```
#3 CNN


import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import PIL.Image
import tensorflow as tf
import tensorflow_datasets as tfds
import pathlib


dataset_url = "https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz"
data_dir = tf.keras.utils.get_file(origin=dataset_url, fname='flower_photos', untar=True)
data_dir = pathlib.Path(data_dir)


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

```
3670
```

```
roses = list(data_dir.glob('roses/*'))
#PIL.Image.open(str(roses[0]))
PIL.Image.open(str(roses[1]))
```

```
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)

test_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)

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

```
Found 3670 files belonging to 5 classes.
Using 2936 files for training.
Found 3670 files belonging to 5 classes.
Using 734 files for validation.
['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']
```

```
for image_batch, labels_batch in train_ds:
    print(image_batch.shape)
```

```
    print(labels_batch.shape)
    break
```

```
(32, 180, 180, 3)
(32,)
```

```
#CNN
num_classes  =  len(class_names)

model  =  tf.keras.models.Sequential(
       [
              tf.keras.Input(shape=(180,  180,  3)),
              tf.keras.layers.Conv2D(32,  kernel_size=(3,  3),  activation="relu"),
              tf.keras.layers.MaxPooling2D(pool_size=(2,  2)),
              tf.keras.layers.Conv2D(64,  kernel_size=(3,  3),  activation="relu"),
              tf.keras.layers.MaxPooling2D(pool_size=(2,  2)),
              tf.keras.layers.Flatten(),
              tf.keras.layers.Dropout(0.5),
              tf.keras.layers.Dense(num_classes,  activation="softmax"),
       ]
)

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 178, 178, 32)      896

 max_pooling2d (MaxPooling2D  (None, 89, 89, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 87, 87, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 43, 43, 64)       0
 2D)

 flatten (Flatten)           (None, 118336)            0

 dropout (Dropout)           (None, 118336)            0

 dense (Dense)               (None, 5)                 591685

=================================================================
Total params: 611,077
Trainable params: 611,077
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(optimizer="adam",  loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),  metrics=["accuracy"])

history=model.fit(train_ds,  validation_data=test_ds,  batch_size=batch_size,  epochs=10,  verbose=1,)
```

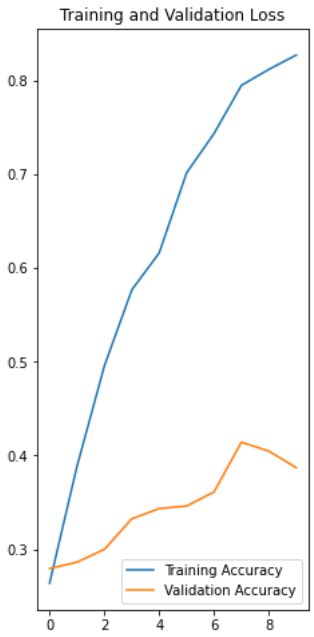
```
Epoch 1/10
/usr/local/lib/python3.8/dist-packages/tensorflow/python/util/dispatch.py:1082: UserWarning: "`sparse_categorical_crossentropy` received `from_logits
  return dispatch_target(*args, **kwargs)
92/92 [==============================] - 145s 2s/step - loss: 59.4540 - accuracy: 0.2636 - val_loss: 1.6245 - val_accuracy: 0.2793
Epoch 2/10
92/92 [==============================] - 143s 2s/step - loss: 1.4513 - accuracy: 0.3883 - val_loss: 1.6282 - val_accuracy: 0.2861
Epoch 3/10
92/92 [==============================] - 142s 2s/step - loss: 1.2586 - accuracy: 0.4956 - val_loss: 1.7989 - val_accuracy: 0.2997
Epoch 4/10
92/92 [==============================] - 143s 2s/step - loss: 1.0864 - accuracy: 0.5766 - val_loss: 1.9552 - val_accuracy: 0.3324
Epoch 5/10
92/92 [==============================] - 150s 2s/step - loss: 1.0023 - accuracy: 0.6161 - val_loss: 2.3140 - val_accuracy: 0.3433
Epoch 6/10
92/92 [==============================] - 155s 2s/step - loss: 0.8285 - accuracy: 0.7016 - val_loss: 2.3446 - val_accuracy: 0.3460
Epoch 7/10
92/92 [==============================] - 146s 2s/step - loss: 0.7328 - accuracy: 0.7435 - val_loss: 2.3810 - val_accuracy: 0.3610
Epoch 8/10
92/92 [==============================] - 144s 2s/step - loss: 0.6385 - accuracy: 0.7950 - val_loss: 2.7168 - val_accuracy: 0.4142
Epoch 9/10
92/92 [==============================] - 143s 2s/step - loss: 0.5895 - accuracy: 0.8120 - val_loss: 3.0230 - val_accuracy: 0.4046
Epoch 10/10
92/92 [==============================] - 141s 2s/step - loss: 0.5288 - accuracy: 0.8273 - val_loss: 3.0149 - val_accuracy: 0.3869
```

```
accuracy = history.history["accuracy"]
val_accuracy = history.history["val_accuracy"]

loss = history.history["loss"]
val_loss = history.history["val_loss"]

range_epochs = range(10)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(range_epochs, accuracy, label="Training Accuracy")
plt.plot(range_epochs, val_accuracy, label="Validation Accuracy")
plt.legend(loc="lower right")
plt.title("Training and Validation Loss")
plt.show()
```




```
plt.plot(history.history['val_accuracy'])
plt.plot(history.history['accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```