THE APPLICATION OF FACE RECOGNITION TECHNOLOGY IN E-COMMERCE

Kelly-(Yunru Qu)
Ares-(Kaichen Jia)
Johnson-(Junxiao Li)

CONTENTS

1

INTRODUCTION

01-1. Background

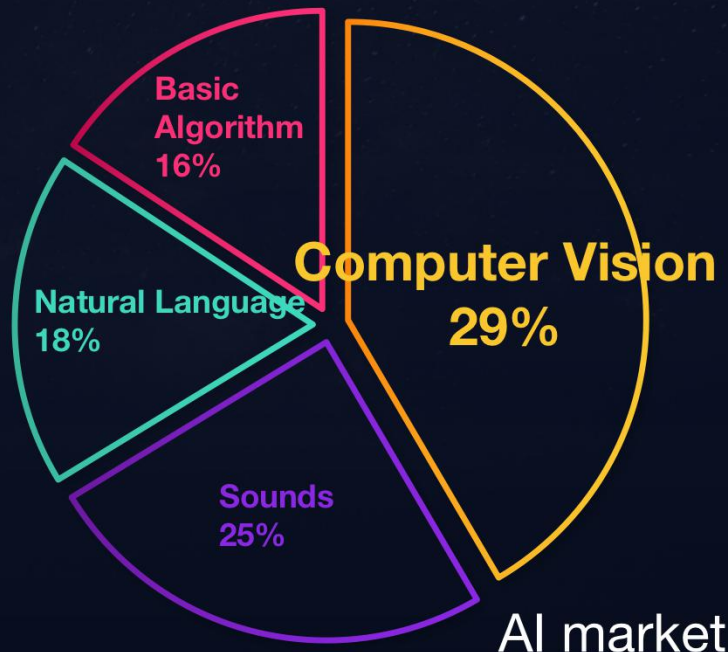
01-2. Opportunities

01-1. Background

Computer vision technology enables the computer to recognize and process images, and then create appropriate output from the analysis.

Computer Vision

Computer vision accounts for 29% in the field of artificial intelligence application, which has become an important support for the development of e-commerce industry.



AI market structure

01-2. Opportunities

At present, computer vision has been applied to many aspects of E-commerce.

01
Brush face payment

Using face recognition for ne twork payment, this technolo gy has been widely used in mobile and financial applicati ons

02
Beauty shopping guide & Try on online

Using image processing technology for beauty shopping guide and try on to im prove user experience

Future
Optimization of product push using user face an d emotion recognition

03
Image information management

Using computer vision to identify and manage the platform pictures.

02

SOLUTION

02-1. Scenario Simulation

02-2. Solution Path

02-1. Scenario Simulation

Recognize who you are

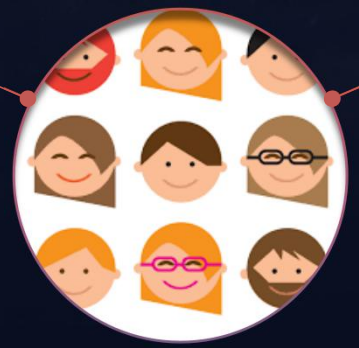
Identify User

Recommend Service

In-app advertising such as products and video content

Happy, Disgust, Neutral...

Detect User Emotion



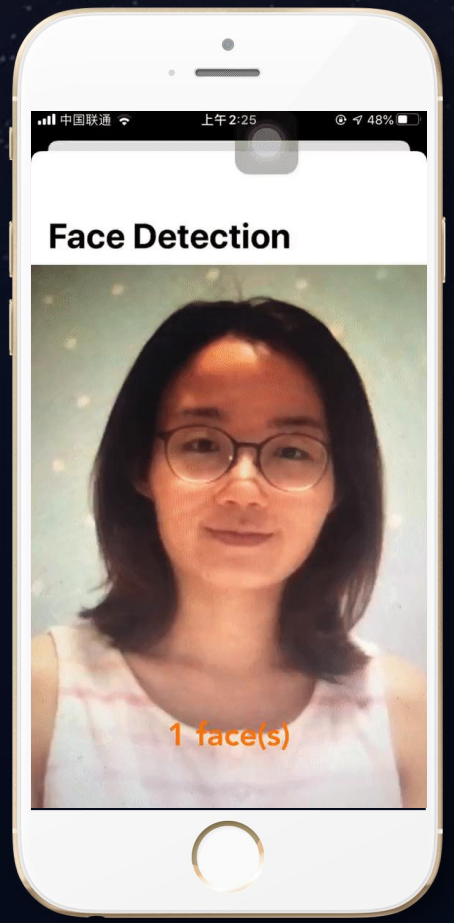
Enrich Personas

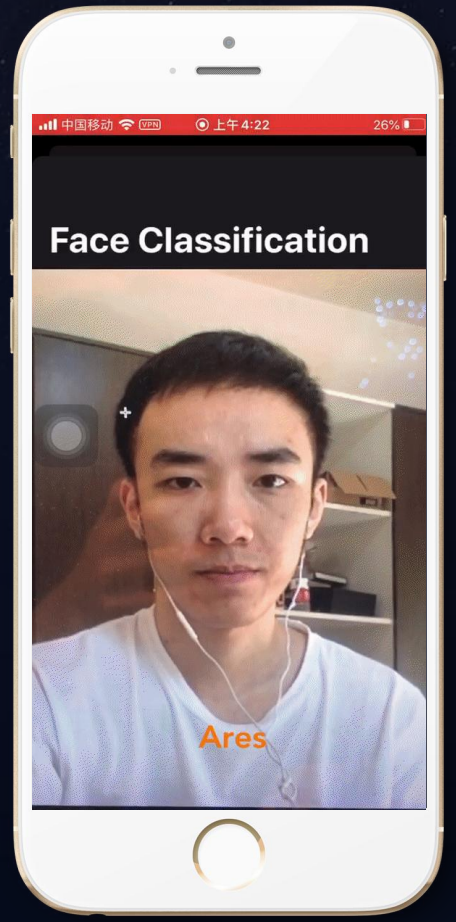
Provide more valuable data about a particular user

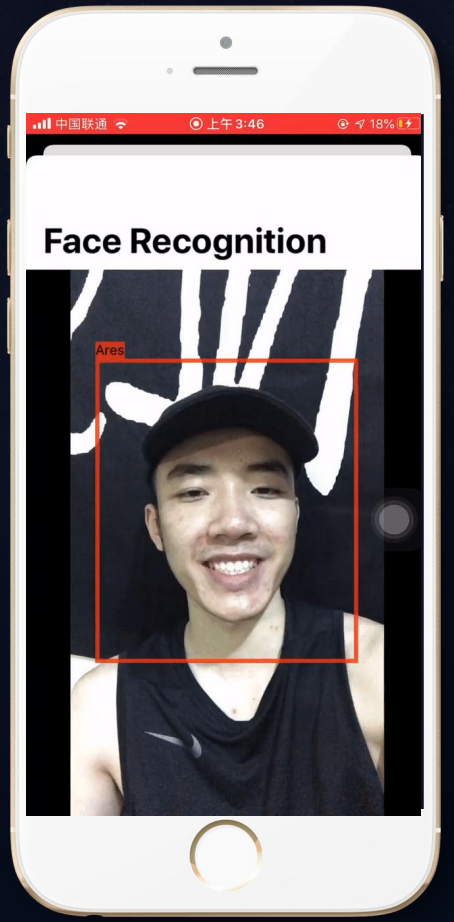
More accurate advertisement to target consumers

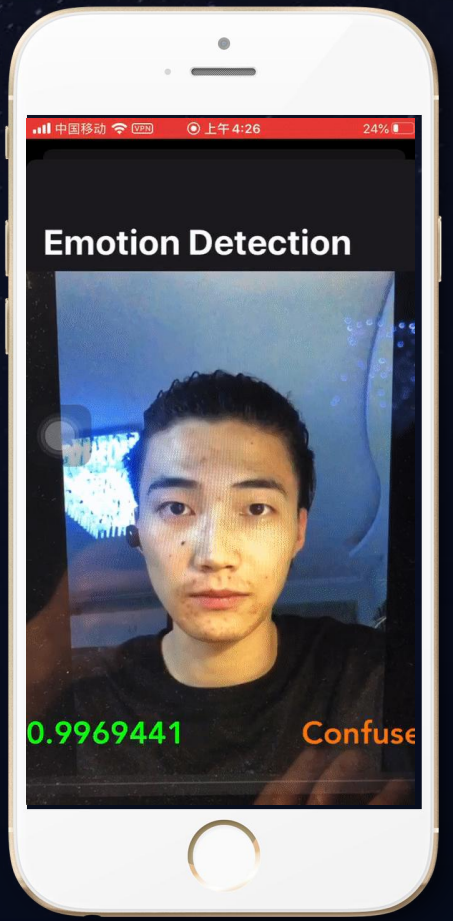
Optimize Recommendations

02-1. Solution Path









1 Face Detection

2 Face Classfication

3 Face Recognition

4 Emotion Detection

03

REALIZATION

03-1. Face Detection

03-2. Face Classification

03-3. Face Recognition

03-4. Emotion Detection

03-5. Mask Detection (Bonus)

03-6. Deploy into iOS Device

Data Source

20 images/ individual

Ares

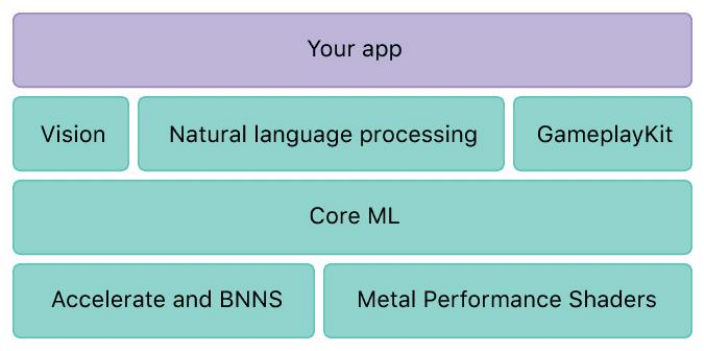
Kelly

ljx

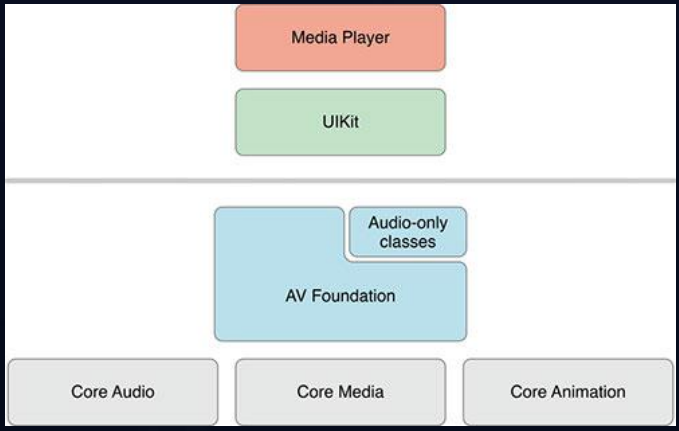
03-1. Face Detection

First, if we want to recognize faces, we need to detect faces.

Vision

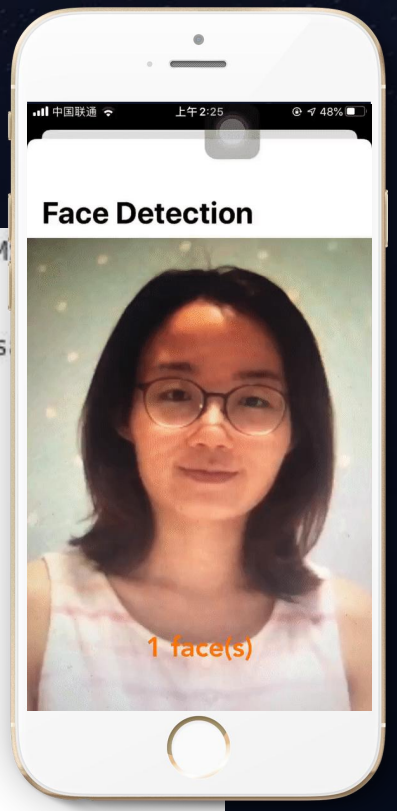


AVFoundation

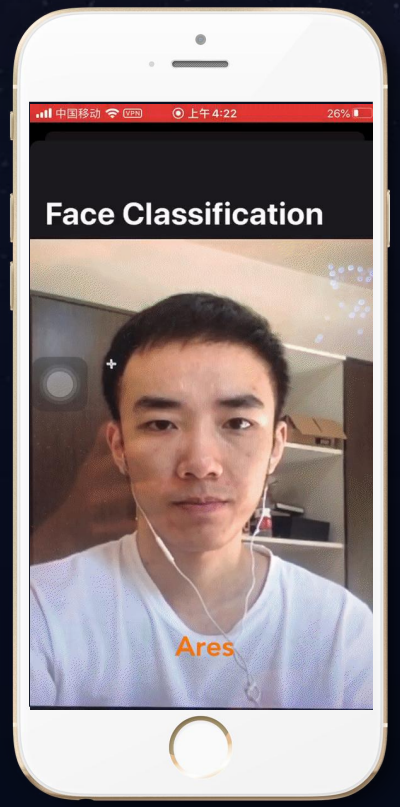


VNDetectFaceRectanglesRequest( )

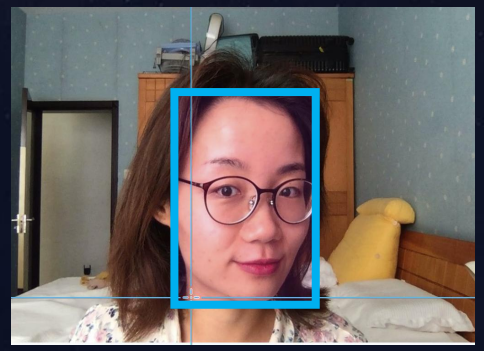
```
func captureOutput(_ output: AVCaptureOutput, didOutput sampleBuffer: CM

    guard let pixelBuffer: CVPixelBuffer = CMSampleBufferGetImageBuffer(s
    let request = VNDetectFaceRectanglesRequest { (req, err) in

        if let err = err {
            print("Failed to detect faces:", err)
            return
        }
        DispatchQueue.main.async {
            if let results = req.results {
                self.numberOfFaces.text = "\(results.count) face(s)"
            }
        }
    }
```



03-2. Face Classification

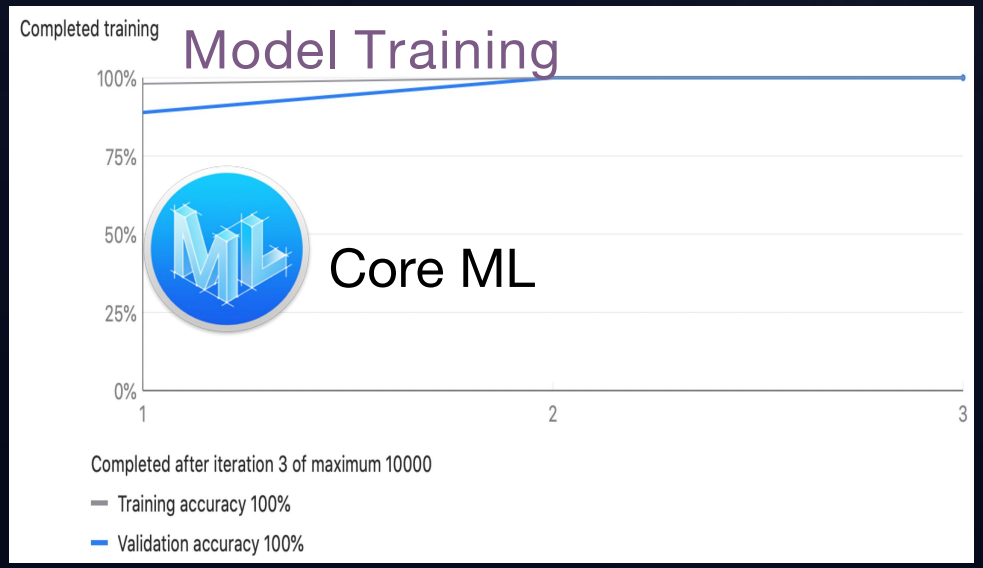


Data Annotation



IBM **Cloud Annotations**

Export
.JSON file



Train

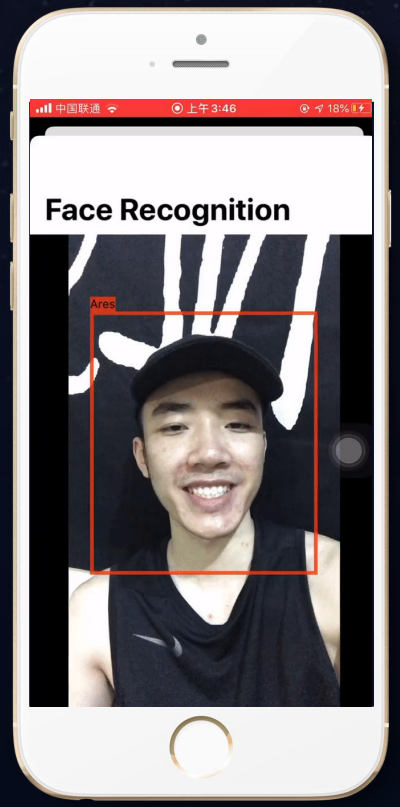
Class	Item count	Percision	Recall
Ares	18	100%	100%
Kelly	18	100%	100%
ljx	18	100%	100%

Validation

Class	Item count	Percision	Recall
Ares	3	100%	100%
Kelly	4	100%	100%
ljx	2	100%	100%

03-3. Face Recognition

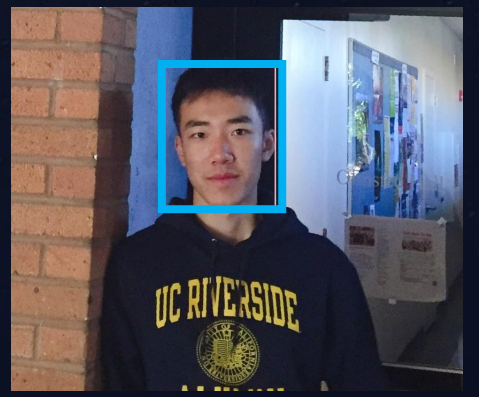
Code: https://colab.research.google.com/drive/1l2dpxkhiBooIFXV_QV0T-P1iylW1oewC?usp=sharing



Data Annotation SFrame

- ❑ Disk backed, tabular data structure
- ❑ Works with text, image, and json

```
[{
    "label"       : "Ares",
    "coordinates": {
    "xMin"        : "324",
    "yMin"        : "274",
    "xMax"        : "563",
    "yMax"        : "581",
    }
},
```




```
// Load annotations & images
annotations =  turicreate.SFrame("annotations.csv")
images = turicreate.load_images("training_images")
```

Model Training Turi Create

- ❑ Python Library for creating Core ML models
- ❑ Task focus APIs
- ❑ Supports various data input
- ❑ Open source

```
Setting 'max_iterations' to 1000
+--------------+--------------+--------------+
| Iteration    | Loss         | Elapsed Time |
+--------------+--------------+--------------+
| 1000         | 0.750628     | 9h 2m        |
+--------------+--------------+--------------+
```

```
// Create a model
model =  turicreate.object_detector.create(train, "annotations")
```

03-3. Face Recognition

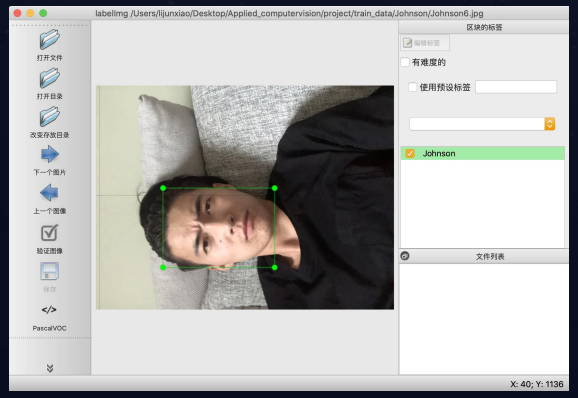
Using tensorflow object detection API to build its own target detection model

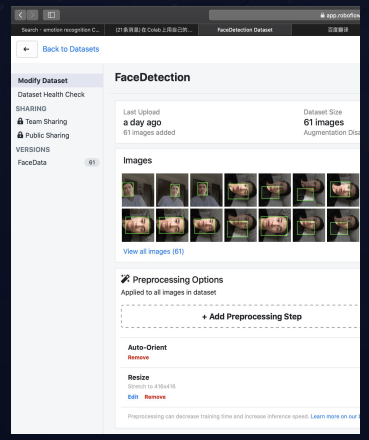
Data Annotation

LabelImg | **Roboflow**

First, label all pictures with LabelImg, and generate .xml file

Then we use the dataset type conversion tool Roboflow





Next, Adding the .record link to colab code.

```
[ ] # UPDATE THIS LINK - get our data from Roboflow
    # !curl -L https://app.roboflow.ai/ds/REPLACE-THIS-LINK > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
    !curl -L "https://app.roboflow.ai/ds/F6kZndfUiP?key=hzpXI6Izcu" > roboflow.zip; unzip roboflow.zip; rm roboflow.zip
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   891  100   891    0     0    909      0 --:--:-- --:--:-- --:--:--   908
100  109k  100  109k    0     0  70190      0  0:00:01  0:00:01 --:--:-- 70190
```

Training Model

Tensorflow Object Detection API

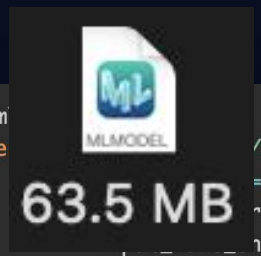
Next, we use GPU to train the data on collab.

Train the model

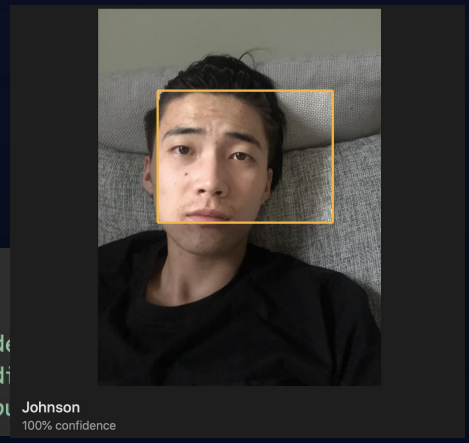
```
!python /content/models/research/object_detection/model_main.py \
    --pipeline_config_path={pipeline_fname} \
    --model_dir={model_dir} \
    --alsologtostderr \
    --num_train_steps={num_steps} \
    --num_eval_steps={num_eval_steps}
```

Convert .pd to .mlmodel

```
import tfcoremlm
tfcoreml.conve
```



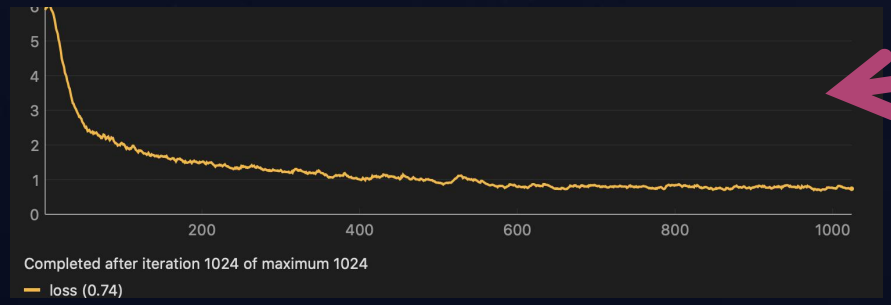

```
/model_0.pb",
="./model.mlmode
re_names=['Predi
nape_dict={'Inpu
```



03-3. Face Recognition

Comparison

```
Setting 'max_iterations' to 1000
+--------------+--------------+--------------+
| Iteration    | Loss         | Elapsed Time |
+--------------+--------------+--------------+
| 1000         | 0.750628     | 9h 2m        |
+--------------+--------------+--------------+
```



	TOOLS	DATA ANNOTATION	MODEL TRAINING
01	Turi Create √	Output Format: CSV Time Cost: 10min	Time Cost: 9h CPU Loss: 0.750
02	Create ML √	Output Format: JSON Time Cost: 10min	Time Cost: 5h CPU Loss: 0.740
03	TensorFlow Rcognition A✗	Output Format: XML-record Time Cost: 30min	Text Cost: 9h GPU Loss: 0.888

```
I0412 23:22:26.453252 139865577109376 basic_session_run_hooks
INFO:tensorflow:loss = 0.88838947, step = 500 (91.184 sec)
```

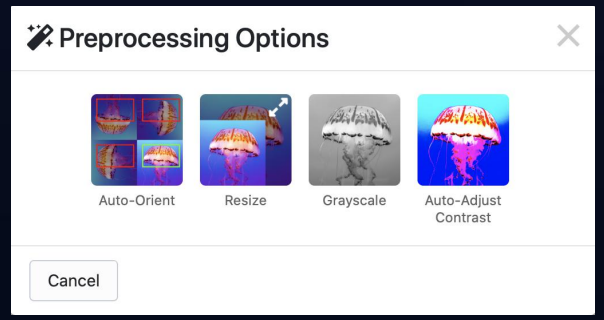
03-4. Emotion Detection

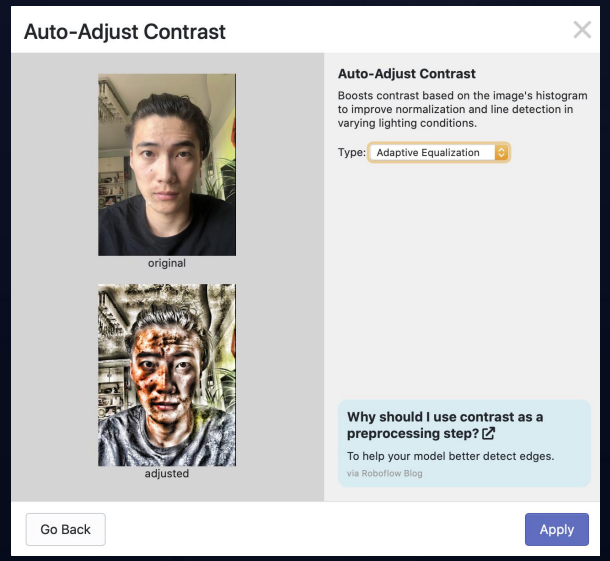
* DataBase of 6 kinds of emotions

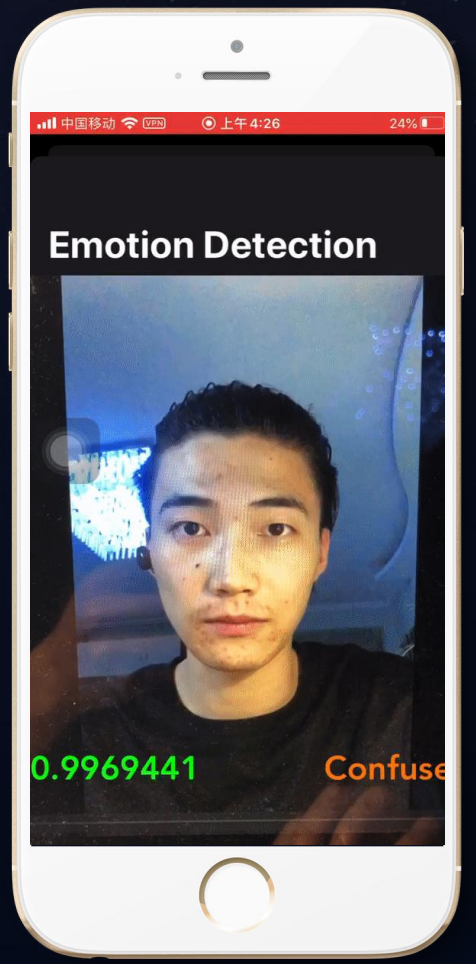
- Happy
- Angry
- Sad
- Natural
- Surprised
- Confused

* Generate more images

- Online Image Augmentation
- Enlarge DataBase







03-5. Mask Detection

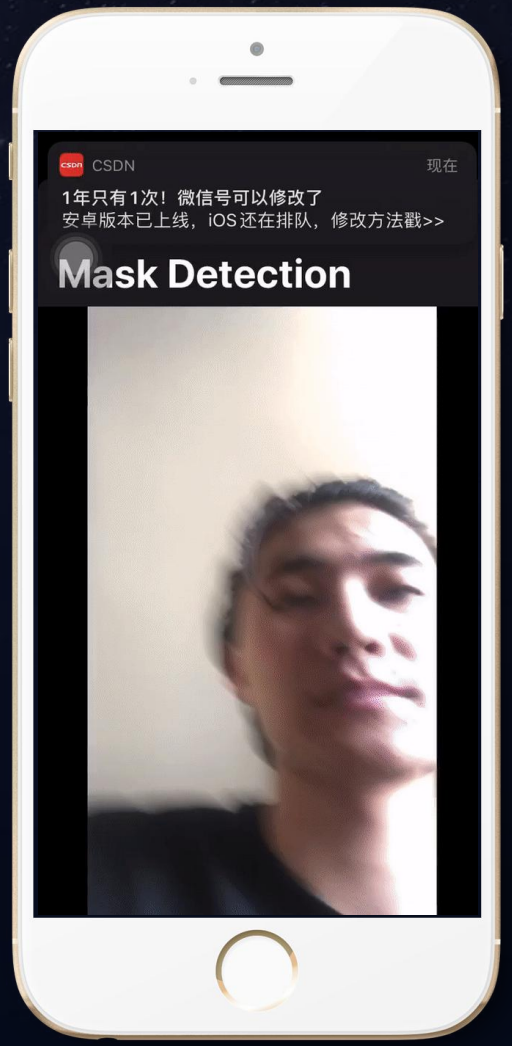
* DataBase IBM Cloud Annotations

30 images →

annotation type

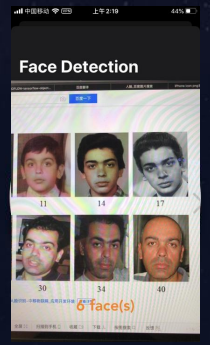
* Model

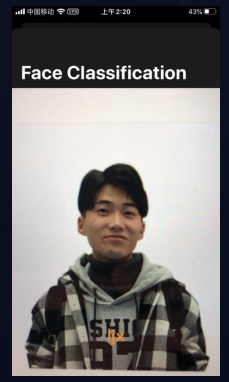
Core ML



0.3-6. Deploy into iOS Device

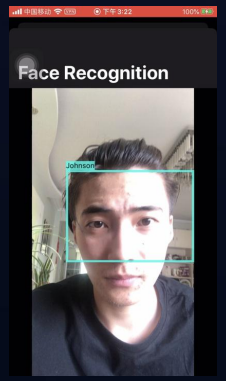
Face detection

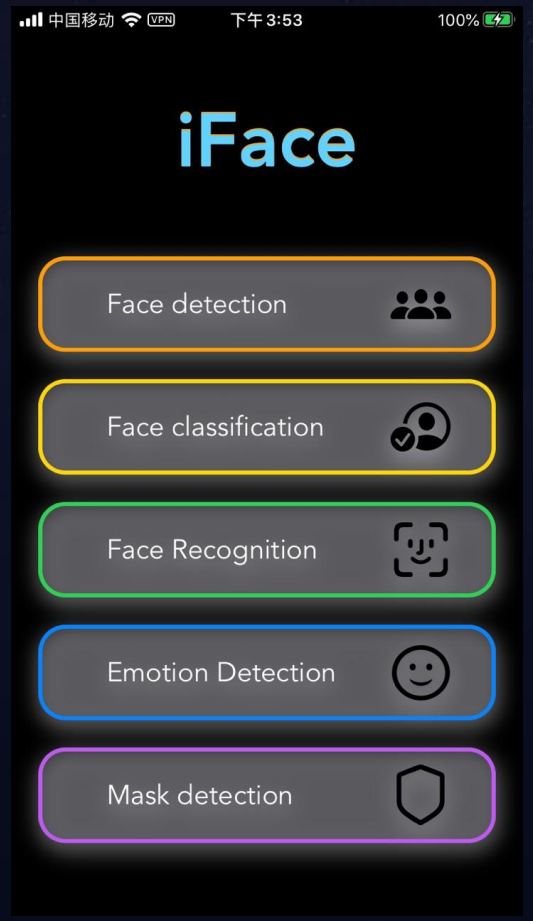


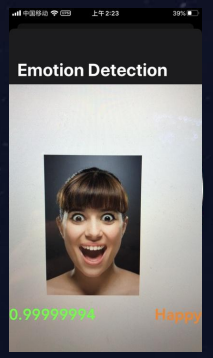


Face classification

Face Recognition

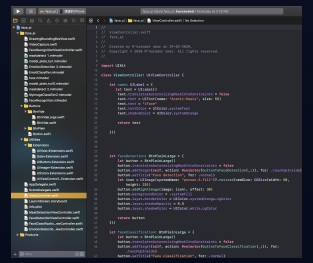






Emotion Detection

Mask Detection



Xcode Prevoew

4

SUMMARY

04-1. Project Summery

04-2. Future Outlook

0.4-1 Project Summery

what have we learned during this project?

Five functions are realized

Tools for using multiple annotation datasets

Training model with 3 different methods

Learn IOS development & deployment model

Improvement

- Expand dataset size
- Preprocessing datasets
- Increase control over model training process

0.4-2 Future Outlook

We will use the face recognition and emotion detection functions we have implemented to optimize the final simulation of the product push scene.

Data sharing between emotions and user identity

Using reinforcement learning algorithm to optimize product label

Repeat iteration to optimize product push


THANKS