**L03 AWS MLU Reflection**

**Lab 01: Getting Started with PyTorch**

**Key Learnings:**

This lab was split into two different parts, the first part focusing on an introduction to the deep learning framework of PyTorch. I learned how to setup PyTorch with installing the libraries, and also what tensors were and some of their basic features. I got information about common operations it was able to complete and how to perform them on data.

This first part also went over an important use of indexing and slicing tensors, with how to convert them into different objects. Then the last part of the lab I learned about was differentiation and how training between CPU/GPUs are different, as well as the difference/conversion of NumPy and PyTorch tensors.

The second part of the lab just had me examine a neural network to see different parts of its architecture. I learned how to use the basic components within a neural network, and the implementation of one through PyTorch. I also went over how to generate simulated datasets, and how to train a neural network.

I don’t feel like I learned anything that’d I’d consider “key” parts to summarize. The lab felt like a general overview where I can’t piece out what I learned outside of the summary of the lab itself. But I definitely think the most interesting part was going over the architecture of the neural network, as even now they’re confusing to understand.

**Insights/Understanding:**

One of the insights I gained during the first portion of the lab was how much PyTorch simplifies complex operations, like with only a few lines of code PyTorch is able to completely create and perform calculations on a matrix. With how much PyTorch simplifies these It makes the process of training models way easier and faster.

I also think I gained a better understanding of a neural network’s base architecture, I feel like I can understand the basics of how to alter one if I was just given a model and need to find where it’s not working, although I’m definitely not confident enough to build a model yet.

**Challenges:**

The main difficulty from this first lab wasn’t necessarily to do with the lab itself, but the way AWS was setup and the outdated instructions. I wasn’t able to properly access the material since the location of the lab had changed, but this was resolved once I found the new location and the lab had no problems afterwards.

**Application/Relevance:**

The first part of the lab taught me some fundamentals on PyTorch and the use of tensors. Since PyTorch is a popular program and uses python as a base, it’s able to work well with the development and training of machine models for various purposes. This lab’s knowledge can be applied to any future use of PyTorch operations.

**Code/Experimentation:**

**Lab 02: Creating a Multilayer Perceptron and Dropout Layers**

**Key Learnings:**

In this lab I focused on training and defining a single dense-layer neural network model, and about the importance of dropout layers and how to add them.

One of the key learnings I kept were the uses of dropout layers, which basically drop a random percent of neurons/nodes/data within the hidden layer’s input. These help prevent overfitting of data which causes the model to only recognize the data it was trained on, rather than being able to make more diverse outputs and predictions.

The other main point I learned was about Multilayer Perceptrons (MLP), which is a simple type of neural network with multiple connected input neurons. The three necessary layers within an MLP being the input, hidden, and output layer.

Other than these, the lab mainly just went over another neural network model which I already felt pretty familiar with.

**Insights/Understanding:**

I only gained a few insights from this lab, one of them being on the importance of making sure that good data isn’t overcrowded or has too many values. Even if a dataset has a lot of information, without the uses of regularization or cutting out certain portions, it can become overfit and trained to predict something specific. I like to connect this to how we think like most concepts within neural networks, if a person is told that things are a certain way, and only see these as truth then it’ll be hard for us to recognize or accept things are different. One main example I can think of this is with an office chair, if all I’m trained on and see in my lifetime are office chairs, then I’m left with a folding chair and told it’s a chair,

Another minor insight I gained although isn’t too important is how visualizing accuracy and loss of data in graphs makes it a bit easier to comprehend. Instead of just having a value on how accurate something is, I can see the performance over a set of time which might help me recognize faults later on.

**Challenges:**

The lab wasn’t too difficult or gave me much of a challenge, although the questions asked about each of the graphs made me think about challenges I might face in the future. When comparing the two graphs before, and after the dropout layer I saw that the accuracy increased after the dropout was added. With how I have to tweak the settings for epochs, or learning rate I might not recognize that I have too much data. Without adding a dropout layer or deploying some other measure against overfitting I can see how people can be stuck with models that don’t have an obvious issue of overfitting data.

**Application/Relevance:**

I think I’m able to use my insights up to this point to help me understand better methods and the way neural networks are trained. With how easy it is to compare neural networks to brains, since they’re structured like one I can apply a lot of concepts on what models might need or how they interpret data.

Although more relevant to the lab itself, I think including dropout layers within models in the future to properly manage data is a simple way and should be a common method to help cut parts of data. The other point being with how much better graphs visualize data and accuracy trends, this can easily convey where the model may fault and inform people that might not be able to read code very well on how models are performing.

**Code/Experimentation:**

**A graph of a curve

AI-generated content may be incorrect.A graph with a line

AI-generated content may be incorrect.Graphs Before Dropout Layer**

**A graph with a line

AI-generated content may be incorrect.A graph with a line

AI-generated content may be incorrect.Graphs After Dropout Layer**

**Lab 03: Building an End-to-End Neural Network Solution**

**Key Learnings:**

This lab was focused on working with a neural network again, but this time using text data instead of image data like the last lab. I used an animal center’s dataset, working with data like Pet ID, Name, Sex, Breed, Color, and other schema to process.

The first key point that stuck with me was performing exploratory data analysis (EDA). Where it manually laid out the data into a usable form, and split the categorical, numerical, and text features. It wasn’t obvious to me at first that each of these needed to be split up since it wasn’t just working with image data anymore, but text data which can have different uses.

Then the next key learning was working with the processing pipeline and using different methods on the three features. Having to use different methods/code for each individual feature was way more complex than the image model I worked with in the last lab.

**Insights/Understanding:**

I would say a lot of my understanding and insights came from working with EDA, Preprocessing, and the Training/Validation. Both the EDA and Preprocessing made it clear to me that a data set’s text really does need to be split into different variables since something like numerical and categorical variables would work differently. They’d need to be processed and understood in a different way, which stands out from something like training on the image datasets which I think seems like it can all run on similar processing/training.

Messing around with the NeuralNetwork() cell on the training/validation, on top of learning from the graphs in the last lab made me realize how important visuals are for identifying accuracy. Seeing a massive dip and rise back up makes me interested in why exactly the epochs end up varying so much, but I also understand more about how these models need to be retrained and ran over and over to reach a good result and can’t just be done once.

**Challenges:**

The first challenge I faced was some incompatible packages with the current version of pandas, which caused an error on the first run but strangely it fixed itself just by reloading the notebook. But I’d say the main challenge came from working on creating the multilayer perceptron, and getting it to a higher accuracy.

The first attempt kept giving me errors and I couldn’t figure out why it wasn’t working, but I got help making a structure from class which I ended up using. It only could reach an accuracy of around 60% which I felt like I could improve, so I started messing with some of the variables until I got them to a better point. I got the accuracy up to 83%, which I felt was good enough for just working on this lab model, although I don’t think when I make an actual model the accuracy should drop below 90%.

**Application/Relevance:**

This lab I think gave some of the most relevant information to use out of the three, although all of them built on top of each other. I know I can apply the knowledge I gained from working with text data, and especially how to split the data like numerical or categorial from one another. With my previous experience of models it was just focused on processing blocks of text or language, but working with text as data is a completely different process.

Working on optimizing the multilayer perceptron near the end of the lab also gave me some experience I can apply to working with actual models in the future. At the very least I know how to put models together and optimize them because of the lab, the one thing I think I’d struggle on is just making them from scratch, but with the available tools and pre-built models I’m not sure I’d need to code them from nothing.

**Code/Experimentation:**

A screenshot of a computer

AI-generated content may be incorrect.

**A screen shot of a computer code

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

**References:**

AWS Academy Module 1: Lesson 1 – 4 [Course Modules: AWS Academy Application of Deep Learning to Text and Image Data](https://awsacademy.instructure.com/courses/107351/modules)