Jade Sanchez

2376 ITAI

2/12/2025

Reflective Journal

Lab 01

Summary of Key Learnings

The first lab provided essential knowledge about the PyTorch deep learning framework while instructing users about tensor operations along with automatic differentiation techniques. The lesson taught me to construct tensors then index their segments and apply arithmetical operations between tensors. Through exercise instructions the lab taught students how to reshape tensors between different Python objects and thus proved PyTorch's capabilities for flexibility. I investigated automatic differentiation since it serves as the primary building block of neural network training.

Insights & Understanding

The fundamental tensor operations of PyTorch as well as their application to deep learning formed the core learning of this laboratory work. Through this learning experience I figured out that GPUs within PyTorch systems perform calculations much faster than CPUs when performing operations. Automatic differentiation with backpropagation became more understandable when I studied the neural network weight optimization process. The provides an easy way for users to compute gradients through its requires_grad functionality.

Challenges Encountered

The process of learning multidimensional slicing and verifying suitable tensor shapes for matrix multiplication operations posed significant challenges to me. Studying reshaped arrays in tensor examples alongside practice time enabled me to understand these concepts more effectively. Identifying how gradients are stored within PyTorch required considerable effort to grasp. I solved the issue through testing the .requires_grad syntax while utilizing debugging methods for result examination.

Application & Relevance

All skills learned during the practical will be vital for conducting neural network training tasks which involve image categorization and natural language processing for AI applications. Effective matrix multiplication combined with gradient calculation represents the basic

fundamental requirement for building learning models with data.

Code and Experimentation
The experiments needed the use of implicit dimension (-1) when reshaping tensors to test random initialization effects on tensor content values. I transformed a tensor data type to observe the method through which PyTorch handles different data types. The conducted experiments proved my previous understanding of tensor properties which enhance computational operations.

Code and Experimentation
The tensor experiments needed -1 as an implicit dimension parameter for modifications and required testing initial value variations on their constituent data. The tensor dtype conversion process let me study how PyTorch handles different data types. My practical work validated my understanding of tensor functionality which drives computational operations.

Lab 02

Summary of Key Learnings

The exercise involved building a neural network through PyTorch implementation which processed data obtained from make_circles. The process involved making a dataset followed by defining the neural network with PyTorch Sequential then applying Merocratic initialization and training with SGD using binary cross-entropy loss. My training demonstrated that PyTorch provides basic neural network features enriched with total manipulation capabilities for each component and variable.

Insights & Understanding

Users gain significant knowledge from the modular nature of the PyTorch API structure. Model development became easier when we created the model definition with Sequential while utilizing built-in Sigmoid activation functions. The software combines gradient computation (loss.backward()) with optimization steps as a unified operation. The single-layer model with its basic dataset produced noticeable improvements through each epoch according to the loss function.

Challenges Encountered

Weight initialization stood as the main obstacle to grasp during the learning process. The application of Xavier initialization prevented the model from producing inconsistent results by either vanishing or exploding gradient effects. The most challenging aspect involved fixing tensor shape mismatches specifically during the conversion of data into PyTorch tensors. To solve this problem, I checked each shape to guarantee uniformity between all operations.

Application & Relevance

A laboratory experiment showed neural network foundation by creating practical two-class systems for categorization purposes. Binary classification fundamentals apply directly to the development of current-day sophisticated processing operations in AI such as image identification and natural language processing. I will implement superior network designs together with optimized approaches using the insights I obtained from this research for my future work.

Code and Experimentation
I ran tests that involved various learning rate values combined with multiple epoch counts. The initial training stability was negatively affected when the learning rate was set to 0.1 because it introduced variations in the loss curve. The learning rate needed adjustment to 0.01 until training gained stability at which point it produced consistent convergence. The implementation of ReLU activation on a hidden layer in the model produced slightly better accuracy during training although the training process became more demanding in terms of epochs.

Lab 03

Summary of Key Learnings

The experiment conducted a complete neural network development to forecast adoption destiny in pets through analysis of Austin Animal Center information. The project work with data preprocessing then feature engineering before finishing with neural network construction through the use of PyTorch for training purposes. I mastered pipeline creation to prepare all features through one-hot encoding and text vectorization of numerical and categorical and textual elements. The work involved me in hands-on training of neural networks while validating these networks through repeated architectural and parameters adjustments to improve their effectiveness.

Insights & Understanding

Working with text data in neural networks delivered multiple comprehensions to me. Through vectorization process text features become compatible with network systems because they can achieve numeric representation. The model's accuracy heavily depends on proper preprocessing for every feature type irrespective of text data because inconsistent data can result in significant performance reduction. The pipelines revealed a valuable method to process features since they managed multiple data types concurrently within a systemized framework.

Challenges Encountered

The main technical obstacle during text vectorization and model training operations involved coping with excessive computational requirements. Optimizing the text processing of the large dataset required setting limits on the maximum feature number in the CountVectorizer module. Building the neural network proved challenging because I experienced problems when deciding between the suitable parameters that ultimately resulted in subpar model performance. My experiments with activation functions and dropout layers together with learning rate adjustments increased the model performance step by step.

Application & Relevance

The laboratory experiment showed the capacity of neural networks to process varied datasets which integrate structured and unstructured information. The practical knowledge about creating complete pipelines as well as model optimization techniques provides direct value for operational AI systems which use predictive modeling and natural language processing. The skills learned during this lab are essential for developing future classification projects that handle data combinations of different types.

Code and Experimentation

The performance assessment required modifications to both hidden layers.count and neuron.count of the neural network throughout the experimental stage. The model gained increased ability to track complex connections as the number of added layers increased yet the threat of overfitting events simultaneously grew in severity. The darüber correction of dropout in combination with the adjustment of learning rates permitted this issue to be resolved effectively.

I found that ReLU activation function delivered optimal results compared to Sigmoid and Tanh activation functions during testing.

Conclusion

The three lab sessions built my competency with PyTorch deep learning applications starting from tensor basics and advancing to network development and optimization for processing genuine dataset information. Students gained full understanding of PyTorch abilities through practical demonstrations which showed its adaptable structure combined with its efficient computing power alongside its usefulness across various Artificial Intelligence tasks.

The experience required me to deal with obstacles alongside chances to improve my technical abilities. My understanding of neural network fundamentals grew stronger through managing tensors and gradients and weights and hyperparameters because this required me to recognize key design choices in neural network development. Solving these difficulties enhanced my self-assurance and delivered problem-solving tools which I can use with complex AI tasks.

The practical exercises revealed the need to match models with operational performance and execution speed requirements for effective real-world use. Through work on binary classification and mixed data type processing I became able to adapt neural networks for specific needs making this learning experience directly applicable to future artificial intelligence and machine learning work.