ECE364 Project Report

Bowen Zhang

Yi Su

Tianyue Jia

Overview

In this project, we build a binary classifier for identifying informative English tweets related to COVID-19[1]. The core of our approach is ALBERT, a lightweight yet expressive transformer model tailored for short text classification. Compared to other compact models, ALBERT showed superior baseline accuracy. Through progressive fine-tuning and several regularization techniques, we achieved a validation accuracy of 89.1% with strong generalization.

1. Model Architecture and Design Choices

We adopted the pretrained ALBERT model and appended a classification head consisting of a dropout layer (rate = 0.3) and a fully connected linear layer for binary output. To accelerate convergence, we assigned a higher learning rate to the classification head than to the frozen transformer backbone during early training. ALBERT has approximately 12.5M parameters, and with the classification head included, it still meets the project requirement of staying under 15M parameters.

2. Training Procedure

- **Base Training Phase:** The initial training was conducted with only the classifier head trainable, while all transformer layers were frozen. We used 7001 given training samples, optimized with **AdamW**, and monitored validation accuracy to trigger early stopping. This stage focused on fast adaptation with minimal disturbance to pretrained features.
- **Progressive unfreezing:** Once the classifier stabilized, we applied **progressive unfreezing**, gradually enabling the training of transformer encoder layers from top to bottom. This technique, implemented via a custom ProgressiveUnfreezer class, allowed the model to adapt deeper semantic representations while avoiding overfitting.
- **Dynamic Learning Rate Scheduling:** To further reduce the risk of overfitting and improve generalization, we dynamically adjust the learning rate when validation accuracy plateaued. This helped escape local minima and supported smoother convergence.

3. Data Processing and Augmentation

- **Data Cleaning:** We standardized text input by converting all characters to lowercase and removing emojis. User mentions (@USER) were removed via regular expressions to eliminate noisy placeholders introduced by anonymization.
- **Augmentation Techniques:** To increase robustness, we introduced paraphrased samples using **Qwen model** for augmentation, which generated alternate sentence structures. However, directly using the augmented dataset will cause serious overfitting because the model can already classify most of the tweets correctly. Therefore, we first performed inference on the augmented dataset using the trained model and collected only those samples that were misclassified. These hard examples were then mixed with the original training set in a 1:1 ratio for fine-tuning. This strategy enhances the model's capability while mitigating risks of overfitting and catastrophic forgetting.
- **Hyperparameter Configuration**

Hyperparameter	Value
Optimizer	AdamW
LR (classifier head)	1e-3

1. *https://github.com/Refursion/UIUC-ECE-364-SP2025-Project-Topic-2*

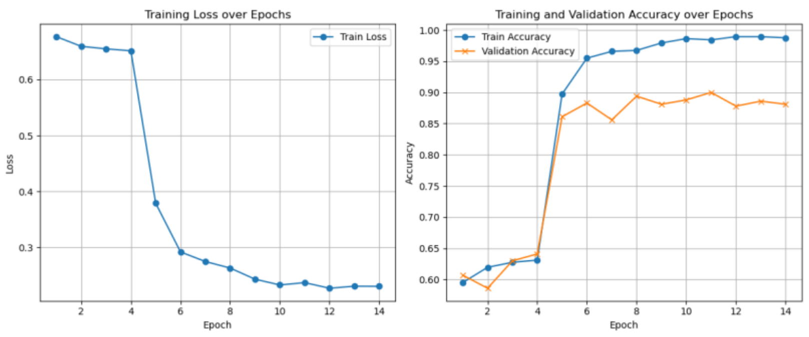
LR (transformer body)	2.5e-5
Batch Size	16
Epochs (max)	10
Scheduler	ReduceLROnPlateau
Dropout Rate	0.3

4. Code Structure Overview

- **batch_train.py:** Orchestrates the training loop, logging, evaluation, and model saving.
- **ProjectDataloader.py**: Defines a PyTorch-compatible Dataset class that handles text preprocessing, tokenization with ALBERT or TinyBERT tokenizer, and label assignment.
- **unfreezer.py**: Implements the progressive layer unfreezing logic. This custom class gradually unlocks layers of the model for better training.
- **generate_csv.py**: Generates the prediction.csv file from the trained model on the test set.
- **test.py**: Loads the trained model and runs evaluation or prediction on the test dataset.
- **DownloadNecessaries.py**: Utility script to download required pretrained models and tokenizers.
- **augment_dataframe.py & run_augmentation.py & filter_misclassified.py:** Codes that are used for data augmentation. Details are included in README.md.
- **finetune.py:** Performs a second stage of training using a mix of original and hard samples with flexible ratio control.

5. Results and Analysis

The result of the first stage training is shown in the figure below. It reaches the maximum valid accuracy of 90.0% at epoch 11. The test accuracy is 88.7%.



Based on this model, we used augmented dataset to finetune, with lower learning rate and epoch number. The final ALBERT model achieved **89.1% prediction accuracy** at test dataset. Compared to TinyBERT trained under similar conditions, our fine-tuned ALBERT achieved a 2.5% higher accuracy and better robustness to noisy labels. Training loss decreased smoothly, and validation metrics remained stable throughout. The use of a learning rate scheduler and dropout proved crucial in preventing overfitting. Augmented data improved robustness, and progressive unfreezing boosted contextual comprehension.

6. Conclusion and Future Work

This project demonstrates that ALBERT, when paired with careful fine-tuning strategies, can serve as a highly effective classifier for short-form COVID-19 tweet classification. Future work includes scaling up data augmentation, exploring semi-supervised learning with pseudo-labels, and combining multiple compact transformer models. Our work shows that even lightweight models can perform competitively with the right training regime and thoughtful design choices.