```
!pip install -q kaggle
!mkdir -p ~/.kaggle
!echo '{"username":"abdulrofi","key":"e05b61e9f371002d45f8283924526695"}' > ~/.kaggle/kaggle.json
!chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -d alxmamaev/flowers-recognition
```

```
Downloading flowers-recognition.zip to /content
 97% 219M/225M [00:01<00:00, 149MB/s]
100% 225M/225M [00:01<00:00, 153MB/s]
```

```
import zipfile

path_to_zip_file = "/content/flowers-recognition.zip"
directory_to_extract_to = "/content/flowers"  # Ganti dengan path folder tujuan ekstraksi

with zipfile.ZipFile(path_to_zip_file, 'r') as zip_ref:
    zip_ref.extractall(directory_to_extract_to)
```

```
import os
base_dir = '/content/flowers/flowers'
```

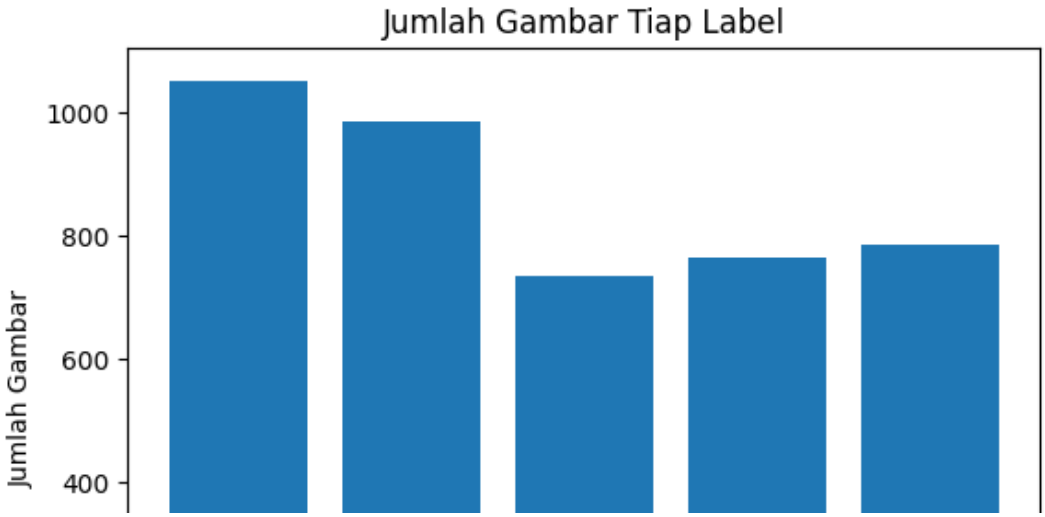
```
print(os.listdir(base_dir))
```

```
['dandelion', 'tulip', 'sunflower', 'daisy', 'rose']
```

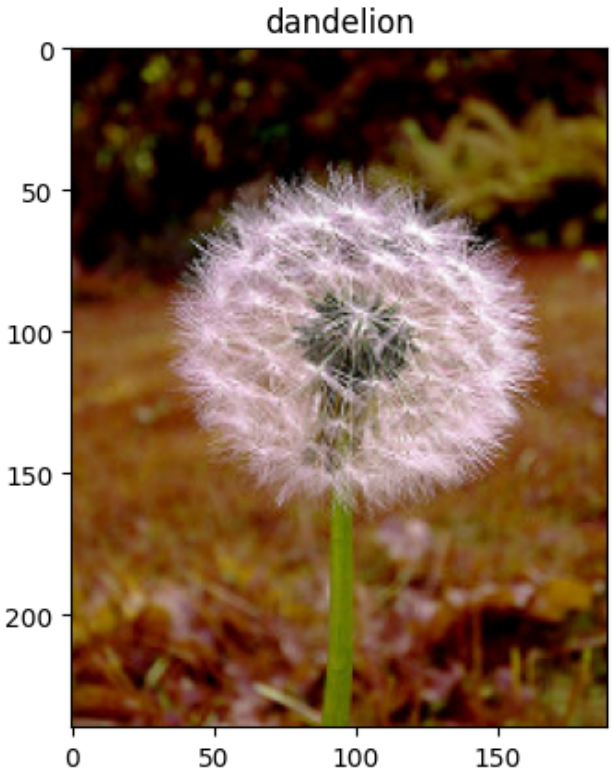
```
# Menghitung jumlah gambar pada dataset
number_label = {}
total_files = 0
for i in os.listdir(base_dir):
    counting = len(os.listdir(os.path.join(base_dir, i)))
    number_label[i] = counting
    total_files += counting

print("Total Files : " + str(total_files))
```

```
Total Files : 4317
```

```
# Visualisasi jumlah gambar tiap kelas
import matplotlib.pyplot as plt

plt.bar(number_label.keys(), number_label.values());
plt.title("Jumlah Gambar Tiap Label");
plt.xlabel('Label');
plt.ylabel('Jumlah Gambar');
```




```
# Menampilkan sampel gambar tiap kelas
import matplotlib.image as mpimg

img_each_class = 1
img_samples = {}
classes = list(number_label.keys())


for c in classes:
    temp = os.listdir(os.path.join(base_dir, c))[:img_each_class]
    for item in temp:
        img_path = os.path.join(base_dir, c, item)
        img_samples[c] = img_path

for i in img_samples:
    fig = plt.gcf()
    img = mpimg.imread(img_samples[i])
    plt.title(i)
    plt.imshow(img)
    plt.show()
```


dandelion
0
50
100
150
200
0
50
100
150


tulip
0
50
100
150
200
250
300
350
0
100
200
300
400


sunflower
0
20
40
60
80
100



```
IMAGE_SIZE = (200,200)
BATCH_SIZE = 32
SEED = 999
```


```
# Menggunakan ImageDataGenerator untuk preprocessing
import tensorflow as tf

datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    validation_split=0.2
)
```


```
# Menyiapkan data train dan data validation
train_data = datagen.flow_from_directory(
    base_dir,
    class_mode='categorical',
    subset='training',
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    seed=SEED
)

valid_data = datagen.flow_from_directory(
    base_dir,
    class_mode='categorical',
    subset='validation',
    target_size=IMAGE_SIZE,
    batch_size=BATCH_SIZE,
    seed=SEED
)
```

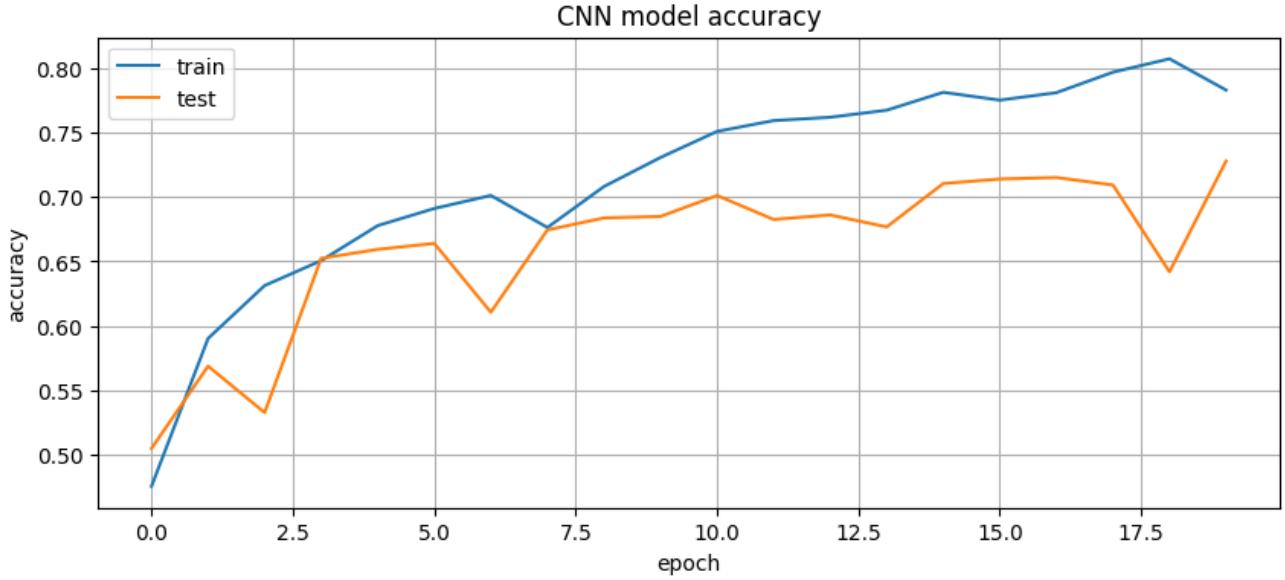
```
Found 3457 images belonging to 5 classes.
Found 860 images belonging to 5 classes.
```

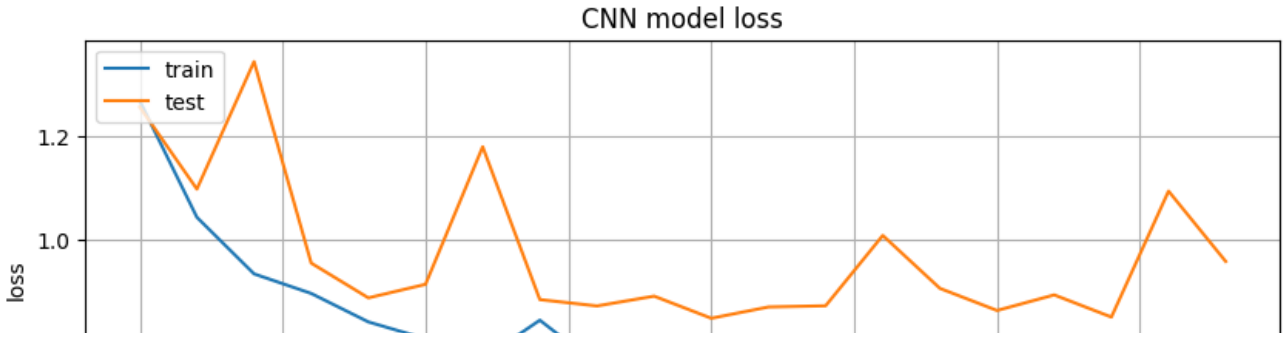
```
# Image Augmentation
data_augmentation = tf.keras.Sequential(
  [
    tf.keras.layers.RandomFlip("horizontal",
                      input_shape=(IMAGE_SIZE[0],
                                   IMAGE_SIZE[1],
                                   3)),
    tf.keras.layers.RandomRotation(0.1),
    tf.keras.layers.RandomZoom(0.1),
    tf.keras.layers.Rescaling(1./255)
  ]
)


# Membuat arsitektur model CNN
cnn_model = tf.keras.models.Sequential([
  data_augmentation,
  tf.keras.layers.Conv2D(32, 3, padding='same', activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Conv2D(64, 3, padding='same', activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Conv2D(64, 3, padding='same', activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Dropout(0.3),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.Dense(5, activation='softmax')
])

# Compiling model
cnn_model.compile(
    loss='categorical_crossentropy',
    optimizer=tf.keras.optimizers.Adam(),
    metrics=['accuracy']
  )


# Training model CNN
cnn_hist = cnn_model.fit(
    train_data,
    epochs=20,
    validation_data = valid_data
)
```

```
Epoch 1/20
109/109 [==============================] - 30s 159ms/step - loss: 1.2669 - accuracy: 0.4753
Epoch 2/20
109/109 [==============================] - 16s 146ms/step - loss: 1.0425 - accuracy: 0.5901
Epoch 3/20
109/109 [==============================] - 16s 147ms/step - loss: 0.9333 - accuracy: 0.6312
Epoch 4/20
109/109 [==============================] - 16s 148ms/step - loss: 0.8955 - accuracy: 0.6506
Epoch 5/20
109/109 [==============================] - 16s 145ms/step - loss: 0.8406 - accuracy: 0.6778
Epoch 6/20
109/109 [==============================] - 17s 154ms/step - loss: 0.8082 - accuracy: 0.6911
Epoch 7/20
109/109 [==============================] - 16s 146ms/step - loss: 0.7688 - accuracy: 0.7012
Epoch 8/20
109/109 [==============================] - 16s 146ms/step - loss: 0.8439 - accuracy: 0.6763
```

```
Epoch 9/20
109/109 [==============================] - 18s 167ms/step - loss: 0.7583 - accuracy: 0.7081
Epoch 10/20
109/109 [==============================] - 16s 145ms/step - loss: 0.6977 - accuracy: 0.7307
Epoch 11/20
109/109 [==============================] - 16s 147ms/step - loss: 0.6629 - accuracy: 0.7509
Epoch 12/20
109/109 [==============================] - 17s 151ms/step - loss: 0.6280 - accuracy: 0.7593
Epoch 13/20
109/109 [==============================] - 16s 145ms/step - loss: 0.6165 - accuracy: 0.7619
Epoch 14/20
109/109 [==============================] - 16s 144ms/step - loss: 0.6027 - accuracy: 0.7674
Epoch 15/20
109/109 [==============================] - 16s 148ms/step - loss: 0.5867 - accuracy: 0.7813
Epoch 16/20
109/109 [==============================] - 16s 145ms/step - loss: 0.5626 - accuracy: 0.7752
Epoch 17/20
109/109 [==============================] - 16s 145ms/step - loss: 0.5681 - accuracy: 0.7810
Epoch 18/20
109/109 [==============================] - 17s 152ms/step - loss: 0.5152 - accuracy: 0.7969
Epoch 19/20
109/109 [==============================] - 16s 147ms/step - loss: 0.5084 - accuracy: 0.8073
Epoch 20/20
109/109 [==============================] - 16s 147ms/step - loss: 0.5742 - accuracy: 0.7830
```

```
# Membuat plot akurasi model CNN
plt.figure(figsize=(10,4))
plt.plot(cnn_hist.history['accuracy'])
plt.plot(cnn_hist.history['val_accuracy'])
plt.title('CNN model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.grid(True)
plt.show()

print()

# Membuat plot loss model CNN
plt.figure(figsize=(10,4))
plt.plot(cnn_hist.history['loss'])
plt.plot(cnn_hist.history['val_loss'])
plt.title('CNN model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.grid(True)
plt.show()
```



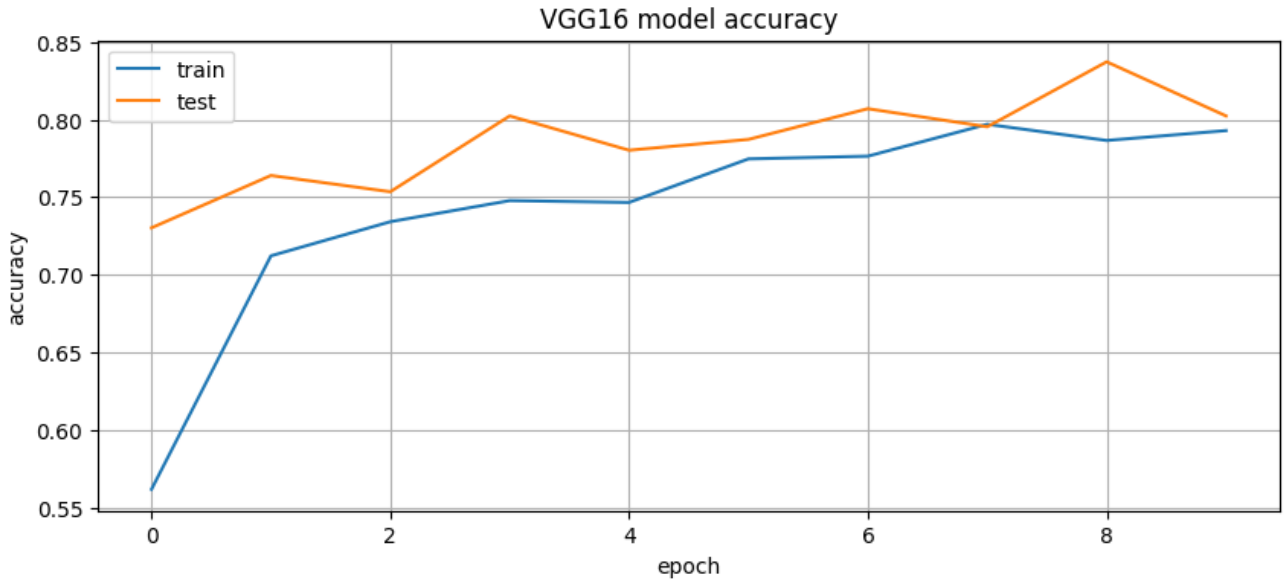

```
import tensorflow as tf
from tensorflow.keras.applications.vgg16 import VGG16

## Loading VGG16 model
base_vgg_model = VGG16(weights="imagenet", include_top=False, input_shape=(IMAGE_SIZE[0], IMAGE_SI
base_vgg_model.trainable = False

# Preprocessing Input
vgg_preprocess = tf.keras.applications.vgg16.preprocess_input
train_data.preprocessing_function = vgg_preprocess
# Transfer learning dengan VGG16
vgg_model = tf.keras.models.Sequential([
  data_augmentation,
  base_vgg_model,
  tf.keras.layers.Dropout(0.7),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.Dense(5, activation='softmax')
])

# Compiling model
vgg_model.compile(
    loss='categorical_crossentropy',
    optimizer=tf.keras.optimizers.Adam(),
    metrics=['accuracy']
  )
```

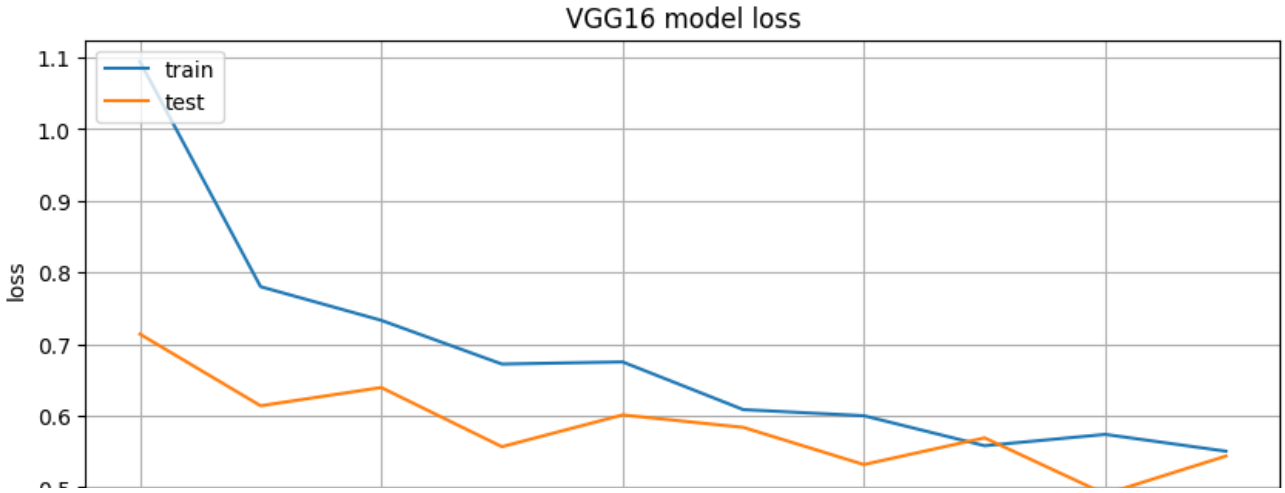
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg
58889256/58889256 [==============================] - 1s 0us/step
```

```
# Melatih model VGG16
vgg_hist = vgg_model.fit(
    train_data,
    epochs=10,
    validation_data = valid_data
)
```

```
Epoch 1/10
109/109 [==============================] - 32s 233ms/step - loss: 1.0941 - accuracy: 0.5618
Epoch 2/10
109/109 [==============================] - 20s 185ms/step - loss: 0.7802 - accuracy: 0.7122
Epoch 3/10
109/109 [==============================] - 20s 186ms/step - loss: 0.7332 - accuracy: 0.7342
Epoch 4/10
109/109 [==============================] - 20s 182ms/step - loss: 0.6722 - accuracy: 0.7478
Epoch 5/10
109/109 [==============================] - 21s 187ms/step - loss: 0.6752 - accuracy: 0.7466
Epoch 6/10
109/109 [==============================] - 20s 187ms/step - loss: 0.6087 - accuracy: 0.7747
Epoch 7/10
109/109 [==============================] - 21s 188ms/step - loss: 0.6001 - accuracy: 0.7764
Epoch 8/10
109/109 [==============================] - 21s 190ms/step - loss: 0.5583 - accuracy: 0.7969
Epoch 9/10
109/109 [==============================] - 20s 186ms/step - loss: 0.5741 - accuracy: 0.7865
Epoch 10/10
109/109 [==============================] - 21s 195ms/step - loss: 0.5506 - accuracy: 0.7929
```

```
# Membuat plot akurasi model VGG16
plt.figure(figsize=(10,4))
plt.plot(vgg_hist.history['accuracy'])
plt.plot(vgg_hist.history['val_accuracy'])
plt.title('VGG16 model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.grid(True)
plt.show()

print()

# Membuat plot loss model VGG16
plt.figure(figsize=(10,4))
plt.plot(vgg_hist.history['loss'])
plt.plot(vgg_hist.history['val_loss'])
plt.title('VGG16 model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.grid(True)
plt.show()
```




```
from tensorflow.keras.applications import ResNet50

# Loading ResNet50 model
base_resnet_model = ResNet50(include_top=False,
                   input_shape=(IMAGE_SIZE[0],IMAGE_SIZE[1],3),
                   pooling='max',classes=5,
                   weights='imagenet')

base_resnet_model.trainable = False

train_data.preprocessing_function = tf.keras.applications.resnet50.preprocess_input


# Transfer learning ResNet50
resnet_model = tf.keras.models.Sequential([
    data_augmentation,
    base_resnet_model,
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation="relu"),
    tf.keras.layers.Dense(64, activation="relu"),
    tf.keras.layers.Dense(5, activation="softmax")
])
# Compiling model
resnet_model.compile(
    loss='categorical_crossentropy',
    optimizer=tf.keras.optimizers.Adam(),
```

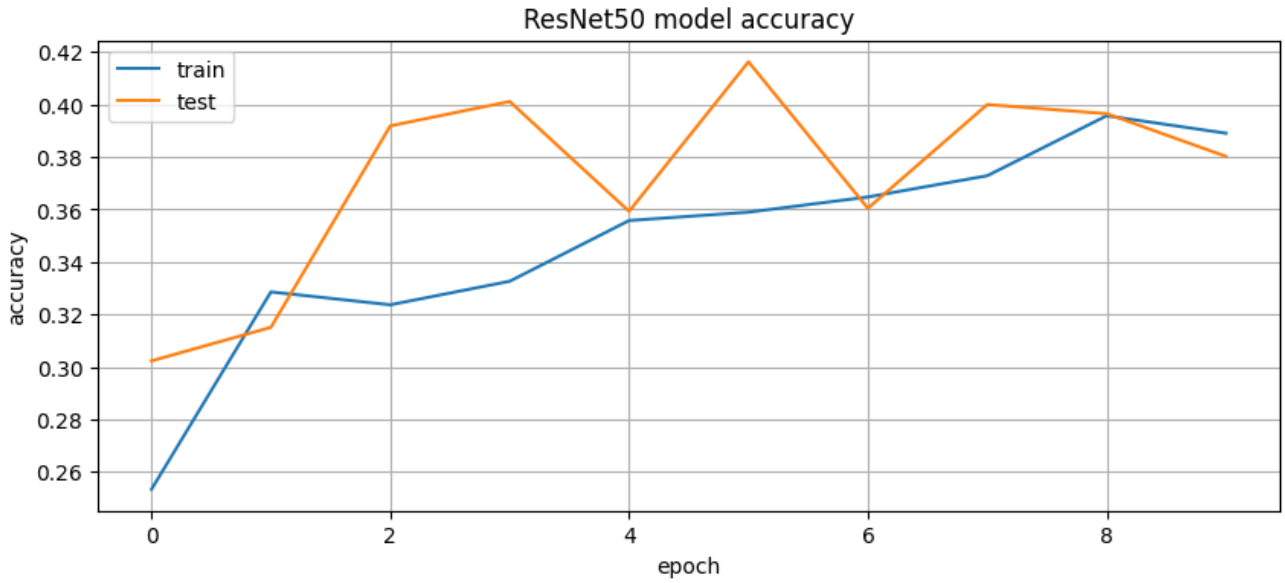
```
    metrics=['accuracy']
  )
```

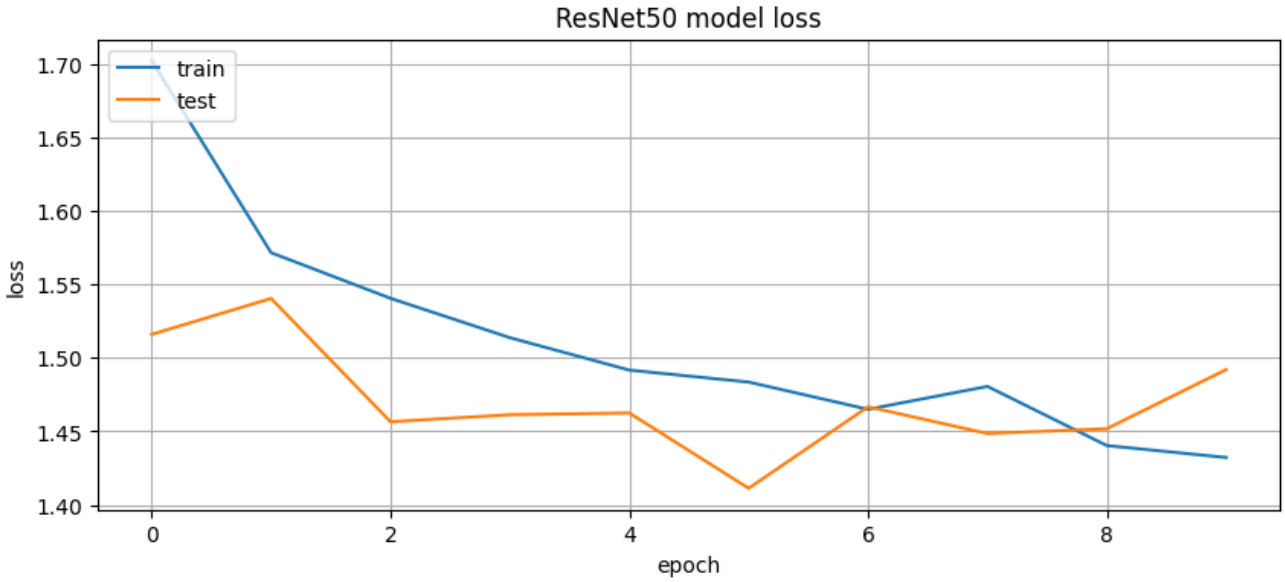
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/re
94765736/94765736 [==============================] - 1s 0us/step
```

```
# Melatih model ResNet50
resnet_hist = resnet_model.fit(
    train_data,
    epochs=10,
    validation_data = valid_data
)
```

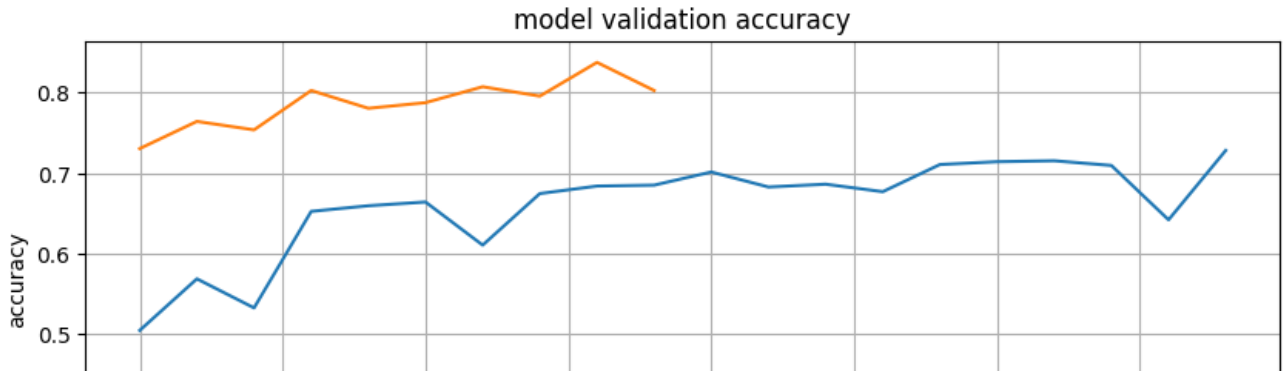
```
Epoch 1/10
109/109 [==============================] - 26s 193ms/step - loss: 1.7021 - accuracy: 0.2534
Epoch 2/10
109/109 [==============================] - 19s 172ms/step - loss: 1.5716 - accuracy: 0.3286
Epoch 3/10
109/109 [==============================] - 18s 167ms/step - loss: 1.5405 - accuracy: 0.3237
Epoch 4/10
109/109 [==============================] - 18s 162ms/step - loss: 1.5137 - accuracy: 0.3327
Epoch 5/10
109/109 [==============================] - 18s 167ms/step - loss: 1.4917 - accuracy: 0.3558
Epoch 6/10
109/109 [==============================] - 18s 163ms/step - loss: 1.4835 - accuracy: 0.3590
Epoch 7/10
109/109 [==============================] - 18s 164ms/step - loss: 1.4650 - accuracy: 0.3648
Epoch 8/10
109/109 [==============================] - 18s 160ms/step - loss: 1.4806 - accuracy: 0.3729
Epoch 9/10
109/109 [==============================] - 17s 160ms/step - loss: 1.4404 - accuracy: 0.3957
Epoch 10/10
109/109 [==============================] - 18s 166ms/step - loss: 1.4322 - accuracy: 0.3891
```

```
# Membuat plot akurasi model ResNet50
plt.figure(figsize=(10,4))
plt.plot(resnet_hist.history['accuracy'])
plt.plot(resnet_hist.history['val_accuracy'])
plt.title('ResNet50 model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.grid(True)
plt.show()

print()

# Membuat plot loss model ResNet50
plt.figure(figsize=(10,4))
plt.plot(resnet_hist.history['loss'])
plt.plot(resnet_hist.history['val_loss'])
plt.title('ResNet50 model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.grid(True)
plt.show()
```




```
# Membuat plot akurasi empat model sebelumnya untuk dibandingkan
plt.figure(figsize=(10,4))
plt.plot(cnn_hist.history['val_accuracy'])
plt.plot(vgg_hist.history['val_accuracy'])
plt.plot(resnet_hist.history['val_accuracy'])
plt.title('model validation accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['CNN', 'VGG16', 'ResNet50'], loc='lower right')
plt.grid(True)
plt.show()
```




```
# Menampilkan daftar kelas atau label gambar
train_data.class_indices
```

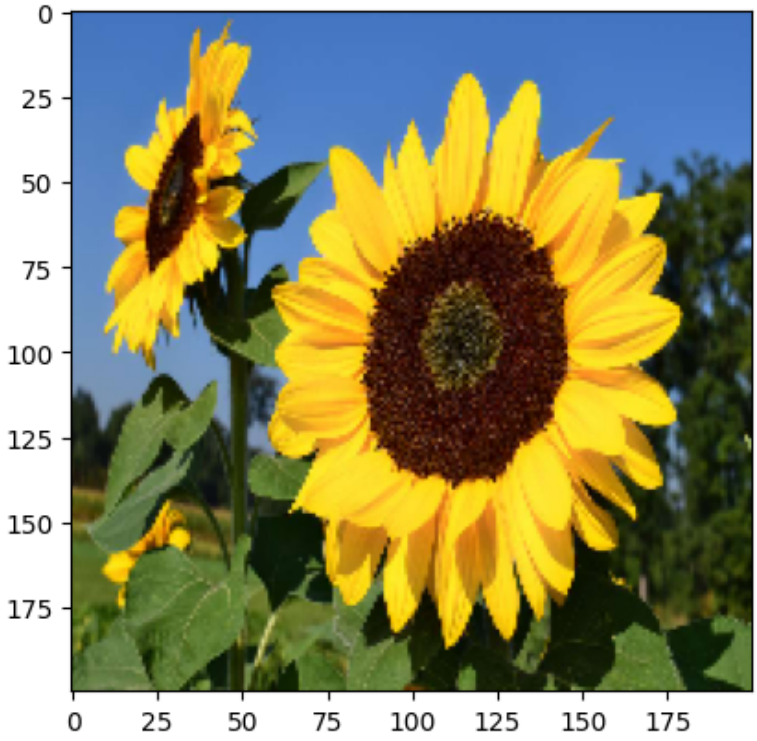
```
{'daisy': 0, 'dandelion': 1, 'rose': 2, 'sunflower': 3, 'tulip': 4}
```

```
# Menguji coba model
import numpy as np
import tensorflow as tf
from tensorflow.keras.applications import VGG16
from tensorflow.keras.preprocessing import image
from google.colab import files
%matplotlib inline


#file upload, kode di bawah in hanya bisa dijalankan di google colab dengan mengimport from google
#atau kalian langsung import file gambarnya langsung
uploaded = files.upload()

for fn in uploaded.keys():

  # prediksi gambar
  path = fn
  img = image.load_img(path, target_size=IMAGE_SIZE)
  imgplot = plt.imshow(img)
  x = image.img_to_array(img)
  x = np.expand_dims(x, axis=0)

  images = np.vstack([x])
  classes = vgg_model.predict(images, batch_size=BATCH_SIZE)
  classes = np.argmax(classes)

  print(fn)
  if classes==0:
    print('daisy')
  elif classes==1:
    print('dandelion')
  elif classes==2:
    print('rose')
  elif classes==3:
    print('sunflower')
  else:
    print('tulip')
```

Pilih File sunflower.jpg

- **sunflower.jpg**(image/jpeg) - 91281 bytes, last modified: 25/6/2023 - 100% done

```
Saving sunflower.jpg to sunflower (5).jpg
1/1 [==============================] - 0s 188ms/step
sunflower.jpg
sunflower
```




0 d selesai pada 08.22