```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import PIL
from tensorflow import keras
import pathlib
import cv2
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
```

```
tf.config.list_physical_devices('GPU')
```

```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

```
# flower_photo/
#   daisy/
#   dandelion/
#   roses/
#   sunflowers/
#   tulips/
```

```
# # dataset_url = "https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz"
# dataset_url =
# data_dir = tf.keras.utils.get_file('flower_photos', cache_dir='/content/drive/MyDrive/A Learning Tensor Flow/
```

```
# data_dir
```

```
data_dir = "/content/drive/MyDrive/A Learning Tensor Flow/CNN-Image Classification/datasets/flower_photos"
```

```
data_dir = pathlib.Path(data_dir)
data_dir
```

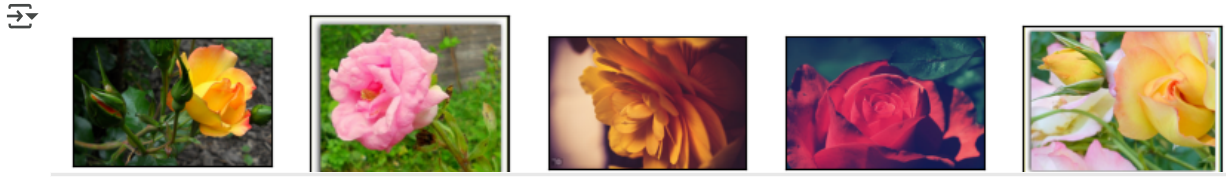
```
PosixPath('/content/drive/MyDrive/A Learning Tensor Flow/CNN-Image Classification/datasets/flower_photos')
```

```
image_count = len(list(data_dir.glob('*/*.jpg')))
print(image_count)
```

```
3670
```

```
roses = list(data_dir.glob('roses/*'))
PIL.Image.open(str(roses[0]))
```

```
fig, axes = plt.subplots(1, 5, figsize=(10, 10))
axes = axes.flatten()
for i in range(5):
    image = PIL.Image.open(str(roses[i]))
    axes[i].imshow(image)
    axes[i].set_xticks([])
    axes[i].set_yticks([])
    axes[i].grid(False)
plt.show()
```



```
tulips = list(data_dir.glob('tulips/*'))
PIL.Image.open(str(tulips[0]))
```

```
flowers_images_dict = {
    'roses': list(data_dir.glob('roses/*')),
    'daisy': list(data_dir.glob('daisy/*')),
    'dandelion': list(data_dir.glob('dandelion/*')),
    'sunflowers': list(data_dir.glob('sunflowers/*')),
    'tulips': list(data_dir.glob('tulips/*')),
}

flowers_labels_dict = {
    'roses': 0,
    'daisy': 1,
    'dandelion': 2,
    'sunflowers': 3,
    'tulips': 4,
}
```

```
flowers_images_dict['roses'][:5]
```

```
[PosixPath('/content/drive/MyDrive/A Learning Tensor Flow/CNN-Image
Classification/datasets/flower_photos/roses/7345657862_689366e79a.jpg'),
 PosixPath('/content/drive/MyDrive/A Learning Tensor Flow/CNN-Image
Classification/datasets/flower_photos/roses/7409458444_0bfc9a0682_n.jpg'),
 PosixPath('/content/drive/MyDrive/A Learning Tensor Flow/CNN-Image
Classification/datasets/flower_photos/roses/9337528427_3d09b7012b.jpg'),
 PosixPath('/content/drive/MyDrive/A Learning Tensor Flow/CNN-Image
Classification/datasets/flower_photos/roses/5736328472_8f25e6f6e7.jpg'),
 PosixPath('/content/drive/MyDrive/A Learning Tensor Flow/CNN-Image
Classification/datasets/flower_photos/roses/7551637034_55ae047756_n.jpg')]
```

```
str(flowers_images_dict['roses'][0])
```

```
img = cv2.imread(str(flowers_images_dict['roses'][0]))
img
```

ndarray (322, 500, 3) show data

```
img.shape
```

(322, 500, 3)

```
batch_size = 32
img_height = 180
img_width = 180


train_ds = tf.keras.utils.image_dataset_from_directory(
    data_dir,
```

```
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size
)

val_ds = tf.keras.utils.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=(img_height, img_width),
    batch_size=batch_size
)
```

```
Found 3670 files belonging to 5 classes.
Using 2936 files for training.
Found 3670 files belonging to 5 classes.
Using 734 files for validation.
```

```
class_name = train_ds.class_names
print(class_name)
```

```
['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']
```

```
AUTOTUNE = tf.data.AUTOTUNE

train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

Training mode

```
num_classes = len(class_name)
model = keras.Sequential([
    keras.layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
    keras.layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu'),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Flatten(),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(num_classes)
])

model.compile(optimizer='adam', loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=[

with tf.device('/device:GPU:0'):
    model.fit(train_ds, validation_data=val_ds, epochs=30)
model.evaluate(val_ds)
```

```
Epoch 4/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.7535 - loss: 0.6793 - val_accuracy: 0.6322 - val_lo
Epoch 5/30
```

```
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9851 - loss: 0.0713 - val_accuracy: 0.6553 - val_lo
Epoch 10/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9935 - loss: 0.0377 - val_accuracy: 0.6485 - val_lo
Epoch 11/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9987 - loss: 0.0141 - val_accuracy: 0.6485 - val_lo
Epoch 12/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9977 - loss: 0.0125 - val_accuracy: 0.6512 - val_lo
Epoch 13/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9995 - loss: 0.0040 - val_accuracy: 0.6649 - val_lo
Epoch 14/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9981 - loss: 0.0094 - val_accuracy: 0.5967 - val_lo
Epoch 15/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9613 - loss: 0.1147 - val_accuracy: 0.6390 - val_lo
Epoch 16/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9862 - loss: 0.0421 - val_accuracy: 0.6417 - val_lo
Epoch 17/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9940 - loss: 0.0273 - val_accuracy: 0.6499 - val_lo
Epoch 18/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9994 - loss: 0.0082 - val_accuracy: 0.6376 - val_lo
Epoch 19/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9891 - loss: 0.0341 - val_accuracy: 0.6158 - val_lo
Epoch 20/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9914 - loss: 0.0297 - val_accuracy: 0.6485 - val_lo
Epoch 21/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9909 - loss: 0.0275 - val_accuracy: 0.6063 - val_lo
Epoch 22/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9797 - loss: 0.0626 - val_accuracy: 0.6376 - val_lo
Epoch 23/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9907 - loss: 0.0335 - val_accuracy: 0.6417 - val_lo
```

```
Epoch 28/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 1.0000 - loss: 4.8565e-04 - val_accuracy: 0.6553 - va
Epoch 29/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 1.0000 - loss: 3.0688e-04 - val_accuracy: 0.6540 - va
Epoch 30/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 1.0000 - loss: 2.7209e-04 - val_accuracy: 0.6526 - va
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.6194 - loss: 3.1875
```

Start coding or generate with AI.

```
# from google.colab import drive
# drive.mount('/content/drive')
```

```
data_augumentation = keras.Sequential([
    keras.layers.RandomFlip("horizontal", input_shape=(img_height, img_width, 3)),
    keras.layers.RandomRotation(0.1),
    keras.layers.RandomZoom(0.1),
])
```

```
num_classes = len(class_name)
model_aug = keras.Sequential([
    data_augumentation,
    keras.layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
    keras.layers.Conv2D(filters=16, kernel_size=(3, 3), padding='same', activation='relu'),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Conv2D(filters=32, kernel_size=(3, 3), padding='same', activation='relu'),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Conv2D(filters=64, kernel_size=(3, 3), padding='same', activation='relu'),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Dropout(0.2),
    keras.layers.Flatten(),
```

```
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(num_classes)
])

model_aug.compile(optimizer='adam', loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metri

with tf.device('/device:GPU:0'):
    history = model_aug.fit(train_ds, validation_data=val_ds, epochs=30)
model_aug.evaluate(val_ds)
```

```
Epoch 4/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.6501 - loss: 0.9185 - val_accuracy: 0.6444 - val_lo
Epoch 5/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.6596 - loss: 0.8866 - val_accuracy: 0.6635 - val_lo
Epoch 6/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.6829 - loss: 0.8134 - val_accuracy: 0.6294 - val_lo
Epoch 7/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.7080 - loss: 0.7886 - val_accuracy: 0.6594 - val_lo
Epoch 8/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.6983 - loss: 0.7787 - val_accuracy: 0.6662 - val_lo
Epoch 9/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.7278 - loss: 0.7401 - val_accuracy: 0.6962 - val_lo
Epoch 10/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.7260 - loss: 0.7185 - val_accuracy: 0.6812 - val_lo
Epoch 11/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.7314 - loss: 0.6898 - val_accuracy: 0.6948 - val_lo
Epoch 12/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.7557 - loss: 0.6582 - val_accuracy: 0.7003 - val_lo
Epoch 13/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.7484 - loss: 0.6619 - val_accuracy: 0.7030 - val_lo
```

```
Epoch 17/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.7952 - loss: 0.5744 - val_accuracy: 0.7030 - val_lo
Epoch 18/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.7803 - loss: 0.5789 - val_accuracy: 0.7248 - val_lo
Epoch 19/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.8010 - loss: 0.5322 - val_accuracy: 0.7139 - val_lo
Epoch 20/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.8117 - loss: 0.5097 - val_accuracy: 0.7398 - val_lo
Epoch 21/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.7950 - loss: 0.5518 - val_accuracy: 0.7098 - val_lo
Epoch 22/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.8111 - loss: 0.5066 - val_accuracy: 0.7112 - val_lo
Epoch 23/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.8051 - loss: 0.5039 - val_accuracy: 0.7193 - val_lo
Epoch 24/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.8153 - loss: 0.4919 - val_accuracy: 0.7439 - val_lo
Epoch 25/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.8442 - loss: 0.4111 - val_accuracy: 0.7493 - val_lo
Epoch 26/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.8422 - loss: 0.4321 - val_accuracy: 0.7548 - val_lo
Epoch 27/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.8360 - loss: 0.4184 - val_accuracy: 0.7398 - val_lo
Epoch 28/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.8394 - loss: 0.4214 - val_accuracy: 0.7289 - val_lo
Epoch 29/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.8375 - loss: 0.4114 - val_accuracy: 0.7589 - val_lo
Epoch 30/30
92/92 ━━━━━━━━━━━━━━━━━━━━ 1s 8ms/step - accuracy: 0.8648 - loss: 0.3865 - val_accuracy: 0.7534 - val_lo
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.7225 - loss: 0.7785
[0.7301371097564697, 0.7534059882164001]
```

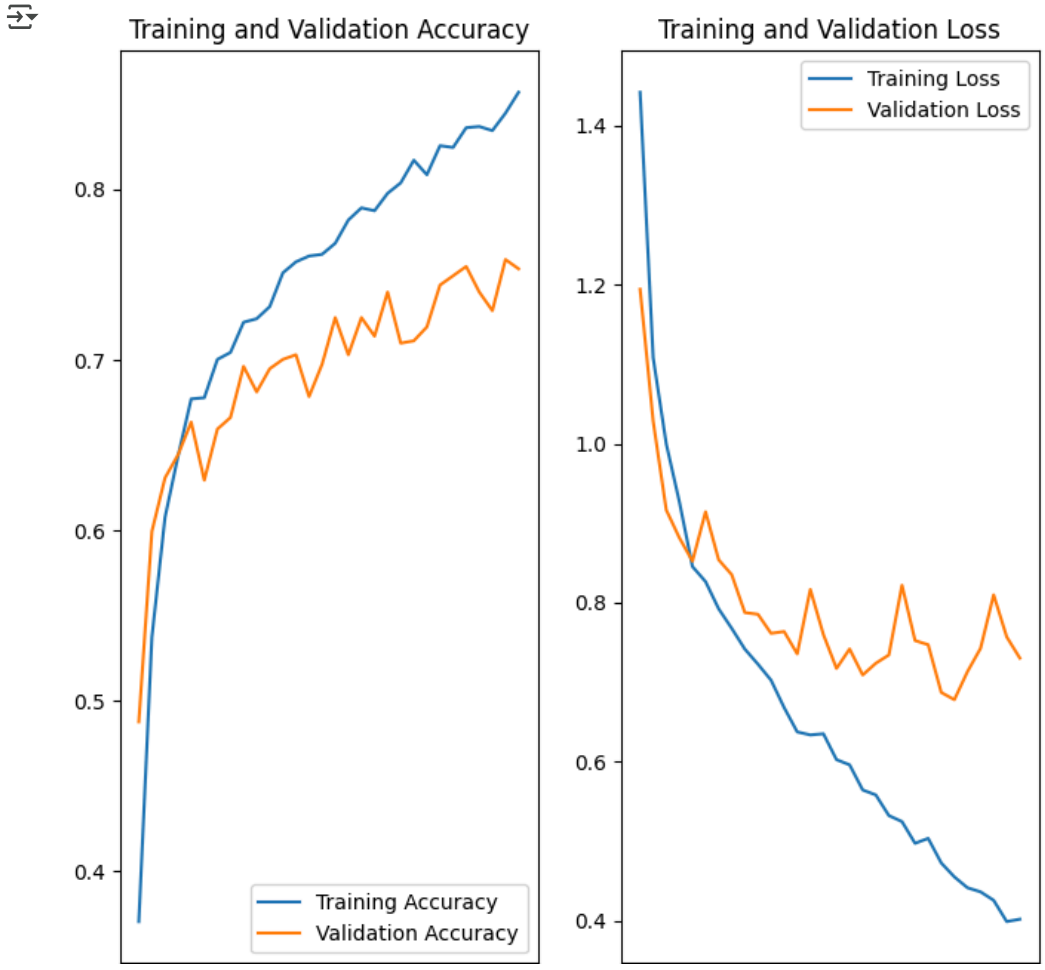
```
epochs = 30
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
```

```
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```


Training and Validation Accuracy
0.8
0.7
0.6
0.5
0.4
Training Accuracy
Validation Accuracy
Training and Validation Loss
Training Loss
Validation Loss
1.4
1.2
1.0
0.8
0.6
0.4

```
class_names = ['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']


# prompt: create code to predict 10 random image from my folder.

import random
import matplotlib.pyplot as plt
import cv2
import os

# Assuming 'data_dir' and 'model_aug' are defined from the previous code.
# If not, define them as shown in the previous code block.

def predict_random_images(num_images=20):
    """Predicts the class of random images from the dataset and displays them."""

    random_image_paths = random.sample(list(data_dir.glob('*/*.jpg')), num_images)
    fig, axes = plt.subplots(4,5, figsize=(20, 10))
    axes = axes.flatten()
    for idx, image_path in enumerate(random_image_paths):
        img = cv2.imread(str(image_path))
        img = cv2.resize(img, (img_height, img_width))  # Resize the image
        img_array = tf.keras.utils.img_to_array(img)
        img_array = tf.expand_dims(img_array, 0) # Create a batch

        predictions = model_aug.predict(img_array)
        predicted_class = class_names[np.argmax(predictions[0])]

        # Get the true label from the image path
        true_label = os.path.basename(os.path.dirname(str(image_path)))
```

```
        # plt.figure()
        axes[idx].imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
        axes[idx].set_title(f"Pred: {predicted_class}, Label : {true_label} ")
        axes[idx].axis("off")
    plt.tight_layout
    plt.show()

predict_random_images()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 105ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 36ms/step
```

Pred: tulips, Label : roses

Pred: dandelion, Label : sunflowers

Pred: tulips, Label : roses

Pred: tulips, Label : daisy

Pred: dandelion, Label : dandelion

Pred: tulips, Label : tulips

Pred: tulips, Label : dandelion

Pred: dandelion, Label : dandelion

Pred: roses, Label : roses

Pred: dandelion, Label : daisy

Pred: roses, Label : tulips

Pred: dandelion, Label : dandelion

Pred: dandelion, Label : dandelion

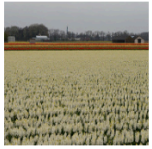
Pred: daisy, Label : daisy

Pred: tulips, Label : sunflowers

Pred: daisy, Label : daisy

Pred: tulips, Label : sunflowers

Pred: roses, Label : tulips



Pred: tulips, Label : sunflowers

Pred: tulips, Label : tulips

```
from sklearn.metrics import classification_report

predict = model_aug.predict(val_ds)
predict = np.argmax(predict, axis=1)
true = np.concatenate([y for x, y in val_ds], axis=0)
print(classification_report(true, predict, target_names=class_name))
```

```
23/23 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step
                precision    recall  f1-score   support

         daisy       0.74      0.68      0.71       129
     dandelion       0.79      0.81      0.80       176
         roses       0.70      0.70      0.70       120
     sunflowers       0.87      0.76      0.81       152
        tulips       0.67      0.78      0.72       157

      accuracy                           0.75       734
     macro avg       0.76      0.75      0.75       734
  weighted avg       0.76      0.75      0.75       734
```

```
# X = []
# y = []
# for flowers_name, images in flowers_images_dict.items():
#     for image in images:
#         img = cv2.imread(str(image))
#         resized_img = cv2.resize(img, (180, 180))
#         X.append(resized_img)
#         y.append(flowers_labels_dict[flowers_name])


# X = np.array(X)
# y = np.array(y)
```