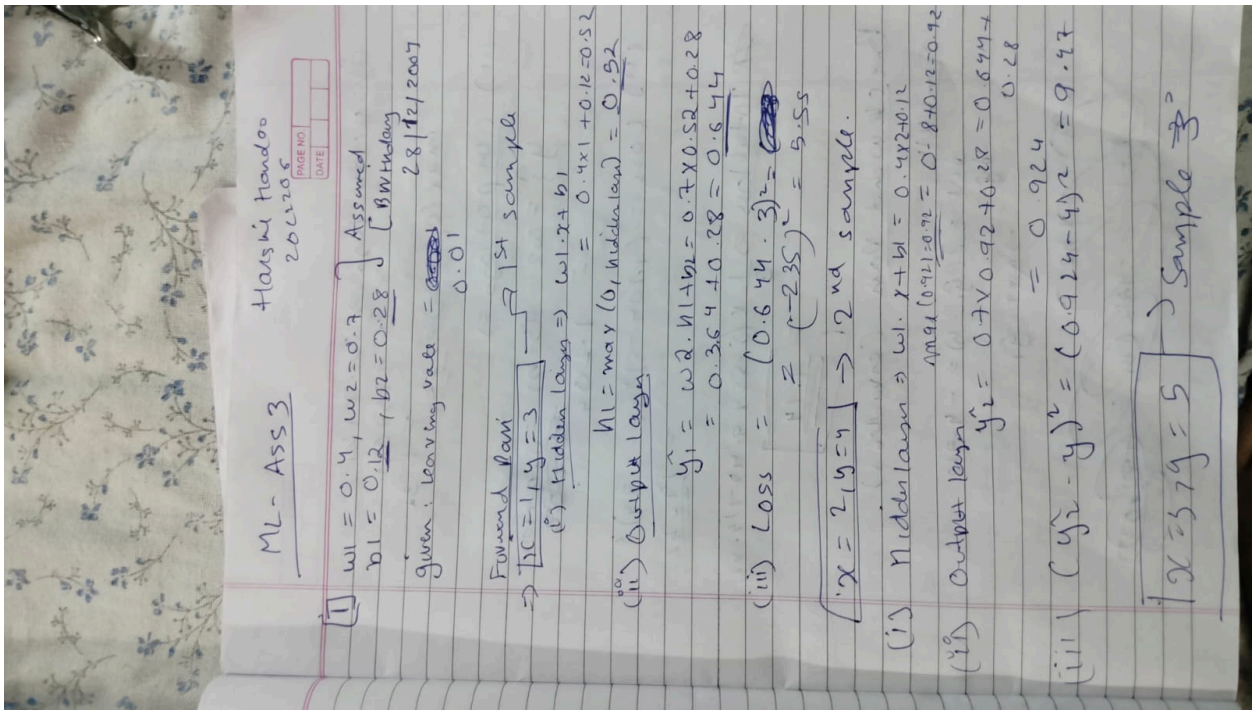
ML ASSIGNMENT 3

Harshil Handoo
2022206
CSAM 2026

PART-A

a)



wrt w_2 & b_2 : $\frac{\partial L_1}{\partial w_2} = -4.712 \times h_1 = -4.712 \times 0.52 = -2.45$

$\frac{\partial L_1}{\partial b_2} = -4.712$

wrt w_1 & b_1 : $h_1 = \max(0, \text{hidden layer})$

~~$\frac{\partial L_1}{\partial w_1} = \ldots \times 0.7$~~ ~~$= -3.298$~~

$\frac{\partial L_1}{\partial b_1} = -4.712$

$\frac{\partial L_1}{\partial w_1} = -4.712 \times 0.7 \times 1 = -3.298$

$\frac{\partial L_1}{\partial b_1} = \frac{\partial L_1}{\partial \hat{y}_1} \times w_2 = -4.712 \times 0.7 = -3.298$

For S2 : $x = 2, y = 4$

wrt $\hat{y_2}$ $\Rightarrow \frac{\partial L_2}{\partial \hat{y_2}} = 2(0.724 - 4) = -6.152$

wrt w_2 & b_2

$\frac{\partial L_2}{\partial b_2} = -6.152$

$\frac{\partial L_2}{\partial w_2} = -6.152 \times 0.92 = -5.660$

wrt w_1 & b_1

$\frac{\partial L_2}{\partial w_1} = -6.152 \times 0.72 = -8.61$

$\frac{\partial L_2}{\partial b_1} = \frac{\partial L_2}{\partial \hat{y_2}} \cdot w_2 = -6.152 \times 0.7 = -4.3064$

(i) Hidden layer $\Rightarrow w_1 \cdot x + b_1$

$= 0 - 4.85 + 0.12$

$= 1.32$

$\max(0, 1.32) = 1.32$

(ii) Output layer $\Rightarrow$

$\hat{y_3} = w_2 \cdot h_3 + b_2 = 0.92 + 0.28$

$= 1.204$

(iii) Loss $= (1.204 - 5)^2 = (-3.79)^2$

$= 14.41$

Total loss $\Rightarrow \frac{1}{3} \times (5.55 + 9.47 + 14.41)$

$= 9.14$

Backward Pass (Gradient)

Sample 1:

$\frac{\partial L_1}{\partial \hat{y}_1} = 2(\hat{y}_1 - y) = -4.712$

$\rightarrow$ wrt $\hat{y}_1$

wrt w_2 & b_2

$\frac{\partial L_1}{\partial w_2} = -4.712 \times h_1 = -4.712 \times 0.52$

$= 2.45$

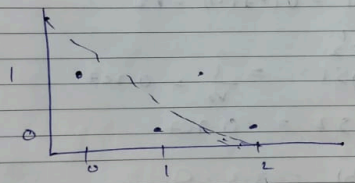
updated parameters

$w_1 = 0.4928$, $b_1 = 0.1631$

$w_2 = 0.7604$, $b_2 = 0.3413$

b) we can make a decision boundary hence the points are linearly seperable. Hyperplane will classify the points without misclassification

Eq: $w_1 x_1 + w_2 x_2 + b = 0$



$x_1 + x_2 = 1.5$ (dashed line)

$w = [w_1, w_2]$ weight vector

$y = +1$ → +ve class

$y = -1$ → -ve class

$y \cdot (w^T x + b) = 1$ → SV

seeing $x_1 + x_2 = 1.5$,

$w = [1, 1]$

$b = -1.5$

S_3 when $x = 3, y = 5$

$wx + y_3$

$\frac{\partial L_3}{\partial \hat{y_3}} = 2(1.264 - 5) = -7.592$

$wx + w_2 \& b_2 = -7.592 \times 1.32 = -10.019$

$wx + w_1 \& b_1$

$\frac{\partial L_3}{\partial w_1} = -1.5923$

$\frac{\partial L_3}{\partial \hat{y_3}} \times w_2 = -5.3144$

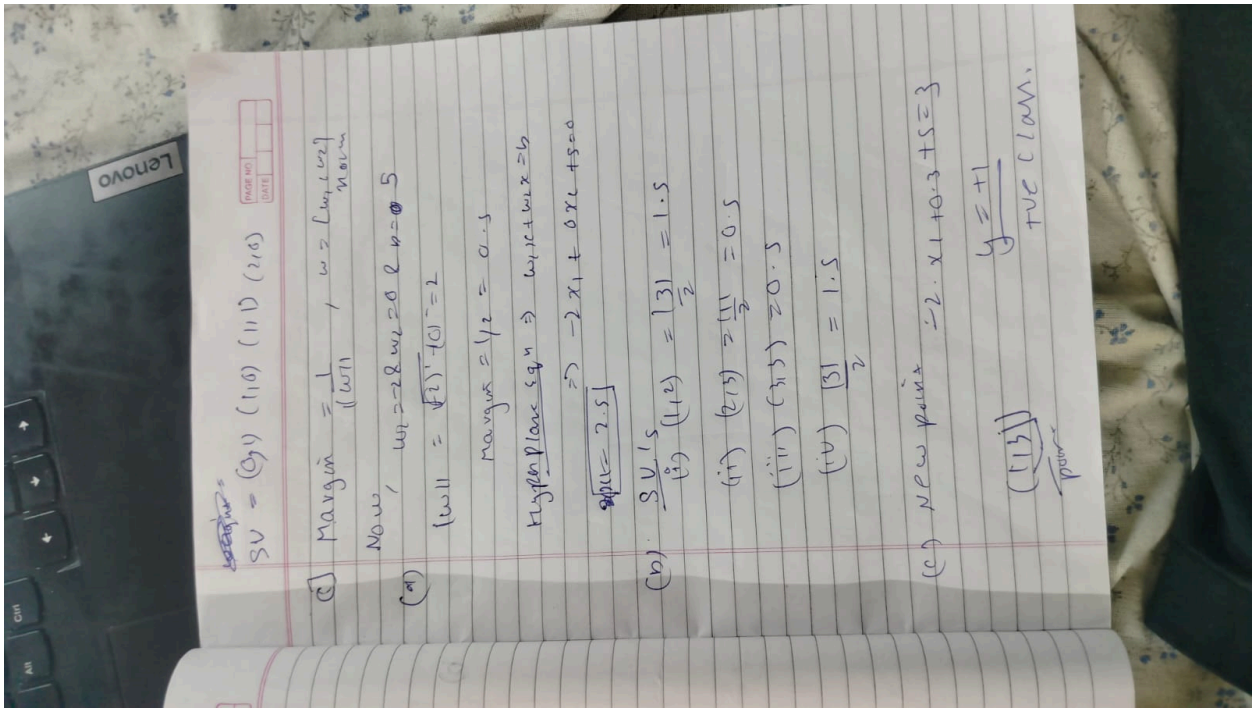
Avg of gradient

$\frac{\partial \text{Total loss}}{\partial w_2} = \frac{-2.455.66 - 10.019}{3} = -6.043$

$\frac{\partial \text{Total}}{\partial b_2} = -6.152$

$\frac{\partial \text{Total loss}}{\partial w_1} = 9.2878$

$\frac{\partial \text{Total loss}}{\partial b_1} = -4.3084$

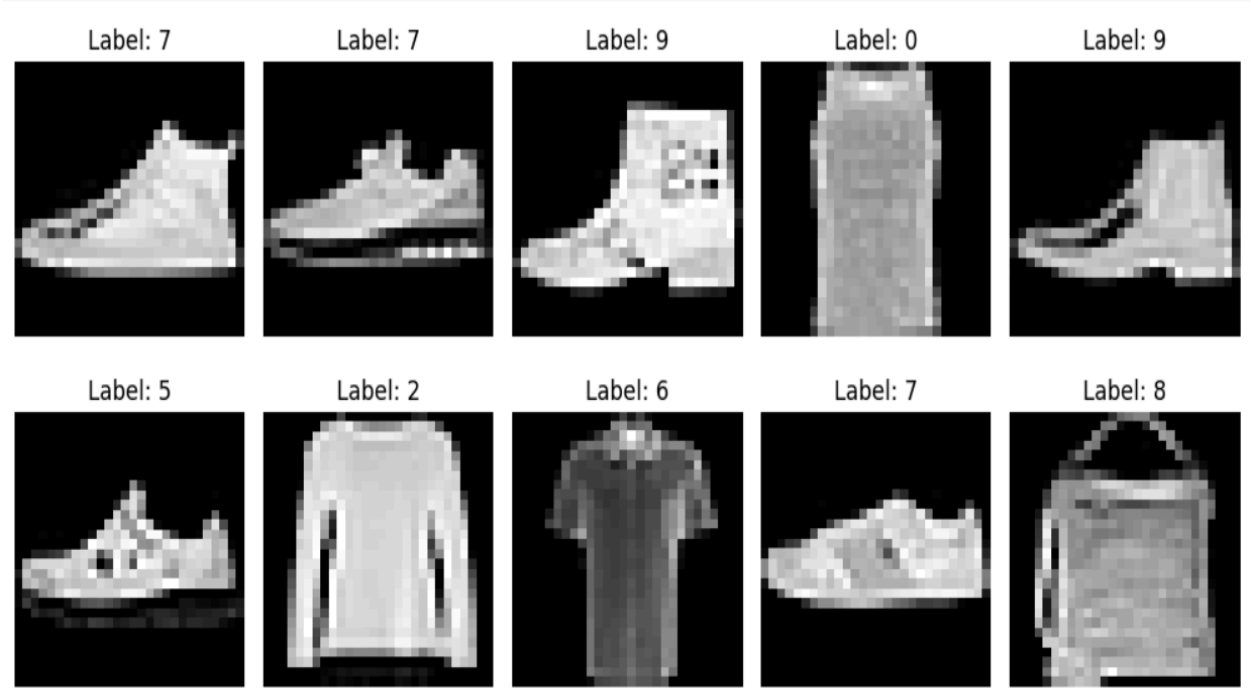


Section C

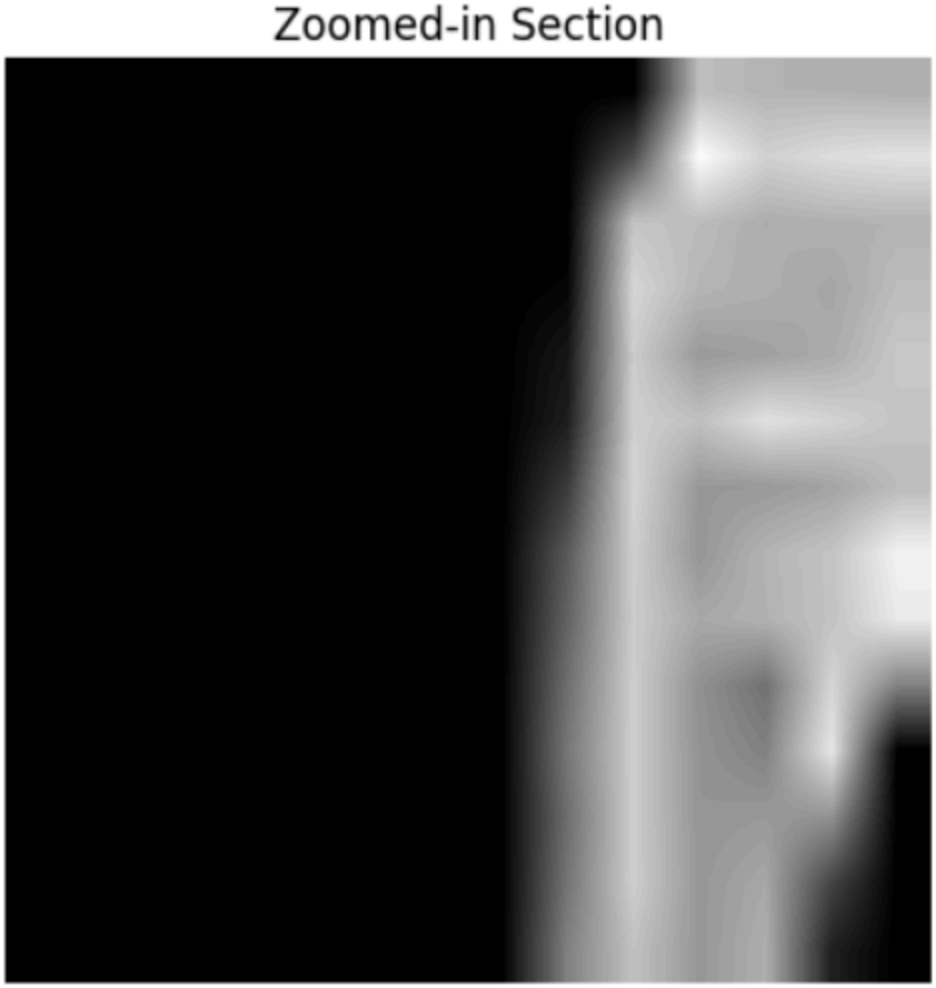
(a)Perform appropriate preprocessing on the data (for eg: normalization)
and visualize any 10 samples from the test dataset.

=> We normalised the dataset

```
X_train /= 255.0
X_test /= 255.0
```



I also zoomed in on images tio get a better view



(b) Train a MLP Classifier from sklearn's neural network module on the training dataset. The network should have 3 layers of size [128, 64, 32], should be trained for 100 iterations using an 'adam' solver with a batch size of 128 and learning rate of 2e-5. Train it using all the 4 activation functions i.e. 'logistic', 'tanh', 'relu' and 'identity'. For each activation function, plot the training loss vs epochs and validation loss vs epochs curves and comment on which activation function gave the best performance on the test set in the report.

=>

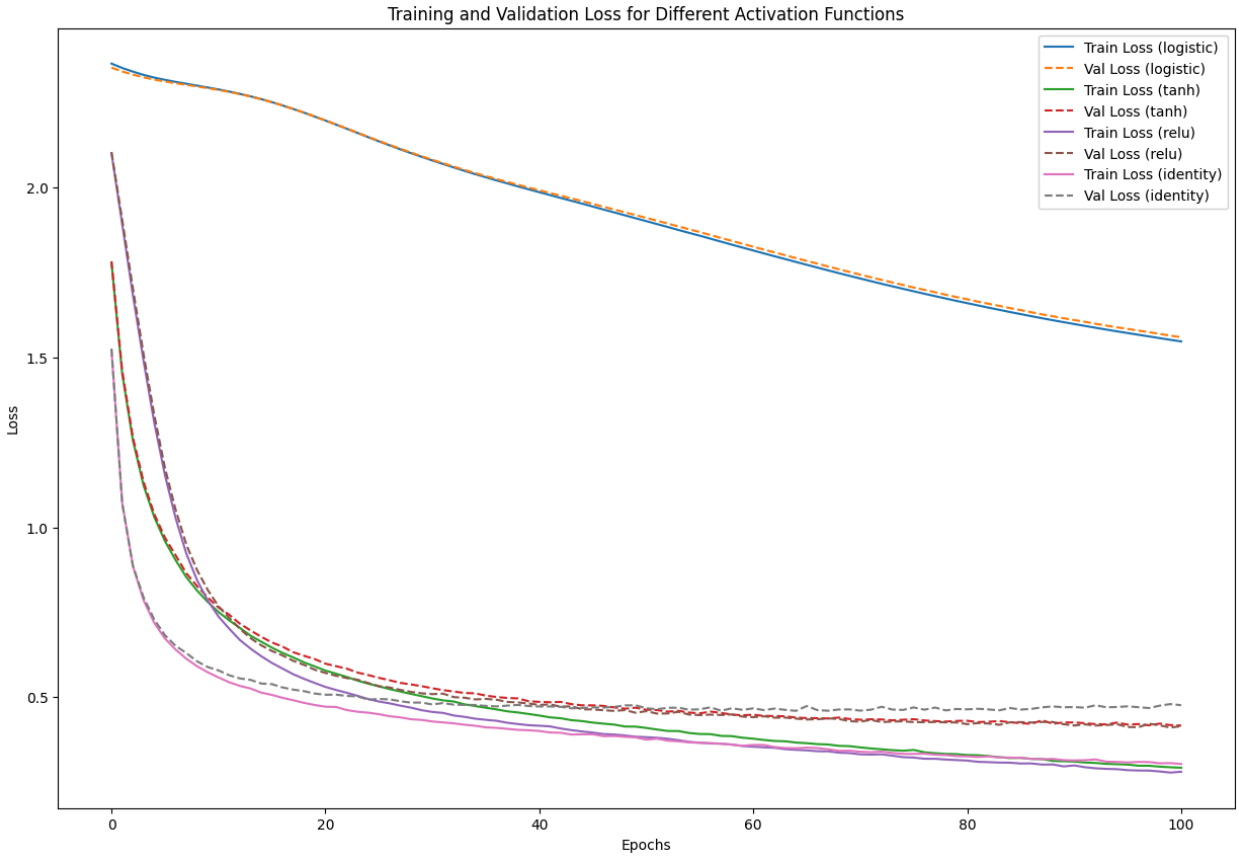
We trained a Multi-Layer Perceptron (MLP) Classifier using four different activation functions (logistic, tanh, relu, and identity) to understand their impact on model performance. The model's architecture consisted of three hidden layers with decreasing sizes [128, 64, 32], allowing it to learn progressively complex features. We used the adam optimizer for efficient training with a batch size of 128 and a low learning rate of 2e-5, all over 100 iterations.

Each activation function influences how the model learns. The logistic activation (sigmoid function) constrains outputs between 0 and 1, which can lead to issues like vanishing gradients in deeper layers. As expected, it yielded a low test accuracy of 0.2945, struggling to converge within the set iterations. The tanh function, which scales between -1 and 1, performed better with a test accuracy of 0.8480.

The relu (Rectified Linear Unit) function generally excels in deep networks by keeping positive gradients stable, which helped it achieve the highest accuracy at 0.8505. Finally, identity, a linear activation, also performed reasonably well with an accuracy of 0.8355, though not as effectively as relu.

```
Test accuracy for activation logistic: 0.2945
/usr/local/lib/python3.10/dist-packages/sklearn/neura
  warnings.warn(
Test accuracy for activation tanh: 0.8480
/usr/local/lib/python3.10/dist-packages/sklearn/neura
  warnings.warn(
Test accuracy for activation relu: 0.8505
Test accuracy for activation identity: 0.8355
```

COMBINED TRAINING AND VALIDATION LOSS GRAPHS FOR DIFFERENT ACTIVATION FUNCTIONS:



(c) Perform grid search using the best activation function from part 2 to find the best hyperparameters (eg: solver, learning rate, batch size) for the MLP classifier and report them in the report.

=>

We found Relu to be the best activation function

Output after running grid search with relu 👍

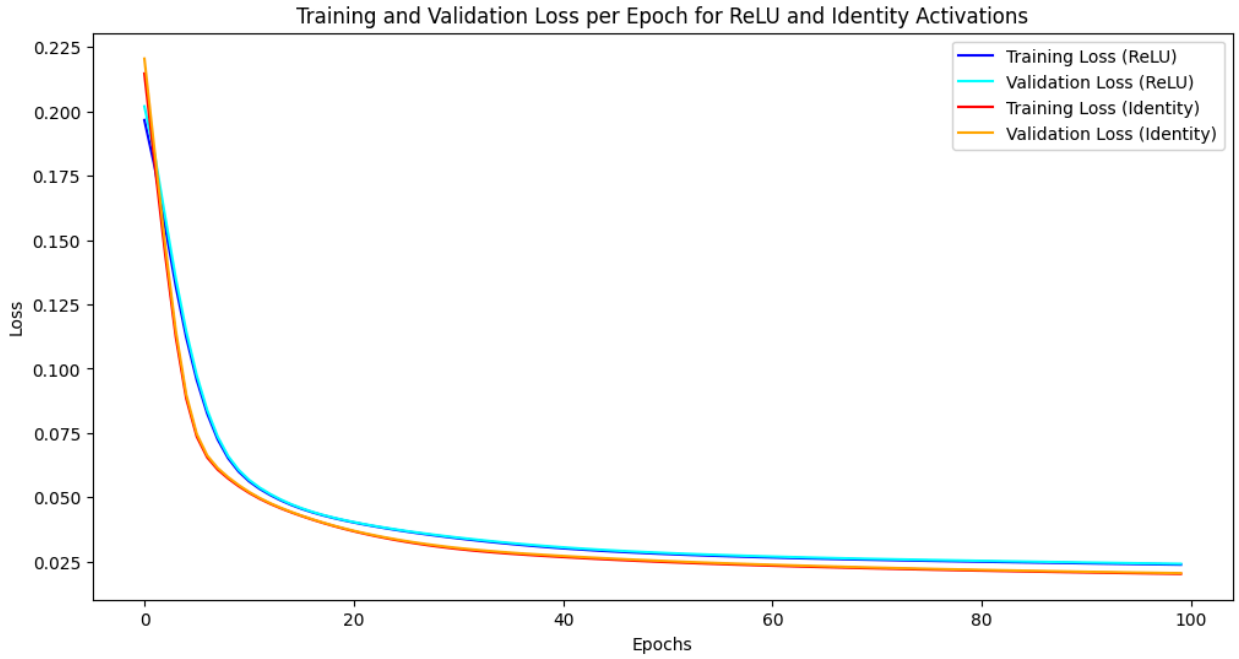
```
Best Hyperparameters from Grid Search:
{'batch_size': 64, 'hidden_layer_sizes': (128, 64, 32), 'learning_rate_init': 0.0001, 'max_iter': 100, 'solver': 'adam'}
Test Accuracy of Best Model: 0.8465
```

Here we can see 64 batch size with 128,64,32 hidden layer sizes and solver adam.

(d you need to train a MLPRegressor from sklearn's neural

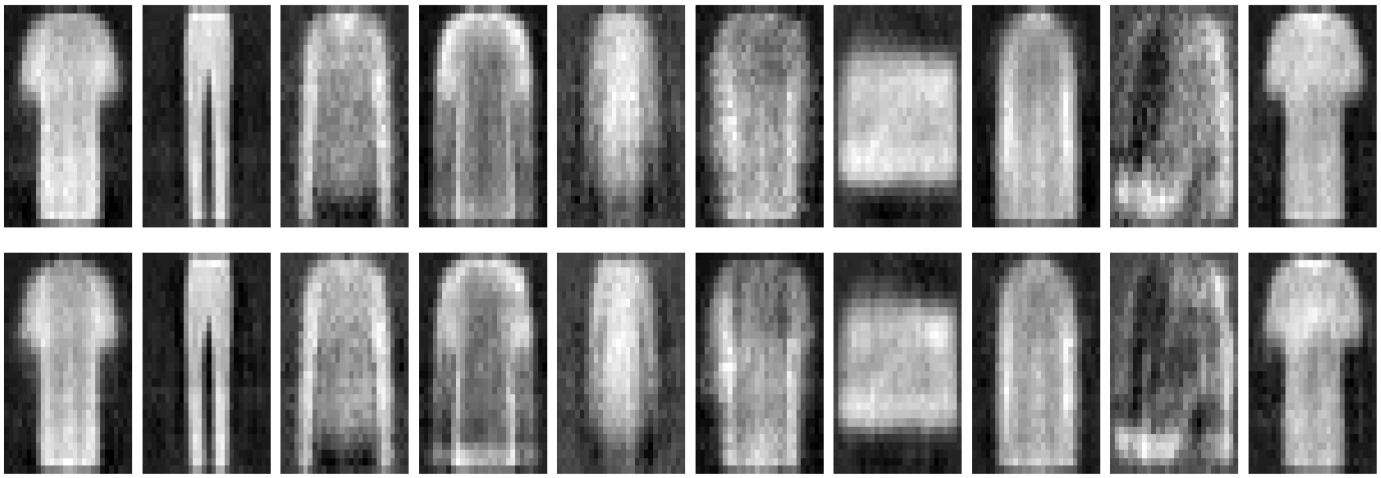
network module on a regeneration task:

Plotting



Here are the generated images 🟰

Generated Images Using ReLU and Identity Activation Functions



After training both networks with `relu` and `identity` activations, the differences in the regenerated images are quite obvious. The network using `relu` produces outputs that are sharper and contain more detail, more or less looking like the originals than those produced by the `identity` network. That seems to be due to the non-linear properties of `relu`, which helped capture the complex interaction between its inputs, which then really worked well in reconstructing the images. On the other hand, the network using `identity` ended up with the images more blurry and less detailly defined since it does not have the non-linearity required to capture finer features. Overall, `relu` did a much better job preserving the shapes and structure of the images compared to `identity`.

Part E results

Layer Configuration: Each classifier has two hidden layers, each of size `a`.

Training Configuration: Both models are configured with the `adam` solver, a learning rate of `2e-5`, and are trained for exactly 200 iterations.