Final Written Reporting

Jennifer Yihan Ruan

1.0 Introduction

In the final project, my aim is using the Xcode to implement Core ML Models to an edge based device, and my outcome for the project are two apps that can classify the dominant object in a photo and real time video. Moreover, instead of focusing on how to generate models by using different platforms, I decided to put more effort on how to create an app in the iOS system with computer vision from begining.

2.0 Methodology

For the first application of Core ML: object classification through a picture, I simply download the example provided by the apple company. This example is also the inspiration of my project. Since I am surprised by how convenient and efficient it is to implement a computer vision app in our smart-phone using an successed model without writing any code, I want to figure out the theory behind creating apps with a training model. With clearly knowing those steps, I will be able to easily implement various computer vision models on my smartphone as an app that can be used in real life.

My second application is to create an app that can achieve real-time object classification, and I have divided my work into three parts: starting up setting camera, implementing Core ML model, designing the storyboard. In order to let our app get the access to the back camera of an iPhone, we first need to go to info.plist to add the privacy for camera usage description. Moreover, we need to add a preview Layer to tell the application to give us the output. In addition, we are also required to specify a frame for the preview Layer. For the second part, we can simply download and drag any models we like in the Xcode file. Then, using *VNCoreMLRequest* and *finishedReq.results* can be contributed to get the result to the model. In order to display the results on the mobile screen, we need to design a storyboard. Creating labels and connecting it into the controller swift can help us to achieve that task. Furthermore, two Core

ML models have been involved in my project: squeezeNet model and ResNet 50, and they are created for the same use: object classification. Therefore, I also decided to compare datas for those two models, and the results turns out that ResNet 50 actually gives us more accurate answers. One reason I believe that might cause this phenomenon is that the squeeze model is only 4.7 megabytes. ResNet 50, on the other hand, is around 102 megabytes, which is much greater.

3.0 Further applications

Some future prospects have also been included during my presentation, and I believe methods for my project can be used to solve some real life problems. For example, implementing Core ML models to an app is able to develop some wearable technology for people with visual impairment to make their lives more convenient, such as letting the app make some noise when it detects something dangerous. Moreover, this project can also be implemented in online searching and online shopping. To illustrate, you can easily get items you want by taking a picture without knowing its name.

4.0 Conclusion

To sum up, I think this project is different from things I have done before, because instead of understanding methods and algorithms, it provides a opportunity for me to actually create something that can be used in our daily life.

5.0 Reference:

Core ML Models(2020) Retrieved May 9, 2020 from: https://developer.apple.com/machine-learning/models/

Elworthy, E. (24 August, 2019) Object Detection CoreML iOS App Tutorial - Swift & Xcode - iPhones & iPads running iOS 11 & Higher. Retrieved May 9, 2020 from: https://www.youtube.com/watch?v=SkgHz8nw5V8

Lets Build That App (14 July, 2017) *CoreML: Real Time Camera Object Detection with Machine Learning - Swift 4.* Retrieved May 9, 2020 from: https://www.youtube.com/watch?v=p6GA8ODlnX0