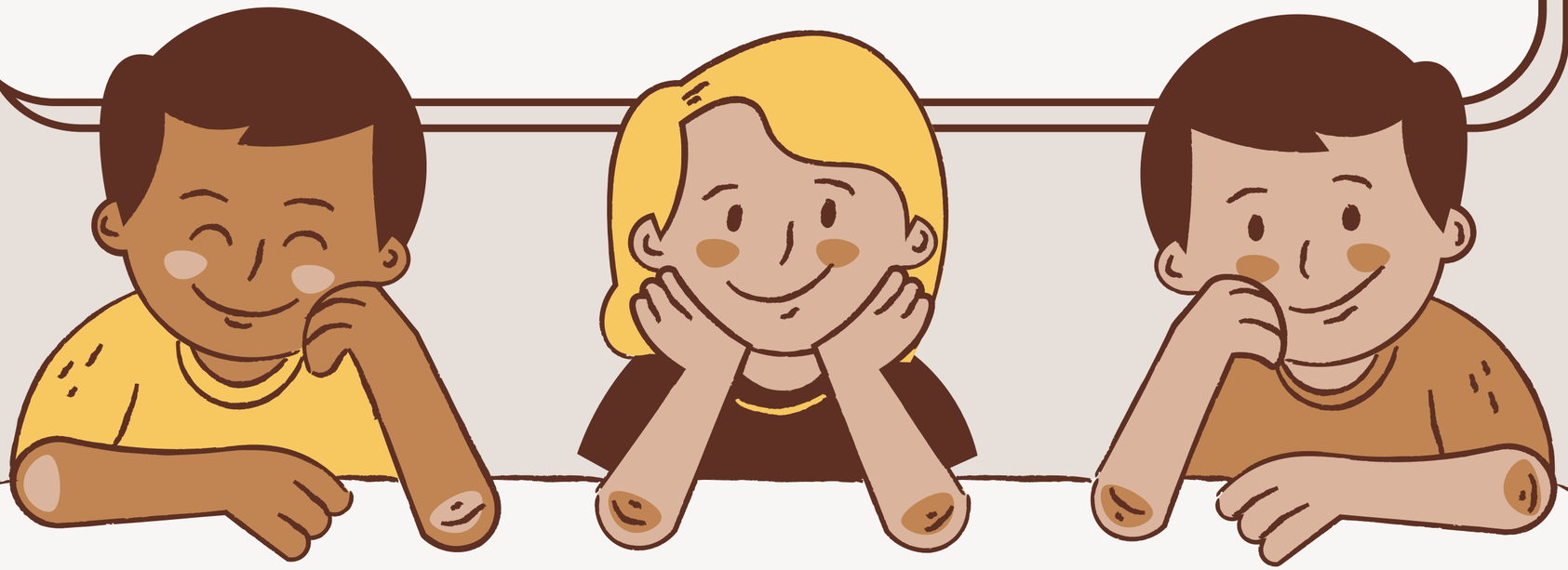
DATING APP

TEAM 05

https://github.com/lon5948/LAIAIAI

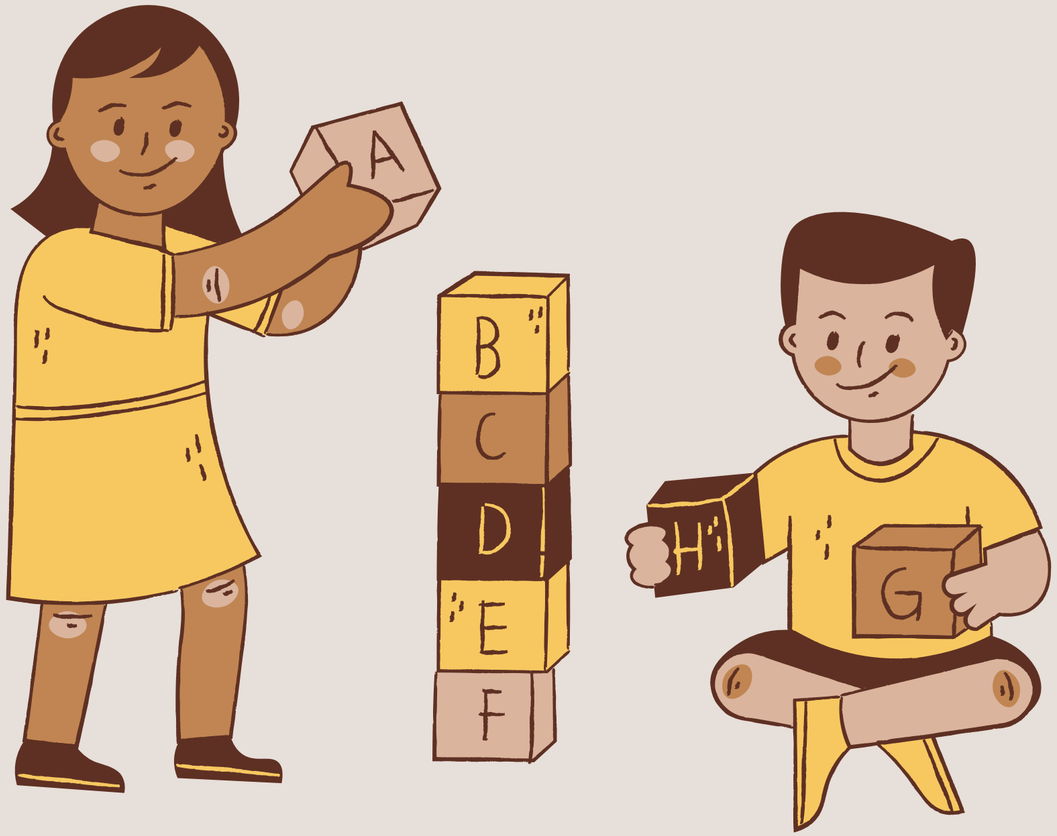


Outline

- introduction
- input/output
- evaluation metric
- four features
- future work

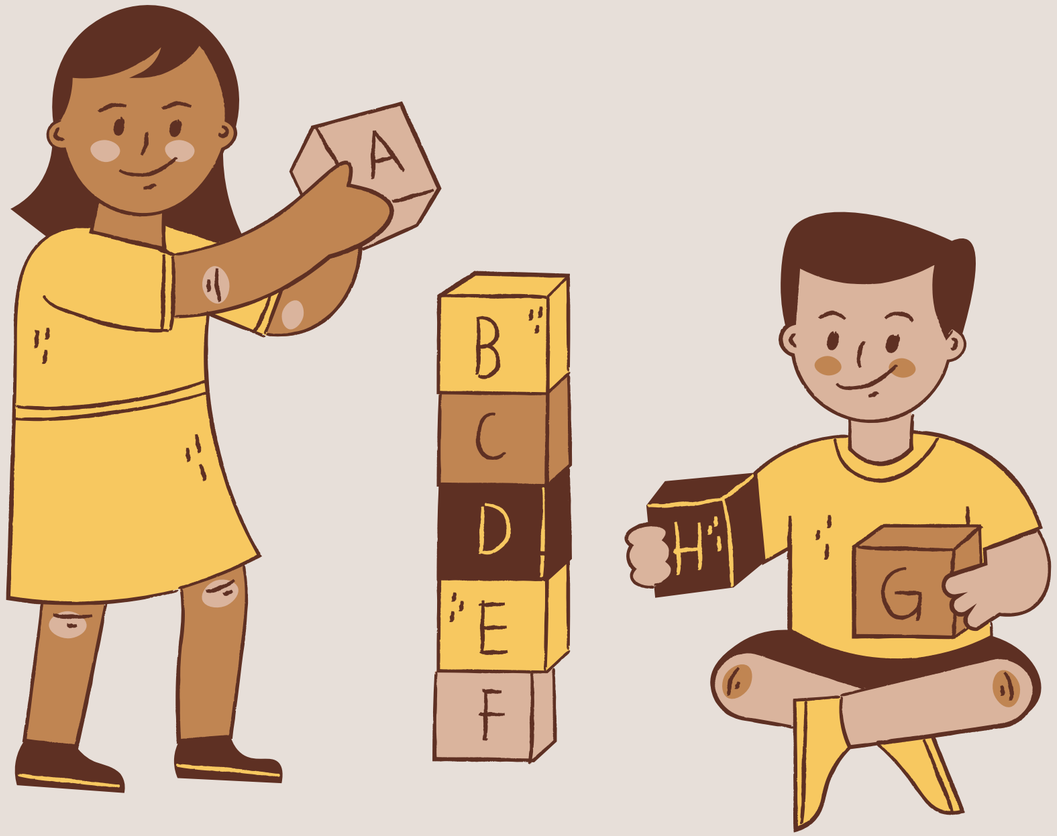
INTRODUCTION

Due to pandemic, online dating has become the safest and most romantic way to meet a partner. But the selection of photos is time consuming, thus, we'd like to introduce to you a Dating App that can classifiy human's faces and output photos with a face that you adore.



In this project, we will implement four classifiers on human faces, gender, face shape, skin color and eye color.

We will give an image of human face as input and output the four attributes that our system judges.



Input: image path

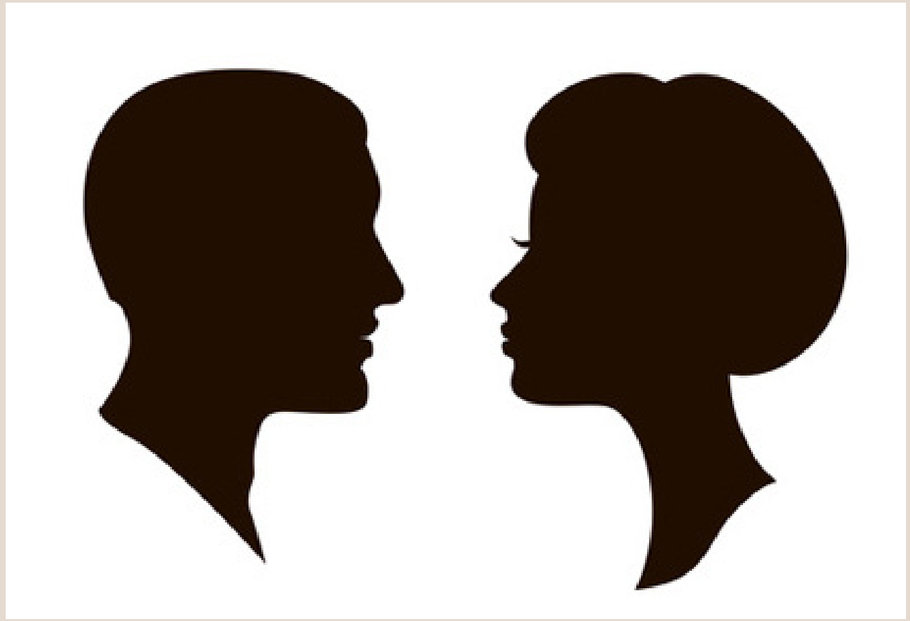
Output: string

- gender： Man / Woman
- face shape： diamond / oblong / oval / round / square / triangle
- race： Asian / Indian / Black / White / Middle eastern / Latino hispanic
- eye color： Black / Brown / Green / Blue

Evaluation Metric

$$accuracy = \frac{1}{n}\sum_{i=1}^{n} f(x) \quad where\ f(x) = \begin{cases} 1 & if\ output = label \\ 0 & otherwise \end{cases}$$

GENDER



Model

BEUARAY CORP.

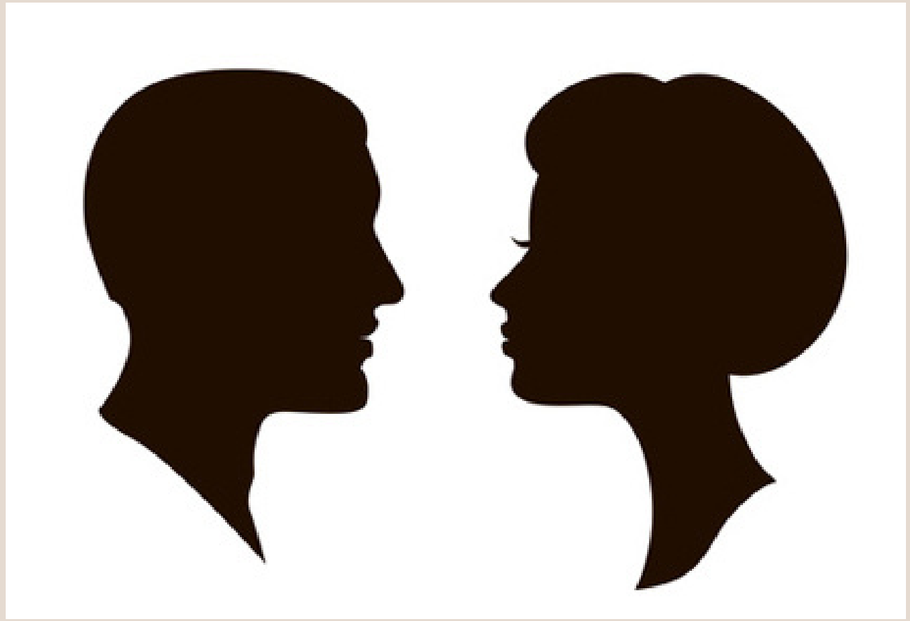
Baseline 1
keras and cvlib

BEUARAY CORP.

Baseline 2
OpenCV and CNN

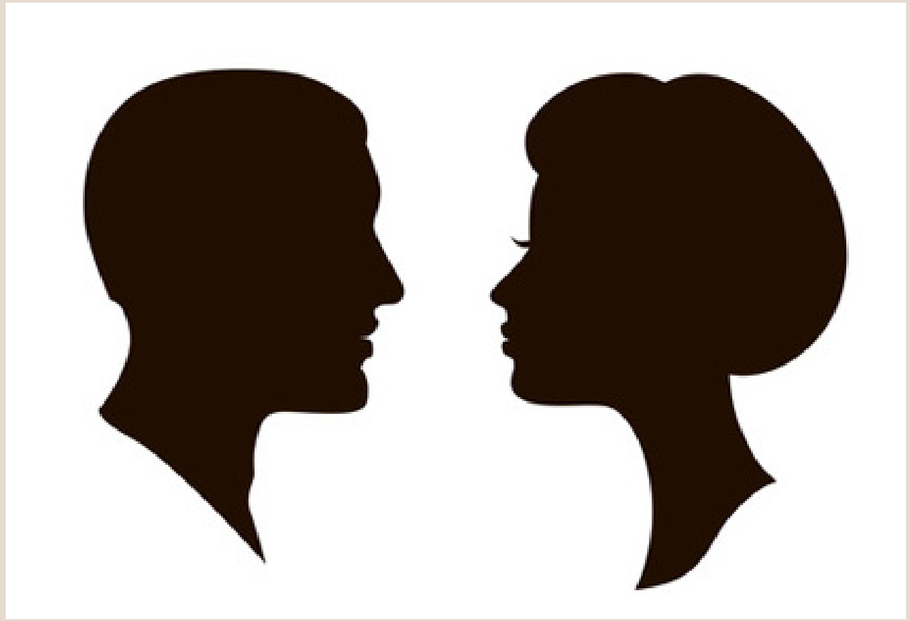
Model
dlib and deepface

Baseline 1

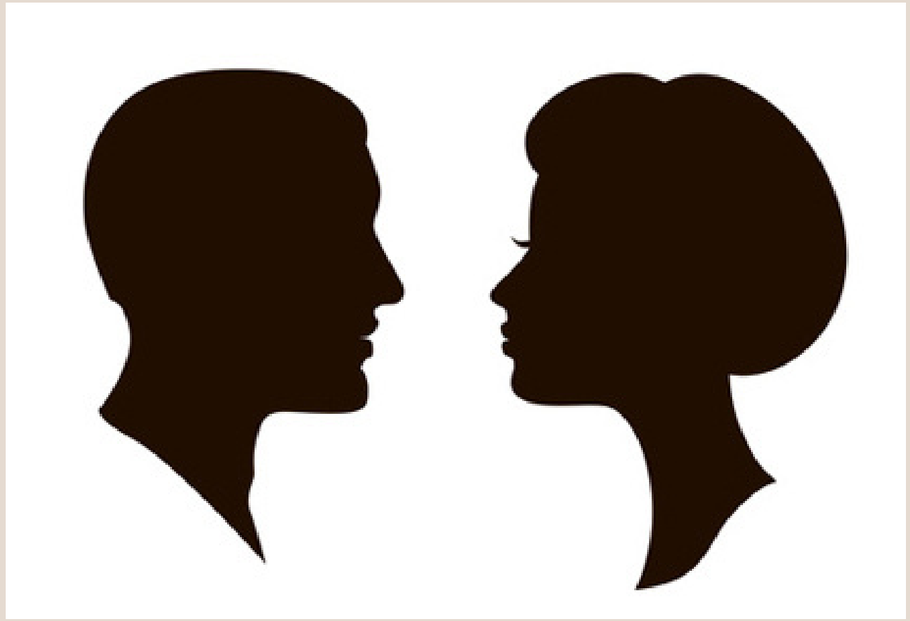


- import cvlib to use detect_face ( )
- import Keras to built model
- convolution, batch normalization, MaxPooling, dropout
- batch normalizaton: gradient vanishing

Baseline 2



- import OpenCV and use haarcascade_frontalface_alt.xml
- import Keras to built model
- convolution, MaxPooling, dropout
- use dropout to avoid overfitting



Model

- import face_recognition to use face_recognition.face_locations ( )
- import deepface and VGGface
- Use pre_trained model gender_model_weights.h5

Detail Implement of the Model

```python
def transform_face_array2gender_face(self,face_array,grayscale=False,target_size = (224, 224)):
    detected_face = face_array
    if grayscale == True:
        detected_face = cv2.cvtColor(detected_face, cv2.COLOR_BGR2GRAY)
    detected_face = cv2.resize(detected_face, target_size)
    #img_pixels = image.img_to_array(detected_face)
    img_pixels = np.asarray(detected_face,'f')
    img_pixels = np.expand_dims(img_pixels, axis = 0)
    #normalize input in [0, 1]
    img_pixels /= 255
    return img_pixels
```

Detail Implement of the Model

```
def predict_gender(self, face_image):
    image_preprocesing = self.transform_face_array2gender_face(face_image)
    gender_predictions = self.model.predict(image_preprocesing )[0,:]
    if np.argmax(gender_predictions) == 0:
        result_gender = "Woman"
    elif np.argmax(gender_predictions) == 1:
        result_gender = "Man"
    return result_gender

def loadModel(self):
    model = VGGFace.baseModel()
    #------------------------
    classes = 2
    base_model_output = Sequential()
    base_model_output = Convolution2D(classes, (1, 1), name='predictions')(model.layers[-4].output)
    base_model_output = Flatten()(base_model_output)
    base_model_output = Activation('softmax')(base_model_output)
    #------------------------
    gender_model = Model(inputs=model.input, outputs=base_model_output)
```

ACCURACY
WITH ONE PERSON PER PICTURE

	BASELINE 1	BASELINE 2	MODEL
MAN	0.85	0.92	0.98
WOMAN	0.75	0.81	0.82

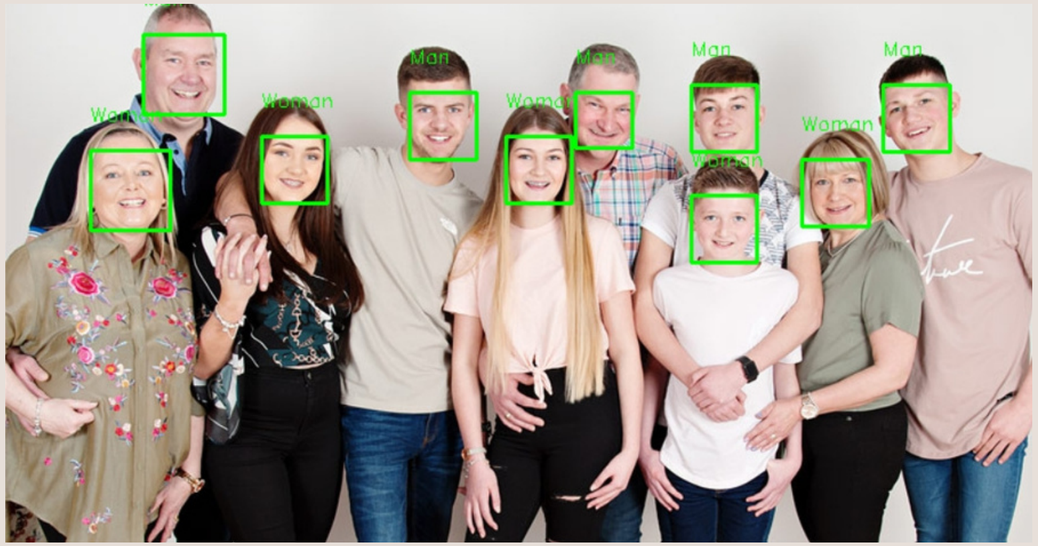
Testing dataset

- US criminal photos, gender are labeled. https://www.kaggle.com/datasets/davidjfisher/illinois-doc-labeled-faces-dataset

- kaggle :https://www.kaggle.com/datasets/ashwingupta3012/male-and-female-faces-dataset

MANY PEOPLE PER PICTURE

MANY PEOPLE PER PICTURE



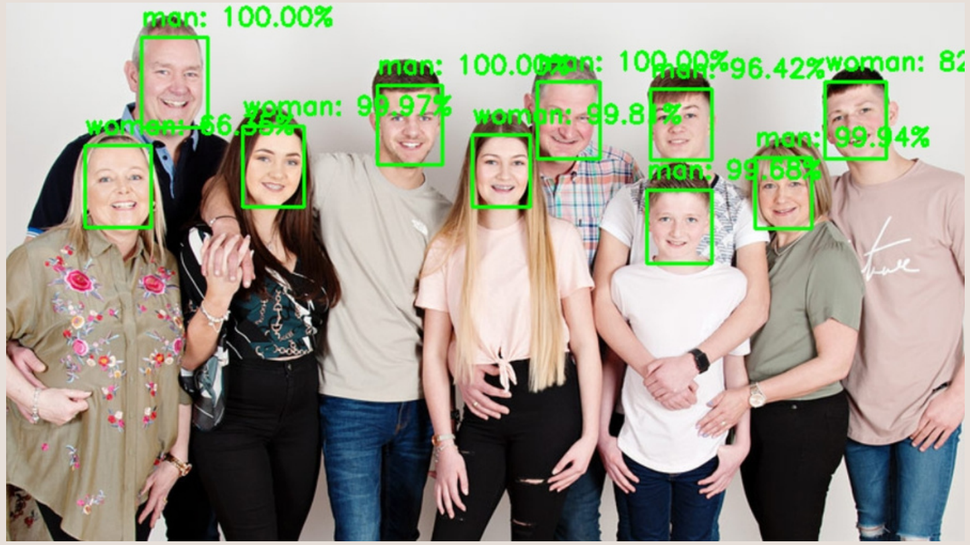
From Left to Right:

W M W M W M M M W M

Model

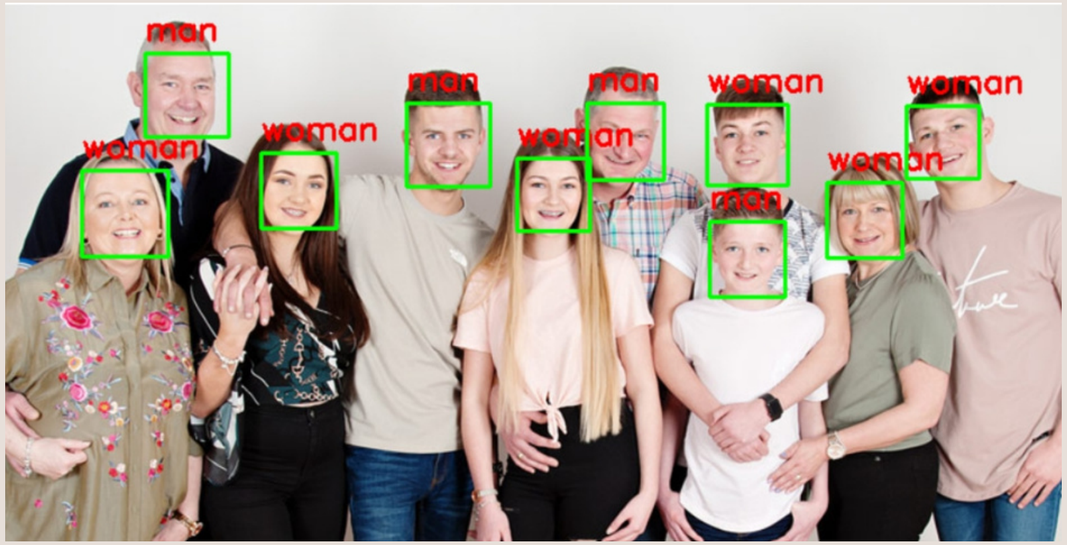
accurancy:10/10

MANY PEOPLE PER PICTURE



Baseline 1

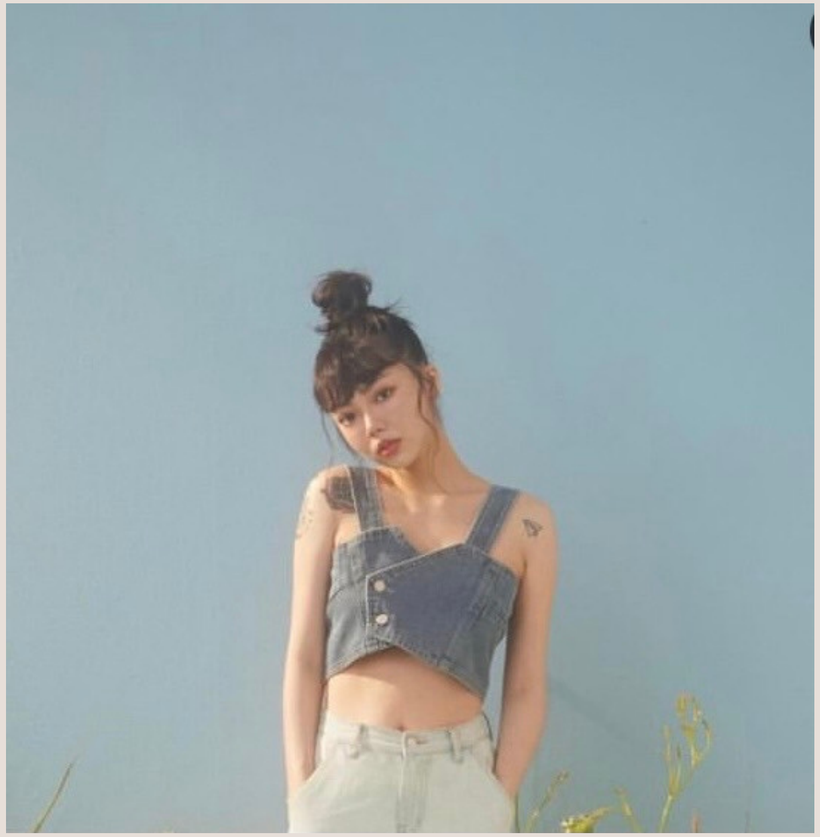
accurancy:8/10



Baseline 2

accurancy:8/10

UNCLEAR PHOTO

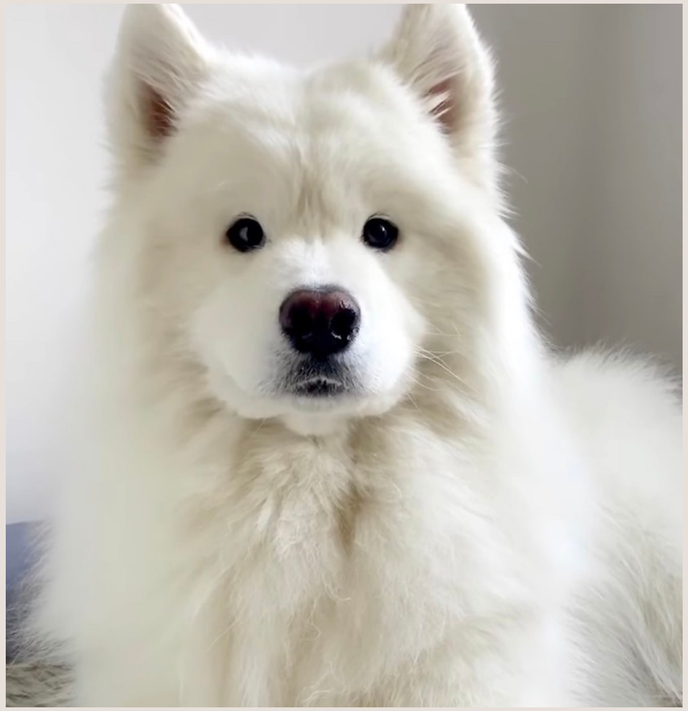


Baseline 1: X

Baseline 2: X

Model: Woman

NOT A PERSON



Baseline 1: Man

Baseline 2: X

Model: X

Baseline 1: Woman

Baseline 2: X

Model: X

REFERENCE

BASELINE 1

https://github.com/arunponnusamy/gender-detection-keras

BASELINE 2

https://github.com/nman7/Facial-Gender-classification

MODEL

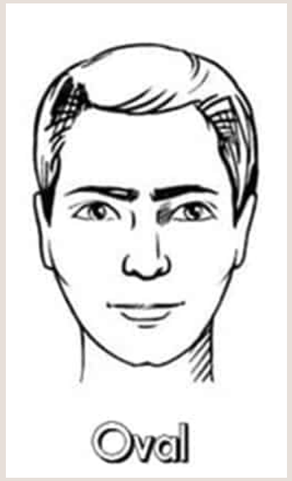
https://github.com/juan-csv/Face_info
https://www.researchbank.ac.nz/bitstream/handle/10652/5395/MCOomp_2021_Changjin%20Wang%20%2B.pdf?sequence=1&isAllowed=y

FACE SHAPE

Related Work

- mood detection
- use distance/angles to determine result
- KNN algorithm

Face Shape



Dataset

kaggle

- no classification
- final test
- 100 pictures

github

- diamond/oblong/oval/round/square/triangle
- 300 pictures each face shape
- calculate the accuracy

Model

BEUARAY CORP.

Baseline 1

face_recognition

BEUARAY CORP.

Baseline 2

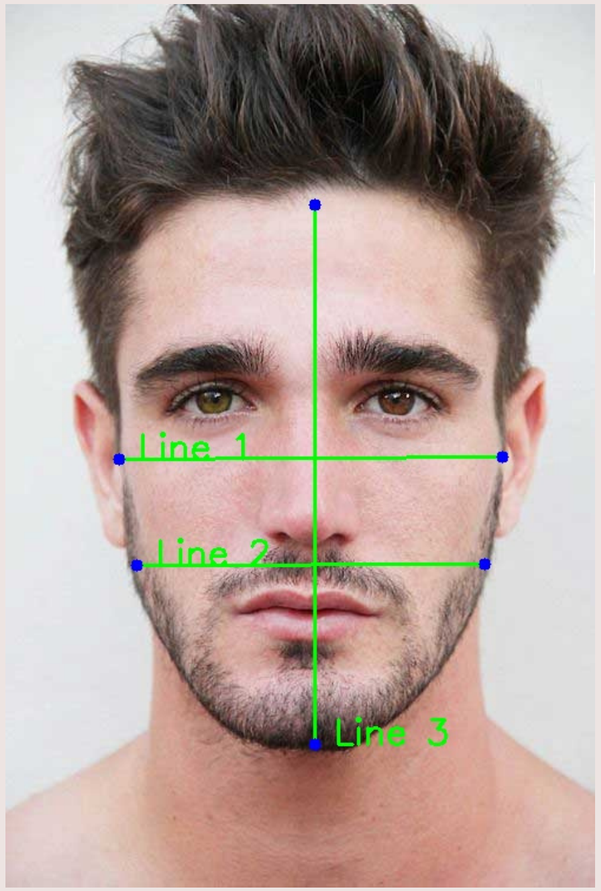
K Means Clustering

Model

K Nearest Neighbor (KNN)

Baseline 1

- import face_recognition
- calculate the angle of jaw
- classification: three types



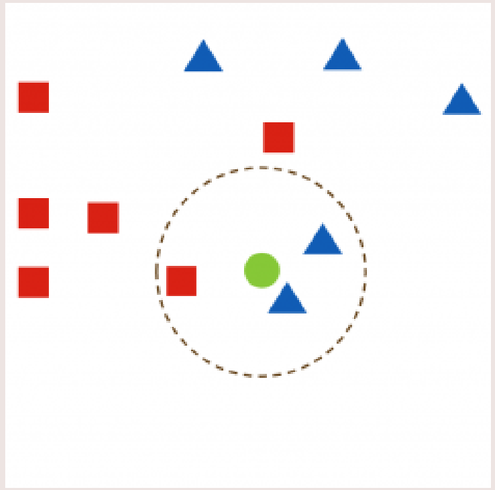
Baseline 2

- K-means Clustering
- draw three lines
- calculate the angle of jaw
- classification: six types
- Unsupervised Learning

K NEAREST NEIGHBORS

- determine k-value
- calculate the distance from yourself to every neughbors
- find k nearest neighbors
- join the group
- if can't join the group, do above steps again and again

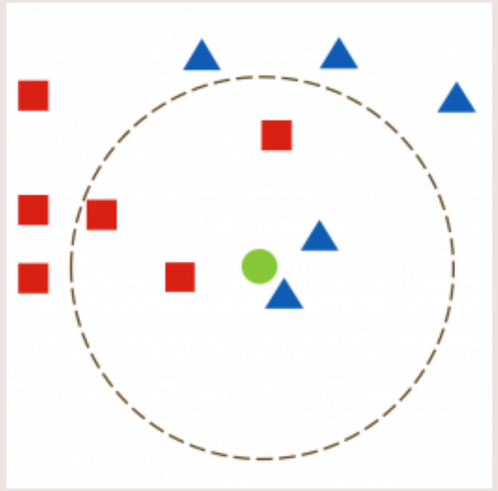
IS K-VALUE IMPORTANT?



find 3 nearest neighbors,
classification is blue triangle

K=3

find 5 nearest neighbors,
classification is red square



K = 5

Training KNN

- data.pickle
- target.pickle
- scikit-learn (KNeighborsClassifier)
- KNN_model.sav

Face Detector

a pretrained face detecting classifier

Landmark Detector

pretrained external algorithm

Implementation

- face_detector → rect
- landmark_detector → 68 points
- Calculate Euclidean Distance
- KNN model
- get face shape

Model

- KNN algorithm
- Supervised Learning
- faster
- classification: six types

RESULT

	BASELINE 1	BASELINE 2	MODEL
SQUARE	47.3 %	37 %	93 %
OBLONG	X	26 %	91 %
OVAL	X	32 %	81 %
ROUND	2.51 %	39 %	88 %
DIAMOND	X	25 %	98 %
TRIANGLE	45 %	13 %	84 %

ANALYSIS

- face shape in some images are ambiguous
- BASELINE 1 only three types but higher accuracy
- BASELINE 2 has more types but lower accuracy
- no matter which face shape, performance of KNN model is the best

ERROR ANALYSIS

Square	93 %
oblong	91 %
OVAL	81 %
round	88 %
diamond	98 %
triangle	84 %

- some images misclassified by both models
  - Oval misclassified as Triangle
  - Mostly “Asian” Oval more mistaken as Round
  - Mostly “White” Round more mistaken as Oval
- six types has the highest performance

OVAL

Fine-Tuning

- oval / triangle classification
- oval / round classification
- according to race, age, gender

REFERENCE

FACE RECOGNITION

https://github.com/ageitgey/face_recognition

K MEANS ALGORITHM

https://github.com/BhagyaPerera/face-shape-recognition-system-

KNN MODEL

https://github.com/AakashBelide/gesture_control_and_mood_detection

PAPER

https://www.researchgate.net/publication/337386649_Face_shape_classification_using_Inception_v3

SKIN COLOR

Dataset

1. YFCC-100M Flicker dataset
2. LFWA+ dataset
3. Illinois DOC labeled faces dataset (Kaggle)

Baseline 1

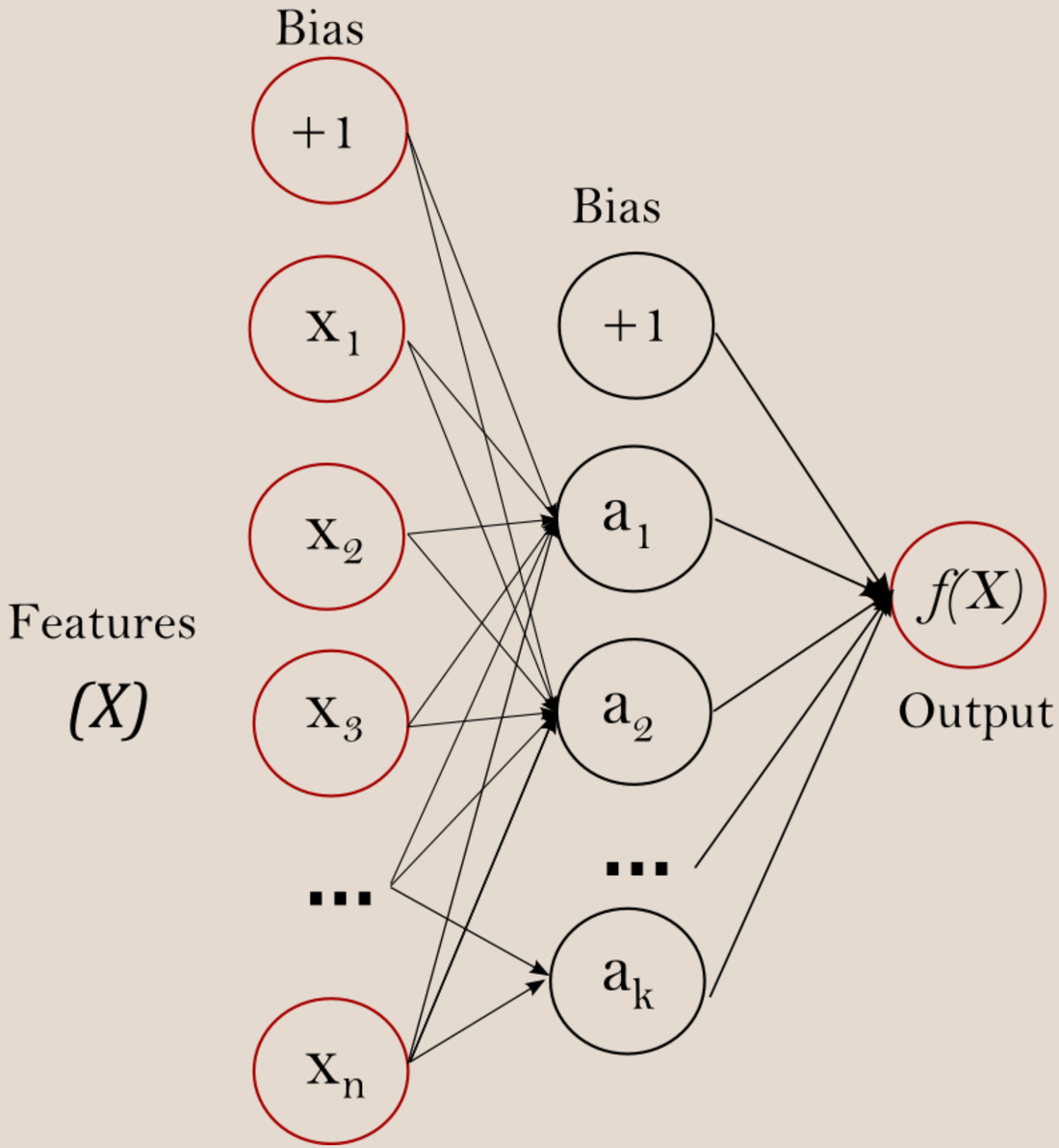
- White / Black / Asian / Indian
- dlib’s CNN-based face detector
- Pretrained model ResNet-34
- FairFace: Face Attribute Dataset for Balanced Race, Gender, and Age for Bias Measurement and Mitigation

Baseline 2

- Asian / Indian / Black / White / Middle eastern / Latino hispanic
- face_recognition.face_locations()
- keras’ sequential model
- DeepFace: Closing the Gap to Human-Level Performance in Face Verification

Model

- White, Black, Asian
- input : images
- output : string
- dlib’s CNN-based face detector
- MLP classifier



MLP
Multilayer Perceptron

solver = ‘adam’, hidden_layer_size = (128, 128),
activation = ‘relu’, max_iter = 5000, tol = 1e-4

RESULT

	BASELINE 1	BASELINE 2	MODEL
ACCURACY	90.93195266272189%	89%	91.02071005917l6%
LATENCY	5 min / 200 images	151 min / 6760 images	57 min / 6760 images

ANALYSIS

- Baseline 2 detect face with HOG
- Baseline usiing ResNet

MODEL RESULT

	N = 100	N = 200	N = 1000	N = 3000	N =6760
ACCURACY	98%	95.5%	93%	92.36666%	91.0207100591716%

ERROR ANALYSIS

- Number of labels :
  - model classifier < testing data

REFERENCE

BASELINE 1

https://github.com/dchen236/FairFace

BASELIN2

https://github.com/juan-csv/Face_info
https://github.com/serengil/deepface

MODEL

https://github.com/wondonghyeon/face-classification

EYE COLOR

Model

BEUARAY CORP.

Baseline 1

MTCNN detector

BEUARAY CORP.

Baseline 2

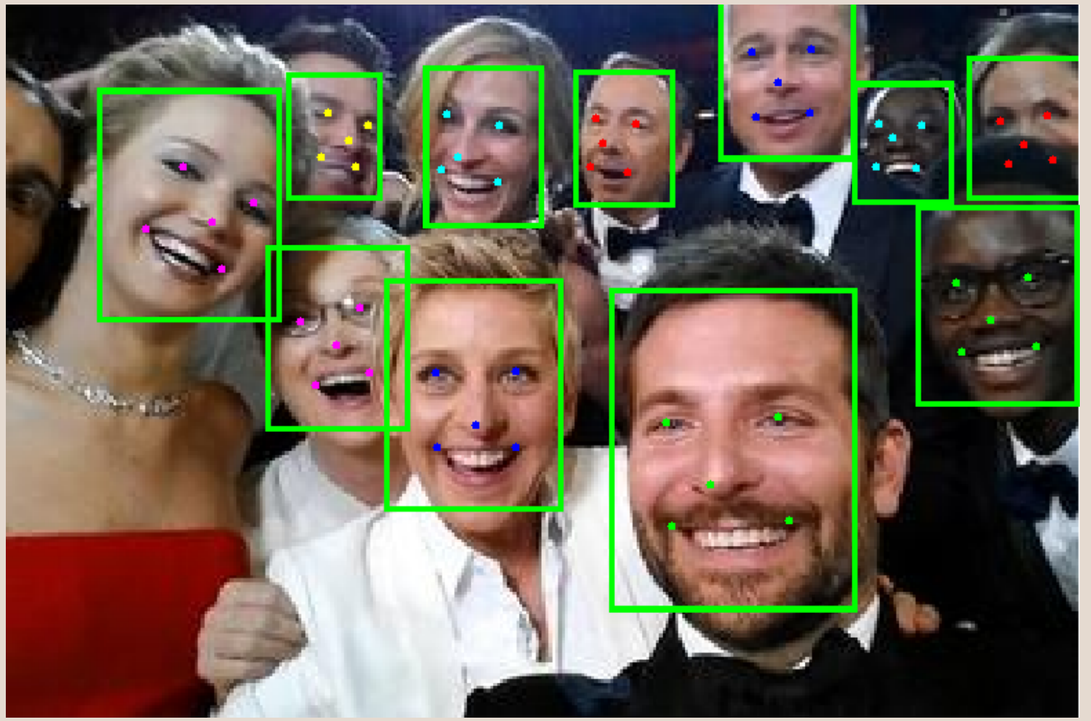
Pupil detection using DLIB

Best model

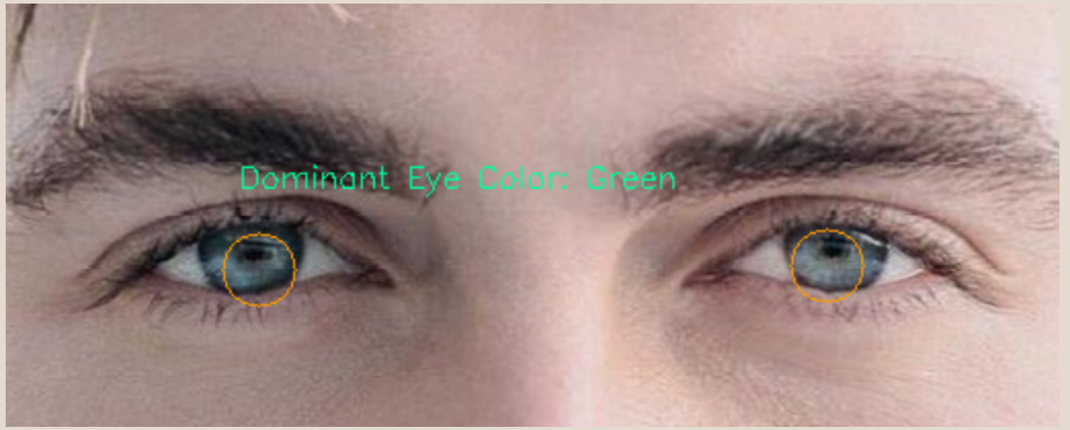
Iris extraction and detection

Baseline 1
MTCNN detector

- use mtcnn to catch 5 points.
- take left eye, right eye and eye's distance.
- approximate eye radius by eye_radius = eye_distance/15
- circle eye area and detect HSV color.

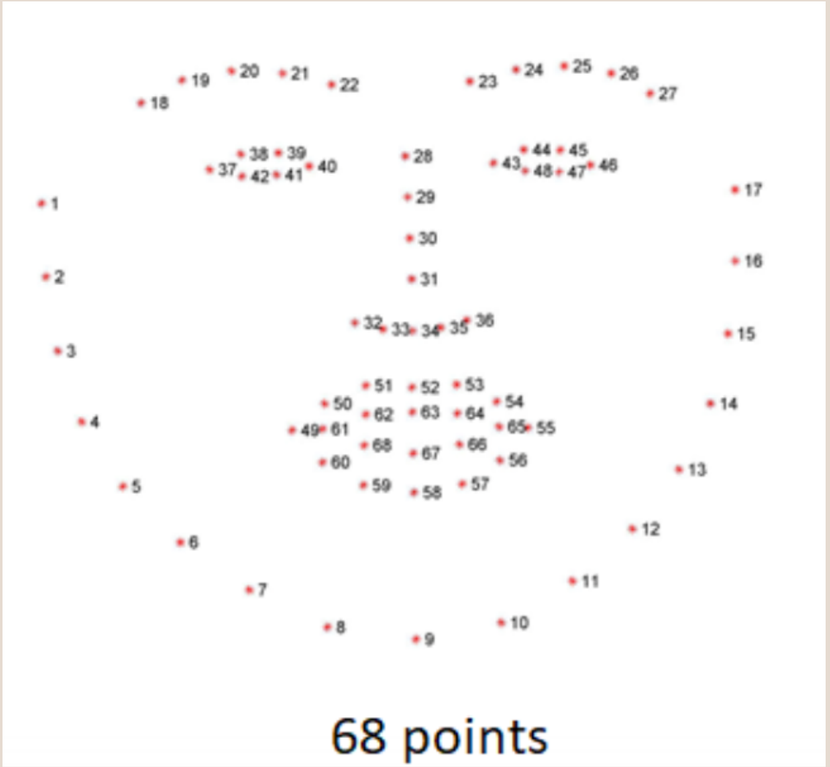


MTCNN detector - output



Baseline 2
Pupil detection using DLIB

- Using DLIB (68 points) to locate whole eye area
- considering eyes' common point: black pupil.
- catch Hough circle / the black part of eye as pupil point
- approximate iris area and detect HSV color.



Detail Implement of Pupil catching - 1

```
def pixelReader(img,startHorizontal,startVertical,height):
    # list for satisfied pixels
    blackColour = []

    # for loops for traversing pixels
    for j in range(-int(height*1.5), 0):
        for i in range(startVertical- height,startVertical+int(height*1.5)):
            # setting lower bound for pixels, termination
            blackLowerRange = [80, 50, 50]
            pixel = startHorizontal + j
            colorCI = img[int(pixel), i]
            if ((colorCI[0] <= blackLowerRange[0] and colorCI[1] <= blackLowerRange[1]
                and colorCI[2] <= blackLowerRange[2])):
                blackColour.append([int(pixel), i])
    return blackColour
```

Judge if the pixel is black and collect the black area (as pupil point).

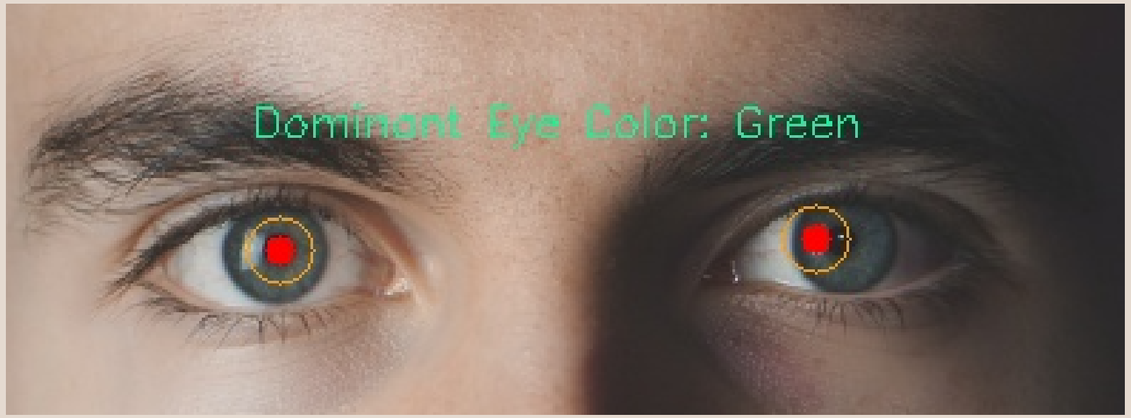
Detail Implement of Pupil catching - 2

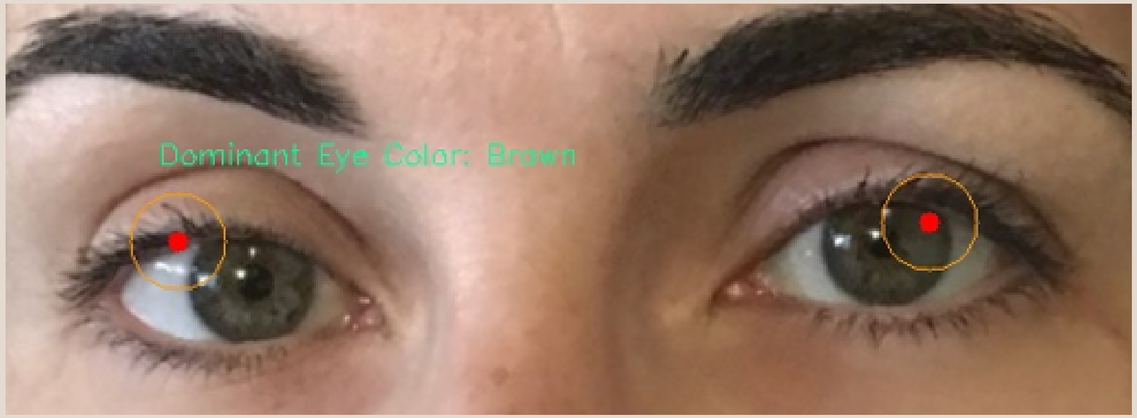
```
# after getting the eyes pixels applying cv2 method to detect circle pixels which we can do using houghcircles
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
circles = cv2.HoughCircles(gray,cv2.HOUGH_GRADIENT, dp = 1,minDist = 5,
    param1=250,param2=10,minRadius=1,maxRadius=-1)

# check if houghcircles has detect any circle or not
if circles is not None:
    circles = np.uint16(np.around(circles))
    for i in circles[0,:1]:
        pupilPoint = [int(eyeTopPointX[0])+ i[0],int(eyeBottomPointY[1]) + i[1]]
else:
    # if HoughCircles is unable to detect the circle than using eyes points to get pupil points
    a = 0
    for j,k in blackCoordinates:
        if a == int(len(blackCoordinates)/2):
            pupilPoint = [k,j]
        a += 1
return pupilPoint
```

1. Use hough circle function to detect circle.
2. Else, return the middle point of black area.

Pupil detection - output





Best model
Iris extraction

→

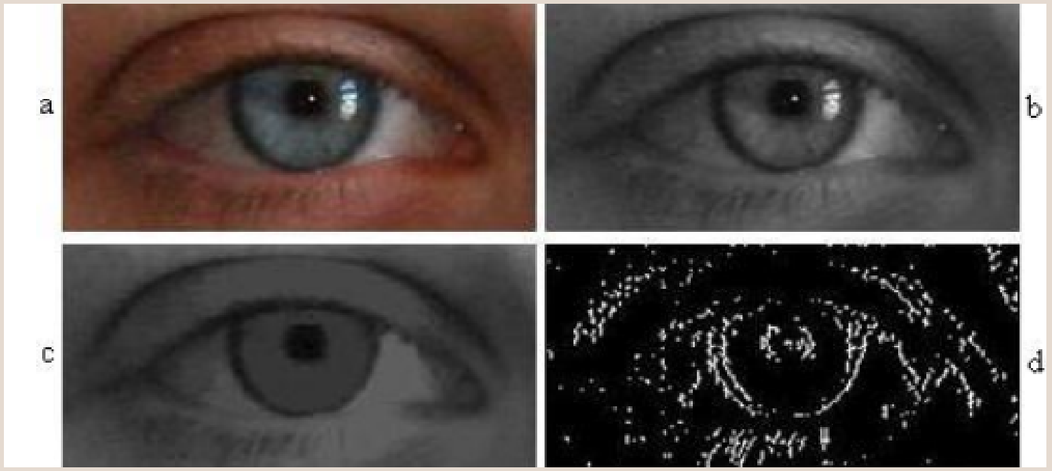
An automatic eye color detection algorithm

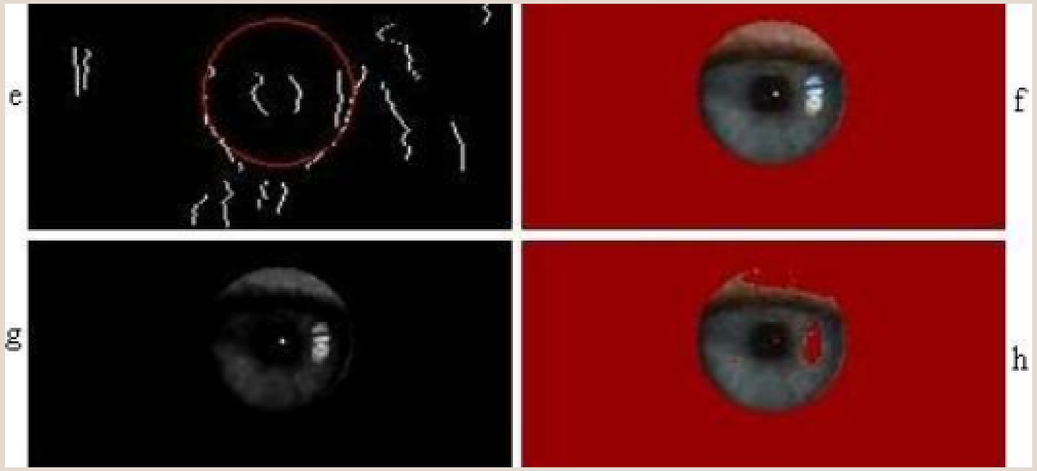
- Automatic iris extraction
- Manual region of interest (ROI) extraction (avoid pupil)
- HSV

```
hsv_g = [95.62,44.44,28.23]
hsv_bl = [205.06,40,53.72]
hsv_br = [27.18,78.05,32.15]
hsv_blck = [0.0,14.28,2.74]
distances = [hsvDist(colour,hsv_br) , hsvDist(colour ,hsv_bl), hsvDist(colour ,hsv_g ) , hsvDist(colour ,hsv_blck )]
majorColorIndex = np.argmin(distances)
```

reference: On the Reliability of Eye Color Classification as a Soft Biometrics Trait

Automatic iris extraction



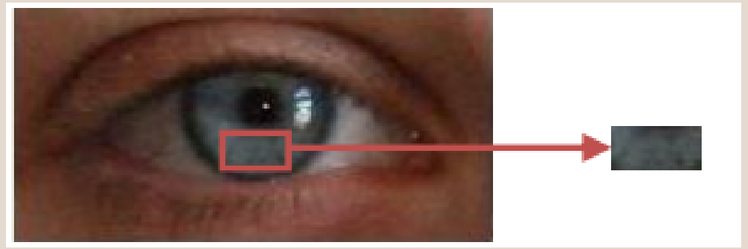


a. Original image
b. Converted to grayscale
c. After filling operation
d. Detected vertical edges

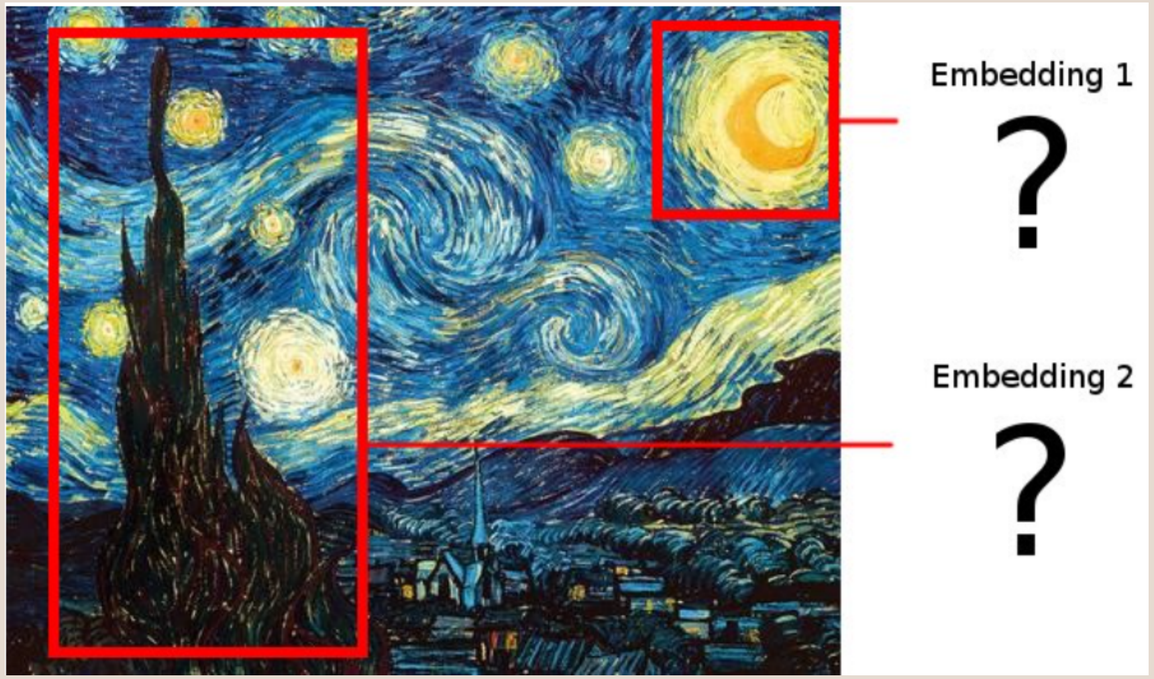
e. processing vertical edges and the detected iris circle
f. Extracted iris
g. Difference image with reflections
h. Final iris image

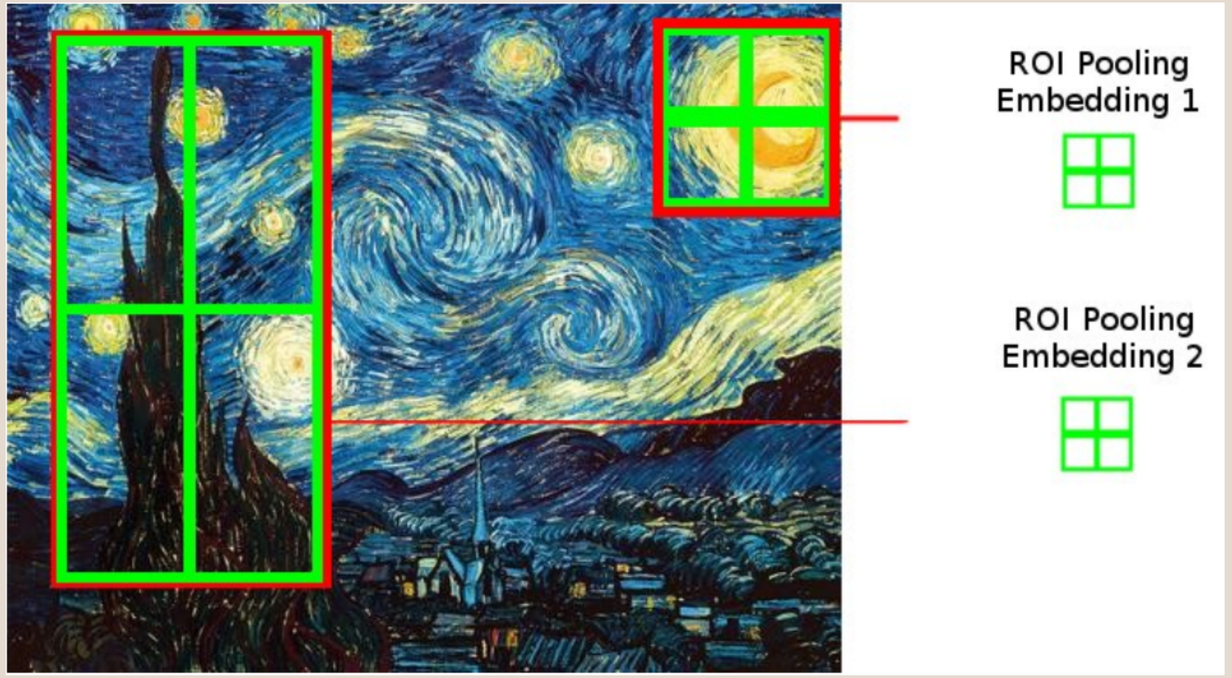
reference: On the Reliability of Eye Color Classification as a Soft Biometrics Trait

ROI extraction



→

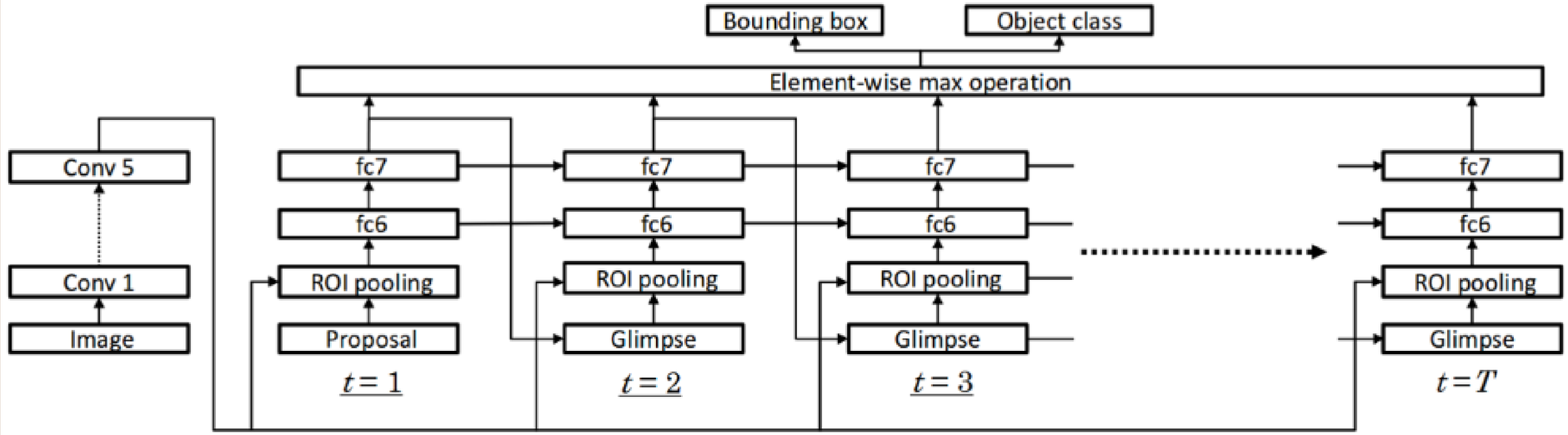




reference: https://medium.com/xplore-ai/implementing-attention-in-tensorflow-keras-using-roi-pooling-992508b6592b

ROI extraction

Fig. 1. Humans have the capability of analyzing image content for visual object detection from multiple fixation points.



Attentional Network for Visual Object Detection showcasing RoI pooling, by Hara et al

reference: https://medium.com/xplore-ai/implementing-attention-in-tensorflow-keras-using-roi-pooling-992508b6592b

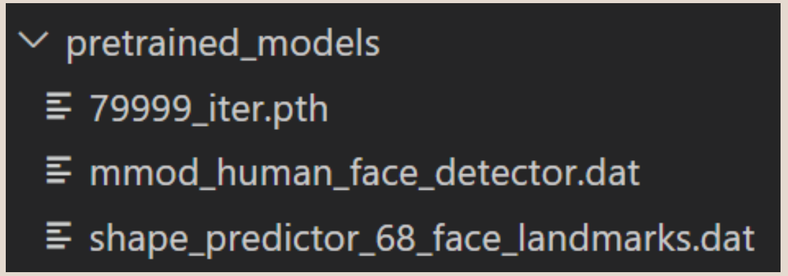
Color analysis

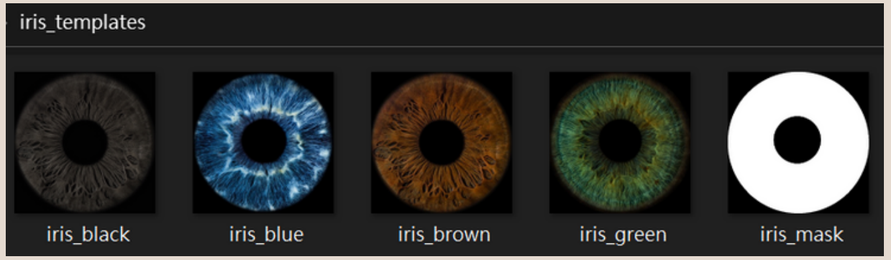
1. If black pixels >=70% -> Black
2. If black is the majority, <= 50% -> second strongest color
3. If black is the majority, <= 50%, and brown == green -> Green

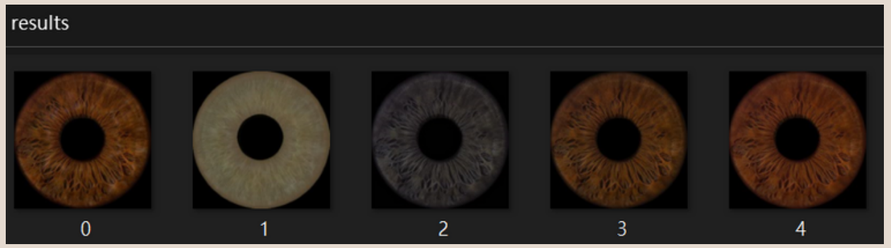
Det\Real	Black	Brown	Green	Blue
Black	90.9%	5.26%		6.25%
Brown		89.47%	14.28%	
Green	4.54%		78.57%	18.75%
Blue	4.54%	5.26%	7.14%	75%

“The UBIRIS.v2: A Database of VisibleWavelength Iris Images Captured On-The-Move and At-ADistance,”

Iris extraction related work →







reference: https://github.com/vanquish630/ExtractIris

Testing dataset

- US criminal photos, eye color are labeled. https://www.kaggle.com/datasets/davidjfisher/illinois-doc-labeled-faces-dataset

- Statistics for eyes:
  - 51714 Brown -> Dark
  - 7808 Blue
  - 4259 Hazel -> Brown
  - 2469 Green
  - 1382 Black -> Dark

- Accuracy = 0.4642857142857143 -> 0.7193877551020408

RESULT & ERROR ANALYSIS

	BASELINE 1	BASELINE 2	MODEL
ACCURACY	0.436681222	0.3755458515	0.7193877551
TESTING DATA	430	430	430

- BL 2 is worse than BL 1, pupil not will detected
- Dataset divergence.
- Influential factors: Illumination, glasses, L/R eye.
- GMM for classification RGB -> HSV

REFERENCE

BASELINE 1

https://github.com/ghimiredhikura/Eye-Color-Detection

BASELIN2

https://github.com/weblineindia/AIML-Pupil-Detection

MODEL

https://github.com/vanquish630/ExtractIris
On the reliability of eye color as a soft biometric trait -- Antitza Dantcheva; Nesli Erdogmus; Jean-Luc Dugelay
https://medium.com/xplore-ai/implementing-attention-in-tensorflow-keras-using-roi-pooling-992508b6592b

DEMO

main.py M

Generate Simulate

LAIAIAI > main.py > ...

```python
from race_analysis.model.pred import racemain
from eye_color.model.test import eye

imagepath = input("\n\n\nEnter your image path: ")
gender = gendermain(imagepath)
faceshape = shapemain(imagepath)
race = racemain(imagepath)
eyecolor = eye(imagepath)
print('\n\n\n-------HERE IS THE OUTPUT-------\n')
print('     gender:    ',gender)
print('     shape:     ',faceshape)
print('     race:      ',race)
print('     eyecolor:  ',eyecolor)
print('\n--------------------------------\n\n\n')
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

TERMINAL

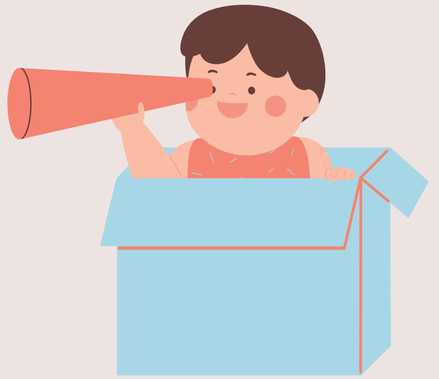
POWERSHELL

```
Windows PowerShell
Copyright (C) Microsoft Corporation. 著作權所有，並保留一切權利。

請嘗試新的跨平台 PowerShell https://aka.ms/pscore6

PS C:\Users\user\LAIAIAI>
```

Future Work



SKIN COLOR

- add more race label
- Use a better face detection model (faster rcnn)

EYE COLOR

- use RGB color space
- decrease illumination variation

GENDER

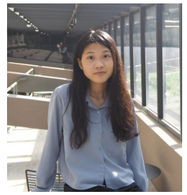
- sexual and gender diversity

FACE SHAPE

- try to compare KNN and CNN
- increase quantity of labels
- extend training to with/without glasses

1. photo to video
2. website, database
3. keep track of user and change their choice cause all some features are subjective
4. add more features

Contribution



109550031 李旻融

face shape recognition

109550027 紀竺均

eyes color recognition

109550005 張可晴

race analysis

109550097 周彤瑾

gender classification

THANK YOU