```
#importing the dataset and libraries
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.datasets import mnist
import plotly.graph_objects as go
```

```
#Checking for GPU availability

if tf.test.gpu_device_name():
    print('GPU device found: {}'.format(tf.test.gpu_device_name()))
    device = '/device:GPU:0'
else:
    print('No GPU device found. Using CPU.')
    device = '/device:CPU:0'
```

```
GPU device found: /device:GPU:0
```

```
(x_train, y_train),(x_test, y_test) = mnist.load_data() #Splitting data into test a

x_train = x_train.reshape(-1, 784) / 255.0             #Flattening the (28X28) dat
x_test = x_test.reshape(-1, 784) / 255.0               #and normalizing the greysc
```

```
x_validation, x_train = x_train[:5000], x_train[5000:]  #5000 data points for valic
y_validation, y_train = y_train[:5000], y_train[5000:]
```

```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

```
((55000, 784), (10000, 784), (55000,), (10000,))
```

```
y_train = tf.keras.utils.to_categorical(y_train)
y_validation = tf.keras.utils.to_categorical(y_validation)
y_test = tf.keras.utils.to_categorical(y_test)
```

```
x_train.shape, x_validation.shape, x_test.shape, y_train.shape, y_validation.shape,
```

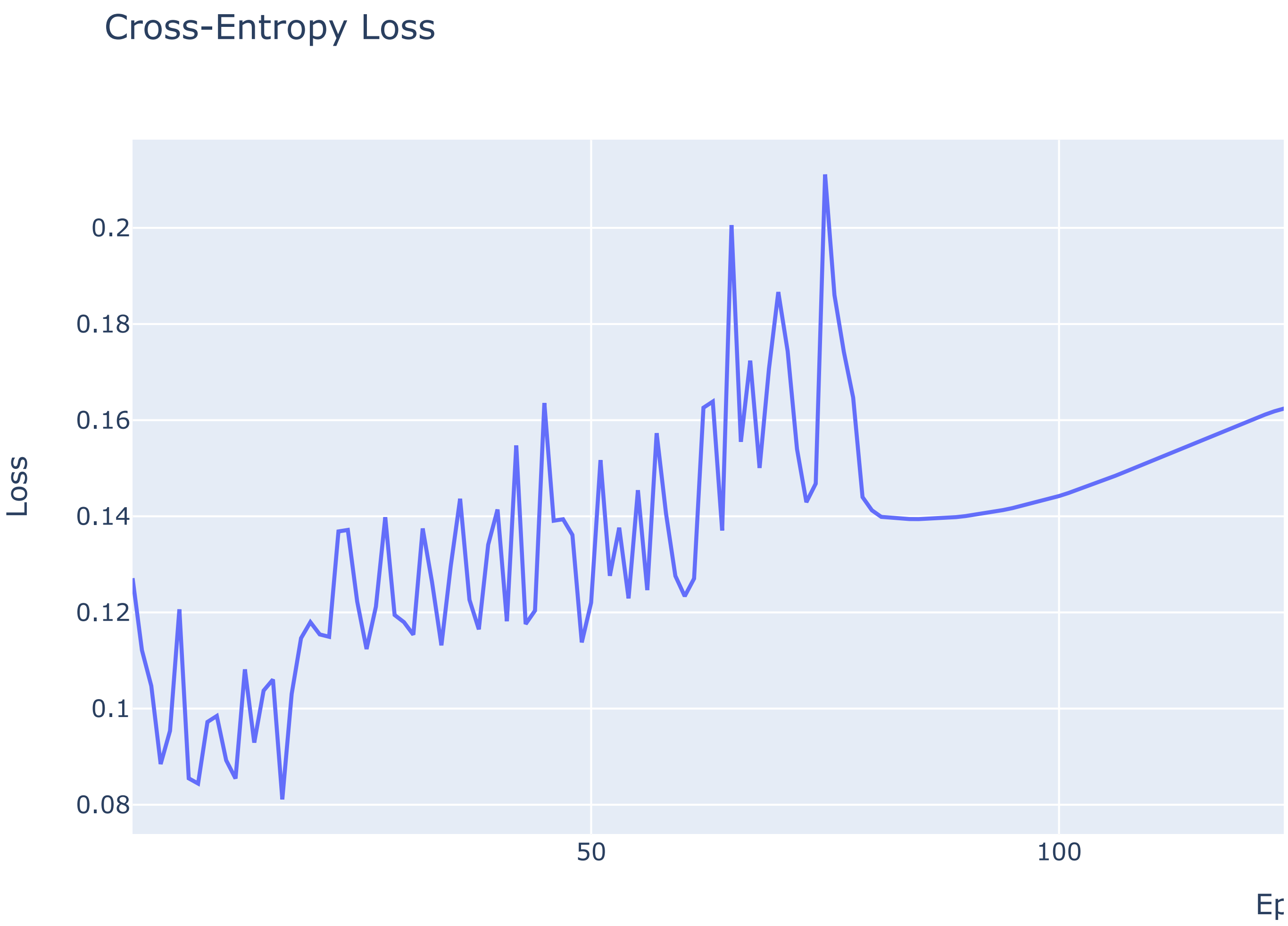
```
((55000, 784), (5000, 784), (10000, 784), (55000, 10), (5000, 10), (10000,
10))
```

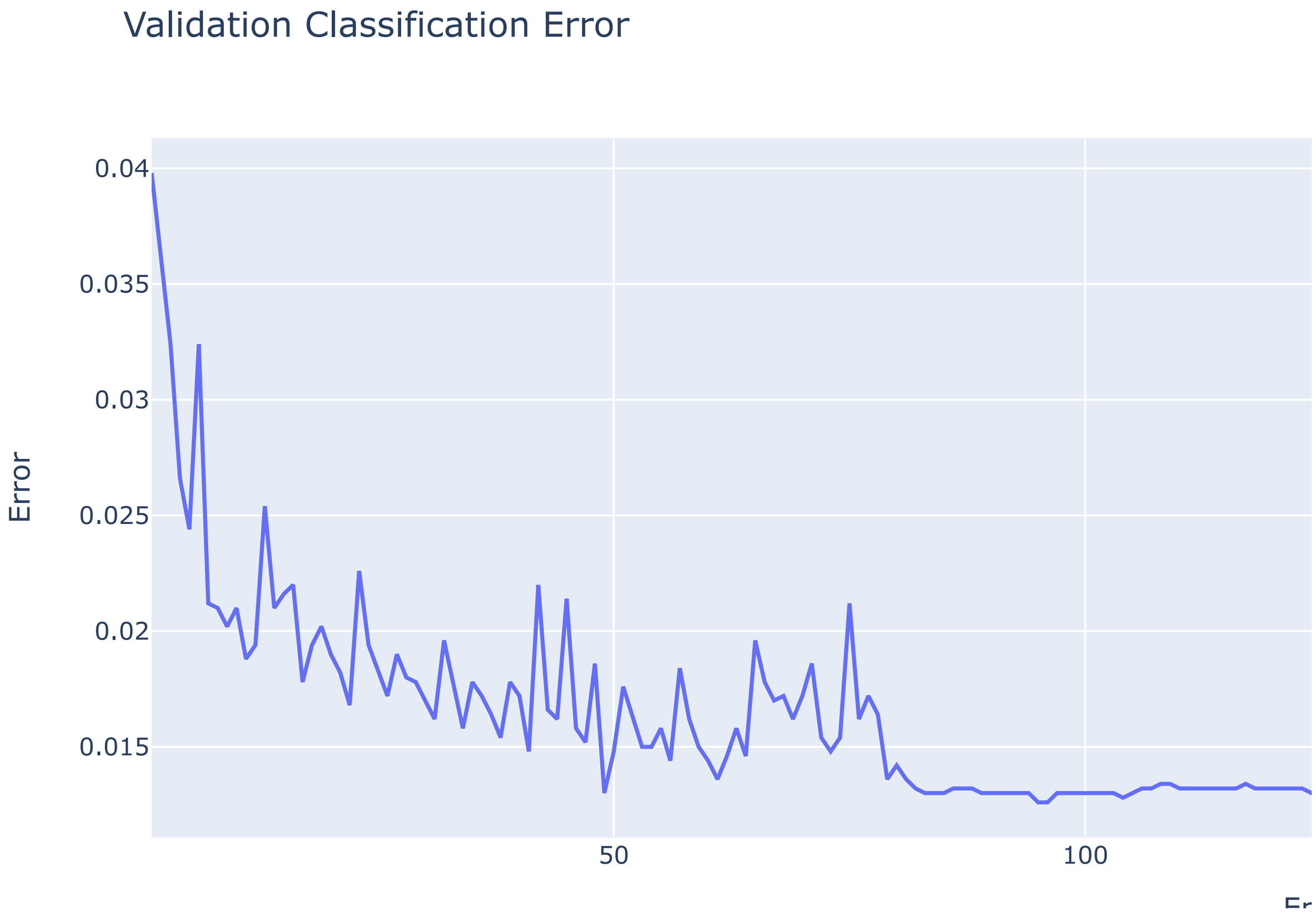
```
#Neural Network with 3 layers, relu as activation for hidden layers
# with softmax as activation for the output layer
model = Sequential()
model.add(Dense(500, activation='relu', input_shape=(784,)))
model.add(Dense(500, activation='relu'))
model.add(Dense(10, activation='softmax'))


#Compiling the model with Adam Optimizer with categorical cross entropy as the loss
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy


#Setting batch size = 128 and epochs to 250
batch_size = 128
epochs = 250


with tf.device(device):

    loss_values = []
    error_values = []

    #each iteration of epochs
    for epoch in range(epochs):

        #each iteration of batch
        for batch_start in range(0, len(x_train), batch_size):
            batch_end = batch_start + batch_size
            x_batch = x_train[batch_start:batch_end]
            y_batch = y_train[batch_start:batch_end]

            model.train_on_batch(x_batch, y_batch)

        #calculating the loss and accuracy
        loss, accuracy = model.evaluate(x_validation, y_validation, batch_size=batc

        #calculating the error
        error = 1 - accuracy
        loss_values.append(loss)
        error_values.append(error)

        print(f"Epoch {epoch+1}/{epochs} - Loss: {loss:.4f} - Error: {error:.4f}")
```

```
Epoch 1/250 - Loss: 0.1271 - Error: 0.0398
Epoch 2/250 - Loss: 0.1121 - Error: 0.0358
Epoch 3/250 - Loss: 0.1047 - Error: 0.0324
Epoch 4/250 - Loss: 0.0885 - Error: 0.0266
Epoch 5/250 - Loss: 0.0954 - Error: 0.0244
Epoch 6/250 - Loss: 0.1207 - Error: 0.0324
Epoch 7/250 - Loss: 0.0855 - Error: 0.0212
Epoch 8/250 - Loss: 0.0844 - Error: 0.0210
Epoch 9/250 - Loss: 0.0972 - Error: 0.0202
Epoch 10/250 - Loss: 0.0985 - Error: 0.0210
Epoch 11/250 - Loss: 0.0892 - Error: 0.0188
Epoch 12/250 - Loss: 0.0855 - Error: 0.0194
Epoch 13/250 - Loss: 0.1082 - Error: 0.0254
Epoch 14/250 - Loss: 0.0929 - Error: 0.0210
Epoch 15/250 - Loss: 0.1037 - Error: 0.0216
Epoch 16/250 - Loss: 0.1060 - Error: 0.0220
Epoch 17/250 - Loss: 0.0811 - Error: 0.0178
Epoch 18/250 - Loss: 0.1030 - Error: 0.0194
Epoch 19/250 - Loss: 0.1146 - Error: 0.0202
Epoch 20/250 - Loss: 0.1180 - Error: 0.0190
Epoch 21/250 - Loss: 0.1154 - Error: 0.0182
Epoch 22/250 - Loss: 0.1149 - Error: 0.0168
Epoch 23/250 - Loss: 0.1369 - Error: 0.0226
Epoch 24/250 - Loss: 0.1372 - Error: 0.0194
Epoch 25/250 - Loss: 0.1223 - Error: 0.0182
Epoch 26/250 - Loss: 0.1124 - Error: 0.0172
Epoch 27/250 - Loss: 0.1212 - Error: 0.0190
Epoch 28/250 - Loss: 0.1398 - Error: 0.0180
Epoch 29/250 - Loss: 0.1195 - Error: 0.0178
Epoch 30/250 - Loss: 0.1180 - Error: 0.0170
Epoch 31/250 - Loss: 0.1154 - Error: 0.0162
Epoch 32/250 - Loss: 0.1375 - Error: 0.0196
Epoch 33/250 - Loss: 0.1263 - Error: 0.0176
Epoch 34/250 - Loss: 0.1132 - Error: 0.0158
Epoch 35/250 - Loss: 0.1297 - Error: 0.0178
Epoch 36/250 - Loss: 0.1437 - Error: 0.0172
Epoch 37/250 - Loss: 0.1226 - Error: 0.0164
Epoch 38/250 - Loss: 0.1165 - Error: 0.0154
Epoch 39/250 - Loss: 0.1341 - Error: 0.0178
Epoch 40/250 - Loss: 0.1414 - Error: 0.0172
Epoch 41/250 - Loss: 0.1182 - Error: 0.0148
Epoch 42/250 - Loss: 0.1548 - Error: 0.0220
Epoch 43/250 - Loss: 0.1176 - Error: 0.0166
Epoch 44/250 - Loss: 0.1204 - Error: 0.0162
Epoch 45/250 - Loss: 0.1636 - Error: 0.0214
Epoch 46/250 - Loss: 0.1391 - Error: 0.0158
Epoch 47/250 - Loss: 0.1394 - Error: 0.0152
Epoch 48/250 - Loss: 0.1361 - Error: 0.0186
Epoch 49/250 - Loss: 0.1138 - Error: 0.0130
Epoch 50/250 - Loss: 0.1220 - Error: 0.0148
Epoch 51/250 - Loss: 0.1517 - Error: 0.0176
Epoch 52/250 - Loss: 0.1276 - Error: 0.0164
Epoch 53/250 - Loss: 0.1376 - Error: 0.0150
```

```
    Epoch 54/250 - Loss: 0.1229 - Error: 0.0150
    Epoch 55/250 - Loss: 0.1454 - Error: 0.0158
    Epoch 56/250 - Loss: 0.1246 - Error: 0.0144
    Epoch 57/250 - Loss: 0.1573 - Error: 0.0184
    Epoch 58/250 - Loss: 0.1406 - Error: 0.0162
    Epoch 59/250 - Loss: 0.1276 - Error: 0.0150
    Epoch 60/250 - Loss: 0.1234 - Error: 0.0144
```

```
# Create the loss plot
loss_plot = go.Scatter(x=list(range(1, epochs+1)), y=loss_values, mode='lines', nam
layout_loss = go.Layout(title='Cross-Entropy Loss', xaxis=dict(title='Epoch'), yaxi
fig_loss = go.Figure(data=[loss_plot], layout=layout_loss)
fig_loss.show()

# Create the classification error plot
error_plot = go.Scatter(x=list(range(1, epochs+1)), y=error_values, mode='lines', r
layout_error = go.Layout(title='Validation Classification Error', xaxis=dict(title=
fig_error = go.Figure(data=[error_plot], layout=layout_error)
fig_error.show()
```




Validation Classification Error
Error
0.04
0.035
0.03
0.025
0.02
0.015
50
100
Ep