# Assignment 4: RNNs, LLMs, and Fine-Tuning

ABE 6033

**Due:** November 22nd

## 1 Introduction

This assignment has two parts. First, you will develop a basic RNN using Numpy, similar to the previous assignment on logistic regressions. You will learn how to implement both basic RNNs and LSTMs. Note that this notebook has a section on backpropagation, which is optional. You will not receive extra credit for completing this section.

In the next problem, you will learn how to fine-tune a Large Language Model (LLM). LLMs are some of the biggest models that you will likely encounter, and working with them poses some practical challenges. **For that reason, this assignment is designed to be completed on HiperGator.**

In this assignment, we will first work through the process of finetuning Llama3-8B on a custom dataset. Then, you will fine-tune the model on a dataset of your choosing. Llama3-8B is the smallest version of Meta's Llama3 model, which achieves state-of-the-art performance on a variety of benchmarks.

As the name implies, Llama3 has 8 billion parameters. Directly fine-tuning a model that large would likely require multiple GPUs and is outside the scope of this assignment. To alleviate this, we will use Low Rank Adaptation (LoRA), which significantly reduces the number of parameters that need to be trained (down to about 40 million).

For this assignment, I will expect your submission to contain all your code and a brief report describing what you did. Because the saved Llama3 model can be very large, you **do not** need to include trained models in your submission.

### 1.1 Grading

This assignment is worth 100 points total. The point breakdown is as follows:

- RNN: 50 points
- LLM fine-tuning: 25 points
- LLM fine-tuning on your own data: 25 points

Note that the grading on the final section (fine-tuning on your own data) will be fairly lenient. You will get most of the points as long as you demonstrate

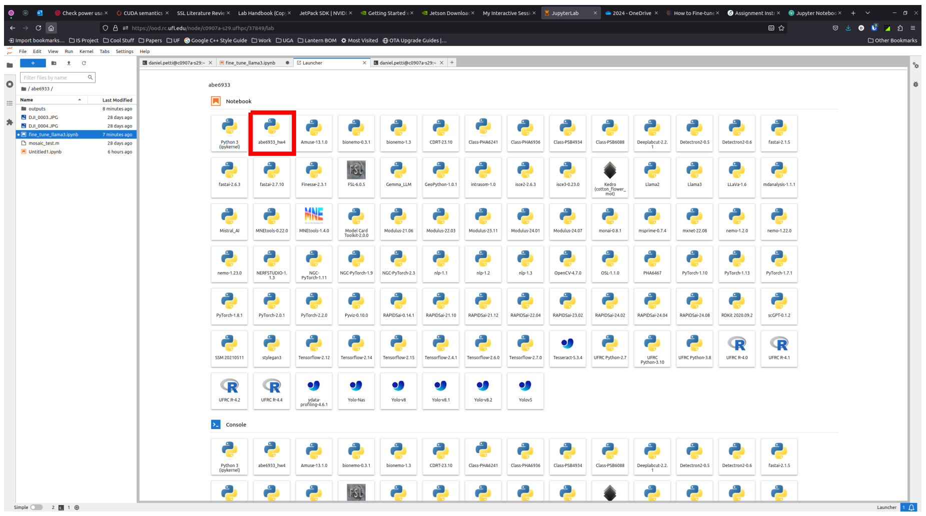

Figure 1: What the new tab page in Jupyter should look like once you have set up the kernel.

that you can train the model on the datasets, even if the resulting model doesn't work very well.

## 2 Set up Jupyter Lab

Start a new Jupyter Lab session on HiperGator. You can experiment with the amount of resources, but I recommend at least 4 cores and 32 GB of RAM. Be sure to also select an A100 GPU. If you don't know how to use HiperGator, please watch this video.

This assignment requires some dependencies that are annoying to install, so I have created a custom iPython kernel for you that has them all installed already. The first time you open Jupyter, you will likely need to add the kernel. Open a terminal, and run the following commands:

```
source /blue/abe6933/share/hw4_venv/bin/activate
ipython kernel install --name "abe6933_hw4" --user
```

Now, when you open a new tab in Jupyter, you should see an option to use the "abe6933_hw4" kernel (Figure 1). If you do not see this, you may have to restart the Jupyter server.

## 3 RNN Exercises

This notebook is located in the "notebooks" folder of the provided .zip file. I suggest that you run it on HiperGator using the "abe6933_hw4" kernel that you set up in the previous step in order to avoid dependency issues. Please note that to receive full credit on this portion of the assignment, all the unit tests must pass. Do not modify any code outside of the labeled blocks.

## 4 Your First Fine-Tuning!

In this section, we will fine-tune Llama3 on a simple dataset. First, open the "fine_tune_llama3.ipynb" file included in this assignment. Note that you might have to adjust the kernel to use the custom one you added. With the correct kernel, you should not have to install any dependencies.

As configured, the notebook will perform the following steps:

- Download the model.
- Configure LoRA
- Download and prepare the Alpaca dataset
- Configure a training loop
- Train the model
- Test model inference
- Save the model locally (if you want).

Note that the training can take a substantial amount of time. Once you are finished, you should see the model produce reasonable output for the final inference test. Note that the output might vary between runs, because the model is configured to sample the output distribution stochastically.

In your submission, please include your Jupyter notebook, with all of the output from each step.

## 5 Fine-Tune on your Own Data

For the last part of this assignment, you will choose another dataset and fine-tune the model on that. You can use any text dataset that you want. Here are some options to get you started:

- Guanaco
- Aya
- UltraChat 200k

- Grade School Math
- Chat Doctor
- OmniMATH
- Ag Injury News Dataset (for advanced users)

You will need to demonstrate that your fine-tuned model works. Include some example prompt-response pairs from your fine-tuned model in your report. Also include any code you wrote along with the intermediate output that it produced.

Feel free to modify any part of the provided notebook. In particular, you might want to experiment with the number of steps you train for (controlled by the "max_steps" or "num_train_epochs" arguments when creating the SFT-Trainer).