APIs:

Detect Number of Faces:

Function: **It will detect the number of faces in an image.**

How To Use: Simply send a request to detect_num_of_face.php in the root library. It takes a file parameter called "file," which should be the image you want our API to analyze at. You can refer to the PHP file to get more information.

How It Works: It has utilized a pre-trained model call "haarcascade_frontalface_default.xml," which is stored under the "data" folder in the root. The model is provided in OpenCV machine learning library. Please refer to the Demo section to see how to try it out by yourself.

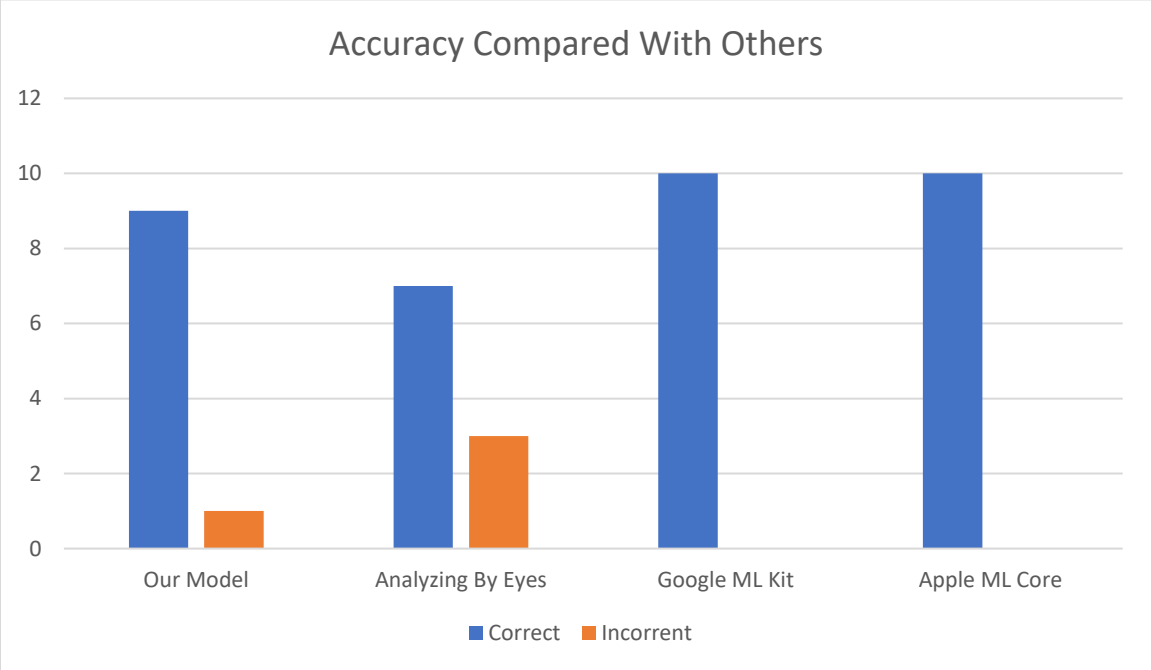
Libraries Used: OpenCV 4

Related Files: "*./detect_num_of_face.php", "./*detect_num_of_face.py", "./data/haarcascades/haarcascade_frontalface_default.xml"

Original Credit: Nagesh Singh Chauhan @ https://medium.com/analytics-vidhya/how-to-build-a-face-detection-model-in-python-8dc9cecadfe9

Additional Note: Please refer to the Common Q&A section's note #1 and note #2 to learn more about our APIs.

Accuracy & Limitations:



Note for the chart: Since for Google ML Kit and Apple ML are not able to be tested in large scale. To make it fair, we have tested on 10 cases. These cases including one face, multiple faces (up to 11 faces), and the environments are classified as pure background, lightly noisy background, moderately noisy background and heavily noisy background. Our model only fails on the case where the background is too noisy and there are only two faces, where the noisy background has affected our model to make it detect 4 faces instead of 2.

Because of the model we have used, our model is only capable of detecting frontal faces in an image. The reason we do not make it able to detect multiple angles of faces is that our test shows it would take a huge amount of time increased in computing where only brings small enhancement to the result, which is conflicted with our Efficiency goal.

Detect Facial expression:

Function: It will detect the facial expression in an image.

How To Use: Simply send a request to detect_mood.php in the root library. It takes a file parameter called "file," which should be the image you want our API to analyze at. You can refer to the PHP file to get more information.

How It Works: It uses four pre-trained models call "haarcascade_frontalface_default.xml," "haarcascade_frontalface_alt2.xml", "haarcascade_frontalface_alt.xml" and "haarcascade_frontalface_alt_tree.xml" under the "data" folder. It also creates a fisherface classifier. Both models and classifier are provided in OpenCV machine learning library.

Libraries Used: OpenCV 4

Related Files: "./detect_mood.php" *"./predict_mood.py"*, "./data/ mood_model.xml", "./data/haarcascades/ haarcascade_frontalface_default.xml", *"./data/haarcascades/haarcascade_frontalface_alt2.xml", "./data/haarcascades/haarcascade_frontalface_alt.xml",*

"./data/haarcascades/haarcascade_frontalface_alt_tree.xml"

Original Credit:

Much of the code is inspired by *van Gent, P* @http://www.paulvangent.com/2016/04/01/emotion-recognition-with-python-opencv-and-a-face-dataset

Image dataset is provided by *Cohn-Kanade database* @http://www.consortium.ri.cmu.edu/ckagree

Additional Note: Please refer to the Common Q&A section's note #1 and note #2 to learn more about our APIs.

Detect the gender :

Function: It will detect the gender in an image.

How To Use: Simply send a request to detect_gender.php in the root library. It takes a file parameter called "file," which should be the image you want our API to analyze at. You can refer to the PHP file to get more information.

How It Works: It has utilized a pre-trained model call "gender_detection.model ," which is stored under the "data/ gender_train_data" folder in the root. The model is provided in Keras machine learning library. Please refer to the Demo section to see how to try it out by yourself.

Libraries Used: OpenCV 4, Tensorflow, Keras

Related Files: "*./detect_gender.php", "./*detect_gender.py", "./data/ gender_train_data / gender_detection.model"

Original Credit: Arun Ponnusamy @ https://github.com/arunponnusamy/gender-detection-keras

Additional Note: Please refer to the Common Q&A section's note #1 and note #2 to learn more about our APIs

Detect the object:

Function: It will identify each object in an image.

How To Use: Simply send a request to object_detection.php in the root library. It takes a file parameter called "file," which should be the image you want our API to analyze at. You can refer to the PHP file to get more information.

How It Works: It has utilized a pre-trained model call "yolov3.cfg" which is stored under the "data\.cvlib\object_detection\yolo\yolov3/" folder in the root. The model is provided in YOLO https://pjreddie.com/darknet/yolo/ trained by the dataset of COCO "http://cocodataset.org/#home". Please refer to the Demo section to see how to try it out by yourself.

Libraries Used: OpenCV 4, Tensorflow, Keras, pillow

Related Files: "./object_detection.*php", "./*object_detection.py", "./ data\.cvlib\object_detection\yolo\yolov3/yolov3.cfg", "./ data\.cvlib\object_detection\yolo\yolov3/yolov3..weights", "./ data\.cvlib\object_detection\yolo\yolov3/ yolov3_classes.txt"

Original Credit: Arun Ponnusamy @ https://github.com/arunponnusamy/cvlib

Additional Note: Please refer to the Common Q&A section's note #1 and note #2 to learn more about our APIs

Face Labeling:

Function: It will identify each object in an image.

How To Use: Simply send a request to FaceRecognizer.php in the root library. It takes a file parameter called "file," which should be the image you want our API to analyze at. It also takes a string parameter called "mode," which is used to detect the current mode the server should run on. Giving "0" will make the server run at normal mode (using the training data the clients have uploaded), where giving "1" will make the server run at demo mode (using the training data we have provided, please refer to the Demo section to get more detail). You can refer to the PHP file to get more information. When using the normal mode, you would need to upload your training data by sending a request to upload_img_for_face_labeling.php. It takes a file parameter called "file," which should be the image of a training data. It also takes two string parameters, one is called "label," which is the label of the image used for training, and another one is called "img_name, " where it is used for the name of each training image (you would like to keep different training image on the server). Also, when you want to start from scratch, you will have to send a request to clear_all_photos_for_labeling.php to delete all previous training data and then upload them again.

How It Works: It has utilized the pretrained face recognition model called "lbpcascades_frontalface.xml," where it is possible to detect frontal faces. Then It is going to train a model with labeled images from clients. After that, it will be able to predict which label the image from clients is belong to.

Libraries Used: OpenCV 4

Related Files: "./FaceRecognizer.*php", "./*FaceRecognizer.py", "./ data\lbpcascades\lbpcascades_frontalface.xml"

Original Credit: Arun Ponnusamy @ https://github.com/arunponnusamy/cvlib

Additional Note: Please refer to the Common Q&A section's note #1 and note #2 to learn more about our APIs

Common Q&A:

1. **The overview of each of the APIs**
   Each of our API contains two types of files. One is a PHP file, which is used to handle your clients' request and return the results. Another one is a Python file, which is used to handle the actual machine learning request. If what you need is just normal use of the API, there is even no need for you to look at the Python file.
2. **When do I need to look at the Python Files**
   Our answer would be: it depends on you! You are the developers, so if you would like to modify the detail of our facial machine learning API, please do not hesitate! You have the choice to choose to change the parameters, retrain our models or even complete replace our model with your own's.

3. **What models do you provide**

   We have provided you some models for you to choose. Some of them are pre-trained models from OpenCV or TensorFlow, some of them are models trained by us and some of them are models ready to be trained by you. By comparing different models, we have chosen the ones that we think is the best fit for most of developers in our current APIs. You can also check out different models we have provided. Of course, if you would like to take use of your own ones, please feel free to do so.

4. **Environment Requirements**

   Minimum Requirements:

   PHP 7.0 or Latter (With GD library support)
   Python 3.*
   OpenCV 4
   TensorFlow
   Numpy

   Suggested Requirements (Our tested environment):

   Ubuntu 18.10
   25GB SSD or Larger
   1GB RAM or Larger
   Apache 2
   PHP 7.0 or Latter (With GD library support)
   Python 3.*
   OpenCV 4
   TensorFlow
   Numpy

   # Please also be notified that there might be more running libraries required when you deploy our APIs on your server, this is varied between the server system you are currently using and the libraries you currently have.

Demo:

1. **What we have provided**

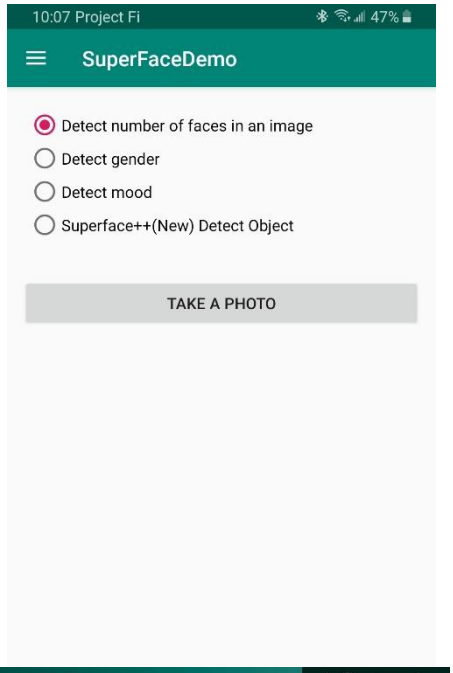
   We have provided you with a fully ready-to-run Android demo for all of our current APIs. You will find a DemoApp.apk under the root folder, where you can also choose to compile the Android APK using Android Studio on the App folder, which is also stored under the root folder. Please be notified that we do not compress the image before uploading to our server, so if you are running the App on Data Connection, the App might consume a lot of data depended on your phone's camera. We are not responsible for your Data connection Charges.

2. **Environment to run it**

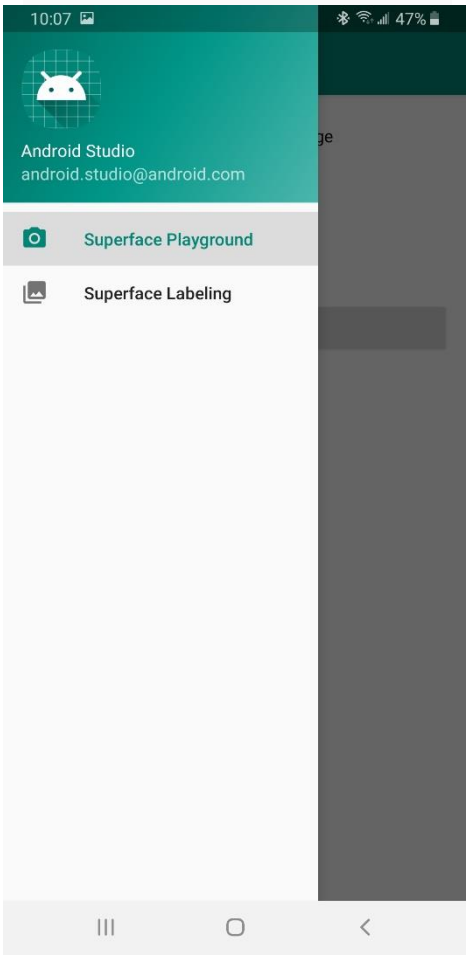
   Our demo App should be able to run on Android with API 23 or newer, which is Android 6.0. We have tested our demo App on Samsung Galaxy S8+ (G955U) running on Android Pie (9.0) which is API 28.

3. How to use it

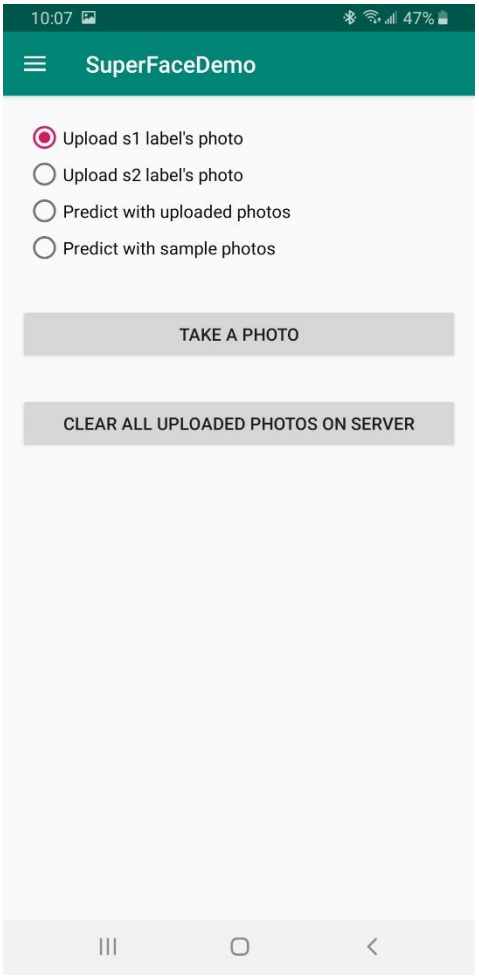
Our Demo App contains two pages, where one is called SuperFace Playground, and another one is called SuperFace Labeling.



The left side's image is the first page (Superface Playground) of our Demo App, where it contains four APIs for you to test on. To try it out, just simply choose the option you would like to try and click take a photo. After taking a photo and confirm it, you will have to wait for a short time for our server to respond you. When the App gets the respond from the server, it would show you the result immediately.



By swiping from left side to the right side, you will see a navigational drawer like the image shown on the left side. After seeing this, you just need to simply select a choice from the two "Superface Playground" and "Superface Labeling."



The left side's image is the second page (Superface Labeling) of our Demo App, where it contains one API for you to test on. There are two modes for you to try, one is the normal mode, and another one is the demo mode. For normal mode, we have provided you with two labels. You will need to upload at least ten photos for the first person (s1 label), and another ten photos for the second person (s2 label). Please be notified that each time you re-open the app, re-enter the page or want to change the person(s) you want to be labeled on, you will have to manually click the "CLEAR ALL UPLOADED PHOTOS ON SERVER, " this is because the photos on the sever will not be automatically delete. However, this does not affect when you choose to try on the demo mode. This is because running this API on the demo mode will delete all previous uploaded photos by the clients. This also means that when you have run this API on demo mode for once, you will have to re-upload both s1 and s2's photos.

When you are running the second page (Superface Labeling) on the demo mode (choosing the forth option “Predict with sample photos”), you will have to take a photo on the left side’s image. You could also find this image at “./data/face_label_test_data/test_img.jpg.”