Assignment 3: Fine-tuning Pretrained Transformers

In this assignment, you will explore how large pretrained Transformer models can be adapted to specific downstream tasks. You are expected to apply the concepts from lectures and implement a working system that demonstrates fine-tuning in practice.

Task Description

- Choose a pretrained Transformer model (e.g., BERT, RoBERTa, DistilBERT, GPT-2). You may use a smaller model if computing resources are limited, and you are welcome to use free cloud platforms (e.g., Google Colab, Kaggle) or other resources. You can use a standard library such as HuggingFace Transformers.
- Select a **domain-specific dataset** (e.g., biomedical, legal, financial, social media, or a general benchmark) for either:
  1. **Text classification** (e.g., sentiment analysis, topic detection), or
  2. **Question Answering (QA)** (e.g., extractive QA from domain-specific passages).
- Implement at least two fine-tuning strategies, such as:
  - **Full fine-tuning:** update all model parameters (optional if resources are insufficient),
  - **Parameter-efficient tuning:** e.g., LoRA, adapters, or prompt-tuning.

  You are encouraged to compare different strategies (e.g., full fine-tuning vs. LoRA).
- Evaluate your model using appropriate metrics:
  - For classification: accuracy, F1 score, or other standard metrics.
  - For QA: exact match (EM), F1 score, etc.
- Report and analyze your results: performance differences, efficiency trade-offs, error cases, and lessons learned.

Report Guidelines

- Maximum length: **6 pages** (including figures/tables, excluding references).
- Your report should clearly describe:
  1. The dataset you chose and the motivation for your choice.
  2. The model(s) and fine-tuning strategies used.
  3. Experimental setup (training details, hyperparameters if relevant).
  4. Results (with tables/plots where appropriate).
  5. Key takeaways and limitations.

Notes

- You are free to choose any dataset and pretrained model, as long as it is non-trivial and your choices are justified.
- You may use any modern framework (e.g., PyTorch + HuggingFace).
- Discussions with peers are allowed, but all code and reports must be completed **individually**.
- For aspects not explicitly mentioned above, you have full freedom — just explain your decisions clearly in the report.

Code Submission

- You must provide a GitHub repository link containing all your code.
- The repository should include:
  - A clear `README.md` with instructions to install dependencies and reproduce your experiments.
  - Clean and well-organized code (with comments where necessary).
  - A simple entry point (e.g., `main.py` or `run.sh`) to reproduce results.
- Do not upload large datasets or pretrained models — instead, provide download links or scripts in your README.
- Make sure the repository is accessible (e.g., add the TA as a collaborator if it is private: rundong25, DoudouZhou).

Grading Policy

- Completing all the requirements in a correct and clear manner will earn you a solid grade.
- To achieve a top grade, you need to go beyond the minimum requirements: demonstrate originality, critical analysis, or particularly strong execution (e.g., insightful error analysis, thoughtful comparisons, creative experiments).
- Do not ask what counts as "originality" or "own ideas" — it is up to you to explore, experiment, and showcase your creativity.

Due date: 10/17.