DIAN MAULIDA

FINAL PROJECT MACHINE LEARNING DEVELOPER

Image Classification

August, 2021

PURPOSE

CREATE AN ARTIFICIAL NEURAL NETWORK PROGRAM USING TENSORFLOW THAT IS ABLE TO RECOGNIZE THE SHAPE OF A HAND THAT FORMS SCISSORS, ROCK, OR PAPER.

Dataset:
https://github.com/dicodingacademy/assets/releases/download/release/rockpaperscissors.zip

August, 2021

STAGES

1

Import TensorFlow on Google Colab

2

Download dataset and extract files by zip method

3

Augment and separate data

4

Build a sequential model

5

Train the model to get the desired accuracy

6

Test the model

August, 2021

PROGRAM SYNTAX

+ Kode + Teks

Hubungkan

Final Submission - Machine Learning Developer

Dian Maulida

dian.maulida103@gmail.com

Mengimport Tensorflow

```
[ ] import tensorflow as tf
    from tensorflow.keras.optimizers import RMSprop
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Mengakses Dataset

Mengunduh dataset

```
[ ] !wget --no-check-certificate \
      https://github.com/dicodingacademy/assets/releases/download/release/rockpaperscissors.zip
```

```
--2021-07-28 04:58:20--  https://github.com/dicodingacademy/assets/releases/download/release/rockpaperscissors.zip
Resolving github.com (github.com)... 140.82.112.4
Connecting to github.com (github.com)|140.82.112.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://github-releases.githubusercontent.com/389502449/e56456b9-62a2-4c27-9318-e26876f42b7d?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F20210728%2Fus-ea
--2021-07-28 04:58:20--  https://github-releases.githubusercontent.com/389502449/e56456b9-62a2-4c27-9318-e26876f42b7d?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F2
Resolving github-releases.githubusercontent.com (github-releases.githubusercontent.com)... 185.199.108.154, 185.199.109.154, 185.199.110.154, ...
```

August, 2021

```
--2021-07-28 04:58:20--  https://github-releases.githubusercontent.com/389502449/e56456b9-62a2-4c27-9318-e26876f42b7d?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A%2F2
Resolving github-releases.githubusercontent.com (github-releases.githubusercontent.com)... 185.199.108.154, 185.199.109.154, 185.199.110.154, ...
Connecting to github-releases.githubusercontent.com (github-releases.githubusercontent.com)|185.199.108.154|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 322873683 (308M) [application/octet-stream]
Saving to: 'rockpaperscissors.zip.1'

rockpaperscissors.z 100%[===================>] 307.92M   158MB/s    in 2.0s

2021-07-28 04:58:22 (158 MB/s) - 'rockpaperscissors.zip.1' saved [322873683/322873683]
```

Mengekstrak file zip

```
import zipfile
import os
```

```
local_zip = '/content/rockpaperscissors.zip.'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/content')
zip_ref.close()
```

Melakukan Augmentasi & Pemisahan Data

Membuat folder

```
folder_kertas = os.path.join('/content/rockpaperscissors/paper')
folder_batu= os.path.join('/content/rockpaperscissors/rock')
folder_gunting = os.path.join('/content/rockpaperscissors/scissors')
```

```
os.listdir("/content/rockpaperscissors/rps-cv-images/train")
os.listdir("/content/rockpaperscissors/rps-cv-images/val")
```

```
['rock', 'scissors', 'paper']
```

```
# membuat direktori kertas pada direktori data training
train_kertas = os.path.join(train_dir, "kertas")

# membuat direktori batu pada direktori data training
train_batu = os.path.join(train_dir, "batu")

# membuat direktori gunting pada direktori data training
train_gunting = os.path.join(train_dir, "gunting")

# membuat direktori kertas pada direktori data validasi
validation_kertas = os.path.join(validation_dir, "kertas")

# membuat direktori batu pada direktori data validasi
validation_batu = os.path.join(validation_dir, "batu")

# membuat direktori gunting pada direktori data validasi
validation_gunting = os.path.join(validation_dir, "gunting")
```

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

test_datagen = ImageDataGenerator(
```

```
folder_batu= os.path.join('/content/rockpaperscissors/rock')
folder_gunting = os.path.join('/content/rockpaperscissors/scissors')

print('Jumlah gambar kertas:', len(os.listdir(folder_kertas)))
print('Jumlah gambar batu:', len(os.listdir(folder_batu)))
print('Jumlah gambar gunting:', len(os.listdir(folder_gunting)))
```

```
Jumlah gambar kertas: 712
Jumlah gambar batu: 726
Jumlah gambar gunting: 750
```

Menampilkan nama file

```
file_kertas = os.listdir(folder_kertas)
print("Kertas:", file_kertas[:3])

file_batu = os.listdir(folder_batu)
print("Batu:", file_batu[:3])

file_gunting = os.listdir(folder_gunting)
print("Gunting:",file_gunting[:3])
```

```
Kertas: ['RFXW2X1DlH7xboth.png', 'aJeRqPY3UVBErDTY.png', 'HuunhhoTMXTrST2Z.png']
Batu: ['Shr7grl23ISlh4Yu.png', 'goDBIj8kMSQbpQZT.png', 'Khsb4fYDiZjyXfBP.png']
Gunting: ['pjOgxx8uZf2XPDFu.png', 'xPDNBvgWZrdijzTm.png', 'dpmWj9s5ily8GuyY.png']
```

Mengelompokkan data

```
base_dir = "/content/rockpaperscissors/rps-cv-images/"
train_dir = os.path.join(base_dir, "train")
validation_dir = os.path.join(base_dir, "val")
```

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

```
train_generator = train_datagen.flow_from_directory(
        train_dir,  # direktori data latih
        target_size=(150, 150),  # mengubah resolusi seluruh gambar menjadi 150x150 piksel
        batch_size=4,
        # karena ini merupakan masalah klasifikasi 3 kelas maka menggunakan class_mode = 'categorical'
        class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
        validation_dir, # direktori data validasi
        target_size=(150, 150), # mengubah resolusi seluruh gambar menjadi 150x150 piksel
        batch_size=4, # karena ini merupakan masalah klasifikasi 3 kelas maka menggunakan class_mode = 'categorical'
        class_mode='categorical')
```

```
Found 1314 images belonging to 3 classes.
Found 874 images belonging to 3 classes.
```

Membangun model squential

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
```

```
[ ]     tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
        tf.keras.layers.MaxPooling2D(2,2),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(512, activation='relu'),
        tf.keras.layers.Dense(3, activation='softmax')
    ])

    model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 148, 148, 16)      448
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 74, 74, 16)        0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 72, 72, 32)        4640
_________________________________________________________________
max_pooling2d_13 (MaxPooling (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 34, 34, 32)        9248
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 17, 17, 32)        0
_________________________________________________________________
flatten_4 (Flatten)          (None, 9248)              0
_________________________________________________________________
dense_12 (Dense)             (None, 128)               1183872
_________________________________________________________________
dense_13 (Dense)             (None, 512)               66048
_________________________________________________________________
dense_14 (Dense)             (None, 3)                 1539
=================================================================
Total params: 1,265,795
Trainable params: 1,265,795
Non-trainable params: 0
```

```
Epoch 8/25
25/25 - 3s - loss: 0.8035 - accuracy: 0.5900 - val_loss: 0.4993 - val_accuracy: 0.8000
Epoch 9/25
25/25 - 3s - loss: 0.6916 - accuracy: 0.7100 - val_loss: 0.4634 - val_accuracy: 0.8000
Epoch 10/25
25/25 - 3s - loss: 0.4375 - accuracy: 0.8300 - val_loss: 0.4894 - val_accuracy: 0.8000
Epoch 11/25
25/25 - 3s - loss: 0.3574 - accuracy: 0.8900 - val_loss: 0.1804 - val_accuracy: 1.0000
Epoch 12/25
25/25 - 3s - loss: 0.4304 - accuracy: 0.8400 - val_loss: 0.6513 - val_accuracy: 0.6500
Epoch 13/25
25/25 - 3s - loss: 0.4091 - accuracy: 0.8300 - val_loss: 0.4847 - val_accuracy: 0.8500
Epoch 14/25
25/25 - 3s - loss: 0.4589 - accuracy: 0.8200 - val_loss: 1.0738 - val_accuracy: 0.6000
Epoch 15/25
25/25 - 3s - loss: 0.5101 - accuracy: 0.8200 - val_loss: 0.7095 - val_accuracy: 0.8000
Epoch 16/25
25/25 - 3s - loss: 0.3397 - accuracy: 0.8800 - val_loss: 0.3620 - val_accuracy: 0.8000
Epoch 17/25
25/25 - 3s - loss: 0.3242 - accuracy: 0.8500 - val_loss: 0.6020 - val_accuracy: 0.8000
Epoch 18/25
25/25 - 3s - loss: 0.4697 - accuracy: 0.8700 - val_loss: 0.9349 - val_accuracy: 0.6500
Epoch 19/25
25/25 - 3s - loss: 0.3596 - accuracy: 0.8600 - val_loss: 0.7038 - val_accuracy: 0.9000
Epoch 20/25
25/25 - 3s - loss: 0.3279 - accuracy: 0.8600 - val_loss: 0.4830 - val_accuracy: 0.9000
Epoch 21/25
25/25 - 3s - loss: 0.1874 - accuracy: 0.9286 - val_loss: 0.6608 - val_accuracy: 0.8500
Epoch 22/25
25/25 - 3s - loss: 0.2862 - accuracy: 0.8900 - val_loss: 0.7463 - val_accuracy: 0.8500
Epoch 23/25
25/25 - 3s - loss: 0.3023 - accuracy: 0.9100 - val_loss: 0.5648 - val_accuracy: 0.9000
Epoch 24/25
25/25 - 3s - loss: 0.1870 - accuracy: 0.9500 - val_loss: 0.9981 - val_accuracy: 0.6500
Epoch 25/25
25/25 - 3s - loss: 0.3465 - accuracy: 0.8800 - val_loss: 0.0840 - val_accuracy: 1.0000
```

```
[ ] import numpy as np
    from google.colab import files
```

```
Non-trainable params: 0
_________________________________________________________________
```

```
[ ] model.compile(loss='categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
```

Melatih Model

```
[ ] history = model.fit(
          train_generator,
          steps_per_epoch=25,
          epochs=25,
          validation_data=validation_generator,
          validation_steps=5,
          verbose=2)
```

```
Epoch 1/25
25/25 - 4s - loss: 1.1223 - accuracy: 0.4200 - val_loss: 1.1010 - val_accuracy: 0.2500
Epoch 2/25
25/25 - 3s - loss: 1.0956 - accuracy: 0.3400 - val_loss: 1.1047 - val_accuracy: 0.4500
Epoch 3/25
25/25 - 3s - loss: 1.0990 - accuracy: 0.3300 - val_loss: 1.1088 - val_accuracy: 0.2500
Epoch 4/25
25/25 - 3s - loss: 1.0874 - accuracy: 0.3800 - val_loss: 1.0844 - val_accuracy: 0.3500
Epoch 5/25
25/25 - 3s - loss: 1.0578 - accuracy: 0.4000 - val_loss: 1.0258 - val_accuracy: 0.4500
Epoch 6/25
25/25 - 3s - loss: 0.9288 - accuracy: 0.5400 - val_loss: 0.5230 - val_accuracy: 0.8500
Epoch 7/25
25/25 - 3s - loss: 0.7408 - accuracy: 0.6500 - val_loss: 0.7552 - val_accuracy: 0.6000
Epoch 8/25
25/25 - 3s - loss: 0.8035 - accuracy: 0.5900 - val_loss: 0.4993 - val_accuracy: 0.8000
```

```
[ ] import numpy as np
    from google.colab import files
    from keras.preprocessing import image
    import matplotlib.pyplot as plt
    import matplotlib.image as mpimg
    import matplotlib.pyplot as plt
    import matplotlib.image as mpimg
    %matplotlib inline

    uploaded = files.upload()

    for fn in uploaded.keys():

      # predicting images
      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]==1:
        print('paper')
      elif classes[0][1]==1:
        print('rock')
      else:
        print('scissors')
```

```
Choose Files  No file chosen    Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving 0ePX1wuCc3et7leL.png to 0ePX1wuCc3et7leL.png
0ePX1wuCc3et7leL.png
scissors
```

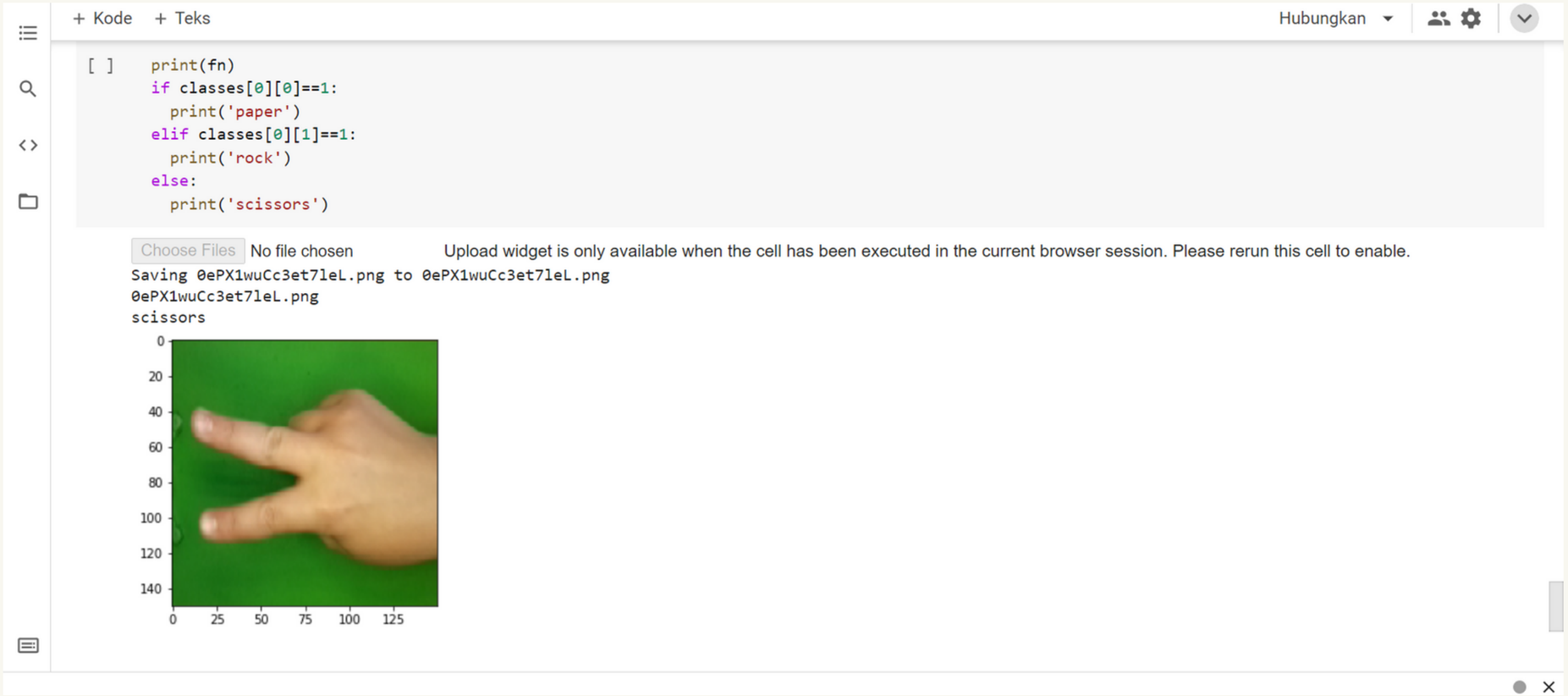
August, 2021

RESULT

The prediction results of the model are shown in the image on the side.

After inputting the test image, the program can predict the image correctly with an accuracy above

95%*



*based on my review rating

August, 2021

THANK YOU

dian.maulida103@gmail.com

Pictures & design by Canva