CSE 665 Deep Learning

TERM PROJECT REPORT

Emotion Insight: Precision Facial Emotion Detection ***(PFED)***

Ercan AKCAN 244201001033

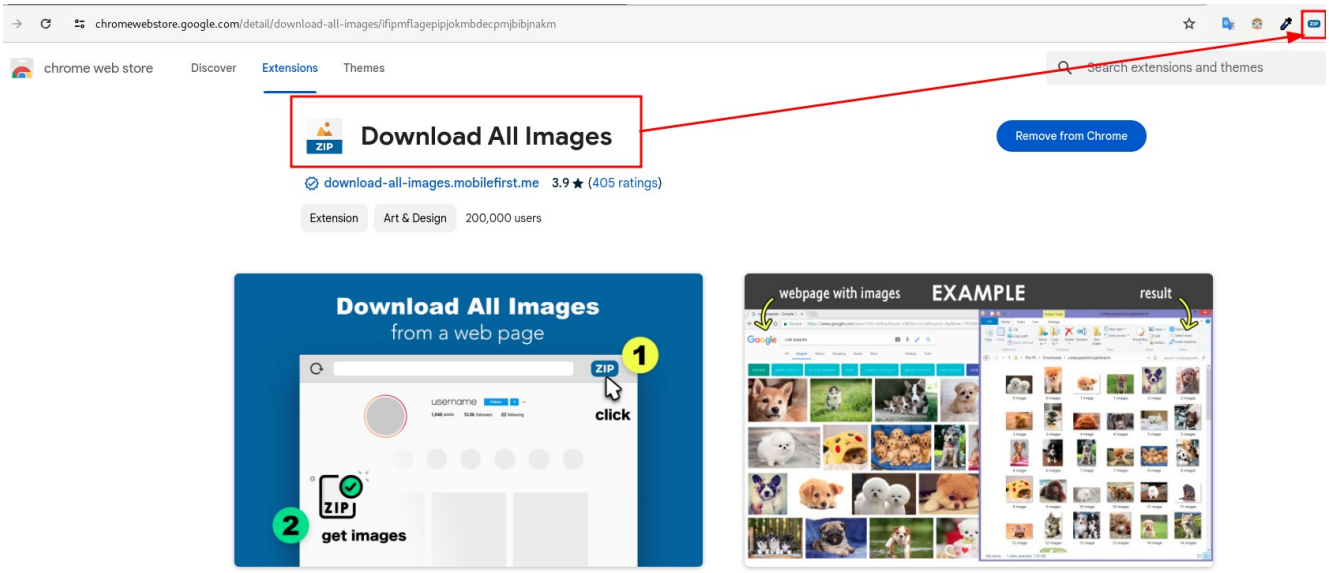
Contents

Introduction

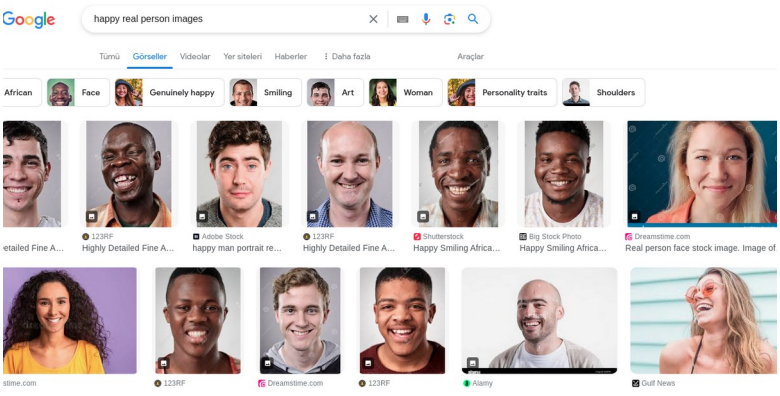
Emotion Insight: *Precision Facial Emotion Detection* (**PFED**); This project focuses on harnessing the power of facial recognition technology to detect and interpret emotions displayed on human faces. By leveraging advanced algorithms, we aim to accurately identify to six classification: happy, sad, angry, afraid, surprised, disgusted.

Data Source

I aim to utilize images sourced from Google Images, particularly those depicting real-life reactions in various daily situations. For this operation, I use a plugin "**Download All Images**" for chrome. Actually, the plugin saves all images in active tab as .zip file. Easily save photos from Google Images.



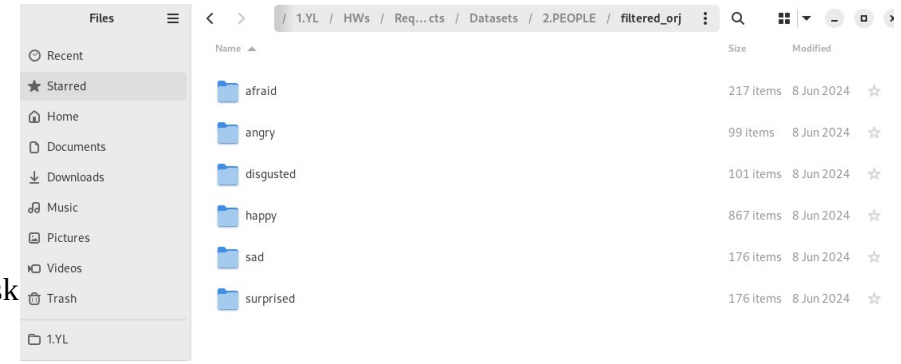
My data source has six classification which includes happy, sad, angry, afraid, surprised, disgusted. So I searched each classification, for example happy person, I searched "*happy real person images*" and went down to page, when I came to last images, ran plugin and zip file is downloaded.



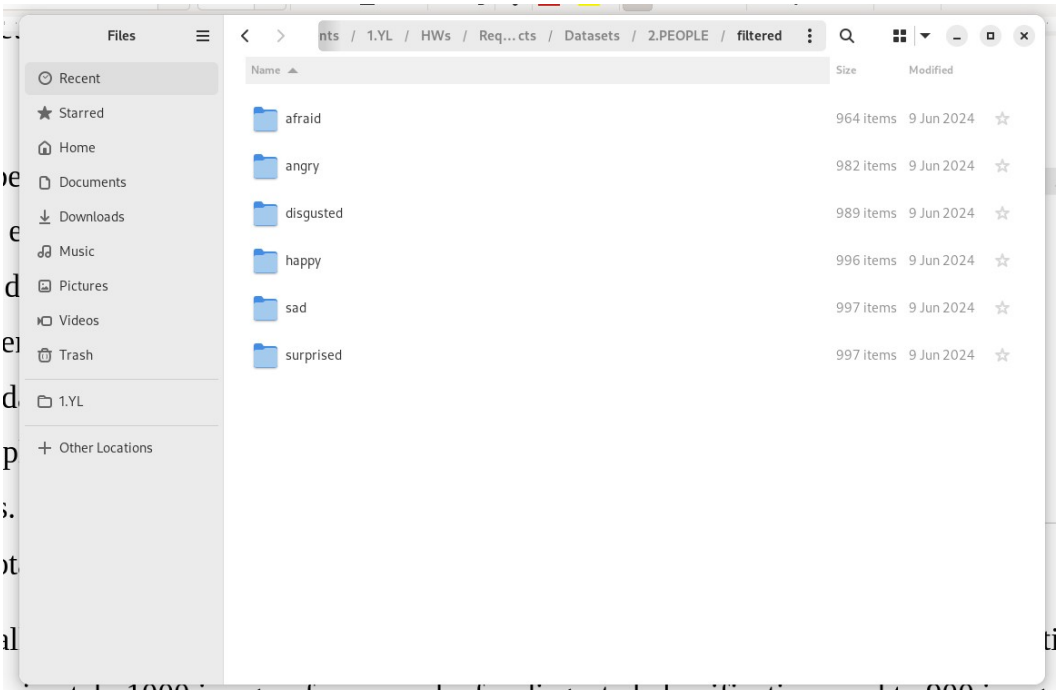
When I saw download file, I saw files which include .webp, .ico and other extensions that didn't contained image data. Moreover, for example I searched happy person, but the images contained sad or unhappy person, I mean my data source didn't clean, it was dirty with other classification images.

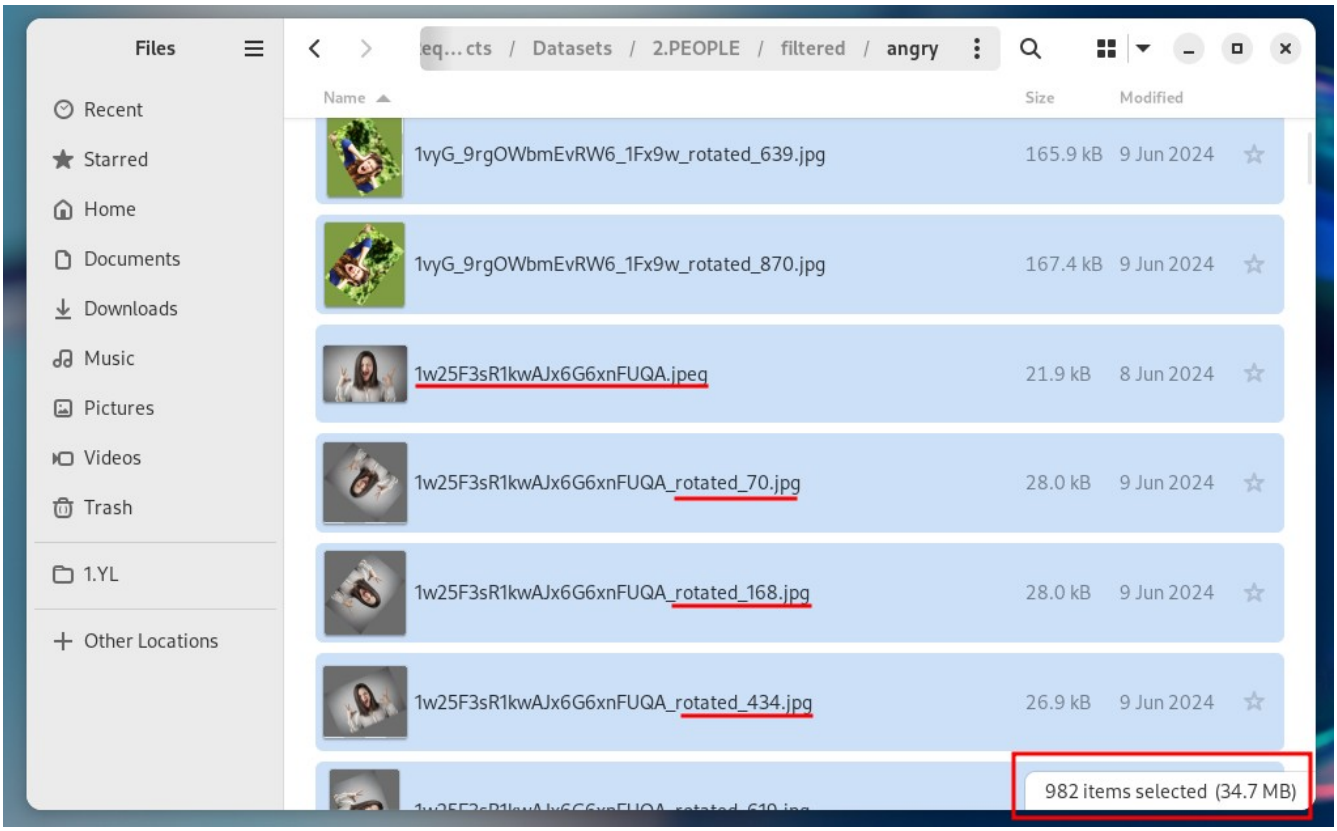
I wrote some code to clean my dataset from dodgy images. When I ran the function, that name is *dirty_image_cleaner()*, clean some files for example .webp, .ico etc. After that operation, I saw my dataset, it had images very limited.

Maybe after training, I might encounter some errors, like vanishing gradient or exploding gradient. I can fix these problems with some data, but I didn't have data, I will, I can augment data, for example some rotations, or mask some filters. I wrote some code script which did rotation called *def rotate_image(image_path, degrees).*



Actually, I wanted them all to have the same number of images. I decided all classification had approximately 1000 images, for example, for disgusted classification need to 900 images, actually 900 rotated images. After added rotated images, my dataset was ready to go, all classification almost had same numbers of images.





Building Model

Get data with keras.utils *data = tf.keras.utils.image_dataset_from_directory(base_dir),* the utils sended me "*Found 5925 files belonging to 6 classes*".

Then I divided it into 3 parts: %70 for training, %20 for validation and %10 for test.

```
[10] 1 train_size = int(len(data)*.7)
     2 val_size = int(len(data)*.2)
     3 test_size = int(len(data)*.1)
```

Executed at 2024.06.11 09:50:57 in 2ms

Model Summary

Model: "sequential_5"

Layer (type)	Output Shape	Param #
conv2d_9 (Conv2D)	(None, 254, 254, 16)	448
max_pooling2d_6 (MaxPooling2D)	(None, 127, 127, 16)	0
conv2d_10 (Conv2D)	(None, 125, 125, 32)	4,640
max_pooling2d_7 (MaxPooling2D)	(None, 62, 62, 32)	0
conv2d_11 (Conv2D)	(None, 60, 60, 16)	4,624
max_pooling2d_8 (MaxPooling2D)	(None, 30, 30, 16)	0
flatten_2 (Flatten)	(None, 14400)	0
dense_4 (Dense)	(None, 256)	3,686,656
dense_5 (Dense)	(None, 6)	1,542

Total params: 3,697,910 (14.11 MB)

Trainable params: 3,697,910 (14.11 MB)

Non-trainable params: 0 (0.00 B)

Training Model

```
hist = model.fit(train, epochs=20, validation_data=val)
```

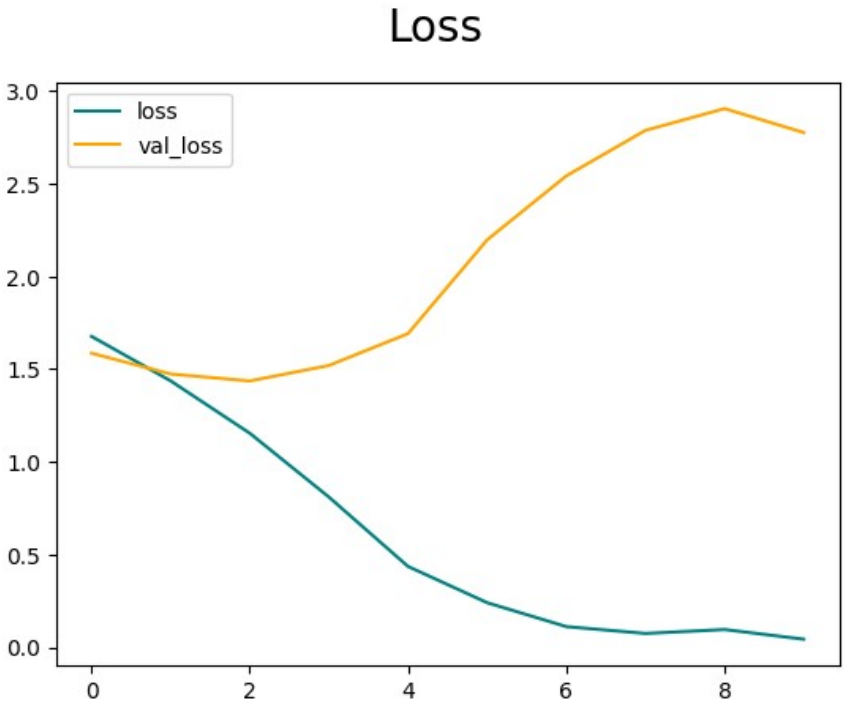
Executed at 2024.06.11 10:03:13 in 10m 8s 175ms

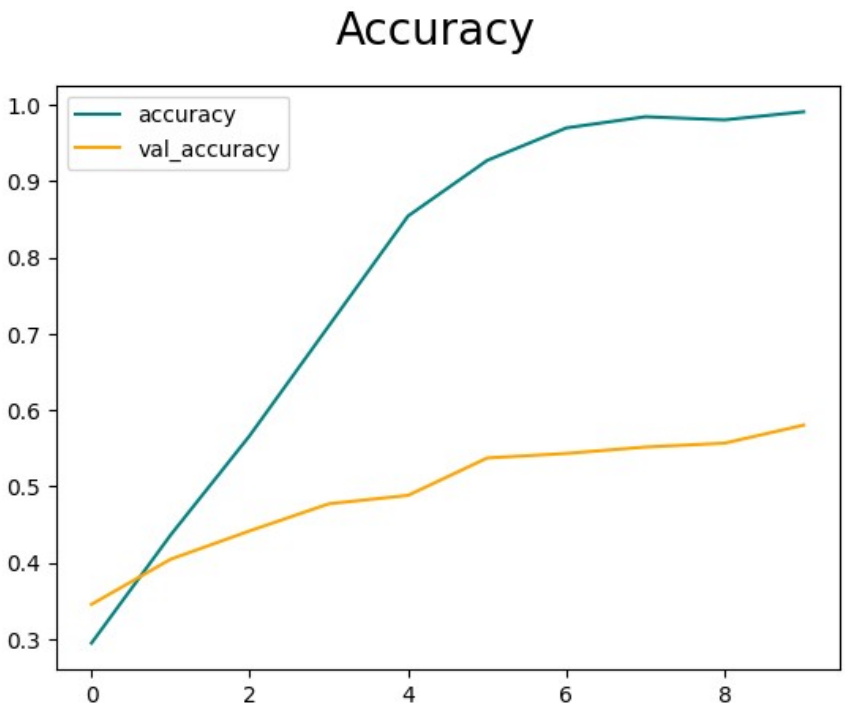
```
Epoch 1/20
130/130 ━━━━━━━━━━━━━━━━━━━━ 32s 239ms/step - accuracy: 0.2007 - loss: 1.9010 - val_accuracy: 0.3353 - val_loss: 1.6453
Epoch 2/20
130/130 ━━━━━━━━━━━━━━━━━━━━ 31s 236ms/step - accuracy: 0.3606 - loss: 1.5940 - val_accuracy: 0.3910 - val_loss: 1.5559
Epoch 3/20
130/130 ━━━━━━━━━━━━━━━━━━━━ 31s 235ms/step - accuracy: 0.4896 - loss: 1.3444 - val_accuracy: 0.4595 - val_loss: 1.4514
Epoch 4/20
130/130 ━━━━━━━━━━━━━━━━━━━━ 31s 236ms/step - accuracy: 0.5963 - loss: 1.0553 - val_accuracy: 0.4975 - val_loss: 1.4007
Epoch 5/20
130/130 ━━━━━━━━━━━━━━━━━━━━ 31s 236ms/step - accuracy: 0.7109 - loss: 0.7966 - val_accuracy: 0.5287 - val_loss: 1.4987
Epoch 6/20
130/130 ━━━━━━━━━━━━━━━━━━━━ 31s 234ms/step - accuracy: 0.8038 - loss: 0.5547 - val_accuracy: 0.5532 - val_loss: 1.5038
Epoch 7/20

2024-06-11 09:58:09.737305: W tensorflow/core/kernels/data/prefetch_autotuner.cc:52] Prefetch autotuner tried to allocate 25165952 bytes after en
element of size 25165952 bytes.This already causes the autotune ram budget to be exceeded. To stay within the ram budget, either increase the ra
element size

130/130 ━━━━━━━━━━━━━━━━━━━━ 31s 234ms/step - accuracy: 0.9494 - loss: 0.1714 - val_accuracy: 0.6014 - val_loss: 1.9252
Epoch 11/20
130/130 ━━━━━━━━━━━━━━━━━━━━ 0s 192ms/step - accuracy: 0.9442 - loss: 0.1835
```

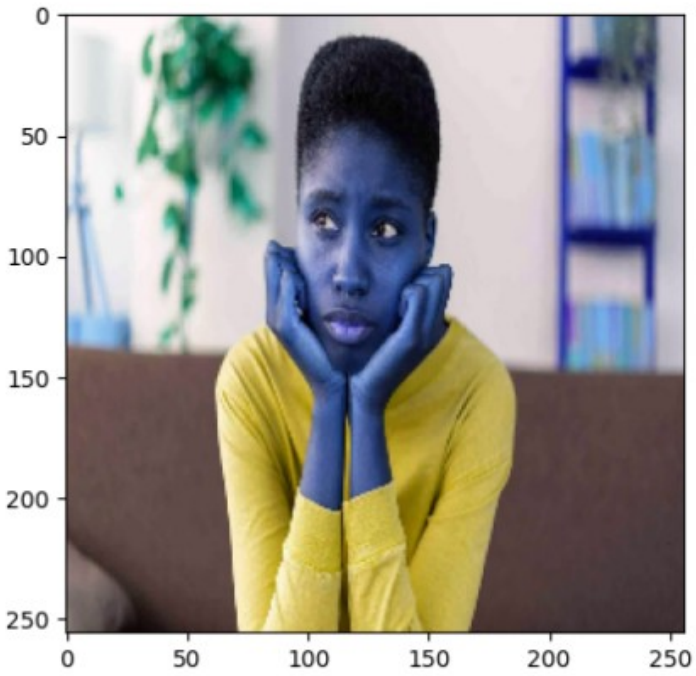
Loss & Accuracy





Test

[86]
```
resized_image = tf.image.resize(img, (256,256))
plt.imshow(resized_image.numpy().astype(int))
plt.show()
```
Executed at 2024.06.11 11:45:43 in 98ms



[100]
```
pred = model.predict(np.expand_dims(resized_image/255, 0))
pred
```
Executed at 2024.06.11 11:54:46 in 34ms

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 12ms/step
```

1 row | 1 rows × 6 columns

	0	1	2	3	4	5
0	0.002166	0.341018	0.000165	0.000108	0.648855	0.007688

[101]
```
class_names[np.argmax(pred)]
```
Executed at 2024.06.11 11:54:48 in 2ms

```
'sad'
```