E0 270: Assignment 2

Due date: April 27, 2024

Instructions

1. Download and extract the assignment template.
2. The report must be typeset in LaTeX, Ensure that everything is in a single pdf file and all pages are in correct order.
3. You must use PyTorch, however we are not imposing any version mandates. It should be fine as long as you use the latest stable version. You will need to additionally install the `transformers` library for loading the pretrained transformers weights from `huggingface`.
4. If you feel any particular problem in underspecified, you can safely assume that it has been done intentionally. You are free to make assumptions, but please specify the assumptions in your report. No further clarifications on the problem will be provided. However feel free to contact us if you believe that there is an error in the problem specification.
5. The grading will be relative based on the final performance achieved by your model. You are encouraged to discuss the problems amongst yourselves and engage in a healthy competition. However copying codes from others verbatim, along with the use of LLM assisted code generators are strictly forbidden, and will be penalized heavily if detected.
6. Use the documentation to read about the common type of layers that are offered by the standard frameworks like PyTorch. This will help you in understanding various deep learning concepts that will in turn allow you to improve the performance of your models.

Data

Consider the COLA dataset https://nyu-mll.github.io/CoLA. Each data point is a sentence containing tokens, which can be tokenized using pretrained tokenizers from huggingface. Tokenization and conversion of tokens to the corresponding token ids have already been done for you in the code template. You may also visit https://paperswithcode.com/sota/linguistic-acceptability-on-cola for checking out the current SOTA (state of the art) models and their performances on the COLA dataset.

Problem 0: Generation

We shall be using GPT2 variants as the pre-trained language models for this assignment. In order to check if all the libraries are setup properly, modify the line 24 in `run.py` and execute:
`python run.py gen <Your5DigitSR>`
You should be able to see the language model predictions. Reference: GPT2 Paper

For the remaining tasks, we recommend that you start out with the `gpt2` (base) model first, and only after everything is working try to fine-tune the `gpt2-medium` model if time permits and you wish to obtain improved accuracy.

Problem 1: Low Rank Adaptation

Now that you have setup everything and you are able to generate new tokens, we shall perform PEFT (Parameter Efficient Fine-tuning) on the GPT model. We will use a technique called Low Rank Adaptation or LoRA (reference below). LoRA is a PEFT where auxiliary linear layers are added (injected) in parallel to the Linear layers in the pre-trained model. Rather than fine-tuning the Linear layers in the models, the difference between the fine-tuned weights and the pre-trained weights is stored in the auxiliary LoRA linear layers. This difference is assumed to be low rank, and thus decomposed into two matrices (say L and R martrics) whose multiplication yields a matrix with the same dimensions as the weights of the linear layer corresponding to which the concerned LoRA linear layers were injected.

1. In the `model.py` file, implement the `LoRALinear` class such that it has two matrics as described above. Also the matrices should have a low-rank bottleneck as described in the below mentioned reference.

2. Complete the remaining `TODOs` in the `model.py` file to complete auxiliary LoRA linear layer injections. Remember to freeze the parameters of all the linear layers for which auxiliary LoRA linear layers were injected. Additionally, also freeze both the token and positional embedding matrices. For sanity check rerun: `python run.py LoRA <Your5DigitSR>`. You should still get some generated text even after injection.

3. Fill in the `train_utils.py` file and fine-tune the GPT2 model for the classification task on the COLA dataset by running `python run.py gen <Your5DigitSR>` You may change the hyper-parameters as required. Report any necessary details including but not limited to: your training strategy, the number of trainable parameters, the reduction in parameters, optimal hyperparameters, maximum accuracy achieved on the COLA validation dataset and the training (loss and metric) plots.

Reference: LoRA - Low Rank Adaptation

Note: Since the parameters in the injected layers are much smaller when compared to the GPT model, multiple LoRA fine-tuned models can be trained and stored for different tasks without taking up much space. There are additional benefits for LoRA PEFT, on top of the storage space being very less for the injected layers. Consider a company who wants to deliver multiple "AI" tools to the customer. All that company now needs to do is inject multiple parallel LoRA layers into a pretrained model. The data can then be passed through the pretrained model along with one of the injected models based on the customer task requirement. Thus, if it is assumed that $\mathcal{O}(n)$ GPUs are required for loading one Large Language Model (LLM) (for one task), without LoRA, $\mathcal{O}(t \times n)$ GPUs would be required for providing services for t different tasks. However, with LoRA, the company can still manage to provide t servies with just $\mathcal{O}(n)$ number GPUs.

Problem 2: Knowledge Distillation

Now that the GPT2 model is trained for classification on the COLA dataset, we shall look at a way to distil the knowledge contained in the fine-tuned GPT2 model to a much smaller model. This process is known as Knowledge Distillation (KD) (reference below).

1. Create a Recurrent Neural Network (RNN) model of your choice by filling up the `DistilRNN` model class in the `model.py` file.

2. Consider the teacher model as the fine-tuned GPT model, and the DistilRNN as the student model. For the COLA classification dataset, transfer the knowledge from the teacher to the student model by running `python run.py distil <Your5DigitSR>`. Report the relevant details including but not limited to: the loss function used for distillation, architecture details of your DistilRNN implementation, number of parameters in the DistilRNN, optimal training hyper-parameters, maximum accuracy acheived on the COLA validation dataset by the DistilRNN model, and the training plots.

3. Use the DistilRNN class to instantiate a new model, and train that model from scratch without knowledge distillation on the COLA dataset for the classification task. Compare the performance of this model trained without KD with the previous KD training on the COLA validation set.

Reference: KD - Knowledge Distillation

Deliverables

- Completed code for LoRA fine-tuning and KD of the LoRA fine-tuned model.
- A report containing the complete details of the algorithm and your implementation, along with justification for any extra assumptions made if necessary. Report just the losses and applicable metrics.
- Training plots corresponding to losses and accuracies for both train and validation set.

Submission

Attach **a single zip file** named in the format `Asst2_FirstName_LastName_5DigitsOfSRNo.zip` to the assignment in Teams, before the due date. The zip file should contain your code that you filled up, your report (in a pdf file named `Report_<5digitSR>.pdf`), and any plots that you put in the report. Ensure to upload the zip file on Teams submission portal on or before April 27, 2024, 11:59 pm. Ensure that you do **not** include the data files and pycache files in the code that you submit.