are able to communicate in free-form natural language, whereas the oracle agent in GuessWhat is restricted to yes, no, and not applicable answers. Second, the questioner and oracle agents are passive observers of the environment (the image), whereas the tourist agent is actively taking actions in this environment. In other words, the Talk the Walk dialogues are linked to both observations and actions. Because of this increased complexity, we established baselines for the full task by training models for the sub-task of tourist localization. Using synthetic language, we achieved high localization accuracy by incorporating the MASC mechanism, which we found to be effective for grounding tourist actions into the guide's map. We found that localization from extracted human utterances is much harder because they often do not talk about the observed landmarks.

Looking forward, there are many exciting directions to extend the work presented in this thesis. Below, I outline a few topics which I believe are fundamental to further advancing this sub-field.

One pressing issue in goal-oriented dialogue is language drifting, a problem which is especially relevant for agents that are optimized with reinforcement learning. As information-seeking dialogue problems are constructed to transfer information between two agents, there exists many synthetic communication protocols that achieve this goal. For that reason, RL agents optimizing for task success tend to converge to a communication protocol that does not resemble natural language, even if the model started from a supervised language model (Kottur et al., 2017; Lewis et al., 2017). In this thesis, we aimed to address this issue by freezing the

pre-trained answering model so that the optimized questioner agent was forced to ask questions that the oracle understood. Others have proposed to optimize both agents with RL but to regularize the produced utterances by their likelihood on a pre-trained language model. So far, the results of these proposals are not fully satisfactory and more research is needed on how to effectively combine supervised learning and reinforcement learning techniques for dialogue tasks. It is also possible that the language drifting problem is the result of optimizing for a single information-seeking task and might be alleviated by bundling a number of dialogue tasks. Human language use is not optimized for a single task either so I believe it is important to train and test our dialogue agents on multiple information-seeking problems too.

Language-and-vision grounding is another research topic that I believe deserves further attention from the community. For many language-and-vision tasks it has recently been reported that language priors overwrite the visual processing (Tejas et al., 2017; Shekhar et al., 2017; Massiceti et al., 2018). The work presented in this thesis has made progress in this direction by introducing better neural architectures and less biased language-vision data sets. However, much work remains to be done. For example, completely eliminating all biases from the dataset seems unrealistic, suggesting the need for developing models that can handle such biases. Very much related to this issue is that current neural architectures lack systematicity and, therefore, fail to generalize to samples outside the training distribution. For example, in the context of visual relationship learning, these systems have difficulties manipulating previously acquired knowledge about relationships (such as "on top of") and objects (e.g. "vase" and "table") into novel configurations ("a vase on top of a table"). I think that this type of compositional generalization is a prerequisite for successful deployment of language-vision models in the real world.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Agrawal, A., Batra, D., and Parikh, D. (2016). Analyzing the Behavior of Visual Question Answering Models. *arXiv preprint arXiv:1606.07356*.

Ahn, L. V. and Dabbish, L. (2004). Labeling images with a computer game. In *Proc. of the SIGCHI conference on Human factors in computing systems*. ACM.

Ahn, L. V., Liu, R., and Blum, M. (2006). Peekaboom: a game for locating objects in images. In *Proc. of the SIGCHI conference on Human Factors in computing systems*. ACM.

Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., Bastien, F., Bayer, J., Belikov, A., Belopolsky, A., Bengio, Y., Bergeron, A., Bergstra, J., Bisson, V., Bleecher Snyder, J., Bouchard, N., Boulanger-Lewandowski, N., Bouthillier, X., de Brébisson, A., Breuleux, O., Carrier, P.-L., Cho, K., Chorowski, J., Christiano, P., Cooijmans, T., Côté, M.-A., Côté, M., Courville, A., Dauphin, Y. N., Delalleau, O., Demouth, J., Desjardins, G., Dieleman, S., Dinh, L., Ducoffe, M., Dumoulin, V., Ebrahimi Kahou, S., Erhan, D., Fan, Z., Firat, O., Germain, M., Glorot, X., Goodfellow, I., Graham, M., Gulcehre, C., Hamel, P., Harlouchet, I., Heng, J.-P., Hidasi, B., Honari, S., Jain, A., Jean, S., Jia, K., Korobov, M., Kulkarni, V., Lamb, A., Lamblin, P., Larsen, E., Laurent, C., Lee, S., Lefrancois, S., Lemieux, S., Léonard, N., Lin,

Z., Livezey, J. A., Lorenz, C., Lowin, J., Ma, Q., Manzagol, P.-A., Mastropietro, O., McGibbon, R. T., Memisevic, R., van Merriënboer, B., Michalski, V., Mirza, M., Orlandi, A., Pal, C., Pascanu, R., Pezeshki, M., Raffel, C., Renshaw, D., Rocklin, M., Romero, A., Roth, M., Sadowski, P., Salvatier, J., Savard, F., Schlüter, J., Schulman, J., Schwartz, G., Serban, I. V., Serdyuk, D., Shabanian, S., Simon, E., Spieckermann, S., Subramanyam, S. R., Sygnowski, J., Tanguay, J., van Tulder, G., Turian, J., Urban, S., Vincent, P., Visin, F., de Vries, H., Warde-Farley, D., Webb, D. J., Willson, M., Xu, K., Xue, L., Yao, L., Zhang, S., and Zhang, Y. (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Anderson, A. H., Bader, M., Bard, E. G., Boyle, E., Doherty, G., Garrod, S., Isard, S., Kowtko, J., McAllister, J., Miller, J., Sotillo, C., Thompson, H. S., and Weinert, R. (1991). The hcrc map task corpus. *Language and Speech*, 34(4):351–366.

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and van den Hengel, A. (2018). Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Andrychowicz, M., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J. W., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. (2018). Learning dexterous in-hand manipulation. *CoRR*, abs/1808.00177.

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Lawrence, Z., and Parikh, D. (2015). Vqa: Visual question answering. In *Proc. of ICCV*.

Artzi, Y. and Zettlemoyer, L. (2013). Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association of Computational Linguistics*, 1:49–62.

Bahdanau, D., Brakel, P., Kelvin, K., Goyal, A., Lowe, R., Pineau, J., Courville, A., and Bengio, Y. (2017). An actor-critic algorithm for sequence prediction. *Proc. of ICLR*.

Baroni, M. (2016). Grounding distributional semantics in the visual world. *Language and Linguistics Compass*, 10(1):3–13.

Barsalou, L. (2008a). Grounded cognition. *Annu. Rev. Psychol.*, 59:617–645.

Barsalou, L. W. (2008b). Grounded cognition. *Annual Review of Psychology*, 59(1):617–645.

Bellman, R. (1966). Dynamic programming. *Science*, 153(3731):34–37.

Ben-Younes, H., Cadène, R., Thome, N., and Cord, M. (2017). MUTAN: Multimodal Tucker Fusion for Visual Question Answering. *arXiv preprint arXiv:1705.06676*.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *JMLR*, 3(Feb):1137–1155.

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.

Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python*. O'Reilly Media, Inc.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2016). Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Boutonnet, B. and Lupyan, G. (2015). Words jump-start vision: A label advantage in object recognition. *Journal of Neuroscience*, 35(25):9329–9335.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Brahmbhatt, S. and Hays, J. (2017). Deepnav: Learning to navigate large cities. *CoRR*, abs/1701.09135.

Brock, A., Donahue, J., and Simonyan, K. (2019). Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*.

Buhrmester, M., Kwang, T., and Gosling, S. (2011). Amazon's Mechanical Turk a new source of inexpensive, yet high-quality, data? *Perspectives on psychological science*, 6(1):3–5.

Castrejon, L., Aytar, Y., Vondrick, C., Pirsiavash, H., and Torralba, A. (2016). Learning Aligned Cross-Modal Representations from Weakly Aligned Data. In *Proc. CVPR*.

Chaplot, D. S., Parisotto, E., and Salakhutdinov, R. (2018a). Active neural localization. *arXiv preprint arXiv:1801.08214*.

Chaplot, D. S., Sathyendra, K. M., Pasumarthi, R. K., Rajagopal, D., and Salakhutdinov, R. (2018b). Gated-attention architectures for task-oriented language grounding. *AAAI*.

Chen, D. L. and Mooney, R. J. (2011). Learning to interpret natural language navigation instructions fro mobservations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-2011)*, San Francisco, CA, USA.

Chen, H., Suhr, A., Misra, D., Snavely, N., and Artzi, Y. (2019). Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12538–12547.

Cho, K., Merriënboer, B. V., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. of EMNLP*. Association for Computational Linguistics.

Choromanska, A., Henaff, M., Mathieu, M., Ben Arous, G., and LeCun, Y. (2015). The loss surfaces of multilayer networks. *Journal of Machine Learning Research*, 38:192–204.

Cirik, V., Zhang, Y., and Baldridge, J. (2018). Following formulaic map instructions in a street simulation environment. In *2018 NeurIPS Workshop on Visually Grounded Interaction and Language*.

Clark, H. and Schaefer, E. (1989). Contributing to discourse. *Cognitive Science*, 13(2):259–294.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 886–893, Washington, DC, USA. IEEE Computer Society.

Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., and Batra, D. (2018). Embodied Question Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Das, A., Kottur, S., Gupta, K., Singh, A., Yadav, D., Moura, J., Parikh, D., and Batra, D. (2017a). Visual Dialog. In *Proc. of CVPR*.

Das, A., Kottur, S., Moura, J., Lee, S., and Batra, D. (2017b). Learning Cooperative Visual Dialog Agents with Deep Reinforcement Learning. *arXiv preprint arXiv:1703.06585*.

Das, A., Kottur, S., Moura, J. M., Lee, S., and Batra, D. (2017c). Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Dauphin, Y. N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2933–2941. Curran Associates, Inc.

Dauphin, Y. N., Vries, H. d., and Bengio, Y. (2015). Equilibrated adaptive learning rates for non-convex optimization. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1504–1512, Cambridge, MA, USA. MIT Press.

de Vries, H., Strub, F., Chandar, S., Pietquin, O., Larochelle, H., and Courville, A. (2016). GuessWhat?! Visual object discovery through multi-modal dialogue. In *Proc. of CVPR*.

de Vries, H., Strub, F., Mary, J., Larochelle, H., Pietquin, O., and Courville, A. C. (2017). Modulating early visual processing by language. In *Proc. of NIPS*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Dodge, J., Gane, A., Zhang, X., Bordes, A., Chopra, S., Miller, A., Szlam, A., and Weston, J. (2016). Evaluating prerequisite qualities for learning end-to-end dialog systems. In *Proc. of ICLR*.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In Xing, E. P. and Jebara, T., editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 647–655, Bejing, China. PMLR.

Dumoulin, V., Shlens, J., and Kudlur, M. (2017). A Learned Representation For Artistic Style. In *Proc. of ICLR*.

Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning. *ArXiv e-prints*.

Eckert, W., Levin, E., and Pieraccini, R. (1997). User modeling for spoken dialogue system evaluation. In *Proc. of ASRU*.

Elliott, D., Frank, S., Sima'an, K., and Specia, L. (2016). Multi30k: Multilingual english-german image descriptions. *arXiv preprint arXiv:1605.00459*.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Escalante, H., Hernández, C., Gonzalez, J., López-López, A., Montes, M., Morales, E., Sucar, E., Villaseñ, L., and Grubinger, M. (2010). The segmented and annotated IAPR TC-12 benchmark. *CVIU*.

Evtimova, K., Drozdov, A., Kiela, D., and Cho, K. (2017). Emergent language in a multi-modal, multi-step referential game. *arXiv preprint arXiv:1705.10369*.

F. Ferreira and M. Tanenhaus (2007). Introduction to the special issue on language–vision interactions. *Journal of Memory and Language*, 57(4):455–459.

Fei-Fei, L., Fergus, R., and Perona, P. (2006). One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*.

Fukui, A., Park, D. H., Yang, D., Rohrbach, A., Darrell, T., and Rohrbach, M. (2016). Multimodal Compact Bilinear Pooling for Visual Question Answering and Visual Grounding. In *Proc. of EMNLP*.

Garrod, S. and Anderson, A. (1987). Saying what you mean in dialogue: A study in conceptual and semantic co-ordination. *Cognition*, 27(2):181 – 218.

Geman, D., Geman, S., Hallonquist, N., and Younes, L. (2015). Visual turing test for computer vision systems. *Proceedings of the National Academy of Sciences*, 112(12):3618–3623.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep Learning. Book in preparation for MIT Press.

Gupta, A., Vedaldi, A., and Zisserman, A. (2016). Synthetic data for text localisation in natural images. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Gupta, S., Davidson, J., Levine, S., Sukthankar, R., and Malik, J. (2017a). Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Gupta, S., Fouhey, D., Levine, S., and Malik, J. (2017b). Unifying map and landmark based representations for visual navigation. *arXiv preprint arXiv:1712.08125*.

Hadsell, R., Sermanet, P., Han, J., Flepp, B., Muller, U., and LeCun, Y. (2007). Online learning for offroad robots: Using spatial label propagation to learn long-range traversability. In *Proc. of Robotics: Science and Systems (RSS)*, volume 11.

He, H., Balakrishnan, A., Eric, M., and Liang, P. (2017). Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1766–1776, Vancouver, Canada. Association for Computational Linguistics.

Hermann, K. M., Hill, F., Green, S., Wang, F., Faulkner, R., Soyer, H., Szepesvari, D., Czarnecki, W., Jaderberg, M., Teplyashin, D., et al. (2017). Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*.

Hinton, G. E., McClelland, J. L., Rumelhart, D. E., et al. (1984). *Distributed representations*. Carnegie-Mellon University Pittsburgh, PA.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

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

Hu, R., Xu, H., Rohrbach, M., Feng, J., Saenko, K., and Darrell, T. (2016). Natural Language Object Retrieval. *Proc. of CVPR*.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proc. of ICML*.

Jabri, A., Joulin, A., and van der Maaten, L. (2016). Revisiting visual question answering baselines. In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 727–739, Cham. Springer International Publishing.

Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Zitnick, C. L., and Girshick, R. (2017). Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proc. of CVPR*.

Kaiming, K., Xiangyu, Z., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proc. of CVPR*.

Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proc. CVPR*.

Kazemzadeh, S., Ordonez, V., Matten, M., and Berg, T. (2014). ReferItGame: Referring to Objects in Photographs of Natural Scenes. In *Proc. of EMNLP*.

Kiela, D. (2017). Deep embodiment: grounding semantics in perceptual modalities (PhD thesis). Technical Report UCAM-CL-TR-899, University of Cambridge, Computer Laboratory.

Kiela, D., Bulat, L., A.Vero, and Clark, S. (2016). Virtual Embodiment: A Scalable Long-Term Strategy for Artificial Intelligence Research. *NIPS workshop in Machine Intelligence*.

Kiela, D., Conneau, A., Jabri, A., and Nickel, M. (2017). Learning visually grounded sentence representations. *arXiv preprint arXiv:1707.06320*.

Kim, J.-H., Lee, S.-W., Kwak, D., Heo, M.-O., Kim, J., Ha, J.-W., and Zhang, B.-Y. (2016). Multimodal Residual Learning for Visual QA. In *Proc. of NIPS*.

Kim, J.-H., On, K. W., Lim, W., Kim, J., Ha, J.-W., and Zhang, B.-T. (2017). Hadamard Product for Low-rank Bilinear Pooling. In *Proc. of ICLR*.

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Kok, P., Failing, M., and de Lange, F. (2014). Prior expectations evoke stimulus templates in the primary visual cortex. *Journal of Cognitive Neuroscience*, 26(7):1546–1554.

Kottur, S., Moura, J. M., Lee, S., and Batra, D. (2017). Natural Language Does Not Emerge 'Naturally' in Multi-Agent Dialog. volume abs/1706.08502.

Krahmer, E. and Deemter, K. V. (2012). Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA. Curran Associates Inc.

Lazaridou, A., Peysakhovich, A., and Baroni, M. (2016). Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324.

Lemon, O. and Pietquin, O. (2007). Machine learning for spoken dialogue systems. In *Proc. of Interspeech*.

Levin, E. and Pieraccini, R. (1997). A stochastic model of computer-human interaction for learning dialogue strategies. In *Eurospeech*, volume 97, pages 1883–1886.

Levin, E., Pieraccini, R., and Eckert, W. (1997). Learning dialogue strategies within the markov decision process framework. In *Proc. of ASRU*.

Lewis, M., Yarats, D., Dauphin, Y. N., Parikh, D., and Batra, D. (2017). Deal or no deal? end-to-end learning for negotiation dialogues. *arXiv preprint arXiv:1706.05125*.

Lin, T., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, L. (2014). Microsoft coco: Common objects in context. In *Proc of ECCV*.

Liu, C., Lowe, R., Serban, I., Noseworthy, M., Charlin, L., and Pineau, J. (2016a). How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*.

Liu, S., Zhu, Z., Ye, N., Guadarrama, S., and Murphy, K. (2016b). Optimization of image description metrics using policy gradient methods. *Under review at CVPR*.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110.

Lowe, R., Pow, N., Serban, I., and Pineau, J. (2015). The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proc. of SIGdial*.

Lu, J., Yang, J., Batra, D., and Parikh, D. (2016). Hierarchical Question-Image Co-Attention for Visual Question Answering. *arXiv preprint arXiv:1606.00061*.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.

Maaten, L. and Hinton, G. (2008). Visualizing data using t-SNE. *JMLR*, 9(Nov):2579–2605.

MacMahon, M., Stankiewicz, B., and Kuipers, B. (2006). Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-2006)*, Boston, MA, USA.

Malinowski, M. and Fritz, M. (2014). A multi-world approach to question answering about real-world scenes based on uncertain input. In *Proc. of NIPS*.

Malinowski, M., Rohrbach, M., and Fritz, M. (2015). Ask your neurons: A neural-based approach to answering questions about images. In *Proc. of ICCV*.

Mao, J., Huang, J., Toshev, A., Camburu, O., Yuille, A., and Murphy, K. (2015). Generation and comprehension of unambiguous object descriptions. *arXiv preprint arXiv:1511.02283*.

Massiceti, D., Dokania, P. K., Siddharth, N., and Torr, P. H. S. (2018). Visual dialogue without vision or dialogue. *CoRR*, abs/1812.06417.

Mei, H., Bansal, M., and Walter, M. R. (2016). Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Proceedings of AAAI*.

Miller, A. H., Feng, W., Fisch, A., Lu, J., Batra, D., Bordes, A., Parikh, D., and Weston, J. (2017). Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476*.

Mirowski, P., Banki-Horvath, A., Anderson, K., Teplyashin, D., Hermann, K. M., Malinowski, M., Grimes, M. K., Simonyan, K., Kavukcuoglu, K., Zisserman, A., et al. (2019). The streetlearn environment and dataset. *arXiv preprint arXiv:1903.01292*.

Mirowski, P., Grimes, M. K., Malinowski, M., Hermann, K. M., Anderson, K., Teplyashin, D., Simonyan, K., Kavukcuoglu, K., Zisserman, A., and Hadsell, R. (2018). Learning to navigate in cities without a map. *CoRR*, abs/1804.00168.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. (2014). On the number of linear regions of deep neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2924–2932. Curran Associates, Inc.

Mooney, R. (2006). Learning language from perceptual context: A challenge problem for AI. In *Proc. of the 2006 AAAI Fellows Symposium*.

Mordatch, I. and Abbeel, P. (2017). Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*.

Ng, A., Harada, D., and Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Proc. of ICML*.

O. Lemon and O. Pietquin (2012). *Data-Driven Methods for Adaptive Spoken Dialogue Systems*. Springer.

Oreshkin, B., Rodríguez López, P., and Lacoste, A. (2018). Tadam: Task dependent adaptive metric for improved few-shot learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 721–731. Curran Associates, Inc.

Parde, N., Hair, A., Papakostas, M., Tsiakas, K., Dagioglou, M., Karkaletsis, V., and Nielsen, R. D. (2015). Grounding the meaning of words through vision and interactive gameplay. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1895–1901. AAAI Press.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS-W*.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global Vectors for Word Representation. In *Proc. of EMNLP*.

Perez, E., Strub, F., De Vries, H., Dumoulin, V., and Courville, A. (2018). Film: Visual reasoning with a general conditioning layer. In *Proc. of AAAI*.

Pietquin, O. and Dutoit, T. (2006). A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech, and Language Processing*.

Pietquin, O., Geist, M., Chandramohan, S., and Frezza-Buet, H. (2011). Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Transactions on Speech and Language Processing (TSLP)*, 7(3):7.

Pietquin, O. and Hastie, H. (2013a). A survey on metrics for the evaluation of user simulations. *The knowledge engineering review*, 28(01):59–73.

Pietquin, O. and Hastie, H. (2013b). A survey on metrics for the evaluation of user simulations. *The Knowledge Engineering Review*, 28(1):59–73.

Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2016). Sequence level training with recurrent neural networks. *Proc. of ICLR*.

Ren, M., Kiros, R., and Zemel, R. (2015). Exploring models and data for image question answering. In *Proc. of NIPS*.

Riezler, S., Simianer, P., and Haas, C. (2014). Response-based learning for grounded machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 881–891.

Roy, D. (2002). Learning visually grounded words and syntax for a scene description task. *Computer speech & language*, 16(3):353–385.

Roy, D. (2005). Grounding words in perception and action: computational insights. *Trends in cognitive sciences*, 9(8):389–396.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252.

Schatzmann, J., Weilhammer, K., Stuttle, M., and Young, S. (2006). A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The knowledge engineering review*, 21(2):97–126.

Serban, I., Lowe, R., Charlin, L., and Pineau, J. (2015a). A survey of available corpora for building data-driven dialogue systems. *arXiv preprint arXiv:1512.05742*.

Serban, I., Lowe, R., Charlin, L., and Pineau, J. (2016). Generative Deep Neural Networks for Dialogue: A Short Review. *NIPS workshop Learning Methods for Dialogue*.

Serban, I., Sordoni, A., Bengio, Y., Courville, A., and Pineau, J. (2015b). Hierarchical neural network generative models for movie dialogues. *arXiv preprint arXiv:1507.04808*.

Shekhar, R., Pezzelle, S., Klimovich, Y., Herbelot, A., Nabi, M., Sangineto, E., and Bernardi, R. (2017). "foil it! find one mismatch between image and language caption". In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL) (Volume 1: Long Papers)*, pages 255–265.

Shekhar, R., Venkatesh, A., Baumgärtner, T., Bruni, E., Plank, B., Bernardi, R., and Fernández, R. (2019). Beyond task success: A closer look at jointly learning to see, ask, and GuessWhat. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2578–2587, Minneapolis, Minnesota. Association for Computational Linguistics.

Shih, K., Singh, S., and Hoiem, D. (2016). Where to look: Focus regions for visual question answering. In *Proc. of CVPR*.

Shortreed, S. M., Laber, E., Lizotte, D. J., Stroup, T. S., Pineau, J., and Murphy, S. A. (2011). Informing sequential clinical decision-making throughÂ reinforcement learning: an empirical study. *Machine Learning*, 84(1):109–136.

Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G. V. D., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T. P., Simonyan, K., and Hassabis, D. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *Proc. of ICLR*.

Singh, S., Kearns, M., Litman, D., and Walker, M. (1999a). Reinforcement Learning for Spoken Dialogue Systems. In *Proc. of NIPS*.

Singh, S., Kearns, M., Litman, D., and Walker, M. (1999b). Reinforcement Learning for Spoken Dialogue Systems. In *Proc. of NIPS*.

Smith, L. and Gasser, M. (2005). The development of embodied cognition: Six lessons from babies. *Artificial Life*, 11(1-2):13–29.

Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J., Gao, J., and Dolan, B. (2015). A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*.

Steels, L. and Hild, M. (2012). *Language grounding in robots*. Springer Science & Business Media.

Strub, F., De Vries, H., Mary, J., Piot, B., Courvile, A., and Pietquin, O. (2017). End-to-end optimization of goal-driven and visually grounded dialogue systems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 2765–2771. AAAI Press.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In Dasgupta, S. and McAllester, D., editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1139–1147, Atlanta, Georgia, USA. PMLR.

Sutskever, I., Vinyals, O., and Le, Q. (2014). Sequence to sequence learning with neural networks. In *Proc of NIPS*.

Sutton, R., McAllester, D., Singh, S., Mansour, Y., et al. (1999). Policy gradient methods for reinforcement learning with function approximation. In *Proc. of NIPS*, volume 99, pages 1057–1063.

Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*.

Tejas, G. Y. K., Douglas, S., B, D., and Devi, P. (2017). Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Proc. of CVPR*.

Thierry, G., Athanasopoulos, P., Wiggett, A., Dering, B., and Kuipers, J. (2009). Unconscious effects of language-specific terminology on preattentive color perception. *PNAS*, 106(11):4567–4570.

Thomason, J., Sinapov, J., Svetlik, M., Stone, P., and Mooney, R. (2016). Learning Multi-Modal Grounded Linguistic Semantics by Playing I Spy. In *Proc. of IJCAI*.

Unknown (2016). Akinator. en.akinator.com/. Accessed: 2016-09.

Vinyals, O. and Le, Q. (2015). A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proc. of CVPR*.

Vo, N., Jacobs, N., and Hays, J. (2017). Revisiting im2gps in the deep learning era. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2640–2649. IEEE.

Vogel, A., Raghunathan, K., and D. (2010). Eye Spy: Improving Vision through Dialog. In *AAAI Fall Symposium Series*.

Walker, M., Litman, D., Kamm, C., and Abella, A. (1997). Paradise: A framework for evaluating spoken dialogue agents. In *Proc. of ACL*.

Weizenbaum, J. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45.

Wen, T., Gasic, M., Mrksic, N., Rojas-Barahona, L., Su, P., Ultes, S., Vandyke, D., and Young, S. (2016). A Network-based End-to-End Trainable Task-oriented Dialogue System. *arXiv preprint arXiv:1604.04562*.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Weston, J., Bordes, A., Chopra, S., Rush, A., van Merriënboer, B., Joulin, A., and Mikolov, T. (2016). Towards ai-complete question answering: A set of prerequisite toy tasks. In *Proc. of ICLR*.

Williams, J. and Young, S. (2007). Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

Williams, R. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Xu, H. and Saenko, K. (2015). Ask, attend and answer: Exploring question-guided spatial attention for visual question answering. In *Proc. of ECCV*.

Xu, K., Ba, J. L., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 2048–2057. JMLR.org.

Yang, Z., He, X., Gao, J., Deng, L., and Smola, A. (2016a). Stacked attention networks for image question answering. In *Proc. of CVPR*.

Yang, Z., He, X., Gao, J., and Smola, L. D. A. (2016b). Stacked attention networks for image question answering. In *Proc. of CVPR*.

Young, S., Gašić, M., Thomson, B., and Williams, J. (2013). POMDP-based statistical spoken dialog systems: A review. *Proc. of the IEEE*, 101(5):1160–1179.

Yu, H., Zhang, H., and Xu, W. (2017). A deep compositional framework for human-like language acquisition in virtual environment. *arXiv preprint arXiv:1703.09831*.

Yu, L., Poirson, P., Yang, S., Berg, A., and Berg, T. (2016). Modeling context in referring expressions. In *Proc. in ECCV*. Springer.

Zhang, J., Wu, Q., Shen, C., Zhang, J., Lu, J., and van den Hengel, A. (2018). Goal-oriented visual question generation via intermediate rewards. In *The European Conference on Computer Vision (ECCV)*.

Zhao, R. and Tresp, V. (2018). Learning goal-oriented visual dialog via tempered policy gradient. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 868–875.

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. (2014). Learning deep features for scene recognition using places database. In *Proc of NIPS*.