Deep Learning Project 2

Spring 2025

Competitions end 11:59pm, April 18, 2025.

Please upload your project report before 11:59pm, April 21, 2025.

This project will comprise 15% of your overall grade. Please perform this project in groups of (at most) 3. If you are looking for project team members, please broadcast your interest in the Class Slack in the `find-a-teammate` channel.

Goal

In this Kaggle competition you are tasked with coming up with a modified BERT architecture with the **highest test accuracy** on a held-out portion of a text classification dataset called "AGNEWS", under the constraint that your model has **no more than 1 million trainable parameters**.

You will start with a specific version of BERT (called RoBERTa). The only type of modification you are allowed to make to BERT is low-rank adaptation [LoRA]. See here or here for an explanation of what LoRA is; the high level idea is that each (frozen) weight matrix in BERT is perturbed by a trainable low-rank matrix, where you can set the rank and the strength of the perturbation. More details are below, and there is also a demo notebook posted on the Kaggle competition site, which uses a popular LLM fine-tuning library called BitsAndBytes.

Details about LoRA

Here is a schematic of how LoRA looks like:

There are two main advantages of doing this. First, there are far fewer trainable parameters, so gradient computation is much easier and can be done on much cheaper hardware. The second is we keep the pre-trained weights frozen and only train (adapt) the new weights to the given task, so it is easy to switch out adapters, merge more than one adapter, etc.

The main control knob in LoRA is the rank of the matrix AB (ie the smaller dimension of either A or B), denoted by r. Another control knob is the *strength*, α, which is a scalar multiplied with AB when added to the original weight. Small values of α are used when we don't want too much change in the base model weights, and vice versa.

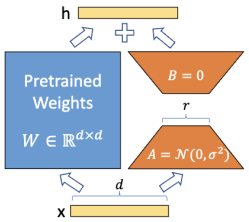

Figure 1: Our reparametrization. We only train A and B.

Figure 1: Block diagram of LoRA

You are free to experiment around and adjust these knobs *per layer* or even *per weight matrix* to gain boosts in accuracy, as long as the *total* number of trainable parameters does not exceed 1 million. (You can use the `torchsummary.summary` function to check the number of trainable parameters in your model.)

You are also free to experiment with:

- any optimizer (ADAM, RMSProp, Muon, etc)
- any data filtering strategy (e.g. you can throw away lengthy or weird reviews according to metrics of your choice.)
- any regularizer
- any choice of learning rate, batch size, epochs, scheduler, etc.
- other tricks such as teacher-student distillation, or quantization (QLoRA), or etc.
- any data augmentation strategy (e.g. you can choose to reword/rephrase/cleanup training samples using whatever technique you like). *This is a very deep rabbit hole so I wouldn't recommend doing this before you have tried other things.*

You are **not** allowed to:

- simply load pre-trained model weights from the web other than the BERT model.
- use other/bigger datasets for training your LoRA weights; use only the given dataset.

Resources (optional)

If you do use external code or coding LLMs, please include a clear citation. You are free to use any other online resources and/or techniques you like, as long as you include citations.

You can also use NYU HPC.

Deliverables

This project has two main deliverables:

- A project report (7 points).
- A project codebase (7 points) in the form of a Github repository.

Teams finishing within the top 20% of the teams in the competition will be given 1 point,

There is a **bonus** 2 points for the top team, and a **bonus** 1 point for the runner-up team.

Projects will be graded on:

- clarity and quality of submitted project report;
- quality of final results. Aim for test accuracy of at least 80% as a minimum baseline.
- and clarity and quality of submitted codebase. Include notebooks with clear plots; in particular, we want to see your code execution. Include statements where you clearly print the final test accuracy and number of parameters.

Project report

Your report has to be **no more than 4 pages long** including all figures, tables, and citations, **typeset in two-column AAAI 2024 camera-ready format**. Any report not in this format will not be graded. Here is a link to the format in LaTeX and MSWord formats; see the "Camera-ready" folder in this link for example documents.

Please upload a PDF of your report to Gradescope. **Only one team member should upload** on Gradescope, as long as they tag all other team members.

In your report, please include:

- Names of all team members
- A short overview of your project, along with a summary of your findings
- A methodology section that explains how you went about designing and training your models, pros and cons of different hyperparameter choices, what lessons you learned during the design process, etc.
- A results section that reports your final test accuracy, model architecture, and number of parameters.
- Any relevant citations.

Project codebase

In the first page of your report, please provide a link **on the first page** to a **publicly accessible** Github repository. Your repository should contain:

- the code necessary to reproduce the results in your report.
- well-documented code and/or Jupyter notebooks for easy visualization and verification of your work.

Once the competition is complete the TAs will reach out to the top performing teams to reproduce your code to make sure your results are valid. Therefore, readability and usability of your code is essential; please prioritize this.