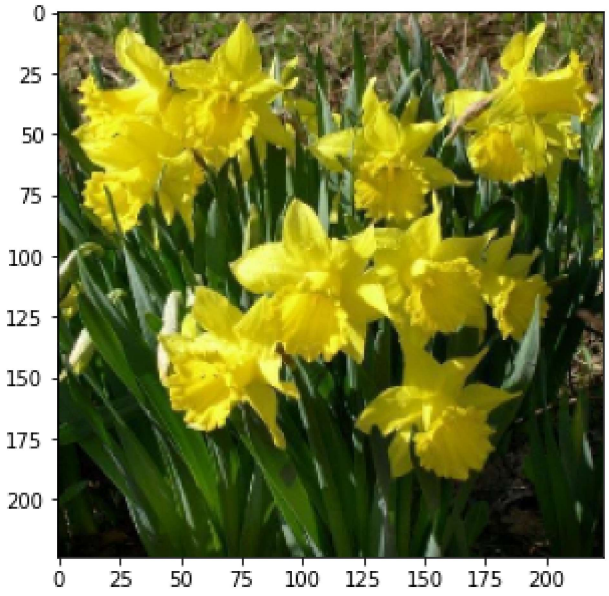
```
import numpy as np
import pandas as pd
import cv2
import os
import tensorflow as tf
import tarfile
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
from keras.applications.vgg16 import VGG16
from keras import layers
from keras import optimizers
from keras import models
from keras.preprocessing.image import ImageDataGenerator



import pathlib
dataset_url = "https://www.robots.ox.ac.uk/~vgg/data/flowers/17/17flowers.tgz"
data_dir = tf.keras.utils.get_file(origin=dataset_url,
                                   fname='flower_photos',
                                   untar=True)
data_dir = pathlib.Path(data_dir)


classes = [ i for i in range(0,17) ]
y = np.repeat(classes, 80)
# one-hot encoding the 17 categories
y = tf.keras.utils.to_categorical(y, num_classes=17, dtype='float32')


import os
import cv2
import numpy as np

images = []
loc = os.path.expanduser('/content/sample_data/flower.jpg')

for filename in sorted(os.listdir(loc)):

    img = cv2.imread(os.path.join(loc,filename))

    if img is not None:
        img = cv2.resize(img, dsize=(224, 224))
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        images.append(img)

images = np.asarray(images)
```

```
import matplotlib.pyplot as plt
plt.figure(figsize = (5,5))
plt.imshow(images[79])
```

```
<matplotlib.image.AxesImage at 0x7fed91192150>
```




```
images.shape
```

```
(1360, 224, 224, 3)
```

```
X_train, X_val, y_train, y_val = train_test_split(images, y, test_size=0.2, random_state=0)
```

```
conv_base = VGG16(weights='imagenet',include_top=False, input_shape=(224, 224, 3))
conv_base.trainable = False
model = models.Sequential()
model.add(conv_base)
model.add(layers.Conv2D(filters=64,kernel_size=(3,3),padding="same", activation="relu"))
model.add(layers.Conv2D(filters=64,kernel_size=(3,3),padding="same", activation="relu"))
model.add(layers.MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(layers.Conv2D(filters=64,kernel_size=(3,3),padding="same", activation="relu"))
model.add(layers.Conv2D(filters=64,kernel_size=(3,3),padding="same", activation="relu"))
model.add(layers.MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(layers.Flatten())
model.add(layers.Dropout(0.2))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(17, activation='softmax'))
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
```

```
=================================================================
 vgg16 (Functional)          (None, 7, 7, 512)         14714688

 conv2d_4 (Conv2D)           (None, 7, 7, 64)          294976

 conv2d_5 (Conv2D)           (None, 7, 7, 64)          36928

 max_pooling2d_2 (MaxPooling  (None, 3, 3, 64)         0
 2D)

 conv2d_6 (Conv2D)           (None, 3, 3, 64)          36928

 conv2d_7 (Conv2D)           (None, 3, 3, 64)          36928

 max_pooling2d_3 (MaxPooling  (None, 1, 1, 64)         0
 2D)

 flatten_1 (Flatten)         (None, 64)                0

 dropout_1 (Dropout)         (None, 64)                0

 dense_3 (Dense)             (None, 512)               33280

 dense_4 (Dense)             (None, 512)               262656

 dense_5 (Dense)             (None, 17)                8721

=================================================================
Total params: 15,425,105
Trainable params: 710,417
Non-trainable params: 14,714,688
_________________________________________________________________
```

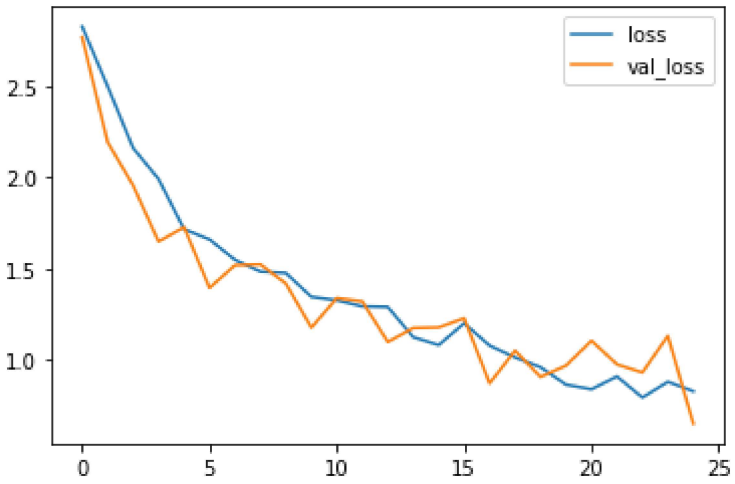
```
train_datagen = ImageDataGenerator (rescale=1./255,
                                    featurewise_std_normalization=True,
                                    zoom_range=0.2,
                                    rotation_range=90,
                                    width_shift_range=0.2,
                                    height_shift_range=0.2,
                                    horizontal_flip=True,
                                    fill_mode='nearest')

val_datagen = ImageDataGenerator (rescale=1./255, featurewise_std_normalization=True)

train_generator = train_datagen.flow(X_train, y_train, batch_size=32)
val_generator = val_datagen.flow(X_val, y_val, batch_size=32)


batch_size = 50
ntrain = len(X_train)
nval = len(X_val)
model.compile(
    optimizer=tf.keras.optimizers.Adam(epsilon=0.00001),
    loss='categorical_crossentropy',
    metrics=['accuracy'])
```

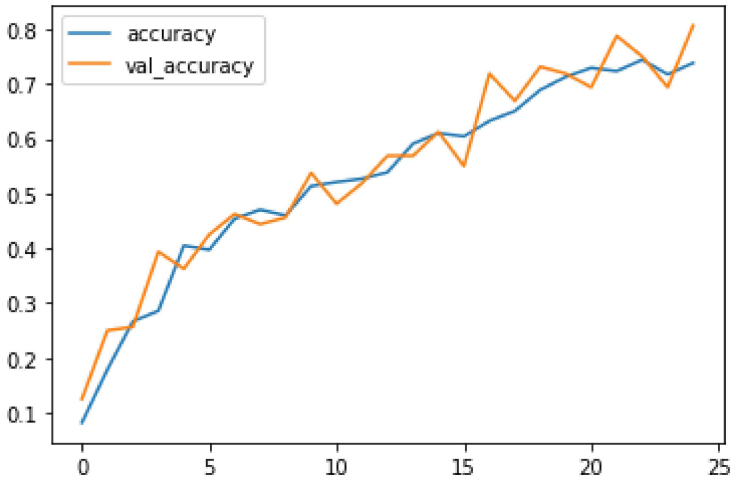
```
        metrics=['accuracy']
    )
    history = model.fit_generator(
        train_generator,
        validation_data=val_generator,
        epochs=25,
        steps_per_epoch = ntrain // batch_size,
        validation_steps = nval // batch_size
    )
```

```
Epoch 1/25
21/21 [==============================] - 423s 20s/step - loss: 2.8291 - accuracy: 0.081
Epoch 2/25
21/21 [==============================] - 420s 20s/step - loss: 2.5008 - accuracy: 0.178
Epoch 3/25
21/21 [==============================] - 421s 20s/step - loss: 2.1631 - accuracy: 0.266
Epoch 4/25
21/21 [==============================] - 421s 20s/step - loss: 1.9929 - accuracy: 0.285
Epoch 5/25
21/21 [==============================] - 420s 20s/step - loss: 1.7177 - accuracy: 0.404
Epoch 6/25
21/21 [==============================] - 420s 20s/step - loss: 1.6609 - accuracy: 0.397
Epoch 7/25
21/21 [==============================] - 418s 20s/step - loss: 1.5496 - accuracy: 0.453
Epoch 8/25
21/21 [==============================] - 418s 20s/step - loss: 1.4851 - accuracy: 0.470
Epoch 9/25
21/21 [==============================] - 418s 20s/step - loss: 1.4772 - accuracy: 0.459
Epoch 10/25
21/21 [==============================] - 418s 20s/step - loss: 1.3462 - accuracy: 0.513
Epoch 11/25
21/21 [==============================] - 418s 20s/step - loss: 1.3274 - accuracy: 0.520
Epoch 12/25
21/21 [==============================] - 419s 20s/step - loss: 1.2942 - accuracy: 0.526
Epoch 13/25
21/21 [==============================] - 420s 20s/step - loss: 1.2909 - accuracy: 0.538
Epoch 14/25
21/21 [==============================] - 420s 20s/step - loss: 1.1249 - accuracy: 0.590
Epoch 15/25
21/21 [==============================] - 419s 20s/step - loss: 1.0816 - accuracy: 0.610
Epoch 16/25
21/21 [==============================] - 419s 20s/step - loss: 1.2013 - accuracy: 0.604
Epoch 17/25
21/21 [==============================] - 419s 20s/step - loss: 1.0786 - accuracy: 0.632
Epoch 18/25
21/21 [==============================] - 419s 20s/step - loss: 1.0135 - accuracy: 0.650
Epoch 19/25
21/21 [==============================] - 419s 20s/step - loss: 0.9606 - accuracy: 0.689
Epoch 20/25
21/21 [==============================] - 419s 20s/step - loss: 0.8645 - accuracy: 0.712
Epoch 21/25
21/21 [==============================] - 418s 20s/step - loss: 0.8393 - accuracy: 0.729
Epoch 22/25
21/21 [==============================] - 419s 20s/step - loss: 0.9096 - accuracy: 0.723
Epoch 23/25
21/21 [==============================] - 419s 20s/step - loss: 0.7936 - accuracy: 0.744
Epoch 24/25
```

```
21/21 [==============================] - 419s 20s/step - loss: 0.8803 - accuracy: 0.717
Epoch 25/25
21/21 [==============================] - 419s 20s/step - loss: 0.8285 - accuracy: 0.738
```

```
history_frame = pd.DataFrame(history.history)
history_frame.loc[:, ['loss', 'val_loss']].plot()
history_frame.loc[:, ['accuracy', 'val_accuracy']].plot();
```





✓ 0s completed at 1:14 PM