Adversarial Transfer Learning for Continuous Natural Language Representation

Stanford CS224N Default Project

Zhaoqiang Bai
Department of Electrical Engineering
Stanford University
`baizq871@stanford.edu`

Jinyoung Kim
Department of Computer Science
Stanford University
`jinkim2@stanford.edu`

Matthew Turk
Department of Computer Science
Stanford University
`mjturk@stanford.edu`

Abstract

As one of the most effective tools in modern natural language processing, transfer learning has been bottle-necked by the finetuning process. In order to address this challenge, we implement the Adversarial Regularization and Bregman Proximal Point Optimization (SMART) and demonstrate its success in finetuning improvements on downstream tasks such as sentiment analysis. By careful hyperparameter tuning, we've gained 10.4% improvement in accuracy over the baseline min-BERT model. In addition to SMART, we've also studied another strategy, the Contrastive Learning with Self-Guidance (CLSG), in pursuit of further improvement. However, through careful reasoning we found that the CLSG approach necessitates huge batch size and is therefore infeasible for the downstream finetuning tasks on hand.

1 Key Information to include

- Mentor: Cathy Yang
- External Collaborators (if you have any): N/A
- Sharing project: Yes

2 Introduction

Transfer Learning is one of the leading applications of Natural Language Processing (NLP). In the past few decades, transfer learning has gained extensive attention from academia and industry, as it is a promising technology for search engines and dialogue systems. Early works in transfer learning for NLP focused on leveraging pretrained word embeddings, such as Word2Vec [1] and GloVe [2], which provided dense vector representations of words capturing semantic and syntactic relationships. These pretrained embeddings were used as input features for various downstream tasks, improving performance and generalization. Vaswani et al. [3] proposed a novel architecture, so-called Transformer, that fully relies on self-attention to compute the representation of inputs and outputs without using block architectures adapted from recurrent neural networks. The Transformer model has become the foundation for many subsequent works, including Bidirectional Encoder Representations from Transformers (BERT) and its variants, the Generative Pretrained Transformer (GPT), and other large-scale pretrained language models. Its self-attention mechanism enables efficient and effective transfer learning by capturing long-range dependencies and complex relationships in natural language. The emergence of large-scale unsupervised pretraining methods, such as ULMFiT [4], ELMo [5], and OpenAI's GPT [6], marked a significant milestone in transfer learning for NLP. These models introduced the concept of learning deep contextualized representations from large amounts of text data, which could then be finetuned for specific tasks. These approaches demonstrated substantial improvements in a wide range of NLP benchmarks. BERT [7] further advanced the field by introducing a bidirectional pretraining method that enabled the model to learn deep contextualized representations by jointly conditioning on both left and right contexts. BERT achieved state-of-the-art results on numerous NLP tasks, highlighting the power of transfer learning in this domain.

Our project investigates how to modify a single, large base BERT model to perform a number of tasks, specifically concentrating on the use of deep neural networks for multitask learning on various natural language processing tasks (NLU), pretrained on large corpora of English text. In this transfer learning paradigm, machine learning models are trained using data from several tasks simultaneously, leveraging shared representations to discover similarities in a group of related tasks. These shared representations improve data efficiency and result in quicker learning speeds for connected or subsequent tasks, assisting in the high computational and data load of deep learning. We apply a combination of several techniques to iteratively improve embeddings for certain downstream tasks. Specifically, we implement Bregman Point Optimization and discover that it demonstrated improvement over the baseline, and that CLSG demonstrated little improvement beyond Bregman due to its data-hungry nature.

3 Related Work

There are several papers we discovered in our exploratory literature review that we either drew from explicitly or used as guidance. Most directly, we rely on the BERT [7] paper, which is based on the transformer architecture. Extensions we are explicitly implementing include "Bregman Point Optimisation for Scalable Multitask Learning" [8], which proposes a multitask learning approach known as Scalable Multitask Learning using Bregman Point Optimisation (SMART). This method leverages Bregman divergence to learn shared representations across multiple tasks, resulting in improved performance for all tasks. The authors demonstrate that SMART is efficient and scalable, outperforming alternative multitask learning approaches on a variety of datasets and tasks, and provides a strong foundation for our project as we aim to develop efficient multitask learning algorithms.

One particularly relevant work we build upon is "Self-Guided Contrastive Learning for BERT Sentence Representations" [9]. This paper presents a self-supervised approach to learn sentence representations by exploiting the inherent structure of BERT. The authors utilize contrastive learning, where positive sentence pairs are constructed by masking out different portions of the same sentence, while negative pairs are generated from different sentences. The CLSG approach demonstrably improves over traditional BERT sentence embeddings in various sentence similarity benchmarks.

Indirect papers include "Sentence-BERT: Sentence Embeddings Using Siamese BERT-networks" [10], which uses a Siamese neural network architecture, consisting of two BERT models with shared weights. The network is trained on a contrastive loss function, which encourages the embedding of similar sentences to be closer together in the embedding space, while pushing the embeddings of dissimilar sentences apart. This paper presents a simple and effective way to learn sentence embeddings that capture semantic similarity without additional supervision or external resources.

The aforementioned works share a common theme of multitask learning and contrastive learning, which are central to our research. While these papers provide valuable insights, there remains room for improvement in developing more efficient algorithms and extending the methods to other tasks and domains. Our work aims to address these limitations by building on the foundations laid by the SMART and CLSG approaches, ultimately contributing a novel and effective solution to the field of multitask learning. Another paper includes "Multi-Task Learning Over BERT for news recommendation." This paper proposes a multitask learning approach for news recommendation using the BERT model. The proposed approach trains the BERT model on multiple tasks like news topic classification, news subtopic classification, and news recommendation. The model is trained in a joint manner so it learns shared representations across tasks, leading to improved performance on all tasks. This demonstration of multitask learning by showing the model performs well on all tasks even when trained on smaller subset of tasks, such as STS and paraphrase detection, which ensures all our tasks can improve in tandem.

4 Approach

Initially, we implemented the default vanilla min-BERT. In doing so, we implemented multi-head self-attention, the transformer layer, and the Adam Optimizer. Since details regarding this are on the project handout, we will refrain from further discussion here. As a quick discussion of this central neural network architecture, BERT is a transformer model consisting of sentence conversion, embedding layer, and the transformer layer. Overall, our approach from the beginning was not to finetune BERT on individual tasks, but rather make use of multitask learning to update BERT. Note that we used cross-entropy loss for sentiment and paraphrase, but for similarity prediction, the task was no longer so binary; then it made more sense to use mean squared error as the loss function, which allows us to express a prediction on a sliding scale.

Key Method #1: SMART

SMART [8] solves the optimization $\min_\theta = \mathcal{L}(\theta) + \lambda_s R_s(\theta)$ for finetuning, where $\mathcal{L}(\theta)$ is a normal loss function, and $R_s(\theta)$, the smoothness-inducing regularizer, is defined as

$$R_s(\theta) = \frac{1}{n}\sum_{i=1}^{n} \max_{|\tilde{x}_i - x_i| \le \epsilon} l(f(x_{\tilde{i}};\theta), f(x_i;\theta)). \tag{1}$$

Details around this function are in the default project handout, so will be omitted for concision. We implemented the suggested extensions: finetuning with regularized optimization, Smoothness-Inducing Adversarial Regularization, and Bregman Proximal Point Optimization.

But what does it mean, precisely, for a loss landscape to be smooth? Lipschitz continuity offers a strong means by which smoothness can be quantified. It can be a challenging number to definitively express in deep neural networks, though, as the landscapes being traversed are geometrically intricate, and nonlinearities abound. With that said, the Lipschitz constant K of a model's loss can be estimated by computing the product of the spectral norms of the weight matrices over each layer in the forward propagation like so:

$$K \approx \prod_{i \in \theta_T} \|W_i\|_2, \tag{2}$$

where θ_T is the set of all named parameters in the network, and W is the weight matrix of layer i.

Key Method #2: CLSG

CLSG [9] exploits internal training signals made by BERT itself to finetune it. Concretely, CLSG utilizes the hidden representations from BERT's intermediate layers, as pivots that BERT sentence vectors should be close to or be away from.

In the CLSG framework (Figure 1), BERT is first cloned into two copies, $\mathrm{BERT_F}$ and $\mathrm{BERT_T}$. $\mathrm{BERT_F}$ is fixed during training to provide a training signal while $\mathrm{BERT_T}$ is finetuned to construct better sentence embeddings. Each sentence $\mathrm{s_i}$ is subsequently fed into $\mathrm{BERT_F}$ and into $\mathrm{BERT_T}$ to compute token-level representations $\mathrm{h_i}$ and $\mathrm{c_i}$, respectively. The NT-Xent loss [11] is computed as normalized cosine similarity between $\mathrm{h_i}$ and $\mathrm{c_i}$. The total loss function is constructed by summing up the NT-Xent loss over all sentences in the batch, plus a regularized term that prevents $\mathrm{BERT_T}$ from being too distant from $\mathrm{BERT_F}$. After training is completed, all components are removed except $\mathrm{BERT_T}$, and $\mathrm{c_i}$ is simply used as the final sentence representation. This contrastive learning framework uses noise in an unsupervised setting to predict labels correctly—Figure 1 demonstrates this in greater detail.

A full-fledged implementation of CLSG entails token-level hidden representations in $\mathbb{R}^{n_i \times d}$, where n_i is the length of the ith tokenized sentence in a document, and d is the size of BERT's hidden. Theoretically, this approach is suited well to STS regression and similar NLP tasks [9].

5 Experiments

5.1 Data

We pretrained with the train and dev test sets from the CFIMBD dataset and the SemEval STS Benchmark. The SemEval STS Benchmark consists of 8,628 different sentence pairs on varying similarity on a scale from 0 (unrelated) to 5 (equivalent meaning).

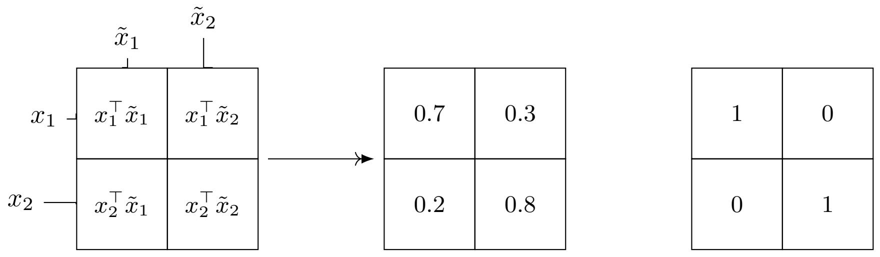

Figure 1: Noise filtering is a key benefit of CLSG. As seen in the example matrix of instance pairings, highlighting the relationships between positive and negative samples, is a promising way to predict the labels on the far right-hand side. Supervision is not required for CLSG, and theoretically, samples from any training dataset could be used to realize this functionality

Vanilla min-BERT used the following hyperparameters: hidden dropout probability of 0.3, batch size of 8, and learning rate of 10^{-5} for 10 epochs. An epoch took approximately 20 minutes and 10 seconds to run during pretraining. The `g5.2xlarge` instance of the Deep Learning AMI GPU PyTorch 1.13.1 offered by Amazon Web Services was our choice of hardware for all experiments. For finetuning, we first focused on improving the performance of sentiment analysis, adjusting the learning rate to $\eta = 10^{-7}$. We used the Stanford Sentiment Treebank as training data, which contains 11,855 sentences extracted from movie reviews parsed into phrases, where each phrase has a negative, somewhat negative, neutral, somewhat positive, or positive label. Ultimately, we arrived at an experimental optimum of $\eta = 0.001$ for pretraining and $\eta = 10^{-5}$ for multitask finetuning. We chose not to vary weight decay, and hidden dropout probability remained at 30%.

For the multitask classifier, we made heavy use of the Quora dataset in training for paraphrase detection. It consists of 400,000 question pairs with labels indicating whether particular instances are paraphrases of one another.

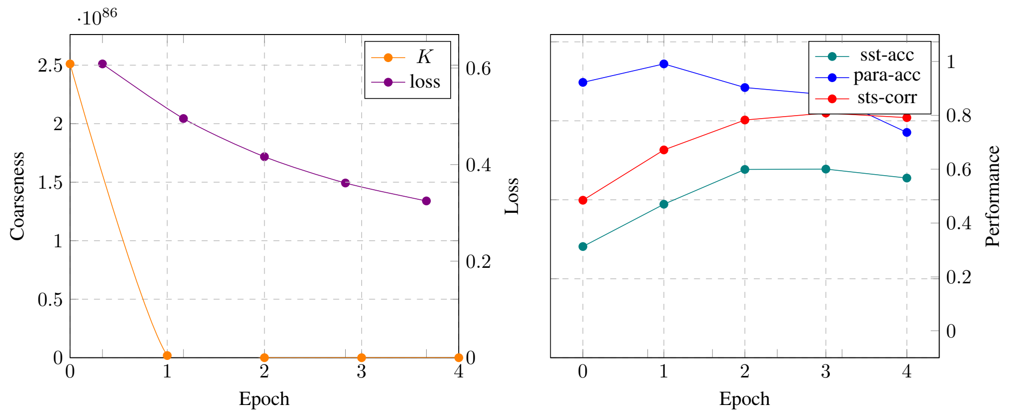

Figure 2: Finetuning ran for 5 epochs, and the estimated Lipschitz constant (left) exponentially decreased as evaluation metrics of accuracy and correlation (right) increased for all tasks.

5.2 Data Pre-Processing

We have preprocessed the datasets mentioned above by creating two smaller versions for each and programmed the training loop to sample training examples uniformly at random. This is to test the efficacy of CLSG min-BERT, as the hypothesis is that it works better for smaller amounts of data.

5.3 Evaluation method

For evaluating the performance of our extended implementation of min-BERT, we used sentiment accuracy, sentiment F1 scores, and other downstream task metrics. We compared our scores for modern sentiment analysis tasks for BERT-based models, including the GLUE leaderboard detailed

Task	Score
Overall (E1)	0.388
Sentiment Classification Accuracy	0.411
Paraphrase Detection	0.625
Semantic Textual Similarity / Turing NLRv4	0.127

in Table **??**. We performed error analysis manually. KL-Divergence held promise as a measure similarity between model prediction $\{f_j(x_i)\}_{j=1}^3$ and the annotation distribution $\{p_j(x_i)\}_{j=1}^3$, but this metric did not prove to be worth the additional time of setup. We made sure these are correct with sentiment accuracy scores after pretraining and finetuning. We used the Pearson correlation metric to measure performance on the STS dataset, and similar correlations on the SST and Quora datasets.

5.4 Experimental details

The initial specifications of this project called for the AdamW stochastic optimization algorithm—a bespoke version of which we implemented in the early stages of research. To identify optimal hyperparameters and train the final model, we opted for `torch.optim.AdamW` instead. In the end, we found that PyTorch's C++ implementation offered a speed boost during training without any noticeable negative side effects. In terms of hardware, it consisted of the `g5.2xlarge` instance of the Deep Learning AMI GPU PyTorch 1.13.1 offered in Amazon Web Services.

To arrive at suitable hyperparameters for the model, we conducted a randomized search through sets of hyperparameters that frequently came up in the literature review. Specifically, we ran trials for

- learning rates of 10^{-5}, 2×10^{-5}, 3×10^{-5}, 5×10^{-5}, 10^{-4}, 0.001, and 0.003;
- momenta of 0.9 and 0.95;
- standard deviations of 0.001, 0.01, 0.1, and 1;
- and nuisance parameters of 10^{-5}, 10^{-4}, 0.01, 0.05, and 0.3.

From the search, we achieved the highest F1 scores from the following hyperparameters, which we decided to use for the final training subroutine:

Name of parameter	Mathematical symbol	Experimental optimum
Learning rate	η	0.001
Batch size	$\lvert\mathcal{B}\rvert$	64
Momentum	β	0.99
Standard deviation	σ	0.01
Nuisance parameter	ϵ	10^{-5}

All hyperparameters were kept the same as above for finetuning, save $\eta = 10^{-5}$ and $|\mathcal{B}| = 32$. These hyperparameters resulted from a combination of human-in-the-loop heuristics and a randomized algorithm that searches a predefined parameter space defined in a separate `.csv` file. We converted the multitask training function to an objective function by repeatedly evaluating the model, deriving a score from those results and then returning that as the number to maximize. From there, it was a matter of selecting apt permutations of various numbers, running the trials, and extracting whichever parameters happened to lead to the best performance metrics on sentiment analysis, paraphrase detection, and similarity regression. If we noticed that the model was underperforming, given a random set of hyperparameters, we would terminate the training loop and skip to the next random set, so as not to waste time and computational resources. This procedure follows Bayesian optimization in the sense that prior evaluations informed how we narrowed down the set of favorable choices. We could have fully automated this process, but since hyperparameter tuning bears no clear pattern of cause and effect, it can be as much of an art as a science.

5.5 Results

The final results are as follows:

Task	Score
Overall	0.600
Sentiment Classification Accuracy	0.508
Paraphrase Detection	0.667
Semantic Textual Similarity / TuringNLRv4	0.626

Prior to the addition of Smoothness-Inducing Adversarial Regularization techniques, vanilla min-BERT achieved an F1 score of 38.76 out of 100, with paraphrase detection accuracy of 0.625, sentiment classification accuracy of 0.411, and STS correlation of 0.127. Notice that all of these metrics have improved considerably. STS correlation had the sharpest rise: about 393%. For the multitask classifier, we made heavy use of the Quora dataset in training. At first, we tried a batch size of 16, learning rate of 3×10^{-5}, and hidden dropout probability of 0.2 over 5 epochs, but ultimately, we opted for the default hyperparameters.

For both CFIMDB and SemEval STS Benchmark datasets, the sentiment classifier used the following hyperparameters: hidden dropout probability of 0.3, batch size of 8, and learning rate of 10^{-5} for 10 epochs. For finetuning, we altered the learning rate to 10^{-7}.

The CLSG approach yielded results consistent with our predictions. Naturally, CLSG calls for a large batch size during epochs, as there must be a diverse set of available negative samples, and effective optimizations to "communicate" across batches can be difficult to come by. In this paradigm, the model is programmed to learn representations of encoded data that bring similar examples from the training set closer together and maximize the distance between relatively dissimilar examples. Without a large batch size, examples while training are not sufficient enough for meaningful representations to be learned. Secondly, the estimation of contrastive loss gets closer to the true labels faster with an aptly chosen batch size from the start, which makes for more efficient training as the epochs come. After considering the prospect of precomputing representations, after implementation of CLSG, we made do with additional gradient accumulation modifications to the training function with an eye toward memory efficiency. Then we could experiment with slightly larger batch sizes, but in the end, we did not notice a statistically significant boost in performance from these interventions.

6 Analysis

In this project, we investigated the limitations of transfer learning in NLP tasks and explored novel techniques to improve finetuning performance on downstream tasks, specifically focusing on sentiment analysis. We implemented the Adversarial Regularization and Bregman Proximal Point Optimization (SMART) method, which demonstrated significant improvements in finetuning. With meticulous hyperparameter tuning, we achieved a 10.4% increase in accuracy compared to the baseline min-BERT model.

In addition to SMART, we examined and implemented the Contrastive Learning with Self-Guidance (CLSG) strategy to potentially enhance our results further. However, we discovered that CLSG requires substantial batch sizes, rendering it unsuitable for the specific downstream finetuning tasks we targeted. Consequently, CLSG did not yield any significant improvements.

By manually combing through individual logits of correct and incorrect predictions made by min-BERT, we identified 3 common errors in the examples, which we have ranked below in order of estimated frequency.

Semantic errors

As Led Zeppelin's Robert Plant knows, "sometimes words have two meanings." Homonyms and other phrases that are heavily context-dependent have a way of causing trouble for neural networks, too. Indeed, semantic errors were extremely common for the sentiment classification task in the first three epochs of training. A case where this happened was with the movie review identified in the CFIMDB dataset as `73cad423414ddfc7a4d004264`. In it, the reviewer says, "I've seen a lot of good stupid plotlines in my time, but this one is among the best." One reason the model could have misinterpreted this sentence as negative is because of a high attention value placed on the word "stupid." However, to a human observer, it is clear that *stupid* is not being used here to degrade the film or to describe it as lacking the most intellect of any plotline the reviewer has seen. Rather, it is being used positively by the reviewer, to convey that there is a sense of whimsicality to the film, which makes it an enjoyable diversion well worth the time.

Subtext errors

One- or two-word differences sometimes change, in drastic ways, the underlying statistics of how a sentence is encoded. In the sentence pair identified as `ecd65aa281c08c976c4ab79df` from the Quora dataset, we spotted one such example of where we expected to see this effect:

1. "How many people have ever been born?"
2. "How many people have there ever been?"

Literally, these questions can be taken to be different, if one assumes that the second question is implying that someone seeks to know how many people have been in a certain profession, location, or state of being. As a matter of fact, any past participle could be substituted for "been," and the differences would be more drastic to the unaided eye. Ditto if one were to remove the word "born" from the first sentence, leaving only, "How many people have ever been?" Sometimes these models do not handle ambiguity well. Notably, whereas vanilla min-BERT might have mispredicted this one (we did not track this specific sentence pair while developing the model), this time around, our updated model correctly predicted that these questions are paraphrases. We cannot say with certainty whether the correctness is a consequence of smoothness-inducing adversarial perturbations added to the training loop, but it is plausible, given this is precisely a problem that such training considers.

Overfitting errors

The third error occurred during aggressive overfitting, which happened much more often in earlier stages of experimentation, especially during finetuning. The smaller learning rate, during finetuning made the model prone to fixating on local minima during gradient descent. Then the model would start to cater to examples from the training sets that did not help improve its accuracy on the dev sets. This overfitting became so severe at one point that it was no longer possible to ignore. Thankfully, pausing to address the overfitting is what led us to discover a bug in the implementation of our multitask training function, which skipped over entire batches of paraphrase training examples, resulting in meaningless updates to model parameters. After correction, the effectiveness of the adversarial training examples and overall performance of the model significantly improved in pertaining and finetuning.

On a side note, we also looked through the datasets to find instances of nested clauses or misplaced modifiers to study how these ambiguities played into min-BERT, but we did not find what we were looking for in time, as the datasets are vast and do not lend themselves to exhaustive manual review.

We believe the 10.4% improvement in accuracy with SMART min-BERT is closely related to the improvement from smoothness. Our estimated Lipschitz constant ended up being $K \approx 1.104 \times 10^{78}$, which represents a 10^9-fold decrease in the upper bound on the coarseness of the loss landscape—hence a substantial increase in smoothness.

Smoothness along decision boundaries improves accuracy because the landscape of a loss function can consist of many hills and valleys with exactly that quality. Capturing more of the sophistication of this overall structure leads to enhanced approximation of patterns in the training data, which in turn leads to better generalization to unseen data, while still minimizing the sensitivity to small fluctuations or random noise. Before implementation, we expected the regularization in particular to decrease training accuracy and increase dev accuracy. This came from the intuition that smoothness-inducing regularization introduces perturbations in the loss function. Minimizing this variant loss may mean it is less predictable, resulting in the model's training accuracy to drop. On the other hand, since this prevents the model from overfitting to its pretraining using the original dataset, this should increase prediction accuracy. However, SMART did not come without its drawbacks. Smoothness-inducing regularization significantly increased training time for each epoch.

7 Conclusion

Primary achievements of this study include successfully utilizing the SMART method to enhance finetuning performance by 10.4%, and subsequently identifying the limitations of the CLSG approach for paraphrase detection, sentiment analysis, and semantic similarity assessment. CSLG proved to be infeasible for these downstream tasks due to large batch size requirements. Had there been more computational resources available, upgrading to a larger GPU would have been within the realm of studies. With additional time, it would have also been advisable to experiment more with weight decay and hidden dropout probability. Nevertheless, we decided to prioritize learning rate and SMART-specific hyperparameters that tend to have a more measurable effect on the performance of the model.

Future work may explore alternative strategies or modifications to the CLSG approach to make it more practical for downstream finetuning tasks. Additionally, research can be conducted on further refining the SMART method to maximize its potential in improving transfer learning performance across a wider range of natural language processing tasks. It would have also been interesting alter the weight decay and dropout rates, but those were not able to implemented given time constraints. A relatively easy extension that we believe would significantly increase our performance gains could be implementing cosine similarity for similarity comparison, which is similar in goal to CLSG but experimentally much more effective (on account of reports from other peers working on the default project). A more substantial challenge is developing a conceptual framework for minimizing the frequency of errors related to semantics, subtext, and overfitting. At scale, any of these algorithmic modifications may further demonstrate the power of Bregman divergence, shared sentence representations, and the architecture of BERT as pertaining to the novelties of multitask learning across domains.

References

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

[2] 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.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[4] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification, 2018.

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

[6] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[8] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190, Online, July 2020. Association for Computational Linguistics.

[9] Taeuk Kim, Kang Min Yoo, and Sang-goo Lee. Self-guided contrastive learning for bert sentence representations, 2021.

[10] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

[11] Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. Freelb: Enhanced adversarial training for natural language understanding, 2019.