```
import cv2
import tensorflow as tf
import matplotlib.pyplot as plt
```

```
a=cv2.imread('/content/b04a3baae251ce9d3348f769179b66f1.jpg',cv2.IMREAD_GRAYSCALE)
```

```
a=cv2.resize(a,(64,64))
```

```
a
```

```
ndarray (64, 64) show data
```

```
a=a/255.0
```

```
a
```

```
array([[0.70588235, 0.71372549, 0.57647059, ..., 0.14901961, 0.71764706,
        0.8745098 ],
       [0.71372549, 0.71764706, 0.60784314, ..., 0.34509804, 0.62352941,
        0.76078431],
       [0.7254902 , 0.72941176, 0.50588235, ..., 0.22745098, 0.30588235,
        0.4       ],
       ...,
       [0.33333333, 0.36078431, 0.19607843, ..., 0.66666667, 0.50588235,
        0.17254902],
       [0.44313725, 0.5254902 , 0.1372549 , ..., 0.62352941, 0.43137255,
        0.36078431],
       [0.74117647, 0.43921569, 0.11764706, ..., 0.54901961, 0.35294118,
        0.31764706]])
```

```
a=a.reshape(1,a.shape[0],a.shape[1],1)
```

```
a.shape    #(batch size, height, weidth, channel) we will give last value as 3 for coloured image
```

```
(1, 64, 64, 1)
```

```
layer=tf.keras.layers.Conv2D(filters=1,kernel_size=(3,3),strides=(1,1),padding='valid')
```

```
pooling_layer=tf.keras.layers.MaxPooling2D(pool_size=(2,2),strides=(2,2))   #(pool size is must)
```

```
con_output=layer(a)
```

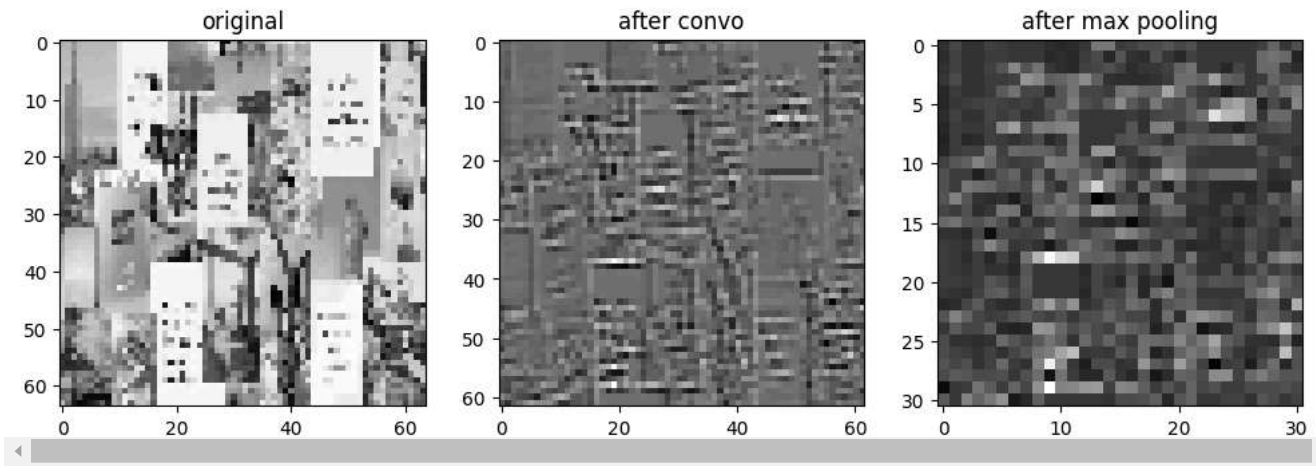
```
pooled_output=pooling_layer(con_output)
```

```
print(a.shape)
print(con_output.shape)
print(pooled_output.shape)
```

```
(1, 64, 64, 1)
(1, 62, 62, 1)
(1, 31, 31, 1)
```

```
fig,axs=plt.subplots(1,3,figsize=(12,4))
axs[0].imshow(a[0,:,:,0],cmap='gray')
axs[0].set_title('original')
axs[1].imshow(con_output[0,:,:,0],cmap='gray')
axs[1].set_title('after convo')
axs[2].imshow(pooled_output[0,:,:,0],cmap='gray')
axs[2].set_title('after max pooling')
```

```
Text(0.5, 1.0, 'after max pooling')
```



18-10-24

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

```
(train_images,train_labels),(test_images,test_labels)=datasets.cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 ━━━━━━━━━━━━━━━━━━━━ 3s 0us/step
```

```
print(train_images.shape)
print(test_images.shape)
```

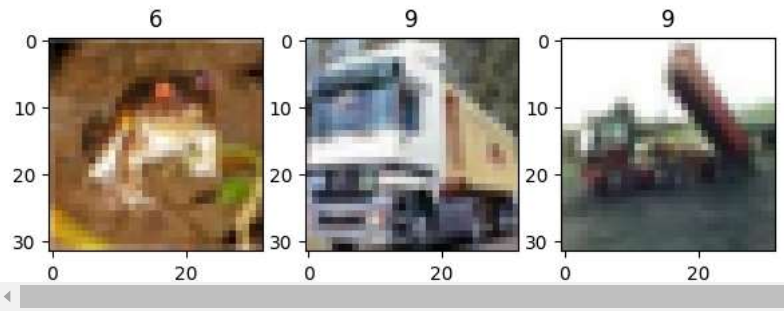
```
(50000, 32, 32, 3)
(10000, 32, 32, 3)
```

```
fig,axs=plt.subplots(1,3,figsize=(7,7))
axs[0].imshow(train_images[0])
axs[1].imshow(train_images[1])
axs[2].imshow(train_images[2])

axs[0].set_title(train_labels[0][0])
```

```
axs[1].set_title(train_labels[1][0])
axs[2].set_title(train_labels[2][0])
```

```
Text(0.5, 1.0, '9')
```




```
train_images=train_images/255.0
test_images=test_images/255.0
```

```
model=models.Sequential([
    layers.Conv2D(32,(3,3),activation='relu',input_shape=(32,32,3)),       # (30,30,32)
    layers.MaxPooling2D((2,2)),                              #(15,15,32)
    layers.Conv2D(64,(3,3),activation='relu'),               # (13,13,64)
    layers.MaxPooling2D((2,2)),                              #(6,6,64)
    layers.Conv2D(64,(3,3),activation='relu'),               # (4,4,64)
    layers.Flatten(),                                        #(1024,1)
    layers.Dense(64,activation='relu'),
    layers.Dense(10,activation='softmax')
    ])
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, pr
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
from tensorflow.keras.losses import SparseCategoricalCrossentropy
model.compile(optimizer='adam',loss=SparseCategoricalCrossentropy(),metrics=['accuracy'])
```

```
from tensorflow.keras.callbacks import EarlyStopping
early_stopping=EarlyStopping(patience=5,restore_best_weights=True)

history=model.fit(x=train_images,y=train_labels,epochs=10,validation_split=0.2,shuffle=True,class_weight={0:1.0,1:2.0},callbacks=[early_stopping],verbose=1)
```

```
Epoch 1/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 65s 51ms/step - accuracy: 0.4992 - loss: 1.4637 - val_accuracy: 0.5698 - val_loss: 1.2096
Epoch 2/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 59s 47ms/step - accuracy: 0.5919 - loss: 1.2069 - val_accuracy: 0.6099 - val_loss: 1.1063
Epoch 3/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 79s 44ms/step - accuracy: 0.6451 - loss: 1.0604 - val_accuracy: 0.6431 - val_loss: 1.0408
Epoch 4/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 83s 45ms/step - accuracy: 0.6752 - loss: 0.9630 - val_accuracy: 0.6520 - val_loss: 1.0179
Epoch 5/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 82s 45ms/step - accuracy: 0.7086 - loss: 0.8748 - val_accuracy: 0.6911 - val_loss: 0.9021
Epoch 6/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 56s 45ms/step - accuracy: 0.7266 - loss: 0.8131 - val_accuracy: 0.6829 - val_loss: 0.9260
Epoch 7/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 80s 44ms/step - accuracy: 0.7403 - loss: 0.7622 - val_accuracy: 0.6850 - val_loss: 0.9144
Epoch 8/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 54s 44ms/step - accuracy: 0.7590 - loss: 0.6924 - val_accuracy: 0.6989 - val_loss: 0.9090
Epoch 9/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 82s 44ms/step - accuracy: 0.7785 - loss: 0.6397 - val_accuracy: 0.6926 - val_loss: 0.9256
Epoch 10/10
1250/1250 ━━━━━━━━━━━━━━━━━━━━ 57s 46ms/step - accuracy: 0.7946 - loss: 0.5977 - val_accuracy: 0.6883 - val_loss: 0.9461
```

```
model.summary()
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_2 (Conv2D)	(None, 4, 4, 64)	36,928
flatten (Flatten)	(None, 1024)	0
dense (Dense)	(None, 64)	65,600
dense_1 (Dense)	(None, 10)	650

```
Total params: 367,712 (1.40 MB)
Trainable params: 122,570 (478.79 KB)
```

```
model.evaluate(test_images,test_labels)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step - accuracy: 0.6963 - loss: 0.9005
[0.9095031023025513, 0.6919999718666077]
```

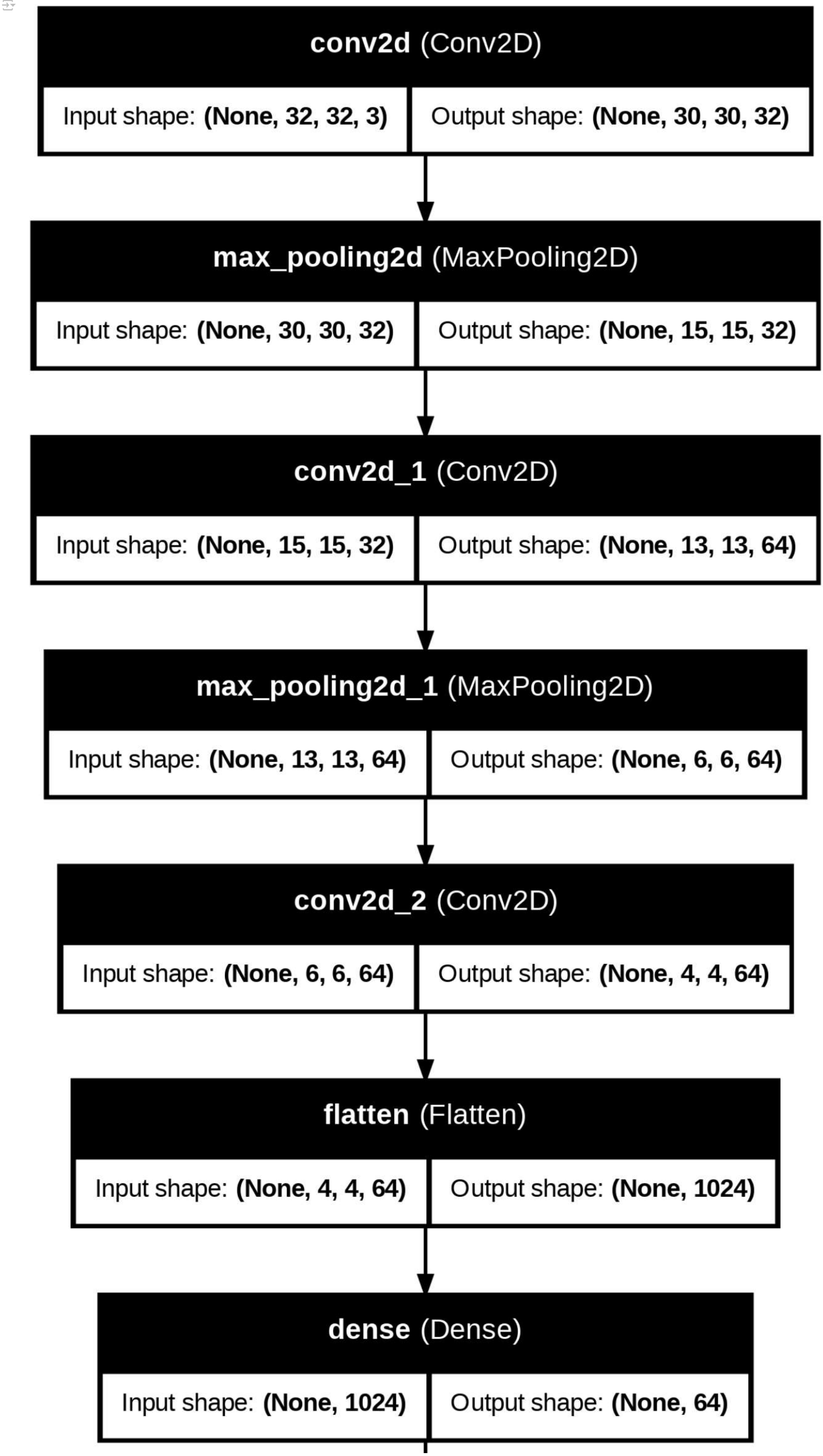
```
# convolutional layer
#parameters=(filter_height*filter_width*input_channels)*number_ of_filter+ no.of filter
#fully_connected_layer/dense_layer
# dense_layer=last_layer_neurons*current_layer_neurons+ current_layers_neurons
```

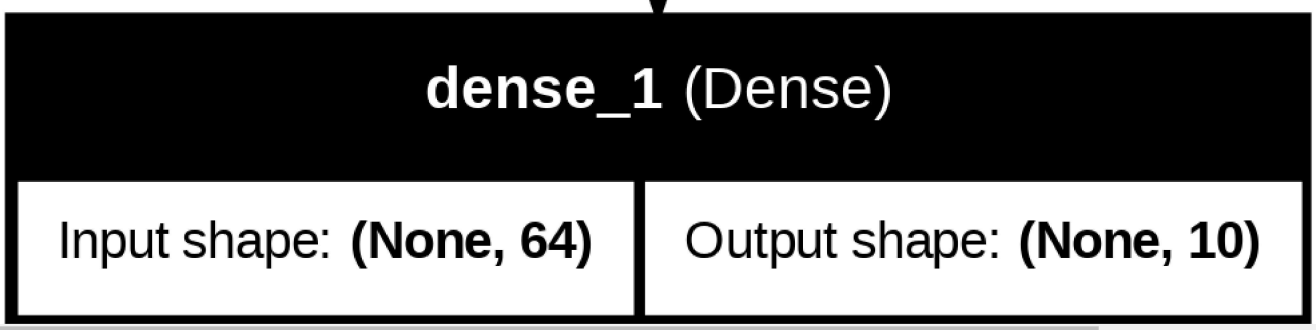
```
pip install pydot graphviz
```

```
Requirement already satisfied: pydot in /usr/local/lib/python3.10/dist-packages (3.0.2)
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-packages (0.20.3)
Requirement already satisfied: pyparsing>=3.0.9 in /usr/local/lib/python3.10/dist-packages (from pydot) (3.1.4)
```

```
from tensorflow.keras.utils import plot_model

plot_model(model, to_file='model_diagrams.png', show_shapes=True, show_layer_names=True)
```


conv2d (Conv2D)
Input shape: (None, 32, 32, 3)
Output shape: (None, 30, 30, 32)
max_pooling2d (MaxPooling2D)
Input shape: (None, 30, 30, 32)
Output shape: (None, 15, 15, 32)
conv2d_1 (Conv2D)
Input shape: (None, 15, 15, 32)
Output shape: (None, 13, 13, 64)
max_pooling2d_1 (MaxPooling2D)
Input shape: (None, 13, 13, 64)
Output shape: (None, 6, 6, 64)
conv2d_2 (Conv2D)
Input shape: (None, 6, 6, 64)
Output shape: (None, 4, 4, 64)
flatten (Flatten)
Input shape: (None, 4, 4, 64)
Output shape: (None, 1024)
dense (Dense)
Input shape: (None, 1024)
Output shape: (None, 64)




```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

plt.show()

plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label = 'val_loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='lower right')

plt.show()
```