REPORT

Problem Statement:

- **Fine-tuning opensource LLM (llama, mistral, phi2, zephyr etc) on a domain specific data**

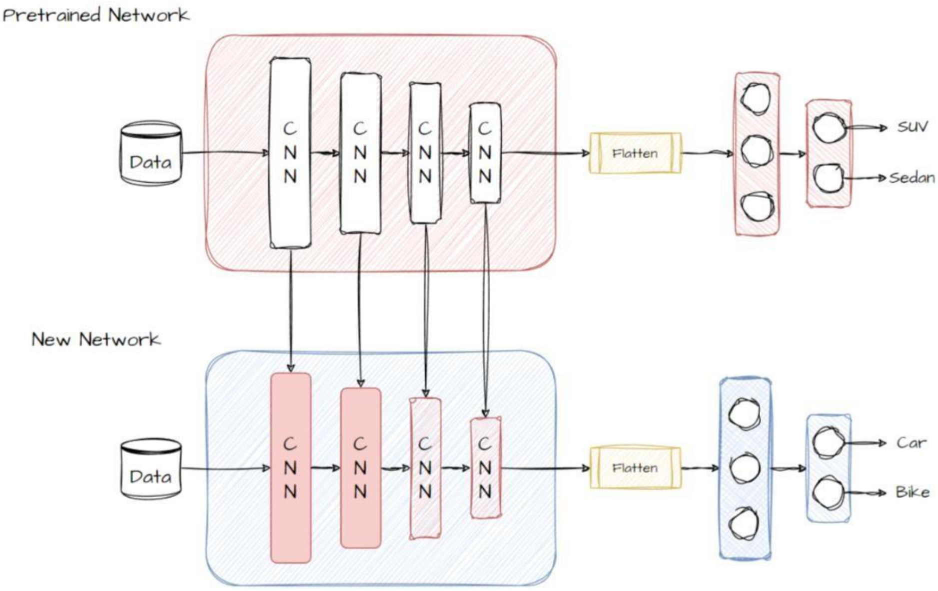
Plan of Action:

1. Deciding what kind of model to develop: A distil-BERT-uncased (HF: distilbert-base-uncased · Hugging Face) model will be fine-tuned using the IMDB Dataset to predict the sentiment of a **Movie Review better than the base model.**
2. Dataset to be used: Sourced from HuggingFace (also available on Kaggle).
3. Pre-processing of the Data.
4. Model Setup.
5. Model Training.
6. Manual Testing.
7. Deployment on Streamlit.

Methodology: I have used Parameter Efficient Fine-Tuning (PEFT), technique where in the original weights of model are retained. This number can be adjusted depending on our computational power.

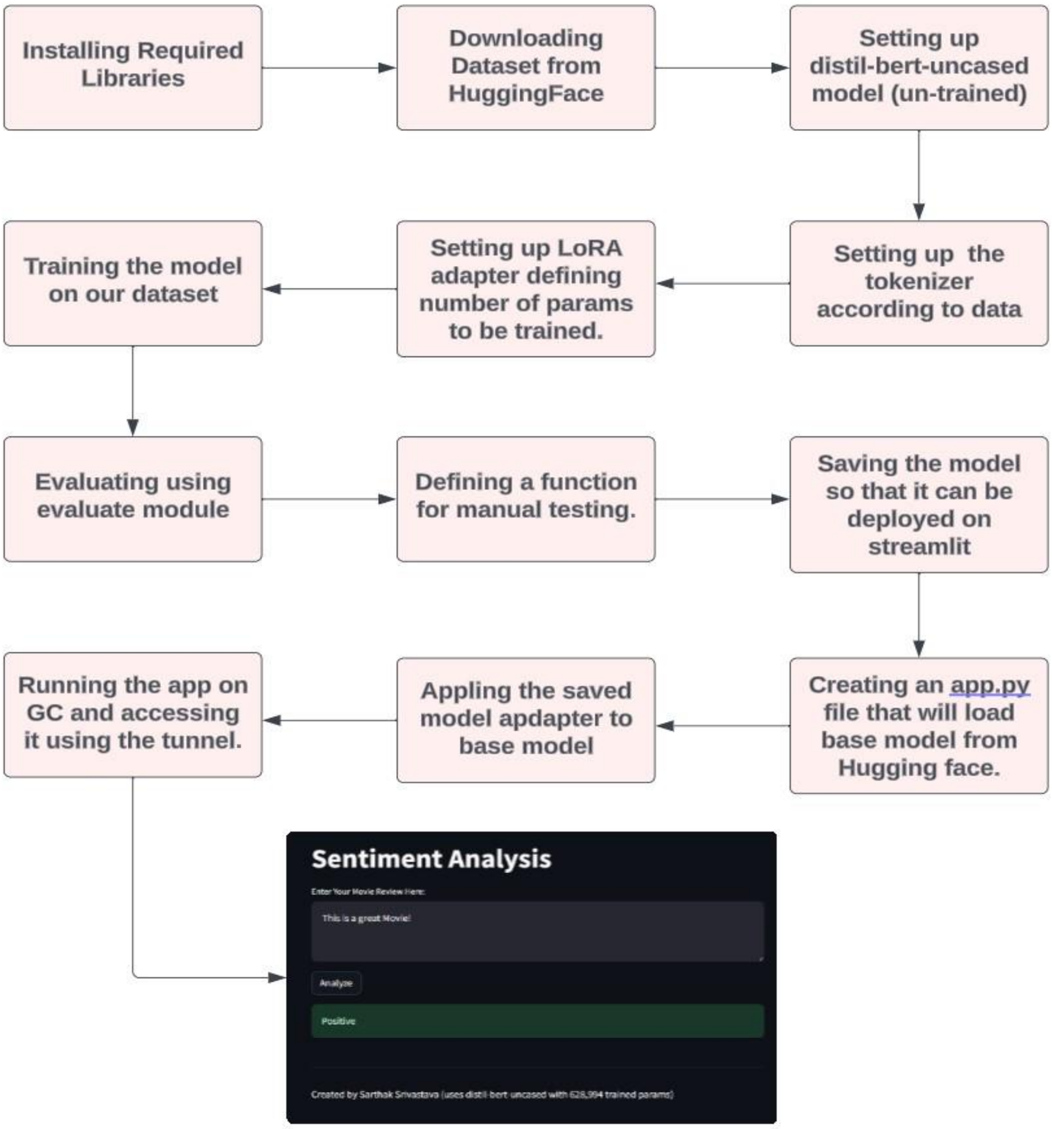
PEFT is a technique used in Natural Language Processing (NLP) to improve the performance of pretrained language models on specific downstream tasks. It involves reusing the pre-trained model's parameters and fine-tuning them on a smaller dataset, which saves computational resources and time compared to training the entire model from scratch.

In the example, I have deployed the model and Streamlit app on Google Collab itself. Distil-bert is a big model, containing around **67 billion params, I have augmented 628,000 of these parameters.** This method can minimize the number of trainable parameters by up to 10,000 times and the GPU memory necessity by 3 times while still performing on par or better than fine-tuning model quality on various tasks. Lets understand the architecture with this given diagram, where a big model which is built to identify type of vehicle i.e SUV, Sedan, MUV is down streamed to only tell if a vehicle in a picture is Bike or Car.



Most weights are retained, only a small number of weights are fine-tuned.

Code Walkthrough:



Steps to replicate the model:

1. Open Google Collab > New Notebook > Upload Notebook > Upload the provided .ipynb file.
2. In the same GC window from the LHS upload the provided 'app.py' file. Make sure the path of this file is '/content/app.py' as this is hardcoded in the code file or else adjust the path manually in the code. (This file will be used later on to deploy streamlit web application).
3. Change runtime from CPU to T4 GPU or faster if available.
4. In the Runtime tab > Run all
5. Wait for the code to run, at the bottom you will be provided with a URL and an IP address (GC server Endpoint).
6. Go on the provided link (Example: https://cyan-berries-pull.loca.lt/) and paste the IP address provided.
7. Voila! You will the application running in your browser, you can share the URL with someone else as well and they can access it too. This is a random URL and will change everytime the file is run.

Results and Conclusion:

The last epoch of the training reported:

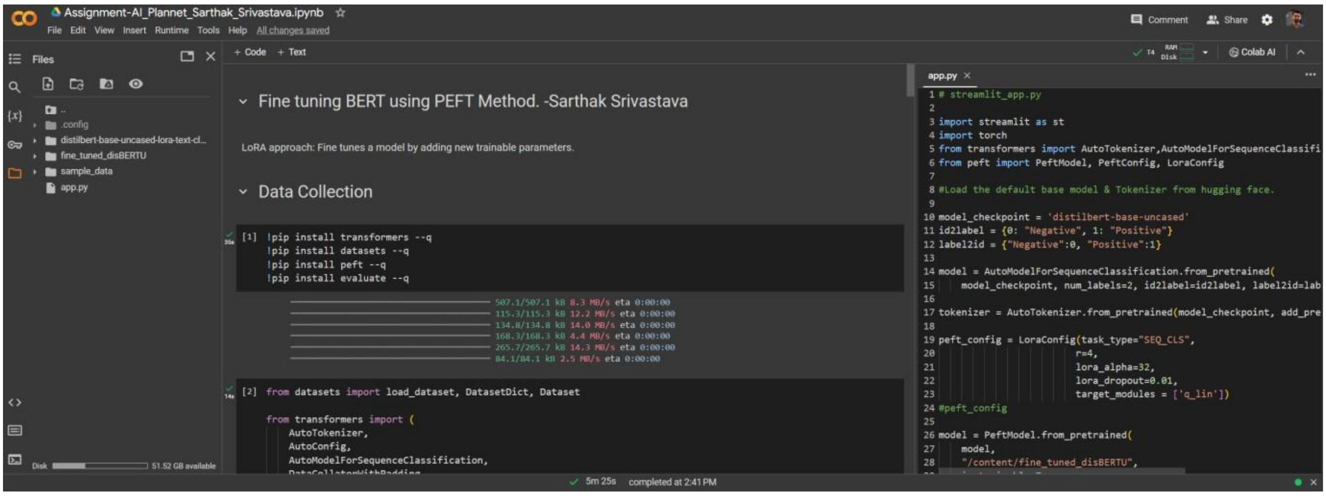
Copied directly from Notebook.

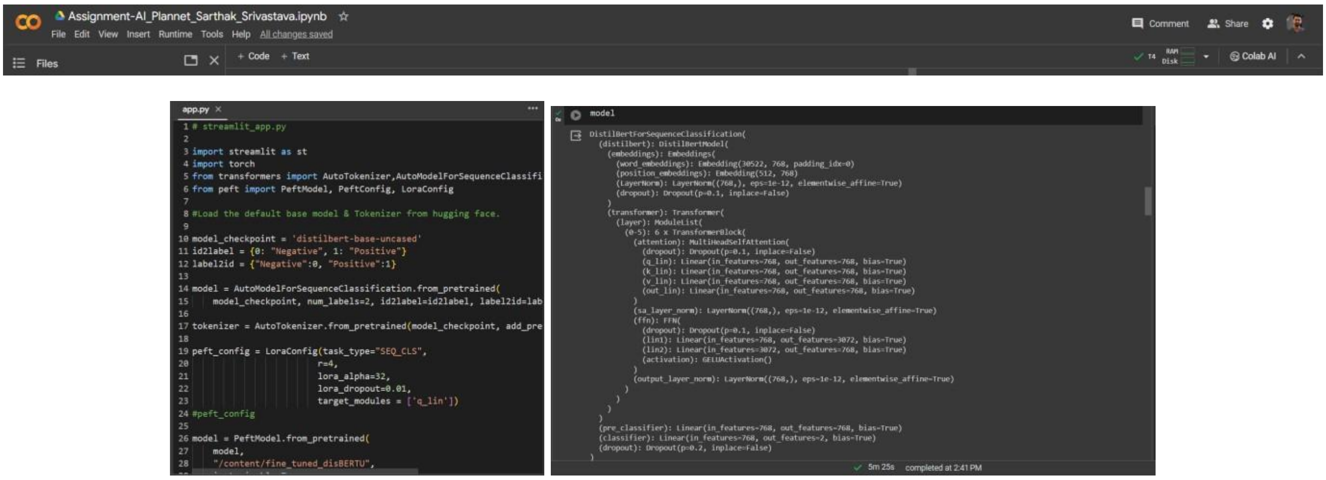
Epoch	Training Loss	Validation Loss	Accuracy
10	0.014700	0.760809	{'accuracy': 0.902}

The results can be easily improved if more computing power is used by:

1. Adding more parameters to train.
2. Running more epochs.
3. Adding more train data.

Some screenshots from the code:





The model was successfully deployed and tested. Please see the demo video for more details.

Sarthak Srivastava
Email: sarthak.s.1603@gmail.com