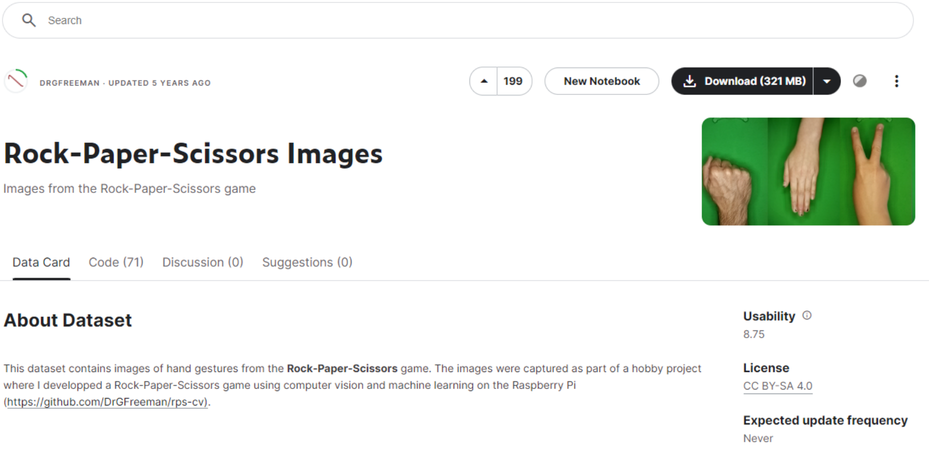
Nama Dosen : Teguh Iman Hermanto, M.Kom
Mata Kuliah : Machine Learning 2
Pembahasan : CNN Rock-Paper-Scissors
Pokok Pemb : Membuat model klasifikasi penyakit diabetes menggunakan ANN

DOWNLOAD DATASET

Import dataset ke dalam google colab menggunakan Alamat dataset berikut :
https://www.kaggle.com/datasets/drgfreeman/rockpaperscissors



Rock-Paper-Scissors Images

Images from the Rock-Paper-Scissors game

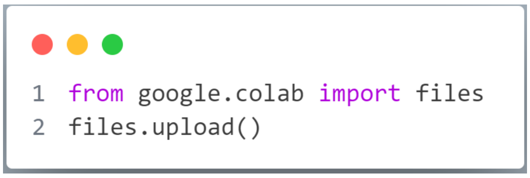
About Dataset

This dataset contains images of hand gestures from the **Rock-Paper-Scissors** game. The images were captured as part of a hobby project where I developped a Rock-Paper-Scissors game using computer vision and machine learning on the Raspberry Pi (https://github.com/DrGFreeman/rps-cv).

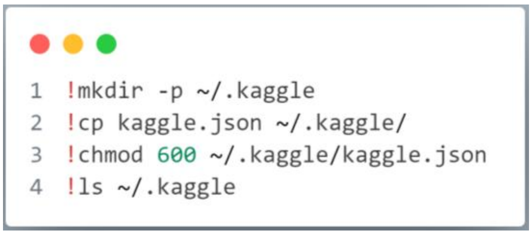
Usability
8.75

License
CC BY-SA 4.0

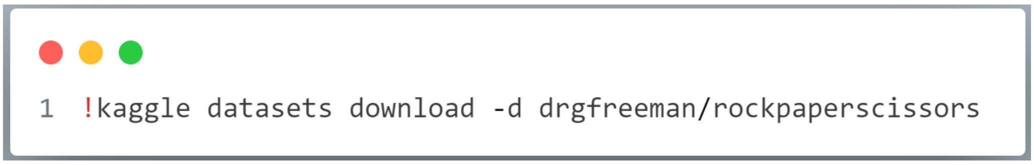
Expected update frequency
Never



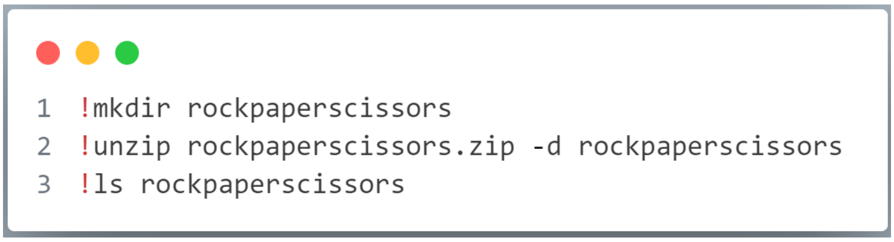
```
from google.colab import files
files.upload()
```



```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
!ls ~/.kaggle
```

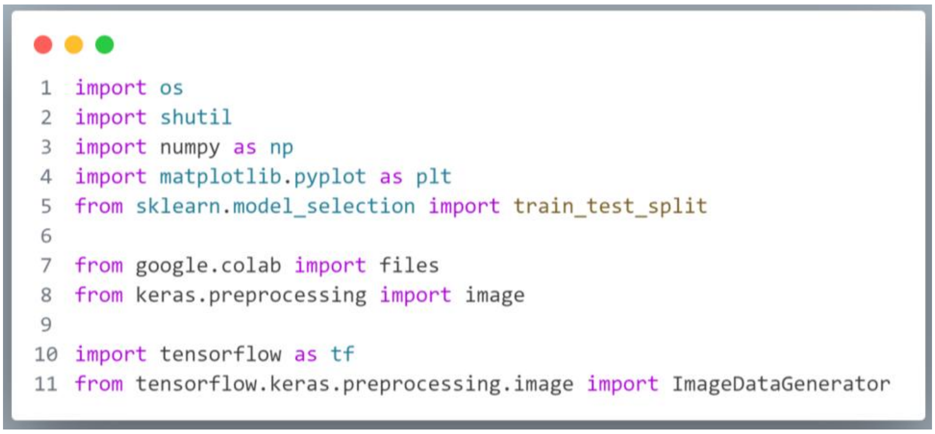


```
!kaggle datasets download -d drgfreeman/rockpaperscissors
```



```
!mkdir rockpaperscissors
!unzip rockpaperscissors.zip -d rockpaperscissors
!ls rockpaperscissors
```

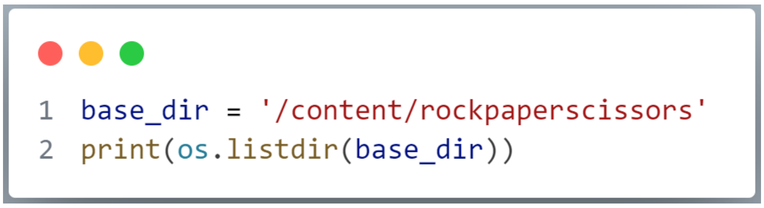
IMPORT LIBARRY



```python
import os
import shutil
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

from google.colab import files
from keras.preprocessing import image

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

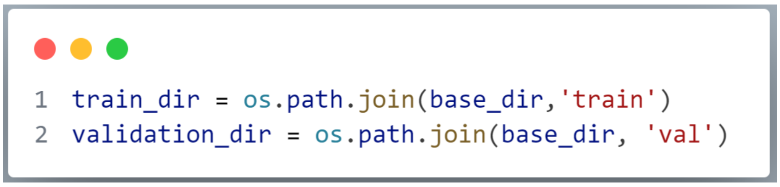
TENTUKAN DIREKTORI IMAGE

Langkah awal buat base_dir sebagai direktori utama dataset menggunakan perintah berikut :



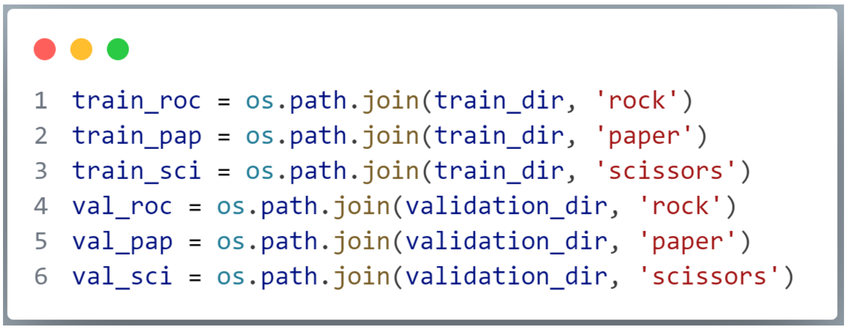
```python
base_dir = '/content/rockpaperscissors'
print(os.listdir(base_dir))
```

Setelah itu tentukan direktori untuk data training dan data validation menggunakan perintah berikut :



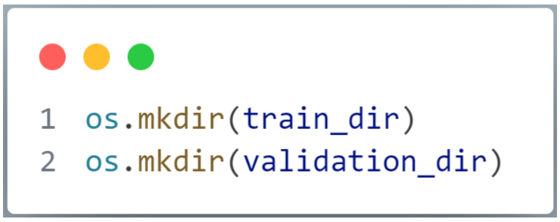
```python
train_dir = os.path.join(base_dir,'train')
validation_dir = os.path.join(base_dir, 'val')
```

Tentukan direktori untuk data training rock, paper, scissor dan tentukan juga direktori untuk data validation rock, paper, scissor



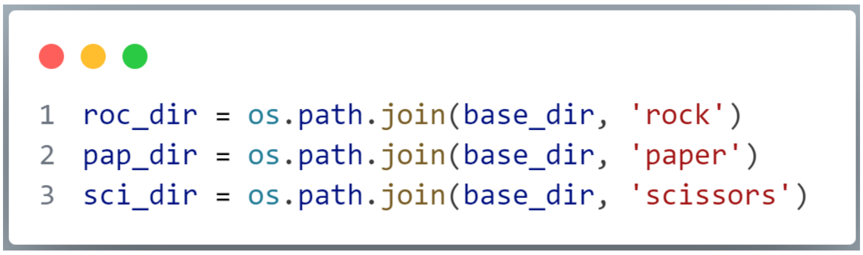
```python
train_roc = os.path.join(train_dir, 'rock')
train_pap = os.path.join(train_dir, 'paper')
train_sci = os.path.join(train_dir, 'scissors')
val_roc = os.path.join(validation_dir, 'rock')
val_pap = os.path.join(validation_dir, 'paper')
val_sci = os.path.join(validation_dir, 'scissors')
```

- Folder train_dir untuk data training
- Folder validation_dir untuk data validation

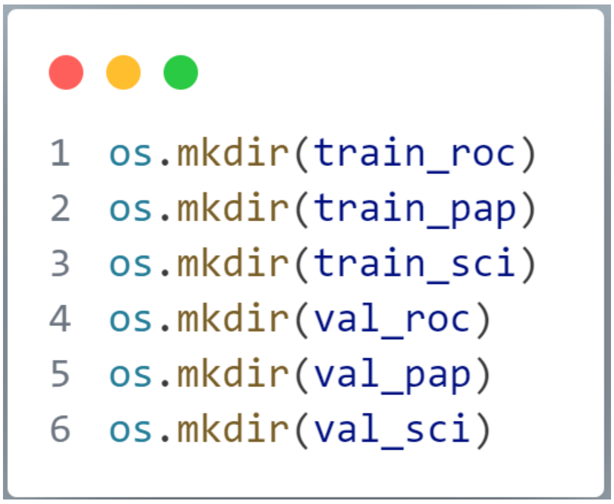


```
os.mkdir(train_dir)
os.mkdir(validation_dir)
```

Tentukan direktori untuk rock, paper dan scissor



```
roc_dir = os.path.join(base_dir, 'rock')
pap_dir = os.path.join(base_dir, 'paper')
sci_dir = os.path.join(base_dir, 'scissors')
```



```
os.mkdir(train_roc)
os.mkdir(train_pap)
os.mkdir(train_sci)
os.mkdir(val_roc)
os.mkdir(val_pap)
os.mkdir(val_sci)
```

Folder train_roc untuk data training rock

Folder train_pap untuk data training paper

Folder train_sci untuk data training scissor

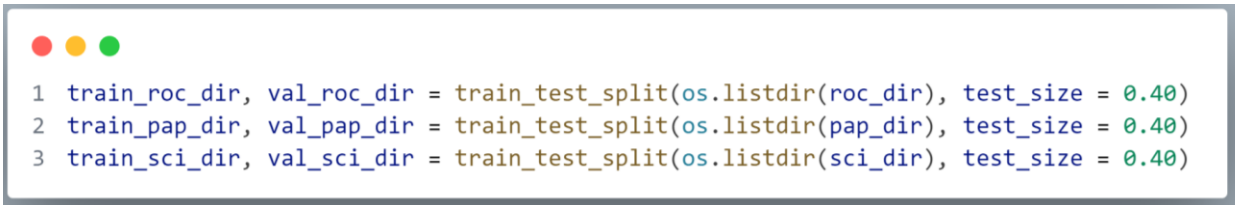
Forder val_roc untuk data validation rock

Folder val_pap untuk data validation paper

Folder val_sci untuk data validation scissor

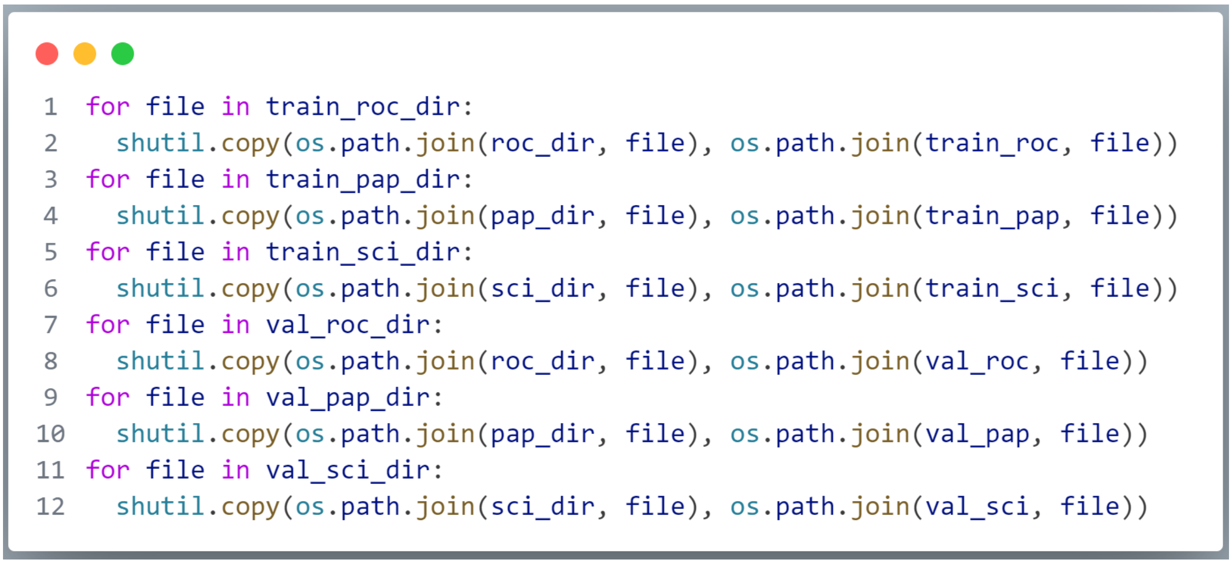
PREPROCESSING DATA

Gunakan perintah berikut untuk split data training dan validation dengan pembagian data test 0.40



```
train_roc_dir, val_roc_dir = train_test_split(os.listdir(roc_dir), test_size = 0.40)
train_pap_dir, val_pap_dir = train_test_split(os.listdir(pap_dir), test_size = 0.40)
train_sci_dir, val_sci_dir = train_test_split(os.listdir(sci_dir), test_size = 0.40)
```

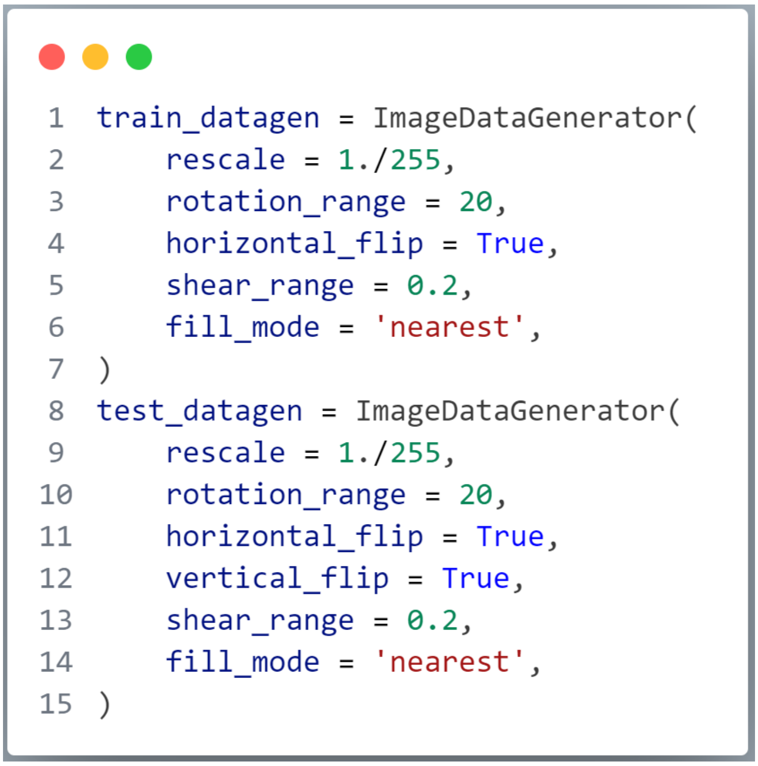
Setelah itu gunakan perintah shutil untuk memindahkan data hasil split ke masing-masing direktori data training dan data validation



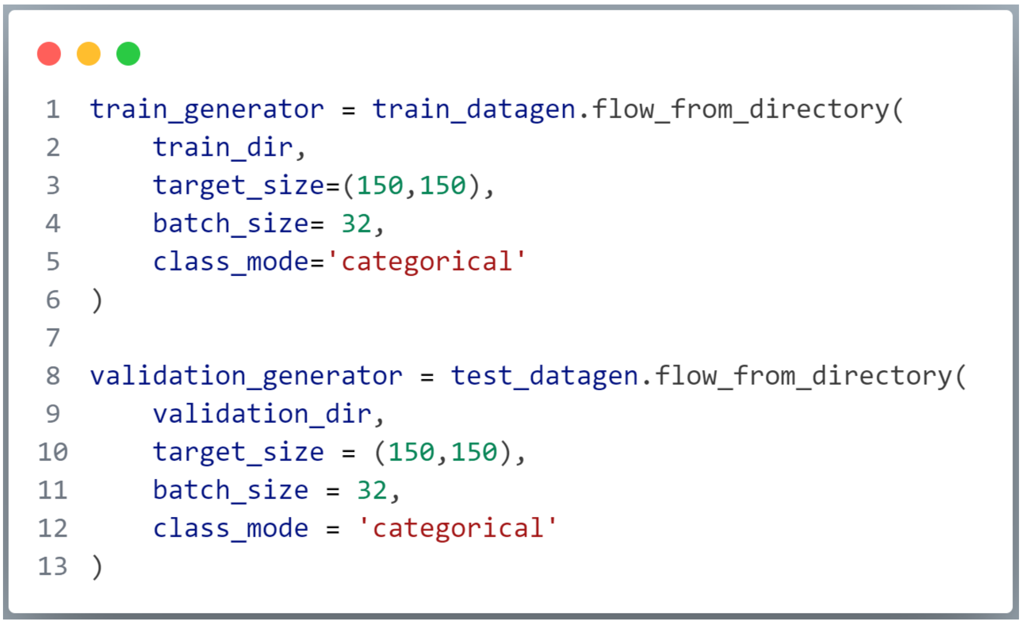
```
for file in train_roc_dir:
  shutil.copy(os.path.join(roc_dir, file), os.path.join(train_roc, file))
for file in train_pap_dir:
  shutil.copy(os.path.join(pap_dir, file), os.path.join(train_pap, file))
for file in train_sci_dir:
  shutil.copy(os.path.join(sci_dir, file), os.path.join(train_sci, file))
for file in val_roc_dir:
  shutil.copy(os.path.join(roc_dir, file), os.path.join(val_roc, file))
for file in val_pap_dir:
  shutil.copy(os.path.join(pap_dir, file), os.path.join(val_pap, file))
for file in val_sci_dir:
  shutil.copy(os.path.join(sci_dir, file), os.path.join(val_sci, file))
```

Image data generator digunakan untuk melakukan augmentasi data dengan parameter yang kita gunakan adalah :

- Rescale untuk membuat skala baru pada gambar
- Rotatation_range kisaran berapa derajat gambar diputar
- Horizontal_flip untuk membalikan gambar secara horizontal
- Shear_range untuk menggeser gambar berlawanan arah jarum jam
- Fill_mode secara default ‘nearest’

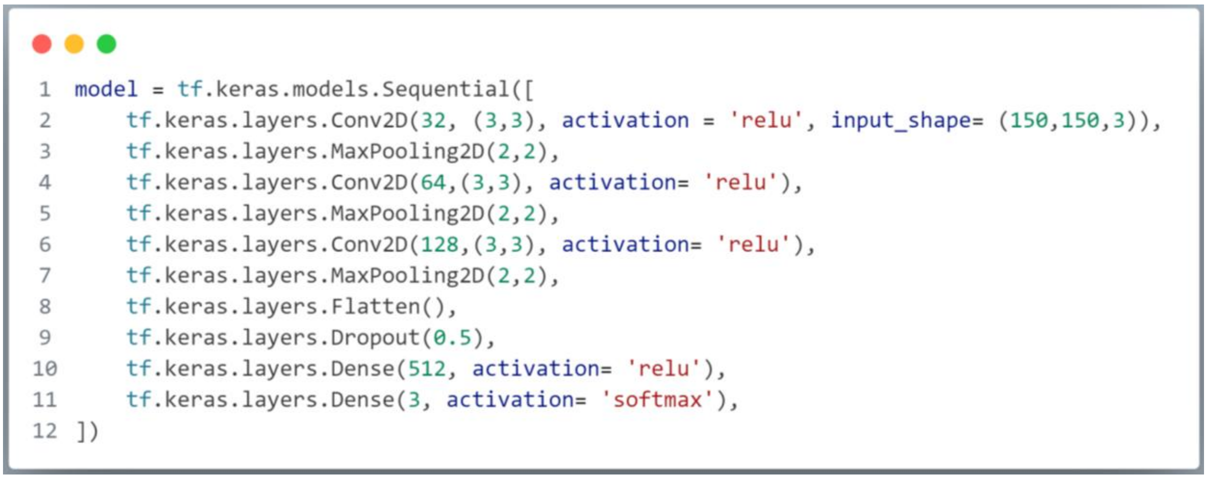


```
train_datagen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range = 20,
    horizontal_flip = True,
    shear_range = 0.2,
    fill_mode = 'nearest',
)
test_datagen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range = 20,
    horizontal_flip = True,
    vertical_flip = True,
    shear_range = 0.2,
    fill_mode = 'nearest',
)
```



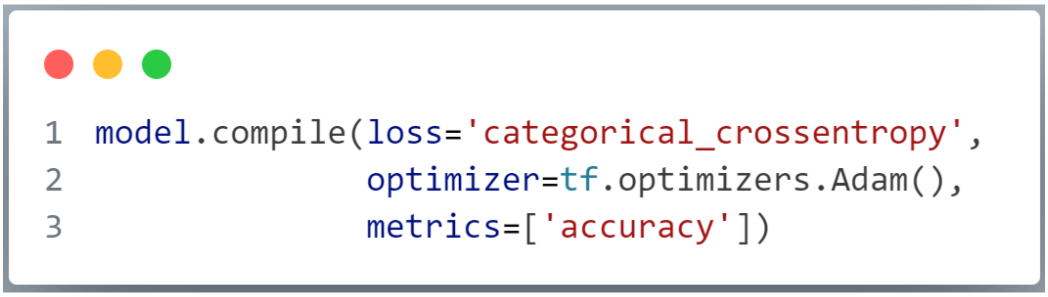
```
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(150,150),
    batch_size= 32,
    class_mode='categorical'
)

validation_generator = test_datagen.flow_from_directory(
    validation_dir,
    target_size = (150,150),
    batch_size = 32,
    class_mode = 'categorical'
)
```

MODELING

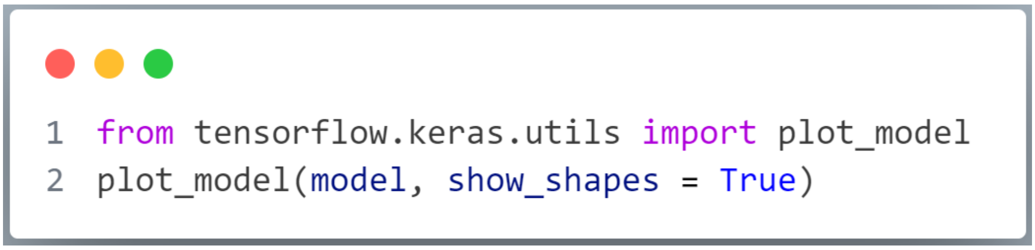


```python
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation = 'relu', input_shape= (150,150,3)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64,(3,3), activation= 'relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128,(3,3), activation= 'relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(512, activation= 'relu'),
    tf.keras.layers.Dense(3, activation= 'softmax'),
])
```

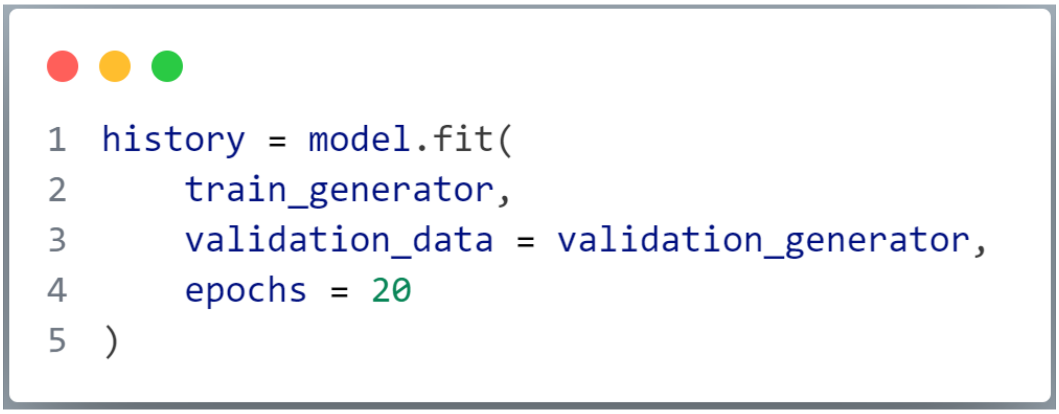
Gunakan perintah “**model.summary**” untuk melihat tipe model, jumlah layer, output shape dan parameter



```python
model.compile(loss='categorical_crossentropy',
              optimizer=tf.optimizers.Adam(),
              metrics=['accuracy'])
```

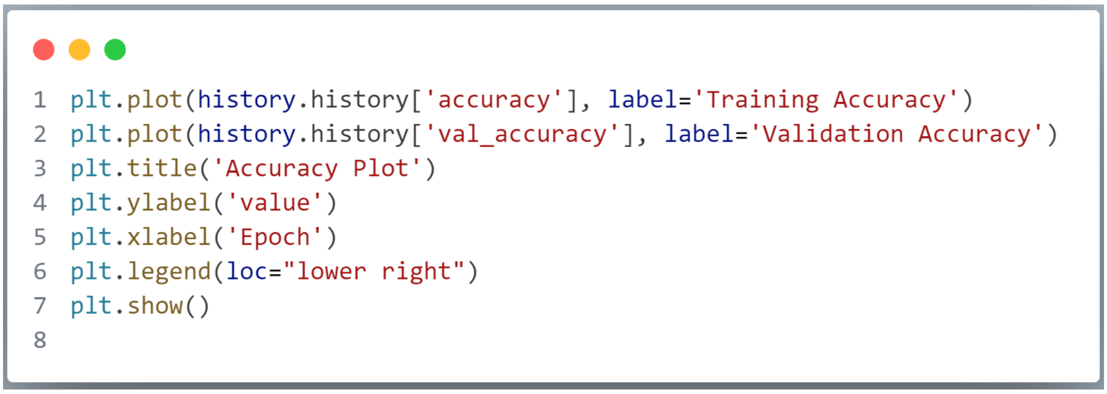


```python
from tensorflow.keras.utils import plot_model
plot_model(model, show_shapes = True)
```



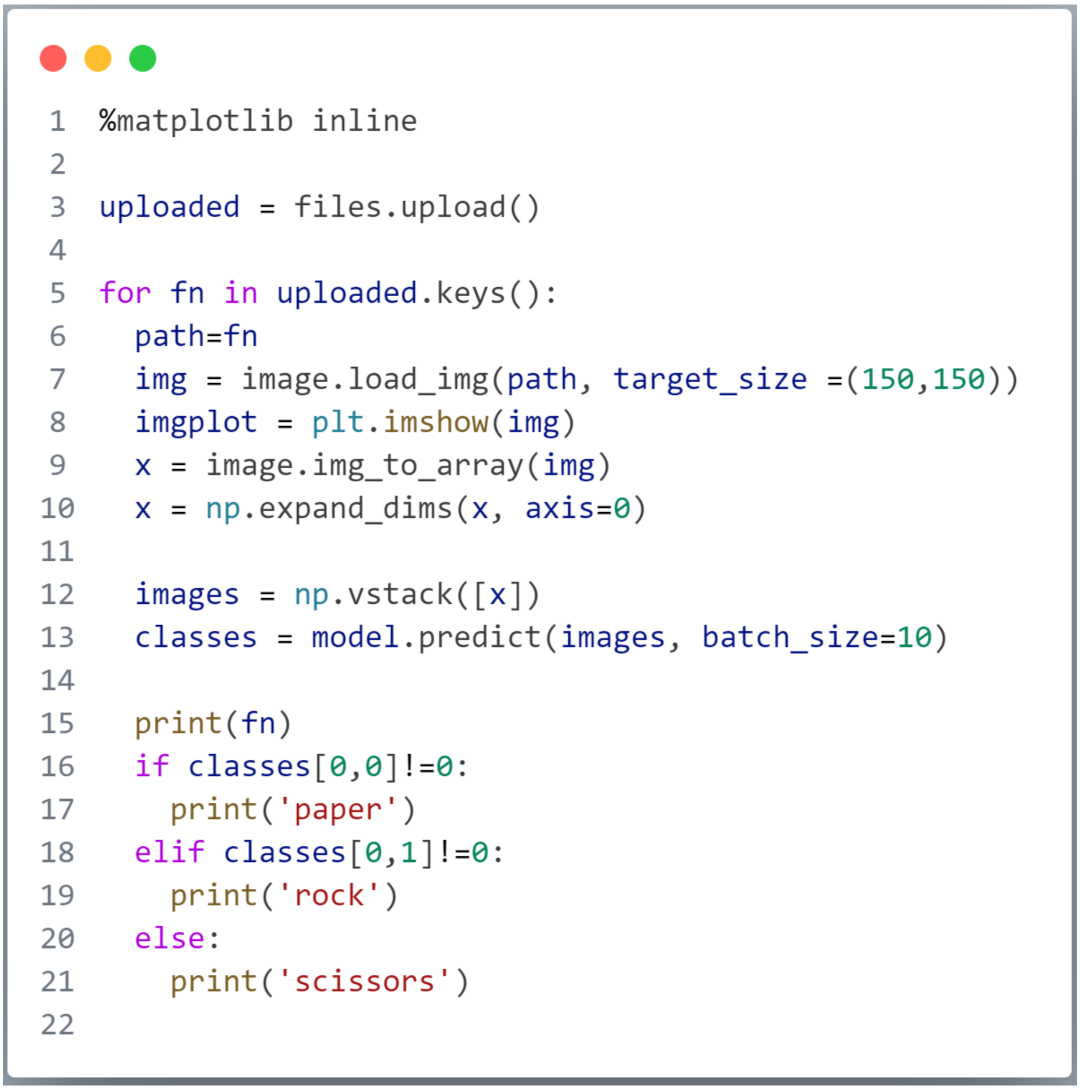
```python
history = model.fit(
    train_generator,
    validation_data = validation_generator,
    epochs = 20
)
```

TESTING MODEL



```
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Accuracy Plot')
plt.ylabel('value')
plt.xlabel('Epoch')
plt.legend(loc="lower right")
plt.show()

```

SIMULASI MODEL



```
%matplotlib inline

uploaded = files.upload()

for fn in uploaded.keys():
  path=fn
  img = image.load_img(path, target_size =(150,150))
  imgplot = plt.imshow(img)
  x = image.img_to_array(img)
  x = np.expand_dims(x, axis=0)

  images = np.vstack([x])
  classes = model.predict(images, batch_size=10)

  print(fn)
  if classes[0,0]!=0:
    print('paper')
  elif classes[0,1]!=0:
    print('rock')
  else:
    print('scissors')

```