Goal and Motivation of the Project

The goal of the project was to build an email classifier capable of separating legitimate emails from phishing emails using a pretrained transformer model DistilBERT. The model is fine tuned by unfreezing a few top layers on the phishing email dataset. Motivation was to learn more about transformer architecture and the use of pretrained models such as DistilBERT. We set out to test the performance of the pretrained model by testing the fully frozen and partially frozen model, and comparing it to performance of simpler networks such as FCN, and a self-made transformer.

The finished project should yield a working email classifier that has been validated and is capable of accurately distinguishing between harmful and safe emails. Additionally, the project provides insights into transformer architecture and its functional basis in context and linguistic details.

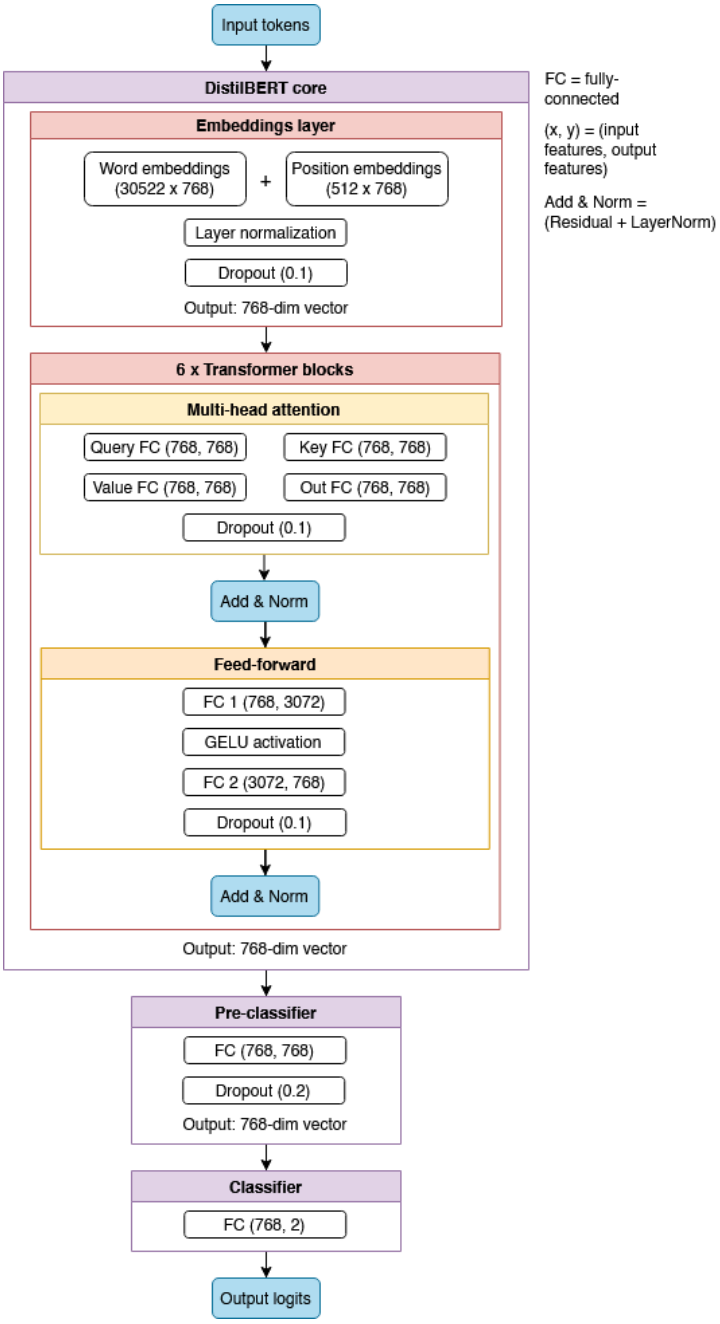
Data Understanding and Preparation

The dataset used is a phishing email dataset composed of legitimate and scam emails from six different sources: SpamAssasin, Nazario, Nigerian_Fraud, CEAS08, Enron, and Ling. The datasets are combined into one dataset with two columns: a text body, which includes the subject line, body, date, and sender email from the initial datasets, if available, and the target labels. Further details can be found in the Jupyter notebook.

To prepare the dataset for model usage, 408 duplicate texts are removed, while keeping only the first instance. Then the texts are tokenized using the pretrained model DistilBERT. We use a pretrained tokenizer, as emails often lack grammatical structure and wider context. We believe that a pretrained tokenizer will allow our model to generalize better, especially when processing emails from contexts outside our six sources. More information can be found in the Jupyter notebook.

Model Architecture and Reasoning

Initially, BERT was considered for the task based on suggestions from our teacher. However, the task doesn’t require the full capacity of the original BERT model, and given our limited computational resources and time, DistilBERT was opted for instead. The model is a distilled version of BERT, but retains nearly similar performance and trains 60% faster. It has approximately 66 million parameters compared to base BERT’s 110 million, making DistilBERT 40% smaller. In other words, it provides an optimal balance between performance and efficiency.



As seen in image 1. The model consists of a DistilBERT encoder core, a pre-classifier layer and a classifier layer.

The DistilBERT core has an embedding layer and six transformer blocks. The embeddings layer converts each token into a 768-dimensional vector by combining word embeddings with positional embeddings. Each transformer block has a multi-head attention mechanism with four fully connected layers (query, key, value, out), layer normalization, a feed-forward network with two fully-connected layers, and dropout for regularization.

The pre-classifier takes the output from the final transformer block and applies another fully-connected layer with dropout to adapt the presentation for spam detection.

The classifier layer transforms the 768-dimensional vector into two output scores representing the model's predictions for each class.

No modifications are done on the architecture. However, the first five layers are frozen (embeddings, 4 transformer blocks) and the rest (2 transformer blocks, pre-classifier, classifier) are fine-tuned.

Image 1. Model architecture diagram

Training and Hyperparameter Tuning

The table below summarizes 6 training cycles with mostly different configurations. The main variations between runs are the number of trainable transformer layers in DistilBERT, as the top-most hidden layer and classification head are always trainable. For all cases, training progress peaked at around the 4th or 5th epoch, with either overfitting or negligible gains present after that point. Reducing the learning rate didn't really help in further improving

performance, nor did it impact training time. Early stopping and restoration of best weights were useful in avoiding useless training and preventing overfitting.

The most important factor that affected performance was the number of trainable layers, although all 3 variations performed exceptionally. We didn't want to experiment with more than 2 trainable layers, as the model exerted above 99% accuracy and minimal validation loss with 2. Every additional trainable layer greatly increased training time, further reducing the appeal of further experimentation.

Variant E was chosen as the final model. Although A had essentially the same hyperparameters, we were unable to re-achieve its better metrics due to random variance. 2 trainable layers was the sweet spot between performance and trainability, which is also confirmed by some other sources training only specific layers of DistilBERT.

Experiment	Trainable Layers	LR / Scheduler	Early Stop	Epochs Run	Best Epoch	Val Acc	Val Loss	Time / Epoch (A100 GPU)	Total Time
A	2	5e-5	No	**10**	**5**	0.9937	0.0172	~3:54	~39 min
B	1	5e-5	No	**10**	**8**	0.9923	0.0253	~3:29	~35 min
C	2	1e-5	Yes	**4**	**3**	0.9890	0.0300	~3:52	~11.5 min
D: ReduceLROnPlateau	2	5e-5 → 5e-6	Yes	**7**	**4**	0.9933	0.0222	~4:02	~28 min
E: ReduceLROnPlateau, Rerun of D	2	5e-5 → 5e-6	Yes	**11**	**8**	0.9939	0.0209	~6:23 (L4 GPU)	~70 min
F: ReduceLROnPlateau	0	5e-5	Yes	**10**	**10**	0.9683	0.0813	-3:00	~30 min

Table 1. Training cycles

Results Analysis and Visualization

Below are the accuracy and loss metrics of model E. Even after the 1st epoch it had an exceptional validation accuracy and loss. This is likely due to DistilBERT being highly performative to begin with and the dataset being comprehensive.

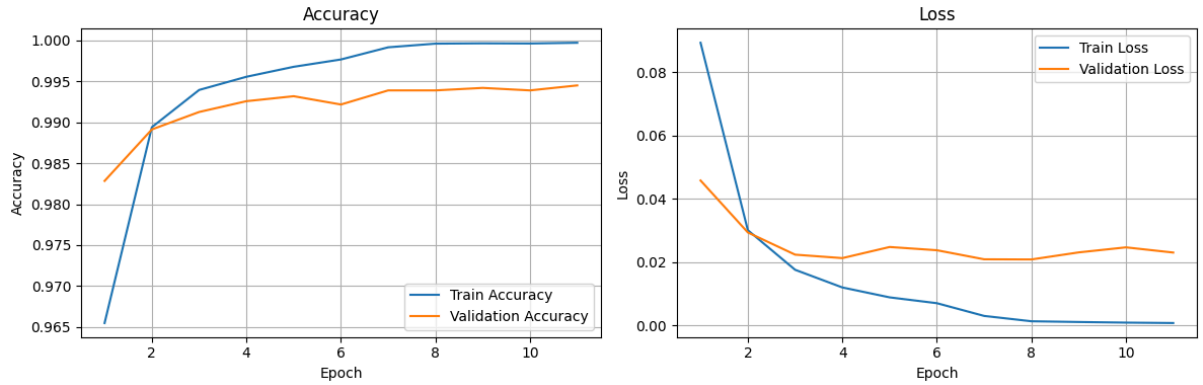


Image 2. Accuracy and Loss curves

The accuracy and loss curves demonstrate stable and efficient learning over the training epochs. Most performance gains occur during the first few epochs, after which both training and validation accuracy stabilize and improve only marginally. The close alignment of training and validation accuracy across epochs indicates good generalization and no significant overfitting. Similarly, training and validation loss decrease rapidly in the early epochs and then stabilize without divergence. This suggests that the model converged effectively and that the selected number of training epochs was sufficient for learning the phishing classification task without unnecessary overtraining.

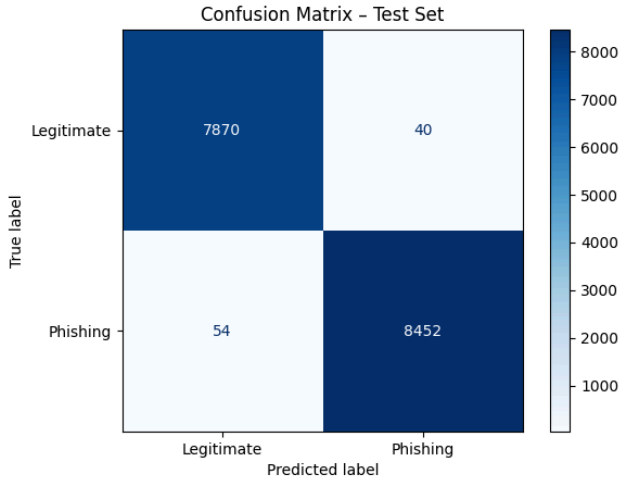


Image 3. Confusion matrix

The confusion matrix indicates strong performance across both classes. The model correctly classifies the vast majority of legitimate and phishing emails, with only a small number of misclassifications. False positives are rare, and while false negatives are more critical in a security context, their occurrence is minimal. Overall, the results confirm that the model reliably distinguishes between legitimate and phishing emails with high accuracy.

Comparison against other models

Our DistilBERT-based model is also compared against the performance of two other models with the same task. One is a more simple transformer model with only 2 total blocks, while the other is a Convolutional Neural Network (CNN) using 1-Dimensional Convolution layers to find relationships between tokens. Performance and training metrics are compared below:

Model	LR / Scheduler	Epochs Run	Best Epoch	Val Acc	Val Loss	Time / Epoch (A100 GPU)	Total Time
Fine-tuned DistilBERT (Variant E)	5e-5 → 5e-6	**11**	**8**	0.9939	0.0209	~6:23 (L4 GPU)	~70 min
Simple Transformer	5e-5 → 5e-6	**5**	**4**	0.9735	0.0745	~5:15	~26 min
CNN	5e-5 → 5e-6	**18**	**16**	0.9916	0.0264	~1:40	~30 min

Table 2. Comparison table

All models took roughly the same time to train (considering that the L4 GPU is roughly twice as slow the A100 GPU), although the CNN model was significantly faster per epoch, while the simple transformer was noticeably slower. All models perform well, although the simple transformer falls noticeably behind the other two in both validation loss and accuracy. Its performance could have been improved by adding more layers and thus enabling it to learn more complex patterns, although this would've defeated the purpose of the fine-tuned DistilBERT model and would've taken forever to train on limited hardware. The CNN model came surprisingly close in performance to the fine-tuned DistilBERT model. Firstly, it also benefits from the DistilBERT tokenizer. Secondly, as spam detection is a more lexical than semantic task, CNN models with 1-dimensional convolutional layers are effective at identifying patterns that are typical of scam.

Challenges and Solutions

One challenge was that the *AutoModelForSequenceClassification*-method did not work properly with Keras TensorFlow. The working approach required us to switch to PyTorch, which we had no experience with since all the project and course material so far was done using TensorFlow. This required some time and consideration on how training, evaluation and model architecture are done with PyTorch.

The pre-trained model introduced some challenges, as the architecture was unknown to us. Understanding it was difficult as there was minimal detailed documentation available, and some of it needed to be assumed based on common practises.

Conclusions and Further Developments

Conclusions

This project demonstrates that fine-tuning a pretrained transformer model, in this case DistilBERT, is an extremely effective approach for phishing email detection. The results show that limited (unfreezing only top 2 layers) fine-tuning yields excellent performance, reaching nearly 99.4% validation accuracy and with very low validation loss. This confirms that most of the linguistic knowledge required for the task is already captured in the pretrained model, and only minimal task-specific adaptation is needed.

An important finding of this project is that simpler architectures can still perform competitively for phishing detection. The CNN-based model achieved surprisingly strong results, nearly matching the fine-tuned DistilBERT model. This can be explained by the lexical nature of phishing emails, where characteristic word-level patterns are often suitable for classification. When combined with a pretrained tokenizer, CNNs can be a computationally efficient alternative, especially if considering large-scale or real-time applications. But in contrast, the self-made transformer model underperformed compared to the other models, showing the advantage of pretrained models over training transformer architectures from scratch with limited data and hardware.

Beyond model performance, the project provided other valuable insights. The transition from TensorFlow to PyTorch was a significant challenge, as the *AutoModelForSequenceClassification* interface did not integrate properly with Keras. Learning PyTorch required understanding training loops, evaluation procedures, GPU utilization, and model parameter freezing. Additionally, interpreting the architecture of a pretrained model proved difficult due to limited low-level documentation, requiring reliance on official resources, common practices, experimentation, and AI-assisted explanations. Random variance in training outcomes also affected reproducibility, emphasizing the inherent non-deterministic behaviour of deep learning and the importance of multiple runs and careful model selection.

Overall, the project achieved its primary goal of building a robust and validated phishing email classifier while deepening the team's understanding of transformer architectures, pretrained models, and fine-tuning strategies in a real-world machine learning task.

Further Developments

Several directions for future work were thought over. One clear improvement would be to train and evaluate the models on larger and more diverse datasets, which could further improve generalization to unseen phishing strategies and email formats. Finding these diverse and balanced datasets just proved to be quite difficult. Exploring alternative pretrained transformer models such as RoBERTa or DeBERTa could also yield performance

gains, as these models are known to capture richer contextual representations than DistilBERT, but at a higher computational cost.

Another promising direction is deploying the model in a real-time email filtering system. In such a setting, careful calibration of the model would be necessary to reduce false positives, as incorrectly classifying legitimate emails as phishing can be even dangerous for users. Tuning the threshold or human-in-the-loop validation could be explored to address this issue.

Finally, further experimentation with some hybrid approaches, such as combining CNN-based lexical pattern detection with transformer-based contextual understanding could offer a strong candidate between performance and efficiency. These extensions would build upon the strong foundation established in this project and move the system closer to practical, deployable phishing detection solutions.

AI-Usage

AI was used to troubleshoot during development. It also advised us how to move from the TensorFlow approach to PyTorch, and generate code. A small portion of our original text was also reworded and formatted using AI.