CSE343: Machine Learning
Assignment-3

Himanshu Raj (2022216)
November 9, 2024

Section B
In this section, we implemented a neural network from scratch using the NumPy package available in Python. The implemented neural network was then tested on the MNIST dataset with different activation functions and weight initialization functions.

1) The neural network classifier is implemented as a class named NeuralNetwork.

The objects of this class need the following parameters during initialization:
N : number of (hidden) layers in the network
A : list of size N specifying the number of neurons in each hidden layer
lr : learning rate
activation : activation function for hidden layers
activation_grad : gradient function for the same activation function
weight_init : weight initialization function to initialize weights
epochs : maximum number of iterations to train the model
batch_size : batch size to use while training the model on the dataset

The class has the following functions:
_softmax : used to apply softmax activation function on the output layer
forward_propagation : does forward propagation and returns the output of the last layer
backward_propagation : does backward propagation and update weights and biases
fit : trains or fits the model on the training data
predict_prob : predict probabilities for all classes (does forward propagation)
predict : predicts the final class for each data point
score : computes accuracy of the trained model
plot_losses : plots training loss and validation loss against epochs

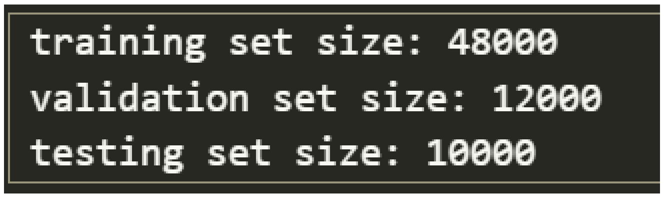
Optional early stopping is also implemented to avoid overfitting of the model on the training set. By default, early stopping is enabled. It can disabled by explicitly setting the `early_stopping` parameter of the fit function as false. If the validation loss increases for a certain number (defined in the `patience` parameter of the fit function) of iterations, we can stop the training early.

2) Activation Functions
sigmoid, tanh, ReLU and Leaky ReLU can be used as the activation functions for the neural network by passing in the activation parameter. Their respective gradients also need to be passed in the activation_grad parameter. softmax activation is used for the last layer, and it is implemented as a static method inside the NeuralNetwork class.

3) Weight Initialization Functions
zero initialization, random initialization and normal initialization can be used as weight initialization functions for the network by passing in the weight_init parameter. The functions are implemented along with appropriate scaling factors.

4) Testing NeuralNetwork on the MNIST dataset

Loaded the dataset and visualised a few samples from the dataset.
Preprocessing includes flattening the features from a 28x28 matrix to a 784-sized 1D array and then normalizing the values in the range [0,1] by dividing by 255.
One-hot encoding is performed on labels as the implementation is for a classifier, so it finds the probabilities for each class for a given data point.
The training set is split to create a validation set, train:validation = 80:20.



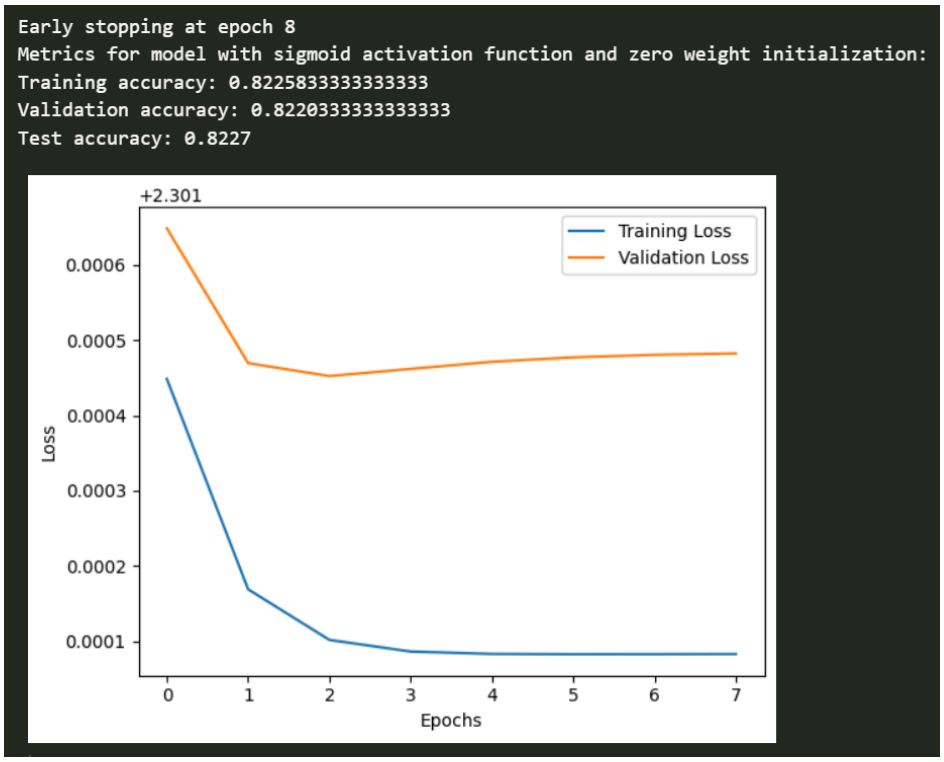
```
training set size: 48000
validation set size: 12000
testing set size: 10000
```

The models are trained on all combinations of activation functions and weight initialization functions. The hidden layer sizes are set as [256, 128, 64, 32], trained for 100 epochs with a constant learning rate of 0.002 with a batch size of 128.

The models with sigmoid activation function reach their convergence at 5th iteration.
The models with tanh activation function reach their convergence at 30th iteration.
The models with ReLU activation function reach their convergence at 25th iteration.
The models with leaky ReLU activation function reach their convergence at 20th iteration.
The models with zero weight initialization always stop early indicating they tend to overfit in further iterations.

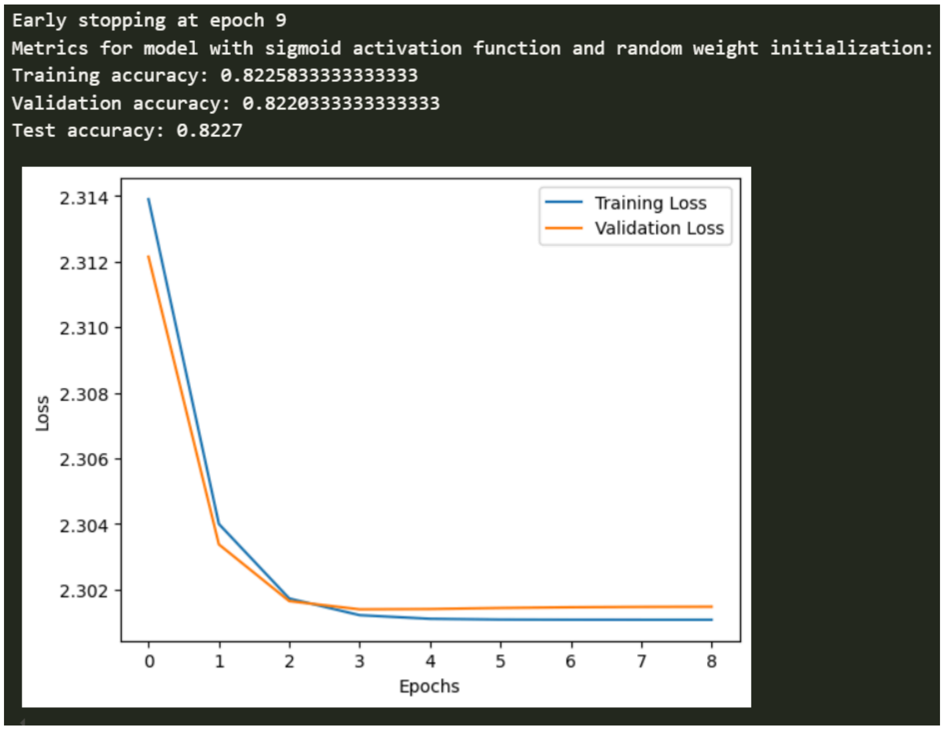
The most optimal combination is ReLU activation with normal weight initialization because of the stable and uniform descent in loss curve and no overfitting tendency, so it will generalize better on unseen data.

Below are the plots and accuracies of all the trained models.



```
Early stopping at epoch 8
Metrics for model with sigmoid activation function and zero weight initialization:
Training accuracy: 0.8225833333333333
Validation accuracy: 0.8220333333333333
Test accuracy: 0.8227
```

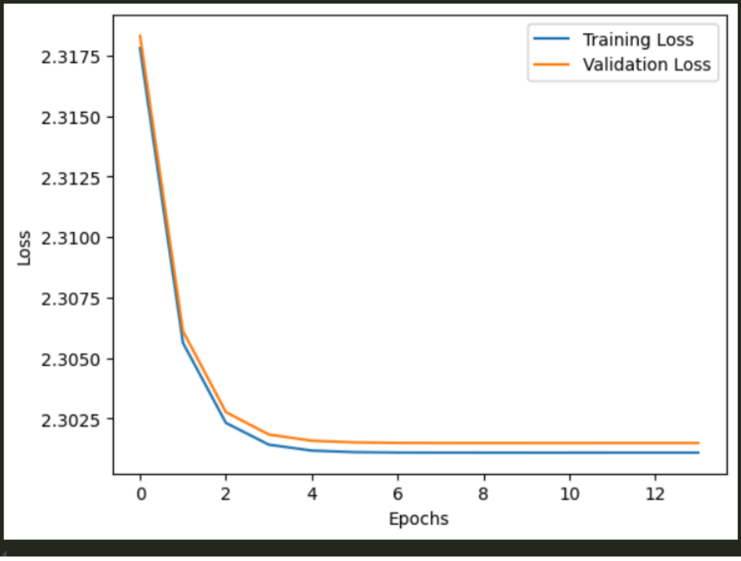
sigmoid activation with zero weight initialization



```
Early stopping at epoch 9
Metrics for model with sigmoid activation function and random weight initialization:
Training accuracy: 0.8225833333333333
Validation accuracy: 0.8220333333333333
Test accuracy: 0.8227
```

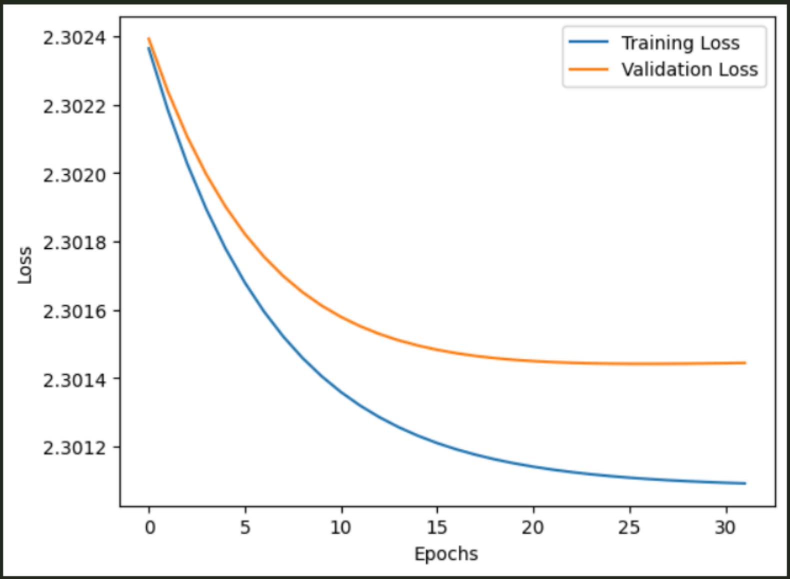
sigmoid activation with random weight initialization

```
Early stopping at epoch 14
Metrics for model with sigmoid activation function and normal weight initialization:
Training accuracy: 0.8225833333333333
Validation accuracy: 0.8220333333333333
Test accuracy: 0.8227
```

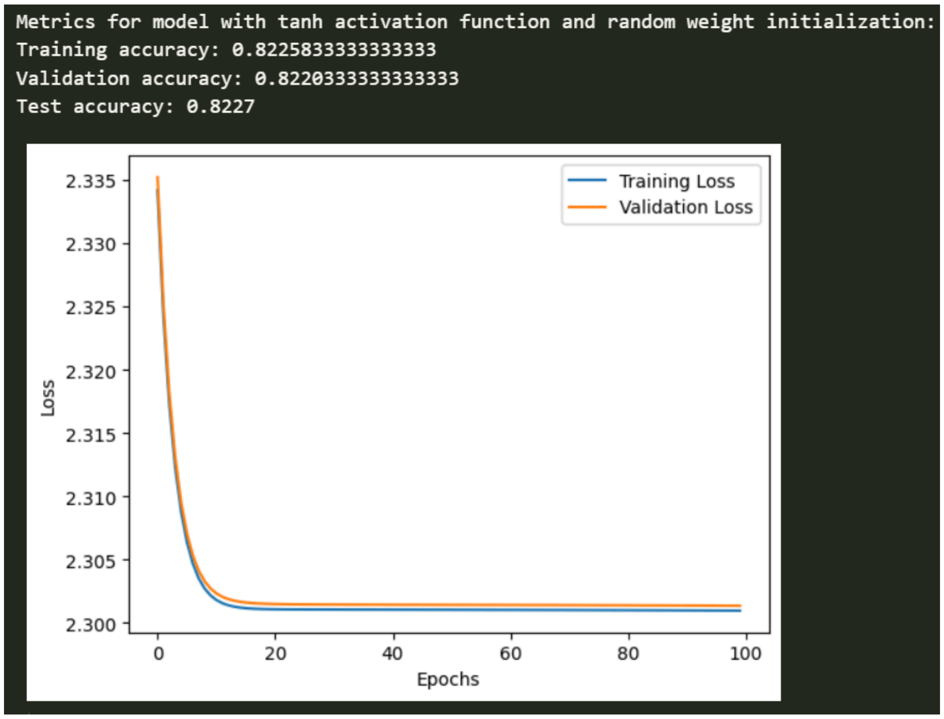


sigmoid activation with normal weight initialization

```
Early stopping at epoch 32
Metrics for model with tanh activation function and zero weight initialization:
Training accuracy: 0.8225833333333333
Validation accuracy: 0.8220333333333333
Test accuracy: 0.8227
```

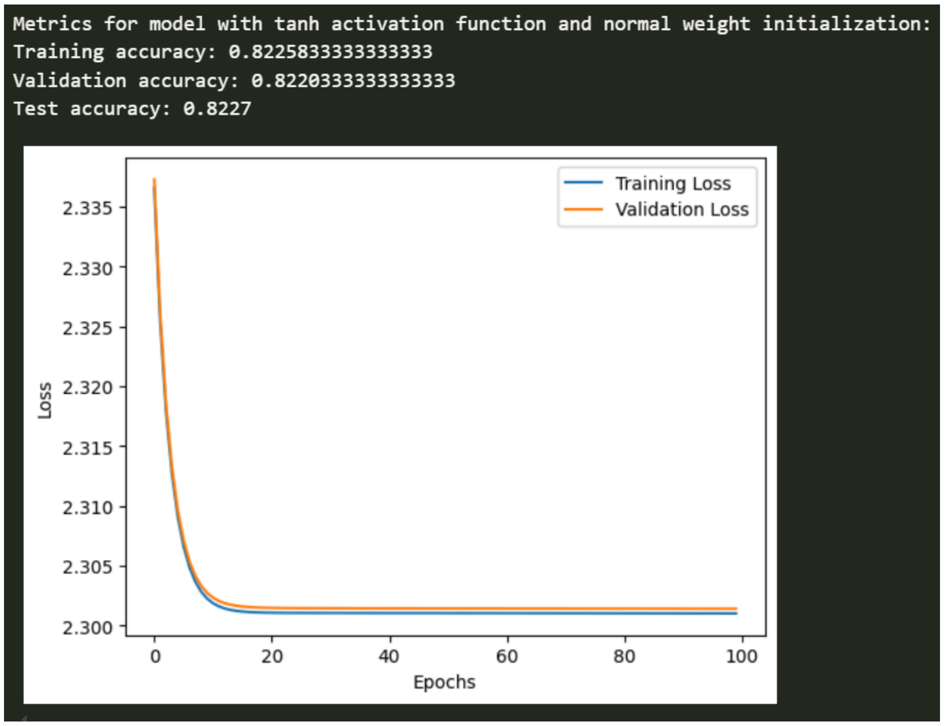


tanh activation with zero weight initialization



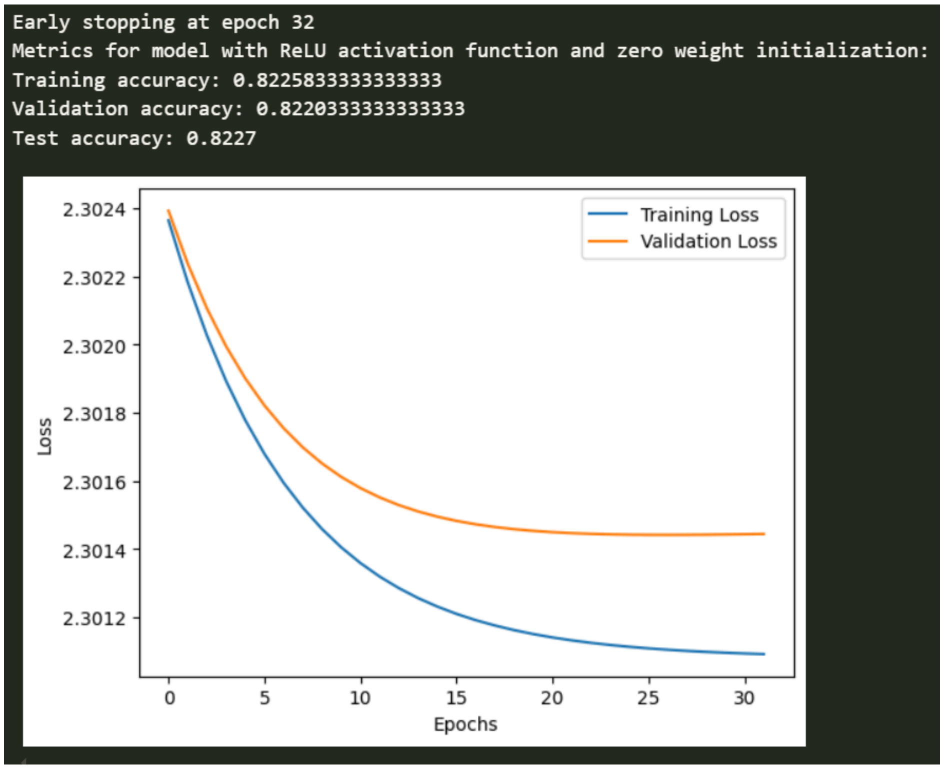
```
Metrics for model with tanh activation function and random weight initialization:
Training accuracy: 0.8225833333333333
Validation accuracy: 0.8220333333333333
Test accuracy: 0.8227
```

tanh activation with random weight initialization

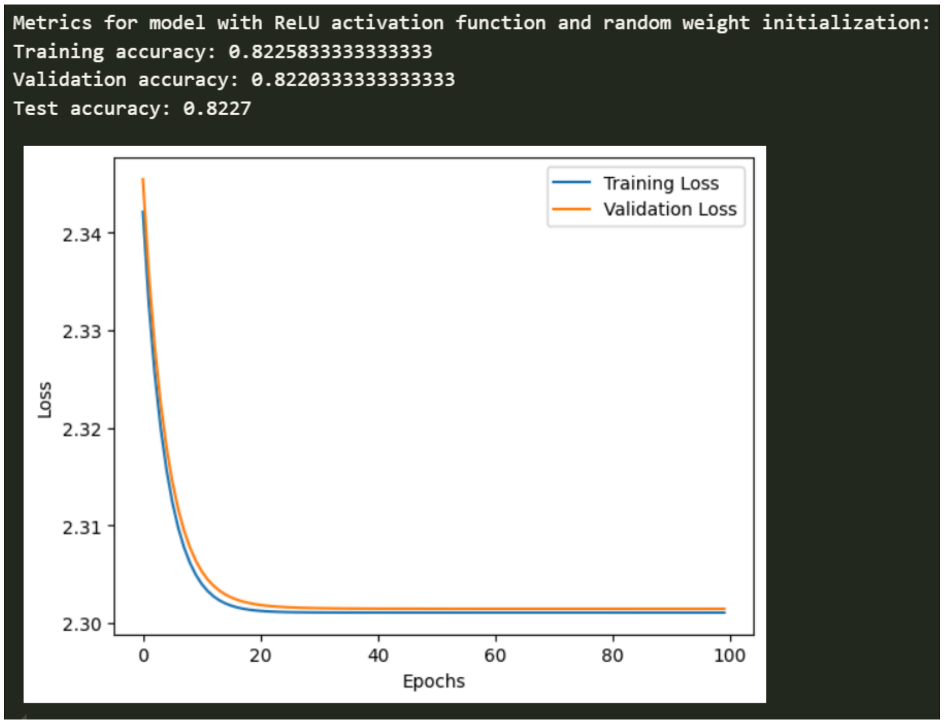


```
Metrics for model with tanh activation function and normal weight initialization:
Training accuracy: 0.8225833333333333
Validation accuracy: 0.8220333333333333
Test accuracy: 0.8227
```

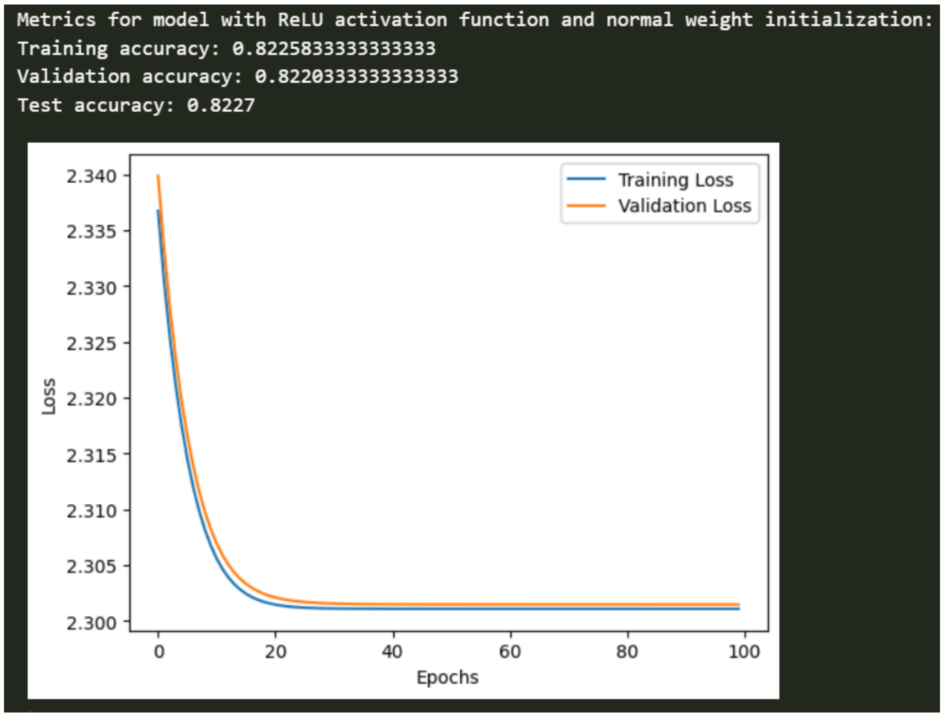
tanh activation with normal weight initialization


```
Early stopping at epoch 32
Metrics for model with ReLU activation function and zero weight initialization:
Training accuracy: 0.8225833333333333
Validation accuracy: 0.8220333333333333
Test accuracy: 0.8227
```

ReLU activation with zero weight initialization

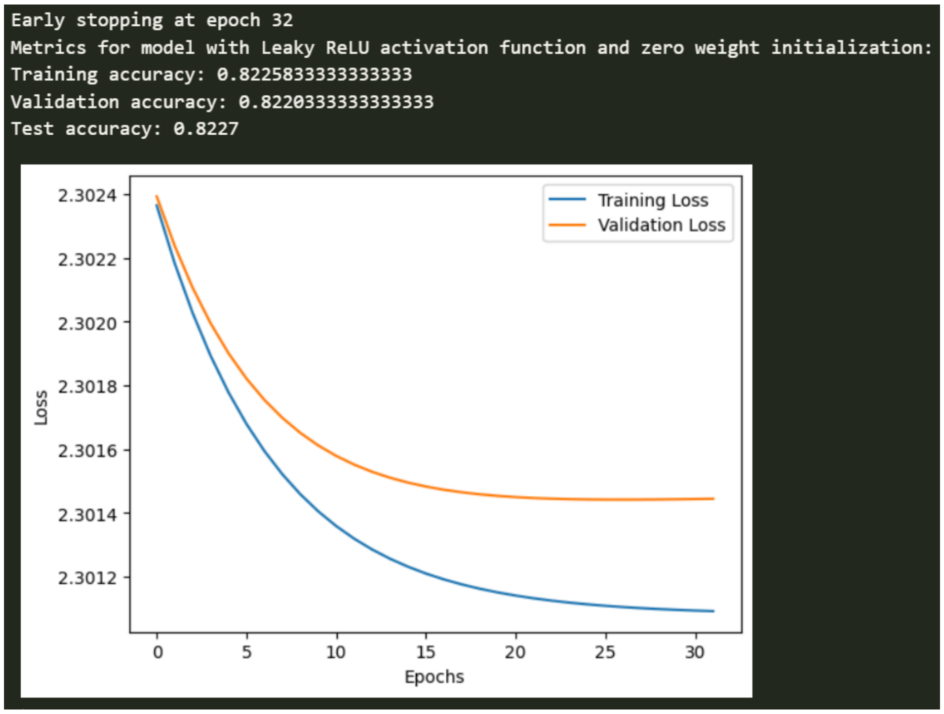

```
Metrics for model with ReLU activation function and random weight initialization:
Training accuracy: 0.8225833333333333
Validation accuracy: 0.8220333333333333
Test accuracy: 0.8227
```

ReLU activation with random weight initialization



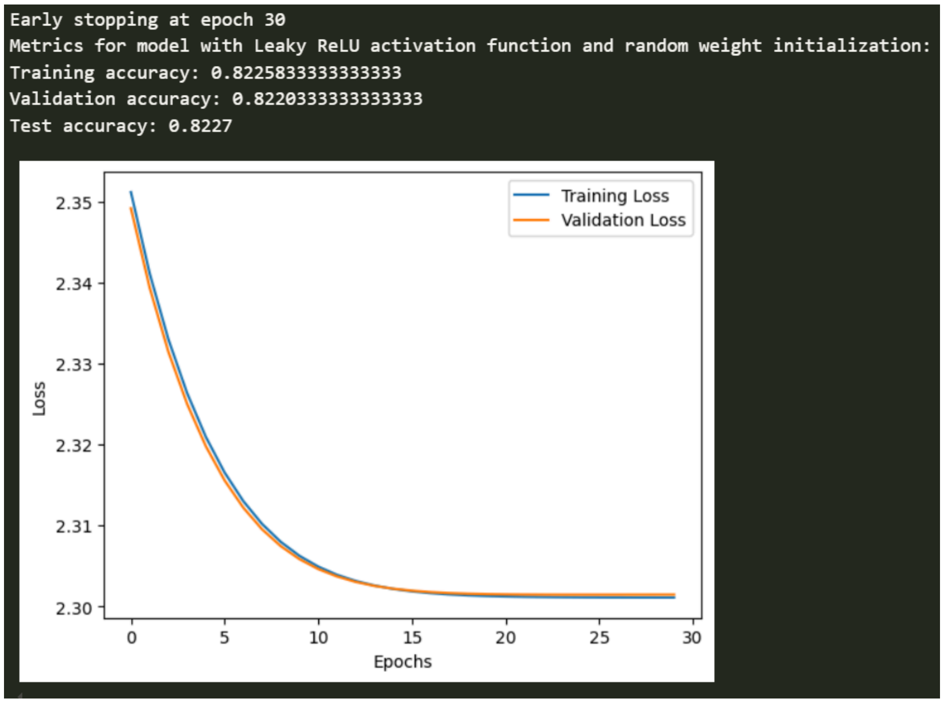
```
Metrics for model with ReLU activation function and normal weight initialization:
Training accuracy: 0.8225833333333333
Validation accuracy: 0.8220333333333333
Test accuracy: 0.8227
```

ReLU activation with normal weight initialization



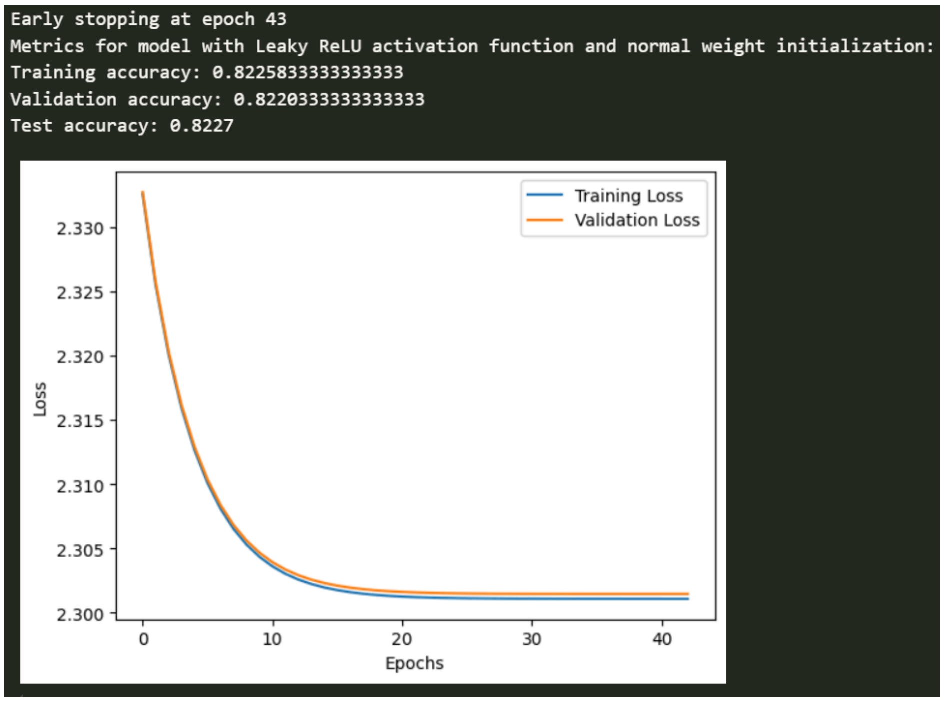
```
Early stopping at epoch 32
Metrics for model with Leaky ReLU activation function and zero weight initialization:
Training accuracy: 0.8225833333333333
Validation accuracy: 0.8220333333333333
Test accuracy: 0.8227
```

Leaky ReLU activation with zero weight initialization



```
Early stopping at epoch 30
Metrics for model with Leaky ReLU activation function and random weight initialization:
Training accuracy: 0.8225833333333333
Validation accuracy: 0.8220333333333333
Test accuracy: 0.8227
```

Leaky ReLU activation with random weight initialization



```
Early stopping at epoch 43
Metrics for model with Leaky ReLU activation function and normal weight initialization:
Training accuracy: 0.8225833333333333
Validation accuracy: 0.8220333333333333
Test accuracy: 0.8227
```

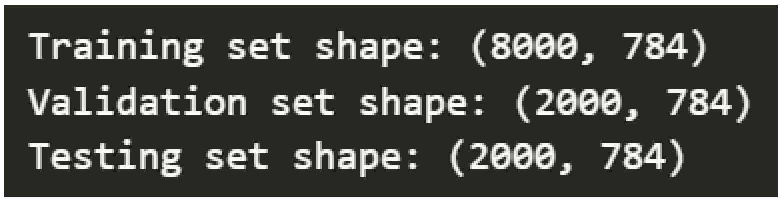
Leaky ReLU activation with normal weight initialization

Section C

In this section, we will use the MLPClassifier and MLPRegressor from the sklearn package to train a neural network on the Fashion-MNIST dataset.

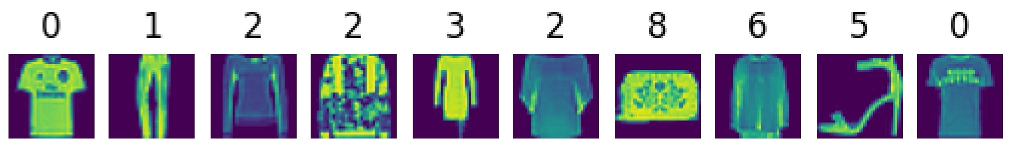
1) Preprocessing and Visualization

The features are already flattened into a 1D array of size 784. Normalize the values in the range [0,1] by dividing by 255. There are 8000 data points in the training set and 2000 data points in the testing and validation set.



```
Training set shape: (8000, 784)
Validation set shape: (2000, 784)
Testing set shape: (2000, 784)
```

This is the visualization of 10 images from the test dataset with their labels.

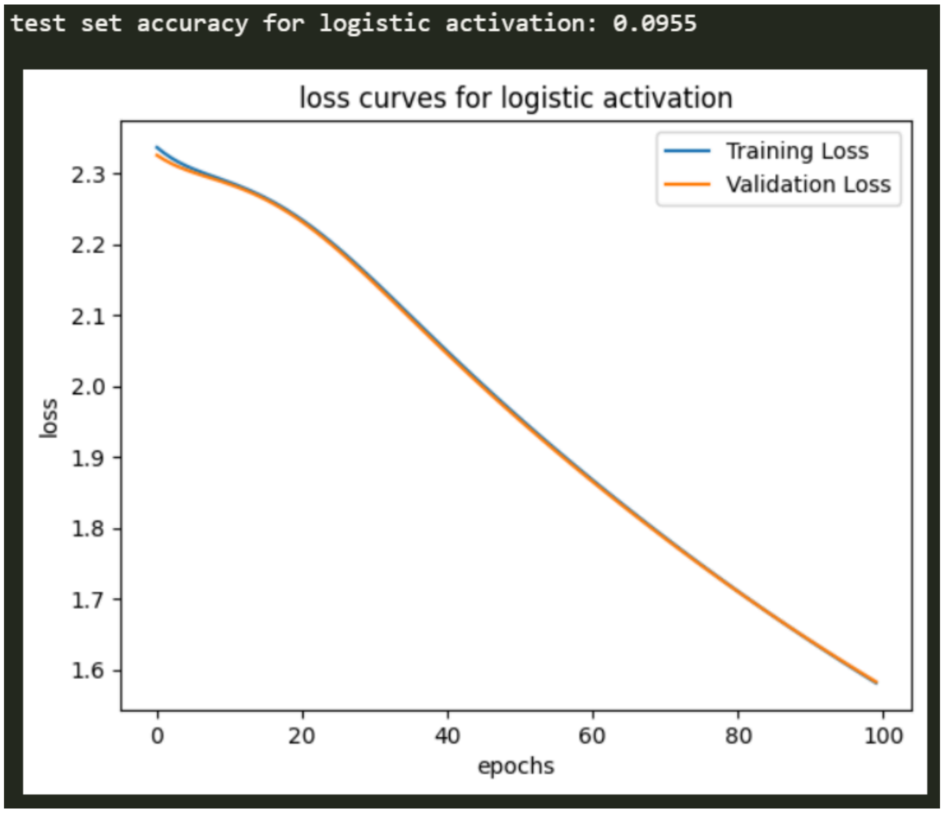


2) Testing models on Fashion-MNIST dataset

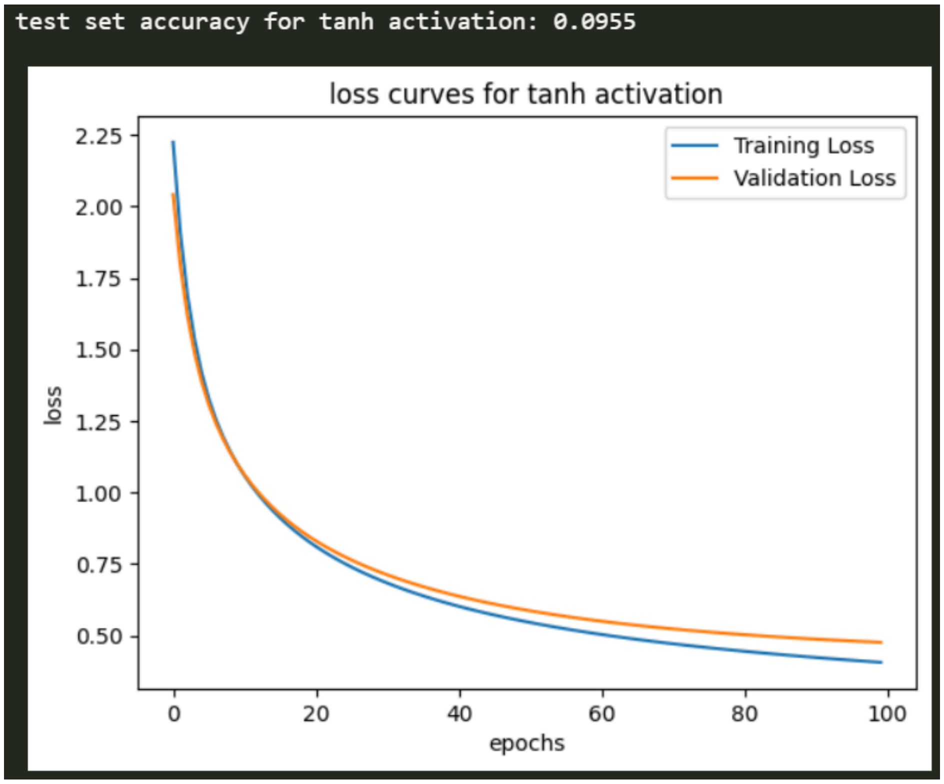
The models are trained with hidden layer sizes set as [128, 64, 32] for 100 epochs with a constant learning rate of 0.00002 with a batch size of 128 using adam solver and random seed set as 42.
We train a model for each activation function, i.e. logistic, tanh, ReLU and identity.

ReLU activation gave the best performance on test set with an accuracy of 9.7 while others had an accuracy of 9.55

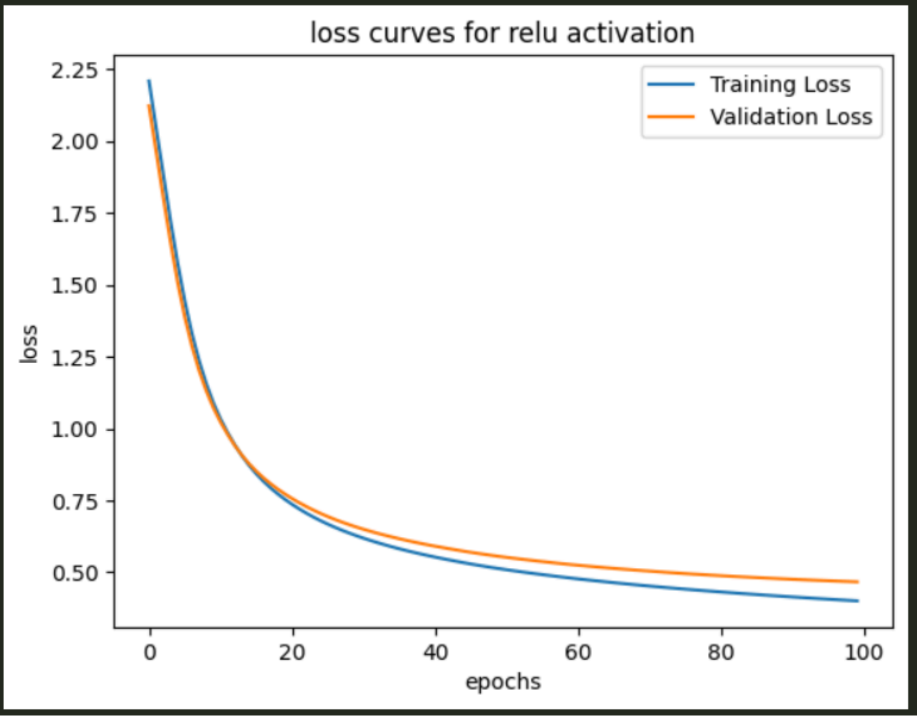
Below are the plots and accuracies of all the trained models.



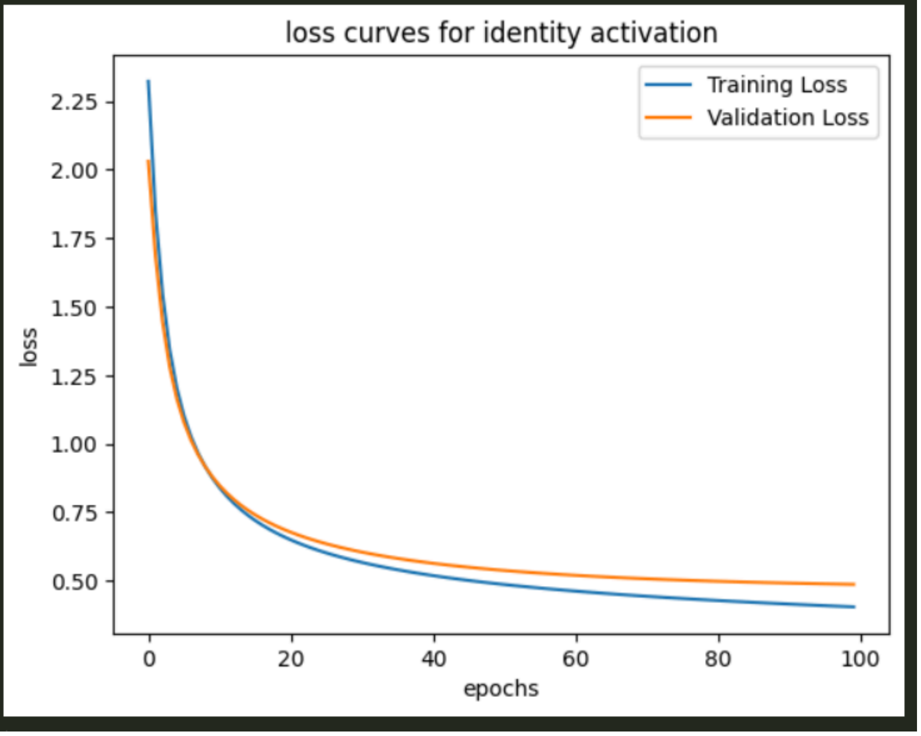
```
test set accuracy for logistic activation: 0.0955
```



```
test set accuracy for tanh activation: 0.0955
test set accuracy for relu activation: 0.097
```

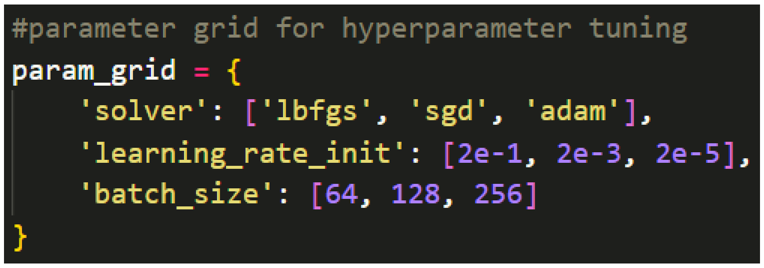


```
test set accuracy for identity activation: 0.0955
```



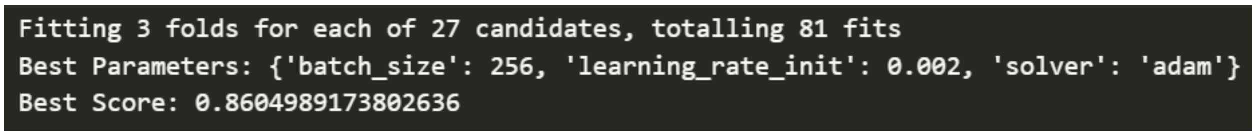
3) Hyperparameter Tuning with Grid Search

Grid search to find the best hyperparameters for the model with best activation function, i.e. ReLU activation. We set hidden layer size as [128, 64, 32], epochs as 100, 3-fold cross validation and scoring parameter as accuracy. We try to find the best solver, learning_rate and batch_size.



```
#parameter grid for hyperparameter tuning
param_grid = {
    'solver': ['lbfgs', 'sgd', 'adam'],
    'learning_rate_init': [2e-1, 2e-3, 2e-5],
    'batch_size': [64, 128, 256]
}
```

The best parameters are

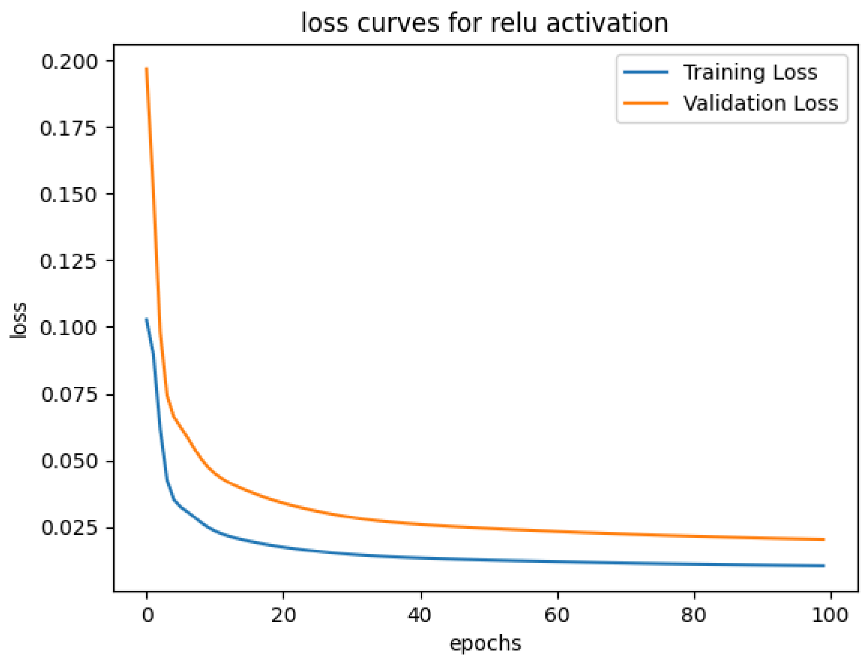


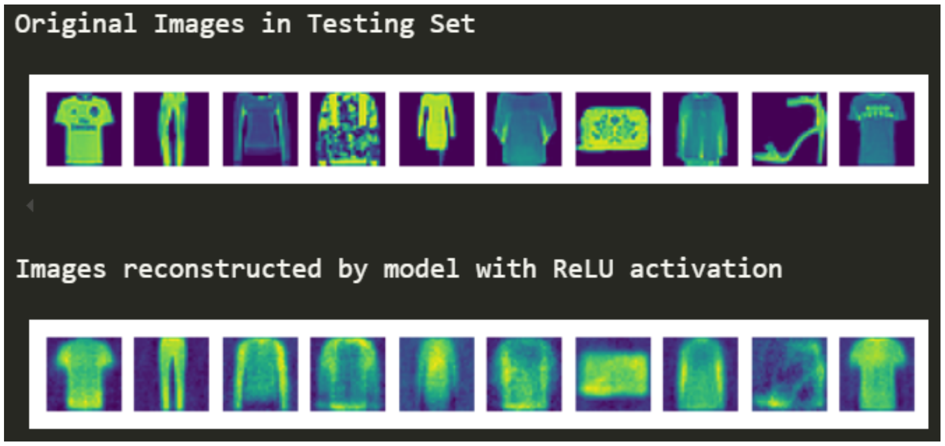
```
Fitting 3 folds for each of 27 candidates, totalling 81 fits
Best Parameters: {'batch_size': 256, 'learning_rate_init': 0.002, 'solver': 'adam'}
Best Score: 0.8604989173802636
```

4) Regenerating images with MLPRegressor

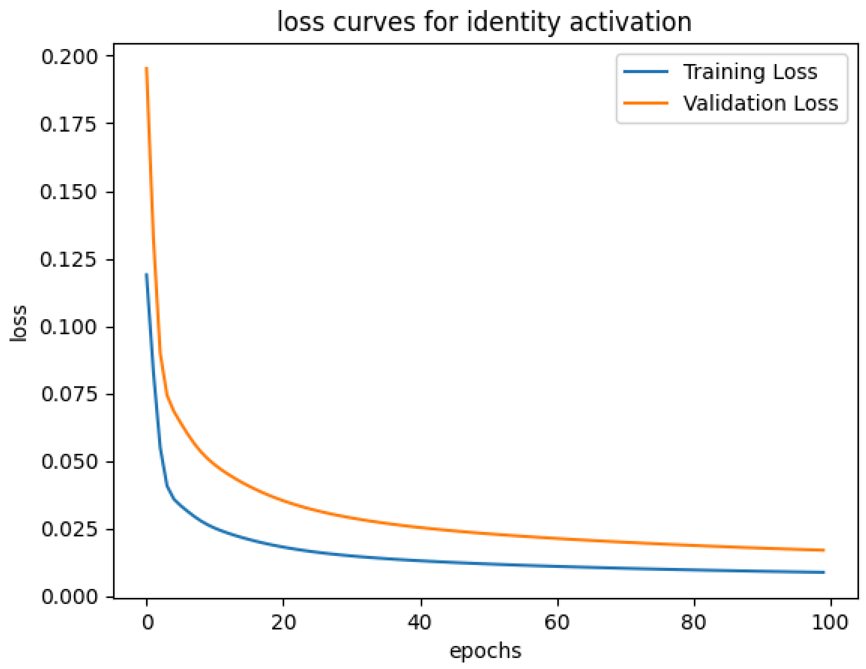
The regressor models are trained with hidden layer sizes set as [256, 128, 64, 128, 256] for 100 epochs with a constant learning rate of 0.00002 with a batch size of 128 using adam solver and random seed set as 42. We use mean squared error to find validation loss.
We train a model for ReLU and identity activation functions.

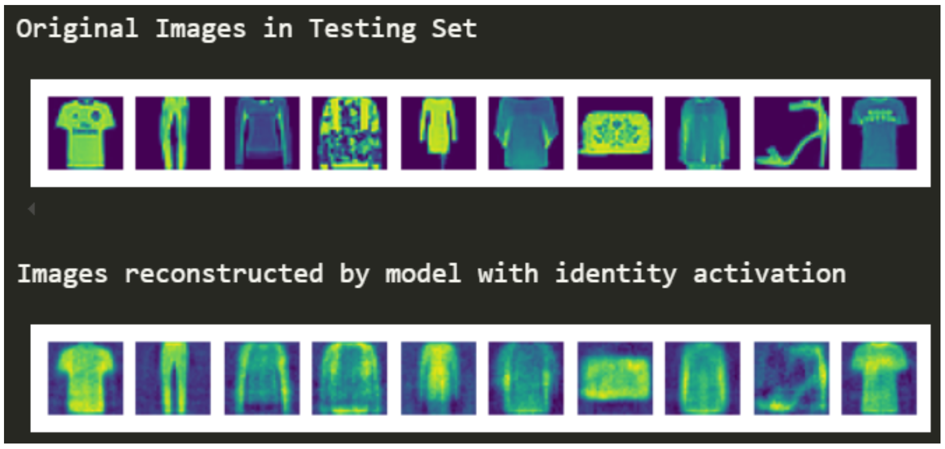
The models are able to capture the features really well and the regenerated images are similar at least visually albeit a bit blurry but they can be assigned a label with a human eye.




```
Original Images in Testing Set

Images reconstructed by model with ReLU activation
```

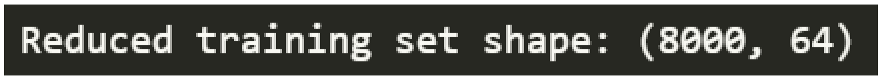



```
Original Images in Testing Set
```

```
Images reconstructed by model with identity activation
```

5) Training Classifiers on features extracted from Regressors

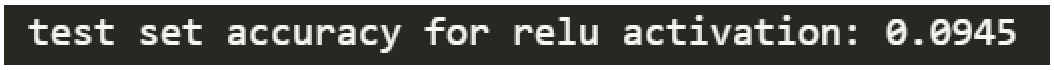
Extract feature vectors from the above trained MLPs of size 64 and train new mini classifiers on these new features.



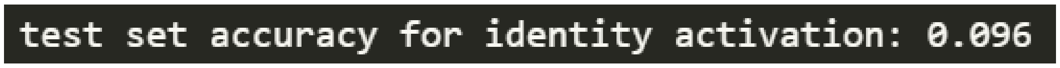
```
Reduced training set shape: (8000, 64)
```

The classifier models are trained with hidden layer sizes set as [64, 64] for 200 epochs with a constant learning rate of 0.00002 with a batch size of 128 using adam solver and random seed set as 42.
We train a model for ReLU and identity activation functions.

Below are the accuracies of these new models on the test set.



```
test set accuracy for relu activation: 0.0945
```



```
test set accuracy for identity activation: 0.096
```

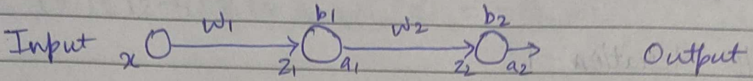
These accuracies are almost similar to what we obtained in part 2, where we considered 784 features.
The features were extracted from a middle layer from an image regeneration regressor and the hidden layers act as feature extractors and discard redundant information and discard outliers. These extracted features essentially represent the original dataset very closely and that is why we get a decent classifier even after the feature size is reduced from 784 (28x28) to 64(8x8).

Himanshu Raj (2022216)

Date: / /
Page No.

ML Asgn-3 Sec A

Q1



Let z_i be the output pre-activation and a_i be output post-activation. So a_2 will be the output and x is input and let y_i be the ~~[illegible]~~ target for x_i.

eq^n for forward propagation -

$$w_1 x_i + b_1 = z_1$$
$$ReLU(z_1) = a_1 = \max(0, z_1)$$
$$w_2 a_1 + b_2 = z_2$$
$$z_2 = a_2$$

We find MSE loss for each data point with eq^n -

$$L = \frac{1}{2}(y_{pred} - y_{true})^2 = \frac{1}{2}(a_2 - y_i)^2$$

eq^n for backward propagation -

$$w_1 = w_1 - \eta \frac{\partial L}{\partial w_1} ; \quad w_2 = w_2 - \eta \frac{\partial L}{\partial w_2} ; \quad b_1 = b_1 - \eta \frac{\partial L}{\partial b_1} ; \quad b_2 = b_2 - \eta \frac{\partial L}{\partial b_2}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial w_2} = (a_2 - y_i) \cdot 1 \cdot a_1$$

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial b_2} = (a_2 - y_i) \cdot 1 \cdot 1$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \frac{\partial a_1}{\partial z_1} \frac{\partial z_1}{\partial w_1} = (a_2 - y_i) \cdot 1 \cdot w_2 \cdot \begin{pmatrix} 1 \text{ if } z_1 > 0 \\ \text{else } 0 \end{pmatrix} \cdot x_i$$

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial b_2} = (a_2 - y_i) \cdot 1 \cdot w_2 \cdot \begin{pmatrix} 1 \text{ if } z_1 > 0 \\ \text{else } 0 \end{pmatrix} \cdot 1$$

Assuming $w_1 = w_2 = 1$ and $b_1 = b_2 = 0$.
$\eta = 0.01$

Iteration - 1

- Sample - 1

$y_1 = 3$, $x_1 = 1$, $z_1 = 1 \times 1 + 0 = 1$, $a_1 = \max(0, 1) = 1$
$z_2 = 1 \times 1 + 0 = 1$, $a_2 = 1$, $L_1 = \frac{1}{2}(1-3)^2 = 2$

~~$\frac{\partial L}{\partial w_2} = (2-3) \cdot 1 \cdot 1 = -1$, $\frac{\partial L}{\partial b_2} = (2-3) \cdot 1 \cdot 1 = -1$~~

~~$\frac{\partial L}{\partial w_1} = (2-3) \cdot 1 \cdot 1 \cdot 1 \cdot 1 = -1$, $\frac{\partial L}{\partial b_1} = (2-3) \cdot 1 \cdot 1 \cdot 1 = -1$~~

~~$w_1 = 1 - 0.01 \times -1 = 1.01$, $b_1 = 0 - 0.01 \times -1 = 0.01$~~
~~$w_2 = 1 - 0.01 \times -1 = 1.01$, $b_2 = 0 - 0.01 \times -1 = 0.01$~~

$\frac{\partial L}{\partial w_2} = (1-3) \cdot 1 \cdot 1 = -2$, $\frac{\partial L}{\partial b_2} = (1-3) \cdot 1 \cdot 1 = -2$

$\frac{\partial L}{\partial w_1} = (1-3) \cdot 1 \cdot 1 \cdot 1 \cdot 1 = -2$, $\frac{\partial L}{\partial b_1} = (1-3) \cdot 1 \cdot 1 \cdot 1 \cdot 1 = -2$

$w_1 = 1 - 0.01 \times -2 = \underline{1.02}$, $b_1 = 0 - 0.01 \times -2 = \underline{0.02}$
$w_2 = 1 - 0.01 \times -2 = \underline{1.02}$, $b_2 = 0 - 0.01 \times -2 = \underline{0.02}$

- Sample - 2

$x_2 = 2$, $y_2 = 4$, $z_1 = 1.02 \times 2 + 0.02 = 2.06$, ~~$a_1 = 2.06$~~
$z_2 = 1.02 \times 2.06 + 0.02 = 2.1212$ ~~$, a_2 = 2.06$~~, $a_1 = \max(0, 2.06) = 2.06$
$a_2 = 2.1212$, $L_2 = \frac{1}{2}(2.1212 - 4)^2 = \frac{3.5299}{2} = 1.7649$

$\frac{\partial L}{\partial w_2} = (2.1212 - 4) \times 1 \times 2.06 = -3.8703$

$\frac{\partial L}{\partial b_2} = (2.1212 - 4) \times 1 \times 1 = -1.8788$

Date : / /
Page No.

$\frac{\partial L}{\partial w_1} = (2.1212-4) \times 1 \times 1.02 \times 1 \times 2 = -3.8328$

$\frac{\partial L}{\partial b_1} = (2.1212-4) \times 1 \times 1.02 \times 1 \times 1 = -1.9164$

$w_1 = 1.02 - 0.01 \times -3.8328 = \underline{1.0583}$
$w_2 = 1.02 - 0.01 \times -3.8703 = \underline{1.0587}$
$b_1 = 0.02 - 0.01 \times -1.8788 = \underline{0.0389}$
$b_2 = 0.02 - 0.01 \times -1.9164 = \underline{0.0391}$

-sample-3

$x_3 = 3$, $y_3 = 5$, $z_1 = 1.0583 \times 3 + 0.0389 = 3.2138$,
$a_1 = 3.2138$, $z_2 = 1.0587 \times 3.2138 + 0.0391 = 3.4416$,
$a_2 = 3.4416$, $L_2 = \frac{1}{2}(3.4416-5)^2 = 1.2143$

$\frac{\partial L}{\partial w_2} = (3.4416-5) \times 1 \times 3.2138 = -5.0085$

$\frac{\partial L}{\partial b_2} = (3.4416-5) \times 1 \times 1 \quad = -1.5584$

$\frac{\partial L}{\partial w_1} = (3.4416-5) \times 1 \times 1.0587 \times 1 \times 3 = -4.94996$

$\frac{\partial L}{\partial b_1} = (3.4416-5) \times 1 \times 1.0587 \times 1 \times 1 = -1.6499$

$w_1 = 1.0583 - 0.01 \times -4.9496 = \underline{1.1078}$
$w_2 = 1.0587 - 0.01 \times$ ~~1.6499~~ $-5.0085 = \underline{1.1087}$
$b_1 = 0.0389 - 0.01 \times$ ~~5.0085~~ $-1.6499 = \underline{0.0554}$
$b_2 = 0.0391 - 0.01 \times -1.5584 = \underline{0.0547}$

Date: / /
Page No.

Q3 $w_1 = -2, \; w_2 = 0, \; b = 5$

a The margin of classifier is defined as the perpendicular dist b/w support vector and the decision boundary.
↳ Stated in Bishop's ML book

$$\text{margin} = \frac{1}{\|w\|}$$
$$= \frac{1}{\sqrt{(-2)^2 + 0^2}} = \frac{1}{\sqrt{4}} = \frac{1}{2}$$

b Support vectors lie on $w^T x + b = \pm 1$.
$w_1 x_1 + w_2 x_2 + b = \pm 1$

1 - $-2 \times 1 + 0 + 5 = 3$
2 - $-2 \times 2 + 0 + 5 = 1$ - SV
3 - $-2 \times 3 + 0 + 5 = -1$ - SV
4 - $-2 \times 4 + 0 + 5 = -3$

Sample 2 (2,3) is a SV belonging to class +1 and Sampl 3 (3,3) is a SV belonging to class -1.

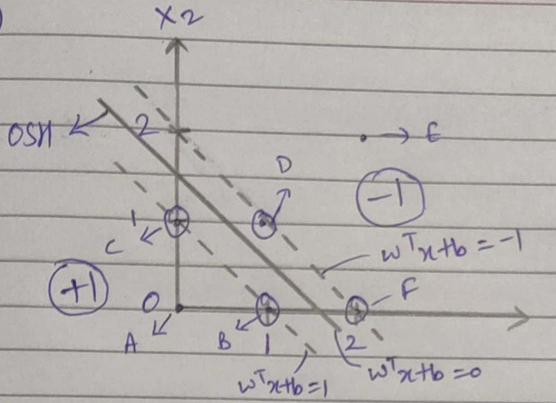
c $x_1 = 1, \; x_2 = 3$

$y_i = w^T x + b = w_1 x_1 + w_2 x_2 + b$
$= -2 \times 1 + 0 \times 3 + 5 = 3$

Since $y_i \geq 1$, the data point belongs to class +1.

Date : / /
Page No.

Q3 - a)



Yes, we ~~[illegible]~~ clearly see that the points are linearly separable from the plot above.

b Let the SVM hyperplane eq^n be $w^T x + b = 0 \Rightarrow$ $w_1 x_1 + w_2 x_2 + b = 0$. From the graph, we can infer that points A & E don't contribute to forming hyperplane. We try to connect remaing 2 points of each ~~by~~ class by a line and observe that the lines are parallel. This helps me to infer that these 4 points are my support vectors and will help me with my hyperplane.

We define SVs as $w^T x + b = 1$ for $+1$ class and $w^T x + b = -1$ for -1 class.

$$\left.\begin{aligned} &B - w_1(1) + w_2(0) + b = 1 \\ &C - w_1(0) + w_2(1) + b = 1 \\ &F - w_1(2) + w_2(0) + b = -1 \\ &D - w_1(1) + w_2(1) + b = -1 \end{aligned}\right\} \text{4 } eq^n \text{ and 3 variables}$$

Solving these, we get $w_1 = -2$, $w_2 = -2$, $b = 3$

w^T for max margin hyperplane is $[-2, -2]$.

Support vectors are points $\underbrace{B, C}_{+\text{class}}, \underbrace{D, F}_{-\text{class}}$.