```python
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
from tensorflow.keras import datasets, layers, models

```

```python

mnist = tf.keras.datasets.cifar10

```

```python
print(x_train.shape)

```

```
(50000, 32, 32, 3)
```

```python
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()
train_images, test_images = train_images / 255.0, test_images / 255.0
train_labels = tf.keras.utils.to_categorical(train_labels, 10)
test_labels = tf.keras.utils.to_categorical(test_labels, 10)

```

```python
test_labels[0]
```

```
array([0., 0., 0., 1., 0., 0., 0., 0., 0., 0.], dtype=float32)
```

```python
from tensorflow.keras import models,layers
def build_lenet():
    model = models.Sequential()
    model.add(layers.Conv2D(6, (5, 5), activation='relu', input_shape=(32, 32, 3)))
    model.add(layers.AveragePooling2D((2, 2)))
    model.add(layers.Conv2D(16, (5, 5), activation='relu'))
    model.add(layers.AveragePooling2D((2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(120, activation='relu'))
    model.add(layers.Dense(84, activation='relu'))
    model.add(layers.Dense(10, activation='softmax'))

    return model
```

```python

# Compile the model
model = build_lenet()
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_4 (Conv2D)           (None, 28, 28, 6)         456       
                                                                 
 average_pooling2d_4 (Avera  (None, 14, 14, 6)         0         
 gePooling2D)                                                    
                                                                 
 conv2d_5 (Conv2D)           (None, 10, 10, 16)        2416      
                                                                 
 average_pooling2d_5 (Avera  (None, 5, 5, 16)          0         
 gePooling2D)                                                    
                                                                 
 flatten_2 (Flatten)         (None, 400)               0         
                                                                 
 dense_6 (Dense)             (None, 120)               48120     
                                                                 
 dense_7 (Dense)             (None, 84)                10164     
                                                                 
 dense_8 (Dense)             (None, 10)                850       
                                                                 
=================================================================
Total params: 62006 (242.21 KB)
Trainable params: 62006 (242.21 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
history = model.fit(train_images, train_labels, epochs=20,
                    validation_data=(test_images, test_labels))
```

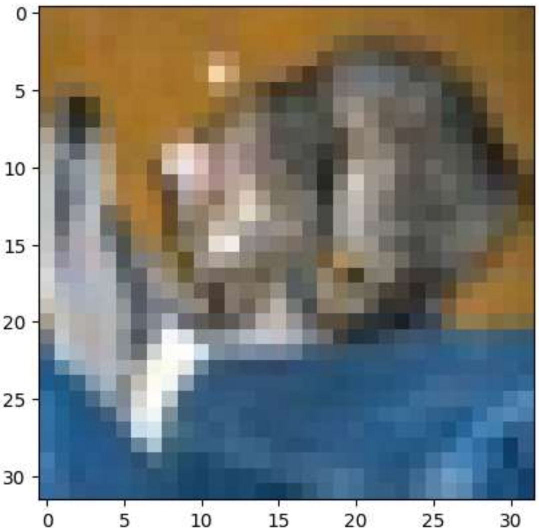
```
Epoch 1/20
1563/1563 [==============================] - 7s 5ms/step - loss: 1.3661 - accuracy: 0.5085 - val_loss: 1.3219 - val_accuracy: 0.5295
Epoch 2/20
1563/1563 [==============================] - 8s 5ms/step - loss: 1.2638 - accuracy: 0.5446 - val_loss: 1.2660 - val_accuracy: 0.5501
Epoch 3/20
1563/1563 [==============================] - 7s 5ms/step - loss: 1.1945 - accuracy: 0.5719 - val_loss: 1.2253 - val_accuracy: 0.5653
Epoch 4/20
1563/1563 [==============================] - 8s 5ms/step - loss: 1.1383 - accuracy: 0.5950 - val_loss: 1.1864 - val_accuracy: 0.5784
Epoch 5/20
1563/1563 [==============================] - 9s 6ms/step - loss: 1.0904 - accuracy: 0.6122 - val_loss: 1.1525 - val_accuracy: 0.5918
Epoch 6/20
1563/1563 [==============================] - 7s 5ms/step - loss: 1.0490 - accuracy: 0.6263 - val_loss: 1.2124 - val_accuracy: 0.5718
Epoch 7/20
1563/1563 [==============================] - 9s 5ms/step - loss: 1.0152 - accuracy: 0.6398 - val_loss: 1.1433 - val_accuracy: 0.6020
Epoch 8/20
1563/1563 [==============================] - 8s 5ms/step - loss: 0.9810 - accuracy: 0.6521 - val_loss: 1.1361 - val_accuracy: 0.6062
Epoch 9/20
1563/1563 [==============================] - 9s 5ms/step - loss: 0.9425 - accuracy: 0.6659 - val_loss: 1.1388 - val_accuracy: 0.6031
Epoch 10/20
1563/1563 [==============================] - 9s 6ms/step - loss: 0.9142 - accuracy: 0.6731 - val_loss: 1.1581 - val_accuracy: 0.6039
Epoch 11/20
1563/1563 [==============================] - 8s 5ms/step - loss: 0.8835 - accuracy: 0.6879 - val_loss: 1.1564 - val_accuracy: 0.6048
Epoch 12/20
1563/1563 [==============================] - 8s 5ms/step - loss: 0.8603 - accuracy: 0.6945 - val_loss: 1.1776 - val_accuracy: 0.5976
Epoch 13/20
1563/1563 [==============================] - 9s 6ms/step - loss: 0.8267 - accuracy: 0.7044 - val_loss: 1.1961 - val_accuracy: 0.5939
Epoch 14/20
1563/1563 [==============================] - 8s 5ms/step - loss: 0.8078 - accuracy: 0.7143 - val_loss: 1.1769 - val_accuracy: 0.6078
Epoch 15/20
1563/1563 [==============================] - 9s 6ms/step - loss: 0.7854 - accuracy: 0.7199 - val_loss: 1.2016 - val_accuracy: 0.6017
Epoch 16/20
1563/1563 [==============================] - 8s 5ms/step - loss: 0.7603 - accuracy: 0.7284 - val_loss: 1.2341 - val_accuracy: 0.6005
Epoch 17/20
1563/1563 [==============================] - 8s 5ms/step - loss: 0.7405 - accuracy: 0.7374 - val_loss: 1.2414 - val_accuracy: 0.5986
Epoch 18/20
1563/1563 [==============================] - 9s 6ms/step - loss: 0.7170 - accuracy: 0.7432 - val_loss: 1.2346 - val_accuracy: 0.5995
Epoch 19/20
1563/1563 [==============================] - 9s 6ms/step - loss: 0.6998 - accuracy: 0.7507 - val_loss: 1.2699 - val_accuracy: 0.5986
Epoch 20/20
1563/1563 [==============================] - 8s 5ms/step - loss: 0.6771 - accuracy: 0.7594 - val_loss: 1.3213 - val_accuracy: 0.5957
```

```
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print(f"Test accuracy: {test_acc}")
```

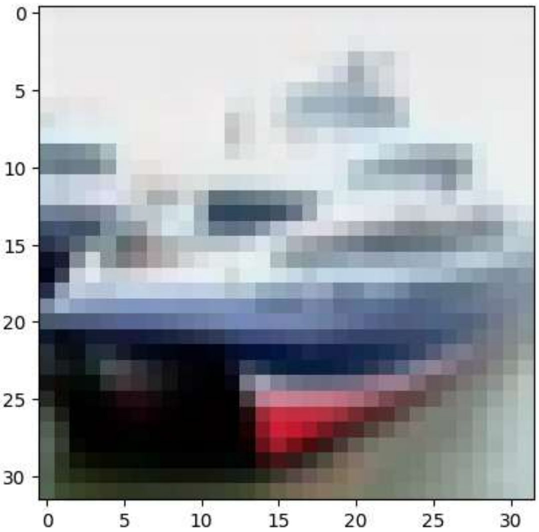
```
313/313 - 1s - loss: 1.3213 - accuracy: 0.5957 - 1s/epoch - 4ms/step
Test accuracy: 0.5957000255584717
```

```
predictions = model.predict(test_images[:5])

# Display the first 5 test images and their predicted labels
for i in range(5):
    plt.imshow(test_images[i])
    plt.show()
    print(f"Predicted label: {np.argmax(predictions[i])}")
```

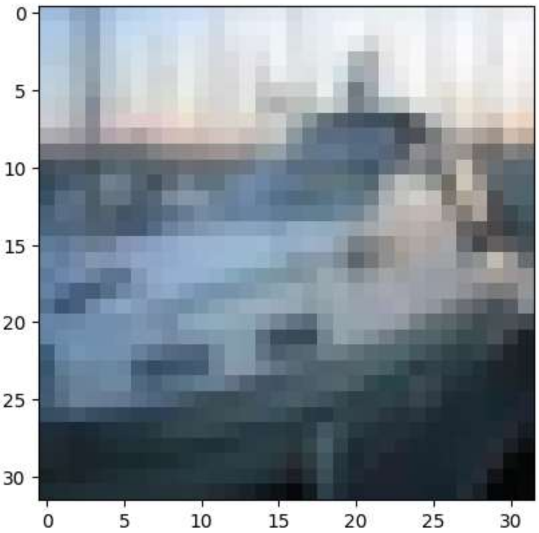
```
1/1 [==============================] - 0s 158ms/step
```



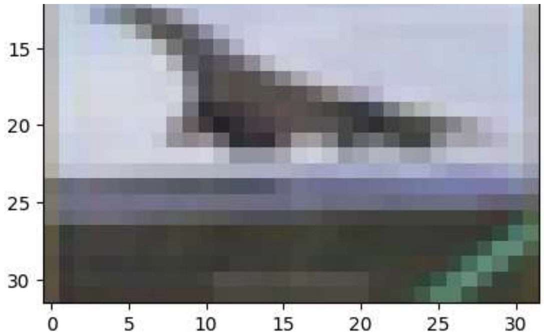
```
Predicted label: 3
```



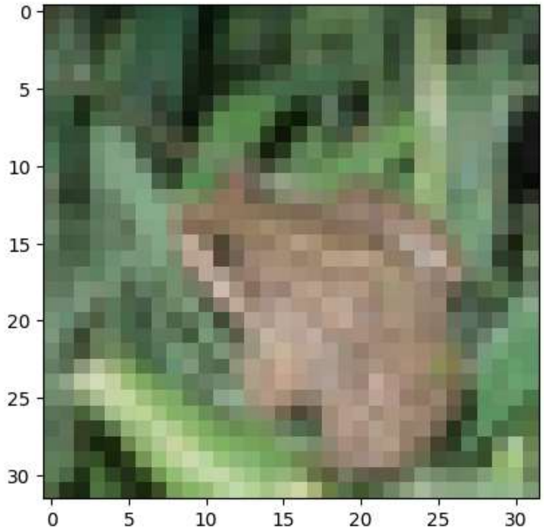
```
Predicted label: 8
```



```
Predicted label: 8
```



```
Predicted label: 0
```

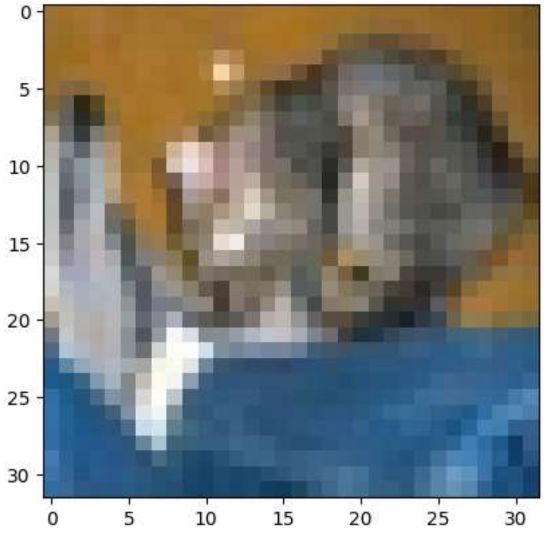


```
Predicted label: 6
```

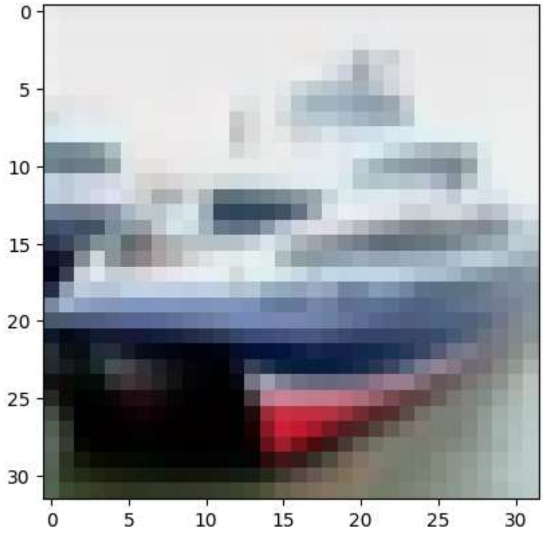
```python
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import numpy as np
import matplotlib.pyplot as plt

# Load and preprocess dataset
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()
train_images, test_images = train_images / 255.0, test_images / 255.0
train_labels = tf.keras.utils.to_categorical(train_labels, 10)
test_labels = tf.keras.utils.to_categorical(test_labels, 10)

# Define the LeNet model
def create_lenet_model():
    model = models.Sequential()
    model.add(layers.Conv2D(6, (5, 5), activation='relu', input_shape=(32, 32, 3)))
    model.add(layers.AveragePooling2D((2, 2)))
    model.add(layers.Conv2D(16, (5, 5), activation='relu'))
    model.add(layers.AveragePooling2D((2, 2)))
    model.add(layers.Flatten())
    model.add(layers.Dense(120, activation='relu'))
    model.add(layers.Dense(84, activation='relu'))
    model.add(layers.Dense(10, activation='softmax'))
    return model

# Compile the model
model = create_lenet_model()
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))

# Evaluate the model
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print(f"Test accuracy: {test_acc}")

# Predict the first 5 images in the test set
predictions = model.predict(test_images[:5])

# Display the first 5 test images and their predicted labels
for i in range(5):
    plt.imshow(test_images[i])
    plt.show()
    print(f"Predicted label: {np.argmax(predictions[i])}")

```

```python

# Train the model
history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))

# Evaluate the model
test_loss, test_acc = model.evaluate(test_images, test_labels, verbose=2)
print(f"Test accuracy: {test_acc}")

# Predict the first 5 images in the test set
predictions = model.predict(test_images[:5])

# Display the first 5 test images and their predicted labels
for i in range(5):
    plt.imshow(test_images[i])
    plt.show()
    print(f"Predicted label: {np.argmax(predictions[i])}")

```

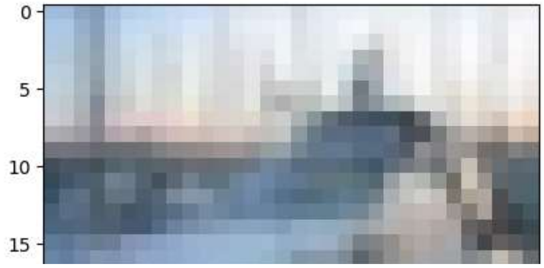
```
Epoch 1/10
1563/1563 [==============================] - 41s 25ms/step - loss: 1.6807 - accuracy: 0.
Epoch 2/10
1563/1563 [==============================] - 36s 23ms/step - loss: 1.4188 - accuracy: 0.
Epoch 3/10
1563/1563 [==============================] - 38s 25ms/step - loss: 1.3149 - accuracy: 0.
Epoch 4/10
1563/1563 [==============================] - 40s 26ms/step - loss: 1.2460 - accuracy: 0.
Epoch 5/10
1563/1563 [==============================] - 43s 28ms/step - loss: 1.1851 - accuracy: 0.
Epoch 6/10
1563/1563 [==============================] - 40s 26ms/step - loss: 1.1360 - accuracy: 0.
Epoch 7/10
1563/1563 [==============================] - 45s 29ms/step - loss: 1.0897 - accuracy: 0.
Epoch 8/10
1563/1563 [==============================] - 39s 25ms/step - loss: 1.0458 - accuracy: 0.
Epoch 9/10
1563/1563 [==============================] - 38s 24ms/step - loss: 1.0073 - accuracy: 0.
Epoch 10/10
1563/1563 [==============================] - 36s 23ms/step - loss: 0.9732 - accuracy: 0.
313/313 - 3s - loss: 1.1774 - accuracy: 0.5928 - 3s/epoch - 10ms/step
Test accuracy: 0.5928000211715698
1/1 [==============================] - 0s 113ms/step
```

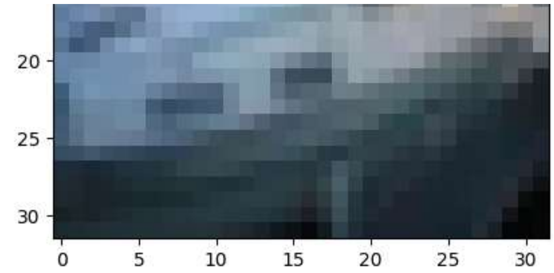


```
Predicted label: 3
```

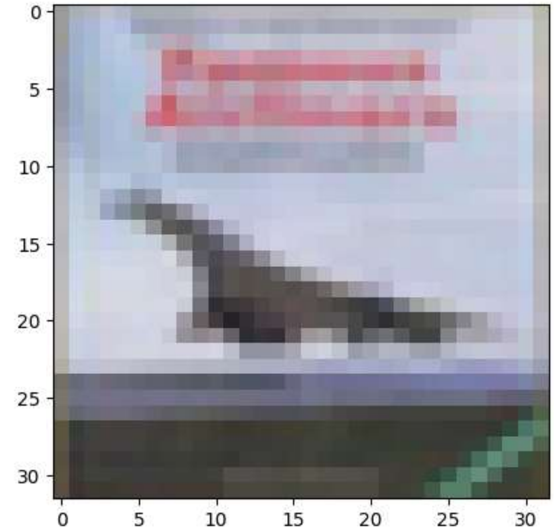


```
Predicted label: 8
```

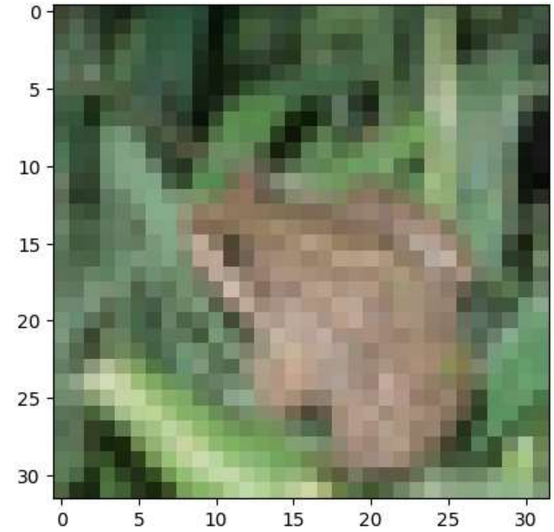




```
Predicted label: 8
```



```
Predicted label: 0
```



```
Predicted label: 3
```

```
def create_vgg16_model():
    vgg16_base = tf.keras.applications.VGG16(weights='imagenet', include_top=False, input_shape=(32, 32, 3))
    model = models.Sequential()
    model.add(vgg16_base)
    model.add(layers.Flatten())
    model.add(layers.Dense(512, activation='relu'))
    model.add(layers.Dense(10, activation='softmax'))

    # Freeze the convolutional base
    vgg16_base.trainable = False

    return model

vgg16_model = create_vgg16_model()
vgg16_model.compile(optimizer='adam',
                    loss='categorical_crossentropy',
                    metrics=['accuracy'])



```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.
58889256/58889256 [==============================] - 4s 0us/step
```

```
vgg16_model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 vgg16 (Functional)          (None, 1, 1, 512)         14714688  
                                                                 
 flatten_3 (Flatten)         (None, 512)               0         
                                                                 
 dense_9 (Dense)             (None, 512)               262656    
                                                                 
 dense_10 (Dense)            (None, 10)                5130      
                                                                 
=================================================================
Total params: 14982474 (57.15 MB)
Trainable params: 267786 (1.02 MB)
Non-trainable params: 14714688 (56.13 MB)
_________________________________________________________________
```

```
vgg16_history = vgg16_model.fit(train_images, train_labels, epochs=20,
                                validation_data=(test_images, test_labels))
```

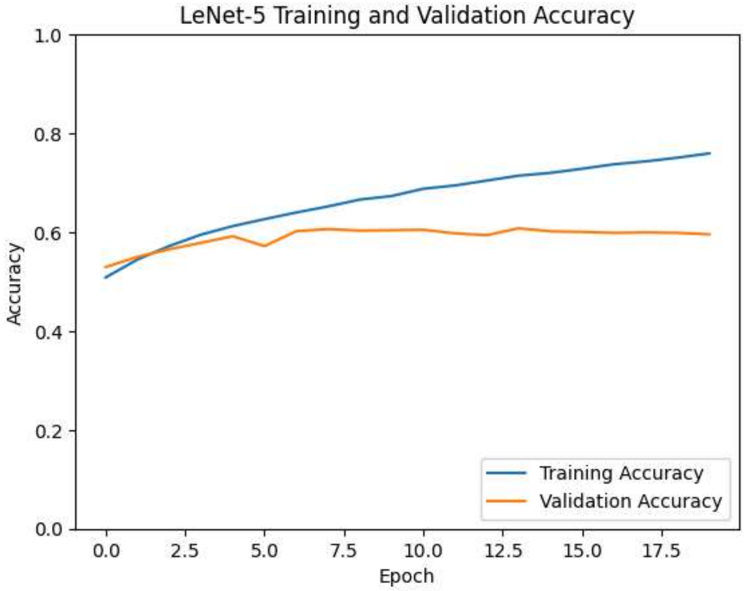
```
Epoch 1/20
1563/1563 [==============================] - 17s 11ms/step - loss: 0.8470 - accuracy: 0.7009 - val_loss: 1.1105 - val_accuracy: 0.6215
Epoch 2/20
1563/1563 [==============================] - 17s 11ms/step - loss: 0.8074 - accuracy: 0.7148 - val_loss: 1.1205 - val_accuracy: 0.6171
Epoch 3/20
1563/1563 [==============================] - 17s 11ms/step - loss: 0.7709 - accuracy: 0.7295 - val_loss: 1.1459 - val_accuracy: 0.6173
Epoch 4/20
1563/1563 [==============================] - 17s 11ms/step - loss: 0.7349 - accuracy: 0.7429 - val_loss: 1.1610 - val_accuracy: 0.6181
Epoch 5/20
1563/1563 [==============================] - 17s 11ms/step - loss: 0.7006 - accuracy: 0.7552 - val_loss: 1.1781 - val_accuracy: 0.6237
Epoch 6/20
1563/1563 [==============================] - 17s 11ms/step - loss: 0.6655 - accuracy: 0.7659 - val_loss: 1.1836 - val_accuracy: 0.6165
Epoch 7/20
1563/1563 [==============================] - 17s 11ms/step - loss: 0.6315 - accuracy: 0.7775 - val_loss: 1.2129 - val_accuracy: 0.6191
Epoch 8/20
1563/1563 [==============================] - 17s 11ms/step - loss: 0.6006 - accuracy: 0.7900 - val_loss: 1.2379 - val_accuracy: 0.6198
Epoch 9/20
1563/1563 [==============================] - 17s 11ms/step - loss: 0.5701 - accuracy: 0.7995 - val_loss: 1.2941 - val_accuracy: 0.6079
Epoch 10/20
1563/1563 [==============================] - 16s 11ms/step - loss: 0.5411 - accuracy: 0.8094 - val_loss: 1.2959 - val_accuracy: 0.6151
Epoch 11/20
1563/1563 [==============================] - 17s 11ms/step - loss: 0.5131 - accuracy: 0.8201 - val_loss: 1.3391 - val_accuracy: 0.6194
Epoch 12/20
1563/1563 [==============================] - 17s 11ms/step - loss: 0.4849 - accuracy: 0.8309 - val_loss: 1.3618 - val_accuracy: 0.6101
Epoch 13/20
1563/1563 [==============================] - 17s 11ms/step - loss: 0.4601 - accuracy: 0.8395 - val_loss: 1.3743 - val_accuracy: 0.6154
Epoch 14/20
1563/1563 [==============================] - 17s 11ms/step - loss: 0.4355 - accuracy: 0.8494 - val_loss: 1.4235 - val_accuracy: 0.6069
Epoch 15/20
1563/1563 [==============================] - 17s 11ms/step - loss: 0.4101 - accuracy: 0.8571 - val_loss: 1.4775 - val_accuracy: 0.6078
Epoch 16/20
1563/1563 [==============================] - 17s 11ms/step - loss: 0.3896 - accuracy: 0.8664 - val_loss: 1.5127 - val_accuracy: 0.6114
```

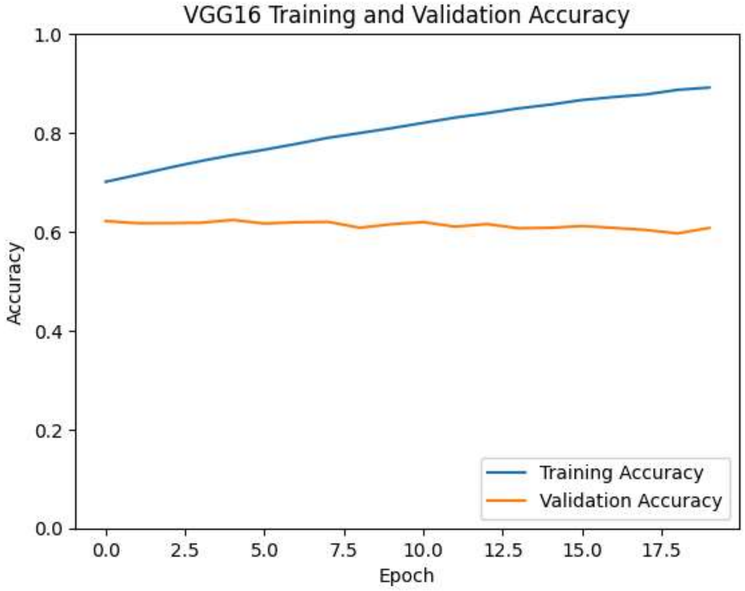
```
Epoch 17/20
1563/1563 [==============================] - 17s 11ms/step - loss: 0.3687 - accuracy: 0.8724 - val_loss: 1.5741 - val_accuracy: 0.6076
Epoch 18/20
1563/1563 [==============================] - 16s 11ms/step - loss: 0.3518 - accuracy: 0.8776 - val_loss: 1.6029 - val_accuracy: 0.6035
Epoch 19/20
1563/1563 [==============================] - 17s 11ms/step - loss: 0.3338 - accuracy: 0.8869 - val_loss: 1.6592 - val_accuracy: 0.5965
Epoch 20/20
1563/1563 [==============================] - 17s 11ms/step - loss: 0.3155 - accuracy: 0.8915 - val_loss: 1.6957 - val_accuracy: 0.6075
```

```python
test_loss, vgg16_moddel = model.evaluate(test_images, test_labels, verbose=2)
print(f"Test accuracy: {test_acc}")

```

```
313/313 - 1s - loss: 1.3213 - accuracy: 0.5957 - 704ms/epoch - 2ms/step
Test accuracy: 0.5957000255584717
```

```python
import matplotlib.pyplot as plt

# Function to plot training history
def plot_history(history, model_name):
    plt.plot(history.history['accuracy'], label='Training Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.ylim([0, 1])
    plt.title(f'{model_name} Training and Validation Accuracy')
    plt.legend(loc='lower right')
    plt.show()

# Plot the training history for LeNet-5
plot_history(history, "LeNet-5")

# Plot the training history for VGG16
plot_history(vgg16_history, "VGG16")

```





```
predictions = vgg16_model.predict(test_images[:5])

# Display the first 5 test images and their predicted labels
for i in range(5):
    plt.imshow(test_images[i])
    plt.show()
    print(f"Predicted label: {np.argmax(predictions[i])}")
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-37-ca0b7faaae12> in <cell line: 1>()
----> 1 predictions = vgg16_model.predict(test_images[:5])
      2 
      3 # Display the first 5 test images and their predicted labels
      4 for i in range(5):
      5     plt.imshow(test_images[i])

AttributeError: 'float' object has no attribute 'predict'
```

Next steps: Explain error

```
predictions = model.predict(test_images[:5])

# Display the first 5 test images and their predicted labels
for i in range(5):
    plt.imshow(test_images[i])
```