THESIS

REDUNDANT COMPLEXITY IN DEEP LEARNING: AN EFFICACY ANALYSIS OF NEXTVLAD IN NLP

Submitted by

Sina Mahdipour Saravani

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Summer 2022

Master's Committee:

Advisor: Indrakshi Ray
Co-Advisor: Ritwik Banerjee

Steven Simske

ABSTRACT

REDUNDANT COMPLEXITY IN DEEP LEARNING: AN EFFICACY ANALYSIS OF NEXTVLAD IN NLP

While deep learning is prevalent and successful, partly due to its extensive expressive power with less human intervention, it may inherently promote a naive and negatively simplistic employment, giving rise to problems in sustainability, reproducibility, and design. Larger, more compute-intensive models entail costs in these areas. In this thesis, we probe the effect of a neural component – specifically, an architecture called NeXtVLAD – on predictive accuracy for two downstream natural language processing tasks – context-dependent sarcasm detection and deepfake text detection, and find it ineffective and redundant. We specifically investigate the extent to which this novel architecture contributes to the results, and find that it does not provide statistically significant benefits. This is only one of the several directions in efficiency-aware research in deep learning, but is especially important due to introducing an aspect of interpretability that targets design and efficiency, ergo, promotes studying architectures and topologies in deep learning to both ablate the redundant components for enhancement in sustainability, and to earn further insights into the information flow in deep neural architectures, and into the role of each and every component. We hope our insights highlighting the lack of benefits from introducing a resource-intensive component will aid future research to distill the effective elements from long and complex pipelines, thereby providing a boost to the wider research community.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisors, Dr. Indrakshi Ray and Dr. Ritwik Banerjee, for their help, guidance, and support during the completion of this thesis. I would like to thank my committee, as well, for their constructive feedback.

My journey would have been impossible without the love and support from my family, and I would like to thank them for everything they did for me. I also would like to thank my friends for their support. Last but not least, I want to thank the Computer Science Department, its faculty, and staff for all their help and support during my time at Colorado State University, and for providing the resources for my research and studies.

DEDICATION

I would like to dedicate this thesis to my mother, who does not hesitate to sacrifice her comfort for my good.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Insights on the importance of scientific design and its impacts is sometimes overlooked in the current deep learning era. Enriching Deep Learning with theories and intuitions would be a key to its further development and real-world application. Studies show that the reliability and reproducibility of scientific research work and their findings has decreased with the popularity of deep learning [1, 2, 3]. It is pivotal and definitely beneficial to make the contributions clear and attribute the success or failure of a system to the correct component. Also, it is valuable to make the effort to include scientific reasoning and justification for components of a deep learning pipeline. Publication of negative results and the findings of research trials and experiments that did not necessarily yield to benchmark improvements but would save the time, efforts, and resources of others, is significantly important in mitigating such concerns. On the other hand, increased interpretability would flourish controllability and trustworthiness, in addition to enabling the incorporation of new, insightful, and scientifically solid ideas.

Efficiency and redundancy of deep learning has been studied [4, 5, 6, 7, 8, 9, 10, 11]; yet, many of these studies focus on compression techniques or other innovative techniques like vector quantization or weight pruning that keep up with the state of the art accuracy with a smaller model size. However, sometimes questioning the components in the proposed or commonly used models or pipelines reveals interesting findings about their efficiency. Due to a general lack of interest in negative results, and the difficulty in publishing such insights and findings, only few researchers present the negative findings of their work [1, 12, 13] and many may decide not to devote time to studying redundant complexity in the form of redundant components of the models, and to distilling effective components from them.

Astonishing accuracy improvements achieved by applying large deep learning models in various domains [14, 15, 16, 17, 18, 19] have become possible by the advances in computational hardware technologies that support the huge amount of computation that they demand. However,

there are important cost factors that should be considered by both the research community and industry players. Despite the success of huge deep learning models, *e.g.*, Transformers [20], and even the accomplishments in technologies and further availability of hardware, the environmental and financial aspects of these computational costs are indeed important to reflect on.

The observed correlation between the increased complexity of models and the increased accuracy, sometimes, encourages trivial enlarging of deep learning models and hoping for the better accuracy. This makes it easy to overlook the associated environmental and financial costs. Financial costs appear in the form of high cost of acquiring the advanced processing hardware, and its resulting limitation of fair accessibility of deep learning. The environmental costs are in the form of energy consumption, carbon emissions, and global warming due to the immense computational demands of these big models. This consumed energy is mainly supplied from non-renewable resources [21].

In an effort to bring these concerns to the broader attention and promote scientific design, we study the application of a neural pooling component called NeXtVLAD [22], which has been employed by others and claimed to improve the accuracy [23], to Natural Language Processing (NLP) across two tasks. We perform a comprehensive set of experiments and even try to provide a helpful feature modification to its input to make the component efficacious, but it proves ineffective in both tasks. We overview both sustainability and interpretability – by *interpretability*, we refer to analytical view to deep learning components and architectures, and use it as an umbrella term that covers this new definition – concerns in creating redundant complexity in deep learning, and encourage researchers to reflect on it and take actions to prevent it.

1.1 Motivational Factors

Deep learning has achieved impressive accuracy on many tasks and oveparameterization [24, 25] in neural networks has proven empirically successful. However, there are subsequent costs and concerns that have motivated our research.

1.1.1 Reproducibility

The concerns about reliability, and thus, about reproducibility, are particularly acute in deep learning. For instance, Reimers and Gurevych [26] demonstrated that the hyperparameter settings have a significant impact on the final results obtained by a model. Crane [1] further showed that other confounding factors such as variation of GPUs, the exact version of a framework, the randomness of a seed value provided to a learning algorithm, and the interaction between multiple such factors, can all impact the obtained results.

Unfortunately, with increasing popularity of deep learning, the reliability of findings in publications that extensively employ deep learning can be expected, in general, to decrease [2]. A relatively high focus in experimental findings and benchmark improvements in Natural Language Processing (NLP) makes it eminently prone to such concerns, and amplifies their importance. In light of this seminal empirical observation and the general difficulty of natural language processing, and especially figurative language processing – our first downstream probing tasks in Chapter 4, it is reasonable to not expect outright success on a benchmark corpus simply based on the use of a neural component. Providing reproducibility details in research publications would help preventing redundant complexity, by channeling others' efforts in the correct directions.

1.1.2 Interpretability for Intuitive Design and Efficiency

Beyond reproducibility, however, lies another pertinent factor: the use of increasingly complex pipelines where multiple sophisticated components are glued together for an important downstream NLP task. In such scenarios, it is not always clear *which* components within the complex system may be responsible for improved outcomes. A simple change in data preprocessing may lead to a significant difference in the final result, for example [27, 28]. In publications that introduce complex NLP pipelines, however, such details have sometimes been omitted.

Interpretability and efficiency are not naturally in tension. William of Ockham's principle of parsimony (Occam's Razor) states *"plurality should not be posited without necessity"* [29].

This is the intuitive rule in practicing interpretability in statistical learning – for the purpose of interpretability, a smaller subset of predictors with the strongest effect is preferred [30].

Interpretability and explainability are critical for AI's application to some domains like healthcare; but they are not *only* useful in making deep learning trustworthy and controllable. Interpretability can create opportunities for further novel improvements in deep learning. Moving away from a non-interpretable, black-box view to deep learning architectures, in addition to improvements in interaction, control, and trustworthiness, enables the use of mathematical tools in a more intuitive and insightful way which can yield to both scoreboard improvements and societally-aware practices. Interpretability-aware perspectives would promote scientific – or at least intuitive – design of the architectures, which inherently avoids redundancy.

1.1.3 Sustainability

Deep learning models are getting larger and larger and they require more time to train. Even using the most advanced available hardware that is not even accessible to many researchers, training huge models like the state-of-the-art language models takes time in the order of couple of weeks and more. Environmental costs and effects of deep learning are increasing and they may become a significant contributor to climate change [31]. The carbon emission of training a big model with hyperparameter search, for example, is much more that a car's in its lifetime [32]. The state-of-the-art deep architectures like Transformers no longer fit into the GPU memory of personal computers. Having access to specialized hardware (such as GPU and TPU) is becoming a privilege. Even many of the graduate students at universities are depending on the limited free cloud resources like Google Colab for their research. Researchers state that the current trends in model size expansion and increasing computational demands would make deep learning environmentally, technically, and financially unsustainable [33, 32, 21]. Rise of workshops like SustaiNLP [34] is an obvious indicator of the importance of the matter in the community.

1.2 Problem Statement and Contributions

The aforementioned motivational factors inspire us to employ two NLP problems in a efficacy analysis to further drive researchers' attention towards the redundant complexity in deep learning. Our experimental objective is to examine a novel use of locally aggregated descriptors – specifically, an architecture called NeXtVLAD [22] – and its effect on the performance of NLP classifiers that utilize the state-of-the-art contextual embeddings from Transformers.

We probe the incorporation of NeXtVLAD, motivated by its accomplishments in computer vision, achieve tremendous success in the FigLang2020 sarcasm detection task [35]. The reported F_1 score of 93.1% is 14% higher than the next best result. We specifically investigate the extent to which the novel architecture is responsible for this boost, and find that it does not provide statistically significant benefits. We also analyze this component's effect in another NLP task – deepfake text classification task – and probe the performance of a collection of deep learning classifiers with and without the aforementioned NeXtVLAD neural component in these two downstream tasks.

Our first probing task, context-dependant sarcasm detection, is the same task as in the original study that applied NeXtVLAD to NLP and suggested its effectiveness [23]. Probing for this task, in addition to providing the opportunity to compare our models with the ones from that research, is specifically interesting as the cross-locality subtraction operator could intuitively fit the sarcasm detection application, where contradictions and oppositions play an essential role in creating sarcasm. Furthermore, this is a relatively difficult task in NLP, due to the nature of figurative language. Our second task is a short-text classification task to discriminate human-written and machine-generated text. Our experiments for this task improve the predictive accuracy, but that is again not due to the incorporation of the NeXtVLAD component, and our ablation study shows its superfluity. Although these are not baseline NLP tasks, we believe they provide a good test bed for our objective. We think they are important tasks, one from a difficulty viewpoint and the other from a motivational angle. Sarcasm detection is important because of its difficulty, which is partly due to the inherent requirement of further knowledge beyond the present text, like commonsense, which makes it a great candidate for becoming a performance measure in future NLP baselines.

The deepfake text detection however, is important from the urgent, time-sensitive angle of misinformation, disinformation, and fake news in the current era of social media and their tremendous effect on various aspects of our lives.

Based on our downstream NLP experiments, even in the case of context-dependent sarcasm detection where we hoped to benefit from the cross-locality subtraction operator in NeXtVLAD, we found the neural component to be ineffective. We encourage researchers to take intuitions in design more seriously, and distill complex NLP pipelines. Simpler models are more explainable and have less environmental impacts. Deep learning approaches are expensive, and we hope our insights highlighting the lack of benefits from introducing a resource-intensive component will aid future research to distill the effective elements from long and complex pipelines, thereby providing a boost to the wider research community.

1.2.1 Publications

The work in this thesis has also been published in the following papers. The first publication mainly focuses on the same objective as of this thesis, and the second one reports the experimental results on a NLP task and focuses on the improved accuracy compared to previously tested models.

- ◇ Sina Mahdipour Saravani, Ritwik Banerjee, and Indrakshi Ray. 2021. An Investigation into the Contribution of Locally Aggregated Descriptors to Figurative Language Identification. In *Proceedings of the EMNLP Workshop on Insights from Negative Results in NLP*. Association for Computational Linguistics.

- ◇ Sina Mahdipour Saravani, Indrajit Ray, and Indrakshi Ray. 2021. Automated Identification of Social Media Bots using Deepfake Text Detection. In *Proceedings of the International Conference on Information Systems Security (ICISS)*. Springer.

Chapter 2

Related Work

To limit the scope of our related work section, we only include research publications that either profoundly relate to our objectives and motivations, or have studied redundancy reduction in deep learning. Further related work for each of our NLP tasks are discussed in their respective sections.

2.1 Sustainability

Strubell et al. [32] conduct an interesting study on quantifying approximate costs of training NLP models and their findings are shocking. They estimate the CO_2 emissions from training a big Transformer with neural architecture search to be five times the emissions from a car in its lifetime. Based on their findings, they recommend i) reporting training time and models' sensitivity to hyperparameters, ii) provision of a government-funded academic compute cloud for equitable access to resources, and iii) prioritization of computationally efficient hardware and algorithms. Our work in this thesis encourages and supports this latest recommendation by experimental findings. They follow up their work with another report in [21]. Based on their study, while the top three cloud compute providers, Amazon AWS, Google, and Microsoft respectively use 50%, 100%, and 50% of their consumed energy from renewable sources, the renewable energy consumption in United States is only at 17% [32]. Another recommendation to researchers and practitioners that is augmented to the three previous ones is to be mindful of the used energy sources and report such information to the degree possible. The online tool[1] provided by Lacoste et al. [36] is a helpful resource to follow such initiatives on reporting carbon footprint.

Predictions show that with the continuation of current trend lines in the increase of computational demands in deep learning, advancements are becoming environmentally, economically, and technically unsustainable [33]. They suggest that such growth in the computational requirements

[1] https://mlco2.github.io/impact/

would finally induce a burden on deep learning's application. They provide a tangible theoretical analysis using statistical learning theory: The benefits of overparameterizing – having more parameters than the number of training data points – has been proven [24], resulting in the number of parameters to grow with the number of training data points. The cost of training grows with the product of the number of parameters and the number of training data points. However, based on the statistical learning theory, the number of data points must increase quadratically to get improvements on the performance. This implies that training an overparameterized model with the goal of introducing improvements would at least require computations in the order of the fourth power of the performance improvements [33]. This is theoretically restrictive. For a review of these computational requirements in practice, we refer the reader to [33]. They include an interesting estimation of the required computational power, carbon emission, and economic cost for hitting target error rates (better than the current state-of-the-art) on benchmark datasets in computer vision and natural language processing. Even their most optimistic extrapolation of current trends in computational power is much worse than the theoretical lower-bound, making it critical to put efforts in designing efficient solutions.

In the same line of work with [36], Anthony et al. [31] present another tool, called Carbontracker, for reporting energy consumption and carbon footprint of training deep learning models. Their tool supports a variety of environments and platforms for easy use, and is capable of stopping the training process if the predicted environmental cost is surpassed. Their recommendations for reducing carbon emissions include training in low carbon intensity regions, training in low carbon intensity times of the day, using efficient algorithms, and choosing efficient hardware.

2.2 Redundancy, Compression, and Interpretability

Increased complexity also adversely affects interpretability. While the usual measure of success or improvement in machine learning is the predictive accuracy, many applications dictate the importance of interpretability [37]. Bratko [37] state that interpretability criterion has been over-

looked and that is probably due to the lack of qualitative assessment measures of interpretability, compared to the easily understood measures of predictive accuracy.

On the topic of efficiency and redundancy in deep learning, Stock [4] discusses equivalence classes of neural networks and their compression. They propose a variant of stochastic gradient decent (SGD) – called Equi-Normalization (ENorm) – that inserts additional steps in-between normal SGD iterations to select a representant of the functional equivalence class to minimize a specific energy function. They also propose a novel technique called Iterative Product Quantization (iPQ) to significantly compress neural networks by vector quantization almost without any accuracy loss. They further put their methods to test by applying them to a real-world application in ultra-low bandwidth video chat using a GAN [38] for reproduction of the video at the receiver's end [5].

Multiple prior work address the problem of compressing neural networks. Pruning, quantization, knowledge distillation, selective attention, and low-rank factorization are common techniques for model compression. Motivated by applications in embedded systems where both memory and computational requirements of deep learning are restrictive, Han et al. [6] develop a three-stage pipeline to i) prune unimportant neural connections, ii) quantize connection weights, and iii) encode quantized weights using Huffman code [39]. They reduce the network size by a factor of 35 to 49 without any loss of accuracy (AlexNet and VGG-16 on ImageNet). They also achieve a 3 to 4 times layer-wise speedup and a 3 to 7 times better energy efficiency. Model distillation and compression in Hinton et al. [40] and Buciluǎ et al. [41] aim to reproduce the accuracy of an ensemble of models with a single model. The ensemble models are used to label new unlabeled data and the resulting labeled data is used to train a single model that almost reproduces the initial accuracy. This idea is further developed to the Teacher Student model where the soft labels from a large teacher model is used to train a smaller, simpler student model that replicates the teacher. Interesting modifications have been applied to this idea that even change the objective from compression to enhancing accuracy [42, 43, 44]. Abbasi et al. [45] provide a good overview of this. Other examples of model compression and redundancy reduction in deep learning include using

agglomerative clustering to produce unique receptive fields that do not extract redundant similar features [7]; using few neural network weight values to predict the rest of the weights [11]; using the cosine distance between the filters in the feature space, and regularization and adaptive feature dropout to remove the features with little or no variations from others, and hence reducing the network size [8, 9]; and deactivating network connections in convolutional layers [10].

Examining Transformers in the computer vision domain, Pan et al. [46] find redundant computations on uncorrelated input tokens and develop a framework to gradually and dynamically drop those computations. They achieve notable reduction in computations – delivering up to $1.4\times$ speed-up with less than 0.7% accuracy loss. Transformers' computational costs extremely exacerbates with the length of input sequence – computation is in the order of input sequence length to the power of four. Hence, reduction in input length would be highly influential on the computational demand. Pan et al. [46] leverage this fact and create a dynamic inference policy network [47, 48] to avoid computations on uninformative input tokens. Such a technique is inspired by the human perception that attends to the input signal with dynamic levels of scrutiny, on a per-input basis (selective attention). Pan et al. [46] seek to achieve both efficiency and interpretability, while saving the flexibility and predictive power of the model. The learned input token selection provides inherent interpretability to their approach, which outperforms other interpretation methods.

Li et al. [49] is another interesting work that investigates model compression with a special focus on interpretability. They leverage the relationship between the input vectors and the $2D$ kernels in CNNs to propose a kernel sparsity and entropy (KSE) indicator for measuring the significance of input features to guide model compression. Basically, they use interpretability to find redundancy in deep learning – another motivation for us to study redundancy and interpretability in parallel.

The huge Transformer architecture is interesting to be studied for interpretability and redundancy. Using matrix decomposition, Brunner et al. [50] have proposed *effective attention* that is the part of the attention matrix in Transformers that actually affects the outputs. Further studies in this line of work would also yield to interesting findings on making more efficient deep learning architectures.

Chapter 3

Studied Neural Architecture and its Prerequisites

In this section, we introduce the necessary neural components and explain their role in the neural architecture that is the basis of our study in this thesis. We also review topics and concepts that are helpful for easier understanding of our work.

3.1 Deep Learning Methodology for Text Classification

The general deep learning approach in text classification follows a set of steps given below.

[Step 1: Preprocessing] The goal of preprocessing is to modify the text to have a form that can be better processed and analyzed by the following NLP components. Usually, preprocessing involves tokenization, lemmatization, stopwords removal, token normalization (lowercasing all tokens for example), and noise removal (removing unknown characters for example) on the text. The helpful preprocessing steps may vary from task to task and they are chosen based on the type of the subsequent NLP components.

[Step 2: Language Representation] Probably the most important component of a NLP pipeline is the language representation model. This component creates numerical representations for the input text, usually by mapping text to a vector space. This makes it possible to apply deep learning algorithms to text data. The simplest form of this component is the one-hot word vectorization where each word is represented by a vector of the same size as the number of words in the vocabulary which has a single *one* value at the corresponding dictionary index of the word and *zeros* in all other positions. Nowadays, however, deep neural networks are utilized to learn vector representations for words and subwords of a language using the masked token prediction and next sentence prediction objectives on the huge amount of available unlabeled text. Word2vec [51], GloVe [52], FastText [53], and BERT [54] are among these models.

[Step 3: Feature Extraction and Learning] This step often applies a neural network architecture to the vector representations from Step 2. The vector representations of words or sentences are fine-tuned and transformed by a sequence of network layers to build features that are helpful for the target classification task. Some of the common neural layers that are used for this step are recurrent neural networks (RNNs), long short-term memorys (LSTMs), and convolutional neural networks (CNNs). Note that, recently and specially after the introduction of BERT, feature extraction and learning layers are often no longer explicitly separable from the language representation layer. Steps 2 and 3 together create and update the vector representations to include richer and more useful linguistic and statistical information to learn and understand the language better.

[Step 4: Pooling] Objective of the pooling layers is to summarize the important information from the huge number of features that previous layers produce into a compact form and also to remove unneeded variance in the feature space. Simple fully-connected layers, and maximum and average pooling layers are the most common choices.

[Step 5: Classification] Finally, a fully-connected network layer, usually with a softmax activation function, is used to produce probability scores for different classes and make the predictions based on the pooled features.

3.2 Language Models

Language models, in general, are models that predict the next token or word in a sequence, and wherefore can be used to generate text. However, the term now may occasionally be used to refer to broader applications for representing text. Masked Language Modeling (MLM) refers to the case where the objective is to predict randomly masked tokens in text. Many of the most recent language models are trained (pre-trained) primarily using the MLM objective to learn the statistics of text data, hoping to implicitly learn the language as well.

3.2.1 Non-contextualized Embeddings

Dense vector representations for words flourished with the introduction of Word2Vec [51]. The backbone philosophy is *"you shall know a word by the company it keeps"* [55]. Using the huge volume of available digital text, co-occurrence statistics of words can be learned. In a semi-supervised machine learning setting, either context words are used to predict a target word, or the target word is used to predict the context words. Maximum likelihood estimation gives the optimal weights of the model – a simple neural network – to maximize the probability of the huge corpus.

Variations of Word2Vec and similar word embedding models dominated other text representations (vectorization) methods from 2013 to 2017 due to the resultant higher accuracy across multiple baseline tasks and their interesting characteristics in terms of proximity and general placement of the embedding vectors in the vector space. However, their biggest drawback is their non-contextual nature where the embedding of a token is constant and generated offline. While providing simplicity and ease in use and storage, the linguistic limitations hurt the performance on more complex tasks and datasets. We include two such models in our experiments in Section 4 (*LSTM on GloVe* and *FastText's Supervised Classifier* [56] in Table 4.9).

3.2.2 Contextualized Embeddings and the Transformer

An important deficiency of constant word embedding models like Word2Vec is their global representation for words without consideration of their context. The prevalent example is the word "bank" which has the same representation in both of the following sentences, while having totally different semantics:

- *She went to the bank to open a checking account.*
- *The erosion of the river bank may cause damages to the underground utilities.*

Attention and the Transformer

The term *attention* with its current meaning in the NLP community appeared in [57] where authors implemented a computational mechanism in an encoder-decoder architecture for machine

translation to enable the decoder to decide which parts of the source to pay attention to. The attention mechanism can be described as a mapping from a query and a set of key-value pairs to an output. Fully-connected layers generate the query, key, and value vectors and a compatibility function between the query and the key produces the weight of the corresponding value. The output is the weighted sum of the values [20]. Transformer [20] is an encoder-decoder neural sequence transduction architecture based on this mechanism. Both the encoder and decoder in Transformer are comprised of stacked self-attention and point-wise fully-connected layers. This entirely attention-based architecture outperforms the previous state-of-the-art, establishing the title of the published paper, "Attention Is All You Need".

BERT

Bidirectional Encoder Representations from Transformers (BERT) is a language representation or embedding model that uses the encoder module of the Transformer to learn a bidirectional representation from the left and right contexts of each token in text. BERT has improved the state-of-the-art performance on eleven NLP tasks [54]. BERT's N-dimensional representation vectors are generated dynamically based on the attention score mechanism [57, 20] which relates the effect of each token to all other tokens and to the task objective. It also creates an encyclopedic representation for the whole input text – the [CLS] (classification) token that is often used as the input to a classification algorithm. However, in the studied architecture and some of the presented experiments in this work, the representations for all and every token is consumed by the subsequent layers. Generating the representations dynamically and based on the input sentence makes BERT's embeddings to be contextualized, and hence, to be different for the two instances of *bank* in the provided example sentences.

The Transformer-based encoder models, like BERT, have been investigated and tweaked in recent years and have been proposed in various configurations and sizes. The parameters of these models are learnt in a pre-training phase using the next sentence prediction and masked language modeling objectives on huge collections of unlabeled data. Although such training has resulted in powerful general language models, as language and text form differs from domain to domain,

they can still be fine-tuned for specific domains or datasets to reach even greater performance. We also conduct experiments with such domain-specific pre-trained Transformer encoder models – including a $\text{BERT}_{\text{Large}}$ model pre-trained on COVID-19-related tweets [58], called CTBERT. The expectation of gaining accuracy improvements by using this domain-specific model is met by discerning our experimental results. Other variations of the Transformer encoder models that we have included in our experiments are XLNet [59], RoBERTa [60], and BERTweet [61].

GROVER

GROVER [62] has the same architecture as GPT2 [63], which is a slightly modified version of the Transformer's decoder [20]. It is a state-of-the-art fake news generation model. However, Zellers et al. [62] also adapt a modified version of their model for discrimination tasks. The major difference of GROVER with GPT2 is in its data structure, where it is tailored for news articles. Each data point contains *domain*, *date*, *authors*, and *headline* metadata fields in addition to the *body* of the news. We conduct some experiments in Chapter 4 to assess the capabilities of GROVER's discriminator on detecting deepfake text, and compare it with other models.

3.3 BiLSTM in the Studied Architecture

The BiLSTM neural architecture is commonly used to capture temporal relations (relations showing the sequential position of the token with respect to other tokens) in the sentence. It is the bi-directional version of LSTM that is from the family of recurrent neural networks. These models had dominated other methods in modelling natural language prior to the development of Transformers. Since a BiLSTM layer has been used by Lee et al. [23], we include it in our experiments in an effort to reproduce their results. While investigating the efficacy of components other than NeXtVLAD is not in the scope of this work, our experiments indicate that this BiLSTM layer is also ineffective, similar to the NeXtVLAD component. Note that BiLSTM here, and in the work of Lee et al. [23], is not used for encoding the whole sequence of tokens into a single representation; instead, it is applied to capture the temporal features and incorporate them to update and fine-tune the vector representation of *each* token. BiLSTM updates the representation of each token based

on its previous tokens in both the left-to-right and right-to-left directions, hence the representation of the last token of the sentence in both directions (the first and the last token) are often pooled and used as the representation of the whole sentence (sentence embedding). However, this is not the case in the studied architecture and instead, all of the tokens are consumed by the subsequent layers. The inherent bias toward tokens that appear at two ends of a sentence in BiLSTM may be another possible intuition of Lee et al. [23] for incorporation of NeXtVLAD in its current position – after the BiLSTM layer – in the studied architecture (Section 3.4.4) in order to mitigate this bias by leveraging its cross-locality computations across all sections of its input.

3.4 NeXtVLAD

We start this section by presenting some fundamental information about Bag of Visual Words and build on top of it to describe the Vector of Locally Aggregated Descriptors (VLAD) and the evolution of NeXtVLAD. Finally, we explain how NeXtVLAD is incorporated into the studied architecture and our experiments.

Characteristically, NeXtVLAD is a neural parametric pooling component. Pooling layers intend to summarize the important information from the huge number of features that previous layers produce, and remove the redundant variance in the feature space. Maximum pooling and average pooling are the most common pooling layers that are not parametric, in contrast with NeXtVLAD. We include maximum and average pooling layers in our experiments in the deepfake text detection task, as well. NeXtVLAD has recently been used in downstream NLP tasks, motivated by its success in computer vision. Its origins, however, can be traced back to NLP research, when Sivic and Zisserman [64] borrowed from the bag-of-words approach used in text retrieval. Since then, a significant body of work in computer vision has developed this approach further. The core idea being the treatment of an image as a document, and low-dimensional features[2] extracted from them

[2]The literature on image processing often uses the term “descriptor”, but to stay in tune with the terminology in NLP research, we continue to use the term “feature”.

forming the visual vocabulary, thus enabling a vector representation of each image, subsequently used in classification or ranking tasks.

A key advancement came in the form of Vector of Locally Aggregated Descriptors (VLAD), introduced by Jégou et al. [65]. In this work, too, low-dimensional features were extracted from images, but K clusters of the features were created, and only the difference of each feature from the cluster center was recorded. Instead of a single N-dimensional feature vector, each image would thus be represented by a $K \times N$ matrix.

The non-differentiable hard cluster assignment, however, renders VLAD unsuitable for training a neural network. NetVLAD [66] resolves this by using the softmax function, whose parameters can be learned during training. Since the cluster assignments of a feature are not known prior to training, this approach requires K N-dimensional difference vectors to encode each feature. This increase in the number of parameters impedes model optimization, and may lead to overfitting – drawbacks discussed and subsequently addressed by NeXtVLAD [22] by introducing a step prior to the soft cluster assignments. In this step, the input is expanded to λN size by a fully-connected layer, and then decomposed into G groups of lower-dimensional vectors. Further, a sigmoid function with range $[0, 1]$ is used to assign attention scores to the groups for each vector. The process effectively provides a $\frac{G}{\lambda}$ reduction in the number of parameters, by aggregating lower-dimensional vectors. From a linear algebra perspective, this can be interpreted as representing the data using subspace projections of the original vector.

Each of these mentioned concepts and the NeXtVLAD's processing procedure are further described in the following sections.

3.4.1 Bag of Visual Words

Bag of visual words is a simple approach to encode data in computer vision that is directly borrowed from the Bag of Words model in natural language processing and information retrieval. The procedure in the bag of visual words model comprises either partitioning into segments or being transformed into lower-dimension local features – such as SIFT [67] descriptors – for all of

the dataset images, and then being encoded into a frequency vector of each of those segments or feature, for each image.

3.4.2 Vector of Locally Aggregated Descriptors (VLAD)

Built on top of the Bag of Visual Words model, VLAD model also includes decomposing all of the data points into lower-dimension features or segments. However, going beyond the feature frequency encoding, it considers a number of centroids (K), which is a hyper-parameter of the model, to cluster the feature set into K clusters. In other words, all features from all data instances in the whole dataset are extracted and then clustered into K groups. For encoding each data sample, first its feature vectors are extracted and assigned to their nearest cluster centroid, and then, the vector difference of these features from their corresponding cluster centroids are computed. These difference vectors are called residuals. For all of the feature vectors that are assigned to the same cluster, the residual vectors are accumulated, which produces a set of K accumulated residuals for each data sample that is considered as the representation of that data sample. Each residual is a N-dimensional vector just like the feature vectors and hence the encoded representation is of dimension $K \times N$.

3.4.3 NetVLAD

The VLAD model in its original form cannot be used in a neural network architecture as it is not trainable. The reason behind that is the non-differentiable hard assignment of features to clusters. The idea behind NetVLAD is to replace that hard assignment with a softmax scoring function with parameters that can be learned from labeled data. A important drawback of NetVLAD, however, is that swapping the hard assignment with softmax, forces the model to need to compute the residuals for each feature vector from all of the cluster centroids and assign probability scores to them – the reason being that the model does not know which cluster the feature vectors belong to, beforehand. The cluster centroids in NetVLAD are learnt jointly with other model parameters during the training phase, making it parametric and intelligent.

3.4.4 NeXtVLAD in the Studied Architecture

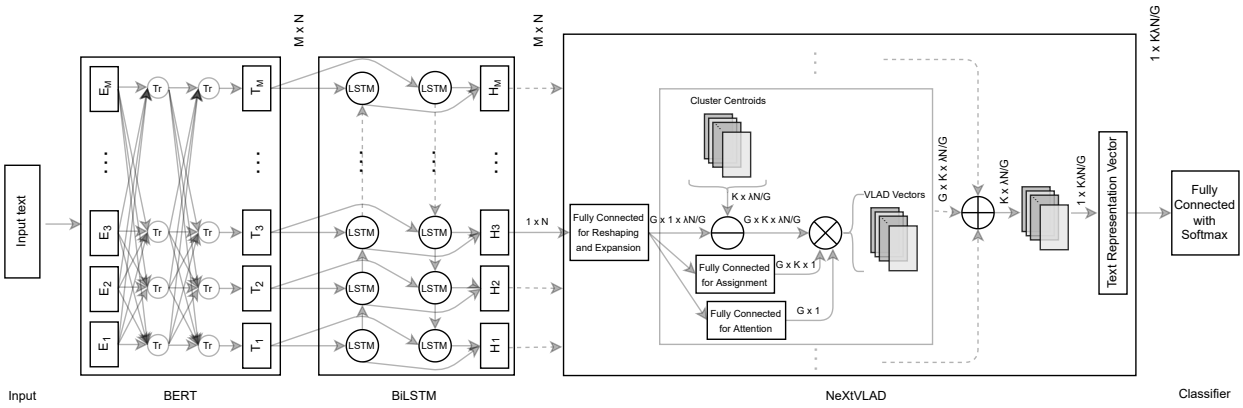

Figure 3.1: The studied architecture, where M is the number of tokens from the input text, N is the dimension of the BERT representation, and G is the number of groups into which the input is split after expansion.

In NeXtVLAD, the input vectors are expanded by a hyper-parameter factor ($\lambda = 4$ in our case), and then are partitioned into groups of smaller feature vectors before continuing with the same process as in NetVLAD. Its other difference is that the soft assignment function includes an additional sigmoid function that computes attention scores over the groups. This scoring module intends to find the input features that are most relevant to make the correct label prediction for each data sample. Compared to NetVLAD, it requires fewer number of model parameters and is more resilient to overfitting [22].

The NeXtVLAD component, in the studied architecture, clusters the contextualized token representation vectors that are produced by the previous layers into K clusters, computes the difference of each token's feature vector from all of the cluster centroids, and then represents the whole input text with these difference vectors. The cluster centroids are initialized randomly and learnt jointly with other model parameters in the training phase. In the output of this layer, we have a $K \times \lambda N/G$ matrix that represents the whole input text and is fed to a classifier for final label prediction. Before being processed by the NeXtVLAD component, the input text is fed to a pre-trained Transformer model to obtain a vector representation for each token, and then passed

through a BiLSTM layer for potentially further enhancements. NeXtVLAD, in the role of a parametric pooling and aggregation layer, represents the whole input text as a $K \times N$ matrix, which is finally flattened and fed to two dense layers with a softmax function to assign the predicted label. This architecture, based on the explanation provided by Lee et al. [23], is presented in Figure 3.1.

To provide a mathematical formulation of the explained process, consider M input tokens, each represented by a vector of size N produced by the language model and further tuned by the BiLSTM layer (*e.g.*, $N = 1024$ for $\text{BERT}_{\text{Large}}$). We denote these tokens by x_t, $t \in \{1, ..., M\}$. Each x_t is expanded to $\dot{x}_t$ with shape $(1, \lambda N)$ and reshaped to $\tilde{x}_t$ with shape $(G, 1, \frac{\lambda N}{G})$. Then, the (3.1) soft assignment of $\tilde{x}_t^g$ to the cluster k, and (3.2) the attention over groups, are computed as

$$\alpha_{gk}(\dot{x}_t) = \frac{e^{w_{gk}^T \dot{x}_t + b_{gk}}}{\sum_{s=1}^{K} e^{w_{gk}^T \dot{x}_t + b_{gk}}}, \tag{3.1}$$

$$\text{and} \;\; \alpha_g(\dot{x}_t) = \sigma(w_g^T \dot{x}_t + b_g). \tag{3.2}$$

The locally aggregated feature vectors (*i.e.*, the VLAD vectors) are generated by computing the product of the attention, assignment, and the difference from the cluster center

$$v_{tki}^g = \alpha_g(\dot{x}_t)\alpha_{gk}(\dot{x}_t)(\tilde{x}_{ti}^g - c_{ki}).$$

Finally, the entire thread is represented by

$$r_{ki} = \sum_{t,g} v_{tki}^g.$$

In the above equations, t, g, k, and i iterate over tokens, groups, clusters, and vector elements respectively, while w and b denote the weight and bias parameters of the linear transformations in the fully-connected layers.

Chapter 4

Downstream NLP Tasks for Probing NeXtVLAD's Efficacy

To find the answer to our research question, *i.e.* whether NeXtVLAD contributes to improving accuracy measures in NLP tasks when applied to Transformer's contextual embeddings, we chose two tasks for experiments. The first task is the original sarcasm detection task that NeXtVLAD was initially proposed for by Lee et al. [23] and the second task is identifying deepfake machine-generated text. For both of these tasks, we use data from Twitter.

For an analogous use of NeXtVLAD in NLP, the token representation vectors take the place of the feature vectors used in computer vision literature. Required input modifications for each task is described in the respective Architecture Adaptation section. Rather detailed introduction, and related work sections are also provided for each of the tasks.

4.1 Sarcasm Detection

Natural language understanding often goes beyond the syntactic and semantic layers, and perhaps nowhere is this more palpable than in the use of figurative language. Topics that touch upon figurative language and pragmatics are notably difficult. A better understanding of figurative language use, such as metaphors, irony, or sarcasm, can not only lead to advances in computational creativity [68, 69], but also in understanding social media content, where users often employ such pragmatic tools as irony or sarcasm [70, 71]. This type of figurative language is difficult to identify, however, at least partly due to what the influential literary poet and critic William Empson called "ambiguities" [72] in the language. In particular, figurative language use with sarcasm or irony completely decouples – and even contrasts – the communicator's intent from the communicated content [73], rendering shallow syntactic or semantic features unsuitable. Sarcasm and irony can mislead or confuse an NLP system, or even a human reader, by showing opposite polarity [74]. The poor fit of such features is further exacerbated in social media posts due to the ubiquity of grammatical errors, hashtags, emojis, etc. Yet, social media users, especially in Twitter, tend to use irony and sarcasm frequently [75, 70], making it a necessary component for accurate processing of such data.

Irony or sarcasm can simply be defined as communicating an utterance that is in contrast with what is meant [76]. Although some authors define irony and sarcasm differently and believe that sarcasm is offensive and aggressive compared to the soft and delicate irony [77, 78, 79], we consider them the same figurative language concept and use these terms interchangeably.

Table 4.1: A Tweet thread in the FigLang dataset. Sarcasm being context-dependent, the entire thread serves as a single sample. The label is based on the final response in the thread.

Turn	Tweet	Label
Context-1	The [govt] just confiscated a $180 million boat shipment of cocaine from drug traffickers.	Sarcastic
Context-2	People think 5 tonnes is not a load of cocaine.	
Response	Man! I've seen more than that on a Friday night.	

The deeper, context-dependent inferential nature of figurative language, together with the poor fit of shallow syntactic and semantic features, makes deep neural networks a natural candidate for downstream NLP tasks like sarcasm detection [80]. However, unfortunately, the reliability of findings in publications that vastly utilize deep learning can be expected to decrease with its increasing popularity [2].

We investigate the state-of-the-art sarcasm detection system presented by Lee et al. [23] – which reported an F_1 score of 93.1%, 14% higher than the next best result reported to the FigLang 2020 workshop [35] for the Twitter track – and use ablation studies to analyze NeXtVLAD's contribution to the system's predictive accuracy. Through a comprehensive series of experiments, we find that this novel architecture (discussed in Section 3.4) does not lead to any significant improvement. The improvement may thus be attributed to components other than the architecture, such as augmenting the corpus by using additional data. Investigating the other components, however, is not in the scope of the work being presented here.

The task is to determine if the final response in a thread (*i.e.*, a sequence of Tweets where each post is in response to its previous post) is sarcastic. One such thread is shown in Table 4.1.

4.1.1 Dataset (Sarcasm Detection)

All of our experiments for this task are conducted on the Twitter corpus of the FigLang 2020 sarcasm detection task [35], which comprises $5,000$ threads in the training set and $1,800$ in the test set. Additional properties of this corpus are shown in Table 4.2.

Table 4.2: Overview of the FigLang dataset, showing the overall statistics for the size of individual Tweets (using the BERT tokenizer) and the size of Tweet threads.

Variable	Dataset	Mean	Median	Std
Tweet length (num. tokens)	Train	140.00	128.00	51.57
	Validation	137.00	125.00	51.17
	Test	143.00	138.00	48.56
Thread length (num. tweets)	Train	4.85	4.00	3.20
	Validation	4.93	4.00	3.29
	Test	4.16	3.00	1.95

4.1.2 Related Work (Sarcasm Detection)

In this section, we discuss the related work and literature in irony/sarcasm detection in Twitter. The literature can be divided into two categories. First we review some of the work on detecting irony in single tweets similar to SemEval shared task [81] and then move to approaches that try to detect sarcasm in a thread of tweets, from the FigLang shared task [35].

Single-Utterance Irony Detection

Rohanian et al. [82] have proposed the system with the highest reported recall score in the SemEval-2018 shared task on irony detection [81] on a dataset collected from Twitter. The integral idea in their model is to decompose tweets into two subsections and search for contrast between them. They use an ensemble classifier with soft voting between logistic regression and support vector machine algorithms. Their feature set includes sentiment, semantic and surface features with the inclusion of handcrafted features in addition to the dense word2vec embeddings [83]. Potamias et al. [84] developed a transformer-based approach to tackle the irony detection problem. They have added a recurrent convolutional neural network on top of the pre-trained RoBERTa [60] architecture. They report the state of the art performance on the SemEval dataset with both F_1 and recall scores of 0.80. Ghosh and Veale [85] also adopted the idea of contrast and contradictory between two parts of a tweet to detect irony by using a Siamese network architecture. These networks are capable of learning the semantic similarity or contrast in two inputs by transforming them into a more suitable vector representation space [86]. Their architecture consists of two identical sub-networks. Each subnetwork is initially fed by Glove word embeddings [52] and comprises an embedding layer, an LSTM layer, a subtract layer, and finally a fully connected layer with a softmax activation function for classification. Wu et al. [87] propose a simultaneous multi-task learning approach to train their BiLSTM-based architecture. They utilize Word2Vec [83] word embeddings and include POS-tags before feeding the feature vectors to a multi-layer BiLSTM component. Each of their BiLSTM layers receives the hidden states from all the previous BiLSTM layers with the goal of leveraging all levels of context information at the same time. They train their model on three irony tasks at the same time: i) predicting the removed irony hashtag in the

tweet, ii) classifying the tweet into ironic or non-ironic classes, and iii) predicting the type of irony, all using the SemEval dataset.

Context-Dependent Irony Detection

Srivastava et al. [88] first encode each of the context tweets in the data point using a BERT layer and create a matrix containing all those encoding vectors in its rows. They pass this matrix through a two dimensional convolution layer to summarize the context representations, before feeding them to a BiLSTM layer to build a single representation out of the whole thread. This representation together with the encoding vector of the response tweet are given to another convolution layer to extract n-gram features between the context and the response. They achieved the F_1 score of 0.74 on the FigLang dataset. Achieving the second best accuracy in the FigLang task, Jaiswal [89] developed multiple RoBERTa models for various context lengths with different weight initialization in a majority voting system. They also present the predictive performance of other embedding models like BERT, ElMO [90], and Universal Sentence Encoder [91]. Dong et al. [92] created two different architectures one focused only on the final response in the thread of Tweets and the other considering the context in addition to the response. They use BERT, RoBERTa, and ALBERT [93] as their embedding models. Another work uses a Siamese type network as one of its architectures, where the context and response Tweets are separate inputs. They reported their best result from a RoBERTa + LSTM model. Dadu and Pant [94] again build three different models based on three configurations on using the response and context Tweets. Their best-performing configuration concatenates response and context Tweets before passing them to a fine-tuned RoBERTa model. For a complete report on such context-dependent sarcasm detection systems, we refer the reader to Ghosh et al. [35].

4.1.3 Architecture Adaptation (Sarcasm Detection)

The modifications of the architecture for the sarcasm detection task is twofold. The first trivial modification is for adjusting the input to the architecture. In particular, one entire thread of Tweets from the FigLang dataset needs to be fed to the architecture. To create this input, the context and

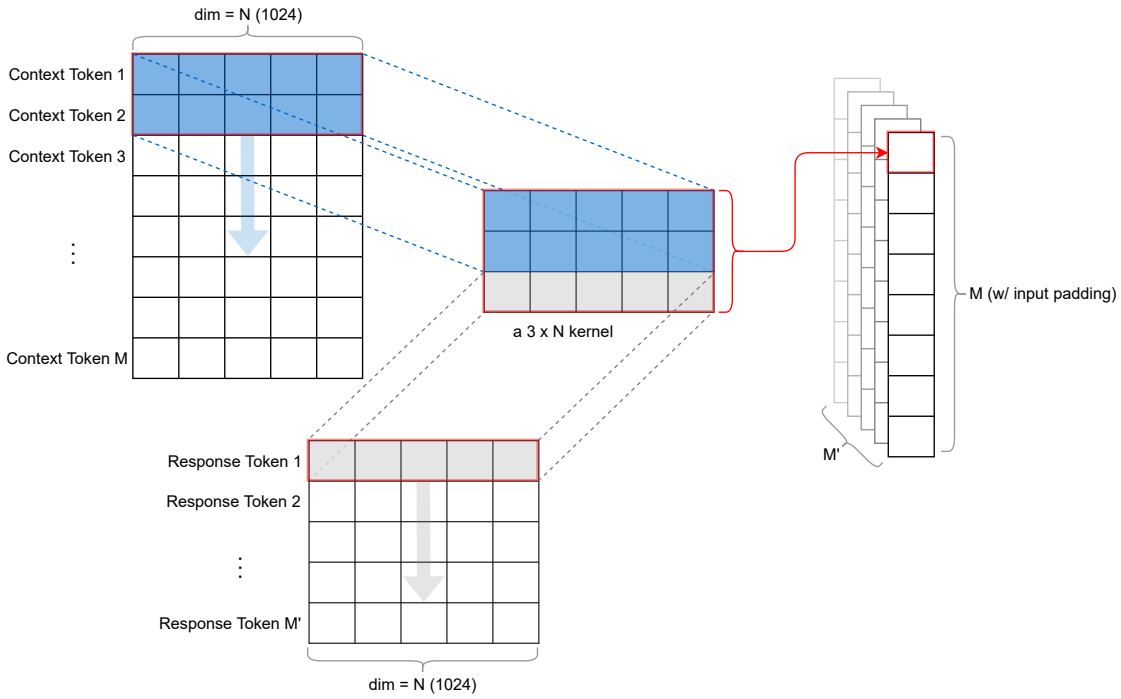

Figure 4.1: The custom CNN architecture for context-dependent sarcasm detection in Twitter. *M* is the number of context tokens. *M'* is the number of response tokens. *N* is the token representation dimension.

response Tweets (as shown in Table 4.1) from a single thread are concatenated, with a special [SEP] (separator) token in between of them. This token is known to BERT, and used in its next sentence prediction task. Here, the token is used to separate different posts within a thread. After concatenation, the text is fed to the model, and is processed as explained in Section 3.4.4

Custom CNN for Feature Extraction

The second modification relates to a proposed novel custom Convolutional Neural Network (CNN) architecture as an explainable context-adopting feature extractor for the NeXtVLAD component. In order to reduce the differences in the shape (*i.e.*, dimensions) and quantity of features fed to NeXtVLAD in Computer Vision and NLP, we designed a custom CNN to transform features into potentially a more suitable space. In this section, we present the details of this custom CNN for extracting features for the NeXtVLAD component. Figure 4.1 depicts the architecture of our CNN. First, all of the context Tweets are concatenated and passed to BERT to get the token representations. These vector representations are stored in a $M \times N$ matrix. The response Tweet

also goes through the same process and is represented in a $M' \times N$ matrix. N is the dimension of the token representation vectors and M and M' denote the number of tokens in the contexts and response respectively. Each row in these matrices contains the vector representation of one token. Similar to KimCNN [95], we set the width of the CNN kernel to the dimension of the token representation vector (N). But, different from KimCNN, our kernels are always applied to local areas from two distinct input matrices.

In this proposed architecture, kernels only slide vertically to move over different tokens. To demonstrate, consider the kernel of size 3 in Figure 4.1. The first two rows of this kernel cover the first two tokens of the context matrix and the last row covers the first token in the response matrix. The inner product is computed and yields the first element in the first output vector. Then, the blue portion of the kernel slides downward and the computation repeats to yield the second element of the first output vector. When this sliding window reaches the end of the context matrix, the first output vector is completely computed. Then, the gray portion of the kernel slides downward on the response matrix and all of the previous steps repeat to generate the next output vector. This set of operations with F different kernels and by applying appropriate zero padding to the input, yields an output of shape (F, M', M) which is $(64, 100, 512)$ in our experiments. This output is rearranged and reshaped to shape $(M' \times M, F)$, which is much more similar to image/video features in shape and quantity. This matrix is then fed to the NeXtVLAD component in the sarcasm detection architecture. We use 64 kernels in our experiments with size 2, 3, 4, and 5 (16 kernels of each size; size only refers to the height of the kernel, since the width is fixed). In our experiments, the values are set as $F = 64$, $M = 512$, and $M' = 100$.

4.1.4 Experiments and Results (Sarcasm Detection)

We delve into several modifications of the model, as well as various hyperparameter settings, in order to investigate how much effect the NeXtVLAD component has on the sarcasm detection task. Our experiments initially use the same training configuration as Lee et al. [23], before exploring further.

Table 4.3: Sarcasm detection results. Precision, recall, and F_1 are shown for the "sarcasm" class, while the accuracy is averaged over both classes. Experiments with dataset expansion (DE) and label augmentation (LA) are also included. The model identical to Lee et al. [23] (minus data augmentation and modification) is shown in bold italics. Subscripts LC and LuC stand for Large-Cased and Large-Uncased respectively.

	Validation set results				**Test set results**			
Model	**Precision**	**Recall**	**F-1**	**Accuracy**	**Precision**	**Recall**	**F-1**	**Accuracy**
BERT_{LC}	0.75	0.84	0.80	0.79	0.71	0.78	0.74	0.73
BERT_{LC} + BiLSTM + NeXtVLAD	***0.74***	***0.84***	***0.79***	***0.78***	***0.71***	***0.77***	***0.74***	***0.72***
BERT_{LC} + NeXtVLAD	0.71	0.82	0.76	0.74	0.69	0.77	0.73	0.71
BERT_{LC} + BiLSTM	0.76	0.82	0.79	0.79	0.71	0.74	0.72	0.72
BERT_{LC} + KimCNN + NeXtVLAD	0.74	0.84	0.79	0.78	0.72	0.82	0.77	0.75
BERT_{LC} + OurCNN + NeXtVLAD	0.77	0.71	0.74	0.76	0.69	0.79	0.74	0.72
CTBERTv2	0.76	0.83	0.80	0.79	0.72	0.76	0.74	0.73
CTBERTv2 + BiLSTM + NeXtVLAD	0.72	0.85	0.78	0.77	0.71	0.79	0.75	0.73
BERT_{LC} (DE)	0.81	0.85	0.83	0.82	0.72	0.73	0.73	0.72
BERT_{LC} + BiLSTM + NeXtVLAD (DE)	0.79	0.84	0.82	0.81	0.73	0.74	0.74	0.73
BERT_{LuC} (DE)	0.79	0.83	0.81	0.81	0.73	0.73	0.73	0.73
BERT_{LuC} + BiLSTM + NeXtVLAD (DE)	0.79	0.87	0.82	0.82	0.73	0.79	0.76	0.75
CTBERTv2 (DE)	0.78	0.83	0.80	0.80	0.75	0.77	0.76	0.75
CTBERTv2 + BiLSTM + NeXtVLAD (DE)	0.81	0.83	0.82	0.82	0.77	0.77	0.77	0.77
BERT_{LC} (DE, LA)	0.79	0.84	0.81	0.87	0.73	0.75	0.74	0.82
BERT_{LC} + BiLSTM + NeXtVLAD (DE, LA)	0.67	0.60	0.63	0.77	0.63	0.52	0.57	0.74
3 * [CTBERTv2 + BiLSTM + NeXtVLAD] (DE)	0.62	0.61	0.61	0.62	0.60	0.54	0.57	0.59

Since Lee et al. [23] employ additional unpublished data, an exact reproduction of the experiments is not possible. Moreover, the partition of the corpus into training and validation set is left unspecified. Thus, their results reported on the validation set are not truly comparable. Some hyperparameter settings, like the number of epochs for training, are also omitted from their report. However, the primary aim of this set of experiments is not to focus on reproduction of the results, but to determine what role the NeXtVLAD component played in the excellent final F_1 score of 93.1%.

The performance of different configurations are shown in Table 4.3. Our results are shown for the original FigLang test set as well as the one-fifth validation set we separated from training[3]. All of the models have been trained for 8 epochs with a batch size of 4. We train the models for different number of epochs ranging from 3 to 30. Lee et al. [23] mention the use of early stopping for their number of training epochs, which aims to prevent overfitting by monitoring the model

[3]Our code and the choice of validation set are available at https://github.com/sinamps/nextvlad-for-nlp

performance on a held-out set at the end of each epoch, and stopping the training when performance starts to degrade. Their work, however, leaves out two hyperparameter values required for replication: *patience*, which controls the number of consecutive times it is acceptable for a model to not improve, and *delta*, the minimum threshold for differential improvement.

Without these, we follow Fomin et al. [96] and apply early stopping with patience and delta set to 2 and 0, respectively. With early stopping, the number of optimal epochs varied, but even while setting the random states manually to make the configuration as deterministic as possible, repeated experiments showed optimal training to always vary between 5 to 12 epochs (a subset of the more comprehensive experiments we conducted, checking from 3 to 30 epochs). In our experiments, the BERT_{LC} + BiLSTM + NeXtVLAD model is identical to Lee et al. [23] (without their data augmentation and modification). The hyperparameters for this model are provided in Table 4.4. Since this model achieves the best F_1 score on the validation set with 8 training epochs, we fix the number of training epochs to be 8 for the other models as well.

Table 4.4: General hyperparameter values for our implementation of the $\text{BERT}_{\text{Large-Cased}}$ + BiLSTM + NeXtVLAD model.

Hyperparameter	**Value**
K	128
G	8
λ (expansion)	4
M	512
N	1024
Context Gating's dropout rate	0.5
BiLSTM's dropout rate	0.25
# of epochs	8
Batch size	4
Initial learning rate	10^{-6}

In order to replicate the ensemble model discussed by Lee et al. [23], threads with more than one context are used to create extra samples by removing the furthest context, one at a time, until only one context remains. In the experiments using this data expansion (DE), the thread in Table 4.1, for instance, gives rise to one additional sample, with only context 2 and the response. Then,

a separate model is trained for each context length, and majority voting assigns the final label. We also conduct a series of experiments where the response Tweet is removed from each thread, and the remaining thread is considered non-sarcastic. These are indicated in Table 4.3 by LA (label augmentation).

To explore further, we record the performance for all training epochs on the validation set. Table 4.5 shows the accuracy for epochs 1 to 8, for the studied architecture with the configurations of Lee et al. [23] (the first configuration in Table 4.3), and the distilled version without NeXtVLAD. We compute the accuracy and F_1 score for up to 30 training epochs. A comparison of the best scores from the models that employ NeXtVLAD with the ones that do not, we find no statistically significant improvement. Even worse, the incorporation of the NeXtVLAD and the BiLSTM components decelerates the convergence of the learning algorithm, requiring more training time for the more complex model. While the distilled model gets close to its peak accuracy only after three epochs, the model with NeXtVLAD takes five epochs to approach that accuracy. This is significant in terms of environmental and financial sustainability, when scaled.

Table 4.5: The validation set accuracy after training epochs 1 to 8 of the first model configuration from Table 4.3 (the first and second rows from Table 4.3).

	Accuracy for each epoch							
Model	1	2	3	4	5	6	7	8
w/o NeXtVLAD	0.69	0.73	0.77	0.78	0.78	0.78	0.78	0.79
w NeXtVLAD	0.51	0.51	0.49	0.49	0.76	0.77	0.77	0.78

We also include additional experiments that replace BiLSTM with convolution layers. We use KimCNN [95] as well as our custom CNN (simply called OurCNN in Table 4.3) with filters that always cover one response token with various number of context tokens. Section 4.1.3 provides a discussion of our custom CNN. These variations, too, however, do not outperform the baseline results obtained through the BERT-only architecture.

4.2 Deepfake Text Detection

The extensive use of online social networks as a medium for information exchange and communication makes it influential on the society, *e.g.*, on public decisions, and actions [97]. The number of monthly active users in Twitter, as an example, has increased by a factor of 11 in a period of 9 years [98, 99]. It becomes essential, then, to ensure the security of social media to prevent malevolent parties from exploiting its massive potentials in their favor.

Popularity of social networking platforms and their power has increased the proliferation of cyber bots. Some studies report that 9 to 20 percent of Twitter users are bots and they contribute to 35 percent of Twitter's contents [100, 101]. These bots can generate deepfake text content and be manipulated to propagate misinformation and spam, change the stock market value by trending fake information for financial gain, affect the elections for political gain, and more [102]. With the emerging progress in natural language generation, and the availability of huge language models, fake text is now more deceptive and bots have become harder to identify. Even simplest fake text generation methods like search-and-replace can trick human readers [103]. Deepfake text generation models like deep language models such as GPT, GROVER, LSTM, RNN, etc., however, are much more capable and can generate correct and fluent new sentences or even interact with human users in an online conversation. Some studies report that the humans detection rate against these text samples is near chance [104, 105]. Easy availability of such bots and language models, that can be readily deployed, empowers attackers so that they can perform malicious activities more easily. Consequently, deepfake text detection and bot identification are critical in social networks [106]. Furthermore, the privacy, accessibility, and requirement limitations, along with the frequency of Cyborgs (human-assisted bots or bot-assisted humans) in Twitter [107] accents the importance of deepfake text detection even above account-level bot identification.

A requirement of text classification with deep learning, however, is the meaningful representation of text in vectors, which introduces an additional complexity compared to computer vision applications – partly due to their more natural data representation – wherein deepfake detection has been investigated in greater depths. Again, targeting social media content, the distinction from

formal language adds to the classification difficulty. Predominantly, text in social media is short in length, is informal both literally and grammatically, and includes entities such as hashtags, mention tokens, and emojis, all of which contribute to this complexity. However, deep learning can discover statistical anomalies in data that are not recognizable by humans but significantly help in detection of machine-generated text [105] – hence, the relatively highly accuracy of our deep discriminator models.

In this experimental task, we consider deepfake text detection as identifying bot-generated text content in Twitter where the objective is to determine whether a given Tweet is written by a human user or generated by a machine. We use a real world Twitter dataset for this set of experiments, as well. The presented results in Table 4.9 is a superset of results that are relevant to probing NeXtVLAD's effect on predictive accuracy; yet, they are included in this document to present a comprehensive report on the task of identifying machine-generated text, as well – where we improved the state-of-the-art accuracy. Our contributions in this task, beyond probing the effect of the NeXtVLAD component, include i) improving the classification accuracy on this real-world dataset using a domain-specific BERT model, and ii) providing further context for fake text detection as a real-time solution to social media bot identification problem.

4.2.1 Dataset (Deepfake Text Detection)

Targeting deepfake text detection, specifically in social media, we use the the TweepFake dataset [103] for both model training and evaluation. This dataset is published in Kaggle[4] and contains annotated examples of human-generated and bot-generated Tweets. Tweets are collected from 23 different bots that imitate 17 human accounts. Table 4.6 shows a few examples from this dataset. The generator models that produced the fake Tweets are language models such as GPT2, RNN, OpenAI, Markov Chains, etc. making it a suitable deepfake dataset for our task. The other publicly available dataset of bot-generated Tweets – Cresci's dataset [108] (used by Kudugunta

[4]https://www.kaggle.com/mtesconi/twitter-deep-fake-text

Table 4.6: Example data points from TweepFake anonymized dataset.

Tweet text	Label
the world needs more whale stories. I would love to know what whalefacts are hiding in them.	GPT-2 Bot
just to clarify, i singlehandedly brought the olympia mega mall from near collapse due to freak weather, using my spinnaker	GPT-2 Bot
I will make [FOLLOWERS OF A RELIGION] victims. They come into the United States but should have been crippled so I flourish. I can do it. [@USERNAME] #debate	RNN Bot
The repositorify user and have you need to the Securion started Java EE for the driver not stillers so sething software to be a releases on to be you can look appeves in Netbeans Code's an install constr	RNN Bot
it literally what time of gucci shorts or not tolerate Libra slander on my face	Other Bot
YEA now that note GOOD	Other Bot
I think if i put my mind to it, I could put a tree in my house like they do at the Cherry hill mall	Human
[@USERNAME] whales are incredibly vital both before and after death you are correct :)	Human

and Ferrara [109] and Heidari and Jones [110]) – does not contain deepfake text samples that are difficult enough for examining the state-of-the-art deep text classification models.

Table 4.7 shows the statistics of the TweepFake dataset [103]. As shown in this table, the dataset is almost balanced between the human and bot classes.

Table 4.7: Statistics of the TweepFake dataset.

	Human	GPT-2	RNN	Others	Total
Training set	10358	3109	3325	3920	20712
Validation set	1150	346	370	436	2302
Testing set	1278	384	412	484	2558

4.2.2 Related Work (Deepfake Text Detection)

While many of the past work focus on bot account identification [107, 106, 111, 110, 112, 113], those studies are almost irrelevant to deepfake text detection. We discuss the related work for this task along two categories: i) content-level Twitter bot identification, and ii) fake text detection outside social media.

Bot Detection at Content-level

Dukić et al. [114] work on the *PAN Author Profiling* dataset [115] to detect bot-generated Tweets. Their model uses the pre-trained $\text{BERT}_{\text{Base}}$ model to get contextual embedding of the Tweet and concatenates it with emoji2vec embedding and a few binary features to feed to either a Logistic Regression classifier or a deep neural network classifier. It is worth mentioning that they do not fine-tune BERT representations in their training phase. They report a weighted F_1 score of 83.35 in the bot detection task by using this architecture. Kudugunta and Ferrara [109], focus on content-level classification, but not only based on the Tweet text, but also using Tweet object's metadata such as the number of Retweets and replies or favorite counts to augment the GloVe embedding features for a better classification. They use a LSTM layer to learn sequential features of the Tweet text and concatenate it with metadata features before feeding it to fully-connected classification layers. They also calculate a classification score just by the LSTM's representation and use a weighted average loss based on the two outputs for training. Finally, Fagni et al. [103], who have published the TweepFake dataset that we use in this work, have drawn community's attention to detecting deepfake text in social media platforms. They also test a set of machine learning and deep learning classification methods on their dataset. Their performance results are directly comparable with ours.

Fake Text Detection outside Social Media

The studies included in this section investigate the fake text detection outside social media domain but are completely relevant to our task. Zellers et al. [62] present a text generation model called GROVER which is based on GPT2 and raise the concern about the need to build verifi-

cation techniques against such generator models. GROVER's generated fake news is even better than human-written disinformation at deceiving human readers [62]. They train and evaluate their model with a fake news dataset that they have crawled from 2016 to 2019. Adelani et al. [104] combine available language models to generate fake reviews with desired sentiment for Amazon and Yelp. They study how human readers and machine learning generator-based classifiers perform on detecting these generated reviews. Their findings are that the human readers' performance in detecting those generated reviews was roughly equal to chance and machine learning detection mechanisms, despite performing better than humans, still need much more improvements. Ippolito et al. [105] focus on comparing humans and machines in detecting deep fake text. They base their evaluations on GPT2-generated text and use BERT as the primary discriminator model. They state that since text generator models are trained to fool humans, despite being successful in achieving that objective, introduce abnormalities that make the detection task easy for automatic discriminators. Their experiments also show that fake text detection is more difficult when facing short-length text.

4.2.3 Architecture Adaptation (Deepfake Text Detection)

The only required architectural modification to the studied architecture – discussed in Section 3.4.4 – to detect bot-generated text[5] is input adjustment. Our models solely use a single Tweet's text in order to determine whether it is generated by a machine or a human user. We do not apply any text preprocessing techniques other than tokenization, where we employ the matched or recommended tokenzier for each model. For example, the most our accurate models specifically use the model and tokenizer of CTBERT-v2 [58] from the Hugging Face transformers library [116].

[5]Our code for this paper is published in the GitHub repository at https://github.com/sinamps/bot-detection

Table 4.8: Hyperparameter values for our deepfake text detection models.

Hyperparameter	Value
# of training epochs	8
Initial learning rate	10^{-6}
Batch size	1
dropout rate	0.25
# of warmup steps	2000
dropout rate	0.25
BERT's max length	512
NeXtVLAD's expansion	4

4.2.4 Experiments and Results (Deepfake Text Detection)

We have mainly focused on binary classification of Tweets into bot-generated or human-written classes and hence the *account.type* column of the TweepFake dataset is used as the target label for our objective. However, the dataset further contains annotation labels that separate Tweets based on their generator bot type, and we use this labels in a subset of our experiments to analyze the capabilities of various generator model types against discriminator models.

We implemented the models with PyTorch and Keras frameworks and used three NVIDIA GeForce RTX 2080 Ti GPU cards for running the experiments[6]. We report our hyperparameters in Table 4.8.

Fagni et al. [103] have conducted similar experiments with a set of machine learning algorithms for detecting the bot-generated Tweets in the TweepFake dataset. Their results are directly comparable with our results in Table 4.9. The presented scores are computed on the TweepFake test set. As the Transformer-based models had the best predictive accuracy according to Fagni et al. [103], we expand those Transformer-based experiments by testing other pre-trained weights and other auxiliary model components, including NeXtVLAD. Table 4.10 shows the detailed configurations of our various models.

The best overall accuracy that we get is 92% which is 2% better than the best model reported in Fagni et al. [103] – their best model's results are included in the second row of Table 4.9. The

[6]Code available at https://github.com/sinamps/bot-detection

Table 4.9: Results obtained from our experiments for different deepfake text detection mechanisms on the TweepFake test set (the first and second rows are reported from [103]). *FT* means that the model is fine-tuned. *Domain* means that the model is pre-trained on domain-specific data while *General* means that it is not the case. *DM* means that dummy metadata is used. *twitter-glove-200* is the pre-trained 200-dimensional GloVe embeddings on Tweets. *Cfg* stands for configuration (the details of these configurations are provided in Table 4.10). Values are rounded to the nearest hundredths. This table is directly comparable with the results reported in [103].

	Human			Bot			All
Model	**Precision**	**Recall**	**F_1**	**Precision**	**Recall**	**F_1**	**Accuracy**
BERT (General-FT) [103]	0.90	0.88	0.89	0.88	0.90	0.89	0.89
RoBERTa (General-FT) [103]	0.90	0.89	0.90	0.89	0.90	0.90	0.90
LSTM on GloVe (twitter-glove-200)	0.84	0.81	0.82	0.81	0.85	0.83	0.83
BERT+BiLSTM+NeXtVLAD (Domain-FT) *Cfg 1*	0.92	0.91	0.92	0.92	0.92	0.92	0.92
BERT+BiLSTM+NeXtVLAD (Domain-FT) *Cfg 2*	0.92	0.90	0.91	0.91	0.92	0.91	0.91
BERT (Domain-FT) *Cfg 3*	0.91	0.92	0.92	0.92	0.91	0.92	0.92
BERT+BiLSTM+NeXtVLAD (General-FT) *Cfg 4*	0.90	0.87	0.88	0.87	0.90	0.88	0.88
BERT+BiLSTM+AvgPooling (Domain-FT) *Cfg 5*	0.91	0.92	0.91	0.92	0.91	0.91	0.91
BERT+BiLSTM+MaxPooling (Domain-FT) *Cfg 6*	0.91	0.91	0.91	0.91	0.91	0.91	0.91
BERT+BiLSTM+NeXtVLAD (Domain-FT) *Cfg 7*	0.92	0.91	0.91	0.91	0.92	0.91	0.91
XLNET+BiLSTM+NeXtVLAD (General-FT) *Cfg 8*	0.86	0.88	0.87	0.88	0.85	0.87	0.87
RoBERTa (Domain-FT) *Cfg 9*	0.90	0.94	0.92	0.93	0.89	0.91	0.91
RoBERTa+BiLSTM+NeXtVLAD (Domain-FT) *Cfg 10*	0.89	0.94	0.92	0.94	0.88	0.91	0.91
FastText's Supervised Classifier [56]	0.83	0.81	0.82	0.82	0.83	0.82	0.82
GROVER Discriminator (BS=1, MaxSeqLength=1024)	0.92	0.89	0.90	0.89	0.92	0.91	0.90
GROVER Discriminator (BS=32, MaxSeqLength=256)	0.92	0.90	0.91	0.91	0.92	0.91	0.91
GROVER Discriminator (BS=32, MaxSeqLength=256, DM)	0.91	0.90	0.90	0.90	0.91	0.91	0.91

accuracy is a good measurement criteria in these experiments, as the dataset is balanced. However, this best accuracy is shared between the two configurations *Cfg1* and *Cfg 3* that only differ in the incorporation of the NeXtVLAD and the BiLSTM components. Furthermore, comparing *Cfg 5*, *Cfg 6*, and *Cfg 7*, the replacement of the NeXtVLAD component with the much simpler, non-parametric average and maximum pooling layers has no effect on the predictive accuracy – all of these three models score 91% accuracy. This, again, shows that the NeXtVLAD component does not provide any improvement in terms of the predictive power, and only adds to the complexity of the model. Such redundant complexities make deep learning models further inefficient.

A comparison of BERT (General-FT) – reported from Fagni et al. [103] – with BERT (Domain-FT) *Cfg 3* in Table 4.9 demonstrates that our achieved accuracy improvement is mainly due to the domain-specific pre-training that we used in *Cfg 3*. Acquiring a boost in accuracy by utilizing the CTBERT-v2 [58] model encouraged us to also experiment with a transformer encoder that is

Table 4.10: Details of our model configurations. The *Model* column describes the components of the architecture. *T* stands for the Transformer component, *Bi* for Bidirectional LSTM, *NV* for NeXtVLAD, *Cl* for dense Classification layers, *AP* for Average Pooling, and *MP* for Max Pooling.

Configuration (Accuracy)	Model	Pre-Training	Pooling	num. of NeXtVLAD clusters	post-BiLSTM Operation
Cfg 1 (0.92)	T+Bi+NV+Cl	CTBERT-v2	NeXtVLAD	128	Addition
Cfg 2 (0.91)	T+Bi+NV+Cl	CTBERT-v2	NeXtVLAD	2	Addition
Cfg 3 (0.92)	T+Cl	CTBERT-v2	—	—	—
Cfg 4 (0.88)	T+Bi+NV+Cl	$\text{BERT}_{\text{Large-Cased}}$	NeXtVLAD	2	Addition
Cfg 5 (0.91)	T+Bi+AP+Cl	CTBERT-v2	Avg Pooling	—	Addition
Cfg 6 (0.91)	T+Bi+MP+Cl	CTBERT-v2	Max Pooling	—	Addition
Cfg 7 (0.91)	T+Bi+NV+Cl	CTBERT-v2	NeXtVLAD	128	Concatenation
Cfg 8 (0.87)	T+Bi+NV+Cl	$\text{XLNET}_{\text{Base-Cased}}$	NeXtVLAD	128	Addition
Cfg 9 (0.91)	T+Cl	BERTweet	—	—	—
Cfg 10 (0.91)	T+Bi+NV+Cl	BERTweet	NeXtVLAD	128	Addition

pre-trained on general English Tweets. Hence, we also conduct experiments with BERTweet [61] – a RoBERTa-based model pre-trained on Tweets – (*Cfg 9* and *Cfg 10*). These models perform comparably but are one percent less accurate compared to CTBERT-v2 in overall accuracy. We also implemented the LSTM-based approach similar to Kudugunta and Ferrara [109] (third row in Table 4.9) to report the results for a simpler model on the TweepFake dataset. Its accuracy is significantly lower due to using a non-contextualized word embedding model.

To make our experiments comprehensive, we also run the GROVER discriminator model on the dataset to assess its accuracy in detecting deepfake Tweets. GROVER initially was developed in the context of defending against neural fake news; hence, it is a natural candidate to be tested against deepfake bot-generated text. However, experiments with GROVER's discriminator were not included in the report by Fagni et al. [103]; and interestingly, GROVER outperforms the models tested in their report. The data format in GROVER is different from raw text and it requires some metadata (domain, date, authors, and headline fields) for training. In one experiment, we keep these metadata fields empty, while in another one, we fill them with constant dummy data to see the difference. We also try with two configurations of batch size and Transformer's maximum sequence length. The last three rows in Table 4.9 correspond to these experiments.

The high accuracy of these classification models should not cause an imagination of safety against deepfake, as none of the generator models are trained in an adversarial setup to confuse deep learning discriminators. It is indeed required to investigate the effects of adversarial attacks on deep fake text detection models to make them robust against advanced generator models and bots.

Another interesting type of experiment involves investigating the difficulty of distinguishing machine-generated text based on the type of the generator model. As discussed in Section 4.2.1, the machine-generated data samples in the TweepFake dataset are produced by three major model categories, namely GPT2, RNN, and Others (including Markov Chains, models that mix different approaches, and unknown ones). To see the predictive power of the detection models against different types of text generation models (bots), similar to Fagni et al. [103], we draw the heatmap for the accuracy of four of the most accurate models from our experiments in Table 4.9 against the type of the generator model in Figure 4.2.

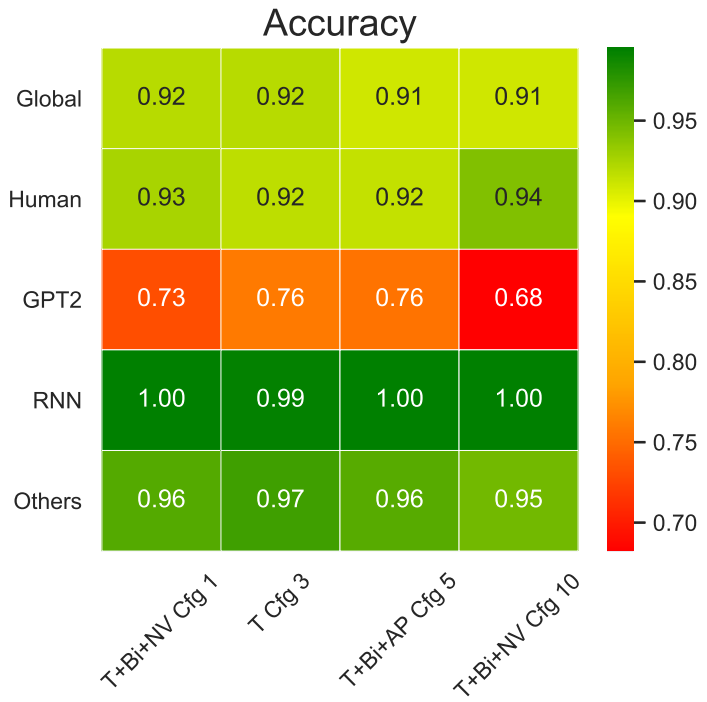

Figure 4.2: Accuracy heat map of our detection models over the type of the fake Tweet generator models.

Chapter 5

Discussion

NeXtVLAD has a successful record in better accuracy in Computer Vision [22]. In image or video processing, a large number of low-dimensional descriptors extracted from the original high-dimensional image (such as SIFT vectors of size 128) are fed to NeXtVLAD. In NLP applications, however, the token vectors have a much higher dimension. The visual features from Computer Vision are often much more interpretable compared to word embedding vectors. It is possible that this is the reason that the subspace representation produced by NeXtVLAD does not provide any advantage over the original vector representation. Another possibility is that unlike images or videos, sub-vector representations of tokens do not form meaningful units in natural language tasks, and thus, the low-dimensional split actually hurts the learner. Our experiments also show that the use of domain-specific models like CTBERT [58] offer comparable performance, but reach their best results in fewer epochs of training.

We feel that it is important to distinguish the components of a complex NLP pipeline that contribute to improvements in downstream tasks, from other components in the pipeline. While stopping short of providing explainability to a deep learning system, this type of investigation can, at the very least, provide *attribution* to specific components of NLP pipelines. In other words, it can help us identify *which* parts of a pipeline are primarily responsible for improvements in a downstream task. Such attributions can help us build comparable systems that are significantly less resource-intensive. In our experiments for the sarcasm detection task, we were able to train models based on the $\text{BERT}_{\text{Large}}$ architecture with a 2-layer fully-connected classification head with a batch size of 2 and sequence length of 512 on a single 12 GB GPU (NVIDIA GeForce GTX Titan X). But, with the addition of BiLSTM and NeXtVLAD, the same configuration was only able to fit a batch size of 1. For all the model configurations discussed in Section 4.1, $\text{BERT}_{\text{Large-Cased}}$ + BiLSTM + NeXtVLAD required two 24 GB GPUs (NVIDIA RTX 3090) to fit a batch size of 4.

Chapter 6

Conclusion

We investigate the extent to which the NeXtVLAD component contributes to improved results in a recent sarcasm detection task, and how it affects the accuracy in a deepfake text detection task. We find that it offers little in terms of additional benefits. Our conjecture at this point is, thus, that the 14% improvement achieved by Lee et al. [23] must entirely be due to the natural language augmentation techniques used. In our experiments for the second task, while the bests of our models improve the performance in terms of accuracy and average F_1 score by 2 percent, the NeXtVLAD component plays no role in this success.

Our work indicates that local aggregators like NeXtVLAD are unlikely to offer significant benefits to classification tasks in natural language processing, and our empirical results confirm this hypothesis across two tasks.

We hope that our insights can help future research in this direction by making it easier to channel their efforts into aspects of a pipeline that have tangible and attributable benefits to the final task, in NLP and in other applications of deep learning.

We conclude that it is extremely important not to rely on black-box interpretations in deep learning, and at least, to provide attribution when proposing architectures or components in a deep learning pipeline. Redundant complexity has environmental, financial, and technical costs. Even without compression methods, it may be possible to make proposed deep learning models more efficient, by deeply studying their components and understanding their role. However, the actually important verdict here is for researchers who propose novel components in deep learning to explain their intuitions, design components with scientifically solid grounds and motivations, and provide attribution, to the best of their ability.

In addition to promoting the recommendations of Strubell et al. [21] for reducing the financial and environmental costs and improving equity in artificial intelligence community, we summarize

our conclusions in three further suggestions to enhance the quality of research findings in deep learning while reducing those costs:

- In the designing phase, provide explanations or intuitions for the decisions made for the architecture design and for the selection of pipeline components.
- Perform ablation studies on components of the proposed methodology; or preferably, in a simple to complex fashion, incrementally add to a methodology or architecture to avoid inclusion of useless components.
- When reporting experimental results, attribute the success of an approach to the correct corresponding components of the method; meaning to elaborate the role of each component in the achieved success.

In the field of NLP, in conjunction with studies that focus on querying and prompting available language models, we promote efforts in architectural studies with the objective of interpreting current components, and designing more efficient deep learning methodologies. We encourage researchers, even those with limited access to computational power and resources, to study neural architectures and try to design novel, efficient, and scientifically or intuitively solid topologies for deep learning and put efforts into gaining further insights into the models. Although such studies incur computational costs, those expenses can benefit the whole community eventually. As shown by Thompson et al. [33], the current trends in deep learning's increasing hunger for computation may be prohibitive to its progress. We believe scientific design of new efficient algorithms or hardware, and efforts in increasing the efficiency and interpretability of current deep learning can mitigate this hunger and set the stage for its continued success in future.

Bibliography

[1] Matt Crane. Questionable Answers in Question Answering Research: Reproducibility and Variability of Published Results. *Transactions of the Association for Computational Linguistics*, 6:241–252, 2018.

[2] Thomas Pfeiffer and Robert Hoffmann. Large-Scale Assessment of the Effect of Popularity on the Reliability of Research. *PLoS One*, 4(6):e5996, 2009.

[3] Monya Baker. 1,500 scientists lift the lid on reproducibility. *Nature News*, 533(7604):452, 2016.

[4] Pierre Stock. *Efficiency and redundancy in deep learning models: Theoretical considerations and practical applications*. PhD thesis, École Normale Supérieure de Lyon, 2021.

[5] Maxime Oquab, Pierre Stock, Daniel Haziza, Tao Xu, Peizhao Zhang, Onur Celebi, Yana Hasson, Patrick Labatut, Bobo Bose-Kolanu, Thibault Peyronel, et al. Low bandwidth video-chat compression using deep generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2388–2397, 2021.

[6] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[7] Babajide O Ayinde and Jacek M Zurada. Nonredundant sparse feature extraction using autoencoders with receptive fields clustering. *Neural Networks*, 93:99–109, 2017.

[8] Babajide O Ayinde, Tamer Inanc, and Jacek M Zurada. Redundant feature pruning for accelerated inference in deep neural networks. *Neural Networks*, 118:148–158, 2019.

[9] Babajide O Ayinde, Tamer Inanc, and Jacek M Zurada. Regularizing deep neural networks by enhancing diversity in feature extraction. *IEEE transactions on neural networks and learning systems*, 30(9):2650–2661, 2019.

[10] Soravit Changpinyo, Mark Sandler, and Andrey Zhmoginov. The power of sparsity in convolutional neural networks. *arXiv preprint arXiv:1702.06257*, 2017.

[11] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. Predicting parameters in deep learning. *Advances in neural information processing systems*, 26, 2013.

[12] Alexandre Kabbach, Corentin Ribeyre, and Aurélie Herbelot. Butterfly effects in frame semantic parsing: impact of data processing on model ranking. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3158–3169, 2018.

[13] Kevin Musgrave, Serge Belongie, and Ser-Nam Lim. A metric learning reality check. In *European Conference on Computer Vision*, pages 681–699. Springer, 2020.

[14] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[15] OpenAI. URL https://openai.com/. Online. [Accessed: April 26, 2022].

[16] Roee Aharoni, Melvin Johnson, and Orhan Firat. Massively multilingual neural machine translation. 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 3874–3884, 2019.

[17] Forest Agostinelli, Stephen McAleer, Alexander Shmakov, and Pierre Baldi. Solving the rubik's cube with deep reinforcement learning and search. *Nature Machine Intelligence*, 1 (8):356–363, 2019.

[18] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.

[19] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2018.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[21] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for modern deep learning research. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13693–13696, 2020.

[22] Rongcheng Lin, Jing Xiao, and Jianping Fan. NeXtVLAD: An Efficient Neural Network to Aggregate Frame-level Features for Large-scale Video Classification. In *Proceedings of the European Conference on Computer Vision Workshops*, pages 206–218. Springer, 2018. URL https://openaccess.thecvf.com/content_ECCVW_2018/papers/11132/Lin_NeXtVLAD_An_Efficient_Neural_Network_to_Aggregate_Frame-level_Features_for_ECCVW_2018_paper.pdf.

[23] Hankyol Lee, Youngjae Yu, and Gunhee Kim. Augmenting Data for Sarcasm Detection with Unlabeled Conversation Context. In *Proceedings of the Workshop on Figurative Language Processing*, pages 12–17. Association for Computational Linguistics, 2020. URL https://aclanthology.org/2020.figlang-1.2.pdf.

[24] Mahdi Soltanolkotabi, Adel Javanmard, and Jason D Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 65(2):742–769, 2018.

[25] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[26] Nils Reimers and Iryna Gurevych. Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks. *arXiv preprint arXiv:1707.06799*, 2017. URL https://arxiv.org/pdf/1707.06799.pdf.

[27] Wael Etaiwi and Ghazi Naymat. The Impact of applying Different Preprocessing Steps on Review Spam Detection. *Procedia Computer Science*, 113:273–279, 2017.

[28] Jose Camacho-Collados and Mohammad Taher Pilehvar. On the Role of Text Preprocessing in Neural Network Architectures: An Evaluation Study on Text Categorization and Sentiment Analysis. In *Proceedings of the Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 40–46. Association for Computational Linguistics, 2018. URL https://aclanthology.org/W18-5406.pdf.

[29] Susan Borowski. The origin and popular use of occam's razor. *Sciencia. url: https://www. aaas. org/origin-and-popular-use-occams-razor*, 2012.

[30] Trevor Hastie, Robert Tibshirani, and Wainwright Martin. Statistical learning with sparsity: The lasso and generalizations, 2016.

[31] Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. ICML Workshop on Challenges in Deploying and monitoring Machine Learning Systems, July 2020. arXiv:2007.03051.

[32] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1355. URL https://aclanthology.org/P19-1355.

[33] Neil C Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F Manso. The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*, 2020.

[34] Second Workshop on Simple and Efficient Natural Language Processing, Nov 2021. URL https://sites.google.com/view/sustainlp2021/home. Online. [Accessed: April 26, 2022].

[35] Debanjan Ghosh, Avijit Vajpayee, and Smaranda Muresan. A Report on the 2020 Sarcasm Detection Shared Task. In *Proceedings of the Workshop on Figurative Language Processing*, pages 1–11. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.figlang-1.1.

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

[37] Ivan Bratko. Machine learning: Between accuracy and interpretability. In *Learning, networks and statistics*, pages 163–177. Springer, 1997.

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

[39] Jan Van Leeuwen. On the construction of huffman trees. In *ICALP*, pages 382–410, 1976.

[40] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[41] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

[42] Ilija Radosavovic, Piotr Dollár, Ross Girshick, Georgia Gkioxari, and Kaiming He. Data distillation: Towards omni-supervised learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4119–4128, 2018.

[43] Lin Miao, Mark Last, and Marina Litvak. Twitter data augmentation for monitoring public opinion on covid-19 intervention measures. In *Proceedings of the 1st Workshop on NLP for COVID-19 (Part 2) at EMNLP 2020*, 2020.

[44] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10687–10698, 2020.

[45] Sajjad Abbasi, Mohsen Hajabdollahi, Nader Karimi, and Shadrokh Samavi. Modeling teacher-student techniques in deep neural networks for knowledge distillation. In *2020 International Conference on Machine Vision and Image Processing (MVIP)*, pages 1–6. IEEE, 2020.

[46] Bowen Pan, Rameswar Panda, Yifan Jiang, Zhangyang Wang, Rogerio Feris, and Aude Oliva. IA-RED^2: Interpretability-Aware Redundancy Reduction for Vision Transformers. *Advances in Neural Information Processing Systems*, 34, 2021.

[47] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8817–8826, 2018.

[48] Xin Wang, Fisher Yu, Zi-Yi Dou, Trevor Darrell, and Joseph E Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 409–424, 2018.

[49] Yuchao Li, Shaohui Lin, Baochang Zhang, Jianzhuang Liu, David Doermann, Yongjian Wu, Feiyue Huang, and Rongrong Ji. Exploiting kernel sparsity and entropy for interpretable cnn compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2800–2809, 2019.

[50] Gino Brunner, Yang Liu, Damian Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. On identifiability in transformers. In *International Conference on Learning Representations*, 2019.

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

[52] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014.

[53] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *TACL*, 5:135–146, 2017. ISSN 2307-387X.

[54] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[55] John R Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, 1957.

[56] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[57] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015.

[58] Martin Müller, Marcel Salathé, and Per E Kummervold. COVID-Twitter-BERT: A Natural Language Processing Model to Analyse COVID-19 Content on Twitter. *arXiv preprint arXiv:2005.07503*, 2020. URL https://arxiv.org/pdf/2005.07503v1.pdf.

[59] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[60] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[61] Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. BERTweet: A pre-trained language model for english tweets. In *EMNLP: System Demonstrations*, pages 9–14, 2020.

[62] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending Against Neural Fake News. In *NIPS*, pages 9054–9065, 2019.

[63] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners.

[64] Josef Sivic and Andrew Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 1470–1477. Institute of Electrical and Electronics Engineers, 2003. URL https://ieeexplore.ieee.org/document/1238663.

[65] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating Local Descriptors into a Compact Image Representation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3304–3311. Institute of Electrical and Electronics Engineers, 2010. URL https://ieeexplore.ieee.org/document/5540039.

[66] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. NetVLAD: CNN architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307. Institute of

Electrical and Electronics Engineers, 2016. URL https://openaccess.thecvf.com/content_cvpr_2016/papers/Arandjelovic_NetVLAD_CNN_Architecture_CVPR_2016_paper.pdf.

[67] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2): 91–110, 2004.

[68] Tony Veale. Creative Language Retrieval: A Robust Hybrid of Information Retrieval and Linguistic Creativity. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 278–287. Association for Computational Linguistics, 2011. URL https://aclanthology.org/P11-1029.

[69] Polina Kuznetsova, Jianfu Chen, and Yejin Choi. Understanding and Quantifying Creativity in Lexical Composition. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1246–1258. Association for Computational Linguistics, 2013. URL https://aclanthology.org/D13-1124.

[70] Antonio Reyes, Paolo Rosso, and Tony Veale. A multidimensional approach for detecting irony in Twitter. In *Language Resources and Evaluation*, volume 47, pages 239–268. Springer Nature, 2013. URL https://link.springer.com/article/10.1007/s10579-012-9196-x.

[71] Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. Sarcasm as Contrast between a Positive Sentiment and Negative Situation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 704–714. Association for Computational Linguistics, 2013. URL https://aclanthology.org/D13-1066.

[72] William Empson. *Seven Types of Ambiguity*. Chatto and Windus, London, 2nd edition, 1947.

[73] Elisabeth Camp. Sarcasm, Pretense, and The Semantics/Pragmatics Distinction. *Noûs*, 46 (4):587–634, 2012.

[74] Aniruddha Ghosh, Guofu Li, Tony Veale, Paolo Rosso, Ekaterina Shutova, John Barnden, and Antonio Reyes. SemEval-2015 Task 11: Sentiment Analysis of Figurative Language in Twitter. In *Proceedings of the International Workshop on Semantic Evaluation*, pages 470–478, 2015. URL https://aclanthology.org/S15-2080.pdf.

[75] Diana G Maynard and Mark A Greenwood. Who cares about sarcastic tweets? Investigating the impact of sarcasm on sentiment analysis. In *Proceedings of the Language Resources and Evaluation Conference*. ELRA, 2014.

[76] Claire Colebrook et al. *Irony*. Psychology Press, 2004.

[77] Christopher J Lee and Albert N Katz. The Differential Role of Ridicule in Sarcasm and Irony. *Metaphor and Symbol*, 13(1):1–15, 1998.

[78] Salvatore Attardo. Irony as relevant inappropriateness. *Journal of Pragmatics*, 32(6):793–826, 2000.

[79] Laura Alba-Juez and Salvatore Attardo. The evaluative palette of verbal irony. *Evaluation in Context*, 242, 2014.

[80] Aniruddha Ghosh and Tony Veale. Fracking Sarcasm using Neural Network. In *Proceedings of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 161–169. Association for Computational Linguistics, 2016. URL https://aclanthology.org/W16-0425.pdf.

[81] Cynthia Van Hee, Els Lefever, and Véronique Hoste. Semeval-2018 task 3: Irony detection in english tweets. In *SemEval*, pages 39–50, 2018.

[82] Omid Rohanian, Shiva Taslimipoor, Richard Evans, and Ruslan Mitkov. Wlv at semeval-2018 task 3: Dissecting tweets in search of irony. In *SemEval*, pages 553–559, 2018.

[83] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[84] Rolandos Alexandros Potamias, Georgios Siolas, and Andreas-Georgios Stafylopatis. A transformer-based approach to irony and sarcasm detection. *Neural Computing and Applications*, pages 1–12, 2020.

[85] Aniruddha Ghosh and Tony Veale. Ironymagnet at semeval-2018 task 3: A siamese network for irony detection in social media. In *SemEval*, pages 570–575, 2018.

[86] Davide Chicco. Siamese neural networks: An overview. *Artificial Neural Networks*, pages 73–94, 2020.

[87] Chuhan Wu, Fangzhao Wu, Sixing Wu, Junxin Liu, Zhigang Yuan, and Yongfeng Huang. Thu_ngn at semeval-2018 task 3: Tweet irony detection with densely connected lstm and multi-task learning. In *SemEval*, pages 51–56, 2018.

[88] Himani Srivastava, Vaibhav Varshney, Surabhi Kumari, and Saurabh Srivastava. A Novel Hierarchical BERT Architecture for Sarcasm Detection. In *FigLang*, pages 93–97, 2020.

[89] Nikhil Jaiswal. Neural sarcasm detection using conversation context. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 77–82, 2020.

[90] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *NAACL*, 2018.

[91] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.

[92] Xiangjue Dong, Changmao Li, and Jinho D Choi. Transformer-based context-aware sarcasm detection in conversation threads from social media. *ACL 2020*, page 276, 2020.

[93] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2019.

[94] Tanvi Dadu and Kartikey Pant. Sarcasm detection using context separators in online discourse. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 51–55, 2020.

[95] Yoon Kim. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751. Association for Computational Linguistics, 2014. URL https://aclanthology.org/D14-1181.pdf.

[96] V. Fomin, J. Anmol, S. Desroziers, J. Kriss, and A. Tejani. High-level library to help with training neural networks in PyTorch. https://github.com/pytorch/ignite, 2020.

[97] Eiman Alothali, Nazar Zaki, Elfadil A Mohamed, and Hany Alashwal. Detecting Social Bots on Twitter: A Literature Review. In *IIT*, pages 175–180, 2018.

[98] J. Clement. Twitter: monthly active users worldwide, Aug 2019. URL https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/. Online. [Accessed: April 26, 2022].

[99] Twitter. Q4 and Fiscal Year 2020 Letter to Shareholders, Feb 2021. URL https://s22.q4cdn.com/826641620/files/doc_financials/2020/q4/FINAL-Q4'20-TWTR-Shareholder-Letter.pdf. Online. [Accessed: April 26, 2022].

[100] Onur Varol, Emilio Ferrara, Clayton Davis, Filippo Menczer, and Alessandro Flammini. Online Human-Bot Interactions: Detection, Estimation, and Characterization. In *ICWSM*, pages 280–289, 2017.

[101] Norah Abokhodair, Daisy Yoo, and David W McDonald. Dissecting a Social Botnet: Growth, Content and Influence in Twitter. In *CSCW*, pages 839–851, 2015.

[102] Daniel Gayo-Avello. Social Media Won't Free Us. *IEEE Internet Computing*, 21(4):98–101, 2017.

[103] Tiziano Fagni, Fabrizio Falchi, Margherita Gambini, Antonio Martella, and Maurizio Tesconi. TweepFake: About detecting deepfake tweets. *Plos one*, 16(5):e0251415, 2021.

[104] David Ifeoluwa Adelani, Haotian Mai, Fuming Fang, Huy H Nguyen, Junichi Yamagishi, and Isao Echizen. Generating Sentiment-Preserving Fake Online Reviews Using Neural Language Models and Their Human- and Machine-based Detection. In *AINA*, pages 1341–1354, 2020.

[105] Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. Automatic Detection of Generated Text is Easiest when Humans are Fooled. In *ACL*, pages 1808–1822, 2020.

[106] Jinyuan Jia, Binghui Wang, and Neil Zhenqiang Gong. Random Walk Based Fake Account Detection in Online Social Networks. In *DSN*, pages 273–284, 2017.

[107] Zi Chu, Steven Gianvecchio, Haining Wang, and Sushil Jajodia. Detecting Automation of Twitter Accounts: Are you a Human, Bot, or Cyborg? *TDSC*, 9(6):811–824, 2012.

[108] Stefano Cresci, Roberto Di Pietro, Marinella Petrocchi, Angelo Spognardi, and Maurizio Tesconi. The Paradigm-Shift of Social Spambots: Evidence, Theories, and Tools for the Arms Race. In *WWW Companion*, pages 963–972, 2017.

[109] Sneha Kudugunta and Emilio Ferrara. Deep neural networks for bot detection. *Information Sciences*, 467:312–322, 2018.

[110] Maryam Heidari and James H Jones. Using BERT to Extract Topic-Independent Sentiment Features for Social Media Bot Detection. In *UEMCON*, pages 0542–0547, 2020.

[111] Nikan Chavoshi, Hossein Hamooni, and Abdullah Mueen. DeBot: Twitter Bot Detection via Warped Correlation. In *ICDM*, pages 817–822, 2016.

[112] Dieudonne Mulamba, Indrajit Ray, and Indrakshi Ray. On sybil classification in online social networks using only structural features. In *PST*, pages 1–10. IEEE, 2018.

[113] Dieudonné Mulamba, Indrajit Ray, and Indrakshi Ray. Sybilradar: A graph-structure based framework for sybil detection in on-line social networks. In Jaap-Henk Hoepman and Stefan Katzenbeisser, editors, *IFIP SEC*, pages 179–193. Springer International Publishing, 2016. ISBN 978-3-319-33630-5.

[114] David Dukić, Dominik Keča, and Dominik Stipić. Are You Human? Detecting Bots on Twitter Using BERT. In *DSAA*, pages 631–636, 2020.

[115] Francisco Rangel and Paolo Rosso. Overview of the 7th Author Profiling Task at PAN 2019: Bots and Gender Profiling in Twitter. In *CEUR Workshop*, pages 1–36, 2019.

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