



IMAGE CLASSIFICATION

Derajat Salim Wibowo

KRITERIA PROGRAM

1. Dataset yang dipakai haruslah dataset berikut :
[rockpaperscissors](https://github.com/dicodingacademy/assets/releases/download/release/rockpaperscissors.zip), atau gunakan link ini pada wget command:
<https://github.com/dicodingacademy/assets/releases/download/release/rockpaperscissors.zip>
2. Dataset harus dibagi menjadi train set dan validation set.
3. Ukuran validation set harus 40% dari total dataset (data training memiliki 1314 sampel, dan data validasi sebanyak 874 sampel).
4. Harus mengimplementasikan augmentasi gambar.
5. Menggunakan image data generator.
6. Model harus menggunakan model sequential.
7. Pelatihan model tidak melebihi waktu 30 menit.
8. Program dikerjakan pada Google Colaboratory.
9. Akurasi dari model minimal 85%.
10. Dapat memprediksi gambar yang diunggah ke Colab

RINGKASAN PROGRAM

Program ini dibuat dalam rangka memenuhi tugas submission dicoding pada kelas machine learning pemula setelah mempelajari seluruh materinya. Program menggunakan tensorflow sebagai library untuk melatih model dan juga dataset rock,paper,scissor yang berasal dari dicoding. Program ini juga diharapkan dapat mencapai akurasi training data lebih dari 85%



PROGRAM

Modul-modul yang diimpor antara lain:

- **tensorflow** untuk menggunakan TensorFlow library.
- **RMSprop** dari **tensorflow.keras.optimizers** untuk mengimpor optimizer RMSprop yang dapat digunakan saat melatih model.
- **ImageDataGenerator** dari **tensorflow.keras.preprocessing.image** untuk memproses data gambar.
- **load_img** dan **img_to_array** dari **tensorflow.keras.utils** untuk membantu dalam memuat gambar dan mengubahnya menjadi larik numerik.
- **zipfile** dan **os** untuk memanipulasi file dan folder terkait dengan dataset.
- **splitfolders** untuk membagi dataset gambar menjadi set pelatihan dan validasi.
- **matplotlib.pyplot** dan **matplotlib.image** untuk melakukan visualisasi gambar dan grafik.

Selain impor modul, beberapa tugas yang dilakukan dalam kode ini adalah menginstal modul **split-folders**, mengatur agar gambar dan grafik ditampilkan di jupyter notebook secara langsung (**%matplotlib inline**), dan mengimpor beberapa fungsi bantu lainnya yang mungkin digunakan dalam pengolahan dan visualisasi gambar, seperti **image** dari **keras.preprocessing** dan **files** dari **google.colab**.

```
#Memanggil libraryyyyyyyyyyyyyyy
import tensorflow as tf
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import load_img, img_to_array
import zipfile,os
!pip install split-folders
import splitfolders
import matplotlib.pyplot as plt
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
```



```
#Dataset berasal dari instruksi dicoding  
!wget --no-check-certificate \  
https://github.com/dicodingacademy/assets/releases/download/release/rockpaperscissors.zip \  
-O /tmp/rockpaperscissors.zip
```

menggunakan perintah **wget** untuk mengunduh file dataset dari URL yang diberikan. File dataset ini berisi gambar-gambar tangan yang memegang batu, kertas, dan gunting, yang biasanya digunakan dalam permainan "rock-paper-scissors" (batu-gunting-kertas).

--no-check-certificate digunakan untuk mengabaikan pemeriksaan sertifikat SSL saat mengunduh file.

-O /tmp/rockpaperscissors.zip mengatur lokasi dan nama file tujuan di mana file zip akan disimpan. Dalam hal ini, file akan disimpan di direktori **/tmp** dengan nama **rockpaperscissors.zip**.

Setelah file zip diunduh, langkah-langkah berikut dijalankan:

1. `local_zip = '/tmp/rockpaperscissors.zip'`: Menyimpan path atau lokasi file zip dataset yang telah diunduh sebelumnya.
2. `zip_ref = zipfile.ZipFile(local_zip, 'r')`: Membuka file zip menggunakan objek `ZipFile` dari modul `zipfile` dengan mode "r" (read) untuk membaca isi file zip.
3. `zip_ref.extractall('/tmp')`: Menggunakan metode `extractall` pada objek `ZipFile` untuk mengekstrak semua file dan folder yang ada di dalam file zip ke direktori tujuan. Dalam hal ini, direktori tujuan adalah `/tmp`, yang merupakan direktori sementara di sistem file.
4. `zip_ref.close()`: Menutup objek `ZipFile` setelah proses ekstraksi selesai.
5. `base_dir = '/tmp/rockpaperscissors/rps-cv-images'`: Menyimpan path ke direktori tempat dataset diekstrak. Dalam hal ini, direktori tersebut adalah `/tmp/rockpaperscissors/rps-cv-images`.
6. `os.listdir(base_dir)`: Menggunakan modul `os` untuk mendapatkan daftar file dan folder yang ada di dalam direktori `base_dir`. Baris ini akan menghasilkan daftar file yang ada dalam direktori tersebut.

```
#Mengekstrak Dataset
local_zip = '/tmp/rockpaperscissors.zip'
zip_ref = zipfile.ZipFile(local_zip, 'r')
zip_ref.extractall('/tmp')
zip_ref.close()
base_dir = '/tmp/rockpaperscissors/rps-cv-images'
os.listdir(base_dir)

['paper', 'scissors', 'rock', 'README_rpc-cv-images.txt']
```

1. `class Callback(tf.keras.callbacks.Callback):` Kode ini mendefinisikan sebuah kelas bernama "Callback" yang merupakan subclass dari "tf.keras.callbacks.Callback". Dalam kelas ini, terdapat sebuah metode `on_epoch_end` yang akan dipanggil pada akhir setiap epoch saat melatih model. Pada metode ini, dilakukan pengecekan apakah akurasi (accuracy) yang dicatat dalam log pelatihan (`logs.get('accuracy')`) telah mencapai 97%. Jika iya, maka pesan "Traning berhenti karena akurasi di atas 97%!" akan dicetak, dan `self.model.stop_training` akan diatur menjadi True untuk menghentikan pelatihan model.

2. `splitfolders.ratio(base_dir, seed=1337, ratio=(.6, .4), group_prefix=None):` Kode ini menggunakan fungsi `ratio` dari modul `splitfolders` untuk membagi dataset menjadi set pelatihan dan validasi. Parameter `base_dir` menyimpan path ke direktori tempat dataset telah diekstrak sebelumnya. Parameter `seed` digunakan untuk mengatur seed acak agar pembagian dataset konsisten. Parameter `ratio` adalah tuple yang menentukan perbandingan pembagian dataset antara set pelatihan dan validasi. Dalam hal ini, dataset akan dibagi dengan perbandingan 60% untuk set pelatihan dan 40% untuk set validasi. Parameter `group_prefix` tidak digunakan dalam kode ini.

```
#Membuat class supaya train berhenti ketika mencapai jumlah target
class Callback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('accuracy') > 0.97):
            print("\n Traning berhenti karena akurasi diatas 97%!")
            self.model.stop_training = True
callbacks = Callback()
```

```
#Membagi dataset menjadi train dan validation
splitfolders.ratio(base_dir,seed=1337, ratio=(.6, .4), group_prefix=None)
```


```
Copying files: 2188 files [00:00, 4171.39 files/s]
```


Kode yang Anda berikan adalah bagian untuk membuat objek `train_datagen` yang menggunakan `ImageDataGenerator` dari TensorFlow untuk melakukan augmentasi gambar pada set pelatihan.

Augmentasi gambar adalah teknik yang digunakan untuk memperluas variasi data pelatihan dengan melakukan transformasi terkontrol pada gambar-gambar pelatihan. Ini membantu mencegah overfitting dan meningkatkan kemampuan generalisasi model.

Pada objek `train_datagen`, beberapa parameter yang digunakan adalah:

- `rescale=1./255`: Mengubah rentang intensitas piksel gambar menjadi antara 0 dan 1 dengan membagi setiap nilai piksel dengan 255. Hal ini dilakukan untuk normalisasi data.
- `rotation_range=20`: Mengacak rotasi gambar hingga 20 derajat.
- `horizontal_flip=True`: Mengacak secara horizontal (membalikkan) gambar.
- `shear_range=0.2`: Mengacak gambar dengan efek shear sebesar 0.2.
- `fill_mode='nearest'`: Mengisi piksel yang kosong setelah perubahan geometri dengan piksel terdekat.
- `validation_split=0.4`: Menggunakan 40% dari data pelatihan sebagai data validasi.



```
#Membuat augmentasi gambar
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    horizontal_flip=True,
    shear_range = 0.2,
    fill_mode = 'nearest',
    validation_split=0.4)
```

[8]

membuat data generator (generator data) untuk set pelatihan dan validasi menggunakan objek train_datagen yang telah dibuat sebelumnya.

1. `train_generator = train_datagen.flow_from_directory(base_dir, target_size=(150, 150), shuffle=True, subset='training')`: Kode ini membuat generator data untuk set pelatihan. Fungsi `flow_from_directory()` dari objek `train_datagen` digunakan untuk menghasilkan batch data gambar secara otomatis dari direktori yang ditentukan (`base_dir`). Parameter `target_size=(150, 150)` menentukan ukuran gambar target yang diharapkan. Parameter `shuffle=True` mengacak urutan data dalam set pelatihan setiap epoch. Parameter `subset='training'` menunjukkan bahwa generator ini digunakan untuk set pelatihan.

2. `validation_generator = train_datagen.flow_from_directory(base_dir, target_size=(150, 150), shuffle=True, subset='validation')`: Kode ini membuat generator data untuk set validasi. Fungsi `flow_from_directory()` kembali digunakan untuk menghasilkan batch data gambar dari direktori `base_dir`. Parameter `target_size=(150, 150)` menentukan ukuran gambar target yang diharapkan. Parameter `shuffle=True` mengacak urutan data dalam set validasi setiap epoch. Parameter `subset='validation'` menunjukkan bahwa generator ini digunakan untuk set validasi.

```
#Membuat data generator Train
train_generator = train_datagen.flow_from_directory(
    base_dir,
    target_size=(150, 150),
    shuffle=True,
    subset='training')
```

Found 1314 images belonging to 3 classes.


```
#Membuat data generator Validation
validation_generator = train_datagen.flow_from_directory(
    base_dir,
    target_size=(150, 150),
    shuffle=True,
    subset='validation')
```

Found 874 images belonging to 3 classes.

Model ini adalah model yang berurutan (sequential) di mana lapisan-lapisan ditempatkan satu per satu dalam urutan yang diinginkan.

Lapisan-lapisan yang digunakan dalam model ini adalah:

1. Conv2D(32, (3,3), activation='relu', input_shape=(150, 150, 3)): Lapisan konvolusi dengan 32 filter, masing-masing berukuran 3x3, dan fungsi aktivasi ReLU. Ini adalah lapisan input pertama yang juga menentukan bentuk input yang diharapkan, yaitu gambar berukuran 150x150 piksel dengan 3 saluran warna (RGB).
2. MaxPooling2D(2, 2): Lapisan pemaksimalan (max pooling) dengan jendela ukuran 2x2. Ini membantu dalam mengurangi dimensi spasial gambar.
3. Lapisan konvolusi dan pemaksimalan yang sama diulang dua kali lagi untuk mendapatkan fitur yang lebih kompleks.
4. Flatten(): Lapisan yang mengubah vektor multidimensi menjadi vektor satu dimensi. Ini digunakan untuk menghubungkan lapisan-lapisan konvolusi dengan lapisan-lapisan selanjutnya yang sepenuhnya terhubung.
5. Dense(512, activation='relu'): Lapisan terhubung sepenuhnya dengan 512 unit neuron dan fungsi aktivasi ReLU. Ini membantu dalam mempelajari representasi yang lebih abstrak dari fitur-fitur yang diekstraksi sebelumnya.
6. Dense(3, activation='softmax'): Lapisan terhubung sepenuhnya terakhir dengan 3 unit neuron dan fungsi aktivasi softmax. Ini menghasilkan output dalam bentuk probabilitas untuk klasifikasi multi-kelas, di mana setiap unit neuron mewakili probabilitas kelas yang berbeda.



```
#Membuat model sequential
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(32, (3,3), activation='relu'),
    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.Dense(512, activation='relu'),
    tf.keras.layers.Dense(3, activation='softmax')
])
```

11]


```
model.summary()
```

```
2]
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 72, 72, 32)	9248
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 32)	0
conv2d_2 (Conv2D)	(None, 34, 34, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 64)	0
conv2d_3 (Conv2D)	(None, 15, 15, 128)	73856
max_pooling2d_3 (MaxPooling2D)	(None, 7, 7, 128)	0
flatten (Flatten)	(None, 6272)	0
...		
Total params:	3,315,811	
Trainable params:	3,315,811	
Non-trainable params:	0	


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

```
14] #Membuat pelatihan model  
train = model.fit(  
train_generator,  
steps_per_epoch=23,  
epochs=20,  
validation_data=validation_generator,  
validation_steps=6,  
verbose=2,  
callbacks=[callbacks])
```

1. `model.compile(loss='categorical_crossentropy', optimizer=tf.optimizers.Adam(), metrics=['accuracy'])`: Kode ini mengompilasi model dengan mengatur fungsi loss, optimizer, dan metrik evaluasi yang akan digunakan saat melatih model. Di sini, kita menggunakan loss function 'categorical_crossentropy' untuk masalah klasifikasi multi-kelas. Optimizer yang digunakan adalah Adam optimizer melalui `tf.optimizers.Adam()`. Metrik evaluasi yang digunakan adalah akurasi (accuracy).
 2. `train = model.fit(train_generator, steps_per_epoch=23, epochs=20, validation_data=validation_generator, validation_steps=6, verbose=2, callbacks=[callbacks])`: Kode ini melakukan pelatihan model menggunakan metode `fit()`. Parameter `train_generator` digunakan sebagai data pelatihan, `steps_per_epoch` menentukan jumlah batch yang akan diproses dalam setiap epoch, `epochs` menentukan jumlah epoch pelatihan yang akan dilakukan, `validation_data` digunakan sebagai data validasi, `validation_steps` menentukan jumlah batch yang akan diproses dalam setiap validasi, `verbose` menentukan tingkat detail keluaran yang ditampilkan selama pelatihan, dan `callbacks` adalah daftar callback yang akan dipanggil selama pelatihan (dalam hal ini, callback `callbacks` yang telah didefinisikan sebelumnya untuk menghentikan pelatihan ketika mencapai akurasi di atas 97%).
- Selama pelatihan, model akan melalui beberapa epoch dan menggunakan data pelatihan dan validasi yang dihasilkan oleh generator data. Metode `fit()` akan mengoptimalkan model berdasarkan fungsi loss yang didefinisikan sebelumnya menggunakan optimizer Adam, dan menghasilkan metrik evaluasi akurasi untuk setiap epoch.

```
Epoch 1/20
23/23 - 20s - loss: 1.0850 - accuracy: 0.4745 - val_loss: 0.8909 - val_accuracy: 0.5729 - 20s/epoch - 886ms/step
Epoch 2/20
23/23 - 21s - loss: 0.5592 - accuracy: 0.7962 - val_loss: 0.4307 - val_accuracy: 0.8125 - 21s/epoch - 892ms/step
Epoch 3/20
23/23 - 20s - loss: 0.3283 - accuracy: 0.8696 - val_loss: 0.3864 - val_accuracy: 0.8854 - 20s/epoch - 884ms/step
Epoch 4/20
23/23 - 19s - loss: 0.2471 - accuracy: 0.9136 - val_loss: 0.1975 - val_accuracy: 0.9271 - 19s/epoch - 820ms/step
Epoch 5/20
23/23 - 20s - loss: 0.2076 - accuracy: 0.9375 - val_loss: 0.4729 - val_accuracy: 0.8542 - 20s/epoch - 880ms/step
Epoch 6/20
23/23 - 19s - loss: 0.2669 - accuracy: 0.9079 - val_loss: 0.1827 - val_accuracy: 0.9688 - 19s/epoch - 825ms/step
Epoch 7/20
23/23 - 20s - loss: 0.1640 - accuracy: 0.9504 - val_loss: 0.1793 - val_accuracy: 0.9271 - 20s/epoch - 850ms/step
Epoch 8/20
23/23 - 20s - loss: 0.1554 - accuracy: 0.9391 - val_loss: 0.3007 - val_accuracy: 0.9115 - 20s/epoch - 858ms/step
Epoch 9/20
23/23 - 22s - loss: 0.1213 - accuracy: 0.9533 - val_loss: 0.3363 - val_accuracy: 0.9062 - 22s/epoch - 961ms/step
Epoch 10/20
23/23 - 20s - loss: 0.1437 - accuracy: 0.9561 - val_loss: 0.1645 - val_accuracy: 0.9375 - 20s/epoch - 848ms/step
Epoch 11/20
23/23 - 21s - loss: 0.0922 - accuracy: 0.9647 - val_loss: 0.1581 - val_accuracy: 0.9583 - 21s/epoch - 924ms/step
Epoch 12/20
23/23 - 22s - loss: 0.1302 - accuracy: 0.9547 - val_loss: 0.1509 - val_accuracy: 0.9427 - 22s/epoch - 961ms/step
Epoch 13/20

Traning berhenti karena akurasi diatas 97%!
23/23 - 20s - loss: 0.0880 - accuracy: 0.9717 - val_loss: 0.0879 - val_accuracy: 0.9635 - 20s/epoch - 850ms/step
```

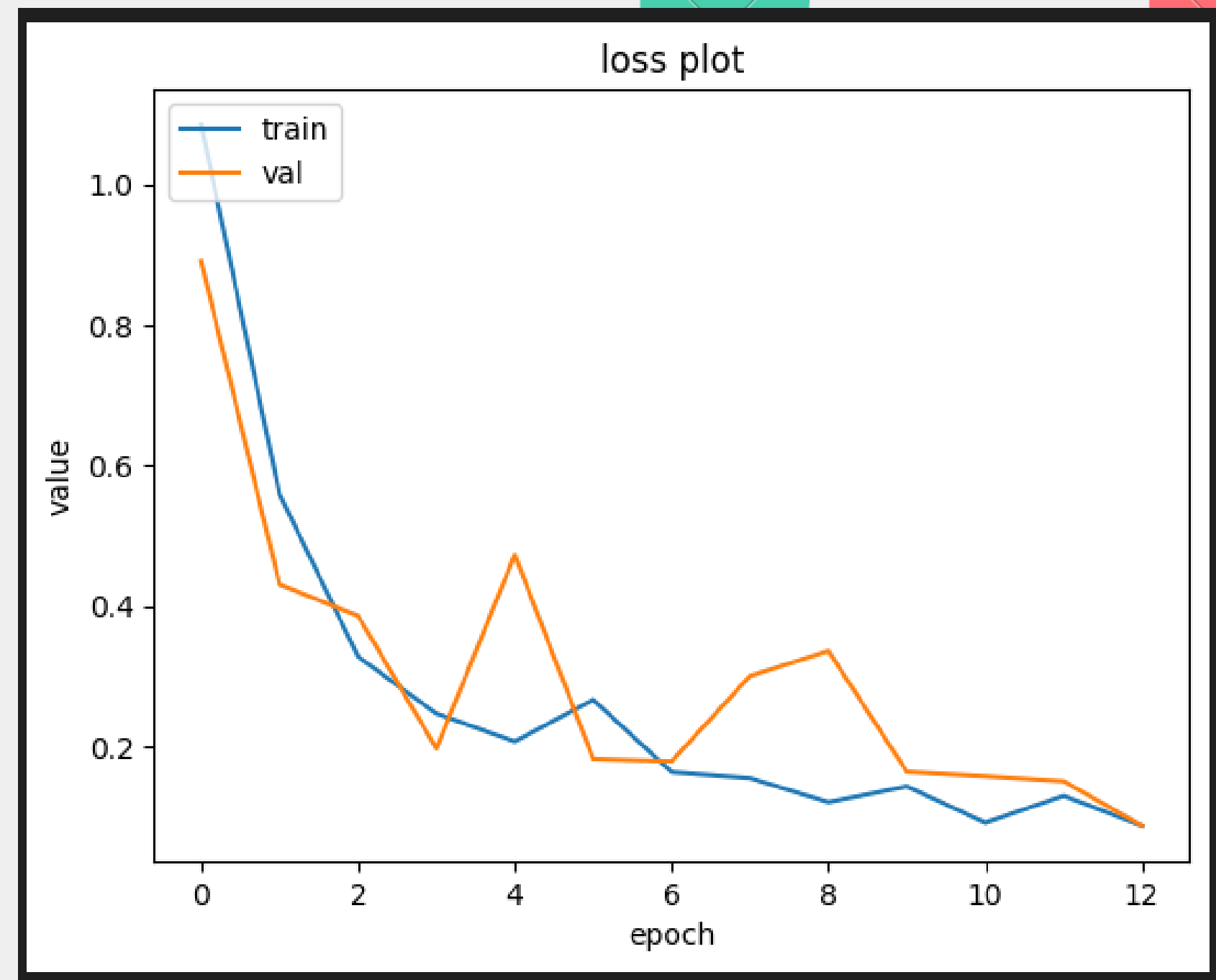
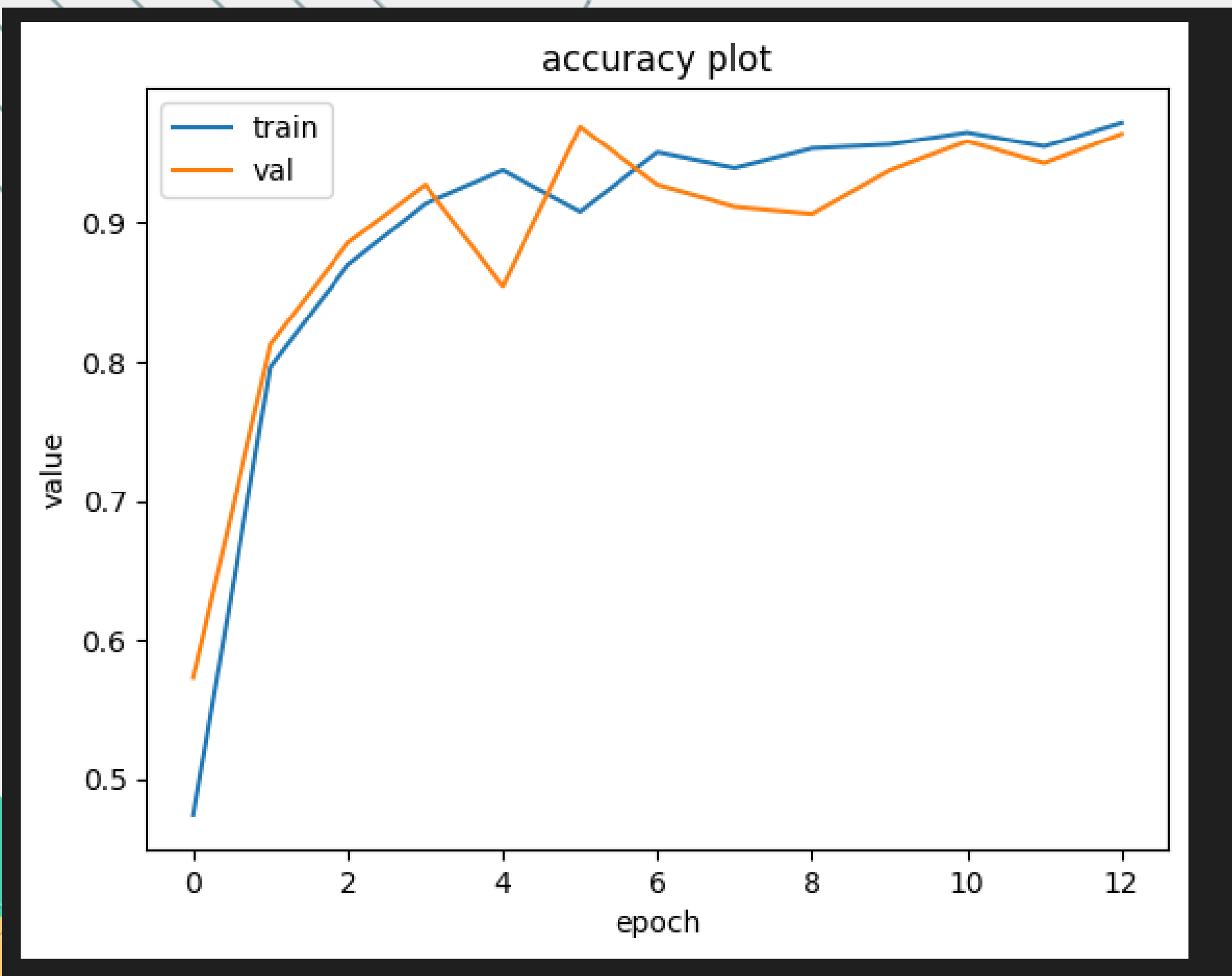
```
#Membuat diagram garis untuk melihat history model
plt.plot(train.history['accuracy'])
plt.plot(train.history['val_accuracy'])
plt.title('accuracy plot')
plt.ylabel('value')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()

plt.plot(train.history['loss'])
plt.plot(train.history['val_loss'])
plt.title('loss plot')
plt.ylabel('value')
plt.xlabel('epoch')
plt.legend(['train', 'val'], loc='upper left')
plt.show()
```

15]

membuat dua diagram garis yang menunjukkan perubahan akurasi dan loss selama pelatihan model.

1. Diagram garis pertama menampilkan perubahan akurasi model pada setiap epoch. Garis biru mewakili akurasi pelatihan, sementara garis jingga mewakili akurasi validasi. Diagram ini membantu melihat seberapa baik model Anda belajar dari data pelatihan dan seberapa baik performanya pada data yang belum pernah dilihat sebelumnya. Anda dapat melihat apakah akurasi meningkat seiring berjalannya epoch atau jika ada overfitting atau underfitting yang terjadi.
2. Diagram garis kedua menampilkan perubahan loss model pada setiap epoch. Garis biru mewakili loss pelatihan, sementara garis jingga mewakili loss validasi. Loss mengukur seberapa baik model Anda memetakan input ke output yang diharapkan. Dalam kasus ini, loss berarti perbedaan antara prediksi model dan label yang sebenarnya. Anda dapat melihat apakah loss berkurang seiring berjalannya epoch, yang menunjukkan model yang semakin baik dalam mempelajari pola-pola dalam data.



untuk memprediksi gambar yang diunggah menggunakan model yang telah dilatih sebelumnya.

1. Pengguna diizinkan untuk mengunggah gambar yang akan diprediksi.
2. Gambar yang diunggah diubah menjadi format yang dapat diterima oleh model, yaitu array numpy dengan ukuran 150x150 piksel.
3. Model digunakan untuk memprediksi kelas gambar yang diunggah. Prediksi dilakukan dengan memasukkan gambar ke dalam model menggunakan metode predict().
4. Hasil prediksi ditampilkan dengan mencetak nama file gambar yang diunggah, diikuti dengan kelas yang diprediksi (antara "paper", "rock", atau "scissors").

Dengan menggunakan kode ini, pengguna dapat mengunggah gambar dan model akan memberikan prediksi kelas gambar tersebut berdasarkan apa yang telah dipelajari selama pelatihan.

```
#Memprediksi Gambar
uploaded = files.upload()
for fn in uploaded.keys():

    path = fn
    img = tf.keras.utils.load_img(path, target_size=(150,150))
    imgplot = plt.imshow(img)
    x = tf.keras.utils.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')
```

Saving 6HsTnF20TAnDZiRT.png to 6HsTnF20TAnDZiRT.png
1/1 [=====] - 0s 172ms/step
6HsTnF20TAnDZiRT.png
rock

