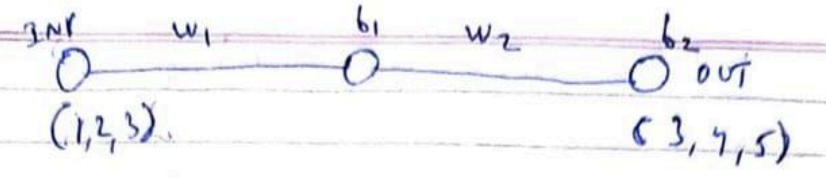
ML Assignment 3

Vikranth Udandarao
2022570

Section A

1.



loss function ⇒ Mean square error

lr = 0.01 (α)

let y_1 be output of $\{ w_1 = 0.3,\ b_1 = 0.1,\ w_2 = 0.4,\ b_2 = 0.2 \}$

for $x = 1$, $t = 3$,

forward propogation, $y_1 = w_1 x_1 + b_1$

$\rightarrow (0.3)(1) + 0.1$

$\rightarrow y_1 = 0.4$

$h = \text{Relu}(y_1) = \max(0, 0.4)$

$= 0.4$

$y = h \cdot w_2 + b_2 = (0.4)(0.4) + 0.2$

$= 0.36$

$MSE\ L = \frac{1}{2}(y - t)^2 = \frac{1}{2}(0.36 - 3)^2$

$= 3.4848$

Backward propogation,

$$\frac{dL}{dw_2} = (\underbrace{h_1 \cdot w_2 + b_2}_{y} - t) \cdot t$$

$$= (0.36 - 3)(0.4)$$

$$= -1.056$$

$$\frac{dL}{db_2} = (y - t)(1) = (0.36 - 3)$$

$$= -2.64$$

As $y_1 = 0.4 > 0$,

$$\frac{\partial b}{\partial y_1} = 1$$

So, $$\frac{\partial L}{\partial w_1} = (y - t)\left(w_2 \cdot \frac{\partial h}{\partial y_1}\, y \cdot x\right)$$

$$= (0.36 - 3)(0.4 \cdot 1)(1)$$

$$= -1.056$$

$$\frac{\partial L}{\partial b_1} = (y - t)\left(w_2 \cdot \frac{\partial b}{\partial y_1}\right)$$

$$= (0.36 - 3)(0.4 \cdot 1)$$

$$= -1.056$$

$b_2 = b_2 - \alpha \cdot \frac{\partial L}{\partial b_2} = 0.2 - (0.01)\ (-2.64)$

$= 0.2264$

$w_1 = w_1 - \alpha \frac{\partial L}{\partial w_1} = 0.3 - (0.01)\ (-1.056)$

$= 0.31056$

$b_1 = b_1 - \alpha \cdot \frac{\partial L}{\partial b_1} = 0.1 - 0.01\ (-1.056)$

$b_1 = 0.11056$

$x = 2;\ t = 4;$

$w_1 = 0.31056,\ b_1 = 0.11056,\ w_2 = 0.41056$
$b_2 = 0.2264$

forward pass, $y_1 = x \cdot w_1 + b_1$

$= 2 \cdot 0.31056 + 0.11056$
$= 0.73168$

$h = \text{ReLU}(y_1) = \max(0, 0.73168)$
$= 0.73168$

so, $y = h \cdot w_2 + b_2 = 0.73168 \cdot 0.41056 + 0.2264$

$= 0.5268$

Back propogation -

$$\frac{\partial L}{\partial w_1} = (y-t)\cdot h = (0.5868-4)\cdot(0.73168)$$

$$= -2.5407$$

$$\frac{\partial L}{\partial b_2} = (y-t) = 0.5268-4$$

$$= -3.4732$$

$$\frac{\partial L}{\partial w_1} = (y-t)\cdot w_2 \cdot \frac{\partial h}{\partial y_1}\cdot 4$$

$$= -2.8532$$

$$\frac{\partial L}{\partial b_1} = (y-t)\cdot w_2 \cdot \frac{\partial h}{\partial y_1}$$

$$= -1.4266$$

Update parameters ;

$$w'_2 = w_2 - \alpha \frac{\partial L}{\partial w_2}$$

$$= 0.41056 - 0.01(-2.5407)$$

$$= 0.4360$$

Date __/__/__ Page ____

$w_1 = w_1 - \alpha \frac{\partial L}{\partial w_1} = 0.3391$

$b_1 = b_1 - \alpha \frac{\partial L}{\partial b_1} = 0.11056 - (0.01)(-1.4266)$

$= 0.1248$

$\rightarrow w_1 = 0.3391, \ b_1 = 0.1248, \ w_2 = 0.436, \ b_2 = 0.2611$

$x = 3, \ t = 5$

$y_1 = x w_1 + b_1 = 3 \cdot 0.3391 + 0.1248 = 1.142$

$h = ReLU(y_1) = \max(0, 1.1421)$

$= 1.1421$

So, $y = h w_2 + b_2 = (1.1421) \cdot (0.4360) + 0.2611$

$= 0.7591$

Loss $L = \frac{1}{2}(y-t)^2 = \frac{1}{2}(0.7591-5)^2$

$= 8.9936$

Back propogation –

$\frac{\partial L}{\partial w_2} = (y-t) \cdot h = (0.7591 - 5) \cdot (1.1421)$

$= -4.8459$

$\frac{\partial L}{\partial} = (y-t) = 0.7591 - 5 = -4.2409$

$$\frac{\partial L}{\partial w_1} = (y-t)\, w_2 \cdot \frac{\partial b}{\partial y_1} \cdot x$$

$$= -5.5503$$

$$\frac{\partial L}{\partial b_1} = (y-t) \cdot w_2 \cdot \left(\frac{\partial b}{\partial y_1}\right)$$

$$= -4.2409 \cdot 0.43882 - 1.8807$$

Updated weights,

$$w_2 = w_2 - \lambda \frac{\partial L}{\partial w_2} = 0.4360 - 0.01(-4.8459)$$

$$= 0.4845$$

$$b_2 = 0.3035$$

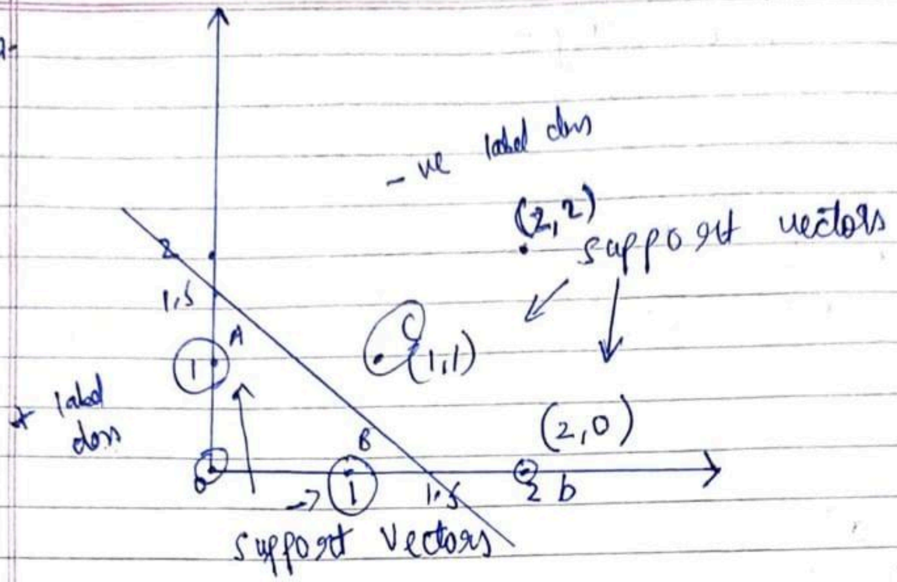
$$w_1 = 0.3946$$

$$b_1 = \cancel{b_1 - \alpha \cdot \frac{\partial L}{\partial b_1}} = 0.1433$$

→ find updated weights –

$w_1 = 0.3946$
$b_1 = 0.1433$
$w_2 = 0.4845$
$b_2 = 0.3035$

1.6-a-



Yes, the points are linearly separable.

b- We can see AB and CD are points closest to each other.

→ Hyperplane must be b/w them.

Line AB is parallel to CD by observation.

$x_1 + x_2 = 1$
$x_1 + x_2 = 2$

→ plane must go b/w them.

$\rightarrow x_1 + x_2 = 1.5$

→ Separating boundary $\Rightarrow x_1 + x_2 - 1.5 = 0$

Weight vector $= \begin{bmatrix} 1 \\ 1 \\ -1.5 \end{bmatrix}$

Date __/__/__
Page ____

cont-

6- (0,0) & (2,2) don't have impact on boundary -

+ve class

$(0,1) \rightarrow w_2 + b = 1$

$(1,0) \rightarrow w_1 + b = 1$

$w_2 - w_1 = 0$

$\rightarrow w_1 = w_2$

-ve class

$w_1 + b = 1$

$(2,0) \rightarrow 2w_1 + b = -1 \rightarrow w_1 = -2$

$(1,1) \rightarrow w_1 + w_2 + b = -1 \rightarrow w_2 = -2$

$\rightarrow b = 4 - 1 = 3$ -

$\rightarrow w_1 = w_2 = -2$

weights $b = 3$ -

$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$ -

$\rightarrow$ Decision boundary $= -2x_1 - 2x_2 + 3 = 0$ -

$y_i(w^T x + b) \geq 1$ -

$\rightarrow g(x) = w^T x + b$ -

$\sqrt{(-2)^2 + (-2)^2} = 2\sqrt{2}$ -

[illegible] $g(x) = 1$

Date ___/___/___
Page ______

+ve class $\rightarrow$ $(1,0) \rightarrow -2(1)+3 = 1$

$(0,1) \rightarrow -2(1)+3 = 1$

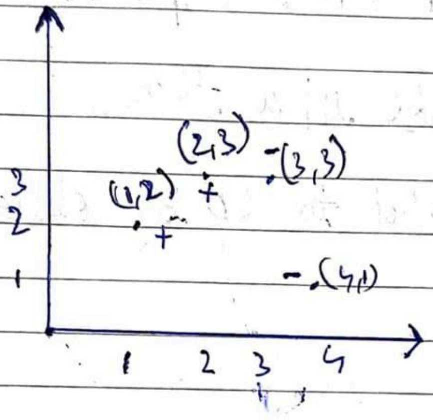
-ve class $\rightarrow$ $(1,1) \rightarrow -2-2+3 = -1$

$(2,0) \rightarrow 2(-2)+3 = -1$

$\therefore$ $(1,0)$, $(0,1)$, $(1,1)$, $(2,0)$

are support vectors.

C-a- (1,0)
(0,1)
(1,1)
(2,0) are support vectors



Boundery $\rightarrow$ $wx+b=0$

$w_1 = -2, w_2 = 0, b = 5$

$\rightarrow y_i \cdot (w^T x + b) = 1$

Margin $= \frac{2 y_i (w^T x + b)}{\|w\|} = \frac{2}{\|w\|}$

$\Rightarrow |w| = \sqrt{2^2+0^2} = 2$

$\rightarrow$ Margin $= \frac{\cancel{2}^1}{\cancel{2}} = 1$

mycompanion

Date ___/___/___
Page ___

b. $w^T = \begin{bmatrix} -2 \\ 0 \end{bmatrix}^T$

1. $x = [1\ 2]^T \rightarrow -2+5 = 3$

2. $x = [2\ 3]^T \rightarrow -4+5 = 1$

3. $x = [3\ 3]^T \rightarrow -6+5 = 1$

4. $x = [4\ 1]^T \rightarrow -8+5 = 3$

$\therefore$ (2,3) & (3,3) are support vectors with ± 1 margin distance.

c. $x_1 = 1,\ x_2 = 3.$

$w^T x + b = \begin{bmatrix} -2 \\ 0 \end{bmatrix}^T [1\ 3]^T + 5$

$= -2 + 0 + 5 = 3.$

$y_i (3) \geq 1.$

$\therefore$ Positive class.

Section B

Observations for Different Activation Functions and Weight Initializations

1. **Sigmoid Activation**
    - **Zero Initialization**: The training, validation, and test losses remain almost constant around a high value (around 2.3), indicating no learning has occurred. This is expected because zero initialization leads to symmetry, preventing the network from learning effectively.
        - **Val Accuracy**: 0.12
        - **Test Accuracy**: 0.12
    - **Random Initialization**: The losses converge relatively quickly, showing a significant decrease, especially early on, and stabilize around a low value. This suggests effective learning, and the accuracy values support this.
        - **Val Accuracy**: 0.95
        - **Test Accuracy**: 0.93
    - **Normal Initialization**: Similar to random initialization, the model converges well, achieving slightly better accuracy than random initialization, suggesting that normal initialization is beneficial for this activation function.
        - **Val Accuracy**: 0.97
        - **Test Accuracy**: 0.96
2. **Tanh Activation**
    - **Zero Initialization**: Similar to the sigmoid zero-initialization case, the model fails to learn effectively, with losses remaining nearly constant around the initial value. This again demonstrates the negative impact of zero initialization.
        - **Val Accuracy**: 0.097
        - **Test Accuracy**: 0.09
    - **Random Initialization**: The model converges efficiently with random initialization, resulting in low loss values for training, validation, and test sets. The accuracy metrics are high, reflecting effective learning.
        - **Val Accuracy**: 0.98
        - **Test Accuracy**: 0.97
    - **Normal Initialization**: Like random initialization, normal initialization also achieves quick convergence and high accuracy. The validation and test accuracy are very close, indicating good generalization.
        - **Val Accuracy**: 0.97
        - **Test Accuracy**: 0.97
3. **ReLU Activation**

    - **Zero Initialization**: Similar to other activations with zero initialization, the ReLU activation fails to converge, as reflected by flat loss curves and poor accuracy scores.
        - **Val Accuracy**: 0.09
        - **Test Accuracy**: 0.1
    - **Random Initialization**: Random initialization yields good convergence, with training and validation losses decreasing initially but stabilizing. There is a slight increase in validation and test loss over epochs, possibly indicating slight overfitting.
        - **Val Accuracy**: 0.98
        - **Test Accuracy**: 0.95
    - **Normal Initialization**: The model converges effectively, achieving stable and low loss values for all datasets. The accuracy results are high, and the model generalizes well.
        - **Val Accuracy**: 0.96
        - **Test Accuracy**: 0.98
4. **Leaky ReLU Activation**
    - **Zero Initialization**: The loss remains almost constant, failing to converge due to zero initialization. Accuracy values are also very low, indicating no learning.
        - **Val Accuracy**: 0.09
        - **Test Accuracy**: 0.09
    - **Random Initialization**: Random initialization with Leaky ReLU shows good convergence, achieving low training and validation losses. This configuration yields high accuracy, reflecting strong learning and generalization.
        - **Val Accuracy**: 0.96
        - **Test Accuracy**: 0.97
    - **Normal Initialization**: Similar to random initialization, normal initialization also leads to effective convergence with low loss values and high accuracy scores.
        - **Val Accuracy**: 0.97
        - **Test Accuracy**: 0.97

Summary and Insights

- **Effect of Initialization**:
    - Zero initialization consistently fails across all activation functions, as it causes neurons to learn the same features due to symmetry, preventing effective learning.
    - Random and normal initializations both perform well, with slight variations in accuracy, suggesting that these initializations break symmetry and allow the model to learn effectively. Normal initialization often yields slightly better accuracy than random initialization.
- **Effect of Activation Functions**:

    - **Sigmoid** and **Tanh** activations show good convergence when paired with non-zero initializations but tend to saturate (slow gradient descent) with deeper networks. However, the losses decrease smoothly, reflecting stable learning.
    - **ReLU** and **Leaky ReLU** generally converge faster and to lower losses compared to Sigmoid and Tanh. ReLU and Leaky ReLU activation functions also demonstrate good generalization in terms of validation and test accuracy, particularly with normal initialization.
- **Best Configuration**:
    - The highest accuracy scores are observed with Tanh (random or normal initialization) and ReLU (normal initialization), indicating that these configurations offer a good balance between learning rate and stability.

1. Loading the Dataset

- The MNIST dataset is loaded from four IDX files, two for training and two for testing. Each file is processed separately:
    - `train-images.idx3-ubyte`: Contains grayscale pixel values for training images.
    - `train-labels.idx1-ubyte`: Contains labels for training images.
    - `t10k-images.idx3-ubyte`: Contains grayscale pixel values for testing images.
    - `t10k-labels.idx1-ubyte`: Contains labels for testing images.
- Two custom functions are defined for reading the IDX files:
    - **`

`load_images`**: Reads the image files by interpreting the first 16 bytes as metadata (including dimensions) and then loads the remaining bytes as pixel values, reshaping each image to 28 x 28 (784 pixels). - **`load_labels`**: Reads the label files by interpreting the first 8 bytes as metadata and then loading the labels for each image.

2. Normalization

- After loading, the pixel values in the images are normalized by dividing by 255.0, converting the range from `[0, 255]` to `[0, 1]`. This rescaling helps in stabilizing the neural network training process, as neural networks typically perform better with smaller input ranges.

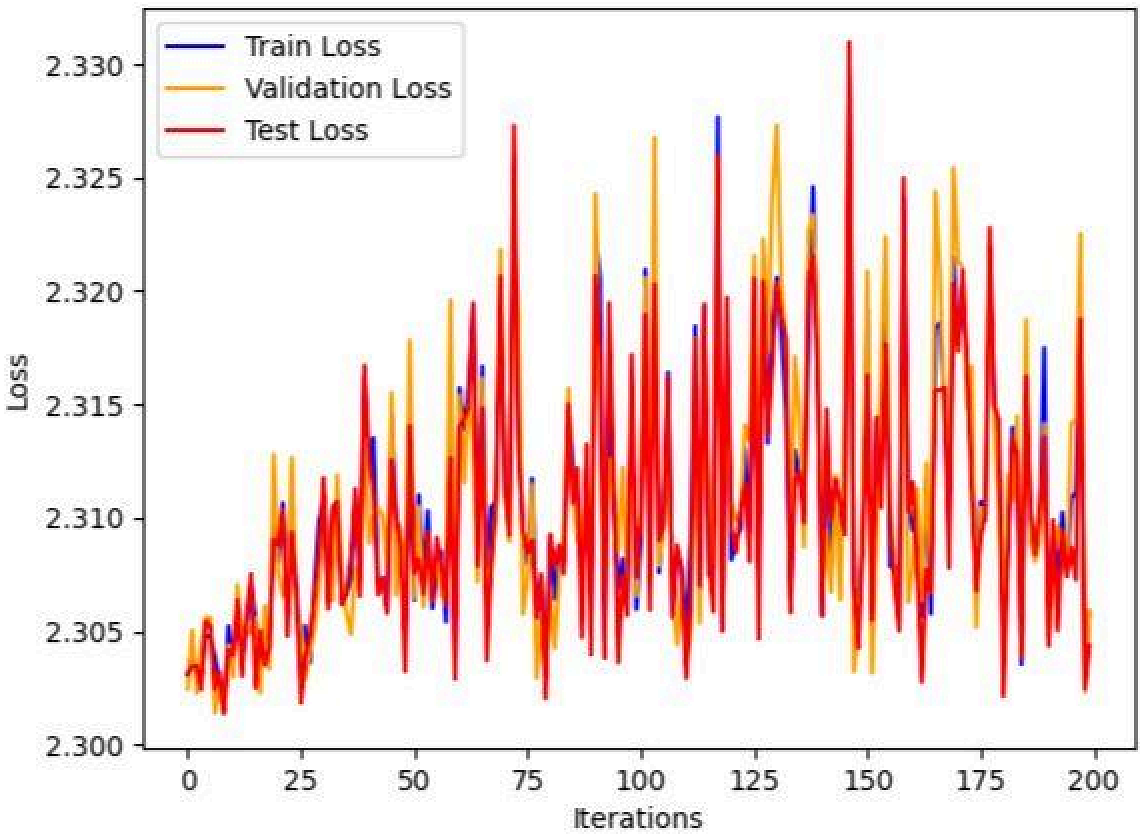
3. One-Hot Encoding

- Labels are one-hot encoded to create a matrix representation where each row corresponds to an image, and each column corresponds to a class. For instance, if an

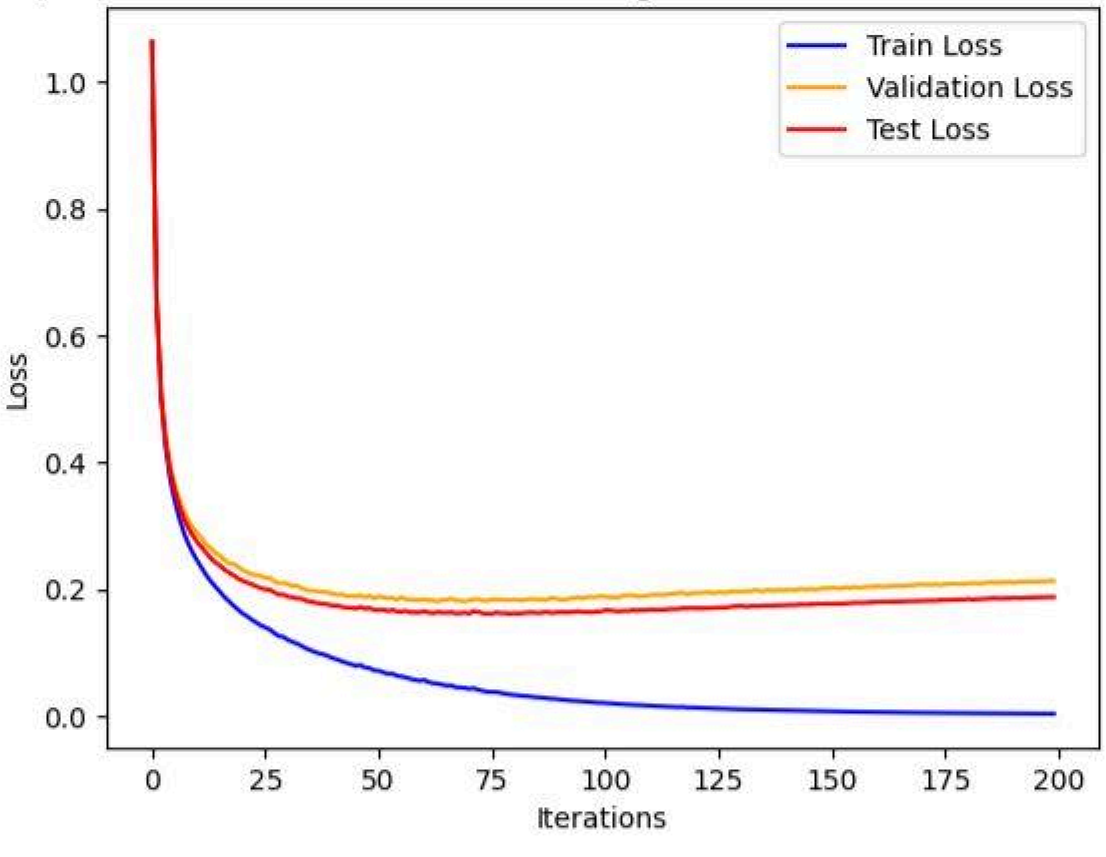
image is labeled as `3`, its one-hot encoded label would be `[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]`.
- This one-hot encoding is essential for multi-class classification tasks and facilitates the use of cross-entropy loss during training.

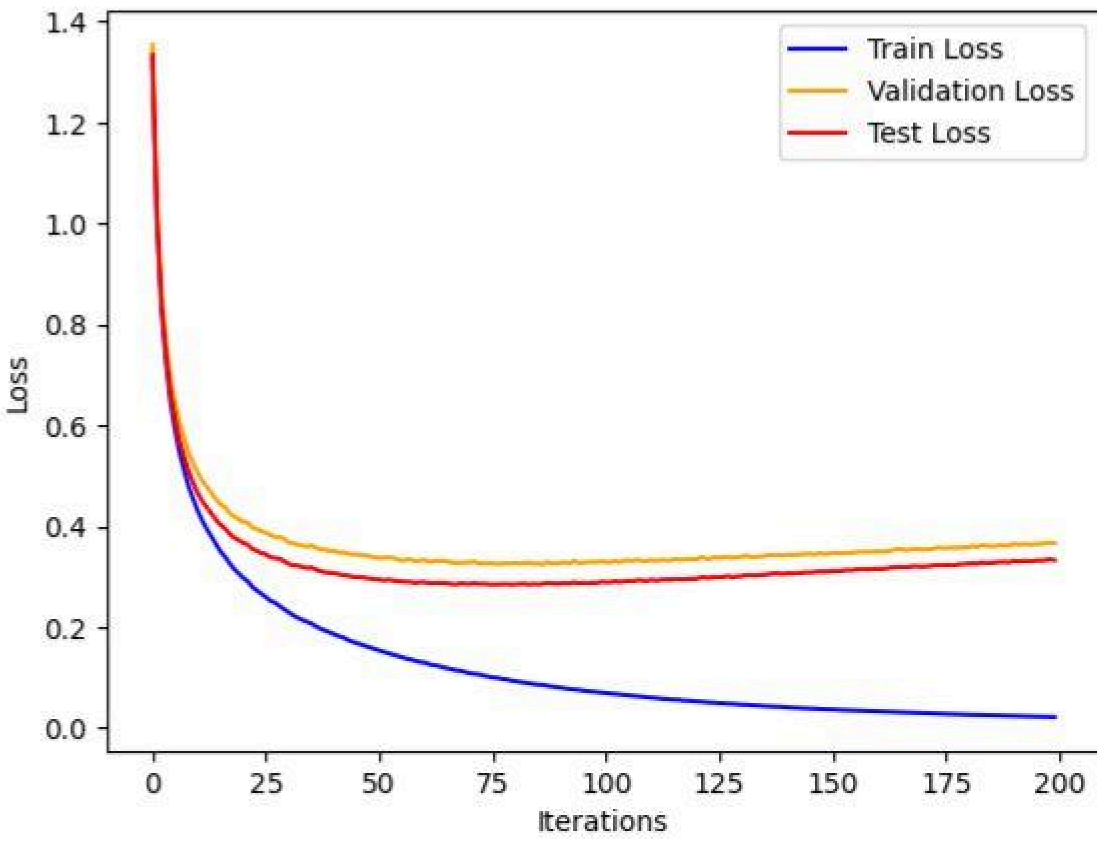
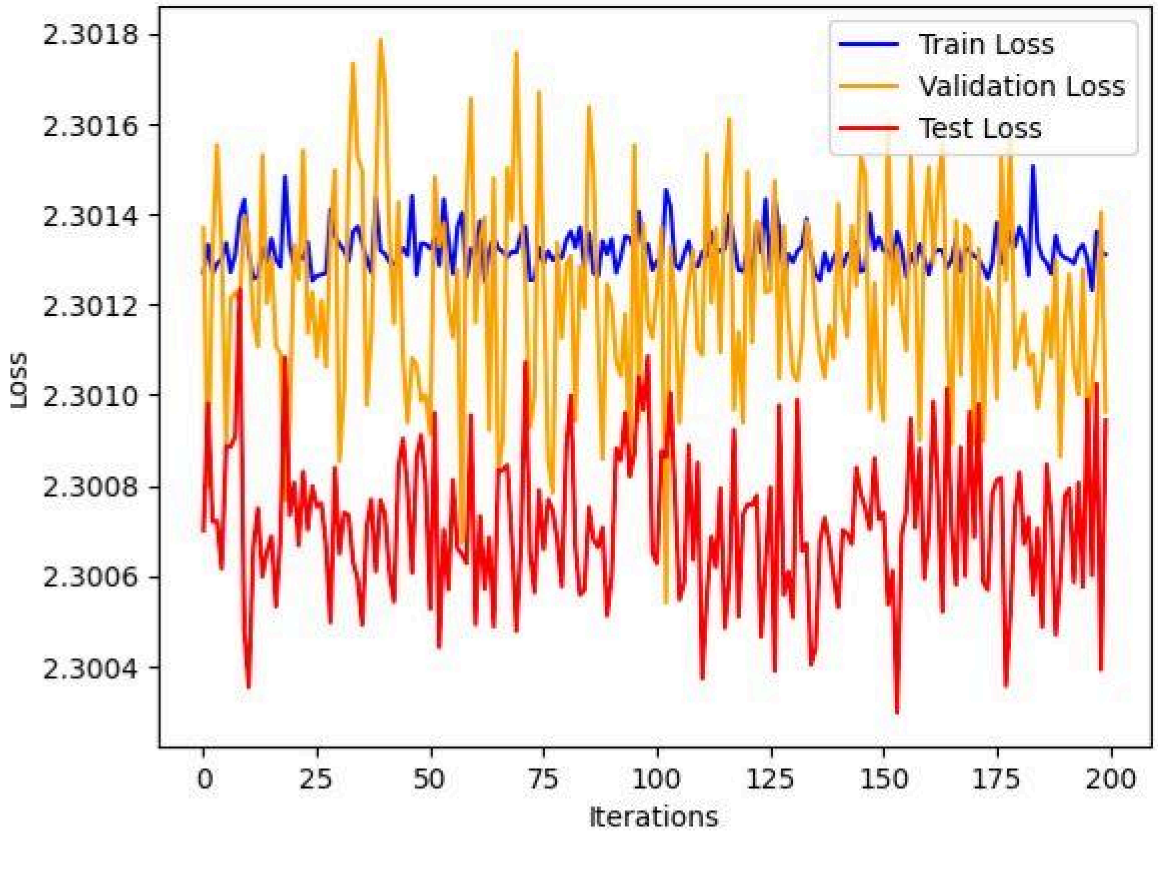
4. Train-Validation Split

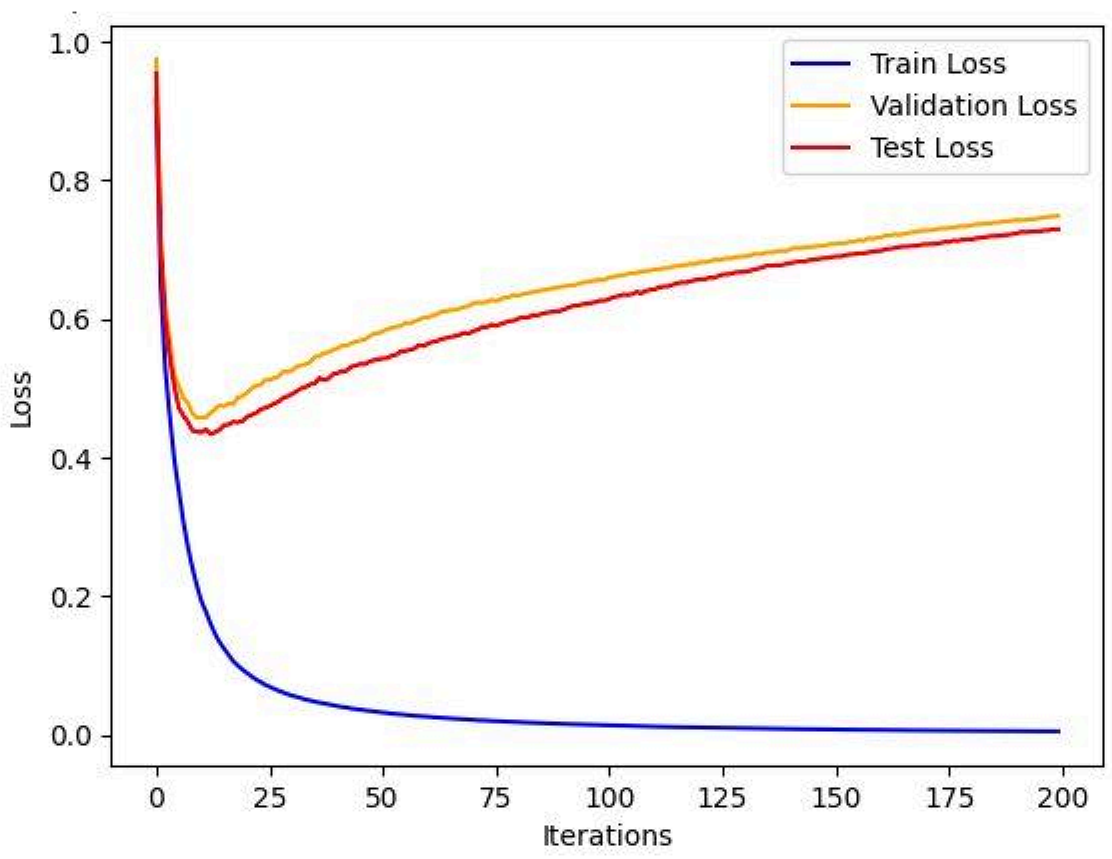
- The training dataset is split further into a training set (90%) and a validation set (10%) using `train_test_split` from `sklearn`. The validation set is used to monitor the model's performance during training and to implement early stopping to prevent overfitting.
- This split ensures that we can assess the model's generalization ability without touching the test set until final evaluation.

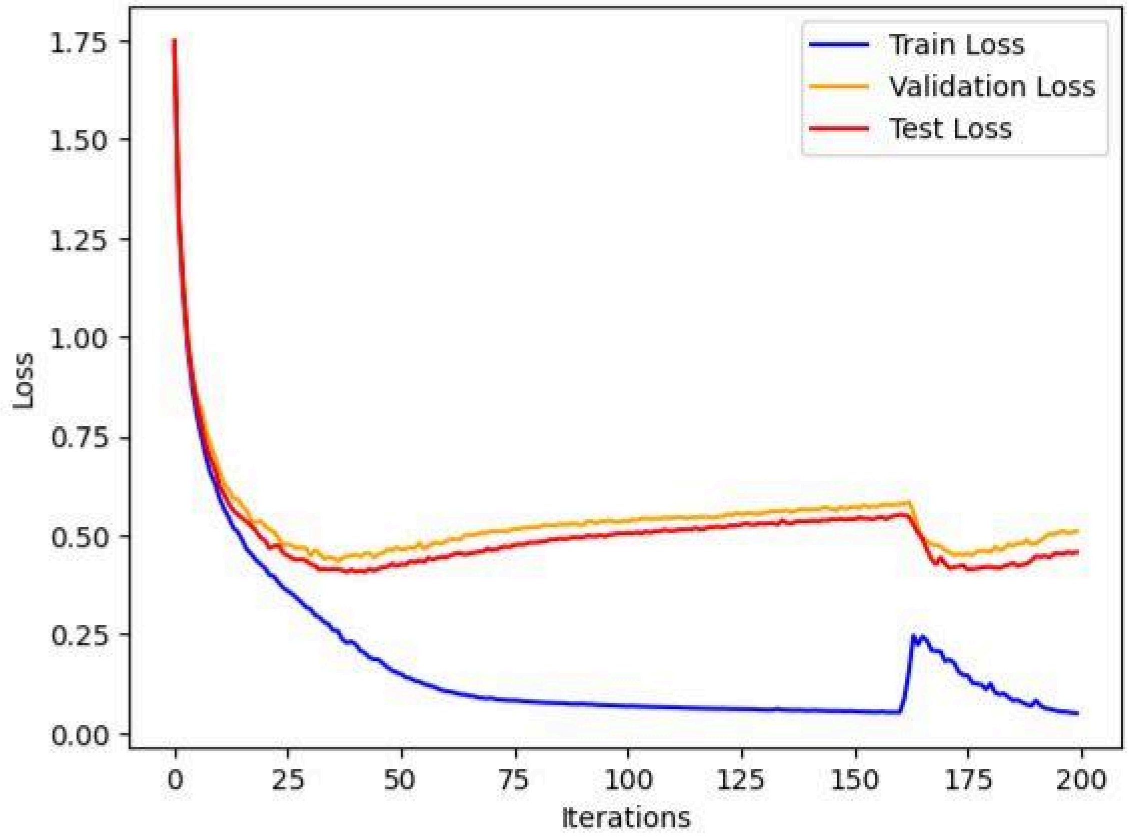

Train Loss
Validation Loss
Test Loss
Loss
Iterations
2.300
2.305
2.310
2.315
2.320
2.325
2.330
0
25
50
75
100
125
150
175
200

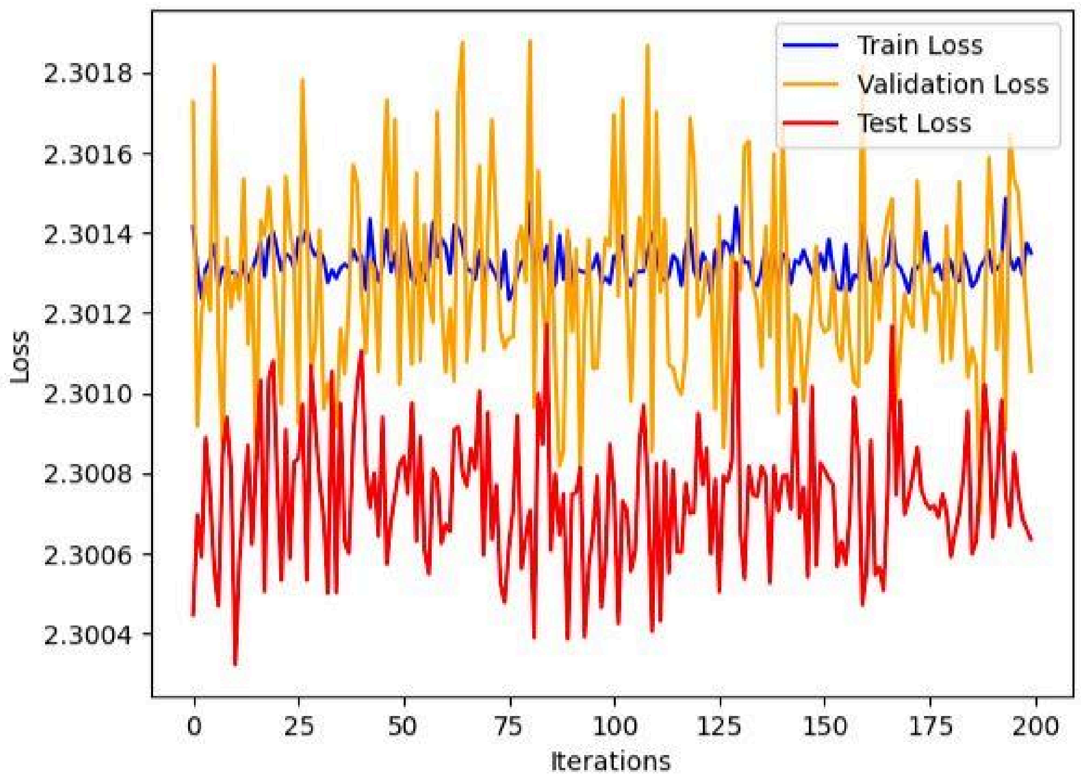
1.

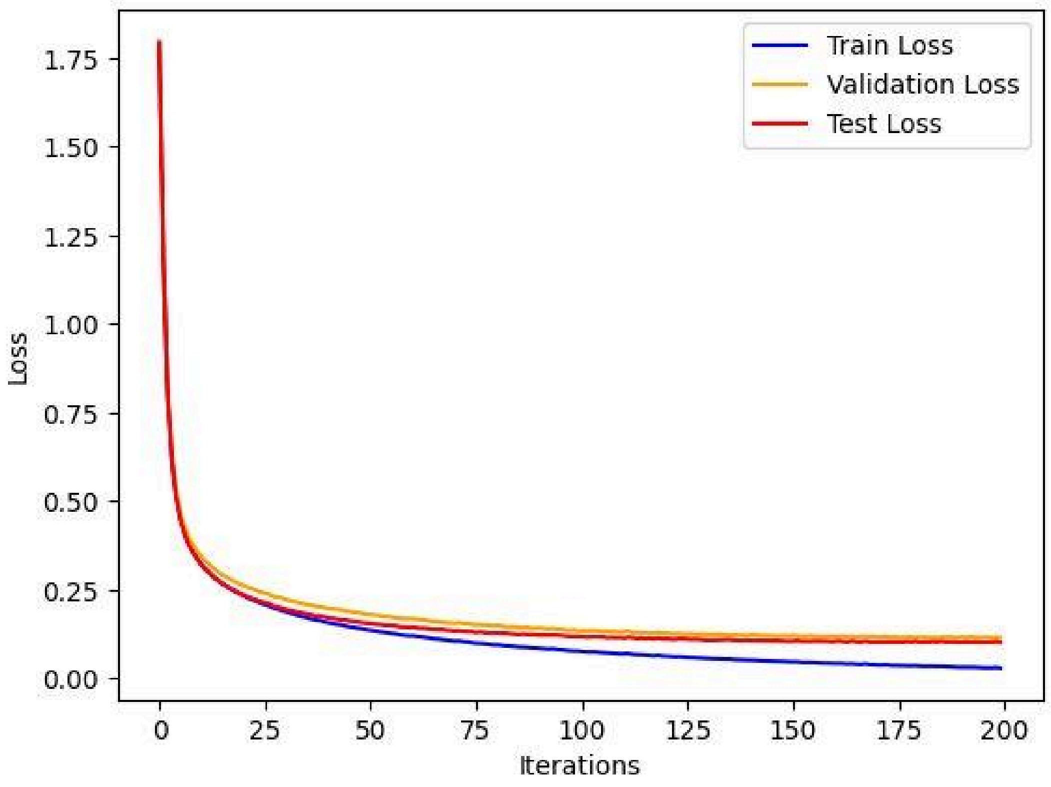

Train Loss
Validation Loss
Test Loss
Loss
Iterations
0.0
0.2
0.4
0.6
0.8
1.0
0
25
50
75
100
125
150
175
200


Train Loss
Validation Loss
Test Loss
Loss
Iterations
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175
200

Train Loss
Validation Loss
Test Loss
Loss
Iterations
2.3018
2.3016
2.3014
2.3012
2.3010
2.3008
2.3006
2.3004
0
25
50
75
100
125
150
175
200


Train Loss
Validation Loss
Test Loss
Loss
Iterations
1.0
0.8
0.6
0.4
0.2
0.0
0
25
50
75
100
125
150
175
200


Train Loss
Validation Loss
Test Loss
Loss
Iterations
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
25
50
75
100
125
150
175
200

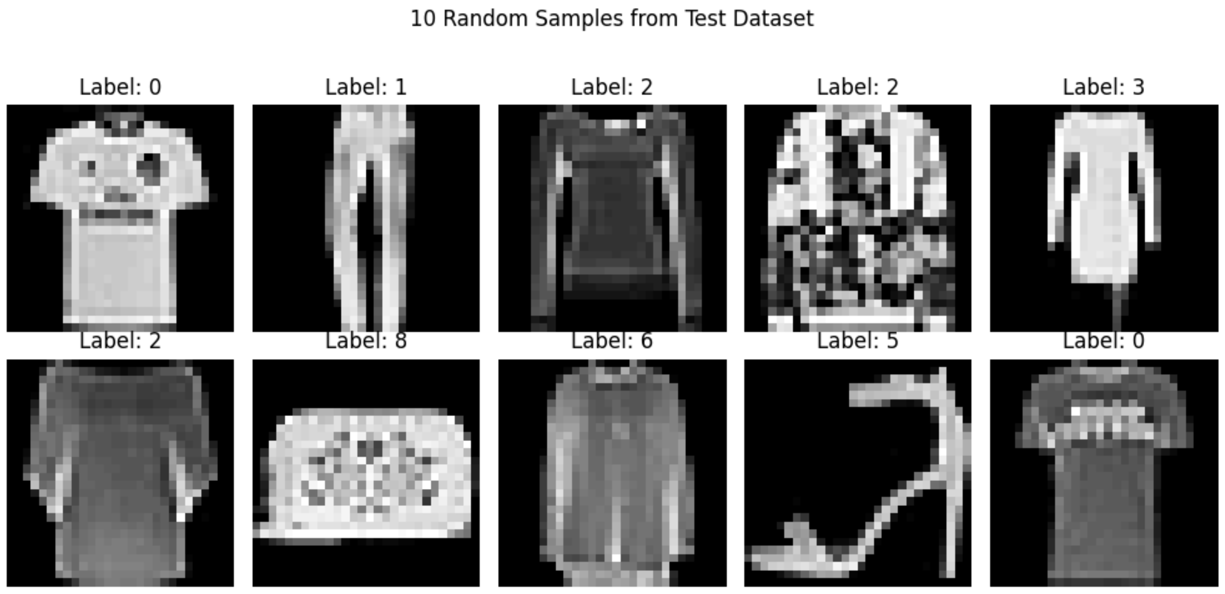


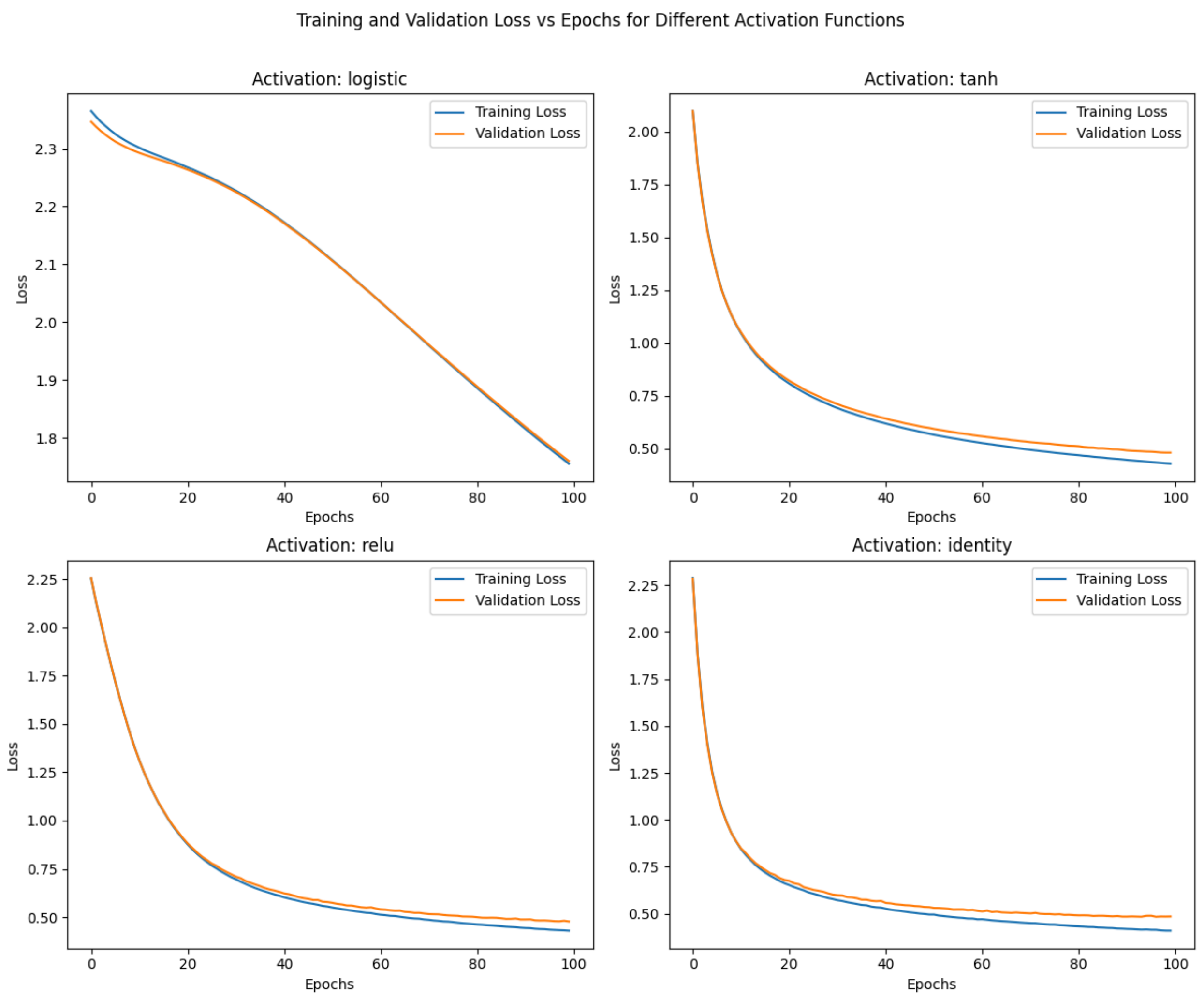


Activation: sigmoid, Init: zero
Val accuracy: 0.12
Test accuracy: 0.12

```
Activation: sigmoid, Init: random
Val accuracy: 0.95
Test accuracy: 0.93

Activation: sigmoid, Init: normal
Val accuracy: 0.97
Test accuracy: 0.96

Activation: tanh, Init: zero
Val accuracy: 0.097
Test accuracy: 0.09

Activation: tanh, Init: random
Val accuracy: 0.98
Test accuracy: 0.97

Activation: tanh, Init: normal
Val accuracy: 0.97
Test accuracy: 0.97

Activation: relu, Init: zero
Val accuracy: 0.09
Test accuracy: 0.1

Activation: relu, Init: random
Val accuracy: 0.98
Test accuracy: 0.95

Activation: relu, Init: normal
Val accuracy: 0.96
Test accuracy: 0.98

Activation: leaky_relu, Init: zero
Val accuracy: 0.09
Test accuracy: 0.09

Activation: leaky_relu, Init: random
Val accuracy: 0.96
Test accuracy: 0.97

Activation: leaky_relu, Init: normal
Val accuracy: 0.97
Test accuracy: 0.97
```

Section C

1. 
10 Random Samples from Test Dataset
Label: 0
Label: 1
Label: 2
Label: 2
Label: 3
Label: 2
Label: 8
Label: 6
Label: 5
Label: 0



2.

The best model among the provided activation functions can be identified by comparing the **validation loss at the last epoch** (epoch 99). Lower validation loss generally indicates better generalization performance on unseen data.

1. **Logistic Activation**:
   - Final Validation Loss: 1.7790
2. **Tanh Activation**:
   - Final Validation Loss: 0.4612
3. **ReLU Activation**:
   - Final Validation Loss: 0.4776
4. **Identity Activation**:
   - Final Validation Loss: 0.4771

From this analysis:

- **Tanh** has the lowest final validation loss (0.4612) compared to the other activations, indicating it performed best in this training session.

- Both **ReLU** and **Identity** activations also achieved reasonable performance but had slightly higher final validation losses than **Tanh**.
- The **Logistic** activation has the highest validation loss by a significant margin, suggesting it was less effective for this task.

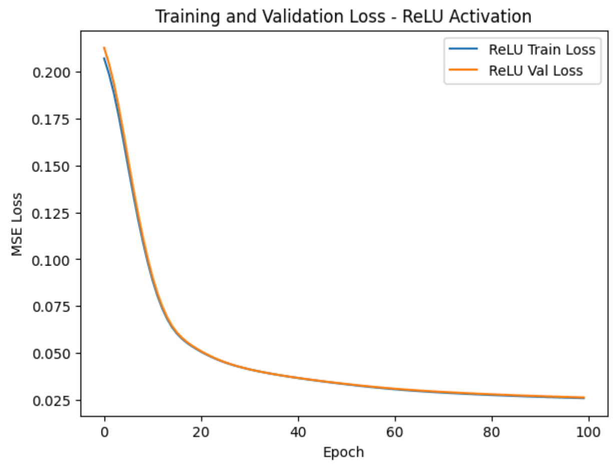
Thus, **Tanh activation** is the best choice based on the training outcome, as it provides the lowest validation loss, indicating that it has learned to generalize better on this particular dataset.

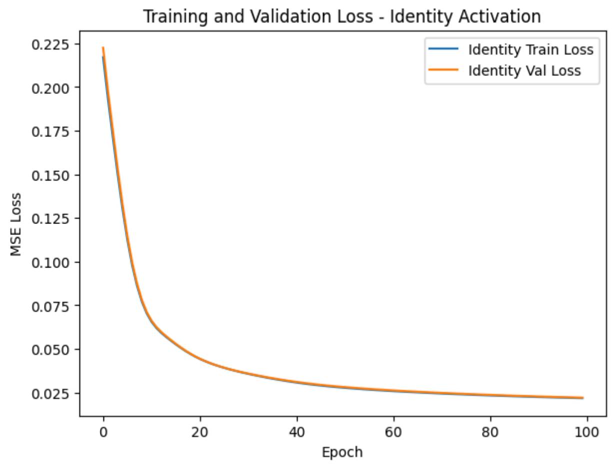
3. Using grid search, the best hyperparameters for the **MLPClassifier** with the **Tanh** activation function were found to be:

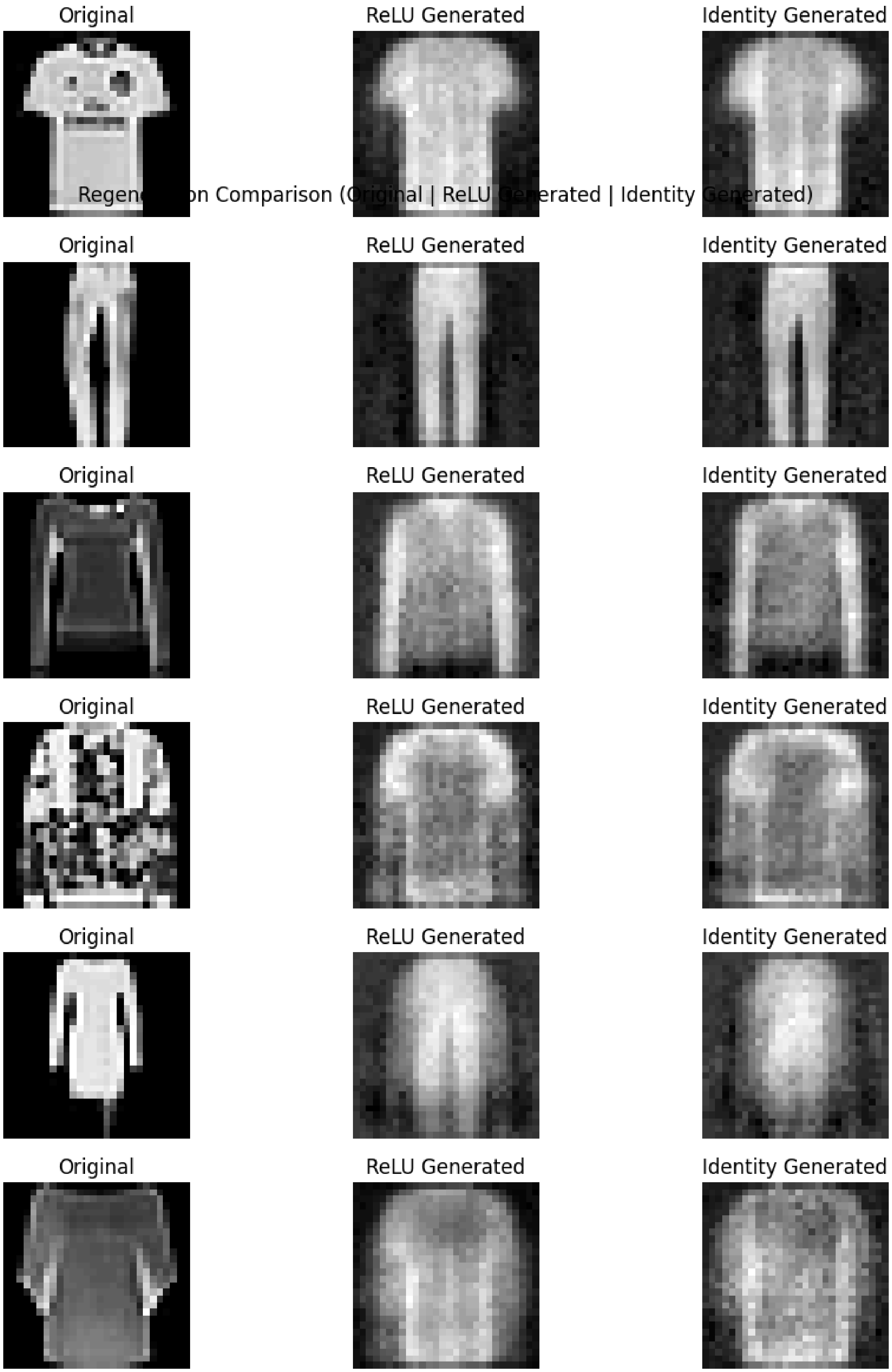
- **Solver**: `adam`
- **Learning Rate (learning_rate_init)**: `0.0001`
- **Batch Size**: `128`

The model achieved a best **log loss** score of approximately `0.4421`.

4.


Training and Validation Loss - ReLU Activation
ReLU Train Loss
ReLU Val Loss
MSE Loss
Epoch
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0
20
40
60
80
100


Training and Validation Loss - Identity Activation
Identity Train Loss
Identity Val Loss
MSE Loss
Epoch
0.225
0.200
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0
20
40
60
80
100


Regeneration Comparison (Original | ReLU Generated | Identity Generated)
Original
ReLU Generated
Identity Generated
Original
ReLU Generated
Identity Generated
Original
ReLU Generated
Identity Generated
Original
ReLU Generated
Identity Generated
Original
ReLU Generated
Identity Generated
Original
ReLU Generated
Identity Generated

5.

ReLU-based feature classifier accuracy: 75.40%

Identity-based feature classifier accuracy: 75.00%

In Part 2, we trained a classifier directly on the original dataset using multiple activation functions and found that the **Tanh activation function provided the best generalization** (lowest final validation loss of 0.4612). Now, after using feature vectors from a reconstruction-based approach (using an MLPRegressor as a feature extractor), both the ReLU and Identity-based classifiers achieved approximately **75% accuracy** on the classification task.

Reasons Why the Feature Vector Approach Provides a Decent Classifier

1. **Feature Extraction and Dimensionality Reduction**:
    - The MLPRegressor models used in the feature extraction step were trained to learn a lower-dimensional representation of the images. By regenerating the input, the models have extracted the most relevant features that capture underlying patterns in the data. This process is similar to using a **bottleneck layer** in an autoencoder, where only the most significant aspects of the input data are retained.
    - This lower-dimensional feature vector helps the classifier focus on the essential attributes of each image, reducing noise and irrelevant information, which often improves classification performance.
2. **Generalization through Pretrained Representations**:
    - The feature vectors obtained from the reconstruction models are likely more **generalized representations**. The MLPRegressor trained to recreate input images would have learned broader data patterns rather than focusing on details specific to classification.
    - When these feature vectors are used as inputs, the smaller classifiers built on top of them benefit from this pretrained representation, which helps them achieve decent performance with a simpler architecture (two layers of size aaa).
3. **Comparison of Classifier Complexity**:
    - In Part 2, the classifiers trained directly on the raw image data needed three layers (sizes 128, 64, and 32) to achieve optimal performance. However, by using the pretrained feature vector, you can reduce the classifier complexity to two layers (each of size aaa) and still achieve a similar level of performance.
    - This suggests that the feature vectors provide a robust, compact representation that simplifies the classification task and reduces the need for deeper networks.
4. **Reduced Overfitting**:
    - By training on the feature vectors rather than the original high-dimensional image data, there is likely a reduction in overfitting. The smaller classifiers trained on the extracted features are less complex and, therefore, less likely to memorize training data, which leads to better generalization on the test set.
5. **Comparison of Validation Loss**:

- The validation loss comparison from Part 2 indicated that **Tanh was the best activation** for direct image classification. However, both ReLU and Identity activations achieved similar performance when using extracted features, suggesting that with a generalized feature representation, the specific choice of activation function may have less impact.
- In essence, the feature extraction has allowed both ReLU and Identity classifiers to perform comparably well, whereas the direct approach required Tanh for the best performance. This indicates that learning a compressed representation first can reduce dependency on specific activation functions for effective classification.