Training Models to Ignore Dataset Bias

Christopher Clark

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2020

Reading Committee:

Luke Zettlemoyer, Chair

Hannaneh Hajishirz

Mark Yatskar

Program Authorized to Offer Degree:
Computer Science & Engineering

University of Washington

Abstract

Training Models to Ignore Dataset Bias

Christopher Clark

Chair of the Supervisory Committee:
Professor Luke Zettlemoyer
Department of Computer Science & Engineering

Modern machine learning algorithms have been able to achieve impressive results on complex tasks such as language comprehension or image understanding. However, recent work has cautioned that this success is often partially due to exploiting incidental correlations that were introduced during dataset creation, and are not fundamental to the target task. For example, sentence entailment datasets can have spurious word-class correlations if nearly all contradiction sentences contain the word "not", and image recognition datasets can have tell-tale object-background correlations if dogs are always indoors. Models that exploit these incidental correlation, which we call dataset bias, can be brittle and perform poorly on out-of-domain examples. In this thesis, we present several methods of solving this issue by preventing models from using dataset bias.

A key challenge for this task is determining which predictive patterns in the training data are bias. This thesis proposes several solutions, ranging from methods that exploit domain expertise when such knowledge is available, to more broadly applicable domain-general solutions. Solving this task also requires preventing complex neural models from exploiting these biased patterns, even though they are often easy to learn and effective on the training data. We present ensembling and data augmentation based methods to handle this difficulty. In all cases, we evaluate our models by showing improved performance on out-of-domain datasets that were built to penalize biased models.

Our first focus is on question answering, motivated by the observation biases can lead to poor performance when applying a model to multiple paragraphs. To solve this task, we propose a modified training scheme that exposes the model to additional paragraphs that do not answer the question. We then consider the case where expert knowledge of the bias can be used to construct a *bias-only* model that captures biased methods. In this case, we can build an unbiased model by ensembling it with the bias-only model in order to disincentive it from learning bias. Finally, we generalize this approach by proposing a method to automatically construct the bias-only model when no such expert knowledge is available. Overall, this thesis shows that it is possible to train unbiased models on biased datasets, and proposes some fundamental answers to question about how bias can be detected and avoided.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I am deeply grateful to my advisor, Luke Zettlemoyer, for his mentorship during my PhD. He gave me many insights into natural language processing and doing research in general, as well as opportunities to explore and develop my own research ideas. Luke's positivity and patience contributed greatly to making this journey more enjoyable and less stressful.

I have had the chance to work with many other amazing people while completing this work. Matt Gardner guided me through completing my first successful natural language processing research project, which became a chapter of this dissertation. Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Lajanugen Logeswaran, and other researchers I met when interning at Google, were great collaborators and exposed me to many new ideas. I am very grateful to Mark Yatskar for providing important insights and enthusiasm when completing our work on debasing. I also owe a lot of ideas and inspiration to discussions with students across the UW-NLP community, who have often impressed me with their thoughtfulness and independence of thought.

I would also like to thank Amos Storkey for taking a chance on me and helping me complete my first research project, which inspired me start my PhD. Santosh Divvala, Tony Fader, and Oren Etzioni also played important roles to get me started as a researcher at the Allen Institute of Artificial Intelligence.

Finally, I greatly appreciate the unwavering support and encouragement from my dad, Peter, who has inspired me since a young age be excited about AI. My siblings, Kevin and Rachel, and my step-mom, Rosann, have also been enormously kind and supportive, something I especially appreciated when I was dealing with the challenges of graduating during the COVID-19 pandemic.

DEDICATION

To my parents

Chapter 1

INTRODUCTION

Modern machine learning algorithms have shown impressive results in recent years, even surpassing human performance on some popular datasets for image and language understanding [Rajpurkar et al., 2016, Deng et al., 2009b]. However, closer examination has shown that these models have a persistent weakness: the tendency to rely on patterns caused by idiosyncrasies in the data collection process, rather than ones fundamental to the target task. Examples include textual entailment models learning the word "not" always implies contradiction [Gururangan et al., 2018], visual question answering (VQA) models learning "2" is almost always the answer to "How many" questions [Jabri et al., 2016], and question answering models selecting entities that occur near question words irrespective of context [Jia and Liang, 2017] (see Section 1.1 for a more detailed overview).

We call these kinds of patterns dataset bias. Models that rely on dataset bias can perform well on in-domain data, but will break when tested on out-of-domain data where the bias no longer applies. This thesis will discuss ways to solve this problem by preventing models from exploiting bias.

Doing this requires a method to distinguish bias from other, better-generalizing features of the data. This is challenging because dataset biases are effective on the training data, and are simple and broadly-applicable enough that standard regularization methods, which primarily aim to prevent models from memorizing examples, are not effective. We discuss several possible approaches, starting from settings where we can exploit prior knowledge of the bias, before moving on to more domain-general methods.

Once biases are identified, it is not obvious how to prevent state-of-the-art models, which are typically black-box neural networks, to avoid learning them. In the question answering

setting, we find a data-augmentation method that exposes the model to additional passages that do not contain the answer. However, we also propose a more general ensemble-based method that explicitly models bias and non-bias components of the training data.

We provide a detailed overview of our methods in Section 1.2. In all cases, we evaluate our work by training the model on biased training data, and then evaluating on out-of-domain test sets where using the bias will harm performance. We compare to various baselines, as well as training the model as normal. We show improved performance across a wide range of tasks and domains, including several question answering tasks, two textual entailment datasets, and a visual question answering and image recognition dataset. Altogether, this thesis shows that although bias is pervasive in modern datasets, it is still possible to use those datasets to build unbiased models.

1.1 Examples of Dataset Bias

In order to better motivate this task, we start by discussing recent observations of dataset bias in more detail. Figure 1.1 provides a visualization of some of these examples.

In natural language understanding, an important example of dataset bias was the observation that language regularities in the popular sentence-pair classification datasets MNLI [Williams et al., 2018] and SNLI [Bowman et al., 2015b] allows models to predict the correct class when given only one of the two input sentences using keywords [Gururangan et al., 2018] or by naively classifying sentences with many shared words as 'entailment' [McCoy et al., 2019b]. Another important example comes from the observation that models trained on the extensively studied SQuAD QA dataset [Rajpurkar et al., 2016] tend to select spans of text that are near question words and match the expected answer type of the question, while ignoring the larger context [Jia and Liang, 2017]. Question answering models are also known to learn positional biases (e.g., the first sentence is much more likely to contain the answer) [Ko et al., 2020]. Additional examples in natural language processing include performing story completion by identifying the author's style [Schwartz et al., 2017] and using superficial associations to answer multiple-choice questions [Clark et al., 2016, 2018].

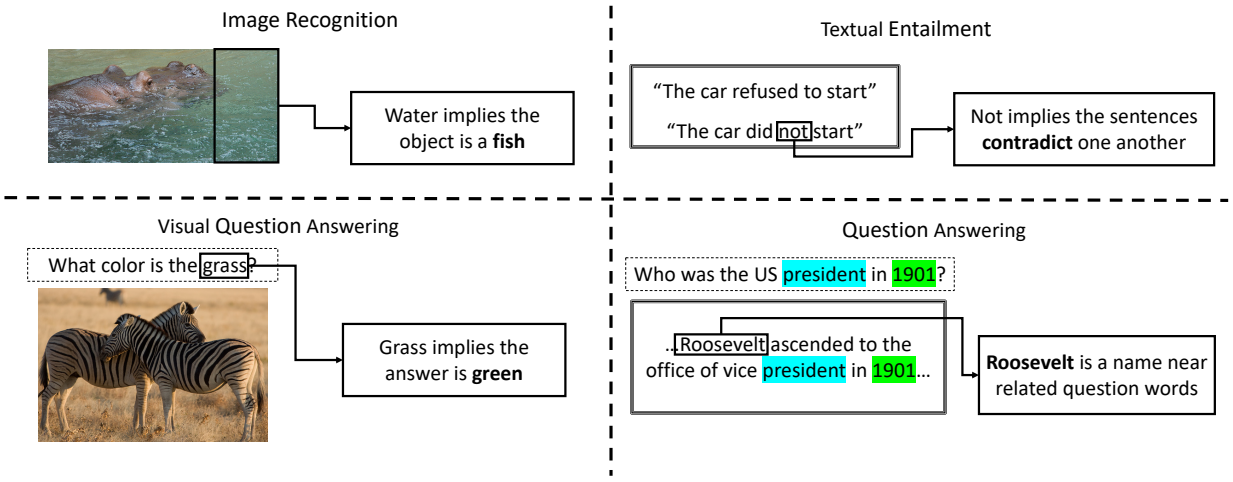

Figure 1.1: Examples of naive heuristics models have been observed to adopt due to fitting to dataset bias. While these heuristics happen to perform well on in-domain data, they are not general solutions to the target tasks, and as a result lead to errors on out-of-domain or adversarial examples as shown here. Upper left: background-class correlation in image recognition [Hendrycks et al., 2019]. Upper right: keyword-class correlations in MNLI [Gururangan et al., 2018]. Lower left: question-type/class correlations in visual question answering [Goyal et al., 2018]. Lower right: matching nearby question words for reading comprehension [Jia and Liang, 2017].

In computer vision, it has long been known image datasets can contain regularities that are unlikely to hold true in practical settings (e.g. the object of interest is never occluded) [Ponce et al., 2006, Torralba and Efros, 2011], which creates opportunities for models to find biased strategies. Image recognition models have also been observed to be overly reliant on texture. For example, while people can often identify an animal even if it has an unusual coloring or texture (e.g., given a silhouette or a painting), image recognition models trained on ImageNet [Deng et al., 2009b] are much less effective at doing so [Geirhos et al., 2019]. Another concern in image recognition is that models learn to identify images based on cues in the background [Arjovsky et al., 2019]. For example, Hendrycks et al. [2019] observe that many of the errors of high-performance models on ImageNet can be attributed to erroneous background cues, and Zhao et al. [2017] observe models will make use of knowledge of stereotypical behavior (e.g., woman are more likely to be in 'cooking' related images than man) when performing image classification.

Models trained on image and language understanding have been observed in many cases to be overly-reliant on the language aspect, often failing to properly account for the image. For example, in visual question answering (VQA) [Goyal et al., 2018] it has been observed models often select an answer because it is a common output for the given question (e.g., 'green' is usually the answer to the question 'What color is the grass?') [Agrawal et al., 2018]. Several recent tasks that require the model to make decisions using visual and language input have been observed to be partially solvable using the language input alone [Thomason et al., 2019], including visual-and-language navigation [Anderson et al., 2018b], VQA in interactive environments [Gordon et al., 2018], and embodied question answering [Das et al., 2018].

Overall, bias has been observed in many datasets across a multitude of domains. This tempers our understanding of how effective modern machine learning models are despite their high in-domain performance. However, our methods provide a way to continue using these datasets while preventing models from adopting these naive solutions.

1.2 Overview of our Approach

1.2.1 Data Augmentation for Question Answering

First, we consider multi-paragraph question answering, where we need to answer a question using a large amount of context text. Modern neural models, such as those based on transformers [Vaswani et al., 2017] or recurrent methods, can scale poorly with the amount of input text, making it impossible to use the model to process the entire context at once. In this case, we can heuristically select a small number of candidate paragraphs that are likely to be related to the question, run the model on each paragraph individually, and return the model's overall most confident prediction.

This approach, however, creates a train/test mismatch: at test time the model is being used on many paragraphs, but it is only being trained on one paragraph at a time due computation restriction on the backpropagation step. We find that this one-paragraph-at-a-time restriction leads to the model learning biases that harm performance in the multi-paragraph setting. For example, models might be quick to learn that if there is one number in the context paragraph for a "How many..." question it must be the answer, even though this heuristic will fair poorly if the model is exposed to a large number of candidate paragraphs. We resolve this issue by proposing a modified training scheme, where the loss is computed across multiple paragraphs, some of which may not contain an answer. While not an exact replica of the test settings, this is sufficient to prevent the models from becoming biased and results in better multi-paragraph performance.

1.2.2 Using Domain Knowledge of the Bias

Next, we look at a case where an expert has prior knowledge of the bias, either from domain expertise or from studying the data beforehand. Here, we propose to use that knowledge to construct a tightly constrained model that can only use bias (e.g., a model that only has access to the question for visual question answering). This model is trained on the training data, resulting in a *bias-only* model that makes predictions only using bias. Then, a *robust*

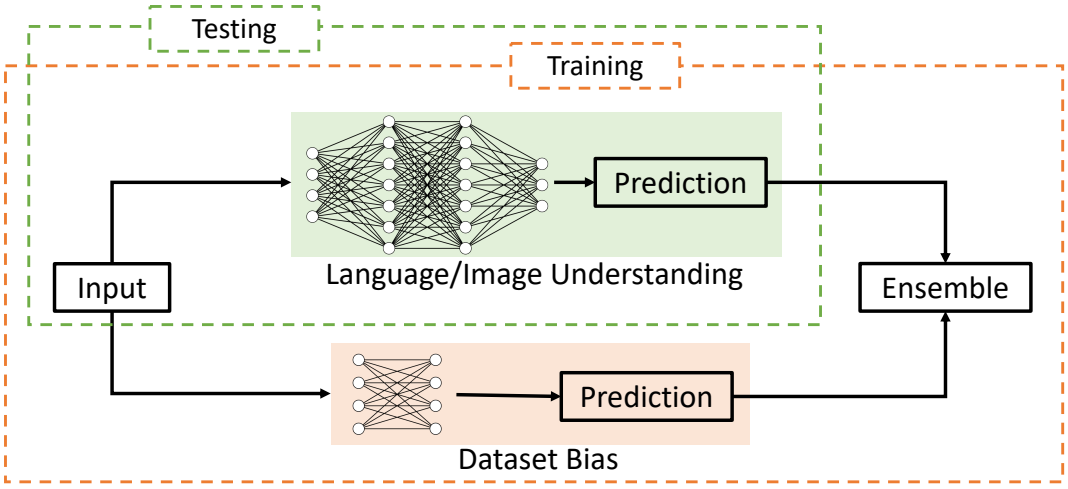

Figure 1.2: An overview of our ensemble-based methods. We train one model that makes predictions using only the bias, and one model the makes predictions using other means. During training, the two models are ensembled, but during testing the unbiased model is used alone.

model is trained in an ensemble with that pre-trained model. The ensemble will use the bias-only model to capture biased patterns, so the robust model has no incentive to learn them. The robust model can then be used alone at test time. Figure 1.2 shows a diagram of this approach.

1.2.3 Handling Unknown Biases

Finally, we look at the setting where we do not have such prior knowledge of the bias. Here, we leverage the hypothesis that biased patterns are likely to be simple, i.e., can be modelled with shallower and less parameter-heavy models then non-biased patterns. While there is no guarantee all examples of dataset bias will fit this criteria, it is commonly true in the examples we have discussed. Additionally, we expect complex dataset biases to be less problematic because models will be less prone to fit to them.

To make use of this hypothesis, we train an ensemble of two models while encouraging relatively simple patterns to be factored into one model and relatively complex patterns into

the other. In particular, we (1) ensemble a lower capacity model, i.e., a model with fewer parameters, and a higher capacity model, which creates a natural tendency for the higher capacity model to capture more complex patterns, (2) put a small weight on the loss of the lower capacity model so that it is preferentially used to capture simpler patterns that it can model, and (3) enforce conditional independence between the models so that they learn non-overlapping strategies. The higher capacity model can then be used alone at test time as an unbiased model.

1.3 Thesis Outline

In Chapter 2, we will provide an overview of the background and related work on dataset bias and debiasing methods. Chapter 3 will present our multi-paragraph question answering method. Chapter 4 will present a method using the pre-trained bias-only model, and Chapter 5 will present an extension of that method to the case where not such bias-only model is available. Finally, we conclude and discuss possible future directions in Chapter 6.

Chapter 2
BACKGROUND AND RELATED WORK

In this section we review the prior work on constructing unbiased datasets, dataset debiasing, and connections between our work and other fields of study.

2.1 Building Unbiased Datasets

A natural suggestion to resolve this issue is to build datasets without bias, and recent datasets have indeed be designed to avoid certain kinds of biases. For example, both CoQA [Reddy et al., 2019] and QuAC [Choi et al., 2018] take steps to prevent annotators from producing questions that closely match the sentence containing the answer, which helps prevent the dataset being solved by naively selecting words near questions words as answers [Jia and Liang, 2017]. Ensuring questions come from a more "natural" distribution, i.e., are constructed by individuals with genuine informational needs rather then because they were prompted to ask a question that is answered by a particular passage, has been suggested to have a similar affect [Clark et al., 2019a, Kwiatkowski et al., 2019]. VQA 2.0 [Goyal et al., 2018] construct pairs questions with multiple images that then lead to different answers in order limit the effectiveness of question-only models.

Another common approach is to filter examples solvable by simple baselines [Yang et al., 2018, Zhang et al., 2018b, Clark et al., 2018, Zellers et al., 2018], which is likely to remove examples that have certain kinds of biases. Some recent work has also proposed augmenting datasets with counter-factuals [Gardner et al., 2020, Kaushik et al., 2019], meaning examples that were minimally altered in ways that change the example's label, in order to force the model to pay attention to the parts that were changed.

This efforts are important, but removing dataset bias is not always easy. For example,

some datasets designed to require multi-hop reasoning can still be partially solved by single-hop methods [Chen and Durrett, 2019, Min et al., 2019], and the adversarial filtering in SWAG [Zellers et al., 2018] was found to be insufficient to prevent the task from containing dataset biases [Zellers et al., 2019].

Additionally, these methods can make datasets less natural, meaning they reflect distributions that are unlikely to occur in the real world, since the dataset distribution has been additionally manipulated. Most importantly, rebuilding datasets every time a bias that we want to avoid is identified is costly and wasteful. The method we present in this document can train unbiased models on biased datasets, which means we can continue to make use of existing datasets and can re-train models if our understanding of what biases we want to avoid evolve.

2.2 Debiasing Methods

2.2.1 Partial Input Methods

Partial-input methods focus on the case where the bias involves the ability to predict the label using only part of the input (for example, predicting the label using only the question in VQA). More generally, the assumption in these cases is that a particular intermediate representation in a network could only contain features that are shallowly indicative of the class due to bias. In these cases, adversarial networks can be used to remove those class-indicative features from the network [Grand and Belinkov, 2019, Wang et al., 2019a, Cadene et al., 2019].

These methods have had some success, but have drawbacks. Adversarial networks my not be completely effective at removing class-indicative features [Elazar and Goldberg, 2018]. Additionally, this can be too blunt of an approach, for example, if the word 'many' in the question correlates with the answer '2' in VQA the only way to remove class-indicative features from the model would be to remove knowledge of the word 'many' from the model's representation, which will likely harm performance overall [Belinkov et al., 2019]. Addition-

ally, there are kinds of bias that are not captured by partial-input methods [Feng et al., 2019].

2.2.2 Using Bias-Only Models

A number of contemporaneous or subsequent works have considered making use of a bias-only model similar to our ensembling methods. Several works have also proposed product-of-experts style ensembling approaches. He et al. [2019] use this method on textual entailment datasets for several biases [Gururangan et al., 2018, McCoy et al., 2019b, Naik et al., 2018]. Karimi Mahabadi et al. [2020] additionally evaluate this approach on the FEVER Symmetric test [Schuster et al., 2019] and show improved performance on several non-adversarial textual entailment datasets. They also propose using the focal loss [Lin et al., 2017] as a reweighting method. Alternatively, REBI [Bahng et al., 2019] employs a conditional independence penalty between a trained model and the bias-only model, HEX [Wang et al., 2019a] tries to ensure the feature space from a model is orthogonal to the features learned by the bias mode.

2.2.3 Identifying Hard Examples

There are also debiasing strategies that identify hard examples in the dataset and re-train the model to focus on those examples [Yaghoobzadeh et al., 2019, Li and Vasconcelos, 2019, Le Bras et al., 2020]. This approach reflects a similar intuition to our work in Chapter 5 that simplicity is connected to dataset bias, although our method is able to explicitly model the bias and does not assume a pool of bias-free examples exist within the training data.

2.2.4 Pretraining

Another way to train to less biased models is to make it easier for the model to learn the unbiased methods of solving the task. Pretraining might help in this regard by training models to use higher-level semantic features of the data. Pretraining might also indirectly

expose the model to a wider distribution of examples, which is likely to help reduce bias.

For example, Lewis and Fan [2018] show generative pre-training can make question-answering systems more robust, and Carlucci et al. [2019] show pre-training on jigsaw-like problems for images has a similar effect for image recognition. Language model pre-training has also been shown to be able to reduce the extent to which models become biased [Tu et al., 2020]. We expect these methods to be complementary to our work; for instance we show we can improve the performance of the extensively pretrained BERT [Devlin et al., 2019] and LXMERT [Tan and Bansal, 2019] models.

2.3 Related Fields

2.3.1 Fairness

Debiasing is related to the task of preventing models from misusing problematic dataset features, such as gender or race. Some works in fairness focus on training models to ignore a feature, which is not applicable to our settings since biases often involve features that are essential to the task (for example ignoring the question-type for VQA would be the question impossible to answer). However training unbiased models can be related to the *Equalized Odds* [Hardt et al., 2016] definition of fairness that requires predictions and a protected feature to be independent conditioned on the label.

However, unlike in fairness, the biases we consider in this work can involve complex predictive strategies (e.g., preferring sentences because they contain many question words in question answering), rather then being based on a particular feature. We also consider cases where the bias is not very precisely defined, or even when it is unknown. Partly because of this, we focus on improving performance on out-of-domain test sets rather then trying to measuring the extent to which models are biased, as is often done in fairness.

Nevertheless, some techniques from the domain of fairness are related to the methods used in debiasing. For example, many works in fairness have used an adversary to remove information from a model's internal representations [Edwards and Storkey, 2016, Beutel et al.,

2017, Wang et al., 2019b], which is also a popular approach in debiasing (see Section 2.2.1). Researchers have also proposed method to ensemble gender-neutral and gender-indicative components of word embeddings [Zhao et al., 2018], which resembles the ensemble based strategies we propose in Chapter 4.

2.3.2 Domain Generalization

A related task is to generalize to unseen test domains when given multiple training datasets from different domains, a task that has been called domain generalization [Muandet et al., 2013]. Most domain generalization methods learn a data representation that is invariant between domains through the use of domain-adversarial classifiers [Ganin et al., 2016, Li et al., 2018b], ensembling domain-specific and domain-invariant representations [Bousmalis et al., 2016, Ding and Fu, 2017] or other means [Arjovsky et al., 2019, Li et al., 2018a, Xu et al., 2014, Ghifary et al., 2015].

Under the assumption that the biases from the individual domains are unlikely to transfer to the other domains, this task can be viewed as partly trying to debias a model. However in that case, simply training the model on the pooled data might be effective since no one dataset bias would be effective on a large portion of the pooled data. It is possible the model could learn to identify which domain each example belongs to and apply a domain-specific bias, in which case bringing in some more explicit de-biasing method might be helpful. It is currently unclear the extent to which these assumptions are correct. Using data from multiple domains remains an intriguing possibility for detecting dataset bias, however given the expense and complexity involved in gathering such data it is unlikely to become a general-purpose solution.

Chapter 3

MULTIPARAGRAPH READING COMPREHENSION

In this chapter we present our method of debiasing question answering models so that they can be used more effectively on a large amount of context text.

3.1 Introduction

Teaching machines to answer arbitrary user-generated questions is a long-term goal of natural language processing. For a wide range of questions, existing information retrieval methods are capable of locating documents that are likely to contain the answer. However modern neural models, such as those based on transformers [Vaswani et al., 2017] or recurrent methods, can scale poorly with the amount of input text, making it challenging to use those models to process these documents in an end-to-end manner.

There are two basic solutions to this problem. Pipelined approaches select a single paragraph from the input documents, which is then passed to a neural model to extract an answer [Joshi et al., 2017, Wang et al., 2018]. Confidence based methods apply the model to multiple paragraphs and returns the answer with the highest confidence [Chen et al., 2017a]. Confidence methods have the advantage of being robust to errors in the (usually less sophisticated) paragraph selection step, however they create a train/test mismatch that can be problematic. In particular, at test time the model is being used across multiple paragraphs, however during training the model is only applied to one paragraph at a time. As we will show, naively trained models often acquire biases that cause them to fail to generalize to this test setting.

In this chapter, we introduce a method for training less biased models, so that they are more effective when used with the confidence method. We also propose an improved

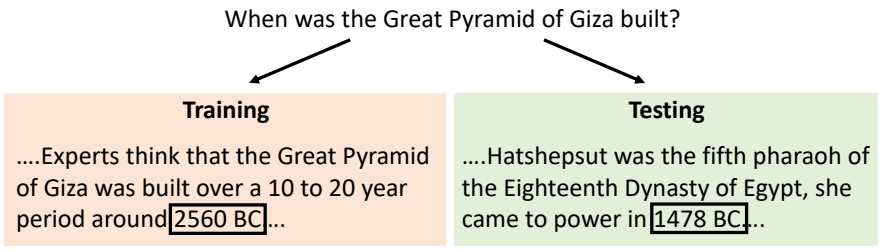

Figure 3.1: Example of a bias in question answering. During training, if the model is only exposed to paragraphs that answer the question (left), it will learn that spans of text that match the answer-type of the question are usually the answer. However, during testing (right), this leads to errors because the model is more likely to be applied to paragraphs that do not answer the question but still contain text that matches the expected answer type.

pipelined method with signficantly improved results.

Our pipelined method focuses on addressing the challenges that come with training on document-level data. We propose a TF-IDF heuristic to select which paragraphs to train and test on. Since annotating entire documents is very expensive, data of this sort is typically distantly supervised, meaning only the answer text, not the locations of the answers in the context, are known. To handle the noise this creates, we use a summed objective function that marginalizes the model's output over all locations the answer text occurs. We apply this approach with a model design that integrates some recent ideas in reading comprehension models, including self-attention [Cheng et al., 2016] and bi-directional attention [Seo et al., 2017a].

Our confidence method extends this approach to better handle the multi-paragraph setting by training a less biased model. Previous approaches trained the model on questions paired with single paragraphs that contain the answer. This leads to biases, such as being overly reliant on detecting answer text that matches the answer-type of the question (see Figure 3.1), or selecting answers near question words without considering context [Jia and Liang, 2017].

To fix these issues, we propose several training regimes that more closely match the test time setting by exposing the model to paragraphs that do not contain an answer. Our most

successful one runs the model independently on a small sample of paragraphs related to the same question, and then normalizes the output probabilities across all paragraphs.

We evaluate our work on TriviaQA web [Joshi et al., 2017], a dataset of questions paired with web documents that contain the answer. We achieve 71.3 F1 on the test set, a 15 point absolute gain over prior work. We additionally perform an ablation study on our pipelined method, and we show the effectiveness of our multi-paragraph methods on TriviaQA unfiltered and a modified version of SQuAD [Rajpurkar et al., 2016] where only the correct document, not the correct paragraph, is known. We release our code[1] to facilitate future work in this field.

3.2 Pipelined Method

We begin by introducing an improved approach to training pipelined question answering systems, where a single paragraph is heuristically extracted from the context document(s) and passed to a paragraph-level QA model. We suggest using a TF-IDF based paragraph selection method and argue that a summed objective function should be used to handle noisy supervision. We also propose a refined model architecture that incorporates some recent modeling ideas for reading comprehension systems.

3.2.1 Paragraph Selection

Our paragraph selection method chooses the paragraph that has the smallest TF-IDF cosine distance with the question. Document frequencies are computed using just the paragraphs within the relevant documents, not the entire corpus. The advantage of this approach is that if a question word is prevalent in the context, for example if the word "tiger" is prevalent in the document(s) for the question "What is the largest living sub-species of the tiger?", greater weight will be given to question words that are less common, such as "largest" or "sub-species". Relative to selecting the first paragraph in the document, this improves the

[1] github.com/allenai/document-qa

chance of the selected paragraph containing the correct answer from 83.1% to 85.1% on TriviaQA web. We also expect this approach to do a better job of selecting paragraphs that relate to the question since it is explicitly selecting paragraphs that contain question words.

3.2.2 Handling Noisy Labels

Question: Which British general was killed at Khartoum in 1885?

Answer: Gordon

Context: In February 1885 Gordon returned to the Sudan to evacuate Egyptian forces. Khartoum came under siege the next month and rebels broke into the city, killing Gordon and the other defenders. The British public reacted to his death by acclaiming 'Gordon of Khartoum', a saint. However, historians have suggested that Gordon defied orders and refused to evacuate...

Figure 3.2: Noisy supervision causes many spans of text that contain the answer, but are not situated in a context that relates to the question, to be labelled as correct answer spans (highlighted in red). This risks distracting the model from learning from more relevant spans (highlighted in green).

In a distantly supervised setup we label all text spans that match the answer text as being correct. This can lead to training the model to select unwanted answer spans. Figure 3.2 contains an example. To handle this difficulty, we use a summed objective function similar to the one from Kadlec et al. [2016], that optimizes the sum of the probabilities of all answer spans. The models we consider here work by independently predicting the start and end token of the answer span, so we take this approach for both predictions. Thus the objective for the span start boundaries becomes:

$$-\log\left(\frac{\sum_{k\in A} e^{s_k}}{\sum_{i=1}^{n} e^{s_i}}\right)$$

where A is the set of tokens that start an answer span, n is the number of context tokens, and s_i is a scalar score computed by the model for span i. This optimizes the negative log-likelihood of selecting any correct start token. This objective is agnostic to how the model

distributes probability mass across the possible answer spans, thus the model can choose to focus on only the more relevant spans.

3.2.3 Model

We use a model with the following layers (shown in Figure 3.3):

Embedding: We embed words using pre-trained word vectors. We also embed the characters in each word into size 20 vectors which are learned, and run a convolution neural network followed by max-pooling to get character-derived embeddings for each word. The character-level and word-level embeddings are then concatenated and passed to the next layer. We do not update the word embeddings during training.

Pre-Process: A shared bi-directional GRU [Cho et al., 2014] is used to map the question and passage embeddings to context-aware embeddings.

Attention: The bi-directional attention mechanism from the Bi-Directional Attention Flow (BiDAF) model [Seo et al., 2017a] is used to build a query-aware context representation. Let $\vec{h_i}$ be the vector for context word i, $\vec{q_j}$ be the vector for question word j, and n_q and n_c be the lengths of the question and context respectively. We compute attention between context word i and question word j as:

$$a_{ij} = \vec{w_1} \cdot \vec{h_i} + \vec{w_2} \cdot \vec{q_j} + \vec{w_3} \cdot (\vec{h_i} \odot \vec{q_j})$$

where $\vec{w_1}$, $\vec{w_2}$, and $\vec{w_3}$ are learned vectors and $\odot$ is element-wise multiplication. We then compute an attended vector $\vec{c_i}$ for each context token as:

$$p_{ij} = \frac{e^{a_{ij}}}{\sum_{j=1}^{n_q} e^{a_{ij}}}$$

$$\vec{c_i} = \sum_{j=1}^{n_q} \vec{q_j} p_{ij}$$

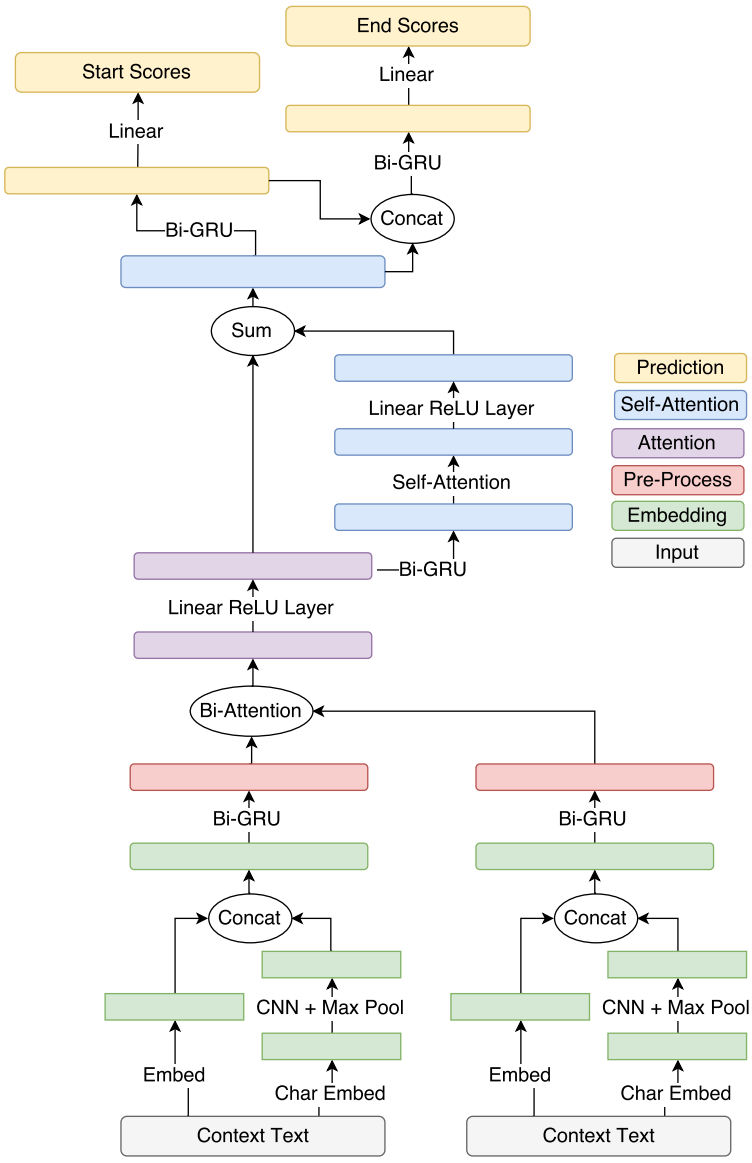

Figure 3.3: High level outline of our model.

We also compute a query-to-context vector $\vec{q_c}$:

$$m_i = \max_{1 \leq j \leq n_q} a_{ij}$$

$$p_i = \frac{e^{m_i}}{\sum_{i=1}^{n_c} e^{m_i}}$$

$$\vec{q_c} = \sum_{i=1}^{n_c} \vec{h_i} p_i$$

The final vector computed for each token is built by concatenating $\vec{h_i}$, $\vec{c_i}$, $\vec{h_i} \odot \vec{c_i}$, and $\vec{q_c} \odot \vec{c_i}$. In our model we subsequently pass the result through a linear layer with ReLU activations.

Self-Attention: Next we use a layer of residual self-attention. The input is passed through another bi-directional GRU. Then we apply the same attention mechanism, only now between the passage and itself. In this case we do not use query-to-context attention and we set $a_{ij} = -inf$ if $i = j$.

As before, we pass the concatenated output through a linear layer with ReLU activations. This layer is applied residually, so this output is additionally summed with the input.

Prediction: In the last layer of our model a bi-directional GRU is applied, followed by a linear layer that computes answer start scores for each token. The hidden states of that layer are concatenated with the input and fed into a second bi-directional GRU and linear layer to predict answer end scores. The softmax operation is applied to the start and end scores to produce start and end probabilities, and we optimize the negative log-likelihood of selecting correct start and end tokens.

Dropout: We also employ variational dropout, where a randomly selected set of hidden units are set to zero across all time steps during training [Gal and Ghahramani, 2016]. We dropout the input to all the GRUs, including the word embeddings, as well as the input to the attention mechanisms, at a rate of 0.2.

Question	Low Confidence Correct Extraction	High Confidence Incorrect Extraction
When is the Members Debate held?	**Immediately after Decision Time** a "Members Debate" is held, which lasts for 45 minutes...	...majority of the Scottish electorate voted for it in a referendum to be held on **1 March 1979** that represented at least...
How many tree species are in the rainforest?	...plant species is the highest on Earth with one 2001 study finding a quarter square kilometer (62 acres) of Ecuadorian rainforest supports more than **1,100** tree species	The affected region was approximately **1,160,000** square miles (3,000,000 km2) of rainforest, compared to 734,000 square miles
Who was Warsz?	....In actuality, Warsz was a 12th/13th century **nobleman** who owned a village located at the modern....	One of the most famous people born in Warsaw was **Maria Sklodowska - Curie**, who achieved international...
How much did the initial LM weight in kg?	The initial LM model weighed approximately 33,300 pounds (**15,000** kg), and...	The module was 11.42 feet (3.48 m) tall, and weighed approximately 12,250 pounds (**5,560** kg)
What do the auricles do?	...many species of lobates have four auricles, gelatinous projections edged with cilia that **produce water currents that help direct microscopic prey toward the mouth**...	The Cestida are ribbon - shaped planktonic animals, with the mouth and aboral **organ aligned in the middle of opposite edges of the ribbon**

Table 3.1: Examples from SQuAD where a paragraph-level model was less confident in a correct extraction from one paragraph (left) than in an incorrect extraction from another (right). Even if the passage has no correct answer, the model still assigns high confidence to phrases that match the category the question is asking about. Because the confidence scores are not well-calibrated, this confidence is often higher than the confidence assigned to the correct answer span.

3.3 Confidence Method

We adapt this model to the multi-paragraph setting by using the un-normalized and un-exponentiated (i.e., before the softmax operator is applied) score given to each span as a measure of the model's confidence. For the boundary-based models we use here, a span's score is the sum of the start and end score given to its start and end token. At test time we run the model on each paragraph and select the answer span with the highest confidence. This is the approach taken by Chen et al. [2017a].

However, training the model as usual can be problematic. During training the model

only takes as input a single paragraph that is known to contain the answer. As result, it possible for the model to learn biased strategies that are effective in this setting, but yields poorly calibrated confidence scores that are not comparable between different paragraphs. Our experiments in Section 3.5 show that this can occur in practice. Table 3.1 shows some qualitative examples of this phenomenon.

We hypothesize that there are two reasons a model's confidence scores might not be well calibrated. First, for models trained with the softmax objective, the pre-softmax scores for all spans can be arbitrarily increased or decreased by a constant value without changing the resulting softmax probability distribution. As a result, nothing prevents models from producing scores that are arbitrarily all larger or all smaller for one paragraph than another. Second, if the model only sees paragraphs that contain answers, it will learn to use biased methods, meaning heuristics or patterns that are only effective when it is known *a priori* that an answer exists. For example, in Table 3.1 we observe that the model will assign high confidence values to spans that strongly match the category of the answer, even if the question words do not match the context. This might work passably well if an answer is present, but can lead to highly over-confident extractions in other cases. Similar kinds of errors have been observed when distractor sentences are added to the context [Jia and Liang, 2017].

We experiment with four approaches to training models to produce comparable confidence scores, shown in the follow subsections. In all cases we will sample paragraphs that do not contain an answer as additional training points.

3.3.1 Shared-Normalization

In this approach all paragraphs are processed independently as usual. However, a modified objective function is used where the normalization factor in the softmax operation is shared between all paragraphs from the same context. Therefore, the probability that token a from

paragraph p starts an answer span is computed as:

$$\frac{e^{s_{ap}}}{\sum_{j \in P} \sum_{i=1}^{n_j} e^{s_{ij}}}$$

where P is the set of paragraphs that are from the same context as p, and s_{ij} is the score given to token i from paragraph j. We train on this objective by including multiple paragraphs from the same context in each mini-batch.

This is similar to simply feeding the model multiple paragraphs from each context concatenated together, except that each paragraph is processed independently until the normalization step. The key idea is that this will force the model to produce scores that are comparable between paragraphs, even when it does not have access to information about the other paragraphs being considered.

3.3.2 Merge

As an alternative to the previous method, we experiment with concatenating all paragraphs sampled from the same context together during training. A paragraph separator token with a learned embedding is added before each paragraph. Our motive is to test whether simply exposing the model to more text will teach the model to be more adept at ignoring irrelevant text.

3.3.3 No-Answer Option

We also experiment with allowing the model to select a special "no-answer" option for each paragraph. First, note that the independent-bounds objective can be re-written as:

$$-\log\left(\frac{e^{s_a}}{\sum_{i=1}^{n} e^{s_i}}\right) - \log\left(\frac{e^{g_b}}{\sum_{j=1}^{n} e^{g_j}}\right) =$$

$$-\log\left(\frac{e^{s_a+g_b}}{\sum_{i=1}^{n}\sum_{j=1}^{n} e^{s_i+g_j}}\right)$$

where s_j and g_j are the scores for the start and end bounds produced by the model for token j, and a and b are the correct start and end tokens. We have the model compute another

score, z, to represent the weight given to a "no-answer" possibility. Our revised objective function becomes:

$$-\log\left(\frac{(1-\delta)e^{z}+\delta e^{s_a+g_b}}{e^{z}+\sum_{i=1}^{n}\sum_{j=1}^{n}e^{s_i+g_j}}\right)$$

where δ is 1 if an answer exists and 0 otherwise. If there are multiple answer spans we use the same objective, except the numerator includes the summation over all answer start and end tokens.

We compute z by adding an extra layer at the end of our model. We compute a soft attention over the span start scores, $p_i = \frac{e^{s_i}}{\sum_{j=1}^{n} e^{s_j}}$, and then take the weighted sum of the hidden states from the GRU used to generate those scores, $\vec{h_i}$, giving $\vec{v_1} = \sum_{i=1}^{n} \vec{h_i} p_i$. We compute a second vector, $\vec{v_2}$ in the same way using the end scores. Finally, a step of learned attention is performed on the output of the Self-Attention layer that computes:

$$a_i = \vec{w} \cdot \vec{h_i}$$

$$p_i = \frac{e^{a_i}}{\sum_{j=1}^{n} e^{a_j}}$$

$$\vec{v_3} = \sum_{i=1}^{n} \vec{h_i} p_i$$

where $\vec{w}$ is a learned weight vector and $\vec{h_i}$ is the vector for token i.

We concatenate these three vectors and use them as input to a two layer network with an 80 dimensional hidden layer and ReLU activations that produces z as its only output.

3.3.4 Sigmoid

As a final baseline, we consider training models with the sigmoid loss objective function. That is, we compute a start/end probability for each token in the context by applying the sigmoid function to the start/end scores of each token. A cross entropy loss is used on each individual probability. The intuition is that, since the scores are being evaluated independently of one another, they will be comparable between different paragraphs.

3.4 Experimental Setup

3.4.1 Datasets

We evaluate our approach on three datasets: TriviaQA unfiltered [Joshi et al., 2017], a dataset of questions from trivia databases paired with documents found by completing a web search of the questions; TriviaQA web, a dataset derived from TriviaQA unfiltered by treating each question-document pair where the document contains the question answer as an individual training point; and SQuAD [Rajpurkar et al., 2016], a collection of Wikipedia articles and crowdsourced questions.

3.4.2 Preprocessing

We note that for TriviaQA web we do not subsample as was done by Joshi et al. [2017], instead training on the full 530k question-document training pairs. We also observed that the metrics for TriviaQA are computed after applying a small amount of text normalization (stripping punctuation, removing articles, ect.) to both the ground truth text and the predicted text. As a result, some spans of text that would have been considered an exact match after normalization were not marked as answer spans during preprocessing, which only detected exact string matches. We fix this issue by labeling all spans of text that would have been considered an exact match by the official evaluation script as an answer span.

In TriviaQA, documents often contain many small paragraphs, so we merge paragraphs together as needed to get paragraphs of up to a target size. We use a maximum size of 400 unless stated otherwise. Paragraph separator tokens with learned embeddings are added between merged paragraphs to preserve formatting information.

3.4.3 Sampling

Our confidence-based approaches are all trained by sampling paragraphs, including paragraphs that do not contain an answer, during training. For SQuAD and TriviaQA web we take the top four paragraphs ranked by TF-IDF score for each question-document pair. We

then sample two different paragraphs from this set each epoch. Since we observe that the higher-ranked paragraphs are much more likely to contain the context needed to answer the question, we sample the highest ranked paragraph that contains an answer twice as often as the others. For the merge and shared-norm approaches, we additionally require that at least one of the paragraphs contains an answer span.

For TriviaQA unfiltered, where we have multiple documents for each question, we find it beneficial to use a more sophisticated paragraph ranking function. In particular, we use a linear function with five features: the TF-IDF cosine distance, whether the paragraph was the first in its document, how many tokens occur before it, and the number of case insensitive and case sensitive matches with question words. The function is trained on the distantly supervised objective of selecting paragraphs that contain at least one answer span. We select the top 16 paragraphs for each question and sample pairs of paragraphs as before.

3.4.4 Implementation

We train the model with the Adadelta optimizer [Zeiler, 2012] with a batch size 60 for TriviaQA and 45 for SQuAD. At test time we select the most probable answer span of length less than or equal to 8 for TriviaQA and 17 for SQuAD. The GloVe 300 dimensional word vectors released by Pennington et al. [2014] are used for word embeddings. On SQuAD, we use a dimensionality of size 100 for the GRUs and of size 200 for the linear layers employed after each attention mechanism. We find for TriviaQA, likely because there is more data, using a larger dimensionality of 140 for each GRU and 280 for the linear layers is beneficial. During training, we maintain an exponential moving average of the weights with a decay rate of 0.999. We use the weight averages at test time.

Model	EM	F1
Baseline [Joshi et al., 2017]	41.08	47.40
BiDAF	50.21	56.86
BiDAF + TF-IDF	53.41	59.18
BiDAF + sum	56.22	61.48
BiDAF + TF-IDF + sum	57.20	62.44
our model + TF-IDF + sum	61.10	66.04

Table 3.2: Results on TriviaQA web using our pipelined method. We significantly improve upon the baseline by combining the preprocessing procedures, TF-IDF paragraph selection, the sum objective, and our model design.

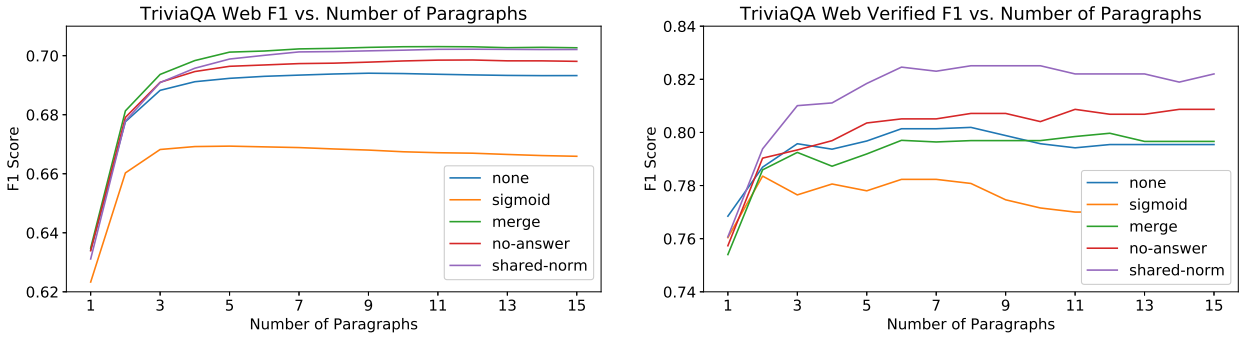

Figure 3.4: Results on TriviaQA web (left) and verified TriviaQA web (right) when applying our models to multiple paragraphs from each document. The shared-norm, merge, and no-answer training methods improve the model's ability to utilize more text, with the shared-norm method being significantly ahead of the others on the verified set and tied with the merge approach on the general set.

Model	All		Verified	
	EM	F1	EM	F1
baseline[Joshi et al., 2017]	40.74	47.06	49.54	55.80
MEMEN* [Pan et al., 2017]	43.16	46.90	49.28	55.83
Mnemonic Reader [Hu et al., 2018]	46.94	52.85	54.45	59.46
Reading Twice for NLU [Weissenborn et al., 2017a]	50.56	56.73	63.20	67.97
S-Norm (ours)	66.37	71.32	79.97	83.70

*Results on the dev set

Table 3.3: Published TriviaQA results when this work was published (10/29/2017). We advance prior work by about 15 points both test sets.

3.5 Results

3.5.1 TriviaQA Web

First, we do an ablation study on TriviaQA web to show the effects of our proposed methods for our pipeline model. We start with an implementation of the baseline from Joshi et al. [2017]. Their system selects paragraphs by taking the first 400 tokens of each document, uses BiDAF [Seo et al., 2017a] as the paragraph model, and selects a random answer span from each paragraph each epoch to be used in BiDAF's cross entropy loss function during training. Paragraphs of size 800 are used at test time. As shown in Table 3.2, our implementation of this approach outperforms the results reported by Joshi et al. [2017] significantly, likely because we are not subsampling the data. We find both TF-IDF ranking and the sum objective to be effective; even without changing the model we significantly improve performance. Using our refined model increases the gain by another 4 points.

Next we show the results of our confidence-based approaches. In this setting we group each document's text into paragraphs of at most 400 tokens and rank them using our TF-IDF heuristic. Then we measure the performance of our proposed approaches as the model

is used to independently process an increasing number of these paragraphs and the model's most confident answer is returned. We additionally measure performance on the verified portion of TriviaQA, a small subset of the question-document pairs in TriviaQA web where humans have manually verified that the document contains sufficient context to answer the question. The results are shown in Figure 3.4.

On these datasets even the model trained without any of the proposed training methods ("none") improves as it is allowed to use more text, showing it does a passable job at focusing on the correct paragraph. The no-answer option training approach lead to a significant improvement, and the shared-norm and merge approach are even better. On the verified set, the shared-norm approach is solidly ahead of the other options. This suggests the shared-norm model is better at extracting answers when it is clearly stated in the text, but worse at guessing the answer in other cases.

We use the shared-norm approach for evaluation on the TriviaQA test set. We found that increasing the paragraph size to 800 at test time, and re-training the model on paragraphs of size 600, was slightly beneficial, allowing our model to reach 66.04 EM and 70.98 F1 on the dev set. We submitted this model to be evaluated on the TriviaQA test set and achieved 66.37 EM and 71.32 F1, firmly ahead of prior work, as shown in Table 3.3. Note that human annotators have estimated that only 75.4% of the question-document pairs contain sufficient evidence to answer the question Joshi et al. [2017], which suggests we are approaching the upper bound for this task. However, the score of 83.7 F1 on the verified set suggests that there is still room for improvement.

3.5.2 TriviaQA Unfiltered

Next we apply our confidence methods to TriviaQA unfiltered. This dataset is of particular interest because the system is not told which document contains the answer, so it provides a plausible simulation of attempting to answer a question using a document retrieval system. We show the same graph as before for this dataset in Figure 3.5. On this dataset it is more important to train the model to produce well-calibrated confidence scores. Note the base

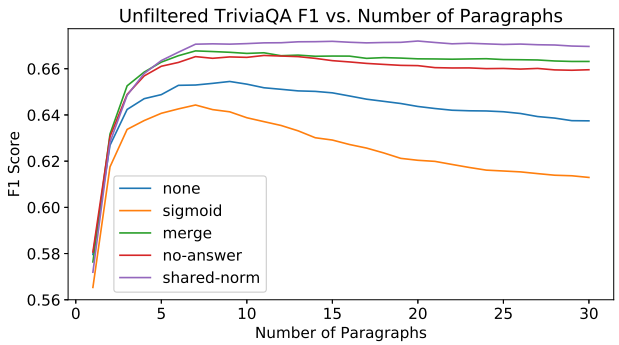

Figure 3.5: Results for our confidence methods on TriviaQA unfiltered. Here we see a more dramatic difference between these models. The shared-norm approach is the strongest, while the base model starts to lose performance as more paragraphs are used.

model and the sigmoid model start to lose performance as more paragraphs are used, showing that errors are being caused by the model being overly confident in incorrect extractions.

3.5.3 SQuAD

We additionally evaluate our model on SQuAD. SQuAD questions were not built to be answered independently of their context paragraph, which makes it unclear how effective of an evaluation tool they can be for document-level question answering. To assess this we manually label 500 random questions from the training set. We categorize questions as:

1. Context-independent, meaning it can be understood independently of the paragraph.

2. Document-dependent, meaning it can be understood given the article's title. For example, "What individual is the school named after?" for the document "Harvard University".

3. Paragraph-dependent, meaning it can only be understood given its paragraph. For example, "What was the first step in the reforms?".

We find 67.4% of the questions to be context-independent, 22.6% to be document-

Model	Dev		Test	
	EM	F1	EM	F1
none	71.60	80.78	72.14	81.05
sigmoid	70.28	79.05	-	-
merge	71.20	80.26	-	-
no-answer	71.51	80.71	-	-
shared-norm	71.16	80.23	-	-

Table 3.4: Results on the standard SQuAD dataset. The test scores place our model as 8th on the SQuAD leader board among non-ensemble models[2]. Training with the proposed multi-paragraph approaches only leads to a marginal drop in performance in this setting.

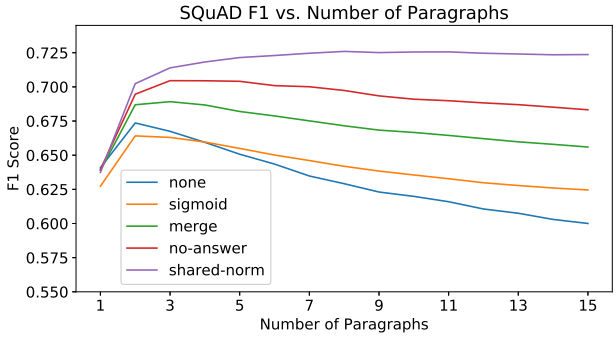

Figure 3.6: Results for our confidence methods on document-level SQuAD. The base model does poorly in this case, rapidly losing performance once more than two paragraphs are used. While all our approaches had some benefit, the shared-norm model is the strongest, and is the only one to not lose performance as large numbers of paragraphs are used.

dependent, and the remaining 10% to be paragraph-dependent. The many document-dependent questions stem from the fact that questions are frequently about the subject of the document, so the article's title is often sufficient to resolve co-references or ambiguities that appear in the question. Since a reasonably high fraction of the questions can be understood given the document they are from, and to isolate our analysis from the retrieval mechanism used, we choose to evaluate on the document-level. We build documents by concatenating all the paragraphs in SQuAD from the same article together into a single document.

The performance of our models given the correct paragraph (i.e., in the standard SQuAD setting), is shown in Table 3.4. Our paragraph-level model is competitive on this task, and our variations to handle the multi-paragraph setting only cause a minor loss of performance.

We graph the document-level performance in Figure 3.6. For SQuAD, we find it crucial to employ one of the suggested confidence training techniques. The base model starts to drop in performance once more than two paragraphs are used. However, the shared-norm approach is able to reach a peak performance of 72.37 F1 and 64.08 EM given 15 paragraphs. Given our estimate that 10% of the questions are ambiguous if the paragraph is unknown, our approach appears to have adapted to the document-level task very well.

Finally, we compare the shared-norm model with the document-level result reported by Chen et al. [2017a]. We re-evaluate our model using the documents used by Chen et al. [2017a], which consist of the same Wikipedia articles SQuAD was built from, but downloaded at different dates. The advantage of this dataset is that it does not allow the model to know *a priori* which paragraphs were filtered out during the construction of SQuAD. The disadvantage is that some of the articles have been edited since the questions were written, so some questions may no longer be answerable. Our model achieves 59.14 EM and 67.34 F1 on this dataset, which significantly outperforms the 49.7 EM reported by Chen et al. [2017a].

---

[2] as of 10/23/2017

3.5.4 Discussion

We found that models that have only been trained on answer-containing paragraphs can perform very poorly in the multi-paragraph setting. The results were particularly bad for SQuAD, we think this is partly because the paragraphs are shorter, so the model had less exposure to irrelevant text. In general, we found the shared-norm approach to be the most effective way to resolve this problem. The no-answer and merge approaches were moderately effective, but we note that they do not resolve the scaling problem inherent to the softmax objective we discussed in Section 3.3, which might be why they lagged behind. The sigmoid objective function reduces the paragraph-level performance considerably, especially on the TriviaQA datasets. We suspect this is because it is vulnerable to label noise, as discussed in Section 3.2.2.

3.6 Related Work in Question Answering

Reading Comprehension Datasets. The state of the art in reading comprehension has been rapidly advanced by neural models, in no small part due to the introduction of many large datasets. The first large scale datasets for training neural reading comprehension models used a Cloze-style task, where systems must predict a held out word from a piece of text [Hermann et al., 2015, Hill et al., 2016]. Additional datasets including SQuAD [Rajpurkar et al., 2016], WikiReading [Hewlett et al., 2016], MS Marco [Nguyen et al., 2016] and TriviaQA [Joshi et al., 2017] provided more realistic questions. Another dataset of trivia questions, Quasar-T [Dhingra et al., 2017], was introduced recently that uses ClueWeb09 [Callan et al., 2009] as its source for documents. In this work we choose to focus on SQuAD and TriviaQA.

Neural Reading Comprehension. Neural reading comprehension systems typically use some form of attention [Wang and Jiang, 2017], although alternative architectures exist [Chen et al., 2017a, Weissenborn et al., 2017b]. Our model follows this approach, but includes some

recent advances such as variational dropout [Gal and Ghahramani, 2016] and bi-directional attention [Seo et al., 2017a]. Self-attention has been used in several prior works [Cheng et al., 2016, Wang et al., 2017, Pan et al., 2017]. Our approach to allowing a reading comprehension model to produce a per-paragraph no-answer score is related to the approach used in the BiDAF-T [Min et al., 2017] model to produce per-sentence classification scores, although we use an attention-based method instead of max-pooling.

Open QA. Open question answering has been the subject of much research, especially spurred by the TREC question answering track [Voorhees et al., 2000]. Knowledge bases can be used, such as in Berant et al. [2013], although the resulting systems are limited by the quality of the knowledge base. Systems that try to answer questions using natural language resources such as YodaQA [Baudiš, 2015] typically use pipelined methods to retrieve related text, build answer candidates, and pick a final output.

Neural Open QA. Open question answering with neural models was considered by Chen et al. [2017a], where researchers trained a model on SQuAD and combined it with a retrieval engine for Wikipedia articles. Our work differs because we focus on explicitly addressing the problem of applying the model to multiple paragraphs. A pipelined approach to QA was recently proposed by Wang et al. [2018], where a ranker model is used to select a paragraph for the reading comprehension model to process.

3.7 Conclusion

We have shown that, when using a paragraph-level QA model across multiple paragraphs, our training method of sampling non-answer containing paragraphs while using a shared-norm objective function can be very beneficial. Combining this with our suggestions for paragraph selection, using the summed training objective, and our model design allows us to significantly improve results on TriviaQA and open SQuAD.

Chapter 4

DEBIASING WITH A KNOWN BIAS

In this chapter, we present a debiasing method that can be applied across a wide range of tasks and biases. It requires a user to have some prior knowledge of the dataset bias (a weakness we address in the following chapter). However, when the bias is known, the methods presented here can be used to leverage that prior knowledge to train a debiased model.

4.1 Introduction

Recent years has seen a large increase in awareness of dataset bias, which in turn has lead to many researchers re-evaluating popular datasets and discovering new biases (see Section 1.1 for examples). In this chapter, we build on these works by showing that, once a dataset bias has been identified, we can improve the out-of-domain performance of models by preventing them from making use of that bias. To do this, we use the fact that these biases can often be explicitly modelled with simple, constrained baseline methods to factor them out of a final model through ensemble-based training.

Our method has two stages. First, we build a bias-only model designed to capture a naive solution that performs well on the training data, but generalizes poorly to out-of-domain settings. Next, we train a second model in an ensemble with the pre-trained bias-only model, which incentivizes the second model to learn an alternative strategy, and use the second model alone on the test set. We explore several different ensembling methods, building on product-of-expert style approaches [Hinton, 2002, Smith et al., 2005]. Figure 4.1 shows an example of applying this procedure to prevent a visual question answering (VQA) model from guessing answers because they are typical for the question, a flaw observed in

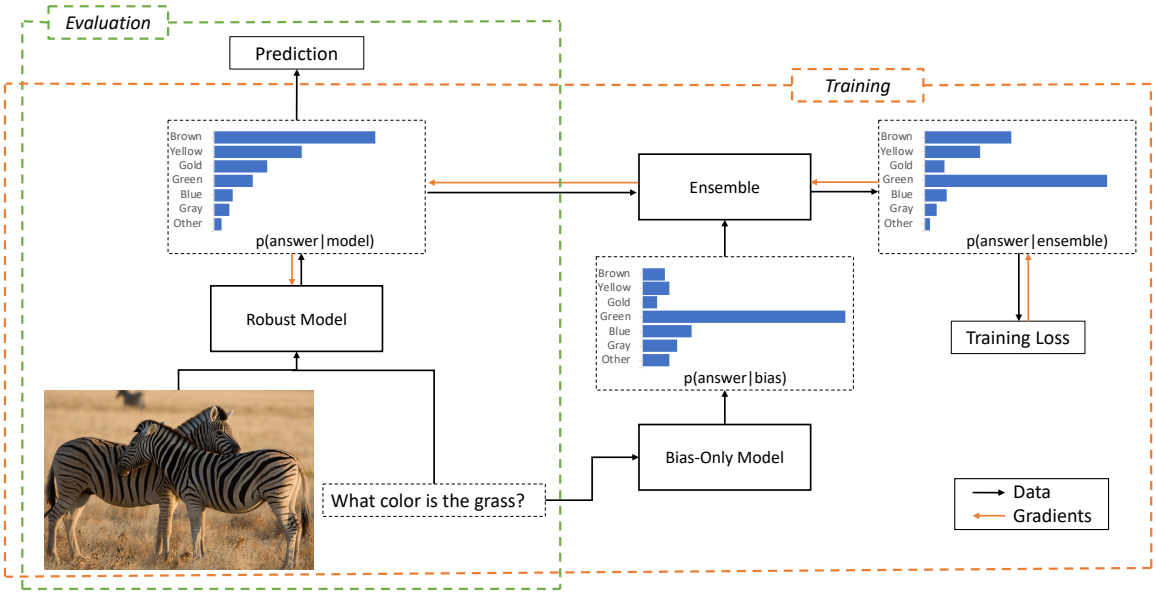

Figure 4.1: An example of applying our method to a Visual Question Answering (VQA) task. We assume predicting green for the given question is almost always correct on the training data. To prevent a model from learning this bias, we first train a bias-only model that only uses the question as input, and then train a robust model in an ensemble with the bias-only model. Since the bias-only model will have already captured the target pattern, the robust model has no incentive to learn it, and thus does better on test data where the pattern is not reliable.

VQA models [Goyal et al., 2018, Agrawal et al., 2018].

We evaluate our approach on a diverse set of tasks, all of which require models to overcome a challenging domain-shift between the train and test data. First, we build a set of synthetic datasets that contain manually constructed biases by adding artificial features to MNLI. We then consider three challenge datasets proposed by prior work [Agrawal et al., 2018, McCoy et al., 2019b, Jia and Liang, 2017], which were designed to break models that adopt superficial strategies on well known textual entailment [Bowman et al., 2015a], reading comprehension [Rajpurkar et al., 2016], and VQA [Agrawal et al., 2018] datasets.

We additionally construct a new QA challenge dataset, TriviaQA-CP (for TriviaQA changing priors). This dataset was built by holding out questions from TriviaQA [Joshi et al., 2017] that ask about particular kinds of entities from the train set, and evaluating on those questions in the dev set, in order to challenge models to generalize between different types of questions.

We are able to improve out-of-domain performance in all settings, including a 6 and 9 point gain on the two QA datasets. On the VQA challenge set, we achieve a 14 point gain, compared to a 3 point gain from prior work. In general, we find using an ensembling method that can dynamically choose when to trust the bias-only model is the most effective, and we present synthetic experiments and qualitative analysis to illustrate the advantages of that approach. We release our datasets and code to facilitate future work.[1]

4.2 Methods

This section describes the two stages of our method, (1) building a bias-only model and (2) using it to train a robust model through ensembling.

---

[1] github.com/chrisc36/debias

4.2.1 Training a Bias-Only Model

The goal of the first stage is to build a model that performs well on training data, but is likely to perform very poorly on the out-of-domain test set. Since we assume we do not have access to examples from the test set, we must apply *a-priori* knowledge to meet this goal.

The most straightforward approach is to identify a set of features that are correlated with the class label during training, but are known to be uncorrelated or anticorrelated with the label on the test set, and then train a classifier on those features.[2] For example, our VQA-CP [Agrawal et al., 2018] bias-only model (see Section 4.4.2) uses the question type as input, because the correlations between question types and answers is very different in the train set than the test set (e.g., 2 is a common answer to "How many..." questions on the train set, but is rare for such questions on the test set).

However, a benefit of our method is that the bias can be modelled using any kind of predictor, giving us a way to capture more complex intuitions. For example, on SQuAD our bias-only model operates on a view of the input built from TF-IDF scores (see Section 4.4.4), and on our changing prior TriviaQA dataset our bias-only model makes use of a pre-trained named entity recognition (NER) tagger (see Section 4.4.5).

4.2.2 Training a Robust Model

This stage trains a robust model that avoids using the method learned by the bias-only model.

Problem Definition

We assume n training examples $\langle x_1, x_2, \ldots, x_n \rangle$, each of which has an integer label y_i, where $y_i \in \{1, 2, \ldots, C\}$ and C is the number of classes. We additionally assume a pre-trained bias-only predictor, h, where $h(x_i) = b_i = \langle b_{i1}, b_{i2}, ..b_{iC} \rangle$ and b_{ij} is the bias-only model's

[2]Since the bias-only model is trained on the same train-set as the robust model care should also be taken to minimize overfitting, although the bias-only model is typically simple enough that this is not an issue.

predicted probability of class j for example i. Finally we have a second predictor function, f, with parameters θ, where $f(x_i, \theta) = p_i$ and p_i is a similar probability distribution over the classes. Our goal is to construct a training objective to optimize θ so that f will learn to select the correct class without using the strategy captured by the bias-only model.

General Approach

We train an ensemble of h and f. In particular, for each example, a new class distribution, $\hat{p_i}$, is computed by combining p_i and b_i. During training, the loss is computed using $\hat{p_i}$ and the gradients are backproped through f. During evaluation f is used alone. We propose several different ensembling methods.

Bias Product

Our simplest ensemble is a product of experts [Hinton, 2002]:

$$\hat{p_i} = softmax(\log(p_i) + \log(b_i))$$

Equivalently, $\hat{p_i} \propto p_i \circ b_i$, where $\circ$ is elementwise multiplication.

Probabilistic Justification: For a given example, x, let x^b be the bias of the example. That is, it is the features we will use in our bias-only model. Let x^{-b} be a view of the example that captures all information about that example except the bias. Assume that x^{-b} and x^b are conditionally independent given the label, c. Then to compute $p(c|x)$ we have:

$$\begin{aligned}
p(c|x) &= p(c|x^b, x^{-b}) && (4.1)\\
&\propto p(c|x^{-b})p(x^b|c, x^{-b}) && (4.2)\\
&= p(c|x^{-b})p(x^b|c) && (4.3)\\
&= p(c|x^{-b})\frac{p(c|x^b)p(x^b)}{p(c)} && (4.4)\\
&\propto p(c|x^{-b})\frac{p(c|x^b)}{p(c)} && (4.5)
\end{aligned}$$

Where 4.2 is from applying Bayes Rule while conditioning on x^{-b}, 4.3 follows from the conditional independence assumption, and 4.4 applies Bayes Rule a second time to $p(x^b|c)$.

We cannot directly model $p(c|x^{-b})$ because it is usually not possible to create a view of the data that excludes the bias. Instead, with the goal of encouraging the model to fall into the role of computing $p(c|x^{-b})$, we compute $p(c|x^b)/p(c)$ using the bias-only model, and train the product of the two models to compute $p(c|x)$.

In practice, we ignore the $p(c)$ factor because, on our datasets, either the classes are uniformly distributed (MNLI), the bias-only model cannot easily capture a class prior since it is using a pointer network (QA), or because we want to remove class priors from model anyway (VQA).

Learned-Mixin

The assumption of conditional independence (Equation 3) will often be too strong. For example, in some cases the robust model might be able to predict the bias-only model will be unreliable for certain kinds of training examples. We find that this can cause the robust model to selectively adjust its behavior in order to compensate for the inaccuracy of the bias-only model, which can lead to errors in the out-of-domain setting (see Section 4.4.1).

Instead we allow the model to explicitly determine how much to trust the bias given the input:

$$\hat{p_i} = \mathit{softmax}(\log(p_i) + g(x_i)\log(b_i))$$

where g is a learned function. We compute g as $\mathit{softplus}(w \cdot h_i)$ where w is a learned vector, h_i is the last hidden layer of the model for example x_i, and the $\mathit{softplus}(x) = \log(1 + e^x)$ function is used to prevent the model reversing the bias by multiplying it by a negative weight. w is trained with the rest of the model parameters. This reduces to bias product when $g(x_i) = 1$.

A difficulty with this method is that the model could learn to integrate the bias into p_i and set $g(x_i) = 0$. We find this does sometimes occurs in practice, and our next method

alleviates this challenge.

Learned-Mixin +H

To prevent the learned-mixin ensemble from ignoring b_i, we add an entropy penalty to the loss:

$$R = wH(softmax(g(x_i)\log(b_i)))$$

Where $H(z) = -\sum_j z_j \log(z_j)$ is the entropy and w is a hyperparameter. Penalizing the entropy encourages the bias component to be non-uniform, and thus have a greater impact on the ensemble.

4.3 Evaluation Methodology

We evaluate our methods on several datasets that have out-of-domain test sets. Some of these tasks, such as HANS [McCoy et al., 2019b] or Adversarial SQuAD [Jia and Liang, 2017], can be solved easily by generating additional training examples similar to the ones in the test set (e.g., Wang and Bansal [2018]). We, instead, demonstrate that it is possible to improve performance on these tasks by exploiting knowledge of general, biased strategies the model is likely to adopt.

Our evaluation setup consists of a training set, an out-of-domain test set, a bias-only model, and a main model. To run an evaluation we train the bias-only model on the train set, train the main model on the train set while employing one of the methods in Section 4.2, and evaluate the main model on the out-of-domain test set. We also report performance on the in-domain test set, when available. We use models that are known to work well for their respective tasks for the main model, and do not further tune their hyperparameters or perform early stopping.

We consider two extractive QA datasets, which we treat as a joint classification task where the model must select the start and end answer token [Wang and Jiang, 2017]. For

Task	Dataset	Domain Shift	Bias-Only Model	Main Model
NLI	Synthetic MNLI	Synthetic indicator features are randomized	Indicator features	Co-Attention
VQA	VQA-CP v2.0	Correlations between question-types and answers are altered	Question-type	BottomUpTopDown
NLI	HANS	Sentence pairs always contain the same words	Shared word features	BERT & Co-Attention
QA	Adv. SQuAD	Distractor sentences are added to the context	TF-IDF sentence selector	Modified BiDAF
QA	TriviaQA-CP	Questions ask about different kinds of entities	NER answer detector	Modified BiDAF

Table 4.1: Summary of the evaluations we perform, Domain Shift refers to what changes between the train and test data, and Bias-Only Model specifies how the bias model we use was constructed. See the main text for details.

these datasets, we build independent bias-only models for selecting the start and end token, and separately ensemble those biases with the classifier's start token and end token output distributions. We apply a ReLU layer to the question and passage embeddings, followed by max-pooling, to construct a hidden state for computing the learned-mixin weights.

We compare our methods to a reweighting baseline described below, and to training the main model without any modifications. On VQA we also compare to the adversarial methods from Ramakrishnan et al. [2018] and Grand and Belinkov [2019]. The other biases we consider are not based on observing only part of the input, so these adversarial methods cannot be directly applied.

4.3.1 Reweight Baseline

As a non-ensemble baseline, we train the main model on a weighted version of the data, where the weight of example x_i is $1 - b_{iy_i}$ (i.e., we weigh examples by one minus the probability the bias-only model assigns the correct label). This encourages the main model to focus on examples the bias-only model gets wrong.

4.3.2 Hyperparameters

One of our methods (Learned-Mixin +H) requires hyperparameter tuning. However hyperparameter tuning is challenging in our setup since our assumption is that we have no access to out-of-domain test examples during training. A plausible option would be to tune hyperparameters on a dev set that exhibits a related, but not identical, domain shift to the test set, but unfortunately none of our datasets have such dev sets. Instead we follow prior work [Grand and Belinkov, 2019, Ramakrishnan et al., 2018] and perform model selection on the test set. Although this presents an important caveat to the results of this method, we think it is still of interest to observe that the entropy regularizer can be very impactful. Future work may be able to either construct suitable development sets, or propose other hyperparameter-tuning methods to relieve this issue.

4.4 Experiments

We provide experiments on five different domains, summarized in Table 4.1, each of which requires models to overcome a challenging domain-shift between train and test data. In the following section we provide summaries of the datasets, main models and bias-only models. We provide more details of the models used, and the TriviaQA challenge dataset we construct, in subsequent sections.

4.4.1 Synthetic Data

Data: We experiment with a synthetic dataset built by modifying MNLI [Bowman et al., 2015a]. In particular, we add a feature that is correlated with the class label to the train set, and build an out-of-domain test set by adding a randomized version of that feature to the MNLI matched dev set. We additionally construct an in-domain test set by modifying the matched dev set in the same way as was done in the train set. We build three variations of this dataset:

Indicator: Adds the token "0", "1", or "2" to the start of the hypothesis, such that 80% of the time the token corresponds to the example's label (i.e., "0" if the class is "entailment", "1" if the class is contradiction, ect.). In the out-of-domain test set, the token is selected randomly.

Excluder: The same as Indicator, but with a 3% chance the added token corresponds to the example's label, meaning the token can usually be used to eliminate one of the three output classes.

Dependent: In the previous two settings, the added bias is independent of the example given the example's label. To simulate a case where this independence is broken, we experiment with adding an additional feature that is correlated with the bias feature, but is not treated as being part of the bias (i.e., it is not used by the bias-only model). In particular,

Debiasing Method	Indicator		Excluder		Dependent	
	Acc.	w/Bias	Acc.	w/Bias	Acc.	w/Bias
None	69.36	86.49	68.06	83.56	63.23	87.90
Reweight	75.44	82.74	70.36	83.29	69.81	85.50
Bias Product	76.27	81.32	77.33	80.41	71.85	84.98
Learned-Mixin	76.29	81.35	77.80	78.86	75.75	77.70
Learned-Mixin +H	76.77	77.65	77.90	78.57	75.79	76.65
Unbiased Training	78.94	78.94	78.94	78.94	78.86	78.86

Table 4.2: Results on MNLI with different kinds of synthetic bias. The Acc columns show the accuracy on the out-of-domain test set, and the w/Bias columns show accuracy on the in-domain test. Unbiased Training is an upper bound constructed by training a model with the same randomized features that are used at test time.

80% of the time a token is added to the start of the hypothesis that matches the label with 90% probability, and the "0" token is appended to the end of the hypothesis. The other 20% of the time a random token is prepended and "1" is appended.

Bias-Only Model: The bias-only model predicts the label using the first token of the hypothesis.

Main Model: We use a recurrent co-attention model, similar to ESIM Chen et al. [2017b]. Details are given in Section 4.6.

Results: Table 4.2 shows the results. All ensembling methods work well on the Indicator bias. The reweight method performs poorly on the Excluder bias, likely because the bias-only model assigns the correct class approximately 50% probability for almost all the training examples, making the weights mostly uniform. This illustrates a general weakness with

reweighting methods: they require at least a small number of bias-free examples for the model to learn from.

The bias product method performs poorly on the Dependent bias. Inspection shows that, when the indicator is 1, the bias product model is anti-correlated with the bias. In particular, it assigns an average of 22.5% probability to the class indicated by the bias, where an unbaised model would assign an average of 33% since the bias is random. The root cause is that, if the indicator is 1, the model knows the bias is likely to be wrong, so it learns to subtract the value the bias-only model will produce from its own output in order to cancel out the bias-only model's effect on the ensemble's output.

The learned-mixin model does not suffer from this issue, and assigns the class indicated by the bias an average of 34.5% probability. Analysis shows that $g(x_i)$ is set to 0.00 ± 0.0001 when the indicator is turned off, and to 1.91 ± 0.285 otherwise, showing that the model learns to turn off the bias-only component of the ensemble as needed, thus avoiding this over-compensating issue. The entropy regularizer appears to be unnecessary on this dataset because $g(x_i)$ does not go to zero.

4.4.2 VQA-CP

Data: We evaluate on the VQA-CP v2 [Agrawal et al., 2018] dataset, which was constructed by re-splitting the VQA 2.0 [Goyal et al., 2018] train and validation sets into new train and test sets such that the correlations between question types and answers differs between each split. For example, "tennis" is the most common answer for questions that start with "What sport..." in the train set, whereas "skiing" is the most common answers for those questions in the test set. Models that choose answers because they are typical in the training data will perform poorly on this test set.

Bias-Only Model: VQA-CP comes with questions annotated with one of 65 question types, corresponding to the first few words of the question (e.g., "What color is"). The bias-only model uses this categorical label as input, and is trained on the same multi-label objective

Debiasing Method	Acc.
None	39.37
Reweight	40.23
Bias Product	40.10
Learned-Mixin	50.16
Learned-Mixin +H	53.81
Ramakrishnan et al. [2018]	41.17
Grand and Belinkov [2019]	42.33

Table 4.3: Results on the VQA-CP v2.0 test set.

as the main model.

Main Model: We use a popular implementation[3] of the BottomUpToDown [Anderson et al., 2018a] VQA model. This model uses a multi-label objective, so we apply our ensemble methods by treating each possible answer as a two-class classification problem.[4]

Results: Table 4.3 shows the results. The learned-mixin method was highly effective, boosting performance on VQA-CP by about 9 points, and the entropy regularizer can increase this by another 3 points, significantly surpassing prior work. For the learned-mixin ensemble, we find $g(x_i)$ is strongly correlated with the bias's expected accuracy[5], with a spearmanr correlation of 0.77 on the test data. Qualitative examples (Figure 4.2) further suggest the model increases $g(x_i)$ when it knows if can rely on the bias-only model.

[3]github.com/hengyuan-hu/bottom-up-attention-vqa

[4]Since the bias sometimes assigns a zero probability to an answer, we additionally add $\sigma(\alpha)$ to the bias probabilities where α is learned parameter to allow the model to soften the bias as needed

[5]Computed as $\sum_j s_{ij} b_{ij} / \sum_j b_{ij}$ where s_{ij} is the score for class j on example i

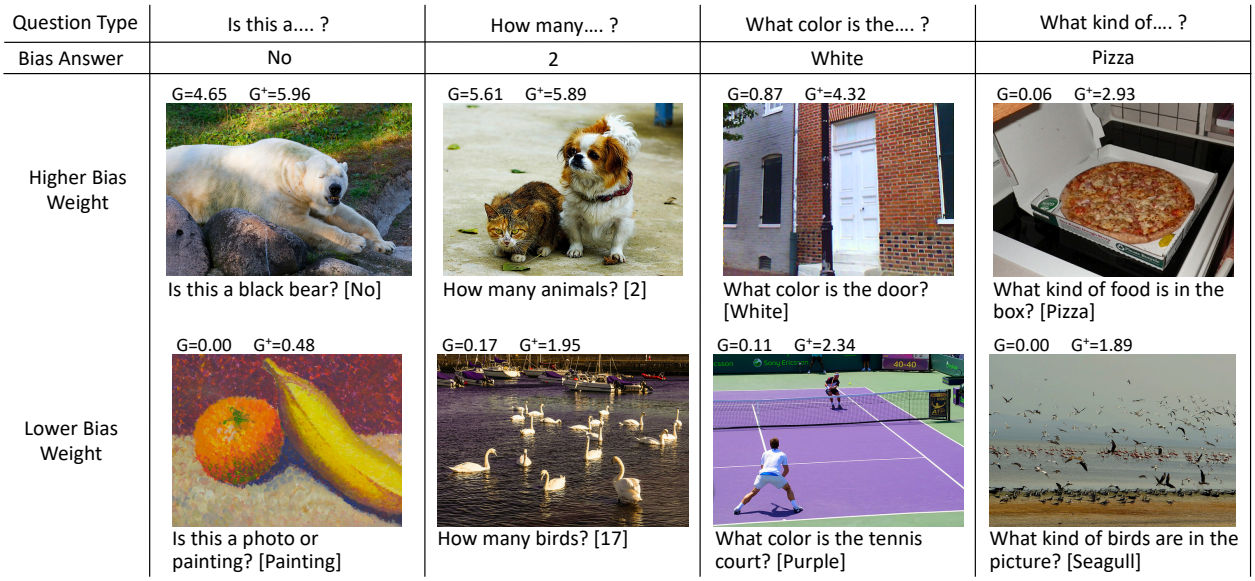

Figure 4.2: Qualitative examples of the values of $g(x_i)$ on the VQA-CP training data for the learned-mixin model (labelled "G") and learned-mixin +H model (labelled "G+"). The question type and the bias model's highest ranked answer for that type are shown above. We find $g(x_i)$ is larger when the bias answers are likely to be correct.

4.4.3 HANS

Data: We evaluate on the HANS adversarial MNLI dataset [McCoy et al., 2019b]. This dataset was built by constructing templated examples of entailment and non-entailment, such that the hypothesis sentence only includes words that are also in the premise sentence. Naively trained models tend to classify all such examples as "entailment" because detecting the presence of many shared words is an effective tactic on MNLI.

Bias-Only Model: The bias-only model is a shallow linear classifier using the following features: (1) whether the hypothesis is a sub-sequence of the premise, (2) whether all words in the hypothesis appear in the premise, (3) the percent of words from the hypothesis that appear in the premise, (4) the average of the minimum distance between each premise word with each hypothesis word, measured using cosine distance with the fasttext Mikolov et al.

Debiasing Method	Co-Attention		BERT	
	HANS	MNLI	HANS	MNLI
None	50.58	78.73	62.40	84.24
Reweight	52.85	77.03	69.19	83.54
Bias Product	53.69	76.63	67.92	82.97
Learned-Mixin	51.65	78.05	64.00	84.29
Learned-Mixin +H	53.35	74.50	66.15	83.97

Table 4.4: Accuracy on the adversarial MNLI dataset, HANS, and the MNLI matched dev set.

[2018] word vectors, and (5) the max of those same distances. We constrain the bias-only model to put the same amount of probability mass on the neutral and contradiction classes so it focuses on distinguishing entailment and non-entailment, and reweight the dataset so that the entailment and non-entailment examples have an equal total weight to prevent a class prior from being learned.

Main Models: We experiment with both the uncased BERT base model [Devlin et al., 2019], and the same recurrent model used for the synthetic data. We use the default hyper-parameters for BERT since they work well for MNLI.

Results: Table 4.4 shows the results. For the recurrent method, both the bias product and learned-mixin +H methods result in about a three point gain. However, for the BERT model, the simpler reweight method is more effective. We noticed high variance in performance between runs in this setting (this observation has been made by other researchers [McCoy et al., 2019a]), and speculate the ensemble methods might be compounding this instability by introducing additional complexity.

We show scores for individual heuristics used in HANS in Table 4.5. Our methods reduce

Model	Debiasing Method	Correct: *Entailment*			Correct: *Non-entailment*		
		Lexical	Subseq.	Const	Lexical	Subseq.	Const
Co-Attention	None	97.83	99.67	97.28	1.37	3.68	3.68
	Reweight	80.10	77.84	73.76	15.68	34.27	35.44
	Bias Product	77.89	76.61	70.95	17.89	35.11	43.71
	Learned-Mixin	94.84	97.25	91.19	3.69	9.57	13.37
	Learned-Mixin +H	67.18	61.05	47.13	27.41	56.82	60.53
BERT	None	96.30	99.58	99.30	49.03	7.88	22.30
	Reweight	67.93	84.34	80.97	77.44	44.87	59.57
	Bias Product	53.67	69.47	70.88	81.34	62.93	69.23
	Learned-Mixin	95.64	99.52	99.14	55.41	8.34	25.96
	Learned-Mixin +H	91.98	98.20	97.98	64.99	13.25	30.48

Table 4.5: Scores on individual heuristics in HANS.

the extent to which models naively guess entailment in all cases. Interestingly, the BERT model shows significantly degraded performance on the entailment examples when using the reweight and bias product method, but largely maintains its performance on those examples when using the learned-mixin method.

4.4.4 Adversarial SQuAD

Data: We evaluate on the Adversarial SQuAD [Jia and Liang, 2017] dataset, which was built by adding distractor sentences to the passages in SQuAD [Rajpurkar et al., 2016]. The sentences are built to closely resemble the question and contain a plausible answer candidate, but with a few key semantic changes to ensure they do not incidentally answer the question. Models that naively focus on sentences that contain many question words are often fooled by the new sentence.

Bias-Only Models: We consider two bias-only models: (1) TF-IDF: the TF-IDF score between each sentence and question is used to select an answer (meaning tokens within the same sentence all get the same score) and (2) TF-IDF Filtered: the same but excluding pronouns and numbers from the words used to compute the TF-IDF scores. The second model is motivated by the fact distractor sentences never include numbers or pronouns that occur in the question.

Main Model: We use an updated version of BiDAF [Seo et al., 2017b], that uses the fasttext words vectors [Mikolov et al., 2018], includes an additional recurrent layer, and simplifies the prediction stage. More details are shown in Section 4.6.

Results: Table 4.6 shows the results. We find the bias product method improves performance by up to 3 points, and the learned-mixin +H model achieves up to a 9 point gain. The importance of including the entropy penalty is explained by the fact that, without the penalty, the model learns to ignore the bias by settings $g(x_i)$ close to zero. For example, on

Debiasing Method	TF-IDF Filtered			TF-IDF		
	AddSent	AddSentOne	Dev	AddSent	AddSentOne	Dev
None	42.54	53.91	80.61	42.54	53.91	80.61
Reweight	41.55	53.06	80.59	42.74	53.83	80.51
Bias Product	47.17	57.74	78.63	44.41	55.73	78.22
Learned-Mixin	42.25	53.51	80.39	42.00	53.46	80.46
Learned-Mixin +H	51.84	60.66	75.94	48.30	58.26	74.14

Table 4.6: F1 scores on Adversarial SQuAD and the standard SQuAD dev set using two different bias-only models.

the AddSent dataset with the TF-IDF filtered bias, the learned-mixin ensemble sets $g(x_i)$ to an average of 0.13, while the learned-mixin +H ensemble increases that to 5.16. The high values are likely caused by the fact the bias-only model is very weak, since it assigns the same score to each token in each sentence, so the model can often scale it by large values. As expected, we get better results using the TF-IDF Filtered bias which is more closely tailored to how the test set was constructed.

4.4.5 TriviaQA-CP

Data: We construct a changing-prior QA dataset from TriviaQA [Joshi et al., 2017] by categorizing questions into three classes, Person, Location, and Other, based on what kind of entity they are asking about. During training, we hold out all the person questions or all the location questions from the train set, and evaluate on the person or location questions in the TriviaQA dev set. Details can be found in Section 4.5.

Bias-Only Model: The bias-only model uses NER tags, identified by running the Stanford NER Tagger [Finkel et al., 2005] on the passage, as input. We only apply the model to tokens

Debiasing Method	Location		Person	
	CP	Dev	CP	Dev
None	41.23	59.27	39.69	55.26
Reweight	40.14	59.18	39.96	55.38
Bias Product	44.42	60.02	40.58	55.20
Learned-Mixin	41.15	61.64	41.31	56.08
Learned-Mixin +H	47.77	57.74	44.37	54.83

Table 4.7: EM scores on two changing priors TriviaQA datasets. The CP column shows scores on the changing priors test set, and Dev shows in-domain scores.

that have a NER tag, and assign all other tokens the average score given to the tokens with NER tags in order to prevent the model from reflecting a preference for entity tokens in general.

Main Model: We use a larger version of the model used for Adversarial SQuAD (see Section 4.6), to account for the larger dataset.

Results: Table 4.7 shows the results. Similar to adversarial SQuAD, the bias product method is moderately effective, and the ensemble method is superior as long as a suitable regularizer is applied. We again observe that the learned-mixin method tends to push $g(x_i)$ close to zero without the entropy penalty (average of 0.25 without the penalty vs. 5.01 with the penalty on the Location dev set). We see smaller gains on the person dataset. One possible cause is that differentiating between people and other named entities, such as organizations or groups, is difficult for the main model, and as a result it does not learn a strong non-person prior even without the use of a debiasing method.

Statistic	Location	Person
Num Train	60,133	52,953
Num Test	1,992	2,865
Avg. Passage Length	318	317
Avg. Question Length	16.7	16.0

Table 4.8: Statistics for the TriviaQA-CP datasets.

4.4.6 Discussion

Despite tackling a diverse range of problems, we were able to improve out-of-domain performance in all settings. The bias product method works consistently, but can almost always be significantly out-performed by the learned-mixin method with an appropriate entropy penalty. The reweight baseline improved performance on HANS, but was relatively ineffective in other cases.

Increasing the out-of-domain performance usually comes at the cost of losing some in-domain performance, which is unsurprising since the biased approaches we are removing are helpful on the in-domain data. TriviaQA-CP stands out as a case where this trade-off is minimal.

4.5 TriviaQA-CP Details

In this section we discuss our changing-prior TriviaQA dataset, TriviaQA-CP, in more detail. This dataset was built by training a classifier to identify TriviaQA [Joshi et al., 2017] questions as being about people, locations, or other topics, and then selecting an answer-containing passage for each question as context. There are two versions of this dataset: a person changing-priors dataset that was built by removing the person questions from the train set and using only person questions from the dev set for evaluation, and a location changing-priors dataset that was built by repeating this process for location questions.

Method	Accuracy	Location			Person		
		Precision	Recall	F1	Precision	Recall	F1
Patterns	72.84	98.30	70.61	82.19	99.12	33.43	50.00
Yago	88.56	95.87	85.31	90.28	94.70	80.00	86.73
Yago + Patterns	91.73	95.44	93.88	94.65	94.70	85.37	89.80
Dist. Supervised Model	92.73	96.60	92.65	94.58	98.28	85.37	91.37
Supervised Model	94.46	95.08	94.69	94.89	93.31	95.82	94.55

Table 4.9: Accuracy, and per-class scores, on the manually annotated questions for the various question classification methods we used when building TriviaQA-CP.

Statistics for these two sets are shown in Table 4.8. We review the three-step procedure we used to construct this dataset below.

Distantly Supervised Classification: We first train a preliminary question-type classifier using distant supervision. We noisily label person and location questions using a manually constructed set of patterns (e.g., questions with the phrase "What is the family name of..." are almost always about people), and by attempting to look up the answers in the Yago database [Suchanek et al., 2007] and checking if the answer belongs to a person or location category. Questions that did not match either of these heuristics are labelled as other.

We use these labels to train a simple recurrent model that embeds the question using the fasttext words vectors, applies a 100 dimensional BiLSTM, max-pools, and then applies a softmax layer with 3 outputs. We train the model for 3 epochs using the Adam optimizer [Kingma and Ba, 2014], and apply 0.5 dropout to the embeddings and 0.2 dropout to the recurrent states and the output of the max-pooling layer.

Supervised Classification: Next we use higher quality labels to train a second linear

classifier to re-calibrate the recurrent model's predictions, and to integrate its predictions with the distantly supervised heuristics. An author manually labelled 1,100 questions, then a classifier was trained on those questions using the predictions from the recurrent model as features, as well as two additional features built from looking up the category of the answer in Yago as before. This classifier was then used to decide the final question classifications.

Table 4.9 shows the accuracy of these classifiers. The final model achieves about 95% accuracy. We find about 25% of the questions are about people and about 20% of the questions are about locations.

Paragraph Selection: In TriviaQA, each question is paired with multiple documents. We simplify the task by selecting a single answer-containing paragraph for each question. We use the approach of Clark and Gardner [2017] to break up the documents into passages of at most 400 tokens, and rank the passages for each question using their linear paragraph ranker. Each question is then paired with the highest ranking paragraph that contains an answer.

4.6 Model Details

Here we specify the models we used in our experiments in more detail.

4.6.1 Co-Attention NLI Model

The model we use for NLI is based on ESIM [Chen et al., 2017b]. It has the following stages:

Embed: Embed the words using a character CNN, following what was done by Seo et al. [2017b], and the fasttext crawl word embeddings [Mikolov et al., 2018], then run a shared BiLSTM over the results.

Co-Attention: Compute an attention matrix using the formulation from Seo et al. [2017b], and use it to compute a context vector for each premise word [Bahdanau et al., 2015]. Then

build an augmented vector for each premise word by concatenating the word's embedding, the context vector, and the elementwise product of the two. Augmented vectors for the hypothesis are built in the same way using the transpose of the attention matrix.

Pool: Run another shared BiLSTM over the augmented vectors, and max-pool the results. The max-pooled vectors from the premise and hypothesis are fed into a fully-connected layer, and then into a softmax layer with three outputs to compute class probabilities.

We apply variational dropout at a rate of 0.2 between all layers, and to the recurrent states of the LSTM, and train the model for 30 epochs using the Adam optimizer [Kingma and Ba, 2014] with a batch size of 32. The learning rate is decayed by 0.999 every 100 steps. We use 200 dimensional LSTMs and a 50 dimensional fully connected layer.

4.6.2 Modified BiDAF QA Model

The model we use for QA is based on BiDAF [Seo et al., 2017b]. It has the following stages:

Embed: Embed the words using a character CNN following Seo et al. [2017b] and the fasttext crawl word embeddings [Mikolov et al., 2018]. Then run a BiLSTM over the results to get context-aware question embeddings and passage embeddings.

Bi-Attention: Apply the bi-directional attention mechanism from Seo et al. [2017b] to produce question-aware passage embeddings.

Predict: Apply a fully connected layer, then two more BiLSTM layers, then a two dimensional linear layer to produce start and end scores for each token.
We apply variational dropout at a rate of 0.2 between all layers. We use the Adam optimizer [Kingma and Ba, 2014] with a batch size of 45, while decaying the learning rate by 0.999 every 100 steps. For SQuAD, we use a 200 dimensional fully connected layer and

100 dimensional LSTMs. For TriviaQA we use a 256 dimensional fully connected layer and 128 dimensional LSTMs, with highway connections between each BiLSTM [Srivastava et al., 2015] and a recurrent dropout rate of 0.2.

4.7 Conclusion

Our key contribution is a method of using human knowledge about what methods will not generalize well to improve model robustness to domain-shift. Extensive experiments show that our method works well on two adversarial datasets, and two changing-prior datasets, including a 12 point gain on VQA-CP.

Chapter 5

DEBIASING WHEN THE BIAS IS UNKNOWN

In this chapter, we extend our previous work to construct a method of training unbiased models without needing a pre-specified bias-only model.

5.1 Introduction

In the previous chapter, we showed it is possible to prevent models from adopting biased methods when the bias is known and can be carefully modeled in advance. Here we present a method that can achieve similar results, but that automatically learns the bias, removing the need for such dataset-specific knowledge. To make this possible, we observe that many known examples of dataset bias involve models learning overly simple patterns [Min et al., 2019, McCoy et al., 2019b, Anand et al., 2018]. This leads us to propose that many dataset biases will be shallower and easier to model than more generalizable patterns. This reflects the intuition that high-quality models for tasks like language comprehension or image understanding will require some minimum amount of complexity (e.g., a visual question answering model should at least consider the question, image, and ways they might correspond), and therefore shallower approaches are likely to be modelling dataset bias.

Our method, called Mixed Capacity Ensembling (MCE), follows prior work [Clark et al., 2019b, He et al., 2019] by training an ensemble of two models, one of which captures bias, and one of which captures other, better generalizing patterns. Prior methods required that the model that captures the bias be pre-specified by domain experts. We instead achieve this separation by jointly training both models while encouraging simpler patterns to be isolated into one model and more complex patterns into the other. In particular, we (1) ensemble a lower capacity model, i.e., a model with fewer parameters, and a higher capacity

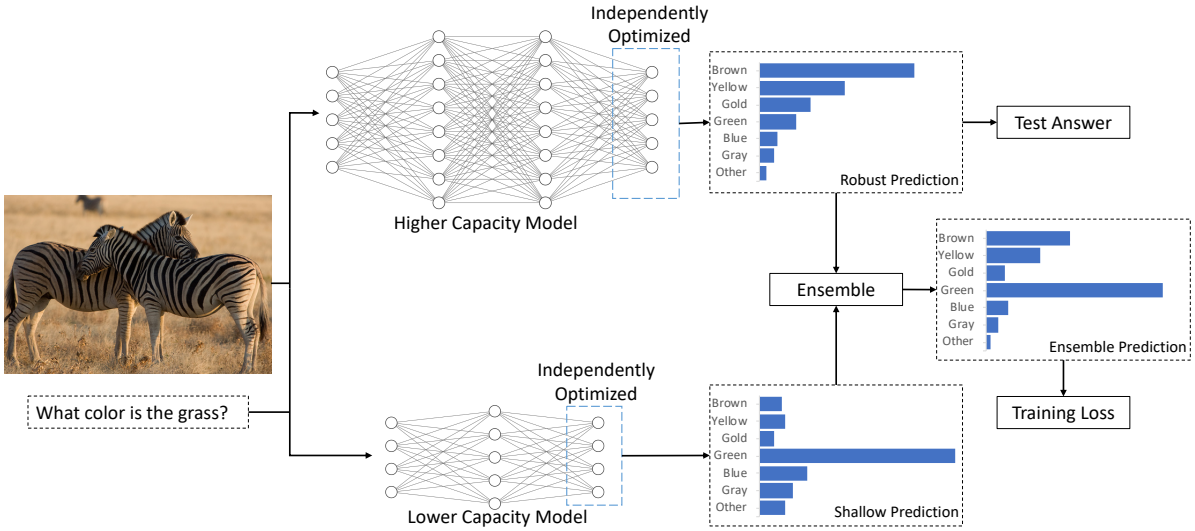

Figure 5.1: An overview of our method. We train a lower capacity model in an ensemble with a higher capacity model. During training simplistic correlations (e.g., "grass is usually green") are captured by the lower capacity model, which frees the higher capacity model to focus on more robust patterns (i.e., matching the question with the image). At test time, the higher capacity model is used alone. We use an independently optimized classifier as the final layer of each model as part of our method to make them conditionally independent (Section 5.2.4).

model, which creates a natural tendency for the higher capacity model to capture more complex patterns, (2) put a small weight on the loss of the lower capacity model so that it is preferentially used to capture simpler patterns that it can model, and (3) enforce conditional independence between the models so that they learn non-overlapping strategies. We show that conditional independence can be achieved by ensuring each classifier makes individually optimal predictions, and we train the ensemble with this constraint using methods from bi-level optimization [Colson et al., 2007].

We evaluate our method by training models on datasets with known biases, and then testing them on out-of-domain datasets built to penalize models that learn to use those biases. First, we construct a series of synthetic datasets to show our method can adapt to multiple kinds of biases. Then, we consider three datasets from prior work that test against question-type biases for visual question answering [Goyal et al., 2018] and hypothesis keyword

biases [Bowman et al., 2015a, Gururangan et al., 2018] or lexical overlap biases [McCoy et al., 2019b] for sentence entailment. Finally, we construct an image recognition dataset using Imagenet [Deng et al., 2009b] that includes a test set of examples with misleading backgrounds (e.g., a fish photographed on dry land) to test our method on background-class biases. We show improved performance in all settings, in some cases nearly matching the results that can be achieved with an upper-bound that does use knowledge of the bias being targeted.

5.2 Mixed Capacity Ensembles

In this section, we present our Mixed Capacity Ensembling method and the motivations behind it. We also discuss an extension to cases where shallow patterns can partially solve the task by eliminating obviously wrong answers.

5.2.1 Problem Definition

Let $\mathcal{X}$ be the domain of the input, $\mathcal{Y} = \{1, 2, \ldots, C\}$ be the space of the labels, and $\mathcal{B}_y$ be the space of probability distributions on $\mathcal{Y}$. Assume we have a training dataset of n examples, $\{(x_i, y_i)_{i=1}^n\}$, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ and x_i, y_i are drawn from the joint distribution $P^{train}(X, Y)$.

Our goal is to learn the parameters θ_h of a differentiable function f_h that returns a vector in $\mathbb{R}^C$ representing a probability distribution over the possible classes, $f_h(\cdot, \theta) : \mathcal{X} \rightarrow \mathcal{B}_y$. For notational simplicity, we will sometimes write $f_h(x_i, \theta_h)$ simply as $f_h(x_i)$. Our goal is to optimize θ_h so that $f_h(\cdot, \theta_h)$ will have high accuracy on an out-of-domain test set drawn from $P^{test}(X, Y)$.

5.2.2 Motivation

Prior work has generally relied on domain-specific assumptions to make this task possible (e.g., question-only strategies will not generalize well to P^{test} for VQA). Our approach is

designed to replace those assumptions with a more domain-general one: that overly simplistic patterns are unlikely to generalize to P^{test}.

While there is no guarantee all examples of dataset bias will fit this criteria, it is commonly true in the growing body of research on dataset bias. Additionally, we expect complex dataset biases to be less problematic because models will be less prone to fit to them.

Achieving this through regularization would be challenging since it is not obvious how to penalize the use of simplistic patterns. This motivates our approach of explicitly modeling simplistic hypotheses during training and discarding them during testing.

5.2.3 Training an Ensemble

Formally, our method introduces a lower capacity model: $f_l(\cdot, \theta_l) : \mathcal{X} \rightarrow \mathcal{Y}$ and additionally computes a class prior $p_y \in \mathcal{B}_y$ by computing the expected value of y in the training data. We then compute predictions for the ensemble, lower capacity model, and higher capacity model as follows:

$$\hat{y}_i^e = softmax(\log(f_h(x_i)) + \log(f_l(x_i)) + \log(p_y)) \tag{5.1}$$

$$\hat{y}_i^l = softmax(\log(f_l(x_i)) + \log(p_y)) \tag{5.2}$$

$$\hat{y}_i^h = softmax(\log(f_h(x_i)) + \log(p_y)) \tag{5.3}$$

We explicitly factor in p_y so that it can be properly integrated into all three predictions (if the class prior was encoded as part of f_l and f_h it would be double-counted when the two functions are ensembled). During training the loss is computed as:

$$Loss = \sum_{i=1}^{n} L(\hat{y}_i^e, y_i) + wL(\hat{y}_i^l, y_i) \tag{5.4}$$

Where L is the cross-entropy loss function and w is a hyperparameter. During testing we make predictions using $\hat{y}_i^h$.

Following our simplicity assumption, we expect both f_l and f_h to be able to model dataset bias, but due to the additional loss penalty on f_l the ensemble will favor using f_l for that

purpose. Additionally, since f_h can better represent more complex patterns, the ensemble will use f_h for that purpose.

5.2.4 Adding Conditional Independence

Although this creates a soft incentive for the models to learn different patterns, there is a risk this separation will not be completely clean (e.g., the higher capacity model might partially capture the relatively simple patterns we hope to model with the lower capacity model). To prevent this, we propose to enforce conditional independence between the two models so $f_l(x) \perp\!\!\!\perp f_h(x)|y$. We do not expect the models to be generally independent, since they will both be predictive of the label.

Most existing methods for enforcing conditional independence requires penalizing some dependency measure between the two models. Here, we propose an alternative approach that takes advantage of the fact the models are being trained in an ensemble.

Theoretical Motivation:: Assume $f_h(x_i, \theta_h)$ can be decomposed as $c_h(g_h(x_i, \theta_h^g), \theta_h^c)$ where $g_h(\cdot, \theta_h^g) : \mathcal{X} \to \mathbb{R}^z$ produces a feature vector and $c_h(\cdot, \theta_h^c) : \mathbb{R}^z \to \mathcal{B}_y$ is a classifier that uses those features. Likewise, assume f_l can be decomposed into g_l and c_l.

Let x be an example from $P^{train}(X, Y)$. We write $x_l = g_l(x)$ and $x_h = g_h(x)$ for notational simplicity. Our method is based on the observation that if x_h and x_l are conditionally independent:

$$
\begin{aligned}
P(y|x_h, x_l) & \qquad (5.5)\\
&\propto P(x_h, x_l|y)P(y) & (5.6)\\
&= P(x_h|y)P(x_l|y)P(y) & (5.7)\\
&= \frac{P(y|x_h)P(x_h)}{P(y)} \frac{P(y|x_l)P(x_l)}{P(y)} P(y) & (5.8)\\
&\propto \frac{P(y|x_h)}{P(y)} \frac{P(y|x_l)}{P(y)} P(y) & (5.9)
\end{aligned}
$$

Where (3) applies Bayes rule, (4) follows from conditional independence, and (5) applies Bayes rule twice.

In words, if x_h and x_l are conditionally independent, then $P(y|x_h, x_l)$ can be computed by computing $P(y|x_h)$ and $P(y|x_l)$ and then combining them as a product-of-experts following equation 6.

Our key idea is to constrain the model so that $P(y|x_h, x_l)$ must be modelled following equation 6, i.e., by modelling $P(y|x_l)$ and $P(y|x_h)$ individually and then combining them as shown. Since equation 6 will only be equal to $P(y|x_h, x_l)$ if x_h and x_l are conditionally independent, this creates a natural incentive for the model to construct conditionally independent feature sets during learning.

Intuitively, this works because if a particular piece of information relevant to predicting y is present in both x_h and x_l, then that information's relationship to y will be captured in both $P(y|x_h)$ and $P(y|x_l)$, and thus will be double-counted when these distributions are combined. Optimization should therefore remove that information from either x_h and x_l to prevent this double counting.

Method: We approximate this constraint by optimizing the classifiers c_h and c_l independently from the other parameters. That is, we compute:

$$\theta_h^{c*} = \arg\min_{\theta_h^c} \sum_{i=1}^{n} L(\hat{y}_i^h, y_i)$$

and then define f_h as $f_h(x_i) = c_h(g_h(x_i, \theta_h^g), \theta_h^{c*})$. Since θ_h^{c*} has been independently optimized, we have $\hat{y}_i^h \approx P(y_i|g_h(x_i))$, in which case $f_h(x_i) \approx P(y|g_h(x_i))/P(y)$ (because we model the class prior separately from f_h when computing $\hat{y}_i^h$). We do the same for $f_l(x_i)$. Therefore the ensemble prediction $\hat{y}_i^e$ is being computed approximately according to equation 6.

This method does not require adding additional hyperparameters, but it does mean optimizing the loss specified in equation 5.4 becomes a bi-level optimization problem Colson et al. [2007] because it contains the subproblem of computing θ_h^{c*}.

5.2.5 Optimization

To make optimization easier, we define c_h and c_l to be residual affine functions so that $c_h(g_h(x_i), \theta_h^c) = g_h(x_i)W_h^c + b_h^c + g_h(x_i)$. In practice, we use the top-level classifier of the model being trained as g_h (after converting the output to log probabilities), and treat c_h as a post-processing step that refines its prediction. We minimize the loss (equation 5.4) using minibatch gradient descent.

Computing Gradients: During the forward pass we directly compute θ_h^{c*} and θ_l^{c*} for the minibatch, which essentially requires optimizing a small logistic regression model, and can be done quickly using existing black-box solvers and warm-starting from previous solutions. Once computed, the gradient of the input features with respect to θ_h^{c*} has a closed-form solutions as shown in Gould et al. [2016], allowing us to backpropagate gradients through θ_h^{c*} and θ_l^{c*} to g_h and g_l.

Addressing Discontinuities: A complication when using gradient descent is that the argmin operations are not smooth: a small change in $g_h(x_i)$ could dramatically change θ_h^{c*} and therefore f_h. To solve this, we add a regularization penalty to both argmin functions so that:

$$\theta_h^{c*} = \arg\min_{\theta_h^c} \sum_{i=1}^{n} L(\hat{y}_i^h, y_i) + \alpha\Omega(W_h^c)$$

Where $\Omega(W_h^c)$ is the L2 norm of W_h^c. This ensures the optimization problem is well-defined, and helps avoid discontinuities during training. α can be tuned by shrinking it as much as possible as long as learning remains smooth on the training data. We generally find that our method remains effective across a range of values for α, so we simply fix $\alpha = 0.002$ for all experiments in this chapter.

Applying the classifiers residually ensures c_h can still only improve the predictions made by g_h, since at worst the weights can be set to zero.

Selecting Minibatches: To ensure the θ_h^{c*} and θ_l^{c*} computed on each minibatch are good approximations of their value if computed on the entire training corpus, we train with large minibatches (at least 256 examples) and stratify examples so each class is approximately equally represented in each batch.

Evaluation: Computing θ_h^{c*} requires using the example's labels, which are not available at test time. Therefore we use a θ_h^{c*} computed on a large sample of the training data when classifying unlabelled examples.

5.2.6 Answer Candidate Pruning

One risk with this approach is that, while simplistic methods should not be relied upon to entirely solve the task, in some cases they could plausibly be used to eliminate some answer options. For example, given a "How many" question in VQA, it can be determined the answer should be a number without looking at the image or the other words. However, our method might factor such simple heuristics into the lower capacity model.

For tasks where this is a possibility, we propose to extract this answer-elimination ability from the lower capacity model through thresholding. At test time we eliminate all answer candidates that the lower capacity model assigns a probability less than some (very conservative) threshold t, and select among the remaining candidates using the higher capacity model. This follows the intuition that the simplistic patterns captured by the lower capacity model might still be expected to eliminate very obviously wrong answers in a generalizable way.

5.3 Experimental Setup

We apply our method to datasets that have known biases, and evaluate the trained model on adversarial datasets that were constructed to penalize models that make use of those biases. This is not a perfect evaluation because models might be unfairly penalized for ignoring biases

that were unaccounted for when the adversarial dataset was constructed, and therefore still exist in the test set. However, positive results on these cases provide good evidence that our method was at least able to identify the bias the adversarial dataset targeted.

All reported results are averages of eight runs unless otherwise specified. We select standard, high-performing models for the higher capacity models. To prove that we do not require the lower capacity model to be naturally prone to modelling the target dataset bias, we select lower capacity models that can achieve high scores when trained alone.

For each dataset, we evaluate on an out-of-domain (OOD) and in-domain (ID) test set. We provide overviews of the datasets and models used alongside the results, and present additional training details in section (TODO).

5.3.1 Applying Prior Work

We attempted to apply the methods from the previous chapter by having the lower capacity model fill the role of a bias-only model. Doing this entails training the lower capacity model on its own, freezing its parameters, and then training the higher capacity model in an ensemble with the pre-trained lower capacity model. However, we found this always leads to poor performance, sometimes to levels close to random chance. The likely cause is that the lower capacity models we use are strong enough to learn far more than just the bias when trained on their own. Therefore those methods, which prevent the higher capacity model from using strategies the lower capacity model learned, leads to degenerate behavior.

5.3.2 Comparisons

Instead, we compare MCE with two ablations, a baseline approach to conditional independence using an adversarial method, and an upper bound that uses domain-specific knowledge of the bias.

No BP: We train the ensemble without backpropagating through the argmin operators that compute the parameters of the top-level classifiers, c_l and c_h. Instead the parameters are

treated as constants. This tests whether it is necessary to optimize the model using bi-level optimization methods, or if a more ad-hoc approach would have been sufficient.

No CI: The ensemble is trained without our approach to enforcing conditional independence.

With Adversary: The ensemble is trained while replacing our conditional independence method with the 'Equality of Odds' adversarial approach from Zhang et al. [2018a]. This approach trains the output of each model to be difficult to predict given the label and the output of the other model.

Pretrained Bias: Following Clark et al. [2019b], we construct a bias-only model by training a model that can only use the target bias (e.g., a hypothesis-only model for MNLI). The high capacity model is then trained in an ensemble with that pre-trained model. This method makes use of precise knowledge about the target bias, so we use it as an approximate upper bound on what could be achieved if the target bias was perfectly factored into the lower capacity model. We give details in chapter (TODO)

5.3.3 Hyperparameters

We use OOD development sets drawn from the same distribution as the OOD test sets for hyper-parameter tuning. As mentioned in the previous chapter, this is not an ideal solution since it means some information about the OOD distribution is used during training. As a best-effort attempt to mitigate this issue, we use a single hyperparameter setting for MCE ($w = 0.2$) on all our experiments. Although setting this parameter did require using examples from the OOD dev sets, the fact a single value works well on a diverse set of tasks suggests that our method will generalize to settings where tuning hyperparameters on the OOD test distribution is not possible.

The With Adversary baseline also requires hyperparameter tuning. We were unable to find a universally effective setting, so we report results using the best setting found on the

Method	Background		Patch		Split	
	OOD Acc	ID Acc	OOD Acc	ID Acc	OOD Acc	ID Acc
MCE (Ours)	81.21	93.92	74.34	86.70	92.36	93.69
No CI	78.75	94.95	69.40	85.53	91.29	93.94
No BP	78.39	94.34	71.93	86.67	92.35	93.78
With Adversary*	80.57	93.37	70.31	82.78	91.24	93.43
None	67.76	95.46	58.53	90.34	88.77	94.72
Pretrained Bias	84.59	91.17	72.96	79.72	91.27	91.61

Table 5.1: Accuracy on MNIST with synthetic biases when those biases are removed (OOD) or preserved (ID).

OOD dev set. This is indicated by a * next to that method.

5.4 Results

5.4.1 Synthetic MNIST

We build synthetic datasets by adding a synthetic bias to the MNIST images [LeCun et al., 1998]. In particular, we design a superficial image modification for each label, apply the modification that corresponds to each image's label 90% of the time, and apply a different, randomly selected modification otherwise. OOD sets are built by applying randomly selected modifications. We use 200 examples for each digit for training, 1000 examples per digit for the OOD dev set, and 1000 per digit for the OOD and ID test sets. Runs are averaged over 100 trials. We use three kinds of modifications in order to demonstrate our method is effective against multiple types of bias.

Background: The background of the digit is colored one of 10 colors depending on the label.

Patch: The background is divided into a 5x5 grid, the upper left patch is colored to match the label, and the rest of the patches are colored randomly.

Split: Following Feng et al. [2019], the label is encoded in two separate locations in the image. Each label is mapped to a set of color pairs. Then the image is divided into vertical stripes, the center strip is colored randomly, and the other strips are colored using a randomly selected color pair for the example label. To avoid an excessive number of color pairs, we only use the digits 1-8, and map those digits to four super classes.

Higher Capacity Model: We use a model with one convolution layer with 8 7x7 filters and ReLU activation, followed by a 128 dimensional ReLU layer and a softmax predictor layer.

Lower Capacity Model: We use a model with a 128 dimensional ReLU layer then a softmax layer.

Results: Table 5.1 shows the results. The Patch bias proves to be the hardest, possibly because the patchwork background distracts the model, while the more subtle Split bias is the easiest. Despite using no knowledge of the particular bias being used, our method improves upon training the model naively by at least four points, in two cases slightly outperforming the Pretrained Bias method. Using the adversary is comparable in some cases, but falls behind on the Patches bias.

5.4.2 VQA

We evaluate on the VQA-CP v2 dataset [Agrawal et al., 2018], which was constructed by re-splitting VQA 2.0 [Goyal et al., 2018] data into new train and test sets such that the correlations between question types and answers differs between each split. For example, "white" is the most common answer for questions that start with "What color is..." in the

train set, whereas "black" is the most common answer for those questions in the test set. We hold out 40k examples from the train set to serve as an ID test set, and 40k of the 200k examples in the test set for an OOD dev set.[1]

Our model follows standard practice of predicting an answer from a set of pre-selected answer candidates. Since many of the answers are uncommon, and thus will be poorly represented in individual mini-batches, we cannot apply our conditional independence method out of the box. Instead, we cluster answer candidates by putting the 10 most common answers in individual clusters and the rest in an 11th cluster, and then apply our method while sharing parameters between answers in the same cluster. Since the model uses a sigmoid prediction function, we make g a simple two-parameter function that rescales and shifts the input.

For this dataset, we additionally show results when using the answer candidate pruning method from Section 5.2.6 for models where it is applicable. We pick a conservative threshold such that the correct label would be pruned less than 0.1% of the time on in-domain data.

Higher Capacity Model: We use LXMERT [Tan and Bansal, 2019], a transformer based model that has been pretrained on image-caption data.

Lower Capacity Model: We make predictions using mid-level representations from the higher capacity model (see the TODO for details).

Results: Table 5.2 shows the results. MCE closes most of the gap between the basic model and the upper bound, suggesting it was able to identify the question-type bias. The baselines underperform MCE by a significant margin, while answer candidate pruning offers a consistent boost.

---

[1]Prior work tuned hyper-parameters directly on the test set, but we think its preferable to shrink the test set slightly in order to have a separate OOD dev set

Method	OOD Acc		ID Acc	
	Acc	w/o AP	Acc	w/o AP
MCE (Ours)	68.44	66.10	74.03	72.72
No CI	59.10	37.32	67.56	49.28
No BP	61.64	47.55	67.99	55.54
With Adversary*	66.08	26.16	72.17	37.47
None	57.65	-	76.95	-
Pretrained Bias	70.32	-	70.78	-

Table 5.2: Results on the VQA-CP OOD test set, and a held out ID test set. We show accuracy with and without answer pruning. Note the ID set is for VQA-CP, and is not comparable to the standard VQA 2.0 dev set.

5.4.3 MNLI

We evaluate on the MNLI Hard sentence pair classification dataset from Gururangan et al. [2018] and the HANS dataset from McCoy et al. [2019b].

The MNLI Hard dataset is built by training a classifier to predict the target class using only the hypothesis sentence, and then filtering out all examples from the dev set that this classifier is able to classify correctly. Our classifier reaches 54% accuracy on the matched dev set (compared to 33% from random guessing) by making use of correlations between certain words and the class (e.g., "not" usually implies the class is "contradiction"). We use 4.4k examples filtered from the MNLI matched dev set as an OOD dev set, and another 4.4k examples filtered from the mismatched dev set as the OOD test set. We use the entire 9.8k mismatched dev set for the ID test set.

The HANS dataset contains 30k examples where both sentences contain similar words, so models that naively classify such sentences as 'entailment' will perform poorly. We do not tune hyper-parameters on HANS, and instead use the same settings as MNLI Hard.

We evaluate each model on both adversarial sets, so doing well requires models to be

simultaneously robust to both the biases being tested for. We build a Pretrained Bias model for hypothesis-only bias, and report the best ensemble result from Clark et al. [2019b] as an upper bound for the HANS dataset.

Higher Capacity Model: We use the pre-trained uncased BERT-Base model [Devlin et al., 2019].

Lower Capacity Model: We use a modified version of the neural bag-of-words model from Parikh et al. [2016][2].

Results: Table 5.3 shows the results. BERT-Base is already reasonably robust to the hypothesis-only bias, with only a 4% gap between the upper bound and the unmodified model, but is more vulnerable to the word-overlap bias in HANS. Our method is able to almost cut the gap in half, with the adversarial approach to conditional independence slightly underperforming MCE.

5.4.4 Imagenet Animals

We construct a dataset from Imagenet [Deng et al., 2009a] to test our methods on background-class correlations. Since training models on Imagenet is computationally expensive, and the large number of classes creates complications when applying our conditional independence approach (i.e., we would have take steps to prevent classes becoming too sparsely represented in each minibatch, as we did for VQA), we build a simplified animal classification dataset by grouping various animal classes into 6 super-classes.

In more detail, we take advantage of the hierarchical structure of Imagenet and its correspondence to wordnet [Fellbaum, 2012]. We select 6 wordnet synsets that have at least 20 hyponyms that exist in Imagenet: fish.n.01, insect.n.01, dog.n.01, bird.n.01, ungulate.n.01,

[2]We were unable to get positive results using layer 3 or 6 of the BERT model, possibly because even the lower layers of BERT have a lot of representational power

Method	Hard	HANS	ID
MCE (Ours)	77.58	64.43	83.28
No CI	77.24	64.25	83.38
No BP	76.44	62.18	83.88
With Adversary*	77.10	63.03	83.35
None	75.62	61.04	84.28
Pretrained Bias	79.78	62.23	76.90
Clark et al. [2019b]	-	67.92	-

Table 5.3: Accuracy on two OOD textual entailment datasets (MNLI Hard and HANS) and the mismatched dev set performance (ID) .

Method	OOD Acc			ID Acc
	5%	10%	20%	All
MCE (Ours)	79.86	81.36	83.56	89.86
No CI	81.02	82.46	84.51	89.97
With Adversary*	80.88	82.36	84.28	89.95
None	78.20	80.28	83.17	90.79
Pretrained Bias	87.71	87.54	86.93	78.38

Table 5.4: Results on Imagenet animal recognition on examples where less than 5%, 10%, or 20% of the image-patch classifiers are accurate (OOD), as well as on the entire test set (ID).

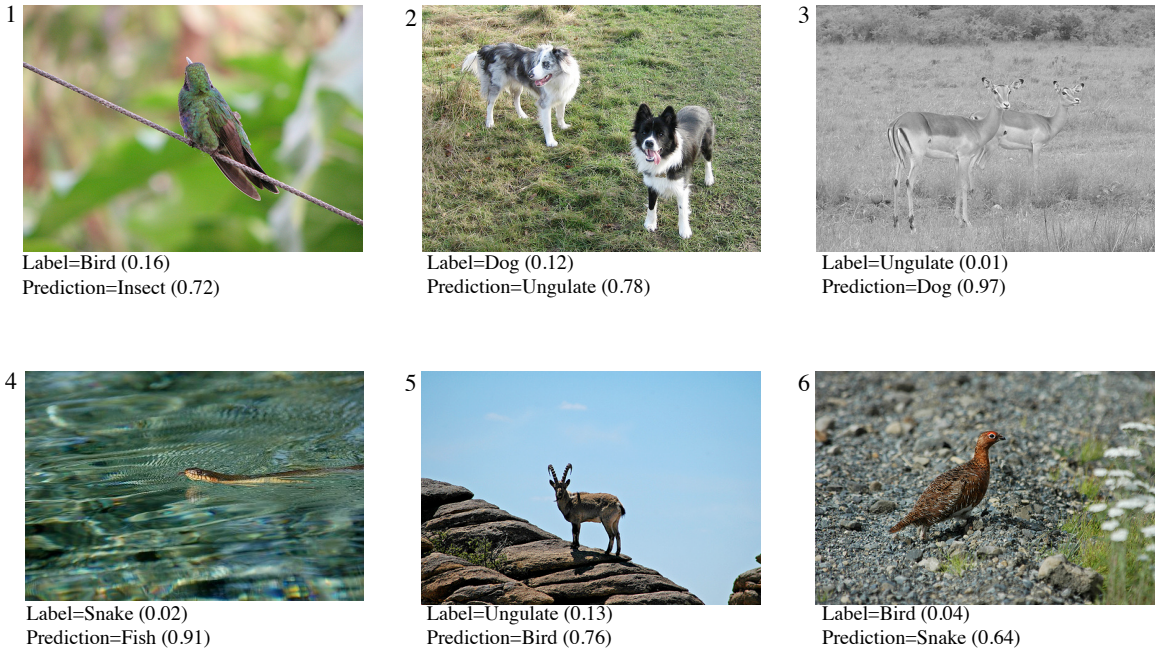

Figure 5.2: Qualitative examples from Imagenet Animals where most of the image-patch classifiers were incorrect. We show images paired with the gold label and the most common prediction made by the image-patch classifiers, with the percent of image-patch classifiers that predicted those labels in parentheses. Errors are often caused by the patch classifiers associating features of the background with the class. In particular, because (1) twigs and leaves are associated with insects, (2) grassy fields are associated with ungulates (e.i., hooved mammals), (3) black and white photos are associated with dogs due to the commonality of black-and-white dog photos in the training data, (4) water is associated with fish, (5) open sky is associated with birds, and (6) close-ups of the ground are associated with snakes.

and snake.n.01. Each synset is used as a class. We gather examples for each class by sampling images randomly from all its hyponyms in the Imagenet training data.

We build a train set with 10k images per class, a dev set with 3k images per class, and a test set with 7k images per class. Similarly to our approach to MNLI, and Hendrycks et al. [2019], we construct OOD datasets by filtering out correct predictions made by a biased classifier. Following the approach of BagNet [Brendel and Bethge, 2019], we modify a ResNet-18 model [He et al., 2016] to build features for 9x9 image patches and then train a classifier to predict the class using those features. Images where most of the classifiers were unable to guess the correct class are assumed to have a misleading background and are used for the OOD test set. Figure 5.2 contains some examples, and demonstrates the kinds of errors the image-patch classifiers make.

Higher Capacity Model: We use ResNet-18 [He et al., 2016].

Lower Capacity Model: We branch the higher capacity model by adding a 1x1 convolution with 256 filters after the first residual block, followed by max-pooling and a 256 dimensional ReLU layer.[3] This model gets around 81% on our dev set if trained alone.

Results: Table 5.4 shows the results. This dataset proves to be challenging; MCE provides a boost over naively training the model, but is still significantly below our estimated upper bound. Our conditional independence method reduces performance here, and training fails to converge for the no-backpropagation ablation, so results for that method are not shown. A possible cause is that the model is able to reach nearly 100% accuracy on the training data which causes the argmin operations to become degenerate.

[3]We found positive, but slightly weaker results using features after the second residual block, and negative results when using features after the third residual block

5.4.5 Discussion

Overall, we are able to improve out-of-domain performance in all settings even though our method does not use dataset-specific information about the target bias. Our conditional independence method generally improved performance while the adversarial baseline, despite getting the benefit of per-dataset hyperparameter tuning, was less effective.

MCE decreases ID performance, which is expected since the bias is helpful on in-domain data and the goal of our method is to remove it. However the decrease is often much less than for the upper bound (e.g., on MNLI MCE is 2 points behind on the OOD test set, but 6 points ahead on the ID test set). A possible cause is that the bias is only being partially factored out. Improving OOD performance without losing much ID performance might also suggest MCE is helping improve the higher capacity model in general. Better understanding and making use of this phenomenon is an interesting avenue for future work.

We find computing the argmin operations adds a moderate computational overhead, requiring about 2x the train time for the ResNet-18 ensemble and about 1.3x the time for the larger BERT and LXMERT ensembles (performance during evaluation is identical). Our implementation performs the optimization on the CPU; a GPU based optimizer might reduce this overhead.

5.5 Conclusion

We have presented a method for improving out-of-domain performance by detecting and avoiding overly simple patterns in the training data. Experiments show this approach successfully prevents the higher capacity model from adopting known biases on several real-world datasets, including achieving a 10-point gain on an out-of-domain VQA dataset.

Chapter 6

CONCLUSION

In this thesis, we have introduced new methods of preventing models from exploiting superficial patterns that happen to work well on the training data, but do not solve the fundamental task.

In chapter 3, we introduced a method of training less biased question answering models in order to improve results in the multi-paragraph setting. Our approach trains the model by using it to process a small sample of paragraphs in parallel, and then computing answer probabilities across all those paragraphs. This prevents the model from learning biases caused when the model is only trained on one paragraph at a time (like over-confidence when only a single entity in the context passage matches the expected answer-type of the question). We showed the resulting unbiased models are much more effective when used to select an answer from a large number of paragraphs, and therefore at open-domain question answering.

In chapter 4, we proposed an ensemble-based method to prevent models from learning known biases. Our method trains a bias-only model that captures the bias, then trains a robust model in an ensemble with the bias-only model in order to prevent it from learning the approach the bias-only model uses. We considered several ensembling methods, the most successful one built on a product-of-expert style ensemble, but gave some additional flexibility to the model by allowing it to control the extent to which the bias-only model was used for each example. We showed this method is effective across a wide range of tasks and biases

In chapter 5, we extended this method to the case where the bias is unknown. To do this, we proposed that biases are often simpler, meaning learnable by less parameter-heavy models, than unbaised methods for language and visual understanding tasks. We exploited

this by training an ensemble as before, but dynamically training the bias-only model to capture simple approaches to solving the problem, while allowing more complex approaches to be captured by the robust model. We showed this approach can be nearly as effective as the previous approach even though it does not make use of expert knowledge of the bias.

Altogether, we have shown bias in datasets need not lead to bias in models. Bias is now known to be pervasive in machine learning datasets, but we have made significant progress training models to ignore that bias.

6.1 Future Work

6.1.1 Debasing for Data Efficiency

A downside to the ensemble-based methods we propose here is that they reduce the need for the model to solve examples the bias-only model (or low-capacity model) already works well on. The ensemble will already be mostly correct for those examples, which means the robust model has less incentive to learn them. The reweighting baseline we used in Chapter 4, and many of the alternative debiasing methods we discuss in Chapter 2, have a similar effect where examples solved by the bias are effectively down-weighted. This serves to reduce the bias, but also effectively reduces the amount of training data since many examples have a low weight.

As a result, these methods are not quite as effective as training models on unbiased data. The synthetic experiments in Section 4.4.1 are a good illustration of this: when we have a bias-only model that perfectly captures a simple bias our methods result in a model that is close, but not quite as good, as a model trained on unbiased data.

Solving this is quite challenging. Given a biased example, we need to receive a strong gradient towards solving that example using the unbiased approach (so the example is not effectively down-weighted), but also receive a small gradient for the biased approach (so the bias will not be learned). This goal cannot be achieved by simply changing the loss function, which could only increase/decrease the gradient for both methods at the same time.

Instead, it would likely require identifying which components of the model are responsible for capturing the bias, but this process itself might be difficult or noisy.

Despite the challenges, this is an interesting line of future work because it opens the possibility of using knowledge of biases to build models that learn to use the unbiased approach more effectively than would be possible with standard training methods. This would result in better unbiased models, and could even be combined with the bias to get strong in-domain performance. This idea is related to ideas from co-training [Blum and Mitchell, 1998] or multi-view learning [Xu et al., 2013] that exploit multiple views of the data to increase performance.

6.1.2 Multiple Biases

Many datasets are likely to exhibit multiple biases. For example, question answering datasets can have both sentence-position biases [Ko et al., 2020] and question key-word biases [Jia and Liang, 2017], and MNLI datasets are known to have both hypothesis-only biases [Gururangan et al., 2018] and world-overlap biases [McCoy et al., 2019b]. In this case, it is natural to ask how our methods can extend to these settings.

For our ensemble-based methods, we could add additional bias-only models into the ensemble. Alternately, we could train a bias-only model that uses multiple biases. The latter approach, however, might have some disadvantages when used with the Mixin methods (Section 4.2) since the model will not be able to control each individual bias. Using an ensemble of models with the mixin method might therefore be more effective.

Our method for dealing unknown biases presented in Chapter 4 should factor *all* biases into the low-capacity model. For example, on MNLI we were able to show our model become more robust to multiple biases (see Section 5.4.3). Factoring that model into multiple models that capture different biases, possibly be leveraging a conditional independence assumption as before, might be have some advantages from an interpretability perspective.

6.1.3 Bias and Explanations

Faithful explanation, i.e., explanations that accurately reflect the processes of the model, will need to either (1) use an unbiased model or (2) surface how bias is being used to users. For example, a faithful VQA explanation will either need to avoid using question-type/answer correlations, or inform the user those correlations are being used. Considering bias is therefore important with dealing with explanations. Our ensemble methods are particularly suited to this task because they explicitly model the bias. Therefore, it would be easy to surface to users the role bias play in the final classification decision.

This can also play a role in bias-discovery since our work from Chapter 5 creates a model that learns the bias. This means users could potentially learn about the bias in a dataset by examining the behavior of that model. However, those models are still black-box, so completing this step will require some additional interpretability effort.

BIBLIOGRAPHY

Aishwarya Agrawal, Dhruv Batra, Devi Parikh, and Aniruddha Kembhavi. Don't Just Assume; Look and Answer: Overcoming Priors for Visual Question Answering. In *CVPR*, 2018.

Ankesh Anand, Eugene Belilovsky, Kyle Kastner, Hugo Larochelle, and Aaron Courville. Blindfold Baselines for Embodied QA. In *ViGIL Workshop at NeurIPS*, 2018.

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering. In *CVPR*, 2018a.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-Language Navigation: Interpreting Visually-Grounded Navigation Instructions in Real Environments. In *CVPR*, 2018b.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant Risk Minimization. *arXiv preprint arXiv:1907.02893*, 2019. version 3.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*, 2015.

Hyojin Bahng, Sanghyuk Chun, Sangdoo Yun, Jaegul Choo, and Seong Joon Oh. Learning De-biased Representations with Biased Representations. In *ICML*, 2019.

Petr Baudiš. Yodaqa: a modular question answering system pipeline. In *POSTER International Student Conference on Electrical Engineering*, 2015.

Yonatan Belinkov, Adam Poliak, Stuart M Shieber, Benjamin Van Durme, and Alexander M Rush. On Adversarial Removal of Hypothesis-only Bias in Natural Language Inference. In *StarSem*, 2019.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic Parsing on Freebase from Question-Answer Pairs. In *EMNLP*, 2013.

Alex Beutel, Jilin Chen, Zhe Zhao, and Ed H Chi. Data decisions and theoretical implications when adversarially learning fair representations. *arXiv preprint arXiv:1707.00075*, 2017. version 2.

Avrim Blum and Tom Mitchell. Combining Labeled and Unlabeled Data with Co-Training. In *COLT*, 1998.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain Separation Networks. In *NIPS*, 2016.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A Large Annotated Corpus for Learning Natural Language Inference. In *EMNLP*, 2015a.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A Large Annotated Corpus for Learning Natural Language Inference. In *ACL*, 2015b.

Wieland Brendel and Matthias Bethge. Approximating CNNs with Bag-of-local-Features models works surprisingly well on ImageNet. *ICLR*, 2019.

Remi Cadene, Corentin Dancette, Matthieu Cord, Devi Parikh, et al. RUBi: Reducing Unimodal Biases for Visual Question Answering. In *NIPS*, 2019.

Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. Clueweb09 data set, 2009.

Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain Generalization by Solving Jigsaw Puzzles. In *CVPR*, 2019.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. In *ACL*, 2017a.

Jifan Chen and Greg Durrett. Understanding Dataset Design Choices for Multi-hop Reasoning. In *NAACL*, 2019.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. Enhanced LSTM for Natural Language Inference. In *ACL*, 2017b.

Jianpeng Cheng, Li Dong, and Mirella Lapata. Long Short-Term Memory-Networks for Machine Reading. In *EMNLP*, 2016.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *EMNLP*, 2014.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. QuAC: Question Answering in Context. In *EMNLP*, 2018.

Christopher Clark and Matt Gardner. Simple and Effective Multi-Paragraph Reading Comprehension. In *ACL*, 2017.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the Surprising Difficulty of Natural Yes/No Questions. In *NAACL*, 2019a.

Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. Don't Take the Easy Way Out: Ensemble Based Methods for Avoiding Known Dataset Biases. In *EMNLP*, 2019b.

Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Turney, and Daniel Khashabi. Combining Retrieval, Statistics, and Inference to Answer Elementary Science Questions. In *AAAI*, 2016.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge. *Computing Research Repository*, arXiv preprint arXiv:1803.05457, 2018. version 1.

Benoît Colson, Patrice Marcotte, and Gilles Savard. An Overview of Bilevel Optimization. *Annals of Operations Research*, 2007.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Embodied Question Answering. In *CVPR*, 2018.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009a.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A Large-Scale Hierarchical Image Database. In *CVPR*. IEEE, 2009b.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*, 2019.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. Quasar: Datasets for Question Answering by Search and Reading. *arXiv preprint arXiv:1707.03904*, 2017.

Zhengming Ding and Yun Fu. Deep Domain Generalization with Structured Low-Rank Constraint. *IEEE Transactions on Image Processing*, 2017.

Harrison Edwards and Amos Storkey. Censoring Representations with an Adversary. In *ICLR*, 2016.

Yanai Elazar and Yoav Goldberg. Adversarial Removal of Demographic Attributes from Text Data. In *EMNLP*, 2018.

Christiane Fellbaum. WordNet. *The Encyclopedia of Applied Linguistics*, 2012.

Shi Feng, Eric Wallace, and Jordan Boyd-Graber. Misleading Failures of Partial-input Baselines. In *ACL*, 2019.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In *ACL*, 2005.

Yarin Gal and Zoubin Ghahramani. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *NIPS*, 2016.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-Adversarial Training of Neural Networks. *The Journal of Machine Learning Research*, 2016.

Matt Gardner, Yoav Artzi, Victoria Basmova, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hanna Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. Evaluating NLP Models via Contrast Sets. *arXiv preprint arXiv:2004.02709*, 2020. version 1.

Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. ImageNet-Trained CNNs are Biased Towards Texture; Increasing Shape Bias Improves Accuracy and Robustness. In *ICLR*, 2019.

Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain Generalization for Object Recognition with Multi-Task Autoencoders. In *ICCV*, 2015.

Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. IQA: Visual Question Answering in Interactive Environments. In *CVPR*, 2018.

Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. On Differentiating Parameterized Argmin and Argmax Problems with Application to Bi-level Optimization. *arXiv preprint arXiv:1607.05447*, 2016.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering. *IJCV*, 2018.

Gabriel Grand and Yonatan Belinkov. Adversarial Regularization for Visual Question Answering: Strengths, Shortcomings, and Side Effects. In *Proceedings of the Second Workshop on Shortcomings in Vision and Language*, 2019.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. Annotation Artifacts in Natural Language Inference Data. In *NAACL*, 2018.

Moritz Hardt, Eric Price, and Nati Srebro. Equality of Opportunity in Supervised Learning. In *NIPS*, 2016.

He He, Sheng Zha, and Haohan Wang. Unlearn Dataset Bias in Natural Language Inference by Fitting the Residual. In *EMNLP Workshop on DeepLo*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural Adversarial Examples. *arXiv preprint arXiv:1907.07174*, 2019. version 3.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching Machines to Read and Comprehend. In *NIPS*, 2015.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. WikiReading: A novel large-scale language understanding task over Wikipedia. In *ACL*, 2016.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations. In *ICLR*, 2016.

Geoffrey E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 2002.

Minghao Hu, Yuxing Peng, Xipeng Qiu, Furu Wei, and Ming Zhou. Reinforced Mnemonic Reader for Machine Reading Comprehension. In *IJCAI*, 2018.

Allan Jabri, Armand Joulin, and Laurens Van Der Maaten. Revisiting Visual Question Answering Baselines. In *ECCV*, 2016.

Robin Jia and Percy Liang. Adversarial Examples for Evaluating Reading Comprehension Systems. In *EMNLP*, 2017.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *ACL*, 2017.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text Understanding with the Attention Sum Reader Network. In *ACL*, 2016.

Rabeeh Karimi Mahabadi, Yonatan Belinkov, and James Henderson. End-to-End Bias Mitigation by Modelling Biases in Corpora. In *ACL*, 2020.

Divyansh Kaushik, Eduard Hovy, and Zachary C Lipton. Learning the Difference that Makes a Difference with Counterfactually-Augmented Data. *arXiv preprint arXiv:1909.12434*, 2019. version 2.

Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In *ICLR*, 2014.

Miyoung Ko, Jinhyuk Lee, Hyunjae Kim, Gangwoo Kim, and Jaewoo Kang. Look at the First Sentence: Position Bias in Question Answering. *arXiv preprint arXiv:2004.14602*, 2020.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural Questions: a Benchmark for Question Answering Research. In *ACL*, 2019.

Ronan Le Bras, Swabha Swayamdipta, Chandra Bhagavatula, Rowan Zellers, Matthew E Peters, Ashish Sabharwal, and Yejin Choi. Adversarial Filters of Dataset Biases. In *ICML*, 2020.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11), 1998.

Mike Lewis and Angela Fan. Generative Question Answering: Learning to Answer the Whole Question. In *ICLR*, 2018.

Haoliang Li, Sinno Jialin Pan, Shiqi Wang, and Alex C Kot. Domain Generalization with Adversarial Feature Learning. In *CVPR*, 2018a.

Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep Domain Generalization via Conditional Invariant Adversarial Networks. In *ECCV*, 2018b.

Yi Li and Nuno Vasconcelos. REPAIR: Removing Representation Bias by Dataset Resampling. In *CVPR*, 2019.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. In *ICCV*, 2017.

R Thomas McCoy, Junghyun Min, and Tal Linzen. BERTs of a feather do not generalize together: Large variability in generalization across models with similar test set performance. *arXiv preprint arXiv:1911.02969*, 2019a. version 1.

Tom McCoy, Ellie Pavlick, and Tal Linzen. Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference. In *ACL*, 2019b.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in Pre-Training Distributed Word Representations. In *LREC*, 2018.

Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. Question Answering Through Transfer Learning from Large Fine-Grained Supervision Data. In *ACL short paper*, 2017.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. Compositional Questions Do Not Necessitate Multi-hop Reasoning. In *ACL*, 2019.

Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain Generalization via Invariant Feature Representation. In *ICML*, 2013.

Aakanksha Naik, Abhilasha Ravichander, Norman Sadeh, Carolyn Rose, and Graham Neubig. Stress Test Evaluation for Natural Language Inference. In *COLING*, 2018.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. 2016. URL `https://www.microsoft.com/en-us/research/publication/ms-marco-human-generated-machine-`

Boyuan Pan, Hao Li, Zhou Zhao, Bin Cao, Deng Cai, and Xiaofei He. MEMEN: Multi-layer Embedding with Memory Networks for Machine Comprehension. *arXiv preprint arXiv:1707.09098*, 2017. version 1.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A Decomposable Attention Model for Natural Language Inference. In *EMNLP*, 2016.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global Vectors for Word Representation. In *EMNLP*, 2014.

Jean Ponce, Tamara L Berg, Mark Everingham, David A Forsyth, Martial Hebert, Svetlana Lazebnik, Marcin Marszalek, Cordelia Schmid, Bryan C Russell, Antonio Torralba, et al. Toward category-level object recognition. In *Toward category-level object recognition*. Springer, 2006.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *EMNLP*, 2016.

Sainandan Ramakrishnan, Aishwarya Agrawal, and Stefan Lee. Overcoming Language Priors in Visual Question Answering with Adversarial Regularization. In *NeurIPS*, 2018.

Siva Reddy, Danqi Chen, and Christopher D. Manning. CoQA: A Conversational Question Answering Challenge. In *TACL*, 2019.

Tal Schuster, Darsh J Shah, Yun Jie Serene Yeo, Daniel Filizzola, Enrico Santus, and Regina Barzilay. Towards Debiasing Fact Verification Models. In *EMNLP*, 2019.

Roy Schwartz, Maarten Sap, Ioannis Konstas, Li Zilles, Yejin Choi, and Noah A Smith. The Effect of Different Writing Tasks on Linguistic Style: A Case Study of the ROC Story Cloze Task. In *CoNLL*, 2017.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional Attention Flow for Machine Comprehension. In *ICLR*, 2017a.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. BiDirectional Attention Flow for Machine Comprehension. In *ICLR*, 2017b.

Andrew Smith, Trevor Cohn, and Miles Osborne. Logarithmic Opinion Pools for Conditional Random Fields. In *ACL*, 2005.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway Networks. In *Deep Learning Workshop (ICML)*, 2015.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *WWW*, 2007.

Hao Tan and Mohit Bansal. LXMERT: Learning Cross-Modality Encoder Representations from Transformers. In *EMNLP-IJCNLP*, 2019.

Jesse Thomason, Daniel Gordon, and Yonatan Bisk. Shifting the Baseline: Single Modality Performance on Visual Navigation & QA. In *NAACL*, 2019.

Antonio Torralba and Alexei A Efros. Unbiased Look at Dataset Bias. In *CVPR*. IEEE, 2011.

Lifu Tu, Garima Lalwani, Spandana Gella, and He He. An Empirical Study on Robustness to Spurious Correlations using Pre-trained Language Models. *arXiv preprint arXiv:2007.06778*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *NIPS*, 2017.

Ellen M Voorhees et al. The TREC-8 Question Answering Track Report. In *TREC*, 2000.

Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning Robust Global Representations by Penalizing Local Predictive Power. In *NIPS*, 2019a.

Shuohang Wang and Jing Jiang. Machine Comprehension Using Match-LSTM and Answer Pointer. In *ICLR*, 2017.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. R3: Reinforced Reader-Ranker for Open-Domain Question Answering. In *AAAI*, 2018.

Tianlu Wang, Jieyu Zhao, Kai-Wei Chang, Mark Yatskar, and Vicente Ordonez. Balanced Datasets Are Not Enough: Estimating and Mitigating Gender Bias in Deep Image Representations. In *ICCV*, 2019b.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated Self-Matching Networks for Reading Comprehension and Question Answering. In *ACL*, 2017.

Yicheng Wang and Mohit Bansal. Robust Machine Comprehension Models via Adversarial Training. In *NAACL*, 2018.

Dirk Weissenborn, Tomáš Kočiskỳ, and Chris Dyer. Dynamic Integration of Background Knowledge in Neural NLU Systems. *arXiv preprint arXiv:1706.02596*, 2017a.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. FastQA: A Simple and Efficient Neural Architecture for Question Answering. *arXiv preprint arXiv:1703.04816*, 2017b.

Adina Williams, Nikita Nangia, and Samuel R Bowman. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *NAACL*, 2018.

Chang Xu, Dacheng Tao, and Chao Xu. A Survey on Multi-View Learning. *arXiv preprint arXiv:1304.5634*, 2013.

Zheng Xu, Wen Li, Li Niu, and Dong Xu. Exploiting Low-Rank Structure from Latent Domains for Domain Generalization. In *ECCV*, 2014.

Yadollah Yaghoobzadeh, Remi Tachet, TJ Hazen, and Alessandro Sordoni. Robust Natural Language Inference Models with Example Forgetting. *arXiv preprint arXiv:1911.03861*, 2019.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A Dataset for Diverse, Explainable Multi-hop Question Answering. In *EMNLP*, 2018.

Matthew D Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701*, 2012.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. SWAG: A Large-Scale Adversarial Dataset for Grounded Commonsense Inference. In *EMNLP*, 2018.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a Machine Really Finish Your Sentence? In *ACL*, 2019.

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating Unwanted Biases with Adversarial Learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, 2018a.

Sheng Zhang, Xiaodong Liu, Jingjing Liu, Jianfeng Gao, Kevin Duh, and Benjamin Van Durme. ReCoRD: Bridging the Gap between Human and Machine Commonsense Reading Comprehension. *arXiv preprint arXiv:1810.12885*, 2018b. version 1.

Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Men Also Like Shopping: Reducing Gender Bias Amplification using Corpus-level Constraints. In *EMNLP*, 2017.

Jieyu Zhao, Yichao Zhou, Zeyu Li, Wei Wang, and Kai-Wei Chang. Learning Gender-Neutral Word Embeddings. In *ACL*, 2018.