Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Neural Network Architectures for Adversarially Robust NLP

Author:
Frank Hua

Supervisor:
Prof. Alessio Lomuscio
Dr. Alessandro De Palma

Submitted in partial fulfillment of the requirements for the MSc degree in Advanced Computing of Imperial College London

September 2023

Abstract

In this study, we explored the influence of transformer architecture design on adversarial robustness in Natural Language Processing (NLP), particularly focusing on the self-attention mechanisms. While transformers have demonstrated superior performance compared to other architectures such as LSTMs, their robustness under adversarial attacks requires further investigation. We address this by conducting experiments and analyses of various self-attention mechanisms within transformers and their impact on robustness.

Our experiments include six distinct adversarial attacks in sentiment analysis, revealing that transformers outperform LSTMs in robustness. We then implement and conduct experiments over multiple self-attention variants, including CosFormer, TransNormer, and Diag Attention. Furthermore, we conduct ablation studies on different components within the transformer. Our experiments have revealed that the number of heads is a significant factor influencing the adversarial robustness of transformers. Additionally, we observed that counter-fitted word embeddings may enhance the adversarial robustness of transformers compared to GloVe embeddings and custom trainable embeddings.

Inspired by the results of the empirical analysis, We use counter-fitted word embeddings to enhance the adversarial robustness of our models. We then propose a simple extension to the Scaled Dot-Product Attention called ReLU Value Attention (ReVA). Furthermore, the integration of ReLU Value Attention with CosFormer yields ReLU Value CosFormer (ReVCos). With similar model capacities, the ReVCos transformer demonstrates a better robustness accuracy trade-off compared to the baseline vanilla transformer. Our study provides insights to future research that progress towards a more robust and resilient transformer architecture against adversarial attacks in the NLP domain.

Acknowledgments

I would like to express my sincere gratitude to Dr. Alessandro De Palma for his continuous support throughout the past six months. His mindful guidance and insightful ideas were critical to the success of this project. Alessandro not only taught me how to complete this Master's thesis but also provided invaluable lessons on how to conduct proper research. I am immensely thankful to him for making this journey enjoyable and enriching.

I would also like to express my gratitude to the entire VAS team, particularly Prof. Alessio Lomuscio and Mr. Edward Stevinson, for sharing their valuable advice on this project.

A special thanks to Lily, who spent much effort on our move, especially during my busiest periods of writing this thesis. Her kindness and encouragement played a pivotal role in helping me stay focused during the final stages.

Lastly, I want to thank The Smiths. While I was "Panic on the streets of London", their music provided the necessary faith for many long and productive nights dedicated to this thesis.

Contents

Chapter 1

Introduction

Deep learning has achieved tremendous success across a wide range of applications in Natural Language Processing (NLP). In recent years, transformers [1] has emerged as the preferred architecture for most NLP tasks, including sentiment analysis and machine translation. It has been shown that transformers, also known as self-attentive neural models, exhibit superior performance compared to other simpler architectures such as LSTM [2, 3]. These models, including the original transformer, Bidirectional Encoder Representations from Transformers (BERT) [4], and Generative Pre-trained Transformer (GPT) [5], use the attention mechanism to learn context-dependent representations. Transformers outperform Recurrent Neural Network (RNN) models in terms of parallelizability and generalizability, particularly in complex tasks like machine translation [1].

Despite the success of deep learning, the existence of adversarial vulnerabilities indicates frailties in existing models and prevents them from being deployed in safety-critical contexts [6]. Moreover, transformers remain vulnerable to various adversarial attacks proposed in the NLP domain [7, 8, 9, 10, 11]. While the investigation carried out by Hsieh et al. [12] highlights the superior adversarial robustness of transformers over LSTMs, their experiments present certain limitations, primarily in the absence of comprehensive information regarding their model capacity and training configurations. Transformers such as BERT are large and typically pre-trained on very large corpora with multi-GPUs. As model capacity proved to be an important factor in adversarial robustness[13, 14, 15, 16], the fairness of this experiment remains questionable.

We are curious if the transformer design still can exceed LSTM's robustness per-

formance under a comparable setting in model capacities. The primary objective of this project is to explore the literature on transformer architecture design and aim to identify variants of transformers that improve the trade-offs between adversarial robustness and standard accuracy, while maintaining a comparable model capacity. Specifically, we will evaluate the adversarial robustness of different novel self-attention mechanisms that exist in the transformer literature against a variety of adversarial attacks, while subject to different training schemes.

To the best of our knowledge, this project brings the following contributions:

- We conduct experiments comparing the adversarial robustness of LSTMs and transformers under six different adversarial attacks in sentiment analysis. The experimental results show that the transformer is more robust than LSTM, despite both models are still very vulnerable under adversarial attacks.
- We implement and evaluate multiple variations of the self-attention mechanism drawn from existing literature.
- We conduct ablation studies which demonstrate certain transformer design choices, such as the choice of word embedding, have a relatively large impact on the robustness of the transformer model.
- Inspired by the results of the empirical analysis, we propose a simple extension to the Scaled Dot-Product Attention called ReLU Value Attention (ReVA). Furthermore, the integration of ReVA with CosFormer yields ReLU Value CosFormer (ReVCos), which is a simple but effective technique that is able to benefit the robustness-accuracy trade-off of the transformer architecture compared to the vanilla transformer.

Chapter 2

Background

This chapter provides a foundational understanding of the project's landscape. It covers neural architectures like LSTM and Vanilla Transformer, word embedding techniques, and introduces adversarial attacks. We then explore various NLP-specific adversarial attacks and conclude by examining the concept of adversarial training as a potential defence strategy. This knowledge forms the basis for our subsequent study on enhancing the transformer's resilience against adversarial challenges.

2.1 RNN and LSTM

Sequential data poses challenges for traditional neural networks due to their inability to retain context across distant time steps. A Recurrent Neural Network (RNN) [17] is a type of neural network designed for processing sequential data by maintaining a hidden state that allows information to persist over time steps.
Given input sequence $x = (x_1, x_2, ..., x_T)$, the hidden state h_t at time t is computed as:

$$h_t = \sigma(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \tag{2.1}$$

The output y_t at time t is computed using the hidden state:

$$y_t = W_{yh}h_t + b_y \tag{2.2}$$

Traditional recurrent neural networks often suffer from the vanishing gradient problem[3]. LSTM[2] was introduced to mitigate these challenges by introducing specialized memory cells and gating mechanisms that facilitate the learning of temporal dependencies.

LSTM can be formulated as:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \tag{2.3}$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \tag{2.4}$$
$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tanh(W_{ic}x_t + b_{ic} + W_{hc}h_{t-1} + b_{hc}) \tag{2.5}$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \tag{2.6}$$
$$h_t = o_t \cdot \tanh(C_t) \tag{2.7}$$

Where x_t is the input at time t, h_{t-1} is the previous hidden state, W and b are weight matrices and bias vectors, respectively. Additionally, σ represents the sigmoid activation function, and $\tanh$ is the hyperbolic tangent activation function.

During each time step, the LSTM processes the input along with the previous hidden state and cell state. The forget gate, input gate, and output gate are determined by learned weights and activation functions, which enable the model to adaptively control the flow of information. LSTM updates hidden states and memory cells sequentially, making it inherently sequential.

Bi-directional LSTM models [18] have gained prominence in handling sequential data. Unlike traditional LSTM models, which process data in a strictly chronological order, bi-directional LSTMs process the data in both forward and backward directions. This enables the model to capture information from past and future contexts simultaneously, allowing for a more comprehensive understanding of the sequence. This is particularly beneficial in tasks where contextual information from both directions is crucial, such as machine translation or sentiment analysis. The final output at time t for the Bi-LSTM can be obtained by concatenating the forward and backward hidden states:

$$h_t^{(\text{Bi-LSTM})} = [h_t^{(\text{forward})}, h_t^{(\text{backward})}] \tag{2.8}$$

Despite its successes, LSTM does have limitations. It has the ability to capture short-term and long-term dependencies within a sequence but it still can struggle with very long paragraphs due to its sequential nature and vanishing gradient problems.

2.2 Vanilla Transformer

Transformers have shown state-of-the-art performance across a variety of NLP tasks such as machine translation [1], question answering [4], and text classification [5].

The original transformer, often referred to as the vanilla transformer, demonstrates its superiority over other architectures through its combination of several key components including the self-attention mechanism. In addition, transformers can process all positions in a sequence simultaneously, enabling more efficient parallelization during training and inference[1]. Both the transformer encoder and decoder can be employed either individually or in combination, depending on the specific requirements of the task. The overall diagram of the transformer architecture can be found in Figure 2.1.

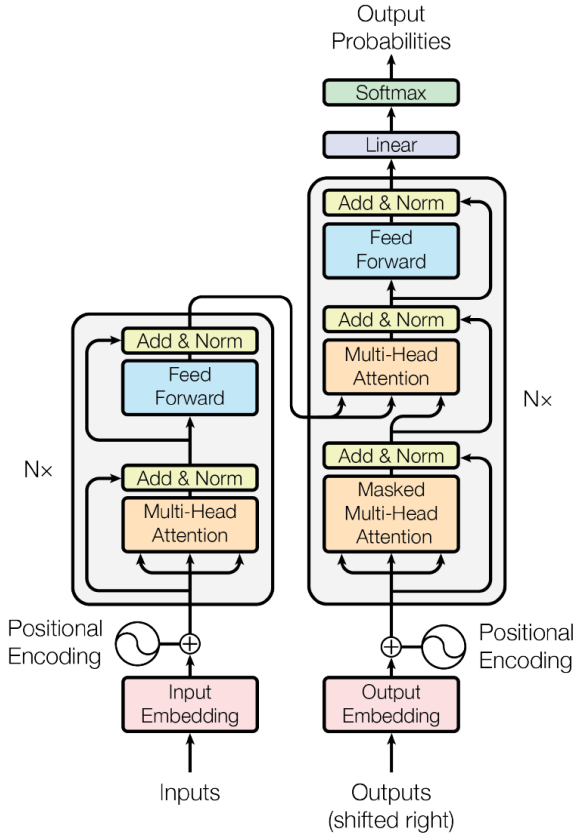

Figure 2.1: Transformer architecture, with encoder on the left and decoder on the right [1]

2.2.1 Scaled Dot-Product Attention

Traditional Seq2Seq models, which often rely on recurrent neural networks (RNNs), suffer from limitations when dealing with long sequences. Transformers, on the other hand, discard sequence-to-sequence operations and rely on the self-attention mechanism [1]. In the vanilla transformer, self-attention is formulated as the Scaled Dot-Product Attention. This mechanism looks at the surrounding words in sequences

to obtain more contextually sensitive word representations by the use of dot-product. Given a sequence of n d_h-dimensional vectors $\mathbf{x} =\in \mathbb{R}^{n\times d_h}$, and a query vector $\mathbf{q} \in \mathbb{R}^{n\times d_h}$, the attention layer parameterized by $W_k, W_q, W_v, W_o \in \mathbb{R}^{d_h\times d_h}$ computes the weighted sum as:

$$Q = \mathbf{q}W_q, K = \mathbf{x}W_k, V = \mathbf{x}W_v \tag{2.9}$$

$$\text{Attention}_{W_k,W_q,W_v,W_o}(\mathbf{x}, \mathbf{q}) = W_o \left(\text{softmax}\left(\frac{QK^\top}{\sqrt{d_h}}\right)V\right) \tag{2.10}$$

Scaled Dot-Product Attention is also referred to as vanilla attention, as in vanilla transformer from Vaswani et al. [1]. The attention mechanism used in Transformers allows them to weigh the importance of different words in a sequence when processing each word. This means that each word can directly consider and attend to all other words, regardless of their distance in the sequence. A diagram of Scaled Dot-Product Attention can be found in Figure 2.2.

2.2.2 Self-Attention vs Cross-Attention

The transformer architecture can be viewed as a collection of multiple transformer encoder blocks and/or decoder blocks stacked together. Both the encoder and the decoder can be used independently or jointly with the other. There exists a distinction between the encoder and decoder in terms of how they calculate their attention scores: the encoder uses self-attention while the decoder uses cross-attention.

Self-attention(transformer encoder) focuses on capturing the relationships between different positions in a single sequence. Therefore the query, key, and value are all derived from the same input sequence, enabling the model to assess the significance of each element with respect to others. In other words, $\mathbf{x}$ also serves as the query $\mathbf{q}$ to calculate a new sequence of representations. Self-attention is sometimes also referred to as casual attention.

Conversely, in cross-attention (transformer decoder), the relationships between different sequences are modelled. This mechanism has significant importance in tasks that involve multiple input sequences or those that require relating information from diverse sources. In cross-attention, queries are derived from one sequence, while keys and values are obtained from another sequence. This configuration enables the model to focus on specific parts of one sequence while considering the entire context of the other. $\mathbf{q}$ is usually a decoder state, while $\mathbf{x}$ corresponds to the encoder output.

2.2.3 Multi-Head Attention

Multi-head Attention(MHA), in addition to Scaled Dot-Product Attention, enables the execution of independently parameterized attention layers in parallel to obtain the final attention result. MHA is usually formulated as:

$$\text{MHAttention}(\mathbf{x}, \mathbf{q}) = \sum_{h=1}^{N_h} \text{Attention}(W_k^h, W_q^h, W_v^h, W_o^h)(\mathbf{x}, \mathbf{q}) \tag{2.11}$$

or

$$\text{MHAttention}(\mathbf{x}, \mathbf{q}) = \text{Concatenate}_{h=1}^{N}(\text{Attention}(W_k^h, W_q^h, W_v^h, W_o^h)(\mathbf{x}, \mathbf{q})) \tag{2.12}$$

where $W_k^h, W_q^h, W_v^h \in \mathbb{R}^{d_h \times d}$ and $W_o^h \in \mathbb{R}^{d \times d_h}$, and each weight matrix is associated with a specific head in the multi-head attention mechanism. When $d_h = d$, Multi-head Attention (MHA) is inherently more expressive than vanilla attention. However, to maintain a constant number of parameters, d_h is often set to $\frac{d}{N_h}$, where N_h represents the number of attention heads. In this case, MHA can be viewed as an ensemble of low-rank vanilla attention layers [19]. A visual representation of MHA can be found in Figure 2.2.

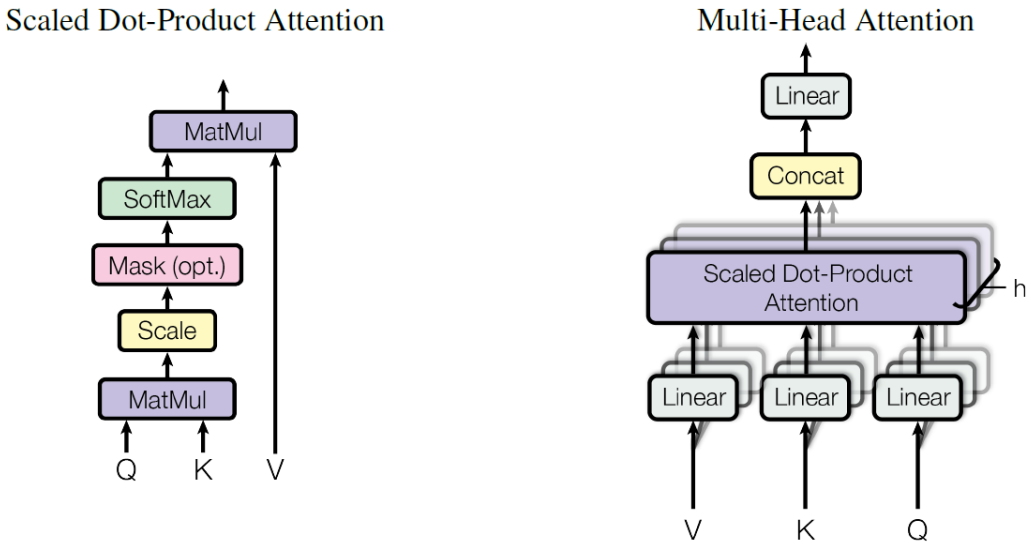

Figure 2.2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel. [1]

Attention mechanisms in their basic form have a quadratic complexity with respect to sequence length, which makes them less efficient for very long sequences. However, the scaled dot-product attention can be computed in parallel for all positions in the

sequence, making the model highly scalable.

2.2.4 Feed-Forward Layer

In addition to the attention mechanism, each sub-layer contains a fully connected feed-forward network which operates on each position separately and identically. The feed-forward layer is made of two linear transformations interleaved with a ReLU activation.

$$FFN(\mathbf{x}) = max(0, \mathbf{x}W_1 + b_1)W_2 + b_2 \tag{2.13}$$

The shapes of each component can be described as:

$$x \in n \times d$$
$$W_1 \in d \times h, b_1 \in h$$
$$W_2 \in h \times d, b_2 \in d$$

Where n is the length of inputs, d is model dimensionality, and h is the hidden size of this feed-forward network. The feed-forward network introduces additional non-linearity to the model, and it acts as a feature extractor to the previous layer and transforms them into a higher-level representation. This network gives the transformer the ability to capture different aspects of context and semantics in the given task.

2.2.5 Positional Encoding

Unlike RNNs, the attention + feed-forward layer combination does not capture any sequential information of the input. The positional encoding serves this purpose by adding position-specific information to the embeddings of tokens in the input sequence. The original transformer uses absolute positional encoding which is formulated as follows:

$$\text{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{2.14}$$
$$\text{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{2.15}$$

where pos is the position and i is the dimension.

2.2.6 Transformer Pre-training

Despite Transformer's recent success in multiple deep learning tasks, training Transformers on small datasets is notably challenging [20, 21, 22]. Consequently, when working with smaller datasets, a common practice is to use simple shallow layers with pre-trained models during the fine-tuning process. A notable benefit of using pre-trained models [4, 5] is the reduced requirement for computational resources during the fine-tuning process on smaller datasets. This approach allows the model to achieve better performance on downstream tasks by conducting fine-tuning on a single GPU.

Nevertheless, the application of pre-trained models does not align with our project's objectives. Our primary goal is to assess how the robustness of the transformer is affected by variations in architectural design. As a result, we need full control over the employed architecture. In this project, we lack the computational resources required to train any proposed architecture modifications from scratch on large language models. Modifying the internal architecture of pre-trained models or fine-tuning them with a different design wouldn't be practical either. Pre-trained models are initially trained with specific architectures, and altering this could lead to unpredictable outcomes. Moreover, distinct pre-trained models consist different number of trainable parameters, which introduces complexity in distinguishing whether better robustness or performance improvements are results of improved design or merely an increase in model capacity. Therefore, we decided not to use pre-trained models in our experiments.

2.3 Word Embedding

Word embeddings are foundational representations that capture semantic relationships and contextual information within textual data. At the input layer of a language model, words are represented as dense, continuous vectors through pre-trained word embeddings, or alternatively, they can be initialized and trained from scratch. These embeddings map words to high-dimensional vector spaces, whereas similar words are positioned closer together, capturing semantic relationships. The Transformer model takes these embeddings as input, allowing it to process and understand textual data. In this section, we will discuss some word embeddings that we use for this study.

2.3.1 Custom Trainable Embedding

One approach involves initializing an embedding model using the Transformer architecture and subsequently training the word embedding end-to-end. In this scenario, the word representations undergo refinement and adjustment over the course of training. This process has an initial dataset pre-processing step, wherein a vocabulary containing N words is constructed, with N being a pre-defined hyperparameter. This vocabulary mechanism prioritizes the N most frequent words while labelling any other words as 'unknown.' Subsequently, input sentences are translated into numeric IDs, which are then mapped to vectors via the word embedding model. These vectors undergo end-to-end training alongside the specific task.

2.3.2 GloVe Embedding

Global Vectors for Word Representation (GloVe), proposed by Pennington et al. [23], is a word embedding model that captures semantic relationships between words by leveraging global statistical information from large text corpora. Unlike Word2Vec [24], which aims to maximize the likelihood of context word prediction, GloVe emphasizes ratios of co-occurrence probabilities, which help it focus on meaningful relationships rather than raw counts. It has been proven valuable across various natural language processing tasks such as word similarity and classification [23]. GloVe embeddings capture both semantic and syntactic relationships between words, but they do not capture contextual information as they assign a fixed vector to each word.

2.3.3 Counter-fitted Embedding

Mrkšić et al. [25] introduces a novel approach to improving synonym matching in word embeddings by incorporating linguistic constraints during the training process. Word embeddings, such as Word2Vec [24] and GloVe [23], may not always align with certain linguistic constraints, such as antonymy (opposite meanings) or synonymy. Counter-fitting can be viewed as an extension to the static word embeddings such as GloVe. The core idea for counter-fitting is to "counter" the direction of the vectors that violate the constraints. For example, for antonyms, if the word vectors of two words point in similar directions, their cosine similarity should decrease; conversely, for synonyms, their similarity should increase. The counter-fitted embeddings achieve state-of-the-art performance on the [26] dataset.

	east	**expensive**	**British**
Before	west	pricey	American
	north	cheaper	Australian
	south	costly	Britain
	southeast	overpriced	European
	northeast	inexpensive	England
After	eastward	costly	Brits
	eastern	pricy	London
	easterly	overpriced	BBC
	-	pricey	UK
	-	afford	Britain

Figure 2.3: Nearest neighbours for target words using GloVe vectors before and after counter-fitting. Figure taken from Mrkšić et al. [25]

Figure 2.3 presents a comparison of the impact of counter-fitting on embedding vectors before and after the process. We can observe that prior to counter-fitting, both synonyms and antonyms are clustered together. However, following counter-fitting, only synonyms display this grouping.
Counter-fitted word embeddings are employed in numerous studies focused on the generation of adversarial attacks [27, 8, 11]. The usage of counter-fitted word embeddings as a defence mechanism is expected to enhance the robustness of the model against adversarial attacks specifically targeting synonyms. In 2019, Huang et al. [28] applied formal verification methods to text classification models against synonym and character flip perturbations, and counter-fitted embeddings were used on top of WordCNN models [29]. With this embedding, the authors observed an increase in the model-verified accuracy and a slight decrease in the standard accuracy, in standard training when compared to baseline GloVe embeddings.

2.4 Adversarial Attack

2.4.1 Overview

Recent breakthroughs in computer vision and natural language processing are bringing trained classifiers into the centre of security-critical systems. However, it's been shown that an adversary is often able to manipulate the input by a very small amount so that the network produces an incorrect output [30]. A widely recognized approach introduced by Goodfellow et al. [31] called the Fast Gradient Sign Method (FGSM) popularized this research topic. FGSM allows adversary to generate the

attack input $\hat{\boldsymbol{x}}$ by this formulation:

$$\boldsymbol{\eta} = \epsilon \cdot \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)) \tag{2.16}$$

$$\hat{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\eta} \tag{2.17}$$

where $\boldsymbol{\theta}$ is the parameters of the model, $\boldsymbol{x}$ is the original input, ϵ is the magnitude of perturbations, y the target of the model, and $J(\boldsymbol{\theta}, \boldsymbol{x}, y))$ is the cost/loss used to train the network. This simple yet effective approach has gained significant popularity in the deep learning community.
FGSM generates effective attacks by taking advantage of the way neural networks learn and make predictions. Neural networks are trained to adjust their parameters in a direction that reduces the error or loss between their predictions and the actual target values. When you calculate the gradient of the loss with respect to the input data (as done in FGSM), you're essentially finding out how sensitive the model's predictions are to changes in the input. This gradient information indicates the direction in which the loss increases the most when you perturb the input.
Following their works, many research efforts have been made to evaluate the robustness of deep neural networks by fooling them with unperceivable perturbations [32]. In Section 2.6, we will focus on the adversarial attacks designed for Natural Language Processing, rather than Computer Vision as it is not the project's primary focus.

2.4.2 Black-box vs White-box

Black-box attack and white-box attack are two different attacking approaches used to exploit vulnerabilities or manipulate the behaviour of a deep learning model [32].

- In a black-box attack, the attacker has limited or no knowledge about the internal workings of the target deep-learning model. They can only interact with the model by providing input data and observing the corresponding output predictions. The attacker doesn't have access to the model's architecture, parameters, or training data. They often rely on heuristics to exploit the model's vulnerability without any prior information. Black-box attacks are more realistic in scenarios where the attacker only has access to the model's inputs and outputs.

- In contrast to black-box attacks, white-box attacks assume the attacker has full knowledge of the target deep learning model. This includes information about

the model's architecture, parameters, gradient information, and possibly even the training data. White-box attacks can be viewed as an approximation of the worst-case attack for a particular model and input. FGSM, which uses gradient information, would be categorized as a white-box attack.

2.5 Adversarial Training

Adversarial training, a defence strategy pioneered by Szegedy et al. [30], involves training a neural network to strengthen its robustness by correctly classifying both normal examples and adversarial examples.

2.5.1 Adversarial Training as a Regularizer

Goodfellow et al. [31] employed explicit training by utilizing adversarial examples generated with FGSM. They discovered that training an adversarial objective function on FGSM can be viewed as a beneficial regularization technique. By incorporating the adversarial component, the model is encouraged to learn features that are less likely to be influenced by noise or irrelevant details in the data. This leads to more robust representations that are better suited for accurate classification. More specifically, with α as a regularization hyperparameter, the adversarial training loss function can be viewed as:

$$\hat{J}(\boldsymbol{\theta}, \boldsymbol{x}, y) = \alpha J(\boldsymbol{\theta}, \boldsymbol{x}, y) + (1 - \alpha) J(\boldsymbol{\theta}, \boldsymbol{x} + \boldsymbol{\eta}, y) \tag{2.18}$$

2.5.2 Adversarial Training as Min-Max Optimization

Another well-known study conducted by Madry et al. [13] demonstrates that a natural saddle point (min-max) formulation can capture the notion of security against adversarial attacks in a principled manner. This strategy is referred to as Min/Max Adversarial Training or Robust Optimization. Rather than incorporating perturbations through a regularization term, this approach directly integrates perturbations into the loss function through modification. Let $\boldsymbol{x}, y$ be the data samples draw from distribution D, and $\boldsymbol{\delta}$ be the attack generated in perturbation space $S \in \mathbb{R}^d$, the saddle point problem can then be formulated as:

$$\min_{\boldsymbol{\theta}} \rho(\boldsymbol{\theta}), \text{ where } \rho(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},y)\sim D} \left[\max_{\boldsymbol{\delta} \in S} J(\boldsymbol{\theta}, \boldsymbol{x} + \boldsymbol{\delta}, y) \right] \tag{2.19}$$

In particular, training with adversarial inputs directly corresponds to approximately optimizing this composition of an inner non-concave maximization problem (attack) and an outer non-convex minimization problem (defence). The outer minimization problem focuses on finding model parameters that minimize the ”adversarial loss” obtained from the inner attack problem.

They further utilized Danskin's theorem which states that gradients at inner maximizers correspond to descent directions for the min-max problem. This implies that back-propagation can still be applied to the saddle point problem. Specifically, instead of just minimizing the loss on the training examples, the model is trained to minimize the maximum possible loss that could occur under adversarial perturbations. Intuitively, robust optimization trains the model to perform well not only on clean/normal examples but also on examples that have been perturbed in ways that maximize the model's loss. This trains the model to be cautious and consider various possible perturbations during its decision-making process.

2.6 Adversarial Attacks In NLP

The attacks described thus far were developed for computer vision applications and cannot be directly applied to textual data and NLP models. The discrete nature of textual data makes it difficult to apply gradient-based methods on them [32]. When employing gradient-based adversarial attacks derived from images on these representations, the resulting adversarial examples consist of invalid characters or word sequences [33]. Another difference is that image perturbations involve subtle changes in pixel values that are difficult for humans to perceive, enabling humans to correctly classify the images and highlighting the weak robustness of deep neural models. Conversely, in the case of adversarial attacks on text, even small perturbations on characters are usually noticeable.

Despite the aforementioned challenges, Jia and Liang [34] published the first adversarial example generation method on textual DNNs. Since their groundbreaking work, numerous adversarial attack generation and defence methods have been proposed in the field of NLP. The majority of these approaches rely on various forms of search algorithms. Now we will introduce several adversarial attacks in the field of NLP that are relevant to our experimental evaluation.

2.6.1 Deepwordbug

Deepwordbug [35] was one of the earliest black-box adversarial attacks that operated on texts. It focuses entirely on character-level transformations as opposed to word-level transformations, utilizing the Levenshtein Edit Distance as a constraint. This method creates adversarial sequences by introducing minor editing operations to a text sequence, for example, character insertions and deletions. These subtle adjustments are designed to generate adversarial words that bear imperceptible differences compared to the original words. Scoring functions were designed for ranking the importance of words to perturb. Through experimentation, Deepwordbug demonstrated its capability to induce a larger reduction in accuracy across various architectures and datasets.

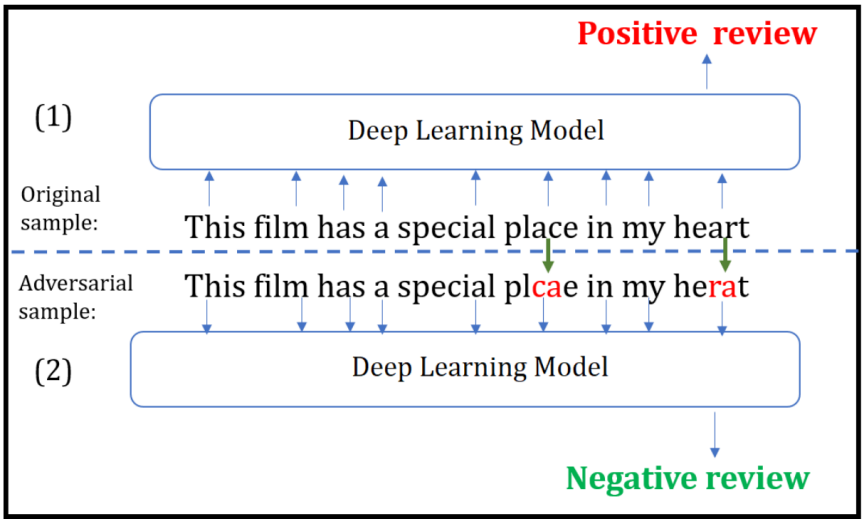

Figure 2.4: Example of a deepwordbug generated adversarial sequence [35]

2.6.2 Textbugger

Unlike previous methods, Textbugger [36] excels in terms of success rate, maintaining the intended meaning of benign text for human readers, and efficiently creating adversarial text even for long texts. Textbugger operates in both white-box and black-box settings. In the white-box scenario, it ranks crucial words by their importance using the classifier's Jacobian matrix and then perturbs them from a set of generated perturbations including multiple character-based transformations such as insertion, deletion and substitution. In the black-box scenario, the framework novelly identifies essential sentences and uses a scoring function to determine significant words for manipulation (see Figure 2.6). The introduced scoring function can accu-

rately capture the importance of words, and compute word scores without knowledge and configuration of the classification model, and it is also computationally efficient. In both white-box and black-box settings, Textbugger employs a greedy approach to search for successful adversarial examples by perturbing words with high importance scores. Textbugger can also generate adversarial examples with word synonym substitutions. A Glove embedding is used to select similar words from the vocabulary. Furthermore, it incorporates a universal sentence encoder (USE) [37] and cosine similarity scores to efficiently ensure the quality of the perturbed words. In our experiments, we used Textbugger with black-box settings. The example adversarial texts generated by Textbugger can be found in Figure 2.5.

Figure 2.5: The example character-based attacks taken from [36]

$$C_{w_j} = \mathcal{F}_y(w_1, w_2, \cdots, w_m) - \mathcal{F}_y(w_1, \cdots, w_{j-1}, w_{j+1}, \cdots, w_m)$$

Figure 2.6: The scoring function C of Textbugger, where $\mathcal{F}_y$ denotes the classification result of the classifier[36]

2.6.3 Textfooler

The focus of Textfooler [8] is on attacking text classification and textual entailment tasks, which involve determining relationships between text passages. The authors demonstrate the effectiveness of Textfooler in generating adversarial text that can

deceive state-of-the-art models. The accuracies after attacks are reduced to around 10%, across multiple datasets (see Figure 2.8). This includes powerful pre-trained models like BERT, as well as convolutional and recurrent neural networks. The approach of Textfooler involves identifying important words for the target model and then systematically replacing them with words that are both semantically similar and grammatically correct. This replacement process continues until the model's prediction is altered. Textfooler further modified the word importance score calculation. In essence, the scoring function quantifies how substituting or removing a word affects the classifier's confidence in the original label Y and potentially other labels as well. This information can be used to select word replacements that are more likely to fool the classifier. With this scoring function, Textfooler uses the same greedy algorithm from Textbugger to search for successful attacks. Unlike the Glove embedding in Textbugger, Textfooler relies on pre-trained counter-fitting word embeddings [25] for adversarial synonym generation. USE [37] is also employed to handle similarity constraints, enabling the generation of effective and efficient attacks on classification and entailment tasks. Some evaluation results of Textfooler can be found in Figure 2.8 and some examples of attacks can be found in Figure 2.7.

Movie Review (Positive (POS) ↔ Negative (NEG))	
Original (Label: NEG)	The characters, cast in impossibly ***contrived situations***, are ***totally*** estranged from reality.
Attack (Label: POS)	The characters, cast in impossibly ***engineered circumstances***, are ***fully*** estranged from reality.
Original (Label: POS)	It cuts to the ***knot*** of what it actually means to face your ***scares***, and to ride the ***overwhelming*** **metaphorical wave** that life wherever it takes you.
Attack (Label: NEG)	It cuts to the ***core*** of what it actually means to face your ***fears***, and to ride the ***big*** **metaphorical wave** that life wherever it takes you.
SNLI (Entailment (ENT), Neutral (NEU), Contradiction (CON))	
Premise	Two small boys in blue soccer uniforms use a wooden set of steps to wash their hands.
Original (Label: CON)	The boys are in band ***uniforms***.
Adversary (Label: ENT)	The boys are in band ***garment***.
Premise	A child with wet hair is holding a butterfly decorated beach ball.
Original (Label: NEU)	The ***child*** is at the ***beach***.
Adversary (Label: ENT)	The ***youngster*** is at the ***shore***.

Figure 2.7: Example attacked texts by Textfooler on text classification datasets, taken from [8]

	WordCNN					**WordLSTM**					**BERT**				
	MR	**IMDB**	**Yelp**	**AG**	**Fake**	**MR**	**IMDB**	**Yelp**	**AG**	**Fake**	**MR**	**IMDB**	**Yelp**	**AG**	**Fake**
Original Accuracy	78.0	89.2	93.8	91.5	96.7	80.7	89.8	96.0	91.3	94.0	86.0	90.9	97.0	94.2	97.8
After-Attack Accuracy	2.8	0.0	1.1	1.5	15.9	3.1	0.3	2.1	3.8	16.4	11.5	13.6	6.6	12.5	19.3
% Perturbed Words	14.3	3.5	8.3	15.2	11.0	14.9	5.1	10.6	18.6	10.1	16.7	6.1	13.9	22.0	11.7
Semantic Similarity	0.68	0.89	0.82	0.76	0.82	0.67	0.87	0.79	0.63	0.80	0.65	0.86	0.74	0.57	0.76
Query Number	123	524	487	228	3367	126	666	629	273	3343	166	1134	827	357	4403
Average Text Length	20	215	152	43	885	20	215	152	43	885	20	215	152	43	885

Figure 2.8: Automatic evaluation results of Textfooler on text classification datasets, taken from [8]

2.6.4 Pwws

Pwws [9] introduces a new approach called Probability Weighted Word Saliency (Pwws). Similar to Textfooler, Pwws is based on a greedy synonym substitution strategy, but it also introduces a novel word replacement order determined by both the saliency of the word and its classification probability. It employs a weight-saliency map to rank the significance of word perturbations prior to launching attacks.
The Pwws algorithm involves two key strategies: Word Substitution Strategy and Replacement Order Strategy.

1. Word Substitution Strategy: For each word w_i in the original input text x, the algorithm aims to replace it with a synonym or alternative word that maximizes the change in the classification probability. L denotes a synonym set of the current word. w' denotes the perturbed word and x' denotes the perturbed input text. The selection of the substitute word w_i^* is based on a score calculated using the difference in classification probabilities before and after substitution.

$$w_i^* = \arg\max_{w_i' \in L_i} [P(y_{\text{true}}|x) - P(y_{\text{true}}|x_i')]$$

2. Replacement Order Strategy: Words that have a higher impact on the classifier's output have a higher "saliency." The algorithm uses a word's saliency value, calculated as the difference in the output probability when the word is replaced with an unknown token, to determine the replacement order. Words with higher saliency values are considered more important for replacement.

$$H(x, x_i^*, w_i) = \phi(S(x))_i \cdot \Delta P_i^*$$

Where $\phi(\cdot)$ is a softmax applied to the word saliency vector, $S(\cdot)$ is the word saliency vector, and ΔP_i^* denotes the change in classification probability for the i th word substitution.

2.6.5 Bae

Bae [10] short for Bert-based Adversarial Examples, leverages a Bert masked language model (MLM) [4] to make contextual perturbations in the input text. The authors address limitations in previous approaches that use rule-based synonym replacement strategies, which often result in unnatural and easily detectable changes. Bae operates by replacing and inserting tokens in the original text using a Bert MLM. The technique involves masking a portion of the input text and then using the Bert

MLM to suggest alternative tokens for the masked positions. The paper emphasizes that Bae generates adversarial examples with improved grammaticality and semantic coherence in comparison to the baseline method TextFooler [8]. The authors showcase that even a small number of token replacement or insertion operations can drastically reduce the accuracy of strong classifiers like BERT by over 80% on certain datasets.

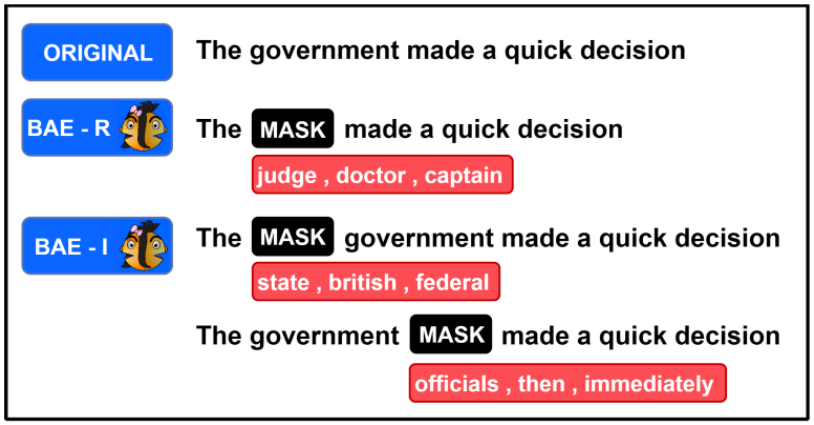

Figure 2.9: Example of how Bae perturb an input sentence, Bae-R uses word replacement and Bae-I uses word insertion [10]

2.6.6 A2t

Existing methods for generating adversarial examples in NLP often involve complex search procedures and resource-intensive sentence encoders. As a result, using adversarial training to enhance NLP models remains challenging and the benefits are not well-explored. A2t (Attacking to Training) [11] is proposed to be a simpler and more efficient adversarial training process for NLP models. Instead of relying on expensive sentence encoders and combinatorial search, A2t employs a gradient-based word importance ranking technique, which is usually cheaper than Textfooler's deletion-based ranking technique as Textfooler needs to loop over the whole input sequence. A2t also caches the top-k nearest neighbours in the counter-fitted word embedding to speed up the attacks. Due to its use of model gradient information, A2t is classified as a white-box attack. In addition, A2t uses a DistilBert semantic textual similarity model as its constraint enforcing method instead of USE [37]. It was reported DistilBert [38] only required 10 times less GPU memory than USE. Adversarial training with both A2t and its variation A2t-MLM improves the adversarial robustness of NLP models, even against types of attacks that were not part of the training process.

Chapter 3

Exploring Model Designs

While transformers are gaining much popularity in the field of Natural Language Processing and even Computer Vision, their ability to handle long sequential data has prompted extensive exploration of their capabilities and potential enhancements in efficiency and performance. As the landscape of transformer architectures diversifies, we are interested in the intricate relationship between efficiency, performance, and robustness in the world of modern neural networks. We aim to explore how these architectural advancements influence adversarial robustness and ultimately contribute to the development of a more resilient design.

In this chapter, we will discuss several relevant literature that focus on enhancing the adversarial robustness of deep neural models from the perspective of architecture design. After that, we will discuss some of the novel transformer architectures that could potentially influence robustness, specifically focusing on the modifications to the self-attention mechanism. Each of these architectural modifications aims to address specific aspects of efficiency or performance enhancement within the transformer framework.

3.1 Designing Towards Robustness

We now detail previous work investigating the relationship between architectural design and adversarial robustness. Note that most of them were solely conducted on deep neural networks for computer vision tasks.

Madry et al. [13] have found that a larger network capacity yields better trade-offs between standard performance and adversarial robustness. This indicates that

the robust decision boundary of the saddle point problem can be considerably more complex compared to a decision boundary that merely separates the benign data points. In addition to this idea, Bubeck and Sellke [14] claim that, for a wide range of data distributions and model classes, over-parametrization is necessary to achieve smooth data interpolation, which in turn guarantees the model Lipschitz properties and thus robustness. In contrast, Huang et al. [15] demonstrate that neural networks with larger capacity and excessive parameters do not necessarily lead to improved robustness. Wu et al. [16] supports the findings of Huang regarding the impact of width on robustness. They suggest that wider networks can achieve better natural accuracy but potentially worse overall model robustness due to decreasing perturbation stability. A recent work done by Zhu et al. [39] investigated this contradicting theory and stated that in the under-parameterized setting, width negatively affects robustness, while in the over-parameterized setting, it improves robustness. The impact of depth depends on model initialization and the training mode.

However, it is worth noting that the aforementioned works are primarily focused on computer vision and convolution networks, and there has been less research conducted on the robustness of the Attention/Transformer architecture in the field of NLP. In this project, our main focus is to investigate the potential impact of different architectural modifications on the adversarial robustness of transformer-based architectures in the domain of NLP.

In 2019 Hsieh et al. [12] compared the adversarial robustness of self-attentive neural networks (Transformers) and recurrent neural networks in language tasks by applying several different adversarial input perturbations. Their experimental results have shown that, compared to recurrent neural models(LSTMs), self-attentive models (BERT Transformers) are more robust against adversarial perturbation in sentiment analysis, entailment, and machine translation. They also provided a theoretical overview regarding why the self-attentive models can have better robustness than recurrent models. However, this paper did not address the network capacity difference between the LSTMs and the transformers they used. It's important to note that the experimental comparison may not have been entirely fair due to this substantial difference in model capacity. Furthermore, the paper lacks a comprehensive discussion on the details of the model configuration and training procedure, which could potentially influence adversarial robustness.

A recent study by Neerudu et al. [40] focuses on the adversarial robustness of fine-

tuned transformers like Bert and GPT-2. Their research produces a comparison of the robustness of various pre-trained transformer models across nine datasets. However, it's worth noting a limitation in these experiments: they only employed basic, random toy example attacks like *Drop first*, *Drop char*, and *Swap words*, without making use of the state-of-the-art adversarial attacks such as Textfooler. Furthermore, they do not consider the modifications to the components in transformers but only consider pre-trained models as a whole. In contrast, we conducted ablation studies to examine the impact of various components within the transformer, including the self-attention mechanism.

3.2 Attention Variants of Interest

Dot-product attention with softmax normalization serves as the fundamental building block within the transformer architecture, but its quadratic space and time complexity relative to the sequence length result in considerable computational overhead, particularly when dealing with lengthy inputs. Linformer [41], SOFT [42], SimA [43], CosFormer [44], and TransNormer [45] emerged as alternative linear transformer architectures. Additionally, it's worth noting that standard self-attention lacks Lipschitz-bound property for unbounded input domains [46], whereas additive attention [47] maintains this property. Furthermore, researchers have investigated the robustness of standard self-attention, leading to the proposal of PAAS attention [48] as a potentially more robust mechanism in vision transformers [49]. In this section, we will describe these modifications in depth and discuss the motivations behind each scheme.

3.2.1 Additive Attention

Additive Attention [47], also known as Bahdanau Attention, was first proposed for Seq2Seq encoder-decoder models. The original transformer paper [1] claims that dot-product attention is faster and more space-efficient in practice due to the speedup gain from optimized matrix multiplication. Despite that, additive attention is able to provide a tighter bound in terms of Lipschitzness [50] while standard dot-product attention is not Lipschitz for unbounded input domain [46]. Lipschitzness has been established to be closely associated with adversarial robustness in certification [51]. Therefore, we are motivated to investigate whether this behaviour exhibits any connection with model robustness within the domain of NLP.

Adopting a similar notation as in 2.9, Additive Self-Attention can be formulated as follows:

$$Q = \mathbf{q}W_q, K = \mathbf{x}W_k, V = \mathbf{x}W_v \tag{3.1}$$

$$\text{Additive Attention }_{W_k,W_q,W_v,W_o}(\mathbf{x}, \mathbf{q}) = W_o\ (S \odot V) \tag{3.2}$$

$$\text{where } S = \text{softmax}(\sigma(\frac{Q+K}{2})W_s) \tag{3.3}$$

And $W_s \in \mathbb{R}^{d\times 1}$, σ is a non-linear function (usually $\tanh$), $\odot$ denoting element-wise multiplication.

3.2.2 Position-Aware Attention Scaling

A variant of Vision Transformer called Robust Vision Transformer, proposed by Mao et al. [48], has demonstrated a 10% robustness performance increase in some computer vision tasks. The application of Position-Aware Attention Scaling(PAAS), employed as a substitute for conventional positional encoding, stands out as a technique adopted by the authors, and the authors claim that PAAS can benefit the robustness of the Vision Transformer.

In PAAS, the original scaled dot-product attention is modified as follows:

$$\text{PAAS Attention} = \text{Softmax}(QK^\top \odot (\frac{W_p}{\sqrt{d_h}}))V \tag{3.4}$$

Where n is the maximum sequence length of inputs, $W_p \in \mathbb{R}^{n\times n}$ is an importance matrix representing the importance of each pair of Q and K. And $\odot$ is the element-wise product. The parameter W_p is independent regarding the input contents, therefore the authors propose to discard the traditional positional encoding and use this modification as positional information. In our experiments, we will present outcomes encompassing both scenarios: one with positional encoding retained and another without.

This paper also discussed the relationship between model robustness and the number of Transformer attention heads. The authors claim that an appropriate number of heads is crucial to robustness, and it benefits the model by supplying diverse informative details from the input. This richer and more unique information may be missed by models with fewer heads.

3.2.3 Linformer

Linformer [41] is one of the early proposals that reduce the computation complexity of self-attention to be $\mathcal{O}(n)$, where n is the length of the input sequence. The authors first show that the self-attention mechanism can be approximated by a low-rank matrix. With this theory in mind they proposed Linformer which adds 2 linear learnable projection matrices $E, F \in \mathbb{R}^{n \times k}$ when computing key and value. k is a hyperparameter which controls the trade-off between model expressivity and computation. It is usually chosen to be much smaller than n to reduce the dimensionality of the multiplied matrices. The Linformer attention can then be computed with:

$$\text{Linformer Attention} = \text{Softmax}(\frac{Q(EK)^\top}{\sqrt{d_h}})FV \tag{3.5}$$

$$\tag{3.6}$$

Note the above operations only require (nk) time and space complexity. Since k is a hyperparameter, therefore the complexity of Linformer self-attention is linear regarding input sequence length n. The authors also prove that if $k \in \mathcal{O}(d/\epsilon^2)$ (independent of n), one can approximate standard dot-product self-attention using Linformer self-attention with ϵ error.

3.2.4 SimA

A simple but effective attention mechanism called Simple Softmax-free Attention (SimA) was proposed by Koohpayegani and Pirsiavash [43] in 2022. The authors criticize the Softmax layer for its computation expensiveness, and they used the l1 norm to mimic Softmax's weight-averaging normalization behaviour.

$$\text{SimA} = \hat{Q}(\hat{K}^\top V) \tag{3.7}$$

$$\hat{Q} = \frac{Q}{|Q|_1} \text{ and } \hat{K} = \frac{K}{|K|_1} \tag{3.8}$$

The l1 normalization is applied across all tokens for every column/feature of Q, K. With Softmax removed, the matrix multiplication in attention can be reordered from $(QK)V$ to $Q(KV)$. As $Q, K, V \in \mathbb{R}^{n \times d_h}$, $Q(KV)$ would have a complexity of nd_h^2 while $(QK)V$ has $n^2 d_h$. This would reduce computation complexity if $n \gg d_h$. In our experiments, we will experiment with both l1 and l2 normalization in Sim Attention, and examine the impact of Softmax operation within the transformer architecture.

3.2.5 SOFT

SOFT, short for Softmax Free Transformer [42], is another transformer variant that achieves linear complexity. In their paper, Lu et al. [42] use a Gaussian kernel to define the similarity (self-attention) function, thereby eliminating the need for subsequent softmax normalization. Following this, they apply matrix decomposition to ensure linear complexity in the self-attention mechanism.
With that in mind, the SOFT attention can be written as

$$\text{SOFT Attention } = SV_j \tag{3.9}$$

$$S = \exp(-\frac{1}{2\sqrt{d_h}} \cdot ||Q_i - K_j||_2^2) \tag{3.10}$$

Where i, j represents the index of Q and K. And $\odot$ is the element-wise product. The SOFT self-attention matrix S has some important characteristics: (1) It is symmetric; (2) All elements are bound within the interval [0, 1]; (3) The diagonal elements, representing self-reinforcement, can be assigned with the highest value of 1, while the lower diagonal elements (corresponding to most dissimilar token pairs) being close to 0. The linearization of SOFT attention is further achieved by approximating a Low-rank regularization of S via matrix decomposition.

3.2.6 CosFormer

Despite previous success in linear transformers, it is worth noting that their performance varies in different tasks/corpus and sometimes suffers from crucial performance drops. Some of the proposed methods rely on constrained theoretical bounds to work. When their assumptions fail, approximation error gets accumulated which leads to worse transformer performances [44].
In 2022, Qin et al. [44] employed analysis and conducted multiple experiments on the Softmax attention. Two important properties of Softmax were identified:

1. Non-negativeness of the attention scores

2. Non-linear re-weighting mechanism as a training stabilizer

Based on these findings, CosFormer was proposed to address these two properties in a Softmax-free manner while achieving linear complexity. In CosFormer, ReLU activation is applied to Q and K to enforce non-negativeness, and a Cosine re-weighting mechanism is introduced to concentrate the distribution of the attention weights.

Formally, the CosFormer can be expressed as:

$$Q' = ReLU(Q), K' = ReLU(K) \tag{3.11}$$

$$Q_i^{cos} = Q'_i cos(\frac{\pi i}{2M}),\ Q_i^{sin} = Q'_i sin(\frac{\pi i}{2M}), \tag{3.12}$$

$$K_j^{cos} = K'_j cos(\frac{\pi j}{2M}),\ K_j^{sin} = K'_j sin(\frac{\pi j}{2M}) \tag{3.13}$$

$$\text{CosFormer Attention at position } i: \tag{3.14}$$

$$O_i = \frac{\sum_{j=1}^{N}(Q_i^{cos}(K_j^{cos})^\top + Q_i^{sin}(K_j^{sin})^\top)V_j}{\sum_{j=1}^{N} Q_i^{cos}(K_j^{cos})^\top + Q_i^{sin}(K_j^{sin})^\top} \tag{3.15}$$

$$= \frac{\sum_{j=1}^{N} Q_i^{cos}((K_j^{cos})^\top V_j) + \sum_{j=1}^{N} Q_i^{sin}((K_j^{sin})^\top V_j)}{\sum_{j=1}^{N} Q_i^{cos}(K_j^{cos})^\top + \sum_{j=1}^{N} Q_i^{sin}(K_j^{sin})^\top} \tag{3.16}$$

Where i, j represent the index of Q and K, and M represents the length of the longest sequence in Q and K. The authors of CosFormer have also performed extensive experiments to demonstrate that CosFormer's performance is on par with the vanilla transformer. In fact, in certain tasks, CosFormer even outperforms the vanilla transformer [44].

3.2.7 TransNormer

A very recent architecture called TransNormer, proposed by the same authors from CosFormer, was published in 2022 [45]. In this paper, they address 2 issues in Linear Transformers which induce performance drawbacks.

1. Unbounded gradients: Kernel-based linear attention suffers from unstable convergence caused by unbounded gradients, while the gradients in the vanilla transformer are bounded because of Softmax.

2. Attention Dilution: In early layers, linear transformers often trivially distribute attention scores over the whole sequence, while vanilla transformers usually preserve more local attention information.

TransNormer was proposed to mitigate both issues. In particular, TransNormer is composed of 2 sub-components: DiagAttention in early layers and NormAttention in later layers.

DiagAttention

DiagAttention uses a none-overlapped block strategy to compute the attention. Q and K are first divided into blocks and for each block with length w and width v, similarity function S_n is computed by

$$S_n = Q_n K_m^\top \tag{3.17}$$

$$\text{And } Q = \text{Concat}_{n=1}^{N} Q_n \tag{3.18}$$

$$K = \text{Concat}_{m=1}^{M} K_m \tag{3.19}$$

N and M are the number of blocks in corresponding Q and K. Each S_n is of size $w \times v$. w, v are hyperparameters of DiagAttention. In our experiments, we are working with self-attentions where both Q and K have the same size. This simplifies the process, as we only need to select w and fix $v = w$.

The key idea of DiagAttention is to keep attention focusing on the local information rather than tokens further away, and DiagAttention can reduce the effect of attention dilution in linear transformers. Figure 3.1 shows a visualization of how DiagAttention achieves similar behaviour with the Vanilla Attention.

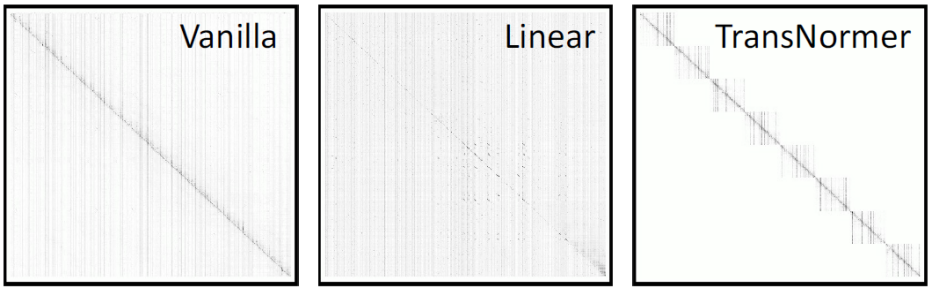

Figure 3.1: Comparing the attention matrices in the initial layers of both models, we can observe that the vanilla attention forces score to be local. The linear model noticeably experiences attention dilution while the proposed TransNormer DiagAttention produces more similar patterns to the original vanilla transformer [45]

However, applying DiagAttention to later layers would lead to a performance drawback, as the early layers focus on neighbouring tokens while the later layers need long-range attention [45].

NormAttention

NormAttention was proposed for the layer layers of TransNormer. It can be formulated as:

$$\text{NormAttention} = \text{XNorm}\ (Q(K^\top V)) \tag{3.20}$$

where XNorm can be some pre-defined normalization function. Examples of XNorm can be LayerNorm[52], RMSNorm[53] or SRMSNorm[54]. It can be proven that NormAttention gradients remain bounded when using these normalization functions. Empirical experiments also confirm the stability of gradients [45].

3.3 Modifications Beyond Attention

Furthermore, several experiments have been conducted to modify components of the transformer beyond attention mechanisms. These designs aim to enhance the standard performance of the Transformer with simple changes on the vanilla Transformer. We will discuss two of them here.
In 2019, Michel et al. [19] studied the effect of the multi-head attention mechanism in the original transformer and pre-trained BERT models. They found that a large number of heads may be removed from the architecture during test time without significantly impacting performance, and some layers can even be reduced to a single head. Another study conducted in 2020 by Press et al. [55] experimented with different ordering patterns of the transformer sub-layers. Their analysis shows that models with more self-attention toward the beginning and more feed-forward sub-layers toward the end may outperform the original transformer architecture proposed by Vaswani et al. [1] in 2017. Given the time constraints, we were unable to carry out these experiments in our current study. Nevertheless, it would be interesting to test the robustness of such architectures in future experiments.

Chapter 4

Experiment Setup

In this chapter, we will detail the setup of our experiments, including components such as dataset analysis, training procedure, and robustness evaluation. The experiments were performed on single GeForce RTX 2080 Ti and Nvidia Tesla A30 GPUs. We used a custom version of the Yelp-polarity dataset [56]. Pytorch [57] and TextAttack [58] were selected as our Python implementation frameworks.

4.1 Dataset Analysis

We present results on a customized version of the Yelp-polarity dataset [56]. We downloaded restaurant and business reviews directly from Yelp and processed them into CSV entries. Each Yelp review includes a "star" field, which rates from 1 to 5. In the current stage of the experiment, we focus on binary classification. Therefore, we consider 1, 2, and 3-star reviews as negative and 4 and 5-star reviews as positive.
Due to limited computing resources, we restricted our analysis to a total of 300,030 text-label pairs which were randomly sampled from Yelp's review database. Among all the reviews, there are 208,463 positive reviews and 91,567 negative reviews. A histogram depicting the distribution of review lengths is presented in Figure 4.1, and a pie chart showing the proportion of data labels is in Figure 4.2.

An example of review text in this dataset:

> *Honestly the food doesn't knock my socks off but other people seem to love this place. I go because my husband likes it as for me I'd rather go to a different BBQ spot. I guess it also depends on what you order.*

The real label for this review is negative. For us humans, we should be able to argue that the reviewer was not impressed by the restaurant. However, for a machine

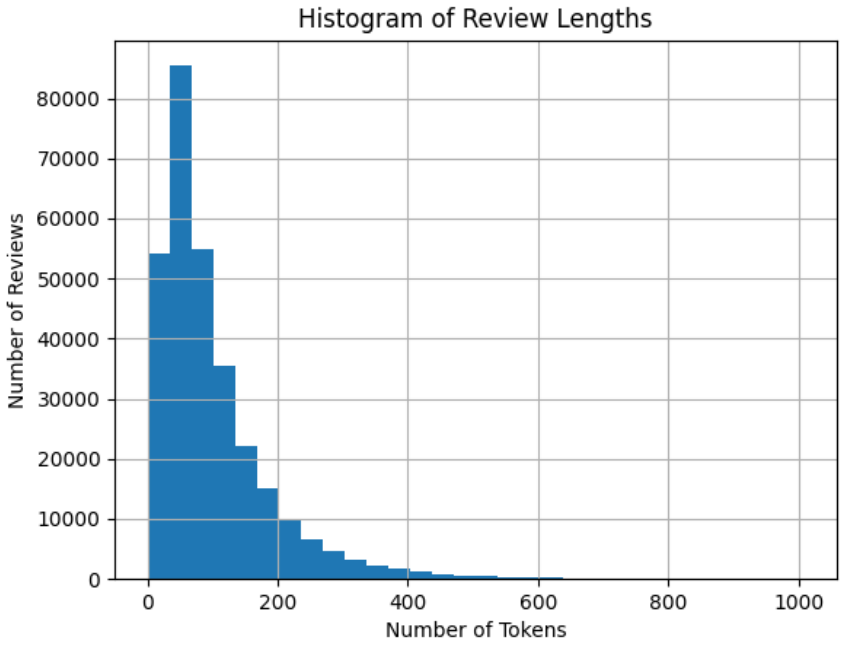

Figure 4.1: The length of Yelp reviews visualized as a distribution

learning model, classifying this review could be moderately challenging due to the nature of the mixed sentiment expressed. There are positive aspects such as ”other people seem to love this place” but also there is a bit of personal preference for a different BBQ spot, and a somewhat neutral tone when saying ”Honestly the food doesn’t knock my socks off”. In short, the sentiment is not straightforwardly positive or negative. A model would need to understand context, sarcasm, and nuanced language to accurately interpret the reviewer’s perspective.

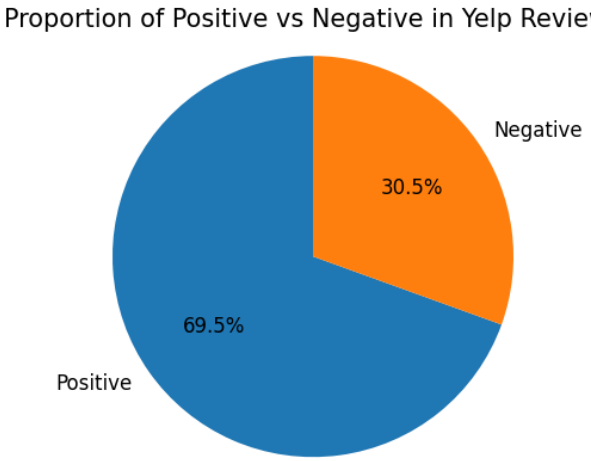

Figure 4.2: Proportion of Positive vs Negative in Yelp review, before up-sampling

4.2 Dataset Pre-processing

We further split the CSV entries into training, validation, and test sets using an 8:1:1 ratio. To address the potential issue of imbalanced data samples, we up-sampled the number of negative reviews by randomly duplicating them at a ratio of 2 during the training data preparation. It's worth noting that this may potentially encourage overfitting. To help this, we used regularization techniques like weight decay to provide additional stability during training. The pie chart after up-sampling can be viewed in Figure 4.3.

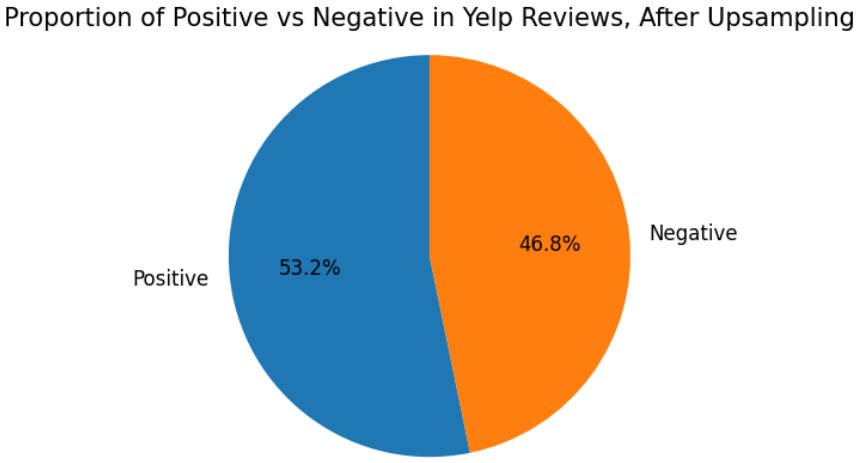

Figure 4.3: Proportion of Positive vs Negative in Yelp review, after up-sampling negative reviews by 2 times

4.3 Model Details

We experimented with two types of architectures: LSTM and Transformer. We will provide detailed explanations for each in the following subsections.

4.3.1 Tokenization

All input sentences are tokenized before feeding into our models. We used NLTK [59] to remove punctuation and apply tokenization to the input texts. In addition, we used NLTK's WordNetLemmatizer for lemmatization. Our tokenization is also uncased(treat all tokens as lowercase). We set the maximum sequence length of the model input to 150 tokens. An example of tokenization can be found in Figure 4.4.

```
>>> from tokenizer import tokenize
>>> text = "Some cat was layin' down some rock 'n' roll 'lotta soul' he said."
>>> print(tokenize(text))
['some', 'cat', 'wa', 'layin', 'down', 'some', 'rock', 'n', 'roll', 'lotta', 'soul', 'he', 'said']
```

Figure 4.4: Example tokenization. As we can see, words such as ”was” are tokenized by NLTK to be ”wa”, as a result of lemmatization.

4.3.2 Word Embeddings

All our models feature a word embedding layer that converts token IDs into vector representations. In our experiments, the models are constructed using one of the three types of word embeddings: custom trainable embedding, GloVe embedding and Paragramcf embedding.

Custom Trainable Embedding

When employing a custom trainable embedding, we first pre-processed a vocabulary object from the entire dataset. This vocabulary object only retains the 8000 most frequently occurring words in the training set. It then applies the conversion of sentence tokens into corresponding IDs, which can then be used as input for the models. The word embedding matrix is of size $N \times K$, where N is the size of the vocabulary (8000 in our case) and K is a hyperparameter of embedding dimension. We set K to be 300, aligning it with the dimensions of GloVe and Paragramcf embeddings, which also possess a K value of 300. The embedding matrix has dimensions of 8000×300, resulting in a total of 2,400,000 parameters. These 2,400,000 parameters must be accounted for as part of the model's trainable parameters, contributing to an increase in the model's overall training capacity.

GloVe Embedding

We can also use a pre-trained embedding such as the GloVe embedding, and keep it fixed throughout the entire process. In this case the embedding matrix functions like a look-up table from tokens to vectors. For the GloVe embedding, we download the vectors directly from Pytorch's TorchText [57]. We used uncased GloVe6B300d which was trained on 6 billion tokens with 400K vocab, and each vector is of dimension 300. The GloVe embedding matrix thus has dimensions of $400\text{K} \times 300$, which equals 120M parameters. But since we keep this matrix fixed, these parameters are not counted as trainable therefore doesn't increase the model's overall training capacity.

Paragramcf Embedding

In 2016, [25] proposed counter-fitted word embeddings and published the fitted word vectors. Counter-fitting makes synonyms closer to each other in the vector space. Intuitively, synonyms with less distance mean that synonym-based adversarial perturbations are closer in the embedding space, making the model more robust. For this project, we used the counter-fitted version of Paragram-SL999 embedding [60], short as Paragramcf. This is the same word embedding that Textfooler uses to generate its synonym attacks. The Paragramcf embedding has dimensions $65,713 \times 300$, vocabulary size 65,173.

4.3.3 LSTM Configuration

We used 4-layer bidirectional LSTM [2] for our experiment. By default, a bidirectional LSTM generates an output vector of size $2\times$ the hidden size. Since we have a binary classification task, we employ a linear layer to reduce the LSTM output vector into a single neuron, which is then followed by a sigmoid activation function. We set the LSTM hidden size to be 200.

4.3.4 Transformer Configuration

We adopted the transformer encoder architecture from [1]. We employed a transformer encoder followed by a linear layer to reduce the model outputs to a single neuron. Our experiment consists of 1, 4 and 8-layer transformer encoders. Before the model outputs are fed into the linear layer, we calculate the average across the sequence dimension to minimize the number of trainable parameters. The model dimensionality is set to 300, and the hidden size of the feed-forward network is set to 1024.

Furthermore, our transformer models employed the multi-head attention mechanism with 5 heads as the default configuration, unless otherwise specified. We also conducted experiments to explore the impact of varying the number of attention heads, which is elaborated upon in Section 5.3.4 and also in Section 6.2. We followed the multi-head split and concatenation approach outlined in [1], where $d_h \times$ Number of Heads $= d$. This strategy ensures that the number of trainable parameters remains consistent, irrespective of the chosen number of heads. With a 4-layer transformer, the number of trainable parameters roughly corresponds to that of the 4-layer LSTM(as detailed in Table 4.1).

	LSTM	Transformer
Model Capacity (# of Trainable Parameters)	3.69M	3.91M

Table 4.1: LSTM vs Transformer model capacity.

4.3.5 Training Configuration

We used Adam [61] as our training optimizer with β = (0.9, 0.98), ϵ = 1e-9, same as the original transformer paper [1]. During each back-propagation step, we apply gradient clipping to reduce large gradients if they exceed an l2 norm of 1, which promotes more stable training. The learning rate is set to 0.0001 for all training. Batch size is set within the range of 64 to 200 for different models, primarily due to variations in GPU memory utilization associated with distinct architectural designs. For weight decay, we make use of the weight_decay parameter available in torch.optim.Adam and configure it within the range of 1e-4 to 1e-7 or opt for no weight decay. The specific choice depends on the observed over-fitting behaviour of our models, which may vary due to certain architectural designs providing implicit regularization. Each model undergoes training for 50 epochs, with the best-performing-epoch model being selected based on its performance on the validation set (early-stopping).

4.4 Robustness Evaluation

We conducted an adversarial robustness evaluation of our victim models using six efficient and well-performing adversarial attacks: Textbugger, Textfooler, Bae, DeepWordBug, PWWS, and A2t. The implementation of these attacks was carried out through the TextAttack framework.

4.4.1 TextAttack

TextAttack [58] is a framework designed for conducting adversarial attacks, data augmentation, and adversarial training in the field of Natural Language Processing (NLP). TextAttack addresses the difficulty of comparing multiple attacks, and it aims to simplify the process of constructing and evaluating adversarial attacks by providing a modular approach that combines multiple components.

TextAttack's approach involves decomposing NLP attacks into four key components: a goal function, a set of constraints, a transformation, and a search method. The

goal function determines the desired outcome of the attack, while the constraints ensure that the perturbations applied to the input text adhere to specific criteria like grammar or semantic similarity. The transformation involves altering the input text, and the search method finds a sequence of transformations that results in a successful adversarial example. All the NLP attacks mentioned above are implemented in the TextAttack Python framework.

Additionally, we did not set any attack query budget limit, meaning that the attacks exhaustively explored all feasible perturbations within specified constraints until either successfully deceiving the model or failing to do so. This ensures the generation of the most powerful attack for the provided text and victim model.

To accommodate TextAttack's requirements, we introduce an additional, non-trainable, simple layer to our trained models after the final sigmoid layer. This layer transforms the single-neuron output, denoted as y, into a vector output of the form $[y, 1-y]$. Both y and $1-y$ are bounded between $[0, 1]$. This adjustment ensures the model output size is equal to the number of labels present in the dataset, as expected by TextAttack.

To illustrate the overall robustness of a model, we computed an average accuracy. This average accuracy under the six attacks is determined by taking the simple mean of the model accuracies after each attack perturbation.

Due to limited computing resources, we conducted our robustness evaluation on a subset of the test set, with 1000 review-label pairs. All six attacks were assessed and results were recorded on this test subset.

4.5 Adversarial Training

In our experiment, we also explored the influence of architecture design on the adversarial training scheme, proposed by Madry et al. [13]. Our approach involved perturbing input sentences using Textfooler before each batch and then feeding these perturbed inputs into the model. It's worth noting that the process of generating these attacks is computationally intensive and time-consuming. Using the same attack strategy for adversarial training alone would require approximately 300 hours for a single epoch.

To mitigate this, we limited Textfooler's query budget to 100, allowing for a more efficient, greedy search for perturbation samples. We selected a subset of the best-performing trained models and conducted one epoch of adversarial training on the training set. The outcomes of this process are discussed in Section 6.3.

Chapter 5

Experiment Results

In this chapter, we present the results and insights from our experiments on the adversarial robustness of transformer models. It's important to note that, unless explicitly mentioned, all models were trained with a capacity of around 3.8 million trainable parameters. This consistent capacity allows us to make fair comparisons regarding the impact of architectural design, without being influenced by model capacity variations. All our models are trained with fixed Paragramcf word embedding except when explicitly mentioned otherwise.

Our ultimate goal is to find an architecture that benefits from adversarial robustness while maintaining a standard performance on par with the baseline vanilla transformer.

5.1 LSTM vs Transformer

It's worth noting that while [12] propose that transformers are significantly more robust than LSTMs, their results may be influenced by the differences in datasets, model sizes, training methodologies, and other variables can considerably influence the outcomes, making direct comparisons less straightforward. To ensure a fair comparison, in our experiments, we try to be consistent in architectural components and training details for both LSTMs and transformers.

In our experiments, we trained both 4-layer LSTMs and transformers, and the best-performing models from the validation set serve as our victim models for evaluation. Detailed information regarding the setup of our models can be found in Section 4.3. For both models, we used Paragramcf word embeddings, and we ensured that both

models had similar model capacity, and approximately 3.8 million trainable parameters, to ensure a fair evaluation between these architectures.

Table 5.1 presents both the standard accuracy and average accuracy across six attacks for the trained LSTM and transformer models. Additionally, Table 5.2 provides a breakdown of accuracy under each of the six attacks.
We observe that the LSTM model exhibits a slightly better performance in terms of standard accuracy, while the transformer model demonstrates greater robustness when subjected to adversarial attacks. A more detailed analysis from Table 5.2 reveals that the transformer outperforms the LSTM in five out of the six NLP attacks, showcasing its improved robustness.

Accuracy in %	LSTM	Transformer
Standard Accuracy	89.10	88.00
Avg Acc Under Attack	20.25	22.18

Table 5.1: LSTM vs Transformer performance.

% Accuracy Under Attacks	LSTM	Transformer
Textbugger	17.90	22.70
Textfooler	0.30	3.70
Bae	25.60	26.10
DeepWordBug	5.10	8.50
Pwws	0.60	1.60
A2t(white-box)	72.00	70.50
Average	20.25	22.18

Table 5.2: LSTM vs Transformer Accuracy under different attacks.

There are a few explanations for why the LSTM does better on the standard inputs:

1. Data Size: LSTMs are known to perform well on smaller datasets [62], while Transformers often require more data to achieve their full potential. Yelp-polarity is a relatively small dataset, and the LSTM may generalize better.

2. Training Duration: Transformers often require longer training times compared to LSTMs [62]. 50 epochs might not give the transformer enough computation to converge to its optimal performance.

Despite the slight drawback in standard performance, the transformer still shows good potential in adversarial robustness over multiple attacks. In the next section, we will discuss how the design of the self-attention mechanism impacts transformer performance outcomes.

5.2 Self-Attention Variants

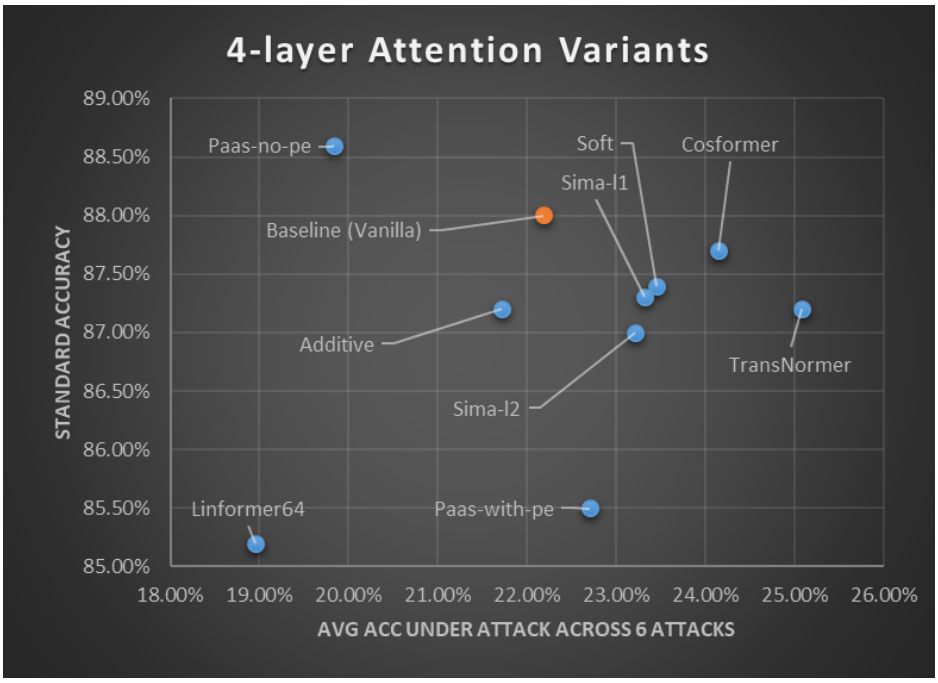

Figure 5.1: The comparison of each attention variant in terms of standard accuracy (y-axis) and average accuracy under attacks(robustness) (x-axis). The orange dot denotes the Baseline Vanilla Scaled Dot-Product Attention. Note that the top-right space denotes models that are more accurate and more robust.

In this section, we will demonstrate and discuss our experimental results for all self-attention variants mentioned in Section 3.2. All transformers we trained in this section have 4 encoder layers and have a similar model capacity of around 3.8 million to 3.9 million trainable parameters. We implemented all these self-attentions on PyTorch and fixed all other architecture components the same as the vanilla transformer for a fair comparison. We also kept all transformer hyper-parameters (except weight decay and batch size) fixed for variants. All self-attention variants were implemented from scratch except CosFormer[44] which has reusable code implementation available online. Figure 5.1 provides a visual representation of each attention variant's relative performance. We will now provide a more detailed overview of the performance of each variant.

5.2.1 Additive Attention

We trained 4-layer additive attention transformers with the same setting as vanilla transformers. In the additive attention mechanism, we selected the $\tanh$ function as

the non-linearity, the same as the original additive attention proposed by Bahdanau et al. [47]. The comparison of accuracy and attack outcomes is presented in Tables 5.3 and 5.4.

Accuracy in %	Additive	Baseline
Standard Accuracy	87.20	88.00
Avg Acc Under Attack	21.72	22.18

Table 5.3: Additive Attention vs Baseline Vanilla Attention performance.

% Accuracy Under Attacks	Additive	Baseline
Textbugger	19.00	22.70
Textfooler	2.50	3.70
Bae	25.50	26.10
Deepwordbug	7.50	8.50
Pwws	3.00	1.60
A2t(white-box)	72.80	70.50
Average	21.72	22.18

Table 5.4: Additive Attention vs Baseline Vanilla Attention Accuracy under different attacks.

Despite theoretical expectations of better Lipschitzness with additive attention (as discussed in Section 3.2.1), empirical experiments have revealed there is no observed superiority in terms of standard or robustness performance with additive attention. In fact, additive attention appears to perform slightly worse compared to the baseline Dot-Product Attention.

5.2.2 Position-Aware Attention Scaling

We trained 4-layer transformers using the Position-Aware Attention Scaling (PAAS) mechanism, considering scenarios both with and without absolute positional encoding. As explained in Section 3.2.2, PAAS introduces a learnable importance matrix, denoted as W_p, with dimensions of $n \times n$, where n corresponds to the maximum sequence length of input data. It's worth noting that although the addition of this matrix does contribute to the model's capacity, the overall increase in parameters is negligible, in total only 3.9 million parameters. Consequently, PAAS models maintain a capacity that is comparable to the baseline transformer.

Accuracy in %	Paas-with-pe	Paas-no-pe	Baseline
Standard Accuracy	85.50	88.60	88.00
Avg Acc Under Attack	22.70	19.83	22.18

Table 5.5: PAAS vs Baseline Vanilla Attention performance. PE stands for positional encoding.

% Accuracy Under Attacks	Paas-with-pe	Paas-no-pe	Baseline
Textbugger	21.00	18.00	22.70
Textfooler	3.20	0.20	3.70
Bae	28.70	25.60	26.10
Deepwordbug	10.90	3.40	8.50
Pwws	3.20	0.00	1.60
A2t(white-box)	69.20	71.80	70.50
Average	22.70	19.83	22.18

Table 5.6: PAAS models accuracy under different attacks. PE stands for positional encoding.

Table 5.5 indicates that PAAS with positional encoding exhibits a marginally better level of robustness to the baseline model but suffers from a reduction in standard performance. Conversely, PAAS without positional encoding displays favourable standard performance but demonstrates poor robustness across nearly all scenarios (refer to Table 5.6). It's also worth noticing that the removal of positional encoding from PAAS results in fast convergence and pronounced overfitting behaviour, plots are included in the Appendix Figure A.1.
The experiments have shown that, while PAAS has demonstrated robustness in Vision Transformers [48], its benefits have not translated effectively into our domain of NLP, as none of its variants have managed to surpass the performance of the baseline transformer.

5.2.3 Linformer

Hyperparameter tuning was performed on the value of K in Linformer, and it was eventually set to 64. The outcomes are summarized in Table 5.7.

Accuracy in %	Linformer64	Baseline
Standard Accuracy	85.20	88.00
Avg Acc Under Attack	18.95	22.18

Table 5.7: Linformer Attention vs Baseline Vanilla Attention performance.

% Accuracy Under Attacks	Linformer64	Baseline
Textbugger	14.20	22.70
Textfooler	2.30	3.70
Bae	23.20	26.10
Deepwordbug	4.50	8.50
Pwws	1.40	1.60
A2t(white-box)	68.10	70.50
Average	18.95	22.18

Table 5.8: Linformer64 vs Baseline Accuracy under different attacks.

While Linformer represented one of the early attempts at linear transformers, it does not emerge as the most promising architecture in our experiments. Neither its standard accuracy nor its robustness appears to be on par with the baseline transformer. Despite its innovative low-rank regularization approach which guarantees linear complexity, Linformer has been observed to under-perform in comparison to the baseline self-attention mechanism. Additionally, it does not demonstrate significant improvements in robustness as a result of this approach.

5.2.4 SimA

Sim Attention (SimA) is a transformer architecture that discarded the softmax layer in self-attention. With SimA's simple nature, we want to investigate whether it may match the performance of the baseline transformer with the Softmax layer. We experimented with both l1 and l2 normalization (which are explained in Section 3.2.4) in Sim Attention and the results can be found in Table 5.9 and 5.10.

Accuracy in %	Sima-l1	Sima-l2	Baseline
Standard Accuracy	87.30	87.00	88.00
Avg Acc Under Attack	23.32	23.22	22.18

Table 5.9: SimA vs Baseline Vanilla Attention performance.

% Accuracy Under Attacks	Sima-l1	Sima-l2	Baseline
Textbugger	21.20	24.30	22.70
Textfooler	2.80	2.20	3.70
Bae	25.70	24.20	26.10
Deepwordbug	13.20	13.20	8.50
Pwws	3.70	3.80	1.60
A2t(white-box)	73.30	71.60	70.50
Average	23.32	23.22	22.18

Table 5.10: SimA vs Baseline Accuracy under different attacks.

While there are no apparent drawbacks to SimA's performance, it's worth emphasizing its strong robustness when facing Deepwordbug attacks. Deepwordbug is unique as it relies entirely on character-based attacks. The fact that SimA exhibits better robustness under Deepwordbug attacks suggests that its normalization strategy is effective in enhancing resilience against character-based adversarial attacks.

5.2.5 SOFT

SOFT is a variant of the transformer model that achieves linear complexity by using a Gaussian kernel for self-attention, eliminating the need for softmax normalization, and applying matrix decomposition to maintain linear complexity in the self-attention mechanism(more explained in Section 3.2.5). The performance results of SOFT are summarized in Table 5.11 and 5.12.

Accuracy in %	SOFT	Baseline
Standard Accuracy	87.40	88.00
Avg Acc Under Attack	23.45	22.18

Table 5.11: SOFT Attention vs Baseline Vanilla Attention performance.

% Accuracy Under Attacks	SOFT	Baseline
Textbugger	24.00	22.70
Textfooler	3.90	3.70
Bae	30.20	26.10
Deepwordbug	7.60	8.50
Pwws	2.20	1.60
A2t(white-box)	72.80	70.50
Average	23.45	22.18

Table 5.12: SOFT vs Baseline Accuracy under different attacks.

While SOFT does experience a very small reduction in standard accuracy, it demonstrates a noteworthy improvement in robustness when subjected to adversarial attacks. In fact, it showcases better performance in five out of the six adversarial attacks evaluated. This promising behaviour motivated us to further investigate additional softmax-free transformer architectures for our task.

5.2.6 CosFormer

CosFormer removes the Softmax layer by dividing it into two sub-components: 1. ReLU layers on query and key, 2. cosine re-weighting mechanism on the attention

scores. The performance of CosFormer is shown below.

Accuracy in %	CosFormer	Baseline
Standard Accuracy	87.70	88.00
Avg Acc Under Attack	24.15	22.18

Table 5.13: CosFormer Attention vs Baseline Vanilla Attention performance.

% Accuracy Under Attacks	CosFormer	Baseline
Textbugger	24.90	22.70
Textfooler	3.90	3.70
Bae	27.70	26.10
Deepwordbug	11.00	8.50
Pwws	3.70	1.60
A2t(white-box)	73.70	70.50
Average	24.15	22.18

Table 5.14: CosFormer vs Baseline Accuracy under different attacks.

CosFormer, despite a minor reduction in standard accuracy, has demonstrated a notable and consistent improvement over the baseline transformer in all six adversarial attack evaluations. The results imply that the ReLU + cosine-reweighting mechanism that CosFormer has (see Section 3.2.6) could be more robust compared to the standard Softmax Dot-Product attention. This promising outcome prompted us to explore the influence of ReLU-induced non-negativity on model robustness, which is detailed in Chapter 6.

5.2.7 TransNormer

TransNormer presents a combination of Diag Attention and Norm Attention. In our TransNormer implementation, we structured it with the first half of attention layers using Diag Attention and the second half using Norm Attention. For instance, in the case of a 4-layer TransNormer, the first two layers are configured as Diag Attention layers, and the last two layers are Norm Attention layers. This aligns with the settings that the TransNormer paper proposes [45]. We conducted experiments using both Layer Norm and SRMS for the Norm Attention layers to comprehensively assess their performance and robustness characteristics. w is a hyperparameter in Diag Attention denoting the size of each sub-attention block. In this experiment, it is set to 15.

Accuracy in %	TransNormer-Layer-Norm	TransNormer-SRMS	Baseline
Standard Accuracy	87.20	86.80	88.00
Avg Acc Under Attack	25.08	23.58	22.18

Table 5.15: TransNormer vs Baseline Vanilla Attention performance.

% Acc Under Attacks	TransNormer-Layer-Norm	TransNormer-SRMS	Baseline
Textbugger	27.30	24.90	22.70
Textfooler	4.80	3.60	3.70
Bae	28.10	25.20	26.10
Deepwordbug	12.70	10.30	8.50
Pwws	3.70	3.00	1.60
A2t(white-box)	73.90	74.50	70.50
Average	25.08	23.58	22.18

Table 5.16: TransNormer Accuracy under Different Attacks.

In this case, TransNormer-Layer-Norm stands out for its good adversarial robustness, accompanied by only a moderate decrease in standard accuracy. On the other hand, SRMS lags behind the Layer-Norm in terms of both standard and adversarial accuracy under attacks.

5.3 Ablation Study

In our previous experiments, we studied how different self-attention mechanisms in the literature may influence the adversarial robustness of transformer models. Now we will provide an analysis of various sub-components or additional model features within the transformer architecture and evaluate their influence on adversarial robustness. These investigations aimed to provide a comprehensive view of the factors that contribute to the overall performance and adversarial robustness of transformer-based models.

5.3.1 DiagAttention

Diag Attention, as a sub-component in TransNormer, can also be evaluated as a type of standalone self-attention in our experiments. w is a hyperparameter in Diag Attention denoting the size of each sub-attention block. Experimentally, we fix w to be 15 so that each block is size 15×15.

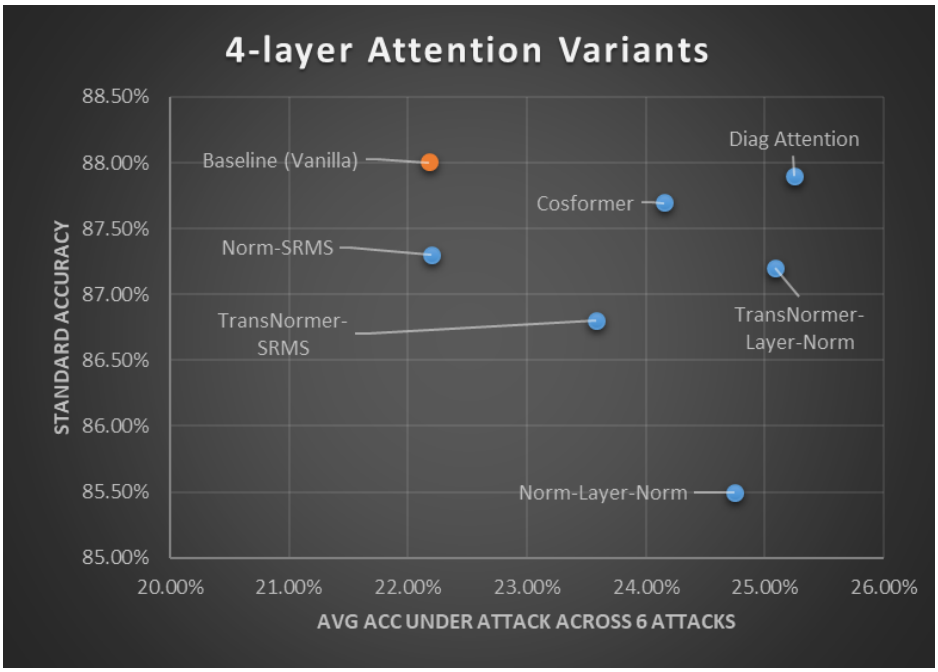

Figure 5.2: Standard accuracy vs Average accuracy under six attacks, comparison of TransNormer and its sub-components. Baseline and CosFormer are added for better visualization. The top-right space denotes models that are accurate and more robust.

Accuracy in %	DiagAttention	Baseline
Standard Accuracy	87.90	88.00
Avg Acc Under Attack	25.25	22.18

Table 5.17: Diag Attention vs Baseline Vanilla Attention performance.

% Accuracy Under Attacks	DiagAttention	Baseline
Textbugger	27.50	22.70
Textfooler	5.10	3.70
Bae	29.90	26.10
Deepwordbug	9.70	8.50
Pwws	4.40	1.60
A2t(white-box)	74.90	70.50
Average	25.25	22.18

Table 5.18: Diag Attention vs Baseline Accuracy under different attacks.

Diag Attention, much like CosFormer, showcases superior robustness while maintaining a comparable standard performance in comparison to the baseline transformer. It can also be viewed as an advanced regularization attention scheme. Its block-based nature restricts attention to tokens within the same block for each token, effectively preventing potential overfitting behaviour. These attributes make Diag Attention a

promising approach not only for regularization but also for model robustness. For instance, if an adversary perturbs a token, only the block containing that token is directly affected in terms of attention scores, while other tokens are less impacted. This characteristic may enhance the model's robustness to adversarial attacks. The performance comparison of Diag Attention with other attentions can be found in Figure 5.2.

5.3.2 NormAttention

Similar to Diag Attention, Norm Attention is another sub-component in TransNormer that could be used standalone. We implemented two versions of Norm Attention, LayerNorm Attention [52, 45] and SRMSNorm Attention [54]. They serve as a simple normalization without Softmax scaling. We trained both settings with 50 epochs and we present the results in Table 5.19, and Table 5.20.

Accuracy in %	Norm-Layer-Norm	Norm-SRMS	Baseline
Standard Accuracy	85.50	87.30	88.00
Avg Acc Under Attack	24.75	22.20	22.18

Table 5.19: Norm Attention models vs Baseline Vanilla Attention performance.

% Accuracy Under Attacks	Norm-Layer-Norm	Norm-SRMS	Baseline
Textbugger	26.40	23.70	22.70
Textfooler	3.60	1.60	3.70
Bae	32.30	23.00	26.10
Deepwordbug	11.20	10.30	8.50
Pwws	2.30	2.10	1.60
A2t(white-box)	72.70	72.50	70.50
Average	24.75	22.20	22.18

Table 5.20: Norm Attention Models Accuracy under Different Attacks.

In the case of layer-norm Norm Attention, it appears to exhibit better robustness but comes at an equivalent cost in standard performance. While SRMS Norm Attention demonstrates slightly worse performance compared to the baseline with no improvement in robustness. However, it's important to note that none of these models demonstrates a significantly better robustness-accuracy trade-off over the baseline model. The performance comparison of two Norm Attentions with other attentions can be found in Figure 5.2.

5.3.3 Word Embedding

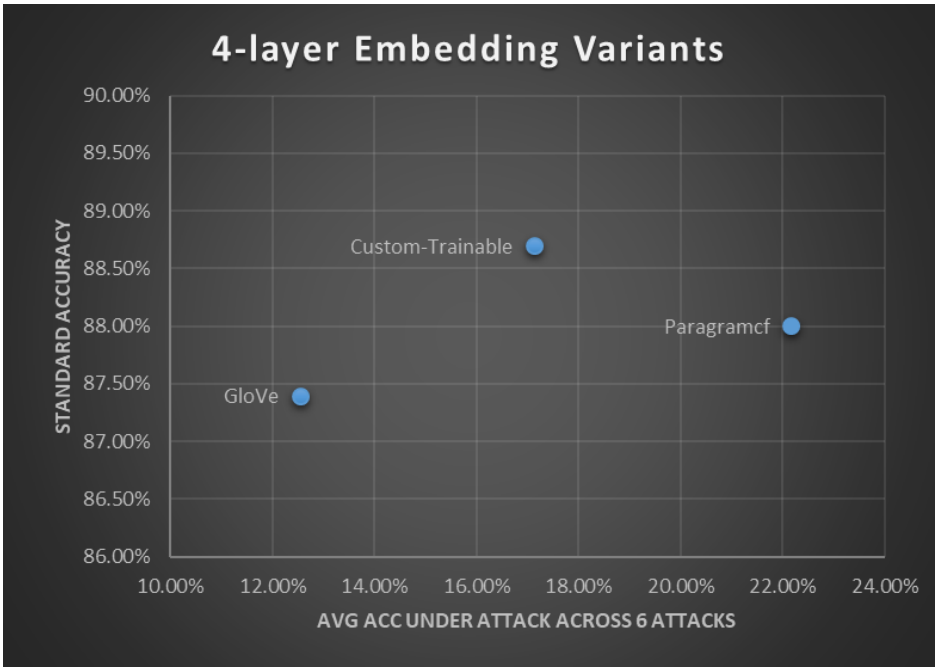

Figure 5.3: Standard accuracy vs Average accuracy under six attacks, comparison of three different word embeddings. The top-right space denotes models that are accurate and more robust.

We conducted ablation experiments on three word embeddings, which are described in Section 2.3, also 4.3.2. We used the standard Scaled Dot-Product Attention on this and kept everything else constant. For all three word embeddings we chose to have an embedding dimension of 300. The experimental results are presented in Figure 5.3, Table 5.21 and 5.22.

Accuracy in %	Custom-Trainable	GloVe	Paragramcf
Standard Accuracy	88.70	87.40	88.00
Avg Acc Under Attack	17.13	12.55	22.18

Table 5.21: Word embedding ablation experiment performance.

% Accuracy Under Attacks	Custom-Trainable	GloVe	Paragramcf
Textbugger	18.20	5.80	22.70
Textfooler	0.10	0.00	3.70
Bae	19.50	21.30	26.10
Deepwordbug	12.90	5.30	8.50
Pwws	1.60	0.40	1.60
A2t(white-box)	50.50	42.50	70.50
Average	17.13	12.55	22.18

Table 5.22: Word embedding ablation experiment accuracy under different attacks.

Despite GloVe's widespread popularity in the field of Natural Language Processing, it has exhibited weaker robustness when compared to the other two word embeddings. Another interesting observation here is the significant improvement in adversarial robustness achieved by counter-fitted word embeddings compared to other embedding methods. As noted in Section 2.3.3, counter-fitted embeddings have similar words closer to each other in the embedding space, which potentially contributes to model robustness against attacks. When comparing fixed word embeddings, it's clear that counter-fitted embeddings outperform GloVe in all six attacks. Moreover, this robustness benefit is not limited to attacks involving word synonym replacements (such as Textfooler, Bae, A2t) but also extends to character-based attacks like Deepwordbug, as shown in Table 5.22.

The model behaviours under Deepwordbug attacks are rather intriguing. The accuracies under Deepwordbug attacks appear to be inversely proportional to the vocabulary size of the word embeddings. The details of this can be found in Table 5.23. This suggests that word embeddings with larger vocabularies may be more vulnerable to character-based attacks, such as those from Deepwordbug. This phenomenon could be attributed to the fact that in large vocabulary embeddings, certain character modifications to input words may result in another word from the vocabulary, creating ambiguities. In contrast, smaller vocabularies tend to categorize perturbed words as unknown, reducing such issues.

	Custom-Trainable	GloVe	Paragramcf
Vocab Size	8,000	400,000	65,713
Acc Under Deepwordbug (%)	12.90	5.30	8.50

Table 5.23: Vocab size in word embeddings vs Accuracies under Deepwordbug.

5.3.4 Number of Heads

To study the influence of adversarial robustness on the number of heads in Multi-Head Attention (MHA), we conducted experiments using the standard Scaled Dot-Product Attention with different numbers of heads. We used the split-concat Multi-Head strategy, where we split the attention input from d dimensional vectors to d_h, where $d =$ number of heads $\times\ d_h$ for attention calculation. This configuration aligns with the setup proposed by Vaswani et al. [1]. In this case, the total computation cost and model capacity are similar to that of single-head attention with full dimensionality.

The results of this ablation experiment are summarized in Table 5.24, and we also present a visual representation of the relationship between Standard accuracy and Robust accuracy in Figure 5.4.

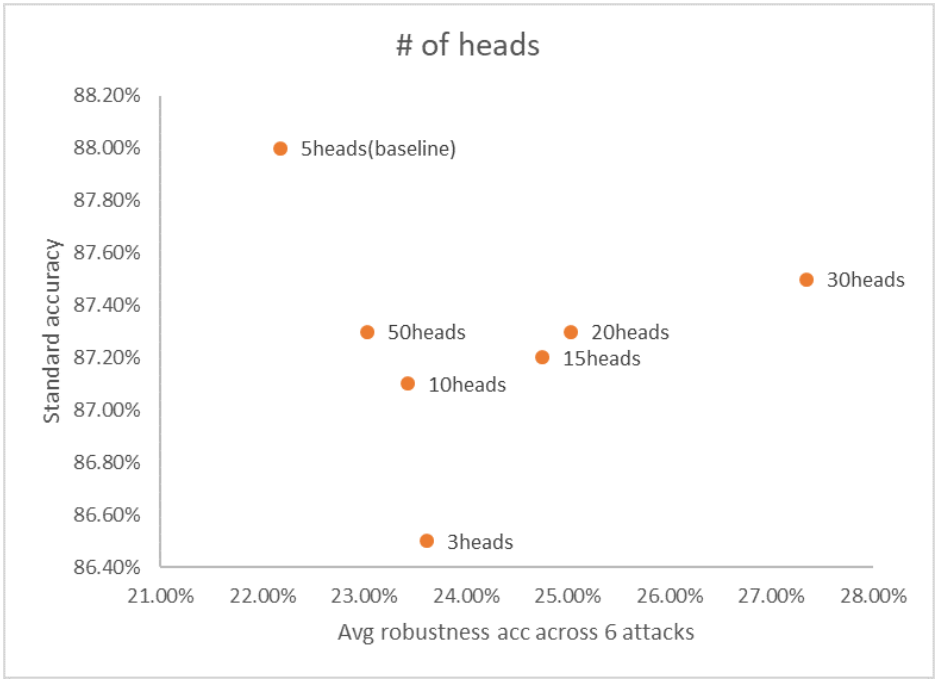

Figure 5.4: Standard accuracy vs Average accuracy under six attacks, ablation study for the number of heads in MHA. The top-right space denotes models that are accurate and more robust.

Similar to the finding in vision by Mao et al. [48], our experiments indicate that the choice of an appropriate number of heads can indeed influence the adversarial robustness of transformers. Specifically, it appears that 30-head's robustness stands out as an optimal setting for the original Scaled Dot-Product Attention mechanism. In fact, the robustness-accuracy trade-off of the 30-head vanilla transformer appears

Accuracy in %	3-head	5-head (Baseline)	10-head	15-head
Standard Accuracy	86.50	**88.00**	87.10	87.20
Avg Acc Under Attack	23.62	22.18	23.43	24.75
Accuracy in %	20-head	30-head	50-head	
Standard Accuracy	87.30	87.50	87.30	
Avg Acc Under Attack	25.03	**27.35**	23.03	

Table 5.24: Head number ablation experiment performance.

to be better than all the architectures described in Section 5.2. It is worth noting that the 30-head vanilla transformer's performance is observed on the test set but not the validation set, and therefore we treat this model also as a variation of the baseline architecture. We will include this architecture setting and discuss more in the end summary of Section 6.5.

However, it's important to note that when we ran similar experiments on Diag Attention, we did not observe the same trend as seen in this section. The detailed results of these two experiments are listed in Section 6.2 where we will provide more results and analysis of this intriguing behaviour.

5.3.5 Non-Negativity

Given the promising robustness results observed in CosFormer (as discussed in Section 5.2), we want to explore deeper into the potential of some of its sub-components. Drawing inspiration from the comprehensive analysis and ablation study on standard performance conducted by Qin et al. [44], we focused our attention on the significance of the ReLU activation function and its role in CosFormer, considering its impact on both standard performance and adversarial robustness.

In pursuit of a better understanding, we conducted experiments where we removed ReLU from the CosFormer architecture. The summarized results of these experiments are presented in Table 5.25.

Accuracy in%	CosFormer-noReLU	CosFormer
Standard Accuracy	83.20	87.20
Avg Acc Under Attack	20.48	24.08

Table 5.25: Ablation experiment of ReLU non-negativity in CosFormer.

The experimental results indicate a significant decrease in performance when ReLU

is removed from CosFormer. This highlights the critical role and advantages of non-linearity in self-attention modules, whether it is achieved through ReLU (as in CosFormer) or Softmax (as in the Vanilla Transformer). The findings underscore the significance of non-linearity in enhancing the model's performance and functionality in various NLP tasks.

Chapter 6

Towards Robust Transformer Design

In the previous chapters, we have detailed the experiments of previous works in various self-attention mechanisms. Our final goal is to enhance the model's robustness without compromising its standard performance in transformer models. With that in mind, we integrate the elements that may potentially enhance the trade-off between robustness and accuracy with the transformer architecture.

As a result, we introduce ReVA and ReVCos self-attention mechanisms, building upon the foundation of Scaled Dot-Product Attention and CosFormer Attention. These architectural variations aim to force non-negativity in the attention output as a means to improve model robustness while maintaining or even enhancing standard performance. In this chapter, we will describe these architectures, present our experimental findings and discuss the scalability of our transformers.

6.1 ReLU Value Attention

Building on the insights gained from our ablation study, and the analysis of ReLU Non-Negativity in CosFormer, we conducted additional experiments that demonstrated favorable results in both standard accuracy and robustness. Encouraged by these findings, we decided to further expand on this concept by introducing ReLU Value Attention (ReVA) and ReLU Value CosFormer (ReVCos).

These two architectures are derived from the Baseline Scaled Dot-Product Attention and CosFormer Attention modules. ReVA is essentially the Scaled Dot-Product Attention with ReLU activation applied to the Value component. On the other hand, ReVCos combines the CosFormer Attention mechanism with the enforced ReLU acti-

vation on the Value component.
More specifically, with the same settings from Equation 2.9, ReVA can be expressed formally as:

$$Q = \mathbf{q}W_q, K = \mathbf{x}W_k, V = \mathbf{x}W_v \tag{6.1}$$

$$\text{ReVA}_{W_k,W_q,W_v,W_o}(\mathbf{x},\mathbf{q}) = W_o\ \left(\text{softmax}\left(\frac{QK^\top}{\sqrt{d_h}}\right) ReLU\,(V)\right) \tag{6.2}$$

And ReVCos Attention can be written formally as:

$$Q' = ReLU(Q), K' = ReLU(K), V' = ReLU(V) \tag{6.3}$$

$$Q_i^{cos} = Q'_i cos(\frac{\pi i}{2M}),\ Q_i^{sin} = Q'_i sin(\frac{\pi i}{2M}), \tag{6.4}$$

$$K_j^{cos} = K'_j cos(\frac{\pi j}{2M}),\ K_j^{sin} = K'_j sin(\frac{\pi j}{2M}) \tag{6.5}$$

$$\text{ReVCos Attention at position } i: \tag{6.6}$$

$$O_i = \frac{\sum_{j=1}^{N}(Q_i^{cos}(K_j^{cos})^\top + Q_i^{sin}(K_j^{sin})^\top)V'_j}{\sum_{j=1}^{N} Q_i^{cos}(K_j^{cos})^\top + Q_i^{sin}(K_j^{sin})^\top} \tag{6.7}$$

$$= \frac{\sum_{j=1}^{N} Q_i^{cos}((K_j^{cos})^\top V'_j) + \sum_{j=1}^{N} Q_i^{sin}((K_j^{sin})^\top V'_j)}{\sum_{j=1}^{N} Q_i^{cos}(K_j^{cos})^\top + \sum_{j=1}^{N} Q_i^{sin}(K_j^{sin})^\top} \tag{6.8}$$

With ReVA and ReVCos transformers, we empirically found them more robust than the baseline Scaled Dot-Product Attention, as shown in Table 6.1 and 6.2.

Accuracy in %	ReVA	RevCos	Baseline
Standard Accuracy	88.00	88.50	88.00
Avg Acc Under Attack	24.92	26.73	22.18

Table 6.1: ReVA vs RevCos vs Baseline Vanilla Attention performance.

% Accuracy Under Attacks	ReVA	RevCos	Baseline
Textbugger	24.60	29.80	22.70
Textfooler	4.50	6.20	3.70
Bae	26.90	30.00	26.10
Deepwordbug	13.80	15.00	8.50
Pwws	4.10	3.70	1.60
A2t(white-box)	75.60	75.70	70.50
Average	24.92	26.73	22.18

Table 6.2: Accuracy of Reva, RevCos, and Baseline under different attacks.

Both ReVA and ReVCos apply ReLU activations on the Value component of attention

mechanisms. This enforced non-negativity and non-linearity can help suppress negative or noisy information in the attention output, making it more robust to adversarial attacks that might exploit such vulnerabilities. The fact that ReVA and ReVCos outperform the baseline under various adversarial attacks suggests that their robustness improvements are not limited to a specific type of attack but are more broadly applicable.

6.2 Number of Heads

As seen in Section 5.3.4, the number of heads plays a critical role in the adversarial robustness of transformers. To further study the relationship between the number of heads and robustness, we decided to conduct further experiments with self-attention variants other than vanilla attention to see if the number of heads still has the same effect.

We trained Diag Attention and ReVCos attention with various numbers of heads. The experimental results are shown in Table 6.3.

A few observations can be made from Table 6.3:

- The number of heads can indeed have a moderate influence on both standard performance and model robustness.
- There is no single optimal choice for the number of heads that applies to all three architectures. The ideal number of heads appears to vary depending on the specific architecture.
- No clear trend could be found between the number of heads and model performance across all three architectures. Instead, it seems that the improvements in standard and robustness performance are attributable to better hyperparameter choices with different architectures.
- Diag Attention appears to be the one with the worst robustness-accuracy trade-off among the three architectures.
- Comparing the best-performing models of ReVCos and the Baseline Vanilla Transformer, it appears that ReVCos exhibit a slight better trade-off between robustness and accuracy.

Accuracy in %	ReVCos	Diag	Baseline
	3-head		
Standard Accuracy	88.20	86.90	86.50
Avg Acc Under Attack	26.32	22.57	23.62
	5-head (Baseline setting)		
Standard Accuracy	**88.50**	87.90	88.00
Avg Acc Under Attack	**26.73**	25.25	22.18
	10-head		
Standard Accuracy	88.90	88.50	87.10
Avg Acc Under Attack	24.88	23.35	23.43
	15-head		
Standard Accuracy	87.40	**87.70**	87.20
Avg Acc Under Attack	25.32	**25.75**	24.75
	20-head		
Standard Accuracy	88.00	87.20	87.30
Avg Acc Under Attack	24.37	23.57	25.03
	30-head		
Standard Accuracy	87.00	87.10	**87.50**
Avg Acc Under Attack	25.75	24.83	**27.35**
	50-head		
Standard Accuracy	87.60	87.50	87.30
Avg Acc Under Attack	24.85	23.67	23.03

Table 6.3: Comparison of different transformer variants with multiple number of heads. The model that exhibits the best robustness-accuracy trade-off is highlighted in bold.

6.3 Adversarial Training

In order to make our transformer models more robust, and also to showcase that the attention schemes that we propose can adopt adversarial training, we conducted adversarial training on the two proposed transformers (5-head) and the baseline transformers(5-head and 30-head). Due to the extensive computation time, we only ran one epoch of adversarial training on half of the training set for all three transformer models and we leave a complete adversarial training to future work. The detailed setup of adversarial training is described in Section 4.5. After one epoch of adversarial training, the results are shown in Table 6.4 and 6.5.

Accuracy in %	ReVA	RevCos	Baseline-5head	Baseline-30head
Standard Accuracy	87.00	87.90	87.00	86.60
Avg Acc Under Attack	32.90	32.73	29.23	23.98

Table 6.4: Performances for models after adversarial training.

% Accuracy Under Attacks	ReVA	ReVCos	Baseline	Baseline-30head
Textbugger	35.60	37.50	30.90	25.90
Textfooler	12.40	10.60	7.90	3.50
Bae	35.40	37.30	35.00	28.30
Deepwordbug	27.40	24.30	20.30	9.50
Pwws	8.00	7.50	5.40	2.10
A2t(white-box)	78.60	79.20	75.90	74.60
Average	32.90	32.73	29.23	23.98

Table 6.5: Accuracy under attacks for models after adversarial training.

Compared to the standard training results in Table 6.3, except for the Baseline-30head model, all other models benefited from adversarial training. Surprisingly, the performance of the Baseline-30head model actually worsened. This unexpected result shows that a larger head number might not be very compatible with adversarial training, but this claim also requires further investigation to understand why this particular setting did not respond positively in our experiment.

On the other hand, ReVCos stands out as a model that consistently exhibits better trade-off between adversarial robustness and standard performance. ReVA also demonstrates good robustness compared to the baseline model. The results of our experiments underscore the potential of adversarial training in further enhancing model robustness. These findings suggest that the integration of non-negativity through ReLU activation in the attention mechanism, as seen in ReVA and ReVCos, may lead to improved overall model performance and adversarial robustness.

6.4 Scaling Capacity

In Section 3.1, we discussed the controversy of the interplay between model capacity and adversarial robustness. This motivated us to conduct experiments to explore whether this phenomenon holds true in the context of NLP attacks.

Furthermore, if our attention mechanisms prove to be effective in slightly larger transformer models, it indicates the potential for scalability. This suggests that our attention mechanisms may continue to demonstrate effectiveness as we scale up to even larger models.

6.4.1 Number of Layers

We compared the performance of 4-layer transformers to that of 8-layer transformers with vanilla attention. The model capacity for these configurations was set at 3.8 million parameters and 7.8 million parameters, respectively. Here are the results:

Accuracy in %	8-layer	4-layer
Standard Accuracy	87.90	88.00
Avg Acc Under Attack	25.00	22.18

Table 6.6: Attention performance for 8-layer vs 4-layer baseline transformers.

% Accuracy Under Attacks	8-layer	4-layer
Textbugger	25.90	22.70
Textfooler	4.70	3.70
Bae	30.80	26.10
Deepwordbug	12.80	8.50
Pwws	2.40	1.60
A2t(white-box)	73.40	70.50
Average	25.00	22.18

Table 6.7: Accuracies under attacks of 8-layer vs 4-layer baseline transformers under different attacks.

The results clearly indicate that increasing the number of encoder layers(from 4 to 8) yields a positive impact on the robustness of the vanilla transformer model without a standard performance drawback.

Additionally, we conducted experiments involving 8-layer models, specifically ReVA, ReVCos, Diag, and Baseline-30heads transformers, to investigate whether scaling up the architecture leads to further improvements in robustness.

Accuracy in %	ReVA	ReVCos	Diag	Baseline-30heads	Baseline
	8-layer				
Standard Accuracy	87.70	89.40	86.20	86.70	87.90
Avg Acc Under Attack	24.68	26.15	24.57	23.50	25.00
	4-layer				
Standard Accuracy	88.00	88.50	87.90	87.50	88.00
Avg Acc Under Attack	24.92	26.73	25.25	27.35	22.18

Table 6.8: Comparison of 8-layer transformer models.

Upon comparing the results presented in Table 6.8, several observations can be made:

- ReVCos exhibits a better robustness-accuracy trade-off after scaling, making it the top-performing architecture among the four.
- ReVA displays a minor drawback in performance after scaling.
- In contrast, Diag Attention experiences a decrease in standard performance when compared to its 4-layer version. This suggests that Diag Attention may not scale as effectively and might not be the optimal architecture when used in isolation.
- Upon scaling, the 30-head Baseline Vanilla transformer demonstrated a reduction in standard accuracy and a notable decrease in average accuracy under attacks. There were no indications of over-fitting during training. This suggests that this particular model may not respond well to scaling in terms of the number of layers, and this head setting might not be the optimal choice for large transformers.

6.4.2 FFN Hidden Size

We also explored the effect of increasing the width of the transformer by doubling the dimension of the Feed-Forward Networks(FFN). Specifically, we changed the dimension from the baseline value of 1024 to 2048, effectively increasing the models' capacity to approximately 6.37 million trainable parameters. This approach allowed us to assess how an increase in model width influences performance and robustness in the transformer architecture.

Accuracy in %	FFN-2048	FFN-1024
Standard Accuracy	87.70	88.00
Avg Acc Under Attack	24.82	22.18

Table 6.9: Attention performance for wider transformers.

% Accuracy Under Attacks	FFN-2048	FFN-1024
Textbugger	26.80	22.70
Textfooler	3.90	3.70
Bae	28.50	26.10
Deepwordbug	12.60	8.50
Pwws	2.80	1.60
A2t(white-box)	74.30	70.50
Average	24.82	22.18

Table 6.10: Accuracy of FFN-2048 vs FFN-1024 under different attacks.

Similar to the number of layers, increasing the dimension of Feed-Forward Networks also yields better adversarial robustness, as shown in Table 6.10.
We also trained ReVA, ReVCos, and Diag transformers with FFN-2048:

Accuracy in %	ReVA	ReVCos	Diag	Baseline
	FFN-2048			
Standard Accuracy	87.10	88.10	87.10	87.70
Avg Acc Under Attack	24.50	25.90	25.18	24.82
	FFN-1024			
Standard Accuracy	88.00	88.50	87.90	88.00
Avg Acc Under Attack	24.92	26.73	25.25	22.18

Table 6.11: Comparison of Models with FFN dimension increase.

Again comparing the results presented in Table 6.11, the results did not reveal any significant improvements in either standard performance or robustness. It appears that increasing the dimension of the Feed-Forward Networks (FFN) to 2048 may not be the most effective strategy for scaling our set of transformers. Although, it is worth noticing that ReVCos exhibits better performance over Baseline in this context as well.

6.5 Summary

In this section, we aim to summarize all valuable contents in our experiments by comparing the overall standard performance and robustness improvements achieved by our proposed models with the vanilla transformers introduced by Vaswani et al. [1]. To ensure a fair comparison, all model comparisons are set to similar model capacities. The two comparative plots are presented in Table 6.1 and 6.2.

Table 6.1 and 6.2 highlight ReVCos as the architecture that demonstrates the most favourable trade-off between robustness and accuracy among all the experimented architectures, in both scenarios. It is worth noting that the Baseline-30head configuration exhibits unexpected behaviour in response to adversarial training and scaling. Despite its initial superior robustness in standard training, this model did not show consistency and even displayed a decrease in performance in multiple training scenarios. This suggests that the observed robustness may be a result of training variability rather than a consistent trend. Further investigation is needed to understand this phenomenon.

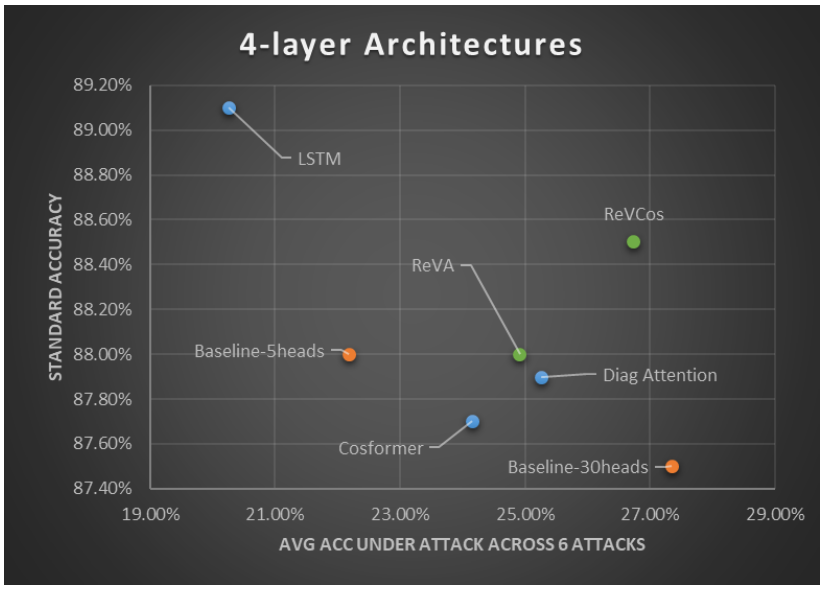

Figure 6.1: Standard accuracy vs Average accuracy under six attacks, for 4-layer architectures (3.8 million trainable parameters). Orange dots denote baseline vanilla transformers, green dots denote architectures that we propose and blue dots denote architectures from the literature. The top-right space denotes models that are accurate and more robust.

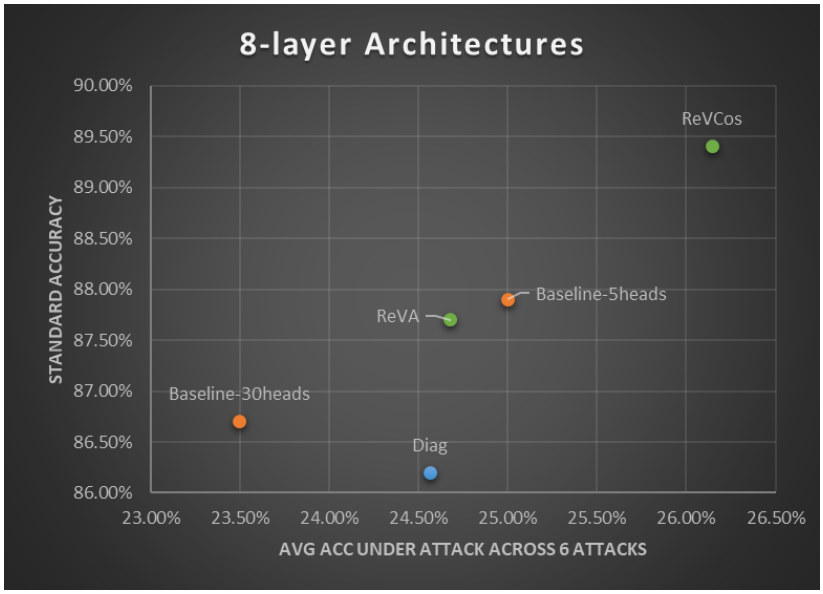

Figure 6.2: Standard accuracy vs Average accuracy under six attacks, for 8-layer architectures (7.8 million trainable parameters). Orange dots denote baseline vanilla transformers, green dots denote architectures that we propose and blue dots denote architectures from the literature. The top-right space denotes models that are accurate and more robust.

Chapter 7

Conclusion and Future Work

In this final chapter, we will first present the conclusion of our study, then we will outline the potential future research directions which provide ideas to further refine and expand upon these findings.

7.1 Conclusion

With comparable model capacity as a guiding principle, we conducted an empirical study of the adversarial robustness of transformer design, with a primary focus on the sentiment analysis task. We first showed that transformer exhibits better robustness than LSTMs. After that, our study involved the implementation and evaluation of multiple variants of the self-attention mechanism drawn from existing literature. Our evaluation included six distinct adversarial attacks, providing a thorough examination of the performance of the models under various adversarial conditions.

The results of our study showcased that certain attention mechanisms, such as CosFormer, TransNormer, and Diag, demonstrated superior robustness over the 5-head baseline vanilla transformer. Conversely, we observed that some other attention mechanisms, including Linformer and PAAS, exhibited comparatively weaker robustness when compared to the vanilla transformer's attention mechanism.

Our ablation study has revealed two interesting findings regarding the robustness of transformers. Firstly, we found that word embeddings have a significant impact on the model's robustness, with counter-fitted embeddings exhibiting superior performance. Secondly, the choice of attention heads can also influence the robustness of transformer models. We subsequently discovered that the vanilla transformer with

30 heads demonstrated a better robustness measure compared to all other models we evaluated from the literature. However, further experiments involving scaling and adversarial training have shown a certain degree of instability and inconsistency with this specific model, indicating that this might not be the reliable setting for large transformer architectures.

Building upon the valuable insights garnered from our experiments, we introduced a simple modification to the self-attention mechanism known as ReLU Value Attention (ReVA). Additionally, the proposed two new transformer designs, ReVA and ReVCos incorporate ReLU activation applied to the Value matrix of the self-attention mechanism. Experimentally, we found ReVCos is able to successfully improve the robustness accuracy trade-off compared to all the baseline vanilla transformers. In addition, our scalability experiments indicated that ReVCos could potentially achieve even greater levels of robustness when scaled to larger model capacities.

These findings underscore the potential for carefully crafted transformer architecture to significantly enhance model robustness without sacrificing standard accuracy. We believe that our experimental insights will pave the way for future research in the development of robust natural language processing architectures, contributing to advancements in the field.

Our results suggest that careful architectural choices in the design of a transformer can lead to improvements in its robustness without compromising performance on clean datasets. We hope that our experiments provide insights to future research that progress towards an adversarial robust NLP architecture.

7.2 Future Work

Given the limitations in time and computational resources, several potentially valuable ideas were identified but could not be explored within the scope of this project. We provide a list of these concepts here, offering a foundation for future research efforts beyond the project's current timeframe.

7.2.1 Adversarial Training & Other Defense Strategies

While our proposed transformer design exhibits notable gains in robustness, it's important to note that its accuracy under attacks doesn't yet qualify it as a fully robust

architecture. For instance, frameworks like Textfooler may still achieve an attack success rate exceeding 90%, given sufficient computational resources. This indicates that there is room for further refinement and exploration in our experiments. Moreover, further investigations employing adversarial training and additional defence strategies, such as verification techniques, could potentially enhance the robustness of this model.

7.2.2 Number of Heads

Given the interesting findings in Sections 5.3.4 and 6.2, it's clear that the relationship between the number of heads in a transformer and its adversarial robustness requires further investigation. Additional experiments are required to explain the reasons behind the observed decrease in adversarial learning for Baseline-30heads, as shown in Table 6.4.

7.2.3 Pre-Trained Models

The absence of experiments on pre-trained models like BERT is primarily due to computational resource limitations. Adapting these models to extensive architectural changes, as seen in our transformer designs, requires re-training from scratch to maintain their strong performance. In the future, it would be interesting to investigate the potential for improving the adversarial robustness of large language models through similar design choices such as ReVA and ReVCos, and explore the implications behind them. Our hope is that the development of large language models would put a focus on robustness, adopting architectural elements that may effectively defend against potential adversarial attacks.

7.2.4 Better Attack Configurations

As implied in Section 5.3.4, model selection through validation may play a large role in the results, but selecting the most robust model is a time-consuming process. In our robustness evaluations, we provided the attacks with an infinite query budget for generating perturbations, which led to extensive computation times. As a result, we had to cap the number of test examples at 1000, and we only evaluated one best-performing model in the clean validation set for each setting.

A more efficient approach for this experiment would be to restrict the query budget, which would significantly accelerate the attack generation process. Then we may

evaluate adversarial robustness on the validation for multiple settings. Finally, we choose the model with the best robustness-standard accuracy trade-off. This would allow us to conduct evaluations for models under various settings in the future.

7.2.5 More Attacks

Recently, there have been several proposals for novel methods to generate gradient-based attacks on textual inputs, that do not rely on search algorithms. In 2021, Facebook AI Research introduced a general-purpose gradient-based attack [63], in which they utilize a parameterized continuous valued distribution to sample the adversarial attacks. Additionally, in 2023, Yuan et al. [64] employed an iterative approach to reconstruct perturbed word embeddings into discrete words, using masked language models such as BERT. This approach enables the utilization of adversarial generation techniques from Computer Vision in NLP models. While the application of adversarial attack methods in computer vision cannot be directly transferred to textual data and NLP models due to the discrete nature of the text, these recent advancements have introduced novel techniques to bridge the gap between adversarial generation techniques from computer vision and NLP models. In future research, it would be valuable to evaluate the performance of the proposed transformer variants and design choices against novel attack methods. This would help ensure the robustness improvements observed in this study remain effective and consistent.

7.2.6 More Datasets

Given the time constraints, our experiments focused exclusively on sentiment analysis tasks using the Yelp-polarity dataset. Additionally, we worked solely on evaluating the self-attention mechanism, and cross-attention was never analyzed. It would be beneficial to extend this research to more complex tasks like text entailment or encoder-decoder tasks such as machine translation. This would provide a more comprehensive understanding of how the proposed improvements in robustness apply to a wider range of NLP applications.

7.2.7 Analysis and Pre-Training on ReVA and ReVCos

While we've seen success in our experiments, a thorough analysis of why applying ReLU to the Value component leads to improved robustness is necessary. A theoretic bound on the attention scores of this architecture would be interesting to explore.

Given our current time constraints, we plan to address these in future work, where we may dedicate more time and resources to explain this intriguing phenomenon.

7.3 Ethical Overview

The goal of this project is to enhance the adversarial robustness of deep neural networks, specifically focusing on the transformer architecture. This research is conducted with a strong commitment to ethical guidelines, ensuring that potential negative social impacts are minimized, particularly with a focus of model adversarial vulnerabilities. All experiments are carried out with a transparent, rigor manner to academic integrity principles. The use of the Yelp dataset strictly follows the terms outlined in their Terms of Use agreement. Additionally, we make considerate efforts to minimize environmental impact by utilizing computational resources efficiently. Finally, this project is dedicated to promoting open science practices by providing access to code, data, and models for the broader research community.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf. pages 1, 4, 5, 6, 7, 22, 28, 33, 34, 50, 60

[2] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735. pages 1, 3, 33

[3] Hasim Sak, Andrew W. Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *CoRR*, abs/1402.1128, 2014. URL http://arxiv.org/abs/1402.1128. pages 1, 3

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423. pages 1, 4, 9, 18

[5] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018. pages 1, 4, 9

[6] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. Adversarial attacks and

defenses in deep learning. *Engineering*, 6(3):346–360, 2020. ISSN 2095-8099. doi: https://doi.org/10.1016/j.eng.2019.12.012. URL https://www.sciencedirect.com/science/article/pii/S209580991930503X. pages 1

[7] Yong Cheng, Lu Jiang, and Wolfgang Macherey. Robust neural machine translation with doubly adversarial inputs, 2019. pages 1

[8] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment, 2020. pages 1, 11, 16, 17, 19

[9] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1103. URL https://aclanthology.org/P19-1103. pages 1, 18

[10] Siddhant Garg and Goutham Ramakrishnan. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.498. URL https://doi.org/10.18653%2Fv1%2F2020.emnlp-main.498. pages 1, 18, 19

[11] Jin Yong Yoo and Yanjun Qi. Towards improving adversarial training of nlp models, 2021. pages 1, 11, 19

[12] Yu-Lun Hsieh, Minhao Cheng, Da-Cheng Juan, Wei Wei, Wen-Lian Hsu, and Cho-Jui Hsieh. On the robustness of self-attentive models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1520–1529, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1147. URL https://aclanthology.org/P19-1147. pages 1, 21, 37

[13] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=rJzIBfZAb. pages 1, 13, 20, 35

[14] Sébastien Bubeck and Mark Sellke. A universal law of robustness via isoperimetry, 2022. pages 1, 21

[15] Hanxun Huang, Yisen Wang, Sarah Monazam Erfani, Quanquan Gu, James Bailey, and Xingjun Ma. Exploring architectural ingredients of adversarially robust deep neural networks, 2022. pages 1, 21

[16] Boxi Wu, Jinghui Chen, Deng Cai, Xiaofei He, and Quanquan Gu. Do wider neural networks really help adversarial robustness?, 2021. pages 1, 21

[17] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. ISSN 0364-0213. doi: https://doi.org/10.1016/0364-0213(90)90002-E. URL https://www.sciencedirect.com/science/article/pii/036402139090002E. pages 3

[18] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. In Włodzisław Duch, Janusz Kacprzyk, Erkki Oja, and Sławomir Zadrożny, editors, *Artificial Neural Networks: Formal Models and Their Applications – ICANN 2005*, pages 799–804, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. pages 4

[19] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/2c601ad9d2ff9bc8b282670cdd54f69f-Paper.pdf. pages 7, 28

[20] Martin Popel and Ondřej Bojar. Training tips for the transformer model. *The Prague Bulletin of Mathematical Linguistics*, 110(1):43–70, Apr 2018. ISSN 1804-0462. doi: 10.2478/pralin-2018-0002. URL http://dx.doi.org/10.2478/pralin-2018-0002. pages 9

[21] Peng Xu, Dhruv Kumar, Wei Yang, Wenjie Zi, Keyi Tang, Chenyang Huang, Jackie Chi Kit Cheung, Simon J. D. Prince, and Yanshuai Cao. Optimizing deeper transformers on small datasets, 2021. pages 9

[22] Asher Trockman and J. Zico Kolter. Mimetic initialization of self-attention layers, 2023. pages 9

[23] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162. pages 10

[24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. pages 10

[25] Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. Counter-fitting word vectors to linguistic constraints, 2016. pages 10, 11, 17, 33

[26] Felix Hill, Roi Reichart, and Anna Korhonen. Simlex-999: Evaluating semantic models with (genuine) similarity estimation, 2014. pages 10

[27] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1316. URL https://aclanthology.org/D18-1316. pages 11

[28] Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. Achieving verified robustness to symbol substitutions via interval bound propagation, 2019. pages 11

[29] Yoon Kim. Convolutional neural networks for sentence classification, 2014. pages 11

[30] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014. pages 11, 13

[31] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015. pages 11, 13

[32] Wei Emma Zhang, Quan Z. Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on deep learning models in natural language processing: A survey, 2019. pages 12, 14

[33] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples, 2018. pages 14

[34] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems, 2017. pages 14

[35] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers, 2018. pages 15

[36] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. TextBugger: Generating adversarial text against real-world applications. In *Proceedings 2019 Network and Distributed System Security Symposium*. Internet Society, 2019. doi: 10.14722/ndss.2019.23138. URL https://doi.org/10.14722%2Fndss.2019.23138. pages 15, 16

[37] Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder, 2018. pages 16, 17, 19

[38] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. pages 19

[39] Zhenyu Zhu, Fanghui Liu, Grigorios G Chrysos, and Volkan Cevher. Robustness in deep learning: The good (width), the bad (depth), and the ugly (initialization), 2023. pages 21

[40] Pavan Kalyan Reddy Neerudu, Subba Reddy Oota, Mounika Marreddy, Venkateswara Rao Kagita, and Manish Gupta. On robustness of finetuned transformer-based nlp models, 2023. pages 21

[41] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020. pages 22, 24

[42] Jiachen Lu, Jinghan Yao, Junge Zhang, Xiatian Zhu, Hang Xu, Weiguo Gao, Chunjing XU, Tao Xiang, and Li Zhang. Soft: Softmax-free transformer

with linear complexity. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21297–21309. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/b1d10e7bafa4421218a51b1e1f1b0ba2-Paper.pdf. pages 22, 25

[43] Soroush Abbasi Koohpayegani and Hamed Pirsiavash. Sima: Simple softmax-free attention for vision transformers, 2022. pages 22, 24

[44] Zhen Qin, Weixuan Sun, Hui Deng, Dongxu Li, Yunshen Wei, Baohong Lv, Junjie Yan, Lingpeng Kong, and Yiran Zhong. cosformer: Rethinking softmax in attention, 2022. pages 22, 25, 26, 39, 51

[45] Zhen Qin, XiaoDong Han, Weixuan Sun, Dongxu Li, Lingpeng Kong, Nick Barnes, and Yiran Zhong. The devil in linear transformer, 2022. pages 22, 26, 27, 28, 44, 47

[46] Hyunjik Kim, George Papamakarios, and Andriy Mnih. The lipschitz constant of self-attention, 2021. pages 22

[47] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016. pages 22, 40

[48] Xiaofeng Mao, Gege Qi, Yuefeng Chen, Xiaodan Li, Ranjie Duan, Shaokai Ye, Yuan He, and Hui Xue. Towards robust vision transformer, 2022. pages 22, 23, 41, 50

[49] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. pages 22

[50] Xiaojun Xu, Linyi Li, Yu Cheng, Subhabrata Mukherjee, Ahmed Hassan Awadallah, and Bo Li. Certifiably robust transformers with 1-lipschitz self-attention. 2022. pages 22

[51] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.,

2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/485843481a7edacbfce101ecb1e4d2a8-Paper.pdf. pages 22

[52] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. pages 28, 47

[53] Biao Zhang and Rico Sennrich. Root mean square layer normalization, 2019. pages 28

[54] Zhen Qin, Dong Li, Weigao Sun, Weixuan Sun, Xuyang Shen, Xiaodong Han, Yunshen Wei, Baohong Lv, Fei Yuan, Xiao Luo, Yu Qiao, and Yiran Zhong. Scaling transnormer to 175 billion parameters, 2023. pages 28, 47

[55] Ofir Press, Noah A. Smith, and Omer Levy. Improving transformer models by reordering their sublayers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2996–3005, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.270. URL https://aclanthology.org/2020.acl-main.270. pages 28

[56] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf. pages 29

[57] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. pages 29, 32

[58] John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp, 2020. pages 29, 34

[59] Edward Loper and Steven Bird. Nltk: The natural language toolkit, 2002. URL https://arxiv.org/abs/cs/0205028. pages 31

[60] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3:345–358, 2015. doi: 10.1162/tacl_a_00143. URL https://aclanthology.org/Q15-1025. pages 33

[61] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980. pages 34

[62] Aysu Ezen-Can. A comparison of lstm and bert for small corpus, 2020. pages 38

[63] Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. Gradient-based adversarial attacks against text transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5747–5757, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.464. URL https://aclanthology.org/2021.emnlp-main.464. pages 65

[64] Lifan Yuan, Yichi Zhang, Yangyi Chen, and Wei Wei. Bridge the gap between cv and nlp! an optimization-based textual adversarial attack framework, 2023. pages 65

Appendix A

Additional Tables and Figures

A.1 PAAS Loss Curves

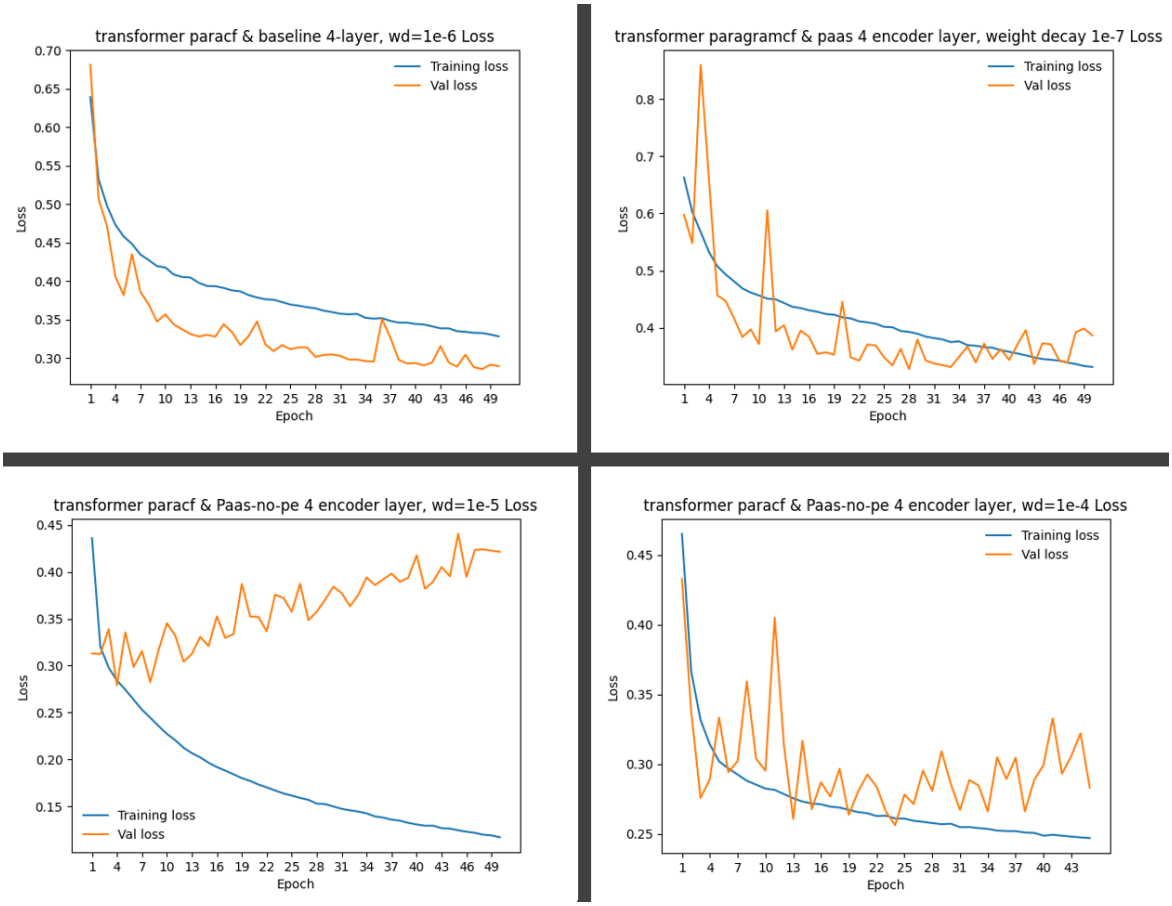

Figure A.1: Training loss curve of baseline(top left), PAAS-with-pe(top right), and PAAS-no-pe(bottom).

The loss curves for the baseline vanilla transformer, PAAS-with-pe, and PAAS-no-pe

are shown in Figure A.1. Upon visual examination, it's noticeable that the top two subplots (Baseline and PAAS-with-pe) exhibit greater stability compared to the bottom two (PAAS-no-pe). Specifically, the bottom two subplots suggest that PAAS-no-pe tends to converge quickly and overfit. Furthermore, increasing the weight decay appears to result in an unstable validation curve, as evidenced by the comparison between the bottom-left and bottom-right plots.