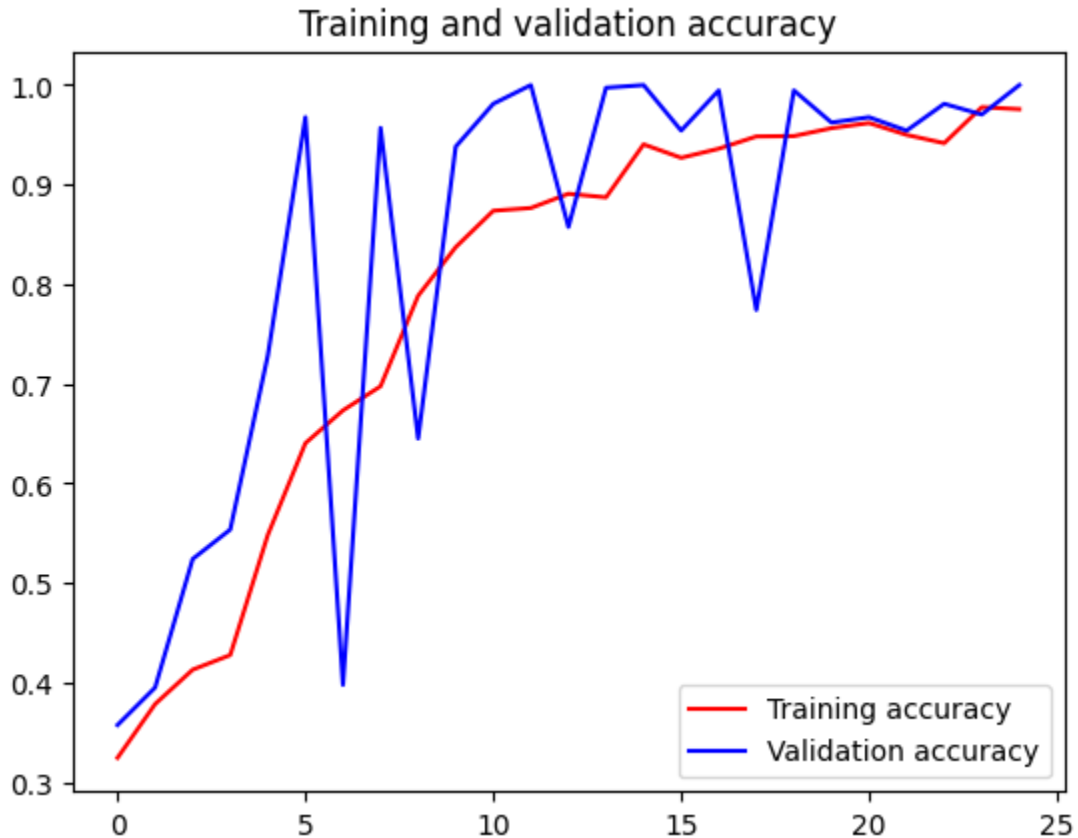


Name: Aadesh Sure

PART 1:

Part one has been implemented the way the tutorials have depicted. The codes are provided in the folder section name Part A.

Result image for part 4 (Rock Paper scissors):



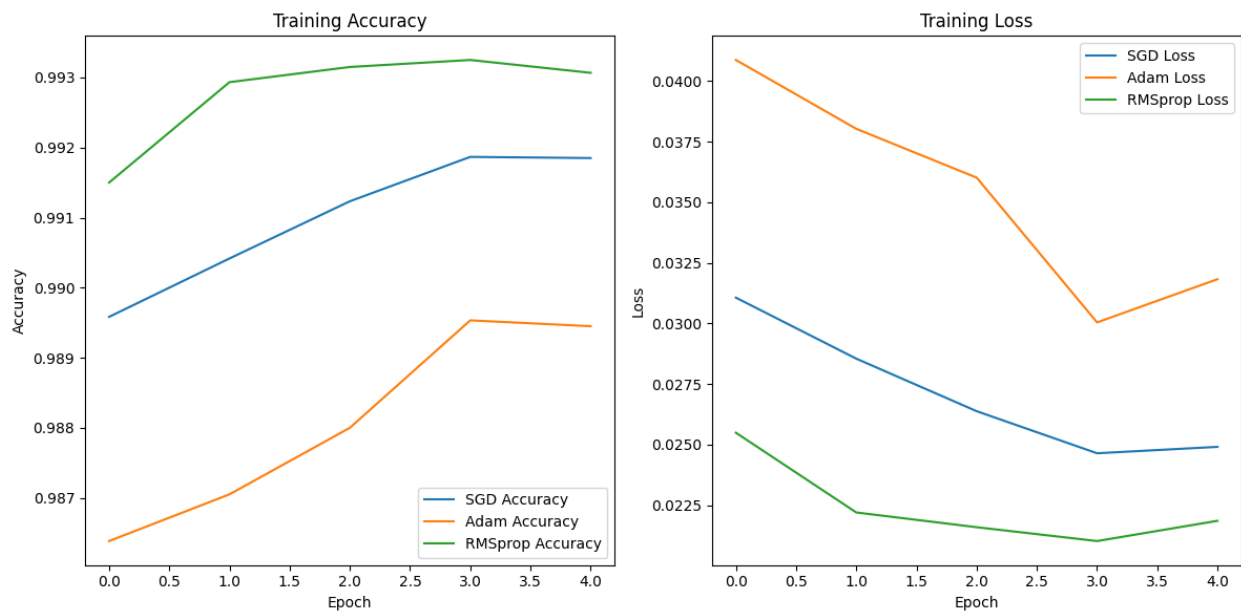
PART 2

A)

In the standard model as per the tutorial

We can observe that the Rmsprop optimizers gives the best results, followed by sgd and adam.

Results :



Optimizer: SGD

313/313 - 1s - loss: 0.0980 - accuracy: 0.9817 - 644ms/epoch - 2ms/step

Test accuracy: 0.9817000031471252

Optimizer: Adam

313/313 - 1s - loss: 0.0980 - accuracy: 0.9817 - 523ms/epoch - 2ms/step

Test accuracy: 0.9817000031471252

Optimizer: RMSprop

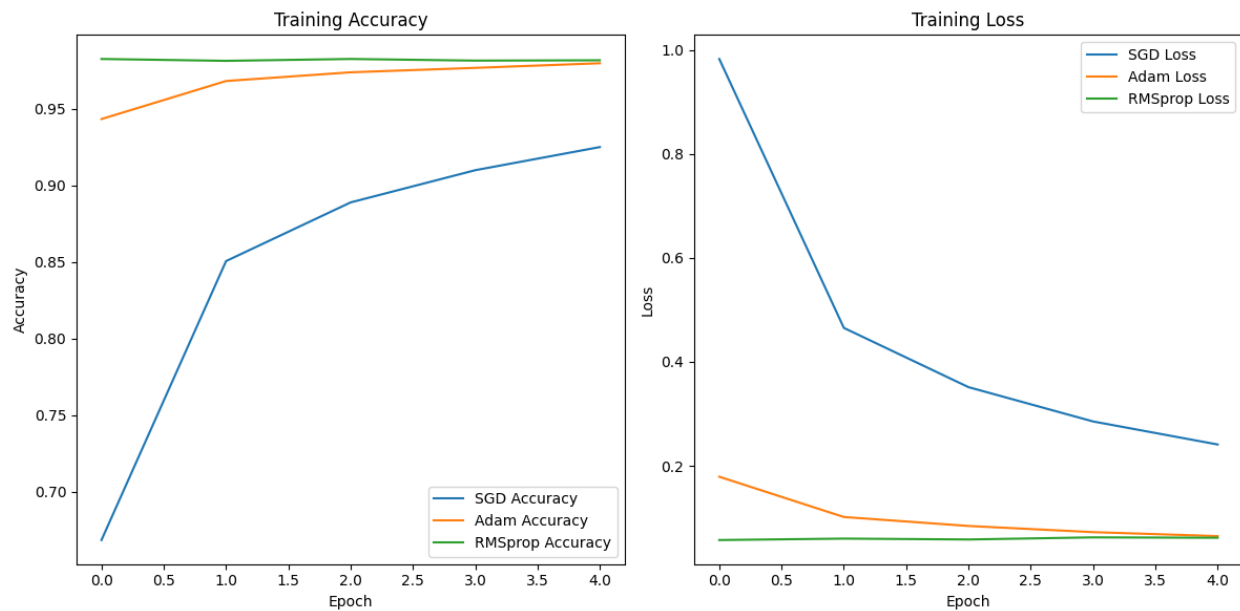
313/313 - 1s - loss: 0.0980 - accuracy: 0.9817 - 526ms/epoch - 2ms/step

Test accuracy: 0.9817000031471252

B)

In my custom model

We can observe that the Rmsprop optimizers gives the best results, followed by adam and sgd.



Optimizer: SGD

313/313 - 3s - loss: 0.0346 - accuracy: 0.9890 - 3s/epoch - 9ms/step

Test accuracy: 0.9890000224113464

Optimizer: Adam

313/313 - 4s - loss: 0.0346 - accuracy: 0.9890 - 4s/epoch - 11ms/step

Test accuracy: 0.9890000224113464

Optimizer: RMSprop

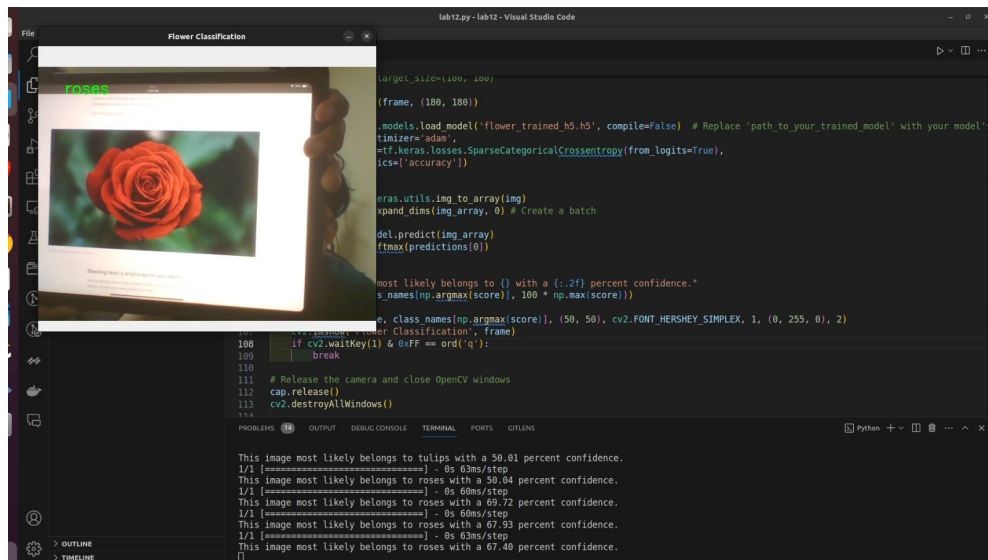
313/313 - 3s - loss: 0.0346 - accuracy: 0.9890 - 3s/epoch - 9ms/step

Test accuracy: 0.9890000224113464

FOR PART C:

I trained the model for 20 epochs on google colab and then loaded the weights and ran on the video recording scripts.

Here are some snippets:



```
target_size=(100, 100)
(frame, (180, 180))

models.load_model('flower_trained_h5.h5', compile=False) # Replace 'path_to_your_trained_model' with your model's
optimizer='adam',
if keras.losses.SparseCategoricalCrossentropy(from_logits=True),
ics=['accuracy'])

keras.utils.img_to_array(img)
expand_dims(img_array, 0) # Create a batch

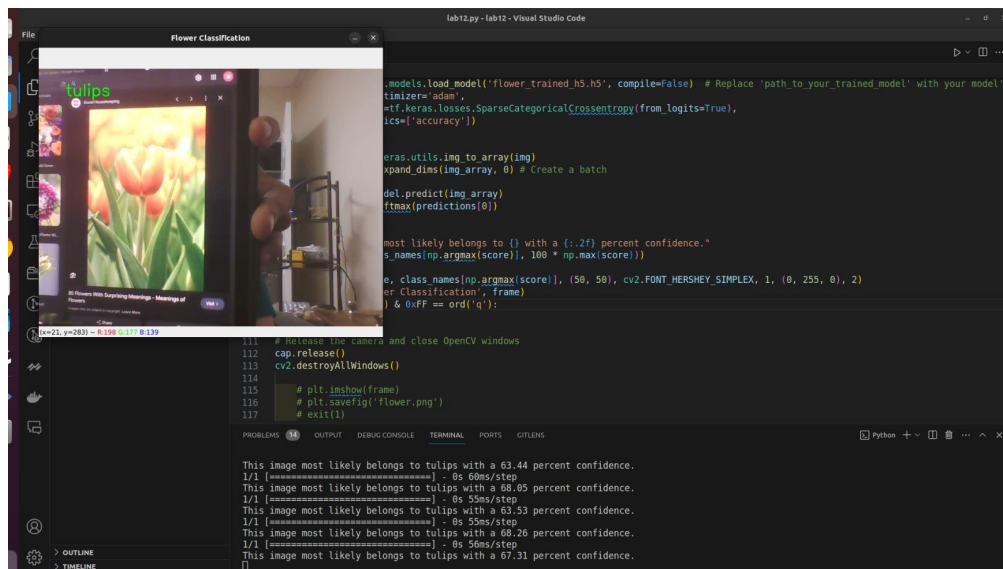
del.predict(img_array)
fmax(predictions[0])

most likely belongs to {} with a {:.2f} percent confidence."
s_names(np.argmax(score)), 100 * np.max(score)))

e, class_names(np.argmax(score)), (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
er Classification', frame)
) & 0xFF == ord('q')):
    break

# Release the camera and close OpenCV windows
cap.release()
cv2.destroyAllWindows()

This image most likely belongs to tulips with a 50.01 percent confidence.
1/1 [=====] - 0s 63ms/step
This image most likely belongs to roses with a 50.04 percent confidence.
1/2 [=====] - 0s 60ms/step
This image most likely belongs to roses with a 69.72 percent confidence.
1/1 [=====] - 0s 60ms/step
This image most likely belongs to roses with a 67.93 percent confidence.
1/1 [=====] - 0s 62ms/step
This image most likely belongs to roses with a 67.40 percent confidence.
```



```
models.load_model('flower_trained_h5.h5', compile=False) # Replace 'path_to_your_trained_model' with your model's
optimizer='adam',
if keras.losses.SparseCategoricalCrossentropy(from_logits=True),
ics=['accuracy'])

keras.utils.img_to_array(img)
expand_dims(img_array, 0) # Create a batch

del.predict(img_array)
fmax(predictions[0])

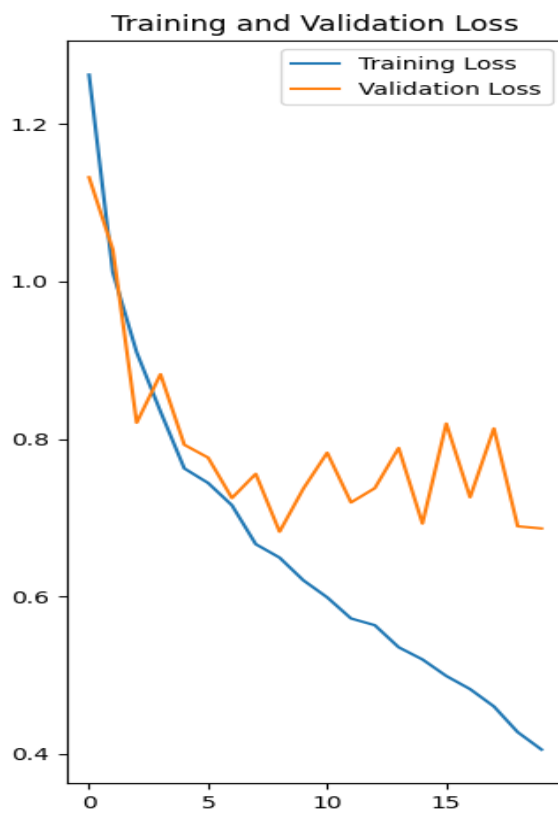
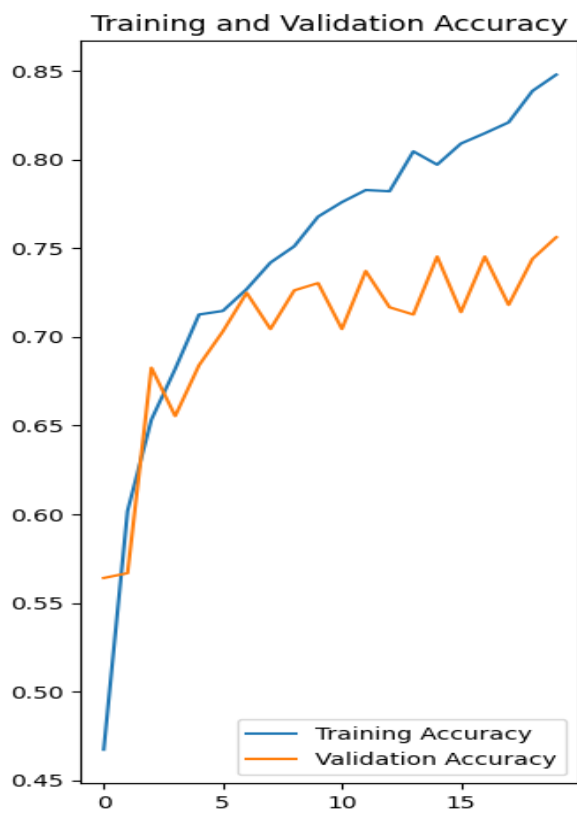
most likely belongs to {} with a {:.2f} percent confidence."
s_names(np.argmax(score)), 100 * np.max(score)))

e, class_names(np.argmax(score)), (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
er Classification', frame)
) & 0xFF == ord('q')):
    break

# Release the camera and close OpenCV windows
cap.release()
cv2.destroyAllWindows()

plt.imshow(frame)
plt.savefig('flower.png')
exit(1)

This image most likely belongs to tulips with a 63.44 percent confidence.
1/1 [=====] - 0s 60ms/step
This image most likely belongs to tulips with a 60.05 percent confidence.
1/1 [=====] - 0s 55ms/step
This image most likely belongs to tulips with a 63.53 percent confidence.
1/1 [=====] - 0s 55ms/step
This image most likely belongs to tulips with a 68.26 percent confidence.
1/1 [=====] - 0s 56ms/step
This image most likely belongs to tulips with a 67.31 percent confidence.
```



BONUS:

