Assignment 04

Name:- Sk Fardeen Hossain

Roll No. :- 2021CSB023

G-Suite Id:- 2021csb023.sk@students.iiests.ac.in

Department:- Computer Science and Technology

Question 01

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).

In [1]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# Install the tensorflow library
!pip install tensorflow
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.17.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow)
(0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=3.10.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.11.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/li
b/python3.10/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (71.0.4)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.64.1)
Requirement already satisfied: tensorboard<2.18,>=2.17 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.17.0)
Requirement already satisfied: keras>=3.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow)
(0.37.1)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.26.4)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflo
w) (0.44.0)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->tensorflow) (13.8.1)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages (from keras>=3.2.0->tensorflow) (0.12.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->te
nsorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorflow)
(3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorfl
ow) (2.2.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorfl
ow) (2024.8.30)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17->tensorf
low) (3.7)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboar
d<2.18,>=2.17->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.18,>=2.17->tensorf
low) (3.0.4)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<
```

```
2.18,>=2.17->tensorflow) (2.1.5)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0->tensor
flow) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich->keras>=3.2.0->tens
orflow) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=
3.2.0->tensorflow) (0.1.2)
```

Question 02

Download the MNIST dataset (contains class labels for digits 0 – 9)

In [3]:
```
from keras.datasets import mnist
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
```

In [4]:
```
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

Out[4]:
```
((60000, 28, 28), (60000,), (10000, 28, 28), (10000,))
```

In [5]:
```
print(X_train[0])
```

```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   3  18  18  18 126 136
  175  26 166 255 247 127   0   0   0   0]
 [  0   0   0   0   0   0   0   0  30  36  94 154 170 253 253 253 253 253
  225 172 253 242 195  64   0   0   0   0]
 [  0   0   0   0   0   0   0  49 238 253 253 253 253 253 253 253 253 251
   93  82  82  56  39   0   0   0   0   0]
 [  0   0   0   0   0   0   0  18 219 253 253 253 253 253 198 182 247 241
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  80 156 107 253 253 205  11   0  43 154
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  14   1 154 253  90   0   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0 139 253 190   2   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0  11 190 253  70   0   0   0
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  35 241 225 160 108   1
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0  81 240 253 253 119
   25   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  45 186 253 253
  150  27   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  16  93 252
  253 187   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0 249
  253 249  64   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0  46 130 183 253
  253 207   2   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0  39 148 229 253 253 253
  250 182   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0  24 114 221 253 253 253 253 201
   78   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  23  66 213 253 253 253 253 198  81   2
    0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0  18 171 219 253 253 253 253 195  80   9   0   0
    0   0   0   0   0   0   0   0   0   0]
```

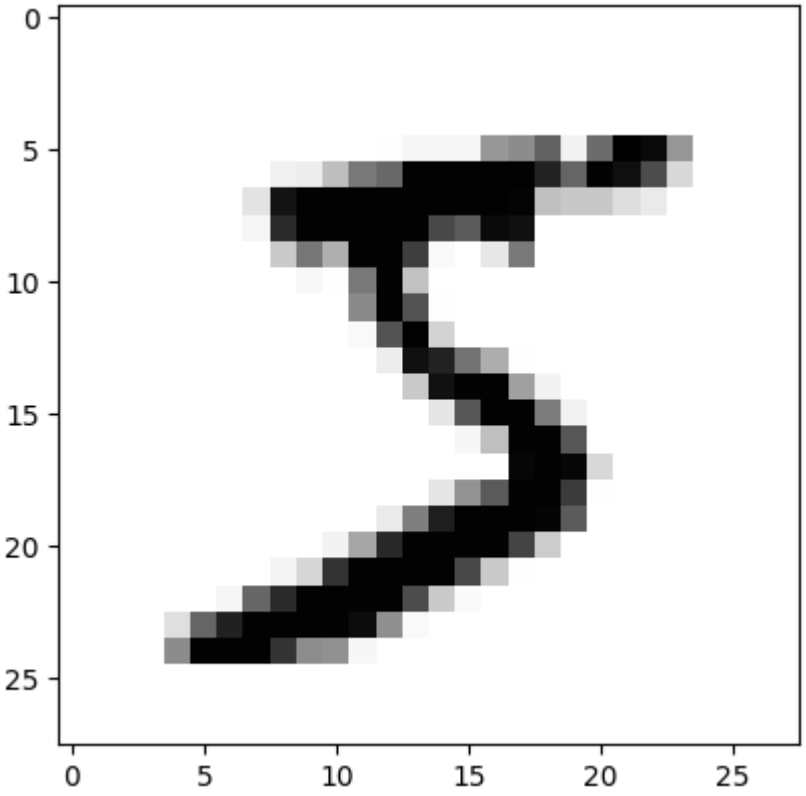
```
[  0   0   0   0  55 172 226 253 253 253 253 244 133  11   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0 136 253 253 253 212 135 132  16   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]
[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0]]
```

As we can see, every image is given as a 28x28 pixel map where every value ranges from 0 to 255 on an intensity scale (0 ---> white and 255 ---> black). So it is basically a gray-scale image.

In [6]:

```python
# Let's see a sample image

import matplotlib.pyplot as plt

plt.imshow(X_train[0],cmap="Greys")

print(f"The image is a {y_train[0]}")
```

```
The image is a 5
```



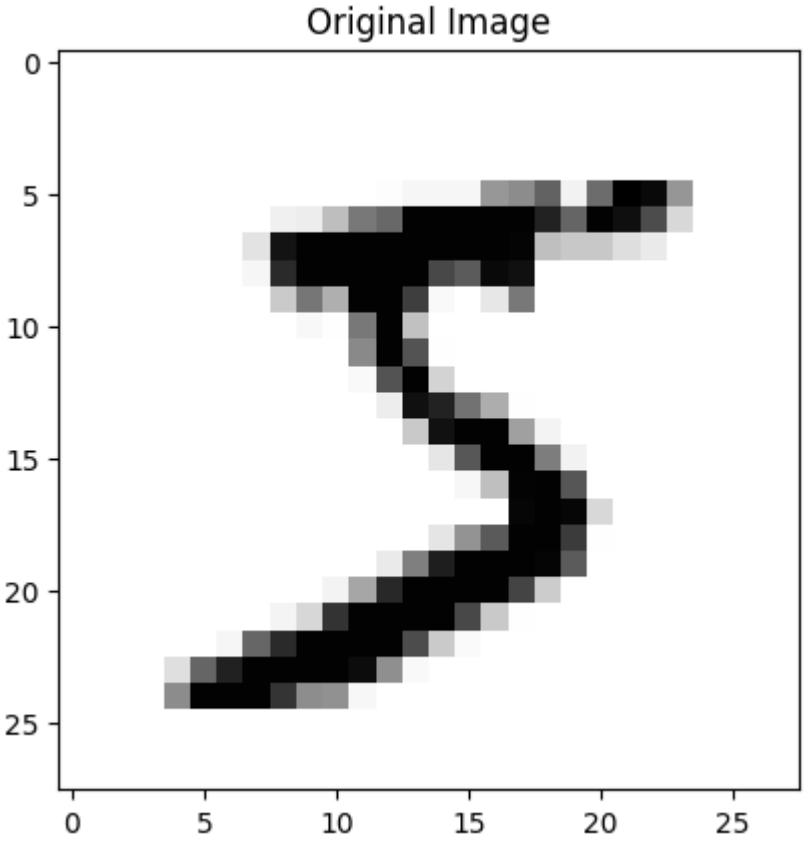
Question 03

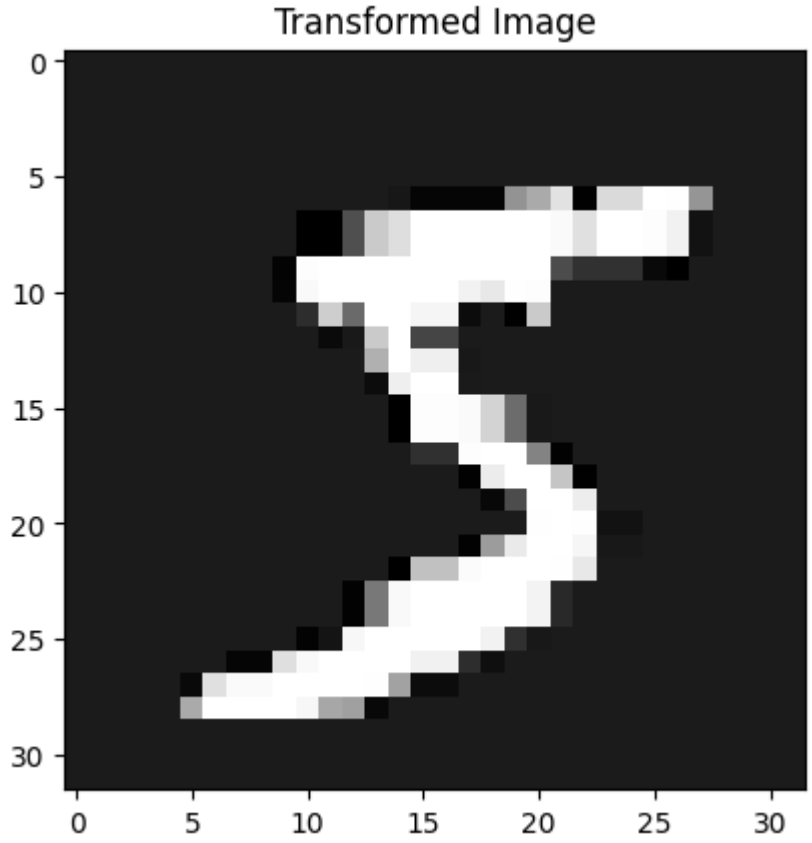
Reduce the training size by 1/10 if computation resources are limited. Define Radial Basis Function (RBF) as `def RBF(x, c, s): return np.exp(-np.sum((x - c) ** 2, axis = 1)/(2 * s ** 2))` , where x is the actual value, c is center (assumed as mean) and s is the standard deviation. Converted 28 × 28 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.

In [7]:
```
import numpy as np

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

In [8]:
```
## Convert a 28x28 image to 32x32 using RBF

# Assuming x_train[0] is a 28x28 image
image = X_train[0]

# Calculate mean and standard deviation of the image
mean = np.mean(image)
```

```python
std = np.std(image)

# Vectorize the RBF function to apply it element-wise on the image
RBF_vectorized = np.vectorize(RBF)

x_new = np.linspace(0, 27, 32).astype(int)  # Mapping 32 points to 28
y_new = np.linspace(0, 27, 32).astype(int)

# Resize the image by selecting pixels from the original 28x28 image
new_image = image[np.ix_(x_new, y_new)]

# Apply the RBF kernel on the resized image element-wise
new_image = RBF_vectorized(new_image, mean, std)

plt.imshow(image,cmap="Greys")
plt.title("Original Image")
plt.show()

plt.imshow(new_image,cmap="Greys")
plt.title("Transformed Image")
plt.show()
```


Original Image
0
5
10
15
20
25
0
5
10
15
20
25



In [9]:
```
## Do this iteratively for all the images
'''
  @params:- A 28x28 image
  @return:- A 32x32 image
  @brief:- Convert a 28x28 image to 32x32 using RBF
'''
RBF_vectorized = np.vectorize(RBF)

def get_transformed_image(image):
  mean = np.mean(image)
  std = np.std(image)
  x_new = np.linspace(0, 27, 32).astype(int)  # Mapping 32 points to 28
  y_new = np.linspace(0, 27, 32).astype(int)

  # Resize the image by selecting pixels from the original 28x28 image
  new_image = image[np.ix_(x_new, y_new)]

  # Apply the RBF kernel on the resized image element-wise
  new_image = RBF_vectorized(new_image, mean, std)
```

```
  return new_image
```

In [10]:
```
# Combine both training and testing data into training for more samples
X_train = np.concatenate((X_train,X_test),axis=0)
y_train = np.concatenate((y_train,y_test),axis=0)
X_train.shape
```

Out[10]: (70000, 28, 28)

In [11]:
```
# Training Data -> 80% , Validation Data -> 10% , Testing Data -> 10%

from sklearn.model_selection import train_test_split

X_train, X_rem, y_train, y_rem = train_test_split(X_train, y_train, test_size=0.2, random_state=4)
X_val, X_test, y_val, y_test = train_test_split(X_rem, y_rem, test_size=0.5, random_state=4)
```

In [12]:
```
from tqdm import tqdm
X_train_transformed = np.empty((X_train.shape[0],32,32))

for i,image in tqdm(enumerate(X_train)):
  X_train_transformed[i] = get_transformed_image(image)

X_train_transformed.shape
```

```
56000it [08:56, 104.33it/s]
```

Out[12]: (56000, 32, 32)

In [13]:
```
X_val_transformed = np.empty((X_val.shape[0],32,32))

for i,image in tqdm(enumerate(X_val)):
  X_val_transformed[i] = get_transformed_image(image)

X_val_transformed.shape
```

```
7000it [01:06, 105.69it/s]
```

Out[13]: (7000, 32, 32)

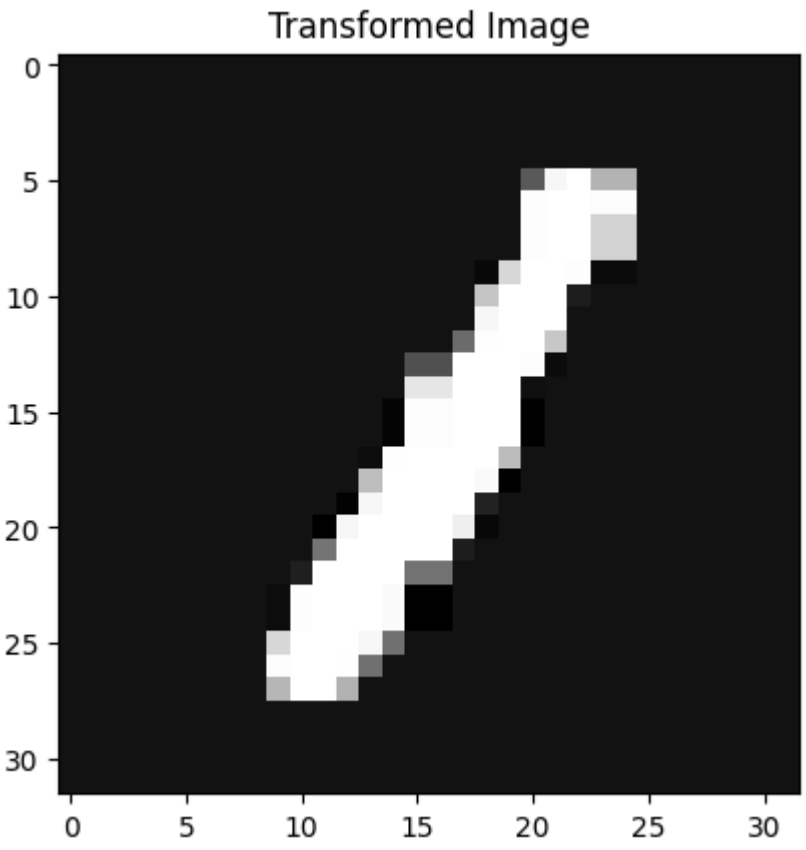
In [14]:
```
X_test_transformed = np.empty((X_test.shape[0],32,32))

for i,image in tqdm(enumerate(X_test)):
  X_test_transformed[i] = get_transformed_image(image)

X_test_transformed.shape
```

```
7000it [01:06, 105.69it/s]
```

Out[14]: (7000, 32, 32)

In [15]:
```
# Renaming
X_train = X_train_transformed
X_val = X_val_transformed
X_test = X_test_transformed
```

In [16]:
```
plt.imshow(X_train[28],cmap="Greys")
plt.title("Transformed Image")
plt.show()
```



In [17]:
```
# One hot encode the target labels [0-9]

import pandas as pd

y_train = pd.get_dummies(y_train,dtype='int').to_numpy()
y_val = pd.get_dummies(y_val,dtype='int').to_numpy()
y_test = pd.get_dummies(y_test,dtype='int').to_numpy()
y_train[28]
```

Out[17]: array([0, 1, 0, 0, 0, 0, 0, 0, 0, 0])

Question 04

Now run the fully connected network after flattening the data by changing the number of hyper-parameters:

a. Use Gradient Descent Optimizer (learning rate = 0.001) and Squared Error Loss
b. 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]

Try all the possible combinations.

In [21]:
```
# Import necessary libraries
from tensorflow.keras import Sequential, Input
from tensorflow.keras.layers import Dense, Flatten, Dropout
from tensorflow.keras.losses import MeanSquaredError,CategoricalCrossentropy
from tensorflow.keras.optimizers import Adam,SGD
from tensorflow.keras.callbacks import EarlyStopping # Stop training when a monitored metric has stopped improving
import time # Calculate training time

# Defining Utility functions
def train_model(
    activation_func: 'str',
    hidden_neurons: 'list(int)',
    optimizer: 'tensorflow.keras.optimizers',
    loss_func: 'tensorflow.keras.losses',
    epochs: 'int',
    dropout_rate: 'float' = None,
```

```
    learning_rate: 'float' = 0.001,

):
  model = Sequential()  # Sequential model (feed forward neural net)
  # Adding the input layer
  model.add(Input(shape=(32,32)))

  # Flatten the input
  model.add(Flatten())

  # Hidden Layer Addition
  for i in range(len(hidden_neurons)):
    model.add(Dense(hidden_neurons[i], activation=activation_func))
    if dropout_rate:
      model.add(Dropout(rate=dropout_rate))

  # Adding the Output Layer
  model.add(Dense(10, activation='softmax'))  # Multiclass classification so softmax activation function

  # Model summary
  model.summary()

  # Compile the model
  model.compile(
        optimizer=optimizer(learning_rate=learning_rate),
        loss=loss_func(),
        metrics=['accuracy']
    )

  start=time.time()

  # Fit the model
  history = model.fit(
      x=X_train,
      y=y_train,
      epochs=epochs,
      validation_data=(X_val, y_val),
      callbacks=[
          EarlyStopping(monitor='val_loss', patience=5,restore_best_weights=True)
      ],
      verbose='auto'
  )

  end=time.time()

  duration = end-start
```

```
  return model, history, duration



def plot_metrics(
    history: 'tensorflow.keras.callbacks.History',
    activation_func: 'str',
    hidden_neurons: 'list(int)',
    dropout_rate: 'float' = None
):

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

  if dropout_rate is None:
    plt.title(f"Loss vs epochs for {activation_func} {hidden_neurons}")
  else:
    plt.title(f"Loss vs epochs for {activation_func} {hidden_neurons} dropout {dropout_rate}")

  plt.legend()
  plt.show()

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

  if dropout_rate is None:
    plt.title(f"Accuracy vs epochs for {activation_func} {hidden_neurons}")
  else:
    plt.title(f"Accuracy vs epochs for {activation_func} {hidden_neurons} dropout {dropout_rate}")

  plt.legend()
  plt.show()
```

In [43]:

```
import pandas as pd

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

```
        'Hidden Neurons',
        'Optimizer',
        'Training Time(in seconds)',
        'Test Loss',
        'Test Accuracy'
    ]
)
```

In [44]:

```
## Model 1 --> SGD, lr =0.001, Mean Squared Error loss, hidden_neurons=[16]

hidden_neurons = [16]
activation_function = 'sigmoid'

model, history, duration = train_model(
    activation_func=activation_function,
    hidden_neurons=hidden_neurons,
    optimizer=SGD,
    loss_func=MeanSquaredError,
    epochs=100,
    dropout_rate=None,
    learning_rate=0.001
)

test_loss, test_acc = model.evaluate(X_test,y_test)

print(f"Test loss={test_loss} Test accuracy = {test_acc}")
print(f"Time required for training the model is {duration} seconds")

result_df.loc[len(result_df.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    'SGD',
    duration,
    test_loss,
    test_acc
]

plot_metrics(history, activation_function, hidden_neurons)
```

Model: "sequential_16"

Layer (type)	Output Shape	Param #
flatten_16 (Flatten)	(None, 1024)	0
dense_56 (Dense)	(None, 16)	16,400
dense_57 (Dense)	(None, 10)	170

Total params: 16,570 (64.73 KB)

Trainable params: 16,570 (64.73 KB)

Non-trainable params: 0 (0.00 B)

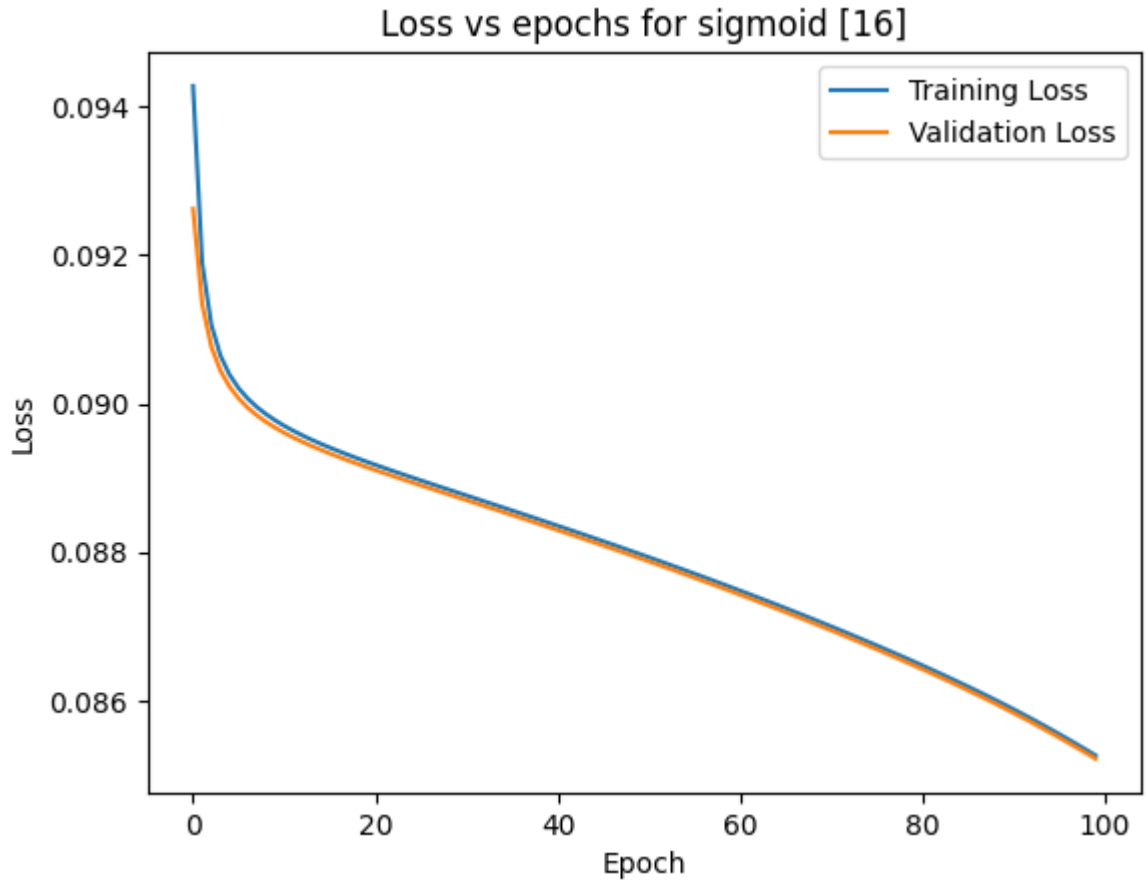
```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.0980 - loss: 0.0954 - val_accuracy: 0.0943 - val_loss: 0.0926
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.0970 - loss: 0.0922 - val_accuracy: 0.0976 - val_loss: 0.0913
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1023 - loss: 0.0912 - val_accuracy: 0.1031 - val_loss: 0.0908
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1053 - loss: 0.0907 - val_accuracy: 0.1040 - val_loss: 0.0904
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1082 - loss: 0.0904 - val_accuracy: 0.1161 - val_loss: 0.0902
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1191 - loss: 0.0903 - val_accuracy: 0.1334 - val_loss: 0.0901
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1364 - loss: 0.0901 - val_accuracy: 0.1534 - val_loss: 0.0899
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1504 - loss: 0.0900 - val_accuracy: 0.1666 - val_loss: 0.0898
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1558 - loss: 0.0899 - val_accuracy: 0.1746 - val_loss: 0.0898
Epoch 10/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1675 - loss: 0.0898 - val_accuracy: 0.1794 - val_loss: 0.0897
Epoch 11/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1738 - loss: 0.0897 - val_accuracy: 0.1806 - val_loss: 0.0896
Epoch 12/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1759 - loss: 0.0896 - val_accuracy: 0.1833 - val_loss: 0.0895
Epoch 13/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1796 - loss: 0.0895 - val_accuracy: 0.1870 - val_loss: 0.0895
Epoch 14/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.1812 - loss: 0.0895 - val_accuracy: 0.1901 - val_loss: 0.0894
Epoch 15/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1839 - loss: 0.0895 - val_accuracy: 0.1917 - val_loss: 0.0894
Epoch 16/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1835 - loss: 0.0894 - val_accuracy: 0.1927 - val_loss: 0.0893
Epoch 17/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.1879 - loss: 0.0894 - val_accuracy: 0.1947 - val_loss: 0.0893
Epoch 18/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 2ms/step - accuracy: 0.1901 - loss: 0.0893 - val_accuracy: 0.1973 - val_loss: 0.0892
Epoch 19/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1907 - loss: 0.0893 - val_accuracy: 0.1996 - val_loss: 0.0892
Epoch 20/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1896 - loss: 0.0892 - val_accuracy: 0.2024 - val_loss: 0.0891
Epoch 21/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1928 - loss: 0.0892 - val_accuracy: 0.2054 - val_loss: 0.0891
Epoch 22/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1953 - loss: 0.0891 - val_accuracy: 0.2066 - val_loss: 0.0891
Epoch 23/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1952 - loss: 0.0891 - val_accuracy: 0.2073 - val_loss: 0.0890
```

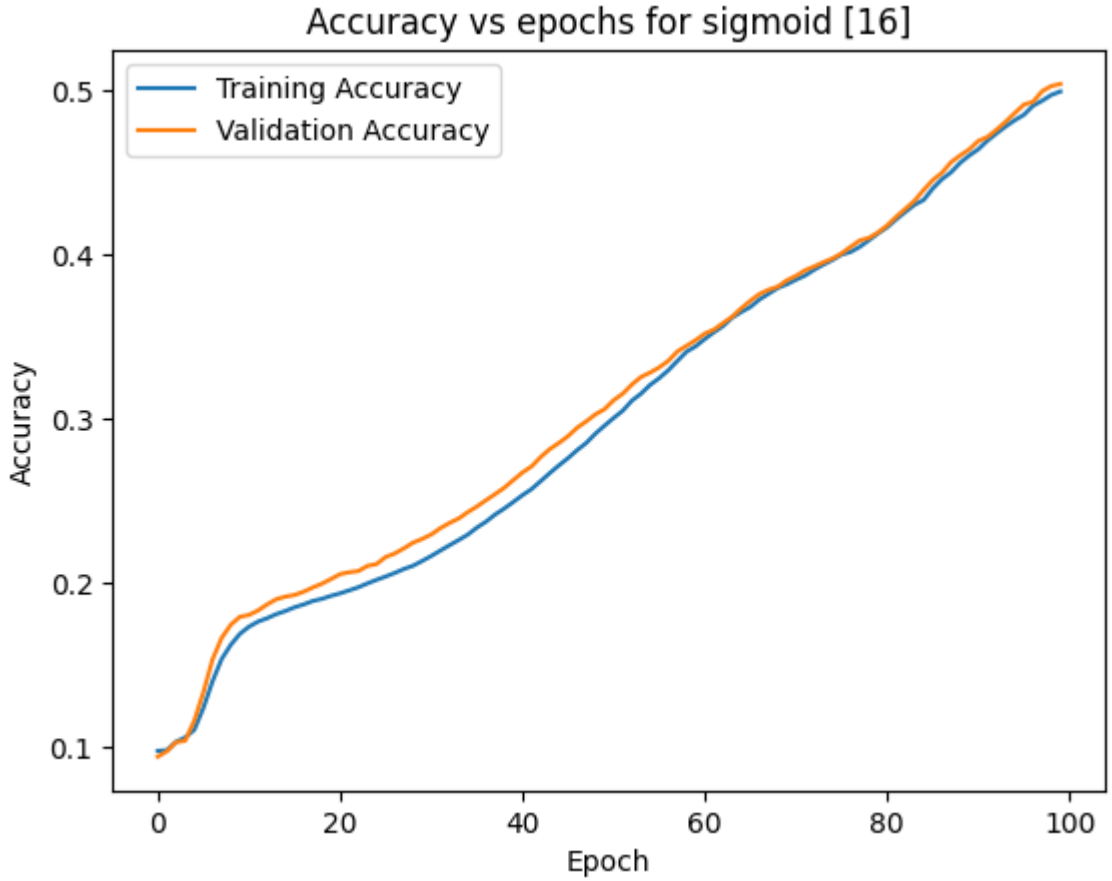
```
Epoch 24/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1972 - loss: 0.0891 - val_accuracy: 0.2104 - val_loss: 0.0890
Epoch 25/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.2036 - loss: 0.0890 - val_accuracy: 0.2116 - val_loss: 0.0889
Epoch 26/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 8s 2ms/step - accuracy: 0.2039 - loss: 0.0890 - val_accuracy: 0.2159 - val_loss: 0.0889
Epoch 27/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.2041 - loss: 0.0889 - val_accuracy: 0.2180 - val_loss: 0.0889
Epoch 28/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.2072 - loss: 0.0889 - val_accuracy: 0.2211 - val_loss: 0.0888
Epoch 29/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.2098 - loss: 0.0889 - val_accuracy: 0.2246 - val_loss: 0.0888
Epoch 30/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.2132 - loss: 0.0888 - val_accuracy: 0.2269 - val_loss: 0.0887
Epoch 31/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.2145 - loss: 0.0888 - val_accuracy: 0.2296 - val_loss: 0.0887
Epoch 32/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.2219 - loss: 0.0887 - val_accuracy: 0.2334 - val_loss: 0.0887
Epoch 33/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.2224 - loss: 0.0887 - val_accuracy: 0.2366 - val_loss: 0.0886
Epoch 34/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.2253 - loss: 0.0886 - val_accuracy: 0.2393 - val_loss: 0.0886
Epoch 35/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.2303 - loss: 0.0886 - val_accuracy: 0.2431 - val_loss: 0.0885
Epoch 36/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.2332 - loss: 0.0886 - val_accuracy: 0.2466 - val_loss: 0.0885
Epoch 37/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.2353 - loss: 0.0885 - val_accuracy: 0.2504 - val_loss: 0.0885
Epoch 38/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.2408 - loss: 0.0885 - val_accuracy: 0.2541 - val_loss: 0.0884
Epoch 39/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.2419 - loss: 0.0885 - val_accuracy: 0.2580 - val_loss: 0.0884
Epoch 40/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.2458 - loss: 0.0884 - val_accuracy: 0.2627 - val_loss: 0.0883
Epoch 41/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.2524 - loss: 0.0884 - val_accuracy: 0.2673 - val_loss: 0.0883
Epoch 42/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.2582 - loss: 0.0883 - val_accuracy: 0.2710 - val_loss: 0.0883
Epoch 43/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.2627 - loss: 0.0883 - val_accuracy: 0.2767 - val_loss: 0.0882
Epoch 44/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.2662 - loss: 0.0882 - val_accuracy: 0.2816 - val_loss: 0.0882
Epoch 45/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.2695 - loss: 0.0882 - val_accuracy: 0.2853 - val_loss: 0.0881
Epoch 46/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.2738 - loss: 0.0881 - val_accuracy: 0.2894 - val_loss: 0.0881
```

```
Epoch 47/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.2819 - loss: 0.0881 - val_accuracy: 0.2946 - val_loss: 0.0880
Epoch 48/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.2882 - loss: 0.0880 - val_accuracy: 0.2984 - val_loss: 0.0880
Epoch 49/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.2903 - loss: 0.0880 - val_accuracy: 0.3027 - val_loss: 0.0880
Epoch 50/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.2922 - loss: 0.0880 - val_accuracy: 0.3059 - val_loss: 0.0879
Epoch 51/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.2975 - loss: 0.0880 - val_accuracy: 0.3113 - val_loss: 0.0879
Epoch 52/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.3047 - loss: 0.0879 - val_accuracy: 0.3153 - val_loss: 0.0878
Epoch 53/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.3120 - loss: 0.0878 - val_accuracy: 0.3209 - val_loss: 0.0878
Epoch 54/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.3160 - loss: 0.0878 - val_accuracy: 0.3254 - val_loss: 0.0877
Epoch 55/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.3202 - loss: 0.0878 - val_accuracy: 0.3281 - val_loss: 0.0877
Epoch 56/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.3285 - loss: 0.0877 - val_accuracy: 0.3313 - val_loss: 0.0877
Epoch 57/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.3291 - loss: 0.0877 - val_accuracy: 0.3353 - val_loss: 0.0876
Epoch 58/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.3359 - loss: 0.0876 - val_accuracy: 0.3410 - val_loss: 0.0876
Epoch 59/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.3412 - loss: 0.0876 - val_accuracy: 0.3444 - val_loss: 0.0875
Epoch 60/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.3436 - loss: 0.0875 - val_accuracy: 0.3479 - val_loss: 0.0875
Epoch 61/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.3467 - loss: 0.0875 - val_accuracy: 0.3519 - val_loss: 0.0874
Epoch 62/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.3534 - loss: 0.0874 - val_accuracy: 0.3543 - val_loss: 0.0874
Epoch 63/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.3562 - loss: 0.0874 - val_accuracy: 0.3583 - val_loss: 0.0873
Epoch 64/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.3580 - loss: 0.0874 - val_accuracy: 0.3621 - val_loss: 0.0873
Epoch 65/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.3614 - loss: 0.0873 - val_accuracy: 0.3673 - val_loss: 0.0872
Epoch 66/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.3693 - loss: 0.0873 - val_accuracy: 0.3719 - val_loss: 0.0872
Epoch 67/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.3678 - loss: 0.0872 - val_accuracy: 0.3760 - val_loss: 0.0871
Epoch 68/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.3731 - loss: 0.0872 - val_accuracy: 0.3786 - val_loss: 0.0871
Epoch 69/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.3771 - loss: 0.0871 - val_accuracy: 0.3804 - val_loss: 0.0870
```

```
Epoch 70/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.3797 - loss: 0.0871 - val_accuracy: 0.3843 - val_loss: 0.0870
Epoch 71/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.3850 - loss: 0.0870 - val_accuracy: 0.3870 - val_loss: 0.0869
Epoch 72/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.3868 - loss: 0.0870 - val_accuracy: 0.3904 - val_loss: 0.0869
Epoch 73/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.3875 - loss: 0.0869 - val_accuracy: 0.3927 - val_loss: 0.0868
Epoch 74/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.3942 - loss: 0.0868 - val_accuracy: 0.3953 - val_loss: 0.0868
Epoch 75/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.3955 - loss: 0.0868 - val_accuracy: 0.3976 - val_loss: 0.0867
Epoch 76/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.4028 - loss: 0.0867 - val_accuracy: 0.4007 - val_loss: 0.0867
Epoch 77/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.4032 - loss: 0.0867 - val_accuracy: 0.4047 - val_loss: 0.0866
Epoch 78/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.4009 - loss: 0.0867 - val_accuracy: 0.4086 - val_loss: 0.0866
Epoch 79/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.4073 - loss: 0.0866 - val_accuracy: 0.4101 - val_loss: 0.0865
Epoch 80/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.4119 - loss: 0.0865 - val_accuracy: 0.4134 - val_loss: 0.0865
Epoch 81/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.4159 - loss: 0.0865 - val_accuracy: 0.4177 - val_loss: 0.0864
Epoch 82/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.4220 - loss: 0.0864 - val_accuracy: 0.4230 - val_loss: 0.0864
Epoch 83/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.4239 - loss: 0.0864 - val_accuracy: 0.4279 - val_loss: 0.0863
Epoch 84/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.4275 - loss: 0.0863 - val_accuracy: 0.4327 - val_loss: 0.0863
Epoch 85/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.4327 - loss: 0.0863 - val_accuracy: 0.4393 - val_loss: 0.0862
Epoch 86/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.4360 - loss: 0.0862 - val_accuracy: 0.4453 - val_loss: 0.0861
Epoch 87/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.4438 - loss: 0.0861 - val_accuracy: 0.4497 - val_loss: 0.0861
Epoch 88/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.4479 - loss: 0.0861 - val_accuracy: 0.4561 - val_loss: 0.0860
Epoch 89/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.4532 - loss: 0.0860 - val_accuracy: 0.4603 - val_loss: 0.0860
Epoch 90/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.4593 - loss: 0.0859 - val_accuracy: 0.4640 - val_loss: 0.0859
Epoch 91/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.4649 - loss: 0.0859 - val_accuracy: 0.4691 - val_loss: 0.0858
Epoch 92/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.4671 - loss: 0.0858 - val_accuracy: 0.4716 - val_loss: 0.0858
```

```
Epoch 93/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.4722 - loss: 0.0858 - val_accuracy: 0.4760 - val_loss: 0.0857
Epoch 94/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.4746 - loss: 0.0857 - val_accuracy: 0.4809 - val_loss: 0.0856
Epoch 95/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.4799 - loss: 0.0856 - val_accuracy: 0.4861 - val_loss: 0.0856
Epoch 96/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.4830 - loss: 0.0856 - val_accuracy: 0.4913 - val_loss: 0.0855
Epoch 97/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.4902 - loss: 0.0855 - val_accuracy: 0.4930 - val_loss: 0.0854
Epoch 98/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.4946 - loss: 0.0854 - val_accuracy: 0.4994 - val_loss: 0.0854
Epoch 99/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.5003 - loss: 0.0853 - val_accuracy: 0.5026 - val_loss: 0.0853
Epoch 100/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.4970 - loss: 0.0853 - val_accuracy: 0.5039 - val_loss: 0.0852
219/219 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.5047 - loss: 0.0853
Test loss=0.08531524986028671 Test accuracy = 0.4977142810821533
Time required for training the model is 428.50232338905334 seconds
```





In [45]:
```
## Model 2 --> SGD, lr =0.001, Mean Squared Error loss, hidden_neurons=[16,32]

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

model, history,duration = train_model(
    activation_func=activation_function,
    hidden_neurons=hidden_neurons,
    optimizer=SGD,
    loss_func=MeanSquaredError,
    epochs=100,
    dropout_rate=None,
    learning_rate=0.001
)

test_loss, test_acc = model.evaluate(X_test,y_test)

print(f"Test loss={test_loss} Test accuracy = {test_acc}")
```

```python
print(f"Time required to train the model is {duration} seconds")

result_df.loc[len(result_df.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    'SGD',
    duration,
    test_loss,
    test_acc
]

plot_metrics(history, activation_function, hidden_neurons)
```

Model: "sequential_17"

Layer (type)	Output Shape	Param #
flatten_17 (Flatten)	(None, 1024)	0
dense_58 (Dense)	(None, 16)	16,400
dense_59 (Dense)	(None, 32)	544
dense_60 (Dense)	(None, 10)	330

Total params: 17,274 (67.48 KB)

Trainable params: 17,274 (67.48 KB)

Non-trainable params: 0 (0.00 B)

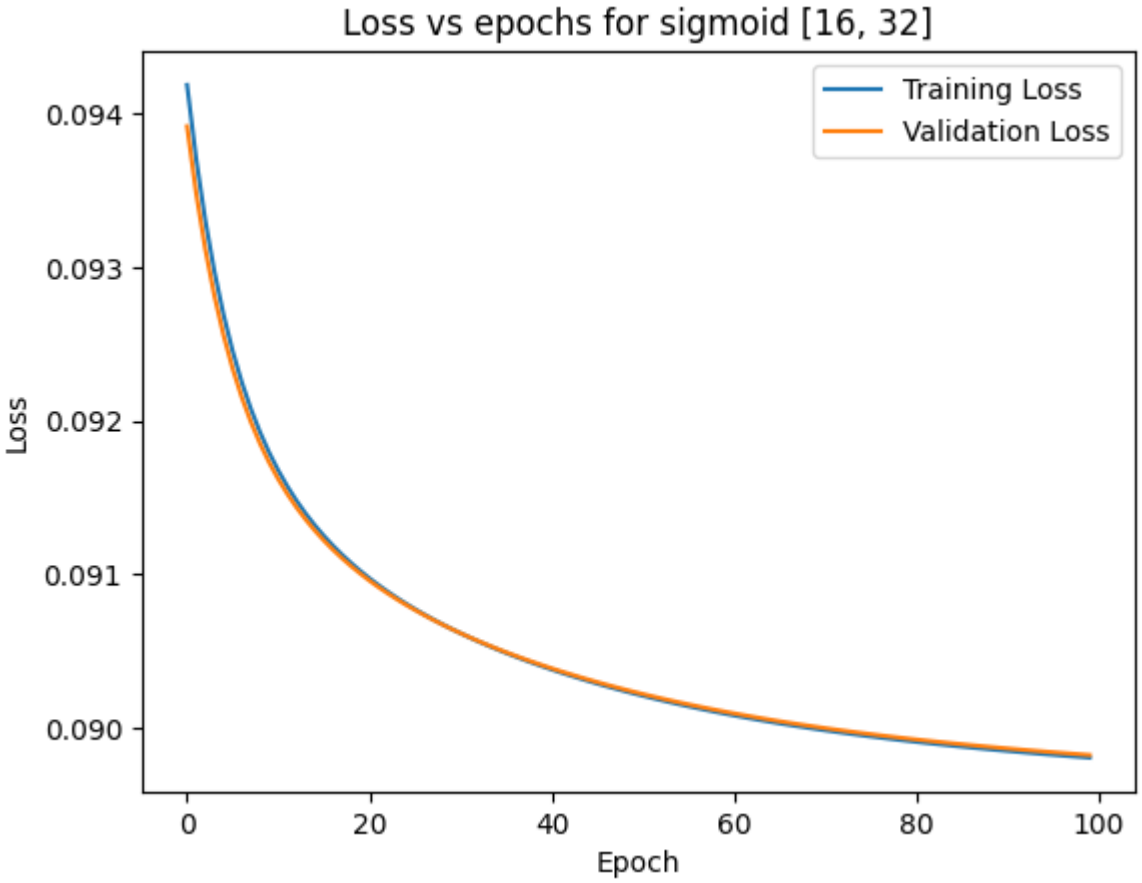
```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1012 - loss: 0.0943 - val_accuracy: 0.1037 - val_loss: 0.0939
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1008 - loss: 0.0938 - val_accuracy: 0.1056 - val_loss: 0.0935
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1024 - loss: 0.0934 - val_accuracy: 0.1083 - val_loss: 0.0931
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1018 - loss: 0.0930 - val_accuracy: 0.1090 - val_loss: 0.0928
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1051 - loss: 0.0928 - val_accuracy: 0.1101 - val_loss: 0.0926
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1031 - loss: 0.0925 - val_accuracy: 0.1100 - val_loss: 0.0923
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1040 - loss: 0.0923 - val_accuracy: 0.1113 - val_loss: 0.0922
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1077 - loss: 0.0921 - val_accuracy: 0.1114 - val_loss: 0.0920
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1100 - loss: 0.0920 - val_accuracy: 0.1127 - val_loss: 0.0919
Epoch 10/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1125 - loss: 0.0918 - val_accuracy: 0.1136 - val_loss: 0.0917
Epoch 11/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1102 - loss: 0.0917 - val_accuracy: 0.1127 - val_loss: 0.0916
Epoch 12/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1138 - loss: 0.0915 - val_accuracy: 0.1157 - val_loss: 0.0915
Epoch 13/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1153 - loss: 0.0915 - val_accuracy: 0.1153 - val_loss: 0.0914
Epoch 14/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1155 - loss: 0.0914 - val_accuracy: 0.1153 - val_loss: 0.0914
Epoch 15/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1160 - loss: 0.0913 - val_accuracy: 0.1166 - val_loss: 0.0913
Epoch 16/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1132 - loss: 0.0913 - val_accuracy: 0.1173 - val_loss: 0.0912
Epoch 17/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1177 - loss: 0.0912 - val_accuracy: 0.1174 - val_loss: 0.0912
Epoch 18/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1173 - loss: 0.0912 - val_accuracy: 0.1173 - val_loss: 0.0911
Epoch 19/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1166 - loss: 0.0911 - val_accuracy: 0.1173 - val_loss: 0.0911
Epoch 20/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1187 - loss: 0.0911 - val_accuracy: 0.1177 - val_loss: 0.0910
Epoch 21/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1224 - loss: 0.0910 - val_accuracy: 0.1187 - val_loss: 0.0910
Epoch 22/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1217 - loss: 0.0909 - val_accuracy: 0.1190 - val_loss: 0.0909
Epoch 23/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1234 - loss: 0.0909 - val_accuracy: 0.1189 - val_loss: 0.0909
```

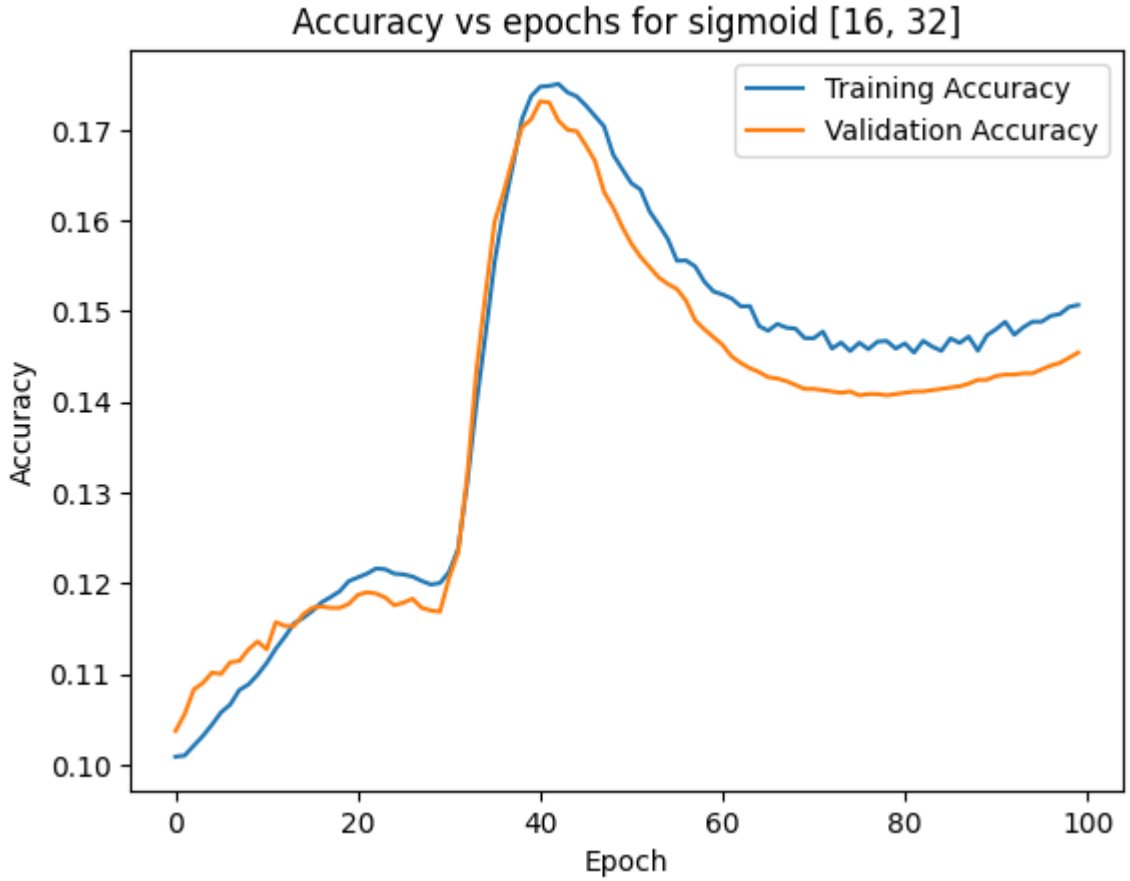
```
Epoch 24/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1208 - loss: 0.0909 - val_accuracy: 0.1184 - val_loss: 0.0908
Epoch 25/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1208 - loss: 0.0908 - val_accuracy: 0.1176 - val_loss: 0.0908
Epoch 26/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1193 - loss: 0.0908 - val_accuracy: 0.1179 - val_loss: 0.0908
Epoch 27/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1221 - loss: 0.0907 - val_accuracy: 0.1183 - val_loss: 0.0907
Epoch 28/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1205 - loss: 0.0907 - val_accuracy: 0.1173 - val_loss: 0.0907
Epoch 29/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1208 - loss: 0.0907 - val_accuracy: 0.1170 - val_loss: 0.0907
Epoch 30/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1210 - loss: 0.0906 - val_accuracy: 0.1169 - val_loss: 0.0906
Epoch 31/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1190 - loss: 0.0906 - val_accuracy: 0.1206 - val_loss: 0.0906
Epoch 32/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1248 - loss: 0.0906 - val_accuracy: 0.1234 - val_loss: 0.0906
Epoch 33/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1276 - loss: 0.0906 - val_accuracy: 0.1319 - val_loss: 0.0906
Epoch 34/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1381 - loss: 0.0905 - val_accuracy: 0.1434 - val_loss: 0.0905
Epoch 35/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1455 - loss: 0.0905 - val_accuracy: 0.1519 - val_loss: 0.0905
Epoch 36/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1531 - loss: 0.0905 - val_accuracy: 0.1599 - val_loss: 0.0905
Epoch 37/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1586 - loss: 0.0905 - val_accuracy: 0.1631 - val_loss: 0.0905
Epoch 38/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1656 - loss: 0.0905 - val_accuracy: 0.1669 - val_loss: 0.0905
Epoch 39/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1707 - loss: 0.0904 - val_accuracy: 0.1703 - val_loss: 0.0904
Epoch 40/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1722 - loss: 0.0904 - val_accuracy: 0.1711 - val_loss: 0.0904
Epoch 41/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1732 - loss: 0.0904 - val_accuracy: 0.1731 - val_loss: 0.0904
Epoch 42/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1757 - loss: 0.0904 - val_accuracy: 0.1730 - val_loss: 0.0904
Epoch 43/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1745 - loss: 0.0903 - val_accuracy: 0.1710 - val_loss: 0.0904
Epoch 44/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1729 - loss: 0.0903 - val_accuracy: 0.1700 - val_loss: 0.0903
Epoch 45/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1720 - loss: 0.0903 - val_accuracy: 0.1699 - val_loss: 0.0903
Epoch 46/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1741 - loss: 0.0903 - val_accuracy: 0.1683 - val_loss: 0.0903
```

```
Epoch 47/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1725 - loss: 0.0903 - val_accuracy: 0.1666 - val_loss: 0.0903
Epoch 48/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1736 - loss: 0.0903 - val_accuracy: 0.1631 - val_loss: 0.0903
Epoch 49/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1658 - loss: 0.0903 - val_accuracy: 0.1614 - val_loss: 0.0903
Epoch 50/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1678 - loss: 0.0902 - val_accuracy: 0.1593 - val_loss: 0.0902
Epoch 51/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1649 - loss: 0.0902 - val_accuracy: 0.1574 - val_loss: 0.0902
Epoch 52/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1649 - loss: 0.0902 - val_accuracy: 0.1560 - val_loss: 0.0902
Epoch 53/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1642 - loss: 0.0902 - val_accuracy: 0.1549 - val_loss: 0.0902
Epoch 54/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1584 - loss: 0.0902 - val_accuracy: 0.1537 - val_loss: 0.0902
Epoch 55/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1581 - loss: 0.0901 - val_accuracy: 0.1530 - val_loss: 0.0902
Epoch 56/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1586 - loss: 0.0901 - val_accuracy: 0.1524 - val_loss: 0.0902
Epoch 57/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1563 - loss: 0.0901 - val_accuracy: 0.1511 - val_loss: 0.0901
Epoch 58/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1547 - loss: 0.0901 - val_accuracy: 0.1490 - val_loss: 0.0901
Epoch 59/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1541 - loss: 0.0901 - val_accuracy: 0.1480 - val_loss: 0.0901
Epoch 60/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1532 - loss: 0.0901 - val_accuracy: 0.1471 - val_loss: 0.0901
Epoch 61/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1519 - loss: 0.0901 - val_accuracy: 0.1463 - val_loss: 0.0901
Epoch 62/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1496 - loss: 0.0901 - val_accuracy: 0.1450 - val_loss: 0.0901
Epoch 63/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1506 - loss: 0.0901 - val_accuracy: 0.1443 - val_loss: 0.0901
Epoch 64/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1502 - loss: 0.0900 - val_accuracy: 0.1437 - val_loss: 0.0901
Epoch 65/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1504 - loss: 0.0901 - val_accuracy: 0.1433 - val_loss: 0.0901
Epoch 66/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1466 - loss: 0.0901 - val_accuracy: 0.1427 - val_loss: 0.0900
Epoch 67/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1496 - loss: 0.0900 - val_accuracy: 0.1426 - val_loss: 0.0900
Epoch 68/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1482 - loss: 0.0900 - val_accuracy: 0.1423 - val_loss: 0.0900
Epoch 69/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1465 - loss: 0.0900 - val_accuracy: 0.1419 - val_loss: 0.0900
```

```
Epoch 70/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1482 - loss: 0.0900 - val_accuracy: 0.1414 - val_loss: 0.0900
Epoch 71/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1502 - loss: 0.0900 - val_accuracy: 0.1414 - val_loss: 0.0900
Epoch 72/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1443 - loss: 0.0900 - val_accuracy: 0.1413 - val_loss: 0.0900
Epoch 73/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1460 - loss: 0.0900 - val_accuracy: 0.1411 - val_loss: 0.0900
Epoch 74/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1463 - loss: 0.0900 - val_accuracy: 0.1410 - val_loss: 0.0900
Epoch 75/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1469 - loss: 0.0899 - val_accuracy: 0.1411 - val_loss: 0.0900
Epoch 76/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1477 - loss: 0.0899 - val_accuracy: 0.1407 - val_loss: 0.0900
Epoch 77/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1475 - loss: 0.0899 - val_accuracy: 0.1409 - val_loss: 0.0900
Epoch 78/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1462 - loss: 0.0899 - val_accuracy: 0.1409 - val_loss: 0.0899
Epoch 79/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1428 - loss: 0.0899 - val_accuracy: 0.1407 - val_loss: 0.0899
Epoch 80/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1474 - loss: 0.0899 - val_accuracy: 0.1409 - val_loss: 0.0899
Epoch 81/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1448 - loss: 0.0899 - val_accuracy: 0.1410 - val_loss: 0.0899
Epoch 82/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1472 - loss: 0.0899 - val_accuracy: 0.1411 - val_loss: 0.0899
Epoch 83/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1471 - loss: 0.0899 - val_accuracy: 0.1411 - val_loss: 0.0899
Epoch 84/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1472 - loss: 0.0899 - val_accuracy: 0.1413 - val_loss: 0.0899
Epoch 85/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1447 - loss: 0.0899 - val_accuracy: 0.1414 - val_loss: 0.0899
Epoch 86/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1454 - loss: 0.0899 - val_accuracy: 0.1416 - val_loss: 0.0899
Epoch 87/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1473 - loss: 0.0899 - val_accuracy: 0.1417 - val_loss: 0.0899
Epoch 88/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1474 - loss: 0.0899 - val_accuracy: 0.1420 - val_loss: 0.0899
Epoch 89/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1456 - loss: 0.0899 - val_accuracy: 0.1424 - val_loss: 0.0899
Epoch 90/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1450 - loss: 0.0899 - val_accuracy: 0.1424 - val_loss: 0.0899
Epoch 91/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1484 - loss: 0.0899 - val_accuracy: 0.1429 - val_loss: 0.0899
Epoch 92/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1486 - loss: 0.0898 - val_accuracy: 0.1430 - val_loss: 0.0899
```

```
Epoch 93/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1480 - loss: 0.0898 - val_accuracy: 0.1430 - val_loss: 0.0899
Epoch 94/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1499 - loss: 0.0898 - val_accuracy: 0.1431 - val_loss: 0.0899
Epoch 95/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1469 - loss: 0.0898 - val_accuracy: 0.1431 - val_loss: 0.0898
Epoch 96/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1476 - loss: 0.0898 - val_accuracy: 0.1436 - val_loss: 0.0898
Epoch 97/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1464 - loss: 0.0898 - val_accuracy: 0.1440 - val_loss: 0.0898
Epoch 98/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1494 - loss: 0.0898 - val_accuracy: 0.1443 - val_loss: 0.0898
Epoch 99/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1501 - loss: 0.0898 - val_accuracy: 0.1449 - val_loss: 0.0898
Epoch 100/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1498 - loss: 0.0898 - val_accuracy: 0.1454 - val_loss: 0.0898
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.1576 - loss: 0.0899
Test loss=0.08983103930950165 Test accuracy = 0.15728572010993958
Time required to train the model is 436.4306344985962 seconds
```





In [46]:
```
## Model 3 --> SGD, lr =0.001, Mean Squared Error loss, hidden_neurons=[16,32,64]

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

model, history, duration = train_model(
    activation_func=activation_function,
    hidden_neurons=hidden_neurons,
    optimizer=SGD,
    loss_func=MeanSquaredError,
    epochs=100,
    dropout_rate=None,
    learning_rate=0.001
)

test_loss, test_acc = model.evaluate(X_test,y_test)

print(f"Test loss={test_loss} Test accuracy = {test_acc}")
```

```python
print(f"Time required to train the model is {duration} seconds")

result_df.loc[len(result_df.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    'SGD',
    duration,
    test_loss,
    test_acc
]

plot_metrics(history, activation_function, hidden_neurons)
```

Model: "sequential_18"

Layer (type)	Output Shape	Param #
flatten_18 (Flatten)	(None, 1024)	0
dense_61 (Dense)	(None, 16)	16,400
dense_62 (Dense)	(None, 32)	544
dense_63 (Dense)	(None, 64)	2,112
dense_64 (Dense)	(None, 10)	650

Total params: 19,706 (76.98 KB)

Trainable params: 19,706 (76.98 KB)

Non-trainable params: 0 (0.00 B)

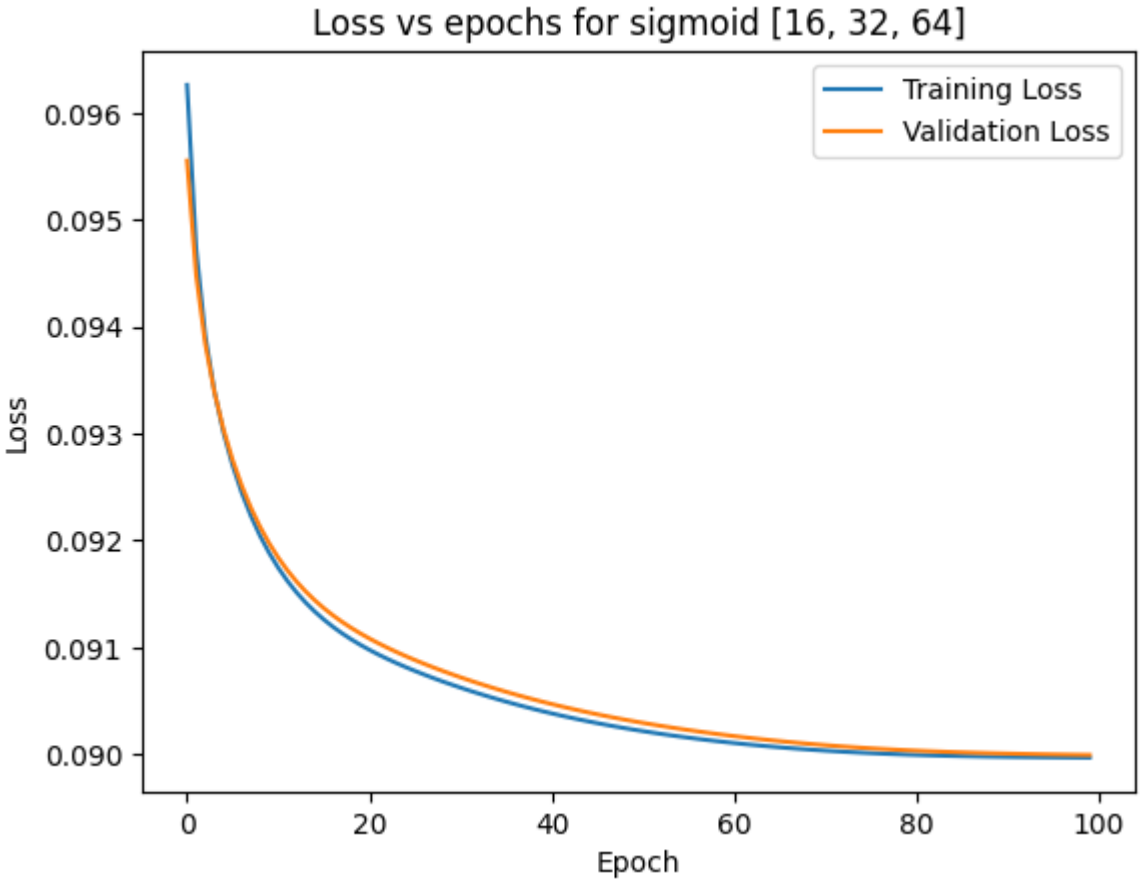
```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.0966 - loss: 0.0969 - val_accuracy: 0.0943 - val_loss: 0.0956
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.0975 - loss: 0.0950 - val_accuracy: 0.0943 - val_loss: 0.0945
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.0970 - loss: 0.0942 - val_accuracy: 0.0943 - val_loss: 0.0938
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.0984 - loss: 0.0935 - val_accuracy: 0.0943 - val_loss: 0.0934
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.0977 - loss: 0.0931 - val_accuracy: 0.0943 - val_loss: 0.0930
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.0994 - loss: 0.0927 - val_accuracy: 0.0943 - val_loss: 0.0928
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.0968 - loss: 0.0926 - val_accuracy: 0.0943 - val_loss: 0.0925
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.0976 - loss: 0.0923 - val_accuracy: 0.0943 - val_loss: 0.0923
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.0958 - loss: 0.0921 - val_accuracy: 0.0943 - val_loss: 0.0921
Epoch 10/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.0996 - loss: 0.0919 - val_accuracy: 0.0943 - val_loss: 0.0920
Epoch 11/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.0963 - loss: 0.0918 - val_accuracy: 0.0943 - val_loss: 0.0918
Epoch 12/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.0965 - loss: 0.0917 - val_accuracy: 0.0943 - val_loss: 0.0917
Epoch 13/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.0983 - loss: 0.0915 - val_accuracy: 0.0943 - val_loss: 0.0916
Epoch 14/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.0971 - loss: 0.0914 - val_accuracy: 0.0943 - val_loss: 0.0915
Epoch 15/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.0983 - loss: 0.0914 - val_accuracy: 0.0943 - val_loss: 0.0914
Epoch 16/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.0978 - loss: 0.0913 - val_accuracy: 0.0937 - val_loss: 0.0914
Epoch 17/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.0972 - loss: 0.0912 - val_accuracy: 0.0859 - val_loss: 0.0913
Epoch 18/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.0926 - loss: 0.0911 - val_accuracy: 0.0716 - val_loss: 0.0912
Epoch 19/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.0768 - loss: 0.0911 - val_accuracy: 0.0699 - val_loss: 0.0912
Epoch 20/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.0768 - loss: 0.0910 - val_accuracy: 0.0866 - val_loss: 0.0911
Epoch 21/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.0925 - loss: 0.0910 - val_accuracy: 0.1006 - val_loss: 0.0911
Epoch 22/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1068 - loss: 0.0909 - val_accuracy: 0.1074 - val_loss: 0.0910
Epoch 23/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1093 - loss: 0.0909 - val_accuracy: 0.1083 - val_loss: 0.0910
```

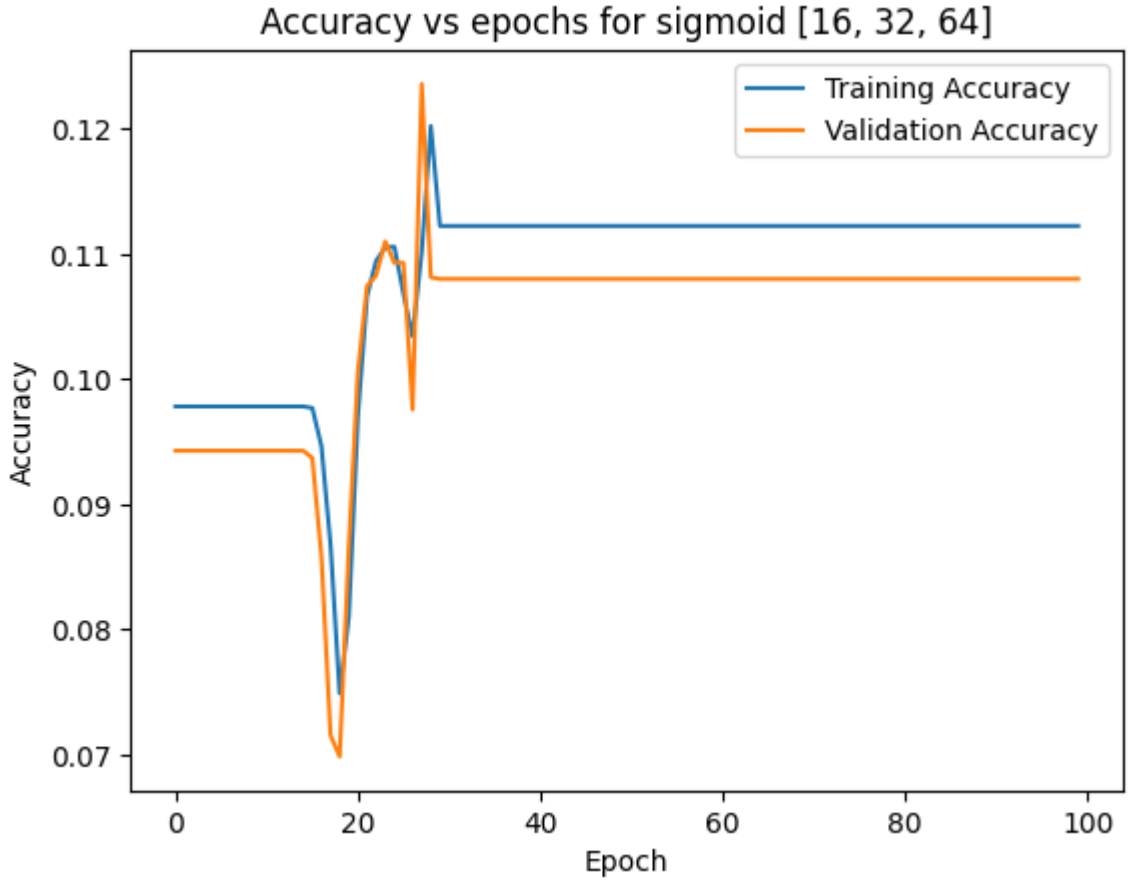
```
Epoch 24/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1109 - loss: 0.0908 - val_accuracy: 0.1110 - val_loss: 0.0910
Epoch 25/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1110 - loss: 0.0908 - val_accuracy: 0.1093 - val_loss: 0.0909
Epoch 26/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1067 - loss: 0.0908 - val_accuracy: 0.1093 - val_loss: 0.0909
Epoch 27/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1055 - loss: 0.0908 - val_accuracy: 0.0976 - val_loss: 0.0908
Epoch 28/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1018 - loss: 0.0908 - val_accuracy: 0.1236 - val_loss: 0.0908
Epoch 29/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1233 - loss: 0.0907 - val_accuracy: 0.1081 - val_loss: 0.0908
Epoch 30/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1122 - loss: 0.0907 - val_accuracy: 0.1080 - val_loss: 0.0907
Epoch 31/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1134 - loss: 0.0906 - val_accuracy: 0.1080 - val_loss: 0.0907
Epoch 32/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1108 - loss: 0.0906 - val_accuracy: 0.1080 - val_loss: 0.0907
Epoch 33/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1124 - loss: 0.0906 - val_accuracy: 0.1080 - val_loss: 0.0907
Epoch 34/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1127 - loss: 0.0905 - val_accuracy: 0.1080 - val_loss: 0.0906
Epoch 35/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.1131 - loss: 0.0905 - val_accuracy: 0.1080 - val_loss: 0.0906
Epoch 36/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1138 - loss: 0.0905 - val_accuracy: 0.1080 - val_loss: 0.0906
Epoch 37/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1118 - loss: 0.0905 - val_accuracy: 0.1080 - val_loss: 0.0906
Epoch 38/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1120 - loss: 0.0905 - val_accuracy: 0.1080 - val_loss: 0.0905
Epoch 39/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1139 - loss: 0.0904 - val_accuracy: 0.1080 - val_loss: 0.0905
Epoch 40/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1110 - loss: 0.0904 - val_accuracy: 0.1080 - val_loss: 0.0905
Epoch 41/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1128 - loss: 0.0904 - val_accuracy: 0.1080 - val_loss: 0.0905
Epoch 42/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1124 - loss: 0.0904 - val_accuracy: 0.1080 - val_loss: 0.0904
Epoch 43/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1116 - loss: 0.0903 - val_accuracy: 0.1080 - val_loss: 0.0904
Epoch 44/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1121 - loss: 0.0903 - val_accuracy: 0.1080 - val_loss: 0.0904
Epoch 45/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1102 - loss: 0.0903 - val_accuracy: 0.1080 - val_loss: 0.0904
Epoch 46/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1124 - loss: 0.0903 - val_accuracy: 0.1080 - val_loss: 0.0904
```

```
Epoch 47/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1131 - loss: 0.0903 - val_accuracy: 0.1080 - val_loss: 0.0904
Epoch 48/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1124 - loss: 0.0903 - val_accuracy: 0.1080 - val_loss: 0.0903
Epoch 49/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1151 - loss: 0.0902 - val_accuracy: 0.1080 - val_loss: 0.0903
Epoch 50/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1123 - loss: 0.0902 - val_accuracy: 0.1080 - val_loss: 0.0903
Epoch 51/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1122 - loss: 0.0902 - val_accuracy: 0.1080 - val_loss: 0.0903
Epoch 52/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1121 - loss: 0.0902 - val_accuracy: 0.1080 - val_loss: 0.0903
Epoch 53/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1141 - loss: 0.0902 - val_accuracy: 0.1080 - val_loss: 0.0903
Epoch 54/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1122 - loss: 0.0902 - val_accuracy: 0.1080 - val_loss: 0.0903
Epoch 55/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1114 - loss: 0.0902 - val_accuracy: 0.1080 - val_loss: 0.0902
Epoch 56/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1150 - loss: 0.0901 - val_accuracy: 0.1080 - val_loss: 0.0902
Epoch 57/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1119 - loss: 0.0901 - val_accuracy: 0.1080 - val_loss: 0.0902
Epoch 58/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1106 - loss: 0.0901 - val_accuracy: 0.1080 - val_loss: 0.0902
Epoch 59/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1115 - loss: 0.0901 - val_accuracy: 0.1080 - val_loss: 0.0902
Epoch 60/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1117 - loss: 0.0901 - val_accuracy: 0.1080 - val_loss: 0.0902
Epoch 61/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1135 - loss: 0.0901 - val_accuracy: 0.1080 - val_loss: 0.0902
Epoch 62/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1120 - loss: 0.0901 - val_accuracy: 0.1080 - val_loss: 0.0902
Epoch 63/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1120 - loss: 0.0901 - val_accuracy: 0.1080 - val_loss: 0.0901
Epoch 64/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1110 - loss: 0.0901 - val_accuracy: 0.1080 - val_loss: 0.0901
Epoch 65/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1106 - loss: 0.0901 - val_accuracy: 0.1080 - val_loss: 0.0901
Epoch 66/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1128 - loss: 0.0901 - val_accuracy: 0.1080 - val_loss: 0.0901
Epoch 67/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1154 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0901
Epoch 68/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1122 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0901
Epoch 69/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1148 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0901
```

```
Epoch 70/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1139 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0901
Epoch 71/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1120 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0901
Epoch 72/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1125 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0901
Epoch 73/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1117 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0901
Epoch 74/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1126 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0901
Epoch 75/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1108 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0901
Epoch 76/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1087 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0901
Epoch 77/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1118 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 78/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1142 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 79/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1159 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 80/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1110 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 81/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1115 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 82/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1125 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 83/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1129 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 84/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1104 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 85/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1142 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 86/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1105 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 87/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1118 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 88/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1121 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 89/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1131 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 90/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1123 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 91/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1127 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 92/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1136 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
```

```
Epoch 93/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1110 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 94/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1131 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 95/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1130 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 96/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1139 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 97/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1094 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 98/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.1114 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 99/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1119 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
Epoch 100/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.1128 - loss: 0.0900 - val_accuracy: 0.1080 - val_loss: 0.0900
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.1219 - loss: 0.0900
Test loss=0.08996734023094177 Test accuracy = 0.11942857503890991
Time required to train the model is 428.4792711734772 seconds
```





In [47]:
```
## Model 4 --> Adam, lr =0.001, CCE loss, hidden_neurons=[16]

hidden_neurons = [16]
activation_function = 'sigmoid'

model, history,duration = train_model(
    activation_func=activation_function,
    hidden_neurons=hidden_neurons,
    optimizer=Adam,
    loss_func=CategoricalCrossentropy,
    epochs=100,
    dropout_rate=None,
    learning_rate=0.001
)

test_loss, test_acc = model.evaluate(X_test,y_test)

print(f"Test loss={test_loss} Test accuracy = {test_acc}")
```

```python
print(f"Time required to train the model is {duration} seconds")

result_df.loc[len(result_df.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    'Adam',
    duration,
    test_loss,
    test_acc
]

plot_metrics(history, activation_function, hidden_neurons)
```

Model: "sequential_19"

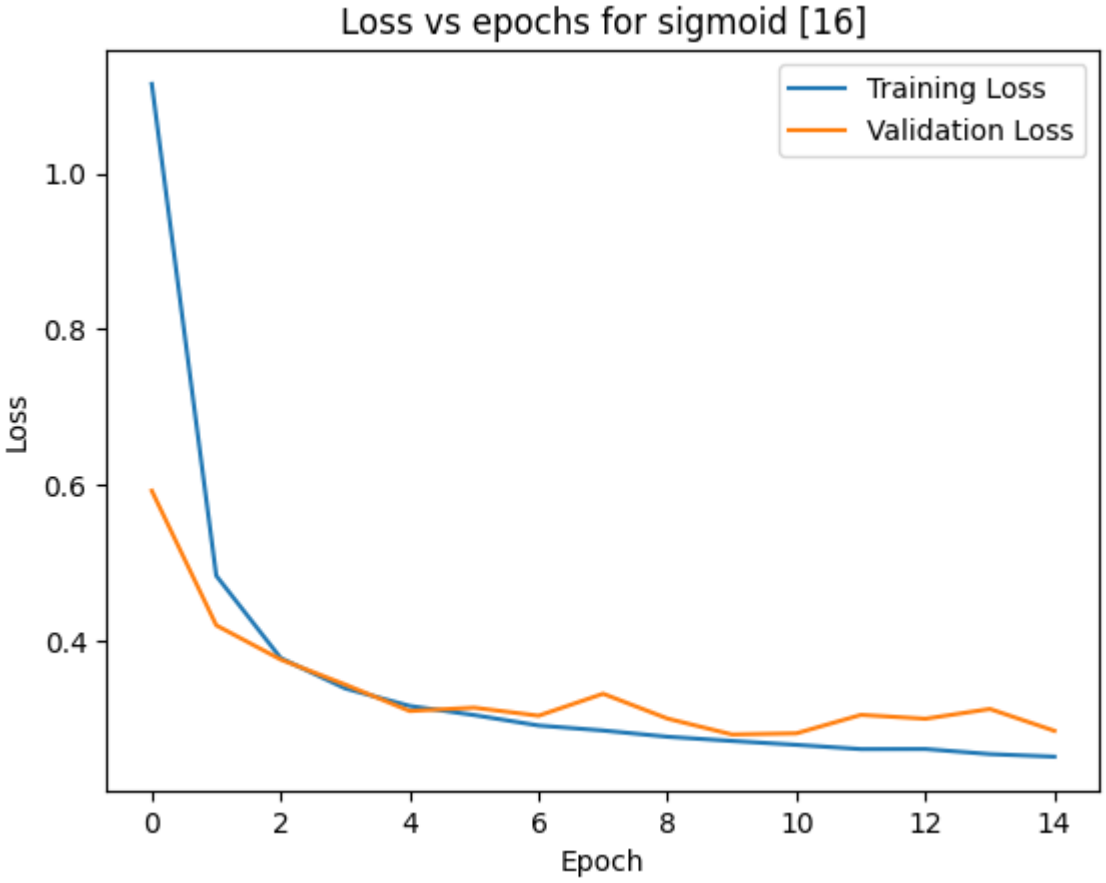
Layer (type)	Output Shape	Param #
flatten_19 (Flatten)	(None, 1024)	0
dense_65 (Dense)	(None, 16)	16,400
dense_66 (Dense)	(None, 10)	170

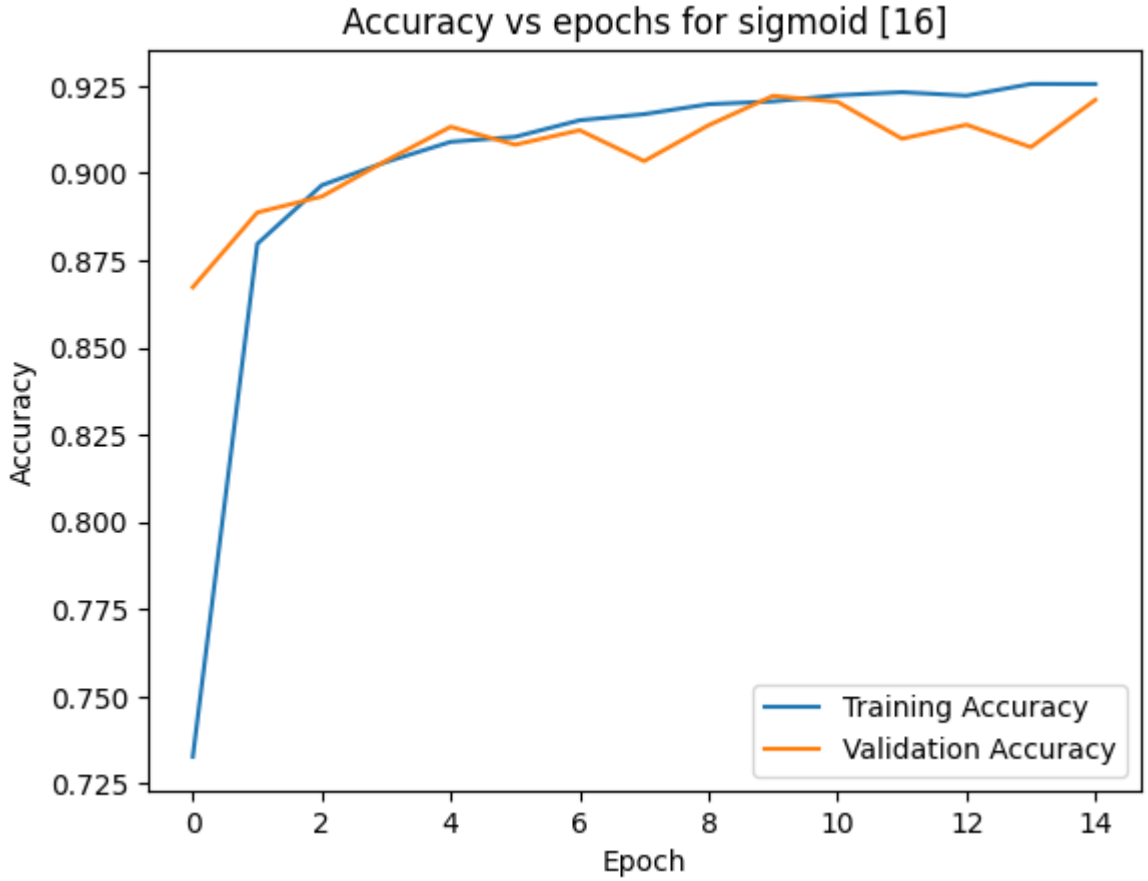
Total params: 16,570 (64.73 KB)

Trainable params: 16,570 (64.73 KB)

Non-trainable params: 0 (0.00 B)

```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.5717 - loss: 1.5646 - val_accuracy: 0.8673 - val_loss: 0.5924
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8753 - loss: 0.5274 - val_accuracy: 0.8887 - val_loss: 0.4196
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8940 - loss: 0.3933 - val_accuracy: 0.8933 - val_loss: 0.3756
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9047 - loss: 0.3397 - val_accuracy: 0.9036 - val_loss: 0.3434
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.9084 - loss: 0.3175 - val_accuracy: 0.9133 - val_loss: 0.3095
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9134 - loss: 0.2966 - val_accuracy: 0.9081 - val_loss: 0.3137
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9127 - loss: 0.2955 - val_accuracy: 0.9123 - val_loss: 0.3034
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9158 - loss: 0.2891 - val_accuracy: 0.9034 - val_loss: 0.3316
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9200 - loss: 0.2760 - val_accuracy: 0.9137 - val_loss: 0.2997
Epoch 10/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9220 - loss: 0.2679 - val_accuracy: 0.9221 - val_loss: 0.2793
Epoch 11/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9226 - loss: 0.2666 - val_accuracy: 0.9204 - val_loss: 0.2808
Epoch 12/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9230 - loss: 0.2583 - val_accuracy: 0.9099 - val_loss: 0.3045
Epoch 13/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9219 - loss: 0.2611 - val_accuracy: 0.9139 - val_loss: 0.2995
Epoch 14/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9259 - loss: 0.2533 - val_accuracy: 0.9074 - val_loss: 0.3121
Epoch 15/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9238 - loss: 0.2526 - val_accuracy: 0.9210 - val_loss: 0.2840
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9234 - loss: 0.2686
Test loss=0.28263649344444275 Test accuracy = 0.9197142720222473
Time required to train the model is 63.880293130874634 seconds
```


Loss vs epochs for sigmoid [16]
Training Loss
Validation Loss
Loss
1.0
0.8
0.6
0.4
0
2
4
6
8
10
12
14
Epoch



In [48]:

```
## Model 5 --> Adam, lr =0.001, CCE loss, hidden_neurons=[16,32]

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

model, history,duration = train_model(
    activation_func=activation_function,
    hidden_neurons=hidden_neurons,
    optimizer=Adam,
    loss_func=CategoricalCrossentropy,
    epochs=100,
    dropout_rate=None,
    learning_rate=0.001
)

test_loss, test_acc = model.evaluate(X_test,y_test)

print(f"Test loss={test_loss} Test accuracy = {test_acc}")
```

```python
print(f"Time taken to train the model is {duration} seconds")

result_df.loc[len(result_df.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    'Adam',
    duration,
    test_loss,
    test_acc
]

plot_metrics(history, activation_function, hidden_neurons)
```

Model: "sequential_20"

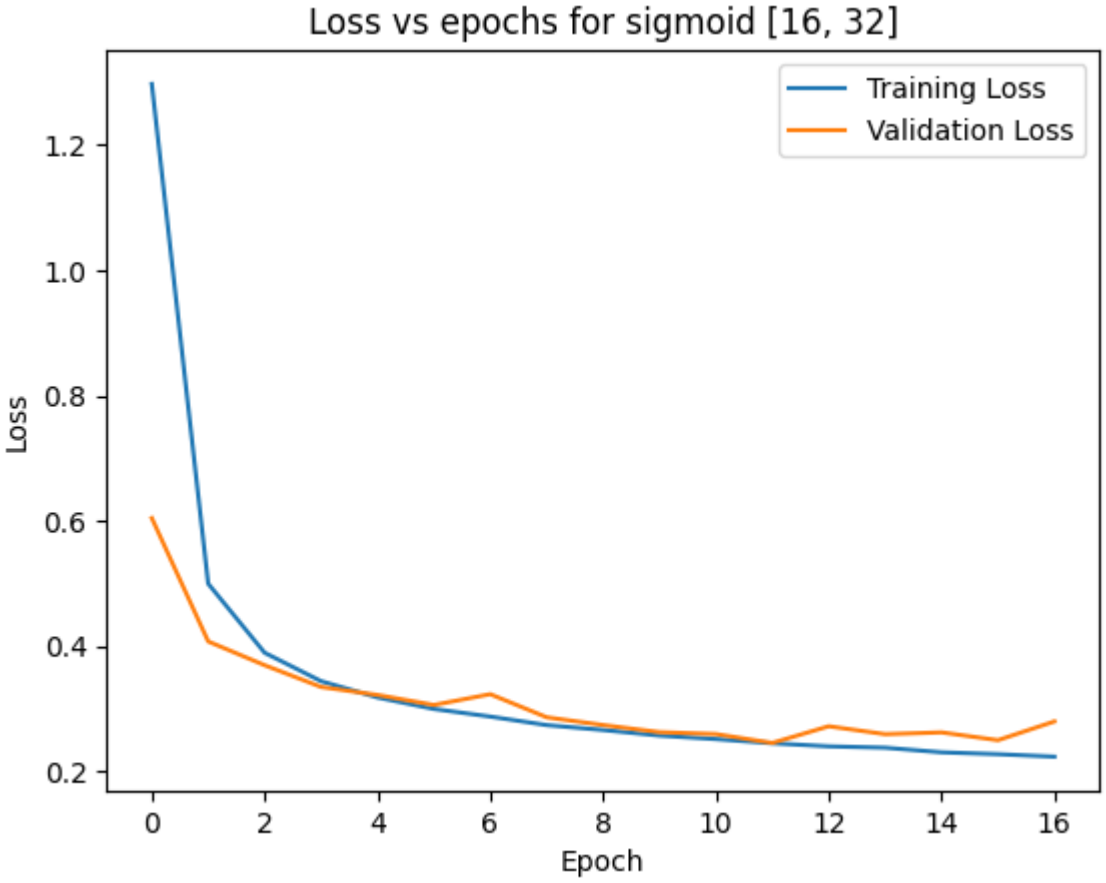
Layer (type)	Output Shape	Param #
flatten_20 (Flatten)	(None, 1024)	0
dense_67 (Dense)	(None, 16)	16,400
dense_68 (Dense)	(None, 32)	544
dense_69 (Dense)	(None, 10)	330

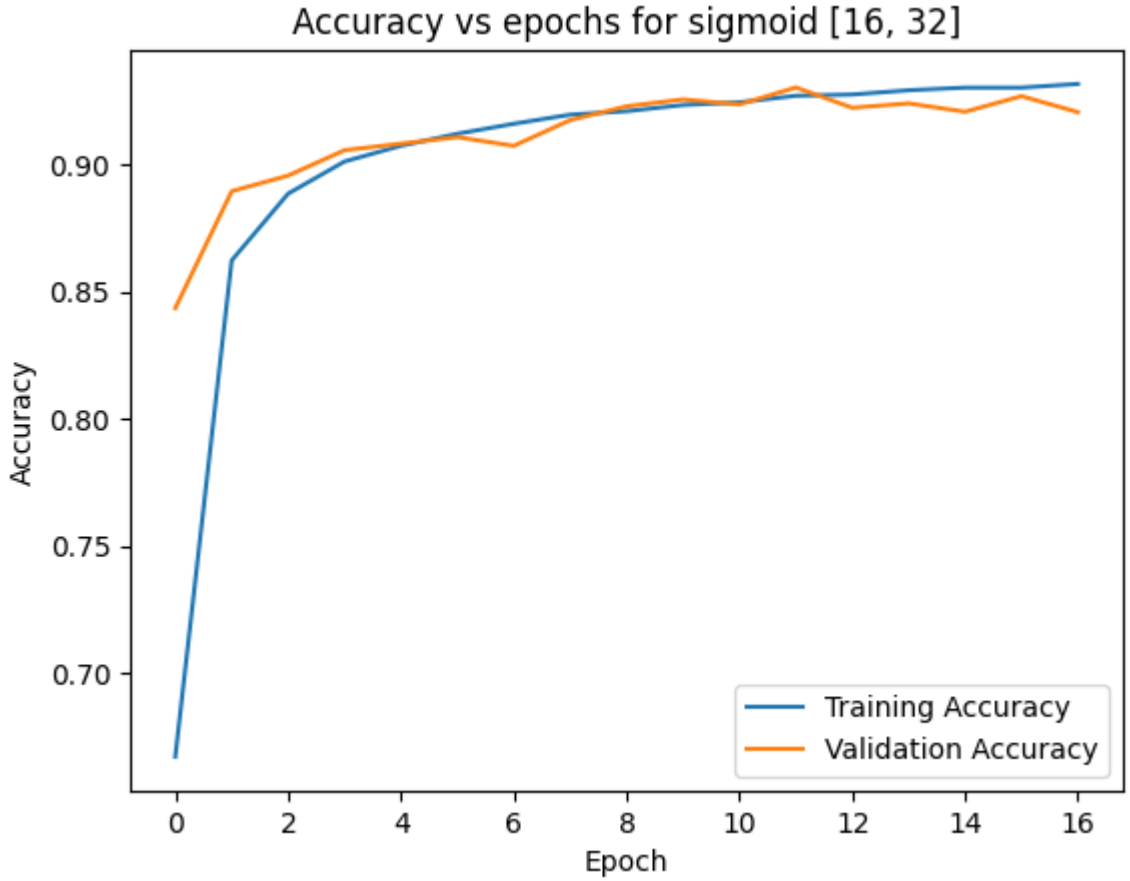
Total params: 17,274 (67.48 KB)

Trainable params: 17,274 (67.48 KB)

Non-trainable params: 0 (0.00 B)

```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.4717 - loss: 1.8169 - val_accuracy: 0.8436 - val_loss: 0.6044
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8507 - loss: 0.5469 - val_accuracy: 0.8896 - val_loss: 0.4074
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8866 - loss: 0.3982 - val_accuracy: 0.8957 - val_loss: 0.3695
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8988 - loss: 0.3551 - val_accuracy: 0.9057 - val_loss: 0.3348
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9074 - loss: 0.3195 - val_accuracy: 0.9083 - val_loss: 0.3220
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9120 - loss: 0.2994 - val_accuracy: 0.9109 - val_loss: 0.3060
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9147 - loss: 0.2884 - val_accuracy: 0.9074 - val_loss: 0.3233
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9195 - loss: 0.2772 - val_accuracy: 0.9176 - val_loss: 0.2864
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9199 - loss: 0.2670 - val_accuracy: 0.9230 - val_loss: 0.2740
Epoch 10/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.9239 - loss: 0.2550 - val_accuracy: 0.9257 - val_loss: 0.2627
Epoch 11/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9252 - loss: 0.2477 - val_accuracy: 0.9237 - val_loss: 0.2596
Epoch 12/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9281 - loss: 0.2439 - val_accuracy: 0.9304 - val_loss: 0.2455
Epoch 13/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9280 - loss: 0.2379 - val_accuracy: 0.9224 - val_loss: 0.2721
Epoch 14/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9296 - loss: 0.2420 - val_accuracy: 0.9241 - val_loss: 0.2594
Epoch 15/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9300 - loss: 0.2330 - val_accuracy: 0.9209 - val_loss: 0.2623
Epoch 16/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.9308 - loss: 0.2288 - val_accuracy: 0.9270 - val_loss: 0.2498
Epoch 17/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9336 - loss: 0.2185 - val_accuracy: 0.9207 - val_loss: 0.2798
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9319 - loss: 0.2338
Test loss=0.24552959203720093 Test accuracy = 0.9307143092155457
Time taken to train the model is 73.31176495552063 seconds
```


Loss vs epochs for sigmoid [16, 32]
Training Loss
Validation Loss
Loss
Epoch
1.2
1.0
0.8
0.6
0.4
0.2
0
2
4
6
8
10
12
14
16



In [49]:
```
## Model 6 --> Adam, lr =0.001, CCE loss, hidden_neurons=[16,32,64]

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

model, history, duration = train_model(
    activation_func=activation_function,
    hidden_neurons=hidden_neurons,
    optimizer=Adam,
    loss_func=CategoricalCrossentropy,
    epochs=100,
    dropout_rate=None,
    learning_rate=0.001
)

test_loss, test_acc = model.evaluate(X_test,y_test)

print(f"Test loss={test_loss} Test accuracy = {test_acc}")
```

```python
print(f"Time taken to train the model is {duration} seconds")

result_df.loc[len(result_df.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    'Adam',
    duration,
    test_loss,
    test_acc
]

plot_metrics(history, activation_function, hidden_neurons)
```

Model: "sequential_21"

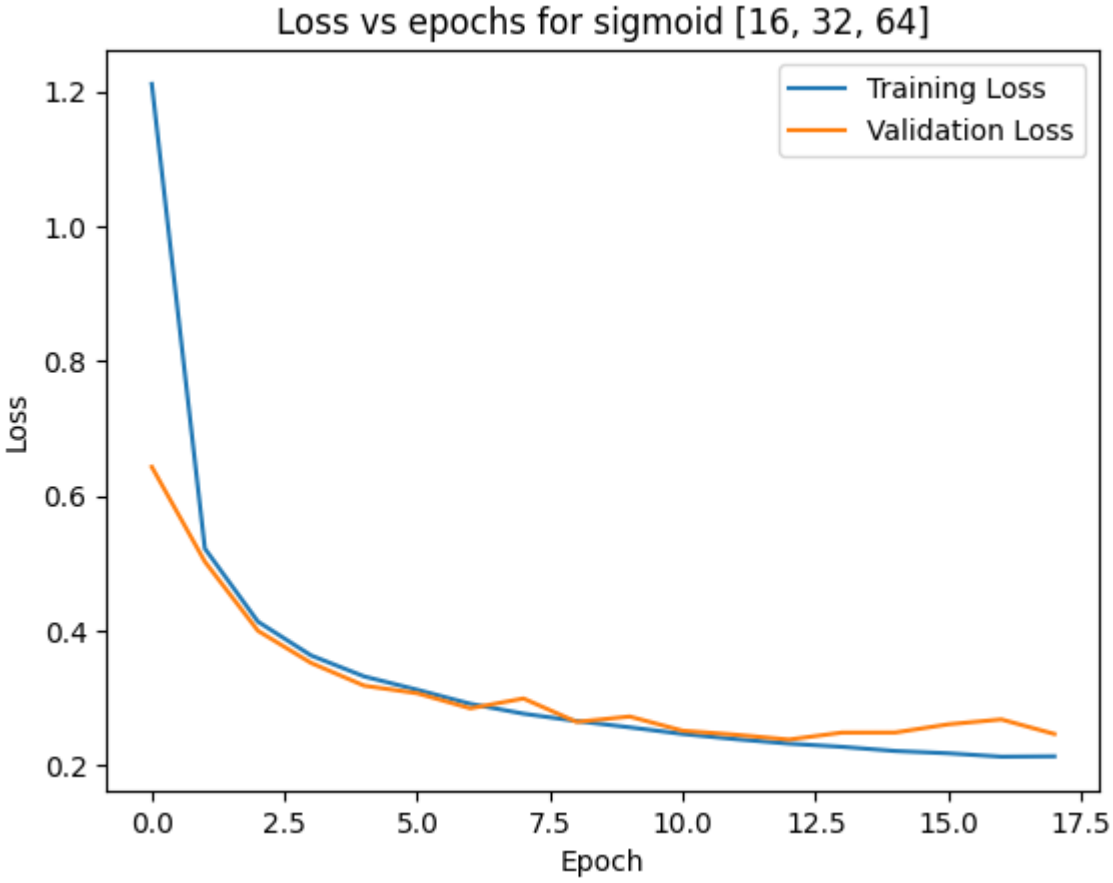
Layer (type)	Output Shape	Param #
flatten_21 (Flatten)	(None, 1024)	0
dense_70 (Dense)	(None, 16)	16,400
dense_71 (Dense)	(None, 32)	544
dense_72 (Dense)	(None, 64)	2,112
dense_73 (Dense)	(None, 10)	650

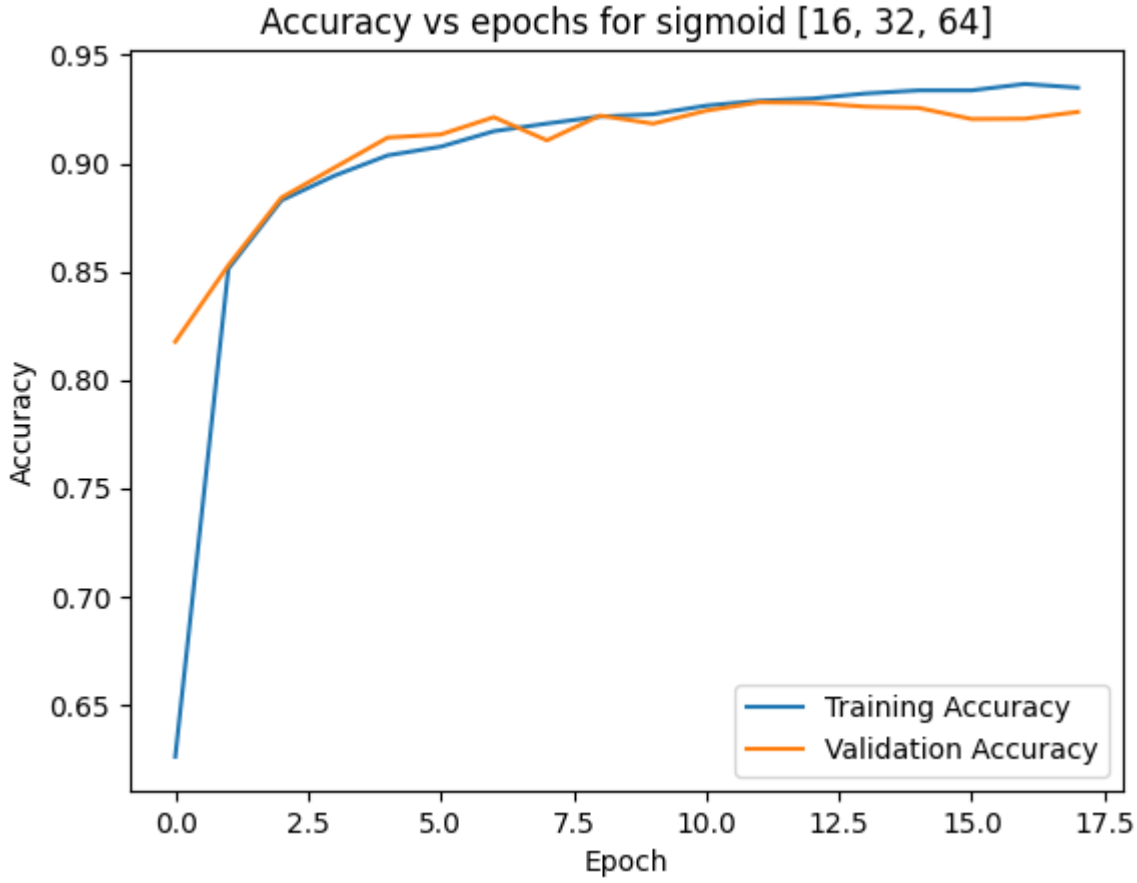
Total params: 19,706 (76.98 KB)

Trainable params: 19,706 (76.98 KB)

Non-trainable params: 0 (0.00 B)

```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.4173 - loss: 1.7356 - val_accuracy: 0.8176 - val_loss: 0.6427
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8375 - loss: 0.5732 - val_accuracy: 0.8529 - val_loss: 0.5020
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.8826 - loss: 0.4201 - val_accuracy: 0.8841 - val_loss: 0.3995
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.8932 - loss: 0.3659 - val_accuracy: 0.8980 - val_loss: 0.3518
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9025 - loss: 0.3347 - val_accuracy: 0.9119 - val_loss: 0.3181
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9079 - loss: 0.3120 - val_accuracy: 0.9133 - val_loss: 0.3068
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9131 - loss: 0.2963 - val_accuracy: 0.9213 - val_loss: 0.2845
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9166 - loss: 0.2805 - val_accuracy: 0.9106 - val_loss: 0.2991
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.9242 - loss: 0.2604 - val_accuracy: 0.9220 - val_loss: 0.2643
Epoch 10/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9233 - loss: 0.2518 - val_accuracy: 0.9183 - val_loss: 0.2723
Epoch 11/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9293 - loss: 0.2408 - val_accuracy: 0.9243 - val_loss: 0.2512
Epoch 12/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9294 - loss: 0.2372 - val_accuracy: 0.9281 - val_loss: 0.2452
Epoch 13/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9304 - loss: 0.2347 - val_accuracy: 0.9279 - val_loss: 0.2381
Epoch 14/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9350 - loss: 0.2225 - val_accuracy: 0.9261 - val_loss: 0.2485
Epoch 15/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9333 - loss: 0.2207 - val_accuracy: 0.9256 - val_loss: 0.2486
Epoch 16/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9348 - loss: 0.2179 - val_accuracy: 0.9204 - val_loss: 0.2607
Epoch 17/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9366 - loss: 0.2104 - val_accuracy: 0.9206 - val_loss: 0.2680
Epoch 18/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9356 - loss: 0.2119 - val_accuracy: 0.9237 - val_loss: 0.2464
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9357 - loss: 0.2305
Test loss=0.24331416189670563 Test accuracy = 0.9314285516738892
Time taken to train the model is 84.13690066337585 seconds
```


Loss vs epochs for sigmoid [16, 32, 64]
Training Loss
Validation Loss
Loss
Epoch
1.2
1.0
0.8
0.6
0.4
0.2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5



In [50]:
```
result_df
```

Out[50]:

	Hidden Layers	Activation Function	Hidden Neurons	Optimizer	Training Time(in seconds)	Test Loss	Test Accuracy
0	1	sigmoid	[16]	SGD	428.502323	0.085315	0.497714
1	2	sigmoid	[16, 32]	SGD	436.430634	0.089831	0.157286
2	3	sigmoid	[16, 32, 64]	SGD	428.479271	0.089967	0.119429
3	1	sigmoid	[16]	Adam	63.880293	0.282636	0.919714
4	2	sigmoid	[16, 32]	Adam	73.311765	0.245530	0.930714
5	3	sigmoid	[16, 32, 64]	Adam	84.136901	0.243314	0.931429

Question 05

v. For the following few tasks, use Adam optimizer (learning rate = 0.001) and Categorical Cross Entropy Loss. Run the network by changing the activation function hyper-parameter:

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

In [31]:
```
## Model 1 --> Adam, lr =0.001, CCE loss, hidden_neurons=[16,32,64]. Sigmoid

result_df = pd.DataFrame(
    columns=[
        'Hidden Layers',
        'Activation Function',
        'Hidden Neurons',
        'Training Time (in seconds)',
        'Test Loss',
        'Test Accuracy'
    ]
)

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

model, history, duration = train_model(
    activation_func=activation_function,
    hidden_neurons=hidden_neurons,
    optimizer=Adam,
    loss_func=CategoricalCrossentropy,
    epochs=100,
    dropout_rate=None,
    learning_rate=0.001
)

test_loss, test_acc = model.evaluate(X_test,y_test)
```

```python
print(f"Test loss={test_loss} Test accuracy = {test_acc}")
print(f"Time taken to train the model is {duration} seconds")

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

plot_metrics(history, activation_function, hidden_neurons)
```

Model: "sequential_8"

Layer (type)	Output Shape	Param #
flatten_8 (Flatten)	(None, 1024)	0
dense_24 (Dense)	(None, 16)	16,400
dense_25 (Dense)	(None, 32)	544
dense_26 (Dense)	(None, 64)	2,112
dense_27 (Dense)	(None, 10)	650

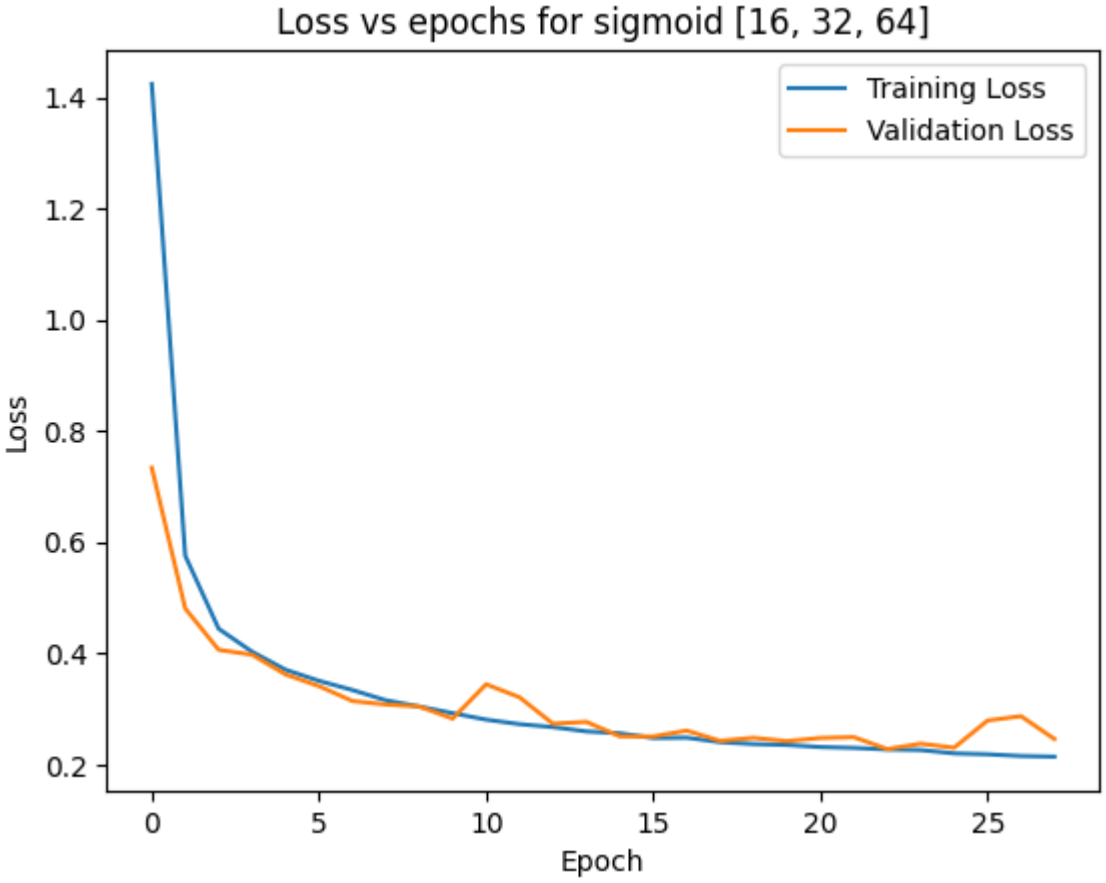
Total params: 19,706 (76.98 KB)

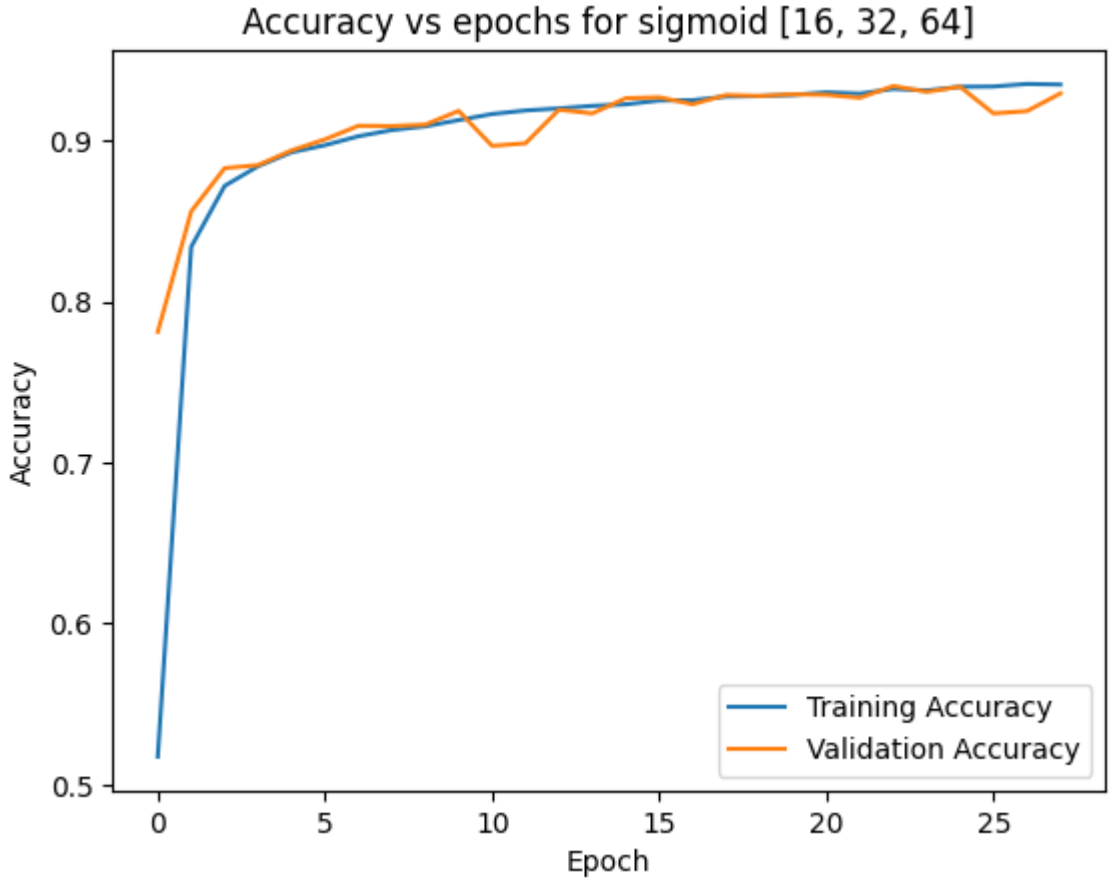
Trainable params: 19,706 (76.98 KB)

Non-trainable params: 0 (0.00 B)

```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 13s 6ms/step - accuracy: 0.3228 - loss: 1.9027 - val_accuracy: 0.7813 - val_loss: 0.7335
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8098 - loss: 0.6485 - val_accuracy: 0.8561 - val_loss: 0.4806
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8665 - loss: 0.4655 - val_accuracy: 0.8830 - val_loss: 0.4062
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8819 - loss: 0.4082 - val_accuracy: 0.8849 - val_loss: 0.3973
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8927 - loss: 0.3756 - val_accuracy: 0.8940 - val_loss: 0.3619
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8940 - loss: 0.3618 - val_accuracy: 0.9009 - val_loss: 0.3409
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9006 - loss: 0.3429 - val_accuracy: 0.9093 - val_loss: 0.3141
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9025 - loss: 0.3270 - val_accuracy: 0.9091 - val_loss: 0.3078
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9090 - loss: 0.3059 - val_accuracy: 0.9100 - val_loss: 0.3043
Epoch 10/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9127 - loss: 0.2916 - val_accuracy: 0.9186 - val_loss: 0.2824
Epoch 11/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9154 - loss: 0.2855 - val_accuracy: 0.8969 - val_loss: 0.3442
Epoch 12/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.9202 - loss: 0.2690 - val_accuracy: 0.8984 - val_loss: 0.3208
Epoch 13/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9209 - loss: 0.2629 - val_accuracy: 0.9193 - val_loss: 0.2733
Epoch 14/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9212 - loss: 0.2627 - val_accuracy: 0.9170 - val_loss: 0.2766
Epoch 15/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9225 - loss: 0.2560 - val_accuracy: 0.9264 - val_loss: 0.2505
Epoch 16/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9250 - loss: 0.2474 - val_accuracy: 0.9270 - val_loss: 0.2503
Epoch 17/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9263 - loss: 0.2429 - val_accuracy: 0.9227 - val_loss: 0.2609
Epoch 18/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.9271 - loss: 0.2426 - val_accuracy: 0.9284 - val_loss: 0.2425
Epoch 19/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9285 - loss: 0.2337 - val_accuracy: 0.9279 - val_loss: 0.2479
Epoch 20/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9293 - loss: 0.2318 - val_accuracy: 0.9289 - val_loss: 0.2420
Epoch 21/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9301 - loss: 0.2302 - val_accuracy: 0.9286 - val_loss: 0.2476
Epoch 22/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9297 - loss: 0.2279 - val_accuracy: 0.9267 - val_loss: 0.2491
Epoch 23/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9335 - loss: 0.2233 - val_accuracy: 0.9340 - val_loss: 0.2277
```

```
Epoch 24/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9331 - loss: 0.2206 - val_accuracy: 0.9304 - val_loss: 0.2373
Epoch 25/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9328 - loss: 0.2211 - val_accuracy: 0.9336 - val_loss: 0.2301
Epoch 26/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9352 - loss: 0.2139 - val_accuracy: 0.9170 - val_loss: 0.2789
Epoch 27/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9349 - loss: 0.2177 - val_accuracy: 0.9184 - val_loss: 0.2869
Epoch 28/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9361 - loss: 0.2082 - val_accuracy: 0.9294 - val_loss: 0.2458
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9338 - loss: 0.2278
Test loss=0.22948826849460602 Test accuracy = 0.9328571557998657
Time taken to train the model is 134.11440420150757 seconds
```





In [32]:
```
## Model 2 --> Adam, lr =0.001, CCE loss, hidden_neurons=[16,32,64]. tanh

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

model, history, duration = train_model(
    activation_func=activation_function,
    hidden_neurons=hidden_neurons,
    optimizer=Adam,
    loss_func=CategoricalCrossentropy,
    epochs=100,
    dropout_rate=None,
    learning_rate=0.001
)

test_loss, test_acc = model.evaluate(X_test,y_test)

print(f"Test loss={test_loss} Test accuracy = {test_acc}")
```

```
print(f"Time taken to train the model is {duration} seconds")

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

plot_metrics(history, activation_function, hidden_neurons)
```

Model: "sequential_9"

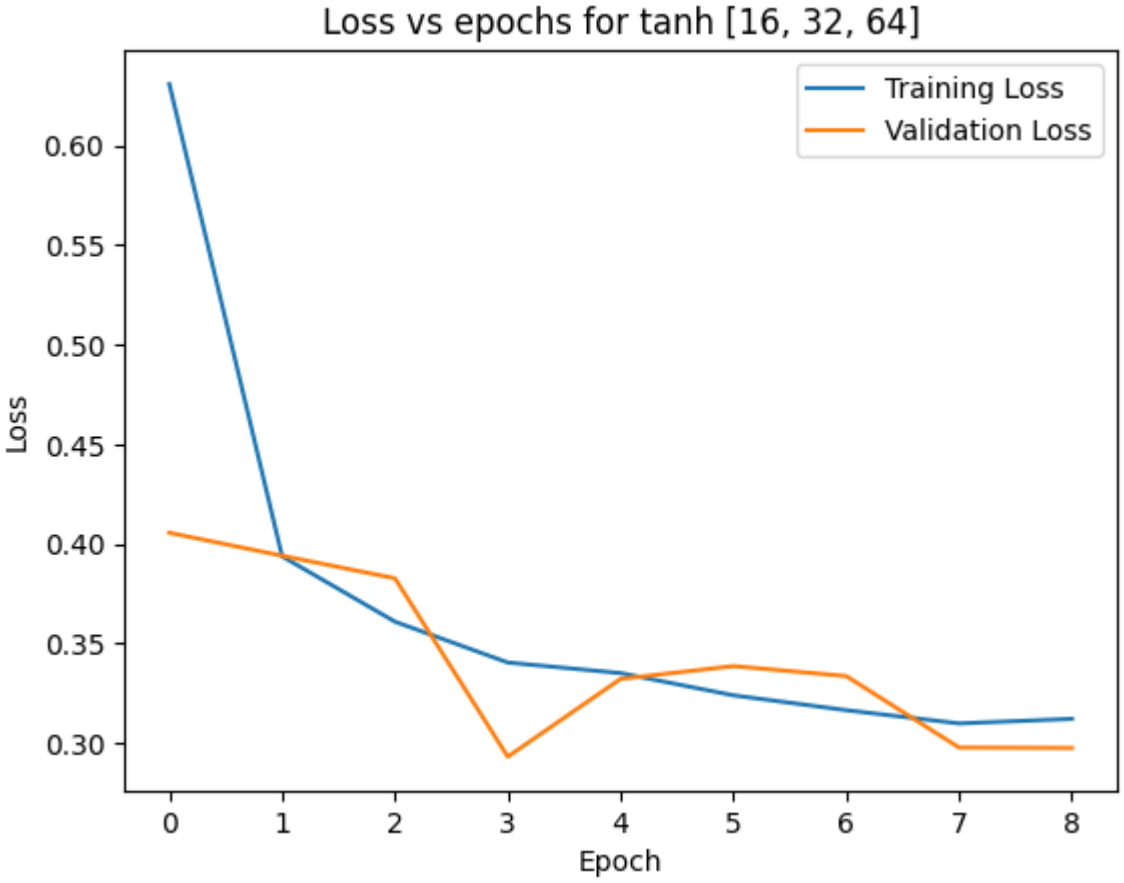
Layer (type)	Output Shape	Param #
flatten_9 (Flatten)	(None, 1024)	0
dense_28 (Dense)	(None, 16)	16,400
dense_29 (Dense)	(None, 32)	544
dense_30 (Dense)	(None, 64)	2,112
dense_31 (Dense)	(None, 10)	650

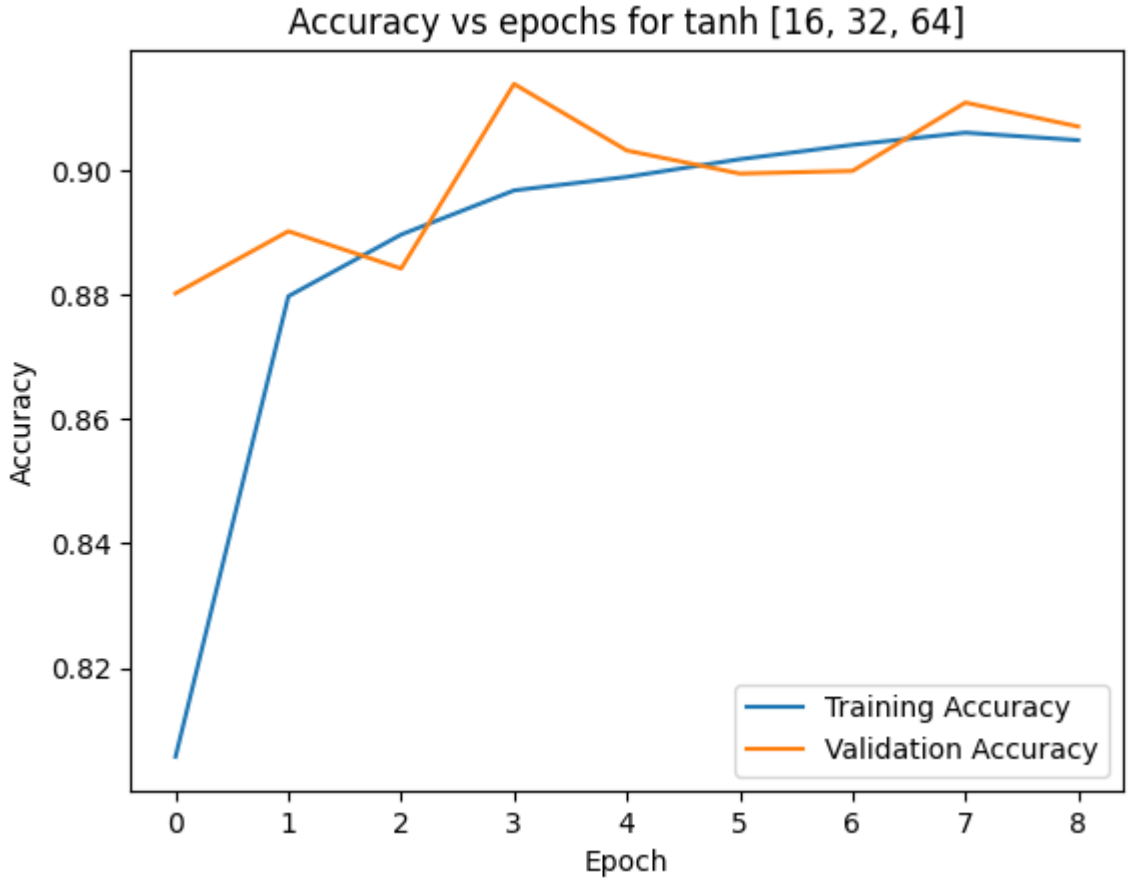
Total params: 19,706 (76.98 KB)

Trainable params: 19,706 (76.98 KB)

Non-trainable params: 0 (0.00 B)

```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.6881 - loss: 0.9802 - val_accuracy: 0.8803 - val_loss: 0.4057
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.8780 - loss: 0.3985 - val_accuracy: 0.8903 - val_loss: 0.3941
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8899 - loss: 0.3602 - val_accuracy: 0.8843 - val_loss: 0.3828
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.8964 - loss: 0.3406 - val_accuracy: 0.9140 - val_loss: 0.2933
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8982 - loss: 0.3361 - val_accuracy: 0.9033 - val_loss: 0.3323
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9023 - loss: 0.3262 - val_accuracy: 0.8996 - val_loss: 0.3387
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9052 - loss: 0.3155 - val_accuracy: 0.9000 - val_loss: 0.3337
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9037 - loss: 0.3185 - val_accuracy: 0.9110 - val_loss: 0.2978
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9060 - loss: 0.3082 - val_accuracy: 0.9071 - val_loss: 0.2976
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9142 - loss: 0.2919
Test loss=0.3065372705459595 Test accuracy = 0.9105714559555054
Time taken to train the model is 41.307180643081665 seconds
```


Loss vs epochs for tanh [16, 32, 64]
Training Loss
Validation Loss
Loss
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0
1
2
3
4
5
6
7
8
Epoch



In [33]:
```
## Model 3 --> Adam, lr =0.001, CCE loss, hidden_neurons=[16,32,64]. relu

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

model, history, duration = train_model(
    activation_func=activation_function,
    hidden_neurons=hidden_neurons,
    optimizer=Adam,
    loss_func=CategoricalCrossentropy,
    epochs=100,
    dropout_rate=None,
    learning_rate=0.001
)

test_loss, test_acc = model.evaluate(X_test,y_test)

print(f"Test loss={test_loss} Test accuracy = {test_acc}")
```

```python
print(f"Time taken to train the model is {duration} seconds")

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

plot_metrics(history, activation_function, hidden_neurons)
```

Model: "sequential_10"

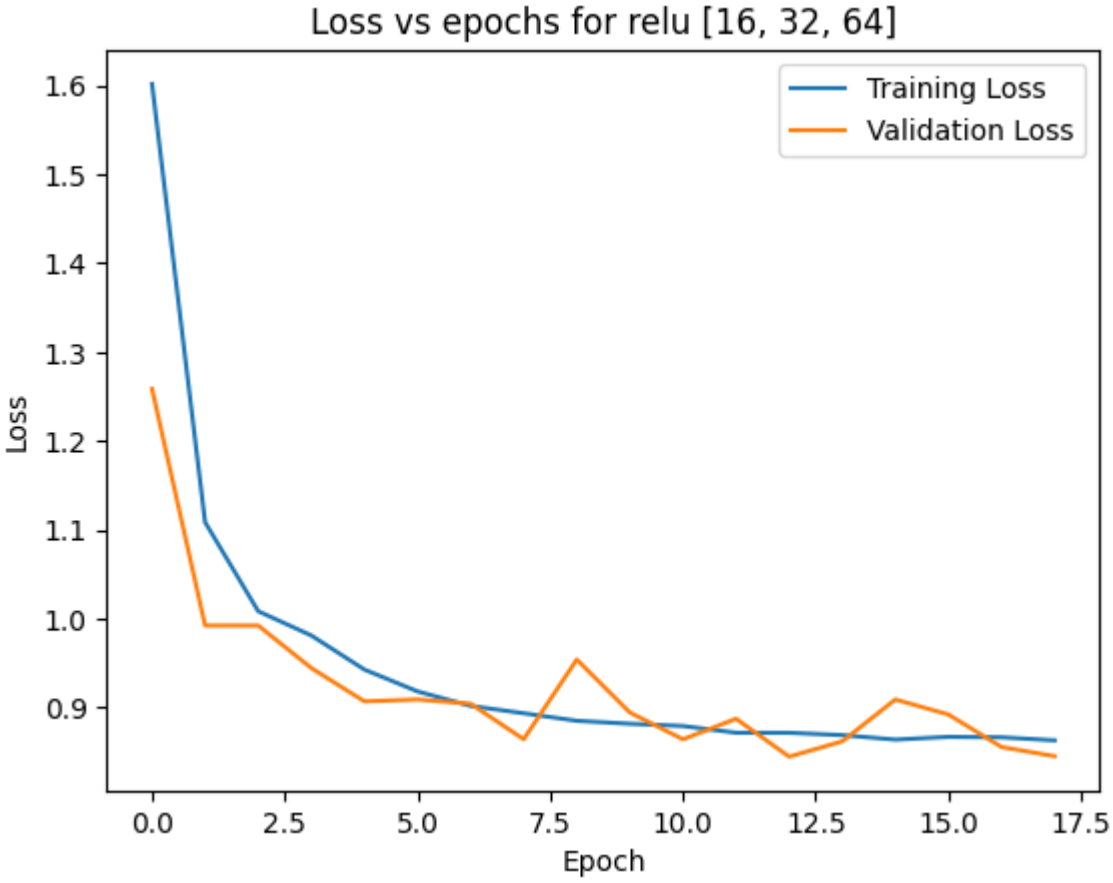
Layer (type)	Output Shape	Param #
flatten_10 (Flatten)	(None, 1024)	0
dense_32 (Dense)	(None, 16)	16,400
dense_33 (Dense)	(None, 32)	544
dense_34 (Dense)	(None, 64)	2,112
dense_35 (Dense)	(None, 10)	650

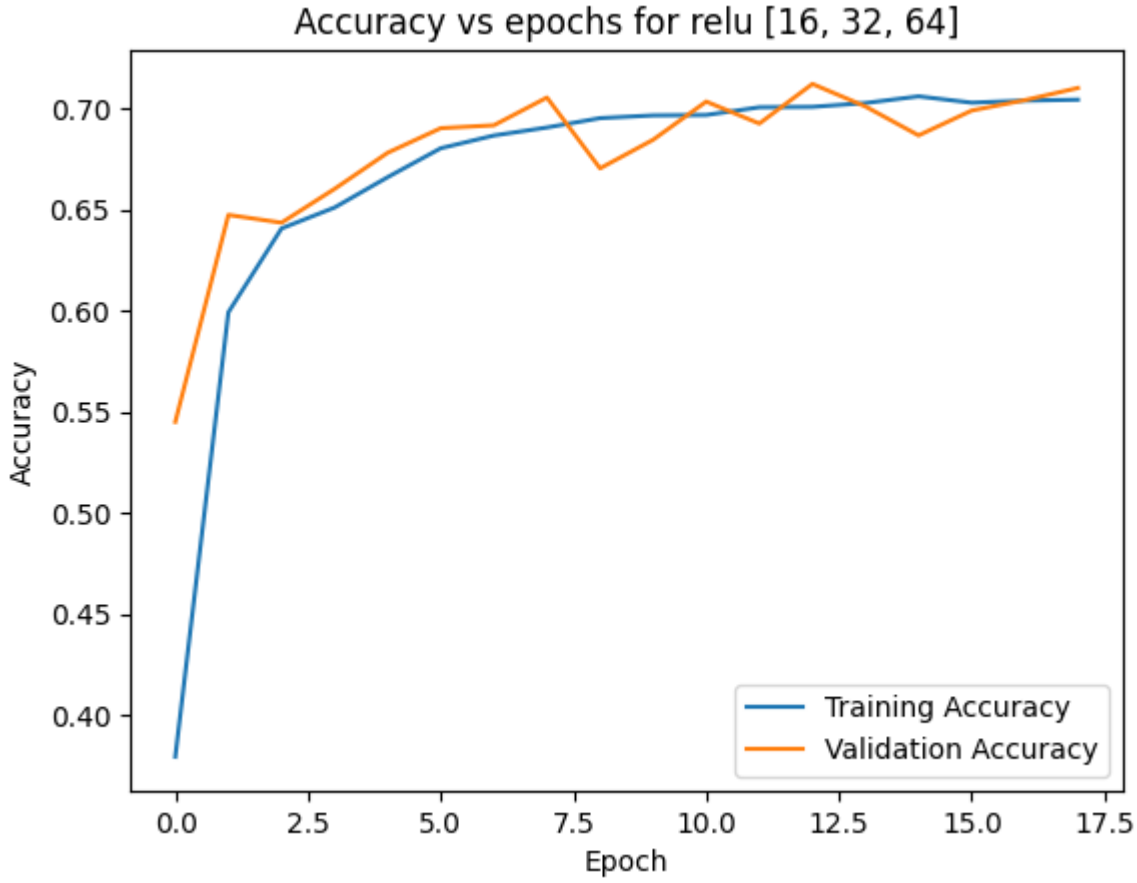
Total params: 19,706 (76.98 KB)

Trainable params: 19,706 (76.98 KB)

Non-trainable params: 0 (0.00 B)

```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.2840 - loss: 1.8179 - val_accuracy: 0.5450 - val_loss: 1.2585
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.5777 - loss: 1.1678 - val_accuracy: 0.6473 - val_loss: 0.9924
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.6378 - loss: 1.0118 - val_accuracy: 0.6434 - val_loss: 0.9923
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.6501 - loss: 0.9843 - val_accuracy: 0.6601 - val_loss: 0.9445
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.6603 - loss: 0.9599 - val_accuracy: 0.6780 - val_loss: 0.9070
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.6788 - loss: 0.9196 - val_accuracy: 0.6901 - val_loss: 0.9092
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.6863 - loss: 0.9003 - val_accuracy: 0.6916 - val_loss: 0.9044
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.6861 - loss: 0.8993 - val_accuracy: 0.7053 - val_loss: 0.8645
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.6986 - loss: 0.8743 - val_accuracy: 0.6703 - val_loss: 0.9541
Epoch 10/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.6951 - loss: 0.8849 - val_accuracy: 0.6844 - val_loss: 0.8945
Epoch 11/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.6954 - loss: 0.8820 - val_accuracy: 0.7033 - val_loss: 0.8643
Epoch 12/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.6994 - loss: 0.8737 - val_accuracy: 0.6924 - val_loss: 0.8875
Epoch 13/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7044 - loss: 0.8581 - val_accuracy: 0.7120 - val_loss: 0.8448
Epoch 14/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.6999 - loss: 0.8727 - val_accuracy: 0.7009 - val_loss: 0.8620
Epoch 15/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.7063 - loss: 0.8580 - val_accuracy: 0.6866 - val_loss: 0.9092
Epoch 16/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7027 - loss: 0.8695 - val_accuracy: 0.6989 - val_loss: 0.8923
Epoch 17/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.7070 - loss: 0.8619 - val_accuracy: 0.7040 - val_loss: 0.8557
Epoch 18/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.7056 - loss: 0.8605 - val_accuracy: 0.7100 - val_loss: 0.8454
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.7070 - loss: 0.8464
Test loss=0.849205493927002 Test accuracy = 0.7095714211463928
Time taken to train the model is 81.0392382144928 seconds
```


Loss vs epochs for relu [16, 32, 64]
Training Loss
Validation Loss
Loss
Epoch
1.6
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5



In [34]:
```
result_df
```

Out[34]:

	Hidden Layers	Activation Function	Hidden Neurons	Training Time (in seconds)	Test Loss	Test Accuracy
0	3	sigmoid	[16, 32, 64]	134.114404	0.229488	0.932857
1	3	tanh	[16, 32, 64]	41.307181	0.306537	0.910571
2	3	relu	[16, 32, 64]	81.039238	0.849205	0.709571

From the above table, we can infer that the best activation funtion is `sigmoid`

Question 06

vi. Run the network by changing the number of 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

In [35]:
```
result_df_dropout = pd.DataFrame(
    columns=[
        'Hidden Layers',
        'Activation Function',
        'Hidden Neurons',
        'Dropout Rate',
        'Training Time (in seconds)',
        'Test Loss',
        'Test Accuracy'
    ]
)
```

In [36]:
```
## Model 1 --> Adam, lr =0.001, CCE loss, hidden_neurons=[16,32,64]. relu, dropout=0.9

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

model, history,duration= train_model(
    activation_func=activation_function,
    hidden_neurons=hidden_neurons,
    optimizer=Adam,
    loss_func=CategoricalCrossentropy,
    epochs=100,
    dropout_rate=dropout_rate,
    learning_rate=0.001
```

```
)

test_loss, test_acc = model.evaluate(X_test,y_test)

print(f"Test loss={test_loss} Test accuracy = {test_acc}")
print(f"Time required to train the model is {duration} seconds")

result_df_dropout.loc[len(result_df_dropout.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    dropout_rate,
    duration,
    test_loss,
    test_acc
]

plot_metrics(history, activation_function, hidden_neurons)
```

Model: "sequential_11"

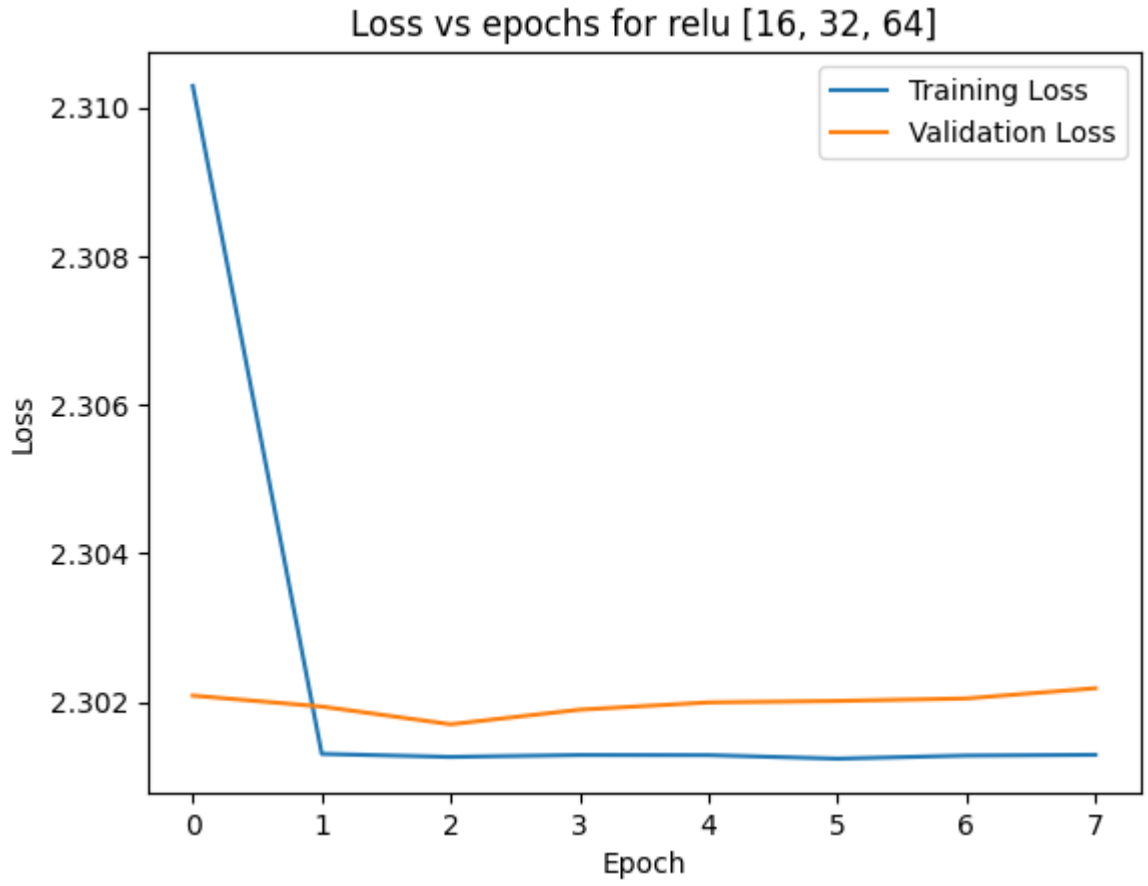
Layer (type)	Output Shape	Param #
flatten_11 (Flatten)	(None, 1024)	0
dense_36 (Dense)	(None, 16)	16,400
dropout (Dropout)	(None, 16)	0
dense_37 (Dense)	(None, 32)	544
dropout_1 (Dropout)	(None, 32)	0
dense_38 (Dense)	(None, 64)	2,112
dropout_2 (Dropout)	(None, 64)	0
dense_39 (Dense)	(None, 10)	650

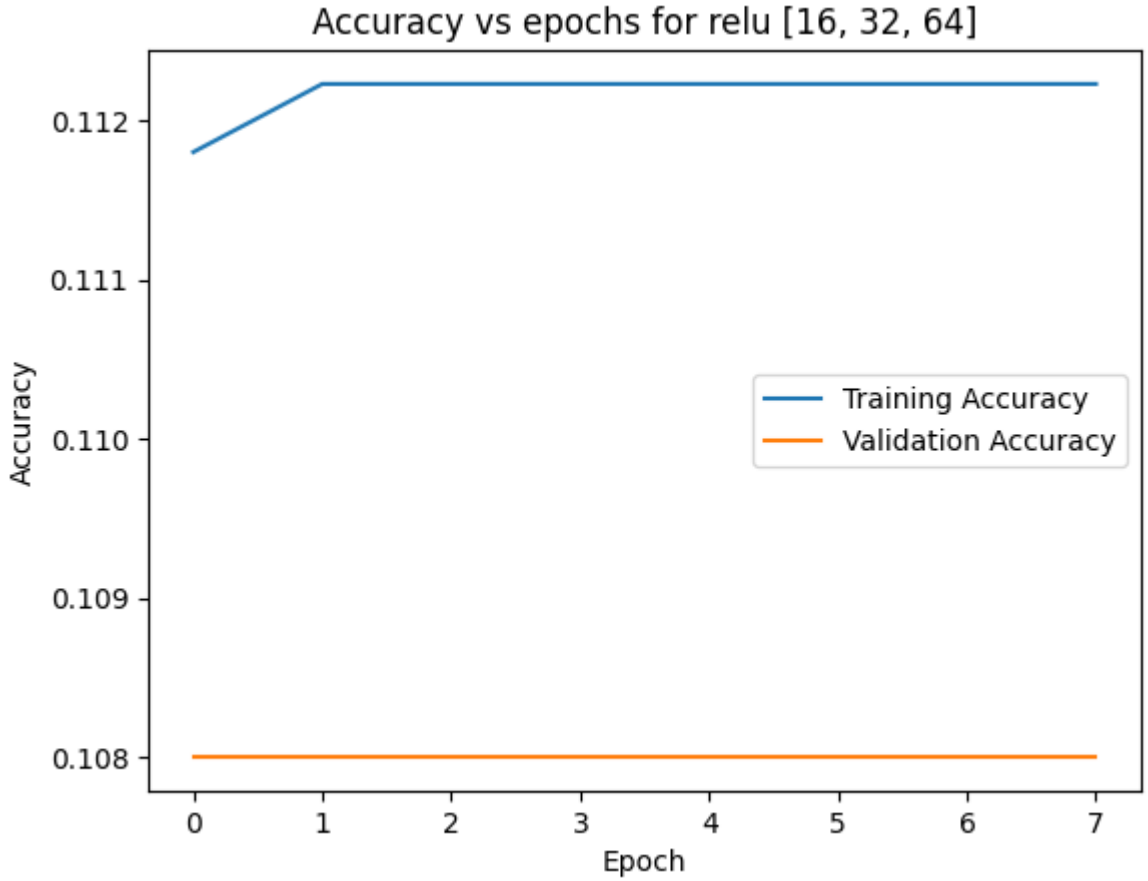
Total params: 19,706 (76.98 KB)

Trainable params: 19,706 (76.98 KB)

Non-trainable params: 0 (0.00 B)

```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.1100 - loss: 2.3640 - val_accuracy: 0.1080 - val_loss: 2.3021
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1111 - loss: 2.3014 - val_accuracy: 0.1080 - val_loss: 2.3019
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1120 - loss: 2.3011 - val_accuracy: 0.1080 - val_loss: 2.3017
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.1106 - loss: 2.3016 - val_accuracy: 0.1080 - val_loss: 2.3019
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1117 - loss: 2.3015 - val_accuracy: 0.1080 - val_loss: 2.3020
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1130 - loss: 2.3014 - val_accuracy: 0.1080 - val_loss: 2.3020
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.1118 - loss: 2.3016 - val_accuracy: 0.1080 - val_loss: 2.3021
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1116 - loss: 2.3015 - val_accuracy: 0.1080 - val_loss: 2.3022
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.1219 - loss: 2.3007
Test loss=2.300971031188965 Test accuracy = 0.11942857503890991
Time required to train the model is 37.40566968917847 seconds
```





In [37]:
```
## Model 2 --> Adam, lr =0.001, CCE loss, hidden_neurons=[16,32,64]. relu, dropout=0.75

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

model, history,duration = train_model(
    activation_func=activation_function,
    hidden_neurons=hidden_neurons,
    optimizer=Adam,
    loss_func=CategoricalCrossentropy,
    epochs=100,
    dropout_rate=dropout_rate,
    learning_rate=0.001
)

test_loss, test_acc = model.evaluate(X_test,y_test)
```

```python
print(f"Test loss={test_loss} Test accuracy = {test_acc}")
print(f"Time required to train the model is {duration} seconds")

result_df_dropout.loc[len(result_df_dropout.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    dropout_rate,
    duration,
    test_loss,
    test_acc
]

plot_metrics(history, activation_function, hidden_neurons)
```

Model: "sequential_12"

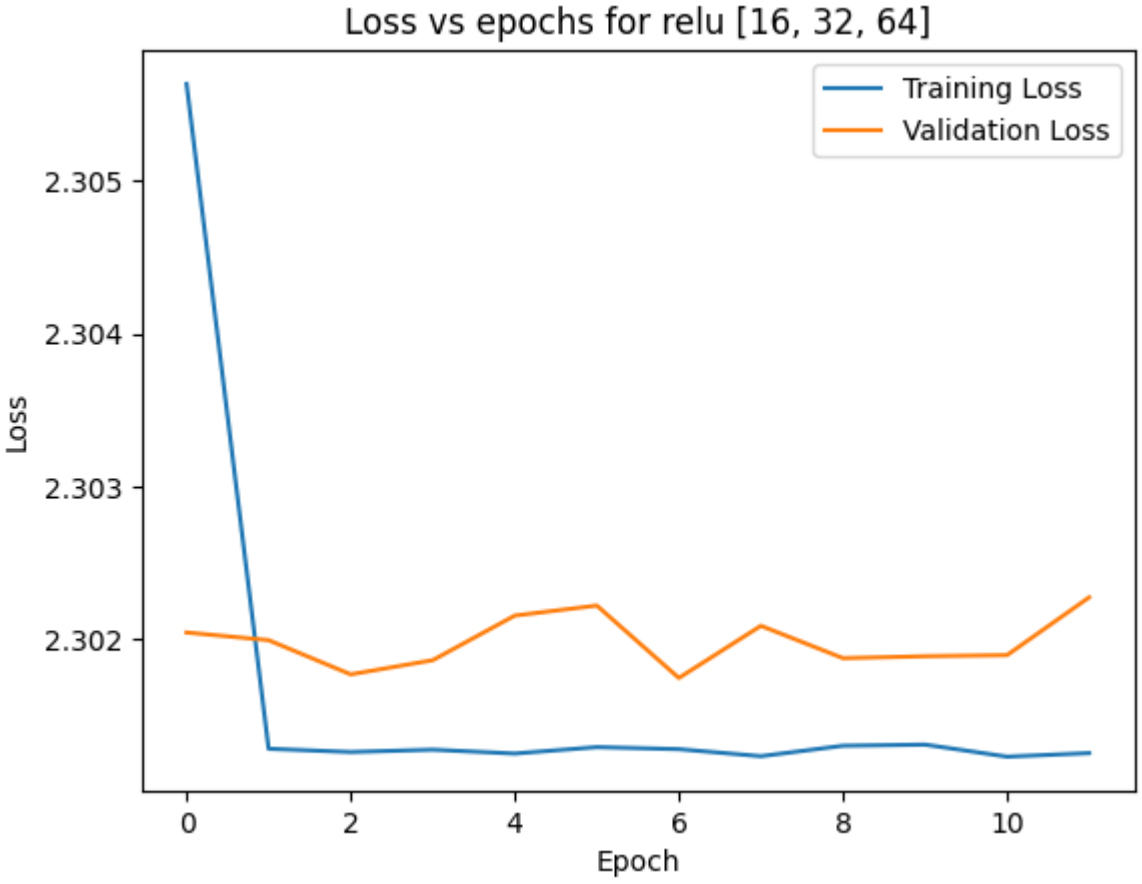
Layer (type)	Output Shape	Param #
flatten_12 (Flatten)	(None, 1024)	0
dense_40 (Dense)	(None, 16)	16,400
dropout_3 (Dropout)	(None, 16)	0
dense_41 (Dense)	(None, 32)	544
dropout_4 (Dropout)	(None, 32)	0
dense_42 (Dense)	(None, 64)	2,112
dropout_5 (Dropout)	(None, 64)	0
dense_43 (Dense)	(None, 10)	650

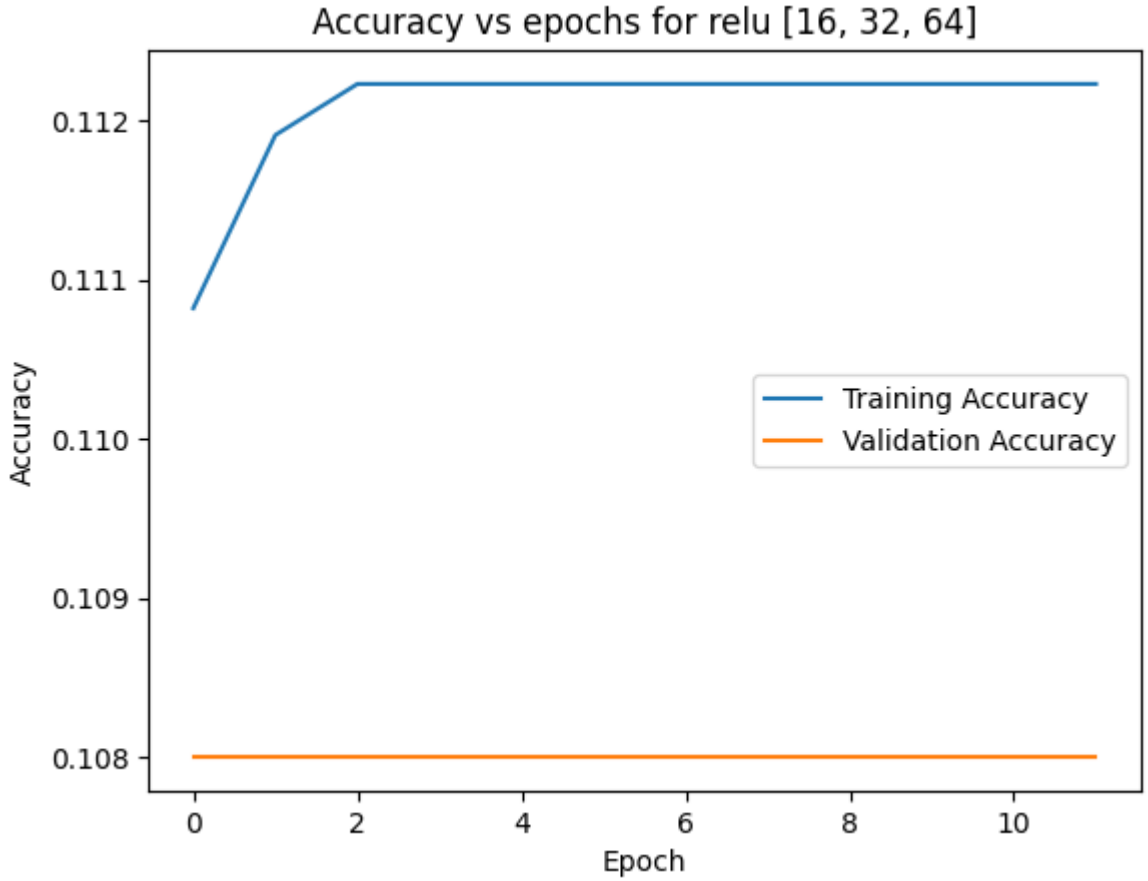
Total params: 19,706 (76.98 KB)

Trainable params: 19,706 (76.98 KB)

Non-trainable params: 0 (0.00 B)

```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.1090 - loss: 2.3317 - val_accuracy: 0.1080 - val_loss: 2.3020
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1107 - loss: 2.3014 - val_accuracy: 0.1080 - val_loss: 2.3020
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1131 - loss: 2.3012 - val_accuracy: 0.1080 - val_loss: 2.3018
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1128 - loss: 2.3011 - val_accuracy: 0.1080 - val_loss: 2.3019
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1131 - loss: 2.3012 - val_accuracy: 0.1080 - val_loss: 2.3022
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1109 - loss: 2.3014 - val_accuracy: 0.1080 - val_loss: 2.3022
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.1135 - loss: 2.3009 - val_accuracy: 0.1080 - val_loss: 2.3017
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1131 - loss: 2.3012 - val_accuracy: 0.1080 - val_loss: 2.3021
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1136 - loss: 2.3011 - val_accuracy: 0.1080 - val_loss: 2.3019
Epoch 10/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.1130 - loss: 2.3011 - val_accuracy: 0.1080 - val_loss: 2.3019
Epoch 11/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.1140 - loss: 2.3008 - val_accuracy: 0.1080 - val_loss: 2.3019
Epoch 12/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.1122 - loss: 2.3015 - val_accuracy: 0.1080 - val_loss: 2.3023
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.1219 - loss: 2.3006
Test loss=2.3008530139923096 Test accuracy = 0.11942857503890991
Time required to train the model is 61.32969856262207 seconds
```


Loss vs epochs for relu [16, 32, 64]
Training Loss
Validation Loss
2.305
2.304
2.303
2.302
Loss
0
2
4
6
8
10
Epoch



In [38]:
```
## Model 3 --> Adam, lr =0.001, CCE loss, hidden_neurons=[16,32,64]. relu, dropout=0.5

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

model, history, duration = train_model(
    activation_func=activation_function,
    hidden_neurons=hidden_neurons,
    optimizer=Adam,
    loss_func=CategoricalCrossentropy,
    epochs=100,
    dropout_rate=dropout_rate,
    learning_rate=0.001
)

test_loss, test_acc = model.evaluate(X_test,y_test)
```

```python
print(f"Test loss={test_loss} Test accuracy = {test_acc}")
print(f"Time taken to train the model is {duration} seconds")

result_df_dropout.loc[len(result_df_dropout.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    dropout_rate,
    duration,
    test_loss,
    test_acc
]

plot_metrics(history, activation_function, hidden_neurons)
```

Model: "sequential_13"

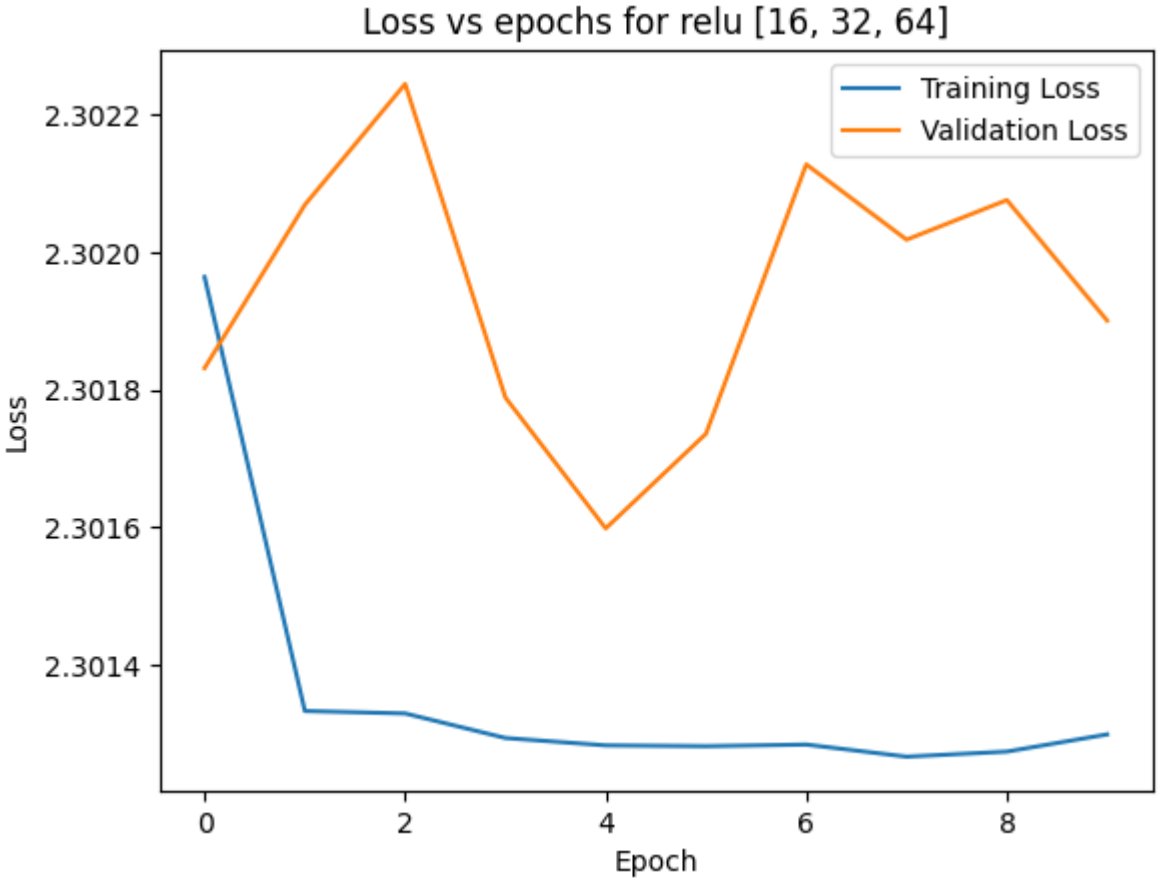
Layer (type)	Output Shape	Param #
flatten_13 (Flatten)	(None, 1024)	0
dense_44 (Dense)	(None, 16)	16,400
dropout_6 (Dropout)	(None, 16)	0
dense_45 (Dense)	(None, 32)	544
dropout_7 (Dropout)	(None, 32)	0
dense_46 (Dense)	(None, 64)	2,112
dropout_8 (Dropout)	(None, 64)	0
dense_47 (Dense)	(None, 10)	650

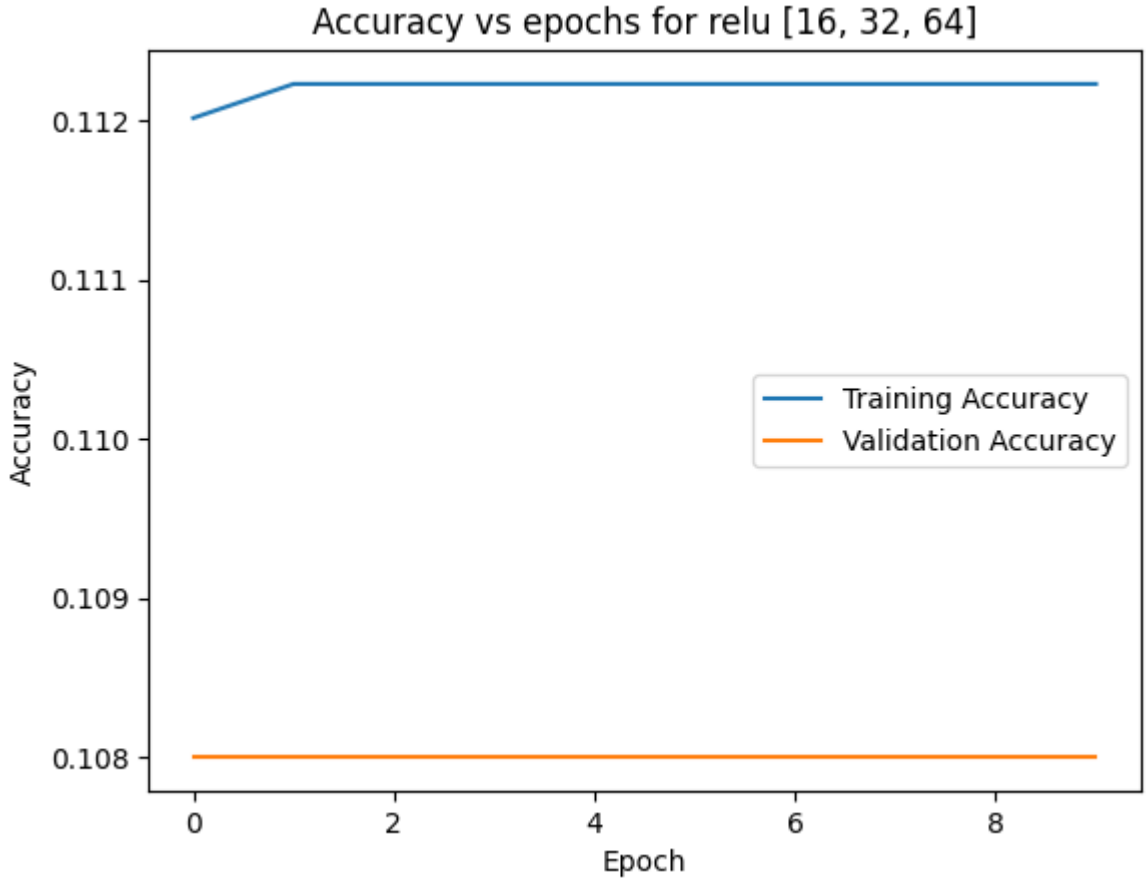
Total params: 19,706 (76.98 KB)

Trainable params: 19,706 (76.98 KB)

Non-trainable params: 0 (0.00 B)

```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1118 - loss: 2.3042 - val_accuracy: 0.1080 - val_loss: 2.3018
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.1134 - loss: 2.3013 - val_accuracy: 0.1080 - val_loss: 2.3021
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1101 - loss: 2.3016 - val_accuracy: 0.1080 - val_loss: 2.3022
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1120 - loss: 2.3012 - val_accuracy: 0.1080 - val_loss: 2.3018
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.1124 - loss: 2.3010 - val_accuracy: 0.1080 - val_loss: 2.3016
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1155 - loss: 2.3010 - val_accuracy: 0.1080 - val_loss: 2.3017
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1139 - loss: 2.3011 - val_accuracy: 0.1080 - val_loss: 2.3021
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1125 - loss: 2.3012 - val_accuracy: 0.1080 - val_loss: 2.3020
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1094 - loss: 2.3016 - val_accuracy: 0.1080 - val_loss: 2.3021
Epoch 10/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.1111 - loss: 2.3014 - val_accuracy: 0.1080 - val_loss: 2.3019
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.1219 - loss: 2.3008
Test loss=2.3010246753692627 Test accuracy = 0.11942857503890991
Time taken to train the model is 51.37219834327698 seconds
```


Loss vs epochs for relu [16, 32, 64]
Training Loss
Validation Loss
2.3022
2.3020
2.3018
2.3016
2.3014
Loss
0
2
4
6
8
Epoch



In [39]:
```
## Model 4 --> Adam, lr =0.001, CCE loss, hidden_neurons=[16,32,64]. relu, dropout=0.25

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

model, history, duration = train_model(
    activation_func=activation_function,
    hidden_neurons=hidden_neurons,
    optimizer=Adam,
    loss_func=CategoricalCrossentropy,
    epochs=100,
    dropout_rate=dropout_rate,
    learning_rate=0.001
)

test_loss, test_acc = model.evaluate(X_test,y_test)
```

```python
print(f"Test loss={test_loss} Test accuracy = {test_acc}")
print(f"Time taken to train the model is {duration} seconds")

result_df_dropout.loc[len(result_df_dropout.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    dropout_rate,
    duration,
    test_loss,
    test_acc
]

plot_metrics(history, activation_function, hidden_neurons)
```

Model: "sequential_14"

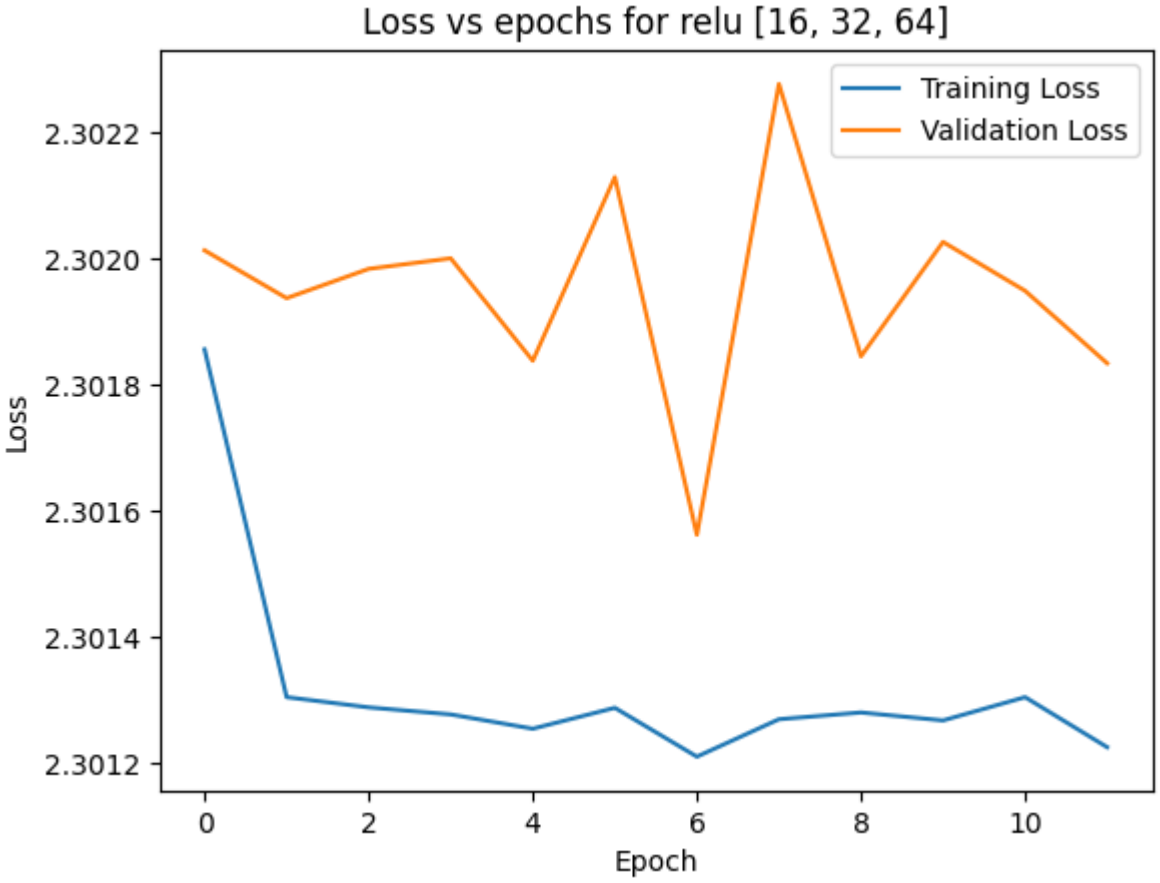
Layer (type)	Output Shape	Param #
flatten_14 (Flatten)	(None, 1024)	0
dense_48 (Dense)	(None, 16)	16,400
dropout_9 (Dropout)	(None, 16)	0
dense_49 (Dense)	(None, 32)	544
dropout_10 (Dropout)	(None, 32)	0
dense_50 (Dense)	(None, 64)	2,112
dropout_11 (Dropout)	(None, 64)	0
dense_51 (Dense)	(None, 10)	650

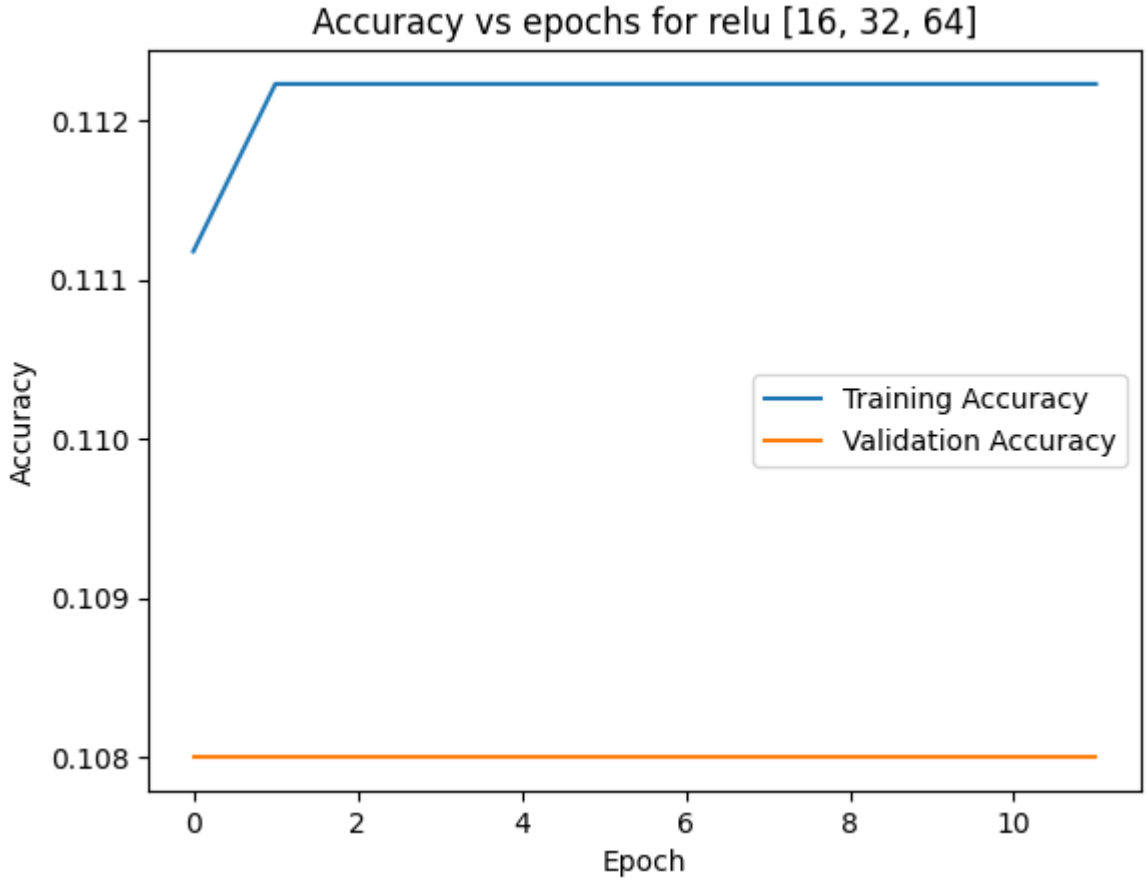
Total params: 19,706 (76.98 KB)

Trainable params: 19,706 (76.98 KB)

Non-trainable params: 0 (0.00 B)

```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1099 - loss: 2.3045 - val_accuracy: 0.1080 - val_loss: 2.3020
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.1137 - loss: 2.3010 - val_accuracy: 0.1080 - val_loss: 2.3019
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1126 - loss: 2.3008 - val_accuracy: 0.1080 - val_loss: 2.3020
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1137 - loss: 2.3011 - val_accuracy: 0.1080 - val_loss: 2.3020
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.1100 - loss: 2.3015 - val_accuracy: 0.1080 - val_loss: 2.3018
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1107 - loss: 2.3015 - val_accuracy: 0.1080 - val_loss: 2.3021
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1142 - loss: 2.3010 - val_accuracy: 0.1080 - val_loss: 2.3016
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.1111 - loss: 2.3014 - val_accuracy: 0.1080 - val_loss: 2.3023
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1136 - loss: 2.3014 - val_accuracy: 0.1080 - val_loss: 2.3018
Epoch 10/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1127 - loss: 2.3013 - val_accuracy: 0.1080 - val_loss: 2.3020
Epoch 11/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.1120 - loss: 2.3012 - val_accuracy: 0.1080 - val_loss: 2.3019
Epoch 12/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1126 - loss: 2.3013 - val_accuracy: 0.1080 - val_loss: 2.3018
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.1219 - loss: 2.3009
Test loss=2.3010637760162354 Test accuracy = 0.11942857503890991
Time taken to train the model is 59.987740993499756 seconds
```


Loss vs epochs for relu [16, 32, 64]
Training Loss
Validation Loss
2.3022
2.3020
2.3018
2.3016
2.3014
2.3012
Loss
0
2
4
6
8
10
Epoch



In [40]:
```
## Model 5 --> Adam, lr =0.001, CCE loss, hidden_neurons=[16,32,64]. relu, dropout=0.1

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

model, history, duration = train_model(
    activation_func=activation_function,
    hidden_neurons=hidden_neurons,
    optimizer=Adam,
    loss_func=CategoricalCrossentropy,
    epochs=100,
    dropout_rate=dropout_rate,
    learning_rate=0.001
)

test_loss, test_acc = model.evaluate(X_test,y_test)
```

```python
print(f"Test loss={test_loss} Test accuracy = {test_acc}")
print(f"Time taken to train the model is {duration} seconds")

result_df_dropout.loc[len(result_df_dropout.index)] = [
    len(hidden_neurons),
    activation_function,
    str(hidden_neurons),
    dropout_rate,
    duration,
    test_loss,
    test_acc
]

plot_metrics(history, activation_function, hidden_neurons)
```

Model: "sequential_15"

Layer (type)	Output Shape	Param #
flatten_15 (Flatten)	(None, 1024)	0
dense_52 (Dense)	(None, 16)	16,400
dropout_12 (Dropout)	(None, 16)	0
dense_53 (Dense)	(None, 32)	544
dropout_13 (Dropout)	(None, 32)	0
dense_54 (Dense)	(None, 64)	2,112
dropout_14 (Dropout)	(None, 64)	0
dense_55 (Dense)	(None, 10)	650

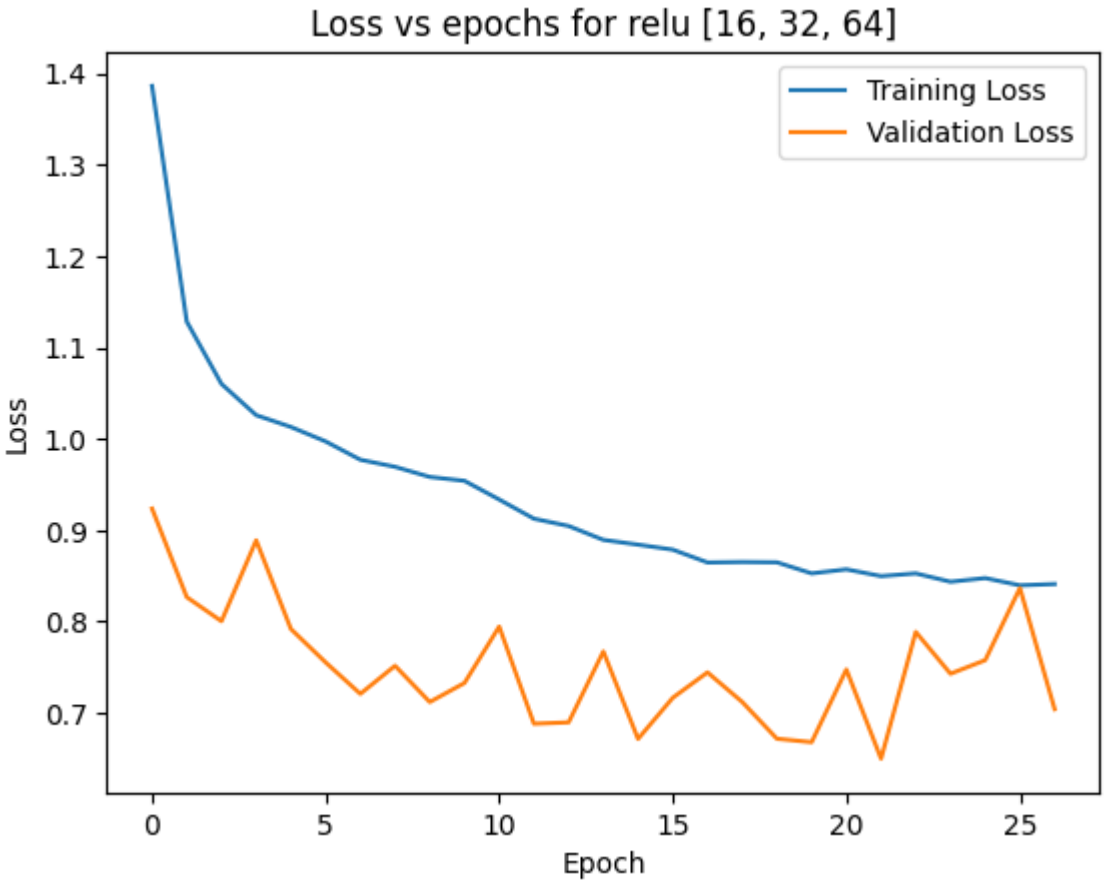
Total params: 19,706 (76.98 KB)

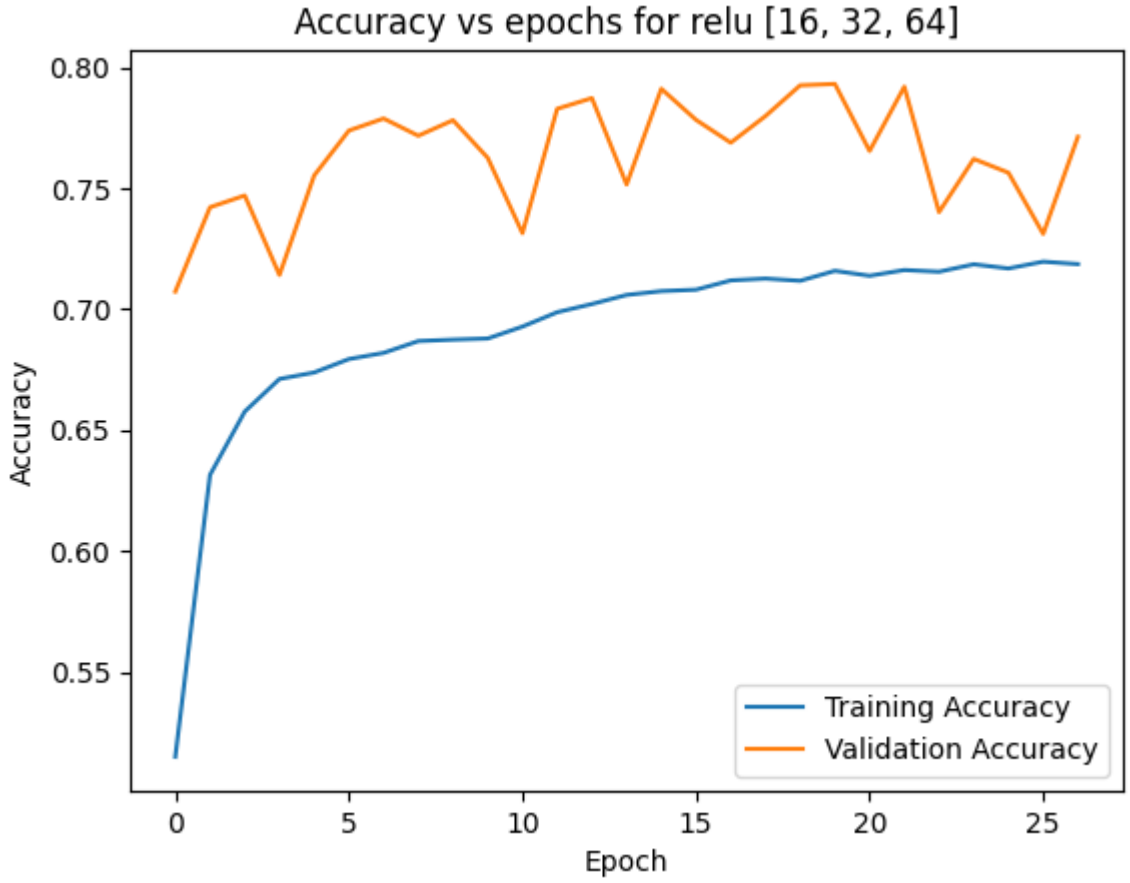
Trainable params: 19,706 (76.98 KB)

Non-trainable params: 0 (0.00 B)

```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.4060 - loss: 1.6379 - val_accuracy: 0.7073 - val_loss: 0.9236
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.6211 - loss: 1.1505 - val_accuracy: 0.7421 - val_loss: 0.8267
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.6544 - loss: 1.0685 - val_accuracy: 0.7470 - val_loss: 0.8004
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.6727 - loss: 1.0284 - val_accuracy: 0.7141 - val_loss: 0.8889
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.6723 - loss: 1.0184 - val_accuracy: 0.7553 - val_loss: 0.7918
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.6791 - loss: 0.9962 - val_accuracy: 0.7739 - val_loss: 0.7553
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.6836 - loss: 0.9714 - val_accuracy: 0.7789 - val_loss: 0.7208
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.6905 - loss: 0.9607 - val_accuracy: 0.7717 - val_loss: 0.7515
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.6872 - loss: 0.9594 - val_accuracy: 0.7781 - val_loss: 0.7119
Epoch 10/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.6858 - loss: 0.9545 - val_accuracy: 0.7626 - val_loss: 0.7327
Epoch 11/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.6922 - loss: 0.9382 - val_accuracy: 0.7314 - val_loss: 0.7945
Epoch 12/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.6985 - loss: 0.9171 - val_accuracy: 0.7829 - val_loss: 0.6881
Epoch 13/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.7034 - loss: 0.8968 - val_accuracy: 0.7873 - val_loss: 0.6896
Epoch 14/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7059 - loss: 0.8926 - val_accuracy: 0.7516 - val_loss: 0.7672
Epoch 15/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.7079 - loss: 0.8829 - val_accuracy: 0.7911 - val_loss: 0.6715
Epoch 16/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.7068 - loss: 0.8835 - val_accuracy: 0.7783 - val_loss: 0.7165
Epoch 17/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.7160 - loss: 0.8552 - val_accuracy: 0.7689 - val_loss: 0.7443
Epoch 18/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.7110 - loss: 0.8725 - val_accuracy: 0.7799 - val_loss: 0.7120
Epoch 19/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7116 - loss: 0.8656 - val_accuracy: 0.7926 - val_loss: 0.6717
Epoch 20/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.7148 - loss: 0.8525 - val_accuracy: 0.7931 - val_loss: 0.6679
Epoch 21/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.7135 - loss: 0.8574 - val_accuracy: 0.7654 - val_loss: 0.7475
Epoch 22/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7156 - loss: 0.8492 - val_accuracy: 0.7920 - val_loss: 0.6498
Epoch 23/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.7127 - loss: 0.8636 - val_accuracy: 0.7401 - val_loss: 0.7886
```

```
Epoch 24/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.7181 - loss: 0.8411 - val_accuracy: 0.7621 - val_loss: 0.7428
Epoch 25/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7129 - loss: 0.8549 - val_accuracy: 0.7564 - val_loss: 0.7577
Epoch 26/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.7206 - loss: 0.8388 - val_accuracy: 0.7311 - val_loss: 0.8366
Epoch 27/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7158 - loss: 0.8482 - val_accuracy: 0.7713 - val_loss: 0.7041
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.7998 - loss: 0.6454
Test loss=0.6585350632667542 Test accuracy = 0.8001428842544556
Time taken to train the model is 138.4717516899109 seconds
```





In [41]:
```
result_df_dropout
```

Out[41]:

	Hidden Layers	Activation Function	Hidden Neurons	Dropout Rate	Training Time (in seconds)	Test Loss	Test Accuracy
0	3	relu	[16, 32, 64]	0.90	37.405670	2.300971	0.119429
1	3	relu	[16, 32, 64]	0.75	61.329699	2.300853	0.119429
2	3	relu	[16, 32, 64]	0.50	51.372198	2.301025	0.119429
3	3	relu	[16, 32, 64]	0.25	59.987741	2.301064	0.119429
4	3	relu	[16, 32, 64]	0.10	138.471752	0.658535	0.800143

The best dropout rate is `0.1`

Question 07

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

Logic

1. As we can see from the above reports for less number of hidden layers, the model performs better since the training dataset is of small size and the number of trainable parameters for more neurons in each layer exceeds the expectation of the dataset. So less neurons and less layers is the way to go for small datasets like MNIST.
2. Sigmoid performs better than tanh and relu, although if dataset size is increased, we will more likely see that relu performs better, owing to its ability to prevent over-fitting.
3. Adam performs better optimization(smooth decrease in loss) as compared to SGD.
4. CategoricalCrossentropy loss is appropriate as compared to MSE since this is a classification task not regression.
5. Low dropout rate is favourable for small datasets.
6. For tanh and relu, early stopping occurs leading to fast convergence due to small dataset but lower accuracy than sigmoid due to early stopping criterion

Question 08

viii. With the best set hyper-parameters 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 runs) from all these five run.

In [53]:
```
best_dropout_rate = 0.1
best_activation_function = 'sigmoid'
hidden_neurons = [16,32,64]

learning_rates = [0.01,0.001,0.005,0.0001,0.0005]

result_df_lr = pd.DataFrame(
    columns=[
        'Hidden Layers',
```

```
        'Hidden Neurons',
        'Learning Rate',
        'Dropout Rate',
        'Activation Function',
        'Training Time (in seconds)',
        'Test Loss',
        'Test Accuracy'
    ]
)

for lr in learning_rates:
    model, history, duration = train_model(
        hidden_neurons=hidden_neurons,
        learning_rate=lr,
        dropout_rate=best_dropout_rate,
        activation_func=best_activation_function,
        loss_func=CategoricalCrossentropy,
        optimizer=Adam,
        epochs=100
    )

    test_loss, test_acc = model.evaluate(X_test,y_test)

    print(f"Learning rate = {lr}")
    print(f"Test loss={test_loss} Test accuracy = {test_acc}")
    print(f"Time taken to train the model is {duration} seconds")

    result_df_lr.loc[len(result_df_lr.index)] = [
    len(hidden_neurons),
    str(hidden_neurons),
    lr,
    best_dropout_rate,
    best_activation_function,
    duration,
    test_loss,
    test_acc
    ]

    plot_metrics(history, activation_function, hidden_neurons)
```

Model: "sequential_22"

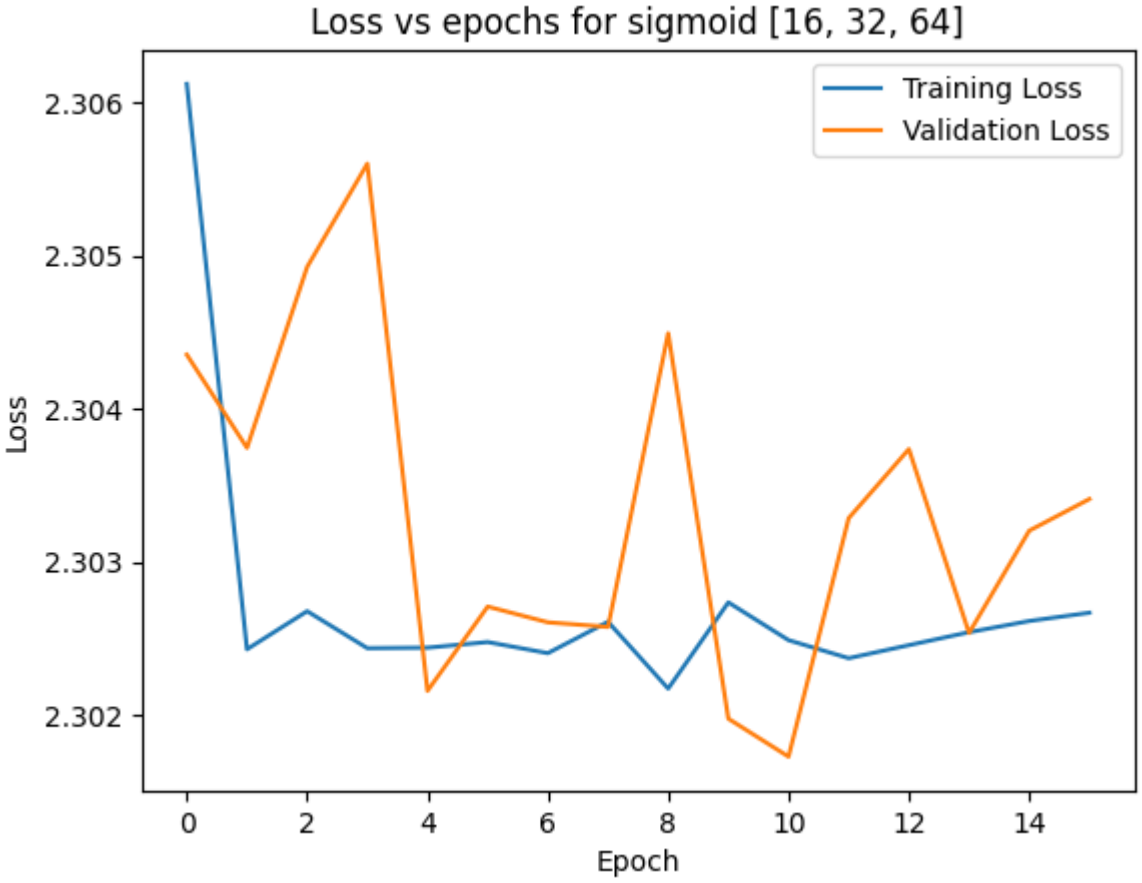
Layer (type)	Output Shape	Param #
flatten_22 (Flatten)	(None, 1024)	0
dense_74 (Dense)	(None, 16)	16,400
dropout_15 (Dropout)	(None, 16)	0
dense_75 (Dense)	(None, 32)	544
dropout_16 (Dropout)	(None, 32)	0
dense_76 (Dense)	(None, 64)	2,112
dropout_17 (Dropout)	(None, 64)	0
dense_77 (Dense)	(None, 10)	650

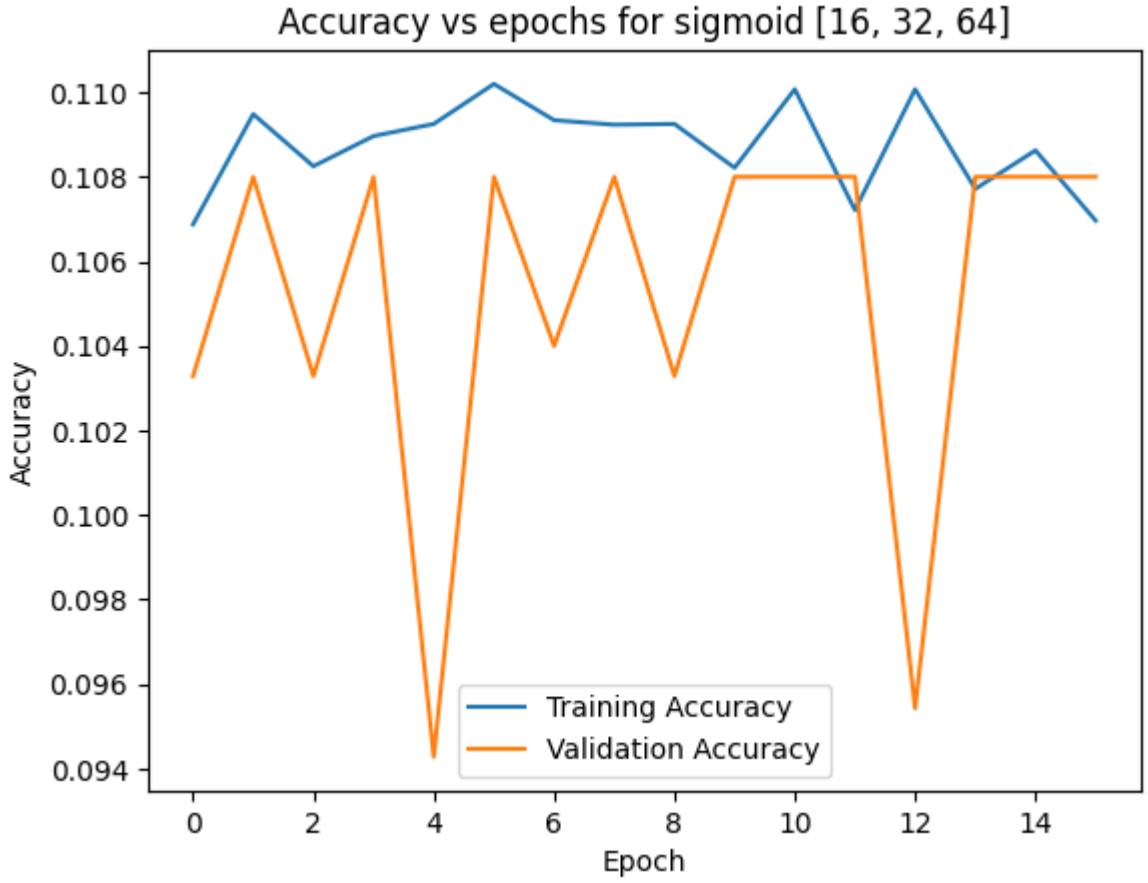
Total params: 19,706 (76.98 KB)

Trainable params: 19,706 (76.98 KB)

Non-trainable params: 0 (0.00 B)

```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1033 - loss: 2.3183 - val_accuracy: 0.1033 - val_loss: 2.3044
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.1074 - loss: 2.3025 - val_accuracy: 0.1080 - val_loss: 2.3037
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1110 - loss: 2.3022 - val_accuracy: 0.1033 - val_loss: 2.3049
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1100 - loss: 2.3025 - val_accuracy: 0.1080 - val_loss: 2.3056
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.1096 - loss: 2.3026 - val_accuracy: 0.0943 - val_loss: 2.3022
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1111 - loss: 2.3021 - val_accuracy: 0.1080 - val_loss: 2.3027
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1097 - loss: 2.3024 - val_accuracy: 0.1040 - val_loss: 2.3026
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1071 - loss: 2.3027 - val_accuracy: 0.1080 - val_loss: 2.3026
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1062 - loss: 2.3025 - val_accuracy: 0.1033 - val_loss: 2.3045
Epoch 10/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1064 - loss: 2.3029 - val_accuracy: 0.1080 - val_loss: 2.3020
Epoch 11/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.1110 - loss: 2.3023 - val_accuracy: 0.1080 - val_loss: 2.3017
Epoch 12/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1103 - loss: 2.3021 - val_accuracy: 0.1080 - val_loss: 2.3033
Epoch 13/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1116 - loss: 2.3024 - val_accuracy: 0.0954 - val_loss: 2.3037
Epoch 14/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.1070 - loss: 2.3026 - val_accuracy: 0.1080 - val_loss: 2.3025
Epoch 15/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1080 - loss: 2.3027 - val_accuracy: 0.1080 - val_loss: 2.3032
Epoch 16/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1093 - loss: 2.3024 - val_accuracy: 0.1080 - val_loss: 2.3034
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.1219 - loss: 2.3007
Learning rate = 0.01
Test loss=2.301191568374634 Test accuracy = 0.11942857503890991
Time taken to train the model is 73.58484530448914 seconds
```


Loss vs epochs for sigmoid [16, 32, 64]
Training Loss
Validation Loss
2.306
2.305
2.304
2.303
2.302
Loss
0
2
4
6
8
10
12
14
Epoch




```
Model: "sequential_23"
```

Layer (type)	Output Shape	Param #
flatten_23 (Flatten)	(None, 1024)	0
dense_78 (Dense)	(None, 16)	16,400
dropout_18 (Dropout)	(None, 16)	0
dense_79 (Dense)	(None, 32)	544
dropout_19 (Dropout)	(None, 32)	0
dense_80 (Dense)	(None, 64)	2,112
dropout_20 (Dropout)	(None, 64)	0
dense_81 (Dense)	(None, 10)	650

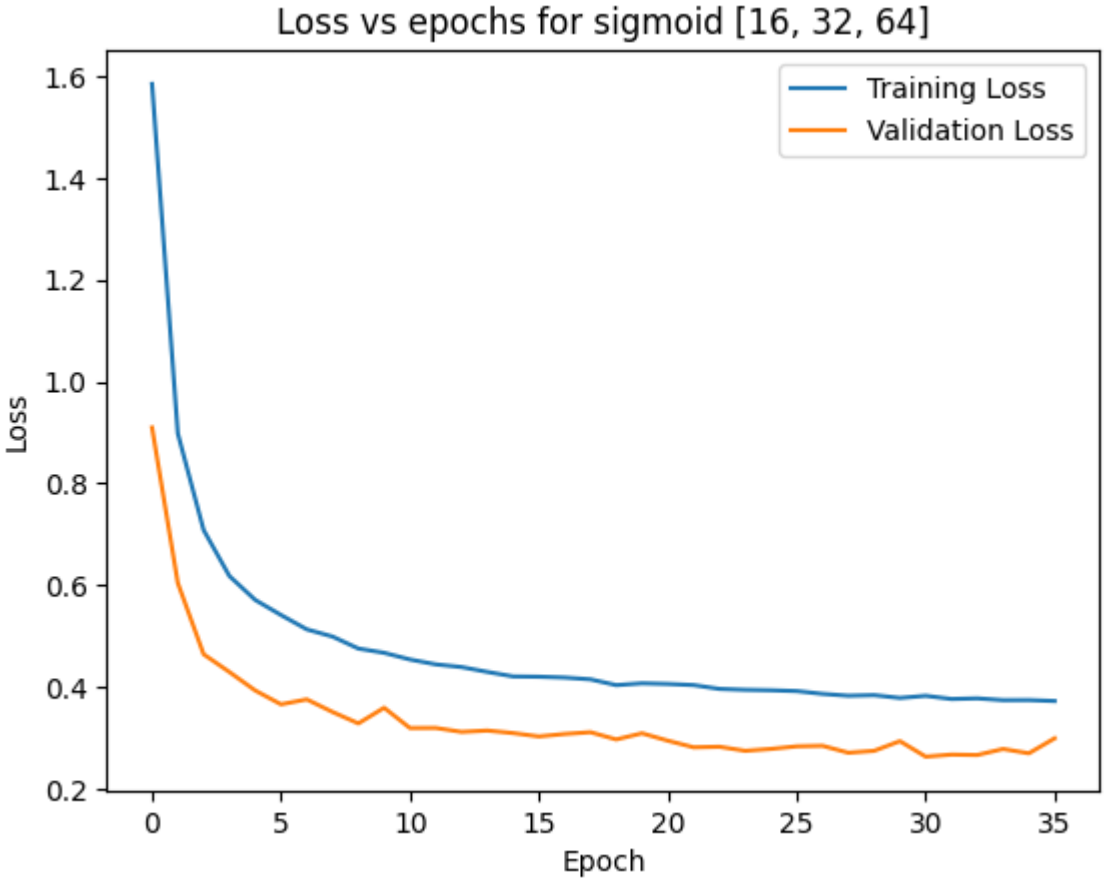
Total params: 19,706 (76.98 KB)

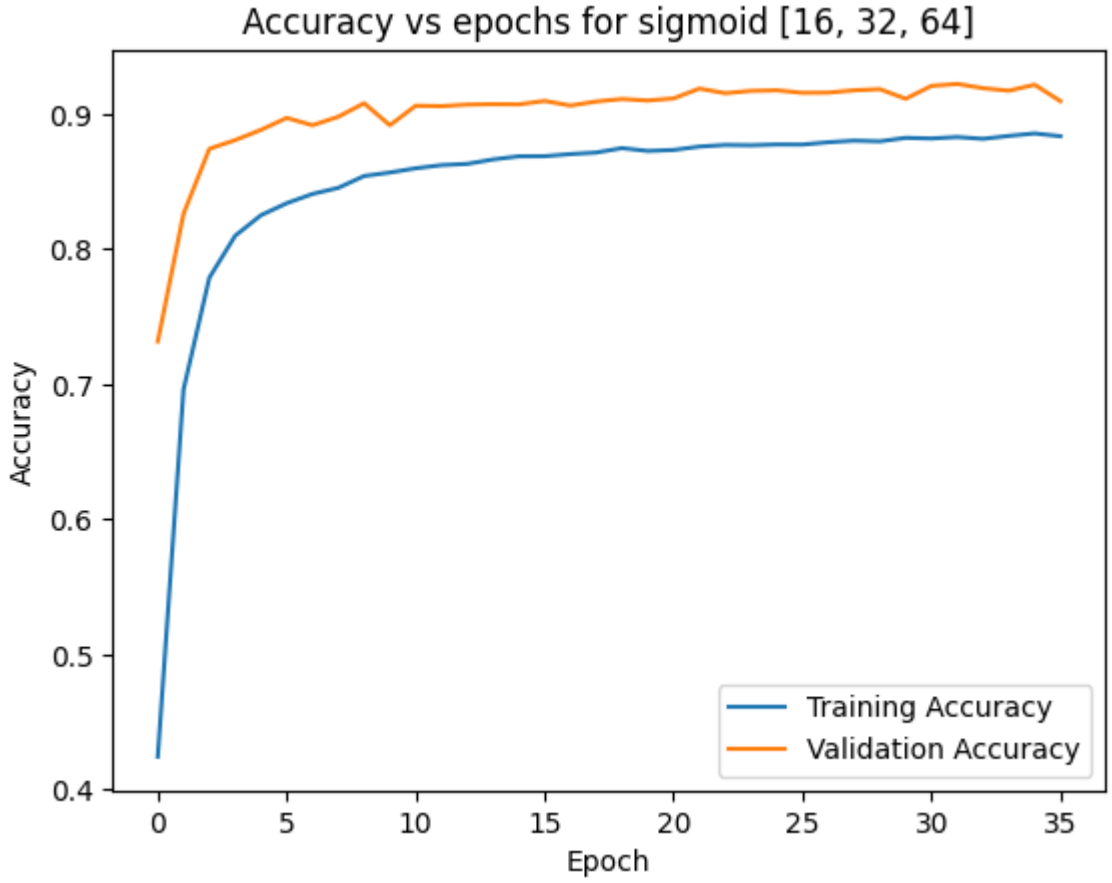
Trainable params: 19,706 (76.98 KB)

Non-trainable params: 0 (0.00 B)

```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.2745 - loss: 1.9652 - val_accuracy: 0.7316 - val_loss: 0.9098
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.6651 - loss: 0.9659 - val_accuracy: 0.8261 - val_loss: 0.6046
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7695 - loss: 0.7357 - val_accuracy: 0.8741 - val_loss: 0.4648
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8049 - loss: 0.6330 - val_accuracy: 0.8806 - val_loss: 0.4299
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8249 - loss: 0.5765 - val_accuracy: 0.8881 - val_loss: 0.3937
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8315 - loss: 0.5479 - val_accuracy: 0.8970 - val_loss: 0.3664
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8389 - loss: 0.5166 - val_accuracy: 0.8916 - val_loss: 0.3762
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8428 - loss: 0.5054 - val_accuracy: 0.8979 - val_loss: 0.3514
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8500 - loss: 0.4850 - val_accuracy: 0.9079 - val_loss: 0.3290
Epoch 10/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8574 - loss: 0.4618 - val_accuracy: 0.8916 - val_loss: 0.3597
Epoch 11/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8597 - loss: 0.4482 - val_accuracy: 0.9060 - val_loss: 0.3199
Epoch 12/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8637 - loss: 0.4390 - val_accuracy: 0.9057 - val_loss: 0.3201
Epoch 13/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.8621 - loss: 0.4413 - val_accuracy: 0.9069 - val_loss: 0.3123
Epoch 14/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8655 - loss: 0.4300 - val_accuracy: 0.9071 - val_loss: 0.3150
Epoch 15/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8708 - loss: 0.4170 - val_accuracy: 0.9070 - val_loss: 0.3100
Epoch 16/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8706 - loss: 0.4131 - val_accuracy: 0.9096 - val_loss: 0.3034
Epoch 17/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8711 - loss: 0.4192 - val_accuracy: 0.9061 - val_loss: 0.3083
Epoch 18/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8699 - loss: 0.4205 - val_accuracy: 0.9091 - val_loss: 0.3117
Epoch 19/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.8740 - loss: 0.4013 - val_accuracy: 0.9110 - val_loss: 0.2975
Epoch 20/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8728 - loss: 0.4069 - val_accuracy: 0.9099 - val_loss: 0.3096
Epoch 21/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8740 - loss: 0.4061 - val_accuracy: 0.9114 - val_loss: 0.2951
Epoch 22/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8774 - loss: 0.3996 - val_accuracy: 0.9187 - val_loss: 0.2821
Epoch 23/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.8766 - loss: 0.4017 - val_accuracy: 0.9153 - val_loss: 0.2832
```

```
Epoch 24/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8753 - loss: 0.3954 - val_accuracy: 0.9170 - val_loss: 0.2752
Epoch 25/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8738 - loss: 0.4073 - val_accuracy: 0.9174 - val_loss: 0.2789
Epoch 26/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8786 - loss: 0.3915 - val_accuracy: 0.9156 - val_loss: 0.2840
Epoch 27/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8803 - loss: 0.3831 - val_accuracy: 0.9157 - val_loss: 0.2850
Epoch 28/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8795 - loss: 0.3879 - val_accuracy: 0.9174 - val_loss: 0.2714
Epoch 29/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8771 - loss: 0.3892 - val_accuracy: 0.9183 - val_loss: 0.2752
Epoch 30/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8819 - loss: 0.3805 - val_accuracy: 0.9111 - val_loss: 0.2942
Epoch 31/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8829 - loss: 0.3784 - val_accuracy: 0.9207 - val_loss: 0.2637
Epoch 32/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8824 - loss: 0.3797 - val_accuracy: 0.9221 - val_loss: 0.2678
Epoch 33/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8801 - loss: 0.3805 - val_accuracy: 0.9190 - val_loss: 0.2670
Epoch 34/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8848 - loss: 0.3703 - val_accuracy: 0.9171 - val_loss: 0.2788
Epoch 35/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8864 - loss: 0.3721 - val_accuracy: 0.9216 - val_loss: 0.2703
Epoch 36/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8823 - loss: 0.3753 - val_accuracy: 0.9094 - val_loss: 0.2998
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9215 - loss: 0.2569
Learning rate = 0.001
Test loss=0.2631760239601135 Test accuracy = 0.920285701751709
Time taken to train the model is 186.7718403339386 seconds
```


Loss vs epochs for sigmoid [16, 32, 64]
Training Loss
Validation Loss
Loss
Epoch
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0
5
10
15
20
25
30
35




```
Model: "sequential_24"
```

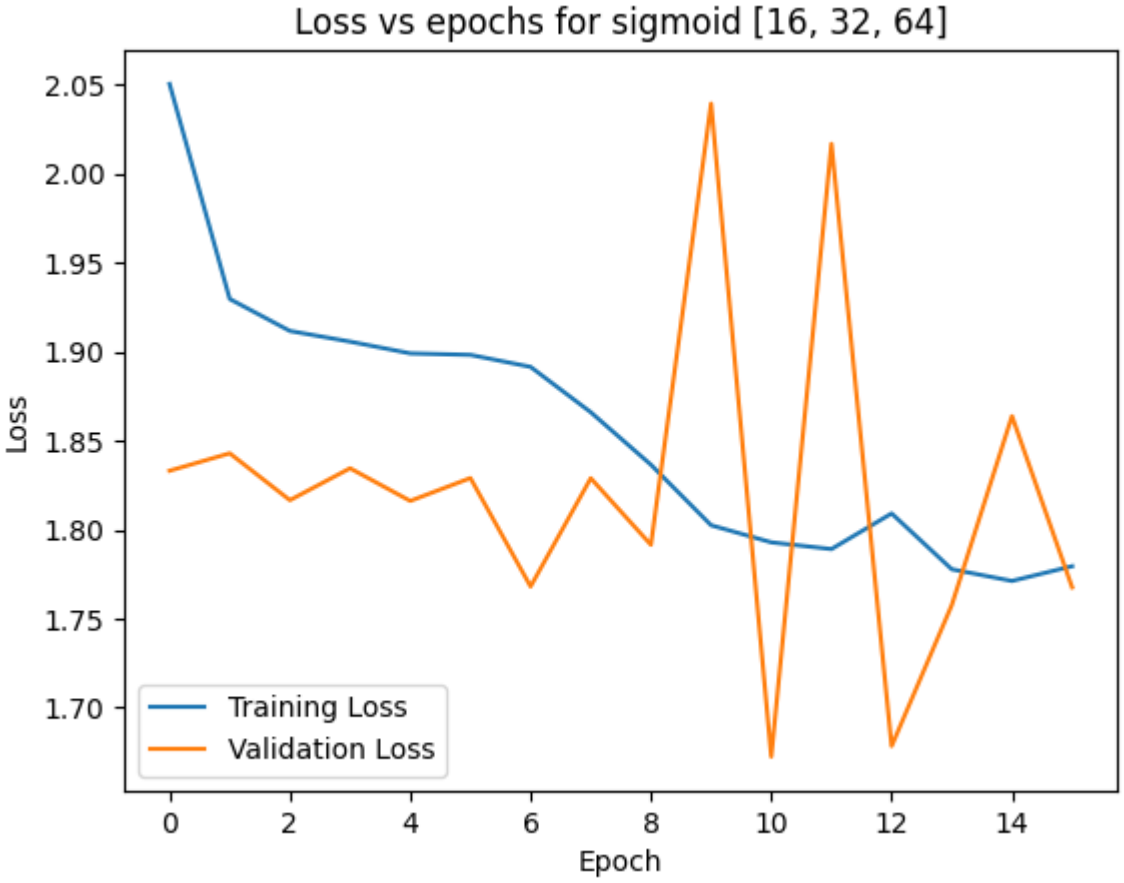
Layer (type)	Output Shape	Param #
flatten_24 (Flatten)	(None, 1024)	0
dense_82 (Dense)	(None, 16)	16,400
dropout_21 (Dropout)	(None, 16)	0
dense_83 (Dense)	(None, 32)	544
dropout_22 (Dropout)	(None, 32)	0
dense_84 (Dense)	(None, 64)	2,112
dropout_23 (Dropout)	(None, 64)	0
dense_85 (Dense)	(None, 10)	650

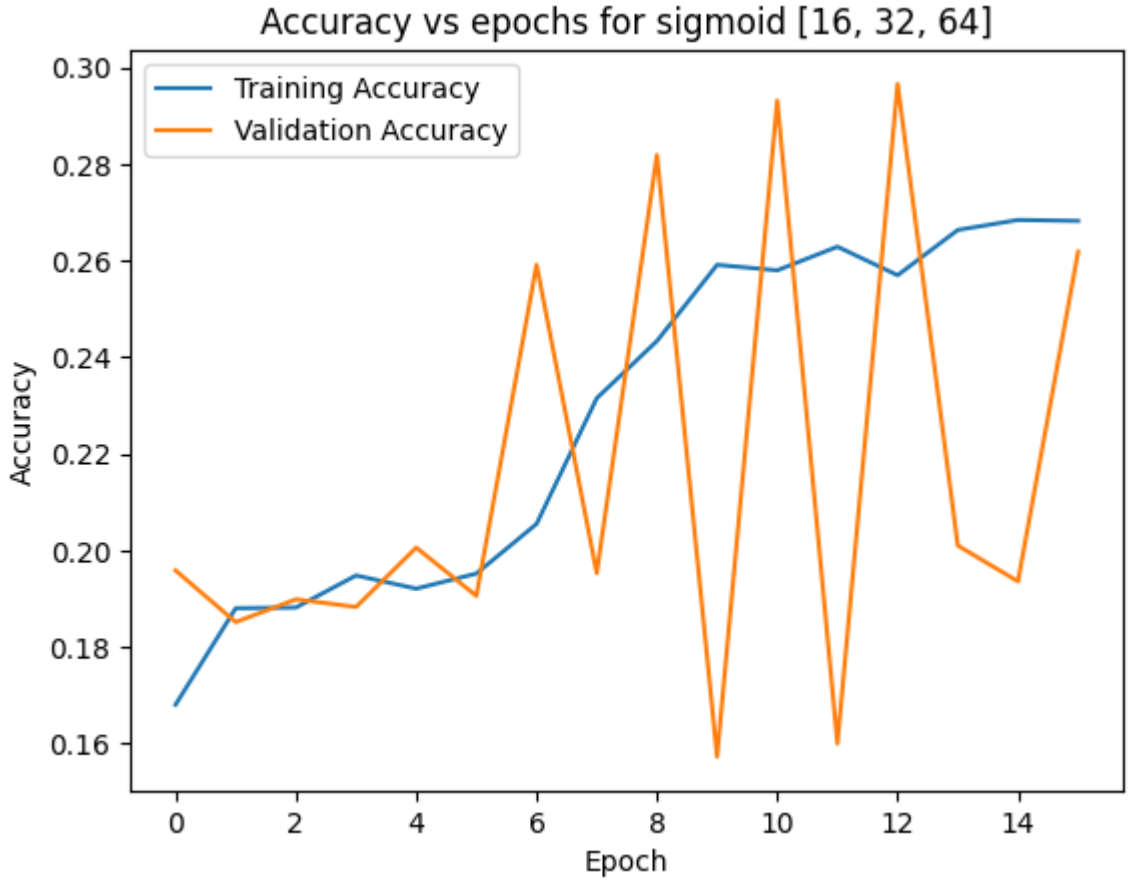
Total params: 19,706 (76.98 KB)

Trainable params: 19,706 (76.98 KB)

Non-trainable params: 0 (0.00 B)

```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.1384 - loss: 2.1863 - val_accuracy: 0.1959 - val_loss: 1.8331
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.1881 - loss: 1.9369 - val_accuracy: 0.1851 - val_loss: 1.8429
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.1908 - loss: 1.9149 - val_accuracy: 0.1899 - val_loss: 1.8165
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.1948 - loss: 1.9022 - val_accuracy: 0.1883 - val_loss: 1.8345
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.1919 - loss: 1.8962 - val_accuracy: 0.2006 - val_loss: 1.8161
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.1952 - loss: 1.8931 - val_accuracy: 0.1906 - val_loss: 1.8290
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.2011 - loss: 1.8921 - val_accuracy: 0.2591 - val_loss: 1.7680
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.2250 - loss: 1.8747 - val_accuracy: 0.1953 - val_loss: 1.8289
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.2396 - loss: 1.8514 - val_accuracy: 0.2819 - val_loss: 1.7915
Epoch 10/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.2550 - loss: 1.8020 - val_accuracy: 0.1573 - val_loss: 2.0394
Epoch 11/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.2573 - loss: 1.7974 - val_accuracy: 0.2931 - val_loss: 1.6724
Epoch 12/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.2638 - loss: 1.7868 - val_accuracy: 0.1600 - val_loss: 2.0168
Epoch 13/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.2562 - loss: 1.8041 - val_accuracy: 0.2966 - val_loss: 1.6783
Epoch 14/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.2663 - loss: 1.7814 - val_accuracy: 0.2010 - val_loss: 1.7576
Epoch 15/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.2718 - loss: 1.7680 - val_accuracy: 0.1936 - val_loss: 1.8638
Epoch 16/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.2632 - loss: 1.7932 - val_accuracy: 0.2619 - val_loss: 1.7675
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.3042 - loss: 1.6644
Learning rate = 0.005
Test loss=1.6627888679504395 Test accuracy = 0.29671427607536316
Time taken to train the model is 75.20755457878113 seconds
```


Loss vs epochs for sigmoid [16, 32, 64]
2.05
2.00
1.95
1.90
1.85
1.80
1.75
1.70
Loss
0
2
4
6
8
10
12
14
Epoch
Training Loss
Validation Loss




```
Model: "sequential_25"
```

Layer (type)	Output Shape	Param #
flatten_25 (Flatten)	(None, 1024)	0
dense_86 (Dense)	(None, 16)	16,400
dropout_24 (Dropout)	(None, 16)	0
dense_87 (Dense)	(None, 32)	544
dropout_25 (Dropout)	(None, 32)	0
dense_88 (Dense)	(None, 64)	2,112
dropout_26 (Dropout)	(None, 64)	0
dense_89 (Dense)	(None, 10)	650

Total params: 19,706 (76.98 KB)

Trainable params: 19,706 (76.98 KB)

Non-trainable params: 0 (0.00 B)

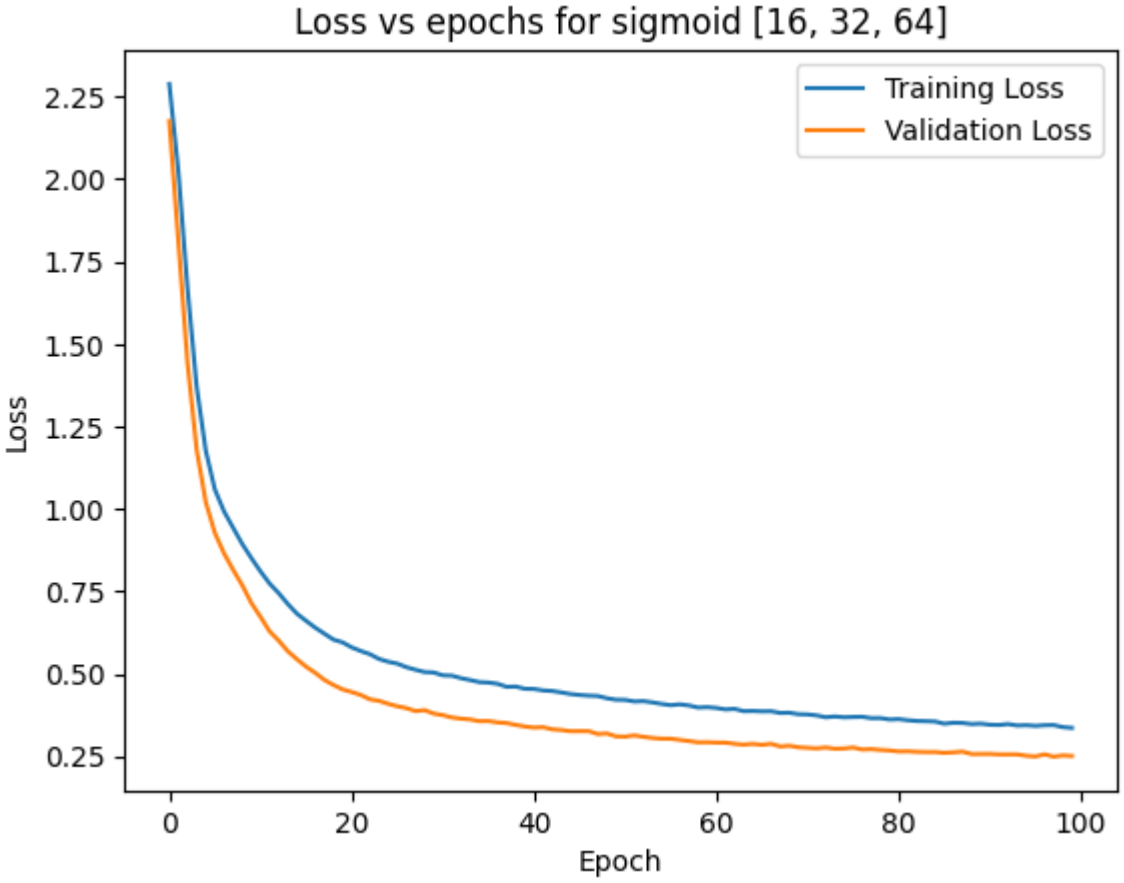
```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.1175 - loss: 2.3349 - val_accuracy: 0.2574 - val_loss: 2.1762
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 8s 2ms/step - accuracy: 0.2592 - loss: 2.1093 - val_accuracy: 0.4377 - val_loss: 1.8098
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.3984 - loss: 1.7545 - val_accuracy: 0.5507 - val_loss: 1.4426
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.5074 - loss: 1.4313 - val_accuracy: 0.6306 - val_loss: 1.1801
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.5710 - loss: 1.2213 - val_accuracy: 0.7029 - val_loss: 1.0208
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.6147 - loss: 1.0800 - val_accuracy: 0.7191 - val_loss: 0.9271
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.6410 - loss: 1.0077 - val_accuracy: 0.7576 - val_loss: 0.8654
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.6644 - loss: 0.9574 - val_accuracy: 0.7600 - val_loss: 0.8153
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.6880 - loss: 0.9034 - val_accuracy: 0.7760 - val_loss: 0.7686
Epoch 10/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.7108 - loss: 0.8617 - val_accuracy: 0.7976 - val_loss: 0.7145
Epoch 11/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.7307 - loss: 0.8184 - val_accuracy: 0.8049 - val_loss: 0.6726
Epoch 12/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7474 - loss: 0.7769 - val_accuracy: 0.8257 - val_loss: 0.6288
Epoch 13/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.7546 - loss: 0.7600 - val_accuracy: 0.8310 - val_loss: 0.6005
Epoch 14/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7724 - loss: 0.7123 - val_accuracy: 0.8420 - val_loss: 0.5682
Epoch 15/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7837 - loss: 0.6930 - val_accuracy: 0.8503 - val_loss: 0.5440
Epoch 16/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.7941 - loss: 0.6685 - val_accuracy: 0.8567 - val_loss: 0.5216
Epoch 17/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7984 - loss: 0.6453 - val_accuracy: 0.8611 - val_loss: 0.5028
Epoch 18/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8069 - loss: 0.6324 - val_accuracy: 0.8667 - val_loss: 0.4815
Epoch 19/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8155 - loss: 0.6002 - val_accuracy: 0.8720 - val_loss: 0.4660
Epoch 20/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8203 - loss: 0.5920 - val_accuracy: 0.8754 - val_loss: 0.4526
Epoch 21/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8237 - loss: 0.5782 - val_accuracy: 0.8770 - val_loss: 0.4447
Epoch 22/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.8304 - loss: 0.5657 - val_accuracy: 0.8814 - val_loss: 0.4356
Epoch 23/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8315 - loss: 0.5588 - val_accuracy: 0.8853 - val_loss: 0.4225
```

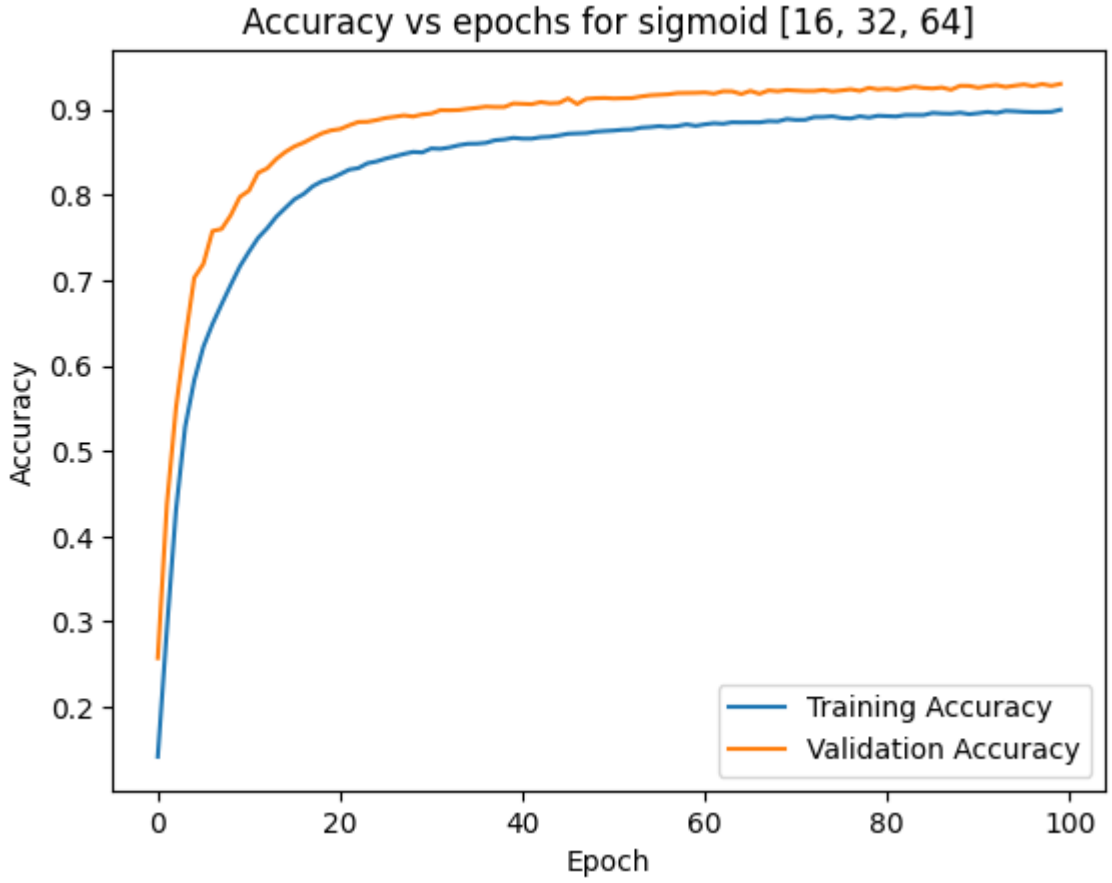
```
Epoch 24/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8370 - loss: 0.5414 - val_accuracy: 0.8856 - val_loss: 0.4178
Epoch 25/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8388 - loss: 0.5352 - val_accuracy: 0.8876 - val_loss: 0.4089
Epoch 26/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8412 - loss: 0.5313 - val_accuracy: 0.8900 - val_loss: 0.4015
Epoch 27/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8435 - loss: 0.5237 - val_accuracy: 0.8914 - val_loss: 0.3959
Epoch 28/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8506 - loss: 0.5052 - val_accuracy: 0.8930 - val_loss: 0.3867
Epoch 29/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8528 - loss: 0.4992 - val_accuracy: 0.8917 - val_loss: 0.3896
Epoch 30/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8482 - loss: 0.5073 - val_accuracy: 0.8941 - val_loss: 0.3792
Epoch 31/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8550 - loss: 0.4946 - val_accuracy: 0.8951 - val_loss: 0.3745
Epoch 32/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8488 - loss: 0.5020 - val_accuracy: 0.8991 - val_loss: 0.3673
Epoch 33/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8534 - loss: 0.4922 - val_accuracy: 0.8990 - val_loss: 0.3638
Epoch 34/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8579 - loss: 0.4831 - val_accuracy: 0.8993 - val_loss: 0.3614
Epoch 35/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8589 - loss: 0.4773 - val_accuracy: 0.9007 - val_loss: 0.3563
Epoch 36/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8591 - loss: 0.4743 - val_accuracy: 0.9019 - val_loss: 0.3567
Epoch 37/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8605 - loss: 0.4721 - val_accuracy: 0.9033 - val_loss: 0.3524
Epoch 38/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8644 - loss: 0.4612 - val_accuracy: 0.9029 - val_loss: 0.3504
Epoch 39/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8641 - loss: 0.4638 - val_accuracy: 0.9029 - val_loss: 0.3449
Epoch 40/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8674 - loss: 0.4502 - val_accuracy: 0.9070 - val_loss: 0.3400
Epoch 41/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8636 - loss: 0.4621 - val_accuracy: 0.9066 - val_loss: 0.3368
Epoch 42/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8645 - loss: 0.4489 - val_accuracy: 0.9061 - val_loss: 0.3380
Epoch 43/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8680 - loss: 0.4469 - val_accuracy: 0.9086 - val_loss: 0.3315
Epoch 44/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8687 - loss: 0.4385 - val_accuracy: 0.9071 - val_loss: 0.3294
Epoch 45/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8710 - loss: 0.4347 - val_accuracy: 0.9076 - val_loss: 0.3258
Epoch 46/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8711 - loss: 0.4353 - val_accuracy: 0.9129 - val_loss: 0.3260
```

```
Epoch 47/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8699 - loss: 0.4375 - val_accuracy: 0.9063 - val_loss: 0.3260
Epoch 48/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8709 - loss: 0.4336 - val_accuracy: 0.9124 - val_loss: 0.3171
Epoch 49/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8760 - loss: 0.4211 - val_accuracy: 0.9131 - val_loss: 0.3189
Epoch 50/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8771 - loss: 0.4150 - val_accuracy: 0.9134 - val_loss: 0.3100
Epoch 51/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8735 - loss: 0.4255 - val_accuracy: 0.9129 - val_loss: 0.3092
Epoch 52/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8764 - loss: 0.4174 - val_accuracy: 0.9131 - val_loss: 0.3131
Epoch 53/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8734 - loss: 0.4239 - val_accuracy: 0.9133 - val_loss: 0.3087
Epoch 54/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8809 - loss: 0.4133 - val_accuracy: 0.9151 - val_loss: 0.3053
Epoch 55/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8793 - loss: 0.4091 - val_accuracy: 0.9166 - val_loss: 0.3030
Epoch 56/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.8808 - loss: 0.4080 - val_accuracy: 0.9171 - val_loss: 0.3030
Epoch 57/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8771 - loss: 0.4116 - val_accuracy: 0.9177 - val_loss: 0.2994
Epoch 58/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8812 - loss: 0.4054 - val_accuracy: 0.9191 - val_loss: 0.2959
Epoch 59/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8810 - loss: 0.4009 - val_accuracy: 0.9193 - val_loss: 0.2920
Epoch 60/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8817 - loss: 0.3977 - val_accuracy: 0.9194 - val_loss: 0.2922
Epoch 61/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.8847 - loss: 0.3874 - val_accuracy: 0.9199 - val_loss: 0.2913
Epoch 62/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8840 - loss: 0.3886 - val_accuracy: 0.9184 - val_loss: 0.2911
Epoch 63/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8839 - loss: 0.3867 - val_accuracy: 0.9213 - val_loss: 0.2875
Epoch 64/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.8843 - loss: 0.3846 - val_accuracy: 0.9211 - val_loss: 0.2850
Epoch 65/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8864 - loss: 0.3846 - val_accuracy: 0.9180 - val_loss: 0.2872
Epoch 66/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8843 - loss: 0.3860 - val_accuracy: 0.9217 - val_loss: 0.2844
Epoch 67/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8836 - loss: 0.3932 - val_accuracy: 0.9183 - val_loss: 0.2875
Epoch 68/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8862 - loss: 0.3799 - val_accuracy: 0.9223 - val_loss: 0.2795
Epoch 69/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8858 - loss: 0.3847 - val_accuracy: 0.9213 - val_loss: 0.2816
```

```
Epoch 70/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8888 - loss: 0.3739 - val_accuracy: 0.9227 - val_loss: 0.2770
Epoch 71/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8877 - loss: 0.3788 - val_accuracy: 0.9221 - val_loss: 0.2754
Epoch 72/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8874 - loss: 0.3751 - val_accuracy: 0.9216 - val_loss: 0.2737
Epoch 73/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8924 - loss: 0.3658 - val_accuracy: 0.9216 - val_loss: 0.2763
Epoch 74/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8909 - loss: 0.3745 - val_accuracy: 0.9229 - val_loss: 0.2726
Epoch 75/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8923 - loss: 0.3673 - val_accuracy: 0.9211 - val_loss: 0.2734
Epoch 76/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8904 - loss: 0.3679 - val_accuracy: 0.9223 - val_loss: 0.2763
Epoch 77/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8893 - loss: 0.3699 - val_accuracy: 0.9236 - val_loss: 0.2703
Epoch 78/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8914 - loss: 0.3639 - val_accuracy: 0.9217 - val_loss: 0.2714
Epoch 79/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 8s 2ms/step - accuracy: 0.8906 - loss: 0.3622 - val_accuracy: 0.9253 - val_loss: 0.2689
Epoch 80/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8917 - loss: 0.3632 - val_accuracy: 0.9234 - val_loss: 0.2674
Epoch 81/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8888 - loss: 0.3740 - val_accuracy: 0.9243 - val_loss: 0.2643
Epoch 82/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8930 - loss: 0.3551 - val_accuracy: 0.9230 - val_loss: 0.2648
Epoch 83/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8950 - loss: 0.3568 - val_accuracy: 0.9249 - val_loss: 0.2631
Epoch 84/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8937 - loss: 0.3539 - val_accuracy: 0.9267 - val_loss: 0.2624
Epoch 85/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8942 - loss: 0.3498 - val_accuracy: 0.9251 - val_loss: 0.2625
Epoch 86/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8949 - loss: 0.3487 - val_accuracy: 0.9246 - val_loss: 0.2603
Epoch 87/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8970 - loss: 0.3461 - val_accuracy: 0.9257 - val_loss: 0.2616
Epoch 88/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8946 - loss: 0.3510 - val_accuracy: 0.9230 - val_loss: 0.2639
Epoch 89/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8967 - loss: 0.3489 - val_accuracy: 0.9277 - val_loss: 0.2564
Epoch 90/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8938 - loss: 0.3551 - val_accuracy: 0.9276 - val_loss: 0.2567
Epoch 91/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8954 - loss: 0.3448 - val_accuracy: 0.9254 - val_loss: 0.2568
Epoch 92/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8967 - loss: 0.3483 - val_accuracy: 0.9273 - val_loss: 0.2553
```

```
Epoch 93/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8946 - loss: 0.3482 - val_accuracy: 0.9284 - val_loss: 0.2554
Epoch 94/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8988 - loss: 0.3428 - val_accuracy: 0.9263 - val_loss: 0.2554
Epoch 95/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8988 - loss: 0.3381 - val_accuracy: 0.9279 - val_loss: 0.2512
Epoch 96/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8955 - loss: 0.3461 - val_accuracy: 0.9293 - val_loss: 0.2491
Epoch 97/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8988 - loss: 0.3398 - val_accuracy: 0.9273 - val_loss: 0.2556
Epoch 98/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8973 - loss: 0.3409 - val_accuracy: 0.9297 - val_loss: 0.2485
Epoch 99/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8979 - loss: 0.3399 - val_accuracy: 0.9277 - val_loss: 0.2522
Epoch 100/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8989 - loss: 0.3356 - val_accuracy: 0.9299 - val_loss: 0.2505
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9338 - loss: 0.2457
Learning rate = 0.0001
Test loss=0.25098052620887756 Test accuracy = 0.9292857050895691
Time taken to train the model is 554.5450575351715 seconds
```


Loss vs epochs for sigmoid [16, 32, 64]
Training Loss
Validation Loss
Loss
Epoch
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0
20
40
60
80
100




```
Model: "sequential_26"
```

Layer (type)	Output Shape	Param #
flatten_26 (Flatten)	(None, 1024)	0
dense_90 (Dense)	(None, 16)	16,400
dropout_27 (Dropout)	(None, 16)	0
dense_91 (Dense)	(None, 32)	544
dropout_28 (Dropout)	(None, 32)	0
dense_92 (Dense)	(None, 64)	2,112
dropout_29 (Dropout)	(None, 64)	0
dense_93 (Dense)	(None, 10)	650

Total params: 19,706 (76.98 KB)

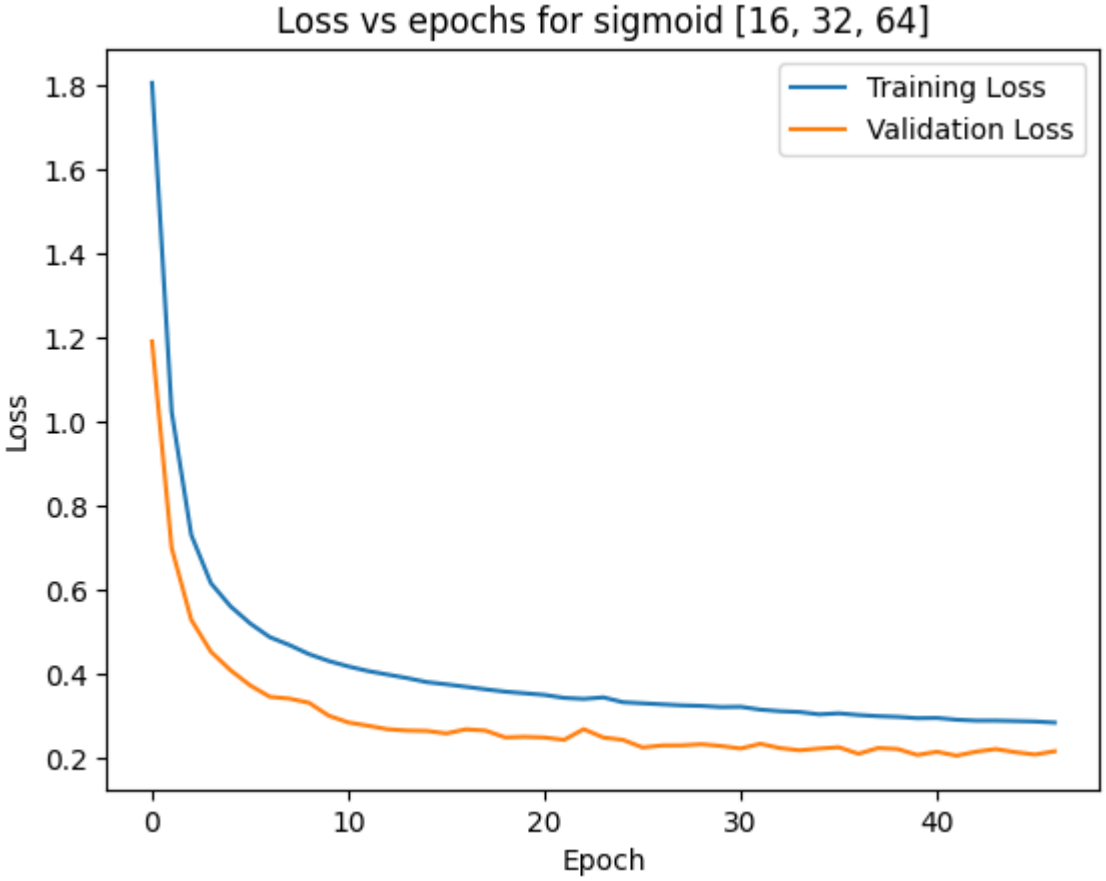
Trainable params: 19,706 (76.98 KB)

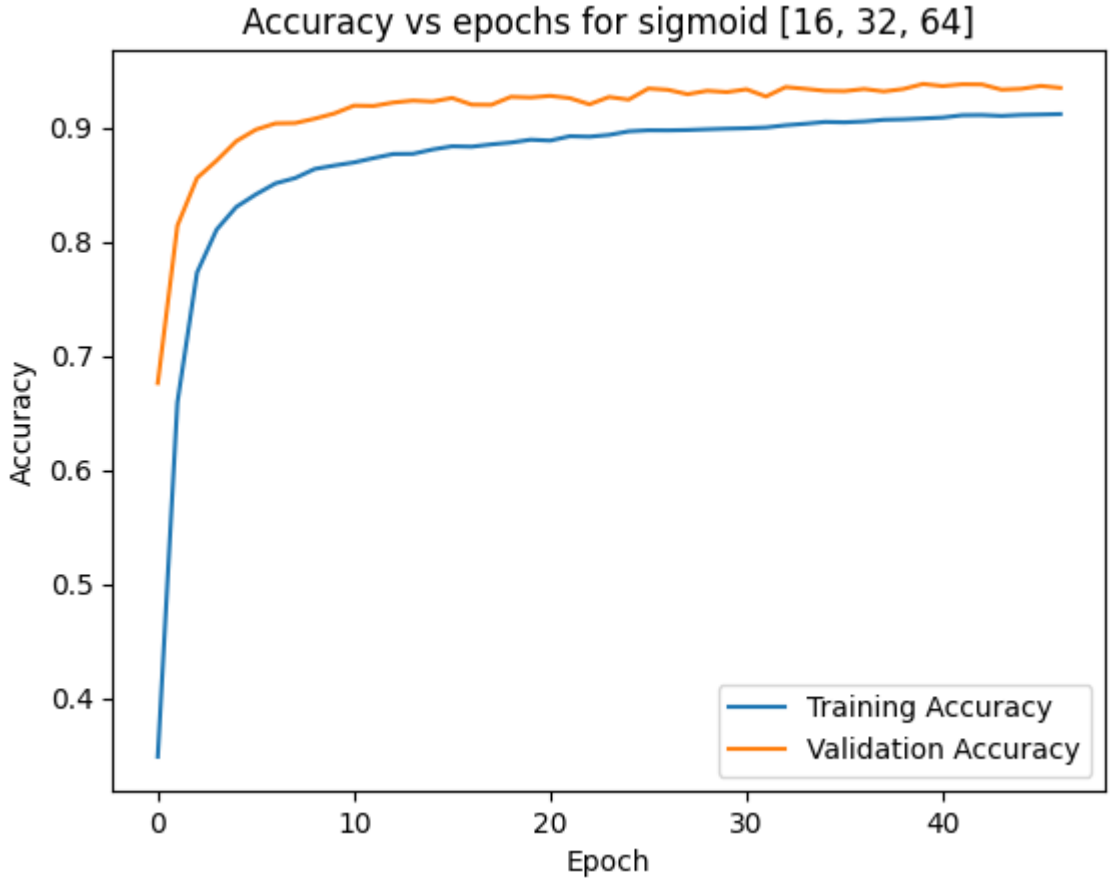
Non-trainable params: 0 (0.00 B)

```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.2199 - loss: 2.1256 - val_accuracy: 0.6759 - val_loss: 1.1909
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.6131 - loss: 1.1406 - val_accuracy: 0.8139 - val_loss: 0.7003
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.7564 - loss: 0.7731 - val_accuracy: 0.8556 - val_loss: 0.5296
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8025 - loss: 0.6327 - val_accuracy: 0.8709 - val_loss: 0.4540
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8264 - loss: 0.5732 - val_accuracy: 0.8879 - val_loss: 0.4100
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8400 - loss: 0.5258 - val_accuracy: 0.8980 - val_loss: 0.3741
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8512 - loss: 0.4889 - val_accuracy: 0.9034 - val_loss: 0.3468
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8541 - loss: 0.4719 - val_accuracy: 0.9037 - val_loss: 0.3426
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8629 - loss: 0.4522 - val_accuracy: 0.9076 - val_loss: 0.3327
Epoch 10/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.8645 - loss: 0.4338 - val_accuracy: 0.9121 - val_loss: 0.3021
Epoch 11/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8678 - loss: 0.4283 - val_accuracy: 0.9190 - val_loss: 0.2859
Epoch 12/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8733 - loss: 0.4042 - val_accuracy: 0.9187 - val_loss: 0.2782
Epoch 13/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8734 - loss: 0.4071 - val_accuracy: 0.9217 - val_loss: 0.2698
Epoch 14/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8752 - loss: 0.3954 - val_accuracy: 0.9234 - val_loss: 0.2668
Epoch 15/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8791 - loss: 0.3873 - val_accuracy: 0.9226 - val_loss: 0.2658
Epoch 16/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8840 - loss: 0.3724 - val_accuracy: 0.9257 - val_loss: 0.2598
Epoch 17/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.8837 - loss: 0.3638 - val_accuracy: 0.9199 - val_loss: 0.2696
Epoch 18/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8854 - loss: 0.3687 - val_accuracy: 0.9197 - val_loss: 0.2665
Epoch 19/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.8872 - loss: 0.3584 - val_accuracy: 0.9267 - val_loss: 0.2503
Epoch 20/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8886 - loss: 0.3580 - val_accuracy: 0.9261 - val_loss: 0.2517
Epoch 21/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8905 - loss: 0.3442 - val_accuracy: 0.9276 - val_loss: 0.2502
Epoch 22/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8941 - loss: 0.3399 - val_accuracy: 0.9256 - val_loss: 0.2446
Epoch 23/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8914 - loss: 0.3409 - val_accuracy: 0.9201 - val_loss: 0.2700
```

```
Epoch 24/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8940 - loss: 0.3410 - val_accuracy: 0.9266 - val_loss: 0.2502
Epoch 25/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8988 - loss: 0.3321 - val_accuracy: 0.9243 - val_loss: 0.2446
Epoch 26/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8968 - loss: 0.3344 - val_accuracy: 0.9340 - val_loss: 0.2266
Epoch 27/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8988 - loss: 0.3231 - val_accuracy: 0.9329 - val_loss: 0.2313
Epoch 28/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8989 - loss: 0.3236 - val_accuracy: 0.9290 - val_loss: 0.2315
Epoch 29/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9008 - loss: 0.3189 - val_accuracy: 0.9320 - val_loss: 0.2342
Epoch 30/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8971 - loss: 0.3266 - val_accuracy: 0.9309 - val_loss: 0.2300
Epoch 31/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9009 - loss: 0.3171 - val_accuracy: 0.9333 - val_loss: 0.2240
Epoch 32/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.9024 - loss: 0.3088 - val_accuracy: 0.9270 - val_loss: 0.2352
Epoch 33/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9023 - loss: 0.3124 - val_accuracy: 0.9353 - val_loss: 0.2250
Epoch 34/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9034 - loss: 0.3058 - val_accuracy: 0.9337 - val_loss: 0.2198
Epoch 35/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9056 - loss: 0.3053 - val_accuracy: 0.9320 - val_loss: 0.2239
Epoch 36/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.9068 - loss: 0.3001 - val_accuracy: 0.9317 - val_loss: 0.2270
Epoch 37/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9029 - loss: 0.3093 - val_accuracy: 0.9336 - val_loss: 0.2115
Epoch 38/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9069 - loss: 0.3009 - val_accuracy: 0.9314 - val_loss: 0.2251
Epoch 39/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9065 - loss: 0.2977 - val_accuracy: 0.9336 - val_loss: 0.2229
Epoch 40/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.9086 - loss: 0.2919 - val_accuracy: 0.9380 - val_loss: 0.2087
Epoch 41/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9083 - loss: 0.2994 - val_accuracy: 0.9361 - val_loss: 0.2163
Epoch 42/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9071 - loss: 0.2999 - val_accuracy: 0.9377 - val_loss: 0.2070
Epoch 43/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9091 - loss: 0.2935 - val_accuracy: 0.9376 - val_loss: 0.2165
Epoch 44/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9100 - loss: 0.2945 - val_accuracy: 0.9330 - val_loss: 0.2227
Epoch 45/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9101 - loss: 0.2899 - val_accuracy: 0.9337 - val_loss: 0.2153
Epoch 46/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.9108 - loss: 0.2875 - val_accuracy: 0.9363 - val_loss: 0.2097
```

```
Epoch 47/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9142 - loss: 0.2818 - val_accuracy: 0.9346 - val_loss: 0.2176
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9422 - loss: 0.2004
Learning rate = 0.0005
Test loss=0.20578503608703613 Test accuracy = 0.9392856955528259
Time taken to train the model is 262.3124701976776 seconds
```





In [54]:
```
result_df_lr
```

Out[54]:

	Hidden Layers	Hidden Neurons	Learning Rate	Dropout Rate	Activation Function	Training Time (in seconds)	Test Loss	Test Accuracy
0	3	[16, 32, 64]	0.0100	0.1	sigmoid	73.584845	2.301192	0.119429
1	3	[16, 32, 64]	0.0010	0.1	sigmoid	186.771840	0.263176	0.920286
2	3	[16, 32, 64]	0.0050	0.1	sigmoid	75.207555	1.662789	0.296714
3	3	[16, 32, 64]	0.0001	0.1	sigmoid	554.545058	0.250981	0.929286
4	3	[16, 32, 64]	0.0005	0.1	sigmoid	262.312470	0.205785	0.939286

As we can see from above the best learning rate is `0.0005`

Question 09

ix. Create five images (of size 28 × 28) containing a digit of your own handwriting and test whether your trained classifier can predict it or not.

In [55]:
```
# Install necessary libraries

!pip install opencv-python
!pip install Pillow
```

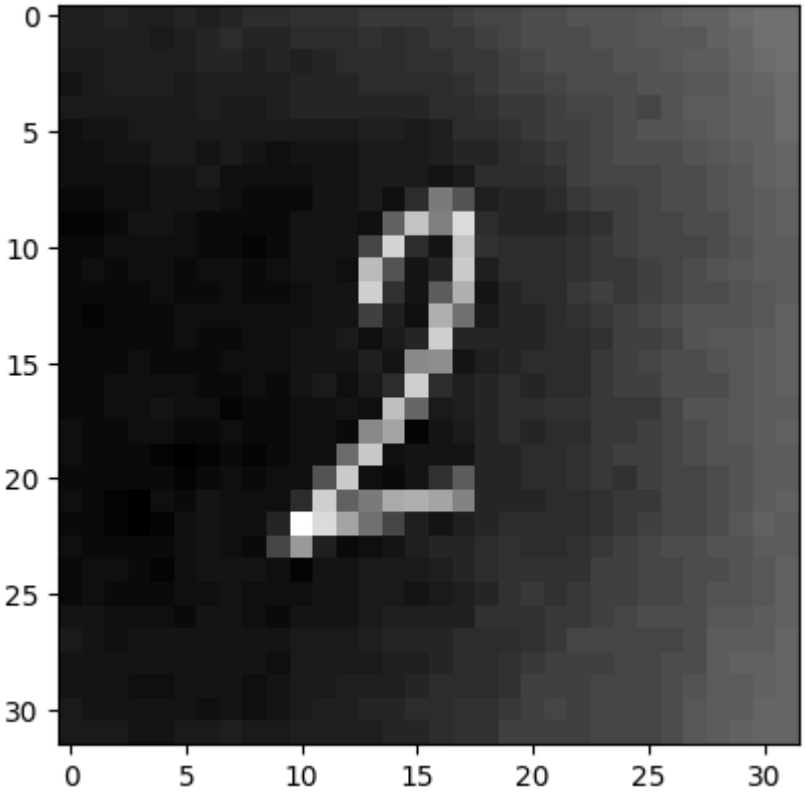
```
Requirement already satisfied: opencv-python in /usr/local/lib/python3.10/dist-packages (4.10.0.84)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from opencv-python) (1.26.4)
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages (10.4.0)
```

In [56]:
```
# Import the libraries

import os
import numpy as np
from PIL import Image
import cv2
```

In [112…
```
def process_image(image_path):
  img = Image.open(image_path).convert('L') # Convert to gray-scale
  img = img.resize((32,32))
  img_array = np.array(img)

  return img_array

# Testing it out
img_array = process_image('/content/sample_images/two.png')
print(img_array)
```

```
[[164 164 163 ... 151 150 150]
 [164 164 164 ... 152 150 150]
 [165 165 164 ... 152 151 150]
 ...
 [166 166 166 ... 154 153 152]
 [165 166 166 ... 154 152 152]
 [165 165 165 ... 153 152 152]]
```

In [113…
```
plt.imshow(img_array, cmap='Greys')
plt.show()
```



The resolution of the image is not upto the mark

In [114…

```
IMG_DIR = '/content/sample_images'

images = []

image_files = [f for f in os.listdir(IMG_DIR) if os.path.isfile(os.path.join(IMG_DIR,f))]

images = []

for image_file in image_files:
  image_path = os.path.join(IMG_DIR,image_file)
  image_array = process_image(image_path)
  images.append(image_array)

images = np.array(images)
images.shape
```

Out[114…

```
(5, 32, 32)
```

In [115…

```python
# Train the model (lr=0.0005), Adam optimizer, sigmoid function, dropout rate=0.1, CCE

model, history, duration = train_model(
        hidden_neurons=hidden_neurons,
        learning_rate=0.0005,
        dropout_rate=best_dropout_rate,
        activation_func=best_activation_function,
        loss_func=CategoricalCrossentropy,
        optimizer=Adam,
        epochs=100
    )

test_loss, test_acc = model.evaluate(X_test,y_test)

print(f"Learning rate = {0.0005}")
print(f"Test loss={test_loss} Test accuracy = {test_acc}")
print(f"Time taken to train the model is {duration} seconds")

plot_metrics(history,best_activation_function,hidden_neurons,dropout_rate=best_dropout_rate)
```

Model: "sequential_27"

Layer (type)	Output Shape	Param #
flatten_27 (Flatten)	(None, 1024)	0
dense_94 (Dense)	(None, 16)	16,400
dropout_30 (Dropout)	(None, 16)	0
dense_95 (Dense)	(None, 32)	544
dropout_31 (Dropout)	(None, 32)	0
dense_96 (Dense)	(None, 64)	2,112
dropout_32 (Dropout)	(None, 64)	0
dense_97 (Dense)	(None, 10)	650

Total params: 19,706 (76.98 KB)

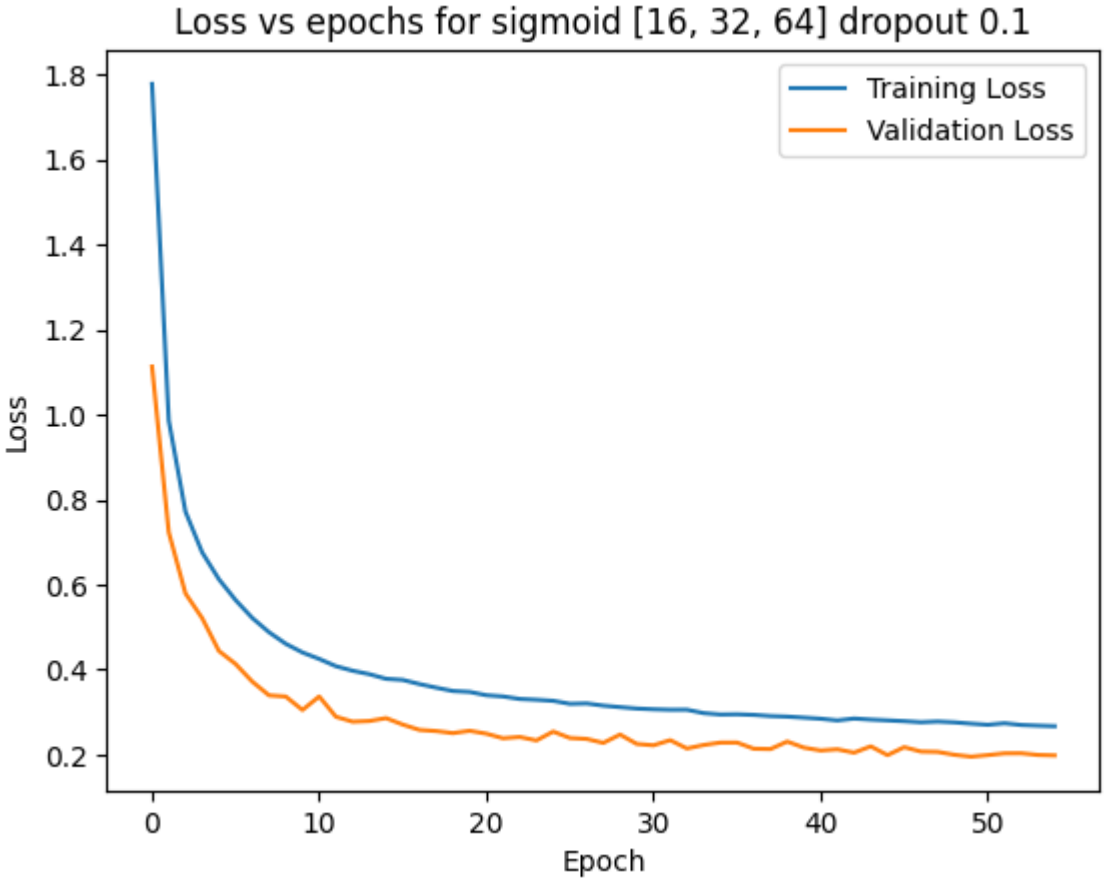
Trainable params: 19,706 (76.98 KB)

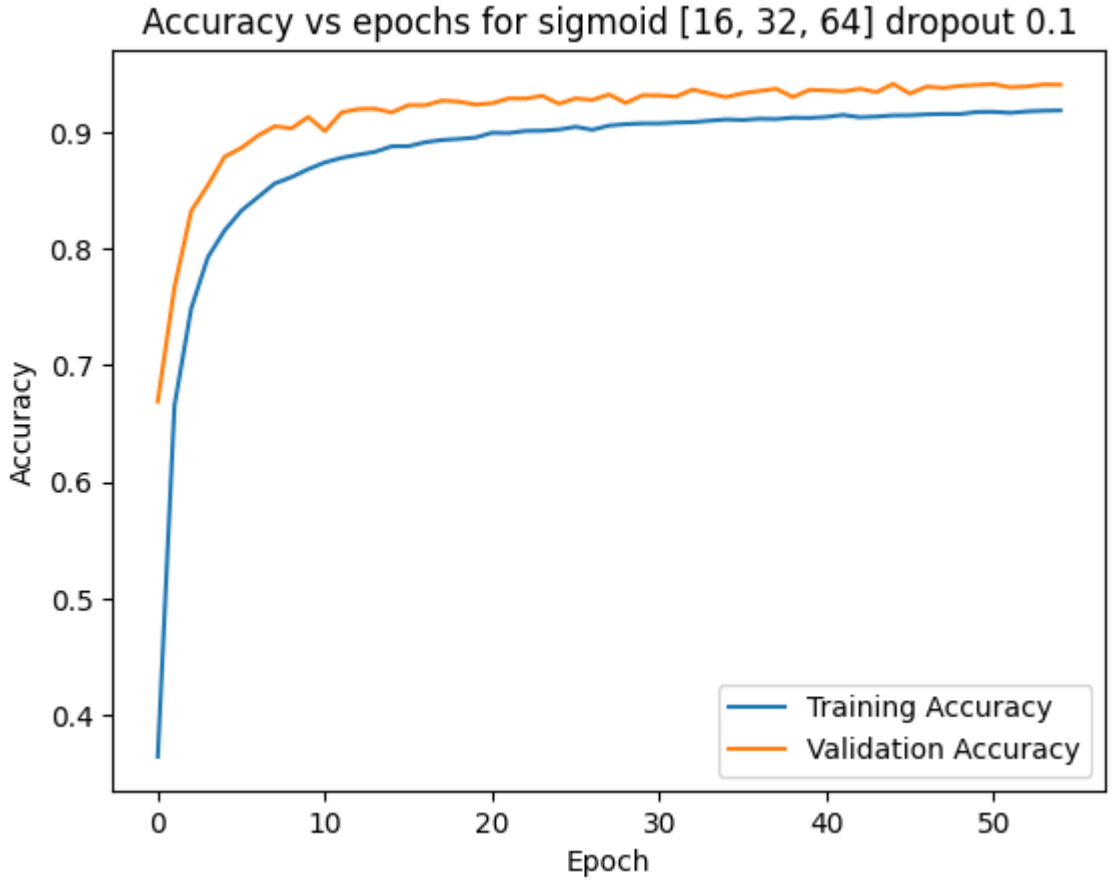
Non-trainable params: 0 (0.00 B)

```
Epoch 1/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.2258 - loss: 2.1062 - val_accuracy: 0.6690 - val_loss: 1.1130
Epoch 2/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.6298 - loss: 1.0791 - val_accuracy: 0.7670 - val_loss: 0.7234
Epoch 3/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.7355 - loss: 0.8068 - val_accuracy: 0.8324 - val_loss: 0.5794
Epoch 4/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.7855 - loss: 0.6915 - val_accuracy: 0.8546 - val_loss: 0.5212
Epoch 5/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8123 - loss: 0.6292 - val_accuracy: 0.8790 - val_loss: 0.4441
Epoch 6/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.8284 - loss: 0.5773 - val_accuracy: 0.8866 - val_loss: 0.4133
Epoch 7/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8434 - loss: 0.5277 - val_accuracy: 0.8971 - val_loss: 0.3718
Epoch 8/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8563 - loss: 0.4882 - val_accuracy: 0.9051 - val_loss: 0.3396
Epoch 9/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8594 - loss: 0.4711 - val_accuracy: 0.9033 - val_loss: 0.3369
Epoch 10/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8665 - loss: 0.4468 - val_accuracy: 0.9131 - val_loss: 0.3052
Epoch 11/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 11s 4ms/step - accuracy: 0.8770 - loss: 0.4149 - val_accuracy: 0.9010 - val_loss: 0.3370
Epoch 12/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 8s 2ms/step - accuracy: 0.8773 - loss: 0.4079 - val_accuracy: 0.9170 - val_loss: 0.2894
Epoch 13/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8788 - loss: 0.4024 - val_accuracy: 0.9199 - val_loss: 0.2777
Epoch 14/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8842 - loss: 0.3862 - val_accuracy: 0.9201 - val_loss: 0.2793
Epoch 15/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8887 - loss: 0.3777 - val_accuracy: 0.9170 - val_loss: 0.2861
Epoch 16/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.8909 - loss: 0.3669 - val_accuracy: 0.9233 - val_loss: 0.2709
Epoch 17/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8931 - loss: 0.3629 - val_accuracy: 0.9231 - val_loss: 0.2583
Epoch 18/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8911 - loss: 0.3584 - val_accuracy: 0.9271 - val_loss: 0.2557
Epoch 19/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8937 - loss: 0.3470 - val_accuracy: 0.9263 - val_loss: 0.2508
Epoch 20/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8928 - loss: 0.3534 - val_accuracy: 0.9237 - val_loss: 0.2563
Epoch 21/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8999 - loss: 0.3419 - val_accuracy: 0.9249 - val_loss: 0.2495
Epoch 22/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8977 - loss: 0.3417 - val_accuracy: 0.9290 - val_loss: 0.2387
Epoch 23/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9043 - loss: 0.3229 - val_accuracy: 0.9289 - val_loss: 0.2417
```

```
Epoch 24/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9017 - loss: 0.3301 - val_accuracy: 0.9313 - val_loss: 0.2336
Epoch 25/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9027 - loss: 0.3240 - val_accuracy: 0.9244 - val_loss: 0.2544
Epoch 26/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9043 - loss: 0.3228 - val_accuracy: 0.9290 - val_loss: 0.2393
Epoch 27/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9031 - loss: 0.3186 - val_accuracy: 0.9277 - val_loss: 0.2372
Epoch 28/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.9065 - loss: 0.3186 - val_accuracy: 0.9324 - val_loss: 0.2278
Epoch 29/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9053 - loss: 0.3203 - val_accuracy: 0.9251 - val_loss: 0.2474
Epoch 30/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9085 - loss: 0.3021 - val_accuracy: 0.9317 - val_loss: 0.2253
Epoch 31/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9073 - loss: 0.3101 - val_accuracy: 0.9316 - val_loss: 0.2222
Epoch 32/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9076 - loss: 0.3085 - val_accuracy: 0.9306 - val_loss: 0.2342
Epoch 33/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9112 - loss: 0.2981 - val_accuracy: 0.9366 - val_loss: 0.2147
Epoch 34/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9112 - loss: 0.2945 - val_accuracy: 0.9333 - val_loss: 0.2228
Epoch 35/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.9114 - loss: 0.2936 - val_accuracy: 0.9301 - val_loss: 0.2285
Epoch 36/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.9083 - loss: 0.2961 - val_accuracy: 0.9334 - val_loss: 0.2286
Epoch 37/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9087 - loss: 0.3005 - val_accuracy: 0.9354 - val_loss: 0.2141
Epoch 38/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9117 - loss: 0.2900 - val_accuracy: 0.9373 - val_loss: 0.2133
Epoch 39/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9154 - loss: 0.2814 - val_accuracy: 0.9301 - val_loss: 0.2309
Epoch 40/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9121 - loss: 0.2878 - val_accuracy: 0.9364 - val_loss: 0.2167
Epoch 41/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9138 - loss: 0.2833 - val_accuracy: 0.9359 - val_loss: 0.2100
Epoch 42/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.9145 - loss: 0.2809 - val_accuracy: 0.9350 - val_loss: 0.2127
Epoch 43/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9152 - loss: 0.2802 - val_accuracy: 0.9373 - val_loss: 0.2048
Epoch 44/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9129 - loss: 0.2869 - val_accuracy: 0.9344 - val_loss: 0.2197
Epoch 45/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9153 - loss: 0.2797 - val_accuracy: 0.9414 - val_loss: 0.1990
Epoch 46/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9153 - loss: 0.2799 - val_accuracy: 0.9331 - val_loss: 0.2178
```

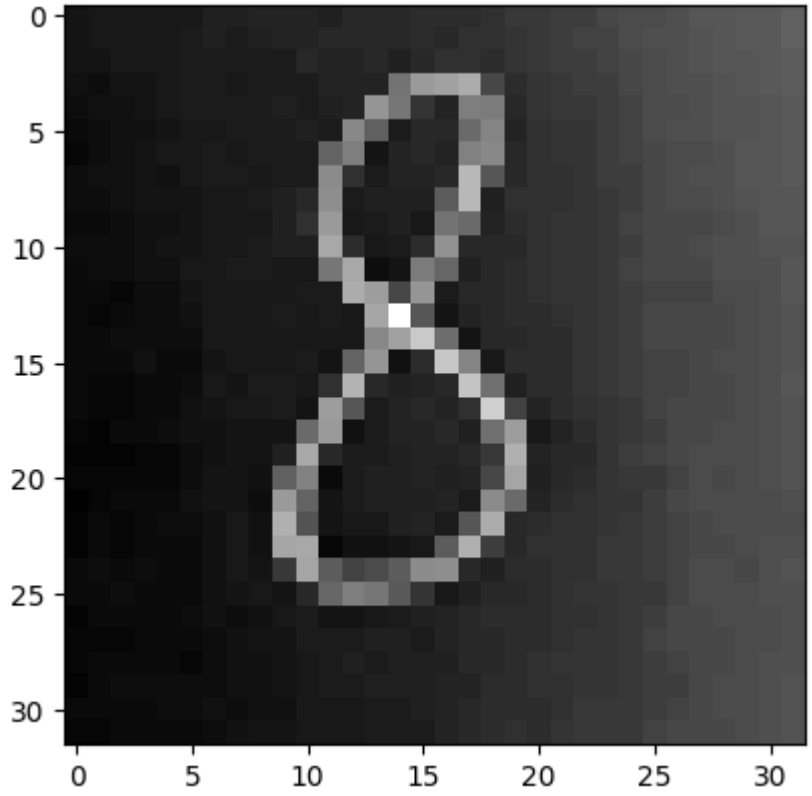
```
Epoch 47/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9178 - loss: 0.2710 - val_accuracy: 0.9391 - val_loss: 0.2075
Epoch 48/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 7s 3ms/step - accuracy: 0.9169 - loss: 0.2739 - val_accuracy: 0.9380 - val_loss: 0.2067
Epoch 49/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - accuracy: 0.9155 - loss: 0.2751 - val_accuracy: 0.9397 - val_loss: 0.1998
Epoch 50/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9156 - loss: 0.2794 - val_accuracy: 0.9406 - val_loss: 0.1954
Epoch 51/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9171 - loss: 0.2717 - val_accuracy: 0.9413 - val_loss: 0.1991
Epoch 52/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 8s 4ms/step - accuracy: 0.9161 - loss: 0.2743 - val_accuracy: 0.9386 - val_loss: 0.2032
Epoch 53/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.9206 - loss: 0.2597 - val_accuracy: 0.9393 - val_loss: 0.2034
Epoch 54/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.9171 - loss: 0.2730 - val_accuracy: 0.9410 - val_loss: 0.1994
Epoch 55/100
1750/1750 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.9183 - loss: 0.2669 - val_accuracy: 0.9407 - val_loss: 0.1984
219/219 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.9454 - loss: 0.1963
Learning rate = 0.0005
Test loss=0.20049400627613068 Test accuracy = 0.9425714015960693
Time taken to train the model is 313.5573968887329 seconds
```


Loss vs epochs for sigmoid [16, 32, 64] dropout 0.1
Training Loss
Validation Loss
Loss
Epoch
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0
10
20
30
40
50

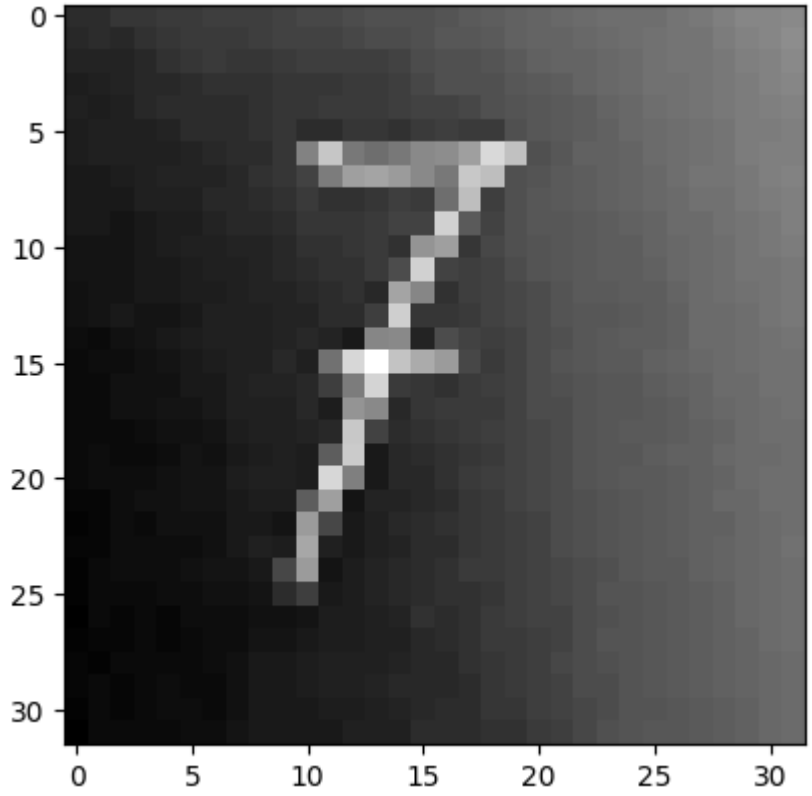


In [117…

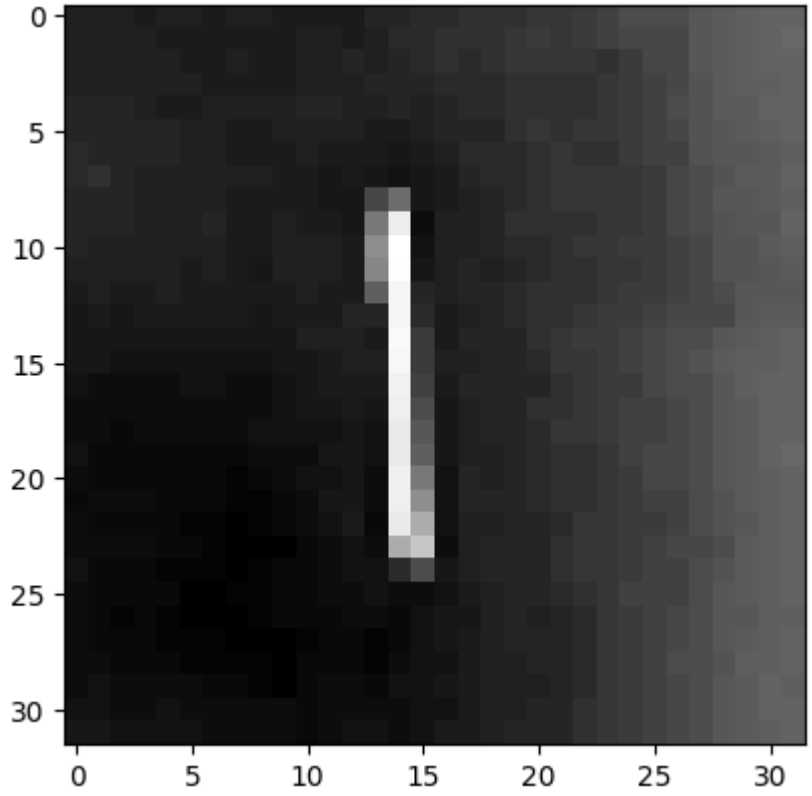
```
# Prediction

for image in images:
  plt.imshow(image,cmap='Greys')
  plt.show()
  prediction_array = model.predict(image.reshape(1,32,32))
  print(f"Prediction: {np.argmax(prediction_array)}")
```



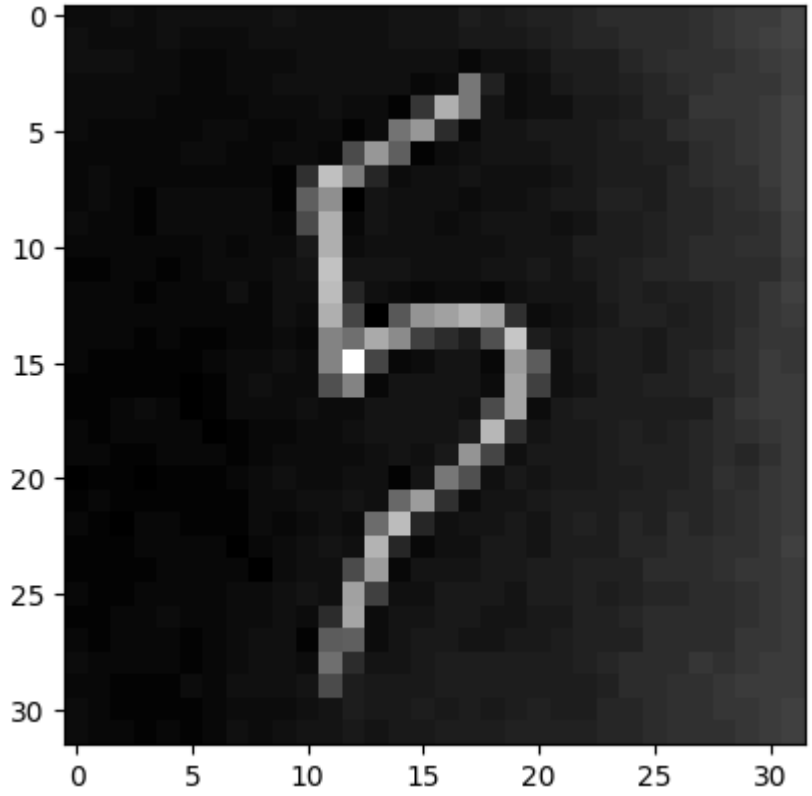

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 246ms/step
Prediction: 0
```



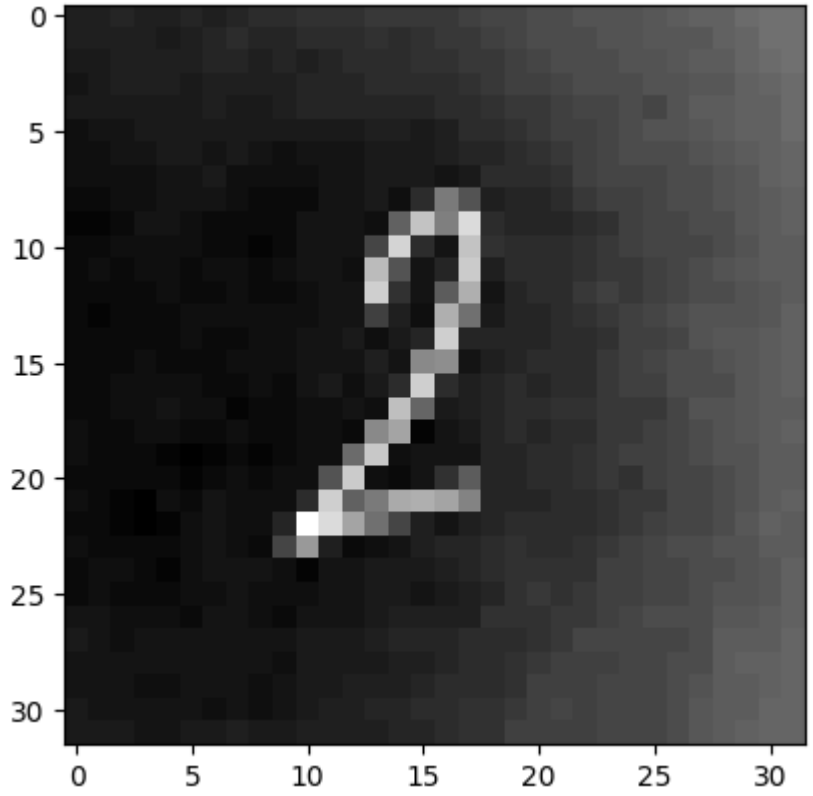

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
Prediction: 0
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
Prediction: 5
```



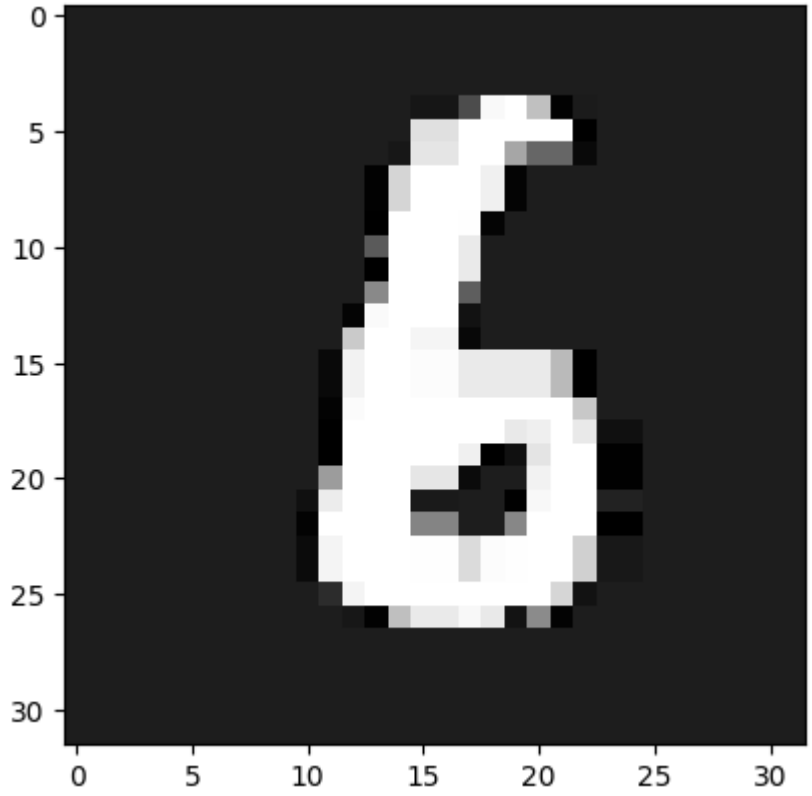

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
Prediction: 5
```



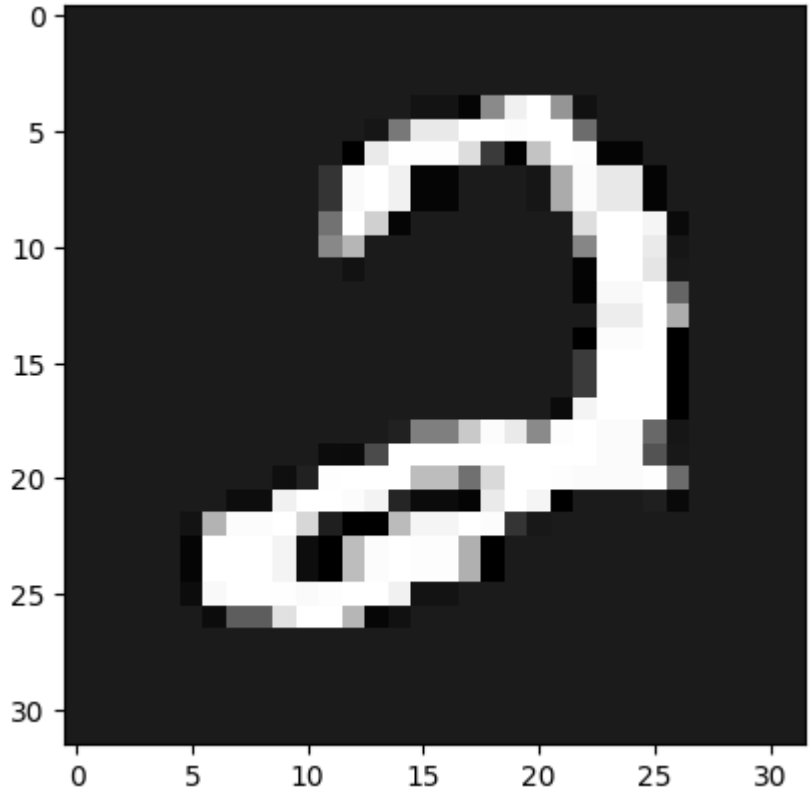

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
Prediction: 5
```

In [118…

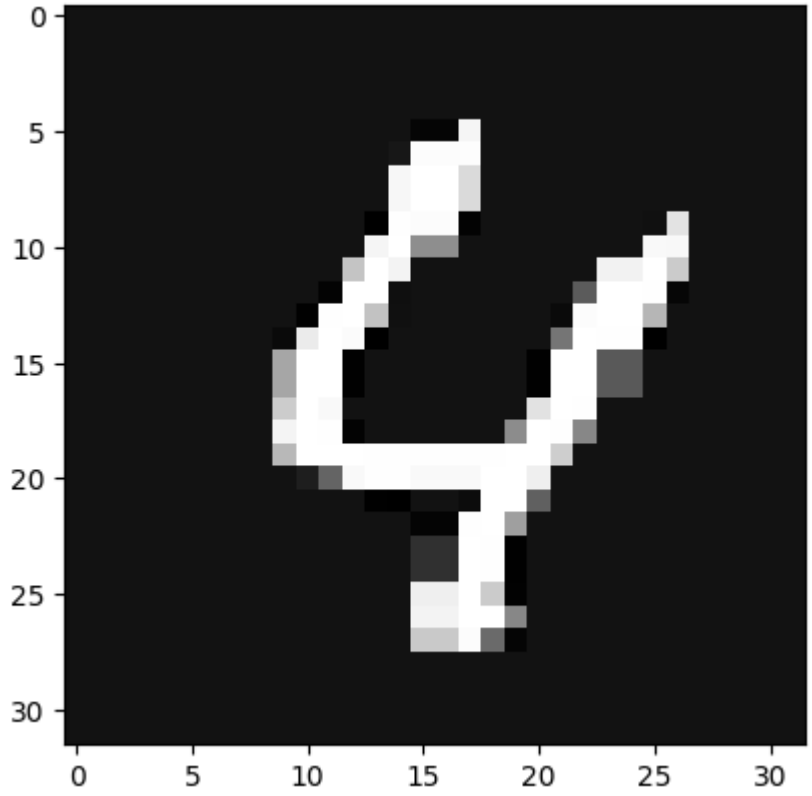
```python
# Testing on taking 10 random samples from MNIST dataset(testing portion)

random_images = X_test[np.random.choice(X_test.shape[0], size=10, replace=False)]

for image in random_images:
  plt.imshow(image,cmap='Greys')
  plt.show()
  prediction_array = model.predict(image.reshape(1,32,32))
  print(f"Prediction: {np.argmax(prediction_array)}")
```



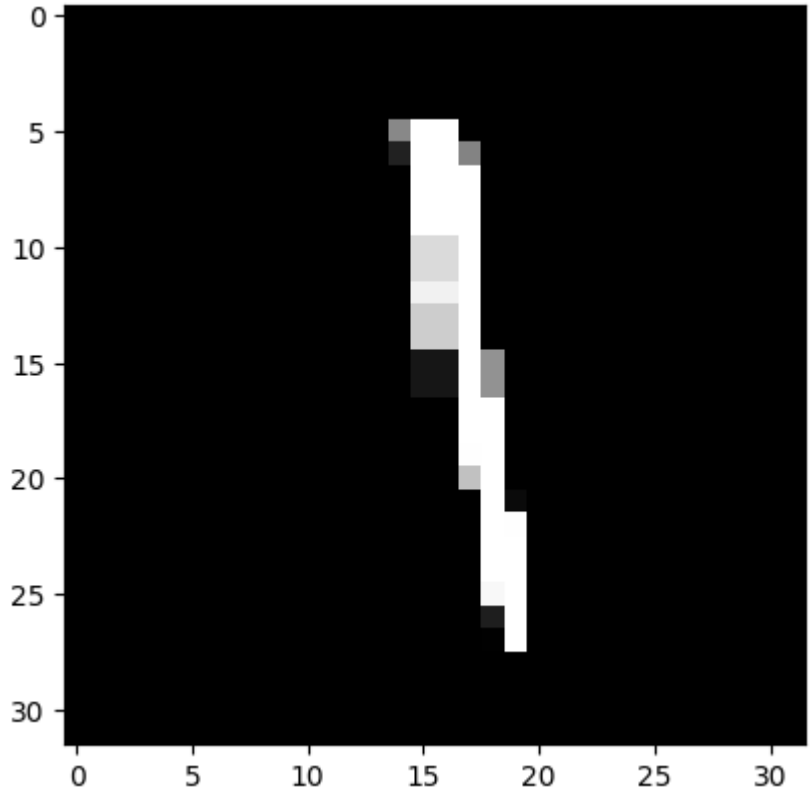

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 116ms/step
Prediction: 6
```



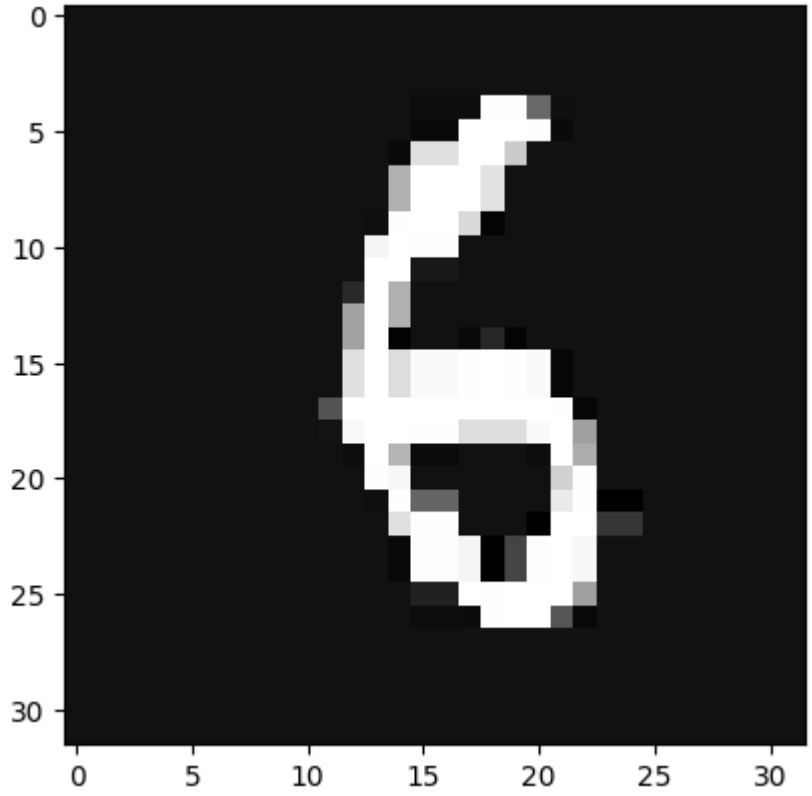

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
Prediction: 2
```



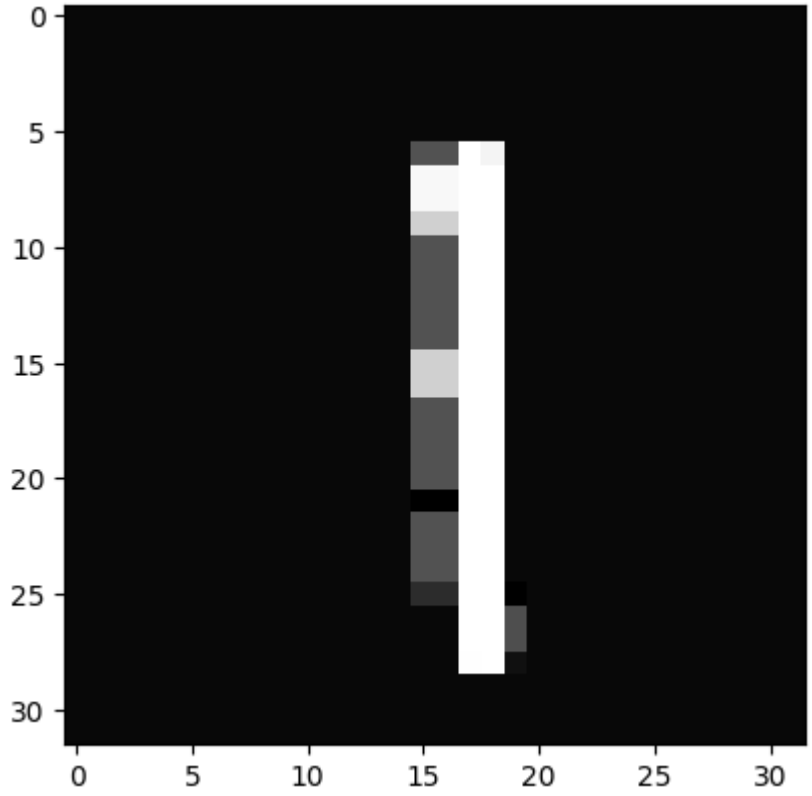

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
Prediction: 4
```



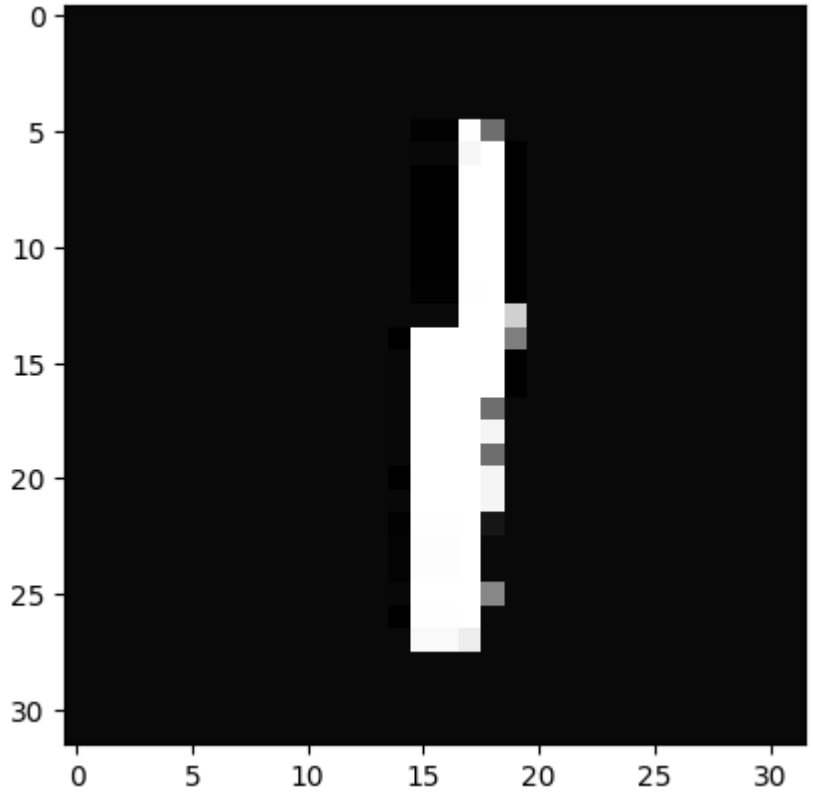

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 162ms/step
Prediction: 1
```



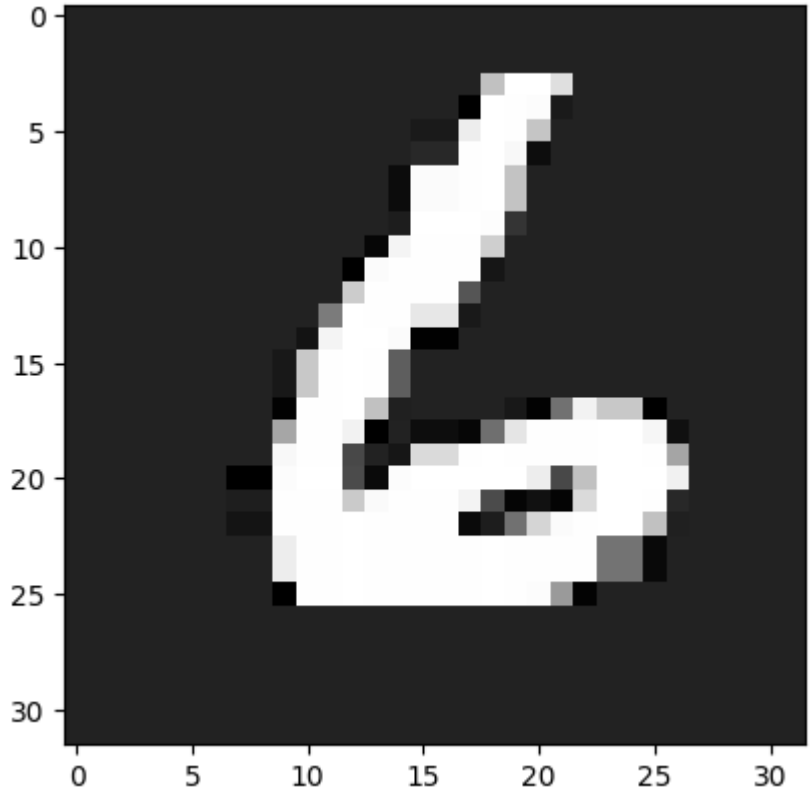

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
Prediction: 6
```



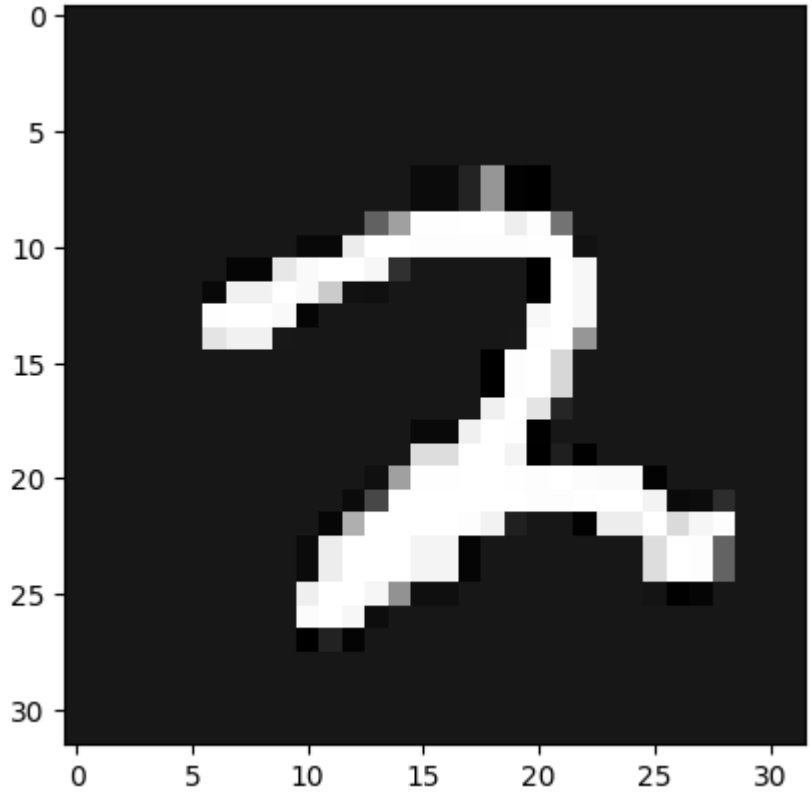

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
Prediction: 1
```



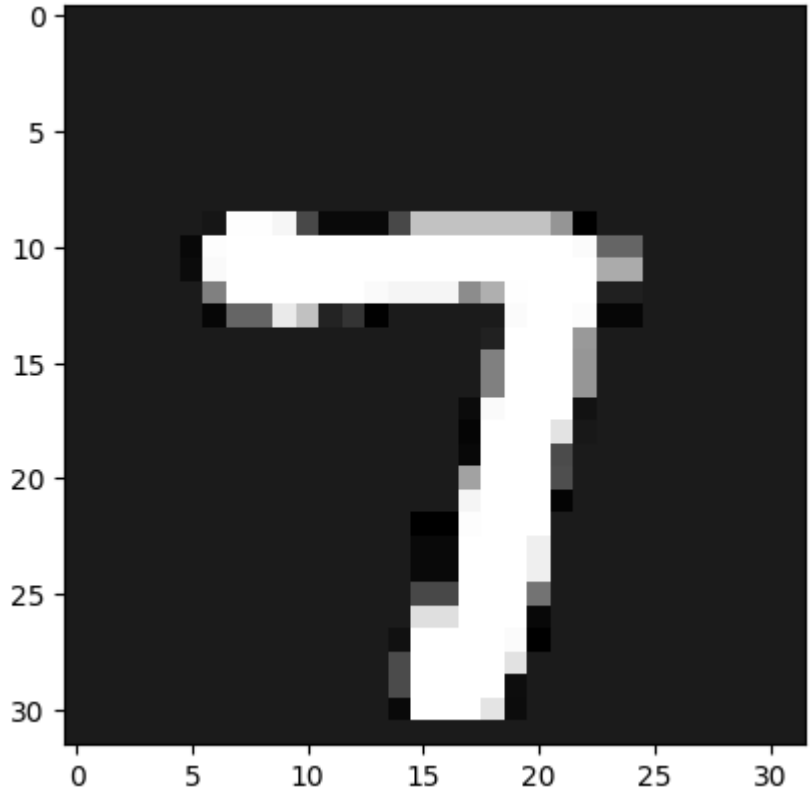

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
Prediction: 1
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
Prediction: 6
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
Prediction: 2
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
Prediction: 7
```

As we can see that taking random samples from a standardised MNIST dataset, there is 100% accurate prediction but for non-standard images the accuracy is 20%