Exploring the Efficacy of Various NLP Models in Predicting Different Categories of Hate Speech

Brian Huang, Abhishek Mohan, Ritaank Tiwari

Abstract

In the advent of the internet virality, web anonymity, and lackluster fact checking, hate speech is a real problem in the 21st century that has seen an anonymous, rapid spread across online forums and microblogging sites. With such an increase in hate speech, the need for accurate, reliable automated tools in this area is needed more than ever. Hate speech is an umbrella term for many different kinds of discriminations, which are fundamentally different in semantics and vocabulary. Solving our task at hand thus requires nuance, precision, and exploration of approaches. Hence, we investigate different models and datasets on the efficacy of recognizing hate speech by performing dynamic, experimental evaluations for different permutations of the problem. We show that while a DistilBERT model, with its pre-trained word representations, retains the highest accuracy overall, a more simple single-stream CNN maintains more resilience to dataset noise.

1 Introduction

Hate speech is defined by the Cambridge Dictionary as "public speech that expresses hate or encourages violence towards a person or group based on something such as race, religion, sex, or sexual orientation." With increases of popularity in online platforms such as Facebook, Twitter, and Instagram to name a few, such speech can be present in these platform's forums, chatrooms, and comments. This topic has become one of much debate and conversation, in addition to being much more relevant in recent years. The subjectivity of hate speech introduces some difficulties, as a neutral sentence can be offensive to one person but not to another. Furthermore, the detection, regulation, and legality of hate speech presents a continuous issue for legislators and individuals alike, with no definitive method of classification being defined and regulations surrounding such speech still being determined.

The following statistics are important to note as they convey the increasing negative presence of hate speech:

- According to the Pew Research Center, among 4248 adults in the United States, 41% have personally experienced harassing behavior online, while 66% have witnessed harassment directed towards others.[4]
- Nearly 22% of adults have experienced online name calling, purposeful embarrassment, physical threats, and sexual harassment - to name just a few forms of harassment.[4]

At the same time, advances in machine learning, specifically in the field of natural language processing (NLP), have allowed for new, effective methods of classifying speech. Text classification has seen great progress over the past decade, with innovative, language-driven approaches being developed and tested. With such approaches, NLP technologies have been explored to better identify and handle hate speech found on the Internet.

Hate speech comes in many shapes and sizes — and we want to explore NLP model structures that are not limited to classifying hate speech in a binary fashion. Specifically, we want to look at the different kinds of hate speech — xenophobic, Islamophobic, racially charged, sexist/misogynistic, anti-LGBTQ statements, and other forms of textual harassment — and the success of different kinds of models in classifying these different forms of hate. By independently evaluating these different kinds of hate speech, we not only learn more about which models work better to classify which types of language problems, but also how to build a more applicable and precise hate speech classifier.

2 Background

We will first look at the datasets used for testing and training, in addition to the models specifically chosen for experimentation. An understanding of these two components will provide us with the background needed to understand our design choices and chosen methodologies.

2.1 Dataset Breakdown

Our ability to classify any corpus of text as containing hate speech or not lies on the fundamental properties of the dataset within. Our datasets used (described in detail below) were carefully selected for their reputability, usage rate, and inter-rater agreement. We presumed that a higher inter-rater agreement indicated a more *precise* dataset. Accuracy – determining the ground truth whether or not a corpus does in fact contain hate speech – is difficult, if not impossible, to define. There exist comments which clearly are hate speech, and there exist comments which clearly aren't, but the exact differentiating line is a bit hazy. As machine learning students and researchers, we have no authority in actually laying groundwork definitions for hate speech ourselves. Thus, we opted for a datasets with a high inter-rater agreement rate to indicate that the labels are precise and consistent in some domain. Our many datasets employ unique parameters and instructions to annotators for defining hate speech. Since our project compares datasets directly, there is a bit of a non-sequitur here. We will discuss it in the Discussion section.

1. **Automated Hate Speech Detection and the Problem of Offensive Language**
    - Size of dataset: 24,802
    - Platform: Twitter
    - This dataset uses a hate speech lexicon containing words and phrases identified by internet users as hate speech, compiled by Hatebase.org. Using the Twitter API, the dataset searched for tweets containing terms from the lexicon, from which a random sample of 25k tweets was selected. Each tweet was then categorized and coded by three or more people. We will construct features (hate/no hate) from these tweets and use them to train our models.
2. **Hate Speech Dataset from a White Supremacy Forum**
    - Size of dataset: 9,916
    - Platform: Stormfront

- This dataset includes text that has been extracted from Stormfront, a white supremacist forum. A random set of forums/posts have been sampled from several smaller forums, which were then split into individual sentences. These sentences were then manually labelled as containing hate speech or not according to specific annotation guidelines.

3. **When Does a Compliment Become Sexist? Analysis and Classification of Ambivalent Sexism Using Twitter Data**
    - Size of dataset: 712
    - Platform: Twitter
    - This dataset includes data that was collected using the public Twitter Search API. The terms queried were common phrases and hashtags that are generally used when exhibiting benevolent sexism; this results in the dataset being sexist in nature to both men and women. Paternalism, gender differentiation and heterosexuality were the three main identifiers for sexism in the tweets

2.2 Model Breakdown

Here, we will discuss the general structure of each model we used to provide some context for our implementation and evaluation. Model selection plays a critical role in text classification, and hence was an important initial step before training and testing. Based on literature review, ease of adaptation, and relevance to the identified problem, the three selections below were the main models used throughout experimentation.

1. **Single-stream Convolutional Neural Network**
    - Convolutional neural networks use layers with convolving filters that are applied to local features, and have been recognized as being effective in semantic parsing, lexical modeling, and other NLP methods. To provide further details, CNNs consist of an input layer, hidden layers that perform the convolutions, and an output layer. These hidden layers can consist of normalization layers, fully connected layers, pooling layers, and other standard CNN infrastructure as desired for the task.
    - The single-stream CNN is simple in that it consists of one convolutional layer on top of word vectors that were obtained from an external unsupervised neural language model, after which max pooling, dropout, and softmax layers are all applied. The vectors were trained on publicly available words from Google News, and are maintained throughout while only the hyperparameters of the model were changed. One key factor in our selection of this model was its inclusion of these specific pre-trained vectors, as they are very effective feature extractors that can easily be applied for different types of classification. This elasticity of feature extraction made it a promising first model for our experimentation.

2. **Parallel Convolutional Neural Network**
    - Similar to our first model, we choose to adapt a CNN-based model, but here we include parallel convolutions and apply a slightly different architecture involving n-

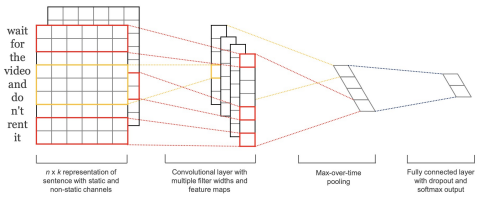

Figure 1: Model architecture of Single-stream CNN [5]

grams of words. What this signifies is that different kernel sizes will be applied to the same input text, after which each of the kernel's outputs will be reduced via max pooling, concatenated, and passed through a final linear layer. It is important to note that the convolutional layers are not stacked, with each layer being defined by a unique kernel size; this is representative of the n-gram. Such a parallel structure could potentially allow for an even more powerful feature extraction ability - a key factor that we wanted to take advantage of and explore.

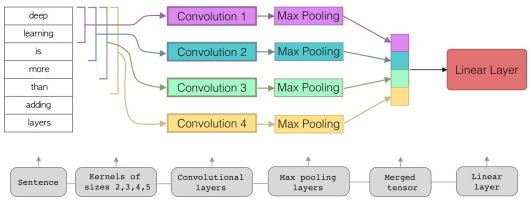

Figure 2: Model architecture of Parallel CNN [6]

3. **Distilled Bidirectional Encoder Representations from Transformers (DistilBERT)**

   - The general BERT model utilizes Transformers, which is an attention mechanism that is capable of learning contextual relationships between different words in a text. It essentially consists of an encoder and decoder, which reads the input and generates a prediction for the task, respectively. VERT allows for the development of a language model that is also bidirectional, allowing it to learn the context of a word from both the left and right sides. With this understanding, DistilBERT maintains the same general architecture as BERT, but now the token-type embeddings and pooler are removed while the number of layers are halved. This exhibits the "distillation" of the BERT model, meaning that it has been compressed into a more compact, concentrated version of the model. Additionally, this allows for the minimization of cross-entropy between the predicted distribution of the model and empirical distribution of training labels. All in all, DistilBERT provides an optimized model that maintains computational efficiency despite having more integrated attention mechanisms.

   - To convert the DistilBERT hidden states into a final softmax for hate speech classification, we tried two different setups of output linear layers. Our original idea involved making a single linear layer from the set of all sequence tokens' hidden states to a final softmax output; however, since this resulted in a very large number of weights to be fully connected, the resulting training was too slow to be feasible. We conjectured that, since hate speech is less dependent on context/holistic aspects of a sentence and more de-

pendent on the presence of certain hateful words or phrases, we could condense the hidden states into a single value for each sequence token, and then convert to the softmax output. In other words, the architecture we actually trained on uses two output linear layers in a row, one from (number of hidden states) $\rightarrow$ 1, and the other from (sequence length) $\rightarrow$ (number of classes), with a flattening layer in between to make the shapes correct. Since this restricts the architecture based on an assumption about the classification task, performance should theoretically be worse, but we conjectured that performance would be approximately very close.

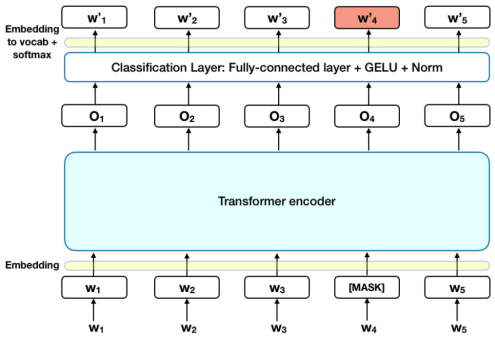

Figure 3: Model architecture of BERT (DistilBERT is an adaptation of this architecture) [7]

3 Related Work

Here, we discuss some previous work that has been completed with similar explorations into hate speech identification and classification. By understanding these past explorations, we hope to gain a better sense of direction with regards to the suite of experiments to run, in addition to obtaining better context of how hate speech can be better classified.

1. **Exploring Multi-Task Multi-Lingual Learning of Transformer Models for Hate Speech and Offensive Speech Identification in Social Media**[8]

    - This paper focused on three subtasks involving hate speech: multitask learning with separate task heads, back-translation, and multilingual training. For each of these tasks, various model architectures were studied, similar to how we are structuring our experiments. The models all went through individual fine-tuning processes, allowing the authors to evaluate different metrics, again, for each of the tasks. Conclusively, the authors were able to determine the promise of multilingual and multi-task models in developing different hate speech detection methodologies.

2. **Feature Explorations for Hate Speech Classification**[9]

    - In this paper, a hate speech classifier for Tweets is presented, with which different optimizations and evaluations are performed. Of the many experiments performed, character n-grams were identified as one of the most effective approaches of classification - something we hope to further explore. Specifically, 4 and 5-grams were best for feature extractions, and the feature extraction and pre-processing methods serve as a great example for ways we can structure and utilize our data for training and testing.

4 Results

4.1 Single-stream CNN

As suggested by (Madukwe et. al., 2020), we initially tried to use unbalanced training sets in our single-stream CNN formulation. Hate speech detection is unique from other sentiment analysis text classification tasks in that it suffers from a class imbalance issue. For most datasets, the hate class is less than 12% of the dataset, and in the case of Twitter, this is around 3%. For the White Supremacy hate speech dataset from Stormfront, this value was 11%. This creates a problem, as either we train on a very small (but balanced) dataset, or we train on a large, but imbalanced dataset. For hate speech, the question of looking at a dataset in its unbalanced form or to make it balanced is an open one, and remains unsolved (Madukwe).

Upon following the described guidelines for training on an imbalanced dataset, we found large overfitting and low accuracy on the downstream classification task. Our testing was done from the same imbalanced dataset – which makes sense since the motivation for using an imbalanced dataset such as this is to represent 'reality' in the model. See Figure 2 for some discussion in the imbalanced dataset task.

We then built a balanced dataset by simple random sampling, with a corresponding balanced test set. Although we had much less data to train with, we achieved an evaluation accuracy of **74.7%** on the test set. For clarity, we are doing a binary classification, so a random classifier would achieve an expected 50% accuracy. This model still was experiencing some overfitting (see Figure 3).

4.2 Parallel CNN

Initially, our parallel CNN model was set up with high-value hyperparameters and a relatively large amount of parallel layers. Our first training experiment on the MLMA dataset results in heavy overfitting; both the training and test accuracies remain at about 0.6 after the first epoch of training, but after 10 epochs, the training accuracy shoots up to 0.97-0.98 while the test accuracy falls between 0.45-0.5. With the drastic difference in the two accuracies as well as the near-perfect training accuracy, we suspect that our parallel CNN essentially memorized the data, implying that its architecture was too large/complex for the classification task on the MLMA dataset.

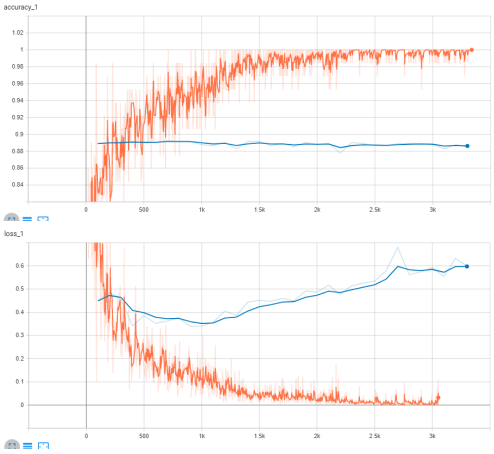

Figure 4: We have train/validation accuracies on top with the train/validation losses below. Our overall accuracy on our test set was **89.9%**, but when considering that only 11% of the data is hate speech, this is not so impressive. We could get 89% accuracy by guessing "not hate" every time. In the real world, misclassifying hate speech as not hate is much more costly than misclassifying some temperate speech as hate speech. It could make more sense to err on the side of caution, modifying our loss function to account for this. We discuss this more in Future Work.

For this reason, we pursued hyperparameter tuning for regularization by focusing more on hyperparameters that directly determined the

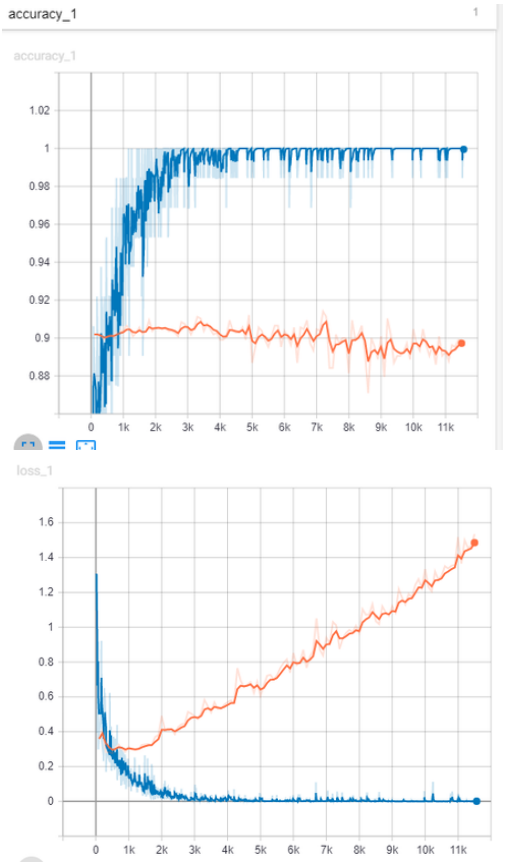

Figure 5: Blue is train and orange is validation. We see overfitting, with the validation loss increasing over time.

architecture and preprocessing, rather than the training-related hyperparameters (batch size, learning rate, etc.). Throughout these tuning experiments, we adjusted a single hyperparameter value and then retrained the parallel CNN, as training was very fast (1 min) on our notebook settings. Our first experiment involved adjusting the vocabulary size, used to tokenize the datasets, from 10000 to 2500 to 1000; we hypothesized that a very small vocab size was appropriate for hate speech classification because the category of hate speech that a document falls under is determined mostly by the presence of certain words such as slurs, swears, or other demeaning adjectives, of which there are relatively few in any language. Our decreases in vocab size only slightly moved the needle towards regularization; the final training accuracy decreased by 0.01-0.02 and the final test accuracy increased by 0.01-0.02, but overall our training performance could be characterized as severe overfitting.

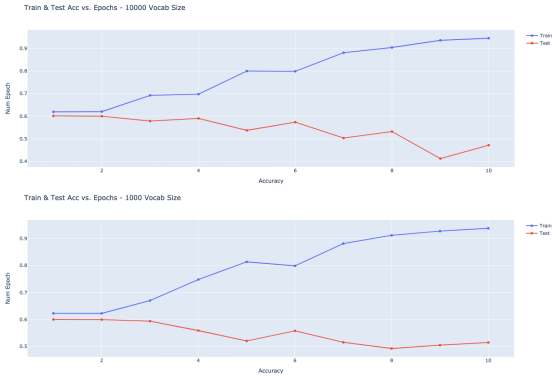

Figure 6: Our initial results for the parallel CNN's training loop, exhibiting heavy overfitting. In reducing the vocab size, our training results experience a positive but very slight improvement.

Another possible approach towards reducing overfitting is enlarging the dataset; although we had used all the data available in the MLMA dataset (∼5000 points), we developed similar preprocessing for a multilingual Twitter dataset with ∼48000 data points.

Our final and largest set of tuning experiments concerned the parallel CNN architecture itself, taking various steps to reduce its complexity. Each parallel layer had the same embedding size and output size but a different kernel size; we removed the parallel layers corresponding to kernel sizes 4 and 5, leaving only two parallel layers of kernel sizes 2 and 3 respectively. Separately, we reduced embedding sizes and output sizes, originally set to values of 64 and 32, respectively; we

continually reduced these by half each time, reaching as low as embedding size 8 and output size 4.

When we trained on all new hyperparameter values simultaneously, i.e. a less complex parallel CNN architecture and a larger dataset preprocessed more specifically towards the hate speech classification task, our resulting performance was much more regularized. The training and test accuracies began at 0.77 and 0.76 respectively, rose to 0.81 and 0.78 after six epochs, and reached 0.82 and 0.775 after ten epochs. While the dramatic simplification of the CNN architecture reduced its overall accuracy in classification, the model was now able to properly learn the classification task rather than memorize the training set. In addition, our switching to the much larger Twitter dataset helped with ensuring performance was more similar across training vs. test sets. As the figure below shows, the CNN still learns optimally at a relatively low number of epochs, around 5 or 6, since we began to see overfitting in our last few epochs.

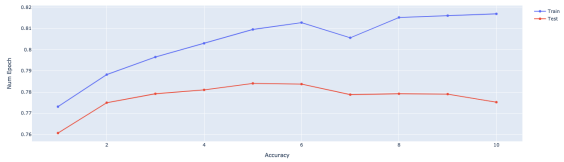

Figure 7: Our parallel CNN avoids overfitting after tuning our preprocessing parameters and culling more complex aspects of the architecture.

We also note that each tuning step above made a positive contribution towards reducing overfitting in our experiments. We omit most of these intermediate training experiments for the sake of space, but give one example below. When training on the Twitter dataset with two parallel layers in the CNN architecture but high embedding and output sizes of 32 and 16, respectively, the training accuracy improves from 0.75 to 0.95 across ten epochs while the test accuracy remains stuck around 0.75.

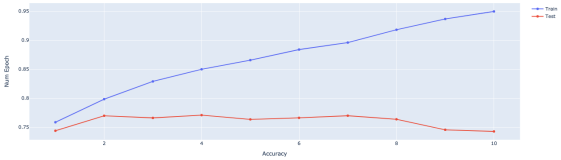

Figure 8: Our parallel CNN avoids overfitting after tuning our preprocessing parameters and culling more complex aspects of the architecture.

4.3 DistilBERT Classifier

We trained the DistilBERT classifier on our Twitter dataset for two epochs, using an AdamW optimizer and learning rate scheduler as described in the original DistilBERT paper. In addition, our training hyperparameters, including learning rate, batch size, and AdamW parameters, are taken from the question answering model training earlier in this class, as we viewed the architectures for the hate speech classifier vs. the question answering model as similar enough to reuse the same training hyperparameters. As a result, we don't perform any hyperparameter tuning in this experiment with the Twitter dataset, and are able to obtain results similar to state-of-the-art performance. The results of training are displayed below.

Figure 9: The change in batch loss across two epochs, which amounts to about 2200 training steps with our dataset and batch sizes.

Epoch	Train Acc	Test Acc
1	0.818	0.800
2	0.847	0.800

We see that the DistilBERT is able to learn most of the complexity of the classification task after about half of the first epoch, as the batch loss falls from its initial 1.4 in the first training step to the 0.4-0.6 range after about 500 training steps; the model stays in this range across the rest of the two epochs. We note that, throughout the second epoch, the batch loss reaches very low values of $\sim$0.3 in a few training steps, and is also more volatile, almost exactly following a see-saw pattern of alternating high and low losses.

One possible reason for this pattern may be related to overfitting; as the model learns finer details of the classification in the second pass through the data, it may alternate between memorizing data from the current group of batches to being punished for overfitting in the next group of batches. Our training and test accuracies after each epoch agree with the idea that the DistilBERT model risks overfitting in the second epoch. Our test accuracy doesn't improve in the second epoch, and when using additional precision in our numbers, it actually decreases slightly from 0.7999 to 0.7995. Meanwhile, the training accuracy is still able to improve by about 0.030. This is a divergence in the training and test set performance that suggests the DistilBERT model begins to overfit after the first epoch.

Overall, when comparing DistilBERT performance in this one experiment to that of the CNNs' best experiments, we see a $\sim$5% increase in test accuracy, a significant improvement. This fits in with our expectations about the nature of each model. For example, the

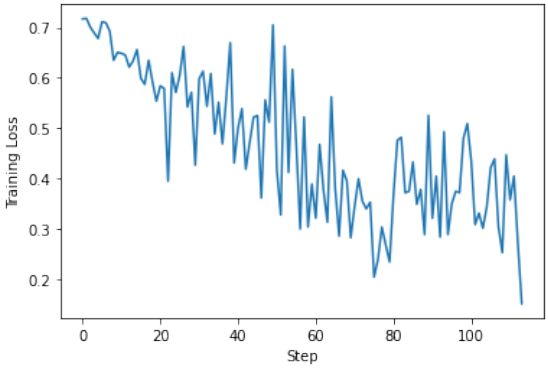

Figure 10: DistilBERT also demonstrates better performance on our white supremacy dataset, with a final test accuracy of 81.1% after the second epoch. This is an 8.6% increase in comparison to the single-stream CNN on the same data.

CNNs' ability to handle context is baked into its filters, which are themselves constrainted by kernel size. If a CNN's filter sizes correspond to n-grams of the same size, it cannot account for context sizes greater than its maximum filter size. By contrast, DistilBERT is able to use attention, so that its context used for a single word in a sentence can account for as much as the entire sentence, making it a better fit for these language-based classification tasks. In addition, CNN weights are trained from scratch, while DistilBERT starts off with pretrained weights for language that are then fine-tuned on our specific hate speech classification task.

5 Further Discussion

Throughout this section, we discuss various technical aspects of the hate speech classification task that may have had an impact on our earlier experiments.

As we discussed earlier, these datasets are fundamentally flawed in their unique, sometimes unclear definitions of hate speech; annotator bias and disagreement; and lack of large

datasets due to the expenses involved. Thus, we decided to experiment with adding noise to our data in an effort to measure how robust a model framework can be to poor data.

We define noise as the purposeful misclassification of training data (changing labels of a random set of our training set). We experimented with different percents of noise, corresponding to the amount of data that gets swapped. Our understanding of model resilience was a bit naive: we considered a model to be resilient if it had minimal decrease in percent accuracy in the presence of noise. This formulation is clearly flawed at the extremes: a model with consistently very low accuracy with and without the noise might be considered 'resilient,' although it would be more aptly characterized as simply, a bad model. Although the DistilBERT has the highest accuracy on the downstream task for all models overall, we find that the single-stream CNN actually demonstrated the most resilience to this noise, with relatively less decrease in accuracy across all amounts of noise (2%, 5%, 10%, and 20% tested).

The figure below depicts our experiment with the single-stream CNN using 2% noise. In comparison to our baseline, we have 97% of our original accuracy, with only a 9% increase in training loss. For the DistilBERT model, we saw 93% of the original accuracy, with a 13% increase in loss. The pattern continues for all noise values tested. Some hypotheses as to why this occurs may first consider the higher baseline efficacy of DistilBERT, since it effectively has "more to lose". Bad training samples might hurt the model more. Another intution could be that the mislabeled data confuses the pre-trained language model, with clearly offensive words being considered not hate speech for some portion of the dataset. The CNN on the other hand, uses unsupervised word embeddings it builds itself.

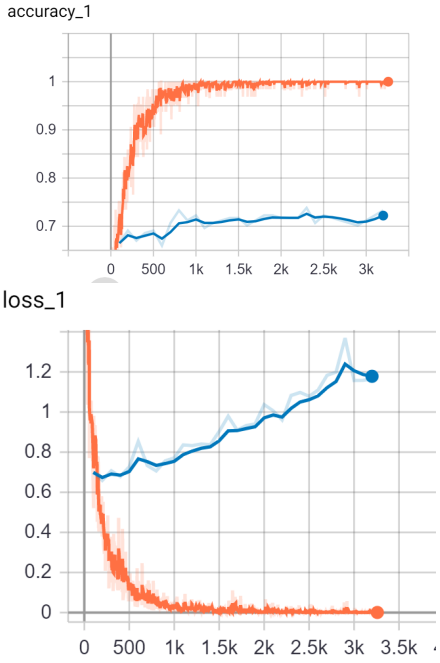

Figure 11: 2% noise with the single-stream CNN.

6 Future Work

There are a few technical approaches that can combat the problem of unbalanced datasets, discussed in our single-stream CNN results. One main way is to use different loss functions that are able to penalize some incorrect answers more than others. For example, since we used cross entropy loss in all of our experiments, we can iterate on our work by using a weighted cross entropy loss, in which the model's misclassification of hate speech as normal corresponds to a high loss, while misclassification of normal speech as hate speech corresponds to a much smaller loss. With larger numbers of classes, including different categories and intensities of hate speech, the tuning of weights in these loss functions can be a difficult task in itself, so we see it as a future direction outside the scope of our work.

Hate speech classification also gives rise to multi-label datasets, as we can describe hate

speech along several different categories such as intensity (normal, moderate, hate speech, etc.) and type (racism, misogyny, etc.). With such datasets, our models, which are designed for single-label classification, can perform separate classification tasks for each label. By comparison, we can design our models around simultaneous multiple classification tasks by having them output multiple softmaxes and having the training loop optimize on multiple cross entropy losses at the same time. We can potentially experiment with models other researchers have designed specifically for multi-label classification in the field of multi-task learning, and compare the performance of those models with our CNNs and DistilBERT.

Finally, we notice that for ambiguity in a hate speech dataset, some data points can be very clearly categorized while others are much more ambiguous and susceptible to annotator bias. As a result, if we have access to information about which points in a dataset annotators found more ambiguous, we can tailor our noise generation in the previous section to focus only on these ambiguous points, which would give us more clear information on the robustness of our models.

7 Conclusion

In this paper, we investigated different model constructions, datasets, and dataset formulations, ultimately performing a large number of experimental evaluations on the problem of hate speech detection. We showed that while a DistilBERT model retains the highest accuracy, using its pre-trained word representations, a more simple single-stream CNN has more resilience to noise in the dataset.

Hate speech is a real problem, as unchecked hate speech leads to hate crimes. We don't mean to trivialize this problem, and its harms, in any way by our quantitative approach to classification. Reading through the thousands of hate-positive samples was undoubtedly a taxing part of this project. Ultimately, while datasets define the boundaries of hate differently, the most vitriolic hate persist in all corners of the web. We hope to improve performance in detection and analysis of said speech.

References

[1] Sean MacAvaney, Hao-Ren Yao, Eugene Yang, Katina Russell, Nazli Goharian, Ophir Frieder. 2019. *Hate speech detection: Challenges and solutions.* PLoS ONE 14(8): e0221152. https://doi.org/10.1371/journal.pone.0221152

[2] Ziqi Zhang, Lei Luo. 2019. *Hate speech detection: A solved problem? The challenging case of long tail on Twitter.* Semantic Web 10(5): 925-945.

[3] Yoon Kim. 2014. *Convolutional Neural Networks for Sentence Classification.* Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP): 1746-1751. https://doi.org/10.3115/v1/D14-1181

[4] Pew Research Center 2017. Online Harassment 2017.

[5] arXiv:1408.5882v2 [cs.CL] 3 Sep 2014

[6] López, Fernando. "Text Classification with CNNs in PyTorch." Medium, Towards Data Science, 19 Sept. 2020, towardsdatascience.com/text-classification-with-cnns-in-pytorch-1113df31e79f.

[7] Hatzipanagos, Rachel. "How Online Hate Turns Into Real-life Violence" https://www.washingtonpost.com/nation/2018/11/30/how-online-hate-speech-is-fueling-real-life-violence/

[8] Horev, Rani. "BERT Explained: State of the Art Language Model for NLP." Medium, Towards Data Science, 17 Nov. 2018, towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

[9] Mishra, S., Prasad, S. Mishra, S. Exploring Multi-Task Multi-Lingual Learning of Transformer Models for Hate Speech and Offensive Speech Identification in Social Media. SN COMPUT. SCI. 2, 72 (2021). https://doi.org/10.1007/s42979-021-00455-5

[10] Scheffler, Tatjana, and Erik Haegert. "Feature Explorations for Hate Speech Classification." Proceedings of the GermEval 2018 Workshop, 21 Sept. 2018.