Data Mining Homework - 2

Classification of Diabetes and Human Activity Dataset on PyTorch and Tensorflow

Link: https://github.com/codewithhritik/Classification-PyTorch-TensorFlow

Introduction

The homework aims to apply machine learning techniques to two distinct datasets using TensorFlow and PyTorch. The goal is to perform classification tasks that predict categorical outcomes based on varying input features. This analysis will not only highlight the capabilities of each framework but also compare how each handles different types of data, with an emphasis on fine-tuning and optimizing the models to achieve the best possible accuracy and efficiency.

Selected Option

I have selected Option 1: Choose two datasets from our dataset list.

Selected Platforms

For this homework, I have selected TensorFlow and PyTorch due to their capabilities in handling large datasets and complex neural network architectures. Both platforms are widely recognized for their flexibility and performance in various machine learning tasks.

Datasets

1. Diabetes Health Indicators Dataset: This dataset comprises demographic, laboratory test results, and survey data from individuals, focusing on their health and diabetes status. The 35 features include age, gender, body mass index, blood pressure readings, glucose levels, and lifestyle factors such as smoking and physical activity
Link: https://archive.ics.uci.edu/dataset/891/cdc+diabetes+health+indicators

2. Human Activity Recognition 70+ (HAR70+): This dataset includes accelerometer data from 18 older adults aged between 70 and 95 years. The participants wore sensors on

their right thigh and lower back, recording their movements during various activities within a semi-structured environment. The dataset provides raw accelerometer readings and annotates periods of activity, which are essential for developing models that can accurately classify movement types in real-time applications.
Link: https://archive.ics.uci.edu/dataset/780/har70

Selected Task

The task selected for this homework is **classification**, where:

The HAR70+ dataset involves classifying types of physical activities.
The Diabetes dataset involves classifying individuals according to their diabetes status.

Selected Solutions

Diabetes Health Indicators Dataset:

Neural Network Architecture:

TensorFlow and PyTorch Models: These used sMLP architectures but were specifically tuned to handle binary classification. The structure was designed to work with the dataset's diverse set of features, including demographic, physiological, and lifestyle indicators.

Layers and Neurons: Configured with a series of dense layers, each followed by dropout and batch normalization layers. The design was aimed at parsing through the intricate relationships between various health indicators and their impact on diabetes.

Activation Functions: ReLU was used in hidden layers to ensure efficient learning of non-linear complexities in the data. The output layer utilized a sigmoid function, which is ideal for binary classification, providing a probability score reflecting the likelihood of diabetes presence.

Regularization Techniques:

Dropout and Batch Normalization: These techniques were applied to prevent overfitting and to ensure that the model generalizes well on unseen data.

Human Activity Recognition 70+ (HAR70+) Dataset:

Neural Network Architecture:

TensorFlow and PyTorch Models: Both frameworks used a multi-layer perceptron (MLP) architecture, which is effective for structured, tabular data.

Layers and Neurons: The networks consisted of several dense layers. Initial layers had a higher number of neurons to capture complex patterns and relationships in the data, with subsequent layers tapering down to focus on feature refinement and reduction.

Activation Functions: ReLU (Rectified Linear Unit) was used in all hidden layers due to its effectiveness in adding non-linearity to the model, helping to learn complex patterns without the vanishing gradient problem. The output layer used a softmax function, suitable for multi-class classification tasks, to predict the type of activity being performed.

Regularization Techniques:

Dropout: Implemented to reduce overfitting by randomly omitting subsets of features at each iteration during the training phase. This technique ensures that the model does not rely too heavily on any single or a group of features.

Batch Normalization: Used to stabilize and accelerate the training process by normalizing the inputs of each layer. It helps in maintaining a mean output close to 0 and an output standard deviation close to 1.

Modifications

Diabetes Health Indicators Dataset:

Categorical Encoding: Due to the presence of numerous categorical variables, such as demographic and lifestyle choices, appropriate encoding methods were necessary. One-hot encoding was used to transform these categorical variables into a format suitable for neural network processing, ensuring that the model could interpret these features without introducing ordinal biases.

Human Activity Recognition 70+ (HAR70+) Dataset:

Preprocessing Adjustments: Before training, the dataset underwent preprocessing to ensure the models could effectively learn from it. Given the nature of physical activity data, windowing techniques were applied to segment the continuous data stream into manageable parts, each representing a specific activity type.

Major Steps

For the Diabetes Health Indicators Dataset:

1. Load the dataset and split it into features (X) and labels (y)
2. Fill missing values in X using the mean of each column
3. One-hot encode categorical features in X
4. Scale numerical features in X using StandardScaler
5. Split X and y into train, validation, and test sets (80% train, 20% test, 25% of train for validation)
6. Convert data to PyTorch tensors and create DataLoaders

TensorFlow:
7. Define the model architecture (4 layers - 256, 128, 64, 1 neuron with ReLU activations)
8. Compile the model (optimizer=Adam(lr=0.0001), loss=binary_crossentropy, metrics=[accuracy])
9. Train the model for 30 epochs with batch_size=1024 and 20% validation split
10. Evaluate on test set and print test loss and accuracy
11. Apply a custom threshold of 0.65 and calculate accuracy
12. Plot ROC, Precision-Recall curve, predicted probability histogram
13. Plot training vs validation loss
14. Plot test loss for every 20th batch

PyTorch:
7. Define model architecture (6 layers - 256, 128, 64, 32 neurons with ReLU/BatchNorm/Dropout)
8. Set loss function as BCELoss and optimizer as Adam(lr=0.001)
9. Train for 30 epochs
10. Evaluate on test set and print test loss and accuracy
11. Apply custom threshold of 0.65 and calculate accuracy
12. Plot ROC, Precision-Recall curve, predicted probability histogram
13. Plot training vs validation loss

14. Plot test loss for every 20th batch

For the Human Activity Recognition (HAR70+) Dataset:

1. Load the dataset
2. Convert timestamp to datetime and adjust labels to start from 0
3. Separate features (X) and labels (y)
4. Scale features using StandardScaler
5. Split data into train, validation (10%), and test (20%) sets

TensorFlow:
6. Define model (2 Dense layers with 64 neurons, Dropout, output Softmax for 8 classes)
7. Compile model (optimizer=adam, loss=sparse_categorical_crossentropy, metrics=[accuracy])
8. Train with EarlyStopping callback for 20 epochs with 10% validation split
9. Evaluate on test set and print test accuracy
10. Save the model
11. Plot training vs validation loss
12. Plot test loss for every 20th batch (up to 800 batches)

PyTorch:
6. Define model (3 layer MLP - 64, 64, num_classes neurons with ReLU/Dropout)
7. Set loss as CrossEntropyLoss and optimizer as Adam(lr=0.001)
8. Train for 20 epochs
9. Evaluate on test set and print test loss and accuracy
10. Plot training vs validation loss
11. Plot test loss for every 20th batch

Performance Comparison

The performance comparison involves evaluating PyTorch and TensorFlow on datasets. To ensure fairness, I used the same models, activation functions, and optimizers for both training and testing the model. Further, I created graphs of loss and class accuracy to assist in the comparison.

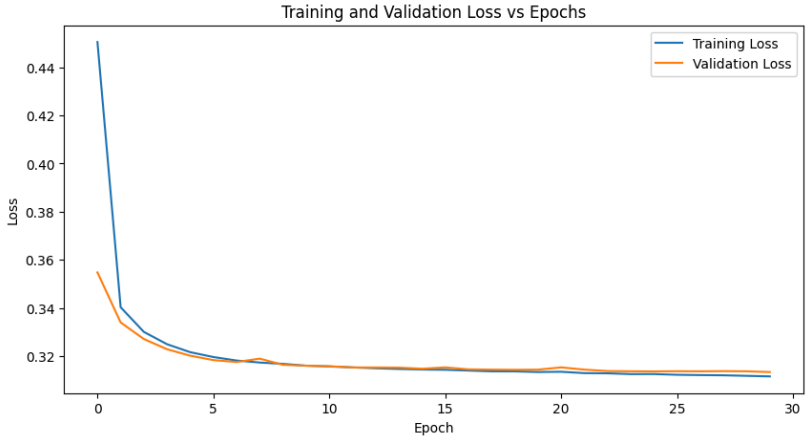
Diabetes Health Indicators Dataset:

TensorFlow:

Test Loss: 0.3086034655570984
Test Accuracy: 0.8681212663650513

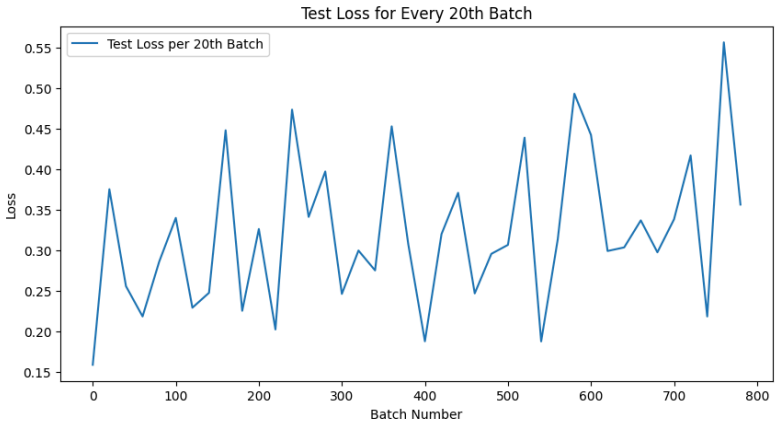
PyTorch:

Test Loss: 0.3109
Test Accuracy: 0.8667

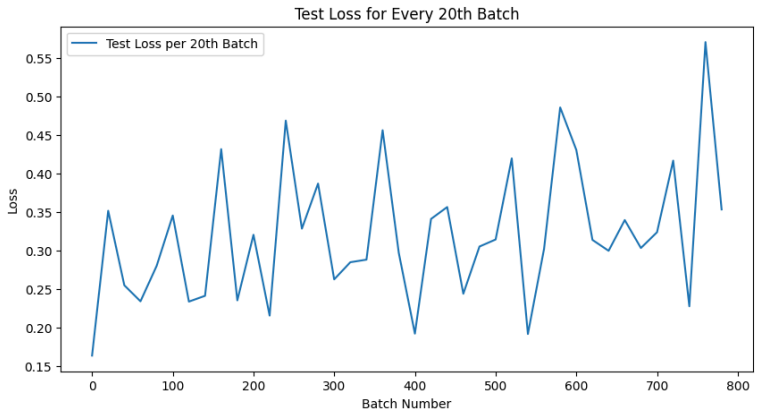


Tensorflow: Losses

PyTorch: Losses



Tensorflow: Test Losses



PyTorch: Test Losses

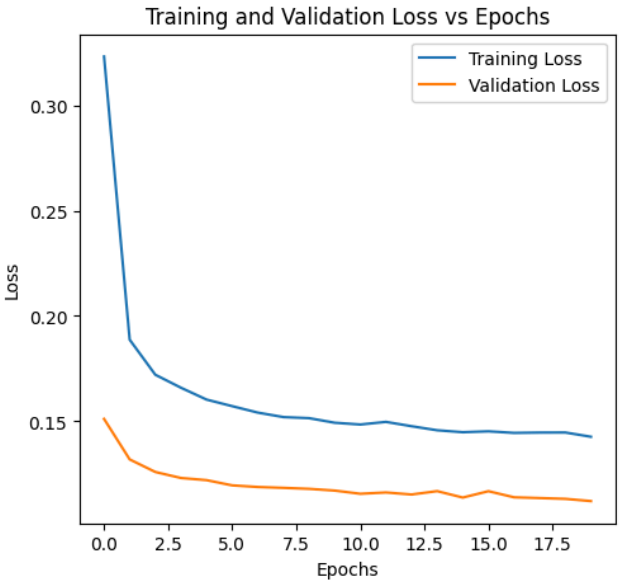
Human Activity Recognition (HAR70+) Dataset:

Tensorflow:

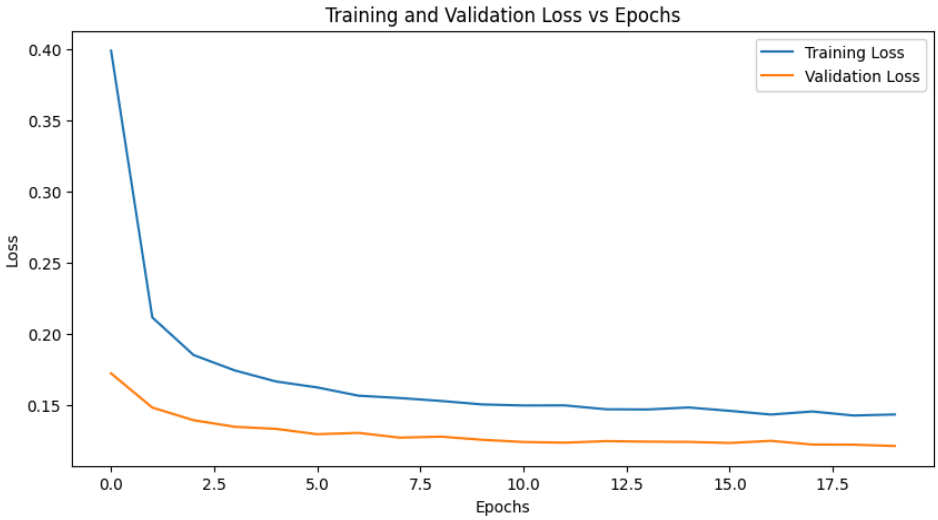
Validation loss after 20th Epoch: 0.1122
Test Accuracy: 96.52%

PyTorch:

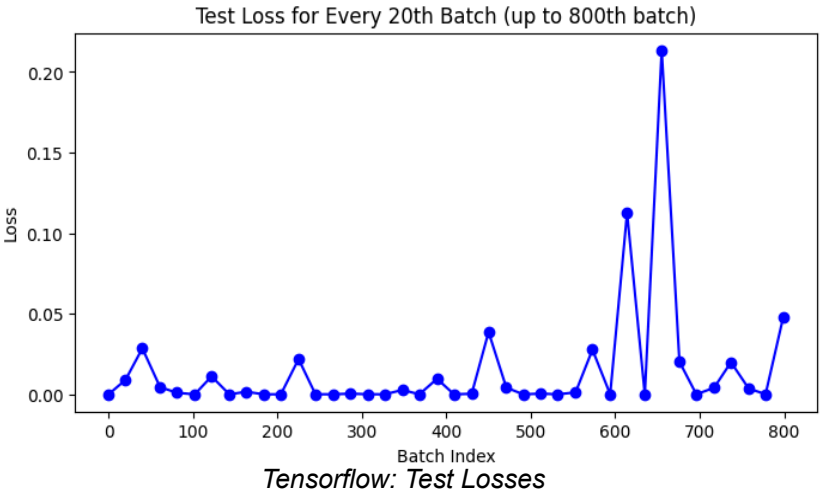
Validation Loss after 20th Epoch: 0.1215
Test Accuracy: 96.54%



Tensorflow: Losses



PyTorch: Losses



Tensorflow: Test Losses

PyTorch: Test Losses

References

[1] "Human Activity Recognition 70+ Dataset," University of California Irvine, [Online]. Available: https://archive.ics.uci.edu/dataset/780/har70.

[2] A. Tanwar, "Adult Subjects 70-95 Years Activity Recognition," Kaggle, [Online]. Available: https://www.kaggle.com/datasets/anshtanwar/adult-subjects-70-95-years-activity-recognition/data.

[3] "Behavioral Risk Factor Surveillance System (BRFSS)," University of California Irvine, [Online]. Available: https://archive.ics.uci.edu/dataset/891/cdc+diabetes+health+indicators.

[4] C. Sahu, "Diabetes Dataset EDA & Prediction," Kaggle, [Online]. Available: https://www.kaggle.com/code/chanchal24/diabetes-dataset-eda-prediction.