```
!pip install -q kaggle
!mkdir -p ~/.kaggle
!echo '{"username":"ardianzyh","key":"7b77075c876942baf04f6b03634f1b5f"}' > ~/.kaggle/kaggle.json
!chmod 600 ~/.kaggle/kaggle.json
!kaggle datasets download -d alxmamaev/flowers-recognition
```

```
flowers-recognition.zip: Skipping, found more recently modified local copy (use --force to force download)
```

```
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)
```

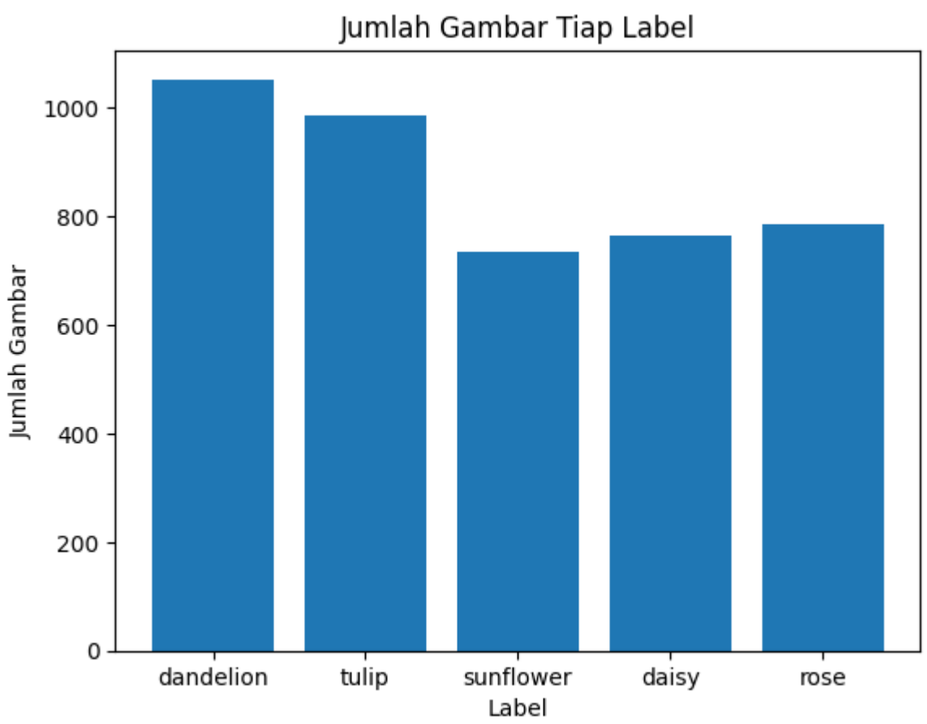
```
import os
base_dir = '/content/flowers/flowers'

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

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

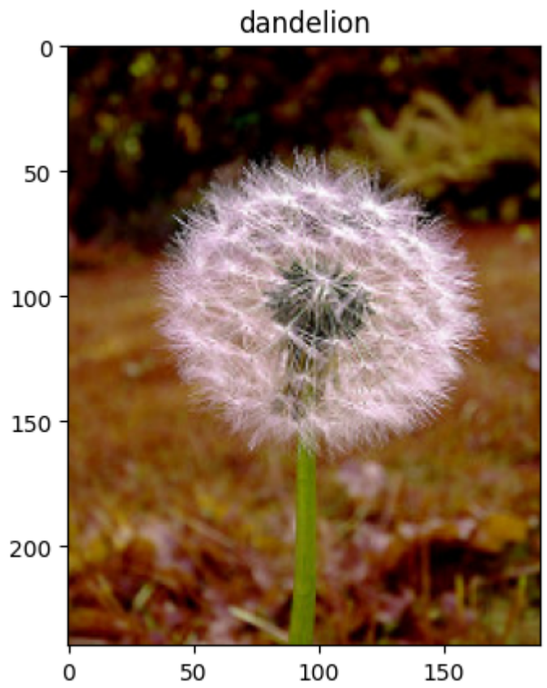
```
# Menghitung jumlah gambar pada dataset
number_label = {}
total_files = 0
for i in os.listdir(base_dir):
    counting = len(os.listdir(os.path.join(base_dir, i)))
    number_label[i] = counting
    total_files += counting

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

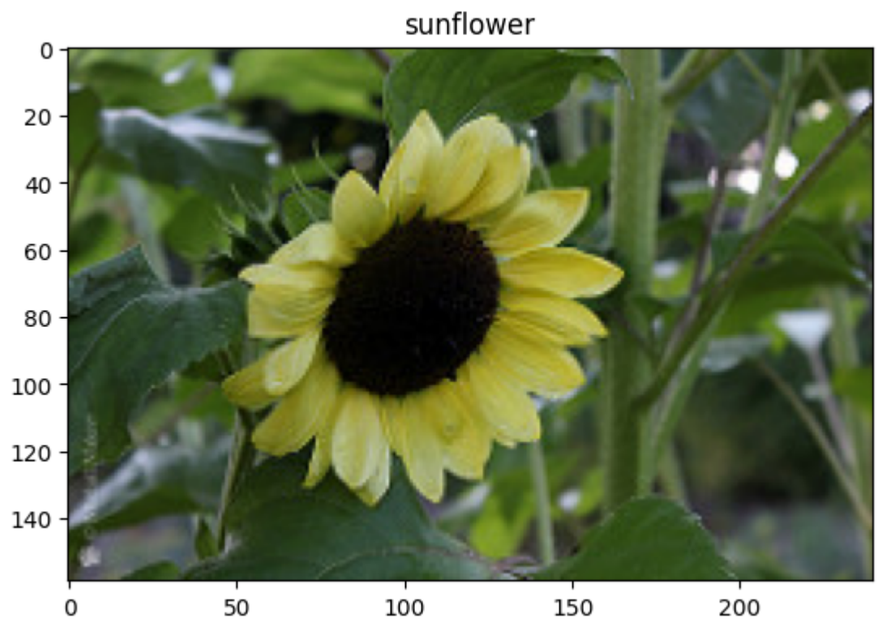
```
Total Files : 4317
```

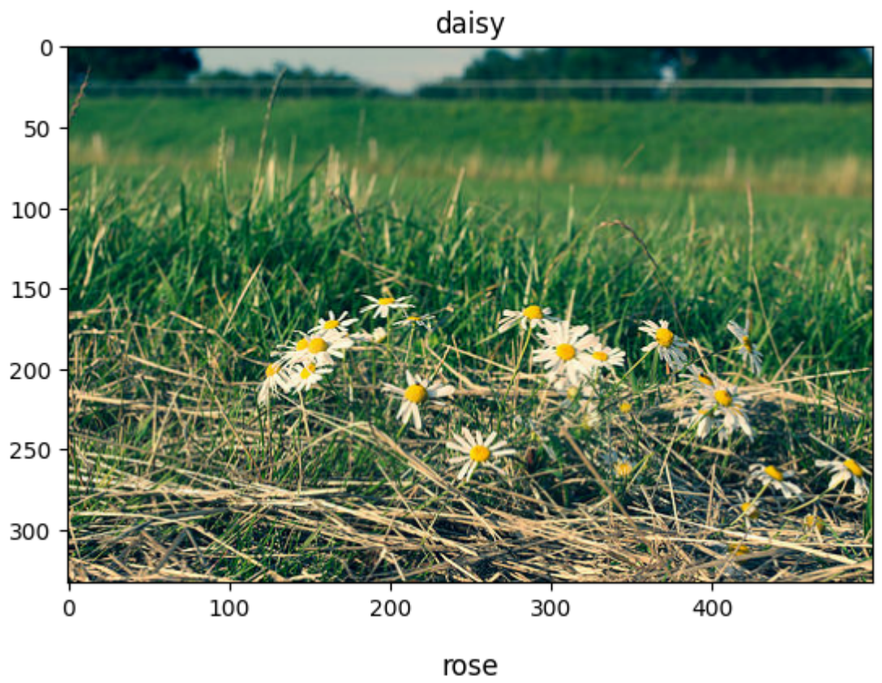
```
# Visualisasi jumlah gambar tiap kelas
import matplotlib.pyplot as plt

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()
```








```
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.
```

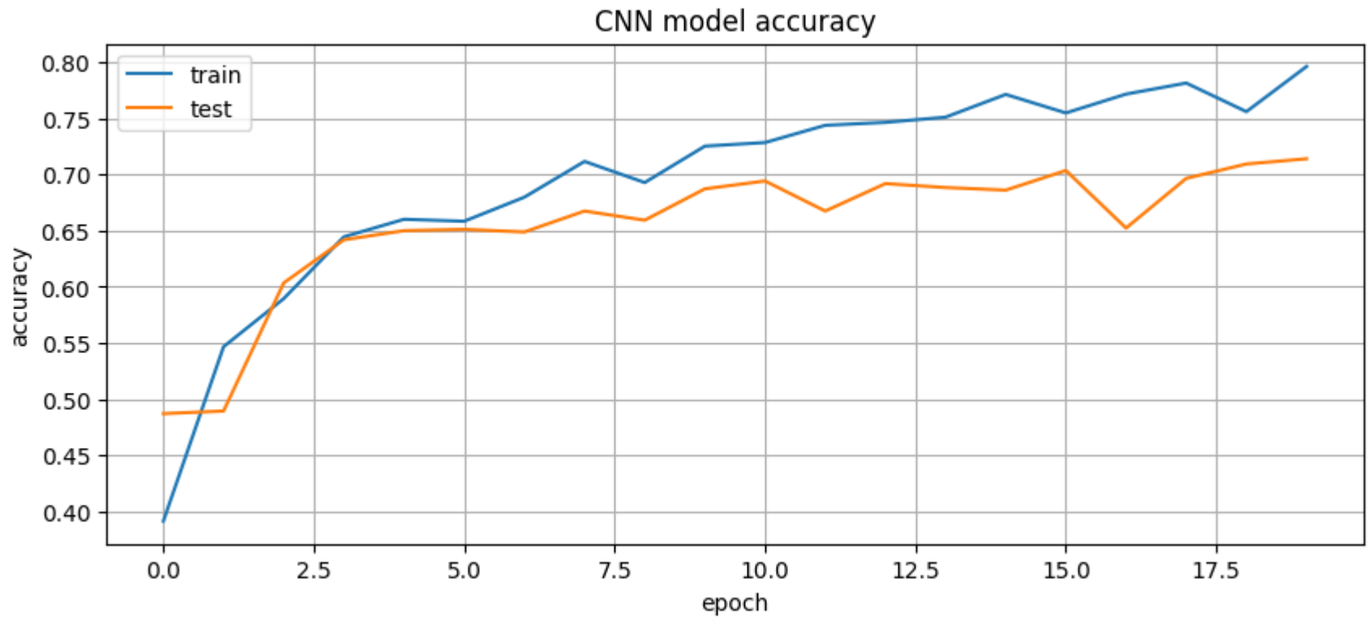
```
# Image Augmentation
data_augmentation = tf.keras.Sequential(
  [
    tf.keras.layers.RandomFlip("horizontal",
                      input_shape=(IMAGE_SIZE[0],
                                   IMAGE_SIZE[1],
                                   3)),
    tf.keras.layers.RandomRotation(0.1),
    tf.keras.layers.RandomZoom(0.1),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

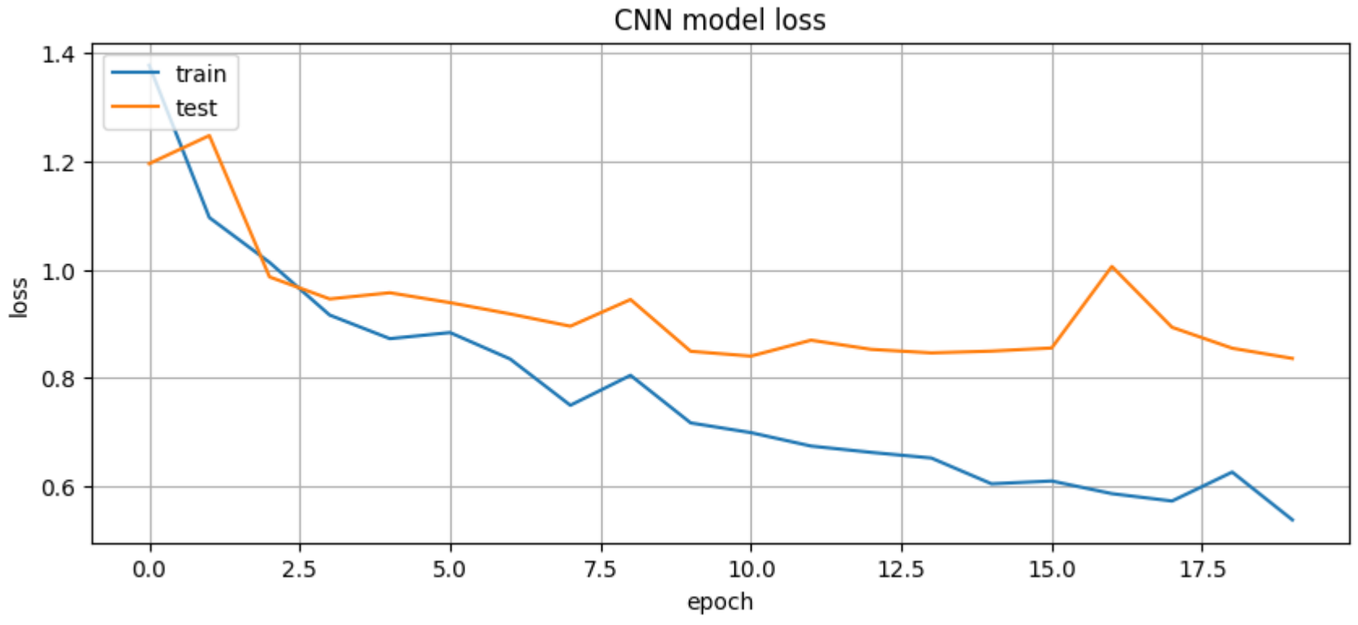
```
# Membuat arsitektur model CNN
cnn_model = tf.keras.models.Sequential([
  data_augmentation,
  tf.keras.layers.Conv2D(32, 3, padding='same', activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Conv2D(64, 3, padding='same', activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  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 [==============================] - 29s 149ms/step - loss: 1.3773 - accuracy: 0.3917 - val_loss: 1.1955 - val_accuracy: 0.4872
Epoch 2/20
109/109 [==============================] - 16s 146ms/step - loss: 1.0960 - accuracy: 0.5467 - val_loss: 1.2475 - val_accuracy: 0.4895
Epoch 3/20
109/109 [==============================] - 15s 141ms/step - loss: 1.0132 - accuracy: 0.5895 - val_loss: 0.9864 - val_accuracy: 0.6035
Epoch 4/20
109/109 [==============================] - 15s 140ms/step - loss: 0.9159 - accuracy: 0.6445 - val_loss: 0.9456 - val_accuracy: 0.6419
Epoch 5/20
109/109 [==============================] - 15s 141ms/step - loss: 0.8723 - accuracy: 0.6601 - val_loss: 0.9571 - val_accuracy: 0.6500
Epoch 6/20
109/109 [==============================] - 16s 141ms/step - loss: 0.8835 - accuracy: 0.6584 - val_loss: 0.9388 - val_accuracy: 0.6512
Epoch 7/20
109/109 [==============================] - 16s 146ms/step - loss: 0.8347 - accuracy: 0.6798 - val_loss: 0.9180 - val_accuracy: 0.6488
Epoch 8/20
109/109 [==============================] - 16s 142ms/step - loss: 0.7492 - accuracy: 0.7116 - val_loss: 0.8953 - val_accuracy: 0.6674
Epoch 9/20
109/109 [==============================] - 18s 165ms/step - loss: 0.8046 - accuracy: 0.6928 - val_loss: 0.9446 - val_accuracy: 0.6593
Epoch 10/20
109/109 [==============================] - 16s 147ms/step - loss: 0.7167 - accuracy: 0.7252 - val_loss: 0.8490 - val_accuracy: 0.6872
Epoch 11/20
109/109 [==============================] - 15s 141ms/step - loss: 0.6987 - accuracy: 0.7284 - val_loss: 0.8400 - val_accuracy: 0.6942
Epoch 12/20
109/109 [==============================] - 18s 163ms/step - loss: 0.6739 - accuracy: 0.7437 - val_loss: 0.8695 - val_accuracy: 0.6674
Epoch 13/20
109/109 [==============================] - 15s 140ms/step - loss: 0.6622 - accuracy: 0.7463 - val_loss: 0.8525 - val_accuracy: 0.6919
Epoch 14/20
109/109 [==============================] - 15s 141ms/step - loss: 0.6518 - accuracy: 0.7509 - val_loss: 0.8460 - val_accuracy: 0.6884
Epoch 15/20
109/109 [==============================] - 16s 146ms/step - loss: 0.6042 - accuracy: 0.7712 - val_loss: 0.8493 - val_accuracy: 0.6860
Epoch 16/20
109/109 [==============================] - 18s 165ms/step - loss: 0.6094 - accuracy: 0.7547 - val_loss: 0.8551 - val_accuracy: 0.7035
Epoch 17/20
109/109 [==============================] - 16s 142ms/step - loss: 0.5860 - accuracy: 0.7715 - val_loss: 1.0057 - val_accuracy: 0.6523
Epoch 18/20
109/109 [==============================] - 15s 141ms/step - loss: 0.5724 - accuracy: 0.7813 - val_loss: 0.8933 - val_accuracy: 0.6965
Epoch 19/20
109/109 [==============================] - 16s 145ms/step - loss: 0.6258 - accuracy: 0.7559 - val_loss: 0.8546 - val_accuracy: 0.7093
Epoch 20/20
109/109 [==============================] - 15s 141ms/step - loss: 0.5376 - accuracy: 0.7961 - val_loss: 0.8358 - val_accuracy: 0.7140
```

```
# 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_SIZE[1], 3))
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']
  )
```

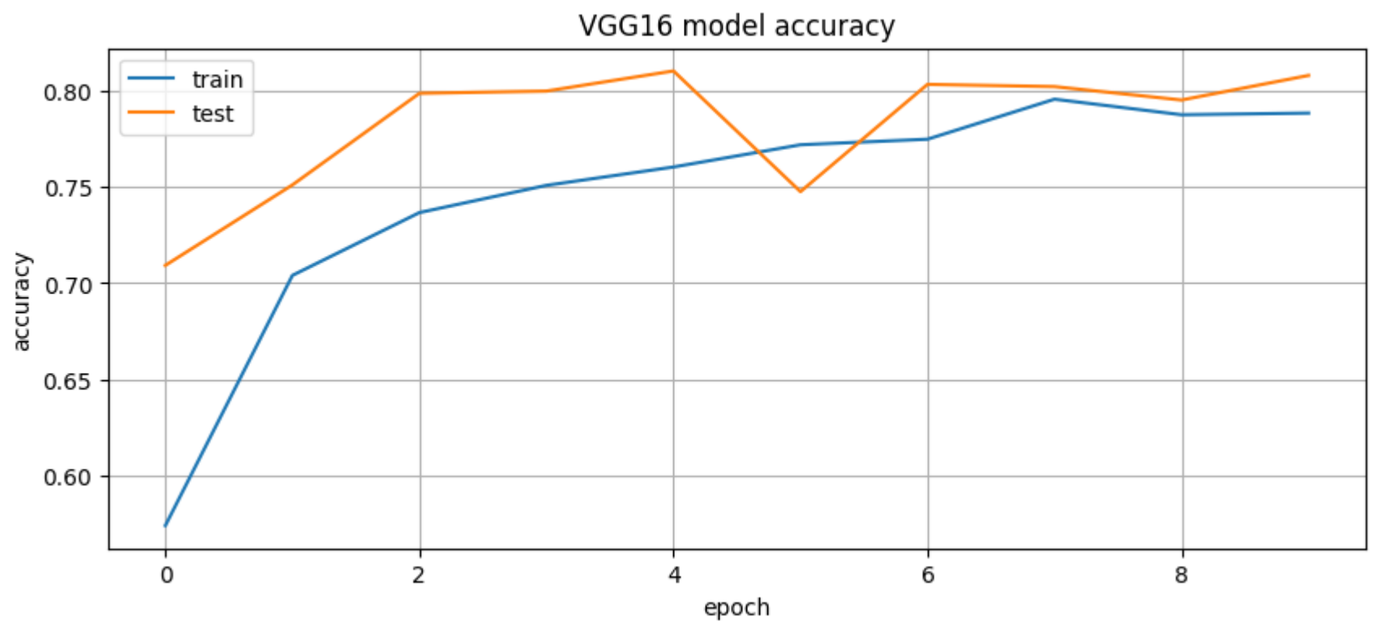
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58889256/58889256 [==============================] - 2s 0us/step
```

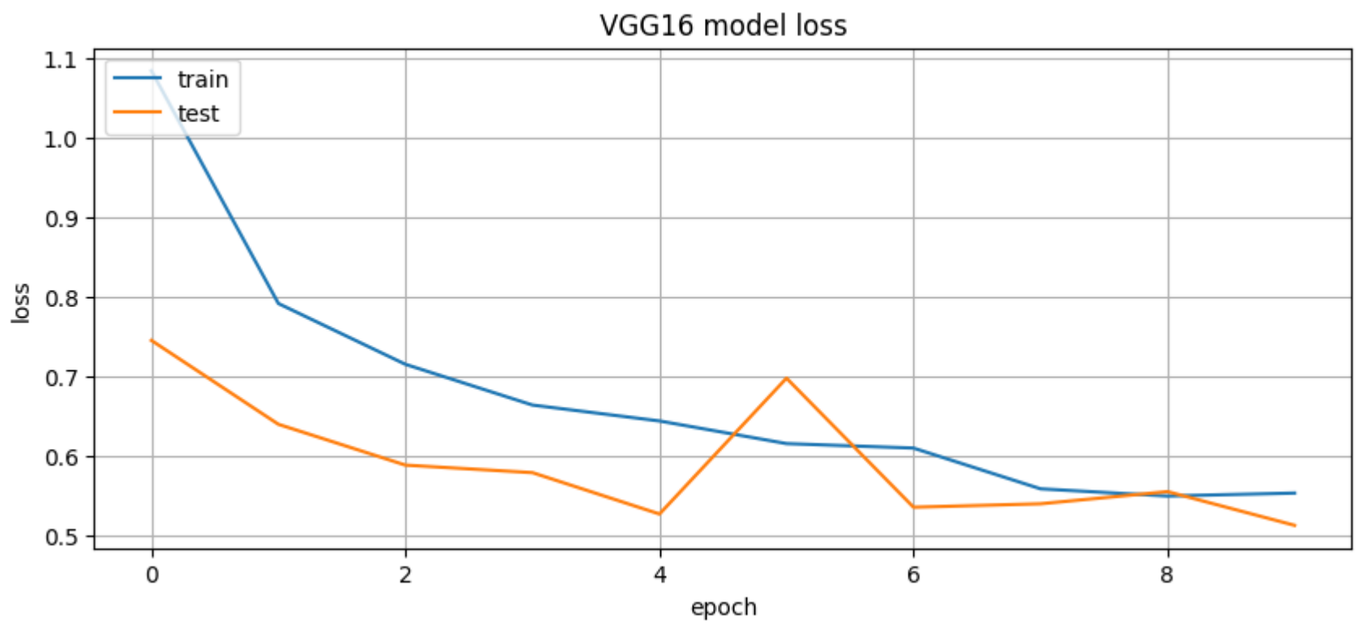
```
# Melatih model VGG16
vgg_hist = vgg_model.fit(
    train_data,
    epochs=10,
    validation_data = valid_data
)
```

```
Epoch 1/10
109/109 [==============================] - 29s 213ms/step - loss: 1.0835 - accuracy: 0.5739 - val_loss: 0.7447 - val_accuracy: 0.7093
Epoch 2/10
109/109 [==============================] - 19s 171ms/step - loss: 0.7910 - accuracy: 0.7041 - val_loss: 0.6393 - val_accuracy: 0.7512
Epoch 3/10
109/109 [==============================] - 19s 176ms/step - loss: 0.7146 - accuracy: 0.7368 - val_loss: 0.5880 - val_accuracy: 0.7988
Epoch 4/10
109/109 [==============================] - 21s 191ms/step - loss: 0.6635 - accuracy: 0.7509 - val_loss: 0.5785 - val_accuracy: 0.8000
Epoch 5/10
109/109 [==============================] - 19s 176ms/step - loss: 0.6435 - accuracy: 0.7605 - val_loss: 0.5265 - val_accuracy: 0.8105
Epoch 6/10
109/109 [==============================] - 19s 178ms/step - loss: 0.6150 - accuracy: 0.7721 - val_loss: 0.6972 - val_accuracy: 0.7477
Epoch 7/10
109/109 [==============================] - 19s 178ms/step - loss: 0.6094 - accuracy: 0.7749 - val_loss: 0.5351 - val_accuracy: 0.8035
Epoch 8/10
109/109 [==============================] - 21s 191ms/step - loss: 0.5582 - accuracy: 0.7958 - val_loss: 0.5394 - val_accuracy: 0.8023
Epoch 9/10
109/109 [==============================] - 19s 176ms/step - loss: 0.5491 - accuracy: 0.7877 - val_loss: 0.5547 - val_accuracy: 0.7953
Epoch 10/10
109/109 [==============================] - 19s 178ms/step - loss: 0.5529 - accuracy: 0.7885 - val_loss: 0.5123 - val_accuracy: 0.8081
```

```
# 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']
  )
```

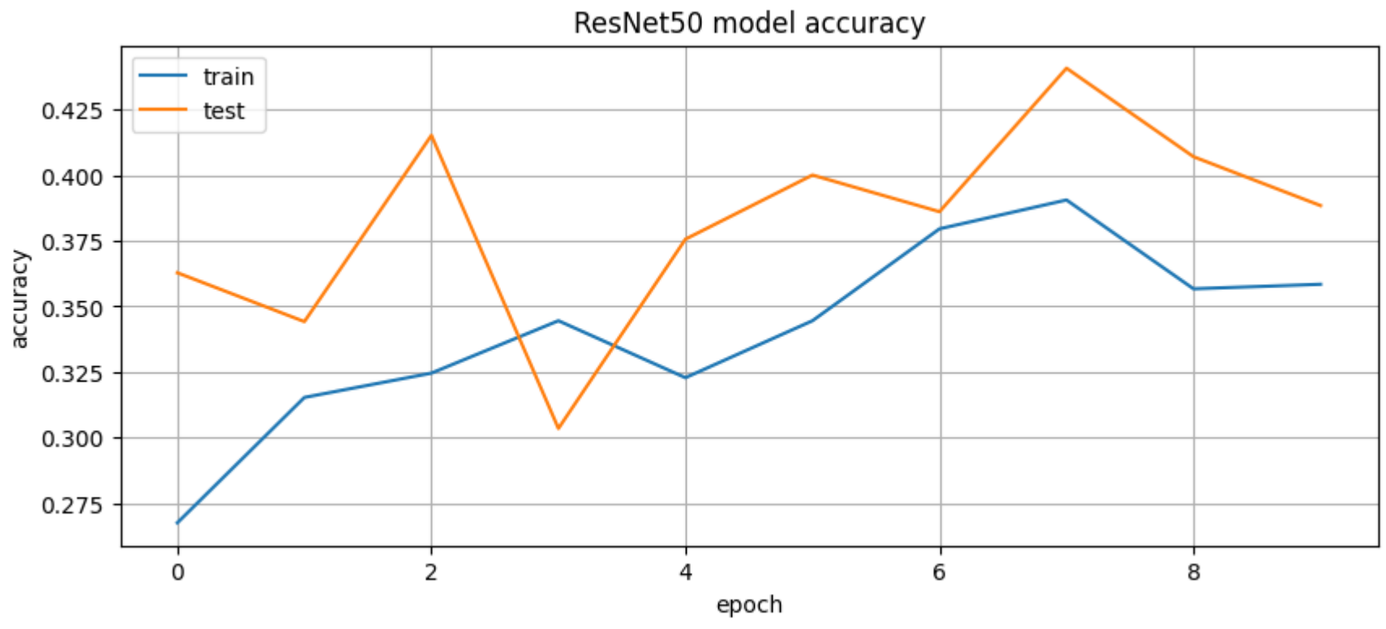
```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 [==============================] - 3s 0us/step
```

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

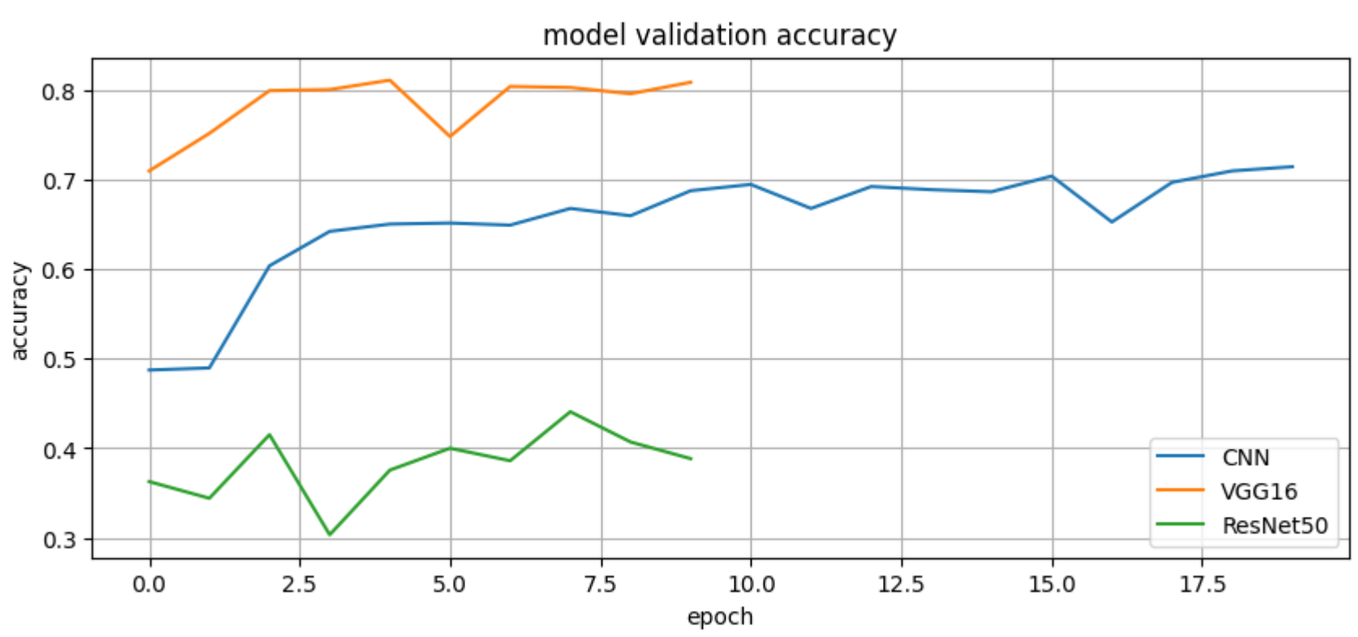
```
Epoch 1/10
109/109 [==============================] - 25s 188ms/step - loss: 1.6694 - accuracy: 0.2676 - val_loss: 1.4765 - val_accuracy: 0.3628
Epoch 2/10
109/109 [==============================] - 17s 157ms/step - loss: 1.5675 - accuracy: 0.3153 - val_loss: 1.4938 - val_accuracy: 0.3442
Epoch 3/10
109/109 [==============================] - 18s 161ms/step - loss: 1.5496 - accuracy: 0.3246 - val_loss: 1.4175 - val_accuracy: 0.4151
Epoch 4/10
109/109 [==============================] - 17s 156ms/step - loss: 1.5148 - accuracy: 0.3445 - val_loss: 1.5020 - val_accuracy: 0.3035
Epoch 5/10
109/109 [==============================] - 17s 159ms/step - loss: 1.5433 - accuracy: 0.3228 - val_loss: 1.4764 - val_accuracy: 0.3756
Epoch 6/10
109/109 [==============================] - 17s 158ms/step - loss: 1.4967 - accuracy: 0.3445 - val_loss: 1.4388 - val_accuracy: 0.4000
Epoch 7/10
109/109 [==============================] - 17s 159ms/step - loss: 1.4554 - accuracy: 0.3795 - val_loss: 1.4172 - val_accuracy: 0.3860
Epoch 8/10
109/109 [==============================] - 17s 155ms/step - loss: 1.4416 - accuracy: 0.3905 - val_loss: 1.3940 - val_accuracy: 0.4407
```

```
Epoch 9/10
109/109 [==============================] - 17s 158ms/step - loss: 1.5102 - accuracy: 0.3567 - val_loss: 1.4295 - val_accuracy: 0.4070
Epoch 10/10
109/109 [==============================] - 19s 173ms/step - loss: 1.4744 - accuracy: 0.3584 - val_loss: 1.4118 - val_accuracy: 0.3884
```

```
# 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.colab import files. Silahkan kalian ganti kodingannya agar bisa upload di jupyter notebook mas:
#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')
```

```
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 7104-1.jpg to 7104-1.jpg
1/1 [==============================] - 0s 281ms/step
7104-1.jpg
sunflower
```