S.Varsha

RA2011026010286

FUR-SEAL FACE RECOGNITION:

This code is an implementation of a deep learning method for fur-seal identification. It relies on the triplet loss defined in FaceNet paper and on novel deep learning techniques as ResNet networks.

Dog faces pictures were retrieved from the web and aligned using three handmade labels. We used VIA tool to label the images. The dataset is available here: [Releases Page](https://github.com/GuillaumeMougeot/DogFaceNet/releases/).

This code contains also an automatic face alignement tool and several implementation of GANs (Generative Adverserial Networks) onto the dataset.

CITATION:

If ever this project (code or dataset) is used for a publication, please cite: [A Deep Learning Approach for Dog Face Verification and Recognition](https://link.springer.com/chapter/10.1007/978-3-030-29894-4_34)

```

@InProceedings{10.1007/978-3-030-29894-4_34,

author="Mougeot, Guillaume and Li, Dewei and Jia, Shuai",

editor="Nayak, Abhaya C. and Sharma, Alok",

title="A Deep Learning Approach for Dog Face Verification and Recognition",

booktitle="PRICAI 2019: Trends in Artificial Intelligence",

year="2019",

publisher="Springer International Publishing",

address="Cham",

pages="418--430",

isbn="978-3-030-29894-4"
```

}

```

# DATASET:

The used data is available for download on the releases page:
(https://github.com/tulip-lab/furseal/tree/main/datasets)

# RUN THE RECOGNITION ALGORITHM:

To run the code you will need:

* python >= 3.6.4

* tensorflow == 1.12.0 (this constraint will be improved)

* numpy >= 1.14.0 * matplotlib >= 2.1.2

* scikit-image >= 0.13.1

* jupyter >= 1.0.0 (optional: only for dev)

* tqdm >= 4.23.4 (optional: only for dev)

Then run the following command from the root directory of the project:

python fur-sealfacenet/fur-sealfacenet.py

To run properly the dataset has to be located in a data/fur-sealfacenet folder or you will have to edit the config part of the fur-sealfacenet.py file.

The above command will train a model and save it into output/model directory. It will also save its history in output/history.

# CONTENT:

As previously described, the stable version is in fur-sealfacenet/fur-sealfacenet.py. It contains:

* the online and offline training modules

* the model definition and training

* the model evaluation (still in development)

The fur-sealfacenet-dev folder contains the developer version of the code. Model evaluation (verification, recognition, clustering, ROC curve, observation on the heatmap, ...) is in developer folder.
```

It will be transfer in stable folder soon. The main dev version is in fur-sealfacenet-dev/fur-sealfacenet_v12-dev.ipynb jupyter notebook file.

* (data: the images of the project) not available right now...

* fur-sealfacenet: stable version of the DogFaceNet project.

* fur-sealfacenet: dataset loading, model definiton and training

* offline/online_training: function for triplet generation

* fur-sealfacenet-dev: the main part, it contains the code on fur-seal face verification and on face alignment (in fur-sealfacenet/labelizer).

* labelizer: contains the data-preprocessing function after labeling the images using VIA

* copy_images: copies the images from the output folder of VIA to the input folder of DogFaceNet

* transform_csv_to_clean_format: edits the output csv file from VIA to a adapted format

* align_face: aligns copied faces using the edited csv file

* fur-sealfacenet-*dataset*: different tries on different dataset

* fur-sealfacenet_v*version_number*: the different version of the code on fur-seal pictures

* dataset: deprecated

* losses: deprecated

* models: deprecated

* triplet_loss: deprecated

* triplet_preprocessing: triplets linked functions

* triplets definition

* triplets augmentation

* hard triplets definition

* GAN: a fur-seal face generator in developement...

* landmarks: a try on automatic facial landmarks detection, still in developement...

* output:

* (model: the trained models not available right now...)

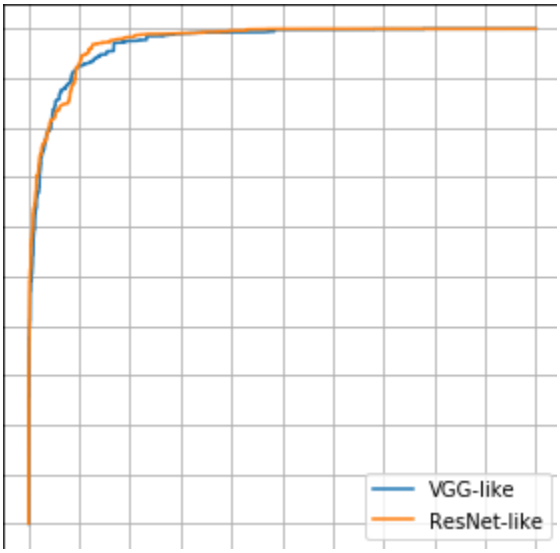
* history: the convergence curves

* tmp: archive of old codes and tests .

RESULTS ON FACE VERIFICATION:

UPDATE: the now available dataset is bigger (around 8600 pictures) than the one presented in the article (around 1400 pictures). The following results (and the one presented in the paper) were found using the small dataset. The results on the big dataset are unfortunately worse than the one on the small dataset (86% accuracy on the open-set): this can be explained knowing that there are much more fur-seals per breed in the test set, it thus makes the problem harder (two different fur-seals of the same breed are very similar).

The current version of the code reaches 92% accuracy on an open-set (48 unknown fur-seals) of pairs of fur-seals pictures. That is to say that for a pair of pictures representing either the same fur-seal or two different fur-seals, the current code could tell if it is the same fur-seal or not with an accuracy of 92%. Here is the corresponding ROC curve:



Here follows some false accepted examples and false rejected ones. The model mistakes are mainly due to light exposure, fur-seals' posture and occlusions.

RESULTS ON FACE CLUSTERING :

The obtained code presents great results on face clustering (even for fur-seal faces that the code hasn't seen before). Here follows is an example of two of these clusters: the left one shows a correct example and the right one shows a mistake.

THE FACE DETECTOR :

Step 1: Install Requirements

```
# clone YOLOv5 repo
!git clone https://github.com/ultralytics/yolov5
```

```
Cloning into 'yolov5'...
```

In [1]:

```
# install dependencies if you do not have a pytorch environment
# %pip install -qr requirements.txt
import torch
import os
from IPython.display import Image, clear_output  # to display images
print(f"Setup complete. Using torch {torch.__version__}
({torch.cuda.get_device_properties(0).name if torch.cuda.is_available() else
'CPU'})") # check the gpu
```

```
Setup complete. Using torch 1.12.0 (NVIDIA GeForce RTX 2070 Super)
```

In [3]:

```
cd yolov5
```

```
D:\GitHub\furseal\yolov5
```

Step 3: Train a custom face detection model based on YOLOv5 model

Here, we are able to pass a number of arguments:

- **img:** define input image size
- **batch:** determine batch size
- **epochs:** define the number of training epochs. (Note: often, 3000+ are common here!)
- **data:** Our dataset locaiton is saved in the ./datasets
- **weights:** specify a path to weights to start transfer learning from. Here we choose the generic COCO pretrained checkpoint (the tiny version: yolov5s).
- **cache:** cache images for faster training

```
!python train.py --img 416 --batch 16 --epochs 150 -data
../datasets/data.yaml --weights yolov5s.pt --cache
```

```
train: weights=yolov5s.pt, cfg=, data=../datasets/data.yaml, hyp=data\hyps\hyp.scratch-low.yaml, epochs=150, batch_size=16, imgsz=416, rect=False, res
ume=False, nosave=False, noval=False, noautoanchor=False, noplots=False, evolve=None, bucket=, cache=ram, image_weights=False, device=, multi_scale=Fa
lse, single_cls=False, optimizer=SGD, sync_bn=False, workers=8, project=runs\train, name=exp, exist_ok=False, quad=False, cos_lr=False, label_smoothin
g=0.0, patience=100, freeze=[0], save_period=-1, seed=0, local_rank=-1, entity=None, upload_dataset=False, bbox_interval=-1, artifact_alias=latest
github: up to date with https://github.com/ultralytics/yolov5
YOLOv5  v6.2-57-gf2b8f3f Python-3.9.12 torch-1.12.0 CUDA:0 (NVIDIA GeForce RTX 2070 Super, 8192MiB)

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1, box=0.05, cls=0.
5, cls_pw=1.0, obj=1.0, obj_pw=1.0, iou_t=0.2, anchor_t=4.0, fl_gamma=0.0, hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5, s
hear=0.0, perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0, mixup=0.0, copy_paste=0.0
Weights & Biases: run 'pip install wandb' to automatically track and visualize YOLOv5  runs in Weights & Biases
ClearML: run 'pip install clearml' to automatically track, visualize and remotely train YOLOv5  in ClearML
TensorBoard: Start with 'tensorboard --logdir runs\train', view at http://localhost:6006/
Downloading https://github.com/ultralytics/yolov5/releases/download/v6.2/yolov5s.pt to yolov5s.pt...

  0%|          | 0.00/14.1M [00:00, ?B/s]
  0%|          | 24.0k/14.1M [00:00<02:21, 105kB/s]
  0%|          | 48.0k/14.1M [00:00<02:15, 109kB/s]
  1%|          | 80.0k/14.1M [00:00<01:29, 165kB/s]
  1%|          | 128k/14.1M [00:00<01:03, 232kB/s]
  1%|          | 192k/14.1M [00:00<00:43, 339kB/s]
  2%|          | 272k/14.1M [00:00<00:31, 467kB/s]
  3%|▏         | 384k/14.1M [00:01<00:22, 639kB/s]
  4%|▏         | 544k/14.1M [00:01<00:15, 900kB/s]
  5%|▏         | 768k/14.1M [00:01<00:11, 1.22MB/s]
  8%|▊         | 1.08M/14.1M [00:01<00:07, 1.82MB/s]
 11%|█         | 1.50M/14.1M [00:01<00:05, 2.50MB/s]
 15%|█▍        | 2.06M/14.1M [00:01<00:03, 3.44MB/s]
 20%|██        | 2.78M/14.1M [00:01<00:02, 4.60MB/s]
 27%|██▋       | 3.81M/14.1M [00:01<00:01, 6.39MB/s]
 37%|███▋      | 5.16M/14.1M [00:01<00:01, 8.56MB/s]
 43%|████▎     | 6.01M/14.1M [00:01<00:00, 8.63MB/s]
 53%|█████▎    | 7.43M/14.1M [00:02<00:00, 10.5MB/s]
 60%|██████    | 8.44M/14.1M [00:02<00:00, 9.68MB/s]
 68%|██████▊   | 9.61M/14.1M [00:02<00:00, 10.4MB/s]
 75%|███████▌  | 10.6M/14.1M [00:02<00:00, 9.77MB/s]
 85%|████████▍ | 12.0M/14.1M [00:02<00:00, 11.0MB/s]
```



```
Overriding model.yaml nc=80 with nc=1

                 from  n    params  module                                  arguments
  0                -1  1      3520  models.common.Conv                      [3, 32, 6, 2, 2]
  1                -1  1     18560  models.common.Conv                      [32, 64, 3, 2]
  2                -1  1     18816  models.common.C3                        [64, 64, 1]
  3                -1  1     73984  models.common.Conv                      [64, 128, 3, 2]
  4                -1  2    115712  models.common.C3                        [128, 128, 2]
  5                -1  1    295424  models.common.Conv                      [128, 256, 3, 2]
  6                -1  3    625152  models.common.C3                        [256, 256, 3]
  7                -1  1   1180672  models.common.Conv                      [256, 512, 3, 2]
  8                -1  1   1182720  models.common.C3                        [512, 512, 1]
  9                -1  1    656896  models.common.SPPF                      [512, 512, 5]
 10                -1  1    131584  models.common.Conv                      [512, 256, 1, 1]
 11                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 12           [-1, 6]  1         0  models.common.Concat                    [1]
 13                -1  1    361984  models.common.C3                        [512, 256, 1, False]
 14                -1  1     33024  models.common.Conv                      [256, 128, 1, 1]
 15                -1  1         0  torch.nn.modules.upsampling.Upsample    [None, 2, 'nearest']
 16           [-1, 4]  1         0  models.common.Concat                    [1]
 17                -1  1     90880  models.common.C3                        [256, 128, 1, False]
 18                -1  1    147712  models.common.Conv                      [128, 128, 3, 2]
 19          [-1, 14]  1         0  models.common.Concat                    [1]
 20                -1  1    296448  models.common.C3                        [256, 256, 1, False]
 21                -1  1    590336  models.common.Conv                      [256, 256, 3, 2]
 22          [-1, 10]  1         0  models.common.Concat                    [1]
 23                -1  1   1182720  models.common.C3                        [512, 512, 1, False]
 24      [17, 20, 23]  1     16182  models.yolo.Detect                      [1, [[10, 13, 16, 30, 33, 23], [30, 61, 62, 45, 59, 119], [116, 90, 156, 1
98, 373, 326]], [128, 256, 512]]
Model summary: 270 layers, 7022326 parameters, 7022326 gradients, 15.9 GFLOPs

Transferred 343/349 items from yolov5s.pt
AMP: checks passed
optimizer: SGD(lr=0.01) with parameter groups 57 weight(decay=0.0), 60 weight(decay=0.0005), 60 bias
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01), CLAHE(p=0.01, clip_limit=(1, 4.0), tile_grid_s
ize=(8, 8))

train: Scanning 'D:\GitHub\furseal\yolov5\..\datasets\train\labels.cache' images and labels... 84 found, 0 missing, 0 empty, 0 corrupt: 100%|████████
```


evaluate custom yolov5 detector performance

Training losses and performance metrics are saved to Tensorboard and also to a logfile.

If you are new to these metrics, the one you want to focus on is mAP_0.5

In [ ]:

```
# Start tensorboard
# Launch after you have started training
# logs save in the folder "runs"
%load_ext tensorboard
%tensorboard --logdir runs
```

Run Inference With Trained Weights

```
!python detect.py --weights runs/train/exp/weights/best.pt --img 416 --conf
0.1 --source ../datasets/test/images --save-crop
```

```
detect: weights=['runs/train/exp/weights/best.pt'], source=../datasets/test/i
mages, data=data\coco128.yaml, imgsz=[416, 416], conf_thres=0.1, iou_thres=0.
45, max_det=1000, device=, view_img=False, save_txt=False, save_conf=False, s
ave_crop=True, nosave=False, classes=None, agnostic_nms=False, augment=False,
visualize=False, update=False, project=runs\detect, name=exp, exist_ok=False,
line_thickness=3, hide_labels=False, hide_conf=False, half=False, dnn=False
YOLOv5  v6.2-57-gf2b8f3f Python-3.9.12 torch-1.12.0 CUDA:0 (NVIDIA GeForce RT
X 2070 Super, 8192MiB)

Fusing layers...
Model summary: 213 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs
image 1/4 D:\GitHub\furseal\datasets\test\images\7T5A6512_JPG.rf.4f6efcbaf8ac
7e03321e5fe3d2997343.jpg: 416x416 1 face, 18.0ms
image 2/4 D:\GitHub\furseal\datasets\test\images\7T5A6714_JPG.rf.e86aeb8b17e9
53af980758bb1f84b022.jpg: 416x416 (no detections), 16.9ms
image 3/4 D:\GitHub\furseal\datasets\test\images\7T5A8563_JPG.rf.3d247a6bf3ab
7b04772cb121c0aeb049.jpg: 416x416 (no detections), 18.1ms
image 4/4 D:\GitHub\furseal\datasets\test\images\IMG_7014_JPG.rf.52e08807a6ec
fd159c28a1b30d38d888.jpg: 416x416 2 faces, 17.1ms
Speed: 0.5ms pre-process, 17.5ms inference, 2.4ms NMS per image at shape (1,
3, 416, 416)
Results saved to runs\detect\exp
```

In [8]:

```
#display inference on ALL test images

import glob
from IPython.display import Image, display

for imageName in glob.glob('runs/detect/exp/*.jpg'): #assuming JPG
    display(Image(filename=imageName))
    print("\n")
```

furseal_face_recognition

```
!pip install face_recognition
```

Face Clustering

In [59]:

```
import face_recognition
import matplotlib.pyplot as plt
from skimage import feature as ft
from skimage.color import rgb2gray
from PIL import Image
```

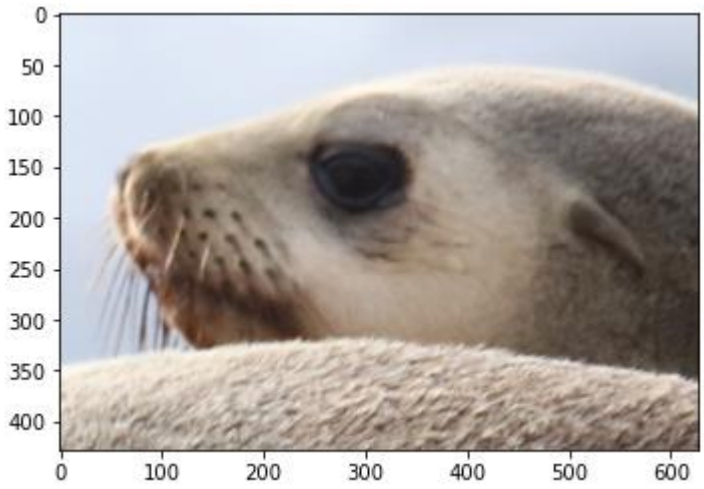
```
from sklearn.cluster import DBSCAN
import numpy as np
import os
```

In [39]:

```
img = Image.open('./yolov5/runs/detect/exp/corps/face/7T5A9436.jpg')
```

In [40]:

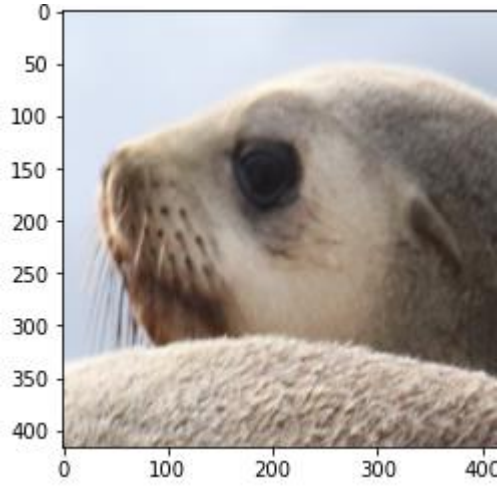
```
plt.imshow(img)
```



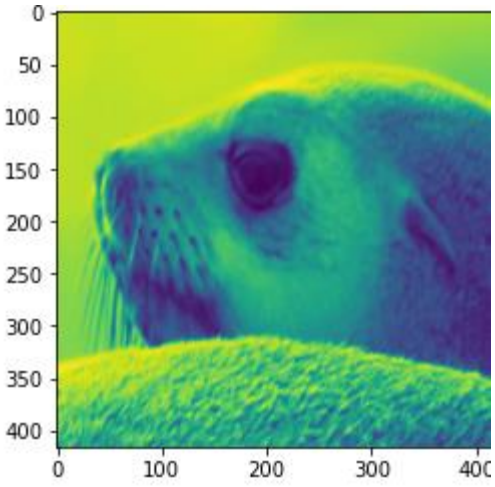

```
img_resize = img.resize((416, 416), Image.Resampling.BILINEAR)
```

In [44]:

```
plt.imshow(img_resize)
```



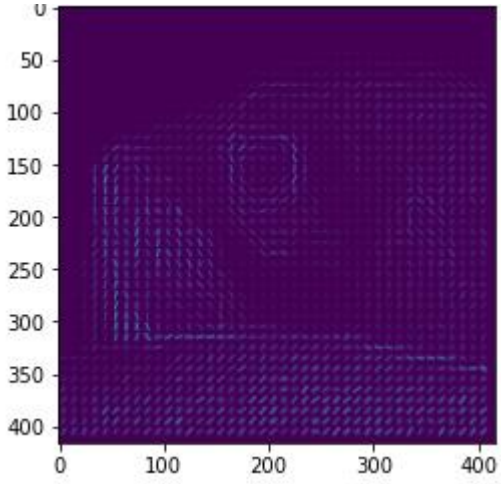

```
gray = rgb2gray(img_resize) / 255.0
plt.imshow(gray)
```




```
features, hog_image = ft.hog(gray,  # input image
                    orientations=12,  # number of bins
                    pixels_per_cell=[10, 10], # pixel per cell
                    cells_per_block=[4, 4], # cells per blcok
                    block_norm = 'L1', #  block norm : str {'L1', 'L1-sqrt',
'L2', 'L2-Hys'}, optional
                    transform_sqrt = True, # power law compression (also known
as gamma correction)
                    feature_vector=True, # flatten the final vectors
                    visualize=True) # return HOG map
```

In [48]:

```
plt.imshow(hog_image)
```




```
features.shape
```

Out[49]:

```
(277248,)
```

In [66]:

```
files = os.listdir('./face/')
result = []
```

```
for file in files:
    img = img=Image.open('./face/'+file)
    img_resize = img.resize((416, 416), Image.Resampling.BILINEAR)
    gray = rgb2gray(img_resize) / 255.0
    features = ft.hog(gray,  # input image
                      orientations=12,  # number of bins
                      pixels_per_cell=[10, 10], # pixel per cell
                      cells_per_block=[4, 4], # cells per blcok
                      block_norm = 'L1', #  block norm : str {'L1', 'L1-sqrt',
'L2', 'L2-Hys'}, optional
                      transform_sqrt = True, # power law compression (also known
as gamma correction)
                      feature_vector=True, # flatten the final vectors
                      visualize=False) # return HOG map
    result.append(features)
```

In [86]:

```
clf = DBSCAN(eps=4, min_samples=3, metric="euclidean", n_jobs=-1)
clf.fit(result)
```

Out[86]:

DBSCAN(eps=4, min_samples=3, n_jobs=-1)

```
np.unique(clf.labels_)
```

Out[92]:

array([-1, 0, 1], dtype=int64)

In [101]:

```
labels = list(clf.labels_)
```

In [108]:

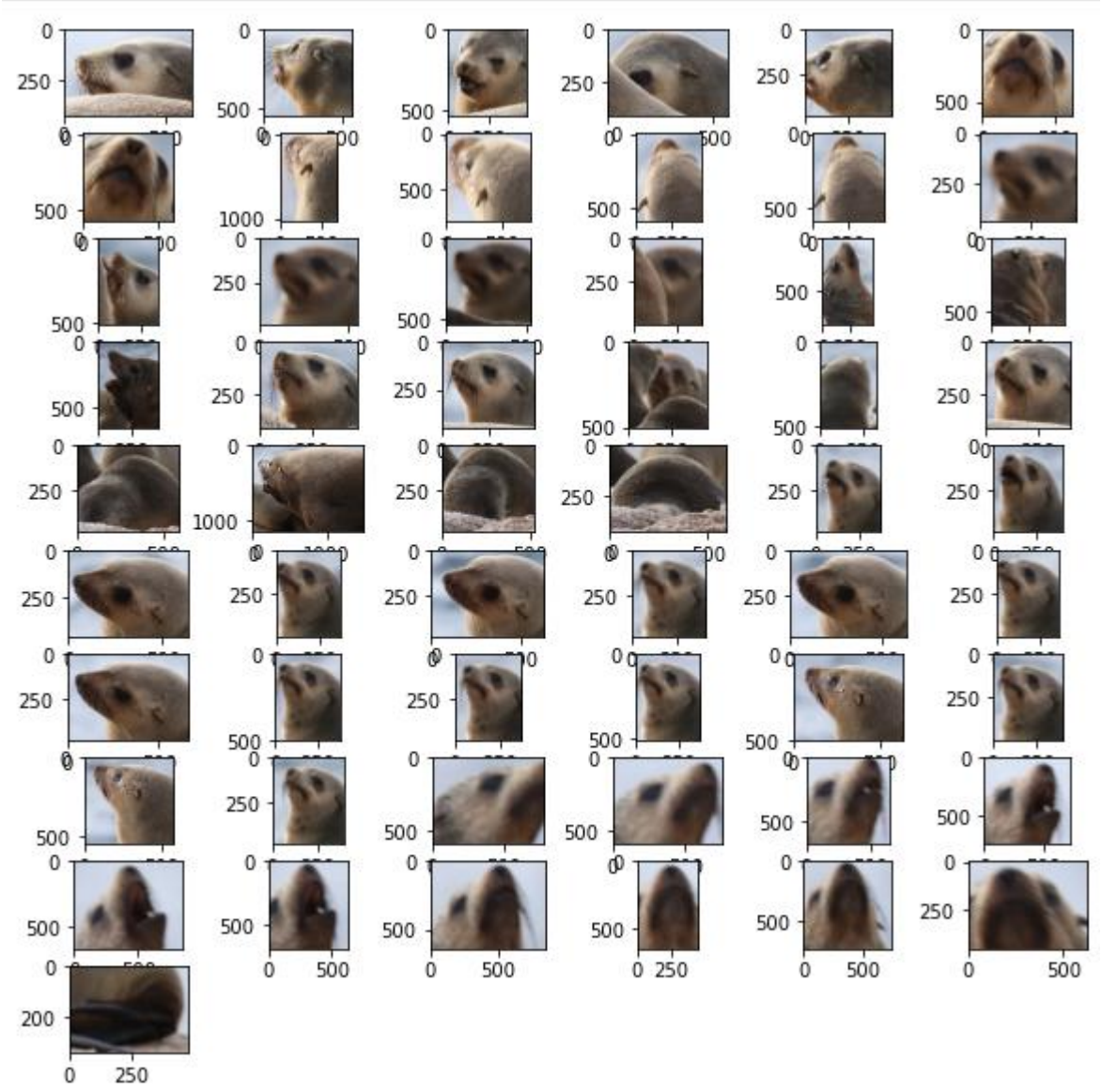
```
[i for i, x in enumerate(labels) if x==1]
```

[34, 36, 41, 42] Out[108]:

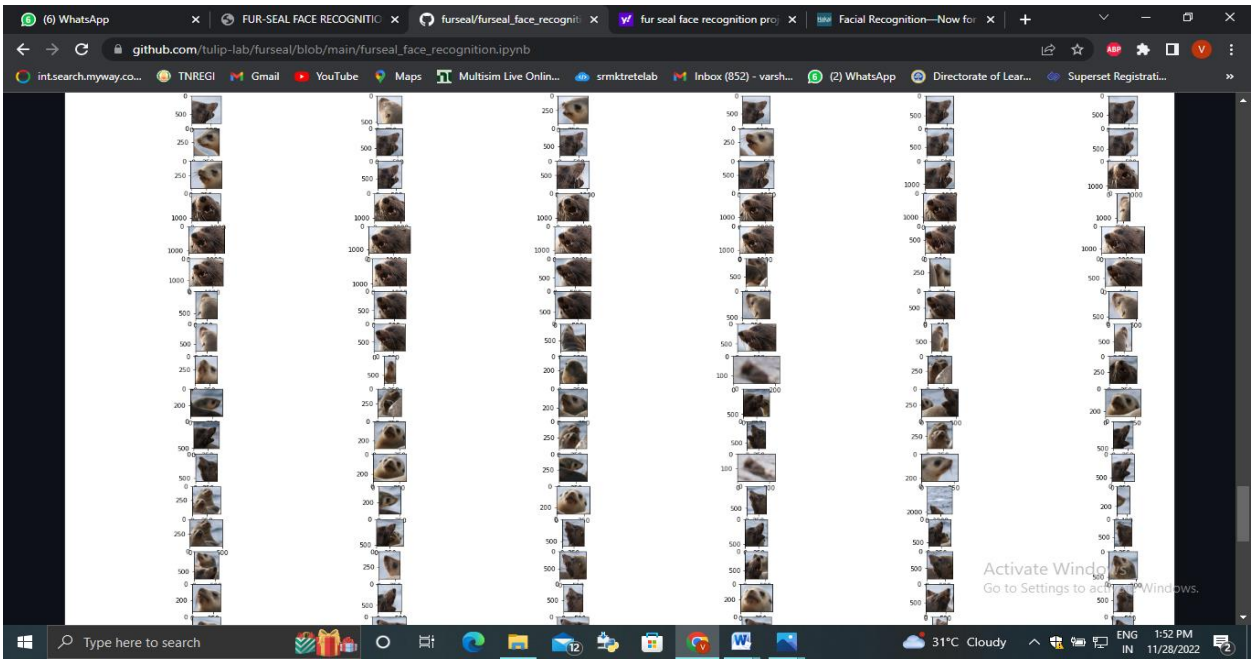
```
plt.figure(figsize=(9, 9))
for m in range(len([i for i, x in enumerate(labels) if x==1])):
    plt.subplot(len([i for i, x in enumerate(labels) if x==1])//6+1, 6 , m +
1)
    img = Image.open('./face/'+files[[i for i, x in enumerate(labels) if
x==1][m]])
    plt.imshow(img)
plt.show()
```

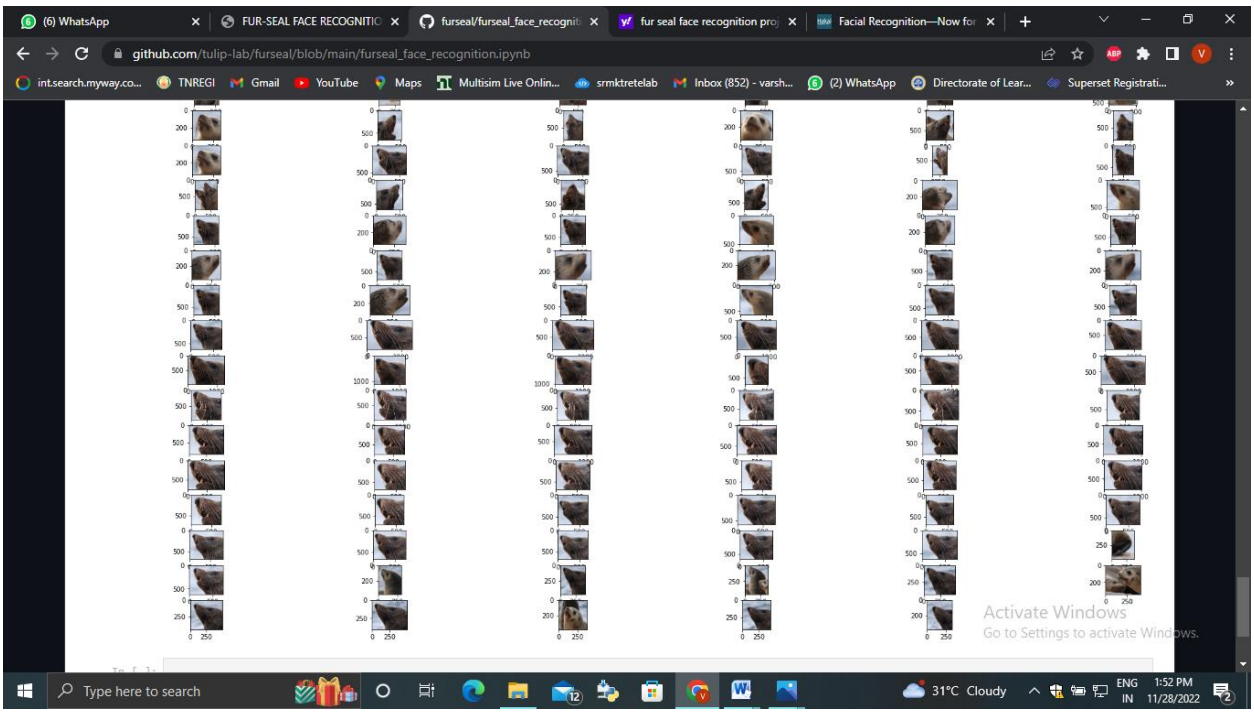

```
plt.figure(figsize=(10, 10))
for m in range(len([i for i, x in enumerate(labels) if x==0])):
```

```
    plt.subplot(len([i for i, x in enumerate(labels) if x==0])//6+1, 6 , m +
1)
    img = Image.open('./face/'+files[[i for i, x in enumerate(labels) if
x==0][m]])
    plt.imshow(img)
plt.show()
```



```
plt.figure(figsize=(30, 30))
for m in range(len([i for i, x in enumerate(labels) if x==-1])):
    plt.subplot(len([i for i, x in enumerate(labels) if x==-1])//6+1, 6 , m +
1)
    img = Image.open('./face/'+files[[i for i, x in enumerate(labels) if x==-
1][m]])
    plt.imshow(img)
plt.show()
```


(6) WhatsApp
FUR-SEAL FACE RECOGNITIO
furseal/furseal_face_recogniti
fur seal face recognition proj
Facial Recognition—Now for
github.com/tulip-lab/furseal/blob/main/furseal_face_recognition.ipynb
int.search.myway.co...
TNREGI
Gmail
YouTube
Maps
Multisim Live Onlin...
srmktretelab
Inbox (852) - varsh...
(2) WhatsApp
Directorate of Lear...
Superset Registrati...
Activate Windows
Go to Settings to activate Windows.
Type here to search
31°C Cloudy
ENG
IN
1:52 PM
11/28/2022


(6) WhatsApp
FUR-SEAL FACE RECOGNITIO
furseal/furseal_face_recogniti
fur seal face recognition proj
Facial Recognition—Now for
github.com/tulip-lab/furseal/blob/main/furseal_face_recognition.ipynb
int.search.myway.co...
TNREGI
Gmail
YouTube
Maps
Multisim Live Onlin...
srmktretelab
Inbox (852) - varsh...
(2) WhatsApp
Directorate of Lear...
Superset Registrati...
Activate Windows
Go to Settings to activate Windows.
Type here to search
31°C Cloudy
ENG
IN
1:52 PM
11/28/2022