report 2022266

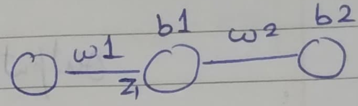
Section A

DATE
PAGE

Section-A

Part A:

Input	Target
1	4
2	5
3	6



(η) learning rate = 0.01

Initializing Weights : $w1 = 0.5$ $\quad w2 = 0.3$

$b1 = 0.1$ $\quad b2 = 0.2$

Forward Pass

1. $x = 1$

hidden layer pre-activation:

$$z_1 = w_1 \cdot x_1 + b_1 = 0.5(1) + 0.1 = 0.6$$

hidden layer activation (ReLU):

$$h = \text{ReLU}(0.6) = 0.6$$

Output Layer:

$$y_1 = w_2 \cdot h + b_2 = 0.3(0.6) + 0.2 = 0.38$$

Error:

$$E_1 = (y_1 - d_1)^2 = (0.38 - 3)^2 = 6.8644$$

2. $x = 2$

hidden layer preactivation:

$$z_1 = w_1 \cdot x_2 + b_1 = 0.5(2) + 0.1 = 1.1$$

hidden Layer activation (ReLU):

$$h = \text{ReLU}(1.1) = 1.1$$

Output layer:

$$y_2 = w_2 \cdot h + b_2 = 0.3(1.1) + 0.2 = 0.53$$

Error:

$$E_2 = (y_2 - d_2)^2 = (0.53 - 4)^2 = 12.0324$$

3. $x = 3$

hidden layer pre-activation:

$z_1 = w_1 \cdot x_3 + b_1 = 0.5(3) + 0.1 = 1.6$

hidden layer activation (ReLU):

$h = ReLU(1.6) = 1.6$

Output Layer

$y_3 = w_2 \cdot h + b_2 = 0.3(1.6) + 0.2 = 0.68$

Error:

$E_3 = \cancel{(d_3 - y_3)^2} = (y_3 - d_3)^2 = (0.68 - 5)^2 = 18.5664$

Computing Loss

$MSE = \frac{1}{3} \sum_n (y_i - d_i)^2 = \frac{1}{3}(E_1 + E_2 + E_3)$

$MSE = \frac{1}{3}(E_1 + E_2 + E_3) = \frac{1}{3}(6.8644 + 12.0329 + 18.5664)$

$MSE = \boxed{E = 12.4879}$

Backward Pass

1. gradient of E wrt w_2:

$$\frac{\partial E}{\partial w_2} = \frac{\partial}{\partial w_2}\left[\frac{1}{3}\sum_n (y_i - d_i)^2\right]$$

$$= \frac{2}{3}\sum_n (y_i - d_i) \cdot \frac{\partial y_i}{\partial w_2}$$

$$\boxed{\frac{\partial E}{\partial w_2} = \frac{2}{3}\sum_n (y_i - d_i) \cdot h_i}$$

– we can use the result again later.

$$\frac{\partial E}{\partial w_2} = \frac{2}{3}\left[(0.38 - 3)(0.6) + (0.53 - 4)(1.1) + (0.08 - 5)(1.6)\right]$$

$= -8.719$

DATE
PAGE

(2) gradient w.r.t b_2 :

Using earlier result:

$$\frac{\partial E}{\partial b_2} = \frac{2}{3}\sum_n (y_i - d_i)$$

~~here~~ $\cancel{\frac{\partial y_i}{\partial b_2} =}$

$$\frac{\partial E}{\partial b_2} = \frac{2}{3}\left[(0.38-3) + (0.53-4) + (0.68-5)\right]$$

$$= -6.94$$

(3) gradient w.r.t w_1 :

$$\frac{\partial E}{\partial w_1} = \frac{2}{3}\sum_n (y_i - d_i)\cdot w_2 \cdot x_i$$

$$\frac{\partial E}{\partial w_1} = \frac{2}{3}\left[(-2.62)(0.3)(1)(1) + (-3.47)(0.3)(1)(2) + (-4.32)(0.3)(1)(3)\right]$$

$$= -4.504$$

(4) gradient w.r.t b_1 :

$$\frac{\partial E}{\partial b_1} = \frac{2}{3}\sum_n (y_i - d_i)\cdot w_2 \cdot 1$$

$$\frac{\partial E}{\partial b_1} = \frac{2}{3}\left[(-2.62)(0.3) + (-3.47)(0.3) + (-4.32)(0.3)\right]$$

$$= -2.52$$

Updating the weights : Final weights.

$$w_1 = w_1 - \eta\frac{\partial E}{\partial w_1} = 0.3 - 0.01(-8.719) = 0.38719$$

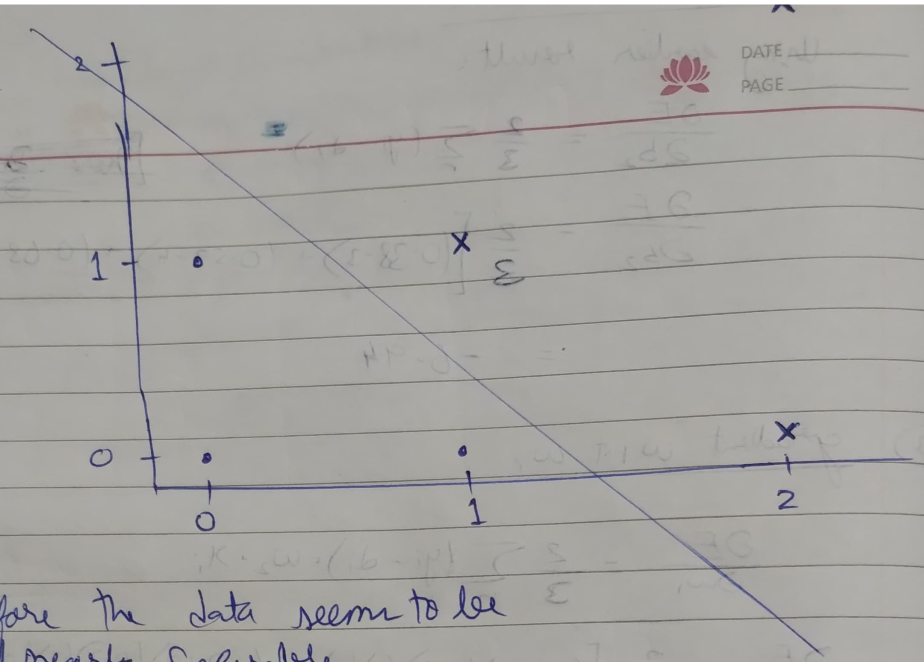
$$b_1 = b_1 - \eta\frac{\partial E}{\partial b_1} = 0.2 - 0.01(-6.94) = 0.2694$$

$$w_2 = w_2 - \eta\frac{\partial E}{\partial w_2} = 0.5 - 0.01(-4.504) = 0.54504$$

$$b_2 = b_2 - \eta\frac{\partial E}{\partial b_2} = 0.1 - 0.01(-2.52) = 0.1252$$

Part B:

(a)



Therefore the data seem to be ~~Linearly Seperable~~.

(b) For correct classification :

$y^i(\omega^T x_i + b) \geq 1$

Support Vectors:

For Class + : points (1,0) & (0,1) seem to be the closest

For Class - : points (1,1) seem to be the closest.

Applying equation for correct classification to support vectors

$(\omega^T x_i + b) y^i \geq 1$

For (0,1) : $\omega_1 \cdot 0 + \omega_2 \cdot 1 + b = 1 \Rightarrow \boxed{\omega_2 + b = 1}$ — ①

For (1,0) : $\omega_1 \cdot 1 + \omega_2 \cdot 0 + b = 1 \rightarrow \boxed{\omega_1 + b = 1}$ — ②

For (1,1) : $\omega_1 \cdot 1 + \omega_2 \cdot 1 + b = -1 \Rightarrow \boxed{\omega_1 + \omega_2 + b = -1}$ ③

using ① & ② $\quad w_2 + b = w_1 + b$

$\Rightarrow \boxed{w_1 = w_2}$

using ① $\quad \Rightarrow \boxed{b = 1 - w_2}$

Then, $\quad w_1 + w_2 + b = -1$

$w_2 + w_2 + 1 - w_2 = -1$

$\boxed{w_2 = -2}$

$\Rightarrow \boxed{w_1 = -2}$

$w_1 + b = 1 \Rightarrow -2 + b = 1 \Rightarrow \boxed{b = 3}$

Hyperplane

$w^T x_i + b = 0$

$\Rightarrow -2 \cdot x_1 - 2 \cdot x_2 + 3 = 0$

$\Rightarrow \boxed{x_1 + x_2 = \frac{3}{2}}$

Support vector:

point (0,1), (1,0), (1,1) ~~are~~ closest to the hyperplane.

Part C:

given that $w = [w_1, w_2] = [-2, 0]$ for decision boundary
$b = 5$

x_1	x_2	y
1	2	+1
2	3	+1
3	3	−1
4	1	−1

(a) Margin:

$$\|w\| = \sqrt{w_1^2 + w_2^2} = \sqrt{(-2)^2 + (0)^2} = 2$$

$$\text{Margin} = \frac{1}{\|w\|} = 0.5$$

(b) Support Vector

they need to Satisfy $y_i(w \cdot x_i + b) = 1$

For (1, 2): $y_1(w_1 \cdot x_{11} + w_2 x_{12} + b) = 1((-2)(1) + (0)(2) + 5) = 3$
For (2, 3): $y_2(w_1 x_{21} + w_2 x_{22} + b) = 1((-2)(2) + (0)(3) + 5) = 1$
For (3, 3): $y_3(w_1 x_{31} + w_2 x_{32} + b) = -1((-2)(3) + (0)(3) + 5) = 1$
For (4, 1): $y_4(w_1 x_{41} + w_2 x_{42} + b) = -1((-2)(4) + (0)(1) + 5) = 3$

There are 2 support vectors:

(2, 3) & (3, 3)

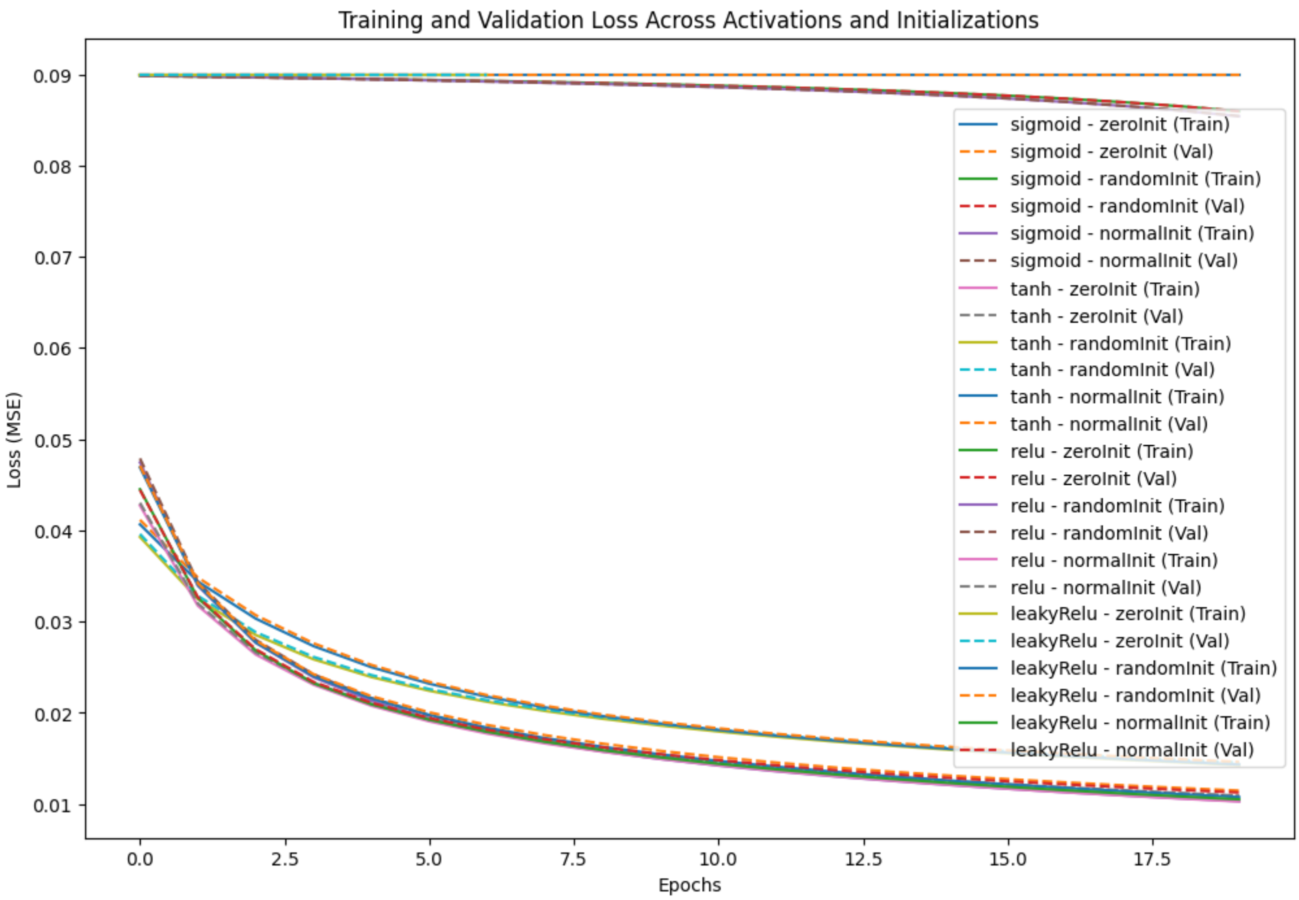
(c) For (1,3) :

$$w_1(1) + w_2(3) + b$$
$$= (-2)(1) + (0)(3) + 5$$
$$= -2 + 5$$
$$= 3$$

Since $w \cdot x_i + b > 0$ hence we predict $\boxed{\text{class } +1}$

Section B

Plot for training loss vs. epochs and validation loss vs. epochs for each activation function and weight initialization function



Training model with sigmoid activation and zeroInit weight initialization.

Epoch: 20, Train Loss: 0.0900, Val Loss: 0.0900, Train Acc: 0.1129, Val Acc: 0.1137

Training model with sigmoid activation and randomInit weight initialization.
Epoch: 20, Train Loss: 0.0859, Val Loss: 0.0859, Train Acc: 0.4032, Val Acc: 0.4032

Training model with sigmoid activation and normalInit weight initialization.
Epoch: 20, Train Loss: 0.0854, Val Loss: 0.0854, Train Acc: 0.4190, Val Acc: 0.4182

Training model with tanh activation and zeroInit weight initialization.
Epoch: 6, Train Loss: 0.0900, Val Loss: 0.0900, Train Acc: 0.1129, Val Acc: 0.1137
Early stopping at epoch 7 for lack of improvement.

Training model with tanh activation and randomInit weight initialization.
Epoch: 20, Train Loss: 0.0143, Val Loss: 0.0146, Train Acc: 0.9492, Val Acc: 0.9482

Training model with tanh activation and normalInit weight initialization.
Epoch: 20, Train Loss: 0.0144, Val Loss: 0.0146, Train Acc: 0.9464, Val Acc: 0.9438

Training model with relu activation and zeroInit weight initialization.
Epoch: 6, Train Loss: 0.0900, Val Loss: 0.0900, Train Acc: 0.1129, Val Acc: 0.1137
Early stopping at epoch 7 for lack of improvement.

Training model with relu activation and randomInit weight initialization.
Epoch: 20, Train Loss: 0.0103, Val Loss: 0.0109, Train Acc: 0.9600, Val Acc: 0.9582

Training model with relu activation and normalInit weight initialization.
Epoch: 20, Train Loss: 0.0103, Val Loss: 0.0109, Train Acc: 0.9605, Val Acc: 0.9563

Training model with leakyRelu activation and zeroInit weight initialization.
Epoch: 6, Train Loss: 0.0900, Val Loss: 0.0900, Train Acc: 0.1129, Val Acc: 0.1137
Early stopping at epoch 7 for lack of improvement.

Training model with leakyRelu activation and randomInit weight initialization.
Epoch: 20, Train Loss: 0.0108, Val Loss: 0.0115, Train Acc: 0.9597, Val Acc: 0.9555

Training model with leakyRelu activation and normalInit weight initialization.
Epoch: 20, Train Loss: 0.0105, Val Loss: 0.0113, Train Acc: 0.9616, Val Acc: 0.9578

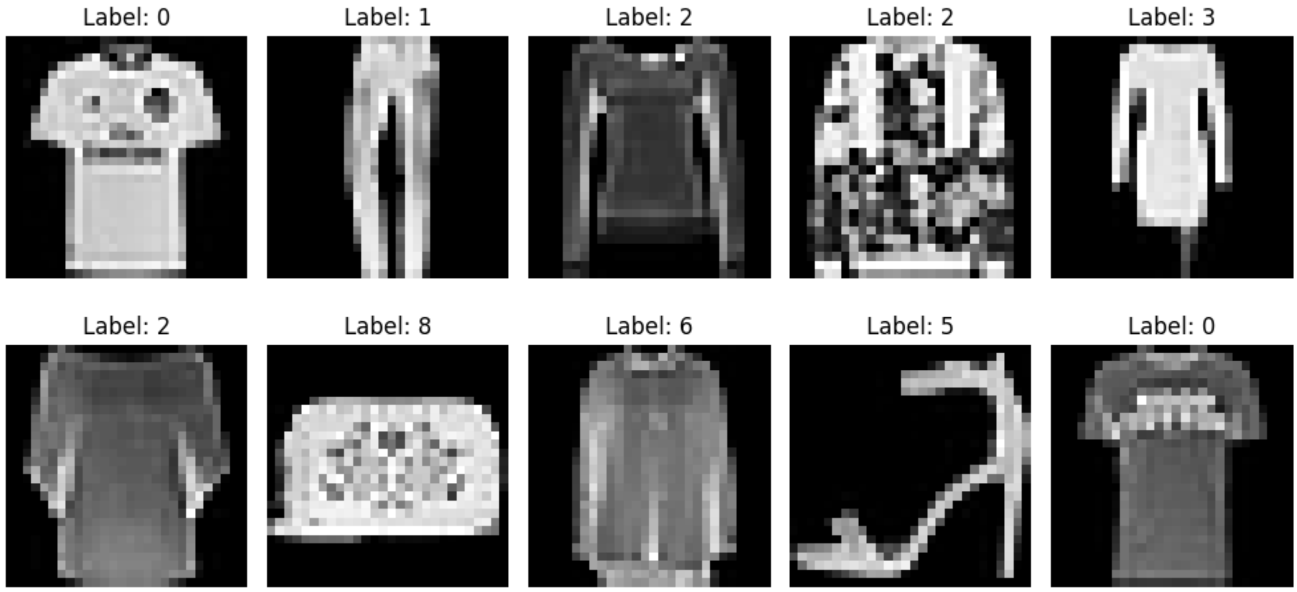
The best performing function combination was : leakyRelu_normalInit_model

Although many combinations had a similar performance

```
Test Accuracy for leakyRelu_zeroInit_model.pkl: 0.1067
Test Accuracy for leakyRelu_randomInit_model.pkl: 0.9525
Test Accuracy for leakyRelu_normalInit_model.pkl: 0.9545
Test Accuracy for relu_zeroInit_model.pkl: 0.1067
Test Accuracy for relu_randomInit_model.pkl: 0.9518
Test Accuracy for relu_normalInit_model.pkl: 0.9530
Test Accuracy for sigmoid_zeroInit_model.pkl: 0.1067
Test Accuracy for sigmoid_randomInit_model.pkl: 0.4010
Test Accuracy for sigmoid_normalInit_model.pkl: 0.4147
Test Accuracy for tanh_zeroInit_model.pkl: 0.1067
Test Accuracy for tanh_randomInit_model.pkl: 0.9442
Test Accuracy for tanh_normalInit_model.pkl: 0.9430
```

Section C

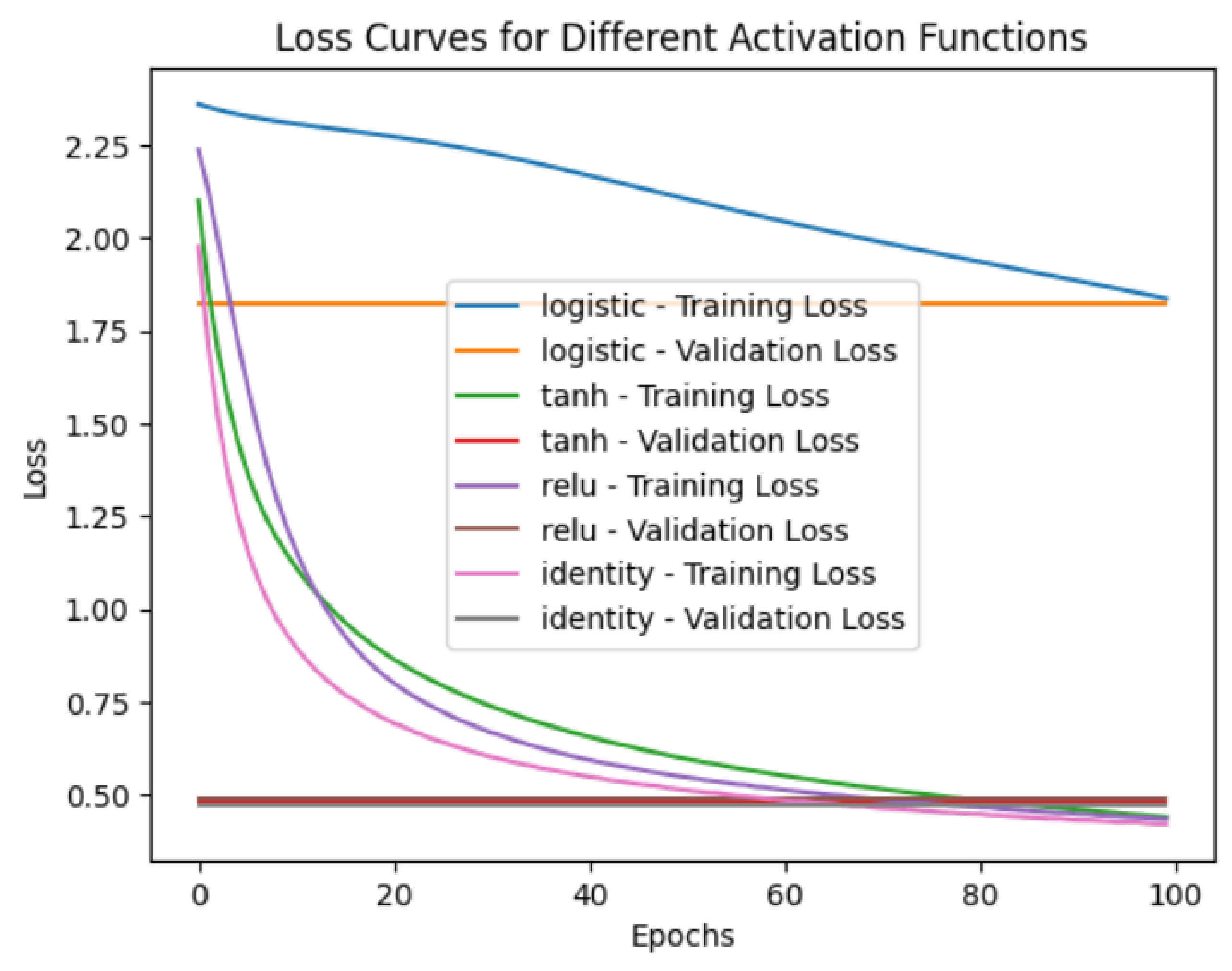
1. Visualizing dataset



2. Plotting Loss for each activation function

tanh, relu, identity all three of them performed equally on test set.

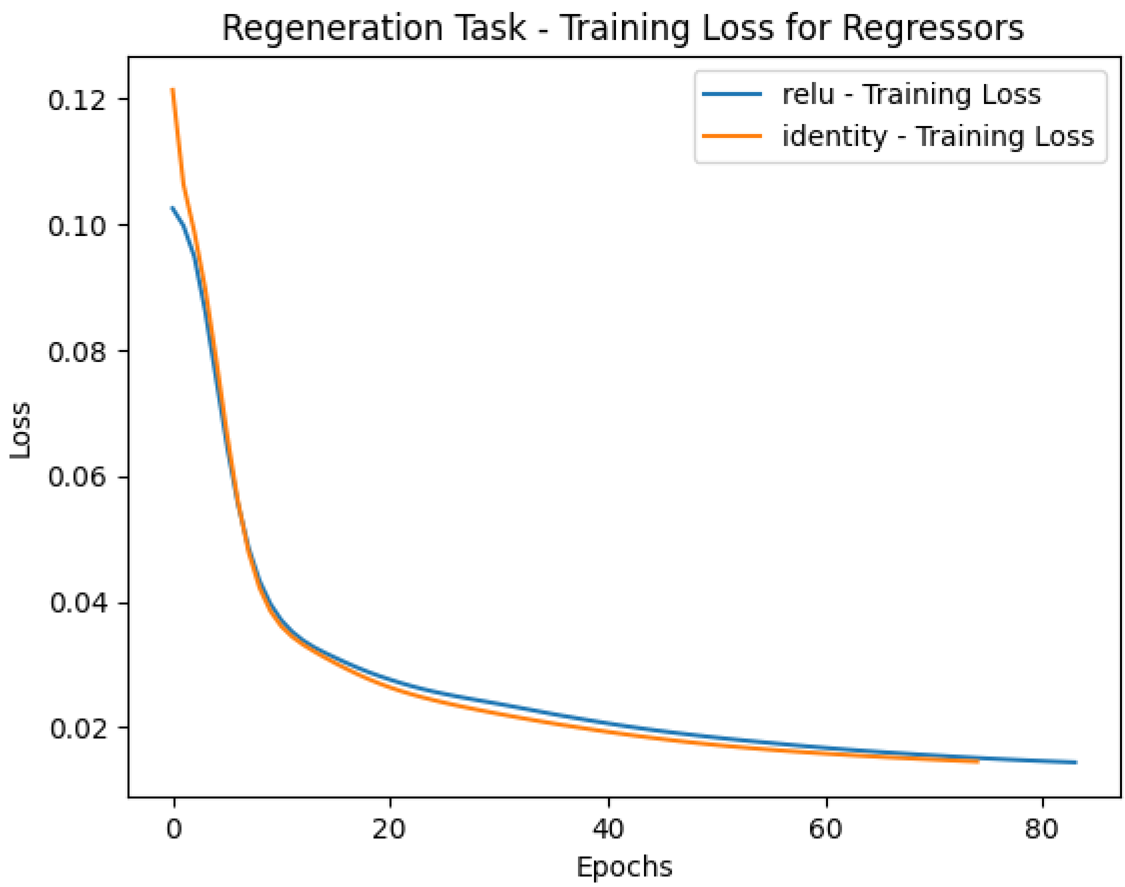
```
Test Accuracy for activation 'logistic': 0.4725
Test Accuracy for activation 'tanh': 0.8462
Test Accuracy for activation 'relu': 0.8462
Test Accuracy for activation 'identity': 0.8475
```

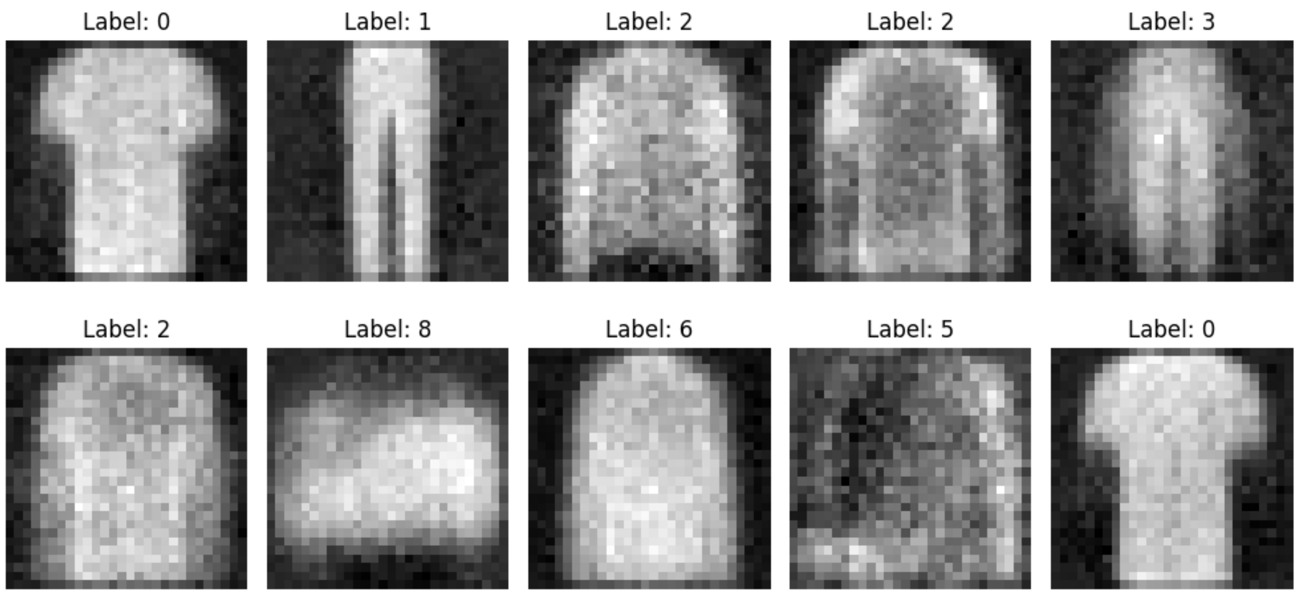


3. The best hyperparameters (eg: solver, learning rate, batch size) for the MLP classifier

```
Best parameters found: {'batch_size': 64, 'learning_rate_init': 5e-05, 'solver': 'adam'}
```

4. Post training :





Observations :

1. **General Shape Capture**: The regenerated images capture the general shape of each clothing item, showing that the networks learned basic features. However, they appear somewhat blurred and lack fine details.
2. **ReLU vs. Identity Activation**: Both activation functions perform similarly, with ReLU achieving slightly lower loss, indicating it might capture features a bit better.
3. **Identifiable Classes**: For items like t-shirts (label 0) and pants (label 1), the images retain enough detail to identify them, showing the networks learned to differentiate broad shapes.

4. **Lack of Fine Detail**: More detailed items, like sandals (label 5) and bags (label 8), are less clear in the regenerated images, suggesting the networks struggle with finer details.

Overall, the networks work well for basic shapes but miss finer details that might be important for more accurate classification.

5. Accuracy with feature vector using 2 different activation functions

```
Accuracy with feature vector using 'relu' activation: 0.7465
Accuracy with feature vector using 'identity' activation: 0.7395
```

The feature vectors extracted from the trained networks provide a decent classifier for a few key reasons:

1. **Pre-learned Features**: The neural networks trained with relu and identity activations learned meaningful features that capture essential patterns in the images, simplifying the classification task. These features serve as a condensed representation, making the classification easier for the smaller MLPs.
2. **Reduced Complexity**: The smaller MLPs focus only on the extracted features rather than the raw pixel data, which reduces the complexity of the problem. This allows them to learn more efficiently and reach a good level of accuracy.
3. **Feature Transferability**: The feature vectors capture common patterns that are useful across samples, even though they may not be optimized for fine details. This transferability enables the MLP classifiers to generalize well without requiring highly complex architectures.

Meaning, using these pre-learned feature vectors provides a strong foundation, allowing the smaller MLP classifiers to perform effectively with fewer layers and parameters.