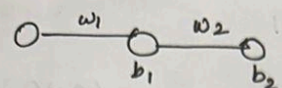
Assignment 3 - ML

Rishi Pendyala - 2022403

SECTION A

SECTION A

1. (a) input → [1,2,3]



output (y).

target $t = [3,4,5]$

loss function — MSE ; $\alpha = 0.01$

let y_1 be the output of the hidden layer
& $w_1 = 0.3$, $b_1 = 0.1$, $w_2 = 0.4$, $b_2 = 0.2$

for input $x = 1$ & target $t = 3$

forward propogation

$$y_1 = w_1 x_1 + b_1 = (0.3)(1) + 0.1$$

$$y_1 = 0.4$$

$$h = RELU(y_1) = max(0, 0.4) = 0.4$$

Output $y = h \cdot w_2 + b_2 = (0.4)(0.4) + 0.2$

$$y = 0.36$$

$$L = \frac{1}{2}(y-t)^2 = \frac{1}{2}(0.36-3)^2 = 3.4848$$

Backward propogation

$$L = \frac{1}{2}(h \cdot w_2 + b_2 - t)^2$$

$$\frac{\partial L}{\partial w_2} = (\underbrace{h w_2 + b_2}_{y} - t) \cdot h.$$

$$= (0.36 - 3)(0.4) = -1.056$$

$$\frac{\partial L}{\partial b_2} = (y-t)(1) = (0.36-3) = -2.64$$

As $y_1 = 0.4$ which is +ve

$$\frac{\partial h}{\partial y_1} = 1$$

So, $$\frac{\partial L}{\partial w_1} = (y-t)\left(w_2 \cdot \frac{\partial h}{\partial y_1}\right) \cdot x = (0.36-3)((0.4)\cdot 1)(1) = -1.056$$

$$\frac{\partial L}{\partial b_1} = (y-t)\left(w_2 \frac{\partial h}{\partial y_1}\right) = (0.36)((0.4)(1)) = -1.056$$

Parameters update

$$w_2 = w_2 - \alpha \frac{\partial L}{\partial w_2} = 0.4 - 0.01(-1.056)$$
$$= \underline{0.41056}$$

$$b_2 = b_2 - \alpha \frac{\partial L}{\partial b_2} = 0.2 - (0.01)(-2.64)$$
$$= 0.\underline{2264}$$

$$w_1 = w_1 - \alpha \frac{\partial L}{\partial w_1} = 0.3 - (0.01)(-1.056)$$

$$w_1 = 0.31056$$

$$b_1 = b_1 - \alpha \frac{\partial L}{\partial b_1} = 0.1 - 0.01(-1.056) = \underline{0.11056}$$

for input $x = 2$ & target $= 4$

$w_1 = 0.31056$, $b_1 = 0.11056$, $w_2 = 0.41056$
$b_2 = 0.2264$

forward pass

$$y_1 = xw_1 + b_1$$
$$= (2)(0.31056) + (0.11056) = \underline{0.73168}$$

$h = \text{RELU}(y_1) = \max(0, 0.73168) = 0.73168$

so, $y = hw_2 + b_2 = (0.73168)(0.41056) + 0.2264$

$= \underline{0.5268}$

loss, $L = \frac{1}{2}(y-t)^2 = \frac{1}{2}(0.5268-4)^2 = 6.0326$

$$\frac{\partial L}{\partial w_2} = (y-t)\cdot h = (0.5268-4)(0.73168)$$

$$= \underline{-2.5404}$$

$$\frac{\partial L}{\partial b_2} = (y-t) = 0.5268-4 = \underline{-3.4782}$$

$$\frac{\partial L}{\partial w_1} = (y-t)\,w_2 \cdot \frac{\partial h}{\partial y_1} \cdot x = (0.5268-4)(0.41056)(1)(2)$$

$$= \underline{-2.8532}$$

$$\frac{\partial L}{\partial b_1} = (y-t)\,w_2 \frac{\partial h}{\partial y_1} = (-3.4732)(0.41056)$$

$$= \underline{-1.4266}$$

Update parameters

$$w_2 = w_2 - \alpha\frac{\partial L}{\partial w_2} = 0.41056 - 0.01(-2.5404)$$

$$= \underline{0.4360}$$

$$b_2 = b_2 - \alpha\frac{\partial L}{\partial b_2} = \underline{0.2611}$$

$$w_1 = w_1 - \alpha\frac{\partial L}{\partial w_1} = 0.3391$$

$$b_1 = b_1 - \alpha\frac{\partial L}{\partial b_1} = 0.1248$$

Updated parameters

$\hookrightarrow w_1 = 0.3391, \quad b_1 = 0.1248, \quad w_2 = 0.4360, \quad b_2 = 0.2611$

Now for $x = 3$ & $t = 5$

$$y_1 = xw_1 + b_1 = (3)(0.3391) + (0.1248) = \underline{1.1421}$$

$$h = \text{RELU}(y_1) = \max(0, 1.1421) = \underline{1.1421}$$

So, $$y = hw_2 + b_2 = (1.1421)(0.4360) + (0.2611)$$

$$= \underline{0.7591}$$

loss $$L = \frac{1}{2}(y-t)^2 = \frac{1}{2}(0.7591 - 5)^2 = 8.9936$$

Back propogation,

$$\frac{\partial L}{\partial w_2} = (y-t) \cdot h = (0.7591 - 5)(1.1421) = \underline{-4.8459}$$

$$\frac{\partial L}{\partial b_2} = (y-t) = 0.7591 - 5 = -4.2409$$

$$\frac{\partial L}{\partial w_1} = (y-t) w_2 \frac{\partial h}{\partial y_1} x = (0.7591 - 5)(0.4360)(1)(3)$$

$$= -5.5503$$

$$\frac{\partial L}{\partial b_1} = (y-t) w_2 \frac{\partial h}{\partial y_1} = (-4.2409)(0.4360) = \underline{-1.8506}$$

Upd

-0.2611

$$w_2 = w_2 - \alpha \frac{\partial L}{\partial w_2} = (0.4360) - (0.01)(-4.8459)$$
$$= \underline{\underline{0.4845}}$$

$$b_2 = b_2 - \alpha \frac{\partial L}{\partial b_2} = 0.2611 - (0.01)(-4.2409)$$
$$= \underline{\underline{0.3035}}$$

$$w_1 = w_1 - \alpha \frac{\partial L}{\partial w_1} = 0.339 - 0.01(-5.5503)$$
$$= \underline{\underline{0.3946}}$$

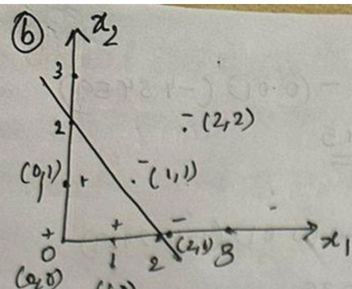
$$b_1 = b_1 - \alpha \frac{\partial L}{\partial b_1} = 0.1248 - 0.01(-1.8506)$$
$$= \underline{\underline{0.1433}}$$

Final updated

$\hookrightarrow$ $w_1 = 0.3946$ $\quad$ $b_1 = 0.1433$

$w_2 = 0.4845$ $\quad$ $b_2 = 0.8035$

(b)



(a) Yes, the points are linearly separable.

(b) $w_1 x_1 + w_2 x_2 + b = y$

(0,0) & (2,2) does not have any impact on boundary.

+ve class $\Rightarrow$

$(0,1) \rightarrow w_2 + b = 1$

$(1,0) \rightarrow w_1 + b = 1$

$\underline{w_2 - w_1 = 0} \Rightarrow w_1 = w_2$

$w_1 + b = 1$

-ve class $\Rightarrow$

$(2,0) \rightarrow 2w_1 + b = -1$

$(1,1) \rightarrow w_1 + w_2 + b = -1$

$b = 4 - 1 = 3$

$w_1 + 1 = -1$

$w_1 = -2$

so

$w_2 = -2$

$\therefore w_1 = w_2 = -2$ & $b = 8$

Weight vector = $\begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$

Decision boundary

$$-2x_1 - 2x_2 + 3 = 0$$

Finding the support vector

$y_i(w^T x + b) \geq 1 \longrightarrow ||w|| = \sqrt{(-2)^2 + (-2)^2} = 2\sqrt{2}$

$f(x) = w^T x + b$

put p

(1,0)

(0,1)

Negat

$\therefore$

put points in $f(x)$ & check if $y_i\,(w^Tx+b) = 1$ or not

$(1,0) \Rightarrow f(x) = -2(1) + 3 = 1 \Rightarrow y_i f(x) = +1$

$(0,1) \Rightarrow f(x) = -2(1) + 3 = 1 \Rightarrow y_i f(x) = 1$

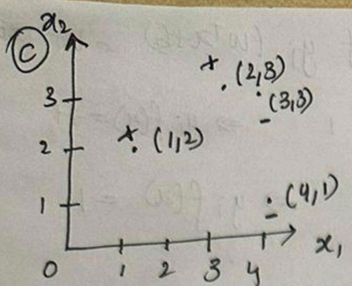
Negative class

$(1,1) \Rightarrow f(x) = -2 - 2 + 3 = -1;\ y_i f(x) = 1$

$(2,0) \Rightarrow f(x) = 2(-2) + 3 = -1 \Rightarrow y_i(f(x)) = 1$

$\therefore$ $(1,0)$, $(0,1)$, $(1,1)$, $(2,0)$ are the support vectors

(c)



boundary $\Rightarrow w x + b = 0$

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

for support vector $\vec{x}_i$

$y_i(w^T x + b) = 1.$

margin $\Rightarrow$ distance from support vector to boundary on both sides.

(a) $\text{margin} = \dfrac{2y_i(w^T x + b)}{\|w\|} = \dfrac{2}{\|w\|}$; $\|w\| = \sqrt{(-2)^2 + 0^2} = 2$

$\text{margin} = \dfrac{2}{2} = 1$

(b) $w^T = \begin{bmatrix} -2 \\ 0 \end{bmatrix}$

for support vectors $\quad y_i(w^T x + b) = 1.$

i) $x = [1\ 2] \Rightarrow 1\left([-2\ 0]\begin{bmatrix}1\\2\end{bmatrix} + 5\right) = -2+5 = 3$

ii) $x = [2,3] \Rightarrow [-2\ 0]\begin{bmatrix}2\\3\end{bmatrix} + 5 = -4+5 = 1$

(iii) $x = [3\ 3] \Rightarrow [-2\ 0]\begin{bmatrix}3\\3\end{bmatrix} + 5 = -6+5 = 1$

(iv) $x_1 = [4\ 1] \Rightarrow [-2\ 0]\begin{bmatrix}4\\1\end{bmatrix} + 5 = -8+5 = 3$

Points [2,3] & [3,3] are support vectors

(c) $x_1 = 1$, $x_2 = 3$

$$w^T x + b = [-2\ 0]\begin{bmatrix}1\\3\end{bmatrix} + 5 = -2+5 = 3$$

$$y_i(w^T x + b) \geq 1 \Rightarrow y_i(3) \geq 1$$

$\therefore$ It belongs to the positive class.

SECTION B

```
        self.N = N

        self.hidden_layer_sizes = hidden_layer_sizes

        self.alpha = alpha

        self.epochs = epochs

        self.batch_size = batch_size

        self.activation = activation.lower()

        self.weight_init = weight_init.lower()

        self.patience = patience
```

These are the attributes that are initialised as parameters.

```
def predict(self, x, intercept=True):
    return np.argmax(self.predict_proba(x, intercept), axis=1)

def predict_proba(self, x, intercept=True):
    x = np.hstack((np.ones((x.shape[0], 1)), x)) if intercept else x
    self.forward_propagate(x)
    return self.activations[-1]

def score(self, x, y, intercept=True):
    return np.mean(self.predict(x, intercept) == y)
```

predict_proba: computes probability distribution over all classes for each input sample. It calls the forward_propagate method to process the input data through layers of the neural network.

predict: returns the predicted class for the input data. It calls predict_proba to get the class probabilities and chooses the class with the highest probability for each sample.

score: calculates the model's accuracy on the given dataset. It compares the predicted labels and true labels y, returning the percentage of correct predictions.

Below are the activation functions and their gradients:

```
def activate(self, x):
    activations = {
        "sigmoid": 1 / (1 + np.exp(-np.clip(x, -500, 500))),
        "tanh": np.tanh(x),
        "relu": np.maximum(0, x),
        "leakyrelu": np.where(x > 0, x, 0.01 * x),
    }
    return activations[self.activation]

def gradient(self, x):
    gradients = {
        "sigmoid": self.activate(x) * (1 - self.activate(x)),
        "tanh": 1 - np.tanh(x) ** 2,
        "relu": np.where(x > 0, 1, 0),
        "leakyrelu": np.where(x > 0, 1, 0.01),
    }
    return gradients[self.activation]
```

The weight initialization function:

```
def _initialize_weights(self):
    layers = [self.x_train.shape[1]] + list(self.hidden_layer_sizes) + [self.classes]
    self.weights = []

    for i in range(len(layers) - 1):
        if self.activation in ['relu', 'leakyrelu']:
            scale = np.sqrt(2.0 / layers[i])
        else:
            scale = np.sqrt(2.0 / (layers[i] + layers[i + 1]))

        if self.weight_init == "zero":
            self.weights.append(np.zeros((layers[i + 1], layers[i])))
        elif self.weight_init == "random":
            self.weights.append(np.random.uniform(-scale, scale, (layers[i + 1], layers[i])))
        else:
            self.weights.append(np.random.normal(0, scale, (layers[i + 1], layers[i])))
```

The data was then split into 80:10:10 proportions for train-validation-test.

Data Normalisation was done by dividing the data by 255

Then the models were run with the required parameters mentioned:

(a) Number of hidden layers = 4.

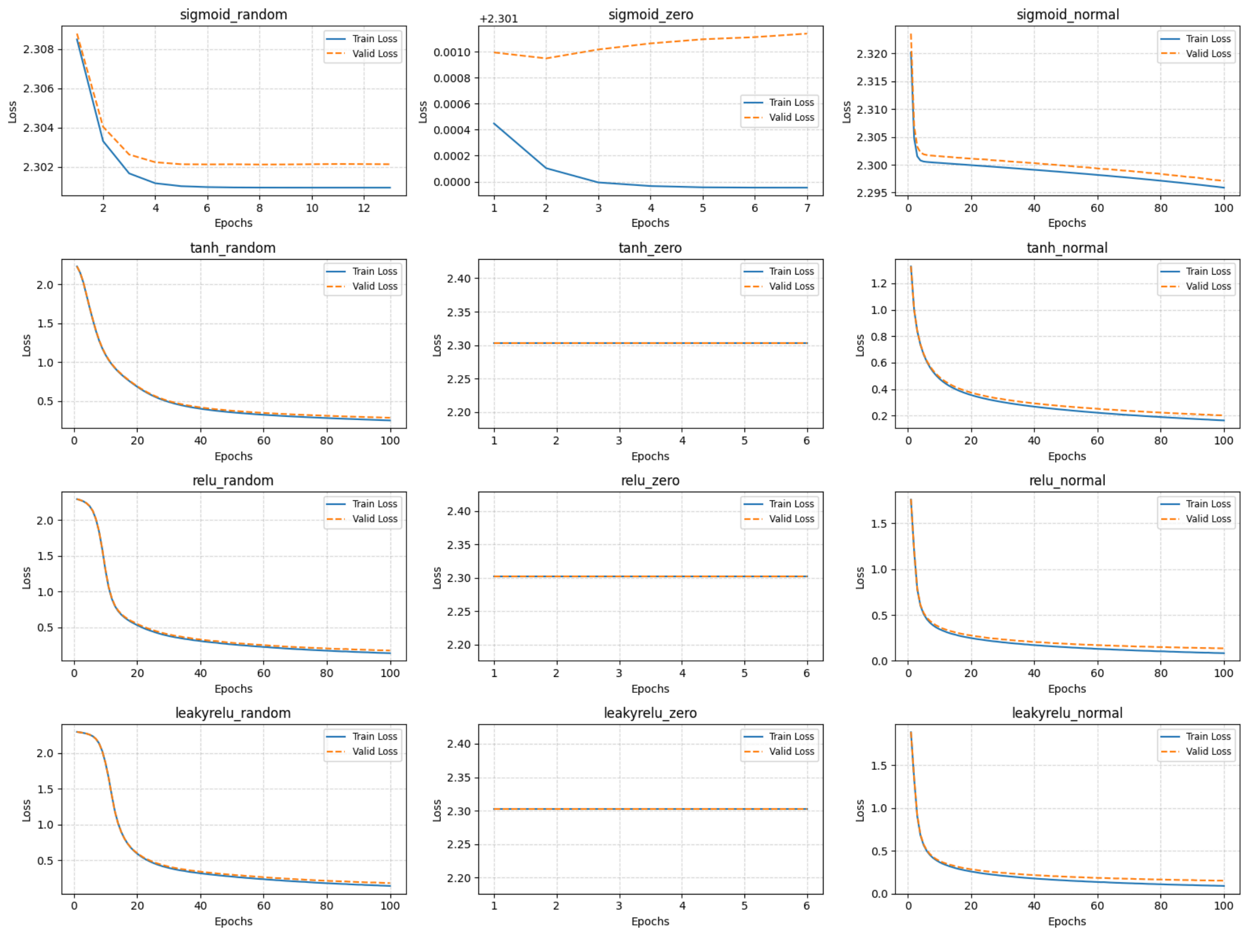
(b) Layer sizes = [256,128,64,32].

(c) Number of epochs = 100 (can be less if computation is taking too long).

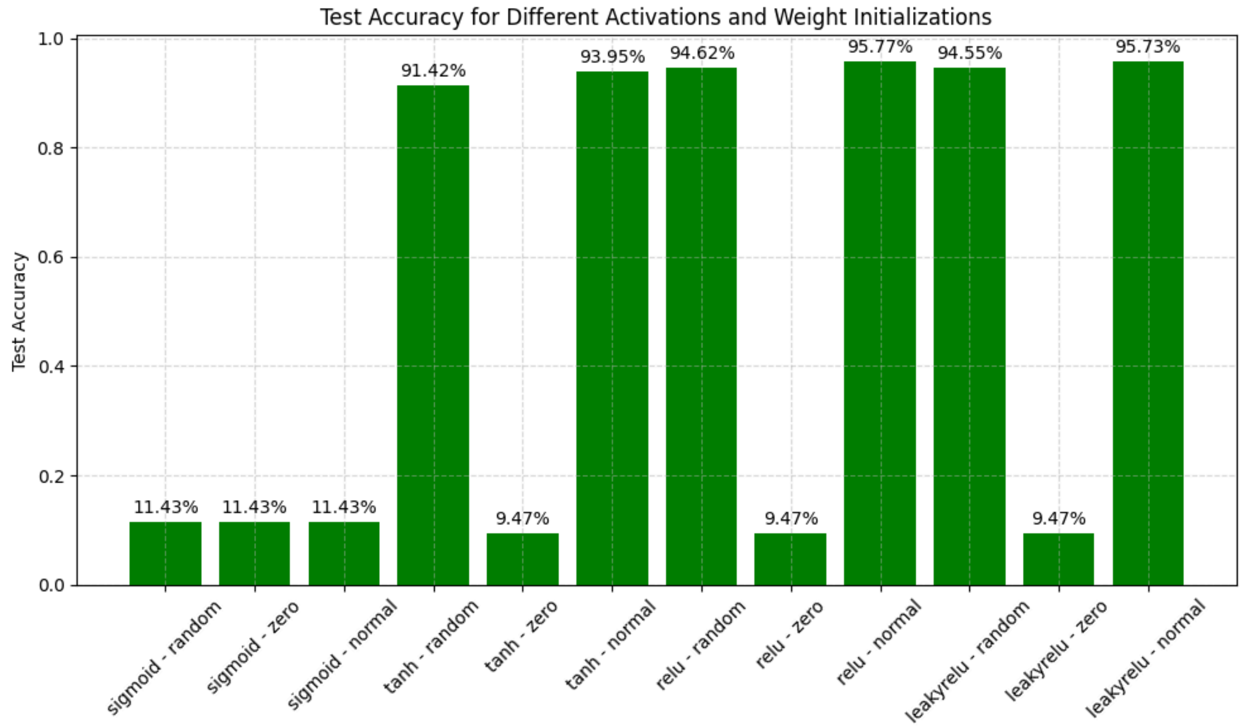
(d) Batch size = 128 (or any other appropriate batch size if taking too long).

(e) Learning rate = 2e-3.

The models were stored to .pkl files and the results were:



sigmoid_random
sigmoid_zero
sigmoid_normal
tanh_random
tanh_zero
tanh_normal
relu_random
relu_zero
relu_normal
leakyrelu_random
leakyrelu_zero
leakyrelu_normal
Train Loss
Valid Loss
Loss
Epochs
+2.301



Findings:

Best performance: ReLU/Leaky ReLU with normal initialization (95.77%, 95.73%)

ReLU/Leaky ReLU with random initialization also strong

Poor performance: Sigmoid/Tanh with zero initialization (below 11.43%, 9.47%)

Loss trends: ReLU/Leaky ReLU – efficient convergence

Sigmoid/Tanh – slow or unstable convergence

Highlights importance of activation and initialization choice

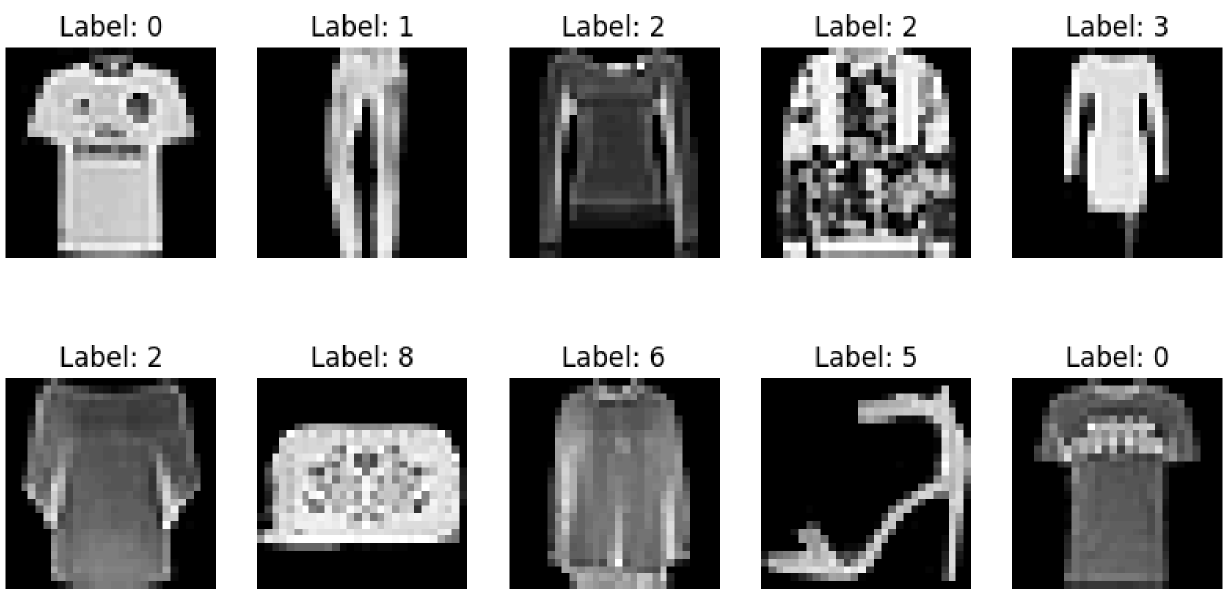
SECTION C

The first 8000 images from the train data made the train set

The first 2000 images from the test data made the test set.
A validation set was created from the train data.

Normalisation of data by scaling by 255

Visualisation:



Trained a MLP Classifier using :

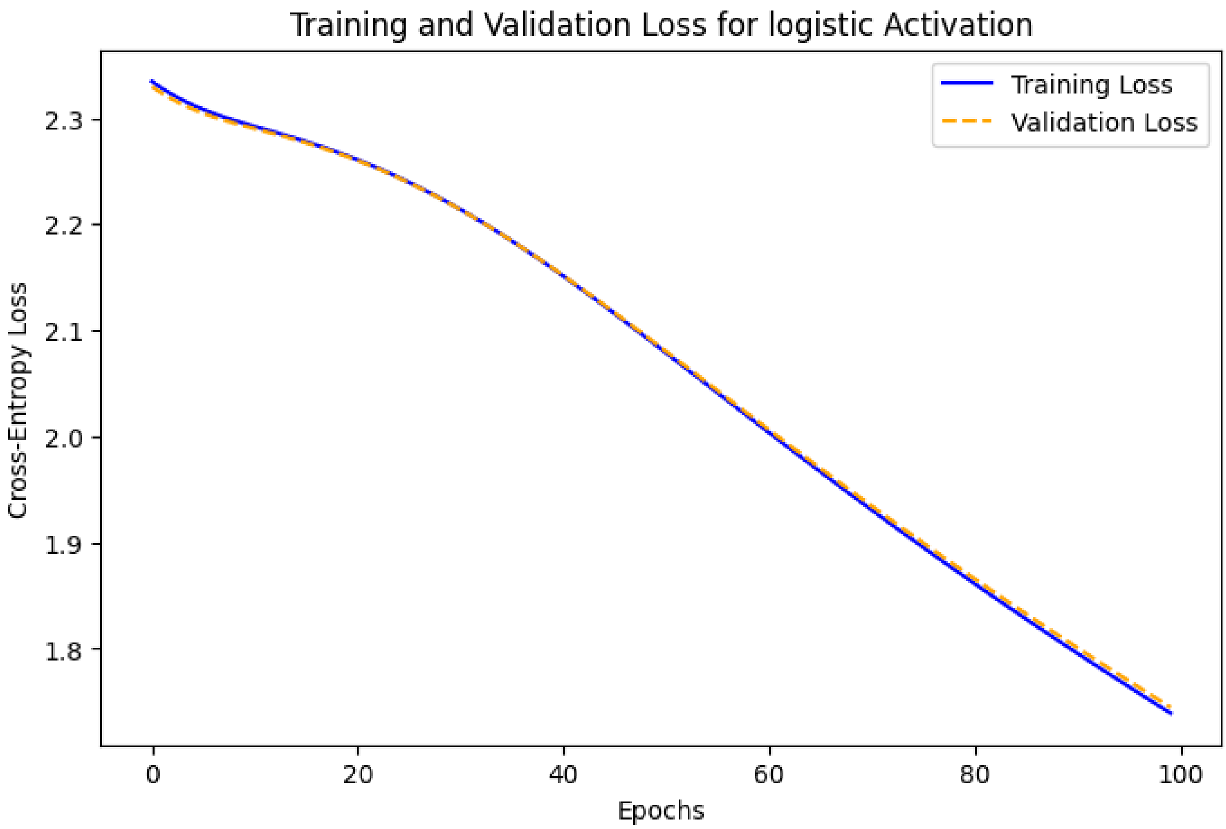
hidden_layers = [128, 64, 32]

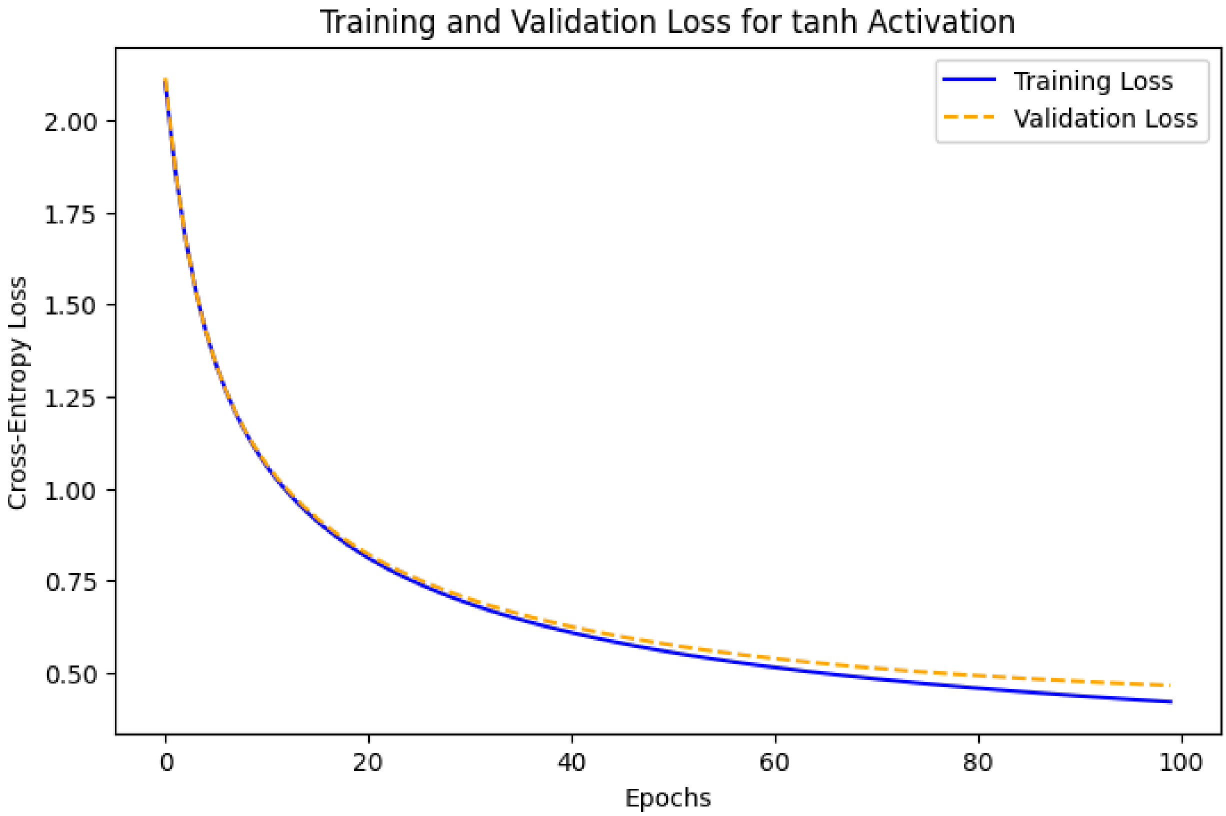
max_iter = 100

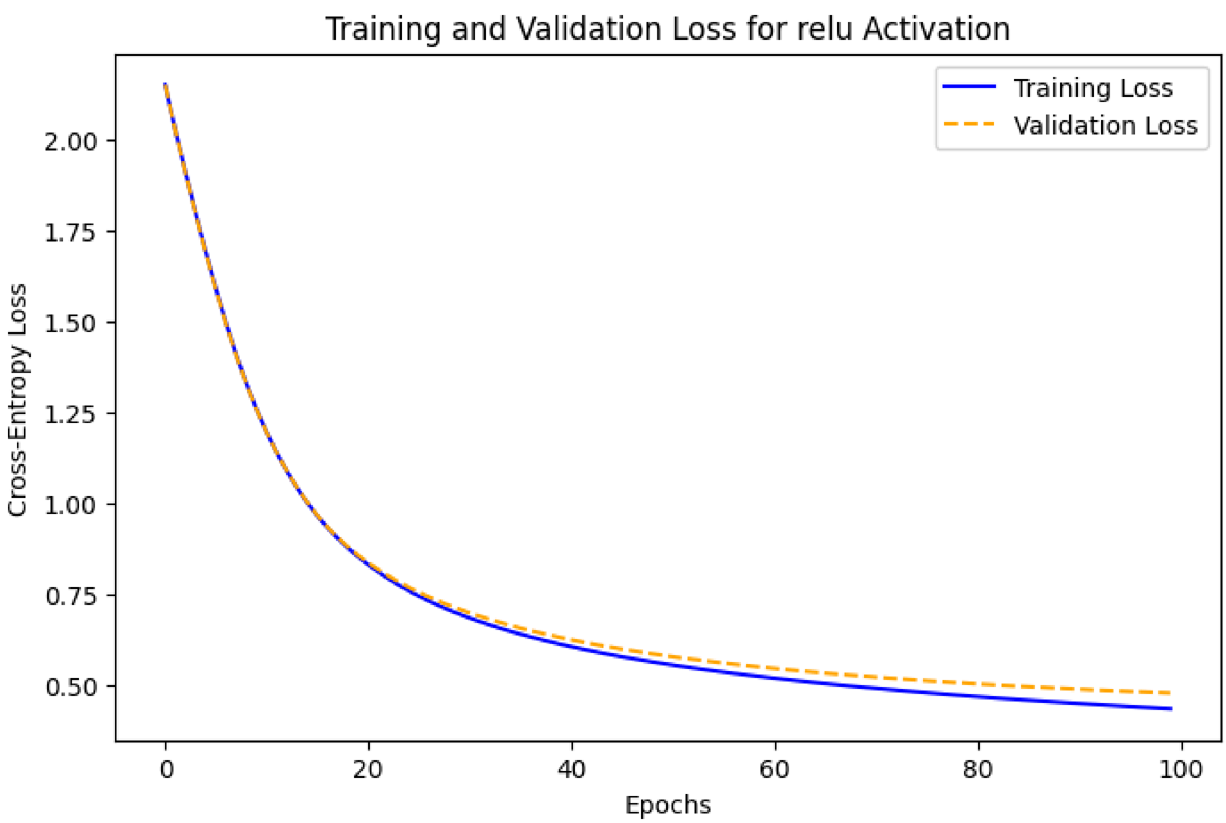
batch_size = 128

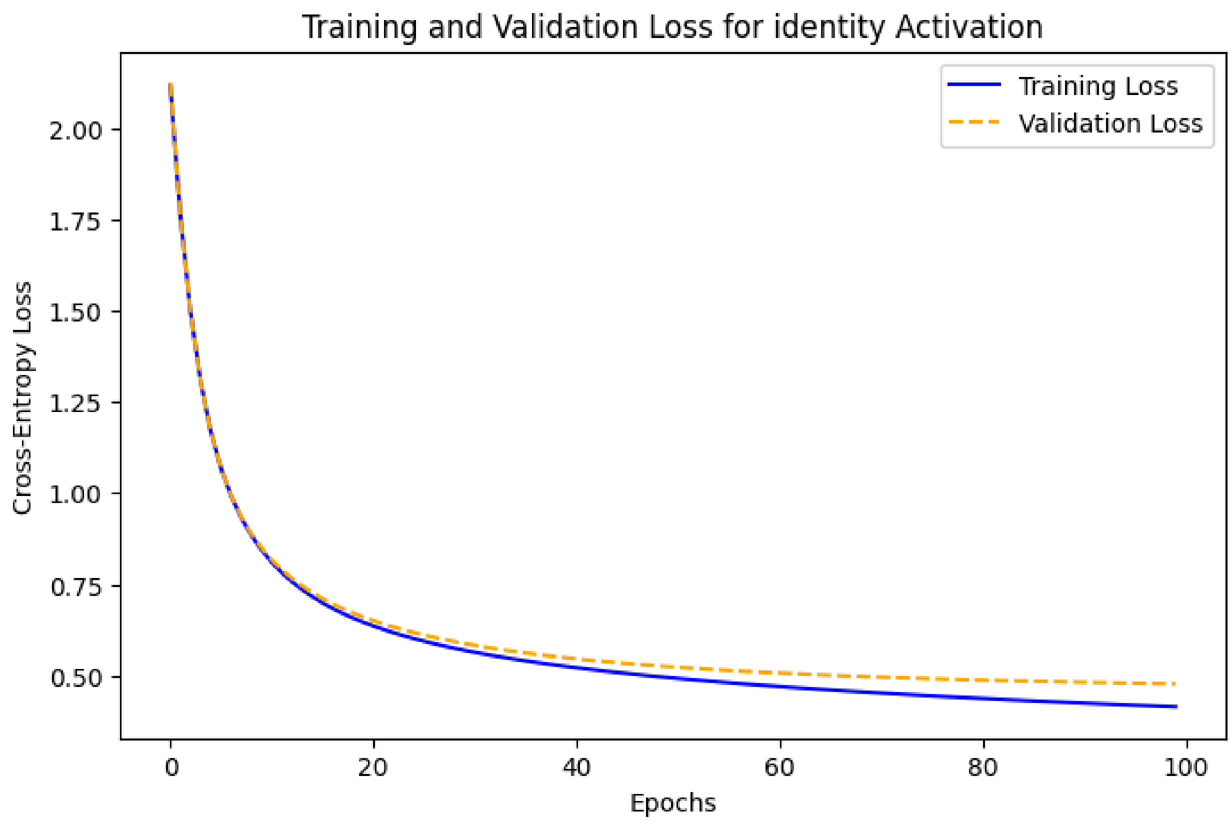
learning_rate = 2e-5

activations = ['logistic', 'tanh', 'relu', 'identity']


Training and Validation Loss for logistic Activation
Training Loss
Validation Loss
Cross-Entropy Loss
Epochs
2.3
2.2
2.1
2.0
1.9
1.8
0
20
40
60
80
100


Training and Validation Loss for tanh Activation
Training Loss
Validation Loss
Cross-Entropy Loss
Epochs
0
20
40
60
80
100
0.50
0.75
1.00
1.25
1.50
1.75
2.00


Training and Validation Loss for relu Activation
Training Loss
Validation Loss
Cross-Entropy Loss
Epochs
0
20
40
60
80
100
0.50
0.75
1.00
1.25
1.50
1.75
2.00



Best activation function was tanh

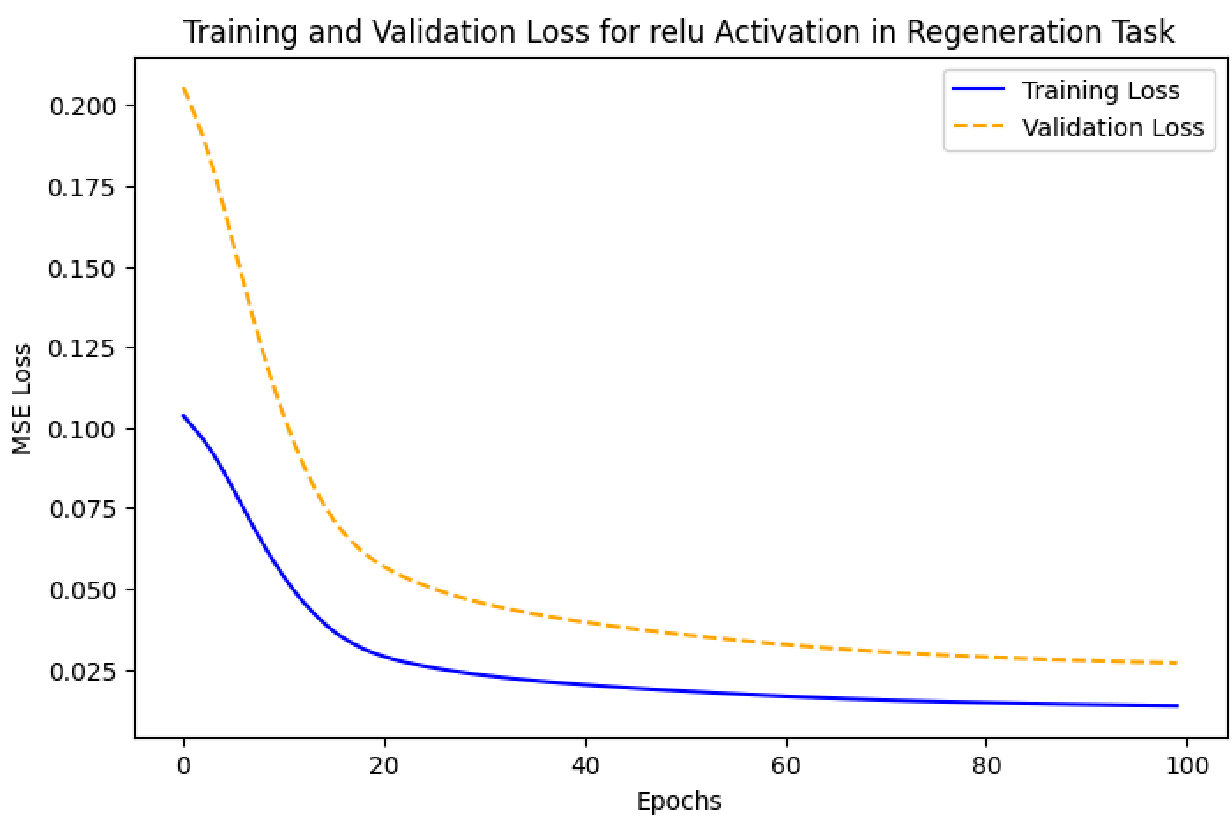
Grid Search:

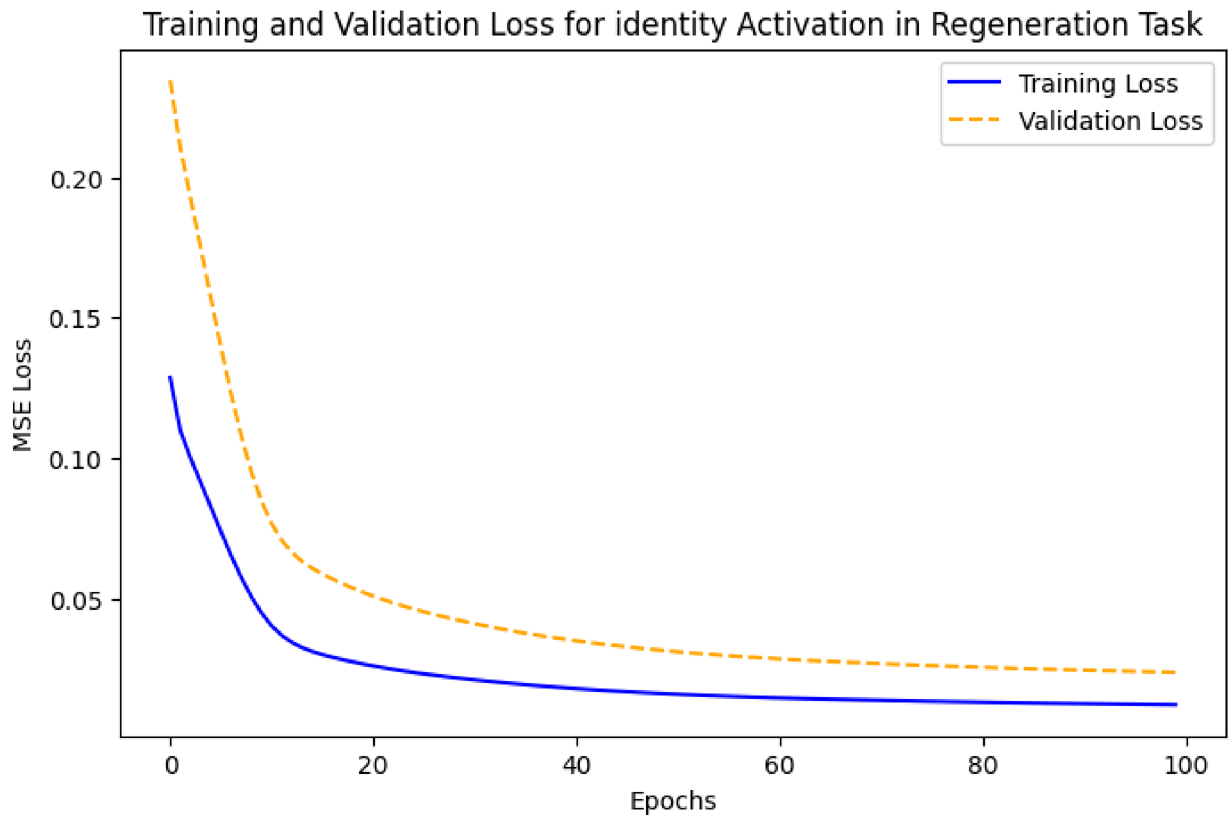
```
param_grid = {
    'solver': ['adam', 'sgd'],
    'learning_rate_init': [1e-4, 1e-5, 1e-6],
    'batch_size': [64, 128, 256]
}
```

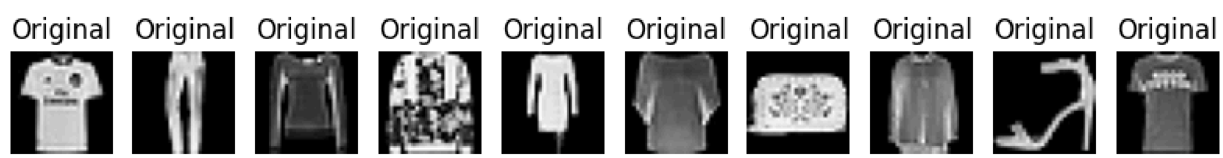
A grid search was performed the best parameters were:

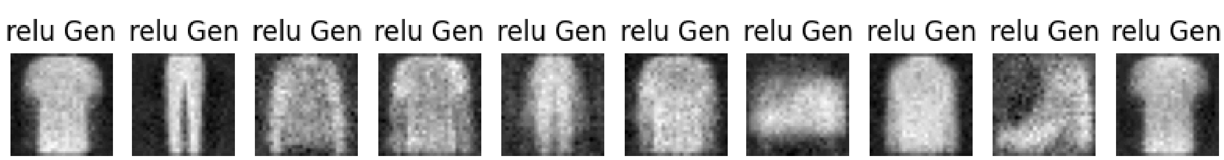
```
best_params = {
    'hidden_layer_sizes': [128, 64, 32],
    'activation': 'tanh',
    'solver': 'adam',
    'batch_size': 128,
    'learning_rate_init': 0.0001,
    'max_iter': 100
}
```

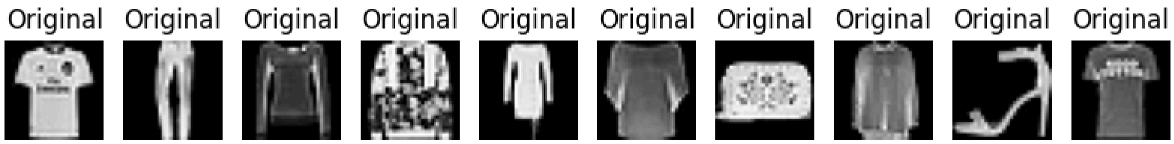
MLP Regressor:

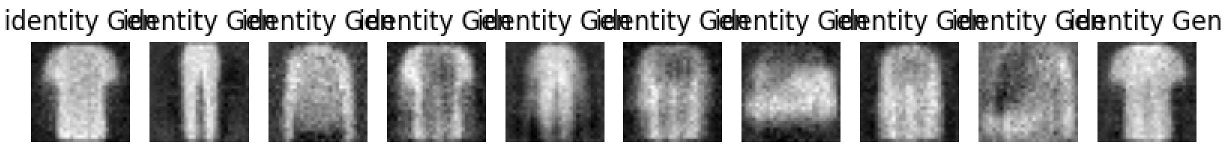



Training and Validation Loss for identity Activation in Regeneration Task
Training Loss
Validation Loss
MSE Loss
0.20
0.15
0.10
0.05
0
20
40
60
80
100
Epochs


Original
Original
Original
Original
Original
Original
Original
Original
Original
Original


relu Gen
relu Gen
relu Gen
relu Gen
relu Gen
relu Gen
relu Gen
relu Gen
relu Gen
relu Gen





MLP Classifier:

Extract feature vectors from trained model using regressor.predict

Use extracted features for both training and testing sets

Define two smaller MLP classifiers (2 hidden layers, size 32 each)

Use best activation, Adam optimizer, low learning rate (2e-5)

Train both classifiers on reduced training data

Test accuracy for smaller MLP Classifier 1: 0.7435

Test accuracy for smaller MLP Classifier 2: 0.7435

- Feature extraction transforms raw images into compact vectors
- Vectors capture key patterns identified by the neural network
- Simplifies data for easier classification
- Decent accuracies due to high-quality feature representation
- Smaller MLPs can classify effectively without large networks
- Essential patterns already embedded in the feature vectors