Machine Learning Project

Name: Singa Saiprasad

Title:-**NEURAL NETWORK-BASED HANDWRITTEN DIGIT RECOGNITION**

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

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.

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
from matplotlib import pyplot as plt
mnist_data = tf.keras.datasets.mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-
keras-datasets/mnist.npz
11490434/11490434 [==============================] - 0s 0us/step
```

```
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]]], 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]],

         [[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([7, 2, 1, ..., 4, 5, 6], dtype=uint8)))
```

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,)
```

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])

def transform(image):
    image = np.pad(image, (2, 2))
    c = np.mean(image)
    s = np.std(image)
    return RBF(image, c, s).flatten()
# flatten reduces each image into a 1-D array by storing it in row-major format


x_train_tf=[]
```

```
x_test_tf=[]
for image in x_train:
    x_train_tf.append(transform(image))
x_train_tf = np.array(x_train_tf)
print("Shape of x_train after transforming: ", x_train_tf.shape)

for image in x_test:
    x_test_tf.append(transform(image))
x_test_tf = np.array(x_test_tf)
print("Shape of x_test after transforming: ", x_test_tf.shape)



# 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}")
```

```
Shape of x_train after transforming:  (60000, 32)
Shape of x_test after transforming:  (10000, 32)
```

```
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)
```

```
print(x_train[25].shape)
```

```
(28, 28)
```

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

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
    # (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
                            )
```

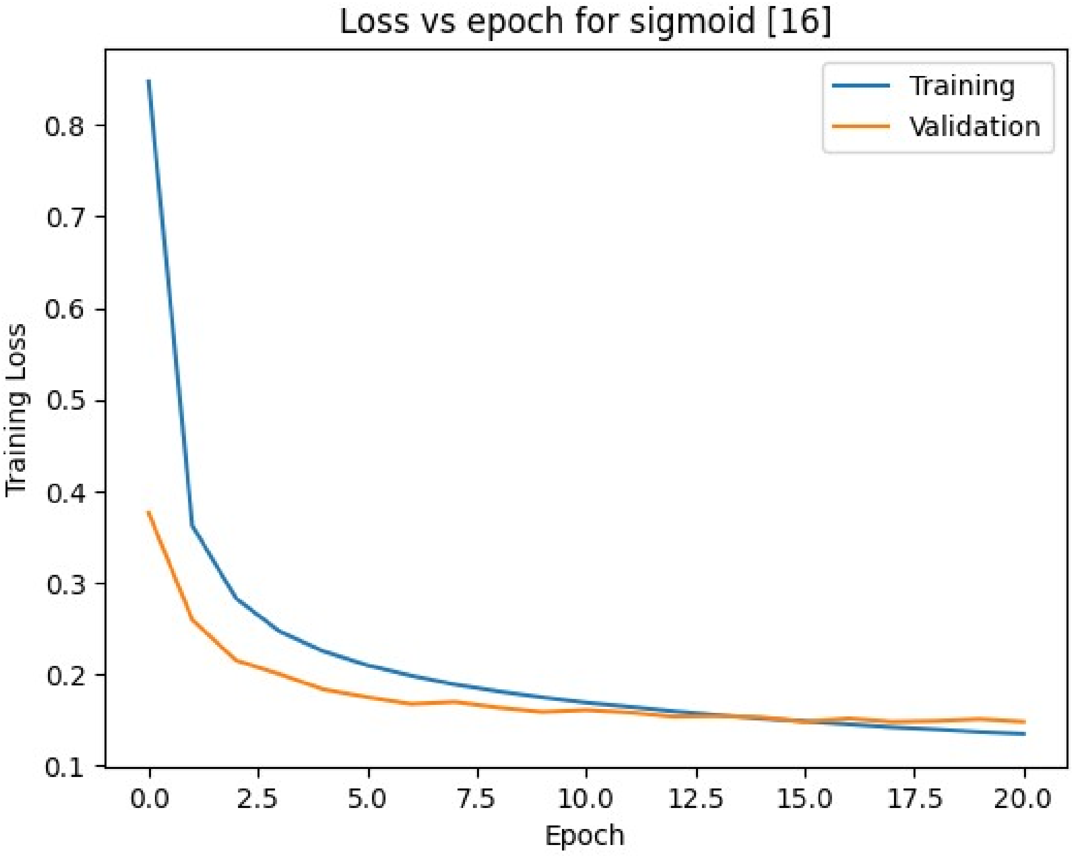
```
                        ],
                        verbose='auto' if verbose else 0
                        )

    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()

    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',
```

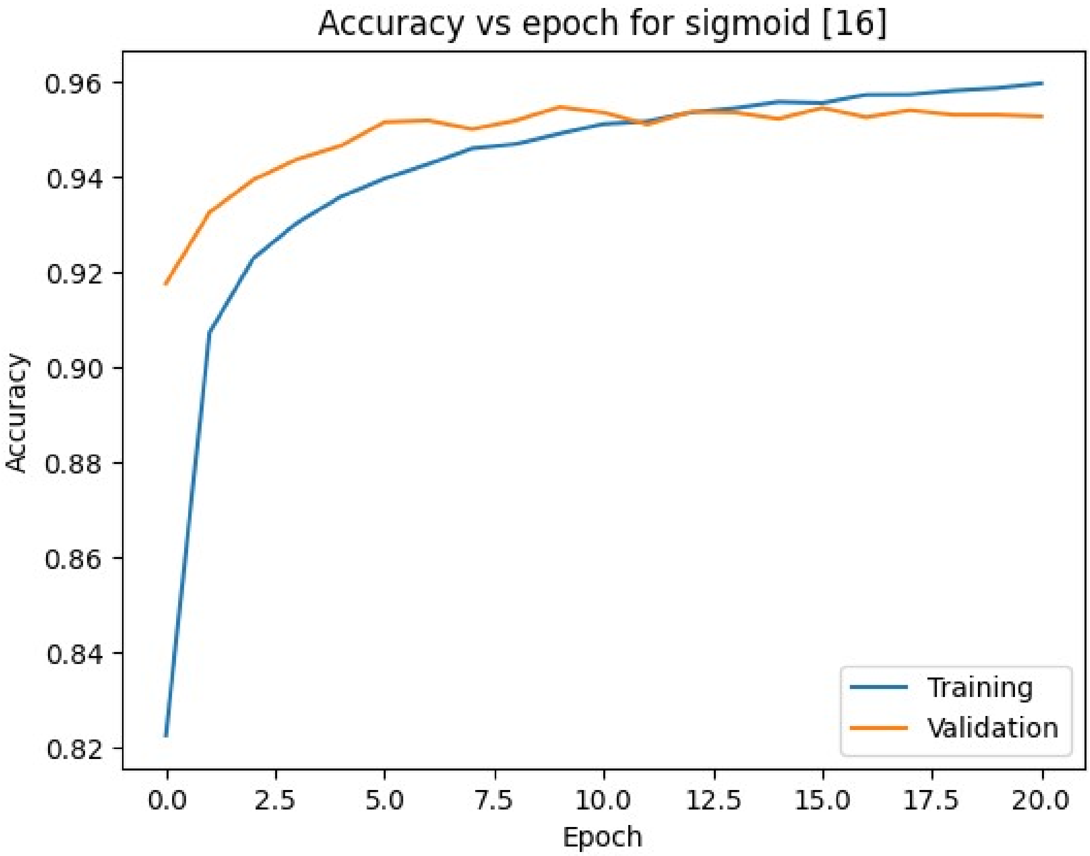
```
        'Activation Function',
        'Hidden Neurons',
        'Test Loss',
        'Test Acccuracy'],
)

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: 12730 (49.73 KB)
Trainable params: 12730 (49.73 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.8473 - accuracy: 0.8224 - val_loss: 0.3763 - val_accuracy: 0.9177
Epoch 2/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.3623 - accuracy: 0.9073 - val_loss: 0.2594 - val_accuracy: 0.9327
Epoch 3/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.2828 - accuracy: 0.9230 - val_loss: 0.2151 - val_accuracy: 0.9395
Epoch 4/100
1688/1688 [==============================] - 4s 3ms/step - loss:
0.2466 - accuracy: 0.9304 - val_loss: 0.2000 - val_accuracy: 0.9438
```

```
Epoch 5/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.2251 - accuracy: 0.9359 - val_loss: 0.1835 - val_accuracy: 0.9467
Epoch 6/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.2098 - accuracy: 0.9397 - val_loss: 0.1750 - val_accuracy: 0.9517
Epoch 7/100
1688/1688 [==============================] - 4s 2ms/step - loss:
0.1985 - accuracy: 0.9428 - val_loss: 0.1675 - val_accuracy: 0.9520
Epoch 8/100
1688/1688 [==============================] - 4s 2ms/step - loss:
0.1889 - accuracy: 0.9461 - val_loss: 0.1697 - val_accuracy: 0.9502
Epoch 9/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.1814 - accuracy: 0.9470 - val_loss: 0.1636 - val_accuracy: 0.9520
Epoch 10/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.1748 - accuracy: 0.9492 - val_loss: 0.1589 - val_accuracy: 0.9548
Epoch 11/100
1688/1688 [==============================] - 4s 3ms/step - loss:
0.1690 - accuracy: 0.9513 - val_loss: 0.1610 - val_accuracy: 0.9537
Epoch 12/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.1644 - accuracy: 0.9518 - val_loss: 0.1582 - val_accuracy: 0.9512
Epoch 13/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.1596 - accuracy: 0.9537 - val_loss: 0.1536 - val_accuracy: 0.9538
Epoch 14/100
1688/1688 [==============================] - 4s 2ms/step - loss:
0.1555 - accuracy: 0.9546 - val_loss: 0.1543 - val_accuracy: 0.9537
Epoch 15/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.1515 - accuracy: 0.9559 - val_loss: 0.1529 - val_accuracy: 0.9523
Epoch 16/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.1487 - accuracy: 0.9557 - val_loss: 0.1480 - val_accuracy: 0.9547
Epoch 17/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.1454 - accuracy: 0.9574 - val_loss: 0.1516 - val_accuracy: 0.9527
Epoch 18/100
1688/1688 [==============================] - 4s 2ms/step - loss:
0.1422 - accuracy: 0.9574 - val_loss: 0.1480 - val_accuracy: 0.9542
Epoch 19/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.1398 - accuracy: 0.9583 - val_loss: 0.1489 - val_accuracy: 0.9532
Epoch 20/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.1370 - accuracy: 0.9588 - val_loss: 0.1509 - val_accuracy: 0.9532
Epoch 21/100
```

```
1688/1688 [==============================] - 4s 2ms/step - loss:
0.1348 - accuracy: 0.9598 - val_loss: 0.1481 - val_accuracy: 0.9528
313/313 [==============================] - 1s 3ms/step - loss: 0.1788
- accuracy: 0.9476
test_loss = 0.17877863347530365 test_acc = 0.9476000070571899
```



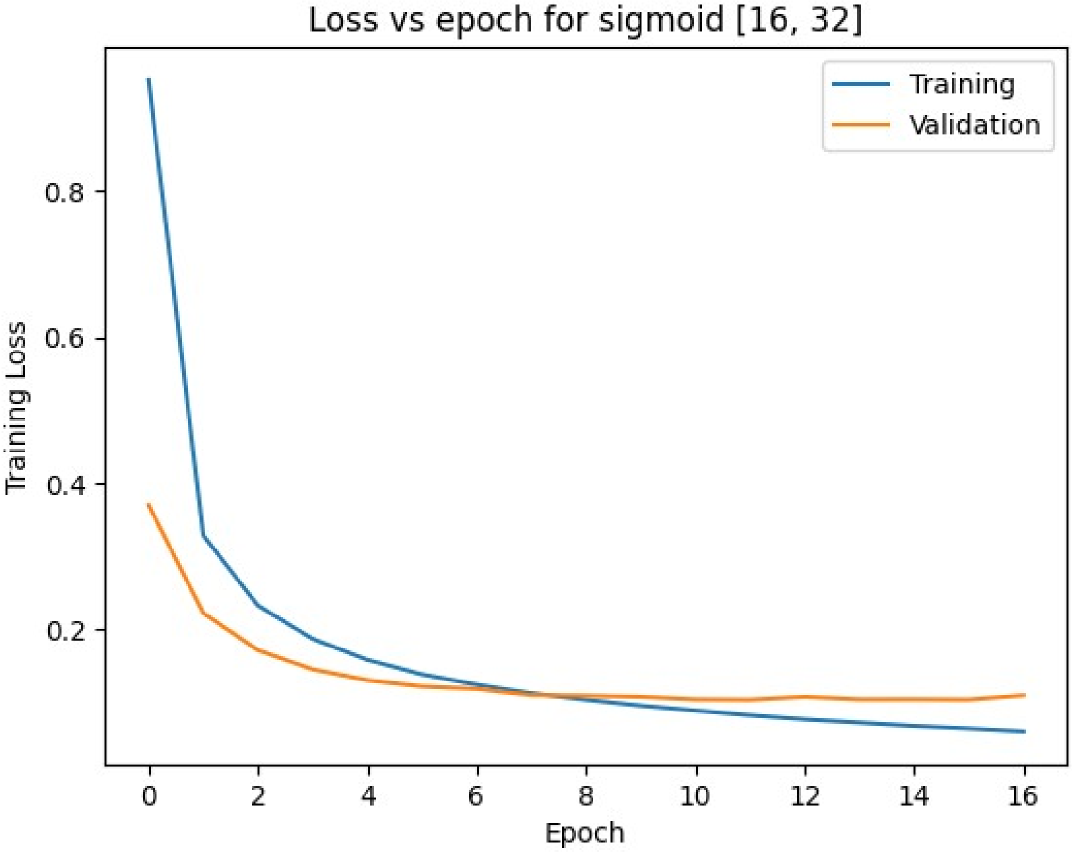


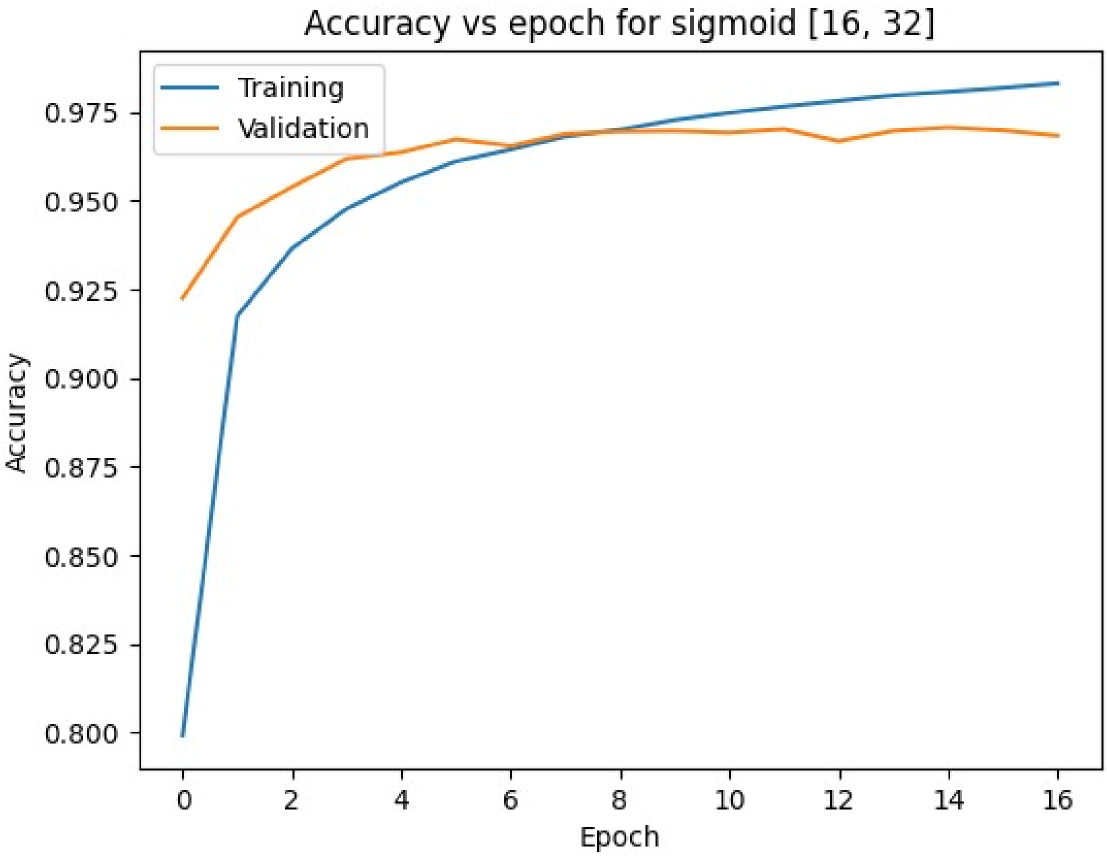
```
activation_function = 'sigmoid'
hidden_neurons = [16, 32]

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: 25818 (100.85 KB)
Trainable params: 25818 (100.85 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.9516 - accuracy: 0.7991 - val_loss: 0.3704 - val_accuracy: 0.9225
Epoch 2/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.3290 - accuracy: 0.9174 - val_loss: 0.2228 - val_accuracy: 0.9453
Epoch 3/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.2330 - accuracy: 0.9365 - val_loss: 0.1721 - val_accuracy: 0.9538
Epoch 4/100
1688/1688 [==============================] - 4s 2ms/step - loss:
0.1876 - accuracy: 0.9477 - val_loss: 0.1459 - val_accuracy: 0.9618
Epoch 5/100
1688/1688 [==============================] - 4s 2ms/step - loss:
0.1590 - accuracy: 0.9552 - val_loss: 0.1311 - val_accuracy: 0.9637
Epoch 6/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.1390 - accuracy: 0.9611 - val_loss: 0.1227 - val_accuracy: 0.9673
Epoch 7/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.1250 - accuracy: 0.9644 - val_loss: 0.1195 - val_accuracy: 0.9655
Epoch 8/100
1688/1688 [==============================] - 4s 3ms/step - loss:
0.1133 - accuracy: 0.9681 - val_loss: 0.1107 - val_accuracy: 0.9688
Epoch 9/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.1043 - accuracy: 0.9700 - val_loss: 0.1099 - val_accuracy: 0.9695
Epoch 10/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.0962 - accuracy: 0.9727 - val_loss: 0.1084 - val_accuracy: 0.9697
Epoch 11/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.0896 - accuracy: 0.9748 - val_loss: 0.1047 - val_accuracy: 0.9692
Epoch 12/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.0831 - accuracy: 0.9766 - val_loss: 0.1042 - val_accuracy: 0.9702
Epoch 13/100
1688/1688 [==============================] - 6s 3ms/step - loss:
```

```
0.0776 - accuracy: 0.9781 - val_loss: 0.1084 - val_accuracy: 0.9668
Epoch 14/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.0731 - accuracy: 0.9797 - val_loss: 0.1047 - val_accuracy: 0.9697
Epoch 15/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.0685 - accuracy: 0.9807 - val_loss: 0.1048 - val_accuracy: 0.9707
Epoch 16/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.0650 - accuracy: 0.9818 - val_loss: 0.1044 - val_accuracy: 0.9698
Epoch 17/100
1688/1688 [==============================] - 4s 2ms/step - loss:
0.0609 - accuracy: 0.9831 - val_loss: 0.1103 - val_accuracy: 0.9683
313/313 [==============================] - 1s 2ms/step - loss: 0.1252
- accuracy: 0.9631
test_loss = 0.12516066431999207 test_acc = 0.963100016117096
```



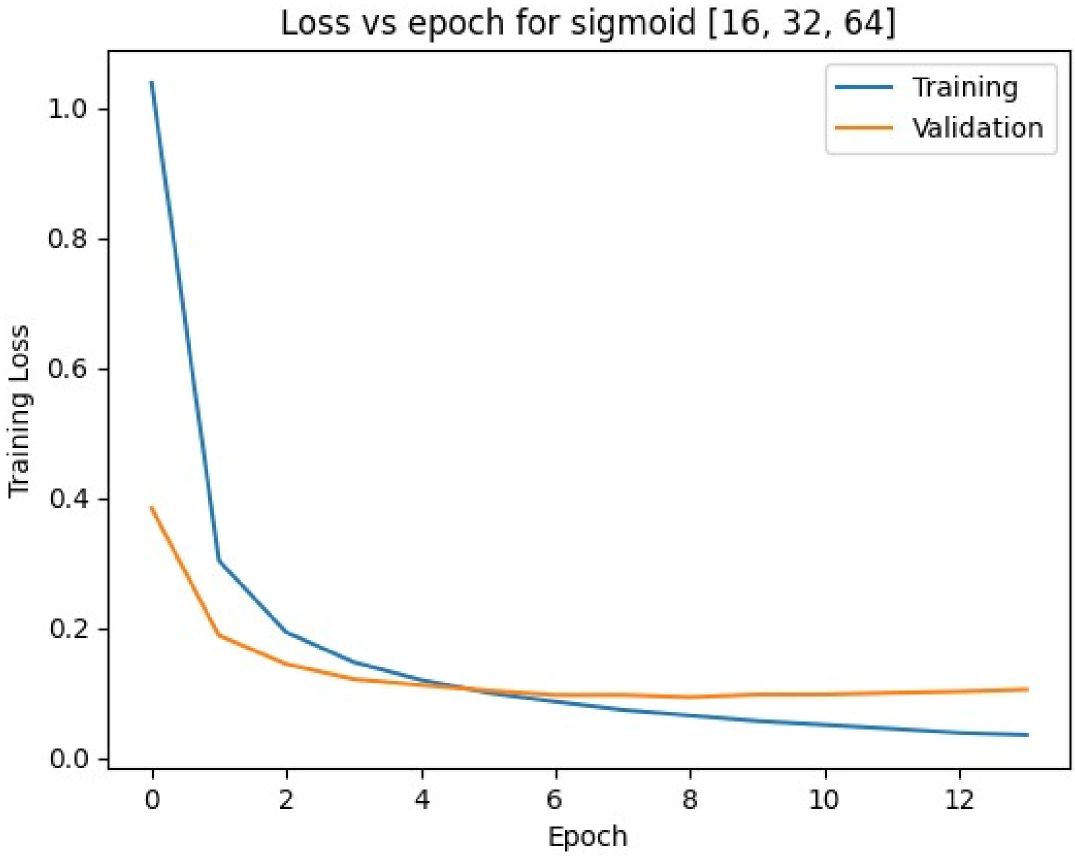


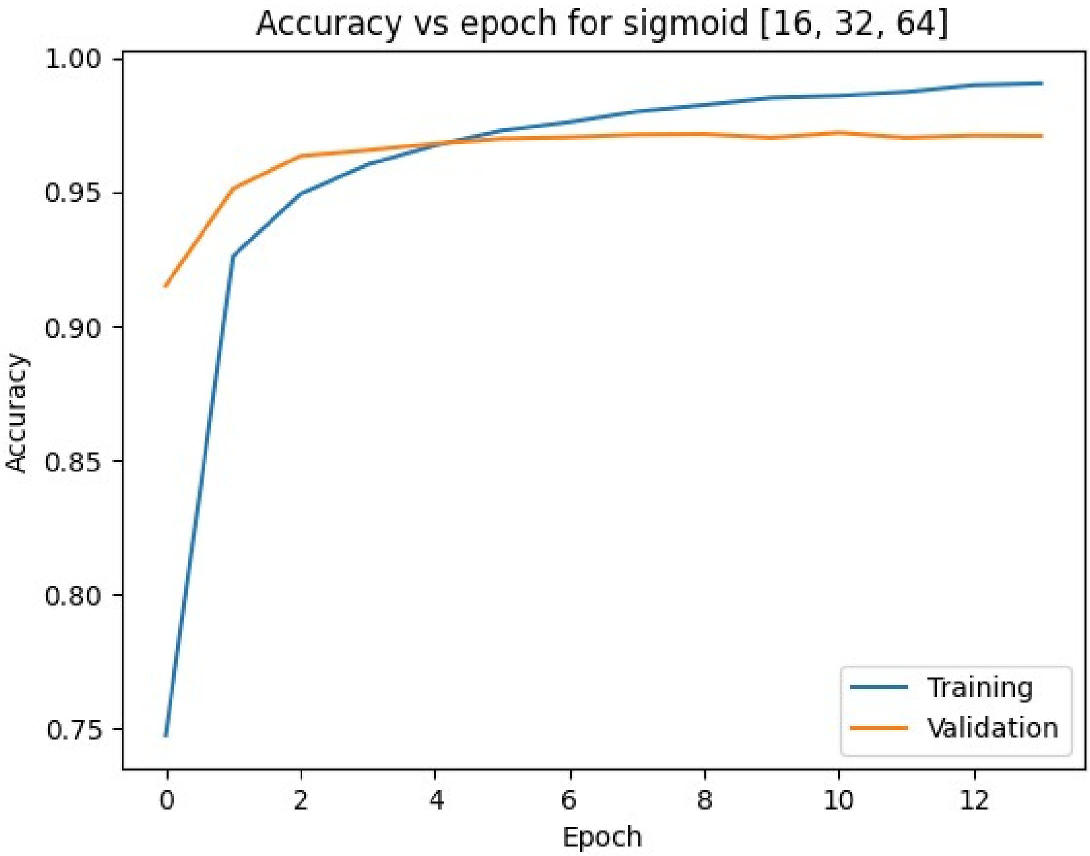
```
hidden_neurons = [16, 32, 64]
activation_function = 'sigmoid'

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: 53018 (207.10 KB)
Trainable params: 53018 (207.10 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 6s 3ms/step - loss:
1.0378 - accuracy: 0.7471 - val_loss: 0.3837 - val_accuracy: 0.9150
Epoch 2/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.3032 - accuracy: 0.9260 - val_loss: 0.1880 - val_accuracy: 0.9512
Epoch 3/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.1930 - accuracy: 0.9491 - val_loss: 0.1441 - val_accuracy: 0.9633
Epoch 4/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.1473 - accuracy: 0.9602 - val_loss: 0.1211 - val_accuracy: 0.9655
Epoch 5/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.1197 - accuracy: 0.9673 - val_loss: 0.1118 - val_accuracy: 0.9680
Epoch 6/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.1000 - accuracy: 0.9728 - val_loss: 0.1033 - val_accuracy: 0.9698
Epoch 7/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.0862 - accuracy: 0.9760 - val_loss: 0.0965 - val_accuracy: 0.9703
Epoch 8/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.0736 - accuracy: 0.9799 - val_loss: 0.0967 - val_accuracy: 0.9713
Epoch 9/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.0649 - accuracy: 0.9824 - val_loss: 0.0933 - val_accuracy: 0.9715
Epoch 10/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.0564 - accuracy: 0.9850 - val_loss: 0.0973 - val_accuracy: 0.9702
Epoch 11/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.0507 - accuracy: 0.9858 - val_loss: 0.0974 - val_accuracy: 0.9720
Epoch 12/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.0448 - accuracy: 0.9872 - val_loss: 0.0999 - val_accuracy: 0.9702
```

```
Epoch 13/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.0383 - accuracy: 0.9898 - val_loss: 0.1019 - val_accuracy: 0.9710
Epoch 14/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.0350 - accuracy: 0.9904 - val_loss: 0.1048 - val_accuracy: 0.9708
313/313 [==============================] - 1s 2ms/step - loss: 0.1146
- accuracy: 0.9675
test_loss = 0.11459528654813766 test_acc = 0.9674999713897705
```





```
result

   Hidden Layers Activation Function Hidden Neurons  Test Loss  Test
Acccuracy
0              1             sigmoid           [16]   0.178779
0.9476
1              2             sigmoid       [16, 32]   0.125161
0.9631
2              3             sigmoid   [16, 32, 64]   0.114595
0.9675
```

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

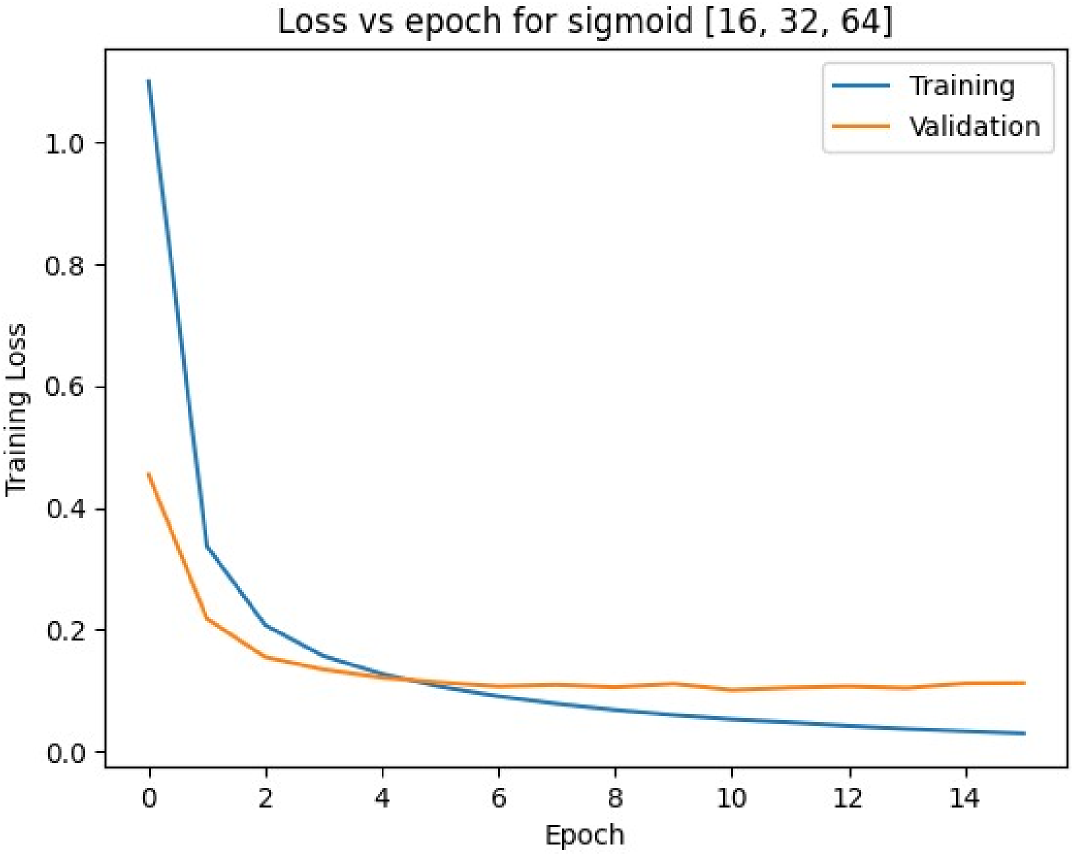
Hidden Layers	Activation Function	Hidden Neurons
3	Sigmoid	[16,32,64]
3	Tanh	[16,32,64]
3	Relu	[16,32,64]

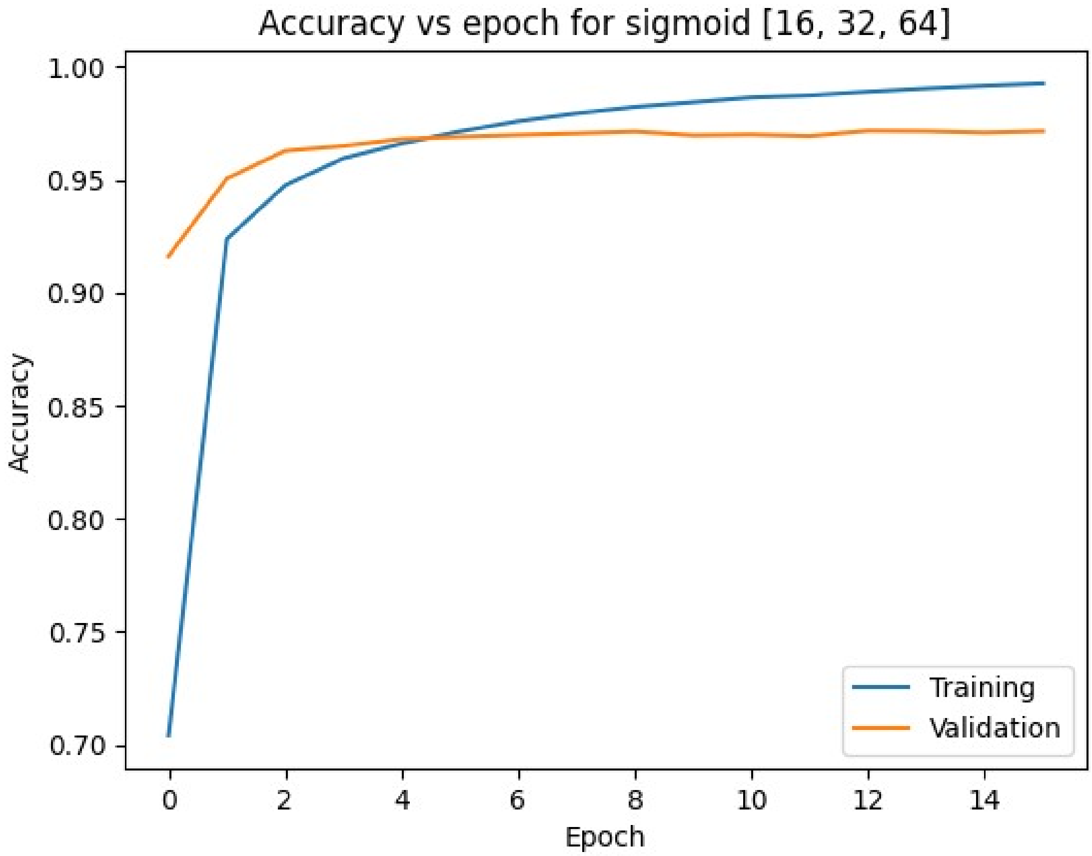
```
result = pd.DataFrame(
    columns=[
        'Hidden Layers',
        'Activation Function',
```

```python
        '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: 53018 (207.10 KB)
Trainable params: 53018 (207.10 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 8s 5ms/step - loss:
1.0987 - accuracy: 0.7039 - val_loss: 0.4540 - val_accuracy: 0.9162
Epoch 2/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.3369 - accuracy: 0.9237 - val_loss: 0.2177 - val_accuracy: 0.9505
Epoch 3/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.2072 - accuracy: 0.9475 - val_loss: 0.1548 - val_accuracy: 0.9628
```

```
Epoch 4/100
1688/1688 [==============================] - 8s 4ms/step - loss:
0.1565 - accuracy: 0.9595 - val_loss: 0.1345 - val_accuracy: 0.9650
Epoch 5/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.1273 - accuracy: 0.9661 - val_loss: 0.1214 - val_accuracy: 0.9680
Epoch 6/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.1065 - accuracy: 0.9714 - val_loss: 0.1135 - val_accuracy: 0.9690
Epoch 7/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.0906 - accuracy: 0.9759 - val_loss: 0.1067 - val_accuracy: 0.9698
Epoch 8/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.0784 - accuracy: 0.9794 - val_loss: 0.1094 - val_accuracy: 0.9705
Epoch 9/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.0682 - accuracy: 0.9822 - val_loss: 0.1053 - val_accuracy: 0.9713
Epoch 10/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.0598 - accuracy: 0.9843 - val_loss: 0.1111 - val_accuracy: 0.9697
Epoch 11/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.0527 - accuracy: 0.9864 - val_loss: 0.1008 - val_accuracy: 0.9700
Epoch 12/100
1688/1688 [==============================] - 9s 5ms/step - loss:
0.0478 - accuracy: 0.9873 - val_loss: 0.1046 - val_accuracy: 0.9693
Epoch 13/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.0422 - accuracy: 0.9890 - val_loss: 0.1068 - val_accuracy: 0.9717
Epoch 14/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.0371 - accuracy: 0.9904 - val_loss: 0.1040 - val_accuracy: 0.9715
Epoch 15/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.0333 - accuracy: 0.9916 - val_loss: 0.1118 - val_accuracy: 0.9708
Epoch 16/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.0296 - accuracy: 0.9926 - val_loss: 0.1124 - val_accuracy: 0.9715
313/313 [==============================] - 1s 2ms/step - loss: 0.1280
- accuracy: 0.9649
test_loss = 0.1279534250497818 test_acc = 0.964900016784668
```

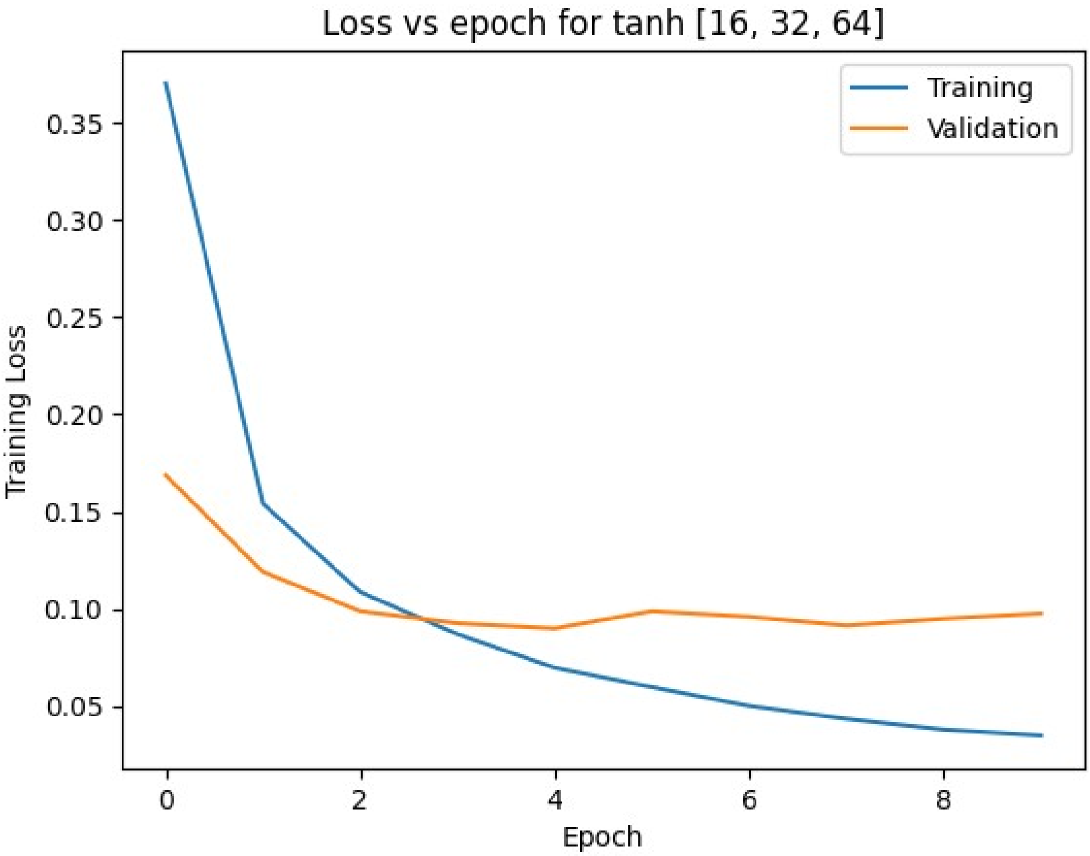

Loss vs epoch for sigmoid [16, 32, 64]
Training
Validation
Training Loss
1.0
0.8
0.6
0.4
0.2
0.0
Epoch
0
2
4
6
8
10
12
14

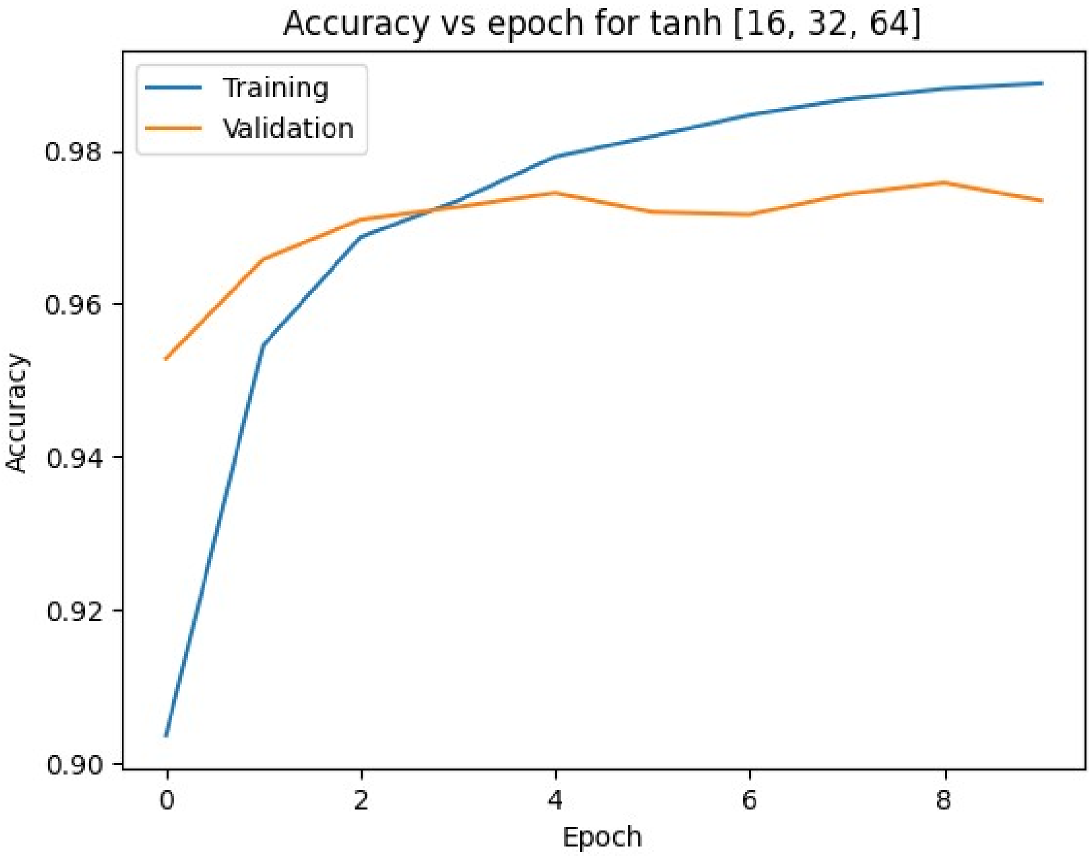


```
hidden_neurons = [16, 32, 64]
activation_function = 'tanh'

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: 53018 (207.10 KB)
Trainable params: 53018 (207.10 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 9s 5ms/step - loss:
0.3701 - accuracy: 0.9036 - val_loss: 0.1686 - val_accuracy: 0.9528
Epoch 2/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.1541 - accuracy: 0.9546 - val_loss: 0.1189 - val_accuracy: 0.9658
Epoch 3/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.1086 - accuracy: 0.9687 - val_loss: 0.0988 - val_accuracy: 0.9710
Epoch 4/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.0869 - accuracy: 0.9735 - val_loss: 0.0925 - val_accuracy: 0.9727
Epoch 5/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.0697 - accuracy: 0.9792 - val_loss: 0.0899 - val_accuracy: 0.9745
Epoch 6/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.0597 - accuracy: 0.9819 - val_loss: 0.0987 - val_accuracy: 0.9720
Epoch 7/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.0500 - accuracy: 0.9847 - val_loss: 0.0959 - val_accuracy: 0.9717
Epoch 8/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.0434 - accuracy: 0.9867 - val_loss: 0.0914 - val_accuracy: 0.9743
Epoch 9/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.0379 - accuracy: 0.9881 - val_loss: 0.0949 - val_accuracy: 0.9758
Epoch 10/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.0348 - accuracy: 0.9888 - val_loss: 0.0975 - val_accuracy: 0.9735
313/313 [==============================] - 1s 3ms/step - loss: 0.0909
- accuracy: 0.9736
test_loss = 0.09091850370168686 test_acc = 0.9735999703407288
```


Loss vs epoch for tanh [16, 32, 64]
Training
Validation
0.35
0.30
0.25
0.20
0.15
0.10
0.05
Training Loss
0
2
4
6
8
Epoch

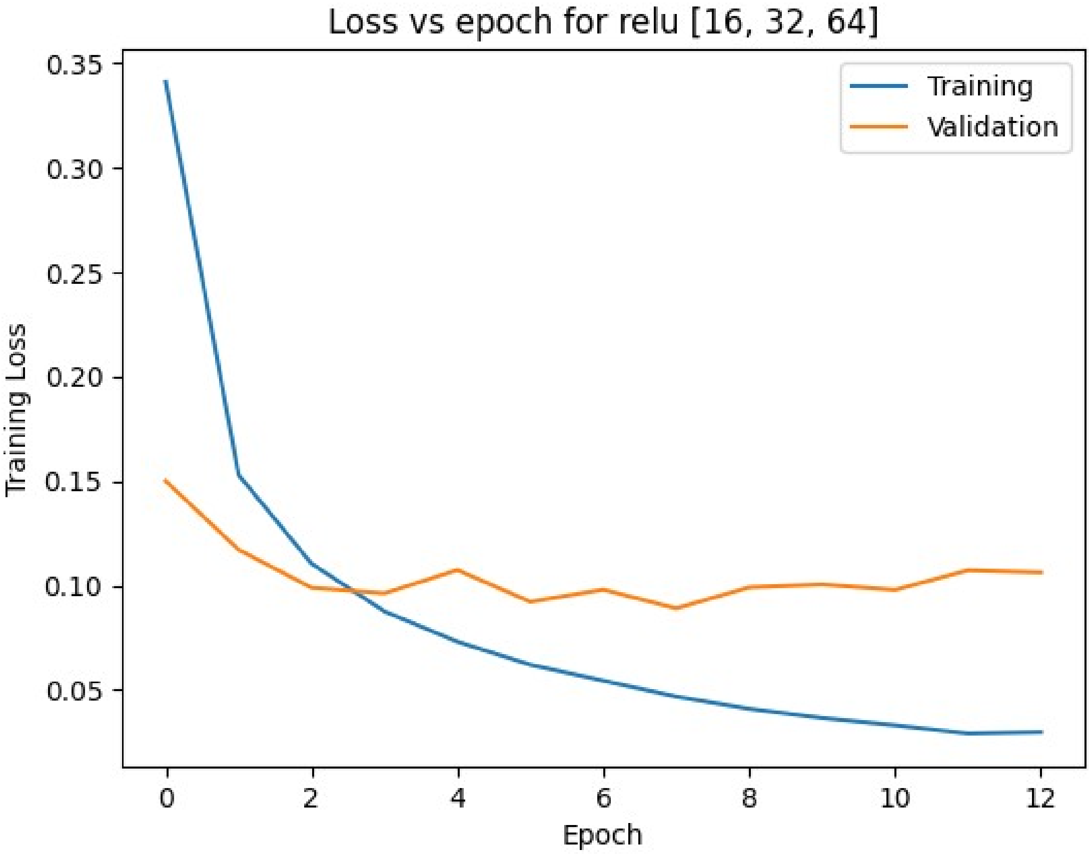


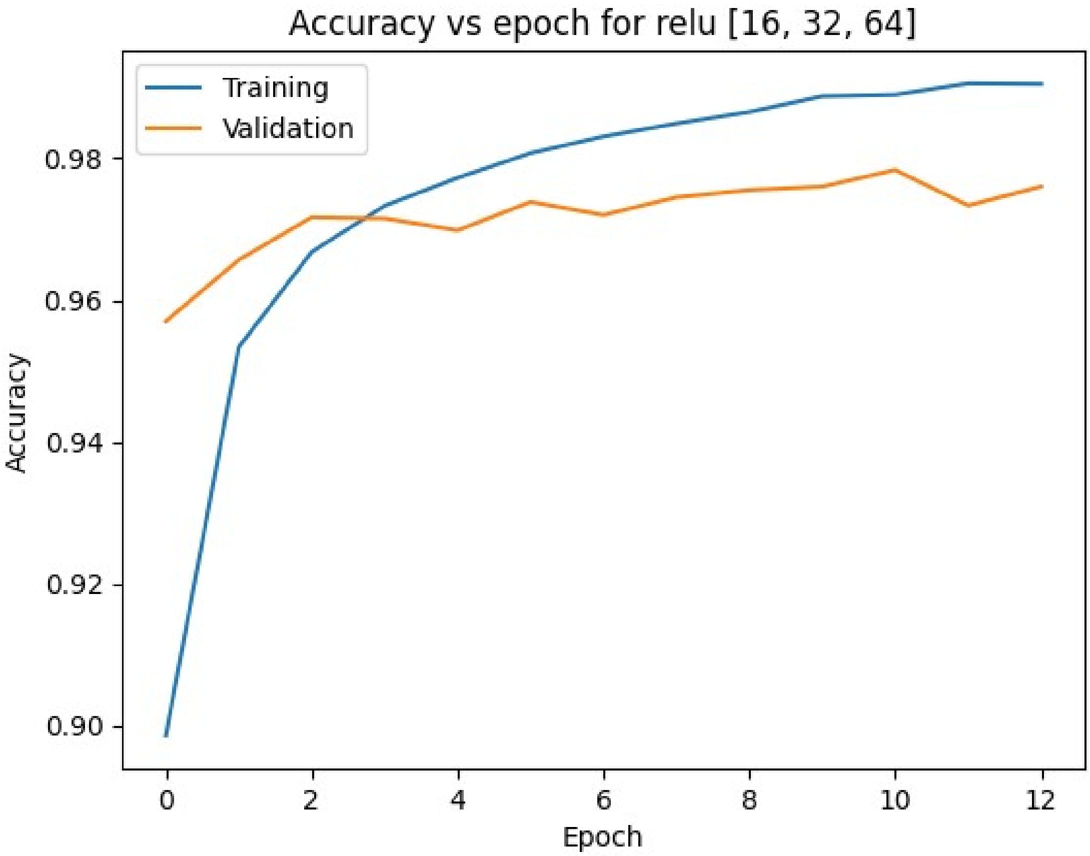
```
hidden_neurons = [16, 32, 64]
activation_function = 'relu'

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: 53018 (207.10 KB)
Trainable params: 53018 (207.10 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 8s 4ms/step - loss:
0.3409 - accuracy: 0.8986 - val_loss: 0.1500 - val_accuracy: 0.9570
Epoch 2/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.1529 - accuracy: 0.9534 - val_loss: 0.1171 - val_accuracy: 0.9657
Epoch 3/100
1688/1688 [==============================] - 8s 4ms/step - loss:
0.1104 - accuracy: 0.9668 - val_loss: 0.0989 - val_accuracy: 0.9717
Epoch 4/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.0876 - accuracy: 0.9733 - val_loss: 0.0962 - val_accuracy: 0.9715
Epoch 5/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.0731 - accuracy: 0.9772 - val_loss: 0.1074 - val_accuracy: 0.9698
Epoch 6/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.0621 - accuracy: 0.9807 - val_loss: 0.0922 - val_accuracy: 0.9738
Epoch 7/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.0544 - accuracy: 0.9831 - val_loss: 0.0980 - val_accuracy: 0.9720
Epoch 8/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.0468 - accuracy: 0.9849 - val_loss: 0.0890 - val_accuracy: 0.9745
Epoch 9/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.0410 - accuracy: 0.9865 - val_loss: 0.0991 - val_accuracy: 0.9755
Epoch 10/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.0366 - accuracy: 0.9887 - val_loss: 0.1005 - val_accuracy: 0.9760
Epoch 11/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.0331 - accuracy: 0.9890 - val_loss: 0.0979 - val_accuracy: 0.9783
Epoch 12/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.0291 - accuracy: 0.9906 - val_loss: 0.1072 - val_accuracy: 0.9733
```

```
Epoch 13/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.0298 - accuracy: 0.9905 - val_loss: 0.1062 - val_accuracy: 0.9760
313/313 [==============================] - 1s 3ms/step - loss: 0.0983
- accuracy: 0.9710
test_loss = 0.09829609096050262 test_acc = 0.9710000157356262
```





```
result

   Hidden Layers Activation Function Hidden Neurons  Test Loss  Test
Acccuracy
0              3             sigmoid   [16, 32, 64]   0.127953
0.9649
1              3                tanh   [16, 32, 64]   0.090919
0.9736
2              3                relu   [16, 32, 64]   0.098296
0.9710

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

best_activation_fn

{"type":"string"}
```

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

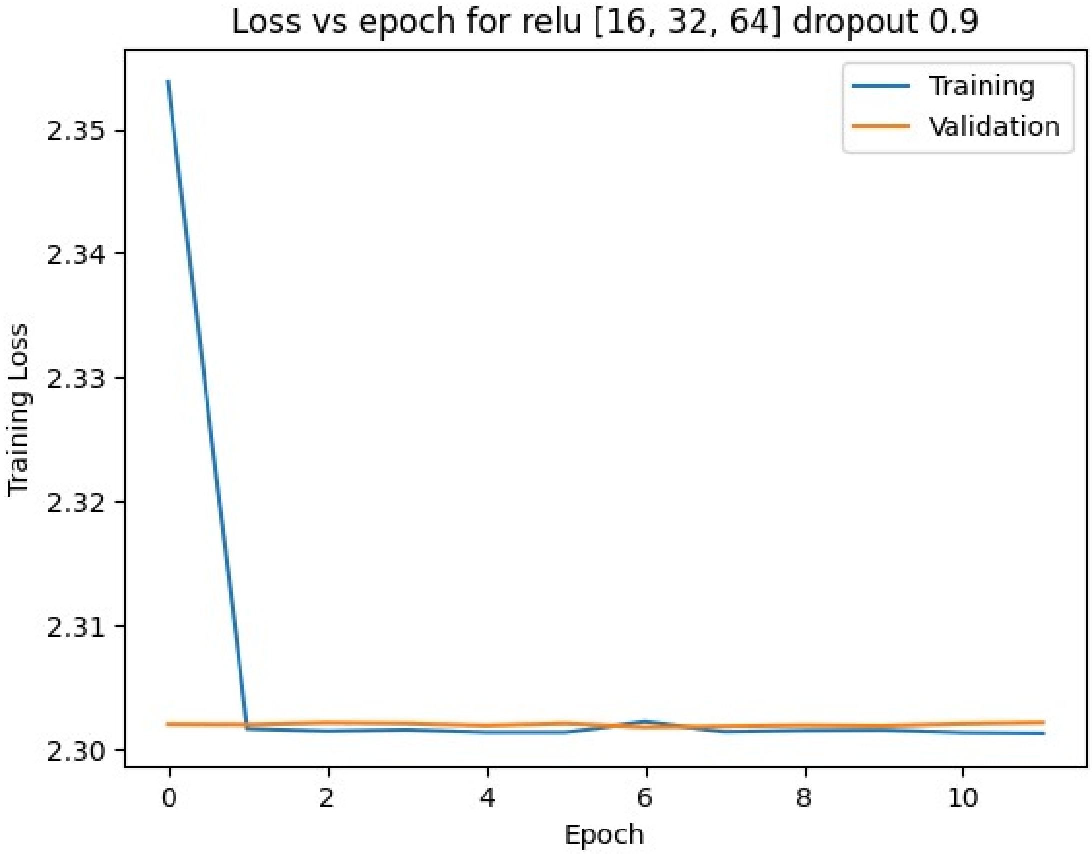
Hidden Layers	Activation Function	Hidden Neurons	Dropout
3	Relu	[16,32,64]	0.9
3	Relu	[16,32,64]	0.75
3	Relu	[16,32,64]	0.5
3	Relu	[16,32,64]	0.25
3	Relu	[16,32,64]	0.10

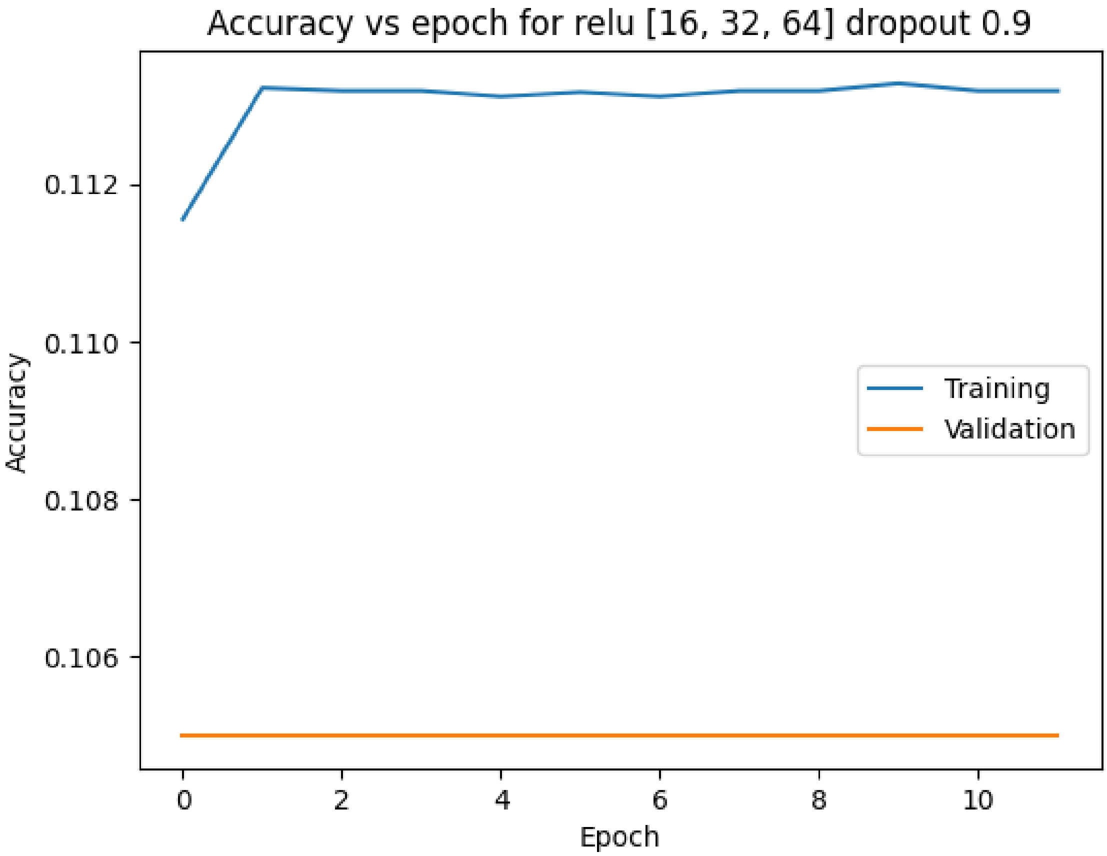
```
result = pd.DataFrame(
    columns=[
        'Hidden Layers',
        'Activation Function',
        'Hidden Neurons',
        'Dropout',
        'Test Loss',
        'Test Acccuracy'],
)

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: 53018 (207.10 KB)
Trainable params: 53018 (207.10 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 7s 4ms/step - loss:
2.3539 - accuracy: 0.1116 - val_loss: 2.3020 - val_accuracy: 0.1050
Epoch 2/100
1688/1688 [==============================] - 9s 5ms/step - loss:
2.3017 - accuracy: 0.1132 - val_loss: 2.3020 - val_accuracy: 0.1050
Epoch 3/100
1688/1688 [==============================] - 7s 4ms/step - loss:
2.3015 - accuracy: 0.1132 - val_loss: 2.3021 - val_accuracy: 0.1050
Epoch 4/100
1688/1688 [==============================] - 10s 6ms/step - loss:
2.3016 - accuracy: 0.1132 - val_loss: 2.3021 - val_accuracy: 0.1050
Epoch 5/100
1688/1688 [==============================] - 7s 4ms/step - loss:
2.3014 - accuracy: 0.1131 - val_loss: 2.3019 - val_accuracy: 0.1050
Epoch 6/100
1688/1688 [==============================] - 7s 4ms/step - loss:
2.3013 - accuracy: 0.1132 - val_loss: 2.3021 - val_accuracy: 0.1050
Epoch 7/100
1688/1688 [==============================] - 8s 4ms/step - loss:
2.3022 - accuracy: 0.1131 - val_loss: 2.3018 - val_accuracy: 0.1050
Epoch 8/100
1688/1688 [==============================] - 6s 3ms/step - loss:
2.3014 - accuracy: 0.1132 - val_loss: 2.3019 - val_accuracy: 0.1050
Epoch 9/100
1688/1688 [==============================] - 8s 5ms/step - loss:
2.3015 - accuracy: 0.1132 - val_loss: 2.3019 - val_accuracy: 0.1050
Epoch 10/100
1688/1688 [==============================] - 7s 4ms/step - loss:
2.3015 - accuracy: 0.1133 - val_loss: 2.3019 - val_accuracy: 0.1050
Epoch 11/100
1688/1688 [==============================] - 7s 4ms/step - loss:
2.3013 - accuracy: 0.1132 - val_loss: 2.3021 - val_accuracy: 0.1050
Epoch 12/100
1688/1688 [==============================] - 7s 4ms/step - loss:
```

```
2.3013 - accuracy: 0.1132 - val_loss: 2.3022 - val_accuracy: 0.1050
313/313 [==============================] - 1s 3ms/step - loss: 2.3012
- accuracy: 0.1135
test_loss = 2.301210641860962 test_acc = 0.11349999904632568
```





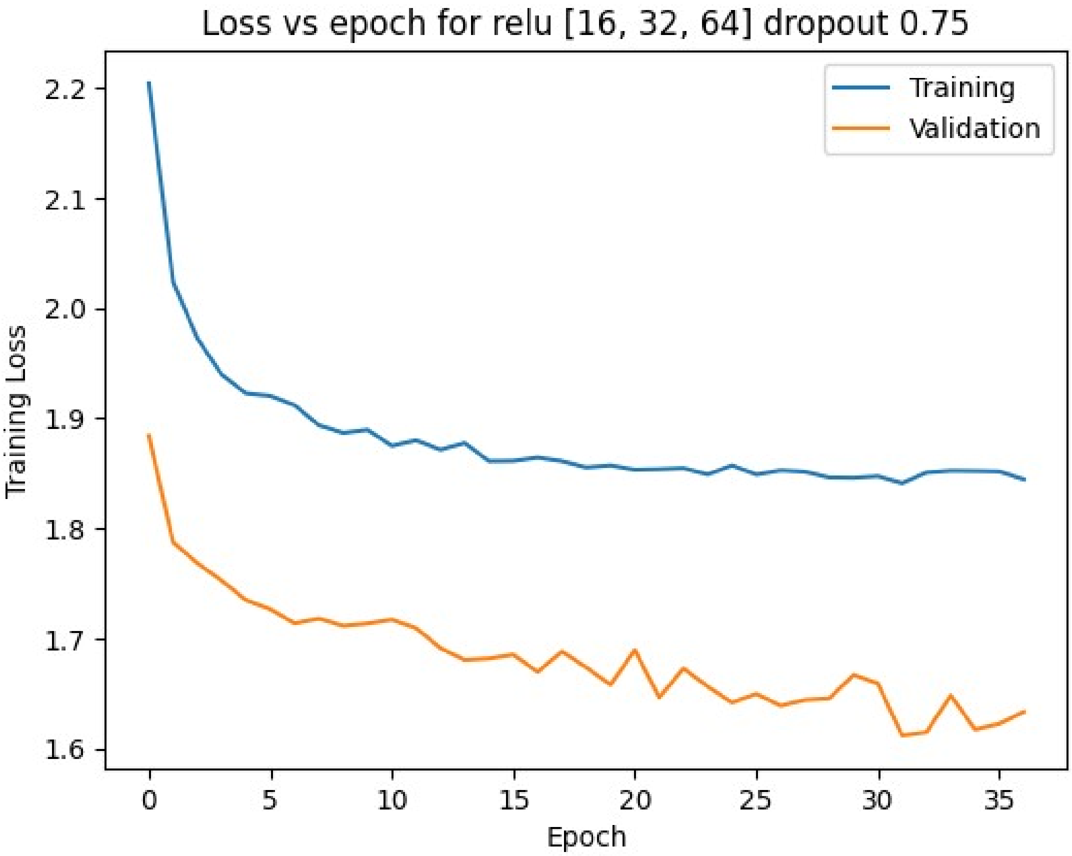
```
hidden_neurons = [16, 32, 64]
activation_function = 'relu'
dropout_val = 0.75

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)
```

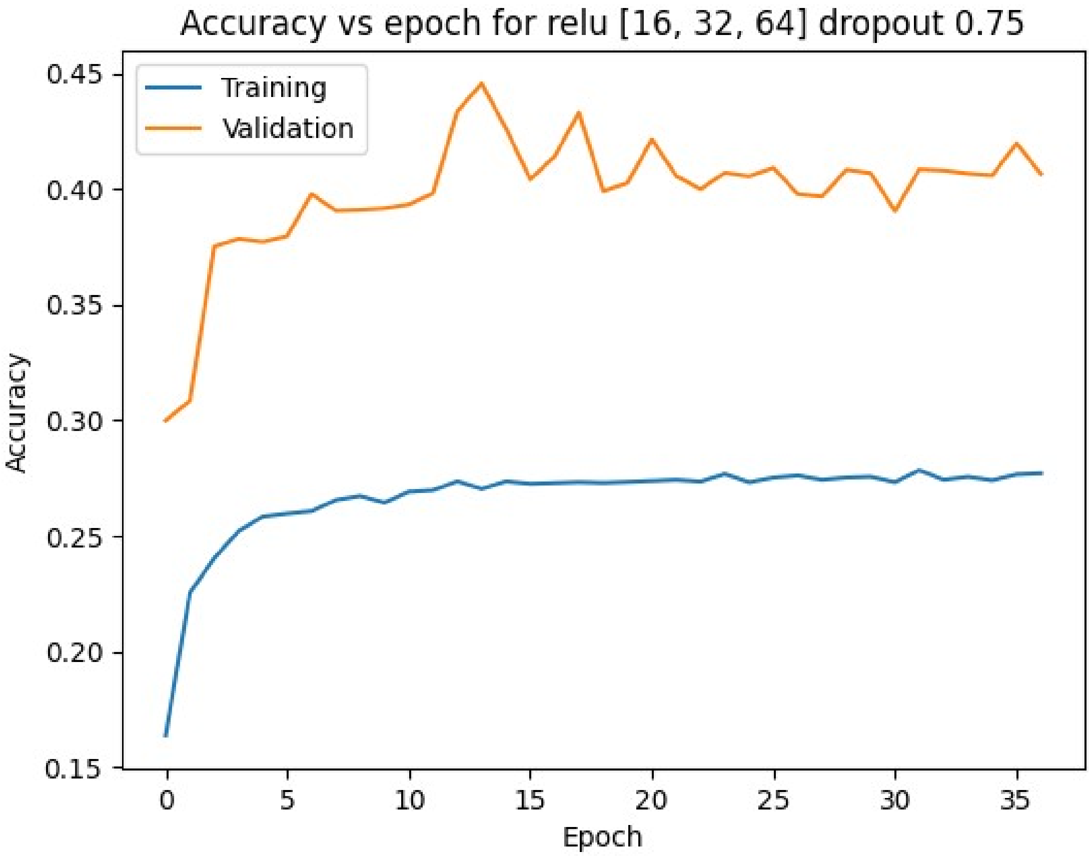
```
Model: "sequential_7"
```

```
 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: 53018 (207.10 KB)
Trainable params: 53018 (207.10 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 8s 4ms/step - loss:
2.2042 - accuracy: 0.1636 - val_loss: 1.8841 - val_accuracy: 0.2998
Epoch 2/100
1688/1688 [==============================] - 9s 5ms/step - loss:
2.0241 - accuracy: 0.2253 - val_loss: 1.7874 - val_accuracy: 0.3083
Epoch 3/100
1688/1688 [==============================] - 6s 4ms/step - loss:
1.9724 - accuracy: 0.2403 - val_loss: 1.7683 - val_accuracy: 0.3750
Epoch 4/100
1688/1688 [==============================] - 8s 5ms/step - loss:
1.9396 - accuracy: 0.2520 - val_loss: 1.7526 - val_accuracy: 0.3783
Epoch 5/100
1688/1688 [==============================] - 8s 5ms/step - loss:
1.9226 - accuracy: 0.2583 - val_loss: 1.7348 - val_accuracy: 0.3770
Epoch 6/100
1688/1688 [==============================] - 6s 4ms/step - loss:
1.9203 - accuracy: 0.2596 - val_loss: 1.7266 - val_accuracy: 0.3795
Epoch 7/100
1688/1688 [==============================] - 8s 5ms/step - loss:
1.9118 - accuracy: 0.2606 - val_loss: 1.7138 - val_accuracy: 0.3977
Epoch 8/100
1688/1688 [==============================] - 6s 4ms/step - loss:
1.8937 - accuracy: 0.2654 - val_loss: 1.7181 - val_accuracy: 0.3905
Epoch 9/100
1688/1688 [==============================] - 9s 5ms/step - loss:
```

```
1.8867 - accuracy: 0.2671 - val_loss: 1.7115 - val_accuracy: 0.3908
Epoch 10/100
1688/1688 [==============================] - 7s 4ms/step - loss:
1.8895 - accuracy: 0.2643 - val_loss: 1.7136 - val_accuracy: 0.3915
Epoch 11/100
1688/1688 [==============================] - 8s 5ms/step - loss:
1.8750 - accuracy: 0.2690 - val_loss: 1.7171 - val_accuracy: 0.3932
Epoch 12/100
1688/1688 [==============================] - 7s 4ms/step - loss:
1.8799 - accuracy: 0.2696 - val_loss: 1.7094 - val_accuracy: 0.3980
Epoch 13/100
1688/1688 [==============================] - 7s 4ms/step - loss:
1.8715 - accuracy: 0.2735 - val_loss: 1.6912 - val_accuracy: 0.4335
Epoch 14/100
1688/1688 [==============================] - 8s 5ms/step - loss:
1.8775 - accuracy: 0.2702 - val_loss: 1.6805 - val_accuracy: 0.4455
Epoch 15/100
1688/1688 [==============================] - 6s 4ms/step - loss:
1.8611 - accuracy: 0.2735 - val_loss: 1.6818 - val_accuracy: 0.4260
Epoch 16/100
1688/1688 [==============================] - 9s 5ms/step - loss:
1.8613 - accuracy: 0.2724 - val_loss: 1.6854 - val_accuracy: 0.4040
Epoch 17/100
1688/1688 [==============================] - 6s 4ms/step - loss:
1.8643 - accuracy: 0.2728 - val_loss: 1.6696 - val_accuracy: 0.4140
Epoch 18/100
1688/1688 [==============================] - 8s 5ms/step - loss:
1.8611 - accuracy: 0.2731 - val_loss: 1.6883 - val_accuracy: 0.4330
Epoch 19/100
1688/1688 [==============================] - 7s 4ms/step - loss:
1.8554 - accuracy: 0.2729 - val_loss: 1.6737 - val_accuracy: 0.3990
Epoch 20/100
1688/1688 [==============================] - 7s 4ms/step - loss:
1.8572 - accuracy: 0.2732 - val_loss: 1.6579 - val_accuracy: 0.4025
Epoch 21/100
1688/1688 [==============================] - 8s 5ms/step - loss:
1.8532 - accuracy: 0.2737 - val_loss: 1.6895 - val_accuracy: 0.4213
Epoch 22/100
1688/1688 [==============================] - 6s 3ms/step - loss:
1.8537 - accuracy: 0.2741 - val_loss: 1.6464 - val_accuracy: 0.4055
Epoch 23/100
1688/1688 [==============================] - 9s 5ms/step - loss:
1.8546 - accuracy: 0.2734 - val_loss: 1.6727 - val_accuracy: 0.3998
Epoch 24/100
1688/1688 [==============================] - 7s 4ms/step - loss:
1.8490 - accuracy: 0.2767 - val_loss: 1.6564 - val_accuracy: 0.4068
Epoch 25/100
1688/1688 [==============================] - 8s 5ms/step - loss:
1.8571 - accuracy: 0.2731 - val_loss: 1.6418 - val_accuracy: 0.4053
```

```
Epoch 26/100
1688/1688 [==============================] - 8s 5ms/step - loss:
1.8489 - accuracy: 0.2751 - val_loss: 1.6497 - val_accuracy: 0.4090
Epoch 27/100
1688/1688 [==============================] - 7s 4ms/step - loss:
1.8527 - accuracy: 0.2760 - val_loss: 1.6390 - val_accuracy: 0.3977
Epoch 28/100
1688/1688 [==============================] - 8s 5ms/step - loss:
1.8515 - accuracy: 0.2741 - val_loss: 1.6442 - val_accuracy: 0.3967
Epoch 29/100
1688/1688 [==============================] - 8s 5ms/step - loss:
1.8462 - accuracy: 0.2751 - val_loss: 1.6455 - val_accuracy: 0.4082
Epoch 30/100
1688/1688 [==============================] - 9s 5ms/step - loss:
1.8459 - accuracy: 0.2755 - val_loss: 1.6669 - val_accuracy: 0.4067
Epoch 31/100
1688/1688 [==============================] - 6s 4ms/step - loss:
1.8474 - accuracy: 0.2731 - val_loss: 1.6592 - val_accuracy: 0.3903
Epoch 32/100
1688/1688 [==============================] - 9s 5ms/step - loss:
1.8411 - accuracy: 0.2783 - val_loss: 1.6119 - val_accuracy: 0.4083
Epoch 33/100
1688/1688 [==============================] - 7s 4ms/step - loss:
1.8508 - accuracy: 0.2742 - val_loss: 1.6148 - val_accuracy: 0.4078
Epoch 34/100
1688/1688 [==============================] - 8s 5ms/step - loss:
1.8524 - accuracy: 0.2755 - val_loss: 1.6483 - val_accuracy: 0.4065
Epoch 35/100
1688/1688 [==============================] - 6s 4ms/step - loss:
1.8521 - accuracy: 0.2740 - val_loss: 1.6174 - val_accuracy: 0.4057
Epoch 36/100
1688/1688 [==============================] - 6s 4ms/step - loss:
1.8517 - accuracy: 0.2765 - val_loss: 1.6226 - val_accuracy: 0.4197
Epoch 37/100
1688/1688 [==============================] - 9s 5ms/step - loss:
1.8445 - accuracy: 0.2769 - val_loss: 1.6332 - val_accuracy: 0.4063
313/313 [==============================] - 1s 2ms/step - loss: 1.6019
- accuracy: 0.4008
test_loss = 1.6018527746200562 test_acc = 0.4007999897003174
```


Loss vs epoch for relu [16, 32, 64] dropout 0.75
Training
Validation
Training Loss
2.2
2.1
2.0
1.9
1.8
1.7
1.6
Epoch
0
5
10
15
20
25
30
35

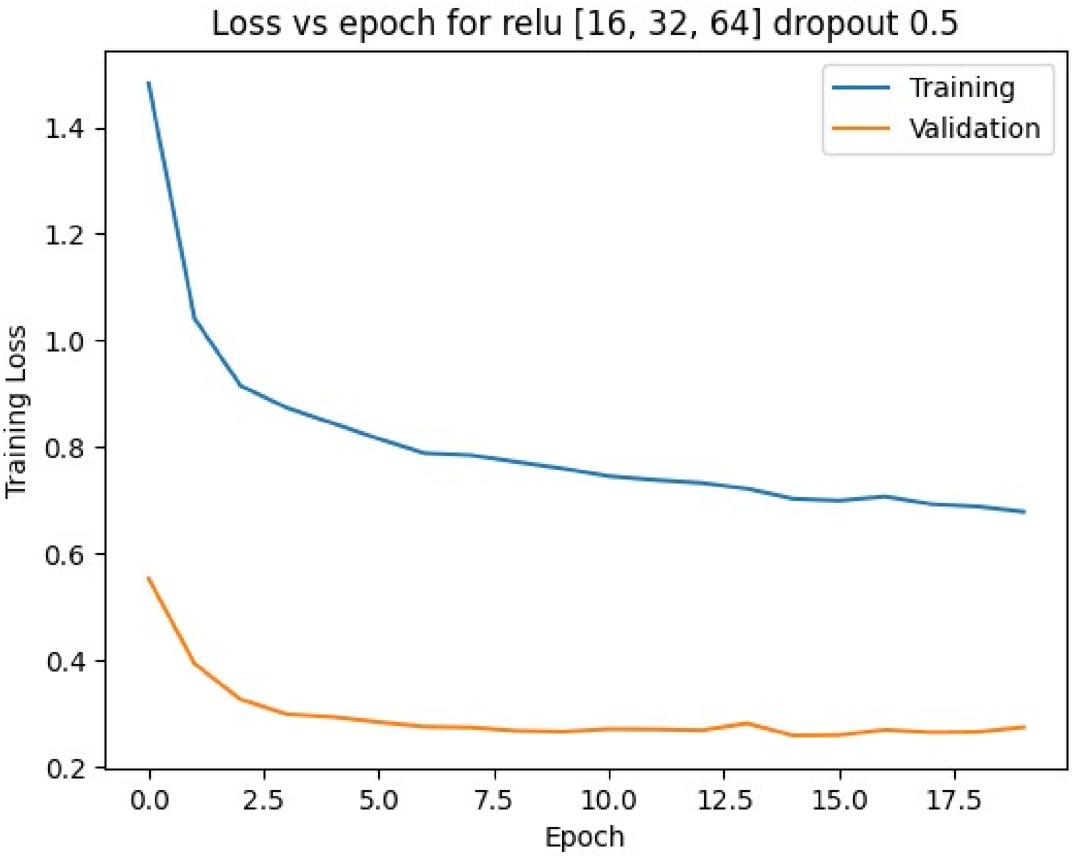


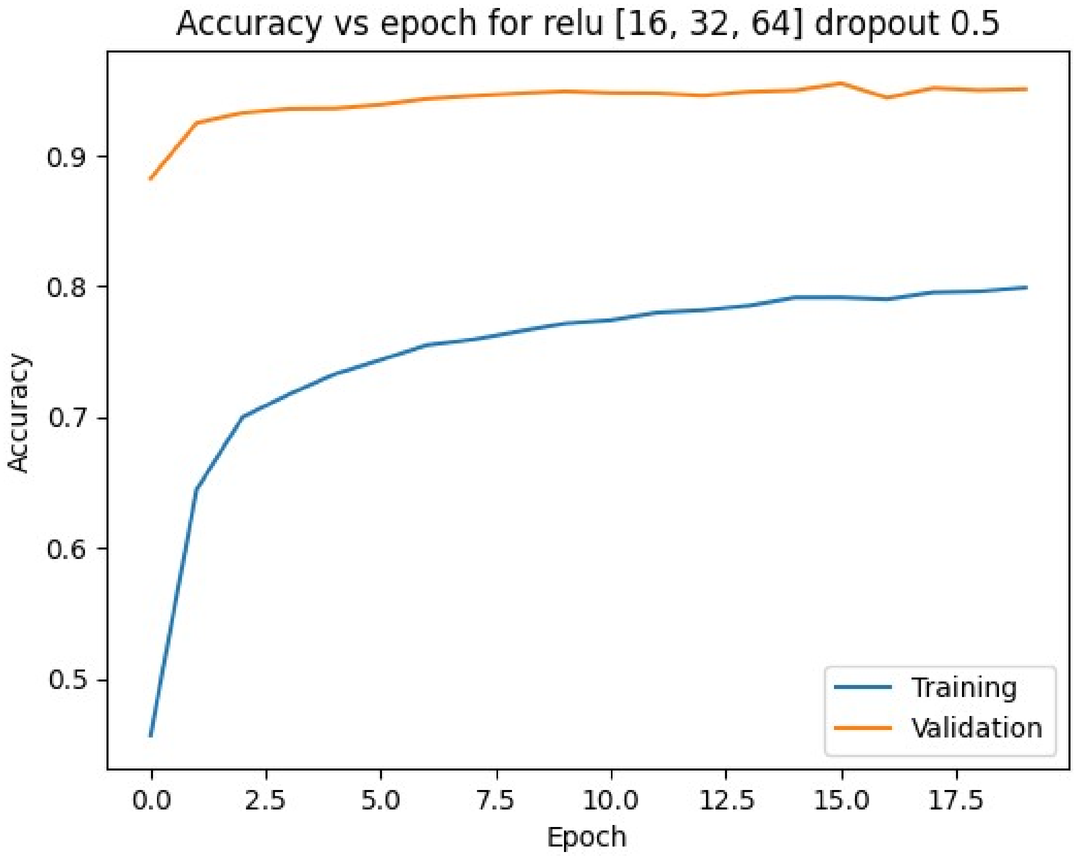
```
hidden_neurons = [16, 32, 64]
activation_function = 'relu'
dropout_val = 0.5

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: 53018 (207.10 KB)
Trainable params: 53018 (207.10 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 9s 5ms/step - loss:
1.4824 - accuracy: 0.4566 - val_loss: 0.5535 - val_accuracy: 0.8820
Epoch 2/100
1688/1688 [==============================] - 7s 4ms/step - loss:
1.0408 - accuracy: 0.6445 - val_loss: 0.3934 - val_accuracy: 0.9245
Epoch 3/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.9151 - accuracy: 0.6999 - val_loss: 0.3265 - val_accuracy: 0.9322
Epoch 4/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.8737 - accuracy: 0.7171 - val_loss: 0.2990 - val_accuracy: 0.9353
Epoch 5/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.8448 - accuracy: 0.7326 - val_loss: 0.2935 - val_accuracy: 0.9357
Epoch 6/100
1688/1688 [==============================] - 9s 5ms/step - loss:
0.8154 - accuracy: 0.7435 - val_loss: 0.2837 - val_accuracy: 0.9385
Epoch 7/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.7886 - accuracy: 0.7547 - val_loss: 0.2755 - val_accuracy: 0.9428
Epoch 8/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.7848 - accuracy: 0.7590 - val_loss: 0.2740 - val_accuracy: 0.9452
Epoch 9/100
1688/1688 [==============================] - 7s 4ms/step - loss:
```

```
0.7720 - accuracy: 0.7653 - val_loss: 0.2676 - val_accuracy: 0.9470
Epoch 10/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.7598 - accuracy: 0.7712 - val_loss: 0.2660 - val_accuracy: 0.9487
Epoch 11/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.7454 - accuracy: 0.7739 - val_loss: 0.2711 - val_accuracy: 0.9473
Epoch 12/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.7384 - accuracy: 0.7795 - val_loss: 0.2705 - val_accuracy: 0.9472
Epoch 13/100
1688/1688 [==============================] - 9s 5ms/step - loss:
0.7329 - accuracy: 0.7815 - val_loss: 0.2687 - val_accuracy: 0.9453
Epoch 14/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.7218 - accuracy: 0.7850 - val_loss: 0.2816 - val_accuracy: 0.9483
Epoch 15/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.7033 - accuracy: 0.7911 - val_loss: 0.2588 - val_accuracy: 0.9493
Epoch 16/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.6998 - accuracy: 0.7912 - val_loss: 0.2596 - val_accuracy: 0.9547
Epoch 17/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.7074 - accuracy: 0.7898 - val_loss: 0.2694 - val_accuracy: 0.9437
Epoch 18/100
1688/1688 [==============================] - 9s 5ms/step - loss:
0.6930 - accuracy: 0.7950 - val_loss: 0.2647 - val_accuracy: 0.9512
Epoch 19/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.6884 - accuracy: 0.7958 - val_loss: 0.2655 - val_accuracy: 0.9495
Epoch 20/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.6787 - accuracy: 0.7987 - val_loss: 0.2744 - val_accuracy: 0.9503
313/313 [==============================] - 1s 3ms/step - loss: 0.3089
- accuracy: 0.9331
test_loss = 0.30893197655677795 test_acc = 0.9330999851226807
```


Loss vs epoch for relu [16, 32, 64] dropout 0.5
Training
Validation
Training Loss
Epoch
0.2
0.4
0.6
0.8
1.0
1.2
1.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

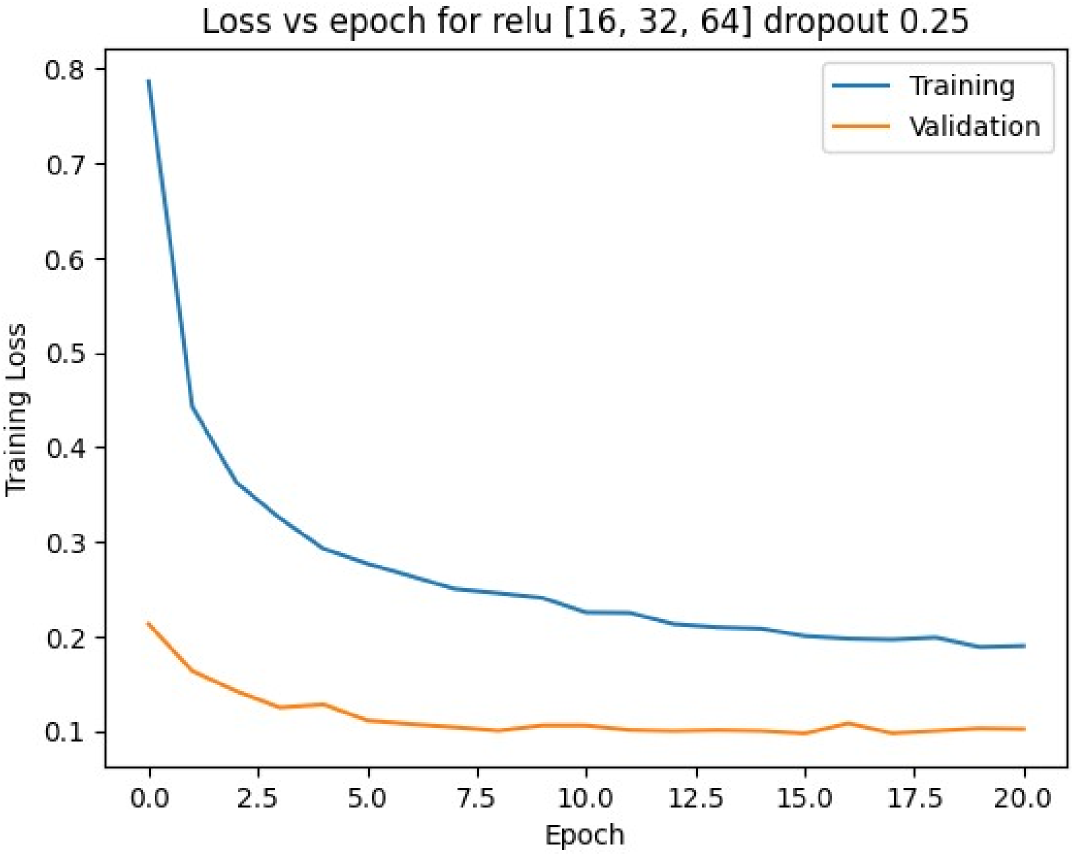


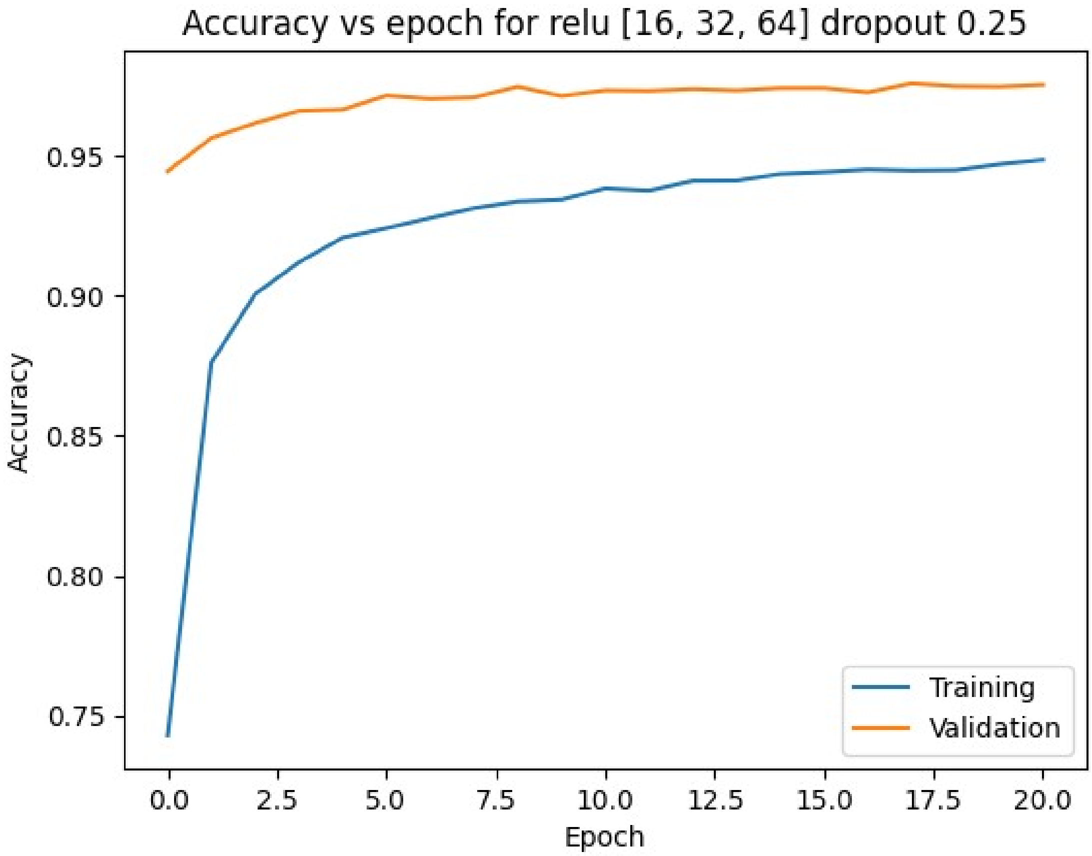
```
hidden_neurons = [16, 32, 64]
activation_function = 'relu'
dropout_val = 0.25

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: 53018 (207.10 KB)
Trainable params: 53018 (207.10 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 8s 4ms/step - loss:
0.7867 - accuracy: 0.7429 - val_loss: 0.2131 - val_accuracy: 0.9445
Epoch 2/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.4429 - accuracy: 0.8761 - val_loss: 0.1635 - val_accuracy: 0.9563
Epoch 3/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.3631 - accuracy: 0.9006 - val_loss: 0.1424 - val_accuracy: 0.9617
Epoch 4/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.3250 - accuracy: 0.9120 - val_loss: 0.1249 - val_accuracy: 0.9660
Epoch 5/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.2928 - accuracy: 0.9207 - val_loss: 0.1282 - val_accuracy: 0.9665
Epoch 6/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.2770 - accuracy: 0.9241 - val_loss: 0.1112 - val_accuracy: 0.9715
Epoch 7/100
1688/1688 [==============================] - 9s 5ms/step - loss:
0.2638 - accuracy: 0.9278 - val_loss: 0.1074 - val_accuracy: 0.9703
Epoch 8/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.2503 - accuracy: 0.9313 - val_loss: 0.1042 - val_accuracy: 0.9708
Epoch 9/100
1688/1688 [==============================] - 7s 4ms/step - loss:
```

```
0.2457 - accuracy: 0.9336 - val_loss: 0.1004 - val_accuracy: 0.9747
Epoch 10/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.2408 - accuracy: 0.9343 - val_loss: 0.1061 - val_accuracy: 0.9713
Epoch 11/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.2256 - accuracy: 0.9384 - val_loss: 0.1060 - val_accuracy: 0.9733
Epoch 12/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.2251 - accuracy: 0.9376 - val_loss: 0.1011 - val_accuracy: 0.9732
Epoch 13/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.2131 - accuracy: 0.9410 - val_loss: 0.1002 - val_accuracy: 0.9738
Epoch 14/100
1688/1688 [==============================] - 8s 4ms/step - loss:
0.2098 - accuracy: 0.9411 - val_loss: 0.1010 - val_accuracy: 0.9733
Epoch 15/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.2083 - accuracy: 0.9435 - val_loss: 0.1003 - val_accuracy: 0.9742
Epoch 16/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.2005 - accuracy: 0.9442 - val_loss: 0.0975 - val_accuracy: 0.9742
Epoch 17/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.1977 - accuracy: 0.9451 - val_loss: 0.1083 - val_accuracy: 0.9727
Epoch 18/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.1968 - accuracy: 0.9447 - val_loss: 0.0977 - val_accuracy: 0.9758
Epoch 19/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.1989 - accuracy: 0.9449 - val_loss: 0.1003 - val_accuracy: 0.9748
Epoch 20/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.1890 - accuracy: 0.9469 - val_loss: 0.1029 - val_accuracy: 0.9747
Epoch 21/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.1902 - accuracy: 0.9486 - val_loss: 0.1022 - val_accuracy: 0.9753
313/313 [==============================] - 1s 2ms/step - loss: 0.1353
- accuracy: 0.9680
test_loss = 0.13534332811832428 test_acc = 0.9679999947547913
```

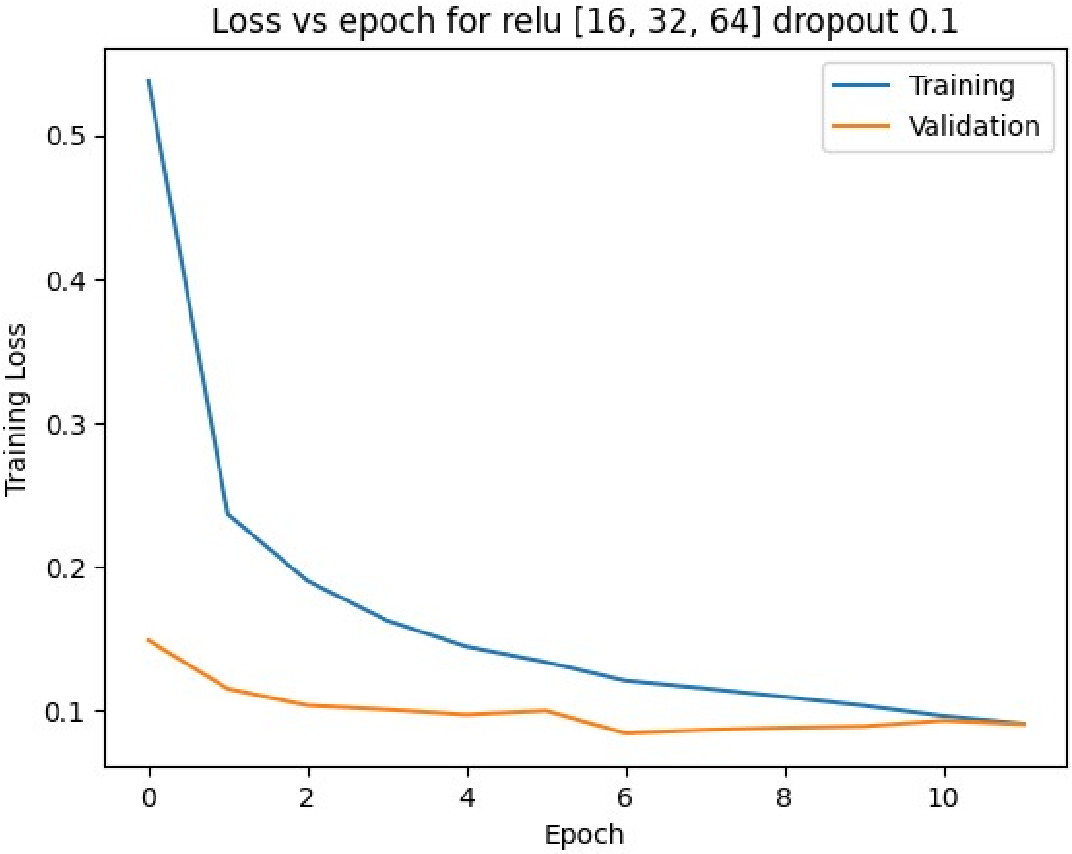

Loss vs epoch for relu [16, 32, 64] dropout 0.25
Training
Validation
Training Loss
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Epoch

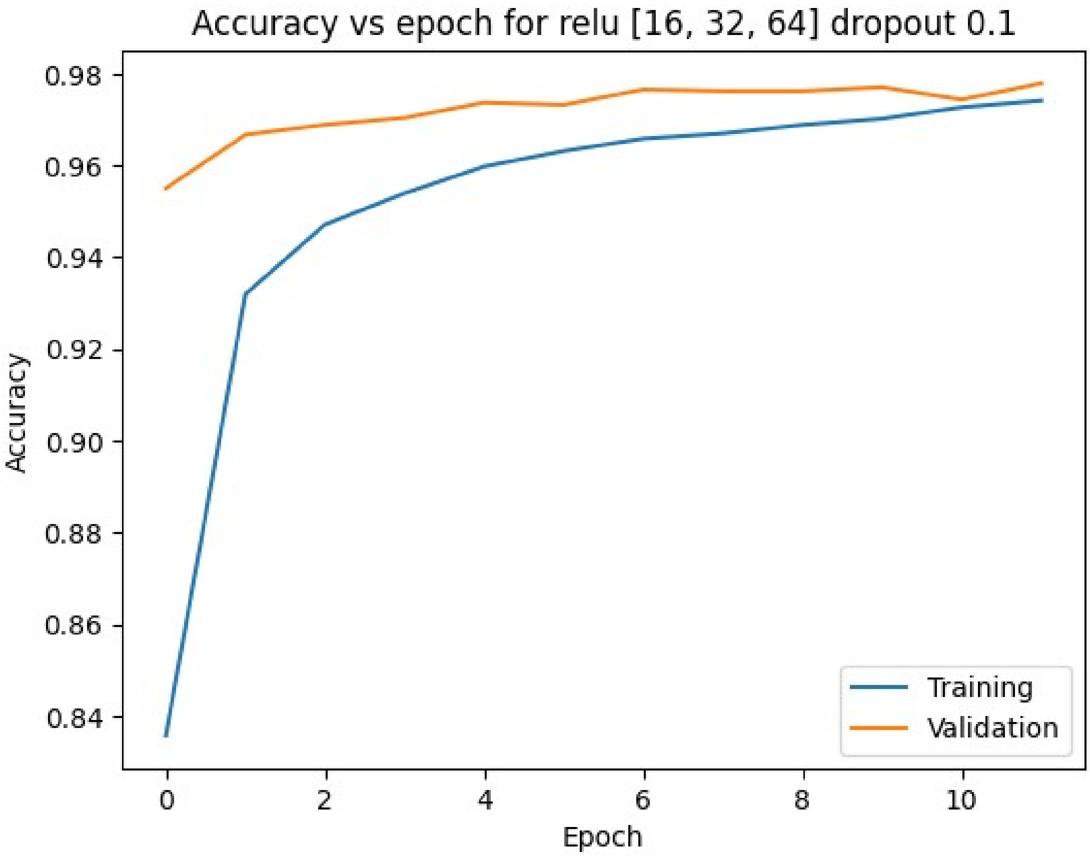


```
hidden_neurons = [16, 32, 64]
activation_function = 'relu'
dropout_val = 0.1

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: 53018 (207.10 KB)
Trainable params: 53018 (207.10 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 10s 5ms/step - loss:
0.5376 - accuracy: 0.8358 - val_loss: 0.1486 - val_accuracy: 0.9550
Epoch 2/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.2364 - accuracy: 0.9319 - val_loss: 0.1149 - val_accuracy: 0.9667
Epoch 3/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.1900 - accuracy: 0.9471 - val_loss: 0.1034 - val_accuracy: 0.9688
Epoch 4/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.1624 - accuracy: 0.9539 - val_loss: 0.1005 - val_accuracy: 0.9703
Epoch 5/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.1442 - accuracy: 0.9597 - val_loss: 0.0969 - val_accuracy: 0.9737
Epoch 6/100
1688/1688 [==============================] - 10s 6ms/step - loss:
0.1334 - accuracy: 0.9631 - val_loss: 0.0996 - val_accuracy: 0.9732
Epoch 7/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.1206 - accuracy: 0.9658 - val_loss: 0.0840 - val_accuracy: 0.9765
Epoch 8/100
1688/1688 [==============================] - 10s 6ms/step - loss:
0.1151 - accuracy: 0.9669 - val_loss: 0.0863 - val_accuracy: 0.9762
Epoch 9/100
1688/1688 [==============================] - 7s 4ms/step - loss:
```

```
0.1093 - accuracy: 0.9688 - val_loss: 0.0878 - val_accuracy: 0.9762
Epoch 10/100
1688/1688 [==============================] - 9s 6ms/step - loss:
0.1032 - accuracy: 0.9701 - val_loss: 0.0888 - val_accuracy: 0.9770
Epoch 11/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.0960 - accuracy: 0.9726 - val_loss: 0.0928 - val_accuracy: 0.9743
Epoch 12/100
1688/1688 [==============================] - 9s 5ms/step - loss:
0.0906 - accuracy: 0.9741 - val_loss: 0.0902 - val_accuracy: 0.9778
313/313 [==============================] - 1s 2ms/step - loss: 0.0967
- accuracy: 0.9736
test_loss = 0.09673555940389633 test_acc = 0.9735999703407288
```





```
result

   Hidden Layers Activation Function Hidden Neurons  Dropout  Test
Loss  \
0              3                relu   [16, 32, 64]     0.90
2.301211
1              3                relu   [16, 32, 64]     0.75
1.601853
2              3                relu   [16, 32, 64]     0.50
0.308932
3              3                relu   [16, 32, 64]     0.25
0.135343
4              3                relu   [16, 32, 64]     0.10
0.096736

   Test Acccuracy
0          0.1135
1          0.4008
2          0.9331
3          0.9680
4          0.9736
```

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

best_dropout
```

```
0.1
```

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.

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: tanh
best dropout value: 0.1
```

```
import time

result = pd.DataFrame(
    columns=[
        'Hidden Layers',
        'Activation Function',
        'Hidden Neurons',
        'Dropout',
        'Adam Learn Rate',
        '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 Layers Activation Function Hidden Neurons  Dropout  Adam 
Learn Rate  \
0              3                tanh   [16, 32, 64]      0.1   
0.0100   
1              3                tanh   [16, 32, 64]      0.1   
0.0010   
2              3                tanh   [16, 32, 64]      0.1   
0.0050   
3              3                tanh   [16, 32, 64]      0.1   
0.0001   
4              3                tanh   [16, 32, 64]      0.1   
0.0005   

   Time Taken  Test Loss  Test Acccuracy  
0  157.511762   0.235740          0.9393  
1  122.233136   0.100066          0.9725  
2   84.804303   0.162639          0.9548  
3  466.381780   0.089396          0.9740  
4  132.471228   0.092584          0.9736  
```

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

best_adam_learn_rate
```

```
0.0001
```

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: 53018 (207.10 KB)
Trainable params: 53018 (207.10 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/100
1688/1688 [==============================] - 10s 5ms/step - loss:
1.1136 - accuracy: 0.7203 - val_loss: 0.5934 - val_accuracy: 0.8740
Epoch 2/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.5996 - accuracy: 0.8561 - val_loss: 0.3821 - val_accuracy: 0.9110
Epoch 3/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.4552 - accuracy: 0.8865 - val_loss: 0.2924 - val_accuracy: 0.9265
Epoch 4/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.3805 - accuracy: 0.9016 - val_loss: 0.2465 - val_accuracy: 0.9355
Epoch 5/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.3373 - accuracy: 0.9091 - val_loss: 0.2174 - val_accuracy: 0.9407
Epoch 6/100
1688/1688 [==============================] - 9s 5ms/step - loss:
0.3067 - accuracy: 0.9164 - val_loss: 0.2015 - val_accuracy: 0.9427
Epoch 7/100
```

```
1688/1688 [==============================] - 6s 4ms/step - loss:
0.2851 - accuracy: 0.9217 - val_loss: 0.1844 - val_accuracy: 0.9478
Epoch 8/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.2695 - accuracy: 0.9246 - val_loss: 0.1733 - val_accuracy: 0.9498
Epoch 9/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.2524 - accuracy: 0.9292 - val_loss: 0.1646 - val_accuracy: 0.9527
Epoch 10/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.2425 - accuracy: 0.9320 - val_loss: 0.1551 - val_accuracy: 0.9558
Epoch 11/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.2296 - accuracy: 0.9343 - val_loss: 0.1474 - val_accuracy: 0.9567
Epoch 12/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.2212 - accuracy: 0.9376 - val_loss: 0.1432 - val_accuracy: 0.9587
Epoch 13/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.2138 - accuracy: 0.9387 - val_loss: 0.1382 - val_accuracy: 0.9612
Epoch 14/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.2076 - accuracy: 0.9409 - val_loss: 0.1319 - val_accuracy: 0.9617
Epoch 15/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.1976 - accuracy: 0.9432 - val_loss: 0.1289 - val_accuracy: 0.9618
Epoch 16/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.1936 - accuracy: 0.9443 - val_loss: 0.1257 - val_accuracy: 0.9630
Epoch 17/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.1872 - accuracy: 0.9460 - val_loss: 0.1205 - val_accuracy: 0.9643
Epoch 18/100
1688/1688 [==============================] - 9s 5ms/step - loss:
0.1833 - accuracy: 0.9464 - val_loss: 0.1188 - val_accuracy: 0.9662
Epoch 19/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.1804 - accuracy: 0.9483 - val_loss: 0.1173 - val_accuracy: 0.9655
Epoch 20/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.1723 - accuracy: 0.9504 - val_loss: 0.1154 - val_accuracy: 0.9658
Epoch 21/100
1688/1688 [==============================] - 9s 6ms/step - loss:
0.1685 - accuracy: 0.9507 - val_loss: 0.1109 - val_accuracy: 0.9670
Epoch 22/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.1650 - accuracy: 0.9516 - val_loss: 0.1121 - val_accuracy: 0.9665
Epoch 23/100
1688/1688 [==============================] - 7s 4ms/step - loss:
```

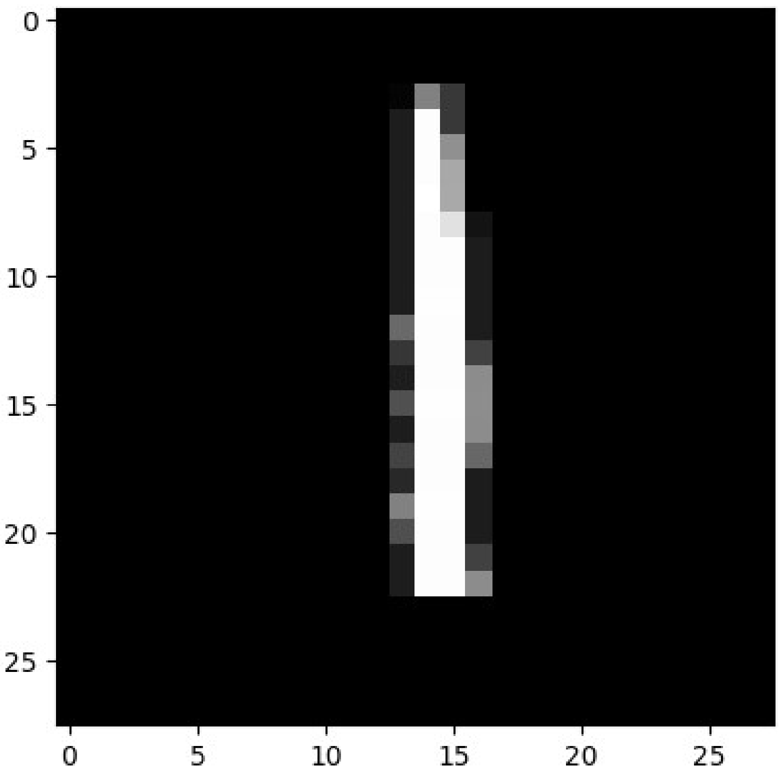
```
0.1629 - accuracy: 0.9528 - val_loss: 0.1088 - val_accuracy: 0.9677
Epoch 24/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.1591 - accuracy: 0.9537 - val_loss: 0.1065 - val_accuracy: 0.9697
Epoch 25/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.1571 - accuracy: 0.9534 - val_loss: 0.1053 - val_accuracy: 0.9685
Epoch 26/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.1519 - accuracy: 0.9552 - val_loss: 0.1054 - val_accuracy: 0.9687
Epoch 27/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.1495 - accuracy: 0.9565 - val_loss: 0.1031 - val_accuracy: 0.9682
Epoch 28/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.1483 - accuracy: 0.9560 - val_loss: 0.1018 - val_accuracy: 0.9697
Epoch 29/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.1471 - accuracy: 0.9565 - val_loss: 0.1005 - val_accuracy: 0.9697
Epoch 30/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.1430 - accuracy: 0.9578 - val_loss: 0.0987 - val_accuracy: 0.9705
Epoch 31/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.1399 - accuracy: 0.9584 - val_loss: 0.1001 - val_accuracy: 0.9713
Epoch 32/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.1377 - accuracy: 0.9598 - val_loss: 0.0982 - val_accuracy: 0.9718
Epoch 33/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.1359 - accuracy: 0.9592 - val_loss: 0.0957 - val_accuracy: 0.9722
Epoch 34/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.1363 - accuracy: 0.9598 - val_loss: 0.0963 - val_accuracy: 0.9712
Epoch 35/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.1338 - accuracy: 0.9612 - val_loss: 0.0947 - val_accuracy: 0.9718
Epoch 36/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.1327 - accuracy: 0.9603 - val_loss: 0.0953 - val_accuracy: 0.9718
Epoch 37/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.1312 - accuracy: 0.9619 - val_loss: 0.0939 - val_accuracy: 0.9715
Epoch 38/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.1278 - accuracy: 0.9619 - val_loss: 0.0942 - val_accuracy: 0.9718
Epoch 39/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.1248 - accuracy: 0.9636 - val_loss: 0.0926 - val_accuracy: 0.9738
```

```
Epoch 40/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.1245 - accuracy: 0.9629 - val_loss: 0.0935 - val_accuracy: 0.9727
Epoch 41/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.1229 - accuracy: 0.9637 - val_loss: 0.0914 - val_accuracy: 0.9717
Epoch 42/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.1211 - accuracy: 0.9641 - val_loss: 0.0911 - val_accuracy: 0.9742
Epoch 43/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.1200 - accuracy: 0.9642 - val_loss: 0.0916 - val_accuracy: 0.9717
Epoch 44/100
1688/1688 [==============================] - 8s 4ms/step - loss:
0.1205 - accuracy: 0.9647 - val_loss: 0.0917 - val_accuracy: 0.9728
Epoch 45/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.1173 - accuracy: 0.9654 - val_loss: 0.0920 - val_accuracy: 0.9733
Epoch 46/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.1156 - accuracy: 0.9651 - val_loss: 0.0908 - val_accuracy: 0.9742
Epoch 47/100
1688/1688 [==============================] - 9s 5ms/step - loss:
0.1131 - accuracy: 0.9664 - val_loss: 0.0882 - val_accuracy: 0.9747
Epoch 48/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.1143 - accuracy: 0.9659 - val_loss: 0.0878 - val_accuracy: 0.9735
Epoch 49/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.1129 - accuracy: 0.9661 - val_loss: 0.0875 - val_accuracy: 0.9742
Epoch 50/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.1106 - accuracy: 0.9674 - val_loss: 0.0872 - val_accuracy: 0.9745
Epoch 51/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.1092 - accuracy: 0.9672 - val_loss: 0.0889 - val_accuracy: 0.9735
Epoch 52/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.1087 - accuracy: 0.9673 - val_loss: 0.0860 - val_accuracy: 0.9738
Epoch 53/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.1067 - accuracy: 0.9684 - val_loss: 0.0873 - val_accuracy: 0.9747
Epoch 54/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.1064 - accuracy: 0.9669 - val_loss: 0.0873 - val_accuracy: 0.9735
Epoch 55/100
1688/1688 [==============================] - 7s 4ms/step - loss:
0.1039 - accuracy: 0.9686 - val_loss: 0.0854 - val_accuracy: 0.9755
Epoch 56/100
```

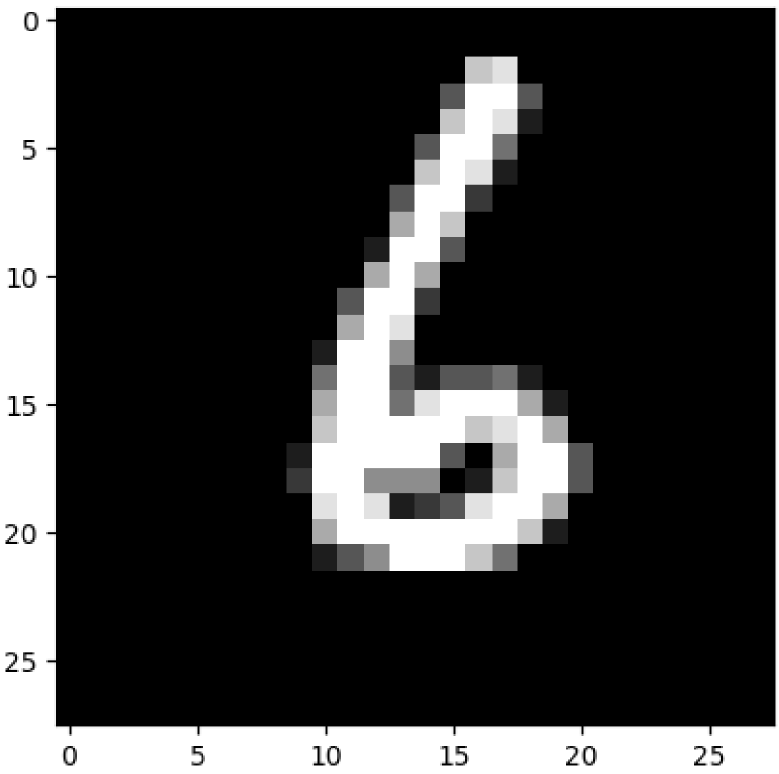
```
1688/1688 [==============================] - 6s 4ms/step - loss:
0.1040 - accuracy: 0.9680 - val_loss: 0.0855 - val_accuracy: 0.9748
Epoch 57/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.1041 - accuracy: 0.9685 - val_loss: 0.0858 - val_accuracy: 0.9743
Epoch 58/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.1021 - accuracy: 0.9696 - val_loss: 0.0854 - val_accuracy: 0.9750
Epoch 59/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.1010 - accuracy: 0.9688 - val_loss: 0.0833 - val_accuracy: 0.9763
Epoch 60/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.0997 - accuracy: 0.9698 - val_loss: 0.0823 - val_accuracy: 0.9748
Epoch 61/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.0990 - accuracy: 0.9710 - val_loss: 0.0823 - val_accuracy: 0.9748
Epoch 62/100
1688/1688 [==============================] - 8s 4ms/step - loss:
0.0985 - accuracy: 0.9700 - val_loss: 0.0841 - val_accuracy: 0.9765
Epoch 63/100
1688/1688 [==============================] - 6s 4ms/step - loss:
0.0987 - accuracy: 0.9704 - val_loss: 0.0843 - val_accuracy: 0.9755
Epoch 64/100
1688/1688 [==============================] - 6s 3ms/step - loss:
0.0988 - accuracy: 0.9708 - val_loss: 0.0846 - val_accuracy: 0.9753
Epoch 65/100
1688/1688 [==============================] - 8s 5ms/step - loss:
0.0958 - accuracy: 0.9709 - val_loss: 0.0839 - val_accuracy: 0.9772
Epoch 66/100
1688/1688 [==============================] - 5s 3ms/step - loss:
0.0931 - accuracy: 0.9718 - val_loss: 0.0838 - val_accuracy: 0.9737
```

```python
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}")
```

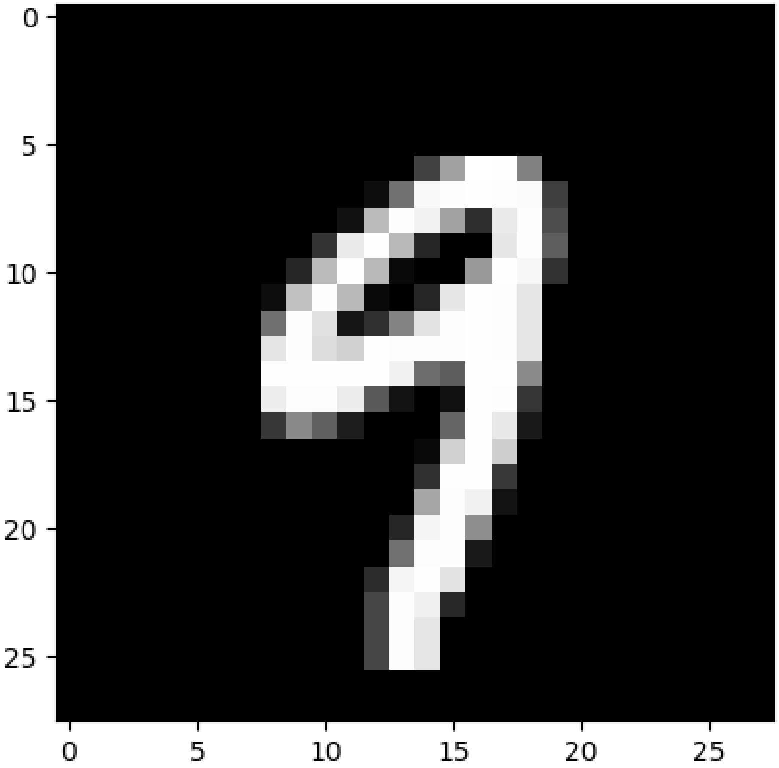
```
1/1 [==============================] - 0s 115ms/step
```



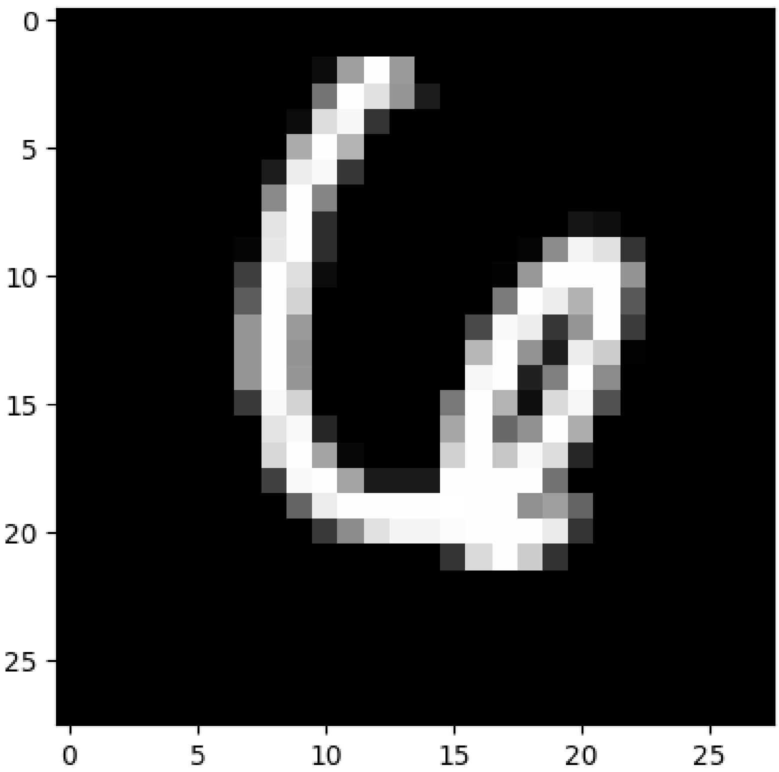
```
predict = 1
```



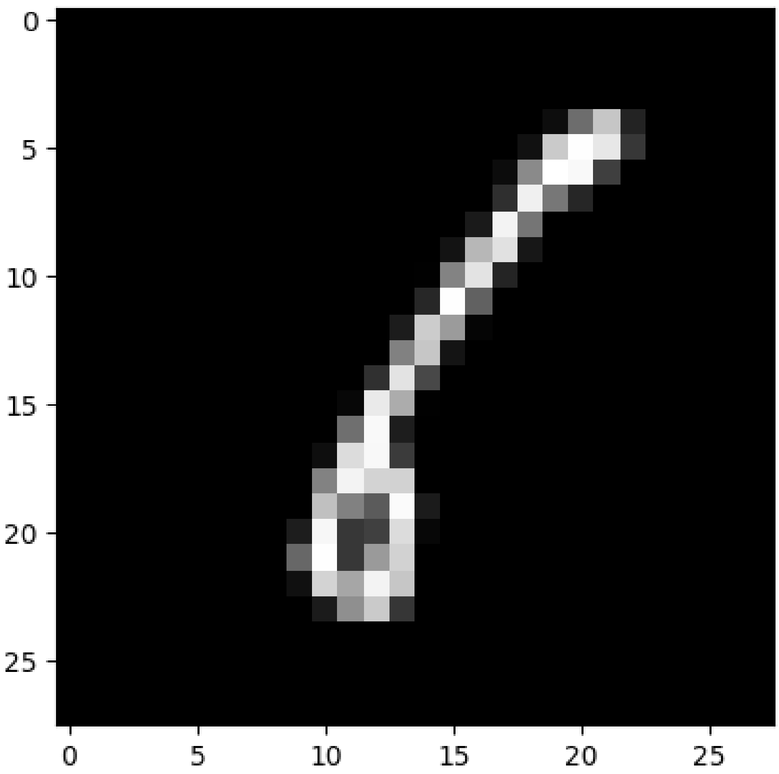
```
predict = 6
```



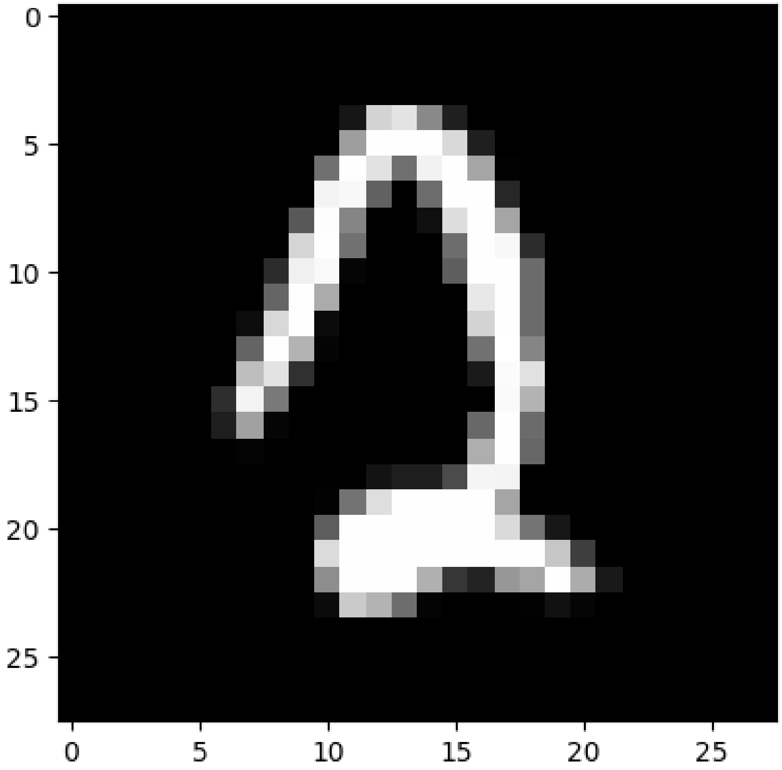
```
predict = 9
```



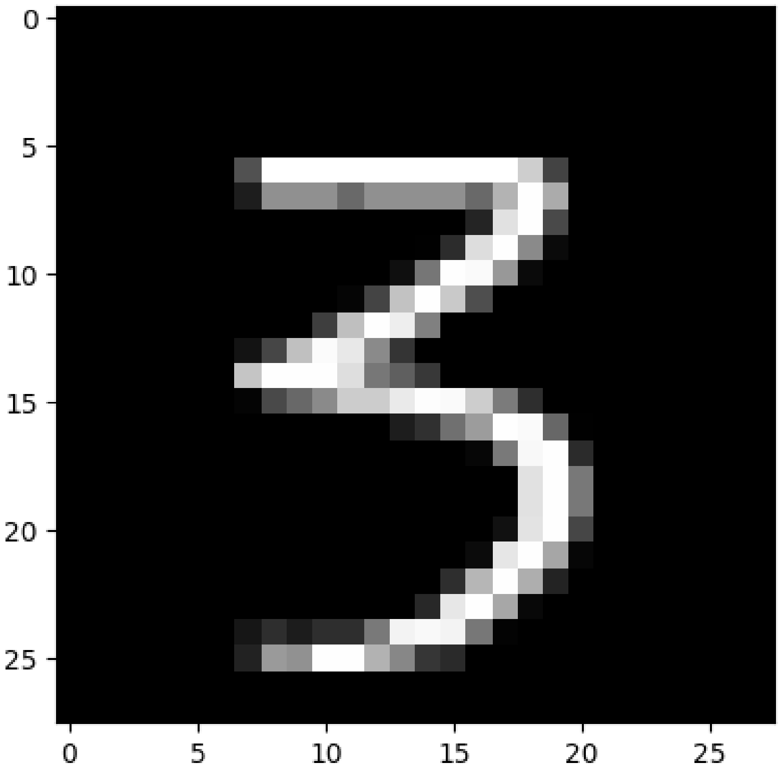
```
predict = 6
```



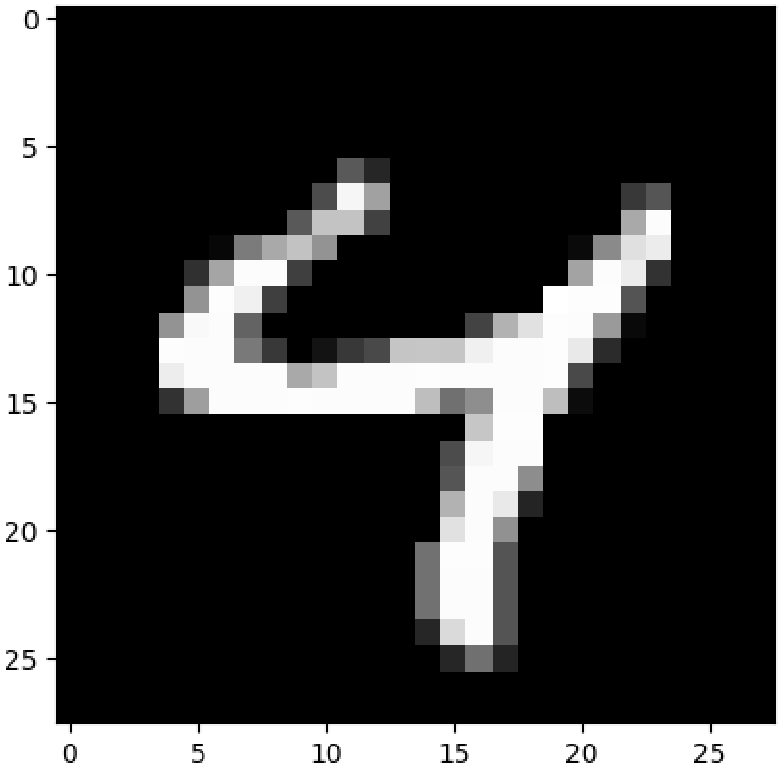
```
predict = 1
```



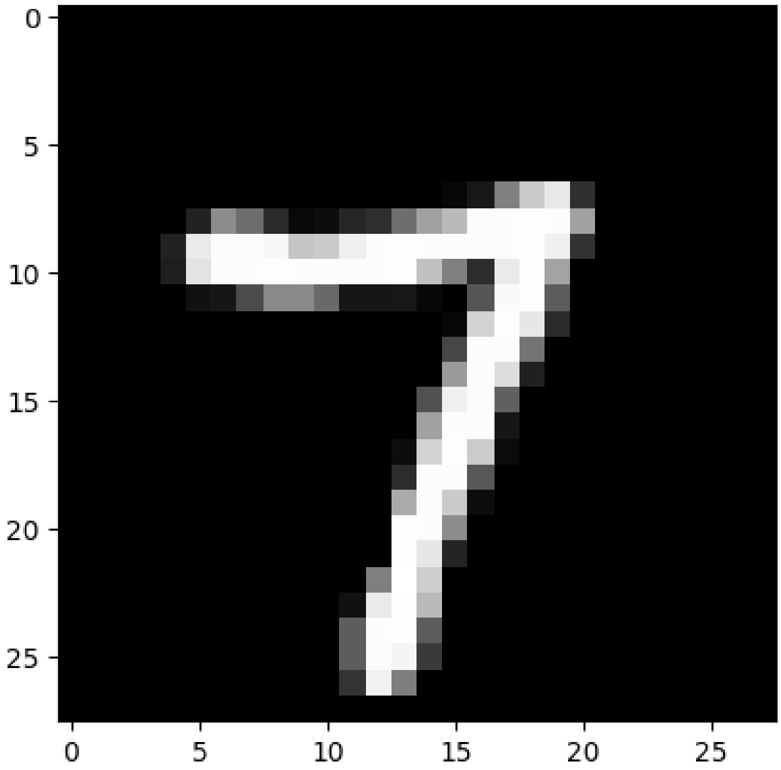
```
predict = 0
```



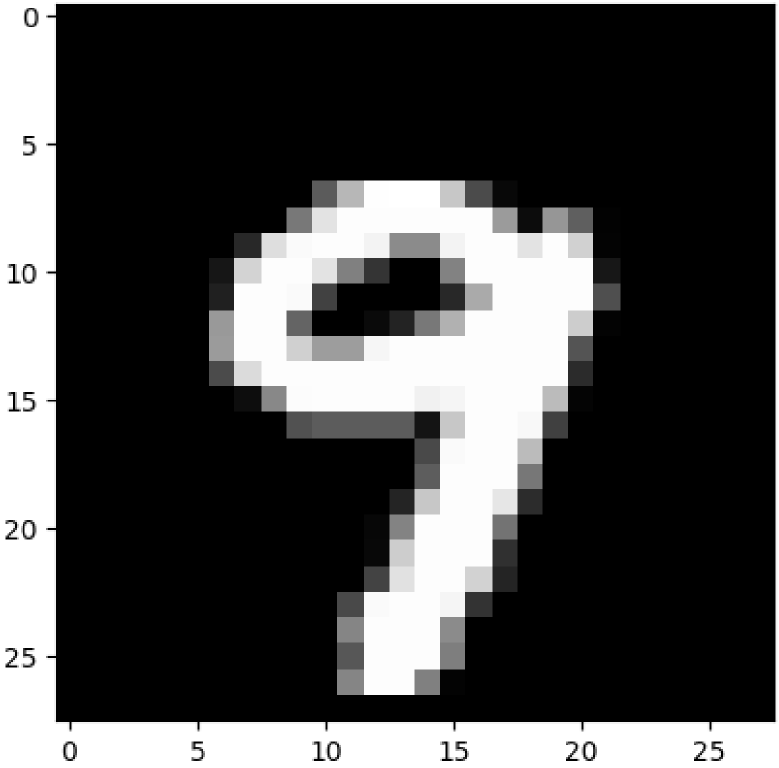
```
predict = 3
```



```
predict = 4
```



```
predict = 7
```



```
predict = 9
```