**Reflective Journal**

**L05 Reflective Journal: AWS Machine Learning University Module 1 Labs (Labs 1–4)**

**Name:** Parnal Sinha

**Course:** ITAI 22376

**Date:** 08/05/24

**Lab Module:** Module 1 — Neural Networks

**Learning Insights**

Throughout Labs 1 to 4 of Module 1, I progressively built my understanding of core neural network concepts, implementation practices, and model evaluation techniques using PyTorch. The labs were carefully scaffolded to transition from basic deep learning fundamentals to designing and training a convolutional neural network (CNN) for image classification.

The early labs emphasized working with data structures like tensors and building fully connected neural networks (multilayer perceptrons, or MLPs). Later labs introduced key regularization techniques (such as dropout) and optimization strategies (including Xavier initialization), reinforcing the practicalities of stable training and model generalization. The culmination of these concepts was the creation and training of a CNN capable of classifying handwritten digit images from the MNIST dataset.

One of the most impactful learning experiences was understanding how architectural choices — such as layer type, activation functions, and feature extraction techniques — affect model performance. Additionally, integrating GPU acceleration and debugging device compatibility issues provided valuable hands-on exposure to real-world deep learning workflows.

**Challenges and Struggles**

A recurring challenge was ensuring that data and models operated consistently across CPU and GPU devices. In both Lab 3 and Lab 4, I encountered device mismatch errors during training, particularly when moving from fully connected layers to convolutional layers. These challenges required careful debugging and the use of .to(device) statements to synchronize data and model locations.

Hyperparameter tuning also posed challenges. Selecting optimal learning rates, batch sizes, dropout rates, and model layer sizes involved iterative experimentation. Each lab presented opportunities to refine these skills, although time constraints limited extensive hyperparameter searches.

Additionally, adapting to the dynamic nature of PyTorch — particularly managing model definitions, optimizers, and loss functions across multiple labs — required attention to detail and cross-referencing documentation regularly.

**Personal Growth**

These labs significantly advanced my understanding of modern NN and CNN espieically. I transitioned from viewing machine learning as a set of abstract concepts to applying concrete solutions in practical tasks like image classification, filter modeling and transfer learning.

One area of personal growth was learning how to balance theoretical understanding with practical engineering. Instead of focusing solely on coding, I learned to ask critical questions:

* Why choose one preprocessing method over another?
* What trade-offs come with deeper models?
* How can pre-trained models accelerate development?

These reflections also improved my patience and perseverance. Not every experiment produced immediate success, but each debugging session enhanced my problem-solving skills to now be able to go through these modules at a much more streamlined fashion. The skills I gained are directly applicable to my academic goals and future professional projects, particularly those involving AI-driven text analysis, sentiment detection, or conversational AI design.

**Critical Reflection**

If I could repeat these labs, I would dedicate more time to experimenting with different neural network architectures and advanced optimization techniques. While I successfully completed the labs as prescribed, deeper experimentation — such as testing additional activation functions or adjusting convolutional parameters — would have provided richer insights into model design trade-offs.

Looking ahead, I aim to explore more advanced neural network architectures, including transfer learning models and custom CNN variations for more complex image datasets. The foundational knowledge gained from these labs will be instrumental in those future endeavors.

These labs also highlighted the broader significance of deep learning workflows, from data preprocessing to model evaluation, which mirrors the industry-standard practices in machine learning and AI development.

**Summary (Lab 1–4)**

**Lab 1: Getting Started with PyTorch**

1. Created and manipulated tensors, including indexing, slicing, and reshaping.
2. Practiced automatic differentiation and verified computation graphs.
3. Explored PyTorch’s dynamic computation graph capabilities.
4. Experimented with moving data to GPU for accelerated computation.

**Lab 2: Creating a Multilayer Perceptron and Dropout Layers**

1. Constructed a multilayer perceptron (MLP) using nn.Sequential.
2. Incorporated dropout layers to mitigate overfitting and improve generalization.
3. Implemented Xavier initialization for stable weight distribution.
4. Trained the network using stochastic gradient descent (SGD) and observed training/validation loss trends.
5. Resolved device management challenges (CPU vs. GPU).

**Lab 3: Building an End-to-End Neural Network Solution**

* Preprocessed and cleaned a real-world tabular dataset (Austin Animal Center pet adoption dataset).
* Designed an MLP with multiple hidden layers and dropout.
* Applied Xavier initialization and experimented with optimizers and learning rates.
* Addressed device mismatch errors during training.
* Evaluated model performance using standard classification metrics.

**Lab 4: Introducing Convolutional Neural Networks (CNNs)**

* Built a convolutional neural network (CNN) to classify handwritten digits from the MNIST dataset.
* Designed the CNN architecture with a Conv2D layer, ReLU activation, Flatten layer, fully connected layers, and a Softmax output.
* Experimented with hyperparameters including kernel size, number of filters, and learning rate.
* Applied Xavier initialization to improve weight distribution and model convergence.
* Addressed and resolved device mismatch errors (CPU vs. GPU), ensuring smooth model training and evaluation.
* Evaluated model performance using classification metrics and analyzed training/validation loss progression across epochs.

**References**

1. AWS Machine Learning University Module 1 Labs and Documentation
2. PyTorch Official Documentation
3. Scikit-learn Documentation
4. Course Lecture Materials