The Application of Face Recognition Technology in E-Commerce

Ares - Kaicheng Jia

June 2020

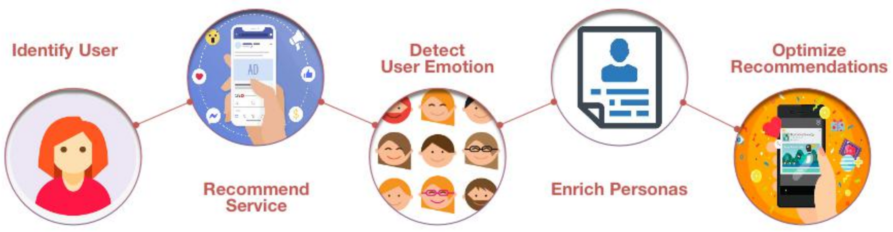
Abstract

In this project, our group provided an interesting solution for a real-world commercial problem with face recognition. By applying computer vision to identify user, and detect users' facial expressions, we collected valuable data for companies to recommend precise online services to users. We chose real human photos taken by ourselves for train set, used multiple tools such as API and AutoML for creating models, and deployed to iPhone device in the end.

Contents

1 Introduction

Nowadays, computer vision has been applied to many aspects of E-commerce, such as face scan payment and makeup try on online service. Moreover, companies like Amazon manage their products data with this technology. However, as more companies are struggling with improving the average revenue per user, which is a key performance index for evaluating business development, there's more we could do with this technology. We hope to identify a user and detect its facial expressions with face recognition technology. The data collected here can be used to enrich the user's personas for more accurate commercial recommendations.



2 Data

For each individual in my group, we provided 20 images into the train set, and we have 60 images in total. To abstract facial characteristics, we used 3 different methods to annotate images, and their outputs are in different formats. These outputs contain the information about the location of face areas and labels.

Annotation Methods:

- **IBM Cloud Annotations** – The output is in JSON format, costing roughly 10 minutes
- **SCV tool** – The output is in CSV format, costing roughly 10 minutes
- **LabelImg** – The output is in XML format, and must be transforms for later use

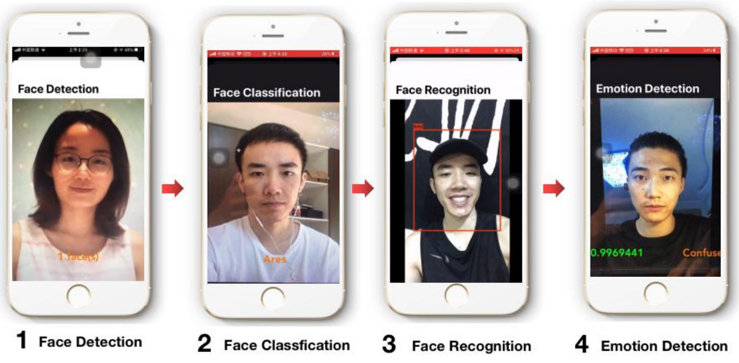
Labels:

- **Identity** – Ares, Kelly, Johnson
- **Emotion** – Happy, Angry, Sad, Natural, Confused, Surprised

3 Model & Methodology

We went through a 4 steps in the process:

We first accomplished face detection, which can detect if there are any faces in the camera. It might be easy but it's fundamental. Then we do face classification, it's able to classify different people's faces. It prints out the name of the person at the bottom. Moreover, we finished face recognition, it reveals people's names with a real-time bounding box. Finally, we realized emotion detection, which helps to analyze facial expression.



- **Face Detection**

We used the **Vision** framework provided by **Xcode** and **AVFoundation** framework to detect and count the number of faces. Vision is a very new and easy-to-use framework that provides solutions to computer vision challenges through a consistent interface. The VNDetectFaceRectangleRequest() method returns an array of bounding boxes for the detected faces. In order to get the face count, we only need to count the number of elements in the array and then update the label accordingly. With AVFoundation, we are able to build the camera into our apps.

(See detailed code at: "FaceDetectionViewControllor.swift"

- **Face Classification**

This step we used **Create ML**. It is a new framework designed to help easily build machine learning models using Swift and Xcode. Here we used IBM Cloud Annotation to process images, and after 1000 iterations, the model showed 100% accuracy both in training and in validation set when we chose type classification to run this model. It's pretty easy to realize under its instructions.

(See model at: "MyImageClassifier2.mlmodel")

- **Face Recognition**

This step we used 3 different ways to achieve this function. **Turi Create** is a python library for creating Core ML models. It has many task foucs APIs and supports various data input.

By transforming the input csv data into SFrame, which is a tabular data structure, we were able to create a model. Besides, we tried using TensorFlow Object Detection API and Create ML for comparison.
(See detailed code at:"Face_Recognition_Turi.ipynb"

- **Emotion Detection**

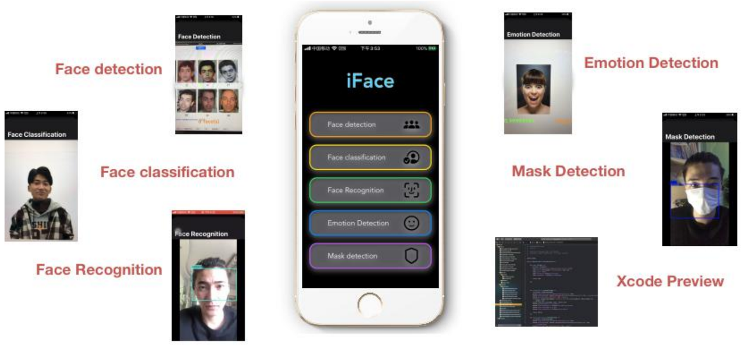
Although we still used Create ML to train the model, we preprocessed the data in an advanced way. Due to the small amount of data, we used the preprocessing options in **Roboflow** to do auto adjust contrast operation with data. It not only increased the size of database, but also increased the diversity.
(See model at: "EmotionDetection 2.mlmodel")

- **Mask Detection (Bonus)**

When we are familiar with the method of deploying face recognition model in IOS, we had implemented an additional function. We used CreateML to train mask datasets and deployed it on iOS to detect whether to wear masks. In today's environment, it is particularly important to realize the function of mask detection.
(See model at: "maskdetect 2.mlmodel")

4 Deployment

On the iOS interface we deployed, an entry will be displayed, including buttons of five functions. If we want to implement a function, just click button, and a view will be generated to perform corresponding operations.
(See detailed code at Xcode project)



5 Performance

The face detection and classification tasks both have nearly 100% accuracy. The emotion detection task has nearly 60% accuracy due to the limited database. We also compared the three methods used in face recognition task.

- **Turi Create** – Time Cost: 9h CPU, Loss: 0.750
- **TensorFlow API** – Time Cost: 5h CPU, Loss: 0.740
- **CreateML** – Time Cost: 9h GPU, Loss: 0.888

Therefore, Turi Create and Create ML have better performances than TensorFlow API here.

6 Conclusion

By analyzing the development of E-commerce and the application of computer vision in the industry, we simulated the scene and realized the function of optimizing product push by using face recognition and emotion recognition. Through the above research, we have gained a lot. We learned to use a variety of tools when working with datasets, including LabelImg and Roboflow. In the training set, we also considered and used different methods, such as CreateML, Turi Create and Tensorflow Object Detection API. We also compared the performances among them. In the process of deploying to iOS devices, since we were not very familiar with swift, we encountered many difficulties at the beginning, but fortunately, we finally deployed the model on the iPhone by finding materials and group discussion.

In the later research, we will use the face recognition and emotion detection functions we have implemented to optimize the final simulation of the product push scene. Firstly, a product picture is on the screen, and output interface of face recognition are extracted in real time, and the label data of emotion detection is extracted to transfer the data between different models. Then bind the user's attitude to the current product and the user's identity, and use reinforcement learning or other optimization methods to make decisions on the product label that will be passed to the user next time, so as to make the user satisfied with the pushed product as much as possible, realize the optimization repeatedly, and finally optimize the product push.