-	Student Name	R.Sai Nishit
Γ	Student Roll Number Download the Dataset	2116190701179
1]:	import numpy as np import tensorflow as tf from tensorflow keras import layers	
	<pre>from tensorflow.keras.models import Sequential import matplotlib.pyplot as plt import os</pre>	
	mage Augmentation	
	<pre>data_aug = Sequential([</pre>	
	<pre>layers.RandomZoom(0.1), layers.RandomZoom(0.1),]</pre>	
	os.listdir("C:\\Users\\Sai\\Flowers-Dataset") 'flowers']	
6]:	<pre>train_data = tf.keras.utils.image_dataset_from_directory("C:\Users\\Harini\\Flowers-Dataset", validation_split=0.25,</pre>	
	<pre>subset="training", seed=120, image_size=(180, 180), batch_size=batch_size)</pre>	
, t	<pre>Found 4317 files belonging to 1 classes. Justing 3238 files for training. val_data_set = tf.keras.utils.image_dataset_from_directory("Ct\\Users\\Harini\\Plowers-Dataset",</pre>	
	<pre>validation_split=0.25, subset="validation", seed=120, image_size=(180, 180),</pre>	
Ţ	batch_size=batch_size) Found 4317 files belonging to 1 classes. Psing 1079 files for validation.	
9]:	class_names = train_data.class_names plt.figure(figsize=(15, 15))	
	<pre>for images, labels in train_data.take(1): for i in range(6): ax = plt.subplot(3, 3, i + 1) plt.imshow(images[i].numpy().astype("uint8")) plt.title(class_names[labels[i]))</pre>	
	100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 - 100 -	
	120 - 120 - 120 - 120 - 120 - 140 - 140 - 140 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 - 160 -	
	20 - 20 - 40 - 40 - 60 - 60 - 60 - 60 - 60 - 6	
0]:	160 160 160 150 0 50 100 150 0 50 100 150 0 50 100 150	
1]:	normalization_layer = layers.Rescaling(1./255) dataset_normalized = train_data.map(lambda x, y: (normalization_layer(x), y)) image_batch, labels_batch = next(iter(dataset_normalized)) first_image = image_batch[0] print(np.min(first_image), np.max(first_image)) 1.0 1.0	
	Create Model dd Layers (Convolution, MaxPooling, Flatten, Dense-(Hidden Layers), Output)	
	<pre>num_classes = len(class_names) model = Sequential([data_aug, layers.Rescaling(1./255, input_shape=(180, 180, 3)), layers.Conv2D(16, 3, activation='relu'), layers.MaxPooling2D(), layers.Conv2D(32, 3,activation='relu'), layers.MaxPooling2D(), layers.Conv2D(32, 3,activation='relu'), layers.MaxPooling2D(), layers.Conv2D(64, 3, activation='relu'), layers.MaxPooling2D(), layers.Flatten(), layers.Dense(128, activation='relu'),</pre>	
	layers.Dense(num_classes)])	
(Committee Model	
0	Compile The Model ompiling model with categorical cross entropy and adam optimizer	
3]:	<pre>ompiling model with categorical cross entropy and adam optimizer model.compile(optimizer='adam', loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy'])</pre>	
3]: F	ompiling model with categorical cross entropy and adam optimizer model.compile(optimizer='adam', loss=tf.kezas.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy']) Fit The Model epochs=15	
63]: F 4]:	ompiling model with categorical cross entropy and adam optimizer model.compile(optimizer='adam', loss=tf.kexas.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=('accuracy')) Fit The Model epochs=15 history = model.fit(train_data,validation_data=val_data_set,epochs=epochs) spoch 1/15 203/203 [========] - 33s 154ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.000 poch 2/15	
F F F F F F F F F F F F F F F F F F F	ompiling model with categorical cross entropy and adam optimizer model.compile(optimizer='adam', loss=tf.kexas.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=('accuracy'!) Fit The Model epochs=15 history = model.fit(trsin_data,validation_data=val_data_set,epochs=epochs) epoch 1/15 203/203 [==============] = 33s 154ms/step = loss: 0.0000e+00 = accuracy: 1.0000 = val_loss: 0.000 epoch 2/15 203/203 [=============] = 28s 140ms/step = loss: 0.0000e+00 = accuracy: 1.0000 = val_loss: 0.000 epoch 4/15 203/203 [=============] = 29s 141ms/step = loss: 0.0000e+00 = accuracy: 1.0000 = val_loss: 0.000 epoch 4/15 203/203 [=============] = 29s 141ms/step = loss: 0.0000e+00 = accuracy: 1.0000 = val_loss: 0.000 epoch 4/15 203/203 [==============] = 29s 143ms/step = loss: 0.0000e+00 = accuracy: 1.0000 = val_loss: 0.000 epoch 5/15	0e+00 - val_accuracy: 1.0000 0e+00 - val_accuracy: 1.0000 0e+00 - val_accuracy: 1.0000
F F F F F F F F F F F F F F F F F F F	model.compile(optimizer='adam', loss=tf.kezas.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy']) Fit The Model epochs=15 history = model.fit(train_data,validation_data=val_data_set,epochs=epochs) fpoch 1/15 epoch 2/15 epoch 2/15 epoch 2/15 epoch 3/15 epoch	0e+00 - val_accuracy: 1.0000 0e+00 - val_accuracy: 1.0000 0e+00 - val_accuracy: 1.0000 0e+00 - val_accuracy: 1.0000
F F F F F F F F F F F F F F F F F F F	model.compile(optimizer='adam', loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=('accuracy'!) Fit The Model epochs=15 history = model.fit(trsin_data,validation_data=val_data_set,epochs=epochs) epoch 1/15 epoch 1/15 epoch 3/15 epoch	0e+00 - val_accuracy: 1.0000
F F 41:	model.compile(optimizer='adam', loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=('accuracy')) Fit The Model epochs=15 history = model.fit(train_data,validation_data=val_data_set,epochs=epochs) Epoch 1/15 203/203 [====================================	0e+00 - val_accuracy: 1.0000
FF	model.compile(optimizer='adam', loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=('accuracy')) Fit The Model epoch=15 history = model.fit(train_data,validation_data=val_data_set,epochs=epochs) epoch 1/15 gpoch 1/15 gpoch 1/15 gpoch 2/15 gpoch 3/15 gpoch 3/1	0e+00 - val_accuracy: 1.0000
F F F F F F F F F F F F F F F F F F F	model.compile(optimizer='adam', loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=('accuracy')) Fit The Model spochs=15 history = model.fit(train_data,validation_data=val_data_set,epochs=epochs) spoch 1/15 spoch 1/15 spoch 2/15 spoch 2/15 spoch 2/15 spoch 3/15 spoch	0e+00 - val_accuracy: 1.0000 0e+00 - val_accuracy: 1.0000 0e+000 - val_accuracy: 1.0000
F F 141:	model.compile (optimirer='adam', loss=tf.kersa.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=('accuracy')) Fit The Model spochs=15 history = model.fit(train_data,validation_data=val_data_set,epochs=epochs) spoch 1/15 spoch 1/15 spoch 1/15 spoch 1/15 spoch 2/15 spoch 2/15 spoch 2/15 spoch 2/15 spoch 3/15 spoc	0e+00 - val_accuracy: 1.0000 0e+00 - val_accuracy: 1.0000 0e+000 - val_accuracy: 1.0000
F F 141:	model.compile(optimirer='adam', lossetf.keras.losses.sparseCategoricalCrossentropy(from_logits=True), metrics=('accuracy')) Fit The Model spochs=15 history = model.fit(train_data,validation_data=val_data_set,epochs=epochs) spoch 1/15 103/203 [====================================	0e+00 - val_accuracy: 1.0000
F F F F F F F F F F F F F F F F F F F	model.compile(optimizer='adam', lossetf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=('accuracy')) Fit The Model spochs=15 history = model.fit(train_data, validation_data=val_data_set, epochs=epochs) spoch 1/15 spoch 1/15 spoch 1/15 spoch 2/15 spoch 2/15 spoch 3/15 spoc	0e+00 - val_accuracy: 1.0000
F F F F F F F F F F F F F F F F F F F	model.compile(optimirer='adam', lossetf.keras.losses.SparseCategoricalCrossentropy(from_logits=True), metrics=['accuracy']) Fit The Model spechs=15 history = model.fit(train_data, validation_data=val_data_set, epochs=epochs) spech 1/15 spech 1/15 spech 1/15 spech 2/15 spech 3/15 spec	0e+00 - val_accuracy: 1.0000 0e+00 - val_accuracy: 1.0000 0e+000 - val_accuracy: 1.0000
F F F F F F F F F F F F F F F F F F F	model.compile (optimizer='adam', lossetf.keras.losses.spareCategoricalCrossentropy(from_logits=True), metricas=['accuracy']) Fit The Model spochs=15 history = model.fit(train_data,validation_data=val_data_set,epochs=epochs) spoch 1/15 spoc	0e+00 - val_accuracy: 1.0000 0e+00 - val_accuracy: 1.0000 0e+000 - val_accuracy: 1.0000
F F F F F F F F F F F F F F F F F F F	model.compile (optimizer*adam*, lossef.karsa.losses.SparseCategoricalCrossentropy(from_logits*True), metricse("Accuracy")) Fit The Model spochsels history = model.fit(train_data,validation_datasval_data_set,epochs*epochs) spoch 1/15 spoch 1/15 spoch 1/15 spoch 2/15 spoch 2/15 spoch 2/15 spoch 3/15 spoch 3/1	0e+00 - val_accuracy: 1.0000 0e+00 - val_accuracy: 1.0000 0e+000 - val_accuracy: 1.0000
F F F F F F F F F F F F F F F F F F F	model.compile (optimizer'sdam', losset, karaa.losses.dparsfatepricalCrossentropy(from_logits=True), metrics=(*accused;*)*) Fit The Model spochs=15 history = model.fit(train_data,validation_data=val_data_set,epochs=epochs) spoch 1/15 spoch 1/1	0e+00 - val_accuracy: 1.0000 0e+00 - val_accuracy: 1.0000 0e+000 - val_accuracy: 1.0000
6]:	model with categorical cross entropy and adam optimizer model.compile (optimizer='adam', nearticese('accuracy')) Fit The Model spocha=15 spocha=16 spocha=	0e+00 - val_accuracy: 1.0000
6]:	model compiles (post interer index), losses (paras Categorical cross entropy and adam optimizer model), compiles (post interer index), settices ("accuracy")) Fit The Model spochs=15 history = model.fit(train_data_varidation_data=var_data_set_epochs=epochs) ppoch 1/15 spochs 1/25	0e+00 - val_accuracy: 1.0000
6]:	model: compiles (optimizers' addes'), matrices' **ecuracy') Fit The Model **geocha=15 history = model.fit(train_data_validation_data_val_data_ext_epocha=spechs) peocha=15 history = model.fit(train_data_validation_data_val_data_ext_epocha=spechs) peocha=16 history = model.fit(train_data_validation_data_val_data_ext_epocha=spechs) peocha=16 history = model.fit(train_data_validation_data_val_data_ext_epocha=spechs) peocha=16 history = model.fit(train_data_validation_data_val_data_ext_epocha=spechs) peocha=16 history = model.fit(train_data_validation_data_val_data_epocha=spechs=spechs) peocha=16 history = model.fit(train_data_validation_data_val_data_epocha=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spechs=spe	0e+00 - val_accuracy: 1.0000
6]:	model complice (post interviale) instructive **Tacoustry** **Tacoustry**	0e+00 - val_accuracy: 1.0000 0e+00 - val_accuracy: 1.0000 0e+000 - val_accuracy: 1.0000
6]:	model complies portions are "alam"; Insert intered "accuracy"; The Model appoint 155 appoint 175 appoint	0e+00 - val_accuracy: 1.0000 0e+00 - val_accuracy: 1.0000 0e+000 - val_accuracy: 1.0000

[17]: model.save("./flowers.h5")

Save The Model

[18]: model.load_weights('./flowers.h5')

Test The Model

[19]: from tensorflow.keras.preprocessing import image import numpy as np

img=image.load_img('C:\\Users\\Sai\\Flowers-Dataset\\flowers\\rose\\5172171681_5934378f08.jpg',target_size=(70,70)) img

[20]

