# Fine - Tuning CovidBert and Bertweet

Anat Korol-Gordon, Itai Peleg

August 21, 2025

**Abstract**

This paper investigates transformer-based models for COVID-19 tweet sentiment classification, comparing domain-specialized COVID-Twitter-BERT against general-purpose BERTweet on a 40,000-tweet dataset with five sentiment categories. We addressed severe overfitting through targeted data augmentation, generating 3,384 synthetic tweets using GPT-5 mini for underrepresented extreme sentiment classes, and implemented both Hugging Face Trainer and native PyTorch training frameworks for comprehensive evaluation. COVID-Twitter-BERT achieved superior performance with 72% accuracy, 72% F1-score, and 91% AUC, outperforming BERTweet across all metrics. To enable practical deployment, we evaluated three compression techniques: quantization (90% size reduction, severe accuracy loss to 30%), structured pruning (minimal benefits), and knowledge distillation (80% size reduction with 4× speedup). Remarkably, knowledge distillation not only compressed models effectively but improved classification accuracy to 84%, suggesting that architectural constraints enhance generalization by reducing overfitting. These findings demonstrate the effectiveness of domain-specific pre-training for specialized tasks and reveal knowledge distillation as a dual-purpose technique for both model compression and performance enhancement in transformer-based sentiment analysis.
Github Repo: `https://github.com/itaipeleg1/corona_virus_NLP`

## 1 Introduction

The COVID-19 pandemic generated unprecedented volumes of social media discourse, making sentiment analysis of pandemic-related tweets an interesting field for NLP research. Transformer-based models have demonstrated superior performance in natural language processing tasks, specifically BERT backbones fine tuned for sentiment analysis. Here, we set to discover how different backbones perform, investigate trade-offs between model size, training set size and fine tuning for specific field. We also set out to explore how the computational demands can be addressed, allowing for practical deployment. This paper investigates the application of distinct BERT variants—Covid-tweeter-BERT and BERTweet —for sentiment classification of COVID-19 tweets, with particular focus on model compression techniques to address computational constraints.
Our contributions include: (1) comparative analysis of specialized COVID-19 language models versus non specific models, (2) targeted data augmentation using synthetic tweet generation to address class imbalance, and (3) comprehensive evaluation of compression techniques including quantization, pruning, and knowledge distillation. Our findings reveal that knowledge distillation not only reduces model size and improves inference speed but unexpectedly enhances classification performance, suggesting potential benefits of architectural constraints for generalization.

# 2 Related Work

Transformer-based models have revolutionized sentiment analysis, with BERT and its variants achieving state-of-the-art performance across multiple benchmarks. Domain-specific adaptations, such as BioBERT [4] for biomedical text and FinBERT [2] for financial documents, have demonstrated the value of specialized pre-training for targeted applications. In the social media domain, BERTweet [7] represents notable advances in tweet-specific language understanding and COVID-Twitter-BERT is a less known fine tune for Covid-19 specific tweets [6] .

Overfitting remains a critical challenge when fine-tuning large transformer models on limited datasets. Data augmentation strategies , including synthetic text generation using large language models, have shown promise for addressing these challenges in transformer-based classification systems [10] [5]. We set out to put this to the test in COVID-19 domain tweets.

Knowledge distillation, originally proposed by Hinton et al. [1], has proven effective for both model compression and implicit regularization. Sanh et al. [9] showed that DistilBERT maintained 97% of BERT's performance while being 60% smaller, with the distillation process often improving generalization compared to direct fine-tuning. Recent studies consistently report that distilled models outperform their teacher models on test sets, suggesting that architectural constraints force learning of more generalizable features—particularly relevant for sentiment analysis tasks

# 3 EDA - Exploratory Data Analysis

## 3.1 The Dataset

Our dataset, the "COVID-19 NLP Text Classification" dataset from Kaggle [3], comprises over 40,000 English tweets related to the COVID-19 pandemic. Each entry includes the tweet text, location, date, and one of five sentiment labels: extremely positive, positive, neutral, negative, and extremely negative. Labels were manually annotated to facilitate sentiment classification. All tweets were collected during March and April 2020, capturing the initial outbreak period of COVID-19 worldwide.

## 3.2 Exploration of the Data

Our EDA process involved iterative analysis alongside model training. First, we set out to explore our dataset and found out it had 41,157 unique tweets from distinct usernames. Sentiment distribution was imbalanced, with fewer samples in extreme sentiment categories.

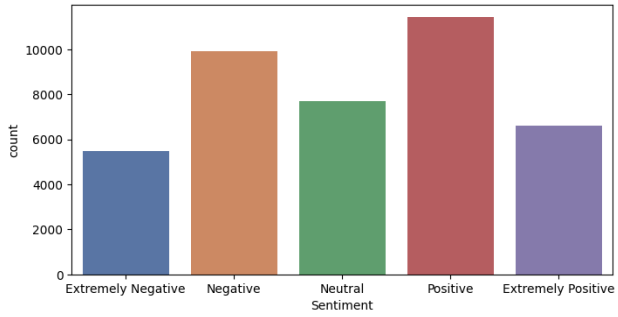



Figure 1: Sentiment labels distribution

We investigated potential patterns in sentiment distribution across various dimensions but found no significant variations: sentiments were evenly distributed across months, countries, and user types (we

identified premium users who exceeded the 280-character limit). Similar distribution patterns appeared across popular hashtags and mentions. Therefore, we focused on tweet content itself.

We applied preprocessing to remove URLs while initially retaining mentions and hashtags, hypothesizing they might convey meaningful information for the models. Word cloud visualization revealed COVID-related concepts such as "coronavirus," "supermarket," and "food," reflecting the food scarcity concerns prevalent during that period. Sentiment-specific word clouds showed subtle differences: "panic" appeared frequently in extremely negative tweets, while "hand sanitizer" and "help" were more common in positive tweets. These patterns suggested potential features that fine-tuned models could leverage.


Figure 2: Wordclouds for different sentiment categories.

We split our data into train (0.8) and evaluation. We created a CovidTweetDataset class for data to be processed in a manner that fits Bert and Roberta based models. Identical pipeline was introduced to the test set.

# 4 Fine Tuning

## 4.1 Model selection

After exploring the HuggingFace model library, we selected two complementary models that balance model size, training scale with task relevance:

1. BERTweet (vinai/bertweet-base): A popular model fine tuned on RoBerta, specifically designed for English tweets, fine tuned on 850M tweets with substantial community adoption ( 30,000 monthly downloads). This served as our general-purpose Twitter sentiment classifier.

2. CovidBERT (digitalepidemiologylab/covid-twitter-bert): A domain-specialized model. Fine tuned on BERT architecture, this model was bigger in size, but fine tuned on a smaller dataset - 22.5M COVID-related tweets. Despite its reduced training scale, we hypothesized that its domain specificity would compensate through better contextual understanding of pandemic-related discourse. While CovidBERT was trained on COVID-related tweets from the same time period as our dataset, data leakage is not a concern as CovidBERT underwent training for language understanding, not sentiment classification.

These model choices allowed us to compare broad tweet understanding (BERTweet) against domain expertise (CovidBERT) for COVID-19 sentiment classification.

## 4.2 Training Framework Comparison: Hugging Face Trainer vs. PyTorch Implementation

Both approaches utilized the same hyperparameter optimization technique using Optuna's objective function, with the same search spaces. learning rate (5e-6 to 5e-4), weight decay (1e-4 to 3e-2), batch

sizes (32, 64, 128), patience values (5-7), and trainable layers (1-2). The core training logic remained consistent across both implementations, including CrossEntropyLoss, AdamW optimizer (an improved variant of Adam optimizer), early stopping mechanisms, and identical evaluation metrics (accuracy, F1-score, precision).

However, there are significant differences in implementation complexity and control. The PyTorch implementation provided explicit control over the training loop, allowing manual batch processing, gradient computation, and metric calculation at each epoch, as well as custom implementation of early stopping logic. In contrast, the HuggingFace Trainer abstracted these complexities through high-level APIs, automatically handling the training loop. This allowed us to implement sophisticated configurations to the training process such as evaluation scheduling, learning rate warm-up (warmup_ratio=0.1), gradient clipping (max_grad_norm=1.0), model checkpointing and more via TrainingArguments configuration.

## 4.3 Overfitting Mitigation and Training Refinement

Our initial fine-tuning approach employed a vanilla methodology with broad hyperparameter search ranges, including learning rate (1e-7 to 1e-3), weight decay (1e-6 to 1e-3), and trainable layers (1-4). This exploratory approach aimed to identify optimal parameter spaces. However, both BERTweet and CovidBERT exhibited severe overfitting, with CovidBERT achieving 99.9% training accuracy while validation accuracy remained at 70%, indicating poor generalization capabilities.

## 4.4 Ultimate training

To address overfitting, we conducted a systematic analysis, beginning with confusion matrix on the evaluationt dataset with the best-performing CovidBERT PyTorch model (Figure 3). The analysis revealed a significant misclassifications between extreme and non extreme sentiment categories. This assured us that our models have a learning curve (they almost never confused negative with extreme positive sentiment), but the imbalance suggested that the model's poor generalization stemmed partly from insufficient representation of these minority classes in the training data.

### 4.4.1 Data cleaning and Augmentation

Following the vanilla training round, we returned to EDA to implement more substantial modifications. We removed hashtags and mentions, hypothesizing that these patterns might lead to overfitting rather than meaningful learning. Subsequently, we implemented a targeted data augmentation strategy using GPT-5 mini through OpenAI API [8]. Rather than generating arbitrary tweets, we provided the model with structured system instructions to create tweets with similar semantic meaning and syntactic structure to the original dataset. For example: ""Write ONE Twitter/X-style tweet: 8–22 words, ¡=280 chars, very positive tone, as if during the COVID-19 pandemic. No hashtags, no @mentions, no links. Do NOT copy the reference". Each request was accompanied with a random reference from the corresponding sentiment class to maintain authenticity and domain relevance. This process yielded 2,112 extremely negative tweets and 1,272 extremely positive tweets, which were exclusively added to the training set while preserving the original validation set integrity. When looking closely at random samples from the original dataset, we noticed some very poor labeling, for example: "this morning i tested positive for covid 19. i feel ok,... i will keep you updated on how im doing ???? no panic" as extremely negative sentiment, which could potentially explain suboptimal results of fine tuning. In comparison, our synthetically generated data seemed more reliable, and being an addition of around 12% to original data, it seemed like a valuable contribution to the fine tuning process.

We then refined our hyperparameter search strategy to better prevent overfitting, implementing higher weight decay values, adjusting learning rate ranges, and constraining trainable layers to 1-2 to reduce

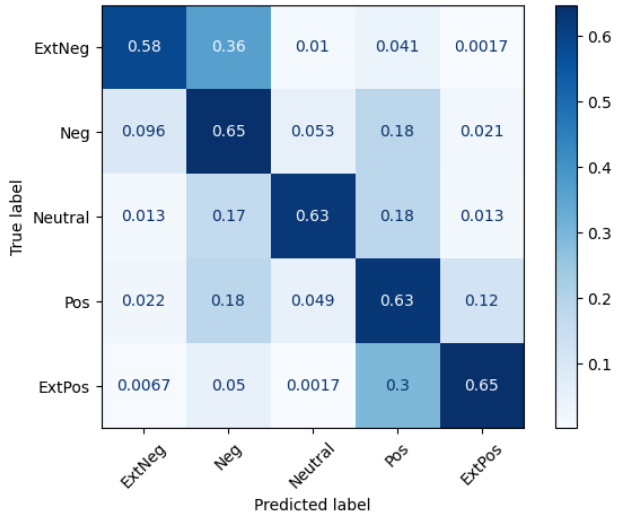



Figure 3: Confusion matrix on evaluation dataset after vanilla fine tuning

model complexity and improve regularization.

### 4.5 Results

The refined training methodology yielded better results, but it did mitigate overfitting and achieved strong test performance. CovidBERT outperformed BERTweet across all metrics, with the HuggingFace implementation of CovidBERT achieving the highest AUC of 0.911 and reaching 0.72 in accuracy and F1. BERTweet achieved 0.620-0.663 accuracy and 0.86-0.89 AUC. Neither PyTorch not HuggingFace Trainer showd consistent advantages.

| Model | Acc | F1 | AUC |
|---|---|---|---|
| Bertweet HF | 0.62 | 0.62 | 0.86 |
| Bertweet pytorch | 0.66 | 0.66 | 0.89 |
| CovidBert HF | 0.72 | 0.72 | 0.91 |
| CovidBert pytorch | 0.69 | 0.67 | 0.9 |

Table 1: Results

## 5 Compression

### 5.1 Compression Techniques

After evaluating our original fine tuned models, we investigated their resilience to compression to understand the trade-offs between model size, inference speed, and performance. We implemented three compression techniques: dynamic quantization of linear layers, structured pruning of attention heads, and knowledge distillation using two student models—DistilRoBERTa for the RoBERTa-based BERTweet and DistilBERT for the BERT-based Covid-Bert.

### 5.2 Compression pipeline

Our compression pipeline was implemented iteratively with performance-based modifications. For quantization, we applied post-training dynamic quantization to linear layers, reducing precision from float32

to 8-bit integers. After observing poor initial results, we excluded the classifier layer from quantization to improve performance. For pruning, we explored both global unstructured pruning of linear layers and structured pruning of attention heads, testing various percentages of pruned neurons and attention heads respectively. Knowledge distillation involved carefully selected student models with similar architectures but significantly reduced sizes: DistilRoBERTa with 313 MB compared with Our 514 MB fine tuned BerTweet, and DistilBERT with only 255 MB compared with CovidBERT ( 1278 MB). Knowledge distillation employed a combined loss function incorporating both the standard cross-entropy loss on ground-truth labels and a distillation loss based on the soft probability distributions from the teacher model, allowing the model to learn the nuanced decision boundaries captured by the teacher model. Despite achieving reasonable results after one epoch, we extended training to five epochs for optimal performance. While quantization and pruning executed rapidly, knowledge distillation required approximately 30 minutes on GPU.

### 5.3 Compression results

Compression techniques yielded markedly different outcomes with a clear performance leader. Quantization achieved 90% size reduction and 66% speed improvement but caused severe accuracy degradation (Acc = 0.30). Pruning techniques maintained reasonable accuracy (0.56-0.65) but provided minimal size and speed benefits. Aggressive optimization for size and speed resulted in accuracy collapse to 0.25 with no optimal balance achieved. Additional fine-tuning might have recovered accuracy losses. In contrast, knowledge distillation achieved astounding results, with 80% size reduction and 4x inference speedup while actually improving accuracy to 0.836. This counter-intuitive improvement suggests that learning from softer probabilties is indeed enriching for the model, and that the smaller architecture which by default caused stronger regularization, reduced overfitting, thereby enhancing generalization capabilities.

Table 2: Evaluation results for Covidbert (pytorch) and compressed models

| **model** | **acc** | **F1** | **auc** | **(Size (MB** | **Inference time (ms) cpu** | **Inference time (ms) gpu** |
|---|---|---|---|---|---|---|
| original | 0.72 | 0.72 | 0.91 | 1278.49 | 5727.56 | 31.60 |
| quantized | 0.31 | 0.23 | 0.70 | 121.64 | 3764.08 | NaN |
| pruned | 0.67 | 0.66 | 0.89 | 1182.42 | 5360.52 | 28.63 |
| distilled | 0.84 | 0.84 | 0.97 | 255.43 | 1454.20 | 4.46 |

## 6 Conclusions

Our project showed how fine tuning, compressing, and correct handling of data can improve model results in sentiment analysis tasks. We explored different backbones that differ in size and specialization for task, and also different fine tuning and compression techniques. The comparison between the two fine tuning techniques demonstrated the effectiveness of transformer-based models for COVID-19 sentiment classification, with CovidBERT achieving superior performance over BERTweet, probably due to its domain-specific original capabilities. Our comparison of HuggingFace Trainer versus PyTorch implementations revealed trade-offs between automation and control, with both frameworks achieving competitive results. The targeted data augmentation strategy affected and mitigated overfitting issues encountered in initial training rounds. The successful knowledge distillation, which improved our models not only in speed and size, but also in accuracy, surprised us and demonstrated the powerful tool of knowledge distillation for models, that is rising in popularity in the ML world. We shall point potential problems with

the data labeling, that prevented even higher performance. This, and more optimization of knowledge distillation hyper parameters, can be addressed in future works.

# References

# References


[1] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[2] Allen H Huang, Hui Wang, and Yi Yang. Finbert: A large language model for extracting information from financial text. *Contemporary Accounting Research*, 40(2):806–841, 2023.

[3] Kaggle. Covid-19 nlp text classification, 2020.

[4] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

[5] Junjie Liu, Yuanhe Tian, and Yan Song. Balanced training data augmentation for aspect-based sentiment analysis, 2025.

[6] Martin Müller, Marcel Salathé, and Per E Kummervold. Covid-twitter-bert: A natural language processing model to analyse covid-19 content on twitter. *arXiv preprint arXiv:2005.07503*, 2020.

[7] Dat Quoc Nguyen, Thanh Vu, and Anh Tuan Nguyen. BERTweet: A pre-trained language model for English tweets. In Qun Liu and David Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 9–14, Online, October 2020. Association for Computational Linguistics.

[8] OpenAI. Introducing gpt-5. `https://openai.com/index/introducing-gpt-5/`, 2025.

[9] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[10] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, 2019.


# Appendix

### 6.0.1 Full reports:

Table 3: Bertweet (Hugging Face Trainer) and compression

| **model** | **acc** | **F1** | **auc** | **(Size (MB** | **Inference time (ms) cpu** | **Inference time (ms) gpu** |
|---|---|---|---|---|---|---|
| original | 0.62 | 0.87 | 0.63 | 0.62 | 514.62 | 2547.18 |
| quantized | 0.32 | 0.72 | 0.24 | 0.23 | 188.05 | 2136.98 |
| pruned | 0.59 | 0.85 | 0.61 | 0.59 | 487.59 | 2531.95 |
| distilled | 0.79 | 0.94 | 0.80 | 0.79 | 313.28 | 1327.30 |

Table 4: Bertweet (pytorch) and compressions

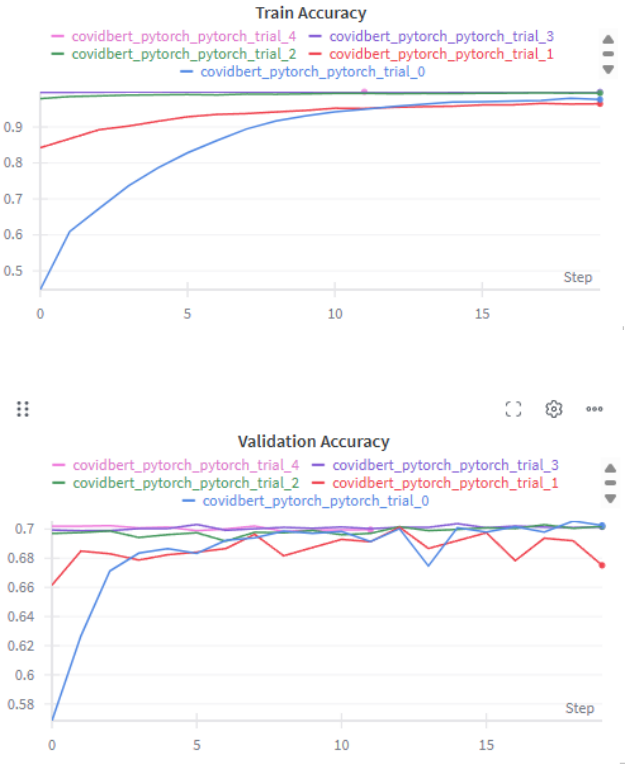
| model | acc | F1 | auc | (Size (MB | Inference time (ms) cpu | Inference time (ms) gpu |
|---|---|---|---|---|---|---|
| original | 0.66 | 0.89 | 0.67 | 0.66 | 514.62 | 2479.02 |
| quantized | 0.17 | 0.63 | 0.07 | 0.06 | 188.05 | 1703.13 |
| pruned | 0.62 | 0.87 | 0.62 | 0.62 | 487.59 | 2860.81 |
| distilled | 0.76 | 0.93 | 0.77 | 0.76 | 313.28 | 1519.21 |

Table 5: Covidbert (hugging face) and compressions

| model | acc | F1 | auc | (Size (MB | Inference time (ms) cpu | Inference time (ms) gpu |
|---|---|---|---|---|---|---|
| original | 0.72 | 0.91 | 0.73 | 0.72 | 1278.49 | 5727.56 |
| quantized | 0.31 | 0.70 | 0.21 | 0.23 | 121.64 | 3764.08 |
| pruned | 0.67 | 0.89 | 0.66 | 0.66 | 1182.42 | 5360.52 |
| distilled | 0.84 | 0.97 | 0.84 | 0.84 | 255.43 | 1454.20 |

#### 6.0.2 Wandb selected experiments:

vanilla training -




with augmentation -


Figure 4: Attention head pruning - accuracy drops with more pruning

### 6.0.3 EDA graphs:

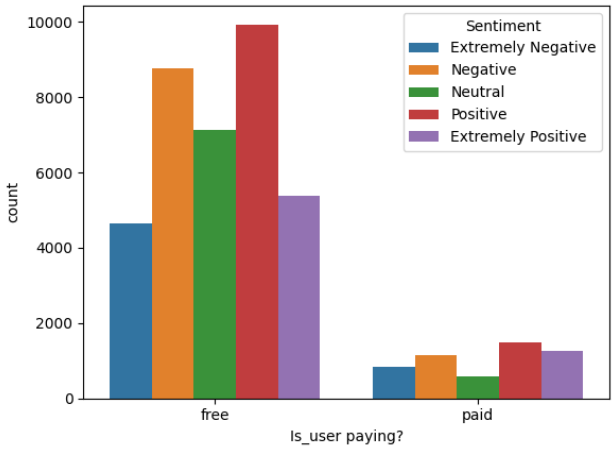



Figure 5: Distribution of classes: Free and Paid users

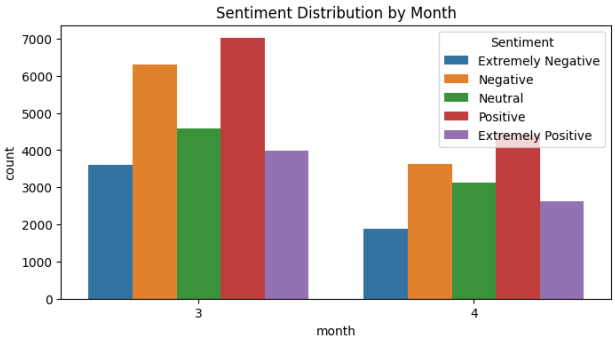



Figure 6: Distribution of classes: date