Data Augmentation for Text Classification Tasks

by

Daniel Tamming

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2020

Abstract

Thanks to increases in computing power and the growing availability of large datasets, neural networks have achieved state of the art results in many natural language processing (NLP) and computer vision (CV) tasks. These models require a large number of training examples that are balanced between classes, but in many application areas they rely on training sets that are either small or imbalanced, or both. To address this, data augmentation has become standard practice in CV. This research is motivated by the observation that, relative to CV, data augmentation is underused and understudied in NLP. Three methods of data augmentation are implemented and tested: synonym replacement, backtranslation, and contextual augmentation. Tests are conducted with two models: a Recurrent Neural Network (RNN) and Bidirectional Encoder Representations from Transformers (BERT). To develop learning curves and study the ability of augmentation methods to rebalance datasets, each of three binary classification datasets are made artificially small and made artificially imbalanced. The results show that these augmentation methods can offer accuracy improvements of over 1% to models with a baseline accuracy as high as 92%. On the two largest datasets, the accuracy of BERT is usually improved by either synonym replacement or backtranslation, while the accuracy of the RNN is usually improved by all three augmentation techniques. The augmentation techniques tend to yield the largest accuracy boost when the datasets are smallest or most imbalanced; the performance benefits appear to converge to 0% as the dataset becomes larger. The optimal augmentation distance, the extent to which augmented training examples tend to deviate from their original form, decreases as datasets become more balanced. The results show that data augmentation is a powerful method of improving performance when training on datasets with fewer than 10,000 training examples. The accuracy increases that they offer are reduced by recent advancements in transfer learning schemes, but they are certainly not eliminated.

Acknowledgements

First and foremost, I would like to thank my supervisor, Dr. Peter van Beek, for his guidance, encouragement, and good conversation. I wish to thank Professors Krzysztof Czarnecki and Jesse Hoey for providing feedback on this thesis. Thanks also to my desk-mates Charupriya Sharma and David Radke for their thoughtful advice. Finally, I am grateful to my undergraduate thesis supervisor, Dr. Abdol-Reza Mansouri, for solidifying my interest in research.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In this chapter, I informally introduce and motivate the study of data augmentation in natural language processing (NLP). I also summarize the contributions and the organization of the thesis.

1.1 Natural Language Processing and the Need for Data Augmentation

NLP is a subfield of computer science that is focused on analyzing, modifying, or generating natural language. It contains several subfields, including information extraction and speech processing and generation. This thesis focuses on text classification, a subfield of information extraction. Text classification is an active area of research with a wide range of uses in industry. Tweet sentiment analysis can be used to predict election outcomes [23]. Automated personal assistants such as Apple's Siri will first classify a user's request by the application to which it relates [5]. Comment section moderators can use a constructiveness classification model to filter out comments that do not contribute to the conversation in a meaningful way. I will focus primarily on three tasks: the classification of sentiment, subjectivity (as defined in Section 2.3.2), and constructiveness.

For much of their history, natural language classification models used a features-based approach to modeling. Using feature engineering to convert text to numerical features, practitioners could apply the computationally inexpensive models that remain popular in tabular data analysis, such as logistic regression and support vector machines. However, with a combination of increased computing power and numerous breakthroughs in deep learning, neural-based models have become the state of the art in most NLP subfields. Neural models not only learn the optimal feature weights, but also learn the optimal *features* themselves. The ability of neural models to learn features is central to their success in NLP and computer vision (CV).

Bag of Words (BoW), a common NLP feature engineering technique, treats text as simply a collection of words, each with a count of the number of times in which they appear. Clearly this approach ignores important information, namely the *context* in which each word appears. Each in their own ways (and with varying degrees of success), neural models outperform BoW-based models by incorporating context into their computations.

Due to the bias-variance tradeoff, the advantages of neural models come with costs. Neural models are able to learn features since they have a high degree of freedom, but this implies high model variance and therefore susceptibility to overfitting. This tendency to overfit is most successfully combated by increasing the size of the training set. However, gathering labeled data tends to be costly and so other methods of limiting overfitting must be relied upon.

Methods of combatting overfitting, specifically in the context of neural NLP and CV models, are usually focused either on the training data, or on the model and its training scheme. Common model and training-focused approaches to combatting overfitting include tuning the learning rate, adjusting the number of epochs, model regularization, and replacing the model with a more biased, less variant model. Aside from gathering more labeled data, data augmentation is the only data-focused method.

Transfer learning is an increasingly popular method of combatting overfitting that does not fit neatly into either of the categories outlined in the previous paragraph. The idea behind transfer learning is that a model can perform better on a target task if it is first trained on a large dataset for a related problem. Bidirectional encoder representations from transformers (BERT) is a popular model that is foundational in modern NLP, largely because it lends itself to transfer learning.

Another frequent issue that must be dealt with in machine learning is that of imbalanced classification datasets. Training a machine learning model on an imbalanced dataset tends to bias the model towards the majority class label. Imbalance can be addressed by removing majority class examples or duplicating minority class examples, respectively known as undersampling and oversampling. These two methods both artificially rebalance the training set, but also introduce issues of their own. Undersampling discounts labeled data and therefore ignores information that could be used to improve the model. Oversampling places an added emphasis on each training example that belongs to the minority class, so it can lead to overfitting on the minority class training examples.

This thesis focuses on the effectiveness of data augmentation techniques for improving text classification models, specifically those trained on small and medium sized training sets. It examines the relationship between data augmentation effectiveness and training set size. It also compares dataset rebalancing through data augmentation to rebalancing with undersampling and oversampling.

This study is motivated by three observations. First, the smaller the training set, the more susceptible a model is to overfitting, and therefore the more it can gain from data augmentation. Second, compared to data augmentation in CV, data augmentation in NLP is understudied and underused. Third, dataset rebalancing through data augmentation is commonplace in machine learning with tabular data but is rarely used in NLP.

1.2 Contributions of the Thesis

This thesis has three central contributions. First, I introduce the concept of augmentation distance in the context of three different data augmentation approaches, and I provide an intuitive understanding of when certain distances are advantageous. Second, I compare sampling-based dataset rebalancing to augmentation-based rebalancing. Third, I test the effects of data augmentation on the fine-tuning of BERT.

1.3 Organization of the Thesis

- Chapter 2 lays the groundwork for the rest of the thesis by describing the relevant NLP models and datasets, and common methods of dealing with imbalanced datasets.
- Chapter 3 provides an overview of data augmentation techniques for both CV and NLP tasks.
- Chapter 4 explains how I will improve upon the methods introduced in Chapter 3, and will study their effects in greater detail. It explains the various settings in which each method will be tested.
- Chapter 5 shows where data augmentation was successful and where it failed to offer any performance improvement.
- Chapter 6 concludes the thesis, offering a summary of the findings and possible directions for future work.

Chapter 2

Background

In this chapter, I briefly review the necessary background in NLP and machine learning more broadly.

2.1 Pre-Machine Learning Approaches to NLP

The following is a brief history of NLP. This history begins at the field's inception in the 1940s and ends before the emergence of machine learning methods. This summary draws heavily from the historical review of NLP presented by Jones in [14].

2.1.1 Rules-Based

NLP began in the 1940s and focused entirely on Machine Translation (MT). The predominant approach was to process word-by-word, mapping words from one language to another by way of predefined dictionaries. Researchers developed procedures that resolved ambiguous word meanings by incorporating local context into this dictionary lookup.

2.1.2 Logic-Based

Logic-based approaches to NLP attempted to represent language rules as mathematical logics. Applications of this approach focused on the construction, manipulation, and communication of world knowledge. For example, the BASEBALL question answering system, published in 1961, uses stored baseball-related data and a question interpretation procedure to answer questions such as “Where did each team play on July 7?” [10].

2.1.3 Statistical Language Models

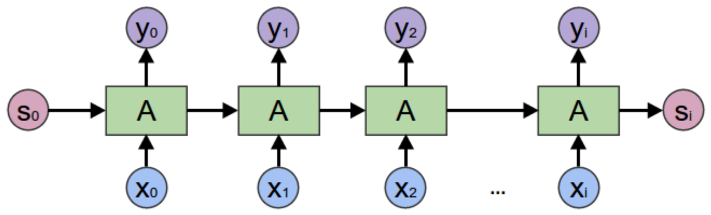
Statistical methods became popular in the 1990s, largely due to increases in processing power and the quantity of text datasets. Rather than hard-coding syntactic rules in a rules-based approach, these methods use massive datasets to infer grammatical rules and probabilities. This wave also laid the foundation for the machine learning models used today. For example, text extraction based on word frequencies is a precursor to feature engineering via bag-of-words (BoW) and term frequency inverse document frequency (TF-IDF).

2.2 Machine Learning NLP Models

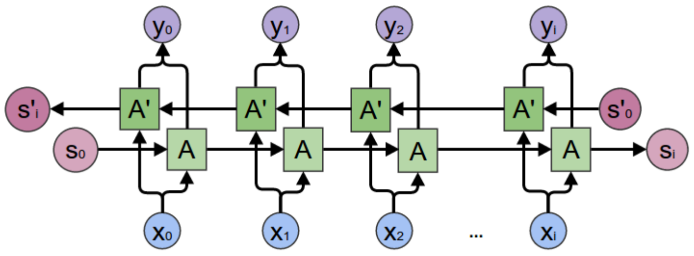
The following is a series of brief explanations of various machine learning models used in NLP. This is not an exhaustive list but it provides a rough overview of the history of machine learning-based NLP models, and an explanation of how each model attempted to improve upon those before it.

2.2.1 Hand-Crafted Features with Traditional Machine Learning Models

Before advances in computing power made neural model training practical, the primary method of NLP modeling was focused on feature-crafting. By transforming text to numerical data, common tabular data models such as regression and support vector machines could be used. During this time, advances in NLP were in the area of feature engineering. The two most common feature engineering methods were BoW and TF-IDF. In BoW, each training example is mapped to a vector of words counts. TF-IDF is BoW but with each word count divided by a function of the word’s frequency in the entire dataset. These methods remain popular in many applications since they are usually fast and tend to require less training data than neural models.



(a) A unidirectional RNN.



(b) A bidirectional RNN.

Figure 2.1: The weights of the subnetworks A and A' are constant across timesteps. Any subset of the outputs, indicated by y and s, may be used for inference. Common choices are the output at each timestep, y_t, $\forall t$, or only the final output(s), s_i, s'_i. Figures published in [25].

2.2.2 Recurrent Neural Network (RNN)

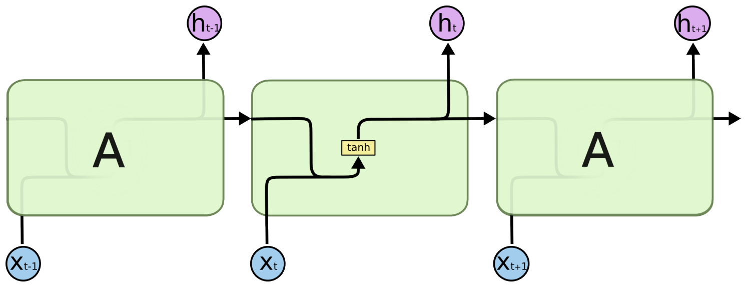
While standard Artificial Neural Networks (ANN) rely on a fixed input size, RNNs allow for variable length input. They do so by accepting not only the input, but also the output of the previous layer (see Figure 2.1). This allows RNNs to learn dependencies in the temporal dimension.

The two most common ways to structure RNNs are unidirectional and bidirectional, where the sequence is processed going forward in time, or once in each direction. Unidirectional RNNs are common in tasks where an emphasis may be placed on the later inputs, such as Language Modeling, an NLP task described in Section 2.4.3. Bidirectional RNNs are better suited for sentiment classification, or other tasks where each timestep can be assumed to have a roughly equal impact on the output. As can be seen in Figure 2.1, bidirectional RNNs are the concatenation of two unidirectional RNNs.

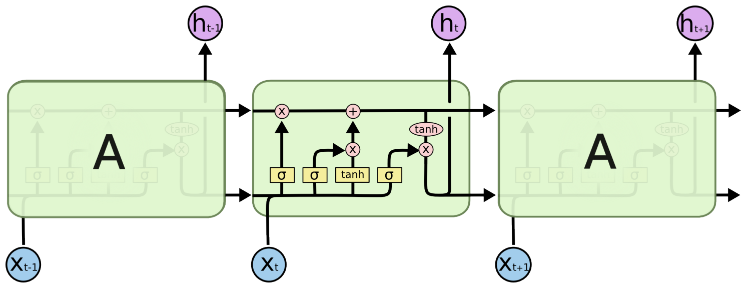
Further development of NLP models sought to address the two major drawbacks of RNNs: the vanishing gradient problem and the fact that they process input sequentially. The vanishing gradient problem is when the loss function decays exponentially with respect to time, so long-term dependencies are not captured by the model. The sequential processing of inputs impedes their training and inference speed since neither can be parallelized. Long short-term memory (LSTM) models improve upon RNNs by mitigating the former of the two problems, and bidirectional encoder representations from transformers (BERT) addresses both issues with its non-sequential structure.

2.2.3 Long Short-Term Memory (LSTM)

LSTMs, a type of RNN so common that they are often simply referred to as RNNs, mitigate the vanishing gradient problem by introducing a memory cell, as shown in Figure 2.2, within each unit. The memory cells, formed by an input, output, and forget gate, govern the information that is remembered and forgotten by the network at each timestep. These gates learn to detect and remember the long-term dependencies that are relevant to the problem at hand. Like general RNNs, LSTMs are usually either unidirectional or bidirectional.



(a) A basic RNN cell.



(b) An LSTM cell.

Figure 2.2: A side by side comparison of RNN and LSTM cells. Pointwise operations and neural network layers are represented by pink circles and yellow rectangles, respectively. Figures published in [25].

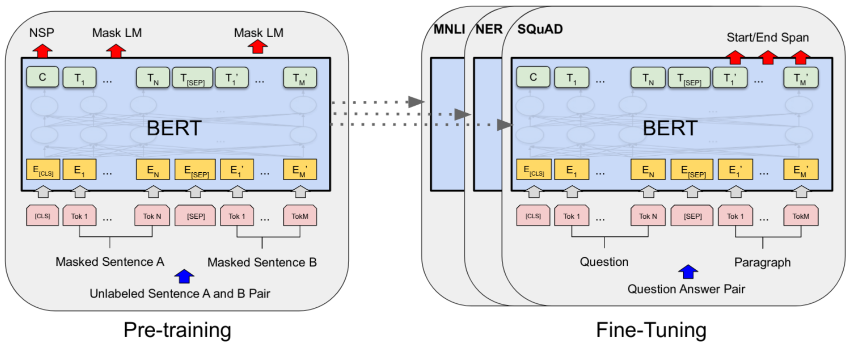


Figure 2.3: BERT pretraining architecture (left) and downstream task architecture (right) introduced in [8]. The pretraining architecture takes two sentences as input in order to adequately prepare the model for two-sentence-input tasks such as question answering; it can be used for single sentence tasks such as topic classification by entering null tokens in the place of sentence B. Figure published in [8].

2.2.4 Bidirectional Encoder Representations from Transformers (BERT)

Convolutional neural networks (CNNs), a type of model that is ubiquitous in CV, are non-sequential and therefore can easily be parallelized and maintain a logarithmic distance between inputs. Note that logarithmic distance between inputs is preferable to linear or exponential distance since text often contains long-term dependencies. Transformer models allow these two desirable features of CNNs to be used in NLP. They consist of encoders and decoders, but the decoder unit is not relevant to this thesis and will not be discussed in detail.

Transformer encoders consist of CNNs and attention mechanisms. Recognizing that word meanings can vary greatly depending on their context, attention mechanisms place each input word in the context of the surrounding words. Rather than simply feeding each word's embedding into the CNNs, the input is a weighted sum of each word's embedding and that of its surrounding words. Transformers were shown to achieve impressive results in [35].

Building on the success of Transformers, BERT achieves state of the art results on a number of NLP tasks by combining multiple Transformer blocks [8]. It has two variants, Base and Large, with 12 and 24 Transformer blocks, respectively.

The advent of BERT has been referred to as the "ImageNet Moment" for NLP. Like ImageNet, BERT lends itself to transfer learning (see Section 2.5.2) wherein knowledge of one problem is leveraged to help learn another. BERT is pretrained on the Masked Language Modeling problem, where the model learns to recover the original version of a corrupted[1] text. This pretraining task is self-supervised, and so BERT is trained on the massive unlabeled Wikipedia and BookCorpus [50] corpora, totaling over 3 billion words. It is now common practice to use transfer learning with a pretrained BERT model for a wide range of NLP tasks.

2.3 NLP Tasks and Datasets

The following tasks and corresponding datasets are common in NLP. This is by no means an exhaustive list, but it provides the necessary background for the tasks that are studied

[1]The text corruption process begins by randomly selecting 15% of the words. Of these 15%, 80% are replaced by a MASK token, 10% by a random word, and 10% by the original word. The loss function depends only on the probabilities that BERT assigns to various words at the randomly selected 15% of the text.

in this thesis and in the related work. The effectiveness of a data augmentation technique can be empirically shown by improving a model's performance on these tasks.

2.3.1 Sentiment Classification

Consisting of 10,662 sentences from the Rotten Tomatoes movie review website, the Stanford Sentiment Treebank (SST) [32] is a sentiment classification corpus. Its original form is SST-5, where each example has a sentiment label in {-2, -1, 0, 1, 2}. The sign and magnitude of the label represent the polarity and intensity of sentiment, respectively. Its binary form, SST-2, contains only the polarity by excluding examples with label 0 and labeling only by the sign of the original label. To my knowledge the highest achieved accuracy on the SST-2 test set is 97.4%, achieved in [30].

2.3.2 Subjectivity Classification

The Subj [26] dataset contains 10,000 sentences taken from movie review websites, balanced and labeled as objective or subjective. Objective sentences give facts about the movie being reviewed; subjective sentences give the writer's opinion about the movie. For example, "in the year 2009, a new drug known as blood heat is developed." and "the script is a tired one, with few moments of joy rising above the stale material." are respectively an objective and a subjective sentence in the dataset. To my knowledge the highest reported 10-fold cross validation accuracy on the dataset is 95.5%, accomplished by the authors of [46].

2.3.3 Constructiveness Classification

The Simon Fraser University Opinion and Comments Corpus (SFU) [17] is a collection of 663,173 comments from the Globe and Mail. Of the 663,173 comments, 1043 were annotated along two axes: toxicity and constructiveness. Toxicity is made an ordinal variable with 4 possible values, and constructiveness is a binary variable. This thesis will focus on the constructiveness label. The purpose of the constructive label is to define comments that encourage a thoughtful dialogue about the article's subject matter. A comment is considered constructive if it creates a dialogue, makes a well-defined point, offers solutions to problems or answers to questions posed by the article or by other commenters, or shares relevant personal experience.

2.4 NLP Concepts

This section discusses several tools and ideas in NLP that are relevant to this thesis. Text preprocessing and embeddings are more prevalent in NLP than are language modeling, POS tagging, and WordNet. However, all five of these ideas and tools are important to this thesis.

2.4.1 Text Preprocessing

Proper preprocessing is essential to any NLP problem. Preprocessing techniques are chosen specifically for the problem and the model being used. All preprocessing involves striking a balance between keeping information and making the text more understandable from the model's point of view. For example, lowercasing all letters reduces the data space by nearly half (thus reducing the necessary amount of training data), but ungrammatical capitalization is widely recognized as a sign of anger or distress.

A common step in text preprocessing is dividing a text into a series of *tokens*, where each token is itself a series of contiguous characters. Usually the division (i.e., *tokenization*) is such that tokens are separated by spaces or between a space and punctuation, but sometimes single words are divided into multiple tokens.

2.4.2 Embeddings

Commonly used as the first layer in NLP models, word embeddings are mappings from words to vectors. They tend to assign similar vectors to words that have similar meanings or often appear together. Frameworks vary, but in general word embeddings are generated by self-supervised training on large corpora. Word2Vec is a common word embedding framework and will be used throughout this thesis. It captures relationships between words that humans can immediately recognize. A famous such relationship is $v_{queen} - v_{woman} \approx v_{king} - v_{man}$, where v_{word} is the Word2Vec embedding of *word*. This can be interpreted as "queen is to woman as king is to man".

Word2Vec and most other word-level embeddings are defined only on a predefined vocabulary of words. When using these embeddings, out-of-vocabulary (OOV) words are either dropped entirely or replaced with a designated OOV token.

2.4.3 Language Model (LM)

Language models are NLP models that learn certain probabilities associated with their corresponding training set. The simplest and most common language modeling problem is next-word prediction, where a model learns the probability distribution over the next word, given the k words that precede it.

The language model used to pretrain BERT is the masked language model (MLM), where rather than predicting the next word, it predicts the word or words that have been masked out. Using language modeling as the transfer learning source task has had remarkable success, likely because it forces the model to learn syntax and semantics. It is a self-supervised problem and therefore does not require labeled training data. Its performance as a pretraining problem for transfer learning is better than autoencoding and translation [44, 37], the two other common source tasks.

2.4.4 POS Tagging

A part of speech (POS) is a class of word distinguished by the function it serves in a sentence [1]. Examples include adjectives, adverbs, nouns, and verbs.

POS prediction, commonly referred to as POS tagging, is an active field of study. While models can be compared by their performance on certain benchmark datasets, which model is state of the art is up for debate. However, the best performing POS tagging models are neural models trained via supervised learning.

2.4.5 WordNet

Princeton's WordNet [22] is a longstanding project wherein expert linguists maintain a structured network of English words. The fundamental components of WordNet are unordered sets of words that are interchangeable in a certain context, referred to as synsets. WordNet includes only the four major word classes: nouns, adjectives, verbs, and adverbs. Synsets are connected through various semantic relations, such as subordinate and superordinate. For instance, synset A is subordinate to synset B if all members of A is a B. I refer to the union of WordNet-generated synsets as the set of *candidates*.

If replacing the current word significantly changes the meaning of the sentence, I say that the candidate is inaccurate. Users can condition on a word's POS tag when querying WordNet, which often decreases the number of inaccurate candidates. For example, by restricting a WordNet query to nouns when finding replacements for “order” in the sentence *he obeyed the order*, the verb synonyms of “order”, such as “arrange” are eliminated from the candidates. Unfortunately POS tags are the only way in which the context of word can be used to inform the query, so there is still a chance of inaccurate candidates. Borrowing from the previous example, “club” is inaccurate but would be a candidate since “club” and “order” are nouns that are synonyms in at least one sense (an association of people).

2.5 Machine Learning Concepts

This section describes imbalanced datasets and transfer learning. The former being a commonly occurring issue and the second an increasingly popular technique, both are central to this thesis.

2.5.1 Imbalanced Datasets

A binary classification dataset is *imbalanced* if its classes have a significantly different number of examples. If it is not imbalanced then it is *balanced*. Imbalance could be defined using a threshold: one could say that a dataset is imbalanced if its minority class contains less than 40% of the examples. Rather than adhering to such a strict definition, this thesis will treat imbalance as a matter of degree.

Training on imbalanced datasets can lead to issues in machine learning since the model will be biased towards classifying datapoints as the majority class. For example, if a binary health classification model’s training data consists of 95% healthy and 5% sick examples, classifying all examples as healthy would give a training accuracy of 95% and could minimize the training loss function.

This section deals with binary classification tasks, but the entire discussion can be extended to multiclass tasks.

Sampling-Based Methods

The two most straightforward methods of restoring class balance for text and image data are removing majority class examples and duplicating minority class examples, respectively known as undersampling and oversampling.

Class weighting artificially balances the training set by incorporating weights into the training loss function that inflate the cost of misclassifying each minority class example. The weights are an arbitrary constant (often chosen to be the size of the majority class) divided by their respective class's size. Adding weight k to minority class examples is equivalent to using k copies of each minority class example, so class weighting could be seen as an efficient implementation of oversampling. To simplify the discussion, I use the term oversampling to refer to both class weighting and to itself.

Undersampling and oversampling both have drawbacks. In undersampling, the model is deprived of the opportunity to learn from data on hand. Oversampling can lead to overfitting the minority dataset by attaching a relatively high importance to each minority example.

SMOTE

A popular method when dealing with imbalanced numerical data is Synthetic Minority Over-Sampling Technique (SMOTE) [4]. SMOTE iterates through the minority class's datapoints, randomly selects one of its k same-class nearest neighbors, and generates a synthetic point at a random point on the line between the point and the selected neighbor. Its implementation details can vary depending on the practitioner's desired effect, but it is usually repeated until the dataset is balanced.

An augmentation method is *safe* if it maintains the validity of the labels of the datapoints to which it is applied. Like instance crossover (arguably SMOTE's NLP equivalent), SMOTE is not necessarily safe. For example, when training SVMs the synthetic points may lie within the opposite class' true decision boundary, and therefore be mislabeled. See Figure 2.4 for an example of how SMOTE can create problematic data, especially when the training data is noisy. It nonetheless generally performs well: SMOTE has become standard practice and is implemented in scikit-learn [27], a popular machine learning library.

SMOTE is not directly applicable to NLP since it is used on numerical data rather than text data. However, it is comparable to two NLP augmentation methods introduced in Section 3.2: instance crossover and contextual augmentation. Like SMOTE, instance crossover generates synthetic training examples by randomly selecting and combining two training examples. Contextual augmentation also fuses the information contained in multiple examples, but rather than combining only two examples it combines a single example with information from the entire training set (by way of a model that has been trained on the training set).

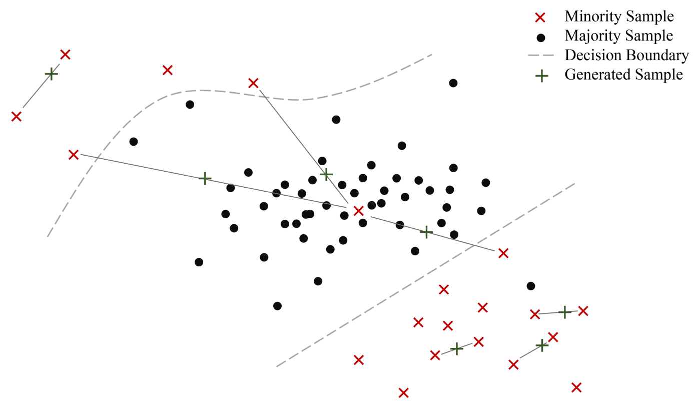


Figure 2.4: An example of SMOTE. The presence of a single mislabeled or noisy point (see centermost minority sample point) greatly increases the probability of problematic data being created. However, in this case and in most cases, a significant majority of the synthetic data is likely to be safe. Figure published in [18].

2.5.2 Transfer Learning

Transfer learning is a method of using knowledge of one problem to assist in learning another. Usually a model is first trained on a large dataset (the "source task"), and this "pretrained" model is further trained on the desired problem (the "target task"). Transfer learning has become ubiquitous in CV since nearly all CV tasks require knowledge of the same low-level and mid-level images features. Concretely, regardless of whether the task is the recognition of humans or the recognition of vehicles, the model must learn edges and shapes, and therefore knowledge of edges and shapes can be transferred from one task to the other.

Using pretrained word embeddings could be considered a form of transfer learning. Pretrained embeddings are low-level feature that are shared between tasks, the NLP equivalent of edges in CV. Recent advancements in NLP, most notably BERT, have shown that this connection between transfer learning in NLP and in CV extends to mid-level features. Just as all image classification tasks require knowledge of shapes, all NLP tasks require knowledge of linguistic representations, syntax, and semantics.

The authors of [11] define a language modeling problem on the Wikitext-103 corpus [20], on which they (1) train an LSTM-based model, then (2) train the model on a language modeling problem for their target dataset, and finally (3) train on the classification problem in the dataset. The call the use of these three steps Universal Language Model Fine-

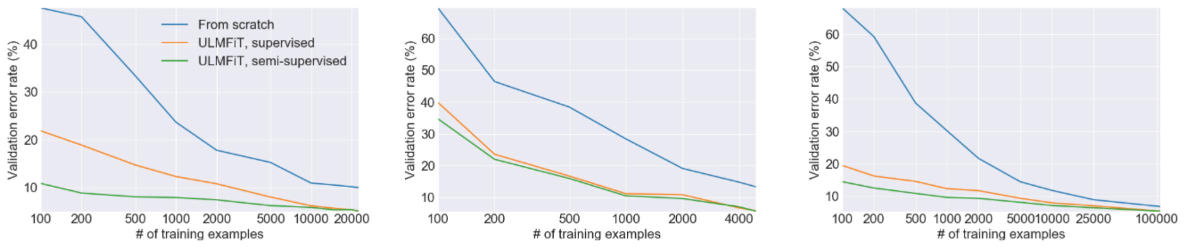


Figure 2.5: The results from [11], on datasets IMDb, TREC-6, and AG (from left to right). Figure published in [11]. "From scratch" corresponds to training without transfer learning; "ULMFiT, supervised" corresponds to pretraining on a language modeling problem for the given classification dataset; "ULMFiT, semi-supervised" corresponds to pretraining first on a language modeling problem for the Wikitext-103 corpus [20], then on a language modeling problem for the given classification dataset.

tuning (ULMFiT). A comparison of this approach with all steps, without step (1), and without steps (1) and (2) is shown in Figure 2.5. Their results show that pretraining offers large accuracy improvements with small training sets, and that the accuracy improvement decreases as the number of training examples increases. On the AG News dataset the non-pretrained model requires more than ten times the training data to achieve the same validation accuracy as a pretrained model.

The model architecture, specifically the input and output layers, often are required to be adjusted between training on the source and the target task. If, for example, the source task is a language modeling problem and the target is a binary classification problem, then the dimensionality of the output layer must be adjusted. In such cases the output layer's weights are randomly reinitialized.

After making the necessary changes to the model architecture, practitioners must decide whether or not to make the non-output pretrained weights trainable. If weights are frozen then the pretrained model acts as a feature extractor for the final layer. Freezing the weights has the benefit of preventing catastrophic forgetting, wherein all knowledge of the source task is overwritten by knowledge of the target task, but significantly restricts the model's ability to learn the target task. Finding an appropriate compromise between lowering the training loss of the target task and avoiding catastrophic forgetting is a central issue in transfer learning.

HuggingFace Transformers [39] is a popular library that implements BERT and other advanced NLP models. In its implementation, when BERT is transferred to sequence classification, a dense layer is concatenated to the final layer and the entire model is retrained.

2.6 Summary

In this chapter, I described the necessary background in NLP and machine learning.

In the next chapter, I present related work in data augmentation for both NLP and a closely related field, CV.

Chapter 3

Related Work

In this chapter, I review previous work on data augmentation in both CV and NLP. I briefly discuss the analogues that certain CV augmentation methods have in NLP. In order to further connect past work in augmentation to the analysis in this thesis, I make note of each method's example-level augmentation distance (ELAD), should one exist.

The ELAD is the distance that each augmented example deviates from its original form. The concept of ELAD is original to this thesis, although other work may discuss a similar notion either directly or indirectly. See Section 4.1 for a more detailed explanation of this term and its relevance to this thesis.

Like in NLP, neural models have become the state of the art for most CV tasks. There are many parallels between the two fields. Both previously relied on engineering context-ignorant features for input to traditional machine learning models. Both benefit from neural models' ability to automatically learn good features for a given problem or dataset. However, while data augmentation is ubiquitous in CV, it has yet to become standard practice in NLP. The highly variant neural models common to both fields can benefit from augmentation, but useful augmentation techniques in NLP are less obvious than in CV.

Many CV tasks (specifically the tasks' labels) are invariant with respect to variance in viewpoint, lighting, and scale [31]. Therefore augmentation by rotation, cropping, or mirroring can increase model robustness. These simple geometric transformations have no clear analogue to NLP. Applying mirroring, for example, to NLP would result in incoherent sentences since language is sensitive to word order.

3.1 Data Augmentation in CV

This section explores common augmentation techniques in CV. It is not an exhaustive list, and more time is spent on the techniques that have analogues in NLP.

Although there have been findings [12, 33] to the contrary, it is generally accepted that a CV augmentation method must be safe (i.e., label-preserving) in order to be effective. Ability to preserve labels will depend on the nature of the label, so an augmentation technique's performance will vary across tasks. Mirroring along its vertical axis, for example, would be detrimental to recognizing certain handwritten digits.

3.1.1 Geometric Transformations and Noise Injection

Effective without being computationally expensive or requiring external information, mirroring, rotation, cropping, translation, and noise injection are standard practice in a wide range of CV applications. Though it is specified indirectly (by way of a parameter), the latter three of the listed methods all have an inherent ELAD while the others do not. For example, the magnitude and coverage of the noise that is injected into an image represents the ELAD of that particular technique.

3.1.2 Image Mixing

Counterintuitively, mixing portions of multiple images has had promising results. Inoue [12] found that SamplePairing, averaging two randomly selected images and assigning the composite image a random choice of the two images' labels, was able to reduce the error rate on the CIFAR-10 dataset from 8.22% to 6.93%. Inoue also showed that SamplePairing reduced the error rate from 43.1% to 31.0% when training on 10% of the CIFAR-10 dataset. These two results show that SamplePairing offers a large performance boost when the training set is small, and as the training set increases in size this boost diminishes but remains pronounced. Surprisingly, constraining image pairing by only selecting those with the same label led to worse results. The nonlinear methods of mixing images shown in Figure 3.1 were explored in [33]. VH-Mixup, the best performing method, reduced the error rate on CIFAR-10 from 5.4% to 3.8%.

It has been suggested that these methods help the model learn lines and other low-level elements [31]. The effectiveness of these techniques should be compared to transfer learning, which also helps models learn low-level elements.

Image mixing does not have a discernible ELAD. Instance crossover, the NLP equivalent of concatenation-based image mixing, is discussed in Section 3.2.4.

3.1.3 Random Erasing and Random Replacement

Random erasing, introduced in [47], is analogous to dropout, but in the data space rather than the model space. The same approach in NLP, namely the random dropping or masking of words, is found to have modest positive results in [38].

Although it is left unstudied, the authors of [21] suggest combining random erasing with image inpainting. Generative Adversarial Networks (GANs) have achieved state of the art results on image inpainting, the task of predicting missing portions of images. This approach would be analogous to contextual augmentation, a method discussed in Section 3.2.3.

The ELAD of both of these methods is the proportion of the image that is erased. Both of these methods are unsafe for a wide variety of tasks. For example randomly erasing or replacing a person in an image would contradict the label in a human localization task.

3.1.4 Artificial Data

GANs have also exhibited excellent performance in artificial data generation, in terms of both results and computational speed. Shorten and Khoshgoftarr [31] note that GANs are the most promising generative modeling technique for use in data augmentation. Traditionally GANs use multilayer perceptron networks as both the generator and the discriminator. The generator creates the images, and the discriminator distinguishes between real and artificial images. The GAN framework, defined by the presence of a generator and discriminator, can take a variety of different forms.

Progressively Growing GANs can raise the resolution of various input images by generating new pixels that conform to the input image and the training set. This approach has shown promising results on facial images [15].

Deep Convolutional Generative Adversarial Networks (DCGANs) use CNNs for both the generator and the discriminator [29]. Since the state of the art models in CV are CNN-based, it is unsurprising that DCGANs have excellent results when it comes to generating high resolution images.

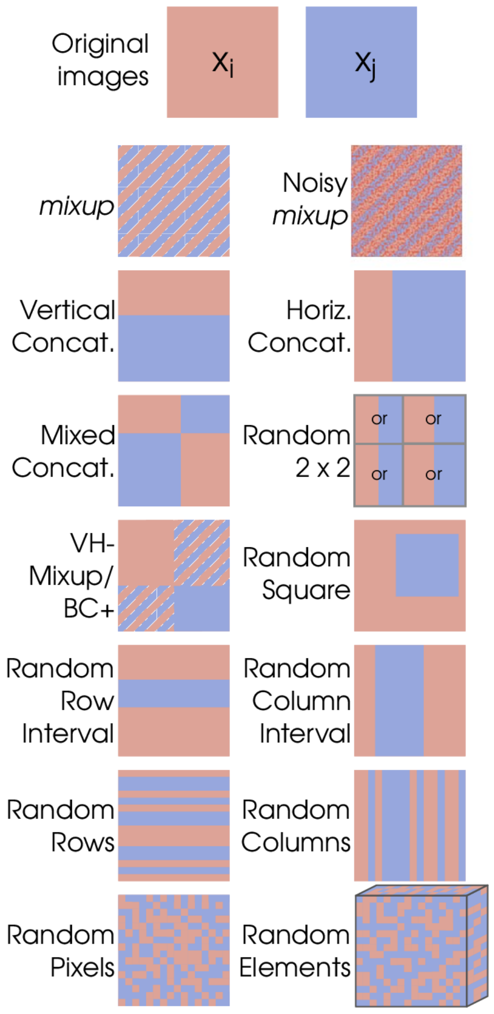


Figure 3.1: Illustrations of the nonlinear image mixing methods explored in [33]. At testing, VH-Mixup achieved the best performance. Figure published in [33].

Since an artificially generated training example is not derived directly from an original training example (it is instead derived from the training set as a whole), the concept of ELAD does not apply to this method.

3.1.5 Style and Class Transfer

Style transfer via neural networks, an approach first introduced in [9], generated significant excitement around deep learning. Numerous improvements to [9] have been published, most notably Fast Style Transfer [13]. A much less computationally expensive algorithm, Fast Style Transfer has allowed style transfer to become more widely used in practice.

CycleGANs can be used to transfer images from one class to another. For example, the authors of [48] are able to convert images of zebras into images of horses and vice versa, all while maintaining the image background. This is achieved by using one generator for each direction, and a discriminator with its usual function. These are used in [49] to artificially balance the Facial Expression Recognition Database by translating images from larger classes to minority classes. This is analogous to SMOTE [4], but with images rather than numerical data. This technique led to a 5%-10% accuracy improvement of the models tested in [49].

The ELAD of this technique is difficult to identify in the abstract. The ELAD would depend on the details of how the style or class transfer is conducted.

3.1.6 Test-Time Augmentation

Test-Time augmentation, performed during testing rather than training, involves passing not only the original example into a model, but also passing one or more augmented versions of that example. The model outputs for each version of the training example are fused in some way, often through a voting scheme. As illustrated in Figure 3.2, it leads to accuracy improvements on a skin lesion detection dataset [28]. Augmentation at train and test time performs better than only at train time or no augmentation at all. For small dataset sizes, training augmentation yields worse performance than no augmentation, yet training and testing augmentation outperforms both of these schemes. Both of these findings are counterintuitive: it is generally accepted that training augmentation is most helpful on smaller datasets. Further, it is unclear why testing augmentation is helpful in cases when training augmentation is harmful.

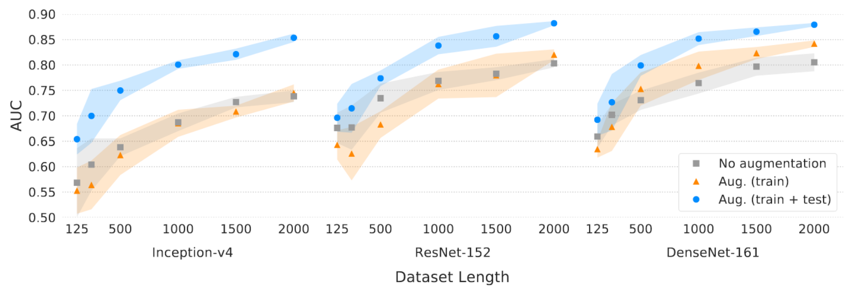


Figure 3.2: Experimental results on the skin lesion dataset in [28] where augmentation is done by random crop, affine transformation, flips along x or y axis, and brightness changes, in that order. Because the augmentation method, and therefore the training set, is stochastic the performance of the model is averaged across six training sets. The mean of each of the six runs are indicated by points, with shading to represent the standard deviation. Figure published in [28].

The main drawback of this technique is the added computation time at testing. While the inputs could be processed in parallel before combining the results, this still requires more computational resources.

3.2 Data Augmentation in NLP

In this section I review previous work on data augmentation for NLP. I describe the work that other researchers have done on this problem and show where these previous approaches are inadequate (and briefly how my approach removes these limitations).

Like the discussion in the previous section, I state what I consider to be the ELAD of each technique. While the ELADs in the CV augmentation methods are well-defined, the ELADs of the NLP augmentation methods tend to be more subjective and debatable. For example, the ELAD of backtranslation could depend either on some measure of distance between the pivot and the original language or on the sampling temperature of the translation models, or both.

3.2.1 Synonym Replacement

Synonym replacement is likely the most obvious augmentation technique for NLP. It does not have a clear analogue in CV augmentation techniques; while text is naturally broken down as a series of words, the only universal segmentation of images is at the pixel level. Unlike words, individual pixels of an image contain very little information. In the literature, synonyms are determined in either of two ways: by a predefined thesaurus or by a word embedding similarity measure.

The ELAD of synonym replacement can be divided into two subdistances: the number of words to be replaced in each example, and the distance between the original word and its replacement synonym. The latter depends on the method of determining the synonyms.

Using WordNet Synsets

Zhang and LeCun [45] generate synonyms using WordNet synsets (see Section 2.4.5). For each training example, they draw both the number of words to be replaced and the index of the replacement word from two independent geometric distributions with parameter 0.5. The authors do not discuss how either parameter was selected, whether it be through a systematic search or an arbitrary choice. Augmentation-related experiments are run on four classification datasets using two different sized CNNs. These are character level CNNs, so no embeddings are used. Across the four datasets the model without augmentation achieves an accuracy as low as 70.45% and as high as 98.27%; augmentation increases the accuracy in all cases by between 0.13% and 0.65%.

Wei and Zou [38] also generate synonyms using WordNet, and test the technique's effect on the performance of an RNN and CNN. They average their performance across five datasets for fixed dataset sizes of 500, 200, and 5000. Augmentation via synonym replacement offers a 2% performance gain when 500 training examples are used. This gain diminishes as the training set becomes larger: it is limited to 0.5% for a training set size of 5000.

Using Cosine Similarity

Synonym replacement using cosine similarity of Word2Vec embeddings (see Section 2.4.2) is studied in [36]. The authors generate augmented data with the same parameters as [45], where each word's list of synonyms is given by the words with which it has cosine similarity of no less than 0.25. Words with higher cosine similarities are assigned higher probabilities of being selected. When the classification model accepts Word2Vec embeddings as input, it is advantageous to use cosine similarity to determine synonyms since the synonym is certain to be in the model's vocabulary. The authors do not test the performance of the models without augmentation, so the effectiveness of this technique cannot be determined.

That being said, this approach can be expected to offer similar performance boosts to those offered by WordNet synsets.

In this approach to synonym replacement, the distance between the original word and its replacement is given by their cosine similarity.

3.2.2 Backtranslation

Backtranslation, wherein text is translated to another language (the "pivot") and back to the original language, has been used to improve translation and to evaluate paraphrasing models. The concept of using it to augment text data is introduced and evaluated in [43]. The authors, using four state-of-the-art translation models trained on publicly available datasets, use both French and German as pivot languages. The promising results of this approach on the SQuAD dataset are shown in Table 3.1.

Training Set (Original:French:German)	EM	F1
(1:0:0)	73.6	82.7
(1:1:0)	74.5	83.2
(1:1:1)	74.8	83.4
(1:2:1)	74.3	83.1
(2:2:1)	74.9	83.6
(2:1:1)	75.0	83.6
(3:1:1)	**75.1**	**83.8**
(4:1:1)	75.0	83.6
(5:1:1)	74.9	83.5

Table 3.1: Exact match and F1 score of the model on the development set when trained on the training set given by each ratio of original to augmented data. Note that a ratio of (1:0:0) corresponds to zero data augmentation, and so each instance of data augmentation improves performance under both metrics.

The authors of [41] perform backtranslation using a pair of English-French translation models, one for each direction. The translation models are trained on the WMT '14 English-French corpus[1], a massive benchmark dataset consisting of human-translated sentence pairs. Both translation models and their checkpoints are made publicly available[2]. They propose a semi-supervised framework, where the total loss function is the sum of

[1] https://www.statmt.org/wmt14/translation-task.html
[2] https://github.com/google-research/uda

two loss functions. The first is a standard supervised learning loss function, and the second is "Unsupervised Consistency Loss", a distance measure between the two outputs of the model for an unlabeled example and the same example, backtranslated. The semi-supervised framework itself is orthogonal to the purpose of this paper, but it should be noted that the authors achieve state of the art results while relying on backtranslation.

While the authors of [43] always select the most likely translation, the authors of [41] randomly sample from the list of possible translations. These lists are generated by random sampling with a tunable temperature parameter. They suggest that the diversity that this stochastic approach introduces is an important factor in the large performance improvements achieved in their experiments.

The ELAD of backtranslation could be defined by some measure of the distance between the original and the pivot language. When backtranslations are generated through random sampling, the ELAD could instead be defined as a function of the sampling temperature parameter.

3.2.3 Contextual Augmentation

Contextual augmentation, the NLP analogue to inpainting in CV (described in Section 3.1.3), uses word prediction models to replace randomly selected words. The word prediction model can be trained on the same training set, on another, or both. The framework, as presented in [16, 40], is similar to synonym replacement (see Section 3.2.1), where the words to be replaced and the replacement words are both chosen by some probability distribution. In this case, however, the words are selected from the k most likely words, as determined by the word prediction model (see Section 2.4.3).

This approach also differs in that it does not offer synonyms, instead suggesting words that have *paradigmatic relations* with the original word. Two words are said to have paradigmatic relations if they can be substituted for one another while preserving the grammatical correctness of the sentence in which they appear. Replacement with any word that shares paradigmatic relations could drastically alter the meaning of the sentence, thus rendering the augmentation technique unsafe. For example, "best" and "worst" share paradigmatic relations, but the sentences "Hitchcock at his *best*" and "Hitchcock at his *worst*" would not share the same label in a movie review sentiment classification dataset. This problem is mitigated by a "label-conditional architecture", wherein the label is built into the input of the word prediction model during both training and inference. In this architecture the training loss is minimized when the model suggests words that do not contradict the label.

The words suggested by a label-conditional model do not necessarily preserve the meaning of the sentence, but compared to a label-independent model they have an increased likelihood of preserving the label. For example, Figure 3.3 shows that "the *actors* are fantastic" may be changed to "the *movies* are fantastic". They do not have the same meaning, but both are positive reviews.

Like in synonym replacement, the ELAD of contextual augmentation can be divided into two values: the number of words to be replaced in each example and the index of the replacement word. The index is given by the output of the contextual augmentation model, sorted by decreasing probability. Note that other definitions of the ELAD for contextual augmentation are possible. For example, the ELAD could also be the number of words being replaced and the inverse of each substitute word's assigned probability.

The following two label-conditional Contextual Augmentation techniques have both had promising results.

Using Bidirectional LSTM

A bidirectional LSTM is used for word prediction in [16]. This model is pretrained on unlabeled WikiText-103 corpus [20], then the architecture is made to be label-conditional and is trained on the task's training set. The word prediction model outputs a probability distribution over the candidate replacement words. They sample replacement words from this probability distribution after transforming it with a temperature parameter τ, where the distribution becomes uniform as $\tau \to 0$ and the highest probability is always selected as $\tau \to \infty$. The authors replace each word with probability q (i.e., if a sample from a Bernoulli distribution with parameter q is true). Both the temperature τ and Bernoulli parameter q are tuned via grid search.

Performance of a CNN and a bidirectional LSTM classifier with no augmentation, label-independent augmentation, and label-dependent augmentation is tested on the SST-2, SST-5, Subj, and three other benchmark datasets. Unsurprisingly, the performance is improved by the label-independent augmentation and further improved by the label-dependent augmentation. Averaged across the 6 datasets, the label-dependent augmentation raises the accuracy of the CNN from 77.53% to 78.83% and the accuracy of the RNN from 77.43% to 77.83%.

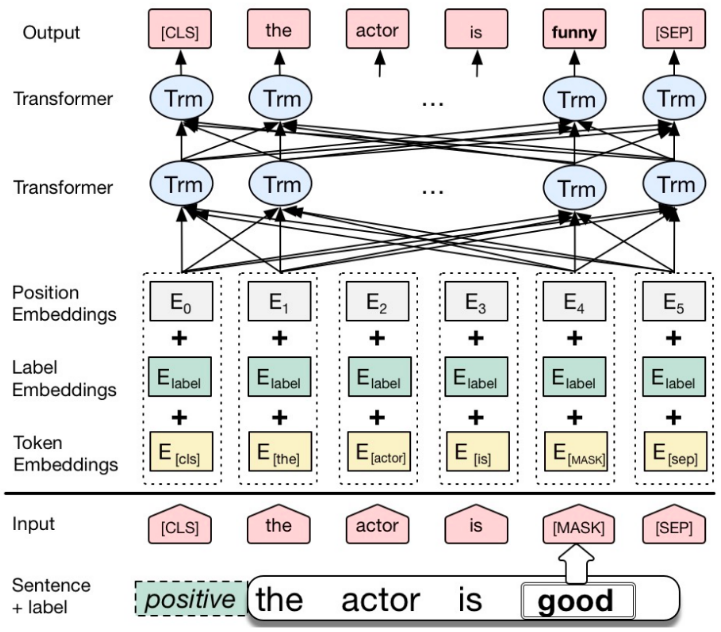


Figure 3.3: The label-conditional BERT architecture used for word prediction in [40]. The label is built into the input by replacing the segmentation embeddings with label embeddings. Figure published in [40].

Using BERT

Pretrained BERT (see Section 2.2.4) is used for word prediction in [40]. Adhering to the transfer learning paradigm, the model is fine-tuned on each task's training set with the architecture and input design shown in Figure 3.3. Note that this approach differs from the bidirectional LSTM in that the label is built directly into the input, rather than concatenated to a middle layer of the model.

3.2.4 Instance Crossover

Introduced in [19], instance crossover consists of splitting tokenized sequences into halves, then randomly sampling one first and one second half whose labels agree, and concatenating. The authors use ensembles of linear classifiers to predict the sentiment of Spanish language tweets. The classifiers use either Bag of Characters (BoC), Bag of Words (BoW), or pretrained embeddings as features for the classifier. All of these features are independent of word order, and therefore are less sensitive to the obvious pitfall of instance crossover: the high likelihood of incoherent training examples due to the discontinuity at the concatenation boundary.

They focus their data augmentation analysis on a dataset with 1126 training examples with sentiment labeled as positive, neutral, negative, or none. Since the dataset is imbalanced the classification metric is the macro-averaged F1-score (M-F1) (the mean per-class F1-score). They tune the augmentation ratio (post-augmentation dataset size to pre-augmentation dataset size) of instance crossover, and find that the accuracy improves for all tested values (4, 8, 12, 16, 20). During ablation testing they find that the full system achieves an accuracy and M-F1 of 64.37% and 52.77%, and that without instance crossover the accuracy remains the same but the M-F1 degrades to 47.57%.

M-F1 is more sensitive to minority classes, so these results may imply that instance crossover offers strong accuracy boosts when using especially small training sets or training sets with very few examples in one or more classes.

Note that this is the NLP equivalent of horizontal concatenation, one of the image mixing CV augmentation techniques described in Section 3.1.2. Instance crossover likely does not help (or could even harm) the performance of NLP neural models since they are dependent upon word order, although the success of image mixing in CV may suggest otherwise.

3.3 Summary

In this chapter, I described the previous work on data augmentation for NLP and CV tasks. I showed that several CV augmentation methods have analogues in NLP. I discussed the limitations of the past studies and summarized the performance of the studied augmentation methods.

In the next chapter, I present my solution and show how it addresses these limitations.

Chapter 4

Study Methodology

In this chapter, I provide an explanation of the methods and techniques that I use to analyze the performance of data augmentation in text classification tasks.

Each experiment will consist of an *augmentation technique* being used for an *application method*, to assist a *classification model*'s performance on a *dataset*. For example, backtranslation, used to rebalance an imbalanced dataset, will assist an RNN's performance on the Subj dataset. Note that when referring to a dataset in this manner I do not necessarily mean the entire dataset. For example, when the application method is dataset balancing, the dataset will be made imbalanced by excluding a portion of the data. Each of the four emphasized terms will be defined and explained in its own section of this chapter.

4.1 Augmentation Distances

Inherent to every augmentation technique is the *distance* that the augmented training set deviates from the original training set. I divide this distance into two subdistances: the ratio of augmented data to original data and the distance that each augmented example deviates from its original form. I call the former the *augmentation ratio* and the latter the *example-level augmentation distance (ELAD)*. The ELAD of word-replacement techniques such as synonym replacement and contextual augmentation can be further divided into the number of words to replace and the 'distance' between the original and the replacement words (according to some similarity measure given by the replacement word lookup method).

Recognizing that the power of data augmentation lies in the diversity that it introduces the training set, I make both subdistances stochastic rather than deterministic. The augmentation distance for each of the three techniques that will be tested is determined by a parameter p, where p is a probability. This parameter governs a geometric distribution, and the augmentation distance increases as p decreases. This parameter governs two subdistances: the ratio of original data to augmented data and the distance between each augmented example and its original form.

4.2 Augmentation Techniques

The following three data augmentation techniques will be studied. In each case, the value of p will be optimized via grid search on a validation set. In order to minimize the time complexity of augmentation during training, I create the augmentation data before model training.

4.2.1 POS-Informed Synonym Replacement

I propose POS-informed synonym replacement, an improvement upon the synonym replacement methods proposed in previous work. This method decreases the probability of erroneous replacement by placing an additional constraint on each word's synonyms.

Similar to [45], lists of candidate synonyms will be generated using WordNet. WordNet synsets are unordered but the same word can appear in multiple synsets, so I sort these lists by number of appearances in synsets, in decreasing order with ties broken randomly. The procedure that generates the list of candidate synonyms is given in Algorithm 1.

Despite its centrality to a word's function within a given sentence, the implementations of synonym replacement in the related work are both agnostic to POS. I will utilize WordNet's POS-specific synsets by constraining the candidate list of synonyms by the word's POS, as inferred by a Natural Language Toolkit (NLTK) POS tagger [2]. I will use NLTK's default POS tagger, the Greedy Average Perceptron tagger. Consider the following sentence: "What vowel do all Esperanto nouns end in ?". When finding POS *independent* synonyms for "end", WordNet returns one invalid and three valid suggestions: "goal", "finish", "cease", and "terminate". However, the NLTK POS tagger recognizes that in this case "end" is functioning as a verb. When this constraint is passed into the synonym search function, "goal" is correctly eliminated from the list of synonyms.

Algorithm 1: Synonym data creation algorithm. POS tags are determined by the NLTK python library. Synonyms are determined by WordNet synsets.

```
1  function get_synonym_dict ($D$);
   Input  : $D$, a list of pairs: tokenized examples and their labels
   Output: $D'$, a list of triplets: the augmentation dictionaries and the original
           example-label pairs
2  $D' \leftarrow$ empty list
3  for example, label pair in $D$ do
4      $d \leftarrow$ empty dictionary
5      for token and its index in example do
6          if token is a word then
7              $d[index] \leftarrow$ list of synonyms of the word with agreeing POS tags
8          end
9      end
10     Append example, label, $d$ to $D'$
11 end
12 return $D'$
```

4.2.2 Contextual Augmentation with BERT

Augmentation with BERT, as proposed in [40], will be implemented. First, the pretrained BERT model is fine-tuned on the training set's label-dependent masked language modeling problem. Then the algorithm loops through every token of every example, and if the token is a word it is masked out. For each such token, the ten words to which the fine-tuned BERT assigns the highest probability are saved. This algorithm is given in greater detail in Algorithm 2.

4.2.3 Backtranslation

Backtranslation is performed using the Fairseq WMT'19 English-German models [24]. Both models are randomly sampled with a temperature parameter. I do not tune this parameter and instead use the same value as [41], a related study of backtranslation discussed in Section 3.2.2. I create and save 25 backtranslations for each training example using the procedure outlined in Algorithm 3.

It is common for the same translation to appear multiple times, and I claim that the ELAD is inversely proportional to the number of times a backtranslated example appears.

Algorithm 2: Contextual augmentation data creation algorithm. "D's Masked Language Model problem" refers to the word-prediction problem on which BERT is pretrained. BERT's tokenizer often breaks words into root words, suffixes, and prefixes. To maintain a high probability of the candidate tokens being consistent with its surrounding tokens, I only mask out entire words, and only accept words as candidate replacement tokens.

```
1  function get_context_dict ($D, n, k$);
   Input  : $D$, a list of pairs: tokenized examples and their labels
            $n$, the number of fine-tuning epochs
            $k$, the number of candidate replacement tokens to collect for each word
   Output: $D'$, a list of triplets: the augmentation dictionaries and the original
            example-label pairs
2  Load the pretrained BERT model
3  Replace BERT segmentation embeddings with label embeddings
4  For $n$ epochs train BERT on $D$'s masked language model problem
5  $D' \leftarrow$ empty list
6  for example, label pair in $D$ do
7      $d \leftarrow$ empty dictionary
8      for token and its index in example do
9          if token is a word then
10             Pass the example, with the current token masked out, into BERT
11             $d[index] \leftarrow$ list of the $k$ words (excluding prefixes and suffixes) to which
                BERT assigns the highest probability of being the masked token,
                ordered by decreasing probability
12         end
13     end
14     Append example, label, $d$ to $D'$
15 end
16 return $D'$
```

This forms an ELAD since backtranslated examples that are closer to the original example's meaning can be expected to be produced by the translation models more often. Varying the temperature parameter of the translation models would likely be a more effective and accurate ELAD, but generating backtranslated examples in advance of training yields significant time savings.

Algorithm 3: Backtranslation data augmentation creation algorithm.

```
1  function get_translation_dict ($D, T, k$);
   Input  : $D$, a list of pairs: tokenized examples and their labels
            $T$, the translation model sampling temperature
            $k$, a square number, the number of replacement examples to create for
   each example
   Output: $D'$, a list of triplets: the augmentation lists and the original
            example-label pairs
2  Load pretrained English to German and German to English translation models
3  $D' \leftarrow$ empty list
4  for example, label pair in $D$ do
5      $L_1 \leftarrow \sqrt{k}$ outputs of the English to German translation model with example as
        input and sampling temperature $T$
6      $L_2 \leftarrow$ empty list
7      for German example in $L_1$ do
8          $L_2 \leftarrow L_2 +$ list of $\sqrt{k}$ outputs of the German to English translation model
            with German example as input and sampling temperature $T$
9      end
10     Append example, label, $L_2$ to $D'$
11 end
12 return $D'$
```

4.3 Classification Models

I will test the augmentation methods with an RNN and with BERT. The former being randomly initialized and one of the most common NLP models and the latter being a foundational model in modern NLP, testing with these will give a sense of how widely these augmentation methods can be used.

4.3.1 RNN

Despite the fact that RNN-based models are generally outperformed by BERT, they remain popular because they are smaller and less computationally expensive. I will use a bidirectional LSTM initialized with randomly initialized weights, and pretrained Word2Vec embeddings with a predefined out-of-vocabulary vector.

4.3.2 BERT

These methods will also be tested on BERT to see if they are able improve the performance of the foundational model of modern NLP. An augmentation's good performance on an RNN does not necessary imply good performance on BERT, since there is a chance that the performance boosts offered by these techniques are already "baked in" to a pretrained BERT model. As discussed in Section 2.5.2, pretrained language models require less training data than randomly initialized models. Therefore it is possible that BERT will benefit from augmentation significantly less than the RNN will, or even that the inherent noise of the augmentation techniques will harm the model's performance.

XLNet, Google Brain's successor to BERT, outperformed BERT on a variety of NLP tasks and datasets, as shown in its introductory paper [42]. I test these augmentation methods using BERT instead of XLNet for two reasons. First, its smaller size will allow more experiments with more random seeds to be conducted in the same amount of time. Second, the effect that these augmentation techniques have on BERT can be assumed to be strongly predictive of their effects on future NLP models. This is because BERT is considered the foundational model for transfer learning in NLP.

In order to avoid catastrophic forgetting, I will use a low learning rate and only 3-4 training epochs, as suggested in the BERT paper [8] and validated in [34]. To my knowledge, this is the first time that any of the listed data augmentation techniques have been used to train BERT.

4.4 Datasets

Experiments will use SST, Subj, and SFU, the classification datasets whose labels are sentiment, subjectivity, and constructiveness, respectively. See Table 4.1 for a statistical summary of these datasets, and Section 2.3 for a brief explanation of each of them.

Testing on these benchmarks will be a strong indication of the efficacy of the various proposed methods and how they are likely to behave in NLP more broadly.

Dataset	l_{mean}	N_{train}	$N_{development}$	N_{test}
SST	19	6,920	872	1,821
Subj	24	10,000	-	-
SFU	62	1,043	-	-

Table 4.1: The mean number of words in each example, and the number of examples in each of the datasets' subsets.

4.5 Application Methods

This section explains the two ways in which augmentation will be applied. Together they allow for a study of how the effects of augmentation change with respect to both the size and the degree of imbalance in a dataset.

4.5.1 General Augmentation

Each augmentation method will be tested on varying percentages of each dataset. These results will be used to construct learning curves that will show how the performance with and without augmentation varies with respect to dataset size. One can also extrapolate from a learning curve, and estimate the dataset size at which the performance boost becomes negligible, if such a point exists.

4.5.2 Augmentation for Dataset Balancing

All datasets are almost exactly balanced, so they will be made imbalanced by only including a percentage of one label's examples. Artificial rebalancing via each augmentation technique will then be compared to oversampling and undersampling.

4.6 Augmentation Algorithms

While Algorithms 1, 2, and 3 generate the augmentation candidates, Algorithms 4 and 5 integrate those candidates into the data during model training. Both algorithms accept as input an original training example, its augmentation candidates, and a probability p. Depending on the result of sampling from a probability distribution governed by p, either the original example or an augmented example is returned. This section explains the two algorithms and justifies certain differences between these procedures and those found in the related work.

4.6.1 Synonym Replacement and Contextual Augmentation

Algorithm 4 is used for synonym replacement and contextual augmentation. This algorithm draws both the number of words to be replaced and the index of each replacement word from independent geometric distributions with parameter p.

This implementation of synonym replacement is in keeping with [45], a related work summarized in Section 3.2.1.

This approach to contextual augmentation differs from the paper in which it is introduced, which samples replacement words from the probability distribution output by the model [40]. My approach uses the output probability distribution to order (but not select) the replacement words. This is to avoid the memory costs of saving all candidate replacement words and their probabilities. Alternatively, the k highest candidate replacement words could be saved along with a transformed version of the original probability distribution. However, I reason that the distortion of the probability distribution due to this transformation would be too large to justify the extra memory costs. Further, my approach of sampling from an ordered list via a geometric parameter allows a single parameter to be used to govern both the number of words to be replaced and the selection of the replacement words.

4.6.2 Backtranslation

Algorithm 5 is used for augmentation via backtranslation. If the Bernoulli distribution with parameter p returns $True$, then a backtranslated example is randomly selected (with each example having an equally likely chance of being selected). This algorithm reflects two ways in which backtranslation is different than the other two augmentation methods. First, the augmentation lists contain entire examples rather than candidate replacement words for specific indices. Second, the augmentation list is unordered.

This backtranslation procedure is the same as that of [41], which is described in Section 3.2.2.

Algorithm 4: Token-level data augmentation algorithm.

```
1  function augment ($s, d, p$);
   Input  : $s$, a dataset example, tokenized
            $d$, a dictionary mapping example indices to an ordered list of candidate replacement tokens
            $p$, a geometric parameter
   Output: $s'$, the augmented example
2  $n \leftarrow$ sample from the geometric distribution governed by $p$
3  $n \leftarrow min(n, len(d))$
4  $s' \leftarrow s$
5  if $n > 0$ then
6      $indices \leftarrow$ randomly sample $n$ keys from d without replacement
7      for $i$ in $indices$ do
8          $j \leftarrow$ sample from the geometric distribution governed by $p$
9          $j \leftarrow min(j, len(d[i]))$
10         $s'[i] \leftarrow d[i][j]$
11     end
12 end
13 return $s'$
```

Algorithm 5: Example-level data augmentation algorithm.

```
1  function augment ($s, L, p$);
   Input  : $s$, a dataset example, tokenized
            $L$, a list of candidate replacement examples
            $p$, a Bernoulli parameter
   Output: $s'$, the augmented example
2  $augment \leftarrow$ sample from Bernoulli distribution with parameter $p$
3  if $augment = True$ then
4      $s' \leftarrow$ random choice from $L$
5  end
6  else
7      $s' \leftarrow s$
8  end
9  return $s'$
```

4.7 Summary

In this chapter, I described my proposed framework for conducting an analysis of data augmentation in NLP.

In the next chapter, I provide more detail on my experimental procedure. I also show my results, explaining where my methods succeeded and where they fell short.

Chapter 5

Evaluation of Data Augmentation for NLP

In this chapter, I evaluate three data augmentation techniques, following the methodology described in the previous chapter. I show how successful these techniques are and attempt to explain the variance in their performance. I discuss the results with a focus on the questions described in Section 5.1.

5.1 Research Questions

As discussed in Chapter 1, data augmentation has proven results in CV, yet is relatively underused in NLP. Both fields use models that require large and fairly balanced training sets, but it is rare for both of these conditions to be present in practice.

The data augmentation methods that will be studied are not original to this thesis (see Section 3.2). However, by addressing the three questions described in this section, I will develop a deeper understanding of how these methods can play a role in addressing issues that frequently arise in NLP. I will also develop an understanding of why these methods do or do not work in certain settings.

The first question is whether NLP data augmentation techniques can reliably yield increases in models' accuracy. Relatedly, I will investigate the relationship between dataset size and the impact of data augmentation.

Next I compare sampling methods to data augmentation in their ability to rebalance imbalanced training sets. This tests whether the data augmentation techniques can outperform either undersampling or oversampling, or both.

Third, I identify the relationship between the optimal augmentation distance (defined in Section 4.1) and two characteristics of a dataset: its size and its degree of imbalance.

5.2 Experimental Setup

In this section I explain and justify the experimental methodology. I explain how consistent the experimental setups are across all datasets, classification models, and augmentation methods. I explain and justify the exceptions to this consistency, most of which are either to accommodate BERT's large size and resultant increased training time or the memory requirements of BERT contextual augmentation.

5.2.1 Model Hyperparameter Tuning

Since the augmentation methods are the focus of this study, I tune only the learning rate and the number of training epochs. The other RNN parameters are set to those used in the related work; the other BERT parameters are set to the values suggested in the BERT paper [8].

I find the optimal learning rate for each of the six dataset/classification model combinations. This learning rate is found by training on the entire training set with no augmentation, measuring performance by the average of 30 trials' validation set accuracy. Note that I assume that the optimal learning rate does not change when augmentation is introduced or as the dataset is made artificially smaller or imbalanced.

RNN

The optimal number of epochs for the RNN and the optimal augmentation parameters are found simultaneously. See Figure 5.1a for an example of the graphs used to select these two parameters when the RNN is the classification model. The augmentation parameter p is selected first (as described in Section 5.2.2) followed by its corresponding optimal number of epochs. Once p is selected, the number of epochs that yields the highest accuracy and is fairly robust (i.e., not a spike at a single epoch) is chosen.

I conducted preliminary tests with 500 training epochs and $p = 0.5$ for each augmentation method on the Subj dataset (the largest of the three datasets). In each case the highest achieved accuracy occurred in the first 100 training epochs. This does not definitively imply that the optimal number of epochs will be less than or equal to 100 for every experiment in this thesis, but I reason that they indicate that any accuracy increases from further training will be insignificant. Considering this finding and the large number of experiments in this thesis, I limited the number of epochs to 100 for the sake of time.

BERT

Like in the case of the RNN, I find that a larger learning rate is beneficial when training BERT on the SFU dataset. The BERT paper [8] suggests limiting the number of training epochs to 3 or 4 in order to avoid catastrophic forgetting (as described in Section 2.5.2). I search over those two values and find that 3 epochs are optimal for the SFU dataset, and 4 is optimal for SST and Subj. See Figure 5.1b for an example of the graphs that are used to select the augmentation parameter p when BERT is the classification model.

For BERT training I implemented a learning rate scheduler that is linear with warmup. This was chosen after tests showed that the corresponding model outperforms one trained with a constant learning rate. Considering the pattern in Figure 5.1b wherein the accuracy is almost always monotonically increasing with respect to the number of training epochs, I also tested a greater number of epochs and found that the optimal number in each case was in fact either 3 or 4. Because the learning rate depends on both the current epoch and the maximum number of epochs, the maximum number of epochs should be chosen prior to augmentation parameter tuning. (This is unlike the RNN wherein the number of epochs and the augmentation parameter can be selected simultaneously.) I therefore fix the maximum number of epochs to the values given in the previous paragraph before tuning p.

5.2.2 Augmentation Distance and Augmentation Parameter Tuning

I conduct a grid search over a range of possible values for p, the geometric parameter that governs augmentation distance. When the RNN is the classification model I search over $0.1, 0.2, \ldots, 0.9$. When BERT is the classification model I make the search roughly half as granular, searching only the odd decimal valued numbers in this list. This adjustment is made in order to accommodate BERT's larger size and resultant increased training time.

As can be seen in Figure 5.1, even after averaging across 30 trials there is still considerable randomness in the RNN training, due to the stochastic nature of both the augmentation techniques and mini-batch gradient descent. The augmentation parameter that has a high accuracy over a relatively large number of epochs is selected. For example, in the shown graph $p = 0.3$ is chosen since its high accuracy is fairly consistent with respect to the number of training epochs, despite the fact that it never reaches the validation accuracy achieved at one point by $p = 0.1$.

5.2.3 Dataset Rebalancing

I compare each augmentation technique's ability to artificially rebalance datasets to oversampling and undersampling. For each percentage of a label's examples that are kept, I sample that percentage x separate times, once for each label. Thus there are $2x$ different training sets, and the results of each of the $2x$ corresponding models are averaged. In the case of synonym replacement and backtranslation $x = 30$ and with contextual augmentation $x = 10$. This increases the results' robustness with respect to the data that is withheld.

5.2.4 General Augmentation

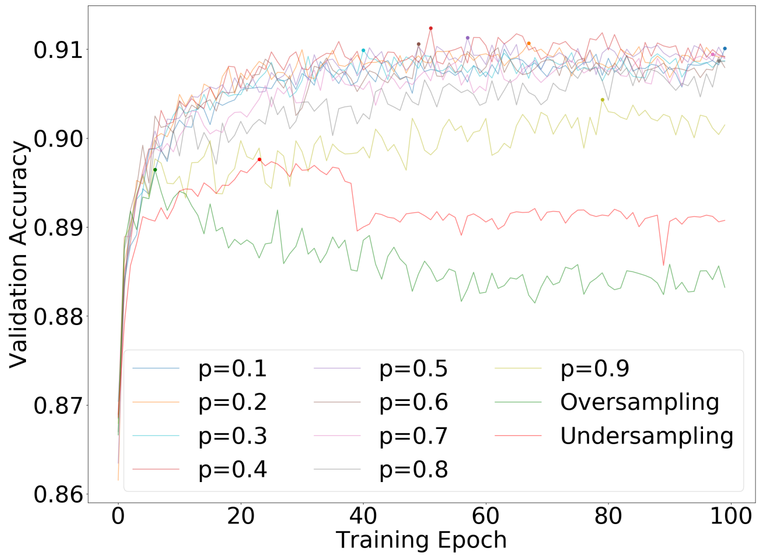
I compare performance with each augmentation technique to performance without augmentation. I conduct these tests on various percentages of the datasets. This allows learning curves to be created, in order to show how the effectiveness of data augmentation changes as datasets become larger. Each percentage is sampled x separate times, with $x = 30$ for synonym replacement and backtranslation, and $x = 10$ for contextual augmentation.

5.2.5 Time-Saving Adjustments to the Analysis

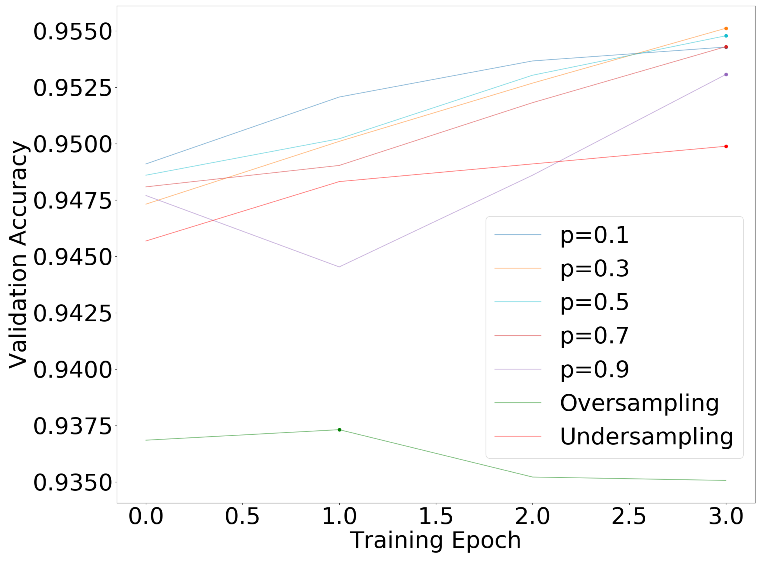
Several adjustments in the experimental procedure are made. These are meant to accommodate certain differences in augmentation techniques, and classification models and datasets.

Classification Model-Specific Adjustments

Because BERT is significantly larger than the RNN and therefore takes longer to train, I vary percentages by 20% rather than 10% in both rebalancing and general augmentation.



(a) RNN



(b) BERT

Figure 5.1: Examples of the graphs used to find the optimal number of training epochs and augmentation parameter p. Both images show experiments with Subj imbalanced to the point of only 20% of one label's examples remaining, with augmentation through backtranslation. The highest achieved accuracy of each parameter setting is marked by a circle.

Dataset-Specific Adjustments

Zero-padding and truncation are used to ensure that all examples in each dataset are of equal length. The examples in the SFU dataset are generally longer than those in the SST and Subj datasets (see Table 4.1). With this in mind, all SFU examples are made to be 128 tokens long and all SST and Subj examples are made to be 25 tokens long. The time costs associated with greater input lengths for SFU are acceptable since that dataset is less than a fifth the size of the other two datasets (as measured by number of training examples).

Because SFU has only 1043 examples, the effects of adjusting the dataset size by 10% are far less pronounced on the SFU dataset than on the others. Therefore in both dataset rebalancing and general augmentation I vary percentages by 20%, rather than 10%.

Augmentation-Specific Adjustments

Contextual augmentation with BERT has greater time and space complexity than backtranslation and synonym replacement. The augmentation model and dataset must be trained specifically for every setting: each combination of dataset, level of imbalance or percentage of dataset remaining, and crossvalidation fold or random seed. This is because the contextual augmentation model must be trained on the entire training set before it is used to infer replacement words. This is not the case for the other two augmentation methods. Backtranslation of any given example depends only on the example and the chosen translation models; candidate synonym creation depends only on the example and on WordNet. To accommodate the increased time and space complexity of contextual augmentation, experimental results with this technique are averaged across 10, rather than 30, trials.

5.2.6 Validation and Testing Procedure

Parameter tuning is conducted on SST's established development set; final testing is conducted on SST's test set. Subj and SFU do not have any official train/development/test splits, so 10-fold cross validation is used both to tune the parameters and to report the test accuracy.

5.3 Experimental Results

This section describes how the dataset size and degree of imbalance affect two values: the accuracy boost offered by augmentation and the optimal augmentation distance.

5.3.1 Augmentation Performance

This subsection show how augmentation accuracy boosts (or lack thereof) change with respect to dataset size and degree of imbalance. Each of the four combinations of application methods and the classification models is discussed separately.

Dataset Balancing, Classification with RNN

Figure 5.2 shows the results of dataset balancing when the RNN is the classification model. All three augmentation techniques generally outperform undersampling and oversampling with both SST and Subj. There are two exceptions to this: when the minority percentage is 80% or higher and when synonym replacement's performance is roughly equal to that of undersampling. The near convergence of all five approaches is likely because their performance differences are muted by the fact that the rebalancing approach simply affects a smaller percentage of the data. When the minority percentage is 90% only 5% of the total data is either ignored (undersampling), duplicated (oversampling), or augmented; when the minority percentage is 20% the rebalancing technique governs 40% of the total data.

On Subj, contextual augmentation and backtranslation both outperform undersampling and oversampling by nearly 1% when the minority percentage is 50% or less. This accuracy improvement is especially notable considering the accuracy is roughly 90% since it is more difficult to increase the accuracy of highly accurate models.

In order to clearly show the results on the largest of the three datasets, the results of classifying Subj are also given in Table 5.1.

General Augmentation, Classification with RNN

In Figure 5.3 I see that all three augmentation techniques boost performance on the Subj dataset when more than 10% of the dataset is used. Synonym replacement's accuracy improvements are modest, while backtranslation and contextual augmentation offer accuracy improvements of nearly 1.0%.

Imb Pct	SR	BT	CA	Over	Under
10	88.42 ± 1.32	**89.41** ± 1.25	89.28 ± 1.11	87.94 ± 1.47	88.50 ± 1.68
20	89.68 ± 1.34	90.83 ± 1.11	**90.96** ± 1.02	89.25 ± 1.41	89.78 ± 1.02
30	90.41 ± 1.28	**91.56** ± 1.14	91.35 ± 0.97	90.08 ± 1.32	90.45 ± 0.96
40	91.46 ± 0.90	**91.86** ± 0.82	91.79 ± 1.16	90.84 ± 1.03	90.92 ± 1.09
50	91.74 ± 1.05	91.89 ± 0.81	**92.00** ± 0.91	91.33 ± 1.32	91.41 ± 0.90
60	92.01 ± 0.96	91.98 ± 0.81	**92.14** ± 0.76	91.76 ± 1.03	91.69 ± 0.84
70	92.17 ± 0.83	92.17 ± 0.85	**92.23** ± 0.88	92.08 ± 0.89	91.87 ± 0.92
80	92.32 ± 0.87	**92.41** ± 0.83	92.12 ± 0.82	92.26 ± 0.95	92.05 ± 0.94
90	**92.54** ± 0.72	92.18 ± 1.08	91.98 ± 1.03	92.38 ± 0.87	92.25 ± 0.85

Table 5.1: Ten-fold cross validation accuracy of dataset rebalancing on the Subj dataset when the RNN is the classification model. All values are percentages. Imb Pct: percentage of minority label examples remaining in the training set, SR: synonym replacement, BT: backtranslation, CA: contextual augmentation, Over: oversampling, Under: undersampling.

Backtranslation on SST consistently yields accuracy increases of roughly 1.5%. Backtranslation harms the performance in the majority of tested percentages of SFU, although all experiments on SFU have a much larger standard deviation.

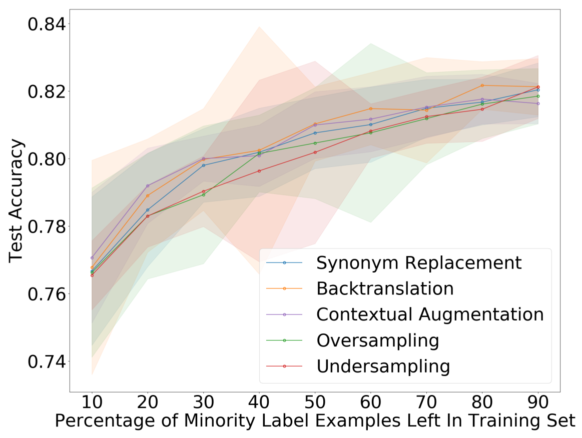
In order to clearly show the results on the largest of the three datasets, the results of classifying Subj are also given in Table 5.2.

Dataset Balancing, Classification with BERT

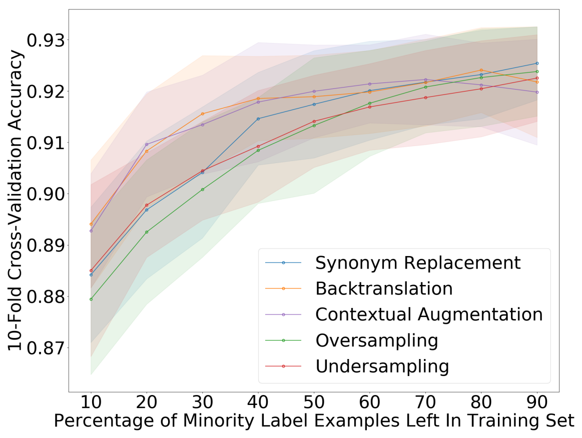
As can be seen in Figure 5.4, all methods of dataset balancing appear to converge as the dataset becomes more balanced. Backtranslation has the best performance of the techniques. Its accuracy tends to be equal to or greater than undersampling. Synonym replacement and contextual augmentation harm performance with few exceptions. Backtranslation's accuracy boosts of nearly 1% in the case of Subj are important considering that the accuracy is already quite high at around 95%.

General Augmentation, Classification with BERT

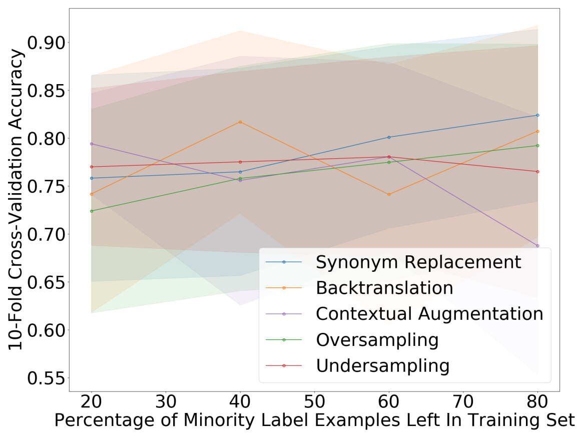
Figure 5.5 appears to show that each augmentation technique is as likely to harm performance as it is to help. Synonym replacement is the best performing, as it consistently provides accuracy improvements of roughly 0.2%.



(a) SST



(b) Subj



(c) SFU

Figure 5.2: Effect of dataset rebalancing techniques on the classification accuracy of the RNN for various levels of imbalance for the SST, Subj, and SFU datasets. Take, for example, when the percentage of minority label examples left in the training set is 40%. In this case a new training set is made by randomly sampling 40% of one label's examples and retaining all of the other label's examples. Each point represents the average of 60 trials (30 for each of the two labels), except the contextual augmentation points, which are averaged across 20 trials (10 for each of the two labels). The shaded area around each line is the standard deviation of the corresponding trials.

Pct	SR	BT	CA	No
10	87.83 ± 2.18	87.95 ± 2.12	86.79 ± 4.27	**88.37** ± 1.16
20	89.81 ± 0.80	89.65 ± 1.23	**90.42** ± 0.98	89.60 ± 1.17
30	90.27 ± 1.18	90.78 ± 1.03	**91.15** ± 1.08	90.38 ± 0.99
40	91.05 ± 0.84	**91.23** ± 1.05	**91.23** ± 0.96	90.77 ± 1.13
50	91.53 ± 1.06	**91.63** ± 1.08	91.57 ± 0.87	91.21 ± 1.07
60	91.90 ± 0.88	**92.09** ± 0.83	91.95 ± 0.84	91.49 ± 1.18
70	91.91 ± 0.83	92.23 ± 0.96	**92.45** ± 0.67	91.89 ± 1.04
80	**92.39** ± 0.84	92.37 ± 0.89	92.38 ± 0.97	92.16 ± 0.93
90	92.46 ± 0.83	**92.60** ± 0.96	92.40 ± 0.69	92.21 ± 1.00
100	92.65 ± 0.82	**92.76** ± 0.77	92.48 ± 1.03	92.38 ± 0.85

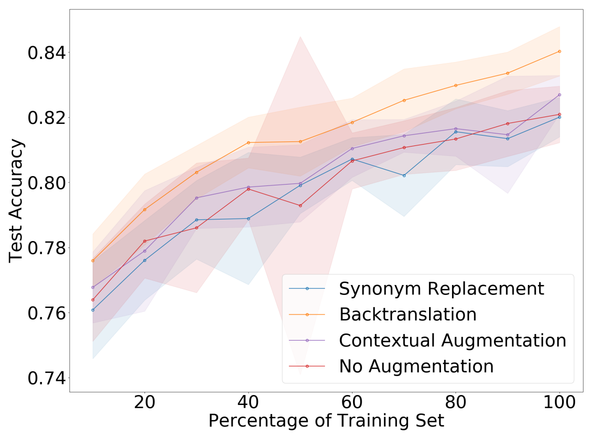
Table 5.2: Ten-fold cross validation accuracy of general augmentation on the Subj dataset when the RNN is the classification model. All values are percentages. Pct: percentage of training examples used, SR: synonym replacement, BT: backtranslation, CA: contextual augmentation, No: no augmentation.

5.3.2 Optimal Augmentation Distance

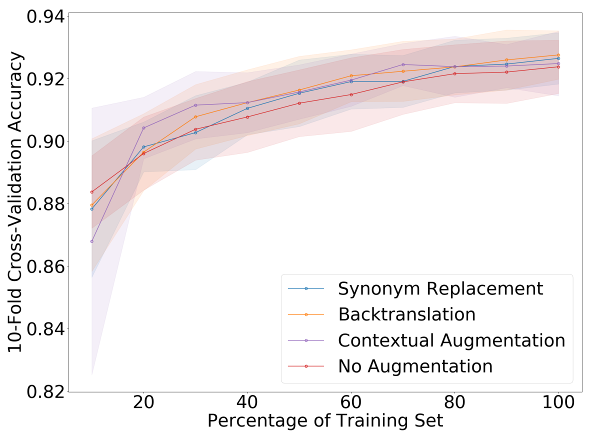
As discussed in Section 4.1, the optimal p values can be used to understand how the optimal augmentation distance changes with respect to the dataset size and degree of imbalance. The RNN is trained for up to 100 epochs while BERT is trained only for three or four. Therefore the impact of varying the augmentation parameter p will be noticeably more consistent and indicative of a general pattern when the RNN is the classification model. For this reason I analyze the optimal p values only for experiments where the text is classified by the RNN.

Augmentation parameters for different augmentation techniques should not be compared with one another: one cannot say that $p = 0.5$ represents the same distance from the original data regardless of the augmentation method. However, I can study and compare the trends in the optimal augmentation distance across the different techniques. Observations involving the optimal augmentation distances should be qualified by the fact that, as discussed in Section 5.2.2, augmentation parameter tuning used subjective human decision making rather than a strict rules-based approach.

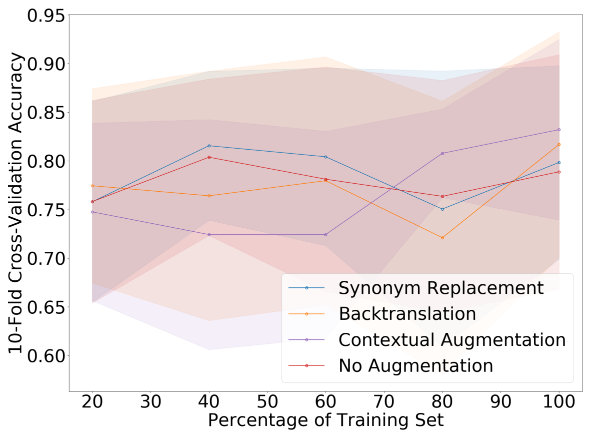
Figures 5.6 and 5.7 show how the optimal distance changes with respect to two variables of interest: degree of imbalance and dataset size.



(a) SST



(b) Subj

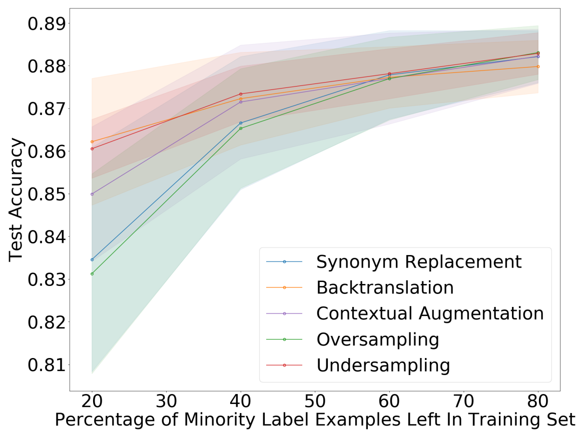


(c) SFU

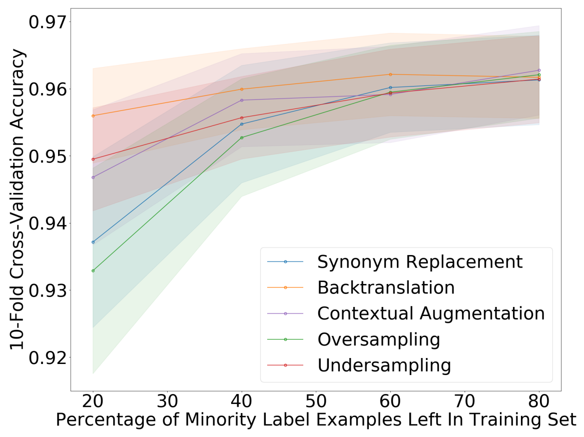
Figure 5.3: Learning curves showing the effect of the augmentation techniques on the classification accuracy of the RNN on the SST, Subj, and SFU datasets. For example, when the percentage of training set is 40%, a new training set is made by randomly sampling 40% of each label's examples. Each point represents the average of 30 trials, except the contextual augmentation points, which are averaged across 10 trials. The shaded area around each line is the standard deviation of the corresponding trials.

Dataset Balancing

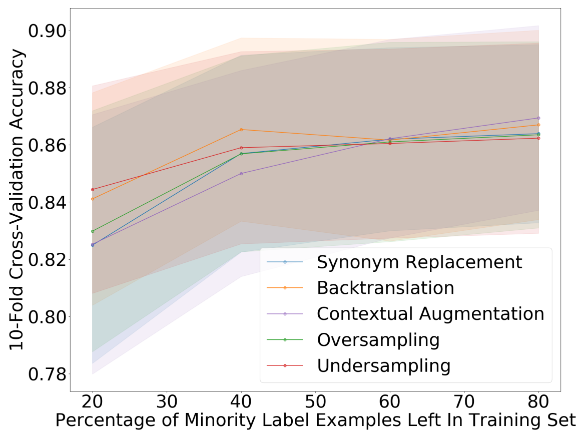
In Figure 5.6 it can be seen that the optimal augmentation distance tends to become larger as the datasets become more imbalanced. This pattern is especially clear in experiments



(a) SST



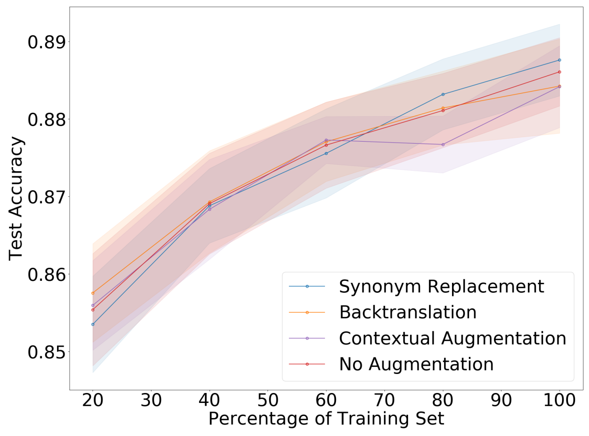
(b) Subj



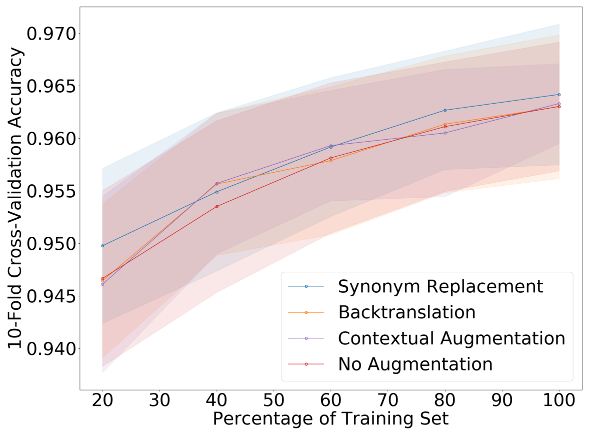
(c) SFU

Figure 5.4: Effect of dataset rebalancing techniques on the classification accuracy of BERT for various levels of imbalance for the SST, Subj, and SFU datasets. Take, for example, when the percentage of minority label examples left in the training set is 40%. In this case a new training set is made by randomly sampling 40% of one label's examples and retaining all of the other label's examples. Each point represents the average of 20 trials (10 for each of the two labels). The shaded area around each line is the standard deviation of the corresponding trials.

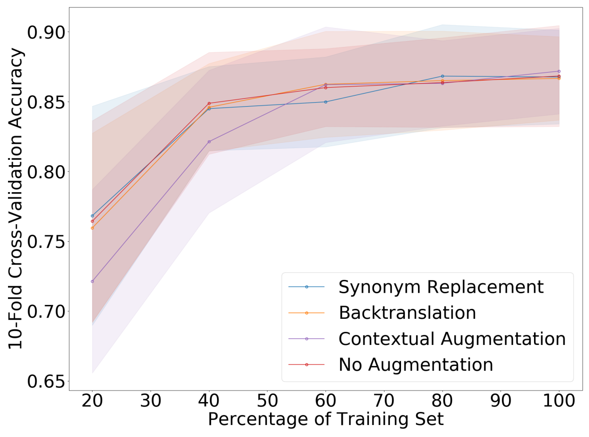
with Subj: the optimal parameter is monotonically nondecreasing as the dataset becomes more balanced.



(a) SST



(b) Subj

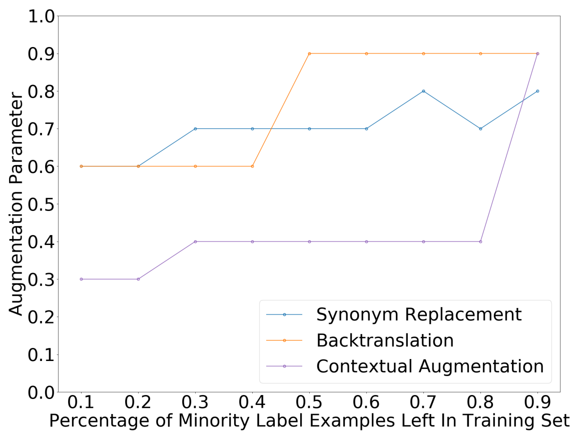


(c) SFU

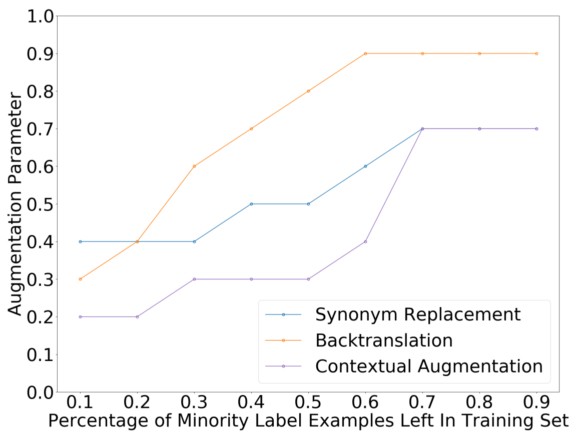
Figure 5.5: Learning curves showing the effect of the augmentation techniques on the classification accuracy of BERT on the SST, Subj, and SFU datasets. For example, when the percentage of training set is 40%, a new training set is made by randomly sampling 40% of each label's examples. Each point represents the average of 10 trials. The shaded area around each line is the standard deviation of the corresponding trials.

General Augmentation

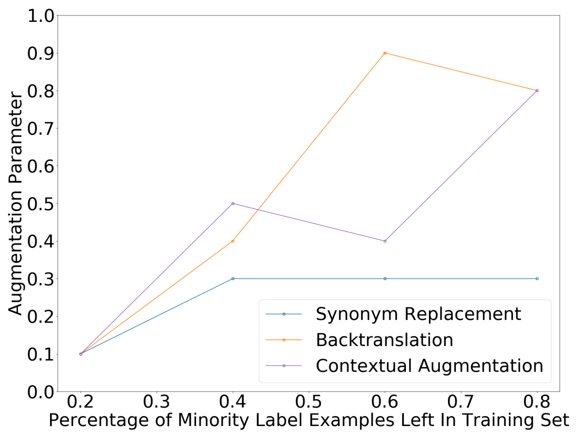
In the case of general augmentation, there does not appear to be a trend in the optimal augmentation distance that holds across datasets and augmentation methods. There is a



(a) SST



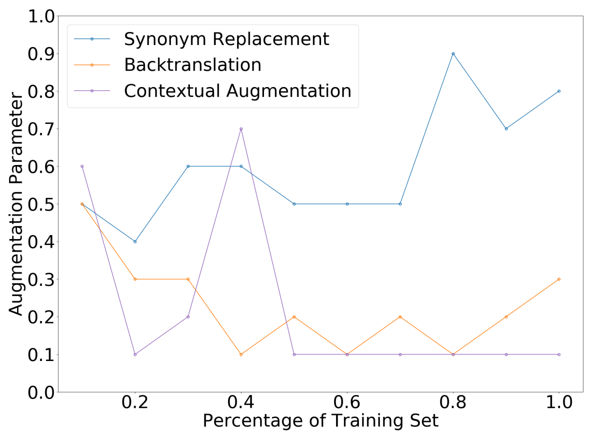
(b) Subj



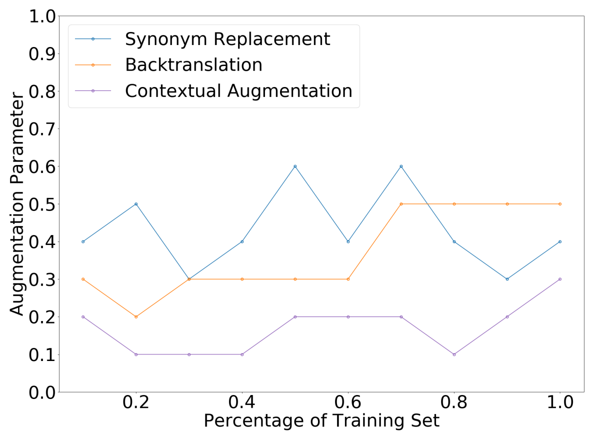
(c) SFU

Figure 5.6: The optimal augmentation parameter as a function of degree of imbalance when the RNN is the classification model. Lower augmentation parameter values correspond to a greater distance from the original data.

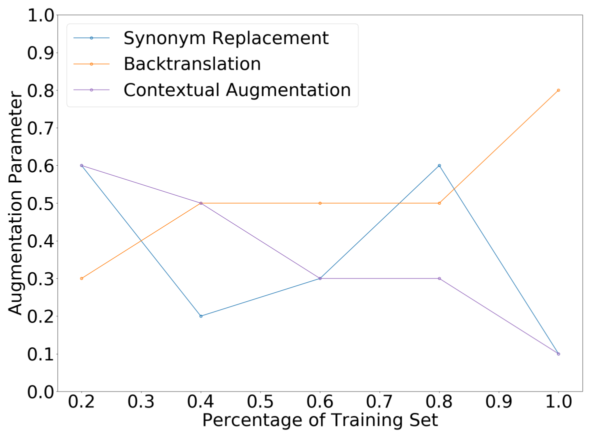
trend in the case of backtranslation with SFU (Figure 5.7c), but the trend is in the opposite direction with backtranslation on SST (Figure 5.7a).



(a) SST



(b) Subj



(c) SFU

Figure 5.7: The optimal augmentation parameter as a function of percent of data used when the RNN is the classification model. Lower augmentation parameter values correspond to a greater distance from the original data.

5.4 Discussion

In this section I highlight several interesting or important findings contained in the experimental results. I discuss possible explanations for various patterns in the results. I also discuss ethical issues that may arise from the use of data augmentation in NLP.

5.4.1 Augmentation for Classification with RNN

All three augmentation methods almost always increase the accuracy of the RNN on the SST and Subj datasets (see Figure 5.2). The performance boost offered by augmentation is especially pronounced when dataset rebalancing: the accuracy with backtranslation and contextual augmentation is up to 1% higher than both undersampling and oversampling. As shown in Figure 5.3, the performance boosts in general augmentation tended to be smaller. However, the results of backtranslation on the SST dataset indicates that general augmentation with the RNN has the potential to offer important accuracy increases.

5.4.2 Augmentation for Classification with BERT

While each of the augmentation techniques had fairly consistent positive performance when used on the RNN classification model, the results for BERT classification are more modest. At most points on the horizontal axes of Figures 5.4 and 5.5 there is an augmentation technique that will boost performance. However, depending on time and resource constraints the performance boosts may be too small to justify the time that it takes to tune the parameters of each of the three techniques.

This relatively low performance could be caused by two attributes of BERT. First, it is already a highly accurate model, and it is generally more difficult to provide accuracy boosts to high accuracy models. Further, comparing Figures 5.3 and 5.5 shows that BERT consistently achieves a roughly 3% greater accuracy than the RNN in each setting. Second, in order to avoid catastrophic forgetting (as described in Section 2.5.2) the authors of BERT recommend limiting the number of training epochs to 3 or 4 [8]. This means that each training example, in either its original or augmented form, is observed on the order of 10 times (the exact number varies — for instance each minority label training example is observed 10 times per epoch when rebalancing with 10% imbalance). Since the RNN is trained for between 25 and 100 epochs, the number of observations of each training examples is an order of magnitude larger. Clearly the effects of augmentation are more muted when the examples are viewed fewer times.

I ran several experiments with a larger number of epochs and a lower learning rate. In this case training with each of the augmentation methods outperformed training without augmentation. However, all of these models were outperformed by BERT trained with no augmentation for 4 epochs. This indicates that the performance gains offered by augmentation over a large number of epochs are counteracted by the performance costs that result from catastrophic forgetting. These results also indicate that if it is possible to avoid catastrophic forgetting while increasing the number of epochs, then BERT may benefit significantly more from augmentation. This idea will be explored in greater detail in Section 6.2.

5.4.3 Synonym Replacement

Synonym replacement's performance varies the most of the three augmentation techniques. In general augmentation with BERT classification it yields 0.3% accuracy increases at all percentages, and in many cases its accuracy roughly equals that of no augmentation.

Synonym replacement is the least expensive of the three augmentation techniques. The POS tagging and WordNet lookup can be performed at training time, but I perform these tasks prior to training and save the candidate synonyms to memory. This approach made training a single epoch approximately twice as fast, yielding considerable time savings. The memory cost of this approach varies depending on the dataset, but in these experiments each of the three augmented datasets use no more than one order of magnitude more memory than the original dataset. If these memory costs are too large for a given task, then candidate synonym generation can be conducted during training.

5.4.4 Backtranslation

Backtranslation appears to offer the most consistent accuracy boosts of the three techniques. It did so with minimal parameter tuning. The augmentation parameter p governed the ratio of augmented to original data, and which of the 25 backtranslated examples were chosen. It did not, however, govern the creation of the backtranslated examples. As discussed in Section 4.2.3, the sampling temperature parameter was set to the same value used in [41]. That backtranslation performs consistently well without tuning this parameter speaks to the power of the technique. Tuning this parameter is likely to yield even better results.

5.4.5 Contextual Augmentation

Contextual augmentation tends to perform best on the largest dataset (Subj) and worst on the smallest (SFU). This is likely because contextual augmentation uses a model that is fit to the training set prior to its use for augmentation. The model will produce more intelligent augmentation candidates as the training set becomes larger. This introduces a dilemma: the augmentation method will create better training examples on larger training sets, but larger training sets generally have a reduced need for data augmentation. Figure 5.3b exemplifies this: contextual augmentation performs poorly for a very small training set (10%), well for a moderately sized training set (20% – 70%), and nearly equals the performance with no augmentation for a large training set (70% – 100%) However, my exploration of this dilemma is limited by the size of the training sets. Without testing on datasets that are an order of magnitude larger (hundreds of thousands of examples), I cannot be entirely confident in my conclusion that contextual augmentation's performance converges to that of no augmentation.

Contextual augmentation's limitations were revealed even in my own methodology. Unlike synonym replacement and backtranslation which rely on a predefined database or pretrained model, this method requires model training and inference. Just as I chose to average contextual augmentation results across 10 (rather than 30) trials, other practitioners will likely have to make time-accuracy tradeoffs when implementing this technique.

5.4.6 Augmentation for Rebalancing

The average accuracy increases achieved by augmentation were greater in dataset rebalancing than in general augmentation. This is likely because imbalanced datasets (specifically their minority class examples) have a greater need for training set diversity. As discussed in Section 2.5.1, undersampling ignores relevant information contained in the available labeled data while oversampling tends to overfit to the minority class. Done well, rebalancing by augmentation allows all majority label examples to be retained while limiting overfitting to the minority class.

The key to limiting overfitting is having a diverse training set, which in theory can be produced by properly tuned augmentation. This goal of maintaining minority class diversity can explain the patterns observed in Figure 5.6. I posit that the optimal level of diversity in the minority set is constant with respect to level of imbalance. It follows that as the minority set decreases in size, each individual example in the minority set must be relied upon to generate more diverse augmentations. This greater contribution of diversity per training example is realized by way of the augmentation distance. When augmentation distance is larger (p is smaller), each training example contributes a greater diversity of augmented examples. This would explain the clear trends observed in Figure 5.6, where greater augmentation distances are optimal for greater degrees of imbalance.

The above analysis explains why a large augmentation distance is helpful when the level of imbalance is high, but does not explain why the optimal augmentation distance decreases (i.e., why the large augmentation distance does not remain helpful) as the training set becomes more balanced. To understand this phenomenon I propose the concept of *augmentation accuracy.* The augmentation accuracy is based on a comparison between a training example before and after its been transformed by an augmentation method. If the post-augmentation training example is more likely to be seen in the test set, I say that the augmentation method has a high augmentation accuracy. If the post-augmentation training example is less likely to be seen in the test set, I say that the augmentation method has a low augmentation accuracy. Note that an augmentation method can be safe (i.e., label-preserving) while having a low augmentation accuracy. For example, suppose an augmentation method introduces label-compatible Old English language into a training example. The transformed example is safe, but the augmentation has low augmentation accuracy since Old English is unlikely to be seen in the test set. Since a machine learning model's performance depends on the similarity of its training and testing set, a consistently low augmentation accuracy would be detrimental to an augmentation method's success.

I suggest that the ELAD of most augmentation methods (including all of the three studied in this thesis) is inversely correlated with their augmentation accuracy. With respect to contextual augmentation, for example, this assumption implies that there is a correlation between the probability that is assigned to a candidate replacement word by BERT and the likelihood of that word, in that context, appearing in the test set. This implication seems reasonable given that the contextual augmentation model is pretrained on a massive dataset, where it learns to predict common words based on their context. Similar arguments can be made with respect to the augmentation accuracy of the other two augmentation methods studied in this thesis. This introduces a tradeoff that varies with respect to the level of imbalance. When the level of imbalance is more extreme, the lesser augmentation accuracy is justified by the need for a greater augmentation distance. As the dataset becomes more balanced, the same low augmentation accuracy would not be justified since the augmentation distance becomes less necessary.

5.4.7 General Augmentation

As discussed in the previous subsection, the accuracy improvement provided by increasing the minority set diversity via augmentation is compounded by the corresponding ability to retain the majority class examples (i.e., avoid undersampling). General augmentation, however, serves only the former function.

The dynamic in dataset rebalancing wherein the optimal augmentation distance becomes smaller as the dataset becomes more balanced does not appear to have an analogue in general augmentation. Figure 5.7 shows no clear relationship between optimal augmentation parameter and percentage of dataset used.

As the training set becomes larger, the accuracy with augmentation does not appear to converge to the accuracy without augmentation. Take for example the largest dataset, Subj. The accuracy with backtranslation, the best performing method when the classification model is the RNN, remains roughly 0.5% greater than that of no augmentation, for all training set percentages greater than 20%. The same can be said with respect to synonym replacement when BERT is the classification model. This finding is somewhat counterintuitive: it would be reasonable to expect the augmentation methods have less to offer since the training set likely becomes more diverse as it becomes larger. Again, this analysis is limited by the size of the training sets; the augmentation methods' accuracy improvements may be closer to 0 for a training set that is an order of magnitude larger.

5.4.8 Comparison to CV Augmentation

The accuracy boosts offered by the augmentation techniques in this thesis are noticeably less impressive than those offered by augmentation in CV. For example, the authors of [7] use augmentation to increase the accuracy of ImageNet from 83.1% to 83.5%. This accuracy boost is considerable not only because the 83.1% was the previous state of the art, but also because ImageNet contains hundreds of classes.

I believe that augmentation is less effective in NLP than it is in CV for reasons are inherent to the respective fields (i.e., the disparity cannot be entirely resolved by more sophisticated NLP augmentation techniques). I suggest that this is for two reasons: (1) the space of possible images that have practically identical content is larger than the space of possible texts that have practically identical content, and (2) this space of images is more easily explored through predefined transformations than is the space of texts.

To illustrate (1), note that many image classification labels are invariant with respect to lighting, occlusion, scale, and horizontal orientation. Text classification labels are usually invariant with respect to the exact word choice and sentence structure, but the number of words that convey the same meaning and the number of ways in which a sentence can be structured are both fairly limited.

To illustrate (2), note that an image in human recognition dataset will have the same label regardless of whether a human in the image is facing left or right, and a model can learn this invariance through a predefined transformation, namely horizontal mirroring. Alternatively, the texts "I greatly enjoyed this movie" and "this movie is one that I greatly enjoyed" are in some sense mirror images of one another that would share the same label in a text classification task. However, reliably transforming the first text to the second or vice versa is a delicate process that likely requires human input.

5.4.9 Ethical Considerations

Biased training sets and biased models are an unfortunately common occurrence in machine learning. For example, the authors of [3] found that gender recognition models often perform worse when classifying darker and more feminine faces.

In theory this bias be overcome by introducing more representative examples into the training set. However, since data augmentation will by definition transform the training set to which it is applied, its potential to introduce biases into the data should be closely scrutinized.

Take, for example, contextual augmentation with BERT. The augmentation model has been trained on Wikipedia articles and BookCorpus [50], both of which tend to contain formal writing, often thoroughly edited. It therefore can be expected to generate less accurate augmentations for less formal English. It is also more likely to suggest words that appeared in its own pretraining set, namely more formal words. This likely explains contextual augmentation's poor performance on the SFU dataset. Newspaper comment sections often contain informal English, and this mismatch between the writing on which the augmentation model is pretrained and the writing to which it is applied likely harms performance. Any language is constantly changing and used in many different ways; BERT augmentation's pretraining set will introduce a bias towards a specific type of writing. In some cases this bias may simply be one that harms performance, but in other cases it could become an ethical issue.

5.5 Summary

In this chapter, I evaluated three NLP data augmentation methods in a variety of settings. I developed an understanding of how augmentation affects classification accuracy, and how the effects change when the size or imbalance of a dataset varies. I conducted a similar analysis with respect to the optimal augmentation distance.

In the next chapter, I summarize the conclusions of the thesis and describe possible directions for future work.

Chapter 6

Conclusion and Future Work

In this chapter, I summarize the contributions of the thesis and give several possible directions for future work.

6.1 Conclusion

This work has shown that synonym replacement, backtranslation, and contextual augmentation can all offer important accuracy increases to already highly accurate models. Backtranslation has the most consistent and strongest performance; synonym replacement is the least expensive (as measured by either time or memory, depending on the implementation); and contextual augmentation tends to have the worst performance and be the most computationally expensive.

With the exception of the smallest dataset, the accuracy of BERT is usually improved by at least one of the augmentation methods; on the same two datasets the accuracy of the RNN is almost always improved by all three augmentation methods. This suggests that the increase in training set diversity offered by data augmentation is less necessary for models that have been pretrained on massive datasets. Still, the results show that properly tuned augmentation can noticeably increase the accuracy of BERT.

When the training set is balanced, there does not appear to be a strong relationship between the training set size and the impact of data augmentation on model accuracy. Experiments with significantly larger training sets may be necessary in order to determine whether such a pattern exists.

Undersampling tends to outperform oversampling, likely because it avoids overfitting to the minority class. However, undersampling has an obvious downside: it ignores relevant information (i.e., labeled data). The diversity that augmentation methods can introduce into the minority class means that minority class overfitting can be mitigated without ignoring labeled data. Therefore the accuracy boost offered by increasing the training set diversity is compounded by the retaining of all majority class examples. As a result, each of the three augmentation methods are usually a powerful alternative to both of the sampling-based techniques of dataset rebalancing. In fact, for the RNN on the two largest datasets, each augmentation technique almost always outperforms both undersampling and oversampling.

The optimal distance from which each augmented example varies from its original form becomes smaller as imbalanced datasets become more balanced. I suggest that this is because of a tradeoff between dataset diversity (as determined by augmentation distance) and augmentation accuracy. The rewards of adding diverse training examples to a minority class are more pronounced, so much so that they justify the corresponding decrease in augmentation accuracy. As the dataset becomes more balanced, the rewards are reduced and therefore the optimal augmentation distance decreases. When a dataset is balanced, however, its size does not appear to be correlated with the optimal augmentation distance.

6.2 Future Work

As discussed in Section 5.4.2, while increasing the number of BERT training epochs allows for more data augmentation, these benefits are counteracted by catastrophic forgetting. Finding a way to increase the number of training epochs without catastrophic forgetting would allow the benefits of augmentation with BERT training to be more fully realized. The authors of [6] developed such a method: they pretrain ULMFit, an LSTM-based model, on an LM pretraining problem and incorporate the pretraining problem into the fine-tuning of the model. They train the model to minimize the joint loss, a weighted sum of the classification loss and the LM loss. The weight of the LM loss is exponentially decaying, which limits the impact of the early training steps of the randomly initialized classification layer on the pretrained model. After light tuning of the exponential decay parameter, their method outperforms standard transfer learning with the same model on all tested datasets. Future work could involve extending this approach to BERT training. It would likely also limit BERT's catastrophic forgetting, in which case it would allow more training epochs and therefore more data augmentation to be used.

Discussed in Section 3.2.4, instance crossover has shown positive results for small datasets. This study was limited to augmentation for models that are ignorant of word order. This method was not tested since I expect it to be outperformed by the three methods tested in this thesis. However, the fact that image mixing, instance crossover's CV analogue, had the promising results outlined in Section 3.1.2, combined with these methods' incredibly low time complexity suggests that it is an area worth exploring in greater detail. Future work could test whether or not this method is helpful for RNNs and BERT, or any other models that are sensitive to context and word order.

I have shown that BERT can benefit from training with the three augmentation techniques. Future work could run similar evaluations of data augmentation techniques as new models are introduced and found to achieve state of the art results on various tasks. For example, XLNet [42] has surpassed the performance of BERT on a range of NLP tasks. Testing these augmentation methods' impact on the accuracy of XLNet and its successors would be a natural direction for future work.

This thesis focused on augmentation for text classification models, but future work could study its impact on a wider variety of NLP tasks. The positive results in text classification would suggest that these methods will be helpful in other NLP tasks, but task-specific considerations must be made. To take question answering as an example, it may be that the question string is so sensitive to noise that augmentation would lead to deteriorations in performance.

As discussed in Section 3.1.6, test time augmentation has had promising results in CV. Future work could study this approach's effectiveness when applied to NLP classification tasks.

This thesis used a single parameter to govern both the augmentation ratio and the ELAD. Future work could separate the two values, and perform parameter tuning on both. The time costs would be considerably larger than those of this study, but it would allow the relationship between the level of imbalance and both the optimal ELAD and the optimal augmentation ratio to be studied.

The patterns observed in the figures in Chapter 5 are illuminating, but they are limited by the size of the training sets used in this thesis. It would be helpful to see how the trends (or lack thereof) change when the datasets are an order of magnitude larger.

References

[1] Mirriam-webster.com, 2016.

[2] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. OReilly Media, Inc., 1st edition, 2009.

[3] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91, 2018.

[4] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[5] Xi C Chen, Adithya Sagar, Justine T Kao, Tony Y Li, Christopher Klein, Stephen Pulman, Ashish Garg, and Jason D Williams. Active learning for domain classification in a commercial spoken personal assistant. *Proc. Interspeech 2019*, pages 1478–1482, 2019.

[6] Alexandra Chronopoulou, Christos Baziotis, and Alexandros Potamianos. An embarrassingly simple approach for transfer learning from pretrained language models. 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 2089–2095, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[7] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, 2019.

[9] LA Gatys, AS Ecker, and M Bethge. A neural algorithm of artistic style. *Nature Communications*, 2015.

[10] Bert F Green Jr, Alice K Wolf, Carol Chomsky, and Kenneth Laughery. Baseball: an automatic question-answerer. In *Papers Presented at the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference*, pages 219–224, 1961.

[11] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[12] Hiroshi Inoue. Data augmentation by pairing samples for images classification. *CoRR*, abs/1801.02929, 2018.

[13] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, pages 694–711. Springer, 2016.

[14] K Sparck Jones. Natural language processing: a historical review. *University of Cambridge*, pages 2–10, 2001.

[15] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017.

[16] Sosuke Kobayashi. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 452–457, 2018.

[17] Varada Kolhatkar, Hanhan Wu, Luca Cavasso, Emilie Francis, Kavan Shukla, and Maite Taboada. The sfu opinion and comments corpus: A corpus for the analysis of online news comments. *Corpus Pragmatics*, pages 1–36, 2019.

[18] Felix Last, Georgios Douzas, and Fernando Bacao. Oversampling for imbalanced learning based on k-means and smote. *arXiv preprint arXiv:1711.00837*, 2017.

[19] Franco M Luque. Atalaya at tass 2019: Data augmentation and robust embeddings for sentiment analysis. *arXiv preprint arXiv:1909.11241*, 2019.

[20] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[21] Agnieszka Mikołajczyk and Michał Grochowski. Data augmentation for improving deep learning in image classification problem. In *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, pages 117–122. IEEE, 2018.

[22] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[23] Bharat R Naiknaware and Seema S Kawathekar. Prediction of 2019 indian election using sentiment analysis. In *2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC) I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC), 2018 2nd International Conference on*, pages 660–665. IEEE, 2018.

[24] Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. Facebook fairs wmt19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 314–319, 2019.

[25] Christopher Olah. Understanding lstm networks, Aug 2015.

[26] Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics, 2004.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[28] Fábio Perez, Cristina Vasconcelos, Sandra Avila, and Eduardo Valle. Data augmentation for skin lesion analysis. In *OR 2.0 Context-Aware Operating Theaters, Computer*

Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis, pages 303–311. Springer, 2018.

[29] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[30] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[31] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, Jul 2019.

[32] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew 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.

[33] C Summers and MJ Dinneen. Improved mixed-example data augmentation. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1262–1270, 2019.

[34] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer, 2019.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[36] Prashanth Vijayaraghavan, Ivan Sysoev, Soroush Vosoughi, and Deb Roy. Deepstance at semeval-2016 task 6: Detecting stance in tweets using character and word-level cnns. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 413–419, 2016.

[37] Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R. Thomas McCoy, Roma Patel, Najoung Kim, Ian Tenney, Yinghui Huang, Katherin Yu, Shuning Jin, Berlin Chen, Benjamin Van Durme, Edouard Grave, Ellie Pavlick, and Samuel R. Bowman. Can you tell me how to get past sesame street? sentence-level pretraining beyond language modeling. In *Proceedings of the 57th Annual Meeting of the Association for*

Computational Linguistics, pages 4465–4476, Florence, Italy, July 2019. Association for Computational Linguistics.

[38] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6383–6389, 2019.

[39] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, pages arXiv–1910, 2019.

[40] Xing Wu, Shangwen Lv, Liangjun Zang, Jizhong Han, and Songlin Hu. Conditional bert contextual augmentation. In João M. F. Rodrigues, Pedro J. S. Cardoso, Jânio Monteiro, Roberto Lam, Valeria V. Krzhizhanovskaya, Michael H. Lees, Jack J. Dongarra, and Peter M.A. Sloot, editors, *Computational Science – ICCS 2019*, pages 84–95, Cham, 2019. Springer International Publishing.

[41] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2019.

[42] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, pages 5754–5764, 2019.

[43] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.

[44] Kelly Zhang and Samuel Bowman. Language modeling teaches you more than translation does: Lessons learned through auxiliary syntactic task analysis. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 359–361, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[45] Xiang Zhang and Yann LeCun. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*, 2015.

[46] Han Zhao, Zhengdong Lu, and Pascal Poupart. Self-adaptive hierarchical sentence model. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[47] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, pages 13001–13008, 2020.

[48] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[49] Xinyue Zhu, Yifan Liu, Jiahong Li, Tao Wan, and Zengchang Qin. Emotion classification with data augmentation using generative adversarial networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 349–360. Springer, 2018.

[50] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27. IEEE, 2015.