#!/usr/bin/env python  
# -\*- coding: utf-8 -\*-

# CNN

## Flower Data

###-----------------  
### Import Libraries  
###-----------------  
  
import os  
import numpy as np  
import pandas as pd  
  
import matplotlib.pyplot as plt  
  
import tensorflow as tf  
  
from utils.helper import fn\_plot\_tf\_hist,fn\_plot\_confusion\_matrix

WARNING:tensorflow:From C:\Users\Administrator.DAI-PC2\anaconda3\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse\_softmax\_cross\_entropy is deprecated. Please use tf.compat.v1.losses.sparse\_softmax\_cross\_entropy instead.

###----------------------  
### Some basic parameters  
###----------------------  
inpDir = '../..\Classwork/input'  
outDir = './output'  
subDir = 'flower\_photos'  
modelDir = './models'  
logDir = './logs'  
altName = 'cnn\_base'  
  
RANDOM\_STATE = 24 # for initialization ----- REMEMBER: to remove at the time of promotion to production  
tf.random.set\_seed(RANDOM\_STATE) # setting for Tensorflow as well  
  
TEST\_SIZE = 0.2  
  
ALPHA = 0.001  
EPOCHS = 2 # number of cycles to run  
PATIENCE = 50  
LR\_PATIENCE = 20  
FACTOR\_LR = 0.1  
BATCH\_SIZE = 32 # inline of Training Rows being 60000  
IMG\_HEIGHT = 190  
IMG\_WIDTH = 190  
  
  
# Set parameters for decoration of plots  
params = {'legend.fontsize' : 'large',  
 'figure.figsize' : (15,10),  
 'axes.labelsize' : 'x-large',  
 'axes.titlesize' :'x-large',  
 'xtick.labelsize' :'large',  
 'ytick.labelsize' :'large',  
 }  
  
CMAP = plt.cm.coolwarm  
  
plt.rcParams.update(params) # update rcParams  
  
plt.style.use('seaborn-v0\_8-darkgrid') # plt.style.use('ggplot')

## Basic Hygiene

physical\_devices = tf.config.list\_physical\_devices('GPU')  
  
if len(physical\_devices) > 0:  
 tf.config.experimental.set\_memory\_growth(physical\_devices[0], True)

print (physical\_devices)

[]

## Import data

'''  
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)  
  
'''  
data\_dir = os.path.join(inpDir, subDir)  
data\_dir

'../..\\Classwork/input\\flower\_photos'

os.listdir(data\_dir)

['daisy', 'dandelion', 'LICENSE.txt', 'roses', 'sunflowers', 'tulips']

## Creating datasets

# create training data  
train\_ds =tf.keras.preprocessing.image\_dataset\_from\_directory(  
 data\_dir, # path the the data directory  
 validation\_split=TEST\_SIZE, # what ratio of validation data  
 subset='training', # purpose  
 seed=RANDOM\_STATE,  
 image\_size=[IMG\_HEIGHT, IMG\_WIDTH], ## @@@ WHAT!  
 batch\_size=BATCH\_SIZE  
)  
# test data  
test\_ds =tf.keras.preprocessing.image\_dataset\_from\_directory(  
 data\_dir, # path the the data directory  
 validation\_split=TEST\_SIZE, # what ratio of validation data  
 subset='validation', # purpose  
 seed=RANDOM\_STATE,  
 image\_size=[IMG\_HEIGHT, IMG\_WIDTH], ## @@@ WHAT!  
 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.

# is it picking class names  
class\_names = train\_ds.class\_names  
class\_names

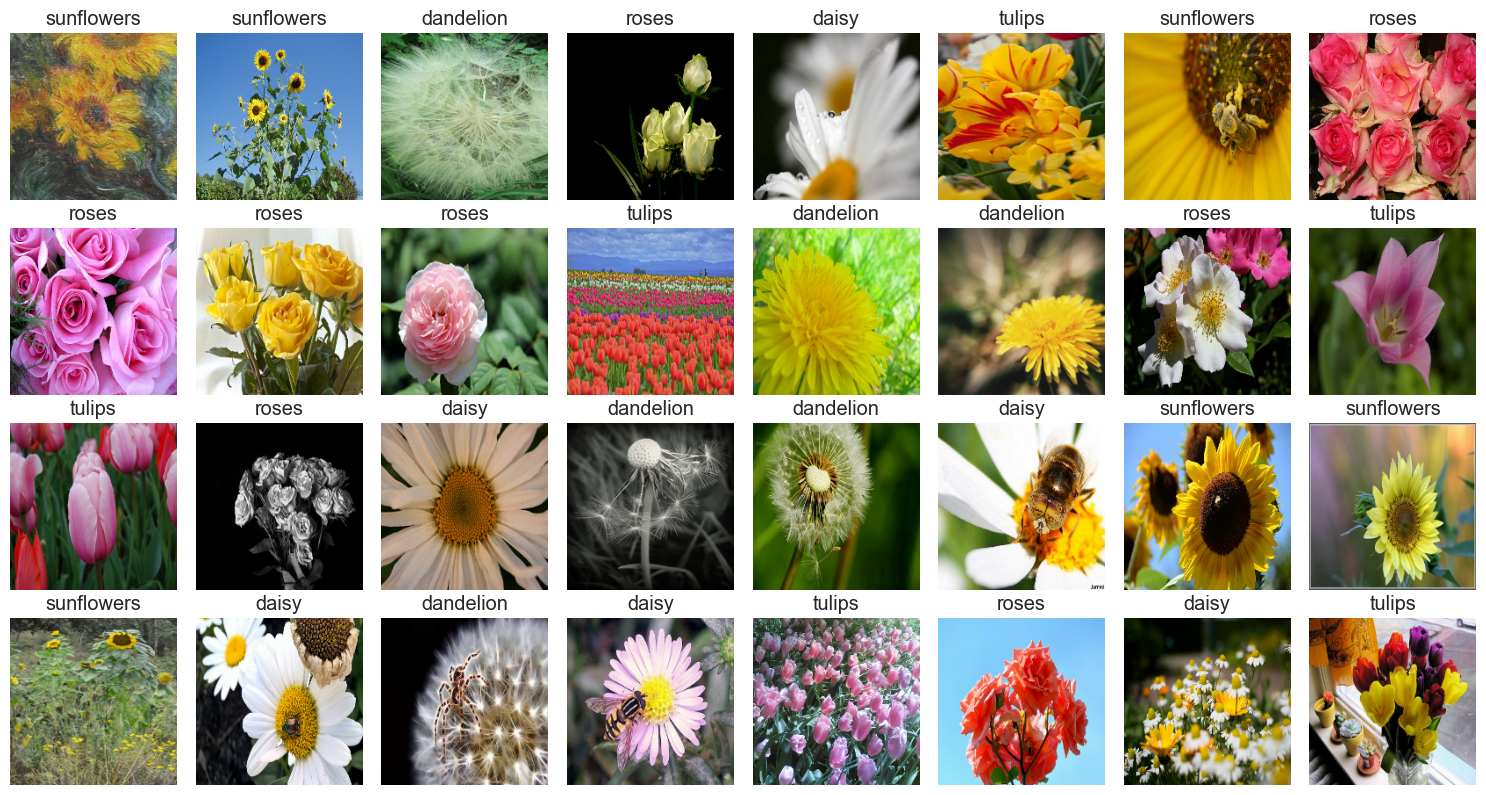
['daisy', 'dandelion', 'roses', 'sunflowers', 'tulips']

class\_dict = {k:v for k,v in enumerate(class\_names)}  
class\_dict

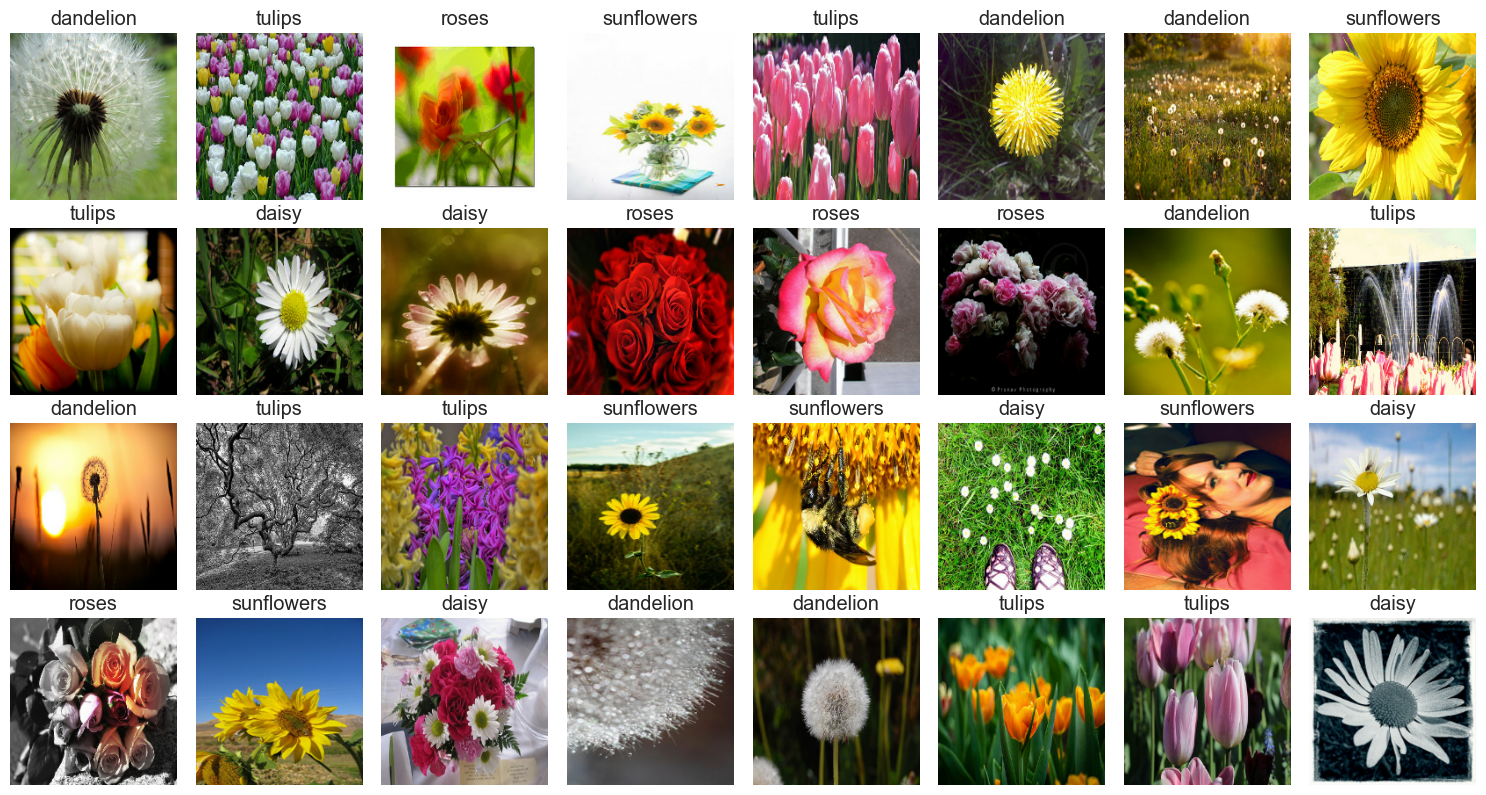
{0: 'daisy', 1: 'dandelion', 2: 'roses', 3: 'sunflowers', 4: 'tulips'}

## Visualize data in train\_ds and test\_ds

plt.figure(figsize=(15,8))  
  
for images, labels in train\_ds.take(1):  
 for i in range (BATCH\_SIZE):  
 plt.subplot(int(BATCH\_SIZE/8), 8, i +1)  
 plt.grid(False)  
 plt.imshow(images[i].numpy().astype('uint8'))  
 plt.title(class\_names[labels[i]])  
 plt.axis('off')  
 plt.tight\_layout()  
plt.show()



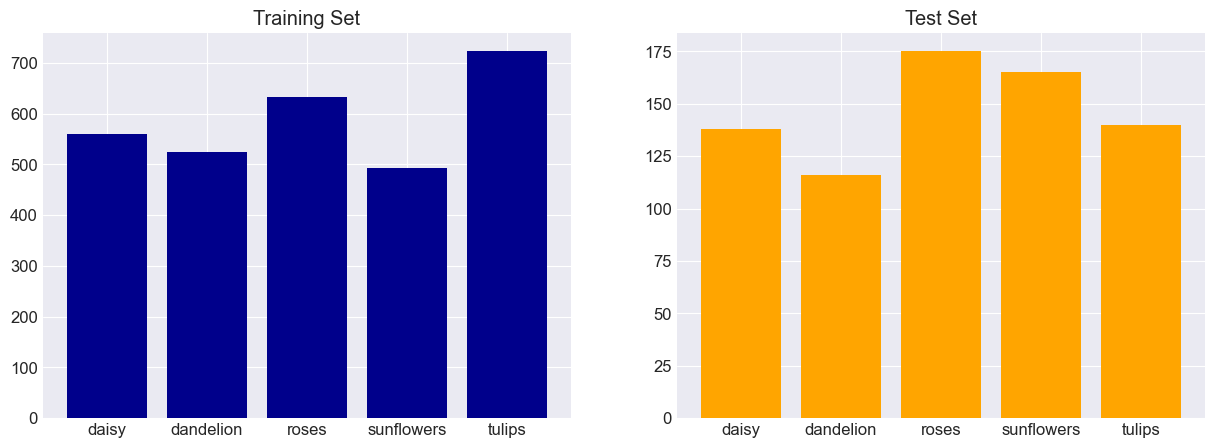
plt.figure(figsize=(15,8))  
  
for images, labels in test\_ds.take(1): # get me one batch  
  
 for i in range (BATCH\_SIZE): # loop over batch  
  
 plt.subplot(int(BATCH\_SIZE/8), 8, i +1) # access the axis  
  
 plt.grid(False) # no to grid  
  
 plt.imshow(images[i].numpy().astype('uint8')) # show image convert to numpy and int  
  
 plt.title(class\_names[labels[i]])  
  
 plt.axis('off')  
  
 plt.tight\_layout()  
  
plt.show()



## To check whether data is balanced or not

def fn\_plot\_label(tr\_ds, ts\_ds):  
  
 plt.figure(figsize = (15,5)) # instantiate the figure  
  
 plt.subplot(1,2,1) # first out of 2  
  
 train\_labels = tf.concat([lbl for img, lbl in tr\_ds], axis = 0).numpy() # get the labels  
  
 unique, \_, counts = tf.unique\_with\_counts(train\_labels) # get counts  
  
 plt.bar(range(len(unique)), counts, align='center', color = 'DarkBlue') # barplot the counts  
  
 plt.xticks(range(len(unique)), class\_names)  
  
 plt.title('Training Set')  
  
 plt.subplot(1,2,2)  
  
 test\_labels = tf.concat([lbl for img, lbl in ts\_ds], axis = 0).numpy()  
  
 unique, \_, counts = tf.unique\_with\_counts(test\_labels)  
  
 plt.bar(range(len(unique)), counts, align='center', color = 'Orange')  
  
 plt.xticks(range(len(unique)), class\_names)  
  
 plt.title('Test Set')

fn\_plot\_label(train\_ds, test\_ds)



## Model Building

from last conv layer to input layer

op size 2\*2

1. conv layer f = 3, stride (s) =1 ip size = 4\*4
2. maxpool layer f = 2,2, stride (s) =2 ip size = 8\*8
3. conv layer f = 3, stride (s) =1 ip size = 10\*10
4. maxpool layer f = 2,2, stride (s) =2 ip size = 20\*20
5. conv layer f = 3, stride (s) =1 ip size = 22\*22
6. maxpool layer f = 2,2, stride (s) =2 ip size = 44\*44
7. conv layer f = 3, stride (s) =1 ip size = 46\*46
8. maxpool layer f = 2,2, stride (s) =2 ip size = 92\*92
9. conv layer f = 3, stride (s) =1 ip size = 94\*94
10. maxpool layer f = 2,2, stride (s) =2 ip size = 188\*188
11. conv layer f = 3, stride (s) =1 ip size = 190\*190 (image size)

6 conv layers

input\_shape = (IMG\_HEIGHT, IMG\_WIDTH, 3)  
num\_classes = len(class\_names)  
input\_shape, num\_classes

((190, 190, 3), 5)

def build\_model (input\_shape, num\_classes):  
  
 krnl\_initializer = tf.keras.initializers.GlorotUniform()  
  
 model = tf.keras.Sequential()  
  
 ## preprocessing (scaling)  
 model.add(tf.keras.layers.Rescaling(1./255.))  
  
 ## 1 layer  
 model.add(tf.keras.layers.Conv2D(32,(3,3),  
 kernel\_initializer = krnl\_initializer,  
 activation = 'relu',  
 input\_shape =input\_shape)) ## output shape expected - 188\*188\*32  
  
 model.add(tf.keras.layers.MaxPool2D(pool\_size=(2,2))) ## 94\*94\*32  
  
 ## 2 layer  
 model.add(tf.keras.layers.Conv2D(64,3,  
 kernel\_initializer = krnl\_initializer,  
 activation = 'relu')) ## output shape expected - 92\*92\*64  
  
 model.add(tf.keras.layers.MaxPool2D(pool\_size=(2,2))) ## 46\*46\*64  
  
 ## 3 layer  
 model.add(tf.keras.layers.Conv2D(128,(3,3),  
 kernel\_initializer = krnl\_initializer,  
 activation = 'relu')) ## output shape expected - 44\*44\*128  
  
 model.add(tf.keras.layers.MaxPool2D(pool\_size=(2,2))) ## 22\*22\*128  
  
 ## 4 layer  
 model.add(tf.keras.layers.Conv2D(256,(3,3),  
 kernel\_initializer = krnl\_initializer,  
 activation = 'relu')) ## output shape expected - 20\*20\*256  
  
 model.add(tf.keras.layers.MaxPool2D(pool\_size=(2,2))) ## 10\*10\*256  
  
 ## 5 layer  
 model.add(tf.keras.layers.Conv2D(512,(3,3),  
 kernel\_initializer = krnl\_initializer,  
 activation = 'relu')) ## output shape expected - 8\*8\*512  
  
 model.add(tf.keras.layers.MaxPool2D(pool\_size=(2,2))) ## 4\*4\*512  
  
 ## 6 layer  
 model.add(tf.keras.layers.Conv2D(1024,(3,3),  
 kernel\_initializer = krnl\_initializer,  
 activation = 'relu')) ## output shape expected - 2\*2\*1024  
  
 ## Head  
 model.add(tf.keras.layers.Flatten())  
 model.add(tf.keras.layers.Dense(1024, activation = 'relu',kernel\_initializer = krnl\_initializer))  
 model.add(tf.keras.layers.Dense(256, activation = 'relu',kernel\_initializer = krnl\_initializer))  
 model.add(tf.keras.layers.Dense(64, activation = 'relu',kernel\_initializer = krnl\_initializer))  
 model.add(tf.keras.layers.Dense(16, activation = 'relu',kernel\_initializer = krnl\_initializer))  
 model.add(tf.keras.layers.Dense(num\_classes))  
  
 return model

model = build\_model(input\_shape, num\_classes)  
model

WARNING:tensorflow:From C:\Users\Administrator.DAI-PC2\anaconda3\Lib\site-packages\keras\src\backend.py:873: The name tf.get\_default\_graph is deprecated. Please use tf.compat.v1.get\_default\_graph instead.  
  
WARNING:tensorflow:From C:\Users\Administrator.DAI-PC2\anaconda3\Lib\site-packages\keras\src\layers\pooling\max\_pooling2d.py:161: The name tf.nn.max\_pool is deprecated. Please use tf.nn.max\_pool2d instead.

<keras.src.engine.sequential.Sequential at 0x166a4dbb5d0>

checkpoint\_path = './weights\_tf\_flower'  
  
model\_checkpoint = tf.keras.callbacks.ModelCheckpoint(  
 filepath = checkpoint\_path,  
 monitor='val\_loss',  
 verbose=2,  
 save\_best\_only=True,  
 save\_weights\_only=True,  
 mode='auto',  
 save\_freq='epoch',  
 initial\_value\_threshold=None  
)  
  
es\_callback = tf.keras.callbacks.EarlyStopping(  
 monitor='val\_loss',  
 min\_delta=0,  
 patience=PATIENCE,  
 verbose=2,  
 mode='auto',  
 baseline=None,  
 restore\_best\_weights=True,  
 start\_from\_epoch=0  
)  
  
lr\_callback = tf.keras.callbacks.ReduceLROnPlateau(  
 monitor='val\_loss',  
 factor=FACTOR\_LR,  
 patience=LR\_PATIENCE,  
 verbose=2,  
 mode='auto',  
 min\_delta=0.000001,  
 cooldown=0,  
 min\_lr=0.0,  
)

## Compile and train

optimizer = tf.keras.optimizers.Adam(learning\_rate=ALPHA)  
  
model.compile(optimizer = optimizer,  
 loss = tf.keras.losses.SparseCategoricalCrossentropy(from\_logits=True),  
 metrics = ['accuracy'])

history = model.fit(train\_ds,  
 validation\_data = test\_ds ,  
 batch\_size = BATCH\_SIZE,  
 epochs = EPOCHS, verbose=2,  
 callbacks=[model\_checkpoint,es\_callback,lr\_callback])

Epoch 1/2

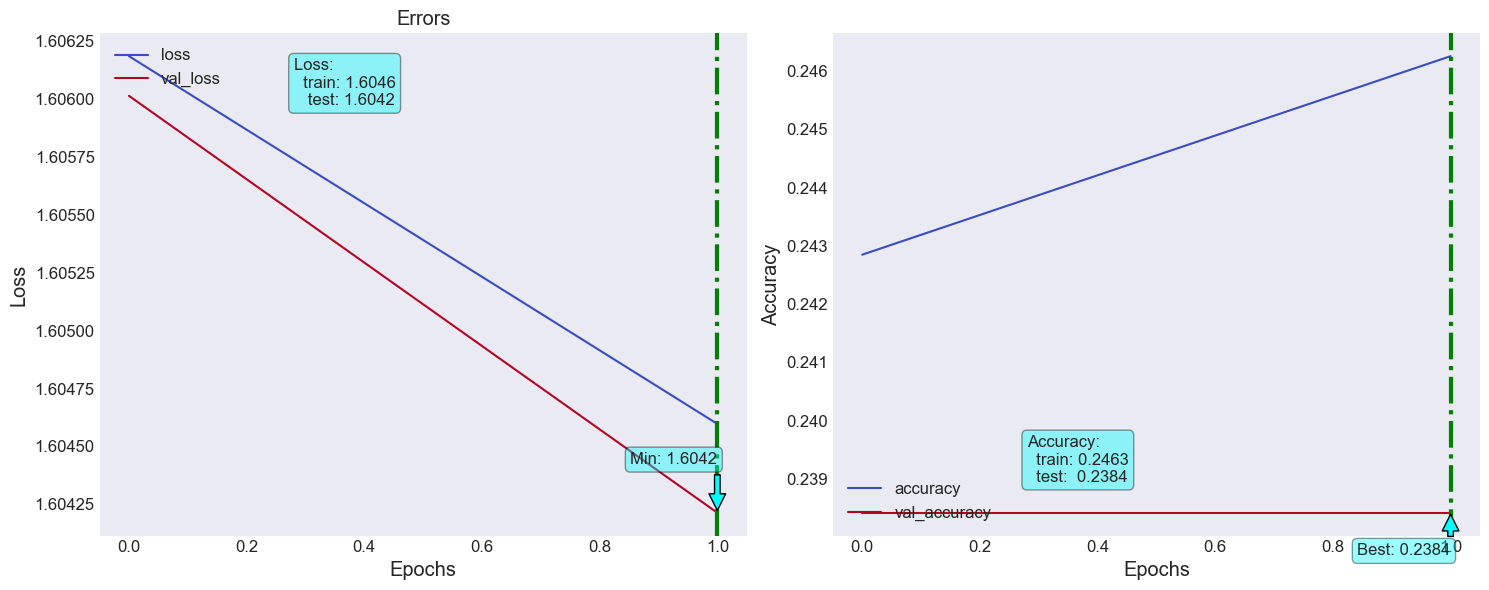
C:\Users\Administrator.DAI-PC2\anaconda3\Lib\site-packages\keras\src\initializers\initializers.py:120: UserWarning: The initializer GlorotUniform is unseeded and being called multiple times, which will return identical values each time (even if the initializer is unseeded). Please update your code to provide a seed to the initializer, or avoid using the same initializer instance more than once.  
 warnings.warn(

WARNING:tensorflow:From C:\Users\Administrator.DAI-PC2\anaconda3\Lib\site-packages\keras\src\utils\tf\_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.  
  
WARNING:tensorflow:From C:\Users\Administrator.DAI-PC2\anaconda3\Lib\site-packages\keras\src\engine\base\_layer\_utils.py:384: The name tf.executing\_eagerly\_outside\_functions is deprecated. Please use tf.compat.v1.executing\_eagerly\_outside\_functions instead.  
  
  
Epoch 1: val\_loss improved from inf to 1.60601, saving model to .\weights\_tf\_flower  
92/92 - 56s - loss: 1.6062 - accuracy: 0.2428 - val\_loss: 1.6060 - val\_accuracy: 0.2384 - lr: 0.0010 - 56s/epoch - 604ms/step  
Epoch 2/2  
  
Epoch 2: val\_loss improved from 1.60601 to 1.60421, saving model to .\weights\_tf\_flower  
92/92 - 50s - loss: 1.6046 - accuracy: 0.2463 - val\_loss: 1.6042 - val\_accuracy: 0.2384 - lr: 0.0010 - 50s/epoch - 546ms/step

model.summary()

Model: "sequential"  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 Layer (type) Output Shape Param #   
=================================================================  
 rescaling (Rescaling) (None, 190, 190, 3) 0   
   
 conv2d (Conv2D) (None, 188, 188, 32) 896   
   
 max\_pooling2d (MaxPooling2 (None, 94, 94, 32) 0   
 D)   
   
 conv2d\_1 (Conv2D) (None, 92, 92, 64) 18496   
   
 max\_pooling2d\_1 (MaxPoolin (None, 46, 46, 64) 0   
 g2D)   
   
 conv2d\_2 (Conv2D) (None, 44, 44, 128) 73856   
   
 max\_pooling2d\_2 (MaxPoolin (None, 22, 22, 128) 0   
 g2D)   
   
 conv2d\_3 (Conv2D) (None, 20, 20, 256) 295168   
   
 max\_pooling2d\_3 (MaxPoolin (None, 10, 10, 256) 0   
 g2D)   
   
 conv2d\_4 (Conv2D) (None, 8, 8, 512) 1180160   
   
 max\_pooling2d\_4 (MaxPoolin (None, 4, 4, 512) 0   
 g2D)   
   
 conv2d\_5 (Conv2D) (None, 2, 2, 1024) 4719616   
   
 flatten (Flatten) (None, 4096) 0   
   
 dense (Dense) (None, 1024) 4195328   
   
 dense\_1 (Dense) (None, 256) 262400   
   
 dense\_2 (Dense) (None, 64) 16448   
   
 dense\_3 (Dense) (None, 16) 1040   
   
 dense\_4 (Dense) (None, 5) 85   
   
=================================================================  
Total params: 10763493 (41.06 MB)  
Trainable params: 10763493 (41.06 MB)  
Non-trainable params: 0 (0.00 Byte)  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

fn\_plot\_tf\_hist(pd.DataFrame(history.history))



## Augmentation

layer = tf.keras.layers.RandomRotation((-0.5,0.5), fill\_mode = 'nearest', seed=RANDOM\_STATE)  
  
img\_num= 0  
  
plt.figure()  
  
for images,labels in train\_ds.take(1):  
   
 out\_img = layer(images)  
   
 plt.subplot(1,2,1)  
 plt.title('Original')  
 plt.imshow(images[img\_num].numpy().astype('uint16'))  
 plt.grid(False)  
  
 plt.subplot(1,2,2)  
 plt.title('Rotated')  
 plt.imshow(out\_img[img\_num].numpy().astype('uint16'))  
 plt.grid(False)

