# Text Classification with BERT Variants

**Rui Miao**
Department of Computer Science
University of Toronto
Toronto, ON M5S 1A1
amyr.miao@mail.utoronto.ca

**Xiyuan Chen**
Department of Computer Science
University of Toronto
Toronto, ON M5B 1L3
garethcxy.chen@mail.utoronto.ca

**Shiyuan Zhou**
Department of Computer Science
University of Toronto
Toronto, ON M5V 3Z1
shiyuan.zhou@mail.utoronto.ca

## Abstract

This paper aims to examine and evaluate the performance and efficiency of two BERT variants: RoBERTa and DistilBERT base uncased, for classifying text by fine-tuning the models. We will perform the sensitivity analysis on the hyperparameters; and dive deep to scrutinize the pros and cons of each model under the Text REtrieval Conference (TREC) Question Classification dataset. Moreover, we will compare the performance of both BERT variants with Random Forest, a conventional machine learning model.

## 1 Introduction

Text classifier is a machine learning model that is designed to automatically classify text documents into one or more predefined categories or labels. Text classification is a common task in natural language processing (NLP), and it can be used in various applications, such as spam detection, sentiment analysis, content categorization, and language identification. Despite the fact that conventional machine learning methods like SVM and Decision Trees have shown promising results in many text classification tasks, they have limited capability for semantic learning and comprehension. In cases of high-dimensional classification or complex scenarios, a language model as the classifier may yield superior performance since these models can effectively capture the intricate nuances of human language and leverage contextual information, leading to more accurate and nuanced classification results.

## 2 Related Work

### 2.1 Random Forest

Random Forest [1] is an ensemble learning method expanded from the bagging method since it employs both bagging and randomness to create an uncorrelated forest of decision trees, which could reduce the risk of overfitting and improve the generalizability and accuracy(as shown in Figure 1). Furthermore, the random forest is a well-known strategy for addressing imbalanced data and its parallel architecture. Overall, Random Forest is a versatile and powerful algorithm for text classification tasks, capable of handling large datasets with high-dimensional features.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

### 2.2 BERT

BERT is a neural network approach to NLP[2]. It uses a multi-layer transformer architecture that processes input data hierarchically(as shown in Figure 2). It is pre-trained on a large corpus of text using unsupervised learning. It learns to generate useful representations of words through two pre-training tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). Once pre-trained, BERT can be fine-tuned for specific NLP tasks by adding a task-specific output layer and training on a smaller dataset.

## 3 Dataset

The dataset utilized in the study is the Text REtrieval Conference (TREC) Question Classification dataset, which contains 5500 labelled questions in the training set and 500 for the test set[3][4]. The dataset has six coarse class labels and 50 fine class labels, with the average length of each sentence being ten and a vocabulary size of 8700. The dataset includes six-class (TREC-6) and fifty-class (TREC-50) versions.

The dataset has three columns:

1. '`text`': Text of the question, i.e. "How did serfdom develop in and then leave Russia?"
2. '`coarse_label`': Coarse class label (six-class (TREC-6))
3. '`fine_label`': Fine class label (fifty-class (TREC-50))

In the study, we would like to predict the fine class label based on the text of the questions to examine each model's performance and investigate the model's sensitivity on the hyperparameters.

## 4 Model Architecture

### 4.1 Random Forest Classifier

Random Forest is a popular ensemble learning algorithm for classification and regression tasks. It works by constructing a collection of decision trees at training time and combining their predictions to obtain a more accurate and robust result. Applying Random Forest to text classification involves representing text data as feature vectors, building decision trees using the text features, and combining the predictions of the trees to obtain the final classification. The data was cleaned and vectorized, and transformed using CountVectorizer and tf-idf Transformer to prepare for training.

Then the random forest is built based on the random feature selection and bagging. Unlike traditional decision trees, Random Forest introduces randomness by selecting a random subset of features at each split. This prevents overfitting and encourages diversity among the trees in the forest, leading to better generalization performance. And the bagging technique randomly selects a subset of the training data with replacement and creates diverse subsets of the training data, which are then used to train individual decision trees. Once all the decision trees are built, the model will be applied to the test set. The predictions of individual trees are combined by taking a majority vote to obtain the final prediction of the Random Forest model.

### 4.2 RoBERTa

RoBERTa (Robustly Optimized BERT Approach) is a variant of the BERT (Bidirectional Encoder Representations from Transformers) language model, which is a type of transformer-based model. RoBERTa extends BERT by making modifications to its training objectives and traing on bigger batch sizes and longer sequences, resulting in improved performance on a wide range of natural language processing (NLP) tasks[6]. There are two key design changes in BERT's architecture[5].

**Removing the Next Sentence Prediction (NSP) losses** matches or slightly improves downstream task performance. NSP is a binary classification task that predict whether two sentences are consecutive in the document by NSP loss. Author trained the model with and without the NSP loss and concluded that removing it yields better performance[5].

**Dynamic Masking** slightly outperformed than static masking. In the original BERT, masking is only performed in data preprocessing stage, which result in static masking strategy during the training stage. Author re-implemented this strategy by duplicating and differently masking each training sample 10 times in 40 epochs. In contrast, dynamic masking strategy generate sequences' masking patterns whenever it is inputted into the model. By experiment, author concluded that dynamic masking has better performance in most of the dataset[5].

### 4.3 DistilBERT base model (uncased)

DistilBERT[7] is a compressed version of the BERT model that aims to address the issue of limited computational resources during both the training and inference stages of natural language processing tasks. Through the application of the technique of knowledge distillation, DistilBERT can achieve a compact version of the original model with faster inference and training times, while still retaining high levels of performance.

The implementation of DistilBERT involves a multi-objective training framework that combines the distillation loss with masked language modeling loss and cosine embedding loss. On top of that, DistilBERT is initialized from the original BERT by taking one layer out of two. It also removes the token-type embeddings and the pooler.

These modifications reduce the overall size of the original BERT model, making it more computationally efficient and faster to do training and inference while maintaining 97% of the language understanding capabilities.

## 5 Experiments and Discussion

### 5.1 Sensitivity Analysis

We conducted sensitivity experiments of RoBERTa and DistilBERT on TREC dataset to evaluate their dependency on hyperparameters, including training batch size, learning rate, number of hidden layers, and dropout probability. All experimental runs were executed with a fixed number of epSochs, specifically, epoch=5. We set our base model as the default setting by using training batch size=16, learning rate=0.00002($2e^{-5}$), number of hidden layers=6, and dropout probabilit=0.1. Remaining the other parameters unchanged, each value selection is compared by runtime, accuracy, F1 score, and training loss. The full testing results are presented in Appendix B, C, and D.

#### 5.1.1 RoBERTa

In the analysis of the training setup for RoBERTa, it is observed that using a smaller batch size results in a decrease in training loss by $0.5$ and an increase in accuracy by $5\%$ to $10\%$ per epoch(Figure 3). Additionally, a smaller training batch size leads to a higher F1 score, indicating better model robustness within the same number of training epochs. However, when a smaller learning rate of $2e-6$ is used, RoBERTa struggles to fit the model within 5 epochs, resulting in significantly higher loss and lower accuracy and F1 score compared to learning rates of $2e-5$ and $2e-4$(Figure 4). Although the training time increases by approximately 5 seconds, a learning rate of $2e-4$ demonstrates superior performance compared to other settings.

In terms of RoBERTa's configuration(Figure 5), it is observed that a higher dropout probability results in a poorly fitted model with high loss and lower accuracy and F1 score, possibly due to increased variance in some layers. Furthermore, increasing the dropout rate slightly increases the training time. Adjusting the number of hidden layers has a significant impact on all evaluation metrics (Figure 6). Training with more hidden layers results in higher training time, but better representation of the data. Specifically, when using 18 hidden layers, accuracy, F1 score, and training loss stabilize after 2 epochs, which can only be achieved by training over 4 epochs for models with 12 hidden layers.

Based on the comparison of all model settings(Figure 11), it is determined that the model with 12 hidden layers is the most competitive. Although it requires a longer running time, it achieves higher accuracy and F1 score compared to models with 6 hidden layers or other configurations. Despite the fact that the model with 18 hidden layers exhibits superior performance in terms of accuracy and F1

score, it is not chosen for implementation due to the considerable increase in training time compared to models with fewer hidden layers. Specifically, the model with 18 hidden layers requires 1.5 times the training time of the model with 12 hidden layers, and three times the training time of the models with only 6 hidden layers.

#### 5.1.2 DistilBERT

We investigate the impact of batch size on model performance(Figure 7), and the results suggest that variations in batch size have a negative effect on performance. Specifically, our experiments indicate that the base configuration with a batch size of 16 outperforms the second configuration with a batch size of 32 by approximately 5%, and is approximately 12% better than the configuration with a batch size of 64 in terms of test accuracy.

Increasing the number of hidden layers from 6 to 12 results in a significant improvement in test accuracy, with a performance boost of approximately 7% while at the cost of increased training time(Figure 10). However, this improvement appears to level off at 12, and further increases do not yield significant improvement.

Of the three learning rates tested, the learning rate of $2e^{-4}$ showed the best performance(Figure 8). Our results suggest that a larger learning rate may lead to better results, but we suspect that the model may not have fully converged yet due to the limited number of training epochs.

Additionally, our experiments indicate that the dropout rate has a negative impact on performance. We discover around a 2% drop in accuracy for every $0.1$ increase in the dropout rate(Figure 9).

Overall, the number of hidden layers contributes to most of the performance improvement. Among the nine different experimental configurations tested, the model with `num_hidden_layers=`$18$ achieved the best performance(Figure 12).

### 5.2 Performance and Efficiency Comparison

We present a comparative analysis of RoBERTa and DistilBERT under three different benchmarks: accuracy, FLOPs(floating point operations), and F1 score. For each benchmark, we select the optimal configuration of both models and report the results in tabular format as shown in Table 1, Table 2, and Table3. Additionally, we attach the result of the Random Forest at the end of each table for reference.

When we select the variants with lowest FLOPs, RoBERTa has higher accuracy and F1 score, showing better performance and model robustness(Table 1). Additionally, RoBERTa also uses less FLOPs, to achieve a 3%her F1 score than DistilBERT, even though it has about 1% lower accuracy(Table 2). Furthermore, comparing the model with highest accuracy from both of the variant, RoBERTa requires less FLOPs and achieve 2% higher F1 score(Table 3). Hence, RoBERTa is more efficient and surpassed than DistilBERT on TREC dataset while Random Forest has a 22% lower accuracy and 25% lower F1 score.

### 5.3 Limitation and Discussion

It is imperative to conduct further experiments on alternative benchmark datasets, considering the potential bias introduced by pre-training RoBERTa and DistilBERT on sentences that are akin to TREC texts, resulting in a high accuracy rate of over 90%. This could potentially skew the comparison with Random Forest, which is not a pre-trained model. Additionally, model with low learning rate requires more epochs to train which could yield better performance but it is also time-consuming.

## 6 Conclusion

Based on the findings from the sensitivity analysis and our experimental evaluation, it has been determined that RoBERTa, a variant of the BERT model, exhibits superior efficiency and outperforms all other algorithms that were compared in our study. This outcome not only substantiates RoBERTa's increased robustness and reliability in comparison to DistilBERT, but also provides evidence for the superiority of advanced language models over conventional models such as Random Forest.

## References


[1] L Breiman. Random forests. *Machine Learning*, 45:5–32, 10 2001.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[3] Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research*, 2001.

[4] Xin Li and Dan Roth. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*, 2002.

[5] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[6] pawangfg. Overview of roberta model, Jan 2023.

[7] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

# Appendix

## Appendix A

Figure 1: Random Forest Architecture

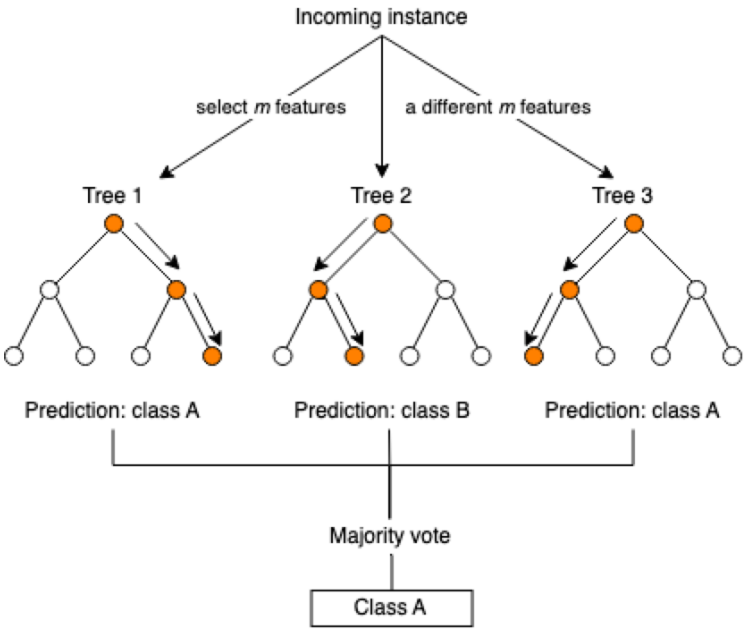
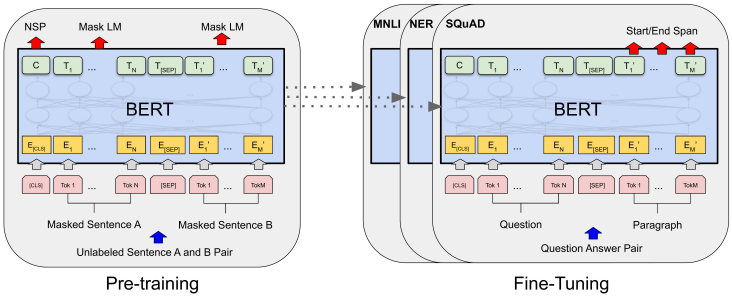
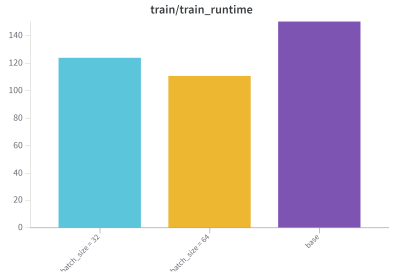



Figure 2: BERT Architecture

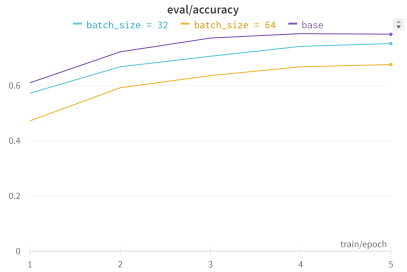


## Appendix B

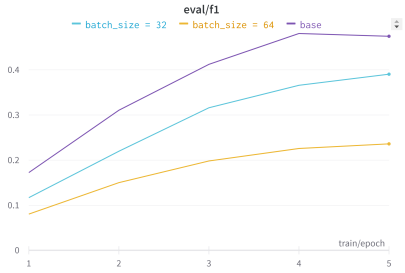
Figure 3: Controlled Experiments on Different Training Batch Size of RoBERTa Training Setting



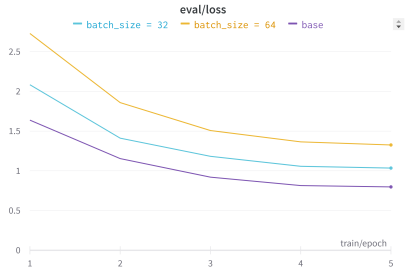

(a) Runtime




(b) Test Accuracy

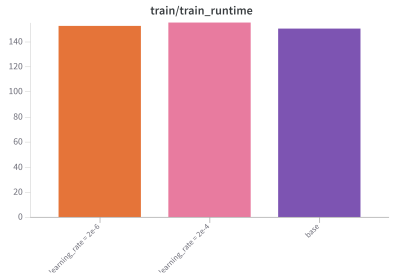



(c) F1 Score

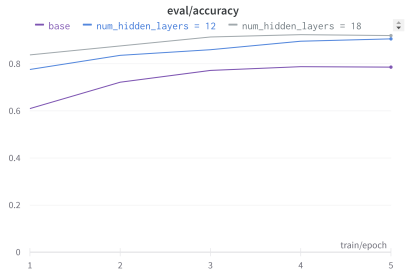



(d) Test Loss

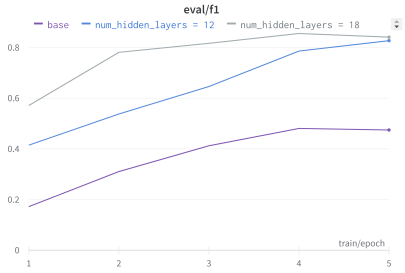
Figure 4: Controlled Experiments on Different Learning Rate of RoBERTa Training Setting



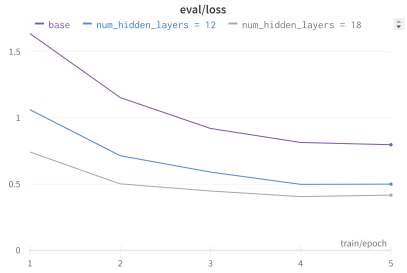

(a) Runtime




(b) Test Accuracy

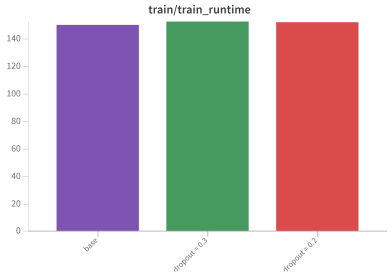



(c) F1 Score

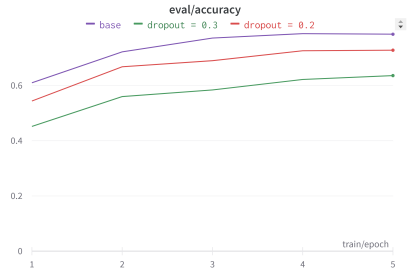



(d) Test Loss

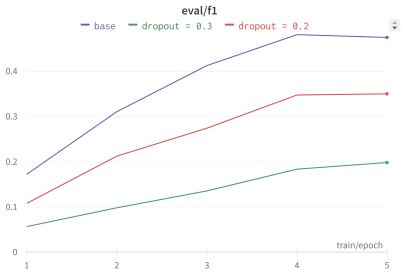
Figure 5: Controlled Experiments on Different Dropout Probability of RoBERTa



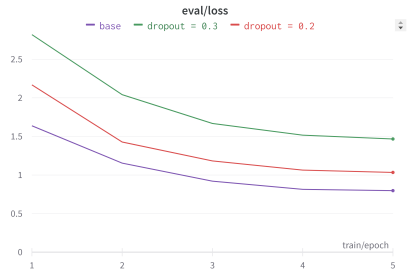

(a) Runtime




(b) Test Accuracy

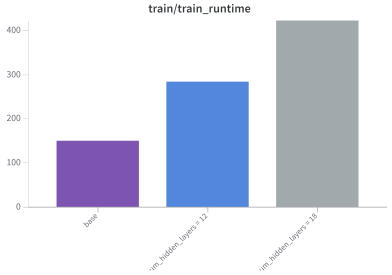



(c) F1 Score

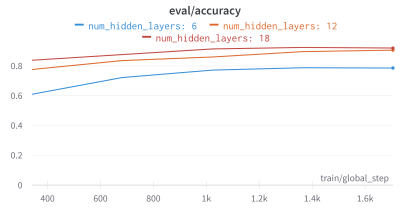



(d) Test Loss

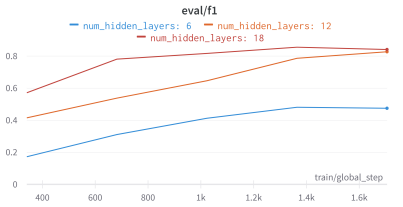
Figure 6: Controlled Experiments on Different Number of Hidden Layers of RoBERTa



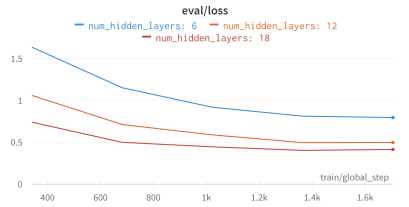

(a) Runtime




(b) Test Accuracy



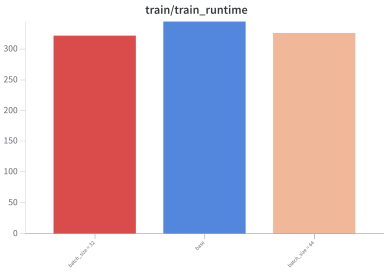

(c) F1 Score



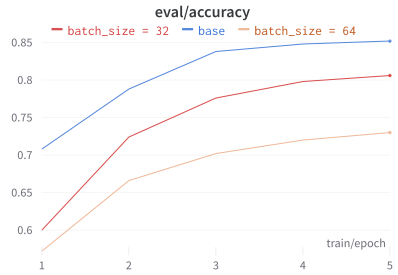

(d) Test Loss

## Appendix C

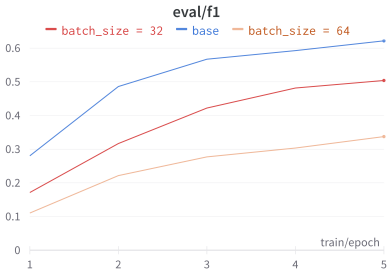
Figure 7: Controlled Experiments on Different Training Batch Size of DistilBERT Training Setting



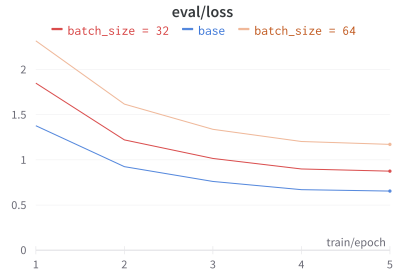

(a) Runtime




(b) Test Accuracy

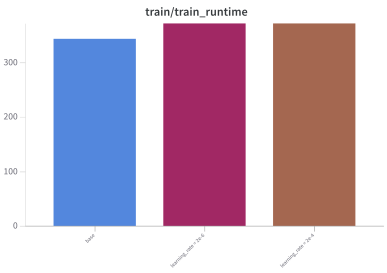



(c) F1 Score

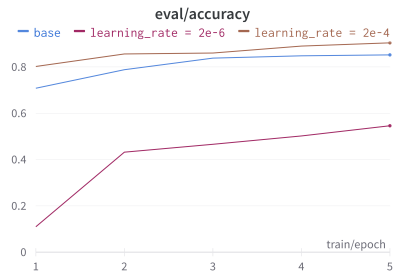



(d) Test Loss

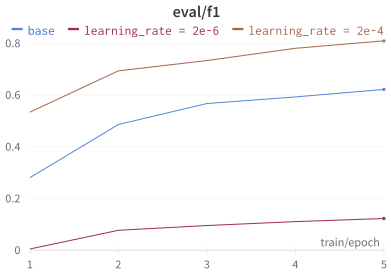
Figure 8: Controlled Experiments on Different Learning Rate of DistilBERT Training Setting



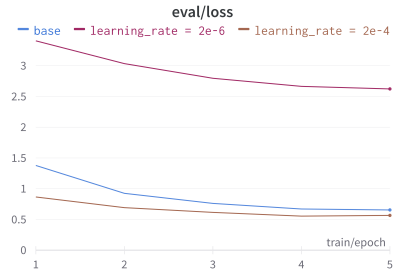

(a) Runtime




(b) Test Accuracy

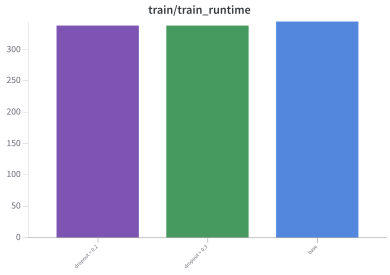



(c) F1 Score

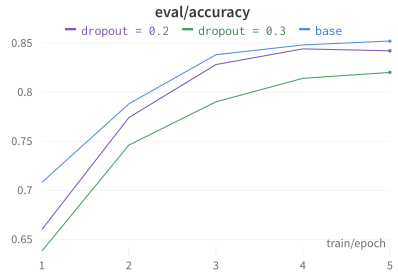



(d) Test Loss

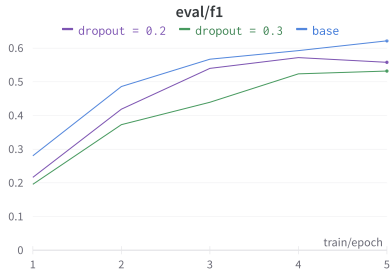
Figure 9: Controlled Experiments on Different Dropout Probability of DistilBERT



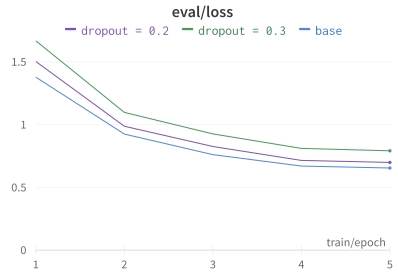

(a) Runtime




(b) Test Accuracy

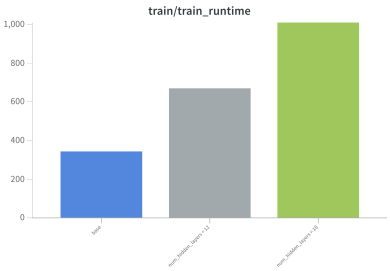



(c) F1 Score

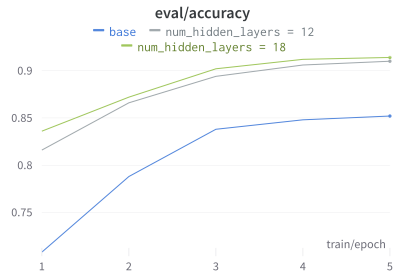



(d) Test Loss

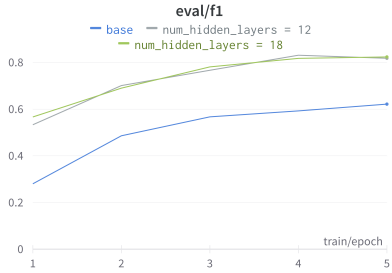
Figure 10: Controlled Experiments on Different Number of Hidden Layers of DistilBERT



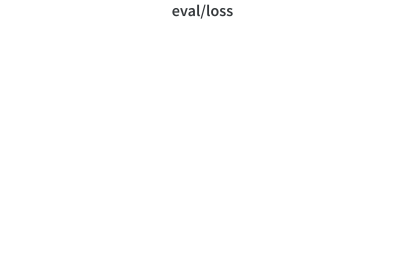

(a) Runtime




(b) Test Accuracy



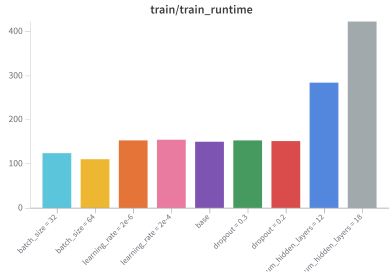

(c) F1 Score



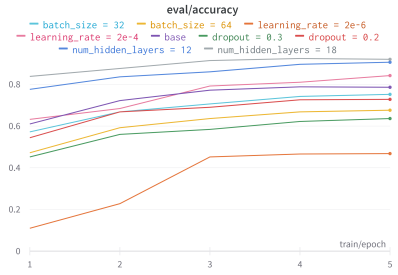

(d) Test Loss

## Appendix D

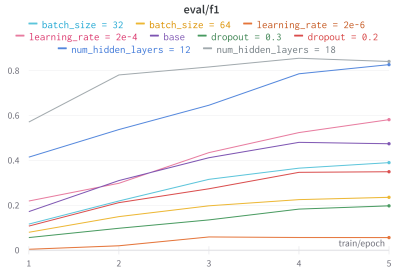
Figure 11: Experiments on RoBERTa



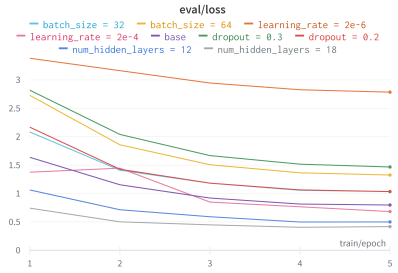

(a) Runtime




(b) Test Accuracy

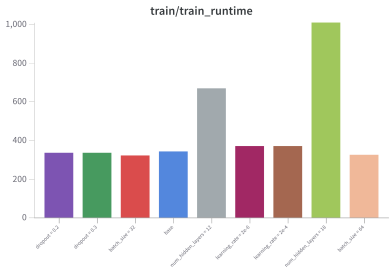



(c) F1 Score

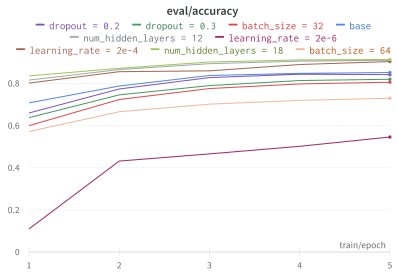



(d) Test Loss

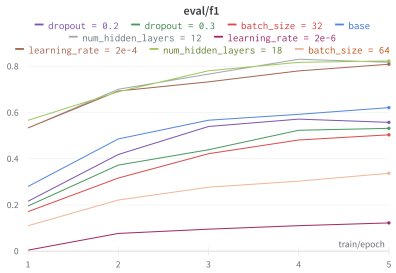
Figure 12: Experiments on DistilBERT



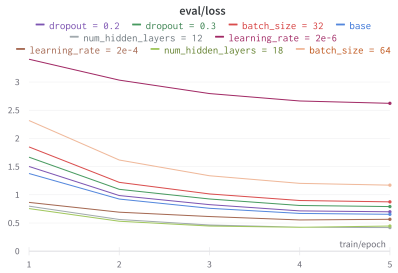

(a) Runtime




(b) Test Accuracy




(c) F1 Score




(d) Test Loss

**Appendix E**

Table 1: Model comparison regarding FLOPs.

| Model | Acc | **FLOPs** | F1 | Hyperparameters | | | |
|---|---|---|---|---|---|---|---|
| | | | | lr | n_layer | batch_size | dropout |
| RoBERTa | 0.9080 | 3.6142e15 | 0.8500 | 2e-4 | 6 | 16 | 0.1 |
| DistilBERT | 0.9040 | 3.6142e15 | 0.8094 | 2e-4 | 6 | 16 | 0.1 |
| Random Forest | 0.6760 | - | 0.5526 | - | - | - | - |

Table 2: Model comparison regarding F1 score.

| Model | Acc | FLOPs | **F1** | Hyperparameters | | | |
|---|---|---|---|---|---|---|---|
| | | | | lr | n_layer | batch_size | dropout |
| RoBERTa | 0.9079 | 3.6142e15 | 0.8500 | 2e-4 | 6 | 16 | 0.1 |
| DistilBERT | 0.9140 | 1.0736e16 | 0.8180 | 2e-5 | 18 | 16 | 0.1 |
| Random Forest | 0.6760 | - | 0.5526 | - | - | - | - |

Table 3: Model comparison regarding accuracy.

| Model | **Acc** | FLOPs | F1 | Hyperparameters | | | |
|---|---|---|---|---|---|---|---|
| | | | | lr | n_layer | batch_size | dropout |
| RoBERTa | 0.9200 | 8.5978e14 | 0.8408 | 2e-5 | 18 | 16 | 0.1 |
| DistilBERT | 0.9140 | 1.0736e16 | 0.8180 | 2e-5 | 18 | 16 | 0.1 |
| Random Forest | 0.6760 | - | 0.5526 | - | - | - | - |