Multi Class Sentiment Analysis using Transformers

Max Mai, Kevin Lucey, Nathan Schambach
3 April 2023

Abstract

In this paper we apply two different models to the task of fine grained sentiment classification on the Stanford sentiment treebank dataset (SST2). We create a bidirectional long short term memory model (Bi-LSTM) as well as a model combining the bidirectional encoder representations from transformers (BERT) architecture with a dilated convolutional neural network (BERT+DCNN), and show that BERT+DCNN is able to compete with the best accuracies achieved for this task and dataset. We show that BERT+DCNN can achieve accuracies which meet or exceed the accuracy of the recursive neural tensor network, or RNTN (Socher et al. 2013) using a five-fold cross validation procedure. Finally, we discuss some of the additional benefits of BERT+DCNN's attention mechanisms over the recursive architecture of the RNTN.

Introduction

Sentiment analysis is a useful tool for quantifying how users feel about a particular product or subject. However, the intricate interrelationships of natural language and the number of dimensions required to properly represent them leads to a number of challenges that need to be overcome. First, sentiment depends highly on the context of words in the sequence which in turn depends on the placement of each and every word. As a result, some of the primary tools applied to this problem have been recurrent and recursive neural networks since they use information about previous words when considering the next word in a sequence. Although robust, the recurrent and recursive nature of these models makes them difficult to parallelize

which is a serious drawback when considering the vast size of most major text databases. Furthermore, the recurrent and recursive models outlined by Socher et al. required the roughly 11,000 reviews contained in the SST2 dataset to be broken into more than 200,000 phrases, each manually labeled.

More recent developments in natural language processing (NLP) have seen staggering levels of success using models based on attention and transformers from “Attention is All you Need” (Vaswani et al 2017). Similar to recurrent and recursive architectures, these models can learn contextual relationships based on the positions and semantic composition of the words, however one of the major benefits is that the entire text is processed concurrently making it easily parallelizable. Additionally, there have been a number of novel, unsupervised training techniques developed around this architecture which can be used to learn general information about natural language.

Our goal is to evaluate an attention based model against the recursive and recurrent techniques employed by Socher et al. both in terms of their fine grain accuracy and in terms of their scalability. We will employ a model based on Bidirectional Encoder Representations from Transformers (BERT Base) consisting of twelve encoder blocks, and a fully-connected, dilated CNN will be used to extract feature maps from the encoded output sequence (BERT+DCNN). As another point of comparison, we will also evaluate a bidirectional long short-term memory (Bi-LSTM) model since it is a well tested and commonly used representative of RNN architecture. This research will provide insight into some of the many challenges of NLP, and propose a model that seeks to meet or exceed the performance of RNTN while incorporating all of the benefits of using an attention based architecture. Lastly, our model will be more easily parallelized and can receive continued training on unlabeled data or fine tuning on labeled data due to its BERT genes.

Data

We will be working with the Stanford Sentiment TreeBank (SST2) dataset which consists of 11,855 single-sentence, movie reviews. Each sentence has been represented as a binary tree of phrases with the root of the tree representing the full sentence, and each leaf representing a single phrase (Figure 1). This method results in 215,154 unique phrases and each has been given a sentiment score which we will be dividing into five categories. We will be using the splits provided by Socher et al. with 8,544 reviews in train, 2,210 in test, and 1,101 in dev. The target sentiment scores are integers between 0 and 4 with a value of 2 representing a neutral sentiment, 4 being the most positive, and 0 being the most negative. Distributions of the sentiment labels for the train and test set are shown in figures 2 and 3. As we can see the labels follow a bimodal distribution with people tending to select intermediate values rather than true neutral or either extreme. Thus, “even a 5-class classification into these categories captures the main variability of the labels” (Socher et al, 2013).

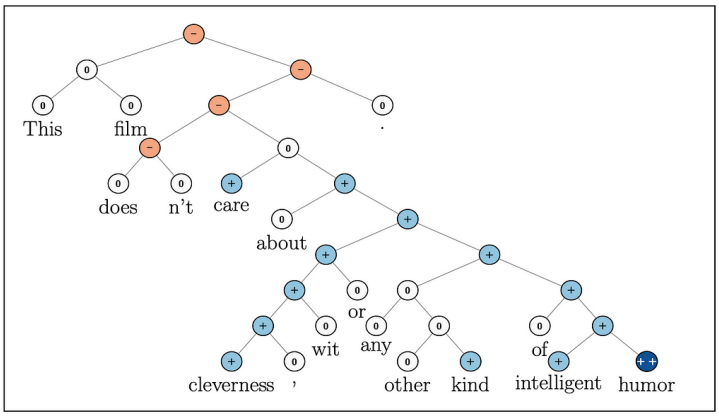

Figure 1: An example binary tree for a single review. Note each node contains a sentiment value

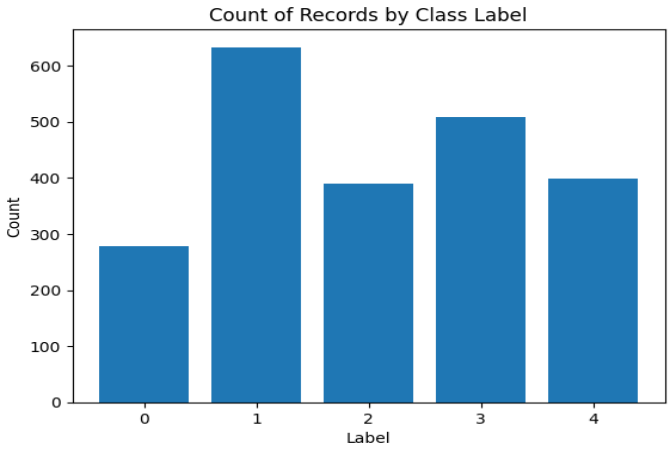

Figure 2: Distribution of sentiment labels in validation data

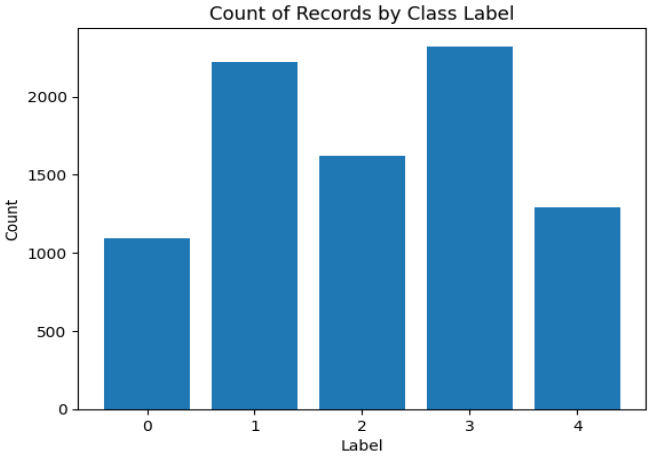

Figure 3: Distribution of sentiment labels in train data

Related Work

Socher et al. developed a Recursive Neural Tensor Network (RNTN) to build upon the group's previous models. They employed an architecture which relies on traversing the binary trees starting from the leaves and relating the embedded vectors of children and parent nodes through a dot product relationship. The model demonstrated exceptional accuracy in the fine grained classification task while simultaneously pruning some of the complexity when compared to other recursive methods. Additionally, the group compared their models' accuracy against a collection of other architectures such as RNNs and SVMs at both the sentence root and phrase

level. They were able to achieve a maximum validation accuracy of 47.5% accuracy at the sentence level using their RNTN architecture.

Model	Fine-grained	
	All	Root
NB	67.2	41.0
SVM	64.3	40.7
BiNB	71.0	41.9
VecAvg	73.3	32.7
RNN	79.0	43.2
MV-RNN	78.7	44.4
RNTN	**80.7**	**45.7**

Figure 4: Accuracy for phrase and root level fine grained sentiment analysis by model (Socher et al. 2013)

Vaswani et al. laid the basis for the transformer architecture which was built solely around the concept of self attention, hence the name “Attention is All You Need”. They demonstrate the predictive power of this architecture as well as discuss some of the key benefits with regards to the ease of computing and the training capabilities.

Dong et al. experimented with BERT alongside a convolution layer to perform a binary sentiment analysis on digital product reviews. It was found that the combination of these two techniques resulted in increased F1 when compared to BERT or CNN alone.

Gu et al. used a dilated convolutional neural network alongside a self attention mechanism for the purpose of cross domain sentiment analysis. They were able to achieve an accuracy of 83% when predicting binary sentiment labels for their dataset. Furthermore, they discuss how attention and convolution make a good pair since they can both be calculated using efficient parallelized methods. It was also found that dilating the convolutional layers allowed for more accurate extraction of the long range contextual relationships between words.

Dos Santos et al. used the Stanford Sentiment Treebank and the Stanford Twitter Sentiment corpus to analyze the sentiment of tweets. The researchers discuss how shorter texts are difficult to accurately label as positive/neutral/negative as there lacks context that is crucial to understand the meaning of a tweet. To get around this issue, the researchers developed a

model to get vector embeddings at the character level, and another model that would take these character embeddings as an input then output word level embeddings.

Preprocessing

In order to process our data for input to both the Bi-LSTM and BERT model, unique integer ids will be assigned to each word in the dataset in order to create a vocabulary. We found that when splitting on whitespace, we required 21,701 token ids in our dictionary to represent all 11,855 sentences in the SST2 dataset. BERT on the other hand uses a dictionary with 30,522 token ids to represent the wikipedia and books corpus containing millions of sentences. Each token id in an input sentence will be embedded into a real valued vector of constant dimensions with initially random values resulting in a matrix Q of size $maximum\ sequence\ length\ \times\ embedding\ dimension.$ Both models will learn to separate these vectors within the space such that words with similar sentiment are close to each other in the embedding space. We found that the Bi-LSTM model performed optimally with an embedding dimension $30\ \leq d\ \leq\ 60$ and we used the provided embedding dimension of 768 in BERT+DCNN.

Additionally, for BERT we also create a positional encoding matrix of the same dimensions as the embedded sentence which carries information about the order in which words appear. The positional matrix is created deterministically with an element-wise application of a piecewise sinusoidal function on Q (Figure 5) with the resulting matrix being added back to our matrix of embedded words.

$$f(k,j) = \left\{ \begin{array}{ll} sin(\frac{k}{10000^{\frac{j}{d}}}), & \text{if } k \text{ even} \\ cos(\frac{k}{10000^{\frac{j-1}{d}}}), & \text{if } k \text{ odd} \end{array} \right\}$$

Figure 5: Function used to create the positional encoding matrix where k is the integer position of a word (or row number), j is the column number or embedding feature, and d is the dimension of the word embedding space.

The purpose of embedding is to reduce the number of dimensions required to represent our vocabulary, and to allow a continuous metric for similarity between words in our space. Specifically, to represent a space of integer tokens we would need to perform a one-hot encoding of each token meaning that the space requires as many dimensions as there are words in our vocabulary. By mapping our integer ids into vectors of real numbers, we can greatly reduce the number of dimensions required to properly represent our input space. In essence we allow the entries of each vector to take on more values meaning that we need fewer entries to encode the same amount of information. The result is a significantly more dense representation of our input space embedded in $\mathbb{R}^d$.

Models

Natural language is an inherently difficult domain due to the challenges that come with quantifying it within a reasonably sized input space. For Bi-LSTM we found that 21,701 tokens were required to represent our vocabulary when splitting tokens on white space. This is an incredibly diverse vocabulary for a relatively small amount of text. For reference, BERT Base, which has been trained on the wikipedia and bookscorpus uses around 31,000 tokens to describe the vocabulary of millions of text examples. Additionally, both models were trained

using categorical cross entropy loss and an Adam optimizer with learning rate = .001, $\beta_1 = .9$, and $\beta_2 = .999$.

To review, bidirectional long short term memory (LSTM) involves passing the input sequence through an LSTM neural network first forward, then backward hence the name 'bidirectional'. Each long short term memory layer consists of a cell, and three gates: input, output, and forget. At a high level, the gates are responsible for controlling the flow of information into and out of the cell. Specifically, since this is a recursive architecture the gates determine what information is remembered from the previous state and which is passed on as input to the next cell in the network. Mathematically, this is achieved through simple sigmoid activation functions along with a tanh nonlinearity. The sigmoid functions map matrices representing the previous hidden state and the predictors, to matrices of the same dimension with real values in [0,1]. These matrices act as a binary mask which, through multiplication with the previous cell state, can ignore certain values by setting them to 0 while retaining others. Since tanh maps values to [-1,1], this nonlinearity is responsible for incrementing and decrementing the values within the cell state through addition. Also, since the absolute values within the cell state are unbounded, another tanh nonlinearity is used to normalize the values in the current cell state before passing it forward to the next cell. One of the main hurdles with recursive neural networks is the unstable gradient problem in which our gradients either vanish or explode in magnitude over many timesteps. This can prevent the model from back propagating meaningful changes to layers towards the input end of the network. Bi-directional LSTM seeks to overcome these issues by reading the sequence in both directions, deciding which inputs to consider, ignore, or forget at each timestep, and by employing a tanh nonlinearity whose second derivative has greater range than that of sigmoid. For our results we used an embedding dimension of 35 with a Bi-LSTM layer containing 64 cells and the standard sigmoid activation and tanh nonlinearity. The output is then passed through a dense layer with

64 units and rectified linear unit activation before a final dense layer with sigmoid activation and 5 units. We also apply dropout in the LSTM at a rate of .4 in order to prevent overfitting. As stated, the input text has a very diverse vocabulary with relatively few training examples and in testing we found that there was not enough information to generalize the patterns learned from our training set. Since the embedding vectors are learned through training, and we have such a low example sequence to vocabulary size ratio it can be difficult for the model to meaningfully differentiate between the word vectors in the embedding space.

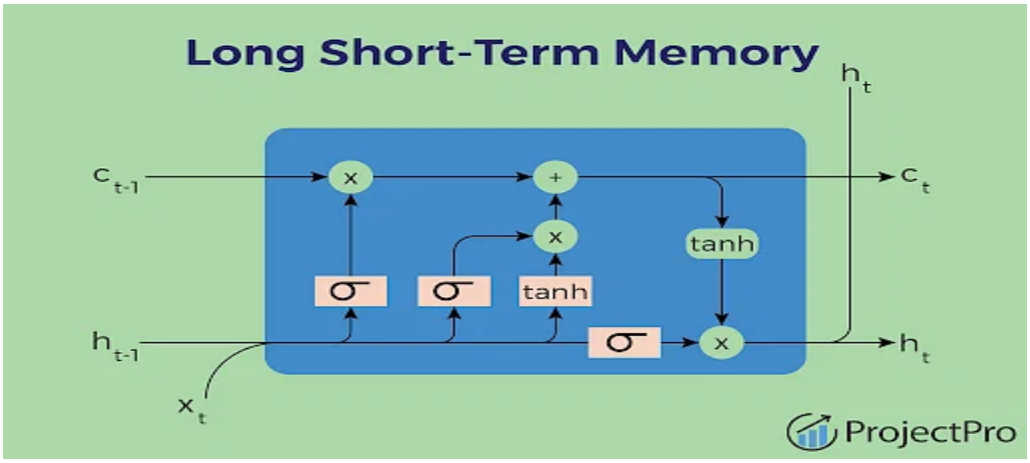

Figure 6: A single LSTM cell

Although this architecture can overcome many of the limitations of simple RNNs, it is still fundamentally constrained by the fact that it must process each element of the input sequentially. Furthermore, it is difficult to continue training with these models due to the volume of manual labeling required. Fortunately, with the advent of the transformer architecture, we can continue to extract contextual relationships between words while evaluating all elements of our input concurrently. Additionally, with the use of a masking layer, models can be trained in a semi-supervised manner by hiding words from the input and forcing the model to guess their value based on the surrounding words.

The BERT Base architecture we use, employs twelve encoder blocks each consisting of a multi headed attention layer alongside linear and normalization layers. Each sequence of

words is encoded using integer tokens and each token is embedded into a 768 dimensional vector of real numbers. As stated this matrix is added to a positional embedding matrix, with the result acting as the key (K), query (Q), and value (V) input matrices for the multi-head attention layers.

At the core of each multi-head attention layer is a block of scaled dot product self attention layers. Each self attention layer has a unique set of weights used to evaluate the scaled dot product attention of the key (K), query (Q), mask (M), and value (V) matrices (Figure 6).

$$Attention(Q, K, V, M) = softmax(\frac{Q \cdot K^T}{\sqrt{d}} + M) \cdot V$$

Figure 7: Scaled Dot product attention, where d is the dimension of our embedding space

The mask (M) can be thought of as a matrix of zeros with a percentage of the values being randomly assigned to $-\infty$. When we add M to the matrix produced by the scaled dot product of Q and K^T, every entry in the product matrix is either left unchanged or set to $-\infty$, which gets mapped to 0 by the softmax function. This operation results in some percentage of the embedding space being hidden or masked from the attention layer. Another important thing to note is that the product of a matrix and its transpose (i.e. QK^T) relates each row of Q to every column of K^T, and given that $K = Q$, the product QK^T relates each word to every other word in a given sentence. Thus each head in our multi-head attention layer can be thought of as attending to a particular type of relationship between each of the embedded word vectors in the input sequence. Furthermore, each head is calculated independently on the same inputs before the outputs are concatenated (Figure 7), meaning that the heads can be evaluated concurrently.

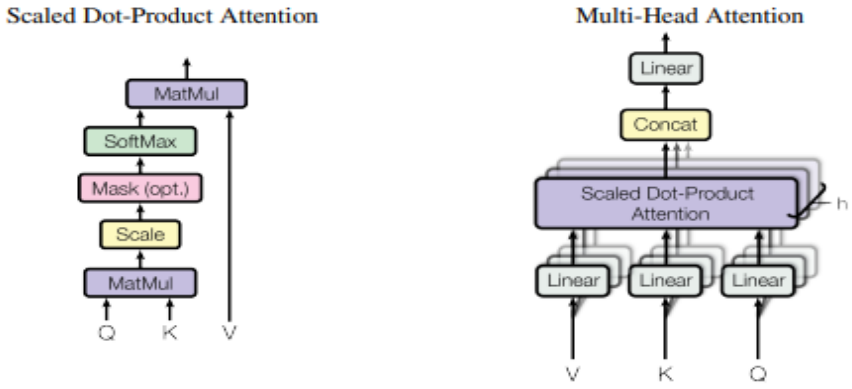

Figure 8: A single scaled dot-product attention head (left), a multi-head attention layer with h scaled-dot product attention heads (right)

The outputs of the multi-head attention layer are then added back to the embedded inputs before normalizing with mean, standard deviation normalization. This operation ensures that the model retains the information learned from the previous layer. Lastly, the summed matrix is passed through a linear layer before finally adding the summed matrix to the outputs of the linear layer and normalizing the result (Figure 8).

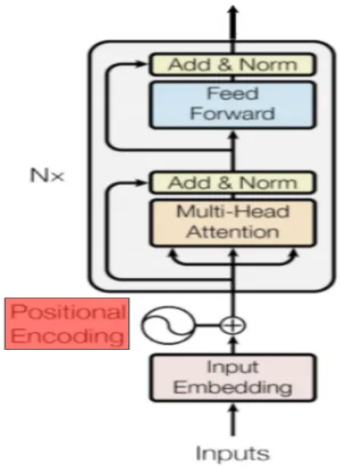

Figure 9: A transformer encoder block

As stated, BERT Base employs 12 of these encoder blocks (Figure 9), before producing two separate outputs. The sequential output is a sequence of embedded token vectors, while the pooled output takes the first embedded token, also called the classification token, and passes it through a dense layer with 768 units and tanh activation before returning. We will use both of these outputs when making our classification.

In order to extract further information from the sequence of embedded token vectors we will use a dilated convolutional neural network consisting of three 1d convolution layers each with a kernel size of three, and 16, 64, and 128 filters respectively. Since a standard 1d convolution kernel is a vector, the feature maps it produces can only relate tokens that are directly neighboring each other. However, we will dilate the kernel by adding space between each unit, allowing us to capture contextual relationships between distant embedded vectors. Furthermore, each stride of a convolution can be calculated concurrently meaning that this addition does not diminish the computational benefits of attention mechanisms.

In BERT+DCNN we use a linearly decreasing rate of dilation starting at three to balance between extraction of long and short range contextual relationships. After each of the first, and second convolutional layers we employ max pooling to reduce the size of our feature maps while retaining the most useful information. After the final convolutional layer we apply global max pooling in one dimension to collapse our input to a single vector and aggregate the values in our feature map. Both of the pooling steps also serve to reduce the total number of parameters in the network. Next, the pooled output and DCNN output are passed through dense layers of 64 units with rectified linear unit activation before being concatenated and passed through a final layer of 5 units with sigmoid activation. The final result is a vector of length 5 encoding a probability distribution for our sentiment classes.

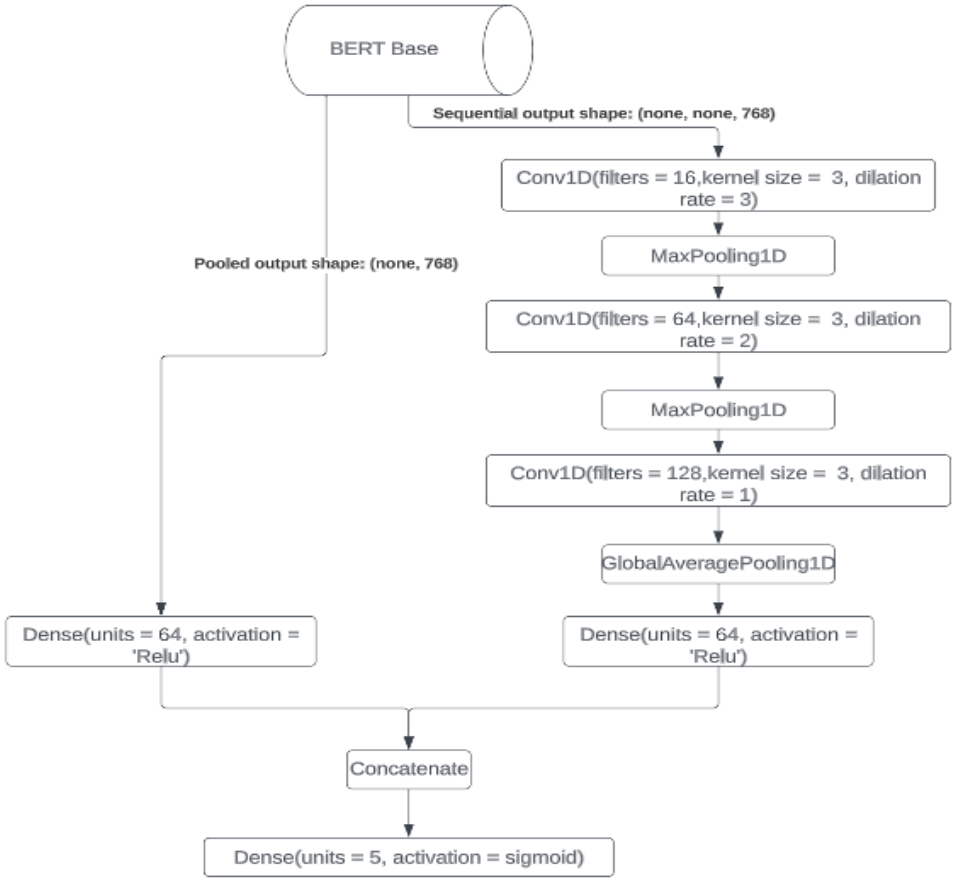

Figure 10: A graph of the BERT + DCNN architecture, where the BERT block represents the BERT Base model pre trained on the wiki books corpus

Experiments and Metrics

As stated, we will be using the splits provided by Socher et al, consisting of 8,544 training reviews and 2,210 validation reviews with approximately equal sentiment label distributions. We will compare the accuracy of Bi-LSTM and BERT+DCNN against each other and against the models tested by Socher et al using a 5-fold cross validation on the same splits with randomly initialized weights and batch shuffling. We will use the mean of the maximum validation accuracy achieved over 10 epochs with a batch size of 32 as our main point of comparison against Socher et al. In addition, since we will be using the same train test split as

them, we can evaluate 95% confidence intervals on the overall accuracy of each model by calculating the standard error of the population proportion (Figure 11). Furthermore, for Bi-LSTM we will graphically evaluate the effects of dropout, along with lasso and ridge regularization in relation to the model's train and test accuracy per epoch. We expect regularization to reduce overfitting since it adds a penalty to our loss function based on the values of our weights.

$$acc \pm 1.96 * \sqrt{\frac{acc*(1-acc)}{n}}$$

Figure 11: Confidence interval for the overall accuracy

Results

The results from our 5-fold validation for the maximum accuracy achieved over 10 epochs with a batch size of 32 are pictured in figure 18. We find that the Bi-LSTM model had poor performance compared to the simple RNN model tested by Socher et al. with a mean accuracy of 38.75% and a standard deviation of .0058. One of the consistent issues we noticed with this architecture is the tendency to overfit the train data set (Figure 12-13).

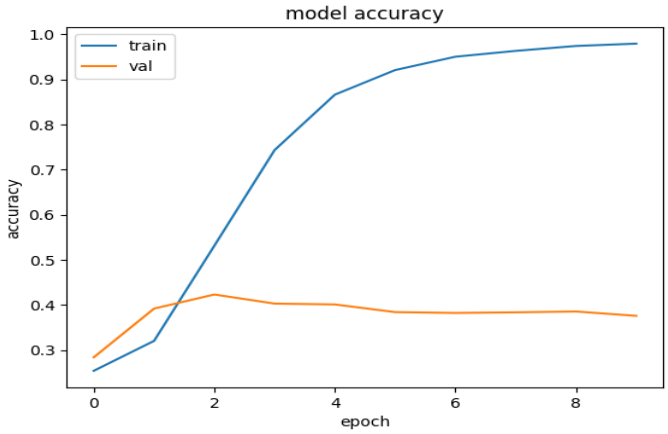

Figure 12: Train vs validation accuracy over 10 epochs with Bi-LSTM, dropout rate of .4, batch size of 32

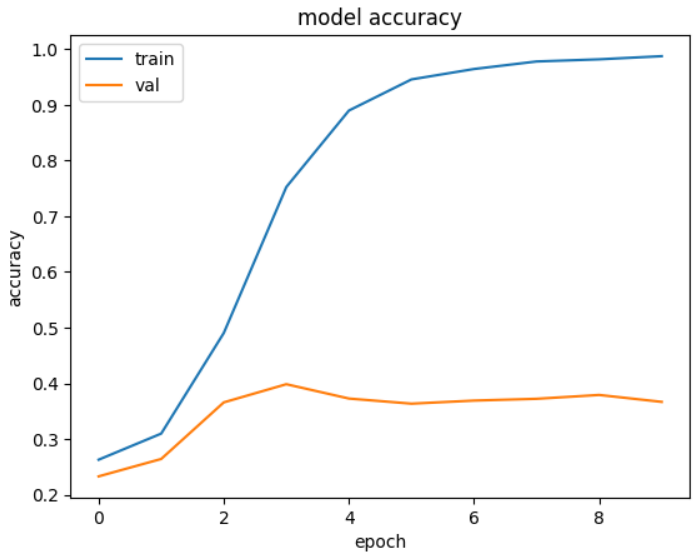

Figure 13: Train vs validation accuracy over 10 epochs with Bi-LSTM, dropout rate of .2, batch size 32

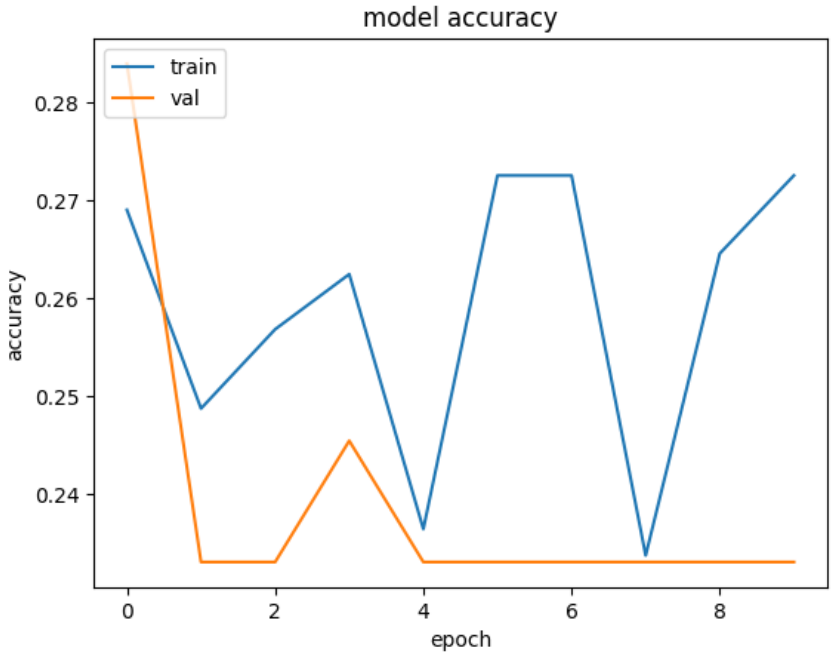

Figure 14: Train vs validation accuracy over 10 epochs with Bi-LSTM, dropout rate of .4, batch size 32, and lasso kernel regularization

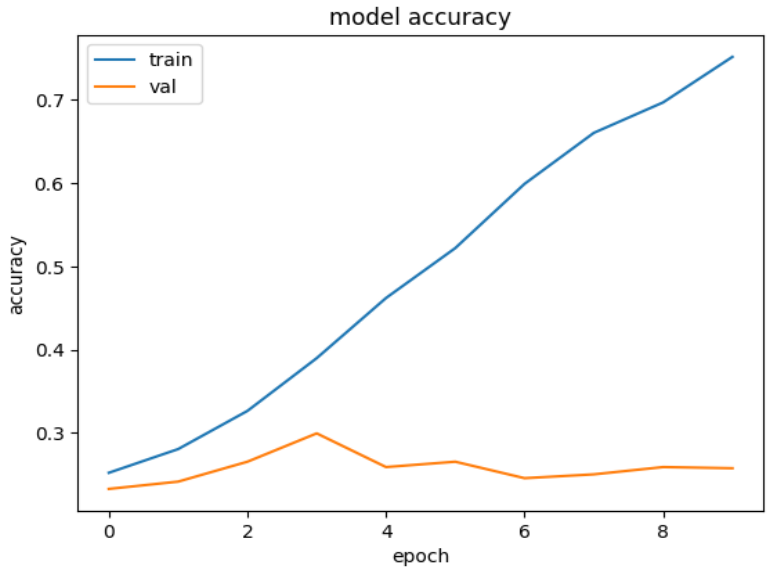

Figure 15: Train vs validation accuracy over 10 epochs with Bi-LSTM, dropout rate of .4, batch size 32, and ridge kernel regularization

Dropout	Regularization	Maximum Validation Accuracy Achieved	95% CI Lower Bound	95% CI Upper Bound
0.2		39.84%	37.80%	41.88%
0.4		40.03%	37.99%	42.07%
0.4	Lasso	24.54%	22.75%	26.33%
0.4	Ridge	29.96%	28.05%	31.87%

Figure 16: Maximum validation accuracy achieved and 95% confidence intervals for various dropout rates and regularization methods on Bi-LSTM

The primary cause for this is due to the diversity of the vocabulary compared to the number of training examples. Each embedded vector is randomly initialized, but learned over

time according to our categorical cross entropy loss function. This means that the embedding space starts off filled with gaussian noise, and the model must see enough examples to meaningfully differentiate between the word vectors in the embedding space. Furthermore, the model ignores words that are not present in its vocabulary. In combination these factors can make it very difficult for the Bi-LSTM model to generalize the patterns it learns during training.

During our experiments we evaluated the effects of adjusting the dropout rate, and adding regularization layers to Bi-LSTM in order to prevent overfitting. We sampled the maximum validation accuracy achieved during a ten epoch training session with a batch size of 32 for dropout values of .2 and .4, as well as for ridge, lasso, and no regularization of the LSTM kernel. We found that although lasso and ridge regularization decreased the tendency to overfit on the training data, they also came with substantial (>10%) decreases in the validation accuracy (Figures 12-16). Furthermore, we noticed that increasing dropout rate simply delayed the models learning on the train data set with no significant effects on validation accuracy. As seen in figure 16, the 95% confidence intervals for each dropout values' validation accuracy have a large region of overlap when using no regularization. We also observe in figures 12 and 13 that the train and test accuracy curves appear to be equivalent, just translated along the x axis. This indicates that we are likely just delaying the overfitting problem rather than solving it by increasing dropout rate, and further shows that we may not have enough examples to properly differentiate the vocabulary in the embedding space.

As stated, our model is built on top of BERT Base, which has been pre trained on the wikipedia and books corpus datasets consisting of millions of words. This means that it has seen enough natural language to map out a large, but well structured embedding space. In other words, the model has explored enough of the space of all English language that it has learned a general 'understanding' of semantic and contextual relationships between word vectors. This means that it will overfit to our train data less severely than the Bi-LSTM which can be seen clearly when comparing figures 12 and 13 to figure 17. Since the positions of vectors in

BERT's embedding space already encodes some information about their context and semantic meaning, we require much fewer labeled examples to fine tune the model.

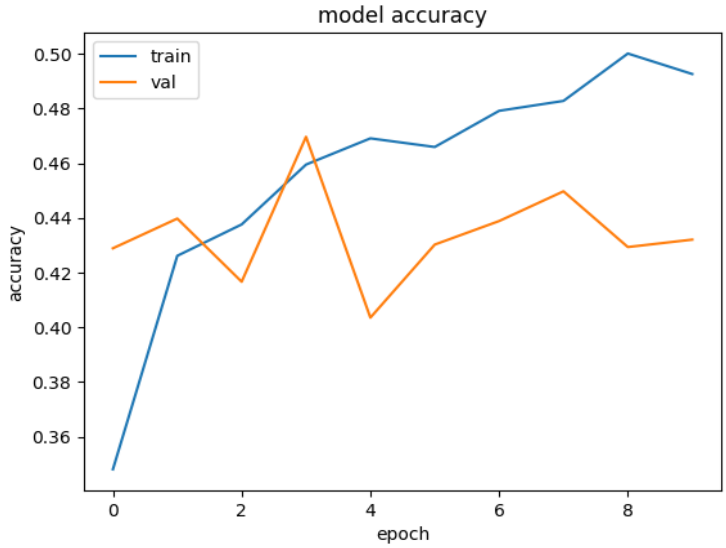

Figure 17: BERT+DCNN train vs validation accuracy per epoch with batch size of 32

To compare our models, we calculated 95% confidence intervals using the mean of the maximum accuracy achieved during our 5-fold cross validation, and the root level accuracy reported by Socher et al. The results for each trial of Bi-LSTM and Bert+DCNN are pictured in Figure 17, and we can see that the mean of the maximum validation accuracies achieved by BERT+DCNN is 46.06% with standard deviation of 0.005849, which exceeds the maximum reported accuracy of all other models under consideration (Figure 18). Furthermore, BERT+DCNN reached validation accuracies as high as 46.97%.

	Trial Number						
Model	1	2	3	4	5	Mean	StdDev
Bi-LSTM	40.03%	38.28%	38.01%	38.83%	38.60%	38.75%	0.00698
BERT+DCNN	46.24%	46.97%	45.15%	46.01%	45.92%	46.06%	0.005849

Figure 18: Maximum validation accuracy achieved in 10 epochs with a batch size of 32 for each trial of the 5-fold cross validation

We found that BERT+DCNN achieved a greater mean validation accuracy during cross validation than RNTN. However, we also see that the 95% confidence intervals for the overall accuracy of BERT+DCNN and RNTN have a significant amount of overlap (Figure 19). In summary, we can say with 95% confidence that the true accuracy of BERT+DCNN lies between 43.98% and 48.14%, and that the true accuracy of RNTN lies between 43.62% and 47.78%. This is not a significant gain in accuracy, however given that the mean accuracy achieved during cross validation was greater than the reported maximum accuracy achieved by RNTN, there is evidence that BERT+DCNN either meets or exceeds the performance of RNTN.

Model	Accuracy	95% CI Lower Bound	95% CI Upper Bound
NB	41.00%	38.95%	43.05%
SVM	40.70%	38.65%	42.75%
BiNB	41.90%	39.84%	43.96%
VecAvg	32.70%	30.74%	34.66%
RNN	43.20%	41.13%	45.27%
MV-RNN	44.40%	42.33%	46.47%
Bi-LSTM	38.75%	36.72%	40.78%
RNTN	45.70%	43.62%	47.78%
BERT+DCNN	46.06%	43.98%	48.14%

Figure 19: Validation accuracy for fine grained sentiment analysis of SST2 at the sentence root level. Bi-LSTM and BERT+DCNN accuracies represent the mean of the maximum validation achieved in our 5-fold cross validation.

Conclusion:

We have shown that BERT+DCNN can compete with the state of the art accuracy in multiclass sentiment analysis for this dataset. In addition, the operations done by BERT+DCNN can be easily parallelized. In the case of multi-head attention, each head can be calculated independently. While in the dilated convolutional neural net, we can compute each stride of a convolution concurrently. Furthermore, due to BERT's implementation of masking, the base model can be easily pretrained on additional unlabelled data to produce more meaningful embeddings and thus more accurate classifications. All of these properties make BERT+DCNN an exceptional model for the task of fine grained sentiment analysis; competing with the best in terms of accuracy without sacrificing speed and scalability.

One question that still needs to be answered, is how sensitive the overall accuracy is in relation to the size of the pre-train and train data. We suspect that adding more labeled data would provide greater gains since it is specific to our downstream task, and will update the

weights of the DCNN through back propagation. Further validation is also required to confirm the performance of RNTN versus BERT+DCNN. Specifically, we only know the reported accuracies of RNTN but we do not know how these accuracies were achieved or sampled. Therefore, a further analysis of RNTN using the same k-fold cross validation technique as we employed for BERT+DCNN is necessary. Additionally, due to limitations in computing resources k was kept small at 5, but we would ideally like to increase that number to ensure a representative sample of each model's accuracy. Further research is also needed in applying BERT+DCNN to datasets with different sequence lengths and subject matter. The SST2 dataset consists only of single sentence movie reviews, so it will be important to evaluate BERT+DCNN on longer texts to confirm that it can extract longer range contextual relationships.

In conclusion, by combining pre-training, attention mechanisms, and a dilated convolutional neural network, we were able to compete with the top overall accuracy scores without losing any of the myriad benefits provided by transformer architectures.

References

Dos Santos, Cicero, and Maira Gatti. "Deep convolutional neural networks for sentiment analysis of short texts." *Proceedings of COLING 2014, the 25th international conference on computational linguistics: technical papers*. 2014.

Gu, T., Xu, G., Liu, X., & Luo, J. (2022, February). Joint Dilated Convolution and Self-attention for Cross-domain Sentiment Analysis. In *Journal of Physics: Conference Series* (Vol. 2188, No. 1, p. 012012). IOP Publishing.

J. Dong, F. He, Y. Guo and H. Zhang, "A Commodity Review Sentiment Analysis Based on BERT-CNN Model," *2020 5th International Conference on Computer and Communication Systems (ICCCS)*, Shanghai, China, 2020, pp. 143-147, doi: 10.1109/ICCCS49078.2020.9118434.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013, October). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1631-1642).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.