ML Assignment 3

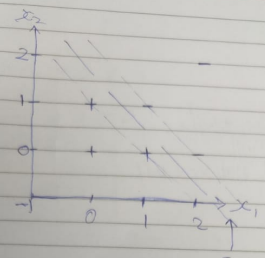
Parth Sandeep Rastogi

IIITD

1 Section:- A Theory

Q2

a)



Seperating boundary

from plot we can see data is linearly separable

b) Support vector ~~between~~ for + class (1,0) and (0,1)
Support vector for − class (1,1), (2,0)

Slope for support vector of both class are same −1

for + it is $x_1 + x_2 = 1$
for − it is $x_1 + x_2 = 2$

So Optimal hyperplane is

$x_1 + x_2 = 1.5$

weigh vector is (1, 1)

Q3 $w_1 = -2$
$w_2 = 0$
$b = 5$

a) $\gamma_i = \frac{|(w x_i + b)|}{\|w\|}$

$\hat{\gamma}_i = \frac{1}{\|w\|} \|w x_i + b\|$

$\hat{\gamma} = \min_{i=1..n} \hat{\gamma}_i$

$\gamma = \min_{i=1..n} \gamma_i$

$\gamma_1 = \frac{3}{2}$ $\quad \hat{\gamma}_1 = 3$

$\gamma_2 = \frac{1}{2}$ $\quad \hat{\gamma}_2 = 1$

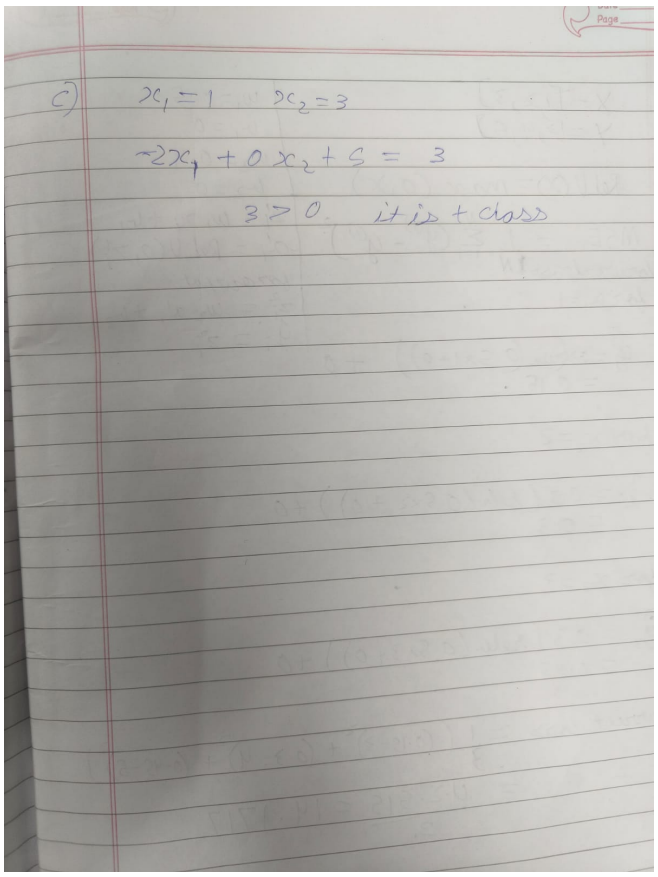
$\gamma_3 = \frac{1}{2}$ $\quad \hat{\gamma}_3 = 1$

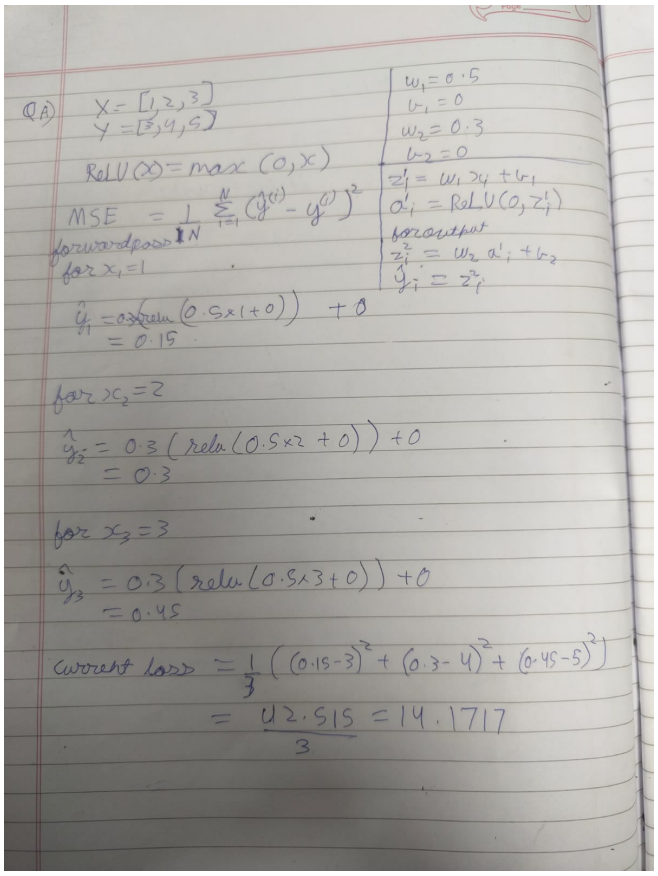
$\gamma_4 = \frac{3}{2}$ $\quad \hat{\gamma}_4 = 3$

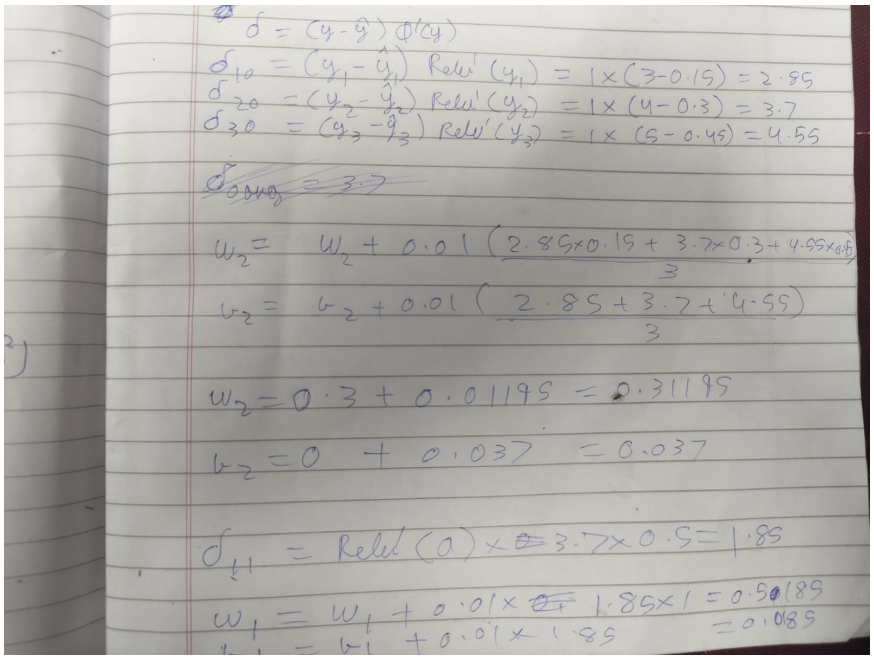
$\min \gamma = \frac{1}{2}$ $\quad \min \hat{\gamma} = 1$

b) Sample 2 and 3 are Support vector of +1 and −1 class respectively

as both have lowest and same margin


c)
$x_1 = 1 \quad x_2 = 3$
$-2x_1 + 0x_2 + 5 = 3$
$3 > 0$ it is + class


Q4)
$X = [1, 2, 3]$
$Y = [3, 4, 5]$
$ReLU(x) = max(0, x)$
$MSE = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}^{(i)} - y^{(i)})^2$
forward pass
for $x_1 = 1$
$\hat{y}_1 = 0.3 relu(0.5 \times 1 + 0) + 0$
$= 0.15$
for $x_2 = 2$
$\hat{y}_2 = 0.3 (relu(0.5 \times 2 + 0)) + 0$
$= 0.3$
for $x_3 = 3$
$\hat{y}_3 = 0.3 [relu(0.5 \times 3 + 0)) + 0$
$= 0.45$
current loss $= \frac{1}{3} ((0.15 - 3)^2 + (0.3 - 4)^2 + (0.45 - 5)^2)$
$= \frac{42.515}{3} = 14.1717$
$w_1 = 0.5$
$b_1 = 0$
$w_2 = 0.3$
$b_2 = 0$
$z_i^1 = w_1 x_i + b_1$
$a_i^1 = ReLU(0, z_i^1)$
for output
$z_i^2 = w_2 a_i^1 + b_2$
$\hat{y}_i = z_i^2$



$\delta = (y - \hat{y})\,\phi'(y)$

$\delta_{10} = (y_1 - \hat{y}_1)\,\text{Relu}'(y_1) = 1 \times (3 - 0.15) = 2.85$

$\delta_{20} = (y_2 - \hat{y}_2)\,\text{Relu}'(y_2) = 1 \times (4 - 0.3) = 3.7$

$\delta_{30} = (y_3 - \hat{y}_3)\,\text{Relu}'(y_3) = 1 \times (5 - 0.45) = 4.55$

$$w_2 = w_2 + 0.01\left(\frac{2.85 \times 0.15 + 3.7 \times 0.3 + 4.55 \times 0.45}{3}\right)$$

$$b_2 = b_2 + 0.01\left(\frac{2.85 + 3.7 + 4.55}{3}\right)$$

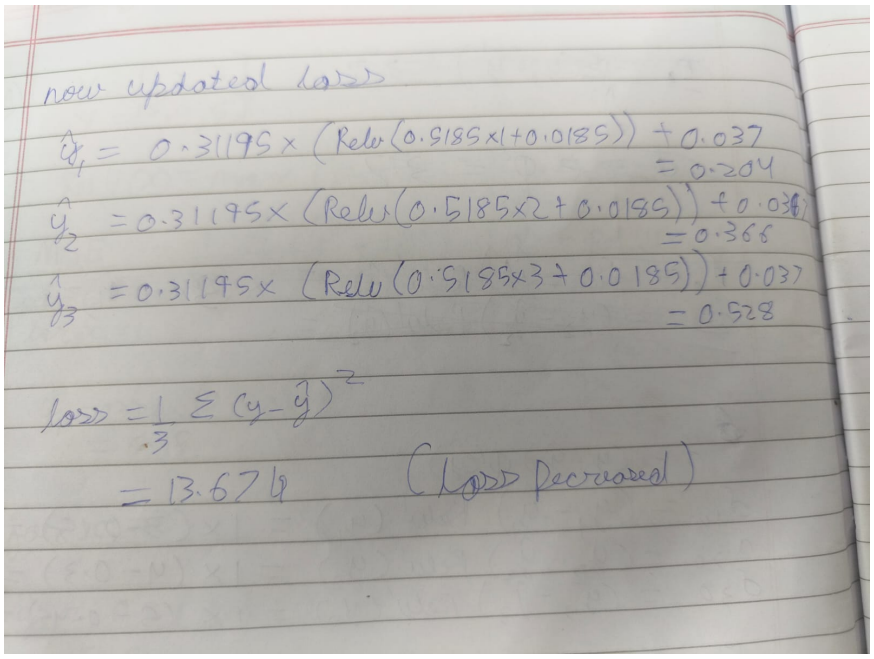
$w_2 = 0.3 + 0.01195 = 0.31195$

$b_2 = 0 + 0.037 = 0.037$

$\delta_{11} = \text{Relu}'(0) \times 3.7 \times 0.5 = 1.85$

$w_1 = w_1 + 0.01 \times 1.85 \times 1 = 0.5185$

$b_1 = b_1 + 0.01 \times 1.85 = 0.0185$



now updated loss>>

$\hat{y}_1 = 0.31195 \times (\text{Relu}(0.5185 \times 1 + 0.0185)) + 0.037 = 0.204$

$\hat{y}_2 = 0.31195 \times (\text{Relu}(0.5185 \times 2 + 0.0185)) + 0.037 = 0.368$

$\hat{y}_3 = 0.31195 \times (\text{Relu}(0.5185 \times 3 + 0.0185)) + 0.037 = 0.528$

$\text{loss} = \frac{1}{3} \Sigma (y - \hat{y})^2$

$= 13.674$ (Loss decreased)

2 Section:-B Scratch Implementation

2.1 Architecture

Here we used [256,128,64,32] as the layers and 128 batch size and trained 12 models

2.2 Results

Leaky Relu and Leaky Relu are giving nan because of exploding gradients

zero initialization is not getting trained because error signal while backpropogating gets a multiplication factor of w which is 0 hence weights are not updating

Activation	Initialization	Train Loss	Val Loss	Train Accuracy (%)	Val Accuracy (%)
Sigmoid	Random	0.0267	0.0301	93.25	91.94
Sigmoid	Normal	0.0349	0.0403	90.53	87.68
Sigmoid	Zero	0.2301	0.2301	11.24	11.22
ReLU	Random	nan	nan	10.22	10.46
ReLU	Normal	nan	nan	9.87	9.88
ReLU	Zero	0.2301	0.2301	10.44	9.92
LeakyReLU	Random	nan	nan	11.24	11.22
LeakyReLU	Normal	nan	nan	9.04	8.30
LeakyReLU	Zero	0.2301	0.2301	11.24	11.22
Tanh	Random	0.0292	0.0747	94.13	81.14
Tanh	Normal	0.1464	0.1869	47.65	37.48
Tanh	Zero	0.2301	0.2301	11.24	11.22

Table 1: Comparison of Activation Functions and Initialization Methods

3 Section:-C Library Implementation

3.1 Part -1)

3.1.1 Preprocessing

From train.csv, I selected the first 8,000 data points as the training set. Additionally, I took the first 2,000 data points and split them into 1,000 validation data points and 1,000 test data points. Since we only have data in CSV format (no .jpg or .png images), we don't need to convert the data. I divided by 255.0 to normalize the data

3.1.2 Visualization

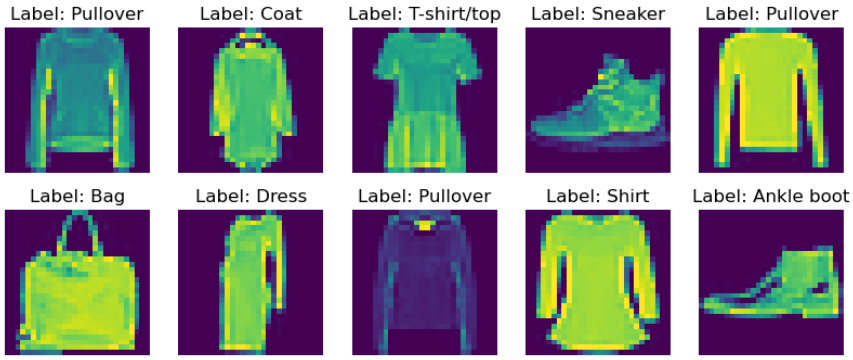
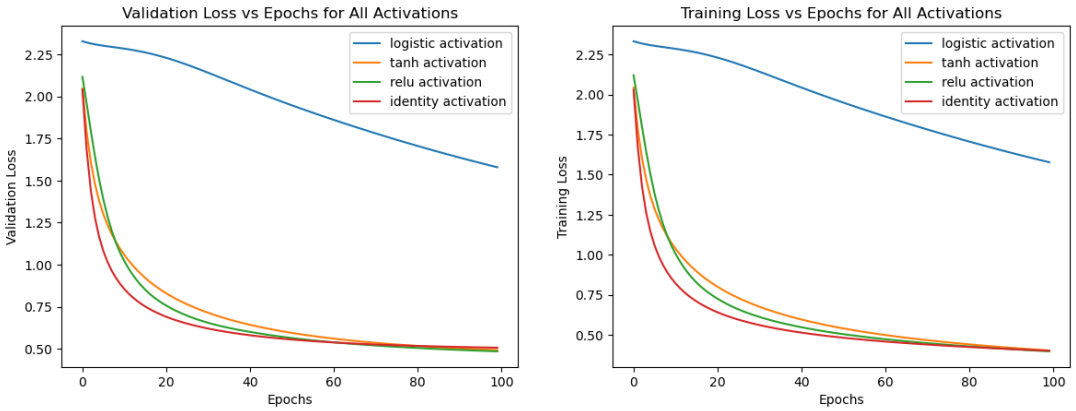


Figure 1: data visualization.

3.2 Part - 2)

3.2.1 Loss vs Epochs



(a) Validation

(b) Training

Figure 2: Comparison between validation and training data

3.2.2 Inference

From the plots we can infer that logistic doesn't converge all other converge but identity converge first but it converge at more loss than tanh and relu . here is given table from which we can say relu performed best .

Table 2: Final Loss and Accuracy for Different Activation Functions

Activation	Train Loss	Train Accuracy	Val Loss	Val Accuracy
Logistic	1.5777	0.46525	1.5787	0.453
Tanh	0.4033	0.87088	0.4932	0.835
ReLU	0.3981	0.8695	0.4858	0.837
Identity	0.4021	0.86738	0.5075	0.826

3.3 Part 3)

3.3.1 Procedure

In this experiment, I performed a grid search to identify the optimal hyperparameters for the MLP Classifier model. The grid search explored different combinations of solver algorithms, learning rate initialization values, and batch sizes to find the best combination for model performance. Also I used cv as 5 .

3.3.2 Parameters Checked in Grid Search

Table 3: Parameters Tested in Grid Search

Parameter	Values Tested
Solver	adam, sgd
Learning Rate Init	0.2, 0.002, 2e-5, 2e-7
Batch Size	64, 128, 256

3.3.3 Best Parameters

Table 4: Optimal Parameters from Grid Search

Parameter	Best Value
Solver	adam
Learning Rate Init	0.002
Batch Size	256

3.3.4 Conclusion

The optimal parameters selected make sense based on the model's requirements and performance. The Adam optimizer was chosen over SGD because it generally achieves faster convergence and better results in deeper networks due to adaptive learning rates. The learning rate of 0.002 provides a balance, offering steady convergence without the instability seen with higher learning rates. Additionally, a larger batch size of 256 helps stabilize the gradients and improves training efficiency, particularly when used in conjunction with Adam.

3.4 Part 4)

3.4.1 Architecture

The Architecture is a 5 Layer MLP with hidden layer as 512, 128, 32, 128, 512.
Relu and Identity are the 2 activation 2e-5 as a learning rate and 100 epochs.
The Model was fitted with X train as data as well as labels

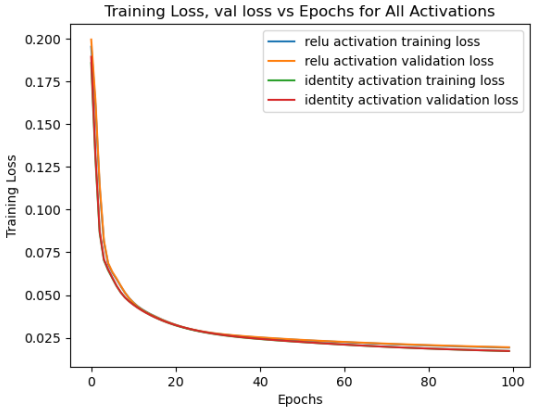


Figure 3: Training and Val losses

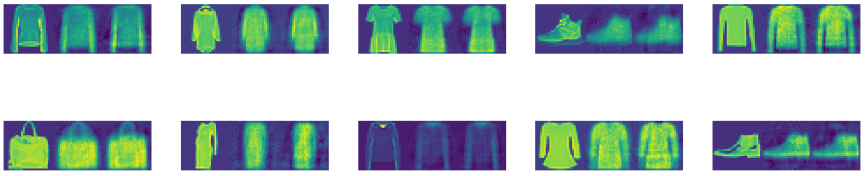


Figure 4: Original vs relu Regenerated Image vs identity activation

3.4.2 Losses for training

3.4.3 Regenerated Image

3.4.4 Conclusion

In experiment comparing ReLU and identity activations for image regeneration using a MLP we observed that the model with identity activation achieved a lower training and validation loss than the model with ReLU activation. Interestingly, the identity-activated model produced regenerated images with sharper edges(makes sense with lower loss). This result contrasts with typical expectations for activation functions, where non-linear activations like ReLU are often assumed to capture complex features better. The identity model's lower loss and sharper outputs indicate that, in this task, it may have been more effective in learning to reconstruct the images more sharp in edge of dress, sole of shoe , handle of bag.

3.5 Part 5)

3.5.1 Introduction

We here see the effectiveness of feature vectors extracted from two neural networks (using `relu` and `identity` activations) trained on a regeneration task. These networks acted as autoencoders by learning to reconstruct the input images, with the feature vectors of size 32 serving as compressed representations. By using these encoded features instead of the raw input data, we aim to explore how well the classifiers can perform using the essential information captured in a reduced dimensional space.

Activation	Accuracy (New Feature space)	Accuracy (Initial data)
relu	0.791	0.837
identity	0.817	0.826

Table 5: Comparison of Classifier Accuracies with Extracted Features and Initial Configurations

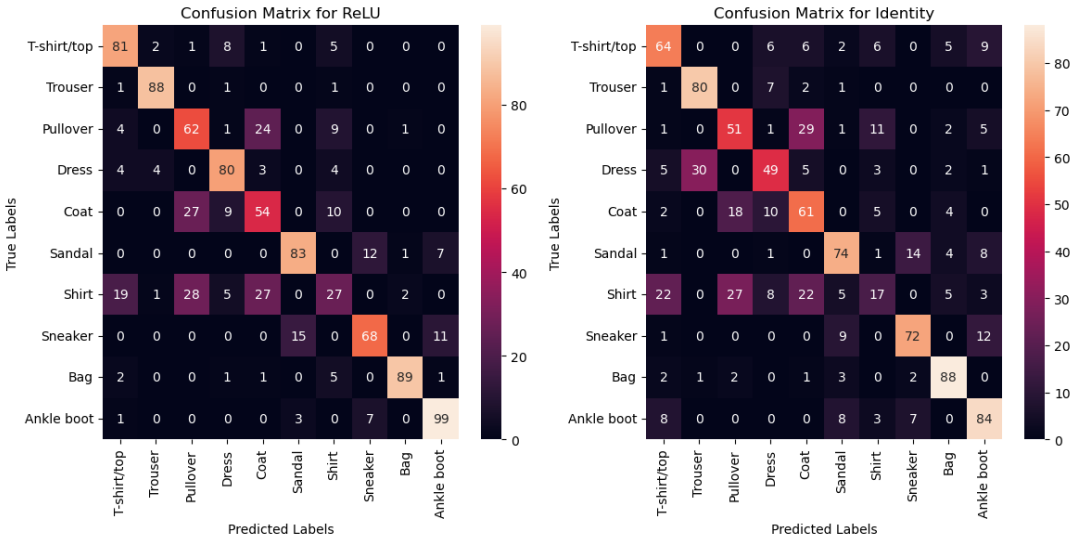


Figure 5: Confusion matrix

3.5.2 Discussion

The classifiers trained on the feature vectors extracted by the autoencoder networks achieved reasonably high accuracies compared to the initial classifiers trained directly on the raw images, with a slight reduction. This performance can be attributed to several factors:

- **Autoencoder Representation:** The pre-trained networks functioned as autoencoders, encoding the most relevant patterns of the input data into a compressed form. This reduced-dimensional representation captures the primary features, making it effective for subsequent classification tasks.
- **Information-Rich Encoding:** The autoencoder-based networks effectively retained essential characteristics of the data in the feature vector of size 32, creating a concise yet information-rich representation that the classifiers could leverage.
- **Transfer Learning Advantage:** By using feature vectors learned in the autoencoder task, the classifiers benefit from transfer learning, allowing for efficient learning even with a smaller model structure.

Additionally to study I plotted the confusion matrix result make sence as from main feature extraction the model can classify general class if it is a top or lower or shoe or bag but distinction between shirt, pullover, coat, t-shirt is hard for encoded representation space

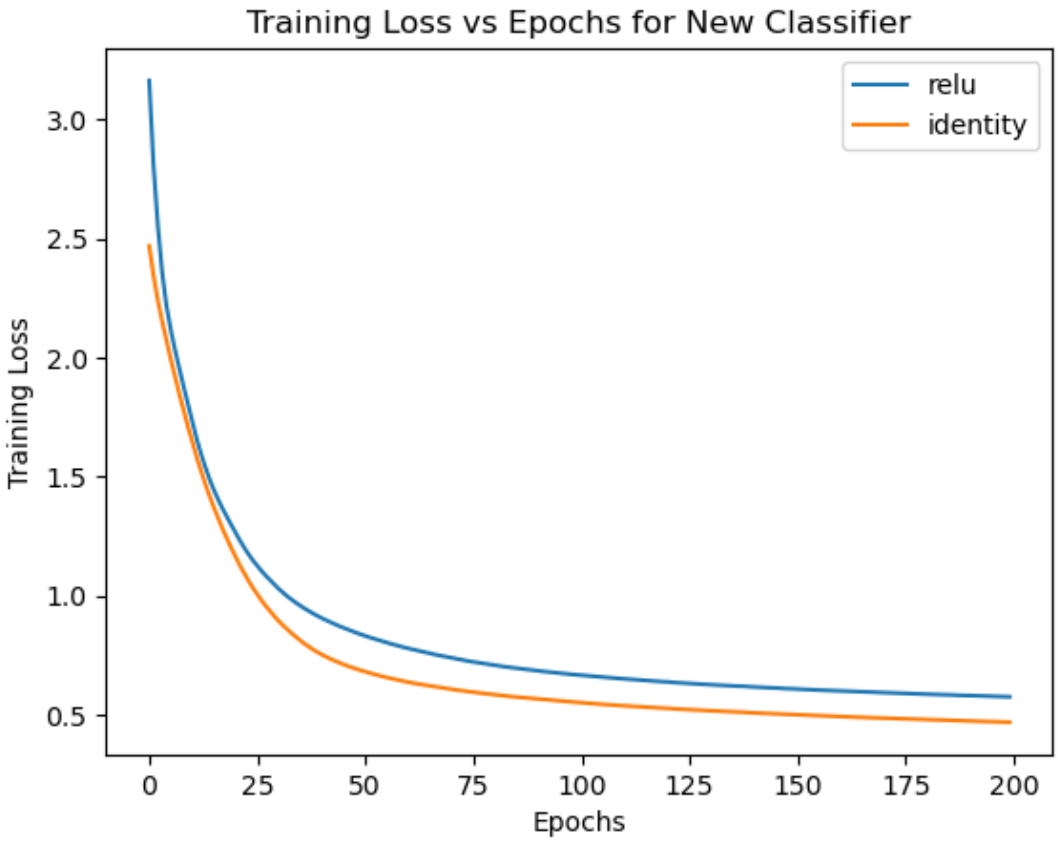


Figure 6: loss curve