

# ASSIGNMENT-3

## Built CNN Model for Classification Of Flowers

Assignment Date	03 October 2022
Student Name	MANIMARAN A
Student Roll Number	E1195028
Maximum Marks	2marks
Team ID	PNT2022TMID45335
Project	AI BASED DISCOURSE FOR BAKING INDUSTRY

### 1. Download the Dataset

```
!unzip Flowers-Dataset.zip

Archive:  Flowers-Dataset.zip
replace flowers/daisy/100080576_f52e8ee070_n.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace flowers/daisy/10140303196_b88d3d6cec.jpg? [y]es, [n]o, [A]ll, [N]one, [r]ename: A
  inflating: flowers/daisy/10140303196_b88d3d6cec.jpg
  inflating: flowers/daisy/10172379554_b296050f82_n.jpg
  inflating: flowers/daisy/10172567486_2748826a8b.jpg
  inflating: flowers/daisy/10172636503_21bededa75_n.jpg
  inflating: flowers/daisy/102841525_bd6628ae3c.jpg
  inflating: flowers/daisy/10300722094_28fa978807_n.jpg
  inflating: flowers/daisy/1031799732_e7f4008c03.jpg
  inflating: flowers/daisy/10391248763_1d16681106_n.jpg
  inflating: flowers/daisy/10437754174_22ec990b77_m.jpg
  inflating: flowers/daisy/10437770546_8bb6f7bdd3_m.jpg
  inflating: flowers/daisy/10437929963_bc13eebe0c.jpg
  inflating: flowers/daisy/10466290366_cc72e33532.jpg
  inflating: flowers/daisy/10466558316_a7198b87e2.jpg
  inflating: flowers/daisy/10555749515_13a12a026e.jpg
  inflating: flowers/daisy/10555815624_dc211569b0.jpg
  inflating: flowers/daisy/10555826524_423eb8bf71_n.jpg
  inflating: flowers/daisy/10559679065_50d2b16f6d.jpg
  inflating: flowers/daisy/105806915_a9c13e2106_n.jpg
  inflating: flowers/daisy/10712722853_5632165b04.jpg
  inflating: flowers/daisy/107592979_aaa9cdfef78_m.jpg
  inflating: flowers/daisy/10770585085_4742b9dac3_n.jpg
  inflating: flowers/daisy/10841136265_af473efc60.jpg
  inflating: flowers/daisy/10993710036_2033222c91.jpg
  inflating: flowers/daisy/10993818044_4c19b86c82.jpg
```

```
inflating: flowers/dandelion/9300335851_cdf1cef7a9.jpg
inflating: flowers/dandelion/9301891790_971dcfb35d_m.jpg
inflating: flowers/dandelion/9472854850_fc9e1db673.jpg
inflating: flowers/dandelion/9517326597_5d116a0166.jpg
inflating: flowers/dandelion/9533964635_f38e6fa3c3.jpg
inflating: flowers/dandelion/9595369280_dd88b61814.jpg
inflating: flowers/dandelion/9613826015_f345354874.jpg
inflating: flowers/dandelion/9617087594_ec2a9b16f6.jpg
inflating: flowers/dandelion/9646730031_f3d5014416_n.jpg
inflating: flowers/dandelion/9719816995_8f211abf02_n.jpg
inflating: flowers/dandelion/9726260379_4e8ee66875_m.jpg
inflating: flowers/dandelion/9759608055_9ab623d193.jpg
inflating: flowers/dandelion/9818247_e2eac18894.jpg
inflating: flowers/dandelion/9853885425_4a82356f1d_m.jpg
inflating: flowers/dandelion/98992760_53ed1d26a9.jpg
inflating: flowers/dandelion/9939430464_5f5861ebab.jpg
inflating: flowers/dandelion/9965757055_ff01b5ee6f_n.jpg
inflating: flowers/rose/10090824183_d02c613f10_m.jpg
inflating: flowers/rose/102501987_3cdb8e5394_n.jpg
inflating: flowers/rose/10503217854_e66a804309.jpg
inflating: flowers/rose/10894627425_ec76bbc757_n.jpg
inflating: flowers/rose/110472418_87b6a3aa98_m.jpg
inflating: flowers/rose/11102341464_508d558dfc_n.jpg
inflating: flowers/rose/11233672494_d8bf0a3dbf_n.jpg
inflating: flowers/rose/11694025703_9a906fedc1_n.jpg
inflating: flowers/rose/118974357_0faa23cce9_n.jpg
```

```
inflating: flowers/rose/9458445402_79e4dfa89c.jpg
inflating: flowers/rose/9609569441_eeb8566e94.jpg
inflating: flowers/rose/9614492283_66020fb4eb_n.jpg
inflating: flowers/rose/9633056561_6f1b7e8faf_m.jpg
inflating: flowers/rose/9702378513_229a96b754_m.jpg
inflating: flowers/rose/99383371_37a5ac12a3_n.jpg
inflating: flowers/sunflower/1008566138_6927679c8a.jpg
inflating: flowers/sunflower/1022552002_2b93faf9e7_n.jpg
inflating: flowers/sunflower/1022552036_67d33d5bd8_n.jpg
inflating: flowers/sunflower/10386503264_e05387e1f7_m.jpg
inflating: flowers/sunflower/10386522775_4f8c616999_m.jpg
inflating: flowers/sunflower/10386525005_fd0b7d6c55_n.jpg
inflating: flowers/sunflower/10386525695_2c38fea555_n.jpg
inflating: flowers/sunflower/10386540106_1431e73086_m.jpg
inflating: flowers/sunflower/10386540696_0a95ee53a8_n.jpg
inflating: flowers/sunflower/10386702973_e74a34c806_n.jpg
inflating: flowers/sunflower/1043442695_4556c4c13d_n.jpg
inflating: flowers/sunflower/1044296388_912143e1d4.jpg
inflating: flowers/sunflower/10541580714_ff6b171abd_n.jpg
inflating: flowers/sunflower/1064662314_c5a7891b9f_m.jpg
inflating: flowers/sunflower/10862313945_e8ed9202d9_m.jpg
inflating: flowers/sunflower/11881770944_22b4f2f8f6_n.jpg
inflating: flowers/sunflower/1217254584_4b3028b93d.jpg
inflating: flowers/sunflower/12282924083_fb80aa17d4_n.jpg
inflating: flowers/sunflower/12323859023_447387dbf0_n.jpg
inflating: flowers/sunflower/1240624822_4111dde542.jpg
inflating: flowers/sunflower/1240625276_fb3bd0c7b1.jpg
```

```
inflating: flowers/sunflower/9738732180_00c0cc53c0_n.jpg
inflating: flowers/sunflower/9783416751_b2a03920f7_n.jpg
inflating: flowers/sunflower/9825716455_f12bcc8d4e_n.jpg
inflating: flowers/sunflower/9904127656_f76a5a4811_m.jpg
inflating: flowers/tulip/100930342_92e8746431_n.jpg
inflating: flowers/tulip/10094729603_eeca3f2cb6.jpg
inflating: flowers/tulip/10094731133_94a942463c.jpg
inflating: flowers/tulip/10128546863_8de70c610d.jpg
inflating: flowers/tulip/10163955604_ae0b830975_n.jpg
inflating: flowers/tulip/10164073235_f29931d91e.jpg
inflating: flowers/tulip/10686568196_b1915544a8.jpg
inflating: flowers/tulip/107693873_86021ac4ea_n.jpg
inflating: flowers/tulip/10791227_7168491604.jpg
inflating: flowers/tulip/10995953955_089572caf0.jpg
inflating: flowers/tulip/110147301_ad921e2828.jpg
inflating: flowers/tulip/112334842_3ecf7585dd.jpg
inflating: flowers/tulip/112428665_d8f3632f36_n.jpg
inflating: flowers/tulip/112428919_f0c5ad7d9d_n.jpg
inflating: flowers/tulip/112650879_82adc2cc04_n.jpg
inflating: flowers/tulip/112651128_7b5d39a346_m.jpg
inflating: flowers/tulip/112951022_4892b1348b_n.jpg
inflating: flowers/tulip/112951086_150a59d499_n.jpg
inflating: flowers/tulip/113291410_1bdc718ed8_n.jpg
inflating: flowers/tulip/113902743_8f537f769b_n.jpg
```

```
inflating: flowers/tulip/8759597778_7fca5d434b_n.jpg
inflating: flowers/tulip/8759601388_36e2a50d98_n.jpg
inflating: flowers/tulip/8759606166_8e475013fa_n.jpg
inflating: flowers/tulip/8759618746_f5e39fdbf8_n.jpg
inflating: flowers/tulip/8762189906_8223cef62f.jpg
inflating: flowers/tulip/8762193202_0fbf2f6a81.jpg
inflating: flowers/tulip/8768645961_8f1e097170_n.jpg
inflating: flowers/tulip/8817622133_a42bb90e38_n.jpg
inflating: flowers/tulip/8838347159_746d14e6c1_m.jpg
inflating: flowers/tulip/8838354855_c474fc66a3_m.jpg
inflating: flowers/tulip/8838914676_8ef4db7f50_n.jpg
inflating: flowers/tulip/8838975946_f54194894e_m.jpg
inflating: flowers/tulip/8838983024_5c1a767878_n.jpg
inflating: flowers/tulip/8892851067_79242a7362_n.jpg
inflating: flowers/tulip/8904780994_8867d64155_n.jpg
inflating: flowers/tulip/8908062479_449200a1b4.jpg
inflating: flowers/tulip/8908097235_c3e746d36e_n.jpg
inflating: flowers/tulip/9019694597_2d3bbdb17.jpg
inflating: flowers/tulip/9030467406_05e93ff171_n.jpg
inflating: flowers/tulip/9048307967_40a164a459_m.jpg
inflating: flowers/tulip/924782410_94ed7913ca_m.jpg
inflating: flowers/tulip/9378657435_89fabf13c9_n.jpg
inflating: flowers/tulip/9444202147_405290415b_n.jpg
inflating: flowers/tulip/9446982168_06c4d71da3_n.jpg
inflating: flowers/tulip/9831362123_5aac525a99_n.jpg
inflating: flowers/tulip/9870557734_88eb3b9e3b_n.jpg
inflating: flowers/tulip/9947374414_fdf1d0861c_n.jpg
inflating: flowers/tulip/9947385346_3a8cacea02_n.jpg
inflating: flowers/tulip/9976515506_d496c5e72c.jpg
```

## Solution:

```
import numpy as np
```

```
import tensorflow as tf
```

```
from tensorflow.keras import layers
```

```
from tensorflow.keras.models import Sequential
```

```
import matplotlib.pyplot as plt
```

```
batch_size = 32
```

```
img_height = 180
```

```
data_dir = "/content/flowers"
```

```
import ImageDataGenerator
```

```
train_datagen = ImageDataGenerator(rescale = 1./255, horizontal_flip = True, vertical_flip = True,  
zoom_range = 0.2)
```

```
x_train = train_datagen.flow_from_directory(r"/content/flowers", target_size = (64,64) ,  
class_mode = "categorical", batch_size = 100)
```

```
[ ] import numpy as np  
import tensorflow as tf  
from tensorflow.keras import layers  
from tensorflow.keras.models import Sequential  
import matplotlib.pyplot as plt  
batch_size = 32  
img_height = 180  
img_width = 180  
data_dir = "/content/flowers"  
  
[ ] from tensorflow.keras.preprocessing.image import ImageDataGenerator  
  
[ ] train_datagen = ImageDataGenerator(rescale = 1./255, horizontal_flip = True, vertical_flip = True, zoom_range = 0.2)
```

```
[ ] x_train = train_datagen.flow_from_directory(r"/content/flowers", target_size = (64,64) , class_mode = "categorical", batch_size = 100)

Found 4317 images belonging to 5 classes.
```

## 2. Image Argumentation

**#Image Augmentation accuracy**

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

### 2. Image augmentation

```
[ ] #Image Augumentation accuracy
data_augmentation = Sequential(
    [
        layers.RandomFlip("horizontal",input_shape=(img_height, img_width, 3)),
        layers.RandomRotation(0.1),
        layers.RandomZoom(0.1),
    ]
)
```

### 3.Create Model

```
from tensorflow.keras.models import Sequential from tensorflow.keras.layers import  
Convolution2D,MaxPooling2D,Flatten,Dense
```

```
model = Sequential()
```

```
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)
```

Found 4317 files belonging to 5

classes. Using 3454 files for training.

```
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 4317 files belonging to 5 classes.

Using 863 files for validation.

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

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

plt.figure(figsize=(10, 10))

for images, labels in train_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
        plt.title(class_names[labels[i]])
        plt.axis("off")
```

### 3. Create Model

```
[ ] from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Convolution2D, MaxPooling2D, Flatten, Dense
    model = Sequential()
```

```
[ ] 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)
```

```
Found 4317 files belonging to 5 classes.
Using 3454 files for training.
```

```
[ ] 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 4317 files belonging to 5 classes.  
Using 863 files for validation.
```

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

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



```
[ ] plt.figure(figsize=(10, 10))
    for images, labels in train_ds.take(1):
        for i in range(9):
            ax = plt.subplot(3, 3, i + 1)
            plt.imshow(images[i].numpy().astype("uint8"))
            plt.title(class_names[labels[i]])
            plt.axis("off")
```

rose



tulip



tulip



dandelion



sunflower



daisy



dandelion



daisy



rose



## 4.Add Layers (Convolution,Max Pooling,Flatten,Dense-(Hidden Layer),Output)

### Solution:

```
model.add(Convolution2D(32, (3,3), activation = "relu", input_shape = (64,64,3) ))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Flatten())

model.add(Dense(300, activation = "relu"))
model.add(Dense(150, activation = "relu")) #multiple dense layers
model.add(Dense(5, activation = "softmax")) #output layer
#Adding the layers for accuracy
num_classes = len(class_names)

model = Sequential([
    data_augmentation,
    layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
    layers.Conv2D(16, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(32, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 3, padding='same', activation='relu'),
    layers.MaxPooling2D(),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
])
```

#### 4.Adding the layers (Convolution,MaxPooling,Flatten,Dense-(HiddenLayers),Output)

```
▶ model.add(Convolution2D(32, (3,3), activation = "relu", input_shape = (64,64,3) ))  
model.add(MaxPooling2D(pool_size = (2,2)))  
model.add(Flatten())  
model.add(Dense(300, activation = "relu"))  
model.add(Dense(150, activation = "relu")) #multiple dense layers  
model.add(Dense(5, activation = "softmax")) #output layer
```

```
[ ] #Adding the layers for accuracy  
num_classes = len(class_names)  
  
model = Sequential([  
    data_augmentation,  
    layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),  
    layers.Conv2D(16, 3, padding='same', activation='relu'),  
    layers.MaxPooling2D(),  
    layers.Conv2D(32, 3, padding='same', activation='relu'),  
    layers.MaxPooling2D(),  
    layers.Conv2D(64, 3, padding='same', activation='relu'),  
    layers.MaxPooling2D(),  
    layers.Flatten(),  
    layers.Dense(128, activation='relu'),  
    layers.Dense(num_classes)  
])
```

## 5.Compile The Model

### Solution:

```
model.compile(loss = "categorical_crossentropy", metrics = ["accuracy"], optimizer = 'adam')
len(x_train)
```

#### 5.Compile The Model

```
[ ] model.compile(loss = "categorical_crossentropy", metrics = ["accuracy"], optimizer = "adam")
len(x_train)
```

44

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

epochs=10

history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs)
```

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

epochs=10
history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=epochs
)
```

```

Epoch 1/10
108/108 [=====] - 128s 1s/step - loss: 1.3816 - accuracy: 0.4091 - val_loss: 1.2008 - val_accuracy: 0.4913
Epoch 2/10
108/108 [=====] - 125s 1s/step - loss: 1.0935 - accuracy: 0.5608 - val_loss: 1.0211 - val_accuracy: 0.5794
Epoch 3/10
108/108 [=====] - 126s 1s/step - loss: 0.9751 - accuracy: 0.6167 - val_loss: 0.9680 - val_accuracy: 0.6130
Epoch 4/10
108/108 [=====] - 126s 1s/step - loss: 0.9249 - accuracy: 0.6372 - val_loss: 0.8913 - val_accuracy: 0.6512
Epoch 5/10
108/108 [=====] - 125s 1s/step - loss: 0.8490 - accuracy: 0.6859 - val_loss: 0.8196 - val_accuracy: 0.6744
Epoch 6/10
108/108 [=====] - 126s 1s/step - loss: 0.8293 - accuracy: 0.6737 - val_loss: 0.9374 - val_accuracy: 0.6477
Epoch 7/10
108/108 [=====] - 125s 1s/step - loss: 0.7899 - accuracy: 0.7006 - val_loss: 0.7637 - val_accuracy: 0.6871
Epoch 8/10
108/108 [=====] - 125s 1s/step - loss: 0.7297 - accuracy: 0.7290 - val_loss: 0.7591 - val_accuracy: 0.7196
Epoch 9/10
108/108 [=====] - 126s 1s/step - loss: 0.7160 - accuracy: 0.7279 - val_loss: 0.8055 - val_accuracy: 0.7115
Epoch 10/10
108/108 [=====] - 125s 1s/step - loss: 0.6868 - accuracy: 0.7276 - val_loss: 0.8471 - val_accuracy: 0.7022

```

**To find the Training and Validation- Accuracy & Loss (Visualization)**

```
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()**

```
[ ] #To find the Training and Validation- Accuracy & Loss (Visualization)

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()
```



## 6. Fit The Model

### Solution:

```
model.fit(x_train, epochs = 15, steps_per_epoch = len(x_train))
```

## 6. Fit The Model

```
model.fit(x_train, epochs = 15, steps_per_epoch = len(x_train))
```

```
Epoch 1/15  
44/44 [=====] - 31s 684ms/step - loss: 1.7914 - accuracy: 0.3588  
Epoch 2/15  
44/44 [=====] - 29s 648ms/step - loss: 1.1730 - accuracy: 0.5045  
Epoch 3/15  
44/44 [=====] - 29s 650ms/step - loss: 1.0967 - accuracy: 0.5529  
Epoch 4/15  
44/44 [=====] - 29s 648ms/step - loss: 1.0351 - accuracy: 0.5939  
Epoch 5/15  
44/44 [=====] - 29s 645ms/step - loss: 0.9920 - accuracy: 0.6127  
Epoch 6/15  
44/44 [=====] - 30s 677ms/step - loss: 0.9659 - accuracy: 0.6259  
Epoch 7/15  
44/44 [=====] - 29s 648ms/step - loss: 0.9129 - accuracy: 0.6426  
Epoch 8/15  
44/44 [=====] - 29s 647ms/step - loss: 0.9005 - accuracy: 0.6433  
Epoch 9/15  
44/44 [=====] - 32s 717ms/step - loss: 0.8597 - accuracy: 0.6620  
Epoch 10/15  
44/44 [=====] - 30s 674ms/step - loss: 0.8350 - accuracy: 0.6824  
Epoch 11/15  
44/44 [=====] - 29s 648ms/step - loss: 0.8420 - accuracy: 0.6718  
Epoch 12/15  
44/44 [=====] - 29s 650ms/step - loss: 0.7857 - accuracy: 0.7030  
Epoch 13/15  
44/44 [=====] - 29s 649ms/step - loss: 0.7868 - accuracy: 0.7000  
Epoch 14/15  
44/44 [=====] - 29s 650ms/step - loss: 0.7542 - accuracy: 0.7132  
Epoch 15/15  
44/44 [=====] - 30s 676ms/step - loss: 0.7467 - accuracy: 0.7107  
<keras.callbacks.History at 0x7f602ce90090>
```

## 7. Save The Model

### Solution:

```
model.save("flowers.h1")  
model.save("flowers.m5")
```

## 7. Save The Model

```
[ ] model.save("flowers.h1")
```

```
[ ] model.save("flowers.m5")
```



## 8. Test The Model

### Solution:

```
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np

model = load_model("/content/flowers.h1")

#Testing with a random rose image from Google

img = image.load_img("/content/daisy.gif", target_size = (64,64) ) img
x = image.img_to_array(img) x.ndim 3
x = np.expand_dims(x,axis = 0) x.ndim

labels = ['daisy','dandelion','roses','sunflowers','tulips'] x_pred=model.predict
x_pred

labels[np.argmax(x_pred)]
```

```
[ ] from tensorflow.keras.models import load_model
    from tensorflow.keras.preprocessing import image
    import numpy as np
```

```
[ ] model = load_model("/content/flowers.h1")
```

```
[ ] #Testing with a random rose image from Google
```

```
    img = image.load_img("/content/daisy.gif", target_size = (64,64) )
```

```
[ ] img
```



```
[ ] x = image.img_to_array(img)
    x.ndim
```

```
3
```

```
[ ] x = np.expand_dims(x,axis = 0)
    x.ndim

4

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

[ ] x_pred=model.predict

[ ] x_pred

<bound method Model.predict of <keras.engine.sequential.Sequential object at 0x7f3901f2d850>>

[ ] labels[np.argmax(x_pred)]

'daisy'
```

**#Testing the model with accuracy**

**daisy\_url = "http://m.gettywallpapers.com/wp-content/uploads/2022/07/Daisy-Wallpaper-Images.jpg"**

**daisy\_path = tf.keras.utils.get\_file('Daisy-Wallpaper-Images', origin=daisy\_url)**

**img = tf.keras.utils.load\_img(**

**daisy\_path, target\_size=(img\_height, img\_width)**

**)**

**img\_array = tf.keras.utils.img\_to\_array(img)**

**img\_array = tf.expand\_dims(img\_array, 0) # Create a batch**

**predictions = model.predict(img\_array) score = tf.nn.softmax(predictions[0])**

```

print("This image most likely belongs to {} with a {:.2f} percent confidence."
      .format(class_names[np.argmax(x_pred)], 100 * np.max(score))
)

```

```

[ ] #Testing the model with accuracy

daisy_url = "http://m.gettywallpapers.com/wp-content/uploads/2022/07/Daisy-Wallpaper-Images.jpg"
daisy_path = tf.keras.utils.get_file('Daisy-Wallpaper-Images', origin=daisy_url)

img = tf.keras.utils.load_img(
    daisy_path, target_size=(img_height, img_width)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(x_pred)], 100 * np.max(score))
)

```

Downloading data from <http://m.gettywallpapers.com/wp-content/uploads/2022/07/Daisy-Wallpaper-Images.jpg>  
 901120/897455 [=====] - 0s 0us/step  
 909312/897455 [=====] - 0s 0us/step  
 This image most likely belongs to daisy with a 94.37 percent confidence.