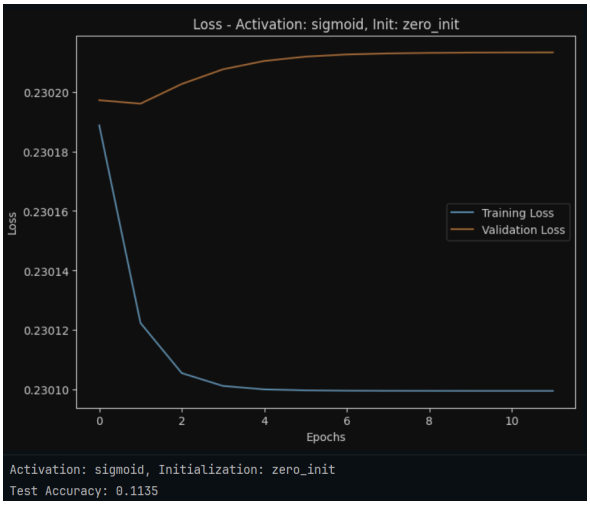
Machine Learning Assignment: 03

Section: B

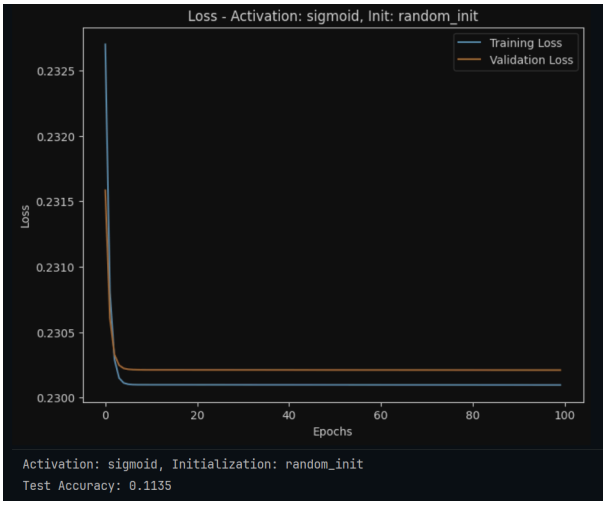
Activation Function: Sigmoid

(a)

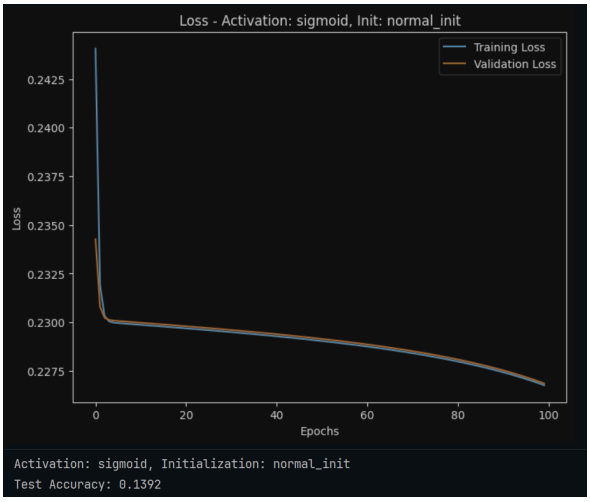
.



(b)



(c)



From all 3 Weight Initialization techniques used for sigmoid, The sigmoid activation performed poorly on the first two initialization techniques (Zero_Init and Random_Init), giving test accuracy of around 11.35%, and for normal initialization also, the model didn't perform well and experienced overfitting, which gave it accuracy 13.92%.

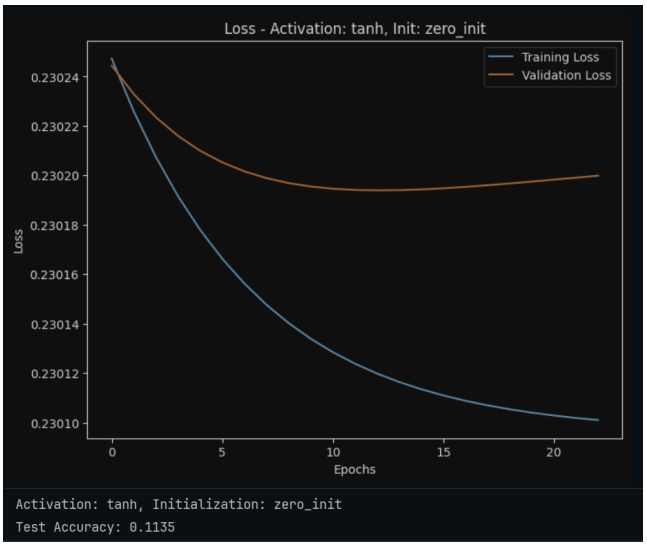
(a) The model experiences high variance and increment in validation accuracy, which results in early stopping because training loss decreases very fast, which shows overfitting, and the model is trained quickly on the training dataset.

(b) Similar results are shown in random initialisation's loss curve in which both training and validation loss decrease rapidly and show high variance as there is a gap between validation and training loss curves.
(c) The model got better results than zero init and random init but didn't perform well, as this model's loss curve also shows overfitting. Although variance was reduced from earlier examples, it shows high model complexity, which resulted in overfitting.

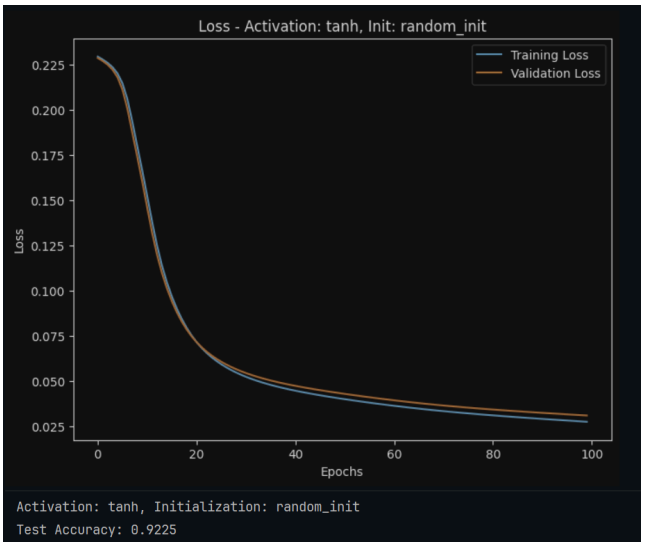
The sigmoid has performed suboptimally and worst among all the activation functions.

Activation Function: Tanh

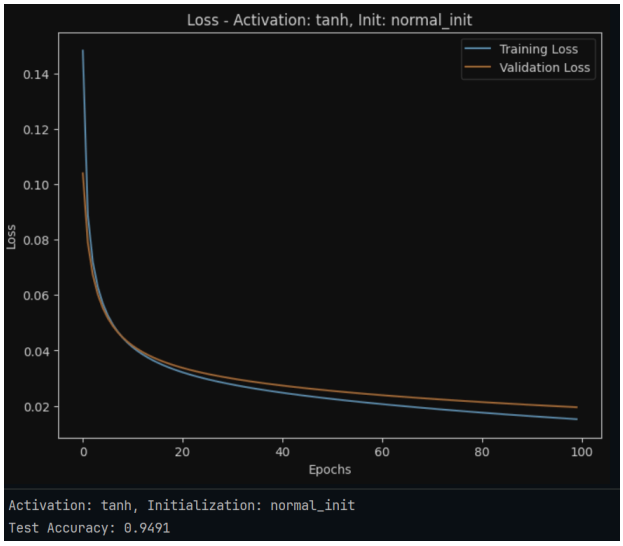
(a)



(b)



(c)



The tanh activation function has performed exceptionally well on random init and normal init and worst on zero init.

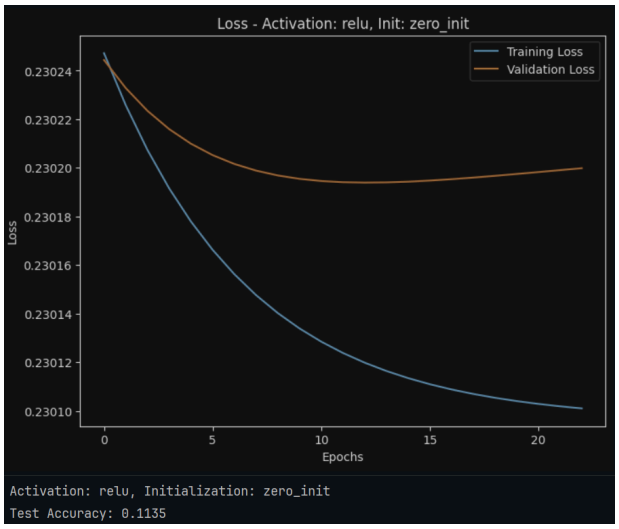
(a) In zero init, the weights didn't get modified a lot, which didn't improve the loss curve for validation and training, and ultimately, early stopping stops the training process. The model gave 11% accuracy, which is poor.

(b) In random init the model showed better results both validation loss and training loss started with high loss but eventually decreased with epochs and led to lower loss there is low variance across the epochs. The accuracy was around 92%.

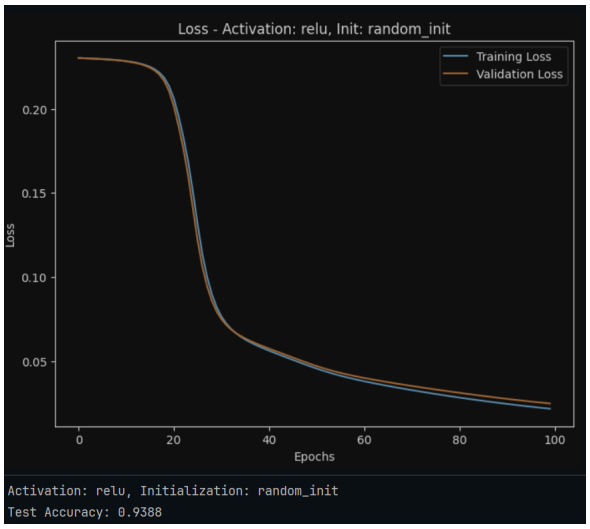
(c) In Normal init the model shows excellent performance the validation loss starts with less than training loss, which shows the variance in the beginning but achieves low loss as epoch increments the loss curve shows low bias and low variance, which ultimately gives accuracy around 94.91% better than random also.

Activation Function: Relu

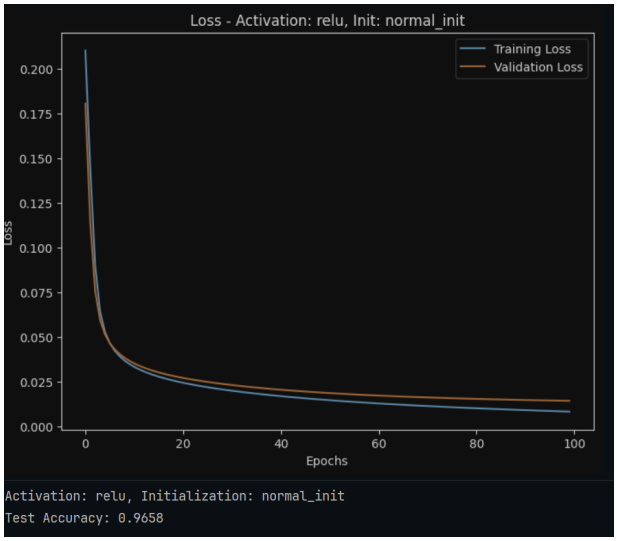
(a)



(b)



(c)

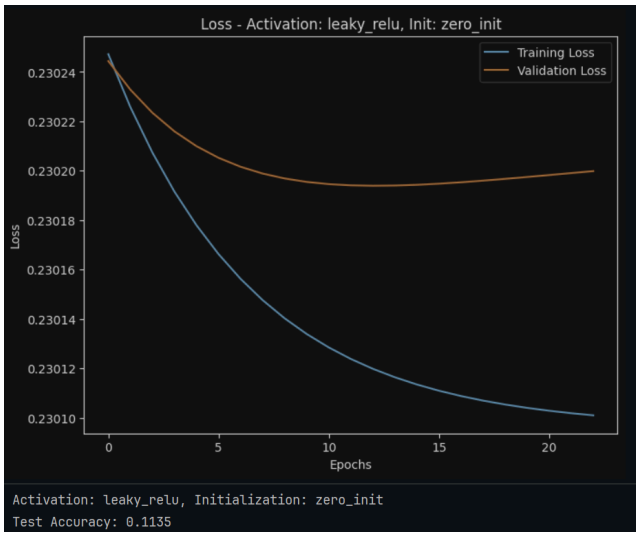


The relu showed even better results than tanh in normal init and a little low in random init, and the same results as sigmoid and tanh in zero init.

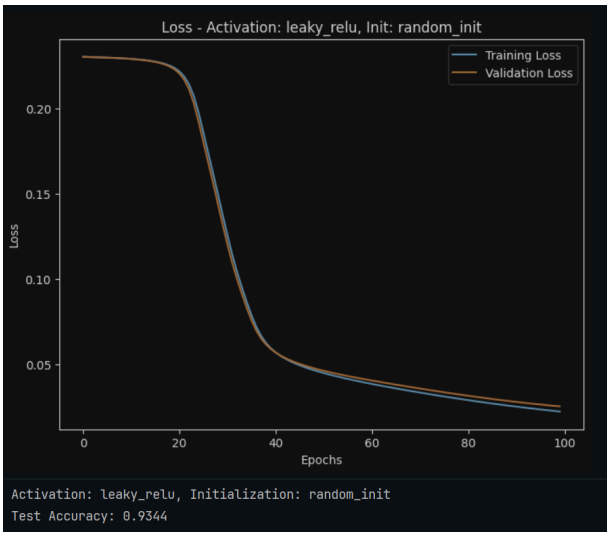
(a) The relu performed the same as Tanh had performed in the zero init case the weights revolved around 0 and didn't get a chance to get themselves updated and decrease the loss therefore had showed the worst performance in relu activation among other init's.
(b) The random init in the beginning did not let weights update faster, showing slower performance more attempts were required to eventually decrease the loss but ultimately showed nice results under 100 epochs, which let it have an accuracy of around 93%.
(c) The Normal unit has performed even better in terms of loss decrease with low bias and low variance. The model quickly achieves the bottom of the gradient descent curve and took a lower number of epochs compared to random init to achieve similar accuracy.

Activation Function: Leaky Relu

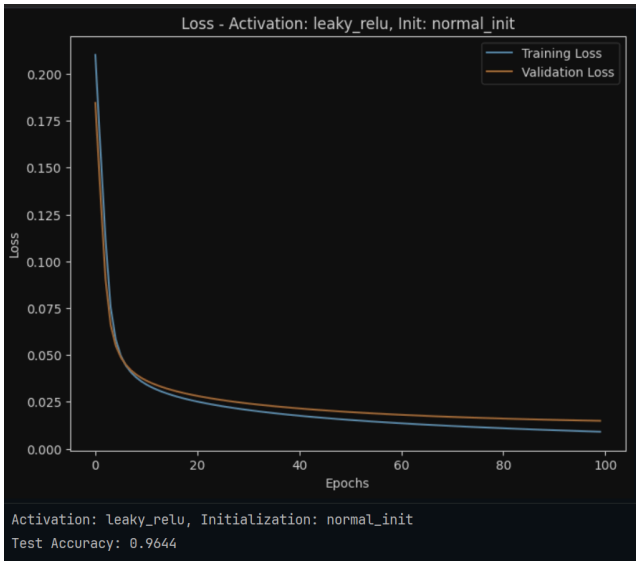
(a)



(b)



(c)



This activation function performed suboptimal then relu in terms of random init and almost similar in the other two init's:

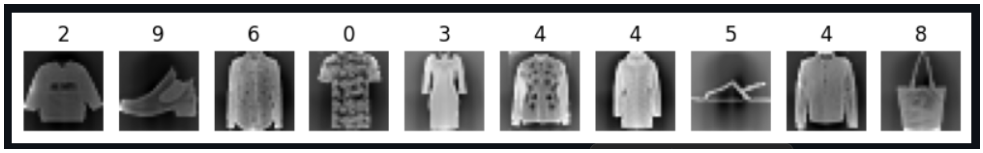
(a) The model performed similarly badly in zero init no model has shown improvement since tanh till now I a get the same results.
(b) The model took more iterations to decrease the slope of the loss function's curve then took in relu but then reached to similar results and almost the same accuracy as the relu activation function.
(c) The model performed excellently, but the validation loss was almost the same as the training loss but with little variance and took no time to converge with the training loss and had showed low bias and low variance and gave an accuracy of around 96%.

Overall Results:

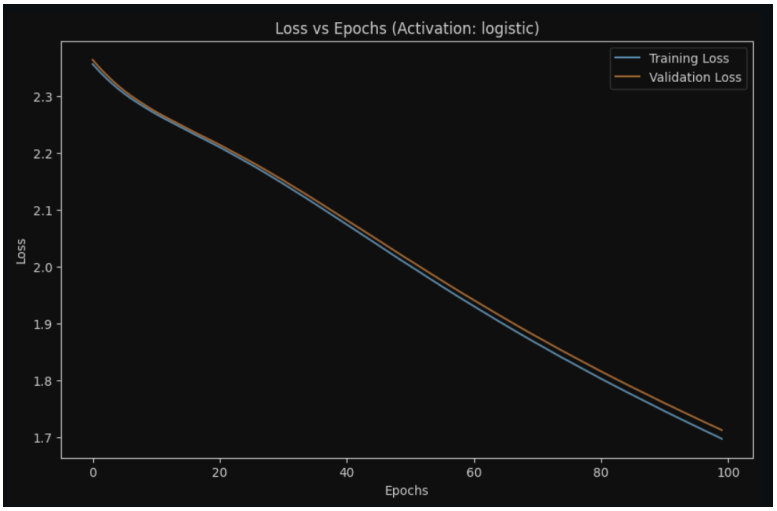
Activation: sigmoid, Initialization: zero_init, Test Accuracy: 11.35% [poor]
Activation: sigmoid, Initialization: random_init, Test Accuracy: 11.35% [poor]
Activation: sigmoid, Initialization: normal_init, Test Accuracy: 13.92% [imrpoved, poor]
Activation: tanh, Initialization: zero_init, Test Accuracy: 11.35% [poor]
Activation: tanh, Initialization: random_init, Test Accuracy: 92.25% [suboptimal]
Activation: tanh, Initialization: normal_init, Test Accuracy: 94.91% [optimal]
Activation: relu, Initialization: zero_init, Test Accuracy: 11.35% [poor]
Activation: relu, Initialization: random_init, Test Accuracy: 93.88% [optimal]
Activation: relu, Initialization: normal_init, Test Accuracy: 96.58% [optimal]
Activation: leaky_relu, Initialization: zero_init, Test Accuracy: 11.35% [poor]
Activation: leaky_relu, Initialization: random_init, Test Accuracy: 93.44% [suboptimal]
Activation: leaky_relu, Initialization: normal_init, Test Accuracy: 96.44% [optimal]

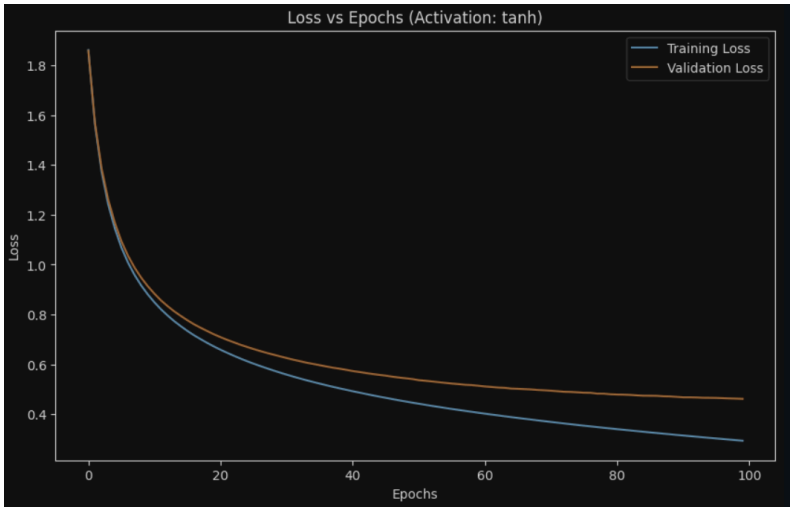
On the basis of the above reasonings and results, I have marks Poor, Suboptimal and Optimal.

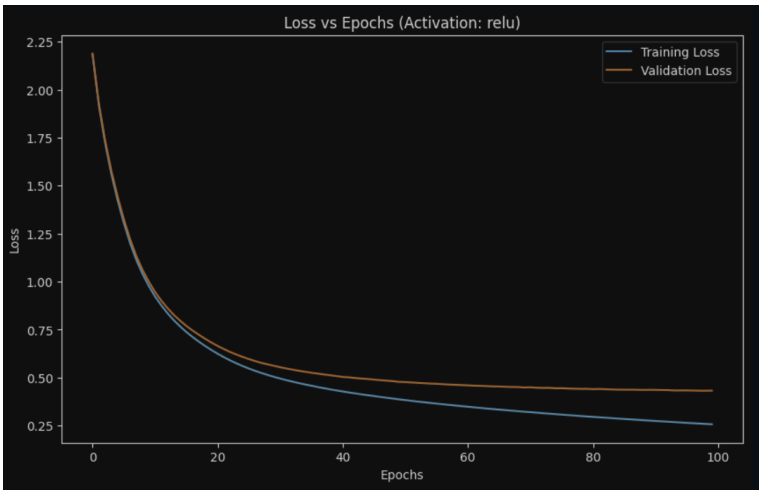
Section: C **(1)**

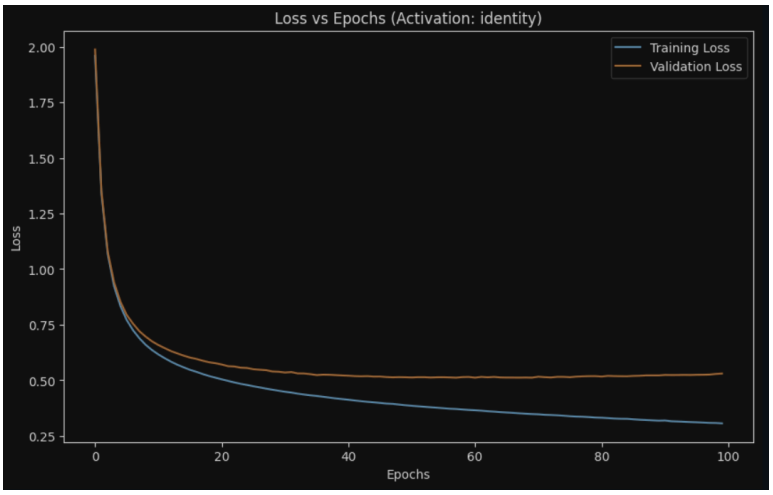


(2)










```
Activation: logistic, Test Accuracy: 0.5435
Activation: tanh, Test Accuracy: 0.8505
Activation: relu, Test Accuracy: 0.8425
Activation: identity, Test Accuracy: 0.8300
```

The best model I chose was one with relu because it showed Good accuracy and also Ended with the lowest validation and training loss among all the models has showed smoother convergence and good gradient descent loss compared to all smoother curve than others even though has less accuracy then tanh but overall had performed better also lowest variance and end among the other models with different activation functions.

Therefore overall the best activation function model was one with the **Relu** activation function.

(3) Grid Search on Relu:

Performed Grid search on these hyperparameters

```
'solver': ['adam', 'sgd'],
'learning_rate_init': [1e-4, 1e-3, 5e-3],
'batch_size': [64, 128, 256]
```

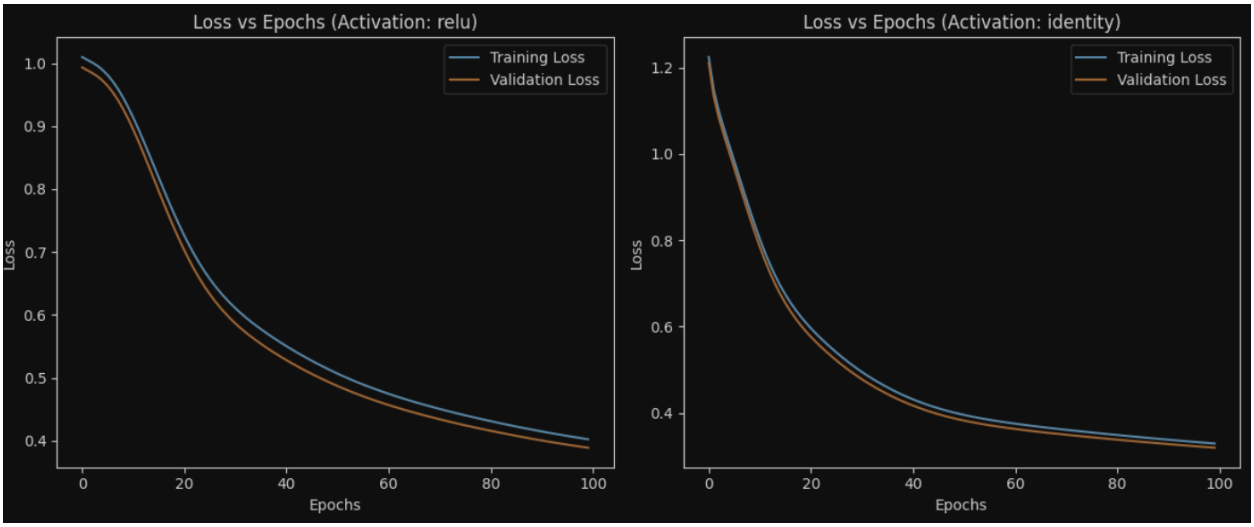
The resultant hyperparameters I got were these:

```
Best Hyperparameters:
{'batch_size': 64, 'learning_rate_init': 0.001, 'solver': 'adam'}
```

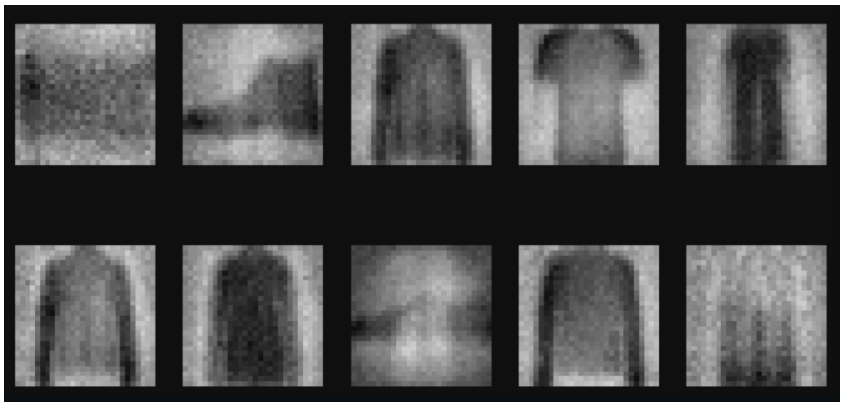
These parameters have performed best among others this configuration, resulted in the Test Accuracy of **86.52%.**

(4)
Layer sizes: [c: 128, b: 64, a: 32]

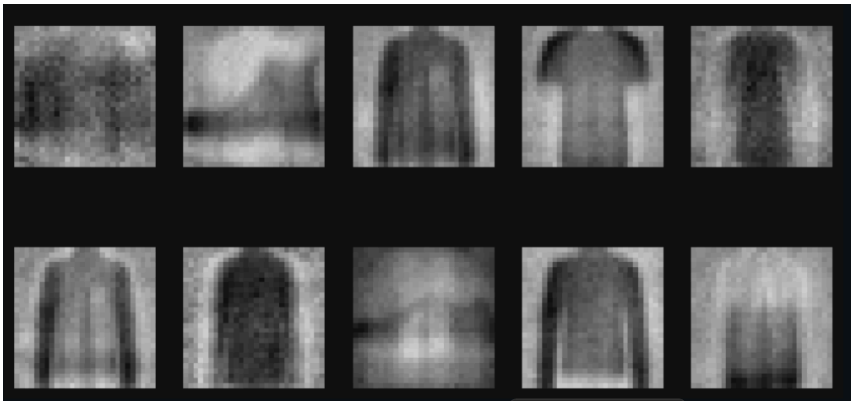
Loss V/S epoch curve for both relu and identity activation function



Regernearted 10 samples from Relu activation function



Regernearted 10 samples from the Identity activation function



Relu got and R2 score of 0.59 and identity got 0.66 that 0.07 increment suggests that the second model has more clearly learned the underlying patterns and performs better therefore the better model on that basis and other metrics is the model with the Identity Activation Function.

(5)

```
Accuracy of MLP Classifier using ReLU features: 0.767
Accuracy of MLP Classifier using Identity features: 0.8045
```

This shows that these two mlp classifiers are still performing well. The reason behind that is, that Extracting features effectively reduces the dimensionality of the data, which often helps to filter out the noise and retain the most critical information. They still perform well because they operate on meaningful, structured feature representations derived from my initial network. This balance between efficiency and performance is often desirable when computational resources are limited or when faster training is needed without significantly sacrificing accuracy.