Proyek_Klasifikasi_Image_from_GDSC_by Isa Aulia Almadani

Project Overview:

Proyek ini bertujuan untuk membangun model klasifikasi gambar yang dapat mengenali gerakan tangan batu, kertas, dan gunting menggunakan jaringan neural dengan TensorFlow. Proyek ini menggunakan dataset gambar yang diambil dari GitHub dan mencakup tahapan mulai dari impor pustaka, preprocessing data, membangun model, hingga prediksi gambar.

Tujuan Proyek:

Membuat model yang dapat mengklasifikasikan gerakan tangan batu, kertas, dan gunting dengan akurasi tinggi untuk aplikasi dalam permainan atau interaksi manusia-komputer.

Fitur Utama:

1. **Import Library:** Mengimpor pustaka yang diperlukan untuk analisis dan pembangunan model seperti TensorFlow, NumPy, Pandas, dan Matplotlib.
2. **Import Dataset from GitHub:** Mengambil dataset gambar gerakan tangan dari GitHub yang berisi gambar-gambar batu, kertas, dan gunting dalam berbagai kondisi pencahayaan dan latar belakang.
3. **Preprocessing:** Melakukan preprocessing pada dataset, termasuk mengubah ukuran gambar, normalisasi nilai piksel, dan pembagian data menjadi set pelatihan dan pengujian.
4. **Building Model with TensorFlow:** Membuat dan melatih model jaringan neural menggunakan TensorFlow. Model ini dirancang untuk mengenali pola dalam gambar gerakan tangan dan mengklasifikasikannya ke dalam salah satu dari tiga kategori: batu, kertas, atau gunting.
5. **Predict Image:** Menggunakan model yang telah dilatih untuk memprediksi klasifikasi gambar baru yang belum pernah dilihat sebelumnya, dan menampilkan hasil prediksinya.

Teknologi yang Digunakan:

- TensorFlow: Untuk membangun dan melatih model jaringan neural.
- NumPy: Untuk manipulasi array dan operasi numerik.
- Pandas: Untuk manipulasi data dan analisis dataset.
- Matplotlib: Untuk visualisasi data dan hasil prediksi.

Hasil yang Diharapkan:

Proyek ini diharapkan menghasilkan model klasifikasi gambar yang dapat secara akurat mengenali dan mengklasifikasikan gerakan tangan batu, kertas, dan gunting. Model ini diharapkan dapat digunakan dalam aplikasi interaktif atau permainan yang memerlukan pengenalan gerakan tangan.

Rencana Pengembangan:

1. Mengoptimalkan model untuk meningkatkan akurasi prediksi.
2. Menggunakan Ide model ini untuk project yang lebih besar dengan data yang sama, yaitu image.

Nama: Isa Aulia Almadani
Domisili: SUkoharjo

```
import zipfile

# Unduh dataset
!wget --no-check-certificate https://github.com/dicodingacademy/assets/releases/download/release/rockpaperscissors.zip

# Ekstrak dataset
with zipfile.ZipFile("rockpaperscissors.zip", "r") as zip_ref:
    zip_ref.extractall(".")
```

```
--2023-10-31 09:23:16--  https://github.com/dicodingacademy/assets/releases/download/release/rockpaperscissors.zip
Resolving github.com (github.com)... 140.82.114.4
Connecting to github.com (github.com)|140.82.114.4|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/391417272/7eb836f2-695b-4a46-9c78-b658671669
--2023-10-31 09:23:16--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/391417272/7eb836f2-695b-4a46-
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 322873683 (308M) [application/octet-stream]
Saving to: 'rockpaperscissors.zip'
```

```
rockpaperscissors.z 100%[==================>] 307.92M   310MB/s    in 1.0s

2023-10-31 09:23:17 (310 MB/s) - ‘rockpaperscissors.zip’ saved [322873683/322873683]
```

```
#Melakukan import beberapa libraries
import cv2
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import to_categorical

# Import dataset
import os
data_dir = "rockpaperscissors"

# Load dataset
x_data = []
y_data = []

for label in ["scissors", "rock", "paper"]:
    label_dir = os.path.join(data_dir, label)
    for img_file in os.listdir(label_dir):
        img = cv2.imread(os.path.join(label_dir, img_file))
        x_data.append(img)
        y_data.append(label)

# Mengubah list (x_data) ke dalam format array NumPy
x_data = np.array(x_data)

# Untuk melatih model klasifikasi karena model memahami data dalam bentuk numerik/int.
label_encoder = LabelEncoder()
y_data = label_encoder.fit_transform(y_data)

# One-Hot Encoding, untuk mengubah bilangan bulat dari langkah diatas menajdi vektor biner.
y_data = to_categorical(y_data)

# memisahkan data set gambar dan label menjadi data pelatihan dan data validasi.
x_train, x_val, y_train, y_val = train_test_split(x_data, y_data, test_size=0.4, random_state=123)

```

```
#Pelatihan model jaringan saraf tiruan menggunakan TensorFlow dan Keras.
# Import library dari beberapa modul
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
# Menyertakan image data generator for training set
train_datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    # zoom_range = 0.2,
    # fill_mode = 'nearest',
    vertical_flip=True,
    # brightness_range=[0.7, 1.3],
    validation_split=0.2
)

# Menyertakan image data generator untuk validation set
valid_datagen = ImageDataGenerator(rescale=1.0 / 255)

# Fit image data generator to train set
train_datagen.fit(x_train)
valid_datagen.fit(x_val)

# Mendefinisikan model dan membangun model dengan Seruential.
image_height = 200
image_width = 300


model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(image_height, image_width, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(3, activation='softmax')
])

# compile the model.
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
# Melatih model.
history = model.fit(train_datagen.flow(x_train, y_train, batch_size=32),
                    validation_data=valid_datagen.flow(x_val, y_val, batch_size=32),
                    epochs=10)
#Mengevaluasi model.
evaluation = model.evaluate(valid_datagen.flow(x_val, y_val, batch_size=32))
print("Validation Accuracy: {:.2f}%".format(evaluation[1] * 100))
```

```
Epoch 1/10
41/41 [==============================] - 33s 490ms/step - loss: 1.3695 - accuracy: 0.3422 - val_loss: 1.0906 - val_accuracy: 0.3356
Epoch 2/10
41/41 [==============================] - 18s 446ms/step - loss: 1.0938 - accuracy: 0.3796 - val_loss: 1.0325 - val_accuracy: 0.3836
Epoch 3/10
41/41 [==============================] - 19s 474ms/step - loss: 1.1018 - accuracy: 0.3971 - val_loss: 1.0478 - val_accuracy: 0.3973
Epoch 4/10
41/41 [==============================] - 18s 450ms/step - loss: 1.0473 - accuracy: 0.4512 - val_loss: 1.3435 - val_accuracy: 0.3379
Epoch 5/10
41/41 [==============================] - 19s 451ms/step - loss: 0.9534 - accuracy: 0.5534 - val_loss: 0.6002 - val_accuracy: 0.8584
Epoch 6/10
41/41 [==============================] - 19s 474ms/step - loss: 0.6737 - accuracy: 0.7584 - val_loss: 0.2296 - val_accuracy: 0.9635
Epoch 7/10
41/41 [==============================] - 18s 451ms/step - loss: 0.4495 - accuracy: 0.8445 - val_loss: 0.1466 - val_accuracy: 0.9475
Epoch 8/10
41/41 [==============================] - 20s 478ms/step - loss: 0.3733 - accuracy: 0.8704 - val_loss: 0.1014 - val_accuracy: 0.9772
Epoch 9/10
41/41 [==============================] - 18s 448ms/step - loss: 0.4896 - accuracy: 0.8628 - val_loss: 0.1116 - val_accuracy: 0.9703
Epoch 10/10
41/41 [==============================] - 20s 483ms/step - loss: 0.2608 - accuracy: 0.9055 - val_loss: 0.1039 - val_accuracy: 0.9726
28/28 [==============================] - 1s 29ms/step - loss: 0.1039 - accuracy: 0.9726
Validation Accuracy: 97.26%
```

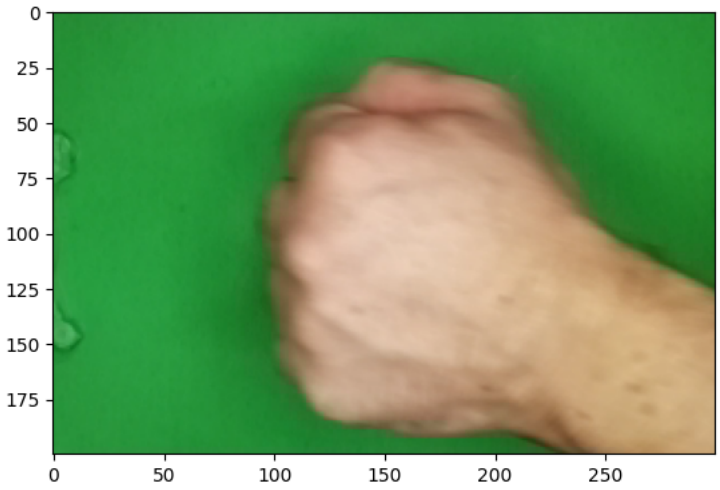
Klik dua kali (atau tekan Enter) untuk mengedit

Klik dua kali (atau tekan Enter) untuk mengedit

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

# Memuat gambar
uploaded = files.upload()

# Memuat setiap gambar ke variabel img
for fn in uploaded.keys():
    # predicting images
    path = fn
    img = image.load_img(path, target_size=(200, 300))

    # Menampilkan gambar di notebook dan mengonversi gambar di notebook
    imgplot = plt.imshow(img)
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    images = np.vstack([x])

    # Memprediksi kelas gambar menggunakan model Keras
    classes = model.predict(images, batch_size=10)
    output_class = np.argmax(classes)
    print(fn)

    if output_class == 0:
        print('paper')
    elif output_class == 1:
        print('rock')
    else:
        print('scissors')


```

Pilih File Tidak ada file yang dipilih Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving 2NmrcDGkc7FQuu12.png to 2NmrcDGkc7FQuu12.png
1/1 [==============================] - 0s 18ms/step
2NmrcDGkc7FQuu12.png
rock
```




```
# Analisis kesalahan model
predictions = model.predict(x_val)
true_labels = np.argmax(y_val, axis=1)
predicted_labels = np.argmax(predictions, axis=1)
misclassified_indices = np.where(true_labels != predicted_labels)[0]

plt.figure(figsize=(10, 5))
```