# Default Final Project: Improving minBERT with Entailment Learning

Stanford CS224N Default Project

**Akash Chaurasia**
Dept. of Computer Science
Stanford University
`akashc@stanford.edu`

**Saksham Consul**
Dept. of Electrical Engineering
Stanford University
`sconsul@stanford.edu`

**Carlota Parés-Morlans**
Dept. of Electrical Engineering
Stanford University
`cpares@stanford.edu`

## Abstract

This paper explores the use of entailment learning to develop a single language model capable of simultaneously solving multiple downstream natural language processing tasks. Focusing on sentiment analysis, paraphrase detection, and semantic evaluation, we first establish a baseline level of performance on these tasks using an minimalist implementation of BERT. Then, we use counterfactual entailment supervision to unify the three tasks with sentence-pair encodings. Employing various training methods, such as dataset batch interleaving, we obtain a model capable of performing well on the SST, QQT, and STS datasets with an overall accuracy of $0.746$. Our experiments highlight the use of entailment learning for generalizability, which opens up an exciting new avenue of research for a "one model fits all" approach.

## 1 Introduction

Devlin et al. (2018)'s work on Biderectional Encoder Representation (BERT) was seminal in the field of Large Language Models (LLM) for introducing the concept of pre-training on a large corpus of data. This step allows the model to capture the statistical regularities of language, which are later fine-tuned for high performance on specific tasks. The success of this fine-tuning process varies depending on the task and the number of available training examples. Given the community trend of scaling LLMs to massive sizes, it has become increasingly important to build a 1-model-fits-all kind of approach Hoffmann et al. (2022). Such a model should be able to perform well on multiple downstream tasks, optimizing overall computational costs.

In this paper, we explore the use of entailment learning to create a uniform sentence-pair structure with the three tasks of interest and develop a model which uses a single backbone and three different heads to simultaneously perform well on the SST, QQT, and STS datasets. We extend the work of Wang et al. (2021) and add the required strategies for fine-tuning a minimalistic version of BERT. Our results demonstrate the power of such unified structure, which led to a tremendous improvement in performance. Moreover, we show the impact of dataset interleaving on BERT's ability to generalize to multiple natural language processing tasks using contextualized embeddings.

## 2 Related Work

There have been several studies exploring different strategies for fine-tuning LLMs for downstream NLP tasks. For instance, the work of Howard and Ruder (2018) showed how transfer learning is an effective strategy for fine-tuning the text classification task on various datasets. Meta-learning has also been heavily explored for fine-tuning multi-language translation and other downstream tasks (Gu et al., 2018; Brown et al., 2020). Meta-learning in the context of language models means the model develops a broad set of skills and pattern recognition abilities at training time, and then uses those abilities at inference time to rapidly adapt to unseen tasks.

Liu et al. (2020) explored the use of adversarial training to fine-tune BERT for text classification. The authors introduced an adversarial loss to encourage models to generate robust features invariant to small perturbations in the input. Wang et al. (2021) showed how entailment can be used as unified method to model all classification tasks. The key idea is to convert the class label into a natural language sentence which can be used to describe the label, and determine if the label description entails the example.

In this work, we build on the approach of Wang et al. (2021) and extend label descriptions to work for fine-grained sentiment analysis. Modifying the given minBERT approach to account for sentence-pair embeddings, we show the impact of adding entailment as a unification method.

## 3 Approach

In this project, we used a pretrained minimalistic version of BERT (minBERT) (Devlin et al., 2018), which includes 12 transformer layers, each of which includes multi-headed self-attention, followed by an additive and normalization layer with a residual connection, a feed-forward layer, and a final additive and normalization layer with a residual connection. Following this architecture, we added the required layers to complete the tasks of interest, which we fine-tuned together with minBERT weights.

In the subsequent sections, we provide a detailed explanation of the different methods we implemented, including the data we used.

### 3.1 Entailment

Entailment involves reformulating the task into predicting whether one sentence entails another. To this end, we implemented an entailment reformulation to unify learning for the 3 tasks. In the case of sentiment analysis as a multiclass classification problem, rather than taking in a sentence S and predicting a class label {0, 1, 2, 3, 4}, we provide a pair of sentences (e.g. ”Light, silly, photographed with colour and depth, and rather a good time” is accompanied by an entailment statement “The sentiment of the sentence is positive”) and predict if such pair is entailed or not. For each true example, we create counterexamples with the 4 other labels (see Figure 1). Hence, with entailment, the multiclass classification problem was reduced to a binary classification problem which is trained in a contrastive manner. (The true example provides a "pull" signal, whereas the false examples provide a "push" signal)

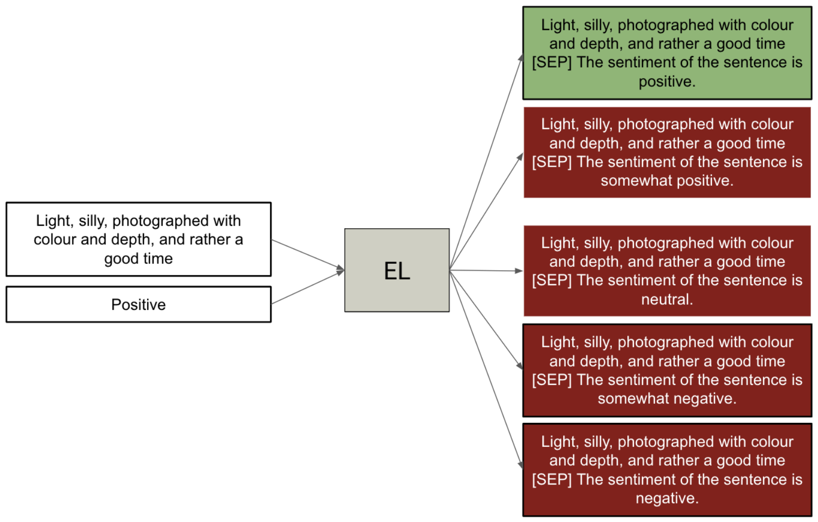

Light, silly, photographed with colour and depth, and rather a good time
Positive
EL
Light, silly, photographed with colour and depth, and rather a good time [SEP] The sentiment of the sentence is positive.
Light, silly, photographed with colour and depth, and rather a good time [SEP] The sentiment of the sentence is somewhat positive.
Light, silly, photographed with colour and depth, and rather a good time [SEP] The sentiment of the sentence is neutral.
Light, silly, photographed with colour and depth, and rather a good time [SEP] The sentiment of the sentence is somewhat negative.
Light, silly, photographed with colour and depth, and rather a good time [SEP] The sentiment of the sentence is negative.

Figure 1: An example of how entailment transforms a single input example in 5-way classification to 5 binary classification inputs: 1 positive example (from the original class) and 4 negative examples.

Table 1 describes the various templates used for the 3 problems, showing the unified structure.

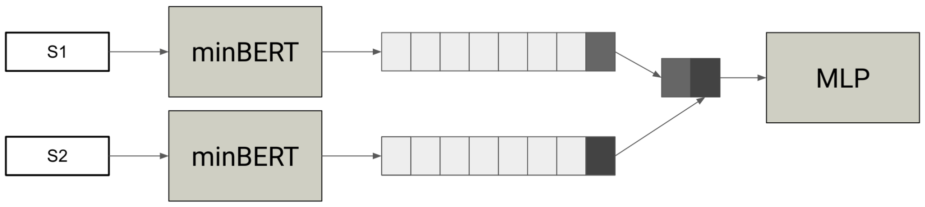
| Dataset | Template of Entailment Input |
|---|---|
| SST | [CLS] sentence$_1$ [SEP] the sentiment of this sentence is <label> [EOS] |
| QQT | [CLS] sentence$_1$ [SEP] sentence$_2$ [EOS] |
| STS | [CLS] sentence$_1$ [SEP] sentence$_2$ [EOS] |

Table 1: Entailment Template

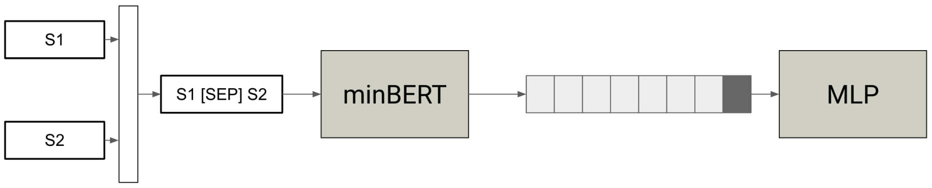
### 3.2 Tokenization

In the provided implementation, the [CLS] token was obtained by concatenating the embeddings along the channel dimension corresponding to the [CLS] token from minBERT after passing the two sentences separately. As such, the combined token would not be capturing the inter-relation of the two sentences, and instead was only capturing the meaning of the two sentences individually. This method is problematic for our approach, given that the [CLS] token of the input sentence for SST would be identical for different sentence pairs.

In our implementation, we passed sentence$_1$ [SEP] sentence$_2$ [EOS] into minBERT and took the embeddings for the [CLS] token, capturing both the meaning of the two sentences as well as the relationship between the two sentences (see Figure 2).



(a) Original Sentence-Pair Tokenization



(b) Modified Sentence-Pair Tokenization

Figure 2: Changes introduced to sentence-pair tokenization

### 3.3 Architecture

Given the goal of developing a single model that can simultaneously perform sentiment analysis (SST), paraphrase detection (QQT), and semantic analysis (STS), we attached a per-task multi-layer perceptron (MLP) to the minBERT backbone (see Figure 3). Each head includes 3 layers, with GELU activation (Hendrycks and Gimpel, 2016) and 0.5 dropout (Srivastava et al., 2014) included in the first 2 layers, and a sigmoid activation in the last layer. Each task's head outputs a single logit, which is used as described in section 3.5.

For the sentiment analysis and the paraphrase detection tasks, we use BCE as our loss given they are classification tasks. On the other hand, for semantic textual similarity, we use MSE as it is a regression task.

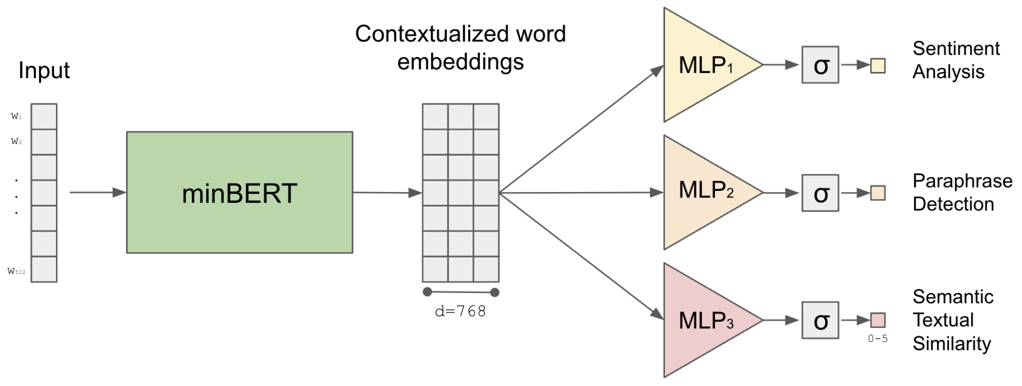


Figure 3: Multitask minBERT

## 3.4 Data

In this project, we have used the following task-associated datasets. We split each dataset in 3 categories: train, dev, and test.

**Stanford Sentiment Treebank (SST) dataset**

Sentiment analysis is the task of classifying the polarity of a sentence - whether the expressed opinion is positive or negative. The Stanford Sentiment Treebank Socher et al. (2013) consists of 8544 training examples, 1101 dev examples, and 2210 test examples. The dataset was parsed with the Stanford parser, with each phrase being annotated by 3 human judges with a label of negative, somewhat negative, neutral, somewhat positive, or positive.

**Quora Dataset**

Paraphrase Detection is the task of finding paraphrases of texts in a large corpus of passages. Quora released a dataset that labeled whether different questions were paraphrases of each other Aghaebrahimian (2017). The Quora dataset consists of 141498 train, 20212 dev, and 40431 test examples.

**SemEval STS Benchmark Dataset**

The semantic textual similarity (STS) task seeks to capture the notion that some texts are more similar than others Agirre et al. (2013). The SemEval STS dataset consists of 6040 training, 863 dev, and 1726 test examples. Each sentence-pair includes a label with a value between 0 (unrelated) to 5 (equivalent meaning).

## 3.5 Training

### 3.5.1 Task batch sampling

As seen in section 3.4, the QQT dataset is much larger than the SST and STS datasets. Training the tasks sequentially would lead to very poor performance for sentiment analysis and semantic textual similarity, as parameters from the BERT encoder drift to those more optimal for creating representations that are useful for the paraphrase task. Thus, we worked on balancing the datasets and interleaving batches from the datasets together to train our model more effectively.

To implement this, at each iteration, we sample a batch from each of the SST, QQT, STS training datasets, compute the losses for each individual batch given the task, and compute a weighted sum to obtain the cumulative loss (Equation 1) to compute the gradients [1].

[1]In our work, all $\alpha$ are equal. While usually, weighting by the number of examples would make sense, and as such, $alpha_1$ should be smaller, we wished to boost the performance for sentiment analysis, and hence kept equal weights.

$$\mathcal{L} = \alpha_1 \cdot \mathcal{L}_{\text{SST}} + \alpha_2 \cdot \mathcal{L}_{\text{QQT}} + \alpha_3 \cdot \mathcal{L}_{\text{STS}} \tag{1}$$

where $\sum_i \alpha_i = 1$

#### 3.5.2 Training parameters

For all entailment experiments, we took the learning rate to be $1e^{-5}$. For experiments without entailment, we used a learning rate of $1e^{-4}$. The batch size used is 48. We used AdamW as our optimizer Loshchilov and Hutter (2017). AdamW optimizer is a modification of the Adam optimizer Kingma and Ba (2014) used for stochastic gradient descent in neural network training. Using such value, our training time for 35 epochs in an NVIDIA A10 GPU (24GB) was 3h and 20 minutes. We tuned the learning rate together with other hyperparameters such as weight decay. More information can be found in the experiments section 4.

## 4 Experiments

### 4.1 Evaluation metrics

We evaluated sentiment analysis and paraphrase detection using average accuracy, and semantic textual similarity using Pearson coefficient. We provide the performance of different experiments using these metrics on the train, dev, and test datasets.

### 4.2 Experimental details

Through the development of the described approach, we run multiple experiments, including ablations, which helped us understand the power of each of the parts of the proposed method. We first set up a baseline with a three-head network trained on the 3 downstream tasks using the provided sentence tokenization. In this model, we used a cross-entropy loss for the SST dataset. After defining our baseline, we modified the tokenizer as discussed in Section 3.2. Thus, we tested the performance boost by adding sentence relationship context to the embeddings. Subsequently, we implemented entailment with counterfactual examples.

Regarding the total loss, initially, we weighted all three losses equally. However, given that the SST dataset obtained the lowest score, further experiments were run to try to improve the performance of the model, including a weight increase of the SST loss. Since for every true example of entailment in SST, we use $4$ false examples, the effective weights were $\alpha = [0.84, 0.08, 0.08]$. Finally, given that we observed overfitting to the training set, we also ran experiments to understand the effect of weight decay on model performance. To test if the performance boost was coming due to the entailment or from the binary cross-entropy loss formalization, we ran a final extra experiment, where we used $5$ separate heads with binary cross-entropy loss for the SST dataset. We refer to this as "one-vs-rest" strategy. Below, we provide a detailed list of the experiments we run.

1. Multitask learning without entailment, with provided implementation of tokenization. We refer to this as our baseline.
2. Multitask learning without entailment, with modified implementation of tokenization.
   ** All experiments below include the modified implementation of the tokenization step.
3. Multitask learning without entailment and one-vs-rest head for SST
4. Multitask learning with entailment, with equal weights for the three loss functions
5. Multitask learning with entailment with loss weighted in ratio of $[0.84, 0.08, 0.08]$
6. Multitask learning with entailment and weigh decay of $1e^{-6}$

### 4.3 Results

The performance on the dev dataset is shown in Table 2, whereas the performance on the test dataset is seen in Table 3.

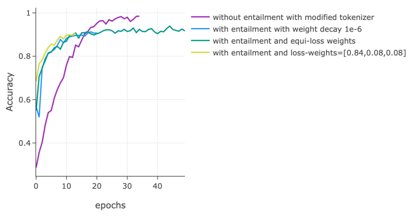
| *Experiment* | **Overall** | **Sentiment Analysis SST (Acc)** | **Paraphrase Detection (Acc)** | **Semantic Textual Similarity (Pearson Correlation)** |
|---|---|---|---|---|
| **Baseline Multitask** | 0.284 | 0.253 | 0.506 | 0.093 |
| **Without entailment modified tokenizer** | 0.740 | 0.504 | 0.857 | 0.844 |
| **Without entailment (one vs all for SST)** | 0.740 | 0.506 | 0.852 | 0.857 |
| **Entailment equi-loss weights** | 0.748 | 0.477 | 0.869 | 0.868 |
| **Entailment + loss weights** $= [0.84, 0.08, 0.08]$ | - | 0.462 | 0.82 | 0.861 |
| **Entailment +** $1e^{-6}$ **weight decay** | - | 0.449 | 0.857 | 0.867 |

Table 2: Results on the dev dataset

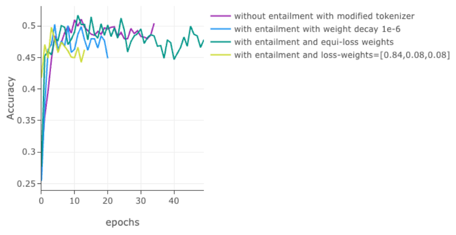
| *Experiment* | **Overall** | **Sentiment Analysis SST (Acc)** | **Paraphrase Detection (Acc)** | **Semantic Textual Similarity (Pearson Correlation)** |
|---|---|---|---|---|
| **Without entailment modified tokenizer** | 0.744 | 0.497 | 0.866 | 0.868 |
| **Without entailment (one vs all for SST)** | 0.744 | 0.522 | 0.851 | 0.60 |
| **Entailment equi-loss weights** | 0.746 | 0.524 | 0.855 | 0.861 |

Table 3: Results on the test dataset

As seen in table 2, modifying the tokenizer led to most improvement in our scores, with scores jumping from $0.284$ to $0.740$ on the dev dataset. Incorporating entailment further improved the scores to $0.748$. Looking at the results, it is evident that the room for most improvement lied in improving the model's performance on the sentiment analysis task. This made sense since this task was the most difficult owing to the multi-class classification nature of the task. This was the motivation to increasing the weight for the losses from SST. Surprisingly, that did not lead to any improvement in the performance in SST. Moreover, adding weight decay to prevent overfitting did not provide a score improvement either. As expected, we verified that the performance boost was indeed coming from the entailment formulation, which was verified by the lack of performance improvement in the one-vs-rest strategy. Figures 4, 5, & 6 show the performance of each of the aforementioned 6 models. We provide the evolution of the train and dev scores for each of the three datasets.

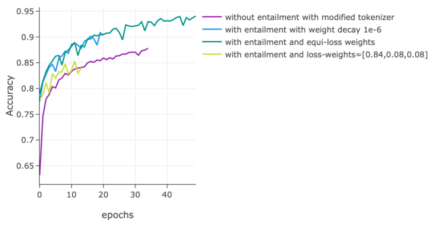


(a) Accuracy on SST-train

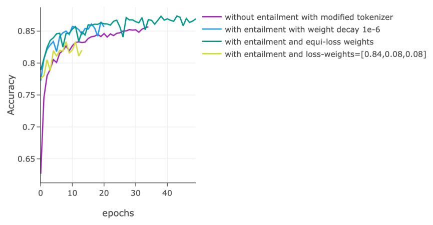


(b) Accuracy on SST-dev

Figure 4: Accuracy on SST dataset

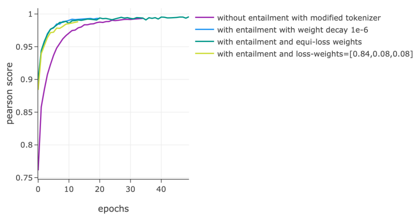


(a) Accuracy on QQT-train

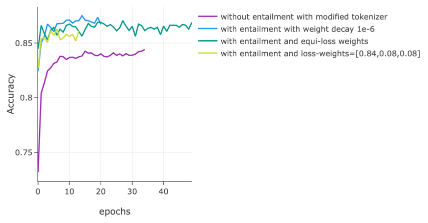


(b) Accuracy on QQT-dev

Figure 5: Accuracy on QQT dataset

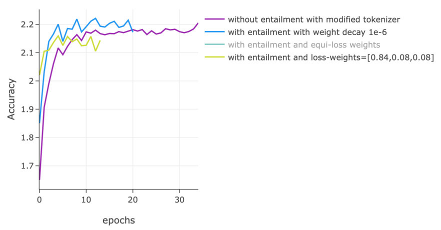


(a) Pearson score on STS-train

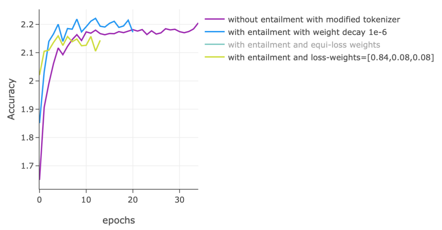


(b) Pearson score on STS-dev

Figure 6: Pearson score on QQT dataset



(a) Accuracy on training data



(b) Accuracy on dev data

Figure 7: Total accuracy curve

## 5 Analysis

Analyzing the performance of the model, we acknowledge that a marginal improvement of the model could have been achieved with more training epochs. Looking at the loss curve, it can be seen that the majority of the learning occurs within the first 20 epochs. Out of the 3 datasets, our model performed the poorest on the SST dataset. However, these results were expected as fine-grained sentiment analysis is a harder task to learn compared to paraphrase detection and semantic evaluation, due to the multi-class nature of the problem. We see that increasing the number of negative samples helps boost the performance of the model. This can be understood with the "push-pull" nature of the contrastive loss method for our approach for training with entailment.

Overall, it became clearer that correctly contextualized embeddings matter a lot for achieving peak performances in natural language processing tasks. When we introduced the modification of the tokenizer to account for sentence-pair embeddings using the `[SEP]` token, we observed the biggest boost in accuracy for the three tasks. Finally, introducing entailment for SST, not only increased the accuracy for that dataset, but it also increased the accuracy of QQT and STS datasets. Thus, the unified structure led to better embeddings for the three tasks.

## 6 Conclusion

All of today's well-known language models have a massive size neural network underpinning them. Despite their unparalleled strength, one of their major shortcomings is the unfeasibility of loading and reloading different models in a system. As such, it has become critical to develop a single model which can solve multiple downstream tasks, reducing the infrastructure load. In this paper, we establish baseline performance on sentiment analysis, paraphrase detection, and semantic similarity tasks using an implementation of the BERT language model.

We experimented with the use of entailment learning to generate robust representations, enabling the use of a single model for multiple downstream language tasks. We used entailment to reformulate the multi-class classification problem into a binary classification problem. For entailment learning to work, it is not only necessary to provide positive examples of entailment but also generate false examples which do not have entailment, as seen in contrastive learning approaches.

Furthermore, by using clever training approaches, such as dataset batch interleaving, we obtain a model which is able to perform well for SST, QQT, and STS datasets at 0.524, 0.855, and 0.861 accuracy. Comparing the performance with the current SOTA models which are trained on one single downstream task and are massive compared to our model architecture, we perform better than models till 2017. Hence we argue, that entailment learning allows to learn more generalizable representations of weights.

In the future, we would like to work on incorporating further techniques to improve the performance for multi-class classification and provide more rigorous study into the cause of success of entailment learning. Besides, we would like to explore different contrastive loss formulations as well further explore hyperparameter tuning.

## References

Ahmad Aghaebrahimian. 2017. Quora question answer dataset. In *Text, Speech, and Dialogue: 20th International Conference, TSD 2017, Prague, Czech Republic, August 27-31, 2017, Proceedings 20*, pages 66–73. Springer.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. * sem 2013 shared task: Semantic textual similarity. In *Second joint conference on lexical and computational semantics (* SEM), volume 1: proceedings of the Main conference and the shared task: semantic textual similarity*, pages 32–43.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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

Jiatao Gu, Yong Wang, Yun Chen, Kyunghyun Cho, and Victor OK Li. 2018. Meta-learning for low-resource neural machine translation. *arXiv preprint arXiv:1808.08437*.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR*, abs/1606.08415.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Xiaodong Liu, Hao Cheng, Pengcheng He, Weizhu Chen, Yu Wang, Hoifung Poon, and Jianfeng Gao. 2020. Adversarial training for large neural language models. *arXiv preprint arXiv:2004.08994*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Sinong Wang, Han Fang, Madian Khabsa, Hanzi Mao, and Hao Ma. 2021. Entailment as few-shot learner. *arXiv preprint arXiv:2104.14690*.