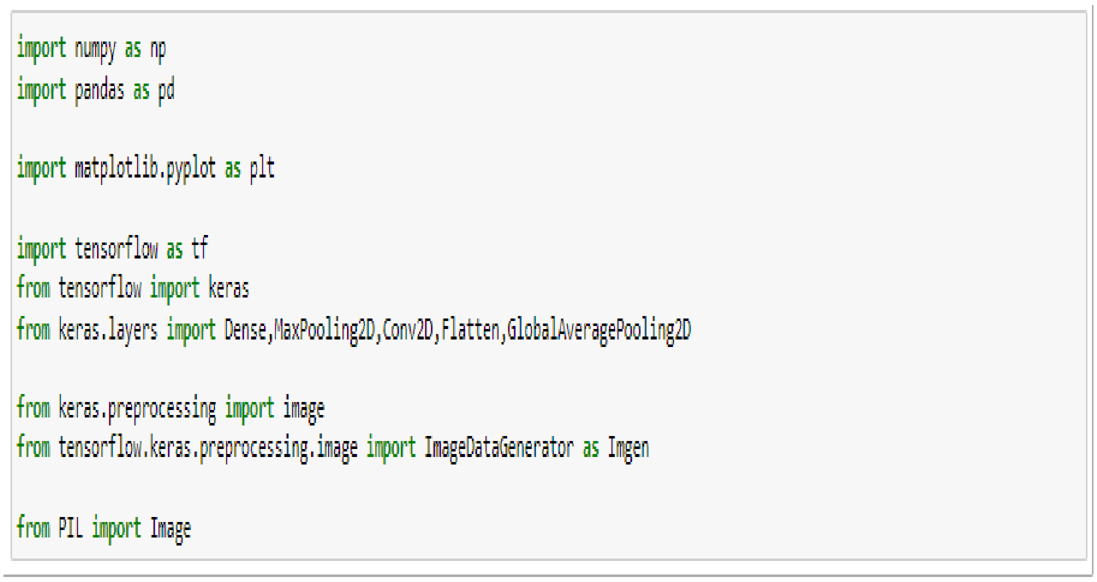
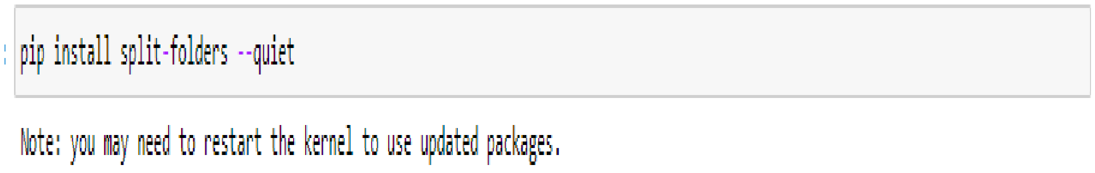
## CODE

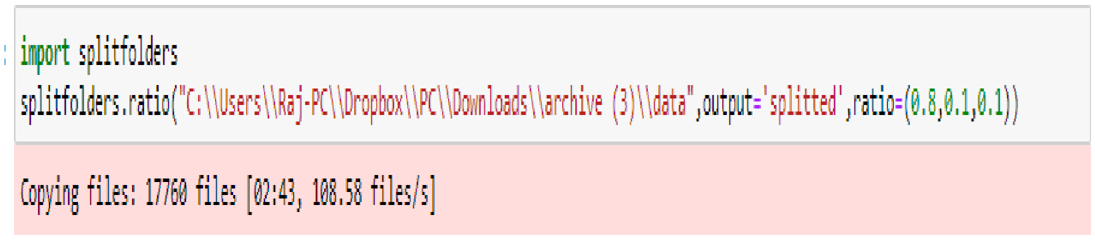


```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow import keras
from keras.layers import Dense,MaxPooling2D,Conv2D,Flatten,GlobalAveragePooling2D

from keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator as Imgen

from PIL import Image
```



```
pip install split-folders --quiet
```

```
Note: you may need to restart the kernel to use updated packages.
```



```
import splitfolders
splitfolders.ratio("C:\\Users\\Raj-PC\\Dropbox\\PC\\Downloads\\archive (3)\\data",output='splitted',ratio=(0.8,0.1,0.1))
```

```
Copying files: 17760 files [02:43, 108.58 files/s]
```

```python
train_ds = Imgen(rescale=1./255).flow_from_directory(
    "./splitted/train",
    seed = 1,
    target_size = (150,150),
    batch_size = 32
)

val_ds = Imgen(rescale=1./255).flow_from_directory(
    "./splitted/val",
    seed = 1,
    target_size = (150,150),
    batch_size = 32
)

test_ds = Imgen(rescale=1./255).flow_from_directory(
    "./splitted/test",
    seed = 1,
    target_size = (150,150),
    batch_size = 32
)
```

```
Found 14207 images belonging to 2 classes.
Found 1775 images belonging to 2 classes.
Found 1778 images belonging to 2 classes.
```

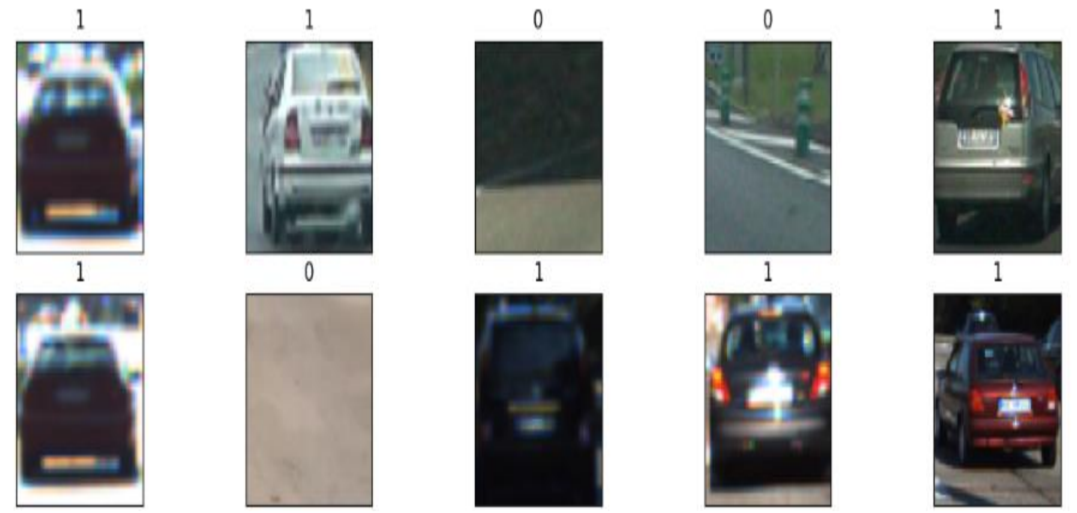
```python
train_ds.class_indices
```

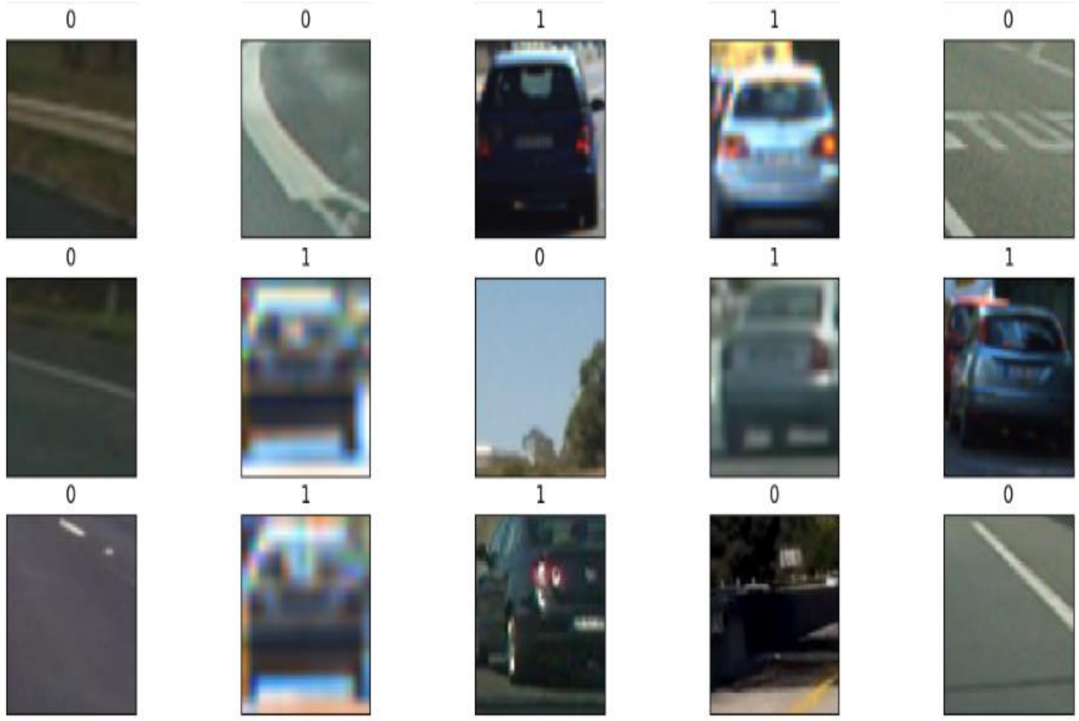
```
{'non-vehicles': 0, 'vehicles': 1}
```

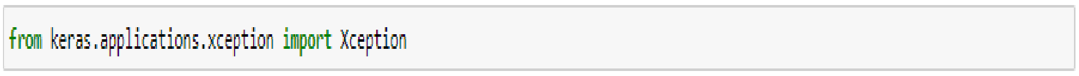
```python
x_train,y_train = next(train_ds)
print(x_train.shape,y_train.shape)
```

```
(32, 150, 150, 3) (32, 2)
```

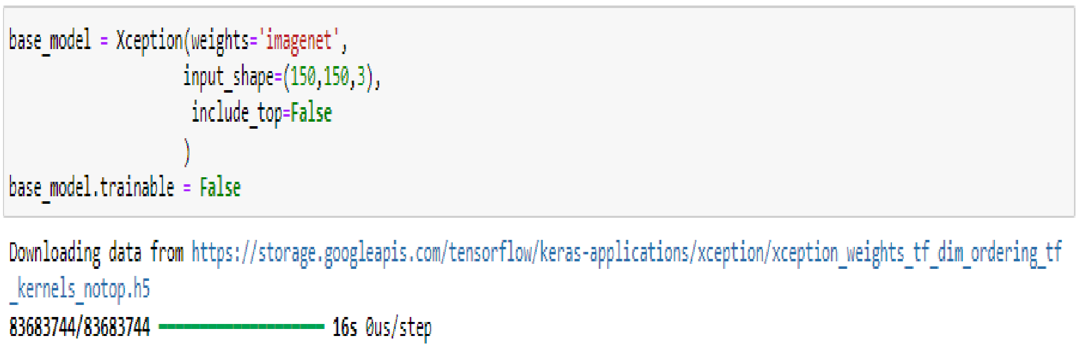
```
plt.figure(figsize=(15,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_train[i])
    plt.title(np.argmax(y_train[i]))
```





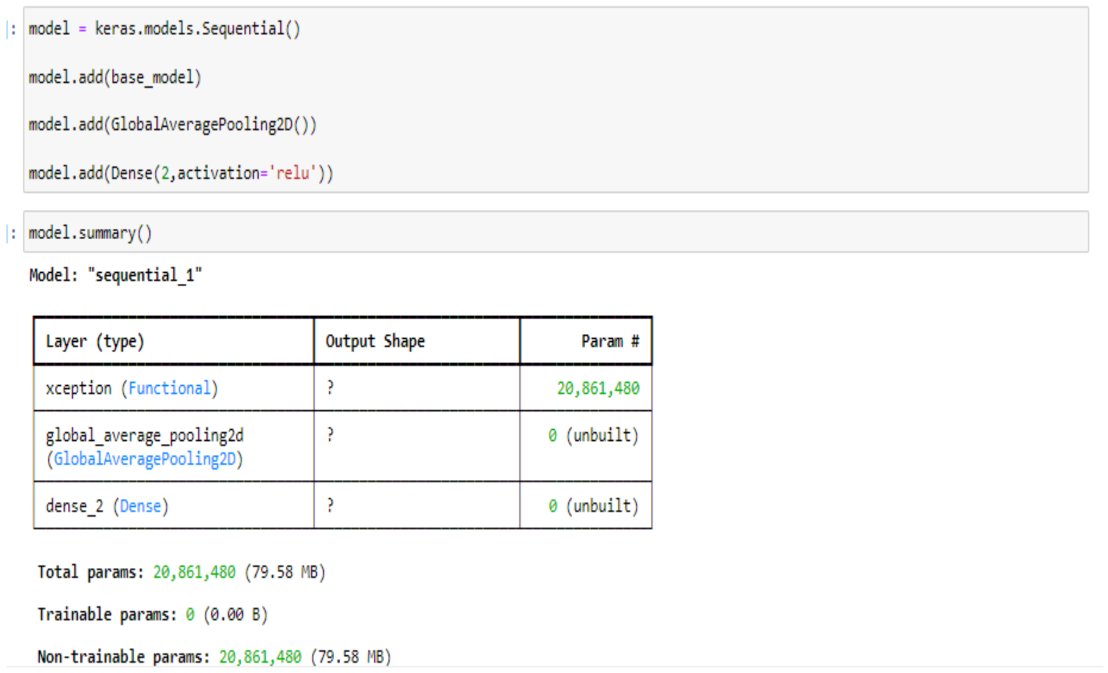


```python
from keras.applications.xception import Xception
```



```python
base_model = Xception(weights='imagenet',
                      input_shape=(150,150,3),
                       include_top=False
                      )
base_model.trainable = False
```

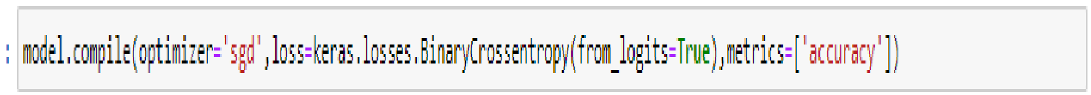
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_weights_tf_dim_ordering_tf_kernels_notop.h5
83683744/83683744 ━━━━━━━━━━━━━━━━━━━━ 16s 0us/step
```



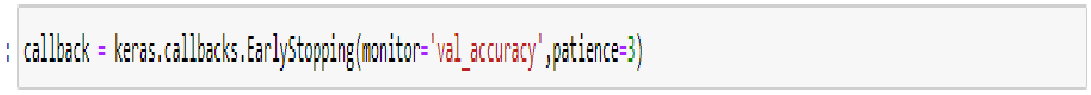
```python
model = keras.models.Sequential()

model.add(base_model)

model.add(GlobalAveragePooling2D())

model.add(Dense(2,activation='relu'))
```

```python
model.summary()
```

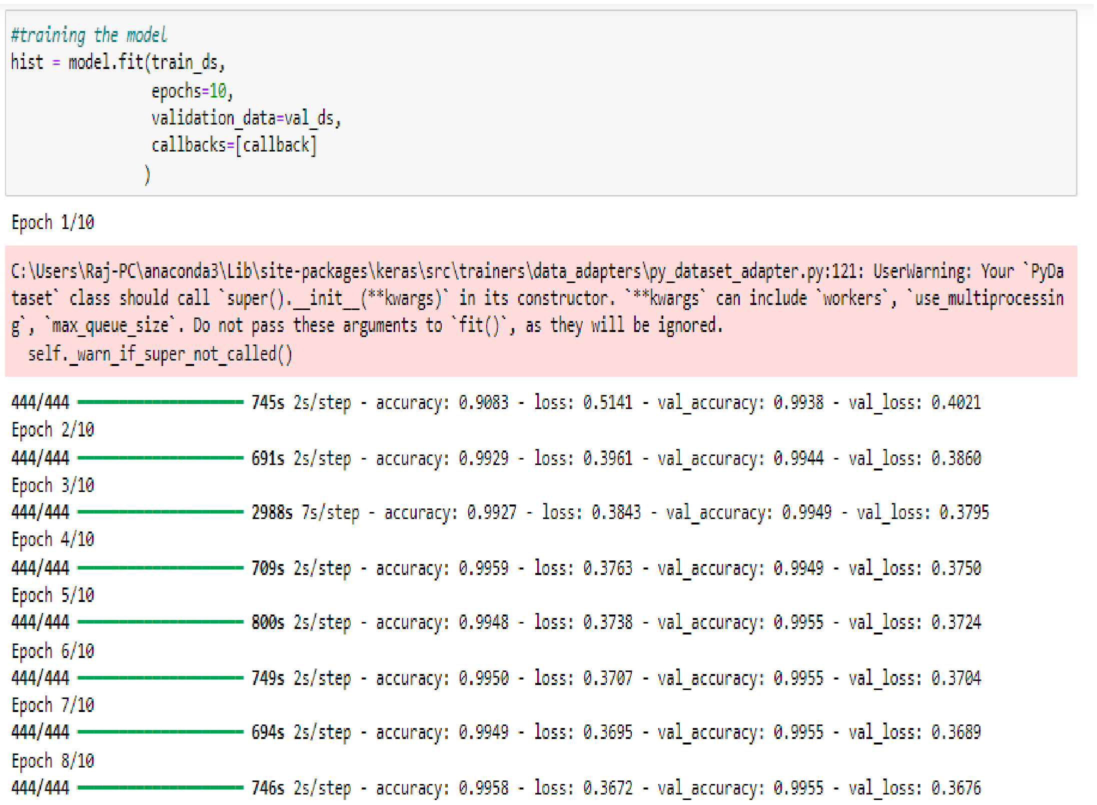
```
Model: "sequential_1"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━┓
┃ Layer (type)                     ┃ Output Shape            ┃        Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━┩
│ xception (Functional)            │ ?                       │     20,861,480 │
├──────────────────────────────────┼─────────────────────────┼────────────────┤
│ global_average_pooling2d         │ ?                       │    0 (unbuilt) │
│ (GlobalAveragePooling2D)         │                         │                │
├──────────────────────────────────┼─────────────────────────┼────────────────┤
│ dense_2 (Dense)                  │ ?                       │    0 (unbuilt) │
└──────────────────────────────────┴─────────────────────────┴────────────────┘

 Total params: 20,861,480 (79.58 MB)

 Trainable params: 0 (0.00 B)

 Non-trainable params: 20,861,480 (79.58 MB)
```



```python
model.compile(optimizer='sgd',loss=keras.losses.BinaryCrossentropy(from_logits=True),metrics=['accuracy'])
```

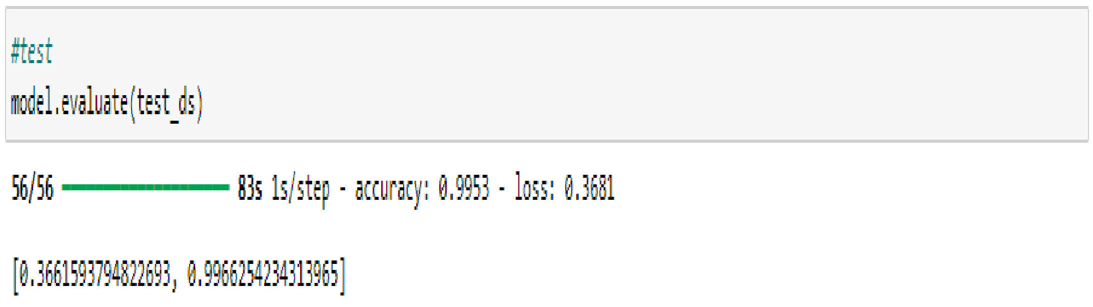


```python
callback = keras.callbacks.EarlyStopping(monitor='val_accuracy',patience=3)
```



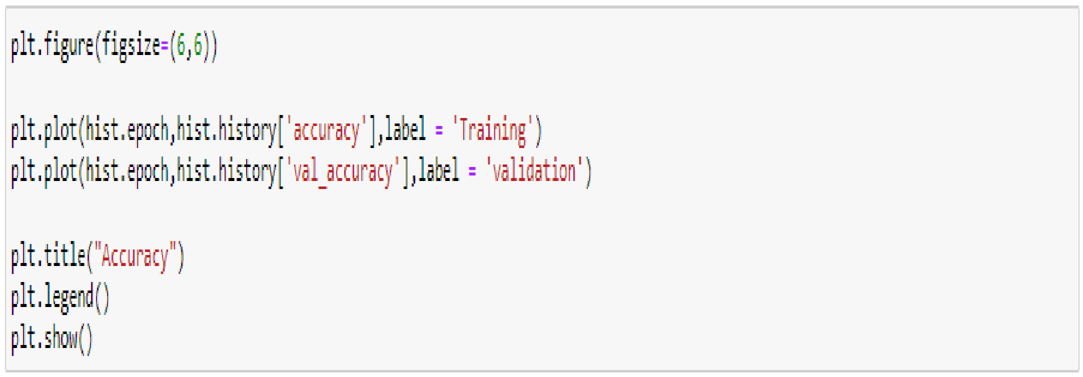
```
#training the model
hist = model.fit(train_ds,
                 epochs=10,
                 validation_data=val_ds,
                 callbacks=[callback]
                )
```

```
Epoch 1/10
C:\Users\Raj-PC\anaconda3\Lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDa
taset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessin
g`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
444/444 ━━━━━━━━━━━━━━━━━━━━ 745s 2s/step - accuracy: 0.9083 - loss: 0.5141 - val_accuracy: 0.9938 - val_loss: 0.4021
Epoch 2/10
444/444 ━━━━━━━━━━━━━━━━━━━━ 691s 2s/step - accuracy: 0.9929 - loss: 0.3961 - val_accuracy: 0.9944 - val_loss: 0.3860
Epoch 3/10
444/444 ━━━━━━━━━━━━━━━━━━━━ 2988s 7s/step - accuracy: 0.9927 - loss: 0.3843 - val_accuracy: 0.9949 - val_loss: 0.3795
Epoch 4/10
444/444 ━━━━━━━━━━━━━━━━━━━━ 709s 2s/step - accuracy: 0.9959 - loss: 0.3763 - val_accuracy: 0.9949 - val_loss: 0.3750
Epoch 5/10
444/444 ━━━━━━━━━━━━━━━━━━━━ 800s 2s/step - accuracy: 0.9948 - loss: 0.3738 - val_accuracy: 0.9955 - val_loss: 0.3724
Epoch 6/10
444/444 ━━━━━━━━━━━━━━━━━━━━ 749s 2s/step - accuracy: 0.9950 - loss: 0.3707 - val_accuracy: 0.9955 - val_loss: 0.3704
Epoch 7/10
444/444 ━━━━━━━━━━━━━━━━━━━━ 694s 2s/step - accuracy: 0.9949 - loss: 0.3695 - val_accuracy: 0.9955 - val_loss: 0.3689
Epoch 8/10
444/444 ━━━━━━━━━━━━━━━━━━━━ 746s 2s/step - accuracy: 0.9958 - loss: 0.3672 - val_accuracy: 0.9955 - val_loss: 0.3676
```

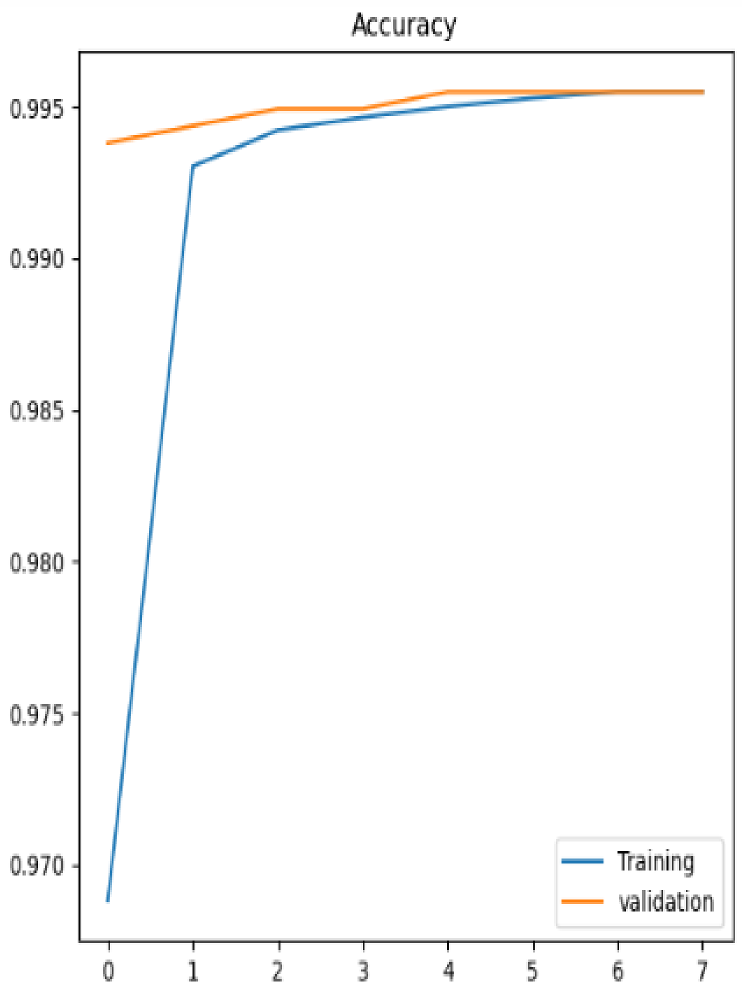


```
#test
model.evaluate(test_ds)
```

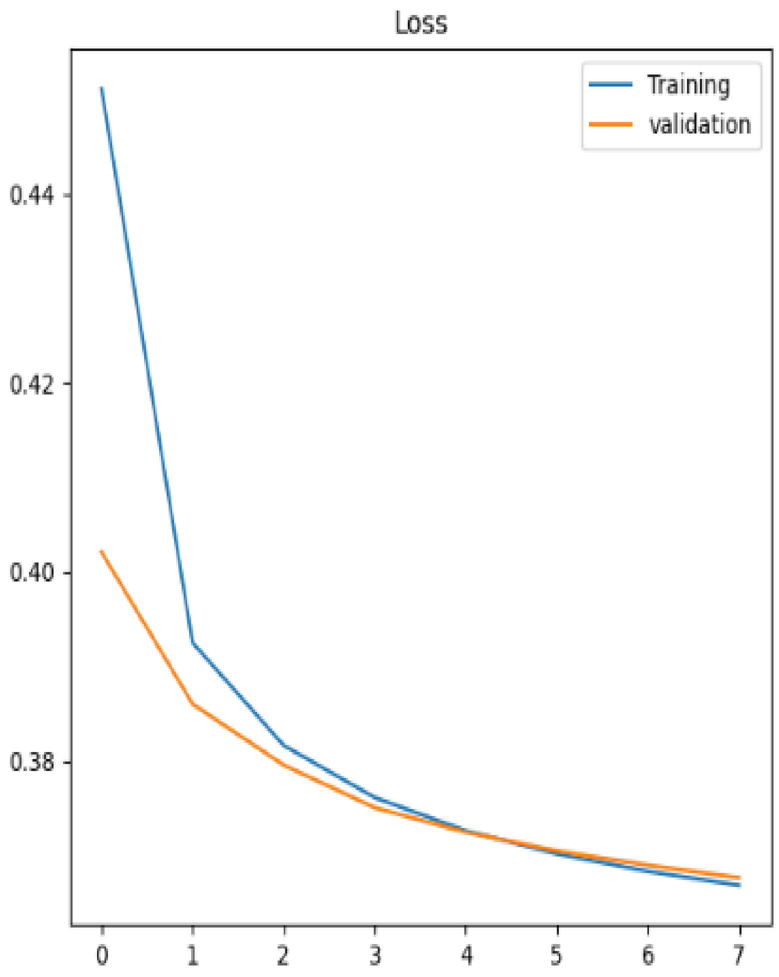
```
56/56 ━━━━━━━━━━━━━━━━━━━━ 83s 1s/step - accuracy: 0.9953 - loss: 0.3681

[0.3661593794822693, 0.9966254234313965]
```



```
plt.figure(figsize=(6,6))

plt.plot(hist.epoch,hist.history['accuracy'],label = 'Training')
plt.plot(hist.epoch,hist.history['val_accuracy'],label = 'validation')

plt.title("Accuracy")
plt.legend()
plt.show()
```



```
plt.figure(figsize=(6,6))

plt.plot(hist.epoch,hist.history['loss'],label = 'Training')
plt.plot(hist.epoch,hist.history['val_loss'],label = 'validation')

plt.title("Loss")
plt.legend()
plt.show()
```



```
x_test,y_test = next(test_ds)
```

```
pred_test = model.predict(x_test)
pred_test = [np.argmax(i) for i in pred_test]
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
```

```
test_dict = test_ds.class_indices
li_pred = list(test_dict.keys())
li_pred
```

```
['non-vehicles', 'vehicles']
```

```python
plt.figure(figsize=(15,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(x_test[i])
    plt.xlabel("Actual: {}".format(li_pred[np.argmax(y_test[i])]))
    plt.ylabel("Pred: {}".format(li_pred[pred_test[i]]))
```