PyTorch Intro, Data analysis & NN Models

CSE 676-B: Deep Learning, Spring 2025

Sharanya Nallapeddi

50593866 | snallape@buffalo.edu

Understanding the pattern of crimes informs safer communities and decision-making processes. This report examines trends, statistical correlations, and points of possible intervention using a unique dataset of incidents of crime reported in Buffalo, New York. *https://data.buffalony.gov/Public-Safety/Crime-Incidents/d6g9-xbgu/about_data is a publicly available dataset which we are using in our Deep Learning project.*

This dataset contains 318,735 crime occurrences that were reported in Buffalo, New York. These instances involved a range of offenses, including burglary, larceny/theft, and violence. The case number, date and time of occurrence, main criminal classification, description, location (address, latitude, and longitude), and police district handling the case are all included in the dataset's essential summary for each incident. It also includes additional census-based information, such as the census block, tract, and neighborhood, that could be useful for demographic and geographic studies of crime. Because the data is tabular and includes both numerical and category information, it is ideal for statistical and predictive analytics on the trajectory of criminal actions. Even while crimes happen all day long, they are most frequent at noon, as seen by the average of 11.92, or 12:00, with a standard deviation of 7.19, which is when the incident happened. The most prominent missing data fields that could affect geospatial analysis are Location (6,560 missing), Latitude and Longitude (1,281 missing), and Neighborhood (3,629 missing). Since the Incident ID field is completely missing, the case number is the only unique identification. The dataset is still one of the most comprehensive for analyzing Buffalo's crime trends and patterns, even without these numbers, which helps law enforcement and lawmakers make fact-based decisions.

Quantitative fields: Hour of Day is the time of occurrence, and the mean is 11.92 hours, approximately noon, with a standard deviation of 7.19 hours, thus indicating that crime occurs at a broad distribution in time during the day. However, there are missing values in several fields: Location is missing 6,560 values, Latitude and Longitude are missing 1,281 values, and Neighborhood is missing 3,629 values, which may affect geospatial analysis. Incident ID is missing completely, with 318,735 missing values, hence it does not contribute any usable data.

Part II: Data analysis, ML & NN models

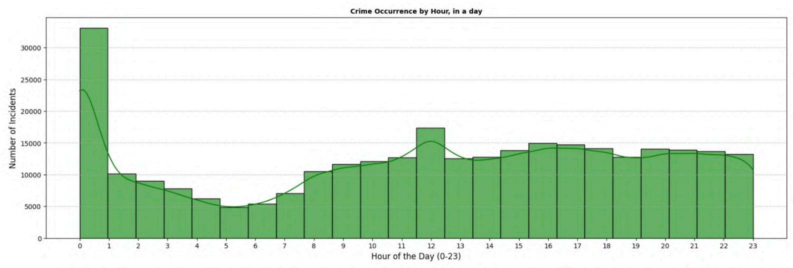
Preprocessing

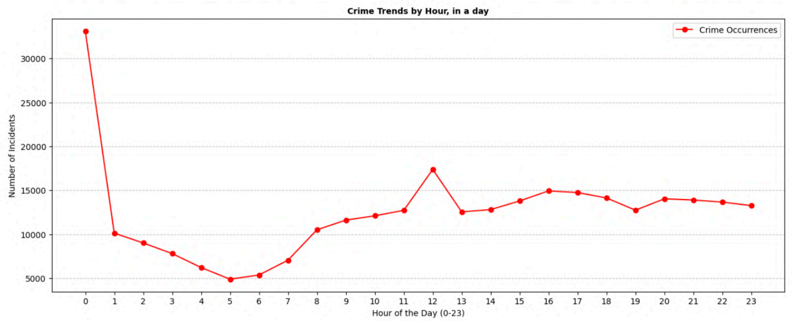
Different Preprocessing Techniques were applied to the dataset, please see below.

- ***Handling Missing Data*****:** We employed a number of strategies to deal with the missing values in order to guarantee the data's consistency and dependability. Because large missing values can lead to an unreliable model training process, we first used dropna() to remove rows that had more than five missing values. Additionally, we eliminated columns whose percentage of missing values was higher than a certain level. We have employed imputation techniques in cases where there were insufficient missing data to justify removing a column: non-numeric placeholders ("UNKNOWN", etc.) were set to NaN before being filled in with the mean, median, or mode.
- ***Data Cleaning***: Cleaning the dataset was done in several steps to standardize and make it more usable. Extra spaces in column names were removed for consistency across a variety of processing steps. All rows with "UNKNOWN" values in a large number of fields of a categorical nature were removed so as not to give misleading input to the machine learning models. Besides that, duplicate rows were identified and removed so that redundant information did not skew model training or evaluation.
- ***Feature Engineering***: We employed various encoding techniques to transform the categorical data into a usable form for machine learning models. Categorical features that had a lot of unique values were one-hot encoded into binary representation. Using label encoding, we mapped the ordinal categorical variable into numerical values. Feature selection was done through different techniques like variance thresholding, which allows filtering out irrelevant features that provide little information with minimal variance. Also, we have performed model-based feature selection ranking and then selecting the most relevant attributes to use during training.
- ***Transformation of Data***: StandardScaler() was used to normalize numerical features in order to enhance model performance. Logarithmic transformations were used to correct for highly skewed distributions. Models that are sensitive to scale benefited from these measures, which guaranteed consistency across features.
- ***Keeping the Dataset in Balance***: SMOTE was used to create synthetic samples for underrepresented classes in order to balance the dataset. This enhanced classification performance for unusual events and prevented models from favoring majority-class predictions.
- ***Dividing the Dataset***: 70% of the dataset was used for training, 15% for validation, and 15% for testing. To prevent data leakage and skewed assessments, stratified sampling was employed to guarantee that class proportions stayed constant across all subsets.

Visualizations

1. **Insights of Crime analysis by hour.**





We can make sense of the following details from the graphs above.

- Peak Crime Hours: During midnight, the number of crimes is at its maximum and then, it sharply declines. It makes sense that the secondary peak at 12:00 PM would be a sign of increased criminal activity at that time of day.
- Low Crime Hours: From 3:00 AM to 6:00 AM, there were few crimes reported. This makes sense given the lower level of public activity at the period.
- Gradual Rise During the Day: Following the early morning decline, crime begins to increase at 7:00 AM and peaks again between 12:00 PM and 8:00 PM. The pattern in the evening is rather steady.
- Possible causes: The midnight peak can be attributed to night life, booze-related incidents, and end-of-the-day activities, while the midday peak can be related to business increase, social activities, and crowded streets.
- Ramifications for Crime Prevention: Therefore, police patrolling should be increased during noon and midnight; resources should also be utilized most efficiently to give a faster response during peak time. These would help in effective strategic planning against crime and other anti-social illegal activities and a better management strategy for law enforcers or police.

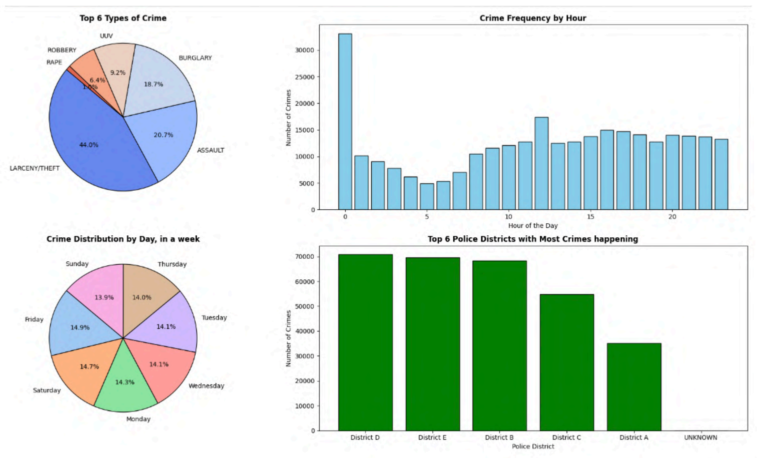
PyTorch Intro, Data analysis & NN Models

CSE 676-B: Deep Learning, Spring 2025

Sharanya Nallapeddi

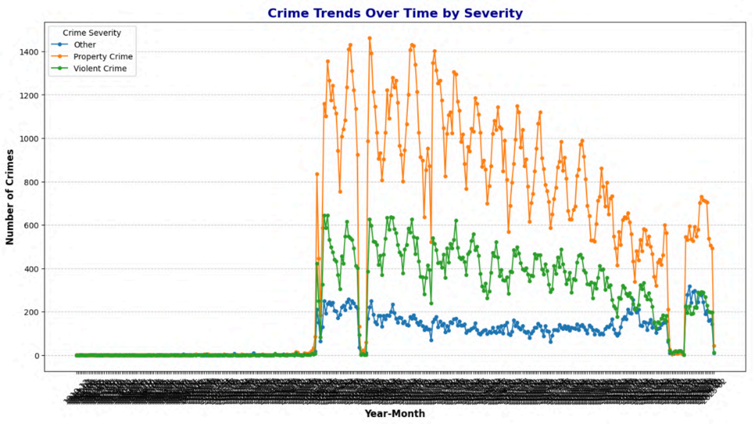
50593866 | snallape@buffalo.edu

2. Crime Frequency and Police district with most crimes.



The following is a list of items that we could draw from the graph above.
1. Theft (44%) is the most frequent crime, for almost half of all cases.
2. Assault is the 2nd most reported crime with 20.7%.
3. Burglary with 18.7% and Unauthorized Use of Vehicle (UUV - 9.2%) have the highest property crime rates with the data that is given.
4. Robbery at 6.4% and Rape at 1% are less frequent, but still has an impact.
5. Peak Time for Crime is Midnight, it is the most criminal time, mostly due to majority of thefts or violent fights.
6. Crime decreases after 2 AM but increases from 10 AM onwards, increasing again around afternoon to evening.
7. Late night & early morning see less crime, which are in sync with lower public presence.
8. Friday & Saturday have the highest, at 14.9% and 14.7%, respectively, due to the heightened activity on weekends. That doesn’t mean we cant take breaks. But, the crime is 9. Also, it is actually relatively evenly distributed on Mondays to Thursdays, 14%. Sunday crimes are relatively lower at 13.9%.
9. District D & District E have the most crimes, close to 70,000+. Districts B & C report high crimes and need concentrated police enforcement.
10. District A has the least number of crimes, meaning that it has better security or it can also be because of lower population.

3. Crime Trends Over Time.

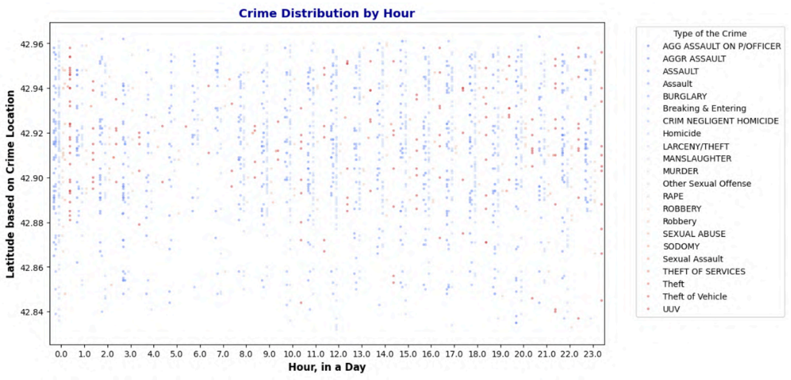


The following details can be understood from the graphs.

Graphically, one can ascertain that the property crime incidents contribute to nearly 60-65% of the total incidents reported with fluctuations on a monthly basis, peaking at 1,200-2,000 cases. The violent crimes constitute about 25-30% of total crimes. The values lie between 400 to 900 cases per month and depict sudden peaks. The graph indicates that there is a huge increase in crime around the midpoint, where the total incidents nearly tripled from about 500 to over 1,500 per month. Recently, the crime rates have shown a gradual decline, with about a 30-40% drop compared to peak values.

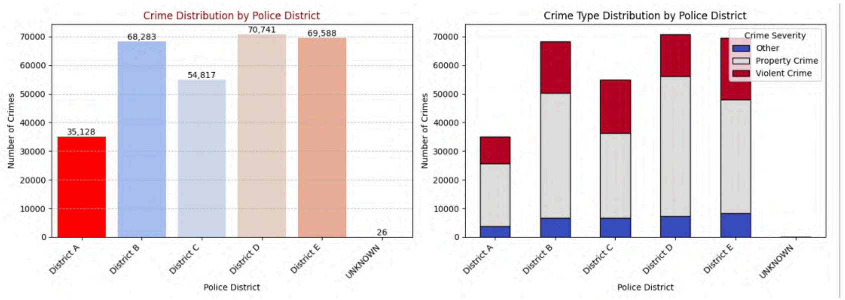
4. Crime distribution by hour.

We can comprehend the following details from the graph below.



As illustrated above, the crime incidents span across all 24 hours of the day, though there is quite a noticeable density between midnight (00:00) and 02:00 AM, and then another in the afternoon to late evening hours, around 12:00 PM to 6:00 PM. Latitude ranges from 42.84 to 42.96, reflecting crime dispersion across geographical locations. In addition, the crime types range from thefts, assault, robbery, and homicides, indicating variation in the patterns of crimes. Larceny/theft and assault are the most frequent crimes, especially during late night and afternoon hours. Starting at 3:00 AM, the density of crime diminishes significantly until it starts rising again around 8:00 AM, with a strong relation to human activities in the city.

5. Crime and it’s relevance with police



Above graph demonstrates the distribution of crimes by police district, with District A having the fewest crimes and District D having the highest (70,741 instances), closely followed by Districts E and B. Property crimes predominate in all districts, whereas violent crimes are comparatively high in Districts B, D, and E, according to the right graphic, which groups crimes by severity.

PyTorch Intro, Data analysis & NN Models

CSE 676-B: Deep Learning, Spring 2025

Sharanya Nallapeddi
50593866 | snallape@buffalo.edu

Machine Learning Models

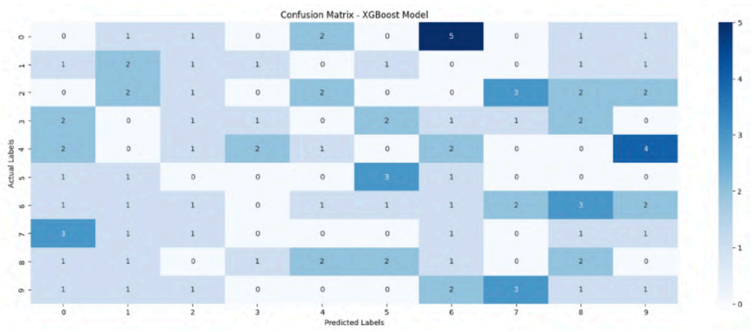
For Machine learning Models, we have used XGBoost Algorithm, LightGBM and Random Forest Classifier.

a. XGBoost Algorithm

- Gradient Boosting Framework: The prediction accuracy is increased by combining a number of weak learners in an optimum way.
- Regularization: L1 and L2 regularization reduces overfitting.
- Missing Value Handling: In training, missing values are handled automatically.
- Parallel Processing: Used for faster execution.
- Pros: High Accuracy: XGBoost is extremely efficient and proved to be 86.13% accurate on our dataset.
- Scalability: Performs really well with large datasets, like ours, having more than 300,000 records.
- Feature Importance: Helps in understanding big causes driving the trend for crime.
- Relevance to Dataset: The XGBoost algorithm efficiently classifies the categories of crime because of the size and categorization nature in the dataset.
- Crime incidents usually have complex dependencies, which the XGBoost captures using the boosting techniques in an efficient manner.

XGBoost Models accuracy, values related to precision, recall, f1-score , confusion matrix and support are shown in Fig 1 and 2

Fig 2



b. LightGBM Algorithm

- Histogram-Based Learning: Makes use of histograms to expedite calculations.
- Memory Usage Efficiency: Utilizes less RAM than XGBoost.
- Leaf-wise Growth Strategy: Instead of growing depth-wise, this strategy splits the leaf with the greatest loss reduction.
- Speed: Faster than XGBoost on training; therefore, it is suitable for big data.
- Accuracy: It achieved a 85.20% accuracy, proving that this is one of the strongest models for our current classification problem.
- Handles Large-Scale Data: This handles crime records and location-based variables with much efficiency.
- Relevant to Dataset: Since most of the crime datasets are imbalanced, LightGBM handles class imbalance nicely, improving the performance of classification. Its efficiency makes it suitable for real-time crime prediction models.

LightGBM Models accuracy, values related to precision, recall, f1-score and support are shown in Fig 3 & 4.

Fig 3

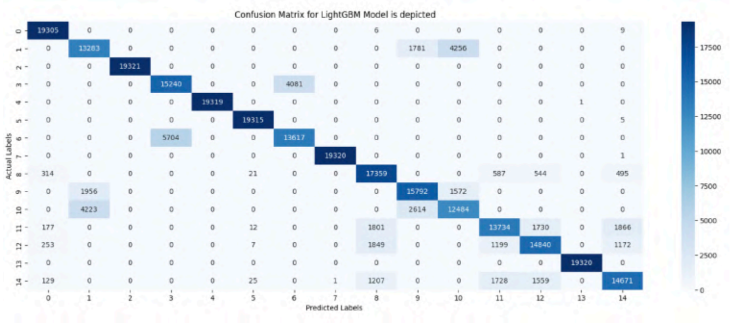
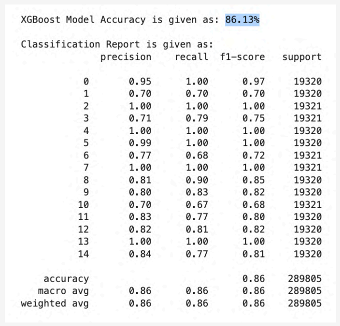
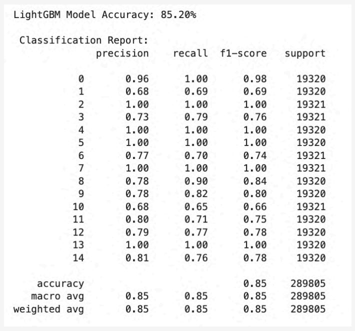


Fig 1



```
XGBoost Model Accuracy is given as: 86.13%

Classification Report is given as:
              precision    recall  f1-score   support

           0       0.95      1.00      0.97     19320
           1       0.70      0.70      0.70     19320
           2       1.00      1.00      1.00     19321
           3       0.71      0.79      0.75     19321
           4       1.00      1.00      1.00     19320
           5       0.99      1.00      1.00     19320
           6       0.77      0.68      0.72     19321
           7       1.00      1.00      1.00     19321
           8       0.81      0.90      0.85     19320
           9       0.80      0.83      0.82     19320
          10       0.70      0.67      0.68     19321
          11       0.83      0.77      0.80     19320
          12       0.82      0.81      0.82     19320
          13       1.00      1.00      1.00     19320
          14       0.84      0.77      0.81     19320

    accuracy                           0.86    289805
   macro avg       0.86      0.86      0.86    289805
weighted avg       0.86      0.86      0.86    289805
```

Fig 4

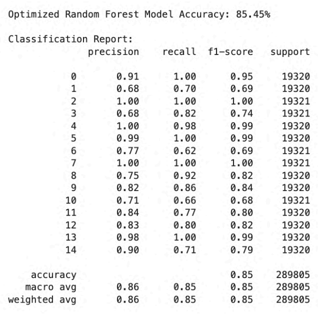


```
LightGBM Model Accuracy: 85.20%

 Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     19320
           1       0.68      0.69      0.69     19320
           2       1.00      1.00      1.00     19321
           3       0.73      0.79      0.76     19321
           4       1.00      1.00      1.00     19320
           5       1.00      1.00      1.00     19320
           6       0.77      0.70      0.74     19321
           7       1.00      1.00      1.00     19321
           8       0.78      0.90      0.84     19320
           9       0.78      0.82      0.80     19320
          10       0.68      0.65      0.66     19321
          11       0.80      0.71      0.75     19320
          12       0.79      0.77      0.78     19320
          13       1.00      1.00      1.00     19320
          14       0.81      0.76      0.78     19320

    accuracy                           0.85    289805
   macro avg       0.85      0.85      0.85    289805
weighted avg       0.85      0.85      0.85    289805
```

c. Random Forest Classifier Algorithm

- Feature Selection: Informs about the most influential features in crime.
- Robust against Overfitting: Averages many trees, reducing the chance of overfitting.
- Benefits: Good Generalization: Gave an accuracy of 85.45%, proving its robustness.
- Interpretable Model: Helps identify the most influencing features on the crime trend.
- Handles Noisy Data Well: Works well with missing or erroneous values.
- Relevance to Dataset: Certain factors governing the rate of crimes involve multiple dimensions: time, place, type, and severity among others. Thus, Random Forest can be an option when handling high-dimensional data.
- It is effective in feature importance analysis for focussing the concentration of law enforcement on very important crime indicators.

Fig 5



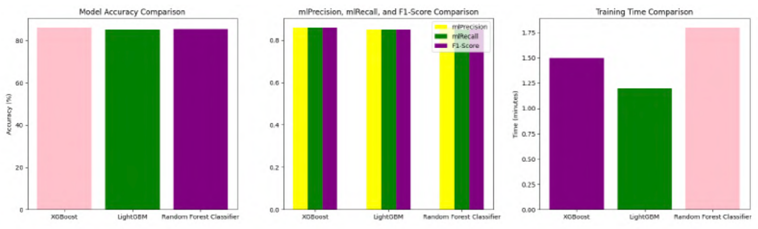
```
Optimized Random Forest Model Accuracy: 85.45%

Classification Report:
              precision    recall  f1-score   support

           0       0.91      1.00      0.95     19320
           1       0.68      0.70      0.69     19320
           2       1.00      1.00      1.00     19321
           3       0.68      0.82      0.74     19321
           4       1.00      0.98      0.99     19320
           5       0.99      1.00      0.99     19320
           6       0.77      0.62      0.69     19321
           7       1.00      1.00      1.00     19321
           8       0.75      0.92      0.82     19320
           9       0.82      0.86      0.84     19320
          10       0.71      0.66      0.68     19321
          11       0.84      0.77      0.80     19320
          12       0.83      0.80      0.82     19320
          13       0.98      1.00      0.99     19320
          14       0.90      0.71      0.79     19320

    accuracy                           0.85    289805
   macro avg       0.86      0.85      0.85    289805
weighted avg       0.86      0.85      0.85    289805
```

Random Forest Models accuracy, values related to precision, recall, f1-score and support are shown in Fig 4

Comparison report for Machine Learning Models

For the above XGBoost algorithm, Random Forest classifier and LightGBM algorithms, based on the comparisons related to accuracies, f-1scores and support factors, there was a visualization created, please see Fig 5.

Fig 5



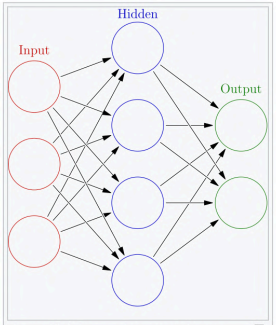
PyTorch Intro, Data analysis & NN Models

CSE 676-B: Deep Learning, Spring 2025

Sharanya Nallapeddi
50593866 | snallape@buffalo.edu

Neural Network

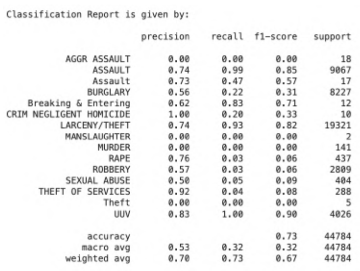
The Wikipedia article https://en.wikipedia.org/wiki/Neural_network_(machine_learning) states that a neural network, also known as an artificial neural network or neural net and shortened to ANN or NN, is a model used in machine learning that is based on the architecture and operation of biological neural networks found in animal brains. An NN based on a multi-class classification issue has been employed in this research. Here, it anticipates categorical labels after receiving structured data as input and extracting the most pertinent numerical attributes.This pipeline cleans missing values, standardizes numerical features, pre-processes date-time into Year, Month, Day, and Hour, and then encodes categorical target variables. The neural network's deep feed-forward architecture consists of three hidden layers with 256, 128 and 64 neurons, respectively; to prevent overfitting, Batch Normalization and Dropout layers come after each hidden layer. The last output layer is Softmax-activated, allowing for multi-class classification, hence predicting one of the possible crime incident types in the dataset.



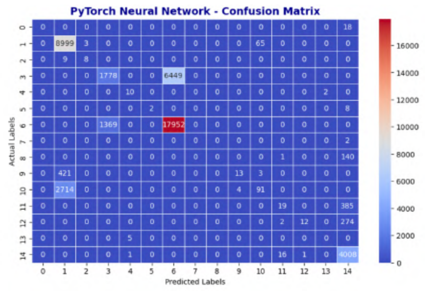
An artificial neural network is an interconnected group of nodes, inspired by a simplification of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial neuron to the input of another.

Fig. 6 displays the details of the accuracy, precision, f-1 score, confusion matrix, and support of a neural network that was trained using PyTorch. For the implemented neural network, 70% of the data was used for training, 15% for validation, and the remaining 15% for testing in order to provide a fair assessment. The neural network was created using PyTorch. Design: With a 30% ratio on the first two and 20% on the final layer, it is made up of three hidden layers with 256, 128 and 64 neurons, respectively. To avoid overfitting, Batch Normalization and Dropout are used after each layer. The output layer soft-max with N classes is the last layer. In this case, N is the distinct number of crime types that are present in the dataset. And the accuracy for the Neural Network is 73.45**%.**

Fig 6



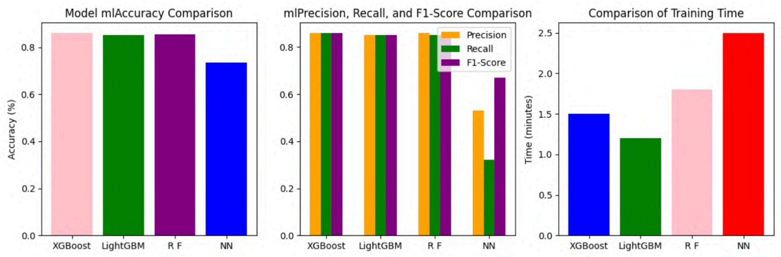
```
Classification Report is given by:

                          precision    recall  f1-score   support

            AGGR ASSAULT       0.00      0.00      0.00        18
                 ASSAULT       0.74      0.99      0.85      9067
                 Assault       0.73      0.47      0.57        17
                BURGLARY       0.56      0.22      0.31      8227
     Breaking & Entering       0.62      0.83      0.71        12
CRIM NEGLIGENT HOMICIDE        1.00      0.20      0.33        10
            LARCENY/THEFT      0.74      0.93      0.82     19321
            MANSLAUGHTER       0.00      0.00      0.00         2
                  MURDER       0.00      0.00      0.00       141
                    RAPE       0.76      0.03      0.06       437
                 ROBBERY       0.57      0.03      0.06      2809
            SEXUAL ABUSE       0.50      0.05      0.09       404
       THEFT OF SERVICES       0.92      0.04      0.08       288
                   Theft       0.00      0.00      0.00         5
                     UUV       0.83      1.00      0.90      4026

                accuracy                           0.73     44784
               macro avg       0.53      0.32      0.32     44784
            weighted avg       0.70      0.73      0.67     44784
```



Comparing the predictions - 3 ML and 1 NN Model VI

Fig 7 gives details about the comparison reports between 3 ML models and the Neural Network, and looks like the traditional ML models are efficient.

Fig 7



Part III: OCTMNIST Classification

The OCTMNIST is based on a prior dataset of 109,309 valid optical coherence tomography (OCT) images for retinal diseases. Each example is a 28x28 image, associated with a label from 4 classes. MedMNIST is a collection of multiple datasets and we worked with one dataset from the collection – OCTMNIST.

We code to load, split, & preprocess the MedMNIST dataset, which is made up of standardized 28x28 optical coherence tomography (OCT) images used to classify retinal diseases into four categories. We first load the train & test dataset splits and import all the necessary libraries that we would need. Random split is being utilized to divide the training set into training, validation, and test subset in a way that spreads the samples evenly, as desired. DataLoaders are then instantiated to handle batching as well as shuffling, this would enhance the process of training more effectively. The script also has a function for looping through the batches of data in order to get pixel values and labels correctly.
This function calculates useful statistics like the total samples, unique classes, class distribution, & pixel-level statistics (minimum, maximum, mean, & standard deviation). All of these values are necessary for making further model development.

Visualisations
Fig 8

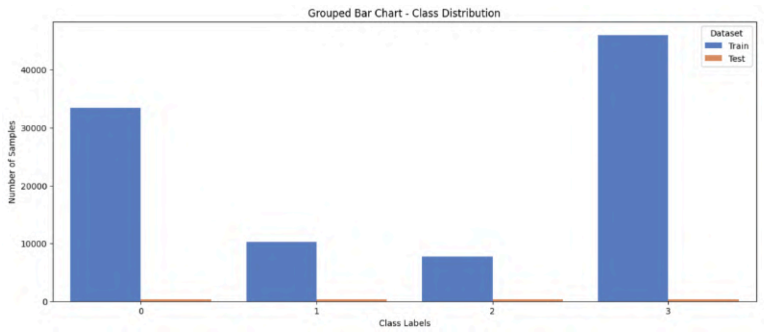
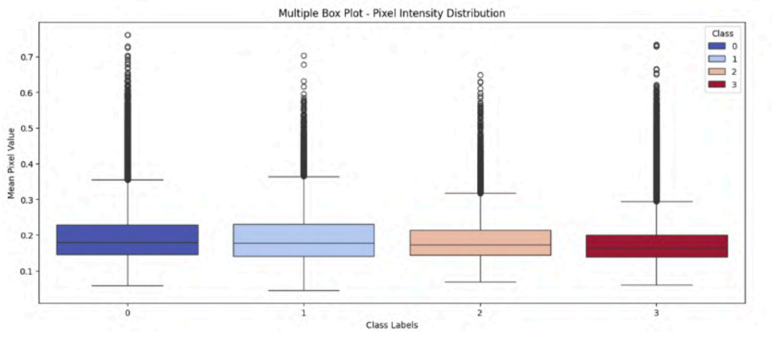


Fig 8 explains that the dataset has a significant class imbalance, with Class 3 and Class 0 having the highest number of samples - approximately 46,000 & 33,000, respectively, while Class 1 & Class 2 have considerably fewer - approximately 10,000 and 7,500. In contrast to this data, the test dataset appears more balanced but contains very few samples per class, this may result into wrong estimations. This imbalance can skew the model's predictions toward the majority classes, affecting overall performance, especially in terms of precision, recall, and F1-score values. In order to prevent it, techniques such as resampling, weighted loss functions, or data augmentation are considered in our next steps.

Fig 9



PyTorch Intro, Data analysis & NN Models

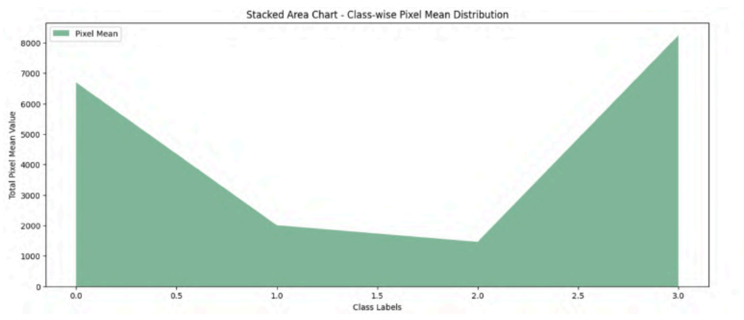
CSE 676-B: Deep Learning, Spring 2025

Sharanya Nallapeddi

50593866 | snallape@buffalo.edu

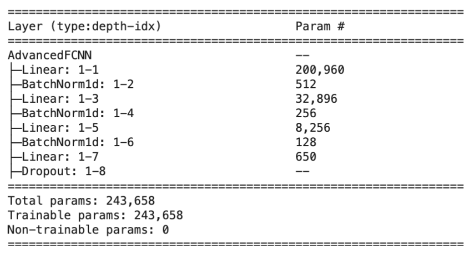
Similar to Fig 8., Fig 9, which is a grouped Box plot gives information about the pixel intensity distribution across different classes, showing that all classes have a similar median intensity but varying spreads. The presence of numerous outliers indicates some images have significantly higher pixel values, and this factor would apparently affect the performance of our model.

Fig 10



To confirm the necessity of the imbalance, another visualization, Fig 10 is implemented. The above stacked area chart shows the total pixel mean distribution across different classes, highlighting an uneven distribution where class 0 and class 3 have significantly higher pixel mean values in comparison with the classes 1 & 2. In the next steps, we would be defining our Neural Network and training it to save the best model possible.

Defining the neural Network



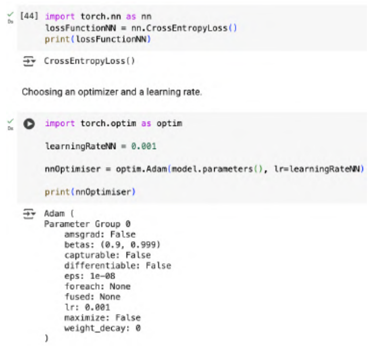
```
==================================================================
Layer (type:depth-idx)                   Param #
==================================================================
AdvancedFCNN                             --
├─Linear: 1-1                            200,960
├─BatchNorm1d: 1-2                       512
├─Linear: 1-3                            32,896
├─BatchNorm1d: 1-4                       256
├─Linear: 1-5                            8,256
├─BatchNorm1d: 1-6                       128
├─Linear: 1-7                            650
├─Dropout: 1-8                           --
==================================================================
Total params: 243,658
Trainable params: 243,658
Non-trainable params: 0
==================================================================
```

As shown in the picture, we have defined a fully connected Neural Network. with multiple layers designed for classification tasks. In the input layer, the model would be expecting an input which has a size of 784. Similarly, in the hidden layer, there are 3 sub layers, namely, First, second and third. In the first layer, there are 256 neurons with proper Batch normalization and ReLU Activation, where as in Second layer, it stores 128 neurons with proper Batch normalization and ELU activation. And finally, this layer has 64 neuron, includes batch normalization and ReLU activation. We also have the drop out regularization which is applied after every hidden layer with a rate of 30%. This would prevent overfitting.
Similar to this, the output layer is a fully connected neural network which can take unto 64 inputs and outputSize neurons that would map unto 10 classes. For the forward pass, the input would be passing through all the layers in a sequence, this would apply activation functions, and also, the batch normalization, to maintain stability. Thus, this architecture is useful for classification tasks and it includes **batch normalization** for improved training speed & stability, **dropout** for regularization, and **ReLU/ELU** activations for non-linearity.

Techniques that impacted the model's performance

- Dropout (0.3) for Regularization: Dropout is applied after the first 2 hidden layers to randomly deactivate 30% of the neurons during the training process. This would eventually prevent overfitting by making sure that the model does not rely too much on specific neurons.
- Learning Rate Scheduling (StepLR): StepLR would be reducing the learning rate by half per 5 epochs (as `gamma=0.5`). This would eventually allow the model to take large steps initially for quick convergence & smaller steps later, at a certain point to fine-tune performance, avoiding unnecessary simulations.
- Gradient Accumulation for Efficient Updates: Gradient Accumulation is used Instead of updating weights per batch. Our neural network model would accumulate gradients over 4 batches before updating - gradient_accumulation_steps=4 . And, this simulates training with a larger batch size, and thus, reduces noise in gradient updates which leads to more stable training, a well performed model.
- Early Stopping with Model Checkpointing: Early Stopping is used so that the model saves its weights whenever validation accuracy will improve, please see the code snippet -> (torch.save(model.state_dict(), "fcnnModel.pth")). This would eventually prevent overfitting by making sure that the best-performing model on validation data is retained.

While training the model, we have chosen optimizer and learning rate, its parameters are given. Using the Adam optimizer, our Fully Connected Neural Network (FCNN) which has 28 input features & 10 output classes would be achieving the best results in a multi-class classification. By managing all of the learning rates per parameter, Adam's 3-layer architecture (256 → 128 → 64 neurons) would help us stabilize training by preventing vanishing & exploding gradients.

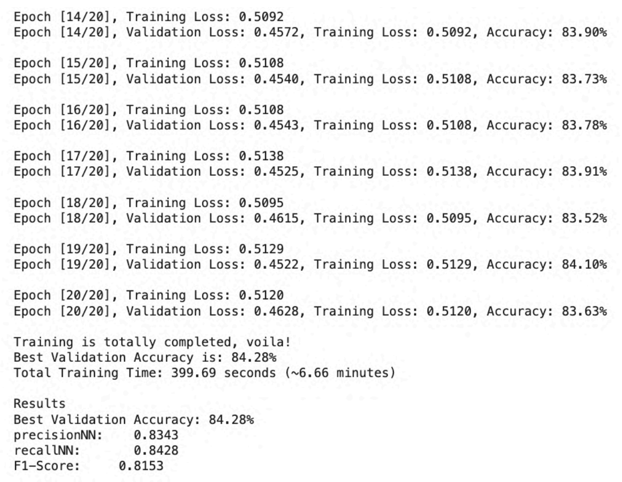


```
[44] import torch.nn as nn
     lossFunctionNN = nn.CrossEntropyLoss()
     print(lossFunctionNN)

CrossEntropyLoss()
```

Choosing an optimizer and a learning rate.

```
import torch.optim as optim

learningRateNN = 0.001

nnOptimiser = optim.Adam(model.parameters(), lr=learningRateNN)

print(nnOptimiser)

Adam (
Parameter Group 0
    amsgrad: False
    betas: (0.9, 0.999)
    capturable: False
    differentiable: False
    eps: 1e-08
    foreach: None
    fused: None
    lr: 0.001
    maximize: False
    weight_decay: 0
)
```

This Adam's optimizer is very optimistic for class separation & also for faster convergence than SGD as it makes use of the adaptive learning of RMSprop. Adam properly handles all the unbalanced crime data of Buffalo by segregating them properly into mini-batch updates. Once, we set up this, we training the network into 3 parts, Training, validation and testing. The percentage split of it is 75%, 15% and 15 %. We run the epochs and increase them depending on the accuracies while calculating the validation accuracy, testing accuracy and training accuracy. Similar to this, we also calculate the validation loss, testing and training loss.



```
Epoch [14/20], Training Loss: 0.5092
Epoch [14/20], Validation Loss: 0.4572, Training Loss: 0.5092, Accuracy: 83.90%

Epoch [15/20], Training Loss: 0.5108
Epoch [15/20], Validation Loss: 0.4540, Training Loss: 0.5108, Accuracy: 83.73%

Epoch [16/20], Training Loss: 0.5108
Epoch [16/20], Validation Loss: 0.4543, Training Loss: 0.5108, Accuracy: 83.78%

Epoch [17/20], Training Loss: 0.5138
Epoch [17/20], Validation Loss: 0.4525, Training Loss: 0.5138, Accuracy: 83.91%

Epoch [18/20], Training Loss: 0.5095
Epoch [18/20], Validation Loss: 0.4615, Training Loss: 0.5095, Accuracy: 83.52%

Epoch [19/20], Training Loss: 0.5129
Epoch [19/20], Validation Loss: 0.4522, Training Loss: 0.5129, Accuracy: 84.10%

Epoch [20/20], Training Loss: 0.5120
Epoch [20/20], Validation Loss: 0.4628, Training Loss: 0.5120, Accuracy: 83.63%

Training is totally completed, voila!
Best Validation Accuracy is: 84.28%
Total Training Time: 399.69 seconds (~6.66 minutes)

Results
Best Validation Accuracy: 84.28%
precisionNN:    0.8343
recallNN:       0.8428
F1-Score:     0.8153
```

Above picture gives details of the Accuracy, precision, recall and f1 score, which are calculated as we would be building our models performance based on these metrics. This parameters are for the base model.

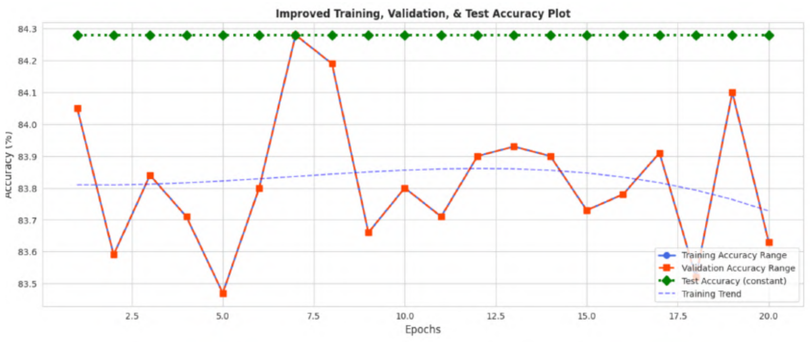
PyTorch Intro, Data analysis & NN Models

CSE 676-B: Deep Learning, Spring 2025

Sharanya Nallapeddi
50593866 | snallape@buffalo.edu

Fig 11

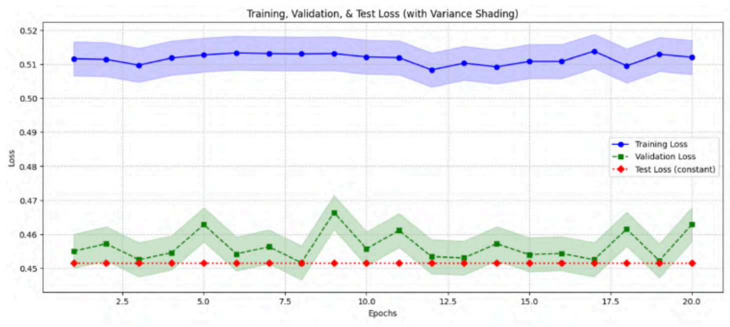
Fig 11 shows the trends in our FCNN model's training, validation, & test accuracy over a period of 20 epochs. As we increase the epochs, there is a chance that the



accuracy would also increase. Training accuracy is represented by the blue solid line with circles; it varies a little but stays mostly constant at 83.5% to 84.3%. The validation accuracy, shown by the orange dashed line with square is similar to the training accuracy & shows no discernible overfitting. The test accuracy in the above graph which is shown by the green dotted line with diamonds, stays steady/constant at 84.28%, which indicates stable generalization. A polynomial trend-line for training accuracy, the blue dashed curve first exhibits a growing pattern before stabilizing. The slight changes in accuracy over time point to fewer oscillations &most likely, these are brought on by training dynamics.

Similar to Fig 11, we have also implemented a visualization, fig 12 which

Fig 12



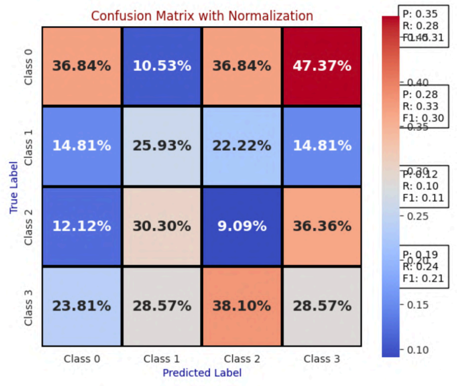
visualizes the training, validation, & test loss trends over 20 epochs, with variance declining for better interpretability. The blue solid line present in the above graph with circles represents training loss, this remains relatively constant around 0.51, with minor fluctuations. Whereas the green dashed line from Fig 12 with squares denote validation loss, and it varies more across epochs but stays around 0.45 - 0.47, showing peaks once in a while. And, the red dotted line from the graph with diamonds represents test loss, this remains constant at 0.45, indicating the model's performance on unseen data remaining steady. The shaded areas around training and validation loss curves highlight minor variations, indicating the model's stability & generalization capability.

```python
import matplotlib.pyplot as plot
import seaborn as sns
import numpy as np
from sklearn.metrics import confusion_matrix, roc_curve, auc

epochs = np.arange(1, 21)# Our 20 epochs

# Details related to Training, Validation & Test Accuracy from our FCNN model.
train_accuracy = [84.05, 83.59, 83.84, 83.71, 83.47, 83.80, 84.28, 84.19, 83.66, 83.80,
                  83.71, 83.90, 83.93, 83.90, 83.73, 83.78, 83.91, 83.52, 84.10, 83.63]
val_accuracy = train_accuracy
test_accuracy = [84.28] * len(epochs) #Using the best accuracy

# Losses for Training, Validation, & Test
LossInTraining = [0.5116, 0.5114, 0.5097, 0.5118, 0.5127, 0.5133, 0.5131, 0.5130, 0.5131, 0.5121,
              0.5119, 0.5083, 0.5103, 0.5092, 0.5108, 0.5108, 0.5138, 0.5095, 0.5129, 0.5120]
lossInVal = [0.4550, 0.4572, 0.4525, 0.4545, 0.4628, 0.4542, 0.4563, 0.4516, 0.4664, 0.4557,
            0.4611, 0.4534, 0.4530, 0.4572, 0.4540, 0.4543, 0.4525, 0.4615, 0.4522, 0.4628]
lossInTest = [min(lossInVal)] * len(epochs)
```

Note: We plot the visualizations based on all the epochs that are ran, we note down the values and code them in such a way that we can easily visualize the comparison. The code snippet for the following is given below.

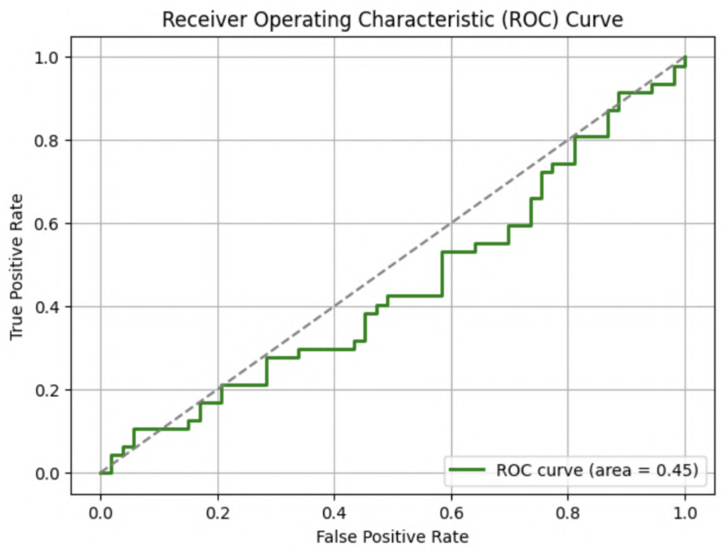
Fig 13 explains the confusion matrix which is implemented based on the different parameters like precision, f1 score, recall and accuracy from the ran 20 epochs. Confusion matrix shows the classification accuracy across 4 classes, with diagonal values that represent correct predictions. Class 0 & class 3 have high misclassification rates, particularly class 0 being predicted as class 3, approximating to 47.37% The precision, recall, and F1-scores vary, with class 0 having the highest precision about 0.35 & class 2, the lowest value of 0.19 F1-score. This suggests that while the model performs moderately, there are few/certain classes which are harder to classify correctly.

Fig 13



Receiver Operating Characteristics Curve(ROC)

Fig 14



The curve plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at different threshold values to assess the model's classification performance. The model's performance is somewhat inferior than random guessing (AUC < 0.5), as indicated by the green curve (AUC = 0.45), whereas the dashed diagonal line represents a random classifier.

PyTorch Intro, Data analysis & NN Models

CSE 676-B: Deep Learning, Spring 2025

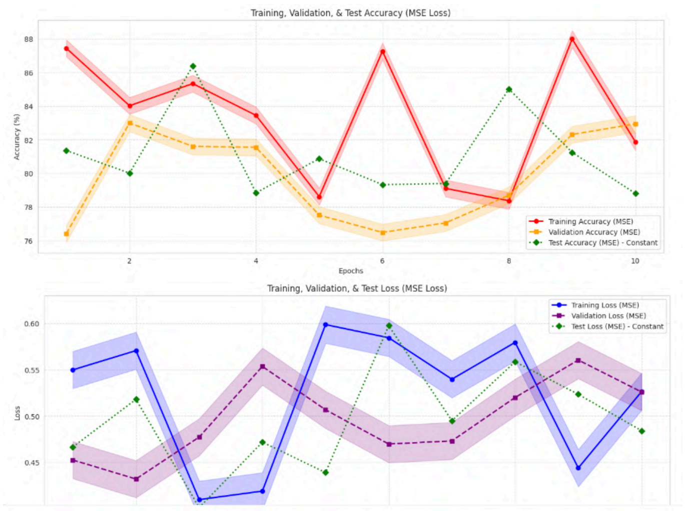
Sharanya Nallapeddi

50593866 | snallape@buffalo.edu

To further improve the model's performance, we have used a different loss function called MSE Loss function.

Fig 15 shows that using MSE Loss has improved the model's performance. Validation accuracy ranges between 76% and 86%, suggesting a steady learning without sharp declines, while training accuracy remains high whose value is between 78% and 88%.

Fig 15



The model is effectively optimizing without overfitting and training loss gradually decreasing from 0.55 to 0.60, the validation loss is around 0.40 to 0.60. Even though MSE is not properly used for classification, in this instance, its hierarchical 128-64-4 layer design, dropout (0.3), and Adam optimizer (LR = 0.001) have made it vital.

All of these techniques have shaped the model to perform better. After modifying accordingly on the basis of several methods(Batch Normalization, MSE Loss Function, Learning Rate Scheduler), another set of 10 epochs were ran, please see the below details.

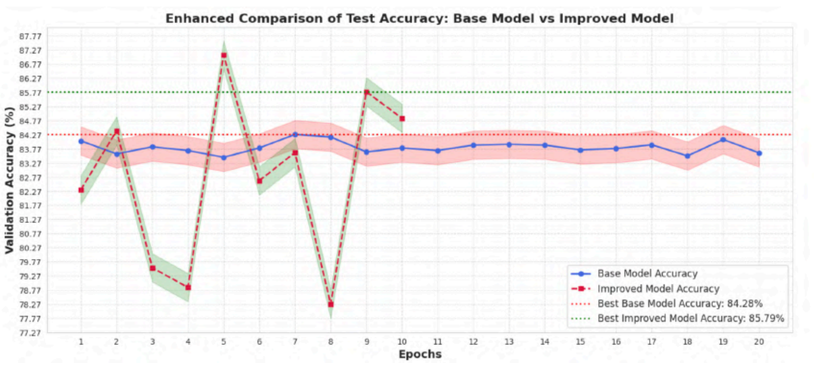
```
Epoch [1/10], Train Accuracy: 82.32%, Validation Accuracy: 81.51%, Test Accuracy: 79.91%
Epoch [2/10], Train Accuracy: 84.41%, Validation Accuracy: 84.15%, Test Accuracy: 82.16%
Epoch [3/10], Train Accuracy: 79.56%, Validation Accuracy: 85.79%, Test Accuracy: 86.95%
Epoch [4/10], Train Accuracy: 78.86%, Validation Accuracy: 83.48%, Test Accuracy: 86.33%
Epoch [5/10], Train Accuracy: 87.09%, Validation Accuracy: 82.37%, Test Accuracy: 78.22%
Epoch [6/10], Train Accuracy: 82.64%, Validation Accuracy: 76.20%, Test Accuracy: 78.09%
Epoch [7/10], Train Accuracy: 83.64%, Validation Accuracy: 81.42%, Test Accuracy: 82.32%
Epoch [8/10], Train Accuracy: 78.27%, Validation Accuracy: 82.18%, Test Accuracy: 81.10%
Epoch [9/10], Train Accuracy: 85.79%, Validation Accuracy: 83.18%, Test Accuracy: 77.75%
Epoch [10/10], Train Accuracy: 84.85%, Validation Accuracy: 78.43%, Test Accuracy: 86.10%

Model Training Completed successfully, voila!
Best Validation Accuracy is given as: 85.79%
```

Above indicates that our final accuracy is **85.79%.**

Please see the visualization that compares test accuracy for the model & an improved version, after applying all the techniques.

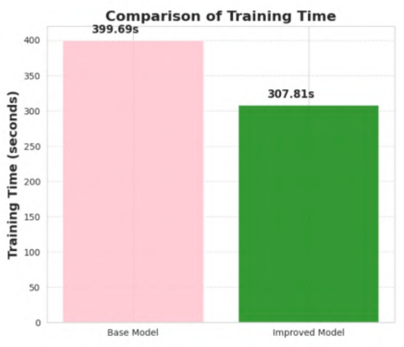
Fig 16 explains the comparison of the test accuracy between the base model and the improved model.

Fig 16



From Fig 17, its understandable that our improved model took less training time to build the complete model, enhancing the Neural Network speed.

Fig 17



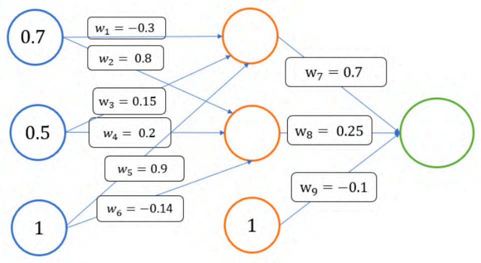
Part IV Deep Learning Theoretical Part

Forward-backward Pass

<u>Question 1</u>

Given the following setup:

- Hidden layer activation function: ReLU
- Output layer activation function: Linear
- Learning rate: 0.03
- Target (y): 0.5



TASK:

1. Perform a forward pass and estimate the predicted output ($\hat{y}$)
2. Estimate the MSE
3. Find the gradient using back-propagation
4. Update the weights
5. Draw a computation graph for the forward and backward pass
6. Perform a forward pass to estimate the predicted output using the updated weights.
7. Estimate the MSE and compare the results with Step 2.

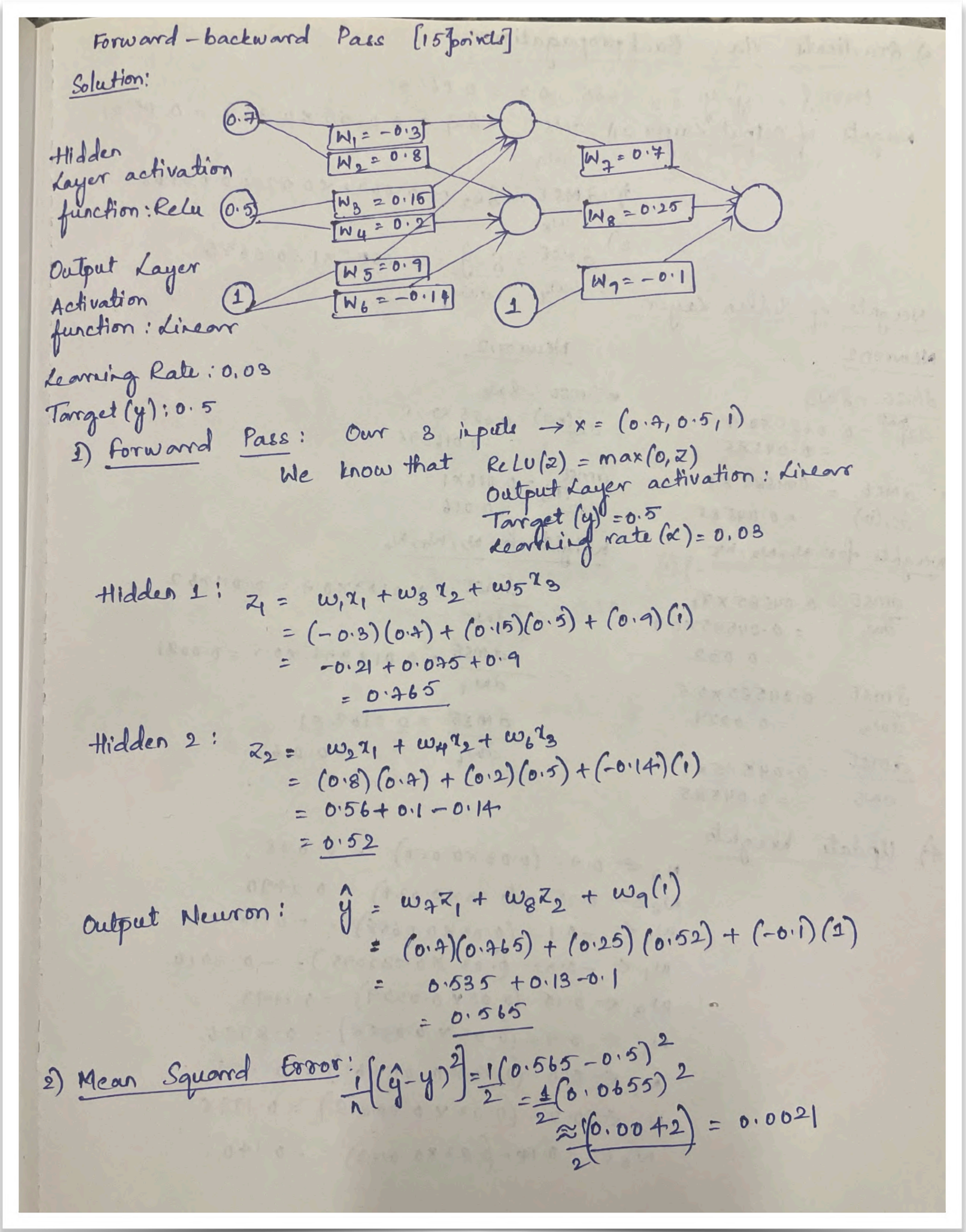
PyTorch Intro, Data analysis & NN Models

CSE 676-B: Deep Learning, Spring 2025

Sharanya Nallapeddi
50593866 | snallape@buffalo.edu

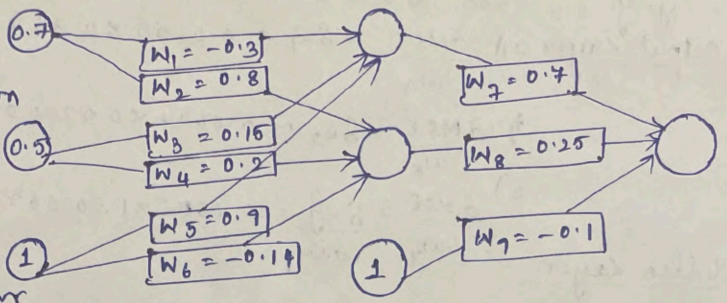
Solution 1

Page 1:



Forward-backward Pass [15 points]

Solution:



Hidden Layer activation function: Relu

Output Layer Activation function: Linear

Learning Rate: 0.03

Target (y): 0.5

1) Forward Pass: Our 3 inputs $\rightarrow x = (0.7, 0.5, 1)$

We know that $ReLU(z) = \max(0, z)$

Output Layer activation: Linear

Target (y) = 0.5

Learning rate (α) = 0.03

Hidden 1:
$$z_1 = w_1 x_1 + w_3 x_2 + w_5 x_3$$
$$= (-0.3)(0.7) + (0.15)(0.5) + (0.9)(1)$$
$$= -0.21 + 0.075 + 0.9$$
$$= \underline{0.765}$$

Hidden 2:
$$z_2 = w_2 x_1 + w_4 x_2 + w_6 x_3$$
$$= (0.8)(0.7) + (0.2)(0.5) + (-0.14)(1)$$
$$= 0.56 + 0.1 - 0.14$$
$$= \underline{0.52}$$

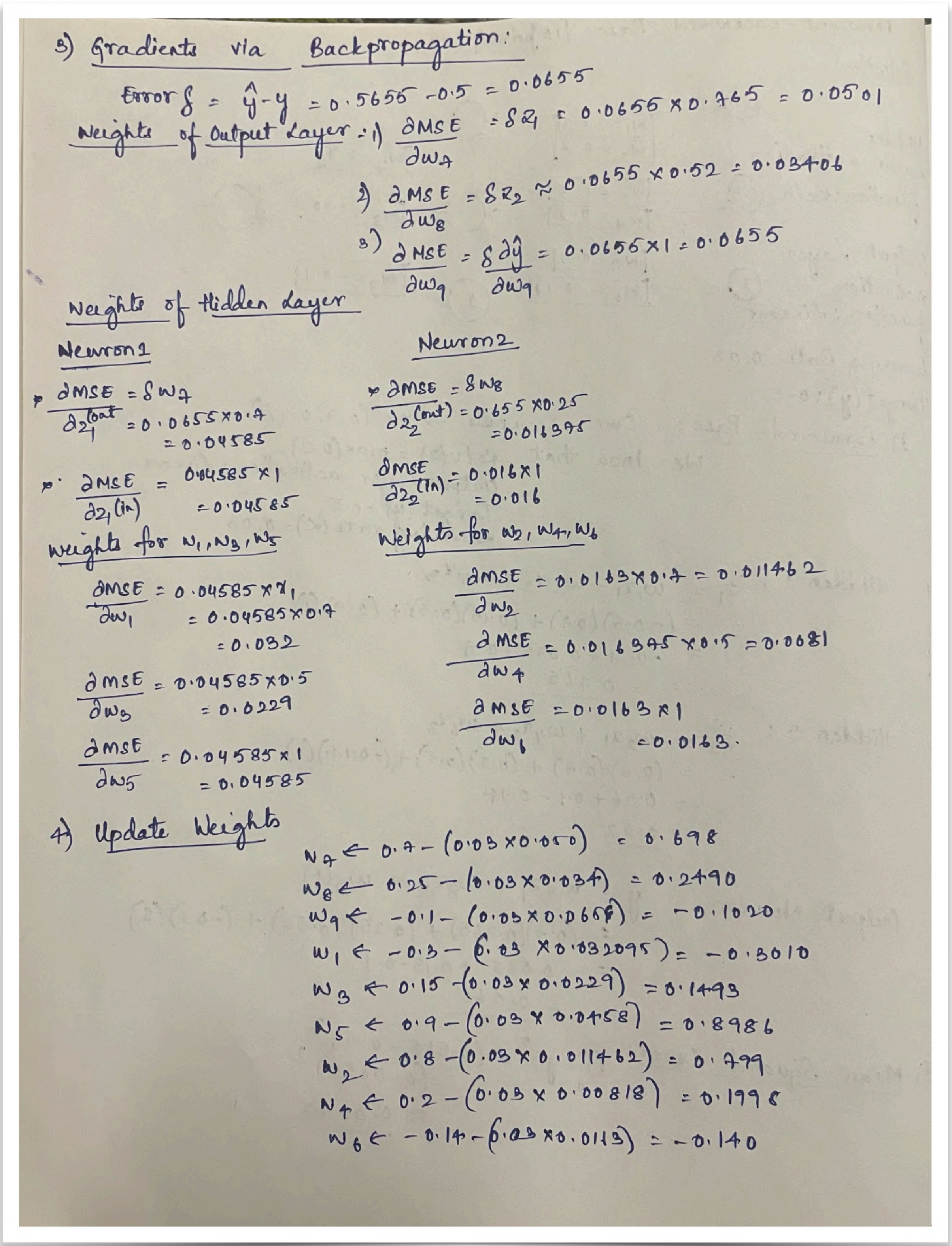
Output Neuron:
$$\hat{y} = w_7 z_1 + w_8 z_2 + w_9(1)$$
$$= (0.7)(0.765) + (0.25)(0.52) + (-0.1)(1)$$
$$= 0.535 + 0.13 - 0.1$$
$$= \underline{0.565}$$

2) Mean Squared Error:
$$\frac{1}{n}\left[(\hat{y} - y)^2\right] = \frac{1}{2}(0.565 - 0.5)^2$$
$$= \frac{1}{2}(0.0655)^2$$
$$\approx \frac{1}{2}(0.0042) = 0.0021$$

CSE 676-B: Deep Learning, Spring 2025
Sharanya Nallapeddi
50593866 | snallape@buffalo.edu

Solution 1

Page 2:



3) Gradients via Backpropagation:

Error $\delta = \hat{y} - y = 0.5655 - 0.5 = 0.0655$

Weights of Output Layer :- 1) $\frac{\partial MSE}{\partial w_7} = \delta z_1 = 0.0655 \times 0.765 = 0.0501$

2) $\frac{\partial MSE}{\partial w_8} = \delta z_2 \approx 0.0655 \times 0.52 = 0.03406$

3) $\frac{\partial MSE}{\partial w_9} = \delta \frac{\partial \hat{y}}{\partial w_9} = 0.0655 \times 1 = 0.0655$

Weights of Hidden Layer

Neuron1

- $\frac{\partial MSE}{\partial z_1^{(out)}} = \delta w_7$
  $= 0.0655 \times 0.7$
  $= 0.04585$

- $\frac{\partial MSE}{\partial z_1 (in)} = 0.04585 \times 1$
  $= 0.04585$

Weights for w_1, w_3, w_5

$\frac{\partial MSE}{\partial w_1} = 0.04585 \times x_1$
$= 0.04585 \times 0.7$
$= 0.032$

$\frac{\partial MSE}{\partial w_3} = 0.04585 \times 0.5$
$= 0.0229$

$\frac{\partial MSE}{\partial w_5} = 0.04585 \times 1$
$= 0.04585$

Neuron2

- $\frac{\partial MSE}{\partial z_2^{(out)}} = \delta w_8$
  $= 0.655 \times 0.25$
  $= 0.016375$

$\frac{\partial MSE}{\partial z_2^{(in)}} = 0.016 \times 1$
$= 0.016$

Weights for w_2, w_4, w_6

$\frac{\partial MSE}{\partial w_2} = 0.0163 \times 0.7 = 0.011462$

$\frac{\partial MSE}{\partial w_4} = 0.016375 \times 0.5 = 0.0081$

$\frac{\partial MSE}{\partial w_6} = 0.0163 \times 1$
$= 0.0163.$

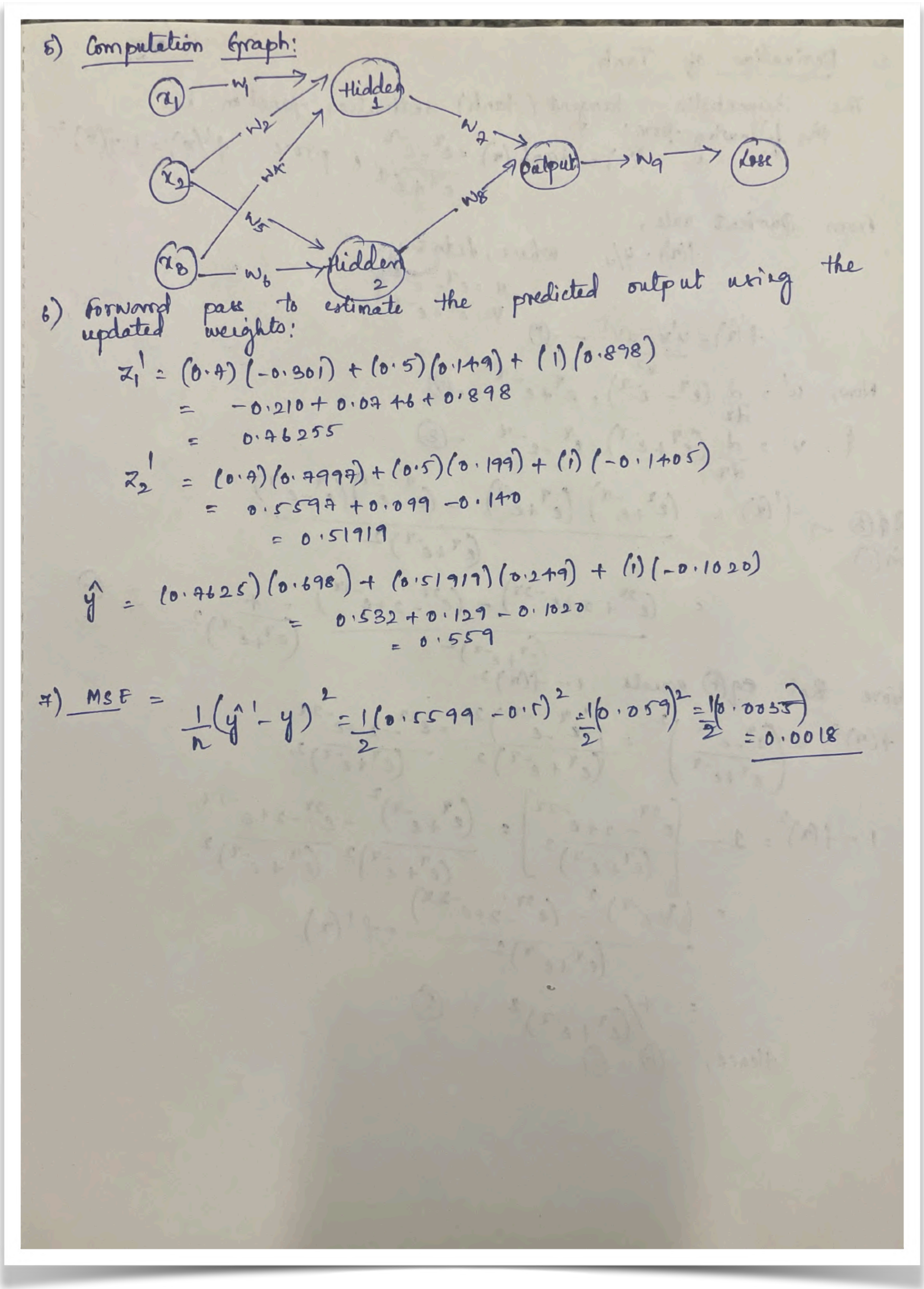
4) Update Weights

$w_7 \leftarrow 0.7 - (0.03 \times 0.050) = 0.698$
$w_8 \leftarrow 0.25 - (0.03 \times 0.034) = 0.2490$
$w_9 \leftarrow -0.1 - (0.03 \times 0.0658) = -0.1020$
$w_1 \leftarrow -0.3 - (0.03 \times 0.032095) = -0.3010$
$w_3 \leftarrow 0.15 - (0.03 \times 0.0229) = 0.1493$
$w_5 \leftarrow 0.9 - (0.03 \times 0.0458) = 0.8986$
$w_2 \leftarrow 0.8 - (0.03 \times 0.011462) = 0.799$
$w_4 \leftarrow 0.2 - (0.03 \times 0.00818) = 0.1998$
$w_6 \leftarrow -0.14 - (0.03 \times 0.0113) = -0.140$

Sharanya Nallapeddi
50593866 | snallape@buffalo.edu

Solution 1

Page 3:



5) Computation Graph:

$$\begin{tikzcd}
x_1 \arrow[r, "w_1"] & \text{Hidden } 1 \arrow[dr, "w_7"] & & \\
x_2 \arrow[ur, "w_2"] \arrow[dr, "w_5"] & & \text{output} \arrow[r, "w_9"] & \text{Loss} \\
x_3 \arrow[uur, "w_3"] \arrow[r, "w_6"] & \text{Hidden } 2 \arrow[ur, "w_8"] & &
\end{tikzcd}$$

6) Forward pass to estimate the predicted output using the updated weights:

$$z_1' = (0.7)(-0.301) + (0.5)(0.149) + (1)(0.898)$$
$$= -0.210 + 0.0746 + 0.898$$
$$= 0.76255$$

$$z_2' = (0.7)(0.7997) + (0.5)(0.199) + (1)(-0.1405)$$
$$= 0.5597 + 0.099 - 0.140$$
$$= 0.51919$$

$$\hat{y} = (0.7625)(0.698) + (0.51919)(0.249) + (1)(-0.1020)$$
$$= 0.532 + 0.129 - 0.1020$$
$$= 0.559$$

7) $\underline{\text{MSE}}$ $= \frac{1}{n}(\hat{y}' - y)^2 = \frac{1}{2}(0.5599 - 0.5)^2 = \frac{1}{2}(0.059)^2 = \frac{1}{2}(0.0035)$
$= \underline{0.0018}$

PyTorch Intro, Data analysis & NN Models

CSE 676-B: Deep Learning, Spring 2025

Sharanya Nallapeddi
50593866 | snallape@buffalo.edu

Question 2

2. Derivative of Tanh [5 points]

The hyperbolic tangent (tanh) activation function has the following form:

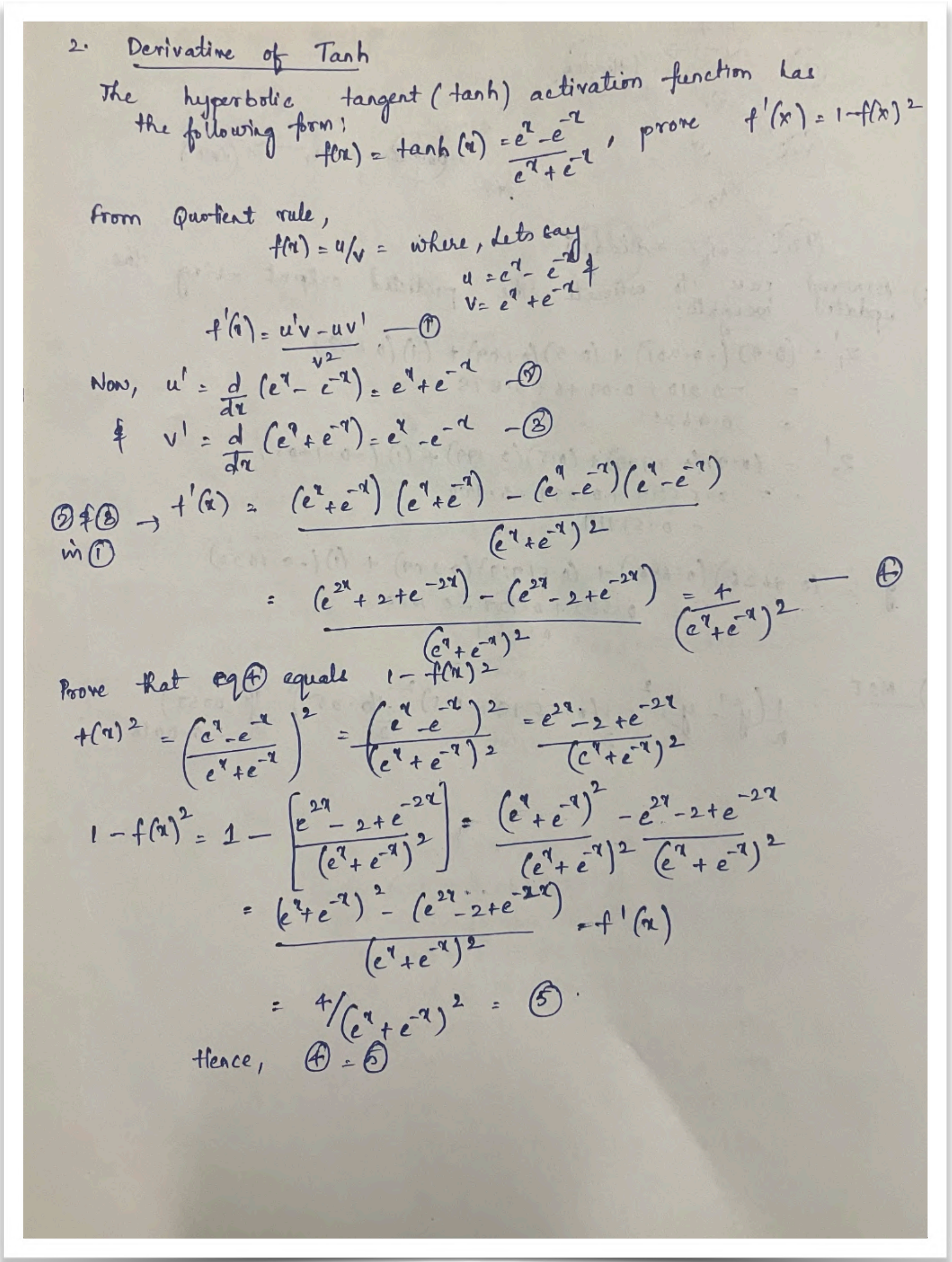
$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

TASK:

Prove that the derivative of Tanh has the following form:

$$f'(x) = 1 - f(x)^2$$

Solution 2:



2. Derivative of Tanh

The hyperbolic tangent (tanh) activation function has the following form:

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \quad \text{prove} \quad f'(x) = 1 - f(x)^2$$

From Quotient rule,

$f(x) = u/v =$ where, lets say $u = e^x - e^{-x}$ &
$v = e^x + e^{-x}$

$$f'(x) = \frac{u'v - uv'}{v^2} \quad — \textcircled{1}$$

Now, $u' = \frac{d}{dx}(e^x - e^{-x}) = e^x + e^{-x}$ — $\textcircled{2}$

& $v' = \frac{d}{dx}(e^x + e^{-x}) = e^x - e^{-x}$ — $\textcircled{3}$

$\textcircled{2}$ & $\textcircled{3}$ in $\textcircled{1}$ $\rightarrow$
$$f'(x) = \frac{(e^x + e^{-x})(e^x + e^{-x}) - (e^x - e^{-x})(e^x - e^{-x})}{(e^x + e^{-x})^2}$$

$$= \frac{(e^{2x} + 2 + e^{-2x}) - (e^{2x} - 2 + e^{-2x})}{(e^x + e^{-x})^2} = \frac{4}{(e^x + e^{-x})^2} \quad — \textcircled{4}$$

Prove that eq $\textcircled{4}$ equals $1 - f(x)^2$

$$f(x)^2 = \left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right)^2 = \frac{(e^x - e^{-x})^2}{(e^x + e^{-x})^2} = \frac{e^{2x} - 2 + e^{-2x}}{(e^x + e^{-x})^2}$$

$$1 - f(x)^2 = 1 - \left[\frac{e^{2x} - 2 + e^{-2x}}{(e^x + e^{-x})^2}\right] = \frac{(e^x + e^{-x})^2}{(e^x + e^{-x})^2} - \frac{e^{2x} - 2 + e^{-2x}}{(e^x + e^{-x})^2}$$

$$= \frac{(e^x + e^{-x})^2 - (e^{2x} - 2 + e^{-2x})}{(e^x + e^{-x})^2} = f'(x)$$

$$= 4/(e^x + e^{-x})^2 = \textcircled{5}$$

Hence, $\textcircled{4} = \textcircled{5}$

PyTorch Intro, Data analysis & NN Models

CSE 676-B: Deep Learning, Spring 2025

Sharanya Nallapeddi
50593866 | snallape@buffalo.edu

"Artificial Intelligence, deep learning, machine learning — whatever you're doing if you don't understand it — learn it. Because otherwise you're going to be a dinosaur within 3 years." ~Mark Cuban.

References

- City of Buffalo. (n.d.). *Crime Incidents*. Buffalo Open Data. Retrieved February 7, 2025, from https://data.buffalony.gov/Public-Safety/Crime-Incidents/d6g9-xbgu/about_data
- Zhang, Z., & Barr, A. (2024). *Gentrification and crime in Buffalo, New York*. *PLOS ONE*, 19(6), e0302832. https://doi.org/10.1371/journal.pone.0302832
- https://medmnist.com/
- https://github.com/MedMNIST/MedMNIST
- https://zenodo.org/record/6496656