Distributions in Semantic Space

by

Kira A. Selby

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2024

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Graham Taylor Professor, School of Engineering, University of Guelph
Supervisor(s):	Pascal Poupart Professor, School of Computer Science, University of Waterloo
Internal Members:	Yaoliang Yu Associate Professor, School of Computer Science, University of Waterloo
	Jesse Hoey Professor, School of Computer Science, University of Waterloo
Internal-External Member:	Ali Ghodsi Professor, Dept. of Statistics & Actuarial Science, University of Waterloo

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This thesis was based on three existing works of research authored by myself, my supervisor Dr. Pascal Poupart, and several co-authors.

For the work 'Robust Embeddings Via Distributions' (discussed in Chapter 3), I developed the original idea, wrote the code to implement the model, and led the experiments. Yinong Wang and Ruizhe Wang assisted in running experiments and made contributions to the code. My coauthors Ahmad Rashid, Peyman Passban and Mehdi Rezagholizadeh contributed to writing the Related Works section. I wrote all other sections of the paper. Ahmad Rashid, Peyman Passban, Mehdi Rezagholizadeh and Pascal Poupart provided feedback and editing.

For the work 'Learning Functions on Multiple Sets Using Multi-Set Transformers' (discussed in Chapter 4), I conceived the idea, conducted all experiments, and wrote the paper. Ahmad Rashid, Ivan Kobyzev, Mehdi Rezagholizadeh and Pascal Poupart provided feedback and editing.

For the work 'Few-Shot Image Generation Using Conditional Set-Based GANs' (discussed in Chapter 5), I conceived of the idea, conducted all experiments and wrote the paper. Pascal Poupart provided feedback and editing.

Abstract

This thesis is an investigation of the powerful and flexible applications of analyzing empirical distributions of vectors within latent spaces. These methods have historically been applied with great success to the domain of word embeddings, leading to improvements in robustness against polysemy, unsupervised inference of hierarchical relationships between words, and even used to shatter existing benchmarks on unsupervised translation.

This work will serve to extend these existing lines of inquiry, with a focus on two key areas of further research:

- Probabilistic approaches to robustness in natural language.
- Approximating general distance functions between distributions in order to infer general hierarchical relationships between words from their distributions over contexts.

Motivated by these initial research directions, the resulting investigations will then demonstrate novel and significant contributions to a diverse range of problems across many different fields and domains - far beyond the narrow scope of word embeddings. The key contributions of this work are threefold:

1. Proposing a probabilistic, model-agnostic framework for robustness in natural language models. The proposed model improves performance on a wide range of downstream tasks compared to existing baselines.
2. Constructing a general architecture for modelling distance functions between multiple permutation invariant sets. The proposed architecture is proved to be a universal approximator for all *partially permutation-invariant* functions and outperforms all existing baselines on a number of set-based tasks, as well as approximating distance functions such as KL Divergence and Mutual Information.
3. Leveraging this architecture to define a novel, set-based approach to few-shot image generation. The proposed approach outperforms all existing image-to-image baselines without making restrictive assumptions about the structure of the training and evaluation sets that might limit its ability to generalize, making it a promising candidate for scaling to true zero-shot generation.

Acknowledgements

I would like to thank my supervisor Dr. Pascal Poupart for his patience, encouragement and insightful critique. I would also like to thank my co-authors Mehdi Rezagholizadeh, Ahmad Rashid, Ivan Kobyzev, Peyman Passban, Yinong Wang and Ruizhe Wang. Finally, I would like to thank the Vector Institute for providing the computational resources necessary to perform the experiments needed for these works.

Dedication

This thesis is dedicated to my mother and my friend Laura, both of whom have been anchors of stability for me through challenging and confusing times, and without whom this would never have been possible.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

A near-universal paradigm in machine learning is the representation of examples from various domains as vector encodings embedded in a continuous latent space. These latent encodings can take many forms. Often they are used as a form of preprocessing - converting images or text into vectors in a continuous latent space that can then be processed by a neural network model. Notable examples of this are *word embeddings* such as the widely-used "Word2Vec" vectors [Mikolov et al., 2013a], or the "WordPiece" vectors [Wu et al., 2016] used by many large language models such as BERT. But such approaches are not limited to preprocessing alone. Latent representations are leveraged by many powerful autoencoder or encoder-decoder models - compressing semantic information about sequences, images, or other complex inputs into vectors that can then be leveraged in many diverse applications.

Before the advent of so-called "large language models", encoder-decoder "sequence-to-sequence" models dominated the field of natural language. These models used an "encoder" module to encode the input sequence into a continuous representation, which was then used by the 'decoder' in order to generate an output sequence [Sutskever et al., 2014b, Vaswani et al., 2017]. In more modern examples, large pretrained transformer models such as BERT [Devlin et al., 2018a] are frequently leveraged to generate powerful latent representations which can be generalized to many different applications. An enormous swath of the modern machine learning literature relies on leveraging these pretrained representations, then finetuning small output modules in order to perform a broad range of downstream tasks. This paradigm of leveraging pretrained representations is not limited to natural language, either. In computer vision, models such as VGG [Simonyan and Zisserman, 2015], Inception [Szegedy et al., 2015], or CLIP [Radford et al., 2021] are often used to encode images as latent vectors in a similar way, which can then be leveraged for applications such as assessing model quality [Heusel et al., 2018], or measuring perceptual similarity [Zhang et al., 2018].

While latent representations are ubiquitous throughout all fields of machine learning, they are often used merely deterministically, as components in larger model pipelines. This narrow, utilitarian focus fails to leverage one of the richest sources of information encoded in these latent representations: the *distributions* of vectors within continuous latent space. As a motivating example, let us consider the Word2Vec word embedding vectors, first proposed in Mikolov et al. [2013a].

1.1 Word2Vec: A Motivating Example

At a high level, Word2Vec consists of two primary components: a set of *word vectors* $U \in \mathbb{R}^{|V| \times d}$ and *context vectors* $V \in \mathbb{R}^{|V| \times d}$. Each word w in the vocabulary W is thus encoded as two d-dimensional latent representations: one as a "center word", and one as a "context word". These vectors are then trained using a *reference corpus*, by maximizing the probability over all words in T of a given context word c occurring nearby a given center word w (for a more detailed presentation, see Section 2.1.1).

$$p(c|w;\theta) = \frac{e^{u_w \cdot v_c}}{\sum_{c' \in V} e^{u_w \cdot v'_c}} \tag{1.1}$$

1.1.1 Distributional Similarity across Languages

The key assumption that underlies Word2Vec is the *distributional hypothesis* [Harris et al., 1954, Firth, 1957]. Concisely stated, the hypothesis states that "a word is characterized by the company it keeps" - i.e. words with similar meanings occur in similar contexts. One critical observation about the distributional hypothesis is that it is not limited to a single language. Consider two languages L_1 and L_2, with vocabularies W_1 and W_2. Let us (naively) assume that each word $w \in W_1$ is a direct translation of a word $w' \in W_2$. Then, any word $w \in W_1$ that is likely to occur in some context of $C = c_i \in W_1$ must have some associated word $w' \in W_2$ that is likely to occur in context $C' = c'_i \in W_2$, where C and C' must be the image of each other under translation. As such, the distribution of word-context pairs should be universal between L_1 and L_2. Since Word2Vec vectors encode this precise distribution, there is a deep connection between the *latent distribution* of word vectors in the trained representations for L_1 and those of the representations for L_2.

In fact, it has been found by many authors [Mikolov et al., 2013b, Artetxe et al., 2016, Lample et al., 2018a] that these two latent distributions are not just similar: they are a *linear transformation* of one another. If we assume V_1 to be the trained Word2Vec vectors for L_1 and V_2 to be the same for L_2, then with very high accuracy we can state that $V_1 = WV_2$ for some linear transformation W. This observation led to a flurry of work in unsupervised translation, culminating in

the work of Lample et al. [2018b]. This paper leveraged this fundamental observation about the *distribution* of latent representations within a continuous vector space in order to shatter existing benchmarks for unsupervised translation, and even surpass the performance of many supervised models.

1.1.2 Distributional Semantics and The Distributional Inclusion Hypothesis

From the distributional hypothesis mentioned in the previous section, we may conclude that the meaning of a given word w to be fundamentally connected to and summarized by its *distribution over contexts* - $p(c|w)$. Given this observation, then *relationships* between words should also imply relationships between their corresponding distributions. Consider the 'entailment' relationship (or 'is-a' relationship) - i.e. a cat *is an* animal, or a car *is a* vehicle. Geffet and Dagan [2005] propose to extend the distributional hypothesis to their "distributional inclusion hypothesis", which states that one word v can be said to *entail* (or be a *hypernym* of) another word w if "the most characteristic contexts of v are expected to be included in all w's contexts (but not necessarily amongst the most characteristic ones for w)". If we thus consider *distributions* over possible contexts $p(c|v)$ and $p(c|w)$, the support of $p(c|w)$ should be a superset of the support for $p(c|v)$. This line of inquiry has been investigated by many different works [Vilnis and McCallum, 2014, Sun et al., 2018, Singh et al., 2020], each of which represented these distributions in different ways.

[Singh et al., 2020] considered expressing these distributions directly as discrete histograms over context words. They proposed a "context-mover's distance", which utilized the Wasserstein distance over a ground metric based on the similarity operator $\circledcirc$ defined by Henderson and Popa [2016] as their ground metric: $D_{ij}^{Hend} = -v_i \circledcirc v_j$.

$$\text{CMD}(w_i, w_j; D) = \text{OT}(P_V^{w_i}, P_V^{w_j}; D) \tag{1.2}$$

Alternatively, since Word2Vec vectors are trained to encode the distribution $p(c|w)$, distributions within the latent space of Word2Vec vectors can also be used to demonstrate the same properties [Vilnis and McCallum, 2014, Sun et al., 2018]. Vilnis and McCallum [2014] propose to use Gaussian mixtures centered on point embeddings of words within the latent space of Word2Vec. These embeddings were then compared against one another by means of distance measures such as the KL Divergence, or the Wasserstein distance in the case of Sun et al. [2018].

1.1.3 Probabilistic Word Embeddings for Polysemy and Homonymy

In a similar fashion to the works of Vilnis and McCallum [2014] and Sun et al. [2018], many other authors have also explored modelling distributions within the latent space of embedding

vectors. One common motivation for these probabilistic approaches is the problem of polysemy and homonymy. A very large proportion of words in the English language can have multiple distinct meanings in different contexts. As such, it may not be sensible to encode the meaning of a given word into a single static vector - instead, the latent meaning of a given token can be viewed *probabilistically* as a mixture over different "senses" or "prototypes". This concept was first introduced in Tian et al. [2014], who proposed a method to extend the well-known Word2Vec embeddings into a mixture over "prototypes" using expectation maximization. Liu et al. [2015] also sought to address this problem by proposing a form of "topical word embeddings". Their proposed method adapted the well-known Latent Dirichlet Allocation of Blei et al. [2003] to infer latent "topics" for each word, and learned different representations of each word vector under each distinct topic. In order to incorporate contextual information, the authors propose a Bayesian approach wherein the distribution over the latent topic z for a given word w with context c is given by $P(z|w,c) \propto p(w|z)p(z|c)$. The "contextual embedding" for w in context c is then found via a mixture over possible topics.

$$w^c = \sum_{z \in T} p(z|w,c) w^z \tag{1.3}$$

Miao et al. [2019] extended the ideas proposed in Tian et al. [2014] with respect to the *output* embeddings of language models, rather than focusing solely on input embeddings. Language models often consist of a layer of input embeddings W_{in}, a decoder f, and a layer of output embeddings W_{out} such that for an input word $\hat{y}_{t-1}$ observed in context c_t:

$$e_t = \text{Lookup}(W_{in}, \hat{y}_{t-1}) \tag{1.4}$$

$$h_t = f(c_t, e_t) \tag{1.5}$$

$$P(y_t = i) = \text{Softmax}(h_t W_{out})_i \tag{1.6}$$

Miao et al suggest that the traditional softmax approach relies on the assumption that each word can be accurately represented by a single vector, and that the context vector h will always be able to approximate the single point vector of the ground truth word. In a similar fashion to Tian et al. [2014], the authors propose to instead model words as mixtures over a dynamically allocated number of *senses*, each with their own vector encodings. The final prediction would then be based on these probabilistic representations rather than single word encodings in order to reduce the ambiguity induced by polysemy and homonymy.

1.2 Core Problems

The examples shown in Section 1.1 highlight how analyzing distributions of vectors within latent spaces of word embeddings can lead to diverse and powerful applications. Two aspects of these

methods in particular will become the foundation on which much of the work within this thesis is built:

1. The representation of word embeddings as *distributions* over many possible semantic encodings, and how this can be used to resolve ambiguities in the meanings of tokens. While these methods were used in works such as Liu et al. [2015], Tian et al. [2014] and Miao et al. [2019] in order to address polysemy and homonymy, ambiguity in word meanings can also be induced by other factors - for example, text that contains noise.

2. The problem of inferring relationships between words by analyzing relationships between their induced distributions over contexts. Rather than use handcrafted distance functions between these distributions such as those discussed in Singh et al. [2020] and Sun et al. [2018], another approach might be to *train* a distance function that will encode a given relationship. This gives rise to the more general question of how to approximate learnable distance functions between multiple distributions or sets of samples.

These motivating questions will provide the backdrop for the contributions detailed in the rest of this thesis, and will lead to a number of discoveries with applications in many diverse areas of machine learning - far beyond the narrow initial scope of word embeddings. I will now discuss these two key focus areas in detail, and lay the groundwork for the contributions that will be discussed later in this work.

1.2.1 The Problem of Noise in Natural Language

As discussed in Section 1.1.3, many previous works have explored the uses of probabilistic representations in resolving ambiguities in the meanings of tokens due to polysemy and homonymy. An interesting extension of these works would be to see if similar methods can be applied to *other* sources of ambiguity - for example, the problem of noise. Noise is a significant challenge for existing methods in the field of natural language due to the difficulties it induces in the process of tokenization and embedding. Embedding schemes using traditional word-based tokenization must use a fixed vocabulary, and cannot handle unknown, or "out-of-vocabulary" (OOV) words. This can be a major problem when dealing with data mined from social media - an increasingly common domain for NLP tasks - as social media data frequently contains misspellings, slang, or other distortions [Belinkov and Bisk, 2017, Khayrallah and Koehn, 2018]. Such noise can cause otherwise highly successful models to completely break down, as demonstrated by Belinkov and Bisk [2017].

Traditional pre-processing approaches to noise such as text normalization or spell-checking can be used to mitigate this, but are often outperformed by end to end approaches [Malykh et al., 2018, Doval et al., 2019]. Another approach is to simply train embeddings on noisy corpora, but,

as noted by Piktus et al. [2019], obtaining meaningful representations for all misspelled tokens may require an impractical expansion of the size of the training data. The compromise solution used by many recent models is to use byte-pair encoding (BPE) for tokenization [Sennrich et al., 2016] (see Section 2.1.1). With BPE, any word can be tokenized and embedded - even words that would otherwise be OOV. The downside to this is that such noisy words will not be embedded as a single token, but rather as multiple subwords or characters. Consider the example of the word 'redact'. If this word is misspelled to 'redavt' (a simple error to make on most keyboards, as the 'c' and 'v' keys are adjacent), then what was previously the single token ['redact'] is now encoded as four tokens ['red' 'a' 'v' 't']! Worse still, none of these tokens carry much semantic information - or any relationship to the meaning of the original word. As such, while approaches such as BPE or subword tokenization enable models to handle noise better than traditional word-based tokenization, they are still not an adequate replacement for a true noise-aware approach.

Fundamentally, the majority of existing approaches fail to fully tackle this problem due to a misalignment between the proposed methods and the underlying problem. The problem of noise in general is one that is intrinsically uncertain and ambiguous. Given a particular misspelled token, there are often multiple possible 'corrections' that one could make to obtain a valid word from the vocabulary - and thus multiple possible semantic roles that a token could perform in a statement. Just as a single representation for the meaning of a polysemous word is insufficient to capture its range of possible meanings, any single correction to a noisy token is also necessarily overconfident for the same reason. Deterministic approaches are thus insufficient to address this problem - in order to truly tackle robustness, the problem of noise must be addressed *probabilistically*. In order to address this problem, existing approaches usually make one of three choices:

1. Proposing single corrections to noisy tokens (e.g. text normalization and most denoising methods). These corrections are necessarily overconfident, as they do not account for the uncertainty in the meanings and semantic roles of the token, and may also remove important semantic information, as discussed in Doval et al. [2019].

2. Proposing alternative scratch-trained vectors which include penalty terms to ensure that words which are lexically close are embedded near each other. These methods essentially do *implicitly* incorporate uncertainty, as we can view the embedding of each word as a sort of mixture over the embeddings of nearby words. These methods can only be used with their particular vectors, however, which limits their applicability.

3. Using subword-level tokenization to remove the out-of-vocab problem. As mentioned earlier, this does compensate for the problem, but can lead to strange ways of tokenizing words that may remove or alter semantic information.

Each of these approaches has significant limitations that prevent it from being a fully general solution to this problem. A better approach is to model this uncertainty in a truly noise-aware way - an approach which requires a probabilistic framework.

1.2.2 Approximating Distances between Distributions

As mentioned earlier in Section 1.1, many previous works have investigated how relationships between the semantic meanings of words can be reflected in the structures of distributions in latent space associated with those words. The details of these works and the relationships between them will be discussed in Section 2.2.2 - but for the moment it suffices to observe that each of these works provides different ways to construct distributional representations of words and compare these representations against one another in order to infer entailment relationships. In each case, a distribution is constructed to represent the latent distribution of contexts induced by a particular word - either by direct histograms over co-occurrence matrices, or continuous distributions within embedding space. These distributions are then compared by some distance function $D(f, g)$, which in existing works was taken to be some handpicked function such as KL Divergence or Wasserstein distance.

While these distance functions are theoretically well-motivated, there is no guarantee that they are optimal for this task. In addition, these distance functions are specialized to the entailment relationship, and do not generalize to other possible relationships between words. Consider instead the notion of *training* a flexible distance metric $D_\theta(f, g)$ between distributions that could be optimized to learn any given set of relationships, given a reference corpus and a training set of existing relationships between some candidate words.

This is really a special case of a far more general problem: the problem of training a neural network to approximate a distance function between two distributions. Solving this problem would have wide-ranging implications far beyond the narrow subject of inferring word relationships. In addition to learning entirely new and bespoke distance functions to perform particular tasks (such as the aforementioned task of unsupervised inference of hierarchical word relationships), a solution to this problem would also allow for improved approximations of *existing* divergences - including exactly those divergences used in the existing works of Vilnis and McCallum [2014], Sun et al. [2018] and Singh et al. [2020]. These well-known distance functions (e.g. KL Divergence, Mutual Information or Wasserstein Distance) are often intractable, and very difficult to effectively approximate outside of specific contexts where they may have well-defined closed form solutions (such as the Gaussian distributions used by Vilnis and McCallum [2014]). This often requires methods to be formulated in restrictive ways in order to effectively leverage these distance functions. Finding a class of neural network functions that could effectively approximate functions between distributions would allow for the pretraining of approximators to compute these previously-intractable functions, which could have wide-ranging and general applications across all of machine learning.

In practice, the distributions under consideration in these settings are often accessible only as discrete sets of samples. In that case, this problem may be specified even further: proposing a neural network architecture that is suitable to compute mappings between *multiple permutation-invariant sets* of inputs. Such an architecture would have applications even beyond the com-

putation or approximation of divergences between distributions, as it would also represent an extension of existing set-based neural networks [Zaheer et al., 2017, Lee et al., 2019] to the more complex domain of functions defined between *multiple* sets. This problem is largely unsolved in the literature, and has many applications - for example, pretraining discriminators to differentiate between sets drawn from different distributions, or predicting the alignment between sets of examples from different domains (such as images and their captions).

This area of inquiry will thus focus on the broad and general problem of approximating functions between distributions or sets of samples using neural networks. Rather than focusing narrowly on the initial problem of hypernymy, I will use that initial question as a springboard to motivate the study of general functions on multiple sets, and how neural network architectures can be used to approximate these functions in a wide variety of settings.

1.3 Contributions & Overall Structure

The problems described in Section 1.2 will provide the foundation for the ensuing explorations throughout this work. These contributions will span a wide variety of different fields and domains, but all are motivated by the power of distributional approaches in understanding and enhancing machine learning models. The focus of this work is threefold:

- Analyzing empirical distributions of points embedded in continuous latent spaces, and how these distributions of vector encodings can be leveraged in powerful applications - for example, improving the robustness of natural language models to noisy inputs by modelling distributions over latent embedding vectors.
- Building powerful tools that can be used to aid in modelling these distributions over encodings, such as novel methods of approximating functions defined on distributions or sets of samples.
- Showcasing how these tools can be applied to solve timely and relevant problems in the landscape of machine learning today - such as building new types of foundation models in image generation.

In Chapter 2, I will review a number of key concepts such as word embeddings, attention and generative adversarial networks that will lay the foundations for the work in subsequent chapters. This chapter will also summarize the existing literature on the problems discussed in Section 1.2, and provide a detailed background on the existing state of the field with respect to these motivating questions.

In Chapter 3, I will focus on the first of the key focus areas defined in Section 1.2. I will demonstrate how the problem of building models robust to noisy or error-ridden data is intrinsically connected to modelling uncertainty. I will propose a probabilistic framework called Robust

Embeddings Via Distributions (or RED) that explicitly models uncertainty in tokens and their meanings, and uses this to improve performance on downstream tasks by combining information about the noisy token itself with information gleaned from the surrounding context. This framework is model-agnostic, and can be used to improve the performance of any natural language model on downstream tasks involving noisy text. I will compare this proposed approach to a number of existing baselines, and demonstrate superior performance across the diverse tasks of the well-known GLUE benchmark.

In Chapter 4, I will answer the fundamental question raised in the second key focus area: how to build a neural network model to approximate learnable functions defined between multiple distributions or sets of samples. I will propose a novel, general-purpose architecture to model functions defined on multiple permutation-invariant sets - a problem for which almost no proposed solutions exist in the literature. I will demonstrate that this architecture is a universal approximator of *partially-permutation-invariant* functions, and show that it outperforms the few other approaches that do exist by significant margins on a variety of different tasks. I will demonstrate the effectiveness of this model on a wide variety of tasks relating to multiple input sets, such as predicting alignment between sets from different domains, as well as distinguishing between sets sampled from the same or different distributions. I will also demonstrate how to use the same principles of permutation invariance to define models that are equivariant with respect to the input dimension, and thus train estimators of statistical distance qualities such as KL divergence and mutual information that can be applied to inputs of any dimensionality.

In Chapter 5 I will highlight the broad applicability of the model from chapter 4 by showing how the task of distinguishing between sets from different distributions can be extended to define a discriminator architecture for a novel type of generative adversarial network. This proposed generative architecture - titled SetGAN - can be used to perform few-shot image generation by generating sets of images conditioned on a set of reference images. The model learns to generate images drawn from the same distribution as the reference set. By using a set-based approach and incorporating the architecture from Chapter 4 into the discriminator, this approach avoids the limitations that plague existing models, and does not rely on restrictive assumptions about the structure of the data. As such, this model has the potential to be scaled to perform truly zero-shot image generation in a style similar to foundation models such as DALL-E [Ramesh et al., 2022] or Stable Diffusion [Rombach et al., 2022].

Finally, Chapter 6 will give an overview of the key contributions from the previous three chapters, and concisely summarize the significance of the work as a whole. In this chapter, I will also discuss potential extensions of the work presented herein, and how this thesis lays the groundwork for promising future lines of inquiry. In particular, I will discuss two key avenues of research stemming from the work in Chapters 4 and 5: pretraining a general estimator for mutual information, and extending the SetGAN model to large-scale datasets such as ImageNet in order to achieve true zero-shot image-to-image generation.

Chapter 2

Background

2.1 Preliminaries

In this section, I will review a number of commonly used techniques in machine learning that will become relevant to later sections of this thesis. While many of these methods are fairly standard in the literature, I provide my own detailed definitions here in the interest of clarity, and for consistency of notation with later chapters. Any references to concepts defined in this section in later chapters can be assumed to follow the notation and definitions detailed here.

2.1.1 Word Embeddings

Skip-Gram with Negative Sampling

Word embeddings first exploded in popularity after the success of Word2Vec, first proposed by Mikolov et al in 2013[Mikolov et al., 2013a]. Word2Vec proposed to embed each word w in the vocabulary V by a specific fixed-size vector v_w, that would somehow encode the semantics of that word and its relationship to other words in the vocabulary. Consider a vocabulary V and corpus $T = [w_1, ..., w_N; w_i \in V]$. In the most succesful variant of the Word2Vec model (known as Skip-gram) the model seeks to predict the likelihood of the context C_i given the center word w_i, where C_i is a "context window" of l words in each direction around the center word - i.e. $C_i = \{w_{i-l}, ..., w_{i-1}, w_{i+1}, ..., w_{i+l}\}$. Skip-gram thus maximizes the loss function: [Goldberg and Levy, 2014]

$$\arg\max_{\theta} \prod_{w_i \in T} \left[\prod_{c \in C_i} p(c|w;\theta) \right] \tag{2.1}$$

This can also be rewritten as follows:

$$\arg\max_{\theta} \prod_{(w,c)\in D} p(c|w;\theta) \tag{2.2}$$

where D is the set of all word-context pairs. In order to parameterize this model, Skip-gram represents each word $w \in V$ by two fixed-size vectors: the "center word representation" u_w and the "context word representation" v_w. Using this parameterization, $p(c|w;\theta)$ can be defined using a softmax as:

$$p(c|w;\theta) = \frac{e^{u_w \cdot v_c}}{\sum_{c' \in V} e^{u_w \cdot v'_c}} \tag{2.3}$$

This equation causes some computational difficulties due to the size of the sum in the denominator of the softmax. While there are some ways to resolve these difficulties, the most popular form of the Skip-gram model actually formulates the problem in a slightly different way to bypass them entirely. This is known as Skip-gram with Negative Sampling (SGNS), and it forms the basis for almost all other embedding models in the literature.

SGNS still considers pairs (w, c) of center and context words respectively, but it does not use a softmax. Instead, SGNS predicts the probability that an arbitrary pair (w, c) is a *true* pair that actually occurs in the training corpus T, rather than a *false* pair that does not. The loss can then be reformulated as follows: [Goldberg and Levy, 2014]

$$\arg\max_{\theta} \prod_{(w,c)\in D} p(D = 1|c, w;\theta) \prod_{(w,c)\in D'} p(D = 0|c, w;\theta) \tag{2.4}$$

where $p(D = 1|c, w;\theta)$ is the predicted probability that (w, c) is a valid pair from the training data, D is the set of all such true pairs that *do* occur in the data, and D' is a set of "negative examples" (thus the name) that *do not* come from the data. This probability is parameterized by

$$p(D = 1|c, w;\theta) = \sigma(u_w \cdot v_c)$$

where

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

By using this definition, taking the log and doing some algebraic manipulation, the loss defined in Eq. 2.4 can be expressed in the forms

$$\arg\max_{\theta} \sum_{w,c\in D} \log \sigma(u_w \cdot v_c) + \sum_{w,c\in D'} \log \sigma(-u_w \cdot v_c) \tag{2.5}$$

or, using l as the negative log logistic function $l(x) = \log(1 + e^{-x})$ and casting the problem as minimization,

$$\arg\min_{\theta} \sum_{w,c\in D} l(u_w \cdot v_c) + \sum_{w,c\in D'} l(-u_w \cdot v_c) \tag{2.6}$$

which is the form often used by other papers built on this method.

In order to obtain vector representations from this algorithm, the matrices U and V are trained in order to maximize this log-loss on the corpus T, and then the matrix V is discarded, keeping only the "center" vectors in U as the vector representations of the words in V.

Byte-Pair Encoding

While word-level methods such as the ones mentioned previously dominated the field for many years, many modern transformer-based models use tokenizations based on subword units or character n-grams. One of the most common and flexible such forms of tokenization is Byte-Pair Encoding (or BPE).

Algorithm 1: Byte-Pair Encoding (Naive)

Input: Corpus T of length n, character set C, dictionary size m
Tokens $\leftarrow \{[c_i] \; \forall c_i \in C\}$
Function `Count(`T, x`):`
  $c \leftarrow$ # occurrences of x in T
  return c
Function `FindNextToken(`T, X`):`
  2Grams $\leftarrow \{[x_1; x_2] \;\; \forall x_1, x_2 \in X\}$
  NextToken $\leftarrow \arg\max\{$`Count(`T,g`)` $\;\; \forall g \in$ 2Grams$\}$
  $x \leftarrow$ 2Grams[NextToken]
  return x

while $|Tokens| < m$ **do**
  Tokens $\leftarrow$ Tokens $\bigcup$ `FindNextToken(`T, $Tokens$`)`
end

Byte-pair encoding was first proposed in Gage [1994b] as a method of data compression, but was adapted for use in tokenization by Sennrich et al. [2016]. The algorithm consists of starting with an initial dictionary of single characters, then iteratively expanding the dictionary by adding a new token consisting of the most common 2-gram of existing tokens within a given reference corpus. A naive version of this algorithm is described in Algorithm 1 in order to demonstrate the method (though this implementation is inefficient and should be used only for illustrative purposes).

This procedure leads to a tokenization method with a flexible dictionary size, which learns common and semantically important subword tokens, as well as many common words. Unlike word-based tokenization methods, BPE-based tokenization has no notion of "out-of-vocabulary"

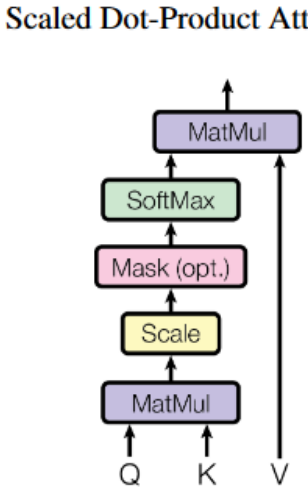


Figure 2.1: The base attention mechanism [Vaswani et al., 2017].

words, as any string of text can always be encoded by the resulting token dictionary - being at worst equivalent to character-level tokenization. This form of tokenization is a flexible intermediary between word-level and character-level tokenization, and is used by nearly all large transformer-based models today.

2.1.2 Attention

The attention operation and the associated transformer architecture have become the backbone of much of modern machine learning - and nowhere more so than in the field of natural language. Variants were first proposed as augmentations to existing sequence-to-sequence models in Sutskever et al. [2014a] and Cho et al. [2014], but quickly exploded after the introduction of the transformer network in Vaswani et al. [2017].

Fundamentally, the attention mechanism (shown in Fig. 2.1) consists of a mapping acting on three input sets: $Q \in \mathbb{R}^{n \times d_k}$, $K \in \mathbb{R}^{m \times d_k}$ and $V \in \mathbb{R}^{m \times d_v}$ - referred to respectively as the *queries*, *keys*, and *values*. Each key is paired with a given value, and in practice the keys and values are frequently chosen to be the same vectors (at least up to a linear transformation). The essential idea of the attention mechanism is that for each *query* $q \in Q$, a scoring function $s(q, k)$ is applied to rank the similarity between q and each of the m *keys* $k \in K$. These scores are then normalized by a softmax function (denoted σ) to achieve a set of weights W defined on the m-dimensional probability simplex:

$$W_q = \sigma(s(q, k) \;\; \forall k \in K \tag{2.7}$$

Finally, each *value* is scaled by the weight of its associated key, and the final output is computed by a weighted sum. When the scoring function is defined as $s(q, k) = q \cdot k$ (as it frequently is), this can be written neatly as:

$$\text{ATTN}(Q, K, V) = \sigma(QK^T)V \tag{2.8}$$

Note that in practice, a scaling factor of $1/\sqrt{d_k}$ is often added to the outputs of the scoring function in order to improve gradient flow.

Multi-Headed Attention

This basic operation led to the birth of the transformer architecture in the seminal work of Vaswani et al. [2017]. This paper defined the *multi-headed attention block*, which makes several key modifications to the architecture detailed above. First, each of the queries, keys and values are first transformed by weight matrices $W^Q, W^K \in \mathbb{R}^{d_{in} \times d_k}, W^V \in \mathbb{R}^{d_{in} \times d_v}$. Second, the mechanism is split into h different parallel*attention heads*, each of which have their own parameter matrices and act independently on the inputs. The outputs are then concatenated and transformed by an output matrix $W^O \in \mathbb{R}^{hd_v \times d_{out}}$ - or, equivalently:

$$\text{MHA}(Q, K, V) = \sum_{i=1}^{h} \sigma\left((QW_i^Q)(KW_i^K)^T)\right) VW_i^V W_i^O \tag{2.9}$$

This is often written as $\text{MHA}(Q, K)$, as identical inputs are frequently used for both the keys K and values V.

Transformers

This multi-headed attention block forms the backbone of the *transformer* architecture. There are often two types of transformer blocks used in practice: the transformer *encoder* block and *decoder* block.

Transformer encoder blocks consist of *self-attention* layers, where the outputs of the previous block (or the initial inputs, in the case of the first block) are used as inputs to the queries, keys, *and* values of the next block. Intuitively, this corresponds to computing intra-sequence relationships between each element of the input with all other elements - hence the name "self-attention", as the sequence is "attending" to itself. Each block is composed of a multi-headed attention block, followed by an elementwise feedforward network that is applied independently to each entry of the output. Usually, both the attention and feedforward layers are followed by layer normalization and residual connections to improve gradient flow. Examples of this structure are

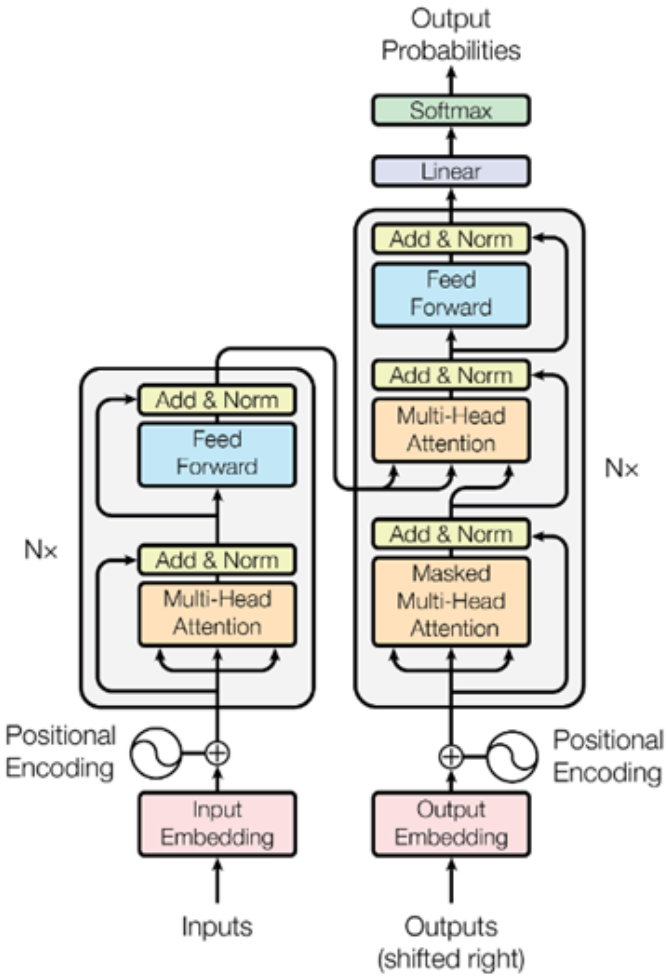


Figure 2.2: The original transformer encoder-decoder model from Vaswani et al. [2017].

shown in the left half of Figure 2.2. Specifically, we can represent a single transformer encoder block by the equations:

$$\begin{aligned} Z &= \mathrm{LN}(X + \mathrm{MHA}(X, X)) \\ T(X) &= \mathrm{LN}(Z + \mathrm{FF}(Z)) \end{aligned} \tag{2.10}$$

wherein LN is the Layer Norm operation [Ba et al., 2016], and FF is a shallow (typically 2-layer) feedforward network applied independently to each element in the set.

The transformer decoder block is structured along similar principles, but is used when there is an external set or sequence to be attended to (for example, the "decoder" of a sequence-to-sequence model generating an output sequence while attending to the input sequence). This block consists of a self-attention layer followed by a cross-attention layer, and a final feedforward layer. This architecture is shown in the right half of Figure 2.2. As above, this can be described by the equations:

$$\begin{aligned} Z_1 &= \mathrm{LN}(X + \mathrm{MHA}(X, X)) \\ Z_2 &= \mathrm{LN}(Z_1 + \mathrm{MHA}(Z_1, Y)) \\ T(X, Y) &= \mathrm{LN}(Z_2 + \mathrm{FF}(Z_2)) \end{aligned} \tag{2.11}$$

Originally these architectures were used together as an encoder-decoder model for sequence-to-sequence generation. Now, however, each of the transformer encoder and decoder models are often used independently in a wide variety of applications. The vast majority of models used in natural language processing today rely heavily on these architectures, and nearly all so-called "foundation models" are built upon one of these two frameworks. Transformers are also increasingly popular in many other domains and applications, including computer vision [Dosovitskiy et al., 2021]. When used in domains such as sequence-to-sequence prediction, the inputs are commonly modified by a *positional encoding*, which adds a variable offset to each embedded vector in the sequence in order to break the permutation-equivariance of the attention operation. This property will become of particular interest in Chapter 4, and will be discussed at greater length at that time.

2.1.3 Generative Adversarial Networks

Generative adversarial networks (or GANs) [Goodfellow et al., 2014] are a well-known class of generative model that have been exceedingly popular since their inception in 2014. These models consist of two components: a *generator* and a *discriminator*, which train jointly in an adversarial fashion by playing a minimax game. As in all generative modelling, the goal is to train a model to generate new samples from an underlying data distribution $\mathcal{D}$ defined on a domain $\mathcal{G}$, given a

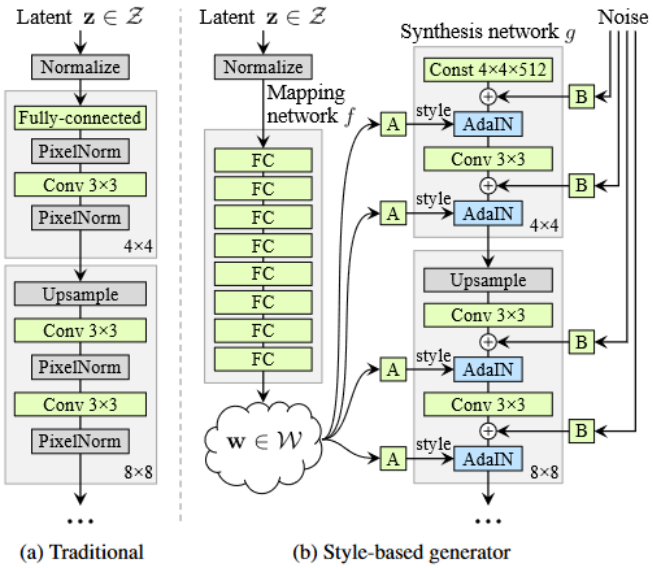


Figure 2.3: A diagram of the StyleGAN generator architecture from Karras et al. [2019] as compared to previous convolutional GANs.

number of empirical samples from said distribution. These samples are referred to as the training set, which I will denote as $\mathcal{D}_{train}$.

The generator $G : \mathcal{Z} \to \mathcal{G}$ consists of a mapping from a high-dimensional latent distribution p_Z defined on a latent space $\mathcal{Z}$ (often taken to be a standard multivariate normal on the space $\mathbb{R}^d$) into the space of the data distribution, and is trained to generate samples that are indistinguishable from the elements of $\mathcal{D}_{train}$. The discriminator, conversely, acts as a mapping $D : \mathcal{G} \to [0, 1]$, which is trained to predict the probability that a given element from the domain belongs to the data distribution $\mathcal{D}$. The adversarial training of the models consists of the following optimization process:

$$\min_G \max_D \mathbb{E}_{x \sim \mathcal{D}_{train}} [\log D(x)] + \mathbb{E}_{z \sim p_Z} [\log 1 - D(G(z))] \tag{2.12}$$

This training process is equivalent to minimizing the Jensen-Shannon Divergence $JSD(\mathcal{D}||G(p_Z)$ between the data distribution $\mathcal{D}$ and the generated distribution $G(p_Z)$.

StyleGAN

One of the most common modern GAN architectures is the StyleGAN architecture proposed by Karras et al. [2019], and further refined in Karras et al. [2020][1]. These works proposed an entirely different structure for the generator G compared to existing works, as shown in Figure 2.3. Rather than reshaping the input latent code into a pixel map that would then become the input to the rest of the convolutional generator, StyleGAN instead uses a learned *constant* input. Instead, the latent vector z is incorporated into the output by way of a series of scaling operations within the convolutions themselves.

First, z is passed through a deep feedforward network known as the *mapping network*, resulting in the *style vector* $w \in \mathcal{W}$. This network has the effect of conditioning the latent space $\mathcal{W}$, essentially acting as a form of regularization to improve the disentanglement of latent factors within the space. After this, the resulting style vector w becomes an additional input to each of the n_s convolutional layers in the generator. At each layer, a learned weight matrix projects the style vector into the space of the feature maps at that convolutional layer, resulting in a series of scaling factors s_i - one for each input channel at that layer. The convolution weights w_{ijk} at that layer are then scaled to become $w'_{ijk} = s_i w_{ijk}$, before they are then applied to the input. StyleGAN also uses skip connections in its generator: at each resolution level, the output at that layer is mapped into an RGB image, then upsampled to the final output resolution. The final generated output is the sum of all these intermediate output images. This has the effect of ensuring that the convolutional blocks at each resolution level focus only on the output features at that resolution scale - early layers of the network affect coarser features, while progressively later layers affect finer and finer details.

Together, these changes have the effect of ensuring that the style vector w_i used as the input to the i-th convolutional layer will control the application and intensity of the particular features attended to by that layer's convolution maps. While during training, the same w vector is used as the input to all layers[2], this is not required at inference time, and different styles can be used at different layers to alter and control the generations. These properties grant the StyleGAN architecture highly interpretable relative to other generative models, and make it easy to interpolate and combine features from different images in many different ways - as explored in depth in the original papers.

[1] The original StyleGAN architecture has a number of differences from the improved StyleGAN2 architecture, but both operate on similar principles. For the rest of this section any references to "StyleGAN" can be assumed to be describing StyleGAN2 wherever there is any difference.

[2] except when using the "style mixing" regularization

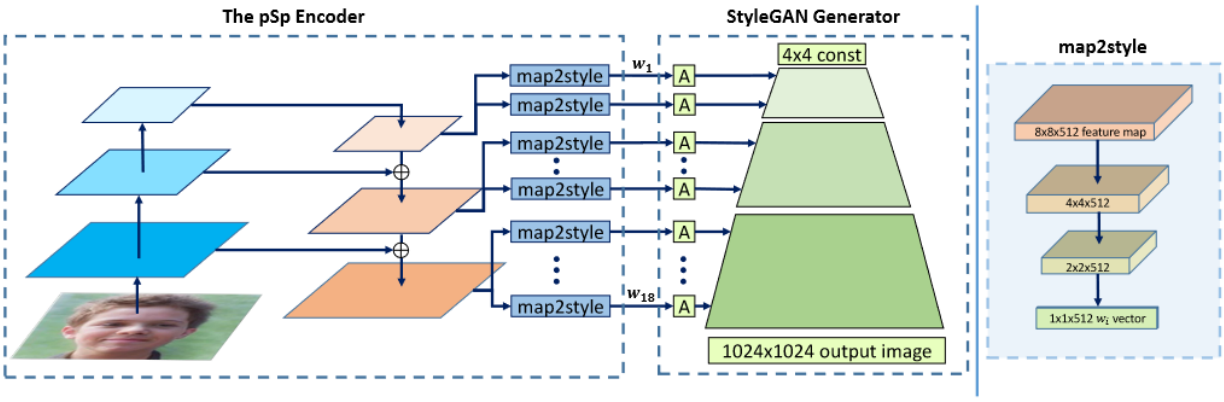


Figure 2.4: The pixel2style2pixel encoder architecture from Richardson et al. [2021]. Feature maps are extracted at three levels of resolution (coarse, medium and fine) using a ResNet. A "map2style" network at each layer learns to extract the style vector from the appropriate feature map (0-2 for coarse, 3-6 for medium, 7-18 for fine).

GAN Inversion

If image generation using GANs is the process of mapping from a latent space $\mathcal{Z}$ to the space of output images $\mathcal{G}$, then GAN inversion is the inverse process of mapping from $\mathcal{G}$ back into the latent space $\mathcal{Z}$. This can be done in a number of ways, including optimization-based methods and encoder-based methods. The areas of manipulating, editing and inverting images using StyleGAN have drawn considerable interest, and there is a large body of existing work on this subject. Much of this work, however, focuses on optimizations in the latent space, or other non-encoder-based approaches. For the purposes of this work, the focus will be on performing encoder-based GAN inversion on the latent space of StyleGAN2. That is to say, defining an encoder $F : \mathcal{G} \rightarrow \mathcal{W}^+$, where $\mathcal{W}^+$ is the expanded latent space $\mathcal{W}^{n_s}$ wherein each of the style vectors are permitted to vary independently.

One such encoder framework is the *pixel2style2pixel* (pSp) architecture proposed in Richardson et al. [2021]. Given a trained StyleGAN network, pSp trains a ResNet-based architecture to map any given image into the series of n_{style} style vectors that, when fed into the StyleGAN generator, will most closely approximate that image. pSp's architecture (shown in Figure 2.4) consists of a deep ResNet backbone that decomposes the input image into feature maps at three levels of resolution: coarse, medium and fine. A series of n_s "map2style" layers then further decompose these feature maps into the desired style vectors. Styles 0-2 are learned from the "coarse" feature map, 3-6 from the "medium" feature map, and 7-18 from the "fine" feature map[3].

[3]These numbers assume a StyleGAN generator that generates 1024x1024 images (corresponding to $n_s = 18$).

This architecture was later refined further by papers such as Alaluf et al. [2021] and Tov et al. [2021], proposing architectures such as ReStyle and E4E. These serve a similar purpose and have a similar structure, though use other refinements such as performing multiple passes through the encoder to more accurately minimize the error between the input and reconstructed image.

2.1.4 Diffusion Models

Another class of generative image model that has achieved great success in recent years is that of the *diffusion model* [Sohl-Dickstein et al., 2015, Ho et al., 2020a]. Diffusion models are a class of probabilistic generative model based on mathematical concepts from statistical thermodynamics. These models consider a *forward diffusion process* as a Markov chain which iteratively adds a small amount of Gaussian noise to a source datapoint $x_0 \sim q(x)$ to generate increasingly noisy samples $x_1, ..., x_T$ using a variance schedule β_t.

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbf{I}) \quad q(x_{1:T}|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{2.13}$$

This forward process is paired with a *reverse process* $p_\theta(x_{t-1}|x_t)$ which iteratively reconstructs the original data sample x_0 from the corrupted noisy samples $x_1 : T$

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)) \quad p_\theta(x_{1:T}|x_0) = \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{2.14}$$

Training is done using the variational bound:

$$L = \mathbb{E}_q \left[-\log p(x_T) - \sum_{t \geq 1} \log \frac{p_\theta(x_{t-1}|x_t)}{q(x_t|x_{t-1})} \right] \tag{2.15}$$

This loss function can be rewritten as a combination of KL divergence terms that are tractable when conditioned on x_0, and allow for efficient training by optimizing random terms of L using stochastic gradient descent. This formulation corresponds with the commonly used *denoising diffusion probabilistic model* (DDPM) framework proposed in Ho et al. [2020a], but many variations and related models have been explored in works such as Song and Ermon [2020], Song et al. [2019b] and Song et al. [2022].

These architectures are commonly used to build powerful image generation models, using large convolutional decoders with a U-Net structure to approximate the reverse process. Models of this nature have been extremely successful, but suffer from significant limitatations in their ability to perform efficient inference, with Song et al. [2022] reporting that "it takes around 20 hours to sample 50k images of size 32 × 32 from a DDPM, but less than a minute to do so from a GAN on a Nvidia 2080 Ti GPU. This becomes more problematic for larger images as sampling 50k images of size 256 × 256 could take nearly 1000 hours on the same GPU".

Latent Diffusion Models

In order to resolve this limitation, Rombach et al. [2022] proposed *latent diffusion models*, which combined the powerful-yet-inefficient diffusion model backbone with a deep convolutional autoencoder network for upsampling. The diffusion model backbone would be used to generate images within the latent space of the autoencoder network, which would then be upsampled by the decoder to produce high-quality, high-resolution outputs without the prohibitive computational cost of running a diffusion model at high output resolution. This approach has been extremely successful, and has become the basis for many so-called "foundation models" for zero-shot text-to-image generation, such as DALLE-3 [Betker et al., 2023] and Stable Diffusion.

2.1.5 Few-Shot Image Generation

Few-shot image generation consists of a dataset $\mathcal{D}$ divided into a number of classes $\{\mathcal{C}_i\}$, which are each composed of some $n_{\mathcal{C}_i}$ images. These classes are partitioned into a disjoint training set $\mathcal{D}_{\text{train}}$ and test set $\mathcal{D}_{\text{test}}$. Given a particular set of *reference images* $\mathcal{C}_{\text{ref}} \subsetneq \mathcal{C}$, the model is trained to infer the latent class $\mathcal{C}$ and generate additional samples $\mathcal{C}_{\text{gen}}$ drawn from $\mathcal{C}$.

During evaluation, each test class $\mathcal{C} \in \mathcal{D}_{\text{test}}$ is divided into two disjoint sets: a *reference set* of n_{ref} images $\mathcal{C}_{\text{ref}}$, and an evaluation set of n_{eval} images $\mathcal{C}_{\text{eval}}$. The model then generates n_{gen} images $\mathcal{C}_{\text{gen}}|\mathcal{C}_{\text{ref}}$ for that class. These generated images are then used to evaluate the model using one of several different evaluation metrics - the most common of which are the Frechet Inception Distance (FID) [Heusel et al., 2018], and Learned Perceptual Image Patch Similarity (LPIPS) [Zhang et al., 2018]. Depending on the metric, these scores can be computed using $\mathcal{C}_{\text{gen}}$ and $\mathcal{C}_{\text{eval}}$ on a class-by-class basis (as is commonly done for LPIPS), or all generated sets and evaluation sets can be aggregated into a single $\mathcal{D}_{\text{gen}}$ and $\mathcal{D}_{\text{eval}}$ (as is commonly done for FID).

Metrics

Frechet Inception Distance [Heusel et al., 2018] is very often used as a measure of generation *fidelity* and *quality* for GANs - in both conditional and unconditional settings. Given a generated set $\mathcal{D}_{\text{gen}}$ and evaluation set $\mathcal{D}_{\text{eval}}$, FID is computed by embedding each image into a space of latent vectors, then measuring the statistical similarity between the distributions of embedded vectors corresponding to the evaluation set and generated sets respectively. A pretrained vision network (commonly the Inception network [Szegedy et al., 2015]) is used to perform the image embeddings, then Multivariate Gaussian distributions are fitted to each dataset of embedded vectors. The alignment between the two distributions is scored by way of the Frechet Distance (or 2-Wasserstein distance) between the resulting distributions. Given image encodings X, Y with

means μ_X, μ_Y and covariance matrices Σ_X, Σ_Y, this can be computed by:

$$\text{FID}(X, Y) = ||\mu_X - \mu_Y||^2 + \text{tr}\left(\Sigma_X + \Sigma_Y - 2(\Sigma_X \Sigma_Y)^{1/2}\right) \tag{2.16}$$

LPIPS [Zhang et al., 2018] is a metric used to measure perceptual similarity between pairs of images in a fashion that correlates with human judgments. It operates by using a pretrained network (often AlexNet or VGG) to evaluate two images, and storing the outputs of all convolutional layers in the network. Given two images x_1, x_2 with image embeddings $\{y_1^l\}, \{y_2^l\} \in \mathbb{R}^{H_l \times W_l \times C_l}$ (wherein l indexes the layer), the LPIPS distance between the two images is computed by averaging the ℓ_2 of the distances between the spatial representations, with the channel features scaled by a learned weight vector $w^l \in \mathbb{R}^{C_l}$.

$$\text{LPIPS}(x_1, x_2) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} ||w^l \odot (y_{1hw}^l - y_{2hw}^l)||_2^2 \tag{2.17}$$

The resulting score produces similarity judgments between pairs of images that track well with human evaluations. This metric is used to measure the *diversity* of generated images by computing pairwise distances between each pair of images in a given set of generations $\mathcal{C}_{\text{gen}}$. These distances are averaged within each class, then a final average over all classes is taken as a measurement of the overall diversity of the generated outputs.

Existing Methods

Existing models in this domain largely fall into one of two categories: generative adversarial networks (see Section 2.1.3) and diffusion models (see Section 2.1.4). Among these categories, the GAN-based approaches are typically divided further into one of three categories: optimization-based methods, fusion-based methods, and transformation-based methods.

Optimization-based methods were among the first models proposed to address this problem. Clouâtre and Demers [2019] and Liang et al. [2020] proposed meta-learning-based approaches in which pretrained generators were fine-tuned on examples from the evaluation class using meta-learning techniques such as MAML [Finn et al., 2017]. These approaches are not competitive with the current state of the art, however, and thus will not be an area of focus.

Fusion-based methods typically work by picking a single reference image as a focus, then combining local features of other reference images into the base image. This approach includes models such as F2GAN [Hong et al., 2020c], LoFGAN [Gu et al., 2021] and WaveGAN [Yang et al., 2022b].

Transformation-based methods have been some of the most successful approaches to this problem in recent years, including models such as DAGAN [Antoniou et al., 2017], DeltaGAN [Hong et al., 2020a] and AGE [Ding et al., 2022]. Transformation-based approaches rely on the fundamental assumption that the latent factors of variation within each of the classes $\mathcal{C} \in \mathcal{D}$ are similar to one another - i.e. that there exists some transformation $T : \mathcal{G} \rightarrow \mathcal{G}$ such that if $T(x_1 \in \mathcal{C}_1) \in \mathcal{C}_1$, then $T(x_i \in \mathcal{C}_i) \in \mathcal{C}_i$ for all other classes $\mathcal{C}_i$. If this property holds, then the class label of the image is effectively *invariant* under the transformation T. In practice, this is used to find some transformation T under which this invariance property holds for the training classes $\mathcal{D}_{\text{train}}$ - and the assumption made by these methods is that this invariance relationship will *continue* to hold for the test classes $\mathcal{D}_{\text{test}}$. Approaches such as AGE [Ding et al., 2022] using these methods have been highly successful in practice, but their applications are generally restricted to highly structured datasets on narrow domains such as images of human faces.

While these GAN-based approaches all focus specifically on the task of few-shot image generation, and often incorporate assumptions about the structure or relationship of the test and training datasets, diffusion-based approaches typically focus instead on *zero-shot generation*. This term is often used loosely to describe models trained on exceptionally large and diverse datasets, and expected to perform well at inference even when used on data belonging to many highly divergent domains, without any further finetuning. The majority of these approaches are designed to perform the distinct but related task of *text-to-image* generation, where the output images $\mathcal{C}_{\text{gen}}$ are conditioned on a text prompt rather than a set of reference images. Some work has been done, however, on adapting models of this type such as Stable Diffusion[Rombach et al., 2022] and DALLE-3 [Betker et al., 2023] to the task of image-to-image generation instead. DreamBooth [Ruiz et al., 2022] and HyperDreamBooth [Ruiz et al., 2023] focus on the closely related task of using large-scale diffusion models to generate images of particular subjects in different contexts - for example, using images of a family pet to generate pictures of that pet in front of the Acropolis in Greece, or swimming underwater. Models such as IP-Adapter [Ye et al., 2023] and Stable Diffusion Image Variations [Pinkney, 2023] adapt these models to the task of few-shot image generation directly, by allowing them to condition on image-based embeddings rather than text.

2.2 Focus Areas

Section 1.2 described the two key focus areas which motivated much of this work. This section will now discuss existing works in these areas in detail, in order to provide background on the current state of the art and how the contributions in this work serve to advance each of the areas under discussion.

2.2.1 Robustness in Natural Language

Existing works on robustness in natural language at the embedding level have typically fallen into one of two categories: denoising-based approaches, and scratch-trained robust vectors.

Denoising Approaches

Examples of the first approach can be found in many text denoising or spellchecking models. An early example of this can be found in the "noisy channel" model of Kernighan et al. [1990], who modelled possible corrections to each noisy token using 'confusion matrices', defined over other possible correction words with an edit distance of 1. Another example of this approach is the work of Sun and Jiang [2019], who proposed to use pretrained masked language models such as BERT [Devlin et al., 2018a] to perform text correction. Their approach proposed evaluating possible corrections to each token with a variable number of masked tokens to model how the addition of noise might distort the BPE tokenization to turn one token into several (see the example mentioned earlier with the word 'redact'). They then compare many possible corrections generated by the contextual model, and choose the one with the lowest edit distance to the noisy token at that location.

Scratch-Trained Vectors

A wide variety of works have proposed methods based on the second approach detailed above, proposing robust word embedding methods such as RoVE [Malykh et al., 2018], MOE [Piktus et al., 2019], Bridge2Vec [Doval et al., 2019] and RobEn [Jones et al., 2020]. While the details of these approaches vary, all apply similar principles of scratch-training a bespoke set of embeddings using various penalty terms to ensure that words with similar character representations are encoded near each other in the latent space.

Robust Word Vectors (RoVe) [Malykh et al., 2018] is a morphological context-dependent robust embedding technique targeting typos in the text and is able to deal with open vocabulary. RoVe derives word embeddings by decomposing words into beginning (B), middle (M), and end (E) components based on the common prefix, main, suffix word structures. Then, the output morphological embedding and both left and right context of the word are fed into an encoder to obtain RoVe embeddings.

The MOE approach proposed by Piktus et al. [2019] is modelled on the classic embedding method known as FastText [Bojanowski et al., 2016]. They use a modified loss function that adds a spelling correction term to improve the robustness of the representations. In addition to a training corpus T_{train}, MOE also considers a misspelling set T_{ms} (harvested from social media), consisting of word pairs (w_{ms}, w_e) such that $w_e \in V$ is a correctly spelled word and w_{ms} is a

misspelling of that word. Their spelling correction loss term encourages each word vector to be close to the vectors for its misspellings.

Doval et al. [2019] proposed a robust word embedding approach based on a modified version of skip-gram by introducing the concept of bridge-words (that is the chain of similar word variants that every two adjacent words are only different in one character, e.g., friend → frind → freind). They augment existing skip-gram-based methods such as FastText [Bojanowski et al., 2016] by altering the training process to include bridge-words in addition to each center word. This technique aims to encourage the vector representations for each word to be similar to the vector representations for nearby bridge-words, and thus to encourage the resulting vectors to be more robust to character-based noise.

Jones et al. [2020] propose a method they refer to as "RobEn" to address adversarial noise in text in a way that remains compatible with pretrained transformer models. They propose a framework for adding an encoding function before the downstream model to map all sentences to a smaller set of robust encodings. They discuss the theoretical properties these encoding functions should have, as well as providing two examples based on clustering algorithms. They show that their method outperforms existing baselines when dealing with adversarial noise - though they do not look at more realistic noise settings.

2.2.2 Hypernymy and Approximating Distances between Distributions

Hypernymy & Inferring Word Relations

As discussed in Section 1.1, the entire framework of cooccurence-based word embeddings rests on the Distributional Hypothesis [Harris et al., 1954, Firth, 1957], which states that the meanings of words are intrinsically characterized by the contexts in which those words occur. Section 1.1.2 also described one interesting extension of this principle: the Distributional Inclusion Hypothesis of Geffet and Dagan [2005]. This principle describes the 'is-a' relationship (or *entailment*) between words - i.e. a cat *is an* animal, or a car *is a* vehicle. The Distributional Inclusion Hypothesis states that one word v can be said to *entail* (or be a *hypernym* of) another word w if "the most characteristic contexts of v are expected to be included in all w's contexts (but not necessarily amongst the most characteristic ones for w)". One of the more interesting results of this is that words which share an entailment relationship also have overlapping supports in their *induced distributions over contexts* - if v entails w, then the support of $p(c|w)$ should be a superset of the support for $p(c|v)$. While these distributions can be expressed at simple histograms over discrete word occurrence matrices, they can also take more complex forms. Often, these distributions are embedded within a latent space, such as those of a space of word embeddings. Since Word2Vec is trained to encode cooccurence information, distributions within its latent space can achieve much the same effect as directly modelling $p(c|w)$ using empirical samples. Examples of these entailment properties using latent space distributions can be seen in Figures 2.5 and 2.6.

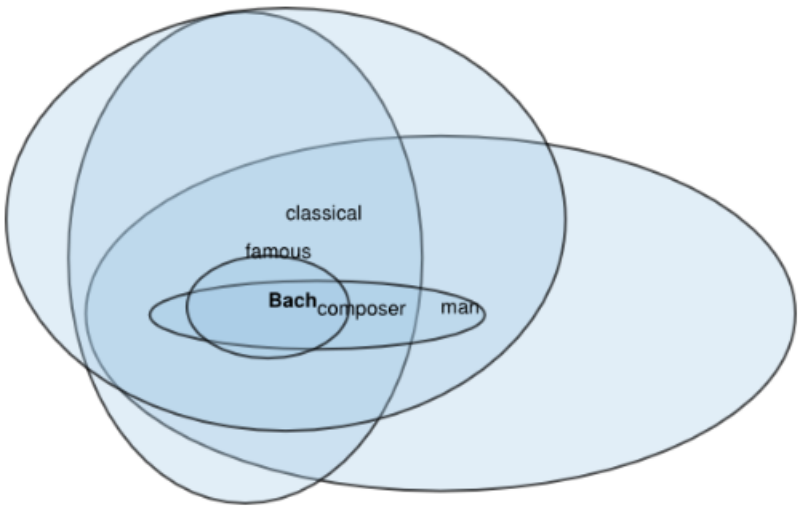


Figure 2.5: Gaussian embeddings of words from Vilnis and McCallum [2014], demonstrating how words with similar meanings could be modelled as distributions with overlapping support.

Many different authors have proposed research along these lines. Vilnis and McCallum [2014] proposed a system of probabilistic embeddings, wherein each word w in the vocabulary W was embedded not as a point vector, but as a gaussian distribution with mean μ_w and covariance Σ_w. They trained these representations using an algorithm highly similar to that of Word2Vec, save for a few small modifications. Vilnis & McCallum replaced the "energy function" $w^T c$ used by Word2Vec with a distribution-based similarity function - either using the inner product kernel $\int f(x)g(x)dx$, or the KL Divergence. Sun et al. [2018] later proposed an extension to this method involving a Wasserstein-based energy function $E(w, c) = -W_2(f_w, f_c) + b$, where W_2 was the 2-Wasserstein distance and b was an offset term used to scale the energy to an appropriate range. These probabilistic representations were evaluated on the task of unsupervised entailment on datasets such as those proposed by Baroni et al. [2012]. Similarity scores between words were determined by either the KL divergence between the distributions or the cosine distance between their means.

A more sophisticated approach to this was introduced by Singh et al. [2020], who defined what they refer to as the "context-mover's distance" between *distributional estimates* of words. They define this *distributional estimate* of a word w as a histogram of its probabilities $p(c|w)$ over contexts c, embedded in a latent space of context embeddings given by $V = (v_c)_{c \in C}$.

$$P_V^w = \sum_{c \in C} (H^w)_c \delta(v_c) \tag{2.18}$$

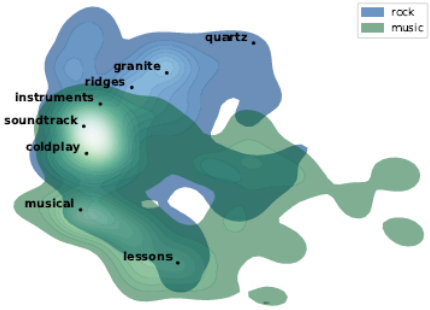


Figure 2.6: Visualizations of Singh et al's distributional estimates for the words "rock" and "music", taken from their paper. "Rock" has a mode that overlaps strongly with the distribution for music (i.e. the rock music as a genre) as well as modes that do not ("rock" as in a stone).

Given some metric D for the ground space of the embedding vectors, Singh et al then define their "context mover's distance" between two words as an optimal transport problem:

$$\text{CMD}(w_i, w_j; D) = \text{OT}(P_V^{w_i}, P_V^{w_j}; D) \tag{2.19}$$

Singh et al discuss how this distance measure can be used to measure entailment by using this context-mover's distance with the similarity operator $\ominus$ defined by Henderson and Popa [2016] as their ground metric: $D_{ij}^{Hend} = -v_i \ominus v_j$.

$$v_i \ominus v_j = \sigma(-v_i) \cdot \log \sigma(v_j) \tag{2.20}$$

Approximating Distances Between Distributions

Existing literature on learning or approximating functions between distributions is largely based on providing tractable methods for approximating specific distance functions such as KL Divergence or Mutual Information. These functions are often notoriously intractable, and require problems to be formulated in restrictive ways where they can be calculated in closed form in order to allow them to be used at all (see, for example, Kingma and Welling [2014]). When approximations are possible, they often use methods such as nearest-neighbours [Kraskov et al., 2004] rather than neural network models. These methods often suffer from troublesome scaling properties [Gao et al., 2015], and are not sufficient as general solutions to this problem.

Those neural networks models that do exist often rely on formulations such as those proposed by Donsker and Varadhan [1983], in which the distance calculation for a *particular pair* of distributions P, Q is formulated as an optimization problem over a class of functions. This class of

functions can then be approximated by a neural network, as was done in the widely-used "MINE" method [Belghazi et al., 2018] for approximating mutual information.

$$MI(X,Y) = \max_{\theta} \mathbb{E}_{X,Y\sim P_{XY}} T_\theta(X,Y) - \log \mathbb{E}_{X,Y\sim P_X\otimes P_Y} e^{T_\theta(X,Y)} \tag{2.21}$$

This approach is not helpful as a general solution to the approximation of divergences, however, as the trained network can be used only to calculate the divergence between the two distributions on which it was trained, and would need to be retrained from scratch in order to evaluate the divergence between any new pair of distributions P', Q'.

Chapter 3

Robust Word Embeddings

3.1 Introduction

This chapter is dedicated to further exploration of the problem defined in Sections 1.2.1 and 2.2.1: developing a probabilistic framework to address the problem of noise in natural language processing. As discussed previously, the majority of existing approaches each have their own significant limitations. Denoising-based approaches are often overconfident and delete important semantic information. Scratch-trained robust vectors can only be applied to certain models, as many models (including most pretrained transformer models) assume certain sets of ground embeddings. BPE-based tokenization removes the out-of-vocabulary problem without actually correcting the token, but allows small amounts of input noise to wildly distort how a given word is tokenized. Fundamentally, most of these problems stem from the fact that these approaches are deterministic, and thus do not model the uncertainty inherent in the problem.

Rather than follow any of these approaches, I propose a probabilistic schema that can be used in concert with any existing embedding vectors or form of tokenization, in order to improve performance on noisy corpora. This model explicitly models the uncertainty over possible meanings by modelling each token as a distribution over embedding vectors. The model combine two components: a token-level stream modelling possible corrections to the noisy token, and a context-level stream evaluating such corrections in the context of the surrounding sentence. I also introduce an 'ensembling' method to turn these distributions into inputs to downstream models in a way that maximizes the available information to the downstream model. This scheme requires no additional training and can be used in combination with any existing embedding methods. The approach is evaluated on several noisy text classification tasks against a number of simple baselines, as well as other robust vector methods and denoising methods from the literature.

3.2 Related work

3.2.1 Robust Methods

As discussed in Section 2.2.1, existing work primarily falls into one of three categories: denoising-based approaches, scratch-trained vectors, or BPE/subword tokenization.

Scratch-trained vector approaches are the most common of the three - at least among explicitly robust approaches. As mentioned previously, these approaches involve training bespoke embedding vectors using penalized losses to ensure lexically similar words are also embedded near each other in semantic space. These approaches are powerful, but cannot be used with models such as BERT or RoBERTa that assume a particular set of input embeddings - at least not without retrainin the entire model pipeline.

Denoising-based approaches include works such as Kernighan et al. [1990] and Sun and Jiang [2019] - as well as many other methods intended purely for spelling correction, which could in principle then be combined with an embedding pipeline. Some of these methods have similarities to my proposed approach, but differ in the application and interpretation. By making only a single correction, these models will often be overconfident - especially in difficult cases where the 'correct' correction is unclear. By using a probabilistic method that expresses distributions over possible 'corrected' sentences and training end to end on downstream tasks, the downstream model is provided with a broader range of information about the noisy sequence and can learn to use that information to make more informed predictions.

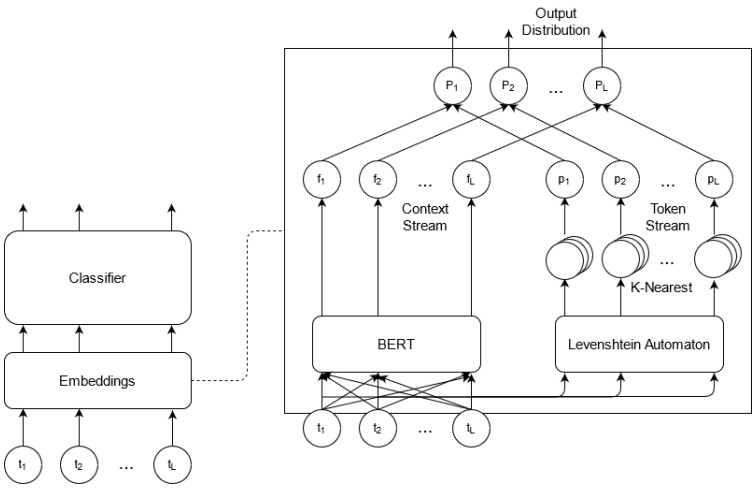


Figure 3.1: Diagram of the robust embedding model and how the output distribution is computed.

3.2.2 Probabilistic Word Embeddings

Many other works have sought to construct probabilistic representations of word embeddings. In particular, many works have used these methods to compensate for ambiguity in word meanings due to factors such as polysemy and homonymy. Section 1.1.3 discusses several of these works in detail - most notably the works of Tian et al. [2014], Liu et al. [2015] and Miao et al. [2019]. Vilnis and McCallum [2014] and Sun et al. [2018] also discuss how these probabilistic representations can be used to reflect uncertainty inherent in word meanings, and measure the similarity of different words. My proposed approach contains many similarities in motivation to these works, but seeks to address a different form of ambiguity: the ambiguity caused by the introduction of noise. These existing techniques are not sufficient to handle this particular case, as they are limited to operating only on words in a fixed dictionary, and have no methods for handling out of vocabulary words. Furthermore, it is not always obvious how to build NLP models for downstream tasks on top of those probabilistic embeddings.

3.3 RED

In order to address to shortcomings of existing approaches, any proposed solution to the problem of noise in natural language models must thus satisfy several criteria. First, such an approach must be tailored to address the fundamental uncertainty and ambiguity inherent to the problem - and thus, must approach the problem probabilistically. Second, an ideal approach should be compatible with a wide variety of existing models, and be usable as a drop-in replacement for existing forms of embedding or tokenization, without requiring retraining. I propose an approach called 'Robust Embeddings Via Distributions' (RED) which satisfies both of these criteria.

3.3.1 Robust Model

Fundamentally, RED is a schema of robust embeddings which represents each token in the noisy sequence as a distribution over many possible word vectors drawn from a given set of ground embeddings. This representation explicitly encodes the uncertainty in the meaning of the noisy token, while also being model-agnostic.

In order to construct this distribution for a particular misspelled word, consider how a human responds to seeing a misspelled word. Once the misspelling is identified, we naturally rely on two sources of information to resolve the misspelling: the characters within the word itself (and their similarity to other words we do know), and the surrounding context. Based on this intuition, I propose a model that is constructed from two streams: a token-stream, and a context-stream. These two streams will each make independent predictions about likely corrections, which will

then be unified to produce a final distribution over output embeddings (see Figure 3.2 for an example).

Suppose we have a sequence of noisy tokens $t_1, ..., t_i, ..., t_l$. In order to produce a robust embedding e_i for token t_i, we first consider the characters within the token t_i itself. Let $w_i^{(1)}, ..., w_i^{(K)} \in V$ be the K nearest words or tokens in our vocabulary V to the noisy token t_i under a particular string distance metric (e.g., the Levenshtein distance [Levenshtein, 1966]), with respective distances $d_i^{(1)}, ..., d_i^{(K)}$. Let $e(w)$ be a function mapping words to embedding vectors. Then, the first component of the model is given by:

$$p\left(e_i = e(w_i^{(j)})\right) = h\left(\vec{d_i}\right)_j \tag{3.1}$$

where h is some normalization function that maps the vector $\vec{d_i} = (d_i^{(1)}, ..., d_i^{(k)})$ of distances to normalized weights, such that larger distances correspond to smaller weights. For the purposes of this investigation we use the softmax function, as shown in Eq. 3.2:

$$h\left(\vec{d}\right) = \text{Softmax}\left(-\frac{\vec{d}}{\tau}\right) \tag{3.2}$$

where τ is a temperature parameter that controls the concentration of the distribution.

The second component of the distribution is a likelihood function that computes the probability of the surrounding context tokens $t_1, ..., t_{i-1}, t_{i+1}, ..., t_l$ given a particular center token. The most obvious model for this task is the famous skip-gram formulation of Word2Vec [Mikolov et al., 2013a], which trains to perform exactly this function. Unfortunately, skip-gram does not consider word order, and thus it evaluates the probability of each word occurring in the context independently. In practice, this ends up being overly naive, and does not lead to good predictions. Pre-trained transformer models such as BERT present an alternative to construct a more sophisticated likelihood function. Since BERT is trained using masked language modelling, it naturally computes a likelihood for the masked tokens in an input sequence conditioned on the rest of the sequence. Given a token t_i at position i in noisy sequence S (with context $S_{-i} = \{t_j; j \neq i\}$), consider the embedded sequence E_{-i} with t_i replaced by a mask token. BERT effectively computes:

$$h_{t_i} = [f_{BERT}(E_{-i})]_i \tag{3.3}$$

$$p_c(e_i = e|C_i) = \frac{\exp h_{t_i} \cdot e}{\sum_{w \in V} \exp h_{t_i} \cdot e(w)} \tag{3.4}$$

In order to combine the predictions of the two streams, we use a product-of-experts model, as discussed by Hinton [2002]. We can treat $p(e_i|t_i)$ and $f_{BERT}(e_i|E_{-i})$ as two expert models

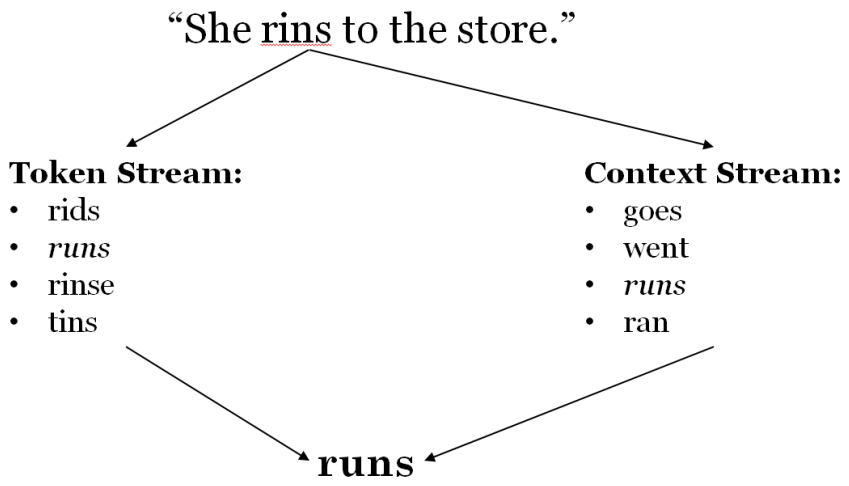


Figure 3.2: Motivating example of how the predictions from both streams are combined to generate the final output.

that make separate predictions based on different evidence. Hinton proposed to combine the probabilities of different experts simply by multiplying them and re-normalizing. To compute our final output distribution $p(e_i)$, we thus compute:

$$p(e_i|t_i, E_{-i}) \propto p_t(e_i|t_i)p_c(e_i|E_{-i}) \tag{3.5}$$

3.3.2 Ensembling

This now results in a method for obtaining a distribution over the set of ground embeddings which represents the robust embedding of the noisy token t_i given its surrounding context. In practice, however, our resulting embedding vectors will need to be fed into other deep models, which require a single vector as input rather than a distribution. There are several solutions to this. The naive solution is to simply sample a vector from the distribution or take the maximum a posteriori vector. This is perfectly adequate, but does not fully utilize the information contained in the distribution. Instead, we propose to use an ensembling method that more fully captures the full expressiveness of the distribution.

Suppose we have a sequence of noisy tokens $S = \{t_1, ..., t_l\}$ that we wish to embed before passing them into another downstream model (e.g., some sort of text classification model). Each token t_i and its associated context $S_{-i} = \{t_1, ..., t_{i-1}, t_{i+1}, ..., t_l\}$ are passed through the robust embedding model, with the intent of producing robust vector representations of each token respectively.

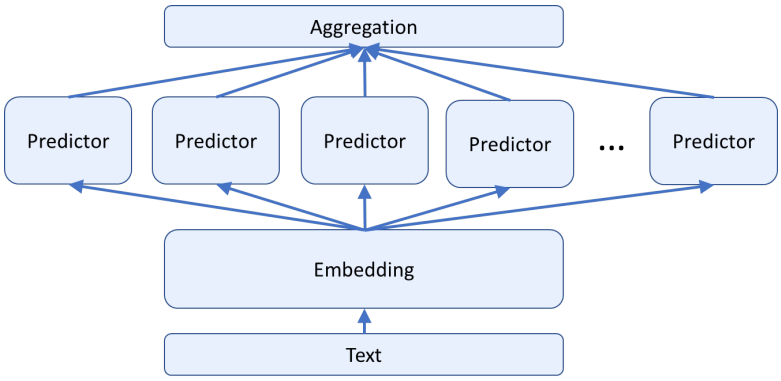


Figure 3.3: Ensembling model for robust embeddings. K sampled vectors for each token are organized into K sequences. These sequences are each passed through the classifier independently, and aggregated at the end into a single prediction.

Instead of producing single embedding vectors $e_1, ..., e_l$, the model instead samples M vectors from the posterior distribution for each token, resulting in ensembles $\{(e_1^{(1)}, ..., e_1^{(M)}), ..., (e_l^{(1)}, ..., e_l^{(M)})\}$. These ensembles of embeddings are then reorganized into an ensemble of embedded sequences $\{s^{(1)}, ..., s^{(M)}\}$, where $s^{(j)} = \{e_1^{(j)}, ...e_l^{(j)}\}$. Each sequence is then independently fed into the downstream model, producing M outputs $o_1, ...o_M$. These outputs can now be aggregated to produce a final consensus in several ways. The most obvious is to simply average the resulting logits, but other possibilities include taking a majority vote of the predicted classes or adding a shallow feedforward network at the end that could be trained to produce a final, aggregated prediction. In practice, it was found a simple average of the logits worked best, but this is a potential site for future exploration.

3.4 Experiments

This method can be evaluated on a variety of downstream tasks. For simplicity, and to focus primarily on the embedding method rather than the details of downstream architectures, we focus on several simple text classification tasks. We use a subset of the tasks from the GLUE text classification benchmark[1] for our experiments [Wang et al., 2018], covering tasks such as paraphrase detection, natural language inference, sentiment analysis, and several other forms of text classification. For each task, we train two different classifiers: a BiLSTM-based approach known as HBMP [Talman et al., 2019], as well as the state of the art pretrained transformer model "RoBERTa" [Liu et al., 2019b]. Evaluation is performed on the test set for MRPC, and

[1] All details and download links can be found at https://gluebenchmark.com/

	RoBERTa					HBMP				
		20%		50%			20%		50%	
Model	Clean	Synth.	Natural	Synth.	Natural	Clean	Synth.	Natural	Synth.	Natural
Naive	**0.890**↑	0.797↓	0.826↓	0.606↓	0.690↓	0.807↑	0.698↓	0.725↓	0.545↓	0.608↓
MOE	x	x	x	x	x	0.795	0.637↓	0.707↓	0.493↓	0.596↓
B2V	x	x	x	x	x	**0.809**↑	0.655↓	0.714↓	0.480↓	0.575↓
Sun&Jiang	0.872↓	0.844↓	0.856↓	0.775↓	0.813↓	0.792	0.757↓	0.770↓	0.681↓	0.724↓
RobEn	0.828↓	0.765↓	0.786↓	0.652↓	0.718↓	0.766↓	0.677↓	0.709↓	0.538↓	0.615↓
RED	0.879	0.860↓	0.866↓	0.822↓	0.836↓	0.796	0.777	0.781	0.734↓	0.748↓
RED-Ens	0.880	**0.862**	**0.869**	**0.833**	**0.843**	0.789	**0.778**	**0.783**	**0.739**	**0.752**

Table 3.1: Scores averaged across selected GLUE tasks with RoBERTa and HBMP classifiers with clean training. An x indicates a method that is not compatible with RoBERTa. ↑ and ↓ signify results that are statistically better or worse respectively than RED-Ensemble with $p < 0.05$ according to the Wilcoxon signed rank test [Wilcoxon, 1945].

the development set for all other datasets (as public test sets are not available), with 20% of the training set withheld as a substitute development set. Several different versions of each evaluation set are created: one that is unaltered and has no noise added, as well as versions that have noise injected with either natural or synthetic noise methods. The noisy corpora are generated by randomly selecting words to have noise injected with a fixed probability of either 20% or 50%. Synthetic noise is injected using the method presented in [Sun and Jiang, 2019]: if a word is selected to have noise injected, there is a 25% chance each of either deleting a character at a random location, inserting a random character at a random location, swapping the characters at two random locations, or replacing a character at a random location with another random character. Natural noise is injected using a dataset of misspellings harvested from social media (we use the dictionary published by Piktus et al. [2019]). A word selected to have natural noise injected will be replaced by a random misspelling for that word taken from the dataset. We show results with noise applied both a) during training and evaluation and b) during evaluation only.

We compare our method against several baselines. First, we compare against the relatively naive baselines of either a) simply using the downstream model with no additional attempt to improve the robustness, or b) correcting each out-of-vocabulary word to the nearest word under the Levenshtein distance. We also compare against several baselines from the literature. For experiments using the RoBERTa classifier, we compare to the denoising method of Sun and Jiang [2019] as well as the robust encoding method "RobEn" [Jones et al., 2020]. For the HBMP classifier, we also compare against scratch-trained vector methods such as Bridge2Vec [Doval et al., 2019] and MOE [Piktus et al., 2019]. Unfortunately, it is not practical to compare against these methods in the case of RoBERTa, as replacing the input embedding layer of RoBERTa would require retraining the entire model. Our model does not suffer from this issue since it can output embeddings from any predefined embedding space and therefore can be inserted below

	RoBERTa				HBMP			
	20%		50%		20%		50%	
Model	Synth.	Natural	Synth.	Natural	Synth.	Natural	Synth.	Natural
Naive	0.842↓	0.859↓	0.732↓	0.805↓	0.748↓	0.764↓	0.678↓	0.714↓
MOE	x	x	x	x	0.731↓	0.752↓	0.675↓	0.712↓t
B2V	x	x	x	x	0.754↓	0.768↓	0.698↓	0.725↓
Sun&Jiang	0.853↓	0.860↓	0.797↓	0.833↓	0.767↓	0.777↓	0.731↓	0.752↓
RobEn	0.793↓	0.802↓	0.744↓	0.758↓	0.731↓	0.742↓	0.690↓	0.717↓
RED	0.868	**0.871**	0.840↓	**0.852**	**0.782**	**0.787**	0.759↓	0.767↓
RED-Ens	**0.869**	**0.871**	**0.852**	0.849	0.777	**0.787**	**0.764**	**0.772**

Table 3.2: Scores averaged across selected GLUE tasks with RoBERTa and HBMP classifiers with noise applied during training time. An x indicates a method that is not compatible with RoBERTa. ↑ and ↓ signify results that are statistically better or worse respectively than RED-Ensemble with $p < 0.05$ according to the Wilcoxon signed rank test [Wilcoxon, 1945].

any pretrained model such as RoBERTa without requiring retraining.

3.5 Results

Experiments were performed using Nvidia T4 and P100 GPUs. FastText vectors were used as the ground embeddings for the HBMP classifier for the Naive method, our method, and all spellchecking/denoising-based methods. Implementation details for the HBMP classifier were taken from their publicly-available code[2], with 400-dimensional hidden layers and default hyperparameter settings. Implementation details for the RoBERTa models and GLUE tasks were taken from the HuggingFace repository[3] [Wolf et al., 2020], with default hyperparameter settings. Implementations of baselines were taken from publicly available source code where possible, and otherwise were re-implemented based on descriptions of the methods. Further details on the experimental setup and baseline implementations are contained in the appendices. For the robust model, the softmax temperature parameter in the token stream was taken to be $\tau = 0.15$, with the top $K = 20$ nearest words under the Levenshtein distance considered, and the ensemble models used $M = 10$ samples each. We use the likelihood function described in Equations 3.3 and 3.4, with the base uncased pretrained BERT model published by HuggingFace. Each experiment was performed ten times, and the results were averaged across the trials. In order to determine statistical significance, the Wilcoxon Signed Rank Test [Wilcoxon, 1945] was used to compare results across the ten trials for each experiment category between different baselines. Table 3.1

[2]https://github.com/Helsinki-NLP/HBMP
[3]https://github.com/huggingface/transformers

		20%		50%	
Model	Clean	Synthetic	Natural	Synthetic	Natural
Token Stream Only	**0.884**↑	0.837↓	0.847↓	0.769↓	0.789↓
Context Stream Only	0.868↓	0.781↓	0.807↓	0.582↓	0.664↓
RED (No Ensemble)	0.879	0.860↓	0.866↓	0.822↓	0.836↓
RED (Ensemble)	0.880	**0.862**	**0.869**	**0.833**	**0.843**

Table 3.3: Results of ablation study on RED components, averaged across the five GLUE tasks. ↑ and ↓ signify results that are statistically better or worse respectively than RED-Ensemble with $p < 0.05$ according to the Wilcoxon signed rank test [Wilcoxon, 1945].

show results of the ten trials for each baseline and noise setting averaged across all trials and all tasks (specifically, the MPRC, SST-2, QNLI, MNLI and QQP tasks from the GLUE benchmark) with no noise applied during training. Table 3.2 shows results when we repeat the experiments while including noise during training. Results marked with a ↓ are statistically worse than RED-Ensemble with $p < 0.05$. Results marked with a ↑ are statistically better than RED-Ensemble with $p < 0.05$. In keeping with standard practice for the GLUE benchmark, we report accuracy for SST-2, MNLI and QNLI, and report an unweighted average of accuracy and F1 score for MRPC and QQP. Full results broken down by individual dataset are also included in Tables 3.8 and 3.9 at the end of this chapter.

3.5.1 Clean Training

The two RED models perform the best (by statistically significant margins) across all datasets and noise settings except for the completely clean case. The model with ensembling outperforms the base RED model on all the high-noise settings across all datasets, and most of the low-noise settings as well. The benefits of ensembling seem highly dependent on the specific task, with the ensemble method being clearly superior on MRPC, SST-2 and QQP, and comparable or slightly superior on MNLI and QNLI. Ensembling also seems to provide a larger boost on the higher noise cases than the lower noise cases in general. The slight decrease in performance compared to the naive baseline on clean data is expected due to the nature of robust methods. As discussed in Tsipras et al. [2019], there is an inherent tradeoff between robustness and accuracy in the completely clean case. A non-robust model can fit more exactly the input data at the cost of becoming extremely brittle to noise and perturbations. Robust models, in contrast, learn representations that are more robust to noise and more similar to human evaluation, but may suffer a small loss of accuracy in the clean case by modeling the possibility of errors that are not there.

Overall, the ensemble model improves performance by larger margins with RoBERTa than

Method	Text
Original Sequence:	Isn't a woman's body her most personal property?
Noisy Sequence:	sIn't a woman's boyd ther kmost persopal property?
Token-Stream:	sin't a woman's boyd their most personal property ?
Context-Stream:	sin't a woman's boyd - a " property ?
RED:	isn't a woman's body the most personal property ?
Sun & Jiang:	sin't a woman's bond the most personal property ?
Original Sequence:	fun , flip and terribly hip bit of cinematic entertainment.
Noisy Sequence	funu, flip land terribly hip bit of cqnematic entertainmenth
Token-Stream	fun , flip land terribly hip bit of cinematic entertainment
Context-Stream	also , flip land terribly hip bit of the .
RED	fun , flip and terribly hip bit of cinematic entertainment
Sun & Jiang	funk , flip land triple hip bit of kinetic entertainment

Table 3.4: Comparison against other baselines as a denoising method on example noisy sequences.

Word	Token Stream	Context Stream	Product	Final Probability
kids	-6.677	-4.292	-10.969	0.719
lids	-0.010	-12.132	-12.142	0.222
lips	-6.677	-7.110	-13.787	0.043
lads	-6.677	-8.589	-15.266	0.010
lies	-6.677	-9.378	-16.054	0.004

Table 3.5: An expanded version of the first table from Fig. 3.4 showing the contributions of each stream to the final distribution over embeddings for the noisy token 'lids'. Values shown are log-scale, save for the final column.

HBMP. We hypothesize that since this classifier is less powerful than RoBERTa, it is less able to fully leverage the information provided by the ensemble, and thus the benefit is less significant.

3.5.2 Noisy Training

The two RED models again outperform all other baselines by statistically significant margins save for the completely clean case. The benefit of ensembling is slightly lower in this setting, especially in the low noise case. Once again, this varies greatly with the dataset, with ensembling providing a more significant boost on MRPC, QQP and SST-2, and being less beneficial on QNLI and MNLI.

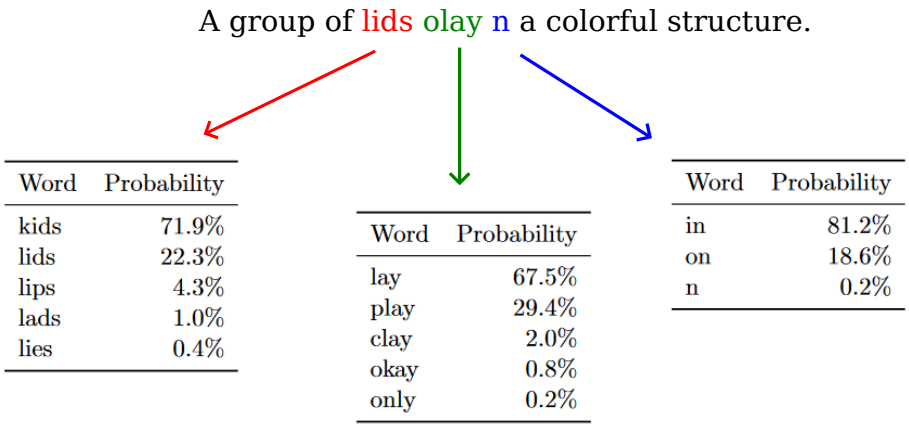


Figure 3.4: Example of the RED model applied to a sample noisy sentence. Tables show the probability distributions over possible embeddings for each of three key tokens in the sentence. Results are shown for all candidate words with probability $> 0.1\%$.

3.6 Analysis

3.6.1 Ablation Study

To compare the effects of the different streams of the model, we performed an ablation study. Experiments were performed on all datasets showing the results of a) RED with the ensemble model, b) RED without the ensemble, c) only using the token-level stream, and d) only using the context-level stream. Results for the RoBERTa classifier averaged across all datasets are shown in Table 3.3 (see Appendices for full results). Using only context-level information gives very poor results, especially on higher noise levels. The token-level stream performs better, but still suffers compared to the full RED model - especially on higher noise levels.

3.6.2 Examples

In order to illustrate the workings of the model, we will now highlight some examples of the model applied to some sample noisy sequences from the data. We can thus compare the RED model to some of the other baselines from the viewpoint of text correction, and show some examples of how the different components of the model contribute.

In Table 3.4, we compare our model against the Sun & Jiang baseline on some difficult noisy sequences taken from the data, as well as showing the predictions made by the separate components of the model. Even in cases where the sequences are very noisy, our model can often generate reasonable corrections where the other models cannot. In the second example, we

Samples
A group of kids play in a colorful structure.
A group of kids play in a colorful structure.
A group of lips lay on a colorful structure.
A group of kids lay on a colorful structure.
A group of kids play in a colorful structure.

Table 3.6: 5 ensemble samples from RED for the sentence in Figure 3.4.

can see how the token-stream model can generate some reasonable corrections, but cannot fully reconstruct the sequence without the added benefit of the context stream to handle the cases where a typo is also a valid word.

To explore this in more detail, consider the example shown in Figure 3.4. This figure demonstrates the action of the RED model on each of three tokens in the noisy sentence "A group of lids olay n a colorful structure". The tables show the top candidates in the output distributions generated by RED at each position in the sequence (truncated to show only results with probability $> 0.1\%$). An expanded version of the first table is also shown in Table 3.5, which shows the contributions of each stream on a log scale, as well as the final unnormalized logits and log probabilities. As shown in the tables, this noisy sentence contains significant ambiguity in several tokens - e.g. do the kids "lay on" the structure or "play in" it? This situation is an excellent example of the utility of the ensemble-based approach; rather than simply making a single choice for how to embed this sentence, the downstream model can be presented with a number of choices, such as the samples shown in Table 3.6. This allows the model to present multiple possibilities for ambiguous tokens, and increases the chance that some of the samples will be correct for highly noisy sequences. By combining different possible sampled sequences, the downstream model has access to more information and can make better predictions.

3.6.3 Computational Cost

One possible concern with the ensembling approach is that it might lead to a prohibitive additional computational cost. In fact, this is not a huge concern. When performing experiments with the robust models, the computations are done in two stages: denoising/embedding, and training. In the first stage, the denoised/robustly embedded representations are computed for every sequence in the dataset. In the second, these representations are then used to train the downstream classifier. Table 3.7 shows the average time (in minutes) required to perform each stage for each of the models on the MRPC and SST-2 datasets with the RoBERTA classifier. As these tables show, the denoising stage is usually significantly more expensive than the training stage, but ensembling does not necessarily yield a higher cost in this portion and when there is a higher cost

	MRPC			SST-2		
Model	Denoise Time	Train Time	Total Time	Denoise Time	Train Time	Total Time
Naive	0.097	2.741	2.837	0.092	29.474	29.566
Roben	0.204	1.334	1.538	0.092	38.550	38.641
Sun & Jiang	219.512	1.283	220.795	301.713	25.521	327.235
RED	66.536	1.946	68.481	224.683	20.820	245.503
RED-Ensemble	67.892	10.032	77.924	190.460	99.302	289.763

Table 3.7: Average time (in minutes) for each stage of computation for each model on the MRPC and SST-2 datasets using the RoBERTA classifier.

it is not significant. Ensembling increases the training time considerably, but this does not lead to a prohibitively higher total time. In theory, ensembling M embeddings could slow down training by a factor of M, but since GPUs parallelize part of the computation for those M embeddings, the slow down is typically less. While M is 10 in the experiments, the observed slow down in Table 3.7 is closer to a factor of 5.

3.6.4 Hyperparameter Analysis

This method makes use of three important hyperparameters: the softmax temperature τ, the K value in the top-k operation in the token stream, and M, the number of samples used for the ensemble. Experiments were performed to test optimal hyperparameter values, and it was found that values of $\tau = 0.15$, $K = 20$, and $M = 10$ worked well for all experiments. See the appendices for further details and figures.

3.7 Conclusion

In conclusion, this chapter proposes a novel approach to robustness at the embedding level that is probabilistic and transferable. This method can be used as a foundation for any other NLP model in order to make that model more robust, without requiring any changes to the architecture. My approach is probabilistic, and thus more fully expresses the uncertainty in meaning within a noisy sequence than other similar models. Using the ensembling approach, the downstream model can have access to all (or at least many) possible meanings of the noisy tokens, and can be trained end to end to decide for itself how to use this information - rather than making overconfident corrections at the preprocessing level. The proposed model demonstrates superior results on a range of noisy tasks, using both synthetic and natural noise.

	RoBERTa					HBMP				
		20%		50%			20%		50%	
Model	Clean	Synth.	Natural	Synth.	Natural	Clean	Synth.	Natural	Synth.	Natural
					MRPC					
Naive	0.868↓	0.764↓	0.801↓	0.368↓	0.523↓	0.799	0.640↓	0.679↓	0.332↓	0.463↓
MOE	x	x	x	x	x	0.769↓	0.568↓	0.680↓	0.388↓	0.567↓
B2V	x	x	x	x	x	0.798	0.541↓	0.643↓	0.198↓	0.361↓
Sun&Jiang	0.864↓	0.857↓	0.855↓	0.773↓	0.819↓	0.793↓	0.756↓	0.773↓	0.642↓	0.707↓
RobEn	0.843↓	0.793↓	0.812↓	0.610↓	0.719↓	0.791↓	0.690↓	0.719↓	0.450↓	0.568↓
RED	0.872↓	0.866↓	0.867↓	0.842↓	0.847↓	0.798	0.775	0.783	0.713↓	0.742
RED-Ens	**0.879**	**0.872**	**0.873**	**0.853**	**0.854**	**0.800**	**0.780**	**0.786**	**0.723**	**0.749**
					QNLI					
Naive	**0.899**↑	0.831↓	0.861↓	0.721↓	0.782↓	**0.804**↑	0.742↓	0.762↓	0.643↓	0.692↓
MOE	x	x	x	x	x	0.788	0.693↓	0.740↓	0.606↓	0.675↓
B2V	x	x	x	x	x	0.808↑	0.736↓	0.764↓	0.642↓	0.706↓
Sun&Jiang	0.887	0.866↓	0.876↓	0.812↓	0.847↓	0.796	0.780↓	0.783↓	0.739↓	0.761↓
RobEn	0.840↓	0.795↓	0.816↓	0.722↓	0.774↓	0.778↓	0.741↓	0.756↓	0.687↓	0.720↓
RED	0.891	**0.879**	0.883	0.849↓	0.863↓	0.800	**0.793**	**0.795**↑	0.773↓	0.777
RED-Ens	0.889	0.877	**0.883**	**0.856**	**0.869**	0.767	0.792	0.794	**0.777**	**0.781**
					QQP					
Naive	**0.890**↑	0.751↓	0.797↓	0.540↓	0.684↓	0.852↑	0.679↓	0.730↓	0.521↓	0.614↓
MOE	x	x	x	x	x	0.853↑	0.575↓	0.706↓	0.356↓	0.535↓
B2V	x	x	x	x	x	**0.858**↑	0.586↓	0.709↓	0.360↓	0.536↓
Sun&Jiang	0.867↓	0.825↓	0.845↓	0.750↓	0.802↓	0.837↓	0.781↓	0.806↓	0.674↓	0.746↓
RobEn	0.858↓	0.750↓	0.800↓	0.582↓	0.714↓	0.837↓	0.691↓	0.751↓	0.515↓	0.631↓
RED	0.873	0.842↓	0.855↓	0.789↓	0.817↓	0.842	0.809	0.818↓	**0.741**	0.770↓
RED-Ens	0.877	**0.847**	**0.858**	**0.794**	**0.823**	0.843	**0.809**	**0.822**	0.739	**0.772**
					SST-2					
Naive	**0.934**↑	0.898	0.897↓	0.818↓	0.809↓	**0.860**↑	0.814↓	0.811↓	0.730↓	0.726↓
MOE	x	x	x	x	x	0.856↑	0.779↓	0.790↓	0.665↓	0.684↓
B2V	x	x	x	x	x	0.862↑	0.818	0.821	0.736↓	0.744↓
Sun&Jiang	0.908↓	0.885↓	0.893↓	0.833↓	0.848↓	0.824↓	0.802↓	0.806↓	0.753↓	0.772↓
RobEn	0.839↓	0.809↓	0.806↓	0.770↓	0.766↓	0.780↓	0.743↓	0.748↓	0.699↓	0.701↓
RED	0.916	0.901	0.901↓	0.884↓	0.878↓	0.825	0.820	0.819↓	0.802↓	0.799
RED-Ens	0.913	**0.903**	**0.908**	**0.895**	**0.886**	0.829	**0.823**	**0.825**	**0.814**	**0.806**
					MNLI					
Naive	**0.859**↑	0.742↓	0.772↓	0.584↓	0.650↓	**0.722**↑	0.618↓	0.642↓	0.497↓	0.545↓
MOE	x	x	x	x	x	0.709	0.568↓	0.620↓	0.449↓	0.518↓
B2V	x	x	x	x	x	0.721↑	0.591↓	0.634↓	0.466↓	0.530↓
Sun&Jiang	0.833↓	0.790↓	0.809↓	0.709↓	0.751↓	0.708	0.667↓	0.683↓	0.595↓	0.632↓
Roben	0.761↓	0.676↓	0.698↓	0.573↓	0.617↓	0.657↓	0.586↓	0.609↓	0.499↓	0.542↓
RED	0.843↑	**0.814**	**0.823**	0.748↓	0.777↓	0.713↑	**0.689**↑	**0.692**	**0.640**	**0.653**↑
RED-Ens	0.840	0.813	0.822	**0.768**	**0.781**	0.707	0.685	0.690	**0.640**	0.650

Table 3.8: Results of baselines by task with clean training. An x indicates a method that is not compatible with RoBERTa. ↑ and ↓ signify results that are statistically better or worse respectively than RED-Ensemble with $p < 0.05$ according to the Wilcoxon signed rank test [Wilcoxon, 1945].

	RoBERTa				HBMP			
	20%		50%		20%		50%	
Model	Synth.	Natural	Synth.	Natural	Synth.	Natural	Synth.	Natural
				MRPC				
Naive	0.819↓	0.836↓	0.749↓	0.766↓	0.760↓	0.768↓	0.726↓	0.745↓
MOE	x	x	x	x	0.722↓	0.734↓	0.709↓	0.724↓
B2V	x	x	x	x	0.762↓	0.765↓	0.73↓5	0.742↓
Sun&Jiang	0.859↓	0.860↓	0.823↓	0.843↓	0.782↓	0.786↓	0.757↓	0.771↓
RobEn	0.814↓	0.823↓	0.765↓	0.795↓	0.761↓	0.765↓	0.731↓	0.752↓
RED	0.866	0.866↓	0.852↓	0.852↓	0.787↓	0.791	0.773↓	0.782↓
RED-Ens	**0.869**	**0.871**	**0.860**	**0.860**	**0.793**	**0.793**	**0.784**	**0.788**
				QNLI				
Naive	0.857↓	0.870↓	0.782↓	0.815↓	0.743↓	0.762↓	0.652↓	0.706↓
MOE	x	x	x	x	0.713↓	0.741↓	0.631↓	0.690↓
B2V	x	x	x	x	0.753↓	0.771↓	0.683↓	0.702↓
Sun&Jiang	0.869↓	0.876↓	0.830↓	0.853↓	0.774↓	0.781↓	0.735↓	0.760↓
RobEn	0.805↓	0.818↓	0.766↓	0.792↓	0.745↓	0.759↓	0.701↓	0.734↓
RED	**0.885**↑	**0.887**	**0.867**	0.872	**0.788**	**0.791**	0.768↓	0.776↓
RED-Ens	0.881	0.885	0.865	**0.874**	0.759	**0.791**	**0.772**	**0.781**
				QQP				
Naive	0.859↓	**0.869**↑	0.823↓	0.847↓	0.777↓	0.805↓	0.712↓	0.774↓
MOE	x	x	x	x	0.776↓	0.809↓	0.721↓	0.781↓
B2V	x	x	x	x	0.788↓	0.817↓	0.733↓	0.788↓
Sun&Jiang	0.844↓	0.854↓	0.812↓	0.835↓	0.801↓	0.814↓	0.761↓	0.791↓
RobEn	0.822↓	0.837↓	0.786↓	0.815↓	0.780↓	0.803↓	0.733↓	0.779↓
RED	0.860	0.865↓	0.841↓	0.852↓	0.818↓	0.825↓	0.787↓	0.807↓
RED-Ens	**0.861**	0.867	**0.844**	**0.857**	**0.821**	**0.829**	**0.794**	**0.813**
				SST-2				
Naive	0.904↓	0.902↓	0.854↓	0.842↓	0.818↓	0.821↓	0.752↓	0.750↓
MOE	x	x	x	x	0.806↓	0.817↓	0.755↓	0.758↓
B2V	x	x	x	x	0.820↓	0.821↓	0.772↓	0.777↓
Sun&Jiang	0.887↓	0.893↓	0.850↓	0.858↓	0.803↓	0.815↓	0.772↓	0.784↓
RobEn	0.806↓	0.811↓	0.782↓	0.775↓	0.752↓	0.755↓	0.719↓	0.726↓
RED	0.905↓	0.907	0.889	0.884	0.826	**0.831**	0.808	0.803↓
RED-Ens	**0.911**	**0.908**	**0.899**	**0.894**	**0.828**	0.830	**0.816**	**0.814**
				MNLI				
Naive	0.769↓	0.818↓	0.450↓	0.754↓	0.643↓	0.661↓	0.550↓	0.595↓
MOE	x	x	x	x	0.6348↓	0.6577↓	0.5588↓	0.6058↓
B2V	x	x	x	x	0.6450↓	0.6684↓	0.5675↓	0.6157↓
Sun&Jiang	0.805↓	0.817↓	0.669↓	0.778↓	0.676↓	0.686↓	0.628↓	0.652↓
Roben	0.716↓	0.724↓	0.619↓	0.610↓	0.614↓	0.628↓	0.564↓	0.594↓
RED	**0.825**↑	**0.828**↑	0.748	**0.802**	**0.691**↑	**0.696**↑	**0.661**↑	**0.667**
RED-Ens	0.821	0.824	**0.794**	0.759	0.685	0.691	0.656	0.666

Table 3.9: Results of baselines by task with with noise applied during training time. An x indicates a method that is not compatible with RoBERTa. ↑ and ↓ signify results that are statistically better or worse respectively than RED-Ensemble with $p < 0.05$ according to the Wilcoxon signed rank test [Wilcoxon, 1945].

Chapter 4

Multi-Set Transformers

4.1 Introduction

The second major application of distributional approaches explored in this work is the use of neural network models to approximate distance functions between distributions. Section 1.1.2 provides a motivating example of how such models could be used to train bespoke distance functions between empirical distributions over word embeddings for unsupervised inference. As discussed in Section 1.2.2, however, the applications of this approach go far beyond this initial motivating problem. More generally, this concept of training models to approximate distance functions between distributions could be used to model troublesome quantities such as Kullback-Leibler Divergence and Mutual Information. These quantities are highly useful in many applications, but notoriously difficult to approximate by conventional means - to the extent that problems are often structured in cumbersome ways to allow them to be computed in closed form.

The primary challenge with this idea is that it requires a model architecture which can be used to predict functions defined on *distributions* - or, at the very least, unordered sets of samples from these distributions. Unfortunately, typical deep learning algorithms are constrained to operate on either fixed-dimensional vectors or ordered sequences of such vectors. While some investigation has now been done into the problem of applying deep learning to functions on sets [Lee et al., 2019, Zaheer et al., 2017], these works all focus on the problem of learning a function on a *single* input set, and do not generalize to the case of multiple sets.

This chapter will detail a novel architecture to do just that: provide a framework to approximate functions defined on multiple permutation invariant sets. I will define the properties of partial permutation invariance and -equivariance that this architecture must satisfy, detail how this architecture can be constructed from modified attention blocks based on the work of Lee et al. [2019], and prove that the proposed architecture is a universal approximator of all partially

permutation invariant or equivariant functions on multiple sets of inputs. I will demonstrate the proposed model's superiority to existing works and naive extensions of single-set architectures across a wide range of applications, including predicting alignment between multiple sets of samples from different domains (for example, images and their captions), or determining whether two sets of samples were drawn from the same distribution (a task whose significance will be explored in further detail in Chapter 5). I will also highlight the particular application of approximating distance functions between distributions by demonstrating the proposed model's effectiveness at approximating the notoriously intractable quantities of KL Divergence and Mutual Information - surpassing the results of widely used existing models such as MINE [Belghazi et al., 2018].

4.2 Related Work

This method is based on the work of Zaheer et al. [2017], who originally drew attention to the problem of using neural networks to approximate functions on permutation-invariant sets. In particular, it draws from the work of Lee et al. [2019], who extended this work to use transformer-based models on sets.

Gui et al. [2021] also address the idea of designing neural networks to learn functions between multiple permutation-invariant sets. Their work, however, focuses on graph embeddings as a primary application, and does not consider estimating distances or divergences. Their method also relies on a more simplistic architecture that has been criticized in Wagstaff et al. [2019], whereas our proposed architecture has a number of theoretical and empirical advantages.

4.3 Method

4.3.1 Background

Consider first the problem of learning a function upon a single set $X = \{x_1, ..., x_n\}$ in $\mathbb{R}^d$. As discussed in the work of Zaheer et al. [2017], this general problem takes on one of two forms: the *permutation-equivariant* and *-invariant* cases. In the permutation-equivariant case, the function takes the form $f : 2^{\mathbb{R}^d} \to 2^{\mathbb{R}^{d'}}$, and must obey the restriction that permutations of the inputs correspond to identical permutations of the outputs - i.e. for a permutation π,

$$f(\pi(X)) = \pi(f(X)) \tag{4.1}$$

In the permutation-invariant case, the function takes the form $f : 2^{\mathbb{R}^d} \to \mathbb{R}^{d'}$, and must obey the restriction that permutations of the inputs correspond to no change in the output - i.e.

$$f(\pi(X)) = f(X) \tag{4.2}$$

Zaheer et al. [2017] proposed an architecture to learn such functions that we will refer to as the *sum-decomposition* architecture (following Wagstaff et al. [2019]). This architecture proposes to learn permutation-invariant functions using the model:

$$f(X) = \rho\left(\sum_{x \in X} \phi(x)\right) \tag{4.3}$$

Each member of the set x is encoded into a latent representation $\phi(x)$, which are then summed and decoded to produce an output. While this architecture is adequate for some purposes, it is also very simple, and has difficulty modeling interactions between multiple elements in the set. In particular, Wagstaff et al. [2019] proved in their paper that *sum-decomposable* architectures such as this require each element to be mapped to a latent vector with latent size at least as large as the maximum number of elements in the input set in order to be universal approximators of functions on sets. Practically speaking, this is a major restriction when working with large sets, because this introduces a hard maximum on the size of the input set for a given latent size. Qi et al. [2017] also propose a very similar architecture to this in their paper on PointNets. While PointNets use a more complicated encoder architecture and a different form of pooling (sum vs max pool) are different, they follow the same general structure of applying a deep elementwise encoding, followed by a final pooling operation.

Lee et al. [2019] improved on this architecture in their paper, proposing an architecture they referred to as 'Set Transformers'. This architecture leverages the fact that the widely-used attention operation (discussed in Section 2.1.2) is itself permutation-equivariant. This means that transformer encoder blocks (as defined in Eq. 2.10) that do not contain positional encodings can be used as the basis for a permutation-equivariant encoder. Their proposed architecture consists of a stack of transformer encoder blocks $\{T_1, ..., T_L\}$, followed by a pooling operation Γ (by default, their proposed 'Pooling by Multiheaded Attention' operator) and a feedforward decoder ρ.

$$f(X) = \rho\left(\prod_{X} T_L(T_{L-1}(...T_1(X))\right) \tag{4.4}$$

The Set Transformer architecture bears some similarities to another model proposed by Santoro et al. [2017]. While these networks do not utilize attention, they also involve elementwise comparisons between pairs of elements from the input set, combined with pooling and decoding operations. Relation networks are defined by the model:

$$f(X) = \rho\left(\sum_{x_i \in X} \sum_{x_j \in X} \theta(x_i, x_j)\right) \tag{4.5}$$

wherein ρ is again a decoder, and θ is a feedforward *pairwise* encoder which encodes the relationship between each pair of elements in X.

General Model

In general, we can consider all three of these architectures to consist of an equivariant encoder ϕ on the set X, followed by a pooling operation Γ and decoder ρ:

$$f(X) = \rho\left(\prod_X \phi(X, X)\right) \tag{4.6}$$

In both the Set Transformer and Relation Network case, this general architecture may be specified even further. Each of these cases consist of encoders which compute pairwise operations on the elements x_i, x_j. In these cases, $\phi(X)$ can be written more explicitly as:

$$\phi(X) = \bigwedge_X \theta(X, X) \tag{4.7}$$

where $\theta(X, X)$ is a pairwise encoder that computes an encoding of each pair (x_i, x_j), then Λ is a form of pooling operation that reduces this $N \times N$ encoding matrix row-wise into a single vector for each element in X. For the base relation network architecture, θ is simply a feedforward neural net, and both pooling operations take the form of sums. For the set transformer architecture, $\phi(X)$ is the multiheaded self-attention operator, with $\theta(X, X)$ being the dot product of the transformed queries and keys and Λ consisting of a softmax and matrix multiplication by the transformed value matrix. The pooling operator Γ in this case is given by the 'Pooling by Multiheaded Attention' (PMA) operator defined in Lee et al. [2019] (though a sum or max pool could also be used).

4.3.2 Multiple Sets

I propose to extend these methods to the case of multiple permutation invariant sets - which I refer to as the case of *partial permutation invariance*. A function $f : 2^{\mathbb{R}^d} \times 2^{\mathbb{R}^d} \to \mathbb{R}^{d'}$ is *partially permutation invariant* if $\forall \pi_1, \pi_2$ it obeys the property:

$$f(\pi_1(X), \pi_2(Y)) = f(X, Y)$$

Similarly, a function $f : 2^{\mathbb{R}^d} \times 2^{\mathbb{R}^d} \to 2^{\mathbb{R}^d} \times 2^{\mathbb{R}^d}$ is *partially permutation equivariant* if it obeys the property:

$$f(\pi_1(X), \pi_2(Y)) = (\pi_1(f_X(X, Y)), \pi_2(f_Y(X, Y)))$$

Note that these definitions are straightforward extensions of the base concepts of permutation invariance and equivariance under a direct product of groups. The definitions in Equations 4.1 and 4.2 can be considered as the invariance or equivariance of the function f under the action of the symmetric group $\mathbb{S}_n$. Similarly, *partial* permutation invariance or equivariance can be

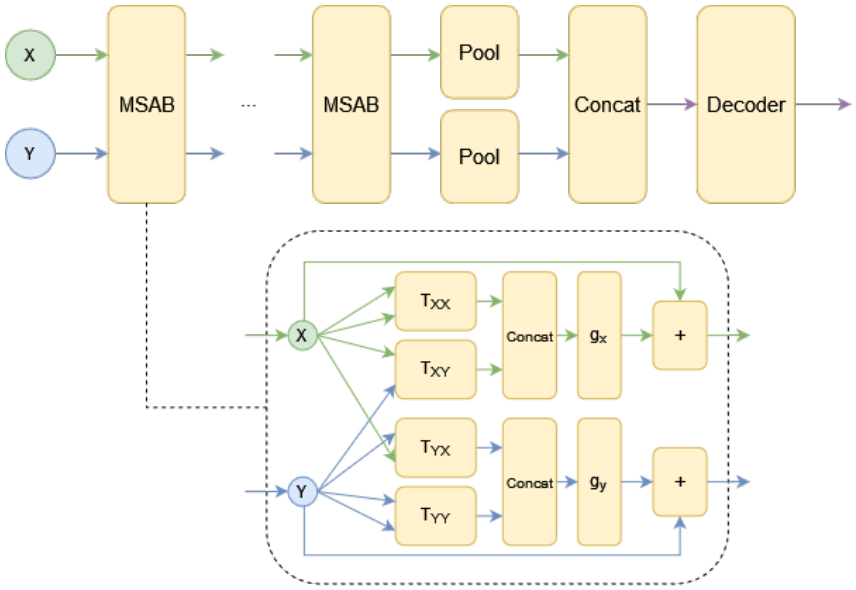


Figure 4.1: Diagram of the Multi-Set Transformer and Multi-Set Attention Block

understood as the invariance or equivariance of the function f under the action of the product $\mathbb{S}_n \times \mathbb{S}_m$.

Gui et al. [2021] propose a similar definition in their work, where they define a partially permutation invariant model:

$$f(X_1, ..., X_m) = \rho\left(\sum_{x \in X_1} \phi_1(x), ..., \sum_{x \in X_m} \phi_m(x)\right)$$

This is essentially a generalization of the sum-decomposition architecture described by Equation 4.3. As mentioned previously, this sum-decomposition architecture has a number of significant theoretical and practical limitations. Rather than build a multi-set architecture on this simple sum-decomposition model, I choose to focus on the general models defined in Equations 4.6 and 4.7, with particular focus on the set transformer model defined in Lee et al. [2019]. This architecture is also advantageous for other reasons; models such as transformers which explicitly model the relationship between pairs of elements in a set carry a useful inductive bias for learning distance functions. Consider the case of computing the Wasserstein distance, wherein computing the ground distance (e.g. euclidean distance) between every pair of elements within the sets is a necessary step. Similarly, for quantities such as KL divergence, methods such as the algorithm of Wang et al. [2009] often rely on nearest-neighbour distances as a useful proxy for the concentration

of the distribution.

4.3.3 The Proposed Model

Let us thus begin from the model presented in Equation 4.6. In order to generalize this to the multi-set case, let us now consider applying these architectures to the case where the single input X is now replaced by $X \bigsqcup Y$ - the concatenation of the two inputs X and Y. When the encoder acts upon this input, $\phi(X)$ can be written by representing the pair encodings $\theta(X \bigsqcup Y, X \bigsqcup Y)$ as block matrices:

$$\phi(X \bigsqcup Y) = \bigwedge_{X \bigsqcup Y} \begin{pmatrix} \theta(X,X) & \theta(X,Y) \\ \theta(Y,X) & \theta(Y,Y) \end{pmatrix}$$

This model is still *equivariant*, rather than *partially-equivariant* - it makes no distinction between elements in X and elements in Y. In order to break this symmetry, two key changes can be made. First, instead of having a single encoder θ learn all four of these relationships, θ can be split into four separate pair encoders: θ_{xx}, θ_{xy}, θ_{yx}, and θ_{yy}. These encoders each have their own parameters, though parameter sharing can be employed when the desired function contains symmetries. Second, rather than pooling over the entirety of the joint set $X \bigsqcup Y$ to reduce the encoding matrix to a single set of vectors, each block matrix $theta_{ab}$ is pooled over only the first input set A. The overall encoder ϕ can thus be divided into its outputs ϕ_x, ϕ_y, which can then be decomposed further into:

$$\begin{aligned} \phi_{xx}(X,X) &= \bigwedge_{X} \theta_{xx}(X,X) \\ \phi_{xy}(X,Y) &= \bigwedge_{X} \theta_{xy}(X,Y) \\ \phi_{yx}(Y,X) &= \bigwedge_{Y} \theta_{yx}(Y,X) \\ \phi_{yy}(Y,Y) &= \bigwedge_{Y} \theta_{yy}(Y,Y) \end{aligned} \tag{4.8}$$

In this manner, each encoder learns the relationships between one of the four pairs of sets separately. This information can then be recombined to produce output encodings for X and Y by concatenating the outputs of the appropriate encoder blocks and applying a shallow feedforward network:

$$\begin{aligned} \phi_x(X,Y) &= g_x(\phi_{xx}(X,X), \phi_{xy}(X,Y)) \\ \phi_y(X,Y) &= g_y(\phi_{yx}(Y,X), \phi_{yy}(Y,Y)) \end{aligned} \tag{4.9}$$

Equation 4.6 then becomes

$$f(X,Y) = \rho\left(\prod_X \phi_x(X,Y), \prod_Y \phi_y(X,Y)\right) \tag{4.10}$$

This structure satisfies the property of partial permutation equivariance, and allows the model to retain the benefits of explicitly representing relationships between each pair of elements and each pair of sets. This general model can now be used to extend any single-set model defined by Equation 4.6 - including both Set Transformers and Relation Networks.

4.3.4 Multi-Set Transformer

The primary architecture under consideration is the *multi-set transformer* architecture, which follows from constructing the encoder defined in Equation 4.9 with transformers as the encoders ϕ. Let the *multi-set attention block* be defined as $\text{MSAB}(X, Y) = (Z_X, Z_Y)$ where

$$\begin{aligned} Z_X &= X + g_x(T_{xx}(X, X), T_{xy}(X, Y)) \\ Z_Y &= Y + g_y(T_{yx}(Y, X), T_{yy}(Y, Y)) \end{aligned} \tag{4.11}$$

and where $T_{ab}(A, B)$ is a transformer block as defined in Eq. 2.10, and the functions g are 1-layer feedforward networks with ReLU activations which are applied to the elementwise concatenation of the outputs of the two transformer blocks. These MSABs can now be treated like regular transformer blocks and stacked to form a deep encoder. A multi-set analogue of Eq. 4.4 can then be defined using Equation 4.10, wherein ϕ is an encoder formed of stacked MSABs, which produces outputs of Z_X and Z_Y. These outputs are then pooled over X and Y independently, concatenated, then passed into a feedforward decoder to produce the final output. A detailed demonstration of how the multi-set attention block is derived from a single-set attention block is presented in Section B.2 of the appendices. Figure 4.1 illustrates the proposed model with MSABs.

4.3.5 Variable-Dimension Encoders

Another application of permutation-invariance that will be particularly useful when discussing distance functions is invariance to the input dimension itself. Incorporating the principles of permutation with respect to the input dimension (i.e. treating each feature of the input in a symmetric fashion) can yield several advantages. First, the model is no longer restricted to operate on inputs of a fixed dimension, and can accept inputs of many different sizes. Second, this representation carries a useful inductive bias for the particular setting of statistical distance functions. A traditional machine learning pipeline in a field such as NLP might learn an embedding scheme in which different dimensions encode different representations of the input sequence, and must thus be treated differently from one another. Statistical distance functions such as the Wasserstein Distance or KL Divergence, however, are often symmetric with respect to the dimensions of their input vectors. In these settings, this invariant representation forces the model to align more closely with the true underlying function, and generally leads to improved performance.

Zaheer et al. [2017] propose a simplistic form of this in their original paper. They demonstrate that a linear layer with a weight matrix that takes the form $\Theta = \lambda I + \gamma 11^T$ is equivariant with respect to the input dimension. This essentially corresponds to a neural network that computes:

$$y_i = \lambda x_i + \gamma \sum_i x_i$$

Each output is thus computed from a constant multiple of the corresponding input added to a multiple of the sum of all inputs. This has some unfortunate properties, however, since it is constrained to an output size that is exactly equal to the input size at every layer. The solution to this is to introduce multiple channels. Instead of mapping each input dimension to a single output dimension, each input dimension can be mapped to a multichannel output. Multiple encoder layers can thus be stacked, each acting only on this multichannel representation of the input dimensions, and treating the input dimension itself as a batch dimension. Then, after these encoder layers are applied, a pooling operation can be introduced over the input dimension to obtain a fixed dimensional output. This procedure allows inputs of any size to be mapped to a fixed dimension encoding.

In accordance with this, I propose an analogous *multichannel transformer block*, wherein the weight matrices are applied as multichannel transformations which treat the input dimension as a batch dimension. A standard multiheaded attention block (see Section 2.1.2) receives inputs $X, Y \in \mathbb{R}^{n \times d}$ and computes:

$$\text{MHA}(X, Y) = \sigma\left((XW_Q)(YW_K)^T\right) YW_V W_O$$

Our multichannel attention block instead is a function from $\mathbb{R}^{n \times d \times n_c} \times \mathbb{R}^{m \times d \times n_c}$ to $\mathbb{R}^{n \times d \times n_c}$ which computes:

$$\begin{aligned} A &= \sigma\left(\sum_{i=1}^{d} (X_{:,i}W_Q)(Y_{:,i}W_K)^T\right) \\ \text{MHA}(X, Y)_{:,i} &= AY_{:,i}W_V W_O \end{aligned} \tag{4.12}$$

This multichannel transformer architecture consists of an initial projection from a single input channel up to k input channels, followed by n_b multichannel transformer blocks, followed by a max pooling over the input dimension d. This is also compatible with the Multi-Set Attention Block architecture, in which case the multichannel transformer blocks are replaced by multichannel MSABs, with a max pooling at the end as before.

4.4 Theoretical Analysis

In order to demonstrate the theoretical effectiveness of the proposed approach, I will demonstrate that the multi-set transformer architecture described in Section 4.3.4 is a universal approximator

on *partially permutation equivariant* functions, and that combined with a pooling layer it is also a universal approximator of *partially permutation invariant* functions.

While the property of universal approximation can of course be achieved by standard feedforward neural networks, these networks have many undesirable properties when modelling functions on equivariant sets. Such networks are limited to inputs of a fixed size n - a restriction which the multi-set transformer architecture does not share. In addition, traditional feedforward networks offer no guarantees of permutation equivariance. As such, there are significant benefits to be offered from demonstrating an alternative architecture which can offer the same theoretical guarantees while also obeying a structure that is naturally more suited to the problem at hand.

First, some preliminaries. I will follow the notation in Yun et al. [2019], for their Theorem 2 forms a foundation for the theorem that will be stated shortly. Let $\mathcal{F}_{PE}$ be the class of all continuous permutation-equivariant functions with compact support from $\mathbb{R}^{d\times n}$ to $\mathbb{R}^{d\times n}$. Given $f, g : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ and $1 \leq p < \infty$, let

$$d_p(f, g) = \left(\int \|f(\mathbf{X}) - g(\mathbf{X})\|_p^p d\mathbf{X} \right)^{1/p} \tag{4.13}$$

Let $t^{h,m,r} : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ denote a transformer block with an attention layer with h heads of size m and a feedforward layer with r hidden nodes. Then, let $\mathcal{T}^{h,m,r}$ define the class of functions $g : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ such that g consists of a composition of transformer blocks of the form $t^{h,m,r}$. Given these definitions, consider the following statement of Theorem 2 from Yun et al. [2019], which is given by:

Theorem 4.4.1. *Let $1 \leq p < \infty$ and $\epsilon > 0$, then for any given $f \in \mathcal{F}_{PE}$ there exists a transformer network $g \in \mathcal{T}^{2,1,4}$ such that $d_p(f, g) \leq \epsilon$. [Yun et al., 2019]*

To extend this to the case of partial permutation equivariance, these definitions must now be modified slightly. Let $\mathcal{F}_{PPE}$ be the class of all continuous partially-permutation-equivariant functions on two sets with compact support from $\mathbb{R}^{d\times n} \times \mathbb{R}^{d\times m}$ to $\mathbb{R}^{d\times n} \times \mathbb{R}^{d\times m}$. Let $c^{h,m,r} : \mathbb{R}^{d\times n} \times \mathbb{R}^{d\times m} \to \mathbb{R}^{d\times n} \times \mathbb{R}^{d\times m}$ denote a multi-set attention block with attention layers with h heads of size m and feedforward layers with r hidden nodes, and let $\mathcal{T}_C^{h,m,r}$ define the class of functions $g : \mathbb{R}^{d\times n} \times \mathbb{R}^{d\times m} \to \mathbb{R}^{d\times n} \times \mathbb{R}^{d\times m}$ such that g consists of a composition of MSABs of the form $c^{h,m,r}$. The modified theorem now states:

Theorem 4.4.2. *Let $1 \leq p < \infty$ and $\epsilon > 0$, then for any given $f \in \mathcal{F}_{PPE}$ there exists a multi-set transformer network $g \in \mathcal{T}_C^{2,2,4}$ such that $d_p(f, g) \leq \epsilon$.*

A proof of Theorem 4.4.2 is given in Section 4.5. If we define $\mathcal{F}_{PPI}$ to be the class of all continuous partially-permutation-invariant functions on two sets with compact support from $\mathbb{R}^{d\times n} \times \mathbb{R}^{d\times m}$ to $\mathbb{R}^d$, this now directly leads to the corollary:

Corollary 4.4.3. *Let $1 \leq p < \infty$ and $\epsilon > 0$, then for any given $f \in \mathcal{F}_{PPI}$ there exists a function $f(X,Y) = \max g(X,Y)$ such that $d_p(f,g) \leq \epsilon$, wherein $g \in \mathcal{T}_C^{2,2,4}$ is a multi-set transformer network.*

Observe that this corollary follows directly from Theorem 4.4.2, since for any function $f \in \mathcal{F}_{PPI}$ we can construct $g \in \mathcal{F}_{PPE}$ such that $g(X,Y)_{:,j} = f(X,Y)\ \forall j$ (i.e., a set of outputs where each entry simply contains $f(X,Y)$). f thus obeys the equation $f(X,Y) = \max_j g(X,Y)_{:,j}$. Therefore, any function in $f \in \mathcal{F}_{PPI}$ can be expressed as a pooling function applied to a function $g \in \mathcal{F}_{PPE}$ and f can thus be approximated by a multi-set transformer network by simply approximating g as per Theorem 4.4.2.

4.5 Proof of Theorem 4.4.2

This proof will closely follow the work of Yun et al. [2019]. In their work, they define the following theorem (stated earlier as Theorem 4.4.1):

Theorem 2 ([Yun et al., 2019])**.** *Let $1 \leq p < \infty$ and $\epsilon > 0$, then for any given $f \in \mathcal{F}_{PE}$ there exists a transformer network $g \in \mathcal{T}^{2,1,4}$ such that $d_p(f,g) \leq \epsilon$.*

The proof of this theorem follows several stages, enumerated here:

1. Approximate $\mathcal{F}_{\text{PE}}$ by piecewise constant functions $\overline{\mathcal{F}}_{\text{PE}}$ on the grid of resolution δ, denoted $\mathbb{G}_\delta = 0, \delta, ..., 1 - \delta^{d \times n}$.
2. Approximate $\overline{\mathcal{F}}_{\text{PE}}$ by a class of modified transformers $\overline{\mathcal{T}}$, with particular choices of parameters and hardmax replacing softmax.

   Proposition 4 ([Yun et al., 2019])**.** *For each $\overline{f} \in \overline{\mathcal{F}}_{PE}$ and $1 \leq p < \infty$, $\exists \overline{g} \in \overline{\mathcal{T}}^{2,1,1}$ such that $d_p(\overline{f}, \overline{g}) = O(\delta^{d/p})$*

3. Approximate the class of modified transformers $\overline{\mathcal{T}}^{2,1,1}$ with $\mathcal{T}^{2,1,4}$

Each of these steps leads to a certain error under d_p, with step 1 and 3 contributing an error of order $\epsilon/3$ and step 2 contributing an error of order $\mathcal{O}(\delta^{d/p})$. This leads to a total error of less than order ϵ, so long as δ is chosen to be sufficiently small. The proof of Theorem 4.4.2 will follow the same procedure. Many of these steps will be omitted, as they remain unchanged from their presentation in the original paper.

Of the steps mentioned above, we will focus our attention on step 2, as the others remain unchanged. The key is to prove a modified version of Proposition 4. The proof of this proposition itself follows the following steps:

1. Given $X \in \mathbb{R}^{d \times n}$, quantize X to $L^{(x)} \in G_\delta^+$, where G_δ is the $[0,1]^{d \times n}$ grid with resolution δ and $G_\delta^+ = G_\delta \cup \{-\delta^{-nd}\}$.

2. Implement a *contextual mapping* $q(X)$ such that all elements of $q(L), q(L')$ are distinct if L, L' are not permutations of each other. This essentially maps each (x_i, X) to a unique representation.

3. Since each (x_i, X) is mapped to a unique representation, we can use feedforward networks to approximate any mapping from these unique representations to the desired outputs up to arbitrary accuracy due to their nature as universal approximators, and thus approximate any equivariant function on X.

In order to translate this proof to the multi-set case, several adjustments need to be made. The first of these steps remains unchanged, and applies in a similar fashion to X, Y. The most critical part of the proof comes in the second step, with the construction of the contextual mapping. Yun et al demonstrate a particular construction of this contextual mapping, and with it they prove their Lemma 6, detailed below:

Lemma 6 ([Yun et al., 2019])**.** *Let $\widetilde{G}_\delta = \{L \in G_\delta |\ L_{:,i} \neq L_{:,j}\ \forall i \neq j\}$. Let $n \geq 2$ and $\delta^{-1} \geq 2$. Then, there exists a function $q(L)$ of the form $q(L) = u^T g_c(L)$ where $u \in \mathbb{R}^d$, and $g_c(L) : \mathbb{R}^{d \times n} \to \mathbb{R}^{d \times n}$ is a function composed of $\delta^{-d} + 1$ self-attention layers using the σ_H operator, as well as constants t_l, t_r such that $q(L)$ has the following properties:*

1. *For any $L \in \widetilde{G}_\delta$, all entries of $q(L)$ are distinct*

2. *For any $L, L' \in \widetilde{G}_\delta$, if L is not a permutation of L' then all entries of $q(L)$ and $q(L')$ are distinct*

3. *For any $L \in \widetilde{G}_\delta$, all entries of $q(L)$ are in $[t_l, t_r]$*

4. *For any $L \in G_\delta^+ \setminus \widetilde{G}_\delta$, all entries of $q(L)$ are not in $[t_l, t_r]$*

The overall structure of this proof will remain largely the same, save for substituting a new version of this lemma:

Lemma 6'. *Let $\widetilde{G}_\delta = \{L \in G_\delta |\ L_{:,i} \neq L_{:,j}\ \forall i \neq j\}$. Let $n \geq 2$ and $\delta^{-1} \geq 2$. Then, there exists a function $q(L^X, L^Y)$ on $L^X, L^Y \in G_\delta^+; L^X \neq L^Y$ consisting of $2\delta^{-d} + 4$ MSAB layers using the σ_H operator, as well as constants $t_l^X, t_r^X, t_l^Y, t_r^Y$ such that $q(L^X, L^Y)$ has the following properties:*

1. *For any $L^X, L^Y \in G_\delta$, all entries of $q(L^X, L^Y)$ are distinct*

2. *For any $L^X, L^Y, L'^X, L'^Y \in G_\delta$, if L'^X is not a permutation of L^X and L'^Y is not a permutation of L^Y then all entries of $q(L^X, L^Y)$ and $q(L'^X, L'^Y)$ are distinct*

3. *For any* $L^X, L^Y \in \widetilde{G}_\delta$, *all entries of* $q^X(L^X, L^Y)$ *are in* $[t_l^X, t_r^X]$

4. *For any* $L^X, L^Y \in \widetilde{G}_\delta$, *all entries of* $q^Y(L^X, L^Y)$ *are in* $[t_l^Y, t_r^Y]$

5. *For any* $L^X \in G_\delta^+ \setminus \widetilde{G}_\delta$, $L^Y \in G_\delta^+$, *all entries of* $q(L^X, L^Y)$ *are not in* $[t_l, t_r]$

6. *For any* $L^Y \in G_\delta^+ \setminus \widetilde{G}_\delta$, $L^X \in G_\delta^+$, *all entries of* $q(L^X, L^Y)$ *are not in* $[t_l, t_r]$

Once the equivalent contextual mapping is defined and Lemma 6' is established, the second stage of the proof is complete. In order to prove the third stage, I then make a slight modification of Yun et al's Lemma 7. This lemma states:

Lemma 7 ([Yun et al., 2019])**.** *Let* $g_c : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ *be the function defined in Lemma 6. Then, there exists a function* $g_v : \mathbb{R}^{d\times n} \to \mathbb{R}^{d\times n}$ *composed of* $O(n(\frac{1}{\delta})^{dn}/n!)$ *column-wise feedforward layers* ($r = 1$) *such that* g_v *is defined by a function* $g_{col} : \mathbb{R}^d \to \mathbb{R}^d$,

$$g_v(Z) = [g_{col}(Z_{:,1}), ..., g_{col}(Z_{:,n})]$$

where $\forall j = 1, ..., n$

$$g_{col}(g_c(L)_{:,j}) = \begin{cases} (A_L)_{:,j} & L \in \widetilde{G_\delta} \\ 0_d & L \in G_\delta^+ \setminus \widetilde{G_\delta} \end{cases}$$

The modified version states instead that:

Lemma 7'. *Let* $g_c : \mathbb{R}^{d\times n} \times \mathbb{R}^{d\times m} \to \mathbb{R}^{d\times n} \times \mathbb{R}^{d\times m}$ *be the function defined in Lemma 6'. Then, there exists a function* $g_v : \mathbb{R}^{d\times n} \times \mathbb{R}^{d\times m} \to \mathbb{R}^{d\times n} \times \mathbb{R}^{d\times m}$ *composed of* $O\left((n+m)(\frac{1}{\delta})^{d(n+m)}/(n!m!)\right)$ *column-wise feedforward layers* ($r = 1$) *such that* g_v *is defined by a function* $g_{col} : \mathbb{R}^d \to \mathbb{R}^d$,

$$g_v(Z) = [g_{col}(Z_{:,1}), ..., g_{col}(Z_{:,n})]$$

where $\forall i = 1, ..., n, j = 1, ...m$

$$g_{col}(g_c(L)_{:,j}) = \begin{cases} (A_L)_{:,j} & L \in \widetilde{G_\delta} \\ 0_d & L \in G_\delta^+ \setminus \widetilde{G_\delta} \end{cases}$$

The construction of $q(L^X, L^Y)$, and the proofs of Lemma 6' and 7' will follow in the subsequent sections.

4.5.1 Construction of the Contextual Mapping

The construction of this function proceeds as follows. First, note that a multi-set attention block can implement functions consisting of multiple feedforward or attention blocks successively by using the skip connections to ignore the other component when needed. It can also implement a function consisting of multiple blocks successively applied to a single one of the input sets by simply letting the action of the block on the other set be the identity. The multi-set attention block can also implement a layer which simply performs the attention computation e.g. $ATTN_{XX}(X, X)$ by letting $g_X(f_{XX}, f_{XY}) = f_{XX}$. As such, multi-set attention blocks can reproduce any function on either X or Y implemented by regular transformer blocks - such as the constructions defined in Yun et al. [2019].

Let $g_c(L)$ be the iterated selective shift network defined in Yun et al. [2019], consisting of n selective shift operations followed by a final global shift layer. This results in the mapping

$$u^T g_c(L)_{:,j} = \widetilde{\ell}_j + \delta^{-(n+1)d}\widetilde{\ell}_n$$

where $\widetilde{\ell}_j$ is the j-th output of the selective shift layers, sorted in ascending order. We will now construct our own analogous network, $g_c(L^X, L^Y)$. Let $g_s(L)$ be the selective shift portion of $g_c(L)$. Then, we let the first δ^{-d} blocks implement $g_s(L^X)$ on L^X alone while performing the identity operation on L^Y ($T_{XY}, T_{YX} = 0$ for this component). The next δ^{-d} blocks do the same thing on L^Y, while performing the identity on L^X. The next block then applies a modified global shift to each set with attention component $\delta^{(n+1)d}\psi(\cdot; 0)$ - the same global shift as in Yun et al. [2019]. This shift is applied with attention over X, and a scaled version is also applied with attention over Y - i.e. shifting L^X by $\delta^{-(n+1)d}\max_k u^T L^X_{:,k}$ and *also* by $\delta^{(m+1)d}\max_k u^T L^Y_{:,k}$ (and the same in reverse for L^Y). This can be implemented by a single MSAB block with $T_{XX} = T_{YY} = \delta^{-(n+1)d}\psi(\cdot; 0)$ and $T_{XY} = T_{YX} = \delta^d\psi(\cdot; 0)$. This comprises our $g_c(L^X, L^Y)$ block, and results in an output of

$$q^X(L^X, L^Y)_j = u^T g^X_c(L^X, L^Y)_{:,j} = \widetilde{\ell}^X_j + \delta^{-(n+1)d}\,\widetilde{\ell}^X_n + \delta^{(m+1)d}\,\widetilde{\ell}^Y_m$$
$$q^Y(L^X, L^Y)_j = u^T g^Y_c(L^X, L^Y)_{:,j} = \widetilde{\ell}^Y_j + \delta^{-(m+1)d}\,\widetilde{\ell}^Y_m + \delta^{(n+1)d}\,\widetilde{\ell}^X_n$$

4.5.2 Proof of Lemma 6'

The proof of Lemma 6' proceeds much as the proof of Lemma 6. We must now check that all conditions are satisfied. In order to do so, we will also require Lemma 10 from Yun et al. [2019], restated here for convenience. This lemma applies to both $\widetilde{\ell^X_n}$ and $\widetilde{\ell^Y_m}$ independently, as both are constructed from the same selective shift operations as in Yun et al.

Lemma 10 ([Yun et al., 2019]). *After n shift operations, $\widetilde{\ell}_n = u^T\widetilde{L_{:,n}}$ satisfies the following bounds:*

$$\delta^{-(n-1)d+1}(\delta^{-d} - 1) \leq \widetilde{\ell}_n \leq \delta^{-nd+1}(\delta^{-d} - 1) - \delta(\delta^{-d} - 1)^2$$

Property 2

For the second property, let us begin by considering the case where L^X, L'^X are not permutations of each other. Then, analogous to Yun et al., we have that

$$u^T g_c^X(L^X, L^Y)_{:,j} \in [\delta^{-(n+1)d}\widetilde{\ell}_n^X + \delta^{(m+1)d}\widetilde{\ell}_m^Y, \delta^{-(n+1)d}\widetilde{\ell}_n^X + \delta^{(m+1)d}\widetilde{\ell}_m^Y + \delta^{-(n+1)d+1} - \delta^{-nd+1})$$

As in Yun et al., $L^X, L'^X \in \widetilde{G_\delta}$ which are not permutations of each other must result in $\widetilde{\ell}_n^X, \widetilde{\ell}_n'^X$ differing by at least δ. By Lemma 10, distinct L^Y, L'^Y can lead to $\widetilde{\ell}_m^Y, \widetilde{\ell}_m'^Y$ differing by a value strictly less than $\delta^{-(m+1)d+1}$. The smallest net change this can result in is $\delta^{-(n+1)d} \cdot \delta - \delta^{(m+1)d} \cdot \delta^{-(m+1)d+1} = \delta^{-(n+1)d+1} - \delta$. Since this is larger than the width of the original interval and the intervals are open on at least one end, the intervals must thus be disjoint, and thus if L^X and L'^X are distinct, Q^X and Q'^X must be distinct. Now, consider the case where $L^X, L'^X \in \widetilde{G_\delta}$ are permutations of each other, but $L^Y, L'^Y \in \widetilde{G_\delta}$ are not. In this case, since $\widetilde{\ell}_m^Y, \widetilde{\ell}_m'^Y$ must differ by at least δ, Q^X and Q'^X must again be distinct. Since $|\widetilde{\ell}_m^Y - \widetilde{\ell}_m'^Y| < \delta^{-(m+1)d+1}$, the resulting change in Q^X must be strictly less than δ. Since $\widetilde{\ell}_j^X, \widetilde{\ell}_k^X$ must be separated by at least δ for $j \neq k$, $Q_j^X \neq Q_k'^X$ for any $j \neq k$. Thus, all entries of Q^X and Q'^X must be distinct in this case as well. These results apply symmetrically for Q^Y and Q'^Y, and thus this proves Property 2.

Note that if L^X, L'^X are not permutations of each other then $u^T g_c^X(L^X, L^Y), u^T g_c^X(L'^X, L'^Y)$ must be separated from each other by at least δ, whereas if L^X, L'^X are permutations of each other, but L^Y, L'^Y are not, $u^T g_c^X(L^X, L^Y), u^T g_c^X(L'^X, L'^Y)$ must be separated by at least δ^{m+2}. In general then, all entries of $u^T g_c^X(L^X, L^Y), u^T g_c^X(L'^X, L'^Y)$ must be separated from each other by at least δ^{m+2} (and conversely with δ^{n+2} for Y).

Properties 3-4

By the same procedure as Yun et al (in B.5.1), we can see that $q^X(L^X, L^Y)$ obeys

$$(\delta^{-2nd+1} + \delta^{2d+1})(\delta^{-d} - 1) \leq u^T g_c^X(L^X, L^Y) < (\delta^{-(2n+1)d+1} + \delta^{d+1})(\delta^{-d} - 1)$$

if $L^X, L^Y \in \widetilde{G}_\delta$. Thus, $\forall L^X, L^Y \in \widetilde{G}_\delta\ q^X(L^X, L^Y) \in [t_l, t_r]$ where

$$t_l^X = (\delta^{-2nd+1} + \delta^{2d+1})(\delta^{-d} - 1)$$
$$t_r^X = (\delta^{-(2n+1)d+1} + \delta^{d+1})(\delta^{-d} - 1)$$

The same holds for $q^Y(L^X, L^Y)$ with

$$t_l^Y = (\delta^{-2md+1} + \delta^{2d+1})(\delta^{-d} - 1)$$
$$t_r^Y = (\delta^{-(2m+1)d+1} + \delta^{d+1})(\delta^{-d} - 1)$$

Property 1

Within Q^X and Q^Y, all entries must be distinct since $\widetilde{\ell}_1 < ... < \widetilde{\ell}_n$. Consider the bounds t_l^X, t_r^X from the previous section. Consider first the case where $n \neq m$. Without loss of generality, suppose $m < n$. If $m = n - k$ then we have $t_r^Y = (\delta^{-(2n+1-2k)d+1} + \delta^{d+1})(\delta^{-d} - 1) < t_l^X$ and thus Q^X and Q^Y belong to disjoint intervals.

If $n = m$, then we can apply a similar argument as we did in proving Property 2, and argue that $\widetilde{\ell}_n^X, \widetilde{\ell}_n^Y$ which are not permutations must differ by at least δ. This results in intervals shifted from each other by at least $\delta^{-(n+1)d+1} - \delta^{d+1}$, which will always be larger than $\delta^{-(n+1)d+1} - \delta^{-nd+1}$, which is the width of the intervals. Thus, in this case too Q^X and Q^Y must be distinct, and Property 1 is proven.

Properties 5-6 - Case 1

Take $L^Y \in G_\delta^+$ and $L^X \in G_\delta \setminus \widetilde{G}_\delta$. This corresponds to an L^X with duplicate columns but within the region of compact support. In this case,

$$\widetilde{\ell}_n^X \leq \delta^{-(n-1)d+1}(\delta^{-d} - 1) - \delta(\delta^{-d} - 1)^2$$

Then,

$$\begin{aligned}
u^T g_c^X(L^X, L^Y)_{:,j} &= \widetilde{\ell}_j^X + \delta^{-(n+1)d}\, \widetilde{\ell}_n^X + \delta^{(m+1)d}\, \widetilde{\ell}_m^Y \\
&\leq (\delta^{-(n+1)d} + 1)(\delta^{-(n-1)d+1}(\delta^{-d} - 1) - \delta(\delta^{-d} - 1)^2) \\
&\quad + \delta^{(m+1)d}(\delta^{-md+1}(\delta^{-d} - 1) - \delta(\delta^{-d} - 1)^2)) \\
&< \delta^{-2nd+1}(\delta^{-d} - 1) + \delta^{-(n-1)d+1}(\delta^{-d} - 1) - \delta^{-(n+1)d+1}(\delta^{-d} - 1)^2 + \delta^{d+1}(\delta^{-d} - 1) \\
&< \delta^{-2nd+1}(\delta^{-d} - 1) + \delta^{-(n-1)d+1}(\delta^{-d} - 1)(1 + \delta^{nd} - \delta^{-2d}(\delta^{-d} - 1)) \\
&< \delta^{-2nd+1}(\delta^{-d} - 1) < t_l^X
\end{aligned}$$

since $\delta^{-d} \geq 2$. Thus, Q^X falls strictly outside $[t_l^X, t_r^X]$, proving Property 6 for the case when $L^Y \in G_\delta^+$ and $L^X \in G_\delta \setminus \widetilde{G}_\delta$. The same applies in reverse for Property 7.

Properties 5-6 - Case 2

Take $L^Y \in G_\delta^+$ and $L^X \in G_\delta^+ \setminus G_\delta$. This corresponds to an L^X that contains at least one element outside the region of compact support. This leads to columns of L^X containing negative values. Note first that for a column $L_{:,j}^X$ containing $-\delta^{-nd}$, $\ell_j^X = \widetilde{\ell}_j^X$ as the selective shift operation does not alter it. From Yun et al, we have that $\widetilde{\ell}_j^X \leq -\delta^{-nd} + \delta^{-d+1} - 1 < 0$, and that the last

layer shifts negative values by $\delta^{-(n+1)d} \min_k \widetilde{\ell}_k^X$. After this shift is applied and our additional attention-based shift is applied,

$$\begin{aligned} u^T g_c^X(L^X, L^Y)_{:,j} &\leq (-\delta^{-nd} + \delta^{-d+1} - 1)(1 + \delta^{-(n+1)d}) + \delta^{(m+1)d}\widetilde{\ell}_m^Y \\ &\leq (-\delta^{-nd} + \delta^{-d+1} - 1)(1 + \delta^{-(n+1)d}) + \delta^{(m+1)d}\delta^{-(m+1)d+1} \\ &\leq -\delta^{-(2n+1)d} + \delta^{-(n+2)d+1} + \delta < 0 < t_l^X \end{aligned}$$

where we use that $\widetilde{\ell}_m^Y \leq \delta^{-(m+1)d+1}$ and $\delta^{-1} \geq 2$. Thus, any negative column is mapped to a value outside of $[t_l^X, t_r^X]$. Note that this holds for any L^Y, including an $L^Y \in G_\delta^+ \setminus G_\delta$.

In the case where *all* columns are negative, the argument proceeds exactly as in Yun et al, and all elements are straightforwardly less than zero as shown above. In the case where only some columns are negative, the negative columns are mapped to negative values as before, and the positive columns satisfy $u^T g_c^X(L^X, L^Y)_{:,i} \geq \delta^{-(2n+2)d+}1$ before we apply our attention shift layer. If $\max_k \widetilde{\ell}_k^Y > 0$, then $u^T g_c^X(L^X, L^Y)_{:,i} \geq \delta^(- (2n+2)d + 1$ still holds. If not, all entries of $\widetilde{\ell}_k^Y$ must be negative, which means $\max_k \widetilde{\ell}_k^Y = -\delta^{-md}(\delta^{-d} - 1)$. We then have

$$u^T g_c^X(L^X, L^Y)_{:,i} \geq \delta^{-(2n+2)d+1} - \delta^d(\delta^{-d} - 1) > t_r^X$$

and thus Property 5 is proved for all cases (and symmetrically for Property 6).

4.5.3 Proof of Lemma 7'

This proof very closely follows the proof shown in Yun et al, with a few small changes. The first layer used to map all invalid entries to strictly negative numbers becomes

$$Z \mapsto Z - (M-1)\mathbf{1}_{n+m}(\phi^X(u^T Z^X) + \phi^Y(u^T Z^Y))$$

where M is the maximum value of the image of $g_c(G_\delta^+, G_\delta^+)$ and

$$\phi^X(t) = \begin{cases} 0 & t \in [t_l^X, t_r^X] \\ 1 & t \notin [t_l^X, t_r^X] \end{cases}$$

$$\phi^Y(t) = \begin{cases} 0 & t \in [t_l^Y, t_r^Y] \\ 1 & t \notin [t_l^Y, t_r^Y] \end{cases}$$

This layer is applied to both X and Y, and ensures that if either X or Y contain invalid elements, the entirety of both sets are mapped to negative values. The next d layers, which map all negative entries to the zero matrix, remain unchanged and are applied again to both X and Y.

The remaining layers must now map $g_c^X(L^X, L^Y)$ to A_L^X and $g_c^Y(L^X, L^Y)$ to A_L^Y. In a similar fashion to Yun et al, we add $\mathcal{O}\left((n+m)(1/\delta)^{d(n+m)}/(n!m!)\right)$ feedforward layers, each of which

maps one value of $u^T g_c^X(L^X, L^Y)$ or $u^T g_c^Y(L^X, L^Y)$ to the correct output while leaving the others unaffected. For a given value of $u^T g_c^X(\bar{L}^X, \bar{L}^Y)_{:,j}$ these layers take the form:

$$Z^X \mapsto Z^X + \left((A_{\bar{L}}^X)_{:,j} - g_c^X(\bar{L}^X, \bar{L}^Y)_{:,j}\right) \phi^X \left(u^T Z^X - u^T g_c^X(\bar{L}^X, \bar{L}^Y)_{:,j} \mathbf{1}_n^T\right)$$
$$Z^Y \mapsto Z^Y + \left((A_{\bar{L}}^Y)_{:,j} - g_c^Y(\bar{L}^X, \bar{L}^Y)_{:,j}\right) \phi^Y \left(u^T Z^Y - u^T g_c^Y(\bar{L}^X, \bar{L}^Y)_{:,j} \mathbf{1}_m^T\right)$$

wherein

$$\phi^X(t) = \begin{cases} 0 & t < -\delta^{m+2}/2 \text{ or } t \geq \delta^{m+2}/2 \\ 1 & -\delta^{m+2}/2 \leq t < \delta^{m+2}/2 \end{cases} \qquad \phi^Y(t) = \begin{cases} 0 & t < -\delta^{n+2}/2 \text{ or } t \geq \delta^{n+2}/2 \\ 1 & -\delta^{n+2}/2 \leq t < \delta^{n+2}/2 \end{cases}$$

If $Z = g_c(L^X, L^Y)$ where L^X is not a permutation of $\bar{L}^X$ and L^Y is not a permutation of $\bar{L}^Y$, then $\phi^X \left(u^T Z - u^T g_c^X(\bar{L}^X, \bar{L}^Y)_{:,j} \mathbf{1}_n^T\right) = 0$ and Z is unchanged. If on the other hand L^X *is* a permutation of $\bar{L}^X$ and L^Y *is* a permutation of $\bar{L}^Y$, with $\bar{L}_{:,j}^X = L_{:,i}^X$, then $\phi^X \left(u^T Z^X - u^T g_c^X(\bar{L}^X, \bar{L}^Y)_{:,j} \mathbf{1}_n^T\right) = (e^{(i)})^T$ and $g_c^X(L^X, L^Y)$ is mapped to $(A_L^X)_{:,i}$. Thus, this layer maps $g_c^X(L^X, L^Y)_{:,j}$ to the correct output for the specific inputs $\bar{L}^X, \bar{L}^Y$ (or permutations thereof), and does not affect any other inputs. By stacking $\mathcal{O}\left((n+m)(1/\delta)^{d(n+m)}/(n!m!)\right)$ of these layers together, we achieve the correct output for any possible inputs L^X, L^Y.

4.6 Experiments

In order to evaluate the model, I consider several tasks, including a number of simple image-based set tasks as well as the aforementioned distance functions. Our model is compared against the PINE model proposed in Gui et al. [2021], as well as a number of single-set models. For those baselines, I take a single-set architecture such as Deep Sets (Single-Set RFF), Relation Networks (Single-Set RN) or Set Transformers (Single-Set Transformer), compute pooled representations for each of X and Y, then concatenate these representations and pass them into a feedforward decoder. Finally, I also compare to a simple transformer baseline wherein a Set Transformer is applied to the union $X \bigsqcup Y$ (Union Transformer).

Several variants and ablations of the proposed model are also examined. The two variants of our model include Multi-Set Transformer and Multi-Set RN (Relation Networks). In the latter, transformer blocks are replaced by relation network blocks for the four encoders, with max pooling operations for both Λ and Γ. In addition to these variants, several ablations of the proposed Multi-Set Transformer model are also considered. First, I consider a variant where $g_x(T_{xx}(X,X), T_{xy}(X,Y)) = T_{xx}(X,X) + T_{xy}(X,Y)$ (referred to as Sum-Merge). Second, I consider modifications to the four-block encoder structure itself by removing T_{XX} and T_{YY} - leaving only the cross terms T_{XY} and T_{YX} (referred as Cross-Only). Finally, the single set transformer baseline (Single-Set Transformer) can itself be considered an ablation of the model with the cross-set blocks removed instead of the same-set blocks. For all experiments, three trials are performed

	d=2	d=4	d=8	d=16
Baselines				
KNN	0.2047	0.5662	4.0584	28.0382
PINE	0.174 ± 0.0003	0.496 ± 0.001	2.080 ± 0.0004	10.534 ± 0.011
Single-Set RFF	0.122 ± 0.011	0.440 ± 0.016	1.777 ± 0.012	8.008 ± 0.164
Single-Set RN	0.156 ± 0.001	0.526 ± 0.002	2.143 ± 0.003	9.596 ± 1.621
Single-Set ST	0.073 ± 0.003	0.260 ± 0.004	1.689 ± 0.034	7.591 ± 0.213
Union Transformer	0.175 ± 0.0004	0.499 ± 0.001	2.267 ± 0.001	9.732 ± 0.046
Our Models				
Multi-Set Transformer	0.073 ± 0.001	**0.190** ± 0.008	0.921 ± 0.019	11.105 ± 0.072
Multi-Set RN	0.106 ± 0.002	0.393 ± 0.014	1.517 ± 0.043	7.400 ± 0.126
Cross-Only	0.079 ± 0.002	0.197 ± 0.001	0.993 ± 0.035	5.421 ± 0.169
Sum-Merge	**0.070** ± 0.001	0.195 ± 0.005	0.932 ± 0.014	10.508 ± 0.066
Multi-Set-Transformer-Equi	0.073 ± 0.002	0.192 ± 0.002	**0.800** ± 0.022	**4.700** ± 0.297

Table 4.1: Mean absolute error of models trained on Gaussian mixture data for estimating KL divergence.

and the average and standard deviation are reported. Hyperparameters and detailed experiment settings are included in Section B.1 of the appendices.

4.6.1 Statistical Distances

One particular application of partially-permutation-invariant models that is worth highlighting is their ability to learn to approximate statistical distances between distributions such as the KL divergence or mutual information. Both mutual information and KL divergence are useful metrics that are widely used in a variety of settings within machine learning, and both are very difficult to estimate for any but the simplest distributions. Results are shown for the base multi-set transformer model, as well as the dimension-equivariant model discussed in Section 4.3.5.

KL Divergence

Training the estimator to learn the KL divergence has unique challenges, as calculating the ground truth requires the log likelihood for both the source and target distributions. In order to train the model to learn the KL divergence between a general class of distributions, it is necessary to find a class of models that are effective universal approximators and also admit a tractable log-likelihood. The most obvious class of models fitting this criteria is that of Gaussian mixture models. Gaussian mixtures are generated with a uniformly random number of components (between 1 and 10) and

mixture weights sampled from a uniform Dirichlet distribution. The means of each Gaussian are generated from a uniform distribution, and the covariance matrices are generated by multiplying a correlation matrix sampled from a Lewandowski-Kurowicka-Joe (LKJ) distribution (with $\nu = 5$) by a vector of covariances distributed according to a log-normal distribution (with $\mu_0 = 0$, $\sigma_0 = 0.3$).

Each training example consists of a random number of points $X \sim p_X$ and a random number of points $Y \sim p_Y$ (with $N_X, N_Y \in [100, 150]$). The ground truth is estimated by a Monte Carlo estimate of the true KL divergence using the closed-form log likelihoods, with the generated points X as the samples. The generated data is normalized by computing the mean and covariance across both X and Y, then applying a whitening transformation

$$[X'; Y'] = \Sigma_{XY}^{-1/2} \left([X; Y] - \mu_{XY}\right) \tag{4.14}$$

under which the KL divergence is invariant.

The model is compared (with and without dimension-equivariance) against the aforementioned baselines, as well as the k-nearest-neighbours estimator of Wang et al. [2009]. Table 4.1 shows the mean average error of each model on Gaussian mixture data, averaged over 3 runs. Our model has the lowest error on all dimensions considered. The dimension-equivariant model performed approximately equal to the standard transformer model in low dimension, but performed significantly better in high dimension.

Convergence was a significant issue with the GMM data in higher dimensions, since as the dimensionality increased the true KL divergence of the generated distributions would often explode. This effect was especially notable when the concentration parameter of the LKJ distribution was small, but always occurs once the dimensionality gets large enough.

Mutual Information

In addition to KL-divergence, I will also show the effectiveness of this method for estimating mutual information. Following previous work [Belghazi et al., 2018, Kraskov et al., 2004], experiments are performed using Gaussians with componentwise correlations $\rho \in (-1, 1)$, with standardized Gaussian marginals. Training examples are generated in a similar fashion as in the KL case, with a ρ sampled uniformly from the interval $(-1, 1)$, then a random number of samples between 100 and 150 drawn from the resulting distribution for each of X and Y. We plot the performance of the proposed model for varying values of ρ compared to both the Kraskov et al. [2004] and MINE [Belghazi et al., 2018] baselines in 2, 10 and 20 dimensions (see Fig. 4.2). The model performs somewhat worse than the other methods shown in the 2-dimensional case, but is almost indistinguishable from the ground truth in the 10 and 20-dimensional cases. Note also that while methods such as MINE must be trained on a particular dataset in order to predict its mutual information, the proposed method need only be trained once, and can then be used for inference on any similar dataset wthout retraining.

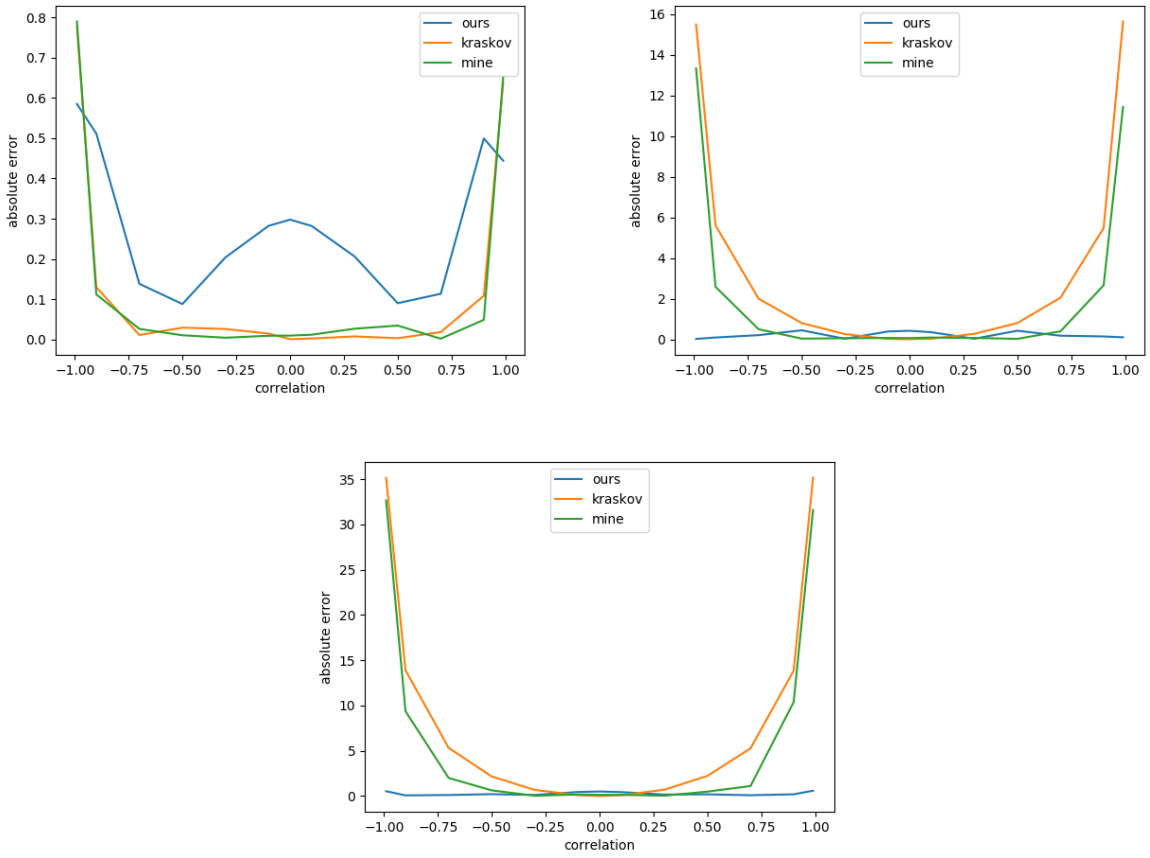


Figure 4.2: Plot of absolute error in predicted mutual information for correlated Gaussian data with 2d, 10d and 20d marginals for the MST model and baselines.

	Omniglot		MNIST	
Model	Acc	L1	Acc	L1
Baselines				
Pine	0.662 ± 0.013	0.824 ± 0.006	0.468 ± 0.004	1.244 ± 0.041
Single-Set RFF	0.631 ± 0.002	0.892 ± 0.042	0.442 ±0.012	1.363 ± 0.028
Single-Set RN	0.672 ± 0.006	0.811 ± 0.009	0.537 ± 0.098	1.097 ± 0.218
Single-Set Transformer	0.724 ± 0.003	0.733 ± 0.006	0.912 ± 0.034	0.466 ± 0.062
Union Transformer	0.630 ± 0.001	0.868 ± 0.009	0.534 ± 0.003	1.111 ± 0.006
Our Models				
Multi-Set Transformer	0.854 ± 0.008	0.543 ± 0.012	0.975 ± 0.004	0.314 ± 0.012
Multi-Set RN	**0.870** ± 0.005	**0.517** ± 0.008	0.978 ± 0.010	0.318 ± 0.056
Cross-Only	0.819 ± 0.034	0.593 ± 0.052	0.972 ± 0.002	0.313 ± 0.011
Sum-Merge	0.854 ± 0.007	0.542 ± 0.008	**0.978** ± 0.001	**0.260** ± 0.015

Table 4.2: Average accuracy and L1 error of each model on the MNIST and Omniglot counting tasks across 3 runs (higher is better for accuracy and lower for L1).

4.6.2 Image Tasks

As a demonstration of the model's ability to perform simple general set-based operations, I begin by looking at a selection of image-based tasks similar to those considered by Lee et al. [2019] and Zaheer et al. [2017]. When working with image or text data, each example was first individually encoded as a fixed-size vector using an appropriate image or text encoder, then passed through the set based model.

Counting Unique Images

For the first task, the models were given input sets consisting of images of characters. The task was to identify the number of unique characters that were shared between the two input sets of a variable number of images drawn from the MNIST [Deng, 2012] and Omniglot [Lake et al., 2015] datasets (6-10 images for Omniglot, 10-30 images for MNIST). For this task, the model used simple CNN encoders that were pretrained on the input datasets as classifiers for a short number of steps, then trained end to end with the set-based model. The multi-set models convincingly outperformed the alternatives - achieving almost 98% accuracy on the MNIST task and 83-85% accuracy on the Omniglot task (see Table 4.2), and outperforming the baselines by considerable margins. The RN-based model outperformed the transformer model by a margin of about 1.5% on the Omniglot task, and they performed equivalently on the MNIST task. The ablations performed largely similarly to the base model, with degraded performance only in the

Model	CoCo	FastText
Baselines		
PINE	0.498 ± 0.007	0.498 ± 0.002
Single-Set RFF	0.496 ± 0.000	0.497 ± 0.003
Single-Set RN	0.775 ± 0.037	0.786 ± 0.013
Single-Set Transformer	0.906 ± 0.004	0.770 ± 0.011
Union Transformer	0.929 ± 0.002	0.732 ± 0.016
Our Models		
Multi-Set Transformer	0.927 ± 0.013	**0.822** ± 0.002
Multi-Set RN	**0.935** ± 0.019	0.763 ± 0.009
Cross-Only	0.919 ± 0.012	0.810 ± 0.007
Sum-Merge	0.930 ± 0.018	0.816 ± 0.009

Table 4.3: Average accuracy and standard deviation of each model across 3 runs on the alignment tasks.

case of the Cross-Only model on the Omniglot dataset.

Alignment

While the first task was purely synthetic, this second task is representative of a general class of applications for this model - predicting alignment between two sets. The first example of this I chose was image captioning on the MSCOCO [Lin et al., 2014] dataset. The models were given a set of 8-15 images and a set of captions of the same size, and tasked to predict the probability that the two sets were *aligned* - i.e. that the given set of captions consisted of captions for the given set of images. For this task, the pretrained models BERT and ResNet-101 were used as encoders for the text and images respectively. The second example of tasks in this category were cross-lingual embeddings. Lample et al. [2018a] show that there is a geometric relationship between learned FastText [Bojanowski et al., 2016] embeddings across languages. As such, the model should be able to predict the alignment between sets of embeddings in one language and sets of embeddings in another. The model is shown a set of 10-30 embeddings in English and another set of the same size of embeddings in French. The model is then tasked to predict whether or not the embeddings in the two sets correspond to the same words.

Results on these tasks are shown in Table 4.3. The proposed multi-set models performed the highest across both tasks. Interestingly, the Single-Set Transformer and Union Transformer models performed quite well on the CoCo task (though not as well as our model), but were significantly worse on the FastText task. No other baseline aside from the Single-Set RN model performed notably better than chance. Given that the unaligned sets consisted of entirely disjoint

Model	Synthetic	Meta-Dataset
Baselines		
PINE	0.501 ± 0.002	0.505 ± 0.003
Single-Set RFF	0.501 ± 0.002	0.783 ± 0.007
Single-Set RN	0.500 ± 0.001	0.798 ± 0.064
Single-Set Transformer	0.604 ± 0.018	0.881 ± 0.009
Union Transformer	0.591 ± 0.006	0.743 ± 0.016
Our Models		
Multi-Set Transformer	0.729 ± 0.035	0.892 ± 0.014
Multi-Set RN	**0.735** ± 0.009	0.868 ± 0.011
Cross-Only	0.635 ± 0.019	**0.904** ± 0.009
Sum-Merge	0.629 ± 0.010	0.868 ± 0.007

Table 4.4: Average accuracy and standard deviation of each model across 3 runs on the distinguishability tasks.

images and captions (i.e., no images and captions overlapped), it's possible that learning whether the net sum of all embedded vectors in each set were aligned might be sufficient, and the model might not need to directly compare individual elements across sets. This might explain the high performance of the Single-Set Transformer and Union Transformer models - though interestingly, this did not hold true for the FastText task (perhaps due to the fact that the FastText task appeared to be more difficult).

Distinguishability

The last task in this category was distinguishability. Given two sets of samples, the models would be asked to predict whether the two input sets were drawn from the same underlying distribution. Two examples were again considered for this task: a synthetic dataset, and a dataset of real-world images. For the synthetic data, sets were sampled from randomly generated 8-dimensional Gaussian mixtures (Gaussian mixture parameters were generated in the same fashion as the data in Section 4.6.1). For the second example, I used Meta-Dataset [Triantafillou et al., 2019][1] - a dataset consisting of 12 other image datasets, each with many subclasses. In each case, each training example consisted of a batch of two sets of 10-30 data points. The data points would be drawn from the same distribution (the same Gaussian mixture for the synthetic data, or the same class from the same parent dataset for Meta-Dataset) with probability 1/2, and generated from different distributions with probability 1/2. The model was tasked to predict the probability of

[1]Pytorch implementation taken from Boudiaf et al. [2021]

the sets being drawn from the same distribution. For the meta-dataset task, images were first encoded using a CNN trained along with the network.

Results are shown in Table 4.4. The Multi-Set Transformer and Relation Network models performed by far the best on the synthetic task. On the image task, the Single-Set Transformer again performed very well, though not as well as the multi-set transformer model. I hypothesize this might be because the distinguishability task relies on recognizing the distribution from which the set is drawn, which is a task that might be possible to do by simply reducing each input set to a single vector and then comparing the resulting vectors.

4.6.3 Analysis

Overall, two different variants of the model were considered (RN-based and transformer-based), as well as several ablations of the transformer model. The base multi-set transformer model performed consistently well across every task, with either the highest accuracy or close to the highest accuracy. The relation-network variant of the model performed slightly better on a number of tasks, but significantly worse on many others. This variant of the model also had significant issues with memory usage, and often required very small batch sizes in order to fit on GPUs. Each of the single-set transformer, cross-only, and sum-merge models can be considered ablations of the multi-set transformer architecture. The cross-only model performed competitively or slightly better on some tasks, but similar to the RN model, it performed worse by notable margins on others. It's possible that the T_{XX} and T_{YY} blocks - which computed the internal relationships between elements in X and Y - were simply not needed for certain tasks, but very helpful for others. The sum-merge model generally performed comparably to the base model, and degraded performance by notable margins only in the case of the distinguishability tasks. This is not entirely unexpected, given that it is the most minor of the ablations, and represents only a small change to the structure of the model.

4.6.4 Scaling

One important consideration when using set-based architectures is how the architectures will scale to large set sizes. Given input sets of size n and m and with model dimension d (assuming that d_{hidden} is of approximately the same order as d_{latent}), Table 4.5 shows the scaling properties of each model with the set sizes and latent dimension. The PINE and Single-Set RFF architectures are the fastest, scaling linearly with set size. All other models contain terms that are quadratic with set size, as they need to compare each element in one set to each element in another set (or the same set). Of these models, the transformer-based models (i.e. Single-Set Transformer, Union Transformer, Multi-Set Transformer, Cross-Only and Sum-Merge) all require approximately $(n+m)d^2+(n+m)^2d$ operations (though some need only nmd or $(n^2+m^2)d$ due to omitting same-set

Model	Ops. Scaling
PINE	$O\left((n+m)d^2\right)$
Single-Set RFF	$O\left((n+m)d^2\right)$
Single-Set RN	$O\left((n^2+m^2)d^2\right)$
Single-Set Transformer	$O\left((n+m)d^2+(n^2+m^2)d\right)$
Union Transformer	$O\left((n+m)d^2+(n+m)^2d\right)$
Multi-Set Transformer	$O\left((n+m)d^2+(n+m)^2d\right)$
Multi-Set RN	$O\left((n+m)^2d^2\right)$
Cross-Only	$O\left((n+m)d^2+nmd\right)$
Sum-Merge	$O\left((n+m)d^2+(n+m)^2d\right)$

Table 4.5: Scaling of the number of operations required for each model with set sizes n, m and dimension d.

terms or cross-set terms, the effect is still a net quadratic scaling with set size). The relation network models have the worst scaling properties, as they scale quadratically with the product of both set size and latent dimension. These scaling properties will remain the same if these architectures were generalized to $K > 2$ sets, with $n + m$ simply replaced by the total size of the union of all input sets.

While the PINE and Single-Set RFF architectures have the best scaling properties, they also demonstrate by far the worst performance, achieving results no better than chance on many of the image tasks. All of the transformer models share approximately the same scaling properties, scaling quadratically with the set sizes n and m. This is a well-known property of transformer-based models, and a site of active research. Many previous works have proposed ways to reduce this quadratic dependency, and find approximations that allow these models to require only linear time (Wang et al. [2020], Choromanski et al. [2021],Kitaev et al. [2019], etc...). All of our proposed transformer models are compatible with any of these approaches, though we leave such explorations for future work. The Relation Network models are the most troublesome, as they scale quadratically with the product of both the model dimension and the set size. While these models do perform very well on some of the tasks discussed, they perform poorly on others, and overall the Multi-Set Transformer models exhibit both better scaling properties and more consistent performance across tasks.

4.7 Discussion and Conclusion

The Multi-Set Transformer model proposed in this chapter performs well at estimating a variety of distance/divergence measures between sets of samples, even for quantities that are notoriously difficult to estimate. It clearly outperforms existing multi-set and single-set architectures, and

beats existing KNN-based estimators in the settings under consideration. In an ideal case, the model could be pretrained once and then applied as an estimator for, e.g., KL divergence in a diverse array of settings. This is a primary area of focus going forward.

Since this model is a universal approximator for partially permutation equivariant functions, its applications are far broader than simply that of training estimators for divergences between distributions. I chose to showcase a number of simple applications with image data, but these are merely meant to be representative of larger classes of applications. The applications in terms of distinguishability, for example, are highly reminiscent of GANs [Goodfellow et al., 2014], and the FastText task from the alignment section bears some similarities to existing work in which GANs are used [Lample et al., 2018a], where this model might lead to improvements. These connections will be explored further in the following chapter.

Chapter 5

SetGAN

5.1 Introduction

The previous chapter introduced a flexible, general-purpose architecture for computing functions on multiple sets, and showcased a number of possible applications on which it could be highly effective. This chapter will highlight just how powerful some of these applications can be, by using this architecture as the foundation for a novel type of generative model: a conditional, set-based generative adversarial network for few-shot image generation.

Consider the 'distinguishability' task discussed in Section 4.6.2, where the model is trained to take two sets as input and determine whether they are drawn from the same distribution. This task is highly reminiscent of the role that a discriminator plays within a generative adversarial network (see Section 2.1.3 for a review of generative adversarial networks). In a regular discriminator, the model accepts only a single image as input at a time - the second distribution is stored *implicitly* in the discriminator's parameters over the course of training. This new formulation can instead *explicitly* compare a *set* of images to another set. This suggests a radically different approach to GAN training: rather than learning to encode a single data distribution and generate single samples unconditionally, consider training a GAN which can accept a set of images as input, and generate a set of output images which are samples from the same distribution. This formulation can act as a form of few-shot or zero-shot learning, similar to the text-to-image paradigm used by celebrated models such as DALL-E 3[Betker et al., 2023] or Stable Diffusion [Rombach et al., 2022].

The architecture proposed in this chapter is a novel few-shot image generation model that is trained to generate images conditioned on sets of reference images of unseen classes. This model, titled "SetGAN", learns to extract relevant features from the unseen reference sets, then generate high-quality, diverse images similar to the reference images at inference time. Once pretrained on

a given image dataset, SetGAN can then generate any number of images for a variety of unseen reference classes, all without any further training or finetuning.

Conceptually, SetGAN uses a similar framework to models such as DAGAN [Antoniou et al., 2017], following a conditional GAN-based approach where the generator generates images conditioned on a given input image, and the discriminator learns to distinguish between the generated images and other true images from the same class. The difference lies in the set-based nature of the proposed approach - SetGAN can condition its generations on multiple reference images rather than just a single image. This allows the model to better understand the variations within the reference class and produce diverse sets of output images conditioned on that class. The discriminator is also able to compare the generated sets of images to the reference set, and judge the generated sets based not only on the individual images' similarity to the reference class, but also on the diversity and factors of variation within each set - leading to generations that more closely match the variations within the true reference class.

Existing works such as AGE [Ding et al., 2022] or DeltaGAN [Hong et al., 2020a] also frequently rely on learning factors of variation within a typical reference class at training time, then applying those variations to a single image at inference time. This makes the assumption that the factors of variation within a given class at training time will be the same as for the unseen test classes - an assumption that does not always hold. Works that follow this methodology also have a tendency to produce generations that are highly similar to the reference image, limiting their diversity. SetGAN does not suffer from these limitations, and can generate truly novel and diverse outputs that are similar to the reference class as a whole without simply reproducing elements of a single input image. SetGAN can also perform inference conditioned on inputs that are grouped across different classes from the dataset (see Section 5.5 and Figure 5.1). This more general form of inference is very challenging for many existing models, due to their strong assumptions of similarity between the organization of test classes and training classes, and their inability to condition on more than one image at a time.

5.2 Related Work

5.2.1 Few-shot GANs

As discussed in Section 2.1.5, revious works on few-shot image generation using GANs generally fall into three categories: optimization-based methods, fusion-based methods, and transformation-based methods. Optimization-based methods [Clouâtre and Demers, 2019, Liang et al., 2020] use meta-learning techniques [Finn et al., 2017] to fine-tune their generative models on small amounts of data, but do not produce results competitive with other approaches. Fusion-based methods [Hong et al., 2020c, Gu et al., 2021, Yang et al., 2022b] condition on several input images by starting with a single base image and incorporating local features from other reference images.

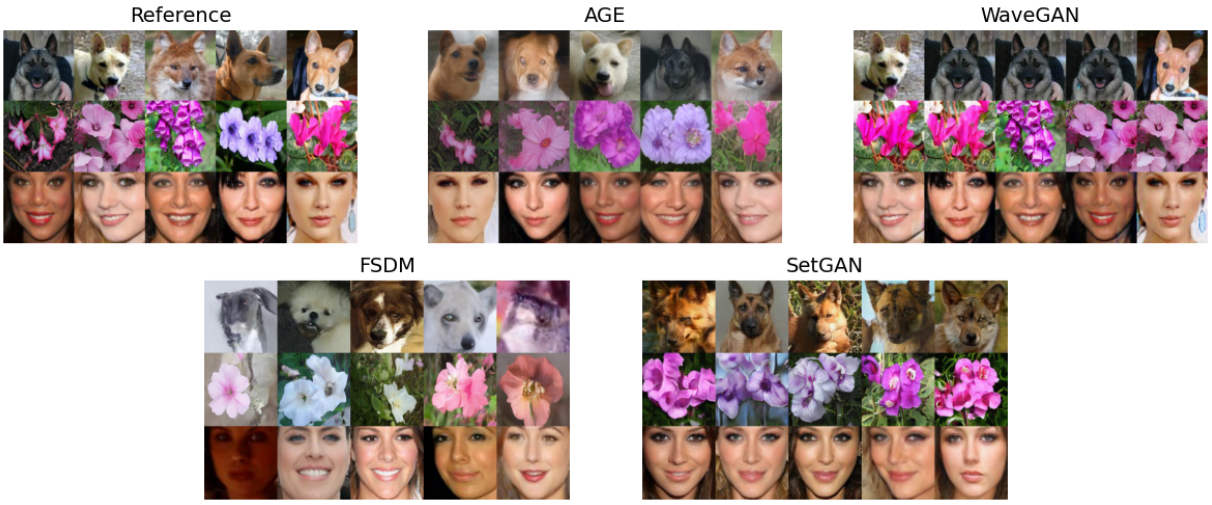


Figure 5.1: Examples of generations using images across many different test classes that share similarities according to other features - e.g. women with heavy eye makeup, animals with long upward-pointing ears, or clusters of pink and purple flowers. SetGAN generates diverse output images that faithfully reproduce these features, whereas other baselines either copy the reference images or generate images which are not faithful to the shared features.

These methods are highly dependent on the images they condition on and sometimes struggle to generalize beyond the features in the input images. Transformation-based methods [Ding et al., 2022, Hong et al., 2020a, Antoniou et al., 2017] learn transformations during training that mimic the typical factors of variation within each training class, then apply these learned transformations to a single test image. These methods can be highly successful at one-shot image generation, but make strong assumptions about the similarity in factors of variation between classes that may not generalize to more diverse datasets. Using only a single image to condition on can also limit diversity, as each generation may be only a slight transformation of the given input image.

5.2.2 Diffusion models

Many diffusion-based approaches such as DALL-E 3 [Betker et al., 2023] and Stable Diffusion [Rombach et al., 2022] have achieved incredible success at text-to-image generation, generating diverse high-resolution images from a wide variety of text-based prompts. While some limited equivalents exist for image-to-image generation such as inpainting Rombach et al. [2022] or image translation Saharia et al. [2022], Sasaki et al. [2021], no such large-equivalents exist for large-scale true few-shot generation of images conditioned on sets of unseen images. Giannone et al. [2022] do propose a framework for few-shot generation with diffusion models, however their model is tested

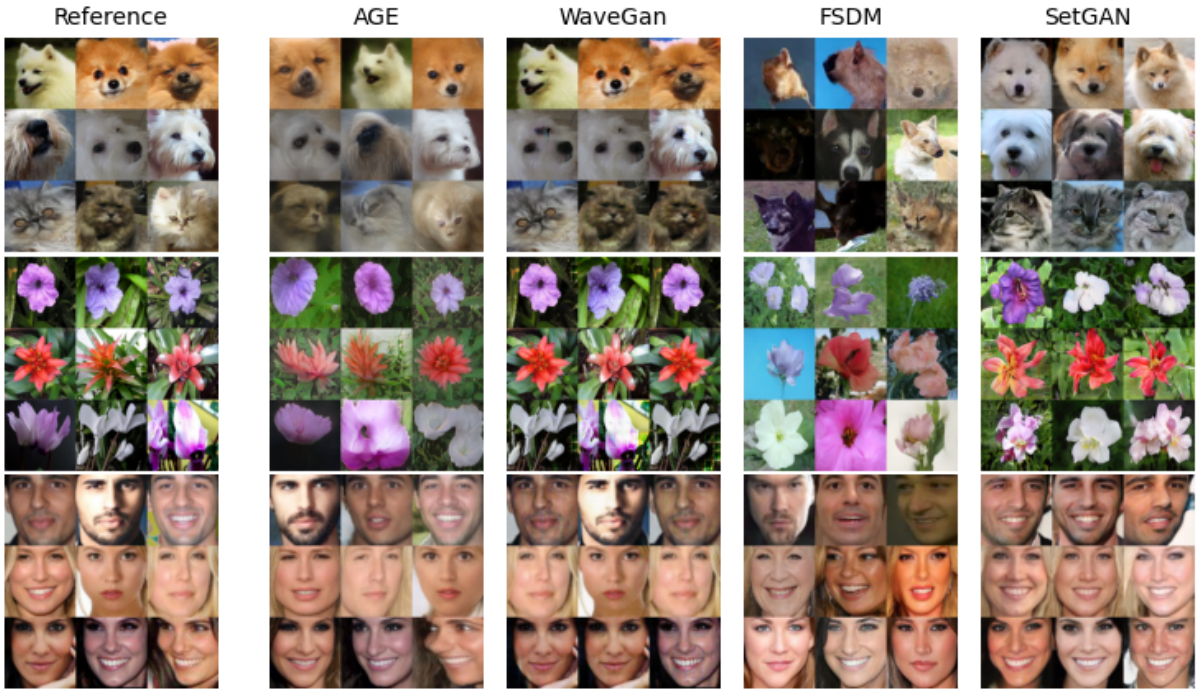


Figure 5.2: Generations from AGE, FSDM, WaveGAN and SetGAN conditioned on 3 reference images from unseen classes of each of the Animal Faces, Flowers and VGGFace datasets.

only on very low-resolution datasets. Their model is also extremely slow to perform inference even at lower resolution scales, and this problem is compounded at higher resolutions.

5.2.3 Image translation

A closely related task to few-shot image generation is image-to-image translation. In this task, the goal is to translate images from one domain to a new domain, often in a few-shot setting. This frequently takes the form of adapting models pretrained on the source domain to the target domain via a minimal number of examples [Li et al., 2020, Ojha et al., 2021]. While this approach to few-shot image translation is a distinct task from few-shot image *generation*, some approaches such as FUNIT [Liu et al., 2019a] have combined these approaches by seeking to translate images between different classes of the same dataset.

5.2.4 Set-based approaches in GANs

Ferrero et al. [2022] proposed an approach where the discriminator is allowed to make decisions based on a set of samples from either training data or the generator in order to increase stability and prevent mode collapse. While this work does also examine the idea of leveraging equivariances for generation, it focuses on improving the stability of unconditional generation rather than performing conditional set-based generation.

5.3 Methods

Formally, the setting under consideration is similar to that laid out in Section 2.1.5, where there is a dataset $\mathcal{D}$ divided into a number of classes $\{\mathcal{C}_i\}$, which are each composed of some $n_{\mathcal{C}_i}$ images. These classes are partitioned into a disjoint training set $\mathcal{D}_{\text{train}}$ and test set $\mathcal{D}_{\text{test}}$. During training, a class $\mathcal{C} \sim \mathcal{D}_{train}$ is sampled, and from this class are drawn two (disjoint) sets of images: a *reference set* $R \in \mathcal{C}^n$, and a *candidate set* $C \in \mathcal{C}^m$.

As in the case of a traditional GAN (see Section 2.1.3), the algorithm consists of a *discriminator* and *generator* model, trained jointly via an adversarial minimax game. The generator is a mapping $G : \mathbb{R}^{m \times d} \times \mathcal{G}^n \to \mathcal{G}^m$ that accepts as input a set of n reference images as well as m latent vectors, and produces a set of m output images. The discriminator is then a model $D : \mathcal{G}^n \times \mathcal{G}^m \to [0, 1]$ which accepts two sets of images, and predicts the probability that the two sets are drawn from the same underlying distribution. The two models are trained via the following minimax game:

$$\min_G \max_D \mathbb{E}_{\mathcal{C} \sim \mathcal{D}_{train}} \mathbb{E}_{R \sim \mathcal{C}^n, C \sim \mathcal{C}^m} \left[\log D(R, C) \right] + \log \left(1 - D(R, G(R)) \right) \tag{5.1}$$

5.3.1 Architecture

Generator

The generator model follows an encoder-decoder structure similar to that of a U-Net [Ronneberger et al., 2015]. StyleGAN2's generator is taken to be the base decoder architecture (discussed in Section 2.1.3), which maps a series of $k = 18$[1] 512-dimensional *style vectors* to a single output image, with each style vector controlling the convolutions at a particular stage of the decoding. It has become common to refer to the extended latent space formed by the concatenation of these k vectors as $\mathcal{W}+$. Similar to Ding et al. [2022], the pixel2style2pixel (pSp) encoder proposed by

[1] Note that the default 18 style vectors correspond to a generation size of 1024x1024 px. All experiments use a generation size of 256x256, and thus in practice use a truncated $\mathcal{W}+$ space of 14 vectors.

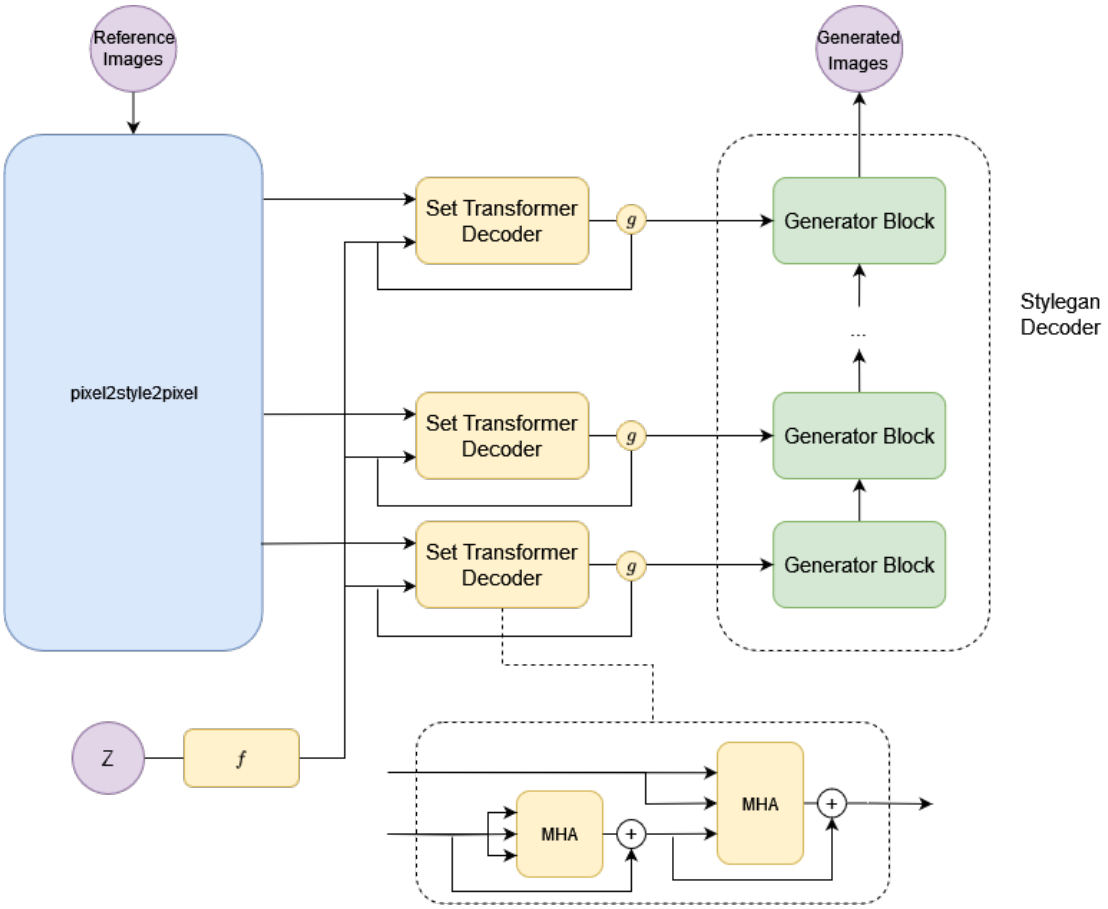


Figure 5.3: Diagram of the SetGAN generator. The pSp encoder maps each input image to the latent space $\mathcal{W}+$. The input style vectors are then passed through the StyleGAN2 mapping network, then passed to a series of conditioning networks which compute conditional styles for each layer of the decoder by attending to the appropriate output layer of the pSp encodings. These conditional styles then become the inputs to the StyleGAN2 generator, which decodes them into images.

Richardson et al. [2021] (see Section 2.1.3) is taken to be the encoder model, which maps a single input image into the space $\mathcal{W}+$ (although this could also be done with other similar encoders such as E4E [Tov et al., 2021] or ReStyle [Alaluf et al., 2021]). This encoder-decoder model is then augmented with a series of attention-based conditioning networks, consisting of a stack of 2 transformer decoder blocks for each of the k style vectors, each surrounded by a skip connection. A diagram of this generator architecture is shown in Figure 5.3.

Given a set of n reference images, each image R_i is encoded into a latent code: $C_i = \text{pSp}(R_i) = \{c_i^0, ..., c_i^k\}$, with the notation c_i^ℓ for style ℓ of the encoding of image i, and $C^\ell = \{c_i^\ell\}$. To generate a set of m candidate images, these latent codes are then combined with a series of m sampled noise vectors $Z = z_{1,...,m} \sim N(0, 1)$. These noise vectors are passed through the decoder's mapping network to generate the base style vectors $W = \{f(z_j)\}$, in the same fashion as StyleGAN. Now, the model takes the base style vectors W and transforms them by attending to the features of the reference encodings C. At each layer ℓ, the model computes the corresponding *conditional style vector*:

$$\omega^\ell = g^\ell(W, T^\ell(W, C^\ell)) \tag{5.2}$$

where T^ℓ is the transformer block associated with the ℓ-th style vector, and g^ℓ is a linear layer applied to the concatenation of the base style vector with the output of the attention blocks. These k conditional style vectors then form the conditional encoding $\omega \in \mathcal{W}+$, which becomes the input to the StyleGAN2 decoder.

In order to stabilize the early training, the layers g^ℓ were initialized as $g^\ell = [I; \Sigma]$, where I is the identity mapping and Σ is a matrix of gaussian noise with $\sigma = 0.2$. This ensures that near the beginning of training the decoder will be given approximately the base style vector, with the effect of the conditioning layers being incorporated gradually as the network learns.

Discriminator

The discriminator now takes the form $D : \mathbb{R}^{n \times d} \times \mathbb{R}^{m \times d} \to \mathbb{R}$, mapping an input reference set R and candidate set C to a single scalar output. To do this, the architecture must accept as input multiple permutation-invariant sets - an ideal use case for the multi-set transformer model proposed in Chapter 4. The input images are first passed through a convolutional encoder to encode each image within the two input sets as fixed-sized vectors, then passed through the multi-set transformer network. Finally, the two pooled output vectors are concatenated and fed to a linear output head. Skip connections are used to connect the outputs of the convolutional encoder to the latent vectors just before the pooling layer of the multi-set transformer. The convolutional architecture of the StyleGAN2 discriminator is used as the base architecture for the discriminator encoder. A diagram of this architecture is shown in Figure 5.4.

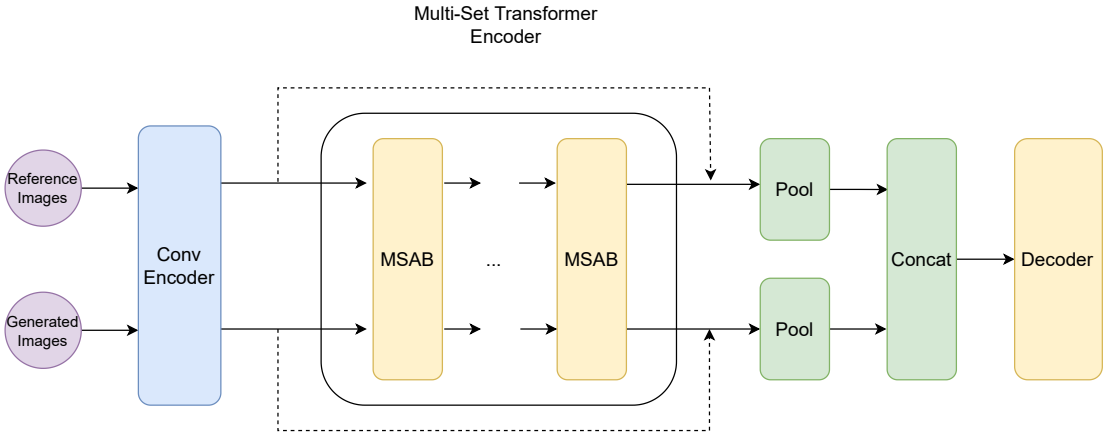


Figure 5.4: Diagram of the SetGAN discriminator. Sets of input images are encoded as fixed-size vectors using a convolutional network. These sets of vectors are then passed through a Multi-Set Transformer (see Chapter 4) consisting of several multi-set attention blocks, followed by a pooling operation performed on each set. These outputs are then concatenated and passed through a feedforward decoder layer to produce a scalar output.

5.3.2 Latent space truncation

SetGAN uses latent space truncation for inference, in a similar manner to StyleGAN2. In order to improve the quality of the generated results, style vectors are shifted towards the mapping network's mean style vector $\bar{w}$ by a given factor λ. Unlike StyleGAN2, however, this truncation may be applied to SetGAN in two ways: either pre-conditioning or post-conditioning.

Given a base style vector w, pre-conditioning truncation is applied in the same manner as it is for StyleGAN: the latent vector is transformed by the procedure:

$$w \to \bar{w} + \lambda_1(w - \bar{w}) \tag{5.3}$$

This ensures that the base style vector used to generate the output images remains in the well-explored region near the mean, and leads to generations of higher quality but slightly lower diversity.

In addition to this, however, truncation may also be applied post-conditioning, to shift the final conditional styles w' towards the mean style vector as follows:

$$w'_j \to \bar{w} + \lambda_2(w'_j - \bar{w}) \tag{5.4}$$

This has a large effect on output quality, but at a much greater cost to output diversity.

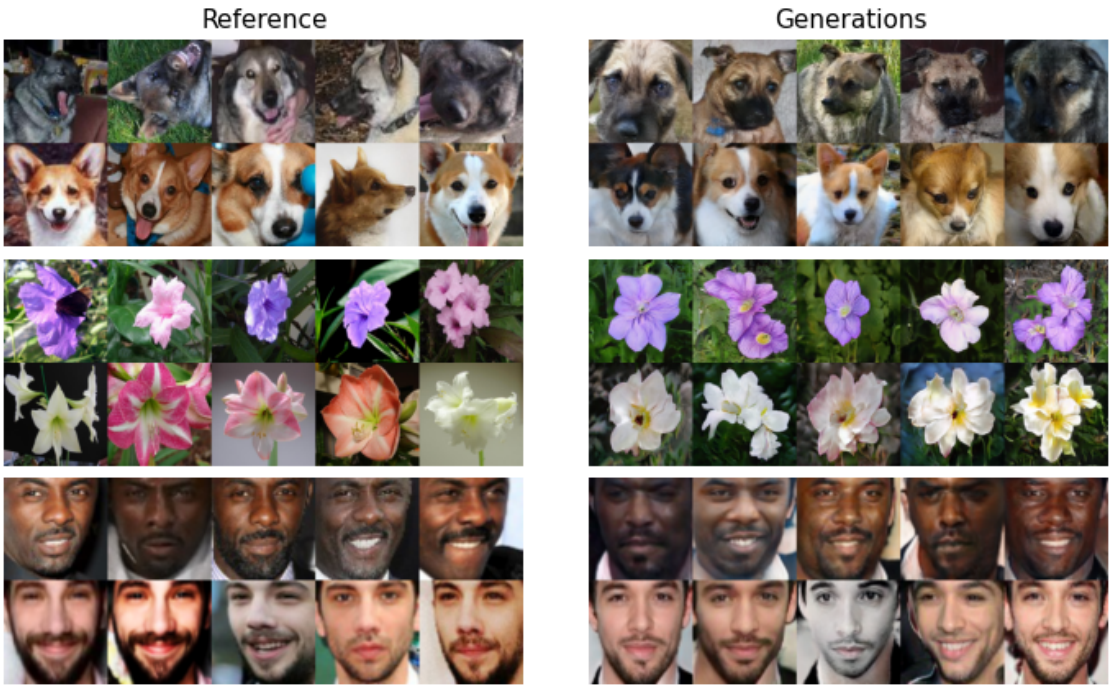


Figure 5.5: Additional generations from SetGAN using reference sets of 5 images.

Both λ_1 and λ_2 truncation provided significant benefits on the Flowers dataset, improving sample quality and MiFID score by considerable amounts. λ_1 truncation improved sample quality and MiFID score for the Animal Faces dataset, but this was not used due to the tradeoff in sample diversity. Truncation provided little benefit on the VGGFace dataset.

5.4 Experiments

5.4.1 Setup

First, a StyleGAN2 model [Karras et al., 2020] is pretrained on the given dataset at 256x256 resolution. Then, a corresponding pSp [Richardson et al., 2021] network is trained to perform GAN inversion on the pretrained StyleGAN2 model to act as the encoder (see Section 2.1.3). These pretrained models are used to instantiate the encoder and decoder for our generator, and are then frozen. The discriminator from the StyleGAN2 model is also used to initialize the encoder for our multi-set discriminator model. These models are then trained following Eq. 5.1 until convergence. The model uses the base training scheme of StyleGAN2 [Karras et al.,

2020] to train SetGAN, using a non-saturating loss with R1 gradient penalty ($\lambda = 10$) and path length regularization. Reference and candidate sizes are sampled uniformly from size 7-10 and 4-6 respectively, so that the model does not learn to assume a specific input size. Models are trained on NVIDIA A40 GPUs with the ADAM optimizer, with a batch size of 2 and learning rate 1e-3.

For inference, latent space truncation is used - as discussed in Section 5.3.2. Results in this work were obtained with $\lambda_1 = \lambda_2 = 0.8$ for the Flowers dataset and $\lambda_1 = \lambda_2 = 1$ for the other two datasets.

5.4.2 Datasets

In keeping with prior works [Hong et al., 2020a, Ding et al., 2022, Yang et al., 2022b, Gu et al., 2021], results are reported on the Animal Faces [Liu et al., 2019a], Flowers [Nilsback and Zisserman, 2008] and VGGFace [Cao et al., 2018] datasets. The same train and evaluation splits proposed in Hong et al. [2020a] are used on Animal Faces and Flowers. For VGGFace, the evaluation set is restricted to the final 53 classes due to the computational requirements of inference for the FSDM baseline.

5.4.3 Baselines

Previous papers often compare directly against the results reported in other prior works. Unfortunately, I found that many existing few-shot image generation models contain significant inconsistencies in the methodologies used for evaluation. For example, the LPIPS metric can be evaluated using either AlexNet or VGG activations, which cannot be compared directly against each other. I found that previous works such as F2GAN and DeltaGAN used AlexNet activations to measure LPIPS score, while WaveGAN and LoFGAN used VGG activations. These previous works also generated results at a variety of different resolutions - and some were then rescaled before applying the metric, while others were not. AGE generated outputs at 256x256, while other works performed their generations at 128x128. WaveGAN, LoFGAN and AGE also rescaled images to 32x32 before computing LPIPS distances, while other works did not.

Different works also used different code for compiling generated images and rescaling them to the target size of the pretrained models used to obtain vector embeddings. As discussed in Parmar et al. [2022], the details of these steps can have a substantial impact on the final results, and inconsistent methodologies between papers can lead to significant discrepancies. In addition to these inconsistencies in methodology, I found that in many cases I was unable to reproduce the reported scores of existing works - despite using code and checkpoints provided by the authors, and consulting with the authors directly.

	$\text{MIFID}_{\text{Inc}}$			$\text{MIFID}_{\text{CLIP}}$			LPIPS		
	1	3	10	1	3	10	1	3	10
Animal Faces									
AGE	71.35	62.23	56.55	14.09	12.77	11.74	0.403	0.560	0.550
WaveGAN	2327.29	1057.39	529.08	603.44	242.81	136.09	0.000	0.421	0.556
FSDM	75.68	73.93	77.37	8.78	8.59	10.38	0.604	0.608	0.609
SetGAN	***61.51***	***52.34***	***47.18***	***6.56***	***5.84***	***5.28***	***0.614***	***0.615***	***0.618***
Flowers									
AGE	81.87	70.15	65.48	16.82	15.03	14.31	0.379	0.553	0.608
WaveGAN	2653.56	1305.31	699.96	851.14	373.62	182.11	0.000	0.484	0.635
FSDM	69.25	62.35	61.47	10.69	10.26	10.18	***0.681***	***0.699***	***0.704***
SetGAN	***62.44***	***59.84***	***59.31***	**10.68**	**9.79**	***9.88***	0.617	0.624	0.628
VGGFace									
AGE	22.12	18.39	16.76	8.20	6.51	5.94	0.260	0.369	0.406
WaveGAN	852.70	36.97	23.12	17.50	9.40	6.65	0.000	0.325	0.430
FSDM	10.51	11.26	12.48	***3.28***	3.47	3.76	0.451	0.448	0.447
SetGAN	***9.60***	***7.93***	***7.83***	4.16	***3.12***	***2.82***	***0.463***	***0.461***	***0.471***

Table 5.1: Scores for conditional generation on the Animal Faces, Flowers and VGGFace datasets for each of the four baselines, conditioned on reference sets of size 1, 3 and 10. Results were averaged over three different random partitions of the test set into D_{eval} and D_{ref}. Lower scores are better for MIFID, higher is better for LPIPS. The best score in each category is bolded. Scores that exceed all others by at least one standard deviation are italicized.

Due to these significant inconsistencies in existing results and methodologies, I chose a selection of the highest performing models from the literature as baselines and computed metrics for each model myself by running the provided models under identical settings to ensure a fair comparison. Code and checkpoints provided by the authors were used wherever possible. The AGE [Ding et al., 2022] and WaveGAN [Yang et al., 2022b] models were selected as representative of the highest-performing GAN-based approaches in the literature, as well as the diffusion-based approach FSDM [Giannone et al., 2022].

5.4.4 Evaluation procedure and metrics

For each dataset, the test dataset is divided by class, then each class is partitioned into a reference set $\mathcal{D}^c_{\text{ref}}$ of size n_{ref} and evaluation set $\mathcal{D}^c_{\text{eval}}$ of size n_{eval}. For each such class, the model is used to generate n_{gen} new images, conditioned on images from D^c_{ref}, to form D^c_{gen}. For some metrics (such as FID or MIFID), these images are then aggregated into a single D_{eval} and D_{gen}. These image

sets are then used to evaluate the generations using a variety of metrics. For the experiments in this work, $n_{\text{eval}} = n_{\text{gen}} = 128$, and n_{ref} varied by experiment (see Section 5.5 for further details). If the number of images in a given evaluation set was lower than 128, all images were used. Each of these evaluations was performed three times with different randomly chosen partitions for each class. The most common metrics used to evaluate models for this purpose are the Frechet Inception Distance [Heusel et al., 2018], or FID, and Learned Perceptual Image Patch Similarity [Zhang et al., 2018], or LPIPS. These metrics are defined in detail in Section 2.1.5.

Limitations of existing metrics

While the aforementioned FID and LPIPS scores are the most widely-used metrics among existing literature, these metrics have significant flaws - particularly FID. Existing works such as Rangwani et al. [2023] and Kynkäänniemi et al. [2023] have already identified flaws in the FID metric related to its bias towards particular features specific to the ImageNet classes it was trained on, leading to arbitrary manipulation of scores via imperceptible changes in generated images. Rangwani et al. [2023] also demonstrate that traditional FID scores sometimes strongly emphasize fidelity over diversity in few-shot generation, and propose FID_{CLIP} in order to address this issue - a modification to the FID method using the large multi-modal CLIP model [Radford et al., 2021] in place of the Inception backbone.

While FID_{CLIP} is an improvement over traditional Inception-based FID scores in some respects, it does not wholly solve the problems with the FID metric. In the experiments, I found that models that generate identical or nearly identical copies of the reference images consistently achieved extremely low FID scores. To test this, I measured the FID scores between the evaluation sets and generated sets constructed solely by copying N random images sampled with replacement from the reference set (denoted as the "Copy" baseline). I also tried the same experiment if the copied images were subjected to a small, imperceptible level of Gaussian noise (denoted as the "Noisy" baseline). I then compared these scores to the best scores among all trained models[2], as well as a theoretical maximum score given by comparing two randomly selected partitions of the test set. As shown in Table 5.2, this baseline of simply copying the reference images achieves FID scores close to the theoretical maximum, and matches or exceeds the score of the best trained baseline in almost every case. As a result, it is clear that *traditional FID scores are not a reliable metric for measuring the performance of few-shot generation.*

[2]WaveGAN was excluded from this, given WaveGAN's propensity to also generate nearly-identical copies of the reference images.

	FID_{Inc}	FID_{CLIP}	$MiFID_{Inc}$	$MiFID_{CLIP}$	FLS_{Inc}	FLS_{CLIP}
			Animal Faces			
Best Model	46.20	5.02	46.20	5.02	125.93	133.38
Noisy	24.05	6.94	109.66	14.44	126.72	143.02
Copy	20.44	1.59	17714.97	1335.72	229.81	168.31
True	13.55	1.05	13.56	1.05	114.93	127.70
			Flowers			
Best Model	57.12	9.29	57.75	9.29	142.59	144.41
Noisy	37.06	4.21	394.61	22.37	144.18	143.83
Copy	36.98	2.56	23408.14	1737.27	164.80	169.98
True	30.18	1.69	30.18	1.69	139.21	132.39
			VGGFace			
Best Model	8.87	2.95	8.87	2.95	134.90	129.20
Noisy	47.96	12.20	56.51	12.64	148.23	145.64
Copy	9.54	0.77	4849.92	438.17	170.63	176.04
True	7.15	0.58	7.15	0.58	134.58	119.17

Table 5.2: Scores for synthetic baselines using a variety of performance metrics. Methods that simply copy the reference set ("Noisy" and "Copy") are disproportionately favored by many scoring methods, outperforming most trained models and even approaching the score for the true test set. MIFID scores are discussed in Section 5.4.4.

Alternative metrics

These issues have also been identified in many other previous works, in the context of training set memorization. Works such as Gulrajani et al. [2020], Bai et al. [2021] and Jiralerspong et al. [2023] have discussed the tendency for traditional GAN evaluation metrics such as FID to overvalue fidelity and fail to penalize training set memorization. The same tools proposed in these papers to measure generalizability beyond the training set can also be equivalently applied here for the reference set. Of the methods discussed in these works, several have obvious difficulties or are not applicable in this case. Conditional FID [Soloveitchik et al., 2022] requires an FID calculation to be computed over the reference set, which does not work for cases with small reference sizes due to the instability of the FID calculation with small numbers of samples. Neural network divergences [Gulrajani et al., 2020] are architecture-specific, and must be trained repeatedly for each inference setting. This makes them different to compare across different models and publications, as well as costly to evaluate. I thus focus on two metrics: MiFID [Bai et al., 2021] and Feature Likelihood Score [Jiralerspong et al., 2023].

Feature Likelihood Score [Jiralerspong et al., 2023] uses a method similar to Kernel Density Estimation to fit a Gaussian Mixture density to the generated samples. The covariances of the mixture components are chosen to maximize the likelihood of the points from the reference set, ensuring that the density will be highly concentrated if the samples are simply copied from the reference data. The score is then calculated by evaluating the likelihood of the test data under the generated density. This scoring method is an interesting candidate, but fails to sufficiently penalize copying - particularly in cases where imperceptible perturbations are applied to the copied image. As shown in Table 5.2, the FLS scores for the "noisy" synthetic baseline nearly match those of the best trained models across multiple datasets.

MIFID

MiFID uses the standard Frechet Inception Distance, scaled by a multiplicative penalty calculated from the similarities between the generations and the reference images:

$$\text{MiFID}(S_g, S_t) = m_\tau(S_g, S_t) \cdot \text{FID}(S_g, S_t) \tag{5.5}$$

wherein S_g is the generated set, S_t is the training set (or reference set, in the case of conditional generation), FID is the standard Frechet Inception Distance, and m_τ is the penalty factor. Specifically, m_τ is defined by:

$$s(S_g, S_t) = \frac{1}{|S_g|} \sum_{x_g \in S_g} \min_{x_t \in S_t} 1 - \frac{|\langle x_g, x_t \rangle|}{|x_g| \cdot |x_t|} \tag{5.6}$$

$$m_\tau = \begin{cases} \frac{1}{s(S_g, S_t)+\epsilon} & s(S_g, S_t) < \tau \\ 1 & \text{else} \end{cases} \tag{5.7}$$

This metric penalizes models that simply reproduce reference images by adding a multiplicative penalty based on the average cosine similarity between the generated images and the nearest reference image. As shown in Table 5.2, this metric successfully penalizes models that simply copy the inputs, while keeping the original FID scores otherwise intact.

I adopt this metric as a drop-in replacement for FID, with the threshold τ determined by the average scale of the test set S_{test}. For each dataset, the test set is divided into two partitions of equal size, S_1 and S_2, then calculate the base score value $\tau_0 = s(S_1, S_2)$ using Equation 5.6. To ensure that models which produce results on a similar scale of variation to the test set are not unfairly penalized, models are only penalized if their scores are at least one standard deviation lower than the mean similarity scale (i.e. $\tau = \tau_0 - \sigma$, where σ is the standard deviation of the summand in Eq. 5.6).

5.5 Results

5.5.1 Quantitative Results

Results for all baselines are shown in Table 5.1. Results are shown for reference sizes of 1, 3 and 10, across each of the 3 datasets. As shown in the table, SetGAN outperforms all existing baselines across nearly all datasets and reference sizes - in some cases by significant margins. In addition to SetGAN's results being of high fidelity and quality, they are also highly diverse. SetGAN achieves high LPIPS scores across all datasets, outperforming or matching all other approaches in almost all settings. The only exception to this is the Flowers dataset, where the FSDM model achieves higher diversity scores. This is likely due to its poor fidelity with the reference class, as discussed in Section 5.5.2.

Notably, WaveGAN performs markedly worse than the other models under the MiFID score - largely due to it being heavily penalized for its tendency to produce nearly-indistinguishable copies of the reference images (see Section 5.5.2). While other models such as AGE often produce images very similar to the reference images, they were not copies, and as such did not fall under the threshold to be penalized.

While Inception-based scores and CLIP-based scores generally ranked models similarly, there were some cases where they demonstrated interesting differences - particularly on the VGGFace dataset. One possible explanation for this might be that the Inception network trained on ImageNet-1k data in which images of human faces were infrequent - unlike CLIP, which used many diverse image datasets from across the internet.

5.5.2 Qualitative Results

Test images with similar factors of variation to the training classes

Figure 5.2 shows images generated by the four models conditioned on reference images from the test classes of each of the three datasets considered. For these experiments, all reference images were drawn from the same unseen test class - measuring the models' effectiveness at generalization along similar factors of variation to the training classes. As shown in the figure, SetGAN generates diverse, high quality images, and avoids many of the struggles that other models demonstrate. Models such as AGE and WaveGAN often simply copy one of the input images, or generate small, subtle variations on it. This causes their generations to be limited in diversity, particularly when conditioned on only a small number of images. WaveGAN in particular very frequently copies the reference image almost exactly, differing from it only in imperceptible high-frequency perturbations. FSDM does succeed at generating diverse images, but often struggles to closely match the input class. This is particularly notable in the results from the flowers dataset, where its generations were often starkly different from the reference class.

Test images with different factors of variation from the training classes

In addition to evaluating the models' generations given images from the same unseen class, we can also examine how the models perform given images from *different* classes. As in the training data, each test class corresponded to images of a single type or breed of animal (for the Animal Faces dataset), a single type of flower (for the Flowers dataset), or a single individual (for the VGGFace dataset). I selected three groups of images wherein each image was taken from a different class, but all images shared common traits. In the first example, the images consisted of animals of many different types or breeds, with the shared trait being long upward-pointing ears. The second example contained flowers of a variety of types, in which all images contained clusters of multiple pink or purple flowers. The third contained images of many different women who were all wearing bold, dark eyeshadow. The resulting generations are shown in Figure 5.1.

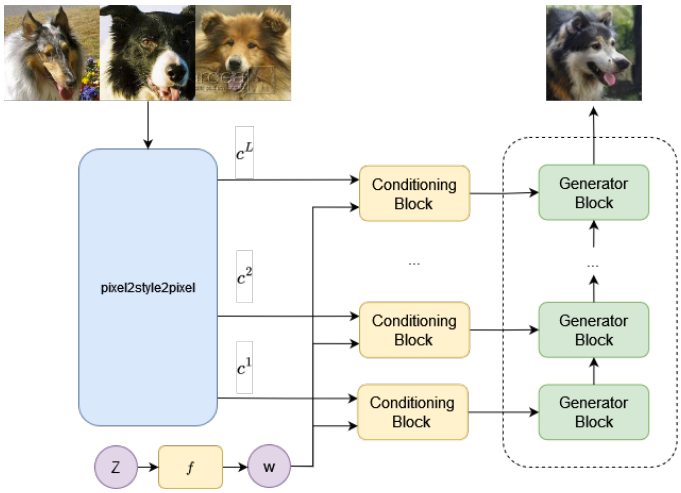
In all cases, SetGAN accurately reproduced the target features while generating a diverse range of output images. Other baselines which conditioned on a single image (i.e. AGE and WaveGAN) each struggled with this - again generating output images either identical to the inputs or very similar with subtle variations. These subtle variations would sometimes lead to deviations from the target features, as the models did not have multiple images to compare to in order to identify which features were shared. For example, the generations from AGE led to some images with short ears, single flowers, or less distinctive makeup. The FSDM baseline *was* also capable of incorporating features from multiple images, but the results were often of lower quality and were less faithful to the target features than those of SetGAN.

5.5.3 Inference Time

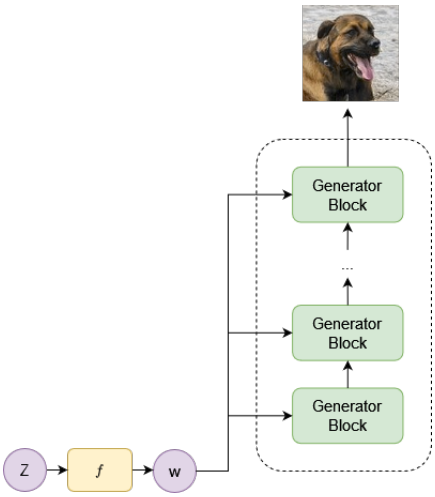
To measure the computational efficiency of each model, the time required for each model to generate a single batch of inputs was recorded, with 3 generated images per set and a batch size of 20. As shown in Table 5.3, all GAN-based models (including SetGAN) are relatively fast to perform inference, with WaveGAN being the fastest. FSDM, as a diffusion-based approach, is extremely slow to perform inference - even at only 128 x 128 resolution.

Model	Time
AGE	00:09.79
FSDM	15:42.49
WaveGAN	00:00.42
SetGAN	00:04.64

Table 5.3: Time to perform a single batch of generations, with batch size 20 and 3 generated images per input set.



(a) The full generation process for a given style w and set of reference images R using SetGAN.



(b) The "template image" generated by the same w with no conditioning layers.

Figure 5.6: Diagrams of an example generation process from SetGAN.

5.6 Analysis

In order to visualize how SetGAN constructs an output image from a given set of inputs, consider the example generation shown in Fig. 5.6a. As explained in section 5.3.1, the generation process begins by encoding each of these reference images into a latent representation C_i using the pSp encoder. The model will then generate a series of m Gaussian noise vectors (one per output image) and pass these through the pretrained StyleGAN2 mapping network to obtain base latent codes $W \in \mathcal{W}$. If these latent codes were fed directly to the generator, they would result in samples from the pretrained StyleGAN2 model, without any conditioning. These unconditional generations can be considered to be a form of "template" images, which will then be transformed and modified to become the final output (see Fig. 5.6b).

In order to incorporate the information from the reference images, a series of attention-based conditioning layers will then combine the base latent codes W with the reference encodings c_i^ℓ at each layer of the network to produce a series of *conditional* style vectors ω. This will have the effect of progressively shifting the template image towards the reference images as it progresses through the network.

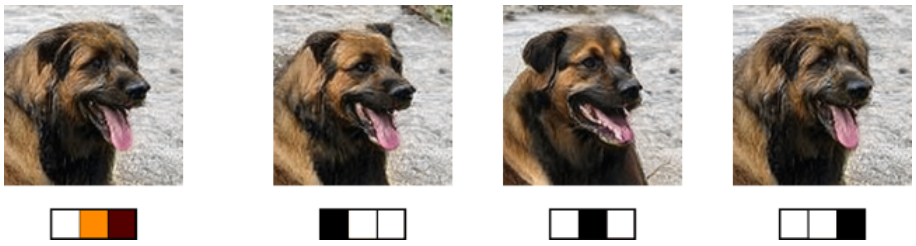


Figure 5.7: Sample generations using the reference images in Figure 5.6 with only the first conditioning layer active. Heatmaps underneath each image indicate the attention weights given to each reference image.

5.6.1 Effect of the conditioning network by reference image

Figure 5.8 shows the relative weight given by the attention blocks in the conditioning network to each of the reference images from the generation process in Figure 5.6a. In order to visualize how the different weight for each image affects the generation outputs, let us consider the effects of the conditioning network on just a single style. With the conditioning network active on only the first style vector, Figure 5.7 shows examples of the output with varying degrees of weight given to each reference image - including examples with the true weights taken from the heatmap in Figure 5.8, as well as 100% weight given to each reference image in turn.

The effect of the varying attention weights at this layer on the final image can be clearly seen from these examples. Features such as the ear shape, ear orientation, fur texture and tongue/mouth position change significantly in accordance with the reference image being most closely attended to at this layer. The effect can be clearly seen on those same features in the final output image. The ears take on a slightly rounded shape, the fur texture becomes shaggy and long, and the open mouth takes on a slight upward lilt that looks almost like a smile - all features strongly similar to the third reference image. This matches the values shown in Figure 5.8, where the weights are indeed highly concentrated around that same image.

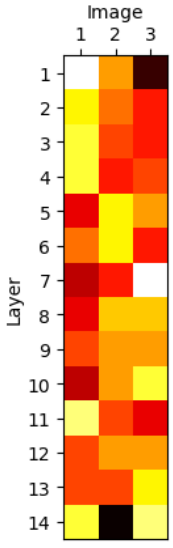


Figure 5.8: Heatmap of attention weights by layer for Fig. 5.6a

5.6.2 Effect of the conditioning network by layer

These previous examples highlighted the effects of the attention layers in attending to and incorporating features from the reference images - but only using a single layer. To see the cumulative effects of these conditioning layers throughout the generation process, I apply the generation

process with a variable number of conditioning layers active. As before, inactive layers use only the base style vector as input. The results of this experiment are shown in Figure 5.9. Initially, no conditioning layers are active, and the generator produces the template image mentioned previously. As more layers are introduced, additional features from the reference image are used to adjust this template image further and further towards the images in the reference set.

Interestingly, the features affected by the introduction of the conditioning layers vary strongly by the position of the layer in the network. Enabling the conditioning layers in the early layers affects coarse features such as fur texture, stripes/patches, head facing and ear position. In contrast, the middle conditioning layers affect the background, fur color, and finer adjustments to face structure/expression. Finally, the last layers in the network affect subtler qualities like color saturation and fine textural details.

This matches closely with the common observation that the layers of the StyleGAN2 network affect the properties of the output image based on their location in the network, with earlier layers affecting coarser features of the image and later layers affecting the finer details. As the SetGAN decoder is directly based on the StyleGAN2 decoder, it is unsurprising to observe the same property here.

Figure 5.9: Generations from Fig. 5.6 with some attention layers inactive. Red boxes indicate active layers.

5.6.3 Effect of the base style vector

One interesting consequence of the many residual or skip connections through SetGAN's architecture is the predominant role played by the base style vector in the generation. As discussed previously, this base style vector represents a sort of "template image", that will then be modified by each of the conditioning layers in turn to attend to the features of the reference images. Despite the significant effects of these layers shown in the previous sections, the initial template image retains a strong effect on the final generation. Figure 5.10 shows a series of generations using the same base style vector as in Fig. 5.6, but different reference images. Notice how all of these images retain similar features in terms of their orientation, head position and overall expression.

To understand the reason for this, consider Equation 5.6.3, which shows how the conditional encodings are incorporated into the styles:

$$\omega^\ell = g^\ell(W, T^\ell(W, C))$$

In this equation, g^ℓ represents a learned transform applied to the concatenation of the base style vector with the conditional style computed

by the appropriate attention block. At the beginning of training, g^{ℓ} is initialized to act as an identity map on the base style, making this essentially a residual connection. As such, the computed conditional encoding will act as an offset *relative* to the base style - anchoring the output generation strongly to the template image.

5.7 Conclusion and Future Work

Figure 5.10: Generations from different reference batches using the same base style.

The task-specific experiments shown in this paper demonstrate that SetGAN can effectively replicate and even surpass the ability of other GAN-based approaches to learn the factors of variation within different classes in a dataset and generalize them to new classes at inference time. In addition, SetGAN shows potential to generalize beyond the structure of the training classes and flexibly perform generation conditioned on reference images sharing features across a wide array of different axes of similarity. It is my hope that in the future, this may be extended to more truly general, zero-shot forms of image generation on larger and more diverse datasets. Other approaches such as Giannone et al. [2022] have shown results on datasets such as CIFAR-100 and Mini-ImageNet, but these datasets are low-resolution and contain a very limited number of classes, which limits the model's ability to generalize to truly diverse and varied unseen classes at inference time. While Giannone et al. [2022] do report some succesful results at few-shot generation with these datasets, they often struggle to adapt to unseen classes at inference time as a result of this, and end up producing samples from unrelated training classes. Instead, my focus is on scaling this approach to truly diverse and large-scale high resolution datasets such as ImageNet. This may provide a path to achieving truly zero shot set-based image-to-image generation, and will be the focus of future work.

Chapter 6

Conclusion

6.1 Contributions

The power of latent representations in machine learning is undeniable. By exploiting and leveraging the distributions of these latent representations, we can gain insights into diverse fields and provide powerful new tools to approach many common tasks. In the field of natural language alone, we can predict relationships between words, improve robustness in natural language, improve the effectiveness of common model architectures, and even translate between languages without parallel data.

In this work, I set out to make my own contribution to the study of distributions within embedding spaces of words. Beginning from existing works such as the distributional representations proposed by Vilnis and McCallum [2014], Tian et al. [2014] and many others, I began with two overarching focus questions:

- How can we leverage the uncertainty inherent to the problem of noise in natural language to create robust representations, in order to improve the performance of natural language models in realistic domains such as social media?
- Can we train neural networks to approximate distance functions between distributions in order to learn bespoke functions that are optimal for particular tasks - for example, unsupervised inference of relationships between words from their induced distributions over contexts?

While these initial questions remained tightly connected to the field of word embeddings, the investigations that followed did not. The insights motivated from those narrow beginnings blossomed out to encompass multiple diverse subfields and applications - from robustness in natural

language, to approximating statistical distance functions, to novel forms of few-shot image generation. In the end, the primary contributions of this work were threefold:

1. Proposing a flexible robust framework for natural language models that could explicitly model the uncertainty inherent to noisy text in order to improve performance on downstream tasks in a model-agnostic fashion. This approach demonstrated superior results to existing methods across many of the tasks in the well-known GLUE benchmark [Wang et al., 2018] using multiple different classifier architectures.

2. Defining a framework for describing functions defined on multiple permutation-invariant sets, and proposing a general model architecture for training neural network models to approximate these functions. The proposed architecture was proved to be a universal approximator of these *partially permutation-invariant* functions, and demonstrated superior performance to any existing methods on a variety of set-based tasks, including approximating notoriously intractable distance measures such as KL Divergence and Mutual Information.

3. Leveraging this multi-set architecture to propose a novel model for few-shot image generation. The proposed model demonstrated superior performance to all existing few-shot image-to-image models, and was not restricted by the limiting assumptions of existing fusion-based or transformation-based methods. As such, this model provides a framework that could be scaled up to true zero-shot image generation in the vein of existing foundation models such as DALL-E or Stable Diffusion - unlike competing approaches.

6.2 Future Work

The works presented in this thesis each offer many promising opportunities for extensions and further work. While embeddings in the vein of Word2Vec have largely fallen out of favor, the large language models that now dominate the field still rely on BPE-based subword embeddings for tokenization. These embeddings do not demonstrate the unique structure of Word2Vec that allowed for rich applications such as MUSE [Lample et al., 2018a] - but many opportunities still exist. Despite the immense power of these models, they must still contend with the same problems as the previous generation of natural language models, and issues such as polysemy and noisy or error-ridden text still remain relevant.

The most critical and promising extensions of this work do not lie within the field of natural language, however. One highly interesting field of exploration motivated by the work presented in Chapter 4 is the approximation of Mutual Information. Mutual Information is a key concept in many areas of machine learning, and is essential to widely-used approaches such as the Information Bottleneck [Tishby and Zaslavsky, 2015]. Mutual information-based approaches are also widely

used in fields such as privacy & fairness [Song et al., 2019a, Zhao and Gordon, 2022, Locatello et al., 2019], contrastive learning [Zhang et al., 2021, Yang et al., 2022a], and many others. Such works are often forced to rely on adversarial approaches to approximate mutual information, which often presents challenges in optimization. The framework in Chapter 4 presents a possible method for creating a *pretrained* estimator of mutual information in a supervised fashion. Such an estimator could be trained once and used in a wide variety of applications. Due to the dimension-equivariance discussed in Section 4.3.5, such an estimator could even be applied to inputs of variable dimension, ensuring that its use could be extended to a wide variety of different problems and model architectures. This is an extremely promising direction for future work.

Finally, as discussed previously, the framework presented in Chapter 5 has the ability to be extended to much larger and more general datasets. While the experiments presented in this chapter were limited to small, task-specific datasets, this is not a requirement of the method. Unlike competing approaches, SetGAN does not make limiting assumptions about the structure of the training and test data. As such, extensions to larger and more general datasets such as ImageNet are a promising next step. Once scaled to such a general setting, SetGAN could then be used for zero-shot generation using reference sets of images from diverse domains and organized by diverse factors of similarity. Even in this limited setting, the model demonstrates some ability to generalize beyond the structure of the training data in this way (see Figure 5.1) - an encouraging sign for the model's potential in larger settings.

References

Yuval Alaluf, Or Patashnik, and Daniel Cohen-Or. Restyle: A residual-based stylegan encoder via iterative refinement, 2021.

Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. 11 2017.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In Jian Su, Kevin Duh, and Xavier Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2289–2294, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1250. URL https://aclanthology.org/D16-1250.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. Learning bilingual word embeddings with (almost) no bilingual data. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1042. URL https://aclanthology.org/P17-1042.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *stat*, 1050:21, 2016.

Ching-Yuan Bai, Hsuan-Tien Lin, Colin Raffel, and Wendy Chi wen Kan. On training sample memorization: Lessons from benchmarking generative modeling with a large-scale competition. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. ACM, aug 2021. doi: 10.1145/3447548.3467198.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. Entailment above the word level in distributional semantics. In Walter Daelemans, editor, *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 23–32, Avignon, France, April 2012. Association for Computational Linguistics. URL https://aclanthology.org/E12-1004.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International conference on machine learning*, pages 531–540. PMLR, 2018.

Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation. *CoRR*, abs/1711.02173, 2017. URL http://arxiv.org/abs/1711.02173.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, Wesam Manassra, Prafulla Dhariwal, Casey Chu, Yunxin Jiao, and Aditya Ramesh. Improving image generation with better captions. 2023.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022, mar 2003. ISSN 1532-4435.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

Malik Boudiaf, Ziko Imtiaz Masud, Jérôme Rony, Jose Dolz, Ismail Ben Ayed, and Pablo Piantanida. Mutual-information based few-shot classification, 2021.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, 2015.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Qiong Cao, Li Shen, Weidi Xie, Omkar M. Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age, 2018.

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 Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179.

Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation, 2018.

Min Jin Chong and David Forsyth. Effectively unbiased FID and inception score and where to find them, 2020.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

Louis Clouâtre and Marc Demers. Figr: Few-shot image generation with reptile, 2019.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

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

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Guanqi Ding, Xinzhe Han, Shuhui Wang, Shuzhe Wu, Xin Jin, Dandan Tu, and Qingming Huang. Attribute group editing for reliable few-shot image generation, 2022.

M. D. Donsker and S. R. S. Varadhan. Asymptotic evaluation of certain markov process expectations for large time. iv. *Communications on Pure and Applied Mathematics*, 36(2):183–212, 1983. doi: https://doi.org/10.1002/cpa.3160360204. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160360204.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

Yerai Doval, Jesús Vilares, and Carlos Gómez-Rodríguez. Towards robust word embeddings for noisy texts. *arXiv preprint arXiv:1911.10876*, 2019.

Alessandro Ferrero, Shireen Elhabian, and Ross Whitaker. Setgan: Improving the stability and diversity of generative models through a permutation invariant architecture, 2022.

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

J. R. Firth. A synopsis of linguistic theory 1930-55. 1952-59:1–32, 1957.

Philip Gage. A new algorithm for data compression. *C Users J.*, 12(2):23–38, February 1994a. ISSN 0898-9788.

Philip Gage. A new algorithm for data compression. *The C Users Journal archive*, 12:23–38, 1994b. URL https://api.semanticscholar.org/CorpusID:59804030.

Shuyang Gao, Greg Ver Steeg, and Aram Galstyan. Efficient Estimation of Mutual Information for Strongly Dependent Variables. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 277–286, San Diego, California, USA, 09–12 May 2015. PMLR. URL https://proceedings.mlr.press/v38/gao15.html.

Maayan Geffet and Ido Dagan. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 107–114, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219854. URL https://aclanthology.org/P05-1014.

Alan E. Gelfand and Adrian F. M. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85(410):398–409, 1990. doi: 10.1080/01621459.1990.10476213. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1990.10476213.

Giorgio Giannone, Didrik Nielsen, and Ole Winther. Few-shot diffusion models, 2022.

Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722, 2014. URL http://arxiv.org/abs/1402.3722.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Zheng Gu, Wenbin Li, Jing Huo, Lei Wang, and Yang Gao. Lofgan: Fusing local representations for few-shot image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8463–8471, October 2021.

Shupeng Gui, Xiangliang Zhang, Pan Zhong, Shuang Qiu, Mingrui Wu, Jieping Ye, Zhengdao Wang, and Ji Liu. Pine: Universal deep embedding for graph nodes via partial permutation invariant set functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44 (2):770–782, 2021.

Ishaan Gulrajani, Colin Raffel, and Luke Metz. Towards gan benchmarks which require generalization, 2020.

Zellig Harris et al. Distributional hypothesis. *Word World*, 10(23):146–162, 1954.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Georg Heigold, Stalin Varanasi, Günter Neumann, and Josef van Genabith. How robust are character-based word embeddings in tagging and MT against wrod scramlbing or randdm nouse? In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 68–80, Boston, MA, March 2018. Association for Machine Translation in the Americas. URL https://www.aclweb.org/anthology/W18-1807.

James Henderson and Diana Popa. A vector space for distributional semantics for entailment. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2052–2062, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1193. URL https://www.aclweb.org/anthology/P16-1193.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020a. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020b.

Yan Hong, Li Niu, Jianfu Zhang, Jing Liang, and Liqing Zhang. Deltagan: Towards diverse few-shot image generation with sample-specific delta. *CoRR*, abs/2009.08753, 2020a. URL https://arxiv.org/abs/2009.08753.

Yan Hong, Li Niu, Jianfu Zhang, and Liqing Zhang. Matchinggan: Matching-based few-shot image generation. *CoRR*, abs/2003.03497, 2020b. URL https://arxiv.org/abs/2003.03497.

Yan Hong, Li Niu, Jianfu Zhang, Weijie Zhao, Chen Fu, and Liqing Zhang. F2GAN: fusing-and-filling GAN for few-shot image generation. *CoRR*, abs/2008.01999, 2020c. URL https://arxiv.org/abs/2008.01999.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. Image-to-image translation with conditional adversarial networks, 2018.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Marco Jiralerspong, Avishek Joey Bose, Ian Gemp, Chongli Qin, Yoram Bachrach, and Gauthier Gidel. Feature likelihood score: Evaluating generalization of generative models using samples, 2023.

Erik Jones, Robin Jia, Aditi Raghunathan, and Percy Liang. Robust encodings: A framework for combating adversarial typos, 2020.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2019.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan, 2020.

Mark Kernighan, Kenneth Church, and William Gale. A spelling correction program based on a noisy channel model. pages 205–210, 01 1990. doi: 10.3115/997939.997975.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models, 2020.

Huda Khayrallah and Philipp Koehn. On the impact of various types of noise on neural machine translation. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 74–83, 2018.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2019.

Alexander Kraskov, Harald Stoegbauer, and Peter Grassberger. Estimating mutual information. *Physical Review E*, 69(6):066138, Jun 2004. ISSN 1539-3755, 1550-2376. doi: 10.1103/PhysRevE.69.066138. arXiv: cond-mat/0305641.

Tuomas Kynkäänniemi, Tero Karras, Miika Aittala, Timo Aila, and Jaakko Lehtinen. The role of imagenet classes in fréchet inception distance, 2023.

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. doi: 10.1126/science.aab3050. URL https://www.science.org/doi/abs/10.1126/science.aab3050.

Guillaume Lample, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043*, 2017.

Guillaume Lample, Alexis Conneau, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. In *International Conference on Learning Representations*, 2018a.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. Phrase-based & neural unsupervised machine translation, 2018b.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR, 2019.

V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707, February 1966.

Vladimir Levenshtein. Binary codes capable of correcting spurious insertions and deletion of ones. *Problems of information Transmission*, 1(1):8–17, 1965.

Yijun Li, Richard Zhang, Jingwan Lu, and Eli Shechtman. Few-shot image generation with elastic weight consolidation. *CoRR*, abs/2012.02780, 2020. URL https://arxiv.org/abs/2012.02780.

Weixin Liang, Zixuan Liu, and Can Liu. Dawson: A domain adaptive few shot generation framework, 2020.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation, 2019a.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. Topical word embeddings. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019b. URL http://arxiv.org/abs/1907.11692.

Francesco Locatello, Gabriele Abbati, Thomas Rainforth, Stefan Bauer, Bernhard Schölkopf, and Olivier Bachem. On the fairness of disentangled representations. *Advances in neural information processing systems*, 32, 2019.

Valentin Malykh, Varvara Logacheva, and Taras Khakhulin. Robust word vectors: Context-informed embeddings for noisy texts. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 54–63, 2018.

Ning Miao, Hao Zhou, Chengqi Zhao, Wenxian Shi, and Lei Li. Kernelized bayesian softmax for text generation. In *Advances in Neural Information Processing Systems*, pages 12487–12497, 2019.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation, 2013b.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729, 2008. doi: 10.1109/ICVGIP.2008.47.

Utkarsh Ojha, Yijun Li, Jingwan Lu, Alexei A. Efros, Yong Jae Lee, Eli Shechtman, and Richard Zhang. Few-shot image generation via cross-domain correspondence. *CoRR*, abs/2104.06820, 2021. URL https://arxiv.org/abs/2104.06820.

OpenAI. Gpt-4 technical report, 2023.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *CVPR*, 2022.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Aleksandra Piktus, Necati Bora Edizel, Piotr Bojanowski, Édouard Grave, Rui Ferreira, and Fabrizio Silvestri. Misspelling oblivious word embeddings. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3226–3234, 2019.

Justin Pinkney. Stable diffusion image variations. https://www.justinpinkney.com/blog/2023/stable-diffusion-image-variations/, 2023.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022. URL https://arxiv.org/abs/2204.06125.

Harsh Rangwani, Lavish Bansal, Kartik Sharma, Tejan Karmali, Varun Jampani, and R. Venkatesh Babu. Noisytwins: Class-consistent and diverse image generation through stylegans, 2023.

Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL http://arxiv.org/abs/1505.04597.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. 2022.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Wei Wei, Tingbo Hou, Yael Pritch, Neal Wadhwa, Michael Rubinstein, and Kfir Aberman. Hyperdreambooth: Hypernetworks for fast personalization of text-to-image models, 2023.

Chitwan Saharia, William Chan, Huiwen Chang, Chris A. Lee, Jonathan Ho, Tim Salimans, David J. Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models, 2022.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30, 2017.

Hiroshi Sasaki, Chris G. Willcocks, and Toby P. Breckon. Unit-ddpm: Unpaired image translation with denoising diffusion probabilistic models, 2021.

Klaus U Schulz and Stoyan Mihov. Fast string correction with levenshtein automata. *International Journal on Document Analysis and Recognition*, 5(1):67–85, 2002.

Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid, August 2020. Version 0.3.0.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://www.aclweb.org/anthology/P16-1162.

Zhuoran Shen, Mingyuan Zhang, Shuai Yi, Junjie Yan, and Haiyu Zhao. Factorized attention: Self-attention with linear complexities. *CoRR*, abs/1812.01243, 2018. URL http://arxiv.org/abs/1812.01243.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

Sidak Pal Singh, Andreas Hug, Aymeric Dieuleveut, and Martin Jaggi. Wasserstein is all you need. *CoRR*, abs/1808.09663, 2018. URL http://arxiv.org/abs/1808.09663.

Sidak Pal Singh, Andreas Hug, Aymeric Dieuleveut, and Martin Jaggi. Context mover's distance & barycenters: Optimal transport of contexts for building representations. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3437–3449. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/singh20a.html.

Samuel L. Smith, David H. P. Turban, Steven Hamblin, and Nils Y. Hammerla. Offline bilingual word vectors, orthogonal transformations and the inverted softmax, 2017.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

Michael Soloveitchik, Tzvi Diskin, Efrat Morin, and Ami Wiesel. Conditional frechet inception distance, 2022.

Jiaming Song, Pratyusha Kalluri, Aditya Grover, Shengjia Zhao, and Stefano Ermon. Learning controllable fair representations. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2164–2173. PMLR, 2019a.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution, 2020.

Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation, 2019b.

Robyn Speer and Catherine Havasi. Representing general relational knowledge in ConceptNet 5. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 3679–3686, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2012/pdf/1072_Paper.pdf.

Chi Sun, Hang Yan, Xipeng Qiu, and Xuanjing Huang. Gaussian word embedding with a wasserstein distance loss. *arXiv preprint arXiv:1808.07016*, 2018.

Yifu Sun and Haoming Jiang. Contextual text denoising with masked language models. *arXiv preprint arXiv:1910.14080*, 2019.

Ilya Sutskever, Oriol Vinyals, and Quoc Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 4, 09 2014a.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014b. URL https://proceedings.neurips.cc/paper_files/paper/2014/file/a14ac55a4f27472c5d894ec1c3c743d2-Paper.pdf.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision, 2015.

Aarne Talman, Anssi Yli-Jyrä, and Jörg Tiedemann. Sentence embeddings in NLI with iterative refinement encoders. *Natural Language Engineering*, 25(4):467–482, 2019.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. A probabilistic model for learning multi-prototype word embeddings. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 151–160, 2014.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle, 2015.

Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik, and Daniel Cohen-Or. Designing an encoder for stylegan image manipulation, 2021.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations*, 2019.

Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Luke Vilnis and Andrew McCallum. Word representations via gaussian embedding. *arXiv preprint arXiv:1412.6623*, 2014.

Edward Wagstaff, Fabian Fuchs, Martin Engelcke, Ingmar Posner, and Michael A. Osborne. On the limitations of representing functions on sets. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6487–6494. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/wagstaff19a.html.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461, 2018. URL http://arxiv.org/abs/1804.07461.

Qing Wang, Sanjeev R. Kulkarni, and Sergio Verdu. Divergence estimation for multidimensional densities via k-nearest-neighbor distances. *IEEE Transactions on Information Theory*, 55(5): 2392–2405, 2009. doi: 10.1109/TIT.2009.2016060.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *CoRR*, abs/2006.04768, 2020. URL https://arxiv.org/abs/2006.04768.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945. ISSN 00994987. URL http://www.jstor.org/stable/3001968.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

Jinyu Yang, Jiali Duan, Son Tran, Yi Xu, Sampath Chanda, Liqun Chen, Belinda Zeng, Trishul Chilimbi, and Junzhou Huang. Vision-language pre-training with triple contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15671–15680, 2022a.

Mengping Yang, Zhe Wang, Ziqiu Chi, and Wenyi Feng. Wavegan: Frequency-aware gan for high-fidelity few-shot image generation, 2022b.

Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. 2023.

Chulhee Yun, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Are transformers universal approximators of sequence-to-sequence functions? In *International Conference on Learning Representations*, 2019.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Han Zhang, Jing Yu Koh, Jason Baldridge, Honglak Lee, and Yinfei Yang. Cross-modal contrastive learning for text-to-image generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 833–842, 2021.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

Han Zhao and Geoffrey J Gordon. Inherent tradeoffs in learning fair representations. *The Journal of Machine Learning Research*, 23(1):2527–2552, 2022.

Jingyuan Zhu, Huimin Ma, Jiansheng Chen, and Jian Yuan. Few-shot image generation with diffusion models, 2023.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks, 2020.

APPENDICES

Appendix A

Appendix: Robust Embeddings

A.1 Experiments and Baselines

Independent noisy versions of the data were constructed for each trial, which were each then used to train each of the two classifiers on the appropriate task, with each robust method used in turn to mitigate the effects of the noise. Basic preprocessing was applied to the noisy data before passing it through the robust models (e.g., lowercasing, stripping non-alphanumeric characters except for basic punctuation, etc.). Experiments were performed using the glue finetuning and evaluation script from HuggingFace, with all training parameters left as default. For the HBMP case, FastText vectors were used as a ground embedding layer for the methods that did not prescribe their own vectors.

Baselines were evaluated using existing source code where possible (Bridge2Vec, RobEn), and otherwise were reimplemented following as closely to the authors' original descriptions as possible (MOE, Sun & Jiang). A value of $\alpha = 0.05$ was used for the MOE training.

A.2 Ablation Results

For full results of the ablation study on all datasets, see Table A.1.

A.3 Hyperparameter Analysis

In order to evaluate the optimal hyperparameter values, we ran a number of experiments with the RED-Ensemble model on two of the GLUE datasets: MRPC and SST-2. Five trials were

	RoBERTa					HBMP				
		20%		50%			20%		50%	
Model	Clean	Synth.	Natural	Synth.	Natural	Clean	Synth.	Natural	Synth.	Natural
					MRPC					
Token Stream	0.875↓	0.848↓	0.855↓	0.788↓	0.799↓	0.799	0.731↓	0.740↓	0.547↓	0.602↓
Context Stream	0.868↓	0.793↓	0.809↓	0.402↓	0.526↓	0.798	0.659↓	0.700↓	0.297↓	0.428↓
RED (Base)	0.872↓	0.866↓	0.867↓	0.842↓	0.847↓	0.798	0.775	0.783	0.713↓	0.742
RED (Ensemble)	**0.879**	**0.872**	**0.873**	**0.853**	**0.854**	**0.800**	**0.780**	**0.786**	**0.723**	**0.749**
					QNLI					
Token Stream	**0.895**↑	0.865↓	0.874↓	0.813↓	0.835↓	**0.801**↑	0.779↓	0.780↓	0.739↓	0.749↓
Context Stream	0.878↓	0.810↓	0.837↓	0.681↓	0.751↓	0.791	0.745↓	0.760↓	0.654↓	0.698↓
RED (Base)	0.891	**0.879**	0.883	0.849↓	0.863↓	0.800	**0.793**	**0.795**	0.773↓	0.777↓
RED (Ensemble)	0.889	0.877	**0.883**	**0.856**	**0.869**	0.767	0.792	0.794	**0.777**	**0.781**
					QQP					
Token Stream	**0.885**↑	0.800↓	0.824↓	0.697↓	0.754↓	**0.857**↑	0.734↓	0.765↓	0.582↓	0.651↓
Context Stream	0.862↓	0.740↓	0.783↓	0.524↓	0.660↓	0.836↓	0.697↓	0.743↓	0.470↓	0.596↓
RED (Base)	0.873↓	0.842↓	0.855↓	0.789↓	0.817↓	0.842	**0.809**	0.818↓	**0.741**	0.770↓
RED (Ensemble)	0.877	**0.847**	**0.858**	**0.794**	**0.823**	0.843	**0.809**	**0.822**	0.739	**0.772**
					SST-2					
Token Stream	**0.917**↑	0.896↓	0.891↓	0.865↓	0.851↓	**0.842**	0.820	0.816↓	0.786↓	0.779↓
Context Stream	0.905↓	0.860↓	0.861↓	0.767↓	0.773↓	0.827	0.786↓	0.788↓	0.709↓	0.710↓
RED (Base)	0.916	0.901	0.901↓	0.884↓	0.878↓	0.825	0.820	0.819↓	0.802↓	0.799
RED (Ensemble)	0.913	**0.903**	**0.908**	**0.895**	**0.886**	0.829	**0.823**	**0.825**	**0.814**	**0.806**
					MNLI					
Token Stream	**0.847**↑	0.774↓	0.790↓	0.679↓	0.705↓	0.718	0.643↓	0.657↓	0.555↓	0.577↓
Context Stream	0.829↓	0.703↓	0.745↓	0.536↓	0.606↓	0.712	0.616↓	0.642↓	0.494↓	0.542↓
RED (Base)	0.843↑	**0.814**	**0.823**	0.748↓	0.777↓	0.713↑	**0.689**↑	**0.692**	**0.640**	**0.653**↑
RED (Ensemble)	0.840	0.813	0.822	**0.768**	**0.781**	0.707	0.685	0.690	**0.640**	0.650
					Average					
Token Stream	**0.884**↑	0.837↓	0.847↓	0.769↓	0.789↓	**0.803**↑	0.741↓	0.752↓	0.642↓	0.671↓
Context Stream	0.868↓	0.781↓	0.807↓	0.582↓	0.663↓	0.793	0.701↓	0.727↓	0.525↓	0.595↓
RED (Base)	0.879	0.860↓	0.866↓	0.822↓	0.836↓	0.796	0.777	0.781	0.734↓	0.748↓
RED (Ensemble)	0.880	**0.862**	**0.869**	**0.833**	**0.843**	0.789	**0.778**	**0.783**	**0.739**	**0.752**

Table A.1: Results of ablations on GLUE tasks with RoBERTa and HBMP classifiers. ↑ and ↓ signify results that are statistically better or worse respectively than RED-Ensemble with $p < 0.05$ according to the Wilcoxon signed rank test[Wilcoxon, 1945].

performed for each experiment, and the results were averaged. In general, the accuracy of the results was not very sensitive to the hyperparameter values, as long as those values were not too extreme. As such, each hyperparameter was evaluated independently of the other two, using the approximately correct values for the other hyperparameters as an ansatz.

A.3.1 τ

Figures A.1 and A.2 show plots of accuracy on MRPC and SST-2 respectively against the softmax temperature value τ, on a range of values from $\tau = 0.01$ to $\tau = 2.0$. All experiments were performed using the RoBERTa classifier and utilized RED with ensembling. Experiments were performed with clean training data, and values of $K = 20$ and $M = 10$ were used as the other hyperparameters. The exact optimal τ value varied by dataset and noise type, but in general all values in the approximate range $0.05 \leq \tau \leq 0.3$ gave reasonable results with only small levels of variation. We chose $\tau = 0.15$ as a good compromise value that was optimal or close to optimal across almost all datasets and noise levels.

A.3.2 K

Figures A.3 and A.4 show plots of accuracy on MRPC and SST-2 respectively against the K value in the top-k operation from the prior, on a range of values from $K = 5$ to $K = 30$. All experiments were performed using the RoBERTa classifier and utilized RED with ensembling. Experiments were performed with clean training data, and values of $\tau = 0.15$ and $M = 10$ were used as the other hyperparameters. Performance tended to increase slightly as K increased, but with diminishing returns after a point. This matches expectations, as the K value should only need to be large enough to include most or all of the reasonable corrections, which in most cases tend to be relatively close to the noisy word. In order to balance improved performance with the constraints of increased memory usage, we used $K = 20$ for all experiments.

A.3.3 M

Figures A.5 and A.6 show plots of accuracy on MRPC and SST-2 respectively against the number of samples used for the ensemble, on a range of values from $M = 5$ to $M = 30$. All experiments were performed using the RoBERTa classifier and utilized RED with ensembling. Experiments were performed with clean training data, and values of $\tau = 0.15$ and $K = 20$ were used as the other hyperparameters. Similar to K, increasing M resulted in a slight increase in performance, but there were again diminishing returns - and after a certain point the performance stopped improving or even worsened. We use $M = 10$ for all experiments to once again balance performance with memory constraints.

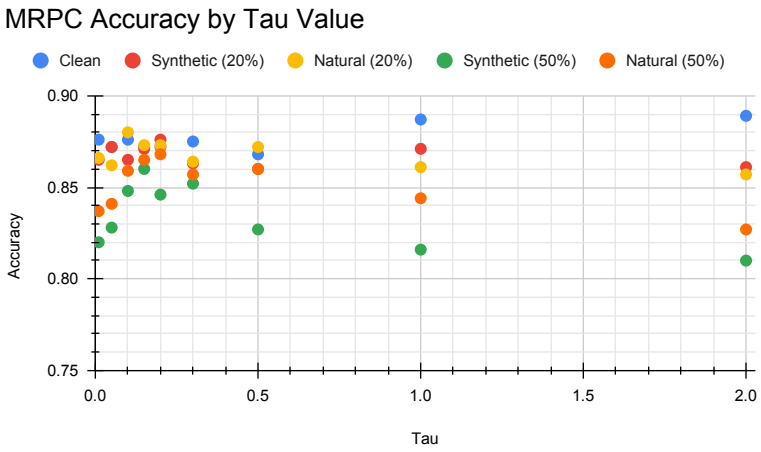


Figure A.1: Plot of MRPC accuracy using the RoBERTa classifier and RED with ensembling by τ value, ranging from $\tau = 0.01$ to $\tau = 2$.

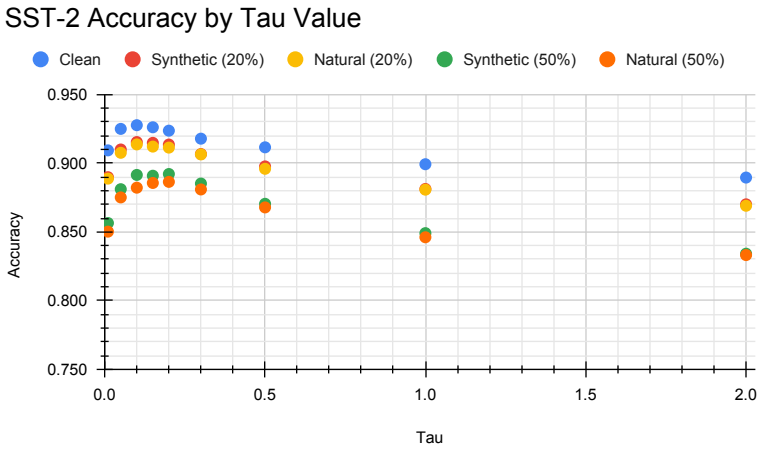


Figure A.2: Plot of SST-2 accuracy using the RoBERTa classifier and RED with ensembling by τ value, ranging from $\tau = 0.01$ to $\tau = 2$.

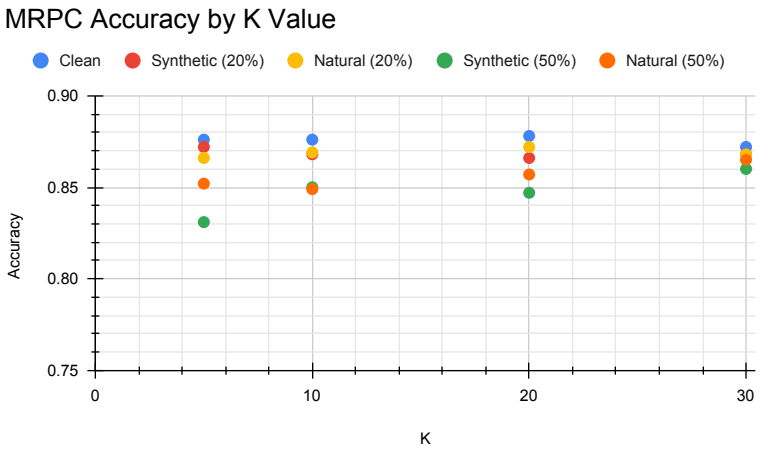


Figure A.3: Plot of MRPC accuracy using the RoBERTa classifier and RED with ensembling by K value, ranging from $K = 5$ to $K = 30$.

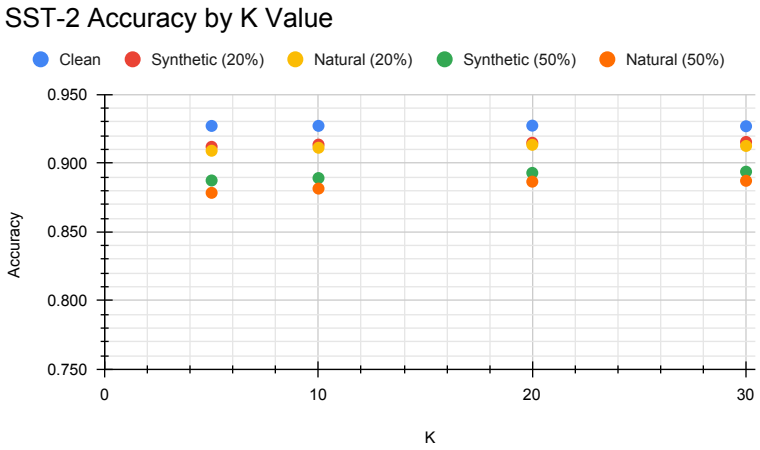


Figure A.4: Plot of SST-2 accuracy using the RoBERTa classifier and RED with ensembling by K value, ranging from $K = 5$ to $K = 30$.

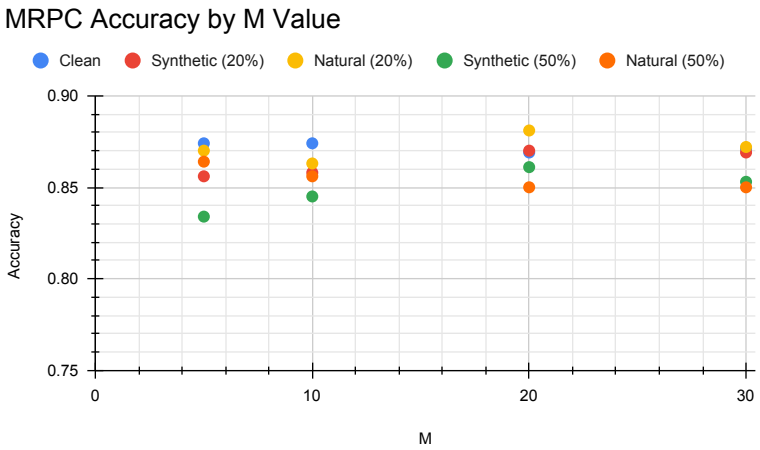


Figure A.5: Plot of MRPC accuracy using the RoBERTa classifier and RED with ensembling by M value, ranging from $M = 5$ to $M = 30$.

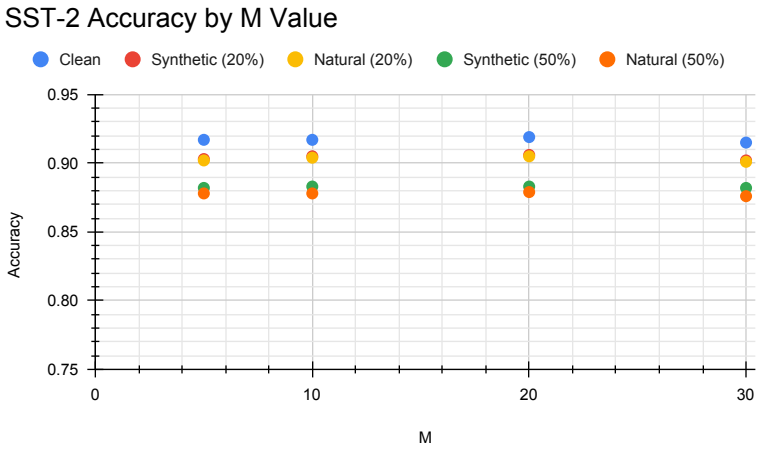


Figure A.6: Plot of SST-2 accuracy using the RoBERTa classifier and RED with ensembling by M value, ranging from $M = 5$ to $M = 30$.

Appendix B

Appendix: Multi-Set Transformers

B.1 Experiment Details

The base architecture used in all experiments was the architecture shown in Figure 4.1, with the MSAB blocks replaced as appropriate for each baseline. The only exception was the PINE model, which followed the architecture described in their paper.

In all cases (except where noted otherwise), we used architectures with 4 blocks, 4 attention heads (for the transformer models), and 1-layer feedforward decoders. We used Pooling by Multiheaded Attention (PMA) (see Lee et al. [2019]) as the pooling layer for the overall network, and max pooling within each relation network block. We used layer norm around each encoder block, as well as within the transformer blocks as per usual. Each block used the same latent and hidden size, and linear projection layers were added at the beginning of the network to project the inputs to the correct dimension if needed.

For the KL and MI experiments, we trained for 100,000 batches of size 64 with a learning rate of 1e-4. The models used a latent size of 16 per input dimension and feedforward size of 32 per input dimension. The dimension-equivariant model was trained across data of multiple dimensions (1-3 for $d = 2$, 3-5 for $d = 4$, 7-9 for $d = 8$ and 14-18 for $d = 16$). Sets were generated as described in Sections 4.6.1 and 4.6.1.

For the Counting experiments, we trained with a batch size of 64 using a latent size of 128 and hidden size of 256. We used a single projection layer as a decoder, with no hidden layers. For MNIST, we used a convolutional encoder with 3x3 convolutional layers of 32 and 64 filters, each followed by a max pool, with a linear projection to the latent size of 128 at the end. This encoder was pretrained for 1000 batches, then the network and encoder were trained end to end for 10,000 batches with a learning rate of 3e-4. Sets were randomly sampled with set size randomly selected in $[10, 30]$. For Omniglot, the convolutional encoder used one 7x7 conv with stride 2 and 32

filters, followed by three blocks of two 3x3 convs each with 32, 64 and 128 filters respectively. Each block was followed by a max pool, and a final linear projection to size 128 was again added at the end. This was pretrained for 300 batches, then the network itself was trained end to end for 10,000 batches with a learning rate of 1e-4. Loss was calculated by mean-squared error. Sets were randomly sampled with set size randomly selected in $[6, 10]$.

The CoCo experiments again used convolutional encoders to obtain fixed size representations of each image, and used transformer encoders to do the same for the captions. This time, the pretrained ResNet-101 model was used as the image encoder, with BERT used as the text encoder. The model was trained for 2500 batches of batch size 48 with learning rate 1e-5, using a latent size of 512 and hidden size of 1024. The set size was increased gradually according to a schedule during training, beginning at sets with size randomly selected in $[1, 5]$ for 1250 batches, $[3, 10]$ for 625 batches and $[8, 15]$ for 625 batches. Standard image preprocessing techniques were applied, with each image rescaled to 256x256, center cropped to size 224, then normalized according to the method expected by PyTorch's pretrained ResNet models (see https://pytorch.org/vision/stable/models.html). The FastText experiments used common crawl FastText vectors for English and French [1], with ground truth translations taken from MUSE [2]. The model was trained for 3125 batches of batch size 128 with learning rate 1e-5, using a latent size of 512 and hidden size of 1024. The set size was increased gradually according to a schedule during training, beginning at sets with size randomly selected in $[1, 5]$ for 1250 batches, $[3, 10]$ for 625 batches, $[8, 15]$ for 625 batches and $[10, 30]$ for 625 batches.

Meta-Dataset experiments used the same convolutional encoder architecture as the Omniglot experiments, though without pretraining. Images were preprocessed in the standard fashion performed by the Pytorch Meta-Dataset library. The model was trained for 7500 batches of batch size 64 with set size randomly selected in $[10, 30]$ and learning rate 1e-5, using a latent size of 512 and hidden size of 1024. The synthetic experiments for distinguishability were trained for 7500 batches of batch size 256 with set size randomly selected in $[10, 30]$ and learning rate 1e-5, using a latent size of 8 and hidden size of 16.

B.2 Attention Derivation

A typical self-attention block with the set $X \in \mathbb{R}^{n \times d}$ as the input queries and keys/values obeys the following equation:

$$\begin{aligned} Z &= \text{MHA}(X, X) \\ &= \left[\sigma\left((XW_Q)(XW_K)^T\right)(XW_V)\right] W_O \\ &= \sigma\left(X(W_Q W_K^T)X^T\right) XW_V W_O \end{aligned}$$

[1] https://fasttext.cc/docs/en/crawl-vectors.html

[2] https://github.com/facebookresearch/MUSE#ground-truth-bilingual-dictionaries

If we now consider the joint set $X \bigsqcup Y \in \mathbb{R}^{n+m \times d}$ and perform self-attention on that, we find the following:

$$\begin{aligned}\begin{pmatrix} Z_X \\ Z_Y \end{pmatrix} &= \text{ATTN}\left(\begin{pmatrix} X \\ Y \end{pmatrix}, \begin{pmatrix} X \\ Y \end{pmatrix}\right) \\ &= \left[\sigma\left(\begin{pmatrix} XW_Q \\ YW_Q \end{pmatrix}\begin{pmatrix} XW_K \\ YW_K \end{pmatrix}^T\right)\begin{pmatrix} X \\ Y \end{pmatrix} W_V\right] W_O \\ &= \begin{pmatrix} A_{xx} & A_{xy} \\ A_{yx} & A_{yy} \end{pmatrix}\begin{pmatrix} X \\ Y \end{pmatrix} W_V W_O\end{aligned}$$

wherein

$$\begin{pmatrix} A_{xx} & A_{xy} \\ A_{yx} & A_{yy} \end{pmatrix} = \sigma\left(\begin{pmatrix} X(W_Q W_K^T)X^T & X(W_Q W_K^T)Y^T \\ Y(W_Q W_K^T)X^T & Y(W_Q W_K^T)Y^T \end{pmatrix}\right)$$

Then, we find that

$$\begin{aligned}Z_X &= A_{xx}XW_VW_O + A_{xy}YW_VW_O \\ Z_Y &= A_{yx}XW_VW_O + A_{yy}YW_VW_O\end{aligned}$$

This function remains entirely equivariant with respect to the order of the elements in the joint set $X \bigsqcup Y$ - there is no distinction between elements in X and elements in Y.

Suppose, however, that we were to let the softmax for $A_{\alpha\beta}$ be computed only over the elements of the set α, and let the parameter matrices be different for each of the 4 terms. Then, we find

$$\begin{aligned}Z_X &= \sigma(X(W_{Q,xx}W_{K,xx}^T)X^T)XW_{V,xx}W_{O,xx} \\ &\quad + \sigma(X(W_{Q,xy}W_{K,xy}^T)Y^T)YW_{V,xy}W_{O,xy} \\ Z_Y &= \sigma(Y(W_{Q,yx}W_{K,yx}^T)X^T)XW_{V,yx}W_{O,yx} \\ &\quad + \sigma(Y(W_{Q,yx}W_{K,yy}^T)Y^T)YW_{V,yy}W_{O,yy}\end{aligned}$$

These are just four separate attention blocks! Now we can simply write

$$\begin{aligned}Z_X &= \text{MHA}_{xx}(X,X) + \text{MHA}_{xy}(X,Y) \\ Z_Y &= \text{MHA}_{yx}(Y,X) + \text{MHA}_{yy}(Y,Y)\end{aligned}$$

Since the attention function is equivariant with respect to the first input and invariant with respect to the second, this meets the conditions for partial equivariance. To make this slightly more general, we can now write

$$\begin{aligned}Z_X &= g_x\left(\text{MHA}_{xx}(X,X), \text{MHA}_{xy}(X,Y)\right) \\ Z_Y &= g_y\left(\text{MHA}_{yx}(Y,X), \text{MHA}_{yy}(Y,Y)\right)\end{aligned}$$

where the function g acts on each vector in the output set independently.

Appendix C

Appendix: SetGAN

C.1 Architecture and training details

All experiments used pretrained StyleGAN2 [Karras et al., 2020] and pSp [Richardson et al., 2021] models provided by Ding et al. [2022] [1]. Layers of these models corresponding to resolutions of 256x256 and lower (i.e., the first 14 layers) were taken to use for the initialization. A standard StyleGAN2 generator is used as the decoder, with 14 layers and a latent dimension of 512. The mapping network uses 8 feedforward layers, with a learning rate multiplier of 0.01. A standard pixel2style2pixel (pSp) architecture was used for the encoder, truncated to 14 style vectors as well. Conditioning networks in the generator used stacks of 2 transformer decoder blocks (see Fig. 5.3), with 16 attention heads and latent size 512. The discriminator used the standard StyleGAN2 discriminator architecture as its encoder, with the last layer removed. The multi-set transformer model in the discriminator used a stack of 4 multi-set attention blocks, with mean pooling, skip connections surrounding the multi-set transformer network, and a linear output projection. Training was performed using the StyleGAN2 training schema, with a nonsaturating loss, R1 gradient penalty ($\lambda = 10$), path length regularization ($\lambda = 2$) and style mixing ($p = 0.9$). The model used lazy regularization, with the R1 penalty applied every 16 steps, and path length regularization applied every 4 steps. The generator weights used an exponential moving average, with $\gamma = 0.999$. Training, validation and test splits for each dataset followed the standard splits in prior work, except as discussed previously (see Ding et al. [2022], Hong et al. [2020a], Gu et al. [2021], Hong et al. [2020c]).

Experiments were performed using NVIDIA A40 GPUs. Each model was trained for approximately 1 week using 2 GPUs in parallel via data parallelism.

[1] https://github.com/unibester/AGE