Assignment 5

Name: Gourav Kumar Shaw

Roll Number: 2020CSB010

Machine Learning Lab

Task 1

Download and install TensorFlow from https://www.tensorflow.org/install/install_sources or using command `sudo pip install tensorflow` alternatively the Keras library can be used.

Task 2

Download MNIST dataset (contains class labels for digits 0-9). using the command:

```
import tensorflow as tf
data = tf.contrib.learn.datasets.mnist.load_mnist()
```

or

```
from keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
import tensorflow as tf
mnist_data = tf.keras.datasets.mnist.load_data()
```

```
mnist_data
```

```
((array([[[0, 0, 0, ..., 0, 0, 0],
          [0, 0, 0, ..., 0, 0, 0],
          [0, 0, 0, ..., 0, 0, 0],
          ...,
          [0, 0, 0, ..., 0, 0, 0],
          [0, 0, 0, ..., 0, 0, 0],
          [0, 0, 0, ..., 0, 0, 0]],

         [[0, 0, 0, ..., 0, 0, 0],
          [0, 0, 0, ..., 0, 0, 0],
          [0, 0, 0, ..., 0, 0, 0],
          ...,
          [0, 0, 0, ..., 0, 0, 0],
          [0, 0, 0, ..., 0, 0, 0],
          [0, 0, 0, ..., 0, 0, 0]],

         [[0, 0, 0, ..., 0, 0, 0],
          [0, 0, 0, ..., 0, 0, 0],
          [0, 0, 0, ..., 0, 0, 0],
          ...,
          [0, 0, 0, ..., 0, 0, 0],
          [0, 0, 0, ..., 0, 0, 0],
          [0, 0, 0, ..., 0, 0, 0]],

         ...,

         [[0, 0, 0, ..., 0, 0, 0],
          [0, 0, 0, ..., 0, 0, 0],
          [0, 0, 0, ..., 0, 0, 0],
          ...,
          [0, 0, 0, ..., 0, 0, 0],
          [0, 0, 0, ..., 0, 0, 0],
          [0, 0, 0, ..., 0, 0, 0]],

         [[0, 0, 0, ..., 0, 0, 0],
          [0, 0, 0, ..., 0, 0, 0],
          [0, 0, 0, ..., 0, 0, 0]
```

```
               [0, 0, 0, ..., 0, 0, 0],
               ...,
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0]],

              [[0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               ...,
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0]]], dtype=uint8),
       array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)),
      (array([[[0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               ...,
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0],
               [0, 0, 0, ..., 0, 0, 0]],
```

mnist_data is a Tuple of NumPy arrays: (x_train, y_train), (x_test, y_test).

x_train: uint8 NumPy array of grayscale image data with shapes (60000, 28, 28), containing the training data. Pixel values range from 0 to 255.

y_train: uint8 NumPy array of digit labels (integers in range 0-9) with shape (60000,) for the training data.

x_test: uint8 NumPy array of grayscale image data with shapes (10000, 28, 28), containing the test data. Pixel values range from 0 to 255.

y_test: uint8 NumPy array of digit labels (integers in range 0-9) with shape (10000,) for the test data.

```
import numpy as np

(x_train, y_train), (x_test, y_test) = mnist_data
```

```
# mapping 0-255 to 0-1
x_train = np.array([img/255 for img in x_train])
x_test = np.array([img/255 for img in x_test])

assert x_train.shape == (60000, 28, 28)
assert x_test.shape == (10000, 28, 28)
assert y_train.shape == (60000,)
assert y_test.shape == (10000,)
```

Task 3

Reduce the training size by 1/10 if computation resources are limited.

Define radial basis function (RBF) as

```
 def RBF(x, c, s):
     return np.exp(-np.sum((x-c)**2, axis=1)/(2*s**2))
```

where, x is the actual value, c is centre (assumed as mean) and s is the standard deviation.

Converted 28*28 image into 32*32 using rbf and store the new dataset with the labels. Split the dataset as 80% training and 10% validation and 10% test.

```
import numpy as np


def RBF(x, c, s):
    return np.exp(-np.sum((x-c)**2, axis=1)/(2*s**2))

# TODO: used simple scaling to upscale the image,
# use rbf to do this in future
# from tensorflow.image import resize

# reshape to convert 28x28 image (assumed greyscale)
# to 28x28x1 (1 denoting only one value per pixel
# [rgb will have three numbers for eg])

# x_train = np.reshape(x_train, (-1, 28, 28, 1))
# x_train = np.array([resize(img, [32, 32]) for img in x_train])
# print(f"x_train shape: {x_train.shape}")

# x_test = np.reshape(x_test, (-1, 28, 28, 1))
# x_test = np.array([resize(img, [32, 32]) for img in x_test])
# print(f"x_test shape: {x_test.shape}")
```

```
import pandas as pd
# convert y to categorical
y_train = pd.get_dummies(y_train).to_numpy()
y_test = pd.get_dummies(y_test).to_numpy()
```

```
y_train[0:9]
```

```
array([[0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
       [1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 1, 0, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 1],
       [0, 0, 1, 0, 0, 0, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 1, 0, 0, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0, 0]], dtype=uint8)
```

```
input_shape = x_train[0].shape
num_classes = len(y_train[0])
```

Task 4

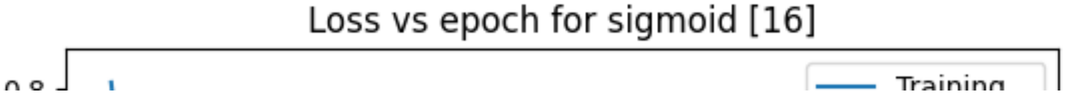
Now run the fully connected network after flattening the data by changing the number the hyper-parameters use adam optimizer(learning rate = 0.001) and categorical cross-entropy loss

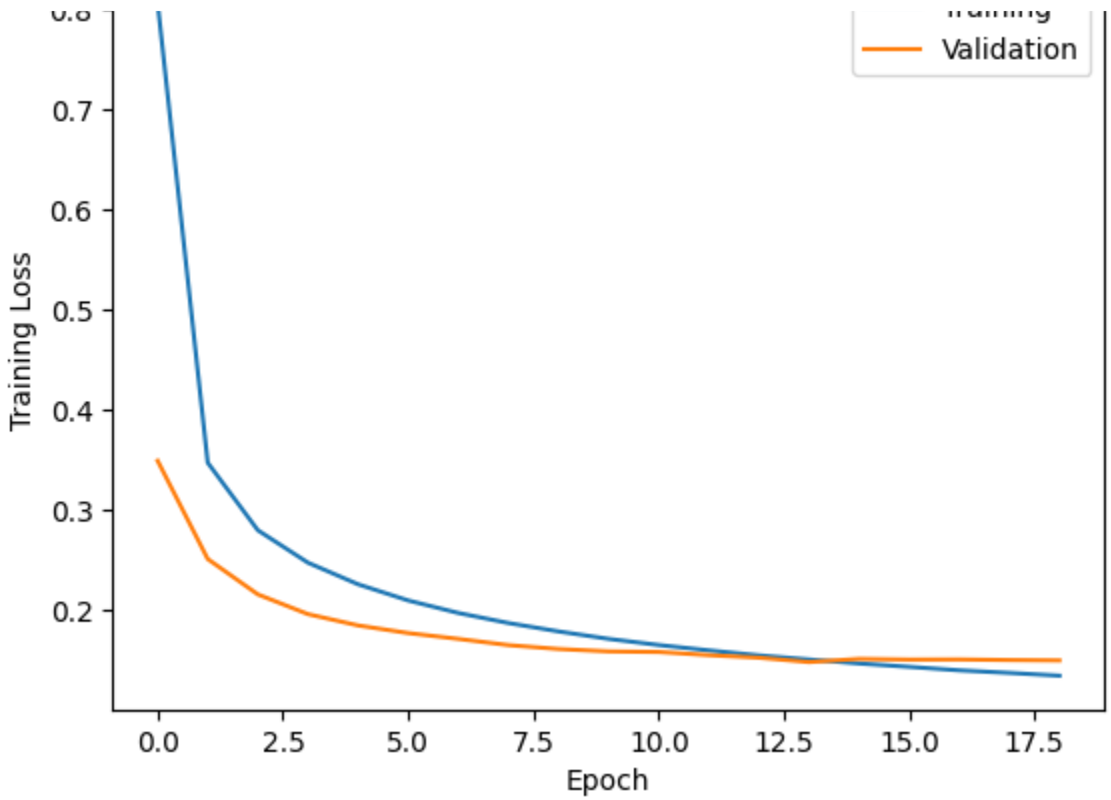
Hidden Layers	Activation Function	Hidden Neurons
1	Sigmoid	[16]
2	Sigmoid	[16,32]
3	Sigmoid	[16,32,64]

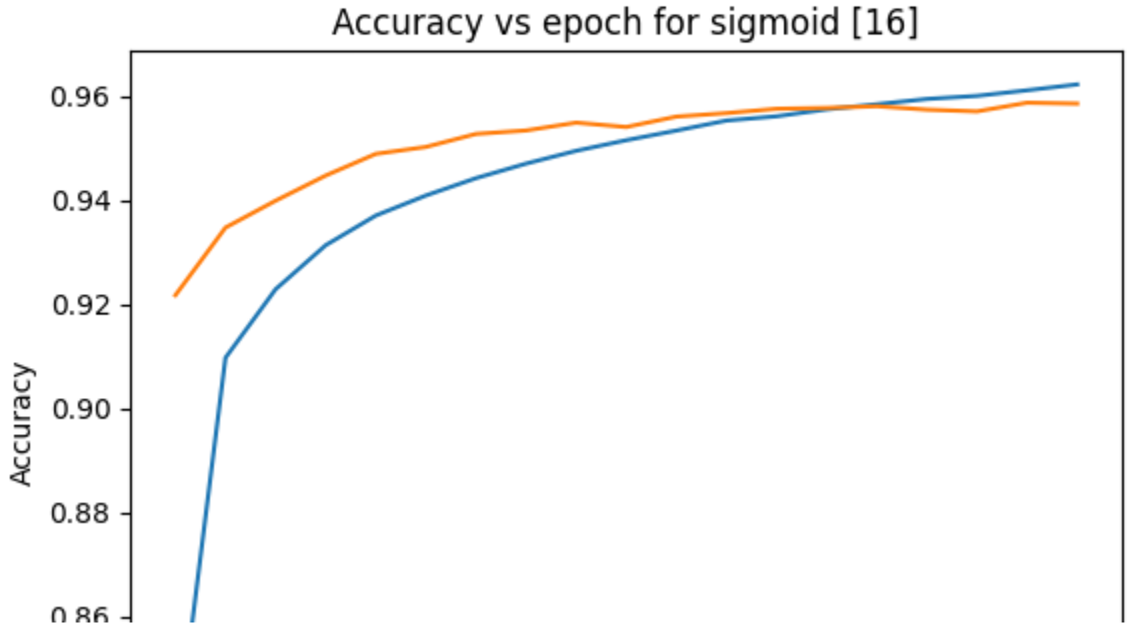
```
from tensorflow.keras import Sequential, Input
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.losses import CategoricalCrossentropy
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam
import matplotlib.pyplot as plt


def train_model(
        activation_function: 'str',
        hidden_neurons: 'list[int]',
        dropout_rate: 'float | None' = None,
        adam_learn_rate=0.001,
        verbose=True):

    model = Sequential()
    model.add(Input(shape=(input_shape)))
    model.add(Flatten())

    for unit in hidden_neurons[::-1]:
        model.add(Dense(unit, activation=activation_function))
        if dropout_rate is not None:
            model.add(Dropout(rate=dropout_rate))

    # softmax as it gives probabilistic value
    # softmax as it gives probabilistic value
    # (sum of all the last nodes will be 1)
    model.add(Dense(num_classes, activation='softmax'))

    if verbose:
        model.summary()

    model.compile(optimizer=Adam(learning_rate=adam_learn_rate),
                  loss=CategoricalCrossentropy(),
                  metrics=['accuracy'])

    history = model.fit(x=x_train,
                        y=y_train,
                        validation_split=0.1,
                        epochs=100,
                        callbacks=[
                            EarlyStopping(
                                monitor='val_loss',
                                patience=5,
                                restore_best_weights= True
                            )
                        ],
                        verbose='auto' if verbose else 0
                        )

    return model, history


def plot_history(
        history: "tf.keras.callbacks.History",
        activation_function: 'str',
        hidden_neurons: 'list[int]',
        dropout_rate: 'float | None' = None):

    plt.plot(history.history['loss'], label='Training')
    plt.plot(history.history['val_loss'], label='Validation')
    plt.ylabel('Training Loss')
    plt.xlabel('Epoch')
    plt.legend()
    plt.legend()

    if dropout_rate is None:
        plt.title(
            f'Loss vs epoch for {activation_function} {hidden_neurons}')
    else:
        plt.title(
            f'Loss vs epoch for {activation_function} {hidden_neurons} dropout {dropout_rate}')

    plt.show()

    plt.plot(history.history['accuracy'], label='Training')
    plt.plot(history.history['val_accuracy'], label='Validation')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')

    if dropout_rate is None:
        plt.title(
            f'Accuracy vs epoch for {activation_function} {hidden_neurons}')
    else:
        plt.title(
            f'Accuracy vs epoch for {activation_function} {hidden_neurons} dropout {dropout_rate}')

    plt.legend()
    plt.show()



result = pd.DataFrame(
    columns=[
        'Hidden Layers',
        'Activation Function',
        'Hidden Neurons',
        'Test Loss',
        'Test Acccuracy'],
)
```

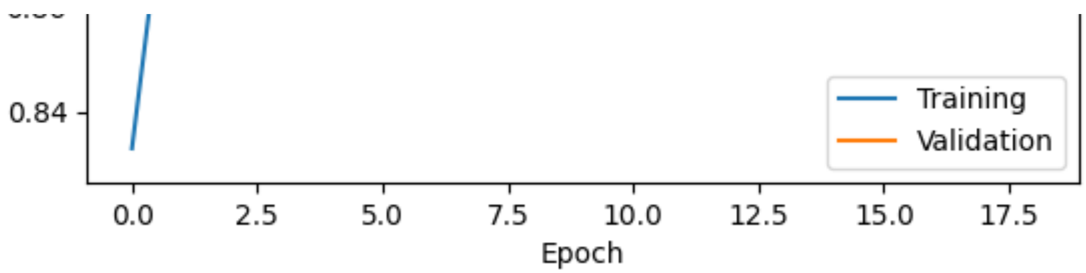
```
hidden_neurons = [16]
activation_function = 'sigmoid'

model, history = train_model(activation_function, hidden_neurons)
test_loss, test_acc = model.evaluate(x_test, y_test)

print(f"test_loss = {test_loss} test_acc = {test_acc}")

result.loc[len(result.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    test_loss,
    test_acc]

plot_history(history, activation_function, hidden_neurons)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 16)                12560

 dense_1 (Dense)             (None, 10)                170

=================================================================
Total params: 12,730
Trainable params: 12,730
Non-trainable params: 0
_________________________________________________________________
2022-10-12 16:13:31.184527: E tensorflow/stream_executor/cuda/cuda_driver.cc:271] failed call to cuInit: UNKNOWN ERR
2022-10-12 16:13:31.184671: I tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not appear
2022-10-12 16:13:31.186869: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimized
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Epoch 1/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.8069 - accuracy: 0.8329 - val_loss: 0.3494 - val_
Epoch 2/100
Epoch 2/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.3475 - accuracy: 0.9096 - val_loss: 0.2512 - val_
Epoch 3/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.2800 - accuracy: 0.9228 - val_loss: 0.2159 - val_
Epoch 4/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.2476 - accuracy: 0.9313 - val_loss: 0.1961 - val_
Epoch 5/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.2260 - accuracy: 0.9370 - val_loss: 0.1849 - val_
Epoch 6/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.2098 - accuracy: 0.9408 - val_loss: 0.1772 - val_
Epoch 7/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1973 - accuracy: 0.9442 - val_loss: 0.1714 - val_
Epoch 8/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1873 - accuracy: 0.9470 - val_loss: 0.1652 - val_
Epoch 9/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1789 - accuracy: 0.9494 - val_loss: 0.1613 - val_
Epoch 10/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1714 - accuracy: 0.9515 - val_loss: 0.1589 - val_
Epoch 11/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1653 - accuracy: 0.9533 - val_loss: 0.1583 - val_
Epoch 12/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1599 - accuracy: 0.9552 - val_loss: 0.1552 - val_
Epoch 13/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1549 - accuracy: 0.9561 - val_loss: 0.1526 - val_
Epoch 14/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1508 - accuracy: 0.9574 - val_loss: 0.1483 - val_
Epoch 15/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1468 - accuracy: 0.9584 - val_loss: 0.1514 - val_
Epoch 16/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1435 - accuracy: 0.9594 - val_loss: 0.1507 - val_
Epoch 17/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1400 - accuracy: 0.9600 - val_loss: 0.1509 - val_
Epoch 18/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1374 - accuracy: 0.9611 - val_loss: 0.1503 - val_
Epoch 19/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1346 - accuracy: 0.9622 - val_loss: 0.1499 - val_
313/313 [==============================] - 1s 2ms/step - loss: 0.1751 - accuracy: 0.9491
test_loss = 0.17512023448944092 test_acc = 0.9491000175476074
```






Accuracy vs epoch for sigmoid [16]

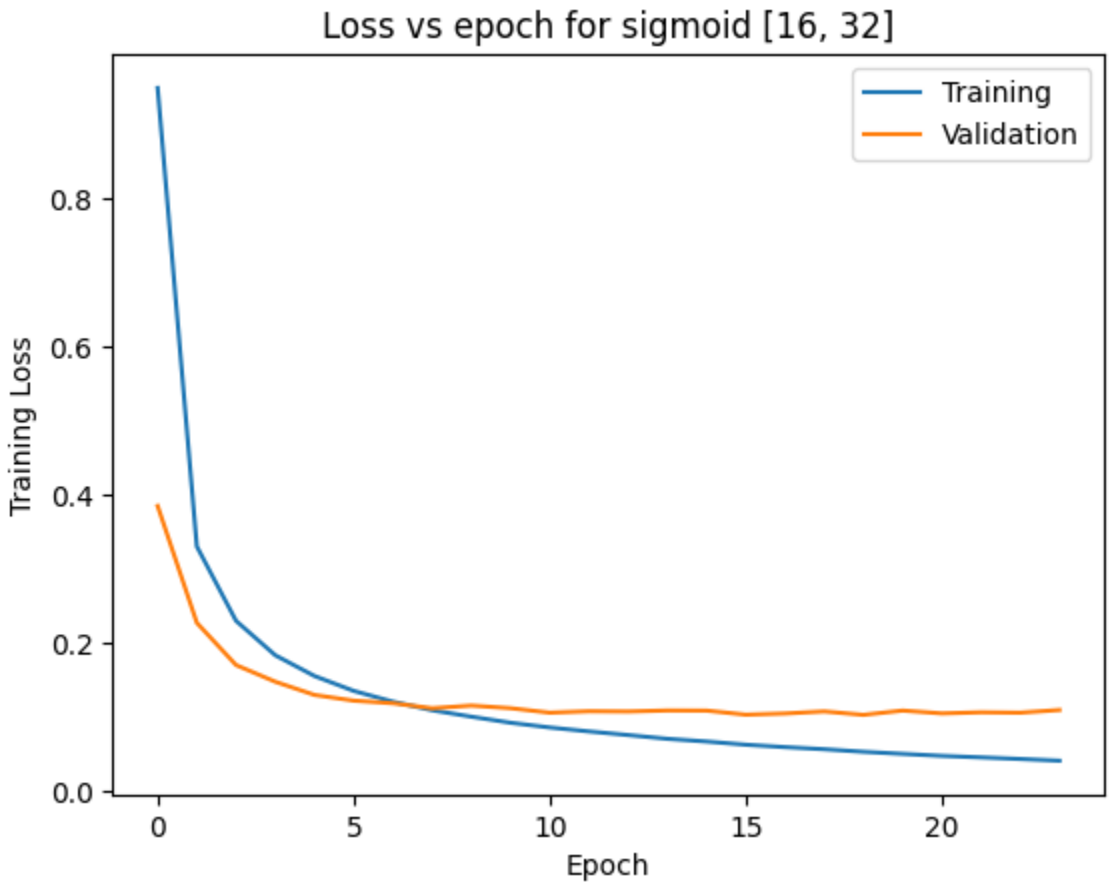


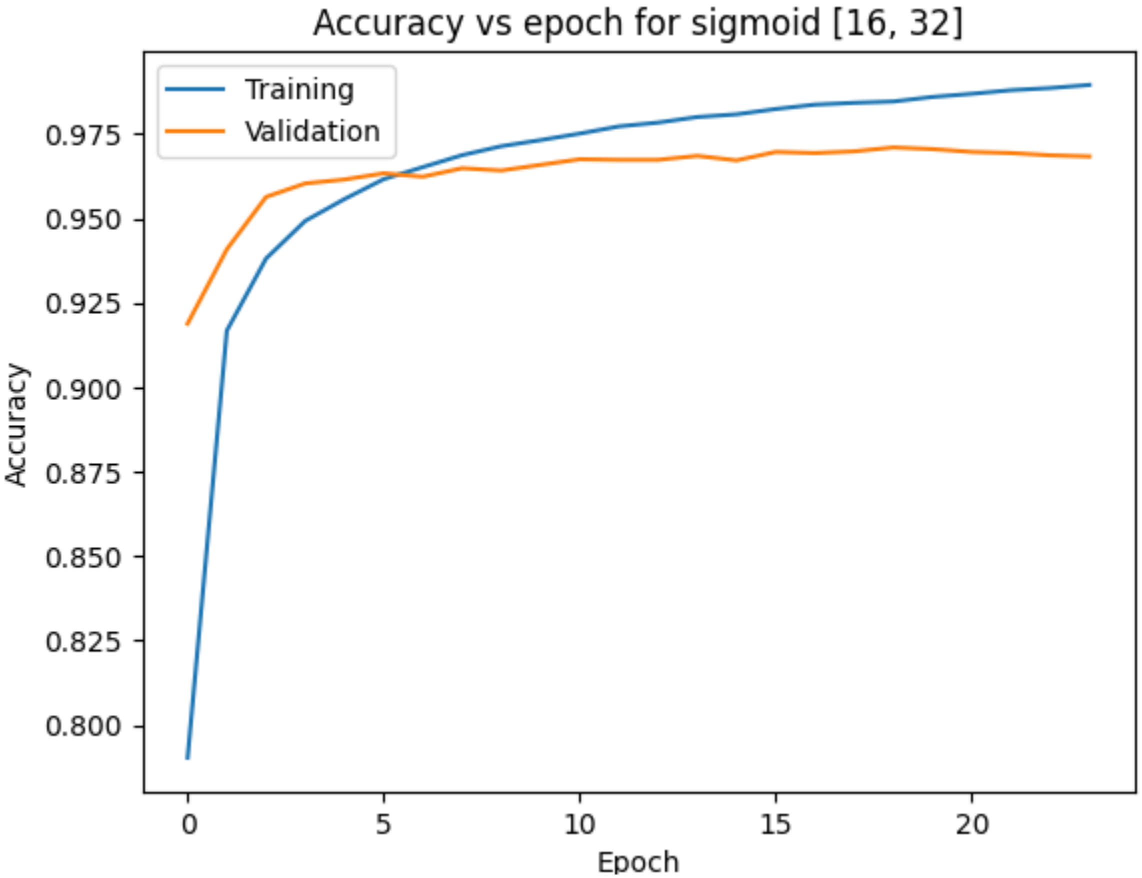
```
activation_function = 'sigmoid'
hidden_neurons = [16, 32]

model, history = train_model(activation_function, hidden_neurons)
test_loss, test_acc = model.evaluate(x_test, y_test)

print(f"test_loss = {test_loss} test_acc = {test_acc}")

result.loc[len(result.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    test_loss,
    test_acc]

plot_history(history, activation_function, hidden_neurons)
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_1 (Flatten)         (None, 784)               0

 dense_2 (Dense)             (None, 32)                25120

 dense_3 (Dense)             (None, 16)                528

 dense_4 (Dense)             (None, 10)                170
```

```
=================================================================
Total params: 25,818
Trainable params: 25,818
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.9483 - accuracy: 0.7903 - val_loss: 0.3847 - val_
Epoch 2/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.3301 - accuracy: 0.9167 - val_loss: 0.2271 - val_
Epoch 3/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.2298 - accuracy: 0.9381 - val_loss: 0.1701 - val_
Epoch 4/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1834 - accuracy: 0.9492 - val_loss: 0.1477 - val_
Epoch 5/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1554 - accuracy: 0.9557 - val_loss: 0.1299 - val_
Epoch 6/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1353 - accuracy: 0.9616 - val_loss: 0.1221 - val_
Epoch 7/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1208 - accuracy: 0.9652 - val_loss: 0.1187 - val_
Epoch 8/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1092 - accuracy: 0.9687 - val_loss: 0.1119 - val_
Epoch 9/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1005 - accuracy: 0.9714 - val_loss: 0.1155 - val_
Epoch 10/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.0924 - accuracy: 0.9732 - val_loss: 0.1120 - val_
Epoch 11/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.0860 - accuracy: 0.9751 - val_loss: 0.1059 - val_
Epoch 12/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.0807 - accuracy: 0.9772 - val_loss: 0.1079 - val_
Epoch 13/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.0758 - accuracy: 0.9784 - val_loss: 0.1076 - val_
Epoch 14/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0707 - accuracy: 0.9800 - val_loss: 0.1089 - val_
Epoch 15/100
1688/1688 [==============================] - 7s 4ms/step - loss: 0.0671 - accuracy: 0.9808 - val_loss: 0.1087 - val_
Epoch 16/100
1688/1688 [==============================] - 7s 4ms/step - loss: 0.0627 - accuracy: 0.9824 - val_loss: 0.1032 - val_
Epoch 17/100
1688/1688 [==============================] - 7s 4ms/step - loss: 0.0595 - accuracy: 0.9837 - val_loss: 0.1048 - val_
Epoch 18/100
1688/1688 [==============================] - 7s 4ms/step - loss: 0.0567 - accuracy: 0.9842 - val_loss: 0.1076 - val_
```

```
1688/1688 [==============================] - 7s 4ms/step - loss: 0.0567 - accuracy: 0.9842 - val_loss: 0.1076 - val_
Epoch 19/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.0532 - accuracy: 0.9846 - val_loss: 0.1030 - val_
Epoch 20/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0504 - accuracy: 0.9859 - val_loss: 0.1088 - val_
Epoch 21/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0477 - accuracy: 0.9869 - val_loss: 0.1050 - val_
Epoch 22/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.0457 - accuracy: 0.9879 - val_loss: 0.1064 - val_
Epoch 23/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0435 - accuracy: 0.9886 - val_loss: 0.1059 - val_
Epoch 24/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.0412 - accuracy: 0.9895 - val_loss: 0.1096 - val_
313/313 [==============================] - 1s 2ms/step - loss: 0.1178 - accuracy: 0.9660
test_loss = 0.11777360737323761 test_acc = 0.9660000205039978
```


Loss vs epoch for sigmoid [16, 32]

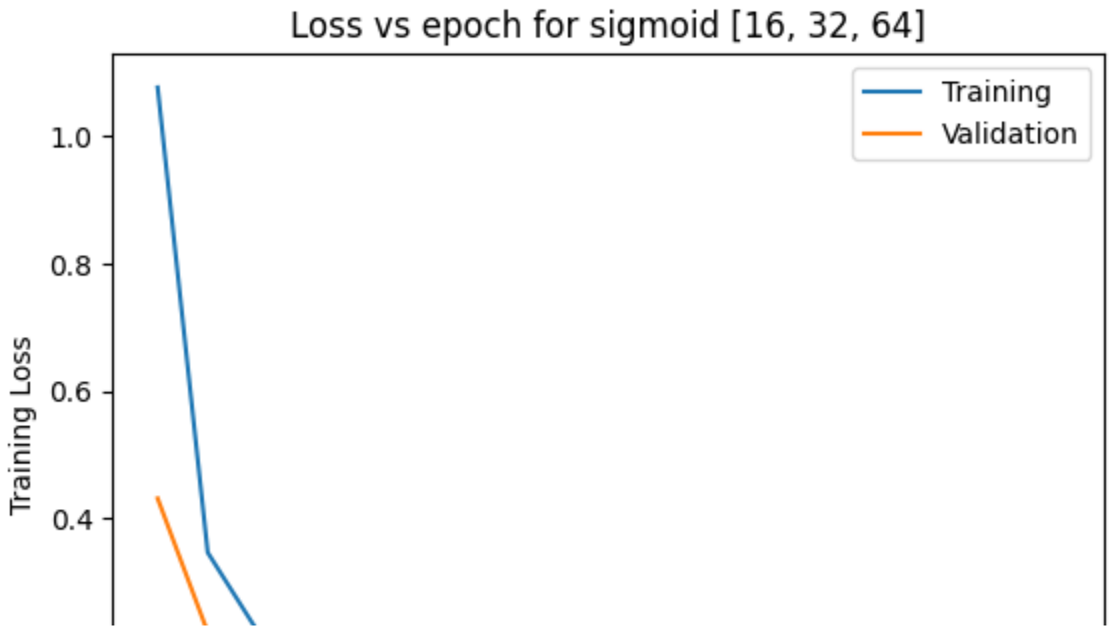


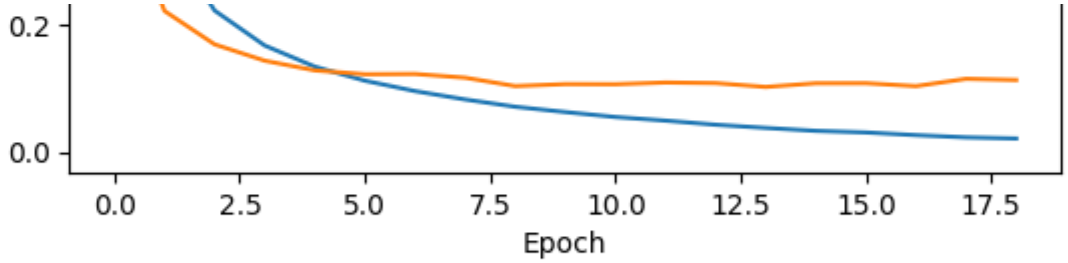
```
hidden_neurons = [16, 32, 64]
activation_function = 'sigmoid'

model, history = train_model(activation_function, hidden_neurons)
test_loss, test_acc = model.evaluate(x_test, y_test)

print(f"test_loss = {test_loss} test_acc = {test_acc}")

result.loc[len(result.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    test_loss,
    test_acc]
```

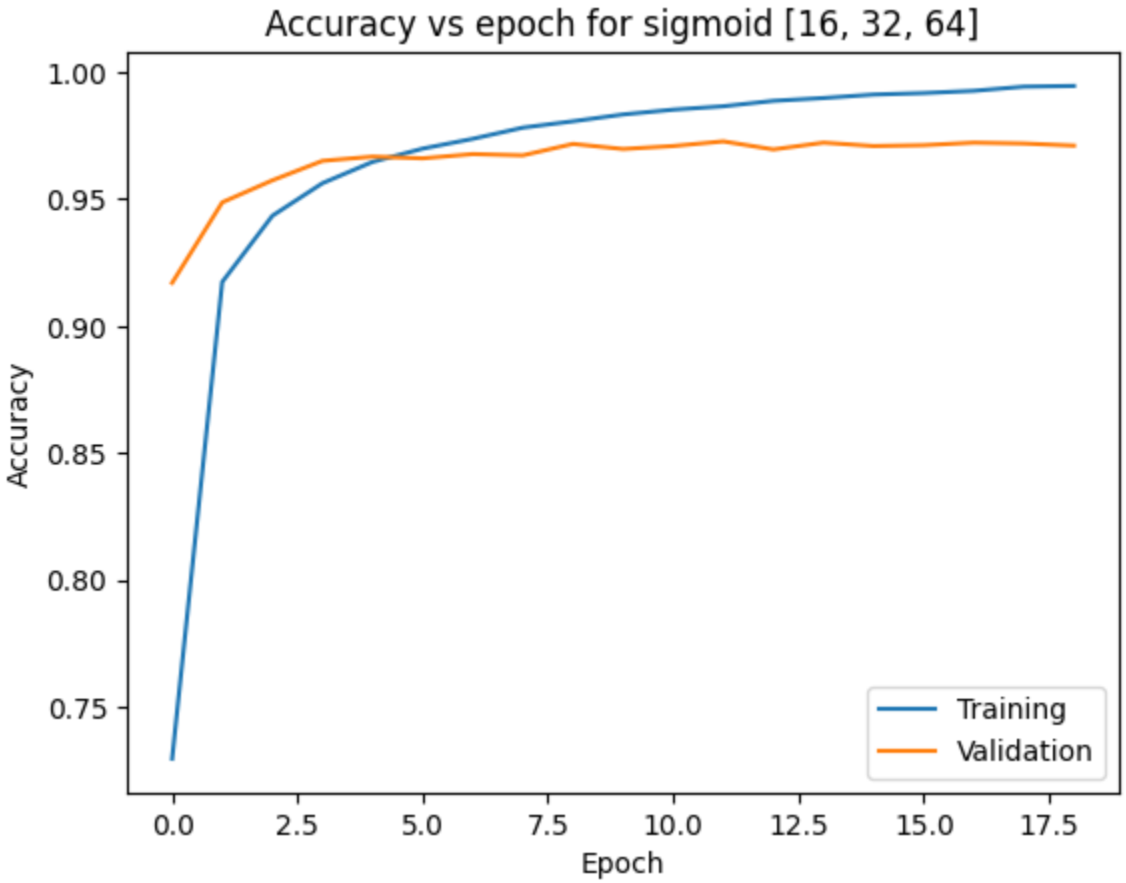
```
plot_history(history, activation_function, hidden_neurons)
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_2 (Flatten)         (None, 784)               0

 dense_5 (Dense)             (None, 64)                50240

 dense_6 (Dense)             (None, 32)                2080

 dense_7 (Dense)             (None, 16)                528

 dense_8 (Dense)             (None, 10)                170

=================================================================
Total params: 53,018
Trainable params: 53,018
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 7s 4ms/step - loss: 1.0763 - accuracy: 0.7298 - val_loss: 0.4315 - val_
Epoch 2/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.3465 - accuracy: 0.9174 - val_loss: 0.2216 - val_
Epoch 3/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.2222 - accuracy: 0.9434 - val_loss: 0.1691 - val_
Epoch 4/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.1674 - accuracy: 0.9562 - val_loss: 0.1435 - val_
Epoch 5/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.1341 - accuracy: 0.9646 - val_loss: 0.1283 - val_
Epoch 6/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.1121 - accuracy: 0.9698 - val_loss: 0.1222 - val_
Epoch 7/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0958 - accuracy: 0.9737 - val_loss: 0.1227 - val_
Epoch 8/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0827 - accuracy: 0.9781 - val_loss: 0.1168 - val_
Epoch 9/100
```

```
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0713 - accuracy: 0.9806 - val_loss: 0.1037 - val_
Epoch 10/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0629 - accuracy: 0.9833 - val_loss: 0.1068 - val_
Epoch 11/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0552 - accuracy: 0.9851 - val_loss: 0.1066 - val_
Epoch 12/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0495 - accuracy: 0.9865 - val_loss: 0.1093 - val_
Epoch 13/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0431 - accuracy: 0.9886 - val_loss: 0.1084 - val_
Epoch 14/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0381 - accuracy: 0.9897 - val_loss: 0.1024 - val_
Epoch 15/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0335 - accuracy: 0.9911 - val_loss: 0.1081 - val_
Epoch 16/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0310 - accuracy: 0.9917 - val_loss: 0.1082 - val_
Epoch 17/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0267 - accuracy: 0.9926 - val_loss: 0.1036 - val_
Epoch 18/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0232 - accuracy: 0.9942 - val_loss: 0.1151 - val_
Epoch 19/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0215 - accuracy: 0.9945 - val_loss: 0.1132 - val_
313/313 [==============================] - 1s 3ms/step - loss: 0.1262 - accuracy: 0.9677
test_loss = 0.12620392441749573 test_acc = 0.9677000045776367
```







```
result
```

	Hidden Layers	Activation Function	Hidden Neurons	Test Loss	Test Acccuracy
0	1	sigmoid	[16]	0.175120	0.9491

```
1    2    sigmoid        [16, 32]    0.117774    0.9660
2    3    sigmoid    [16, 32, 64]    0.126204    0.9677
```

Task 5

Now run the network by changing the number the Activation Function hyper-parameters:

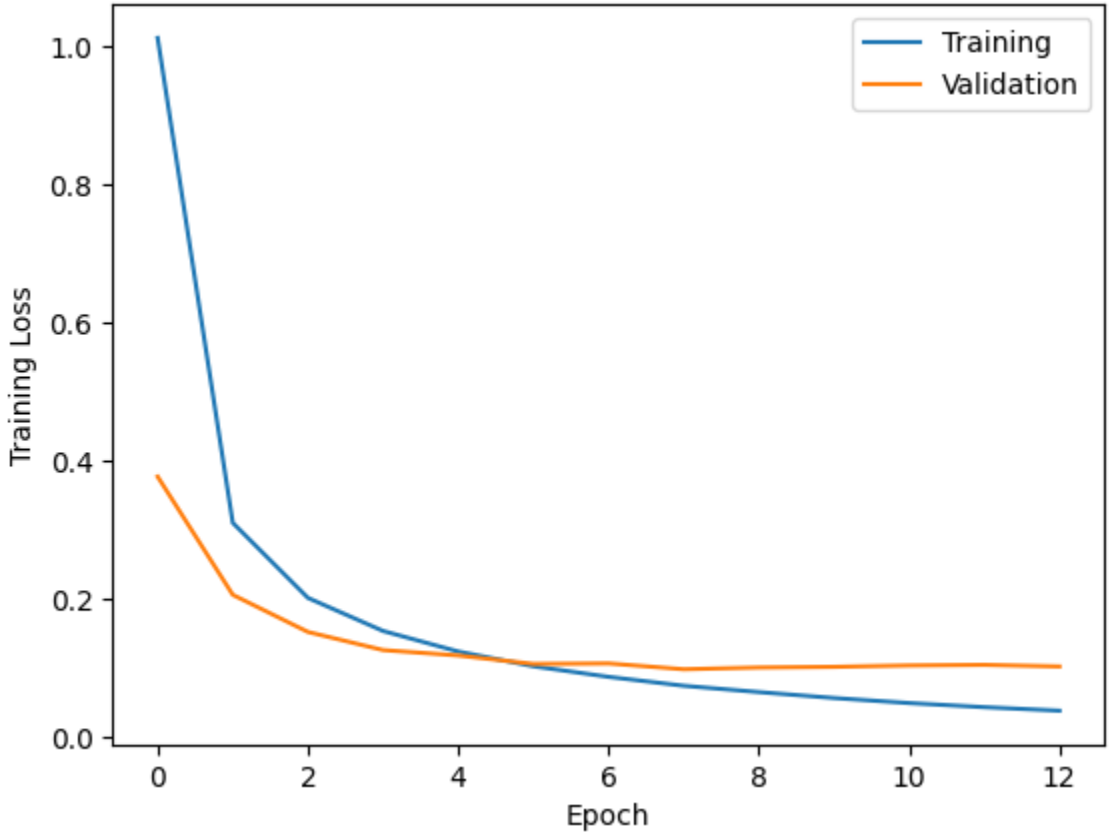
Hidden Layers	Activation Function	Hidden Neurons
3	Sigmoid	[16,32,64]
3	Tanh	[16,32,64]
3	Relu	[16,32,64]

```
result = pd.DataFrame(
    columns=[
        'Hidden Layers',
        'Activation Function',
        'Hidden Neurons',
        'Test Loss',
        'Test Acccuracy'],
)
```

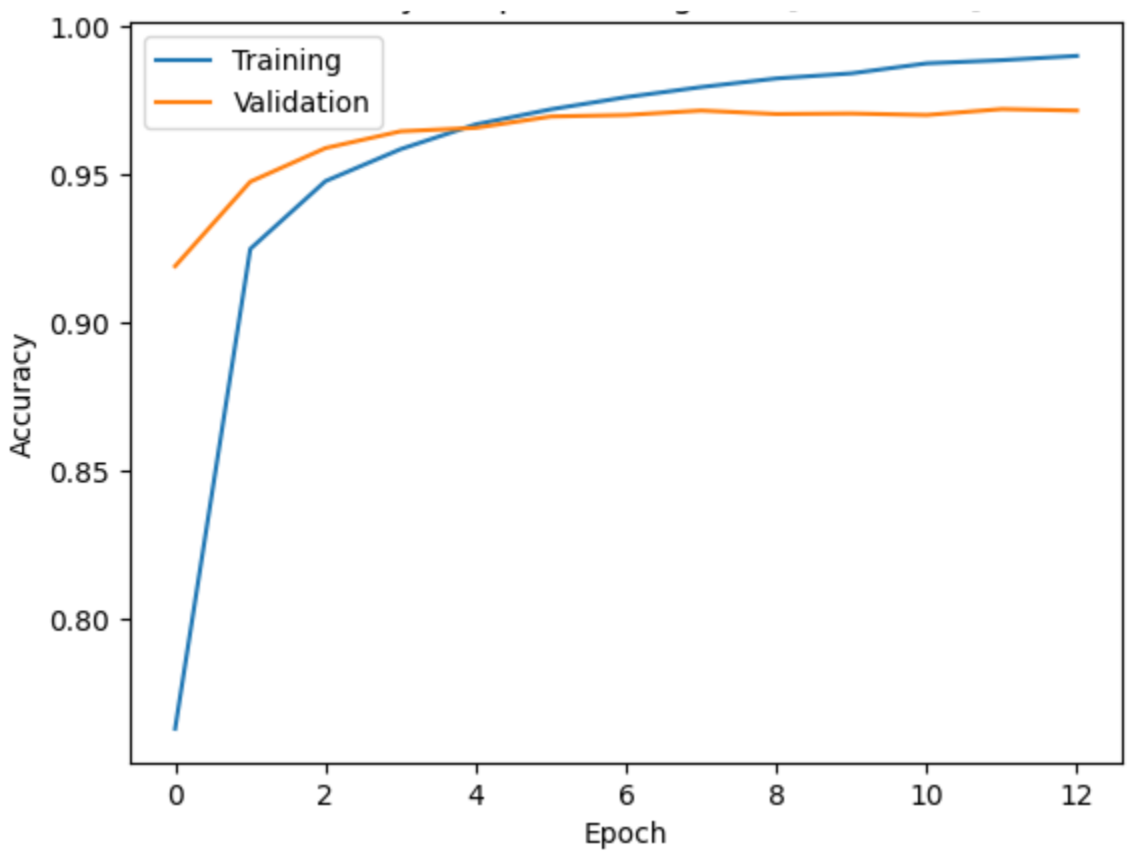
```
hidden_neurons = [16, 32, 64]
activation_function = 'sigmoid'

model, history = train_model(activation_function, hidden_neurons)
test_loss, test_acc = model.evaluate(x_test, y_test)

print(f"test_loss = {test_loss} test_acc = {test_acc}")

result.loc[len(result.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    test_loss,
    test_acc]

plot_history(history, activation_function, hidden_neurons)
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_3 (Flatten)         (None, 784)               0

 dense_9 (Dense)             (None, 64)                50240

 dense_10 (Dense)            (None, 32)                2080

 dense_11 (Dense)            (None, 16)                528

 dense_12 (Dense)            (None, 10)                170

=================================================================
Total params: 53,018
Trainable params: 53,018
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 7s 4ms/step - loss: 1.0124 - accuracy: 0.7628 - val_loss: 0.3773 - val_
Epoch 2/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.3104 - accuracy: 0.9247 - val_loss: 0.2062 - val_
Epoch 3/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.2017 - accuracy: 0.9475 - val_loss: 0.1524 - val_
Epoch 4/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.1538 - accuracy: 0.9583 - val_loss: 0.1262 - val_
Epoch 5/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.1241 - accuracy: 0.9667 - val_loss: 0.1184 - val_
Epoch 6/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.1027 - accuracy: 0.9718 - val_loss: 0.1061 - val_
Epoch 7/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0874 - accuracy: 0.9759 - val_loss: 0.1069 - val_
Epoch 8/100
Epoch 8/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0745 - accuracy: 0.9793 - val_loss: 0.0984 - val_
Epoch 9/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0654 - accuracy: 0.9822 - val_loss: 0.1009 - val_
Epoch 10/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0568 - accuracy: 0.9839 - val_loss: 0.1019 - val_
Epoch 11/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0495 - accuracy: 0.9873 - val_loss: 0.1039 - val_
Epoch 12/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0435 - accuracy: 0.9883 - val_loss: 0.1048 - val_
Epoch 13/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0385 - accuracy: 0.9898 - val_loss: 0.1024 - val_
313/313 [==============================] - 1s 2ms/step - loss: 0.1100 - accuracy: 0.9694
test_loss = 0.11003106832504272 test_acc = 0.9693999886512756
```

Loss vs epoch for sigmoid [16, 32, 64]



Accuracy vs epoch for sigmoid [16, 32, 64]

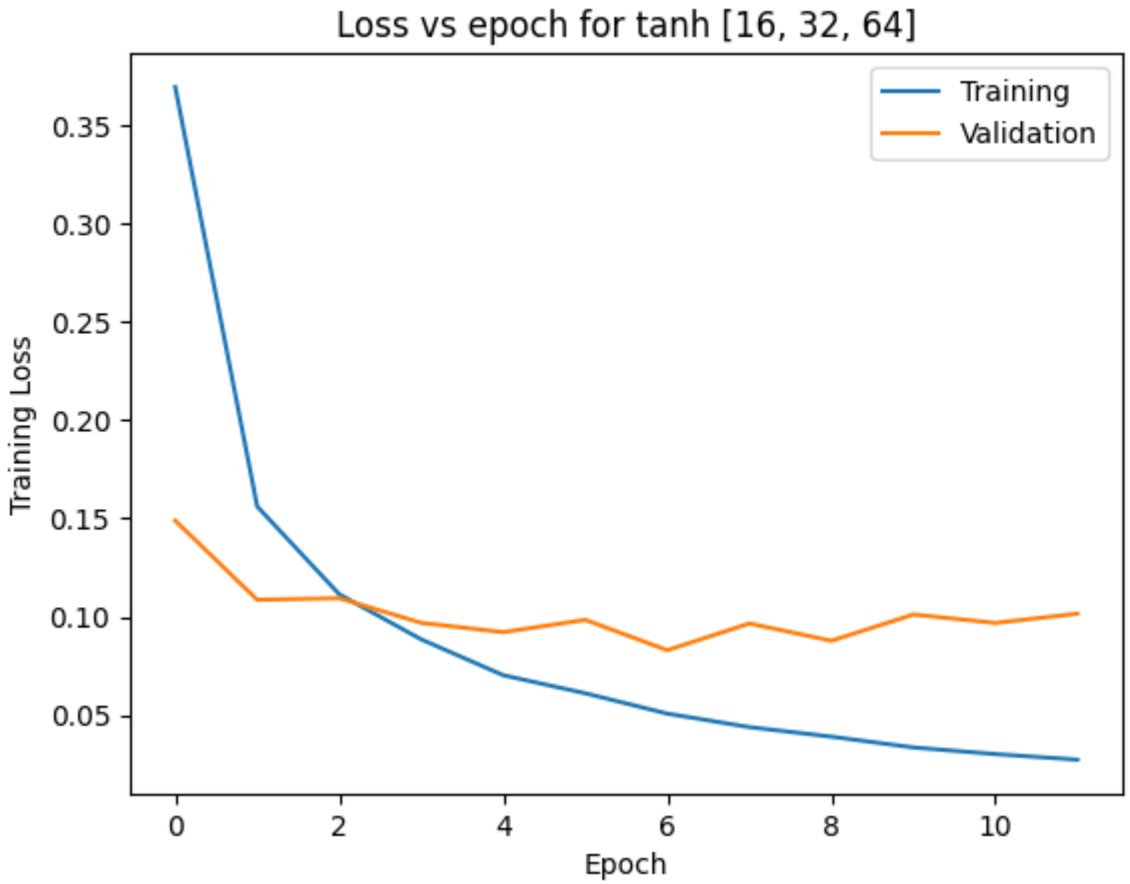


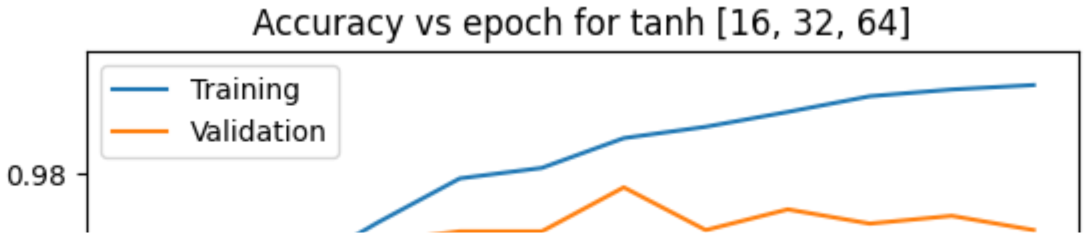
```
hidden_neurons = [16, 32, 64]
activation_function = 'tanh'

model, history = train_model(activation_function, hidden_neurons)
test_loss, test_acc = model.evaluate(x_test, y_test)

print(f"test_loss = {test_loss} test_acc = {test_acc}")

result.loc[len(result.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    test_loss,
    test_acc]
```

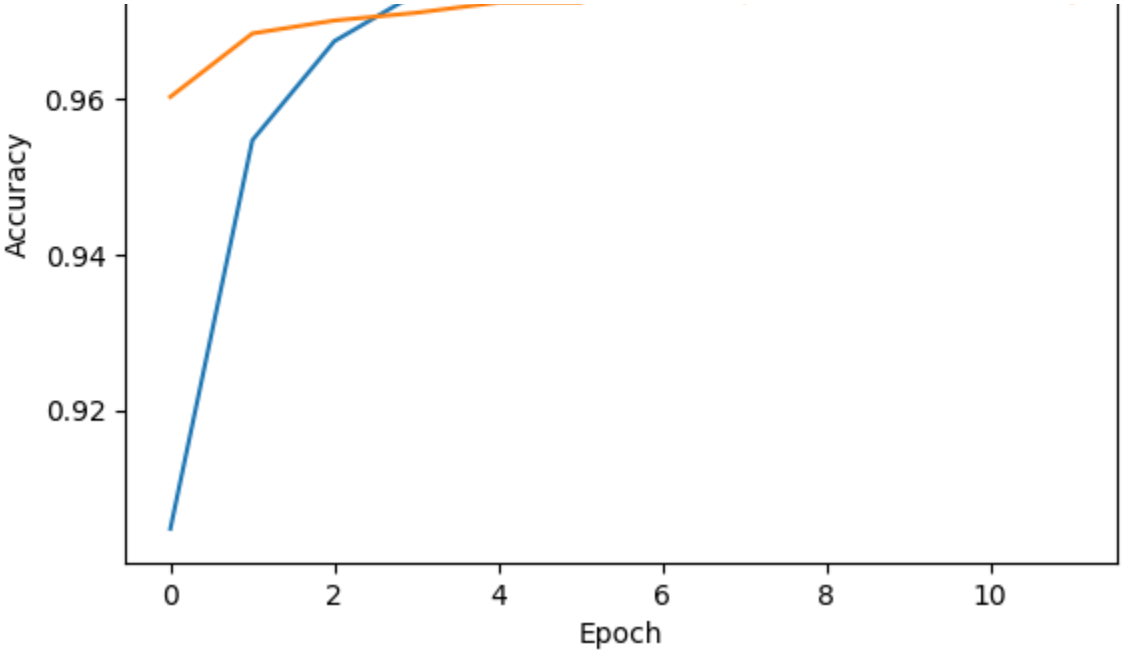
```
plot_history(history, activation_function, hidden_neurons)
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 flatten_4 (Flatten)         (None, 784)               0         
                                                                 
 dense_13 (Dense)            (None, 64)                50240     
                                                                 
 dense_14 (Dense)            (None, 32)                2080      
                                                                 
 dense_15 (Dense)            (None, 16)                528       
                                                                 
 dense_16 (Dense)            (None, 10)                170       
                                                                 
=================================================================
Total params: 53,018
Trainable params: 53,018
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 7s 4ms/step - loss: 0.3694 - accuracy: 0.9047 - val_loss: 0.1489 - val_
Epoch 2/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.1561 - accuracy: 0.9547 - val_loss: 0.1086 - val_
Epoch 3/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.1114 - accuracy: 0.9675 - val_loss: 0.1094 - val_
Epoch 4/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0885 - accuracy: 0.9736 - val_loss: 0.0969 - val_
Epoch 5/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0703 - accuracy: 0.9793 - val_loss: 0.0922 - val_
Epoch 6/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0611 - accuracy: 0.9807 - val_loss: 0.0984 - val_
Epoch 7/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0507 - accuracy: 0.9845 - val_loss: 0.0830 - val_
Epoch 8/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0439 - accuracy: 0.9860 - val_loss: 0.0966 - val_
Epoch 9/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0390 - accuracy: 0.9879 - val_loss: 0.0878 - val_
```

```
Epoch 10/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0336 - accuracy: 0.9899 - val_loss: 0.1012 - val_
Epoch 11/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0302 - accuracy: 0.9908 - val_loss: 0.0969 - val_
Epoch 12/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0273 - accuracy: 0.9913 - val_loss: 0.1016 - val_
313/313 [==============================] - 1s 2ms/step - loss: 0.0955 - accuracy: 0.9726
test_loss = 0.09548316895961761 test_acc = 0.972599983215332
```





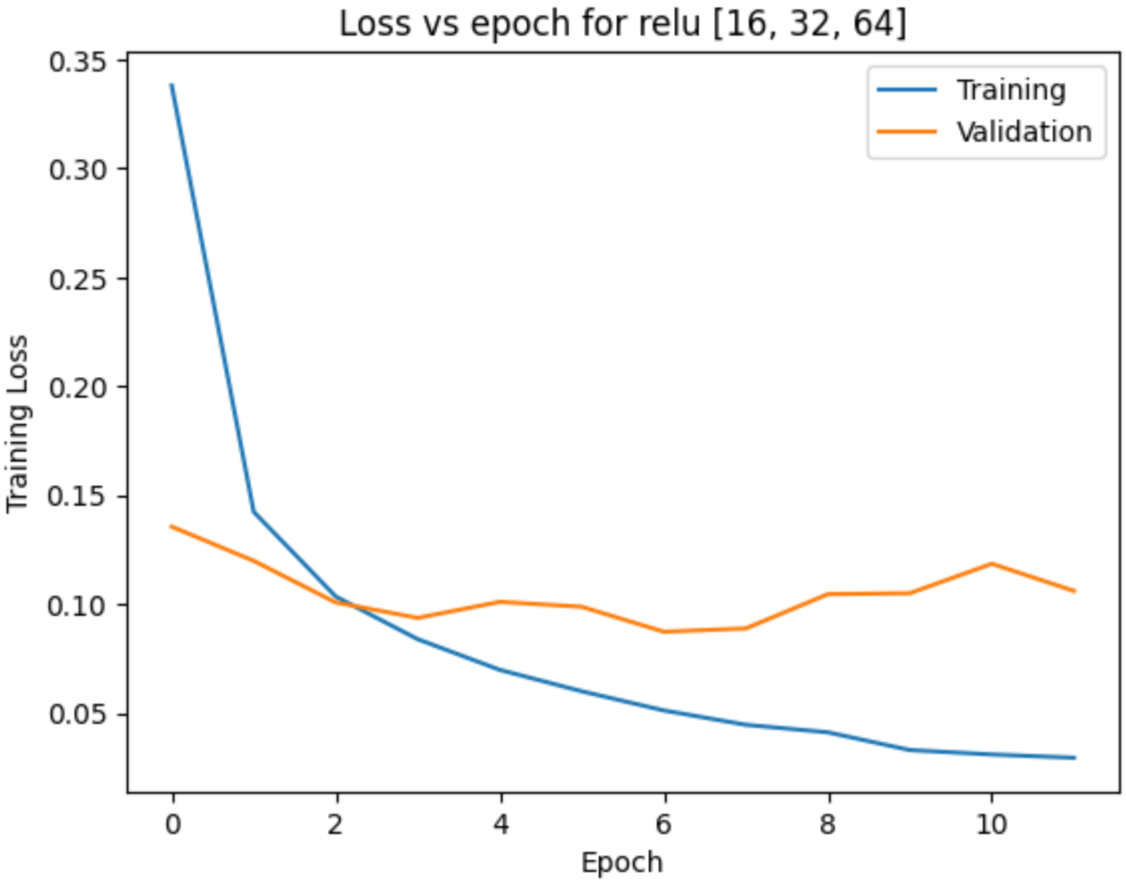


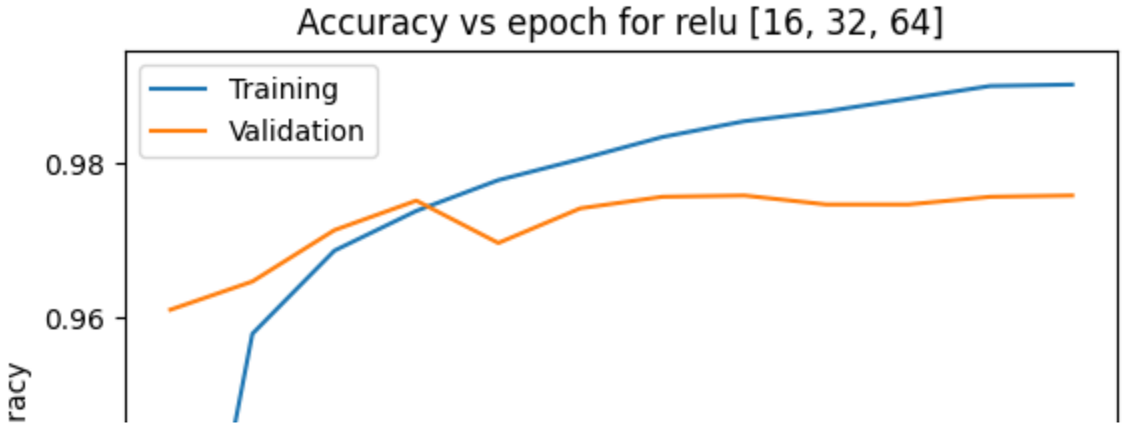
```
hidden_neurons = [16, 32, 64]
activation_function = 'relu'

model, history = train_model(activation_function, hidden_neurons)
test_loss, test_acc = model.evaluate(x_test, y_test)

print(f"test_loss = {test_loss} test_acc = {test_acc}")

result.loc[len(result.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    test_loss,
    test_acc]

plot_history(history, activation_function, hidden_neurons)
```

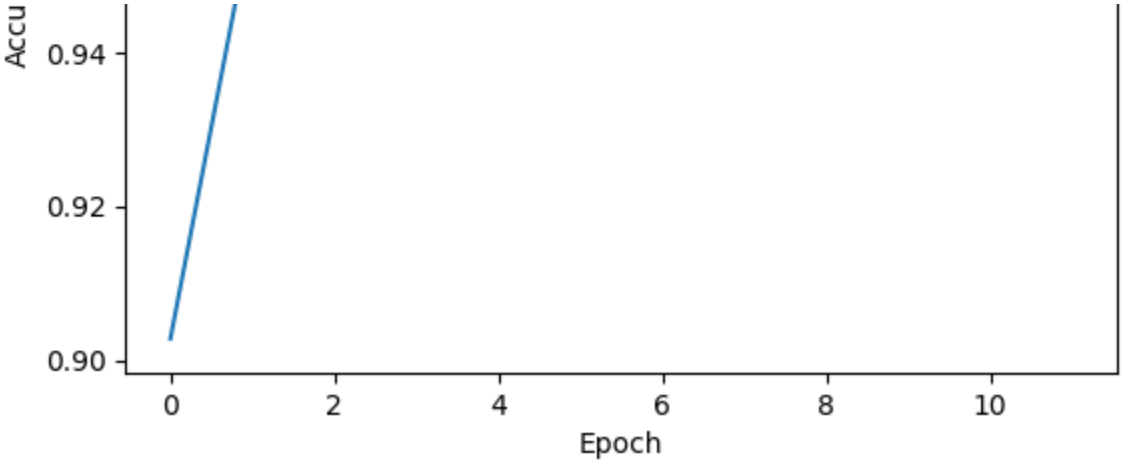
```
Model: "sequential_5"
```

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_5 (Flatten)         (None, 784)               0

 dense_17 (Dense)            (None, 64)                50240

 dense_18 (Dense)            (None, 32)                2080

 dense_19 (Dense)            (None, 16)                528

 dense_20 (Dense)            (None, 10)                170

=================================================================
Total params: 53,018
Trainable params: 53,018
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 7s 4ms/step - loss: 0.3382 - accuracy: 0.9028 - val_loss: 0.1356 - val_
Epoch 2/100
1688/1688 [==============================] - 7s 4ms/step - loss: 0.1424 - accuracy: 0.9579 - val_loss: 0.1199 - val_
Epoch 3/100
1688/1688 [==============================] - 5s 3ms/step - loss: 0.1035 - accuracy: 0.9687 - val_loss: 0.1008 - val_
Epoch 4/100
1688/1688 [==============================] - 7s 4ms/step - loss: 0.0839 - accuracy: 0.9738 - val_loss: 0.0937 - val_
Epoch 5/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0698 - accuracy: 0.9778 - val_loss: 0.1011 - val_
Epoch 6/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0600 - accuracy: 0.9806 - val_loss: 0.0988 - val_
Epoch 7/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0512 - accuracy: 0.9834 - val_loss: 0.0873 - val_
Epoch 8/100
1688/1688 [==============================] - 6s 4ms/step - loss: 0.0446 - accuracy: 0.9855 - val_loss: 0.0888 - val_
Epoch 9/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0412 - accuracy: 0.9868 - val_loss: 0.1046 - val_
Epoch 10/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0330 - accuracy: 0.9884 - val_loss: 0.1050 - val_
Epoch 11/100
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0310 - accuracy: 0.9901 - val_loss: 0.1186 - val_
Epoch 12/100
```

```
1688/1688 [==============================] - 6s 3ms/step - loss: 0.0295 - accuracy: 0.9902 - val_loss: 0.1061 - val_
313/313 [==============================] - 1s 2ms/step - loss: 0.0949 - accuracy: 0.9740
test_loss = 0.09490325301885605 test_acc = 0.9739999771118164
```







```
result
```

	Hidden Layers	Activation Function	Hidden Neurons	Test Loss	Test Acccuracy
0	3	sigmoid	[16, 32, 64]	0.110031	0.9694
1	3	tanh	[16, 32, 64]	0.095483	0.9726
2	3	relu	[16, 32, 64]	0.094903	0.9740

```
best_activation_fn = result.sort_values(
    by=['Test Acccuracy', 'Test Loss'],
    ascending=[False, True]
    )['Activation Function'].iloc[0]

best_activation_fn
```

```
'relu'
```

Task 6

Now run the network by changing the number the Dropout hyper-parameters:

Hidden Layers	Activation Function	Hidden Neurons	Dropout
3	Relu	[16,32,64]	0.9
3	Relu	[16,32,64]	0.75
3	Relu	[16,32,64]	0.5
3	Relu	[16,32,64]	0.25
3	Relu	[16,32,64]	0.10

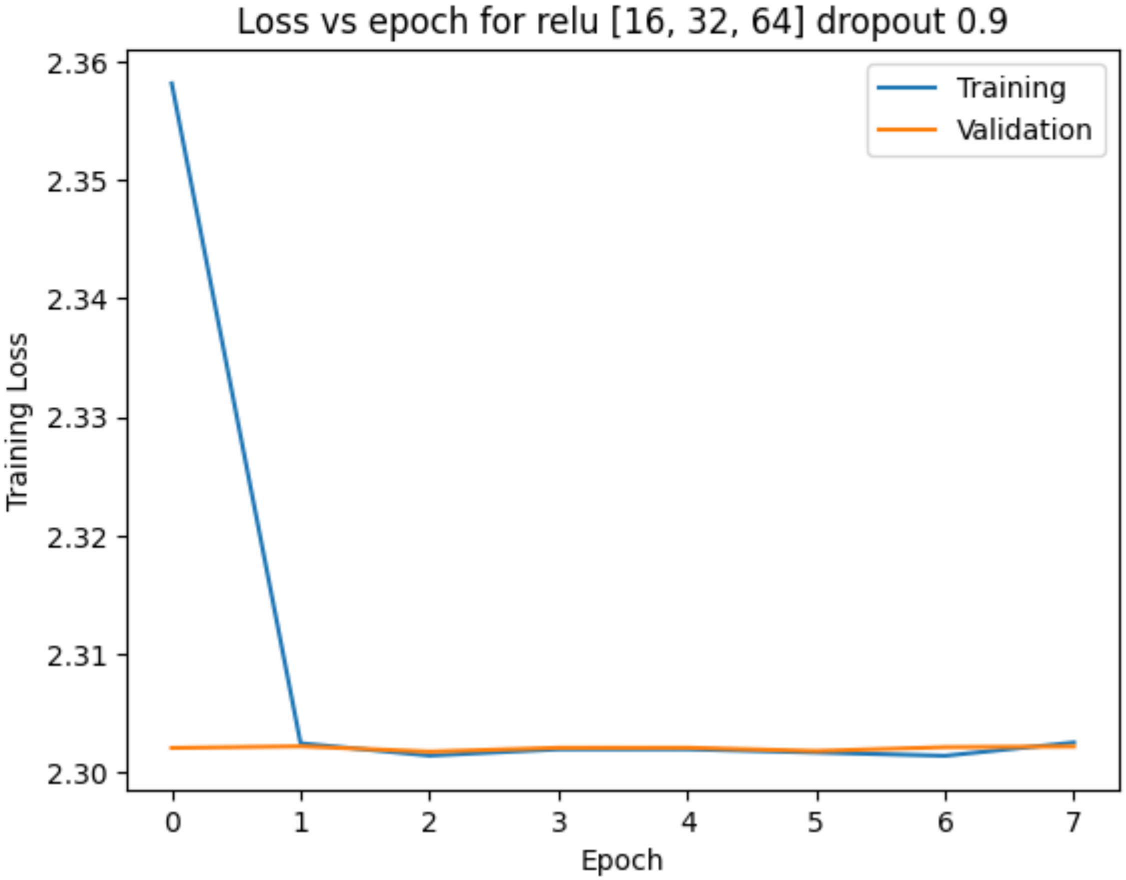
```
result = pd.DataFrame(
    columns=[
        'Hidden Layers',
        'Activation Function',
        'Hidden Neurons',
        'Dropout',
        'Test Loss',
        'Test Acccuracy'],
)
```

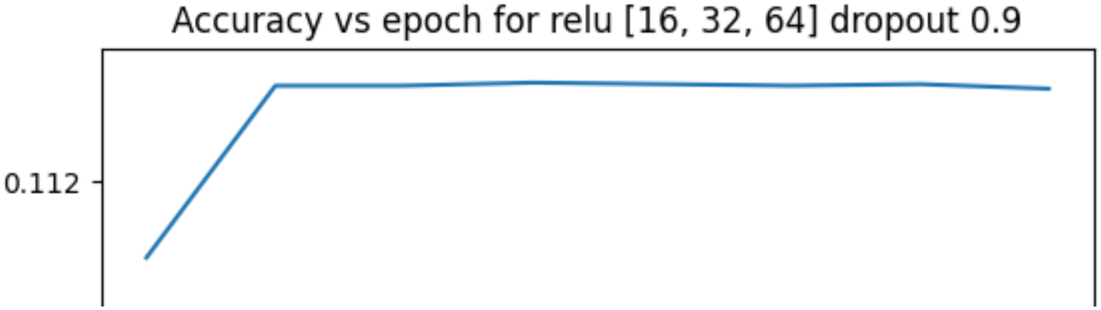
```
hidden_neurons = [16, 32, 64]
activation_function = 'relu'
dropout_val = 0.9

model, history = train_model(activation_function, hidden_neurons,dropout_val)
test_loss, test_acc = model.evaluate(x_test, y_test)

print(f"test_loss = {test_loss} test_acc = {test_acc}")

result.loc[len(result.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    dropout_val,
    test_loss,
    test_acc]
```

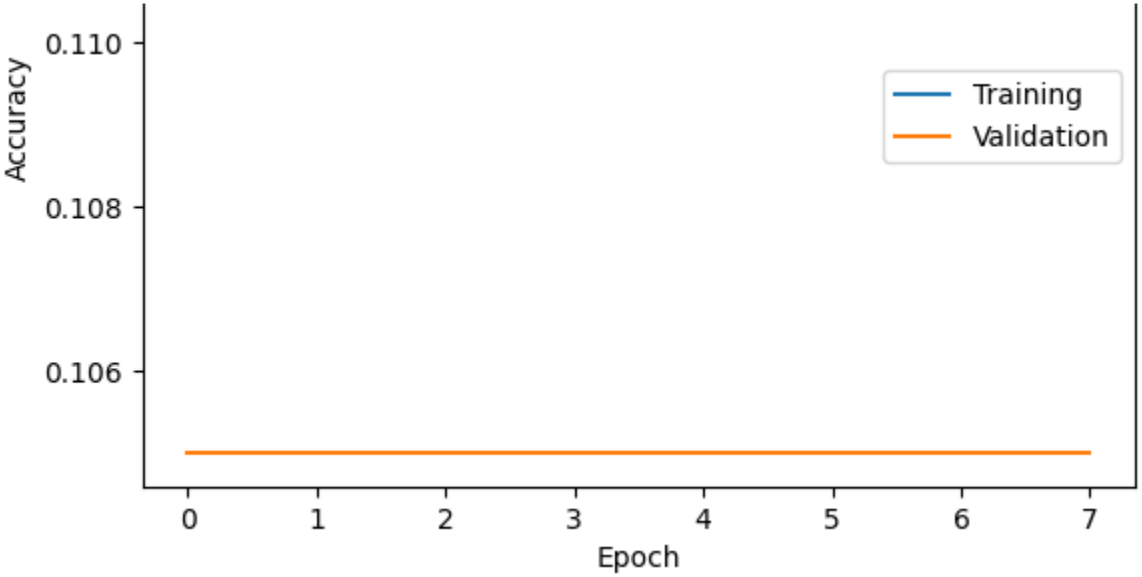
```
plot_history(history, activation_function, hidden_neurons, dropout_val)
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 flatten_6 (Flatten)         (None, 784)               0         
                                                                 
 dense_21 (Dense)            (None, 64)                50240     
                                                                 
 dropout (Dropout)           (None, 64)                0         
                                                                 
 dense_22 (Dense)            (None, 32)                2080      
                                                                 
 dropout_1 (Dropout)         (None, 32)                0         
                                                                 
 dense_23 (Dense)            (None, 16)                528       
                                                                 
 dropout_2 (Dropout)         (None, 16)                0         
                                                                 
 dense_24 (Dense)            (None, 10)                170       
                                                                 
=================================================================
Total params: 53,018
Trainable params: 53,018
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 7s 4ms/step - loss: 2.3582 - accuracy: 0.1111 - val_loss: 2.3021 - val_
Epoch 2/100
1688/1688 [==============================] - 6s 4ms/step - loss: 2.3025 - accuracy: 0.1132 - val_loss: 2.3022 - val_
Epoch 3/100
1688/1688 [==============================] - 6s 4ms/step - loss: 2.3014 - accuracy: 0.1132 - val_loss: 2.3017 - val_
Epoch 4/100
1688/1688 [==============================] - 6s 4ms/step - loss: 2.3019 - accuracy: 0.1132 - val_loss: 2.3021 - val_
Epoch 5/100
1688/1688 [==============================] - 6s 4ms/step - loss: 2.3019 - accuracy: 0.1132 - val_loss: 2.3021 - val_
Epoch 6/100
1688/1688 [==============================] - 6s 4ms/step - loss: 2.3017 - accuracy: 0.1132 - val_loss: 2.3018 - val_
```

```
Epoch 7/100
1688/1688 [==============================] - 6s 4ms/step - loss: 2.3014 - accuracy: 0.1132 - val_loss: 2.3021 - val_
Epoch 8/100
1688/1688 [==============================] - 6s 4ms/step - loss: 2.3025 - accuracy: 0.1131 - val_loss: 2.3022 - val_
313/313 [==============================] - 1s 2ms/step - loss: 2.3011 - accuracy: 0.1135
test_loss = 2.3010966777801514 test_acc = 0.11349999904632568
```





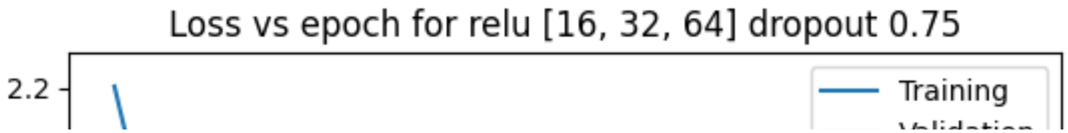


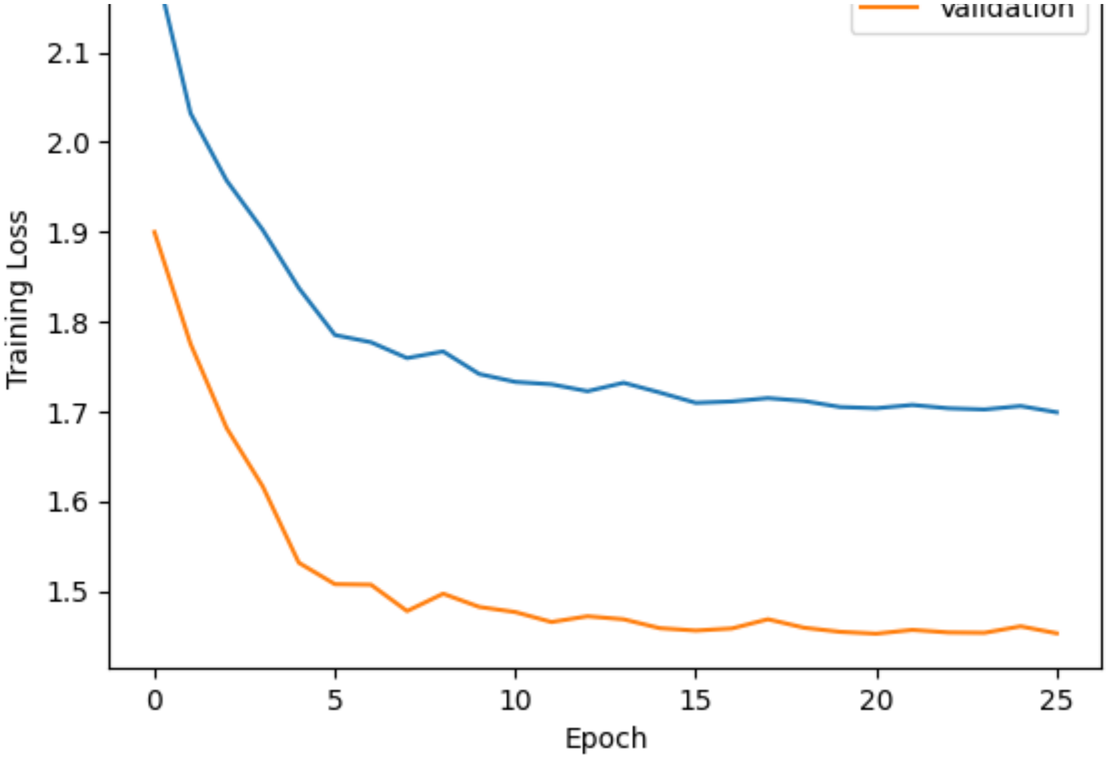
```
hidden_neurons = [16, 32, 64]
activation_function = 'relu'
dropout_val = 0.75

model, history = train_model(activation_function, hidden_neurons, dropout_val)
test_loss, test_acc = model.evaluate(x_test, y_test)

print(f"test_loss = {test_loss} test_acc = {test_acc}")

result.loc[len(result.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    dropout_val,
    test_loss,
    test_acc]

plot_history(history, activation_function, hidden_neurons, dropout_val)
```

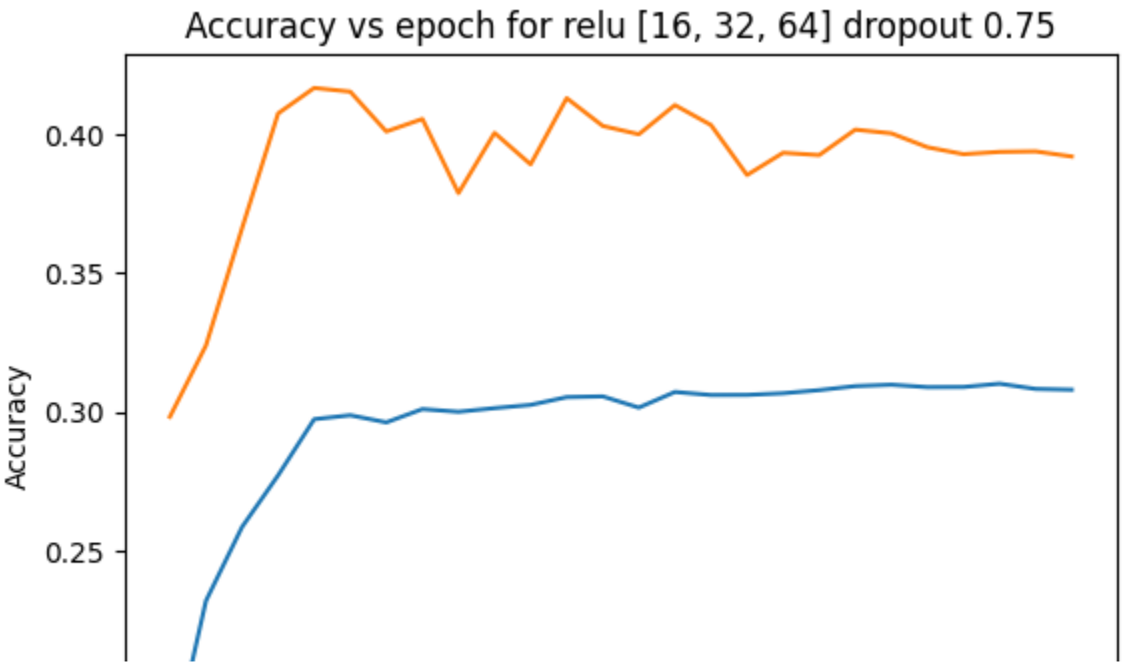
```
Model: "sequential_7"
```

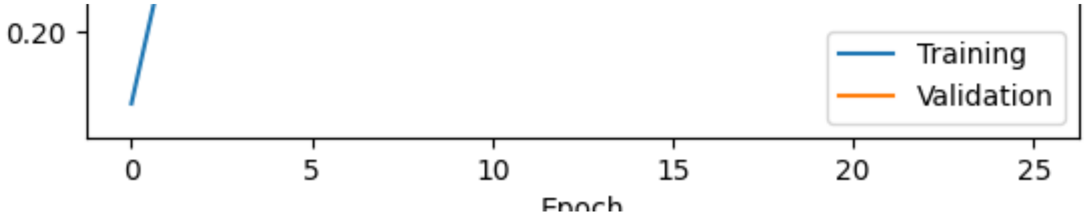
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 flatten_7 (Flatten)         (None, 784)               0         
                                                                 
 dense_25 (Dense)            (None, 64)                50240     
                                                                 
 dropout_3 (Dropout)         (None, 64)                0         
                                                                 
 dense_26 (Dense)            (None, 32)                2080      
                                                                 
 dropout_4 (Dropout)         (None, 32)                0         
                                                                 
 dense_27 (Dense)            (None, 16)                528       
                                                                 
 dropout_5 (Dropout)         (None, 16)                0         
                                                                 
 dense_28 (Dense)            (None, 10)                170       
                                                                 
=================================================================
Total params: 53,018
Trainable params: 53,018
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 7s 4ms/step - loss: 2.2016 - accuracy: 0.1744 - val_loss: 1.8996 - val_
Epoch 2/100
1688/1688 [==============================] - 6s 4ms/step - loss: 2.0319 - accuracy: 0.2319 - val_loss: 1.7752 - val_
Epoch 3/100
1688/1688 [==============================] - 6s 4ms/step - loss: 1.9570 - accuracy: 0.2585 - val_loss: 1.6815 - val_
Epoch 4/100
1688/1688 [==============================] - 6s 4ms/step - loss: 1.9021 - accuracy: 0.2771 - val_loss: 1.6164 - val_
Epoch 5/100
1688/1688 [==============================] - 6s 4ms/step - loss: 1.8374 - accuracy: 0.2973 - val_loss: 1.5315 - val_
Epoch 6/100
1688/1688 [==============================] - 6s 4ms/step - loss: 1.7850 - accuracy: 0.2987 - val_loss: 1.5076 - val_
Epoch 7/100
1688/1688 [==============================] - 6s 4ms/step - loss: 1.7770 - accuracy: 0.2961 - val_loss: 1.5070 - val_
Epoch 8/100
1688/1688 [==============================] - 4s 2ms/step - loss: 1.7594 - accuracy: 0.3009 - val_loss: 1.4776 - val_
Epoch 9/100
```

```
1688/1688 [==============================] - 4627s 3s/step - loss: 1.7667 - accuracy: 0.3000 - val_loss: 1.4970 - va
Epoch 10/100
1688/1688 [==============================] - 4s 3ms/step - loss: 1.7416 - accuracy: 0.3013 - val_loss: 1.4823 - val_
Epoch 11/100
1688/1688 [==============================] - 3s 2ms/step - loss: 1.7328 - accuracy: 0.3024 - val_loss: 1.4765 - val_
Epoch 12/100
1688/1688 [==============================] - 3s 2ms/step - loss: 1.7302 - accuracy: 0.3052 - val_loss: 1.4655 - val_
Epoch 13/100
1688/1688 [==============================] - 3s 2ms/step - loss: 1.7225 - accuracy: 0.3055 - val_loss: 1.4719 - val_
Epoch 14/100
1688/1688 [==============================] - 3s 2ms/step - loss: 1.7319 - accuracy: 0.3015 - val_loss: 1.4685 - val_
Epoch 15/100
1688/1688 [==============================] - 3s 2ms/step - loss: 1.7210 - accuracy: 0.3071 - val_loss: 1.4586 - val_
Epoch 16/100
1688/1688 [==============================] - 3s 2ms/step - loss: 1.7095 - accuracy: 0.3060 - val_loss: 1.4560 - val_
Epoch 17/100
1688/1688 [==============================] - 3s 2ms/step - loss: 1.7110 - accuracy: 0.3060 - val_loss: 1.4583 - val_
Epoch 18/100
1688/1688 [==============================] - 3s 2ms/step - loss: 1.7148 - accuracy: 0.3066 - val_loss: 1.4685 - val_
Epoch 19/100
1688/1688 [==============================] - 3s 2ms/step - loss: 1.7116 - accuracy: 0.3078 - val_loss: 1.4591 - val_
Epoch 20/100
1688/1688 [==============================] - 3s 2ms/step - loss: 1.7049 - accuracy: 0.3092 - val_loss: 1.4545 - val_
Epoch 21/100
1688/1688 [==============================] - 3s 2ms/step - loss: 1.7035 - accuracy: 0.3097 - val_loss: 1.4525 - val_
Epoch 22/100
1688/1688 [==============================] - 3s 2ms/step - loss: 1.7072 - accuracy: 0.3088 - val_loss: 1.4567 - val_
Epoch 23/100
1688/1688 [==============================] - 3s 2ms/step - loss: 1.7034 - accuracy: 0.3089 - val_loss: 1.4539 - val_
Epoch 24/100
1688/1688 [==============================] - 3s 2ms/step - loss: 1.7021 - accuracy: 0.3101 - val_loss: 1.4534 - val_
Epoch 25/100
1688/1688 [==============================] - 4s 3ms/step - loss: 1.7060 - accuracy: 0.3082 - val_loss: 1.4607 - val_
Epoch 26/100
1688/1688 [==============================] - 4s 2ms/step - loss: 1.6990 - accuracy: 0.3079 - val_loss: 1.4529 - val_
313/313 [==============================] - 0s 1ms/step - loss: 1.5254 - accuracy: 0.3913
test_loss = 1.5254249572753906 test_acc = 0.3912999927997589
```




validation
Training Loss
2.1
2.0
1.9
1.8
1.7
1.6
1.5
0
5
10
15
20
25
Epoch

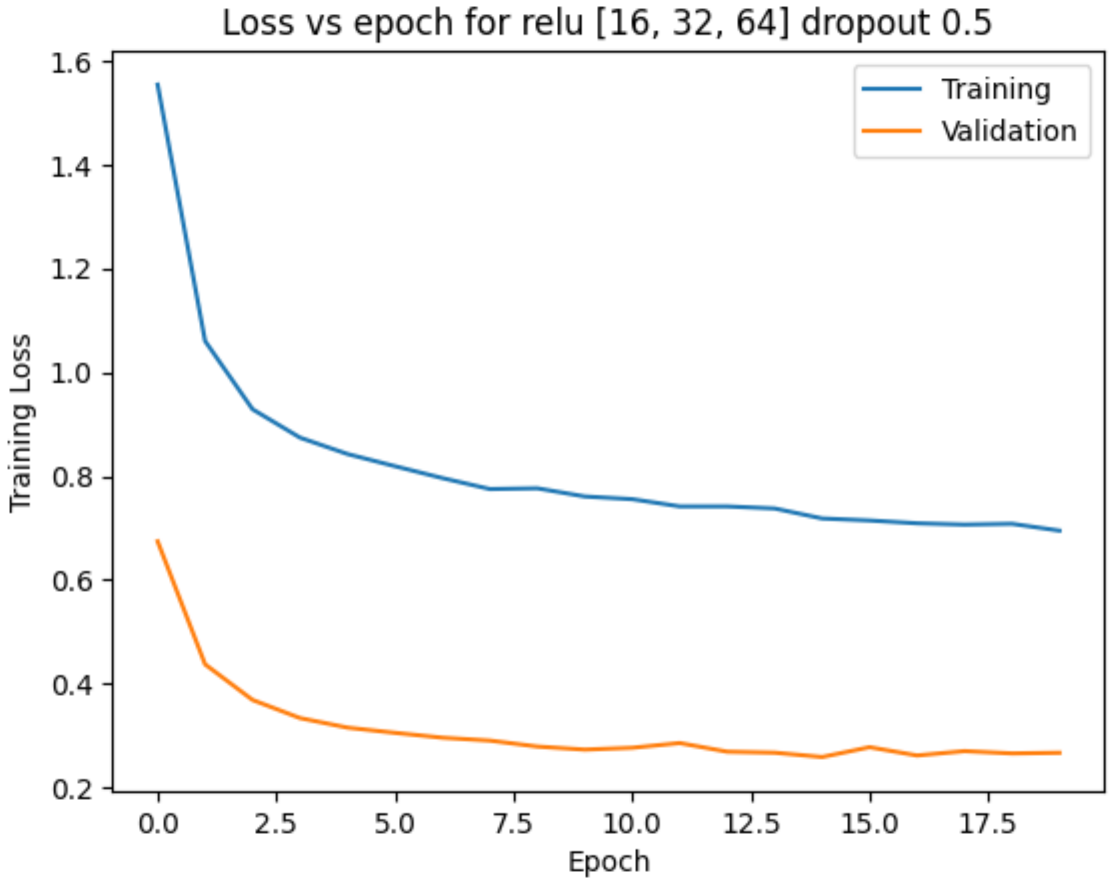

Accuracy vs epoch for relu [16, 32, 64] dropout 0.75
Accuracy
0.40
0.35
0.30
0.25



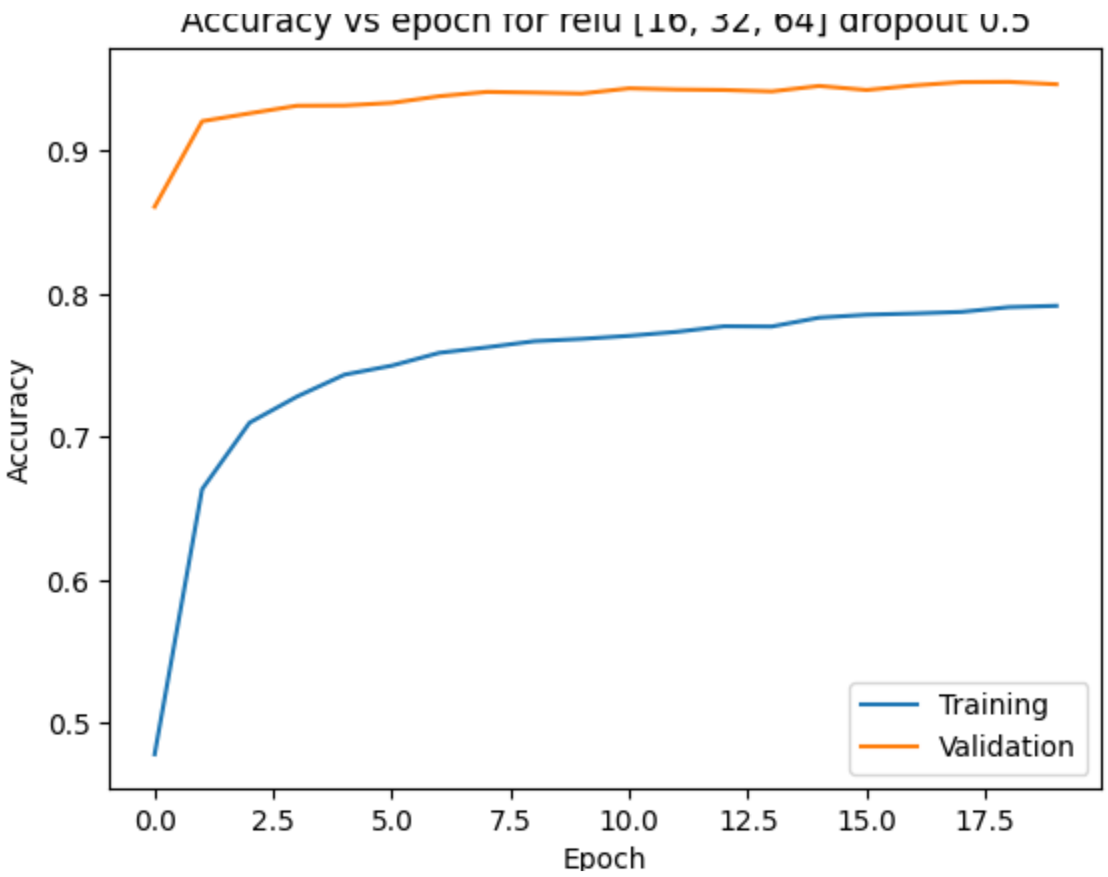
```
hidden_neurons = [16, 32, 64]
activation_function = 'relu'
dropout_val = 0.5

model, history = train_model(activation_function, hidden_neurons, dropout_val)
test_loss, test_acc = model.evaluate(x_test, y_test)

print(f"test_loss = {test_loss} test_acc = {test_acc}")

result.loc[len(result.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    dropout_val,
    test_loss,
    test_acc]

plot_history(history, activation_function, hidden_neurons, dropout_val)
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_8 (Flatten)         (None, 784)               0

 dense_29 (Dense)            (None, 64)                50240

 dropout_6 (Dropout)         (None, 64)                0

 dense_30 (Dense)            (None, 32)                2080
 dropout_7 (Dropout)         (None, 32)                0

 dense_31 (Dense)            (None, 16)                528

 dropout_8 (Dropout)         (None, 16)                0

 dense_32 (Dense)            (None, 10)                170

=================================================================
Total params: 53,018
Trainable params: 53,018
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 5s 3ms/step - loss: 1.5541 - accuracy: 0.4782 - val_loss: 0.6742 - val_
Epoch 2/100
1688/1688 [==============================] - 4s 2ms/step - loss: 1.0602 - accuracy: 0.6635 - val_loss: 0.4373 - val_
Epoch 3/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.9287 - accuracy: 0.7100 - val_loss: 0.3687 - val_
Epoch 4/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.8740 - accuracy: 0.7283 - val_loss: 0.3337 - val_
Epoch 5/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.8425 - accuracy: 0.7436 - val_loss: 0.3154 - val_
Epoch 6/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.8190 - accuracy: 0.7499 - val_loss: 0.3053 - val_
Epoch 7/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7961 - accuracy: 0.7589 - val_loss: 0.2959 - val_
Epoch 8/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7750 - accuracy: 0.7627 - val_loss: 0.2904 - val_
Epoch 9/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.7764 - accuracy: 0.7670 - val_loss: 0.2789 - val_
Epoch 10/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.7607 - accuracy: 0.7686 - val_loss: 0.2730 - val_
Epoch 11/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.7555 - accuracy: 0.7708 - val_loss: 0.2765 - val_
Epoch 12/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7415 - accuracy: 0.7736 - val_loss: 0.2856 - val_
Epoch 13/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7415 - accuracy: 0.7774 - val_loss: 0.2690 - val_
Epoch 14/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.7375 - accuracy: 0.7773 - val_loss: 0.2670 - val_
```

```
Epoch 15/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7183 - accuracy: 0.7835 - val_loss: 0.2587 - val_
Epoch 16/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7144 - accuracy: 0.7855 - val_loss: 0.2777 - val_
Epoch 17/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7090 - accuracy: 0.7864 - val_loss: 0.2617 - val_
Epoch 18/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.7065 - accuracy: 0.7875 - val_loss: 0.2701 - val_
Epoch 19/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.7080 - accuracy: 0.7908 - val_loss: 0.2657 - val_
Epoch 20/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.6948 - accuracy: 0.7917 - val_loss: 0.2669 - val_
313/313 [==============================] - 0s 769us/step - loss: 0.3231 - accuracy: 0.9342
test_loss = 0.3230963349342346 test_acc = 0.9341999888420105
```



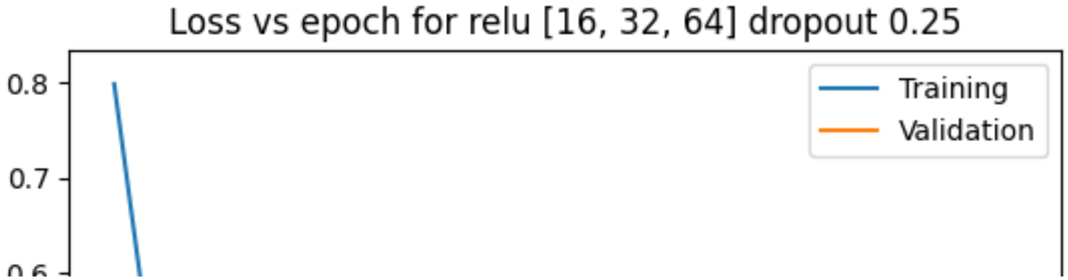
Accuracy vs epoch for relu [16, 32, 64] dropout 0.5

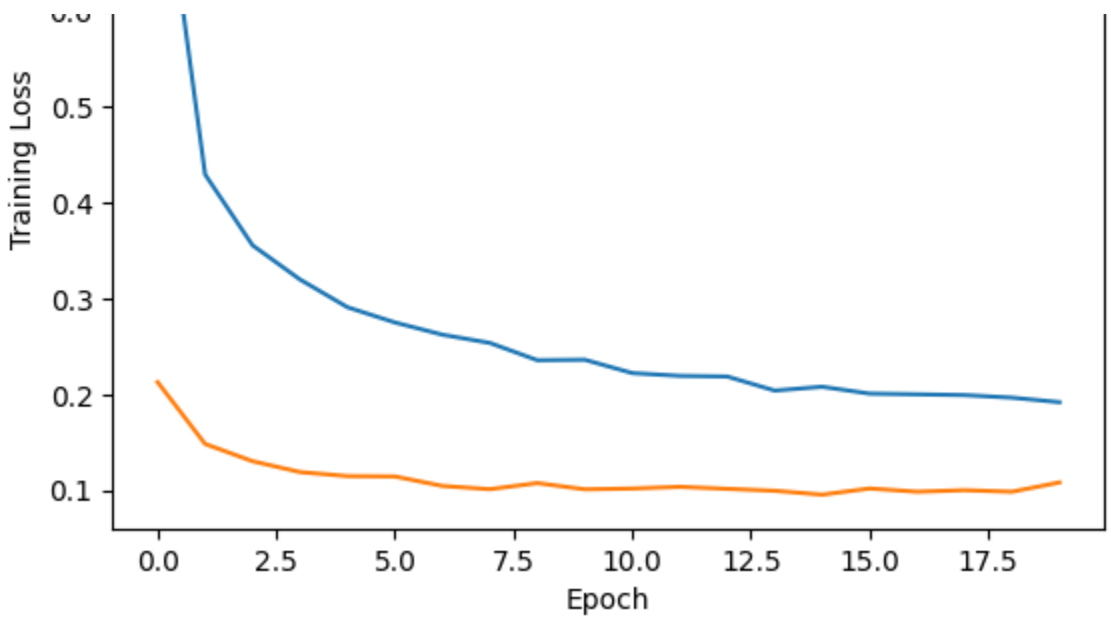


```
hidden_neurons = [16, 32, 64]
activation_function = 'relu'
dropout_val = 0.25

model, history = train_model(activation_function, hidden_neurons, dropout_val)
test_loss, test_acc = model.evaluate(x_test, y_test)

print(f"test_loss = {test_loss} test_acc = {test_acc}")

result.loc[len(result.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    dropout_val,
    test_loss,
    test_acc]

plot_history(history, activation_function, hidden_neurons, dropout_val)
```

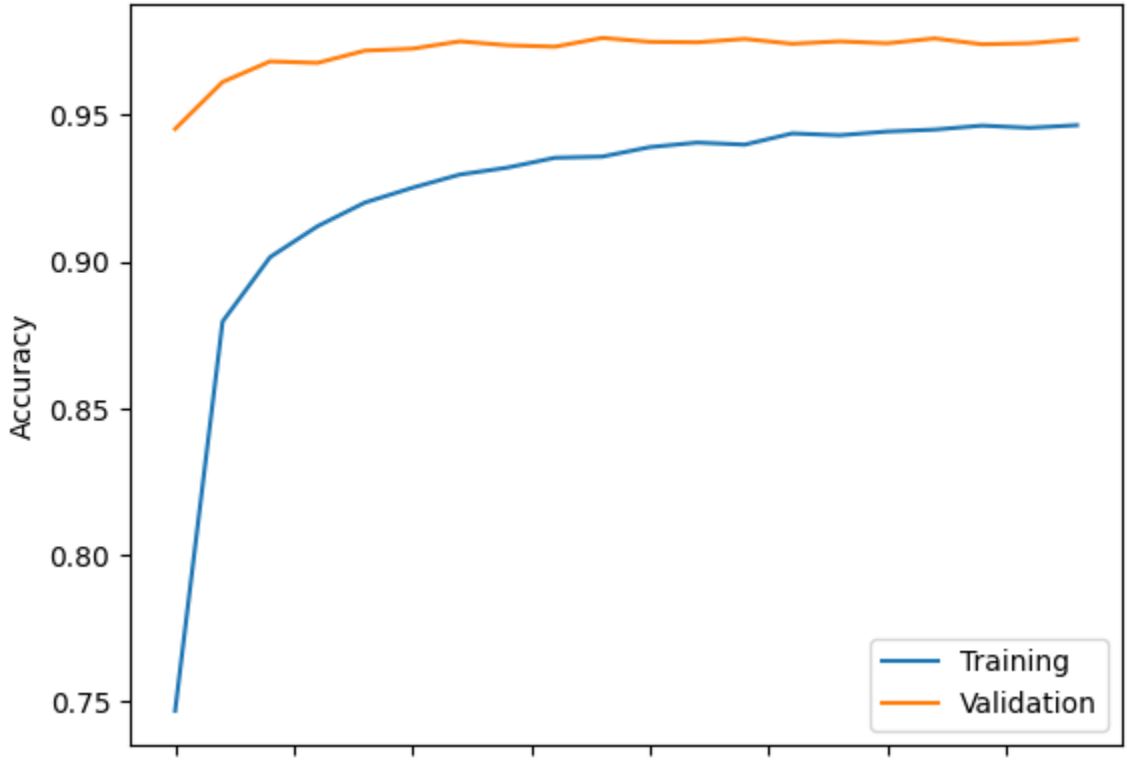
```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_9 (Flatten)         (None, 784)               0

 dense_33 (Dense)            (None, 64)                50240

 dropout_9 (Dropout)         (None, 64)                0

 dense_34 (Dense)            (None, 32)                2080

 dropout_10 (Dropout)        (None, 32)                0

 dense_35 (Dense)            (None, 16)                528

 dropout_11 (Dropout)        (None, 16)                0

 dense_36 (Dense)            (None, 10)                170

=================================================================
Total params: 53,018
Trainable params: 53,018
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.7977 - accuracy: 0.7469 - val_loss: 0.2127 - val_
Epoch 2/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.4295 - accuracy: 0.8796 - val_loss: 0.1481 - val_
Epoch 3/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.3555 - accuracy: 0.9015 - val_loss: 0.1301 - val_
Epoch 4/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.3198 - accuracy: 0.9120 - val_loss: 0.1188 - val_
Epoch 5/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2908 - accuracy: 0.9201 - val_loss: 0.1145 - val_
```

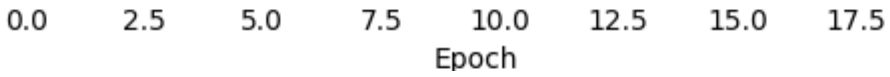
```
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2908 - accuracy: 0.9201 - val_loss: 0.1145 - val_
Epoch 6/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2749 - accuracy: 0.9251 - val_loss: 0.1142 - val_
Epoch 7/100
1688/1688 [==============================] - 3s 1ms/step - loss: 0.2622 - accuracy: 0.9296 - val_loss: 0.1043 - val_
Epoch 8/100
1688/1688 [==============================] - 3s 1ms/step - loss: 0.2536 - accuracy: 0.9320 - val_loss: 0.1009 - val_
Epoch 9/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.2355 - accuracy: 0.9353 - val_loss: 0.1073 - val_
Epoch 10/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2360 - accuracy: 0.9357 - val_loss: 0.1008 - val_
Epoch 11/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2221 - accuracy: 0.9390 - val_loss: 0.1016 - val_
Epoch 12/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.2192 - accuracy: 0.9405 - val_loss: 0.1034 - val_
Epoch 13/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.2185 - accuracy: 0.9398 - val_loss: 0.1013 - val_
Epoch 14/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.2039 - accuracy: 0.9436 - val_loss: 0.0992 - val_
Epoch 15/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2078 - accuracy: 0.9430 - val_loss: 0.0952 - val_
Epoch 16/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.2007 - accuracy: 0.9443 - val_loss: 0.1016 - val_
Epoch 17/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.1999 - accuracy: 0.9449 - val_loss: 0.0983 - val_
Epoch 18/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.1991 - accuracy: 0.9463 - val_loss: 0.0998 - val_
Epoch 19/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.1965 - accuracy: 0.9455 - val_loss: 0.0982 - val_
Epoch 20/100
1688/1688 [==============================] - 3s 1ms/step - loss: 0.1917 - accuracy: 0.9464 - val_loss: 0.1079 - val_
313/313 [==============================] - 0s 795us/step - loss: 0.1285 - accuracy: 0.9676
test_loss = 0.12848332524299622 test_acc = 0.9675999879837036
```





Accuracy vs epoch for relu [16, 32, 64] dropout 0.25



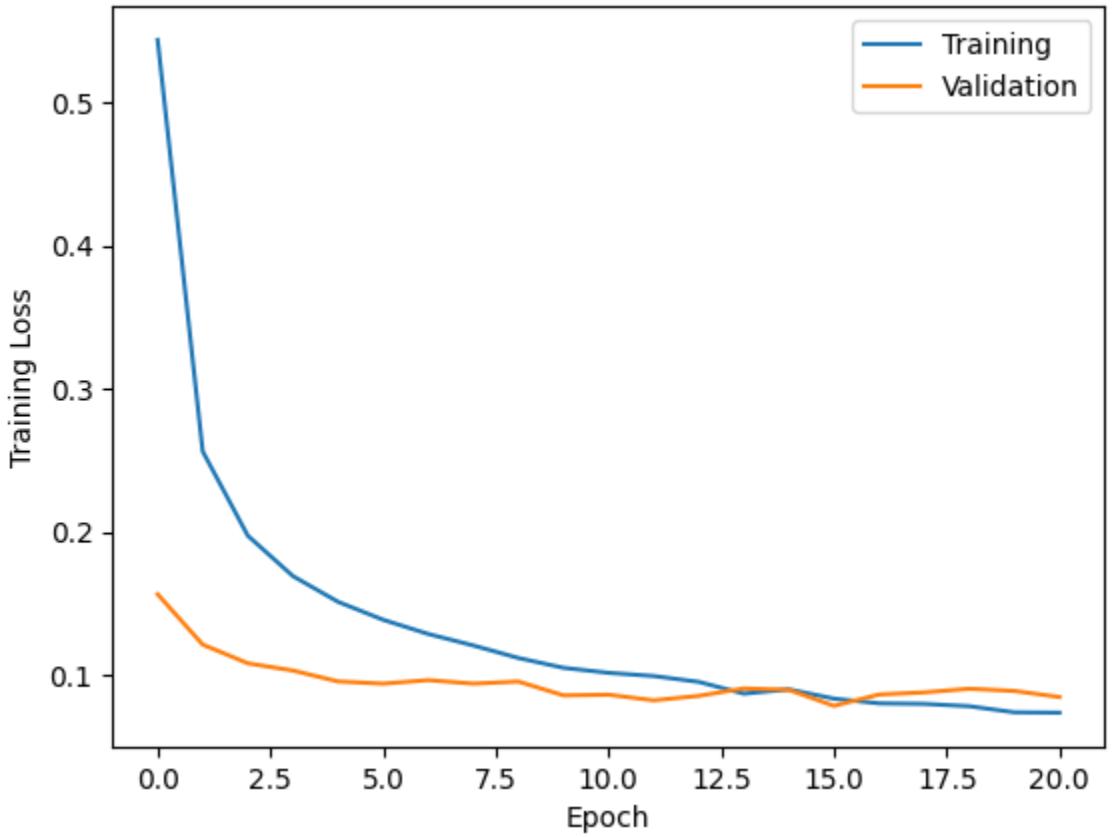


```
hidden_neurons = [16, 32, 64]
activation_function = 'relu'
dropout_val = 0.1

model, history = train_model(activation_function, hidden_neurons, dropout_val)
test_loss, test_acc = model.evaluate(x_test, y_test)

print(f"test_loss = {test_loss} test_acc = {test_acc}")

result.loc[len(result.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    dropout_val,
    test_loss,
    test_acc]

plot_history(history, activation_function, hidden_neurons, dropout_val)
```

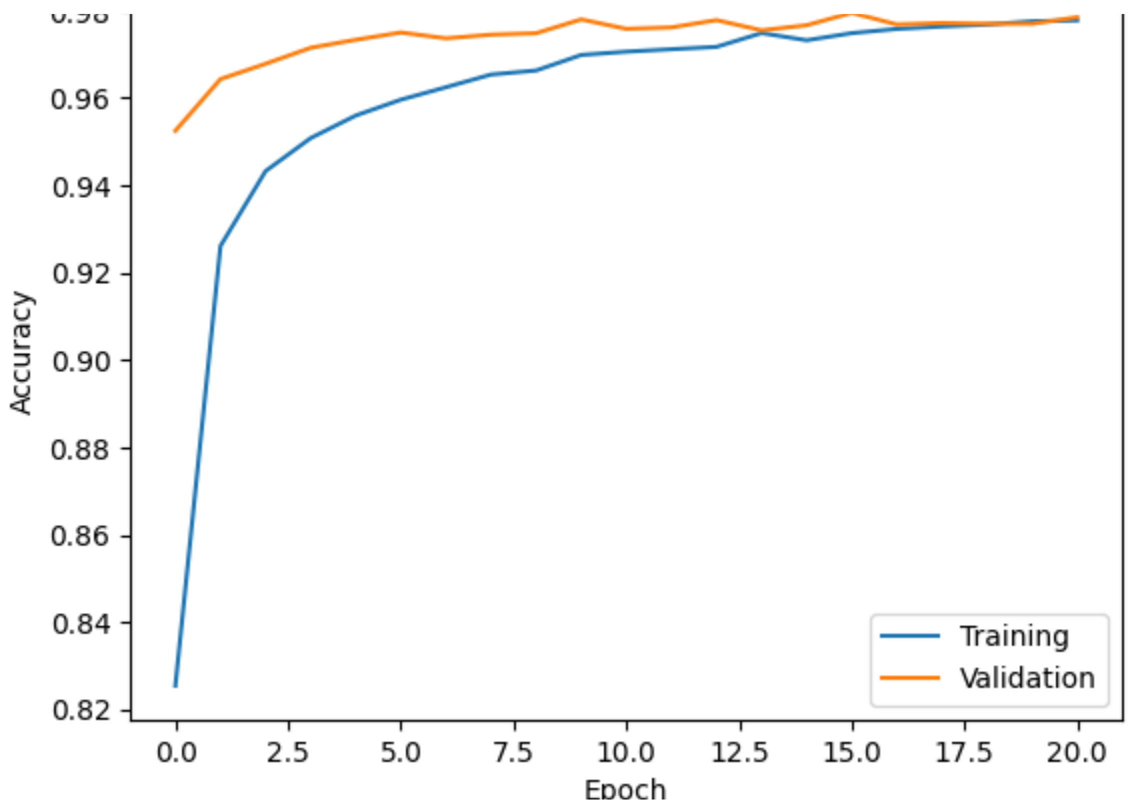
```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 flatten_10 (Flatten)        (None, 784)               0         
                                                                 
 dense_37 (Dense)            (None, 64)                50240     
                                                                 
 dropout_12 (Dropout)        (None, 64)                0         
                                                                 
 dense_38 (Dense)            (None, 32)                2080      
                                                                 
 dropout_13 (Dropout)        (None, 32)                0         
                                                                 
 dense_39 (Dense)            (None, 16)                528       
 dropout_14 (Dropout)        (None, 16)                0

 dense_40 (Dense)            (None, 10)                170

=================================================================
Total params: 53,018
Trainable params: 53,018
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.5433 - accuracy: 0.8254 - val_loss: 0.1566 - val_
Epoch 2/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2563 - accuracy: 0.9261 - val_loss: 0.1215 - val_
Epoch 3/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.1975 - accuracy: 0.9433 - val_loss: 0.1085 - val_
Epoch 4/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.1693 - accuracy: 0.9508 - val_loss: 0.1034 - val_
Epoch 5/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.1514 - accuracy: 0.9560 - val_loss: 0.0958 - val_
Epoch 6/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.1388 - accuracy: 0.9596 - val_loss: 0.0943 - val_
Epoch 7/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.1289 - accuracy: 0.9625 - val_loss: 0.0967 - val_
Epoch 8/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.1209 - accuracy: 0.9654 - val_loss: 0.0943 - val_
Epoch 9/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1122 - accuracy: 0.9663 - val_loss: 0.0957 - val_
Epoch 10/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.1053 - accuracy: 0.9699 - val_loss: 0.0861 - val_
Epoch 11/100
1688/1688 [==============================] - 3s 1ms/step - loss: 0.1018 - accuracy: 0.9706 - val_loss: 0.0866 - val_
Epoch 12/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.0996 - accuracy: 0.9712 - val_loss: 0.0826 - val_
Epoch 13/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.0955 - accuracy: 0.9717 - val_loss: 0.0857 - val_
Epoch 14/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.0873 - accuracy: 0.9749 - val_loss: 0.0909 - val_
Epoch 15/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.0902 - accuracy: 0.9732 - val_loss: 0.0899 - val_
Epoch 16/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.0838 - accuracy: 0.9749 - val_loss: 0.0788 - val_
```

```
Epoch 17/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.0805 - accuracy: 0.9758 - val_loss: 0.0867 - val_
Epoch 18/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.0802 - accuracy: 0.9764 - val_loss: 0.0882 - val_
Epoch 19/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.0785 - accuracy: 0.9768 - val_loss: 0.0907 - val_
Epoch 20/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.0743 - accuracy: 0.9776 - val_loss: 0.0892 - val_
Epoch 21/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.0740 - accuracy: 0.9778 - val_loss: 0.0849 - val_
313/313 [==============================] - 0s 795us/step - loss: 0.1021 - accuracy: 0.9734
test_loss = 0.10208451747894287 test_acc = 0.9733999967575073
```

Loss vs epoch for relu [16, 32, 64] dropout 0.1



Accuracy vs epoch for relu [16, 32, 64] dropout 0.1



```
result
```

	Hidden Layers	Activation Function	Hidden Neurons	Dropout	Test Loss	Test Acccuracy
0	3	relu	[16, 32, 64]	0.90	2.301097	0.1135
1	3	relu	[16, 32, 64]	0.75	1.525425	0.3913
2	3	relu	[16, 32, 64]	0.50	0.323096	0.9342
3	3	relu	[16, 32, 64]	0.25	0.128483	0.9676
4	3	relu	[16, 32, 64]	0.10	0.102085	0.9734

```
best_dropout = result.sort_values(
    by=['Test Acccuracy', 'Test Loss']
    by=['Test Accuracy', 'Test Loss'],
    ascending=[False, True]
    )['Dropout'].iloc[0]

best_dropout

     0.1
```

Task 7

Plot the graph for loss vs epoch and accuracy(train, validation, accuracy) vs epoch for all the above cases. Point out the logic in the report.

Task 8

With the best set hyperparameter from above run vary the Adam Optimizer learning rate [0.01, 0.001, 0.005, 0.0001, 0.0005]. Print the time to achieve the best validation accuracy (as reported before from all run) for all these five run .

```
print(f"best activation function: {best_activation_fn}")
print(f"best dropout value: {best_dropout}")

     best activation function: relu
     best dropout value: 0.1
```

```
import time

result = pd.DataFrame(
    columns=[
        'Hidden Layers',
        'Activation Function',
        'Hidden Neurons',
        'Dropout',
        'Adam Learn Rate',
        'Time Taken'
        'Time Taken',
        'Test Loss',
        'Test Acccuracy'],
)


hidden_neurons = [16, 32, 64]
adam_learn_rates = [0.01,  0.001,  0.005,  0.0001,  0.0005]

for learn_rate in adam_learn_rates:
    start_time = time.time()
    model, _ = train_model(
        activation_function=best_activation_fn,
        hidden_neurons=hidden_neurons,
        adam_learn_rate=learn_rate,
        dropout_rate=best_dropout,
        verbose=False
    )
    end_time = time.time()

    time_taken = end_time - start_time

    test_loss, test_acc = model.evaluate(x_test, y_test, verbose=0)

    result.loc[len(result.index)] = [
    len(hidden_neurons),
    best_activation_fn,
    str(hidden_neurons),
    best_dropout,
    learn_rate,
    time_taken,
    test_loss,
    test_acc]



result
```

	Hidden	Activation	Hidden	Adam Learn	Time	Test	Test

	Hidden Layers	Activation Function	Hidden Neurons	Dropout	Adam Learn Rate	Time Taken	Test Loss	Test Acccuracy
0	3	relu	[16, 32, 64]	0.1	0.0100	28.731697	0.180212	0.9548
1	3	relu	[16, 32, 64]	0.1	0.0010	24.480594	0.100439	0.9731
2	3	relu	[16, 32, 64]	0.1	0.0050	24.704880	0.129875	0.9664
3	3	relu	[16, 32, 64]	0.1	0.0001	66.947391	0.103917	0.9708
4	3	relu	[16, 32, 64]	0.1	0.0005	38.584798	0.095270	0.9725

```
best_adam_learn_rate = result.sort_values(
    by=['Test Acccuracy', 'Test Loss'],
    ascending=[False, True]
    )['Adam Learn Rate'].iloc[0]

best_adam_learn_rate
```

```
0.001
```

Task 9

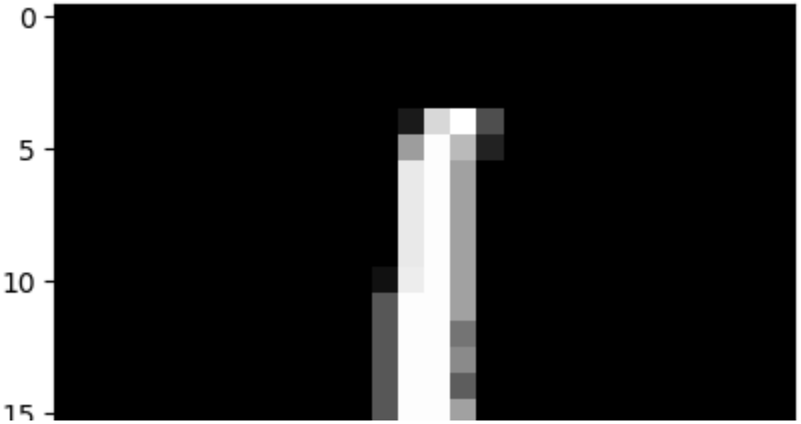
Create five image(size 28*28) containing a digit of your won handwriting and test whether your trained classifier is able to predict it or not.

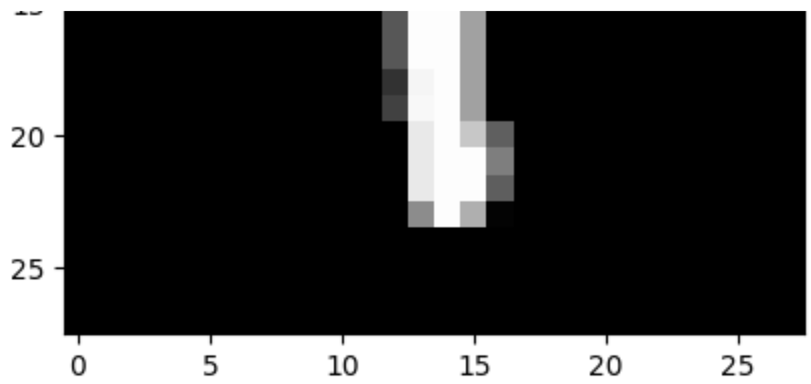
```
model, _ = train_model(
    activation_function=best_activation_fn,
    hidden_neurons=hidden_neurons,
    adam_learn_rate=best_adam_learn_rate,
    dropout_rate=best_dropout,
    verbose=True
)
```

```
Model: "sequential_16"
_________________________________________________________________
```

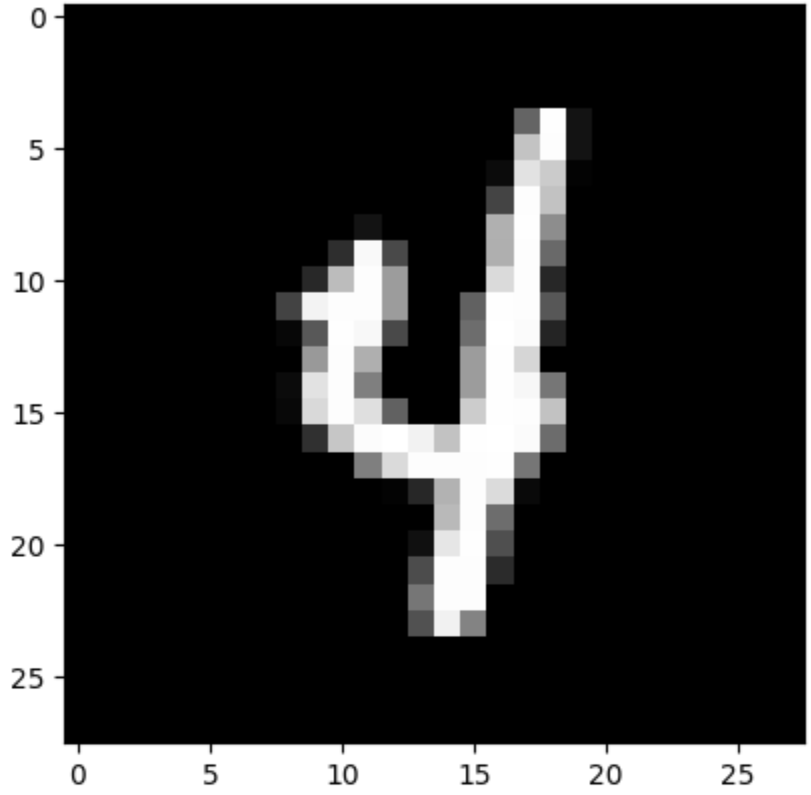
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 flatten_16 (Flatten)        (None, 784)               0         
                                                                 
 dense_61 (Dense)            (None, 64)                50240     
                                                                 
 dropout_30 (Dropout)        (None, 64)                0         
                                                                 
 dense_62 (Dense)            (None, 32)                2080      
                                                                 
 dropout_31 (Dropout)        (None, 32)                0         
                                                                 
 dense_63 (Dense)            (None, 16)                528       
                                                                 
 dropout_32 (Dropout)        (None, 16)                0         
                                                                 
 dense_64 (Dense)            (None, 10)                170       
                                                                 
=================================================================
Total params: 53,018
Trainable params: 53,018
Non-trainable params: 0
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 3s 1ms/step - loss: 0.5321 - accuracy: 0.8379 - val_loss: 0.1541 - val_
Epoch 2/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.2498 - accuracy: 0.9280 - val_loss: 0.1052 - val_
Epoch 3/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.2000 - accuracy: 0.9430 - val_loss: 0.1010 - val_
Epoch 4/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1677 - accuracy: 0.9524 - val_loss: 0.0918 - val_
Epoch 5/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1518 - accuracy: 0.9562 - val_loss: 0.0982 - val_
Epoch 6/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1371 - accuracy: 0.9602 - val_loss: 0.0831 - val_
Epoch 7/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1267 - accuracy: 0.9632 - val_loss: 0.0845 - val_
Epoch 8/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1192 - accuracy: 0.9654 - val_loss: 0.0822 - val_
Epoch 9/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1133 - accuracy: 0.9666 - val_loss: 0.0735 - val_
Epoch 10/100
Epoch 10/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1064 - accuracy: 0.9687 - val_loss: 0.0828 - val_
Epoch 11/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.1044 - accuracy: 0.9697 - val_loss: 0.0813 - val_
Epoch 12/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.0971 - accuracy: 0.9715 - val_loss: 0.0775 - val_
Epoch 13/100
1688/1688 [==============================] - 3s 2ms/step - loss: 0.0936 - accuracy: 0.9724 - val_loss: 0.0784 - val_
Epoch 14/100
1688/1688 [==============================] - 2s 1ms/step - loss: 0.0881 - accuracy: 0.9739 - val_loss: 0.0763 - val_
```

```
import random

random_idx = random.sample(range(0, len(x_test)), 10)

img_to_predict = np.array([x_test[idx] for idx in random_idx])

categorical_predictions = model.predict(img_to_predict)

for img, cat_pred in zip(img_to_predict, categorical_predictions):
    plt.imshow(img, cmap='gray')
    plt.show()
    pred = np.argmax(cat_pred)
    print(f"predict = {pred}")
```

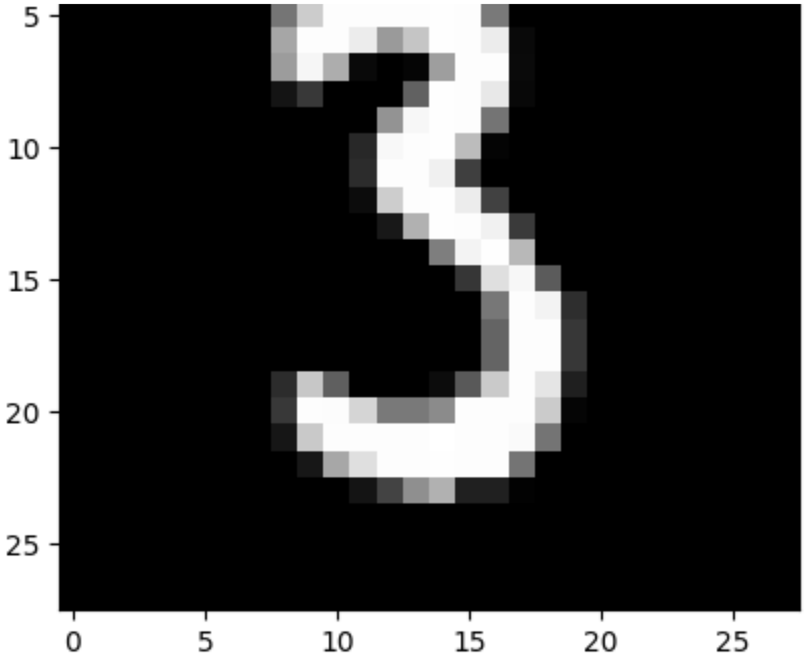




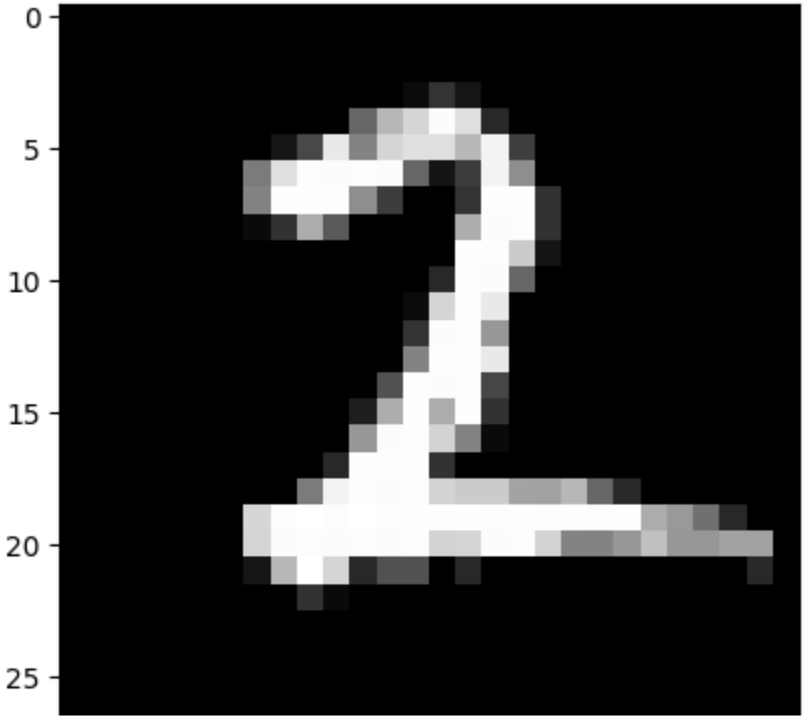
```
predict = 1
```

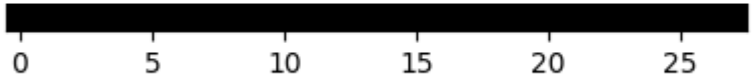


```
predict = 4
```

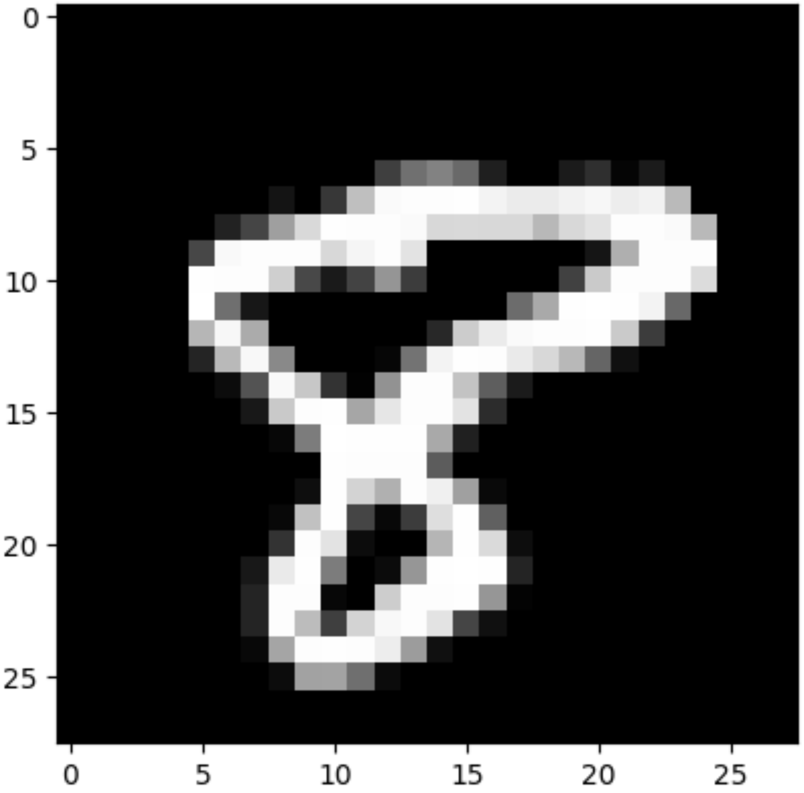


```
predict = 3
```

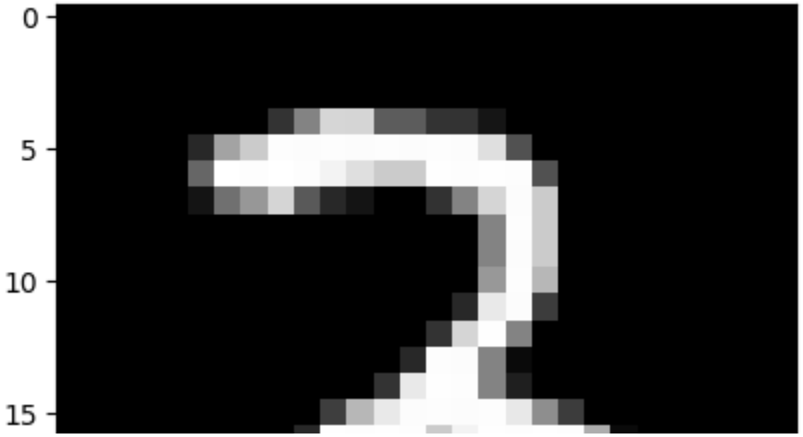


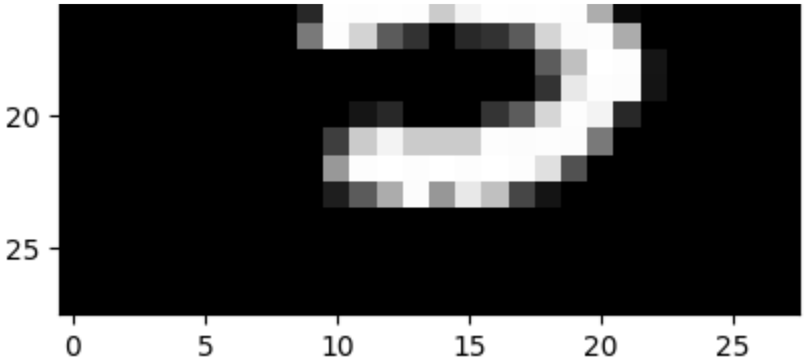


```
predict = 2
```

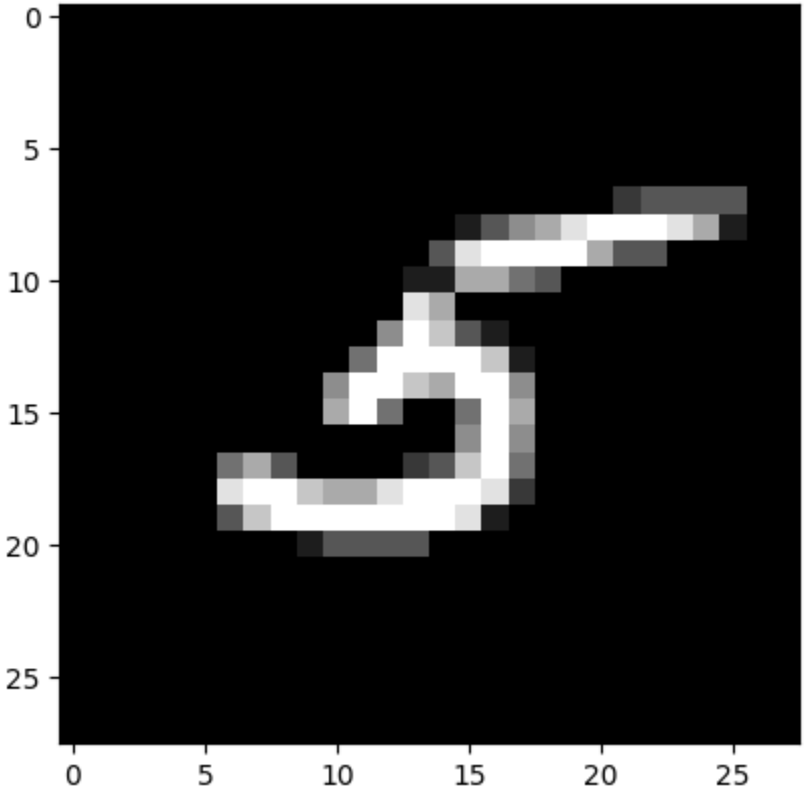


```
predict = 8
```

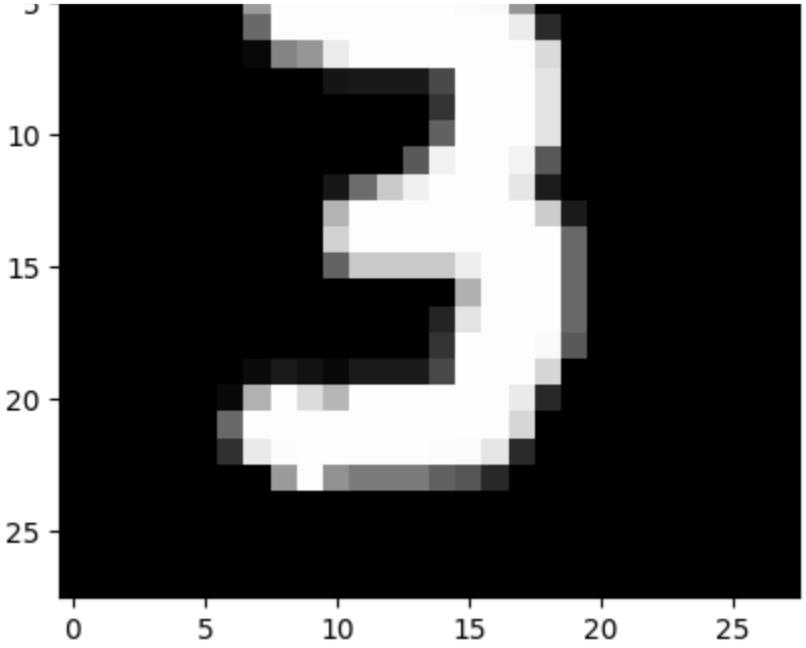




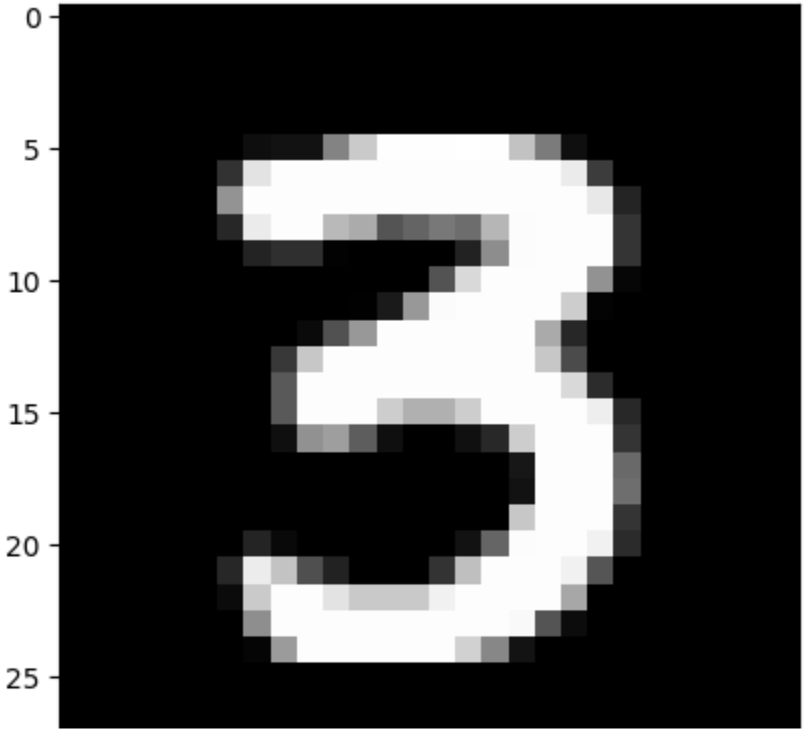
```
predict = 3
```

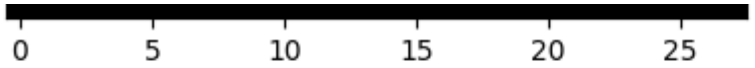


```
predict = 5
```

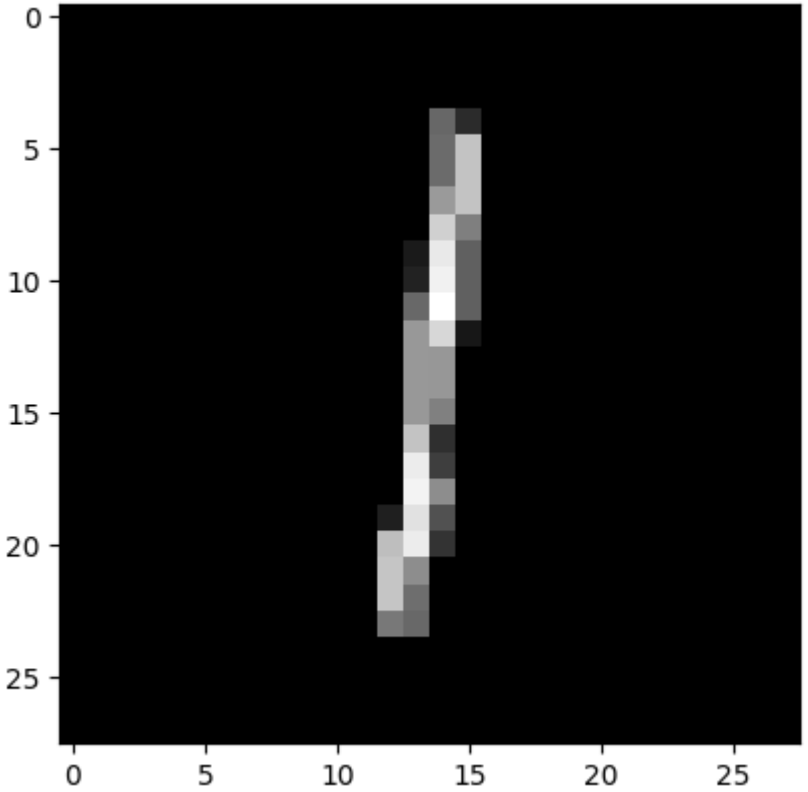


```
predict = 3
```





```
predict = 3
```



```
predict = 1
```