CSCI B657 Computer Vision
Hand gestures recognition on iOS mobile application

Haifeng Lin, Xinquan Wu

April 30, 2018

1 Introduction

Aiming for using hand gestures to control iOS Apps, we develop an App of hand gestures recognition on iOS. We use the front camera of iPhone to capture the real-time images and then use our own model trained by Deep Learning to detect and recognize hand gestures. Then we can use these prediction results to control the music player App.

2 Background and related work

Virtual Reality (VR) and Augmentation Reality (AR) technologies are very popular concepts in these days. They help people to feel like in a virtual environment or interact with some virtual objects in the real world. And more excitingly now, we can easily use these technologies just through our smartphones and cardboards.
However, it's not easy for us to make interactions with the mobile devices, especially when they are put into the cardboard. And also, it's inconvenient to use additional controlling devices.
Computer vision is a way to help us build an interaction method. And by using the Apple Core ML and Vision kit, we can easily deploy them into iOS mobile devices.
Core ML is a development toolkit to help developers to implement Machine Learning techniques more easily. Developers can incorporate the trained machine learning models into the iOS development environment by using Core ML.

3 Methods

3.1 Tools in iOS Development

Apple gives developers a lot of tools for development. In our project we use AVFoundation, Core ML and ARKit.

The AVFoundation framework[2] combines four major technology areas that together encompass a wide range of tasks for capturing, processing, synthesizing, controlling, importing and exporting audiovisual media on Apple platforms.

Core ML[4] is the foundation of domain-specific frameworks and functionality. Core ML supports Vision for image analysis, Foundation for natural language processing (for example, the NSLinguisticTagger class), and GameplayKit for evaluating learned decision trees. Core ML itself builds on top of low-level primitives like Accelerate and BNNS, as well as Metal Performance Shaders.

Augmented reality (AR) describes user experiences that add 2D or 3D elements to the live view from a device's camera in a way that makes those elements appear to imitate the real world. ARKit [1] combines device motion tracking, camera scene capture, advanced scene processing, and display conveniences to simplify the task of building an AR experience.

In our project, we need to get the images from the front camera and use them to make predictions and other operations. In iOS development, we can use AVFoundation framework to get the data from the camera. And in the music playing app, we use the audio functions from AVFoundation framework, which can help us reduce a lot of work. After training the model, we can use Core ML to import the model into our app. And we can use the model functions to make predictions. We also use ARKit to help us to get statistic information like fps and time information and show them on the screen.[3]

3.2 Model Training

Custom Vision[5] Service is a tool for building custom image classifiers. It makes it easy and fast to build, deploy, and improve an image classifier. Custom Vision Service provides a REST API and a web interface to upload your images and then train the data. So we use Custom Vision Service to train our own model.
First, we need to prepare our training data. With any dataset we create, we should be aware of its inherent biases and robustness.[6] So we take hundreds of photos of our hand gestures with different backgrounds and different light environments. We designed six gestures for training, which are "fist-hand", "five-hand", "checkmark-hand" "circle-hand", "index-hand" and "thumb-hand". And each of these gestures has at least 30 pictures. The exapmle is as Figure 1 and Figure 2.
At frist, we label our images separately for left and right hands. And the "recall" of the training result is pretty low. This is because in Custom Vision Service, it flit the images horizontally and vertically to generate more training data. However, we want to train them differently between left and right hands. So this confuses the machine that gives us really low "recall" result. To solve the problem, we merge the same gestures with both hands into same labels. Finally, the training performance give us a jump of recall from 8% to 72%.

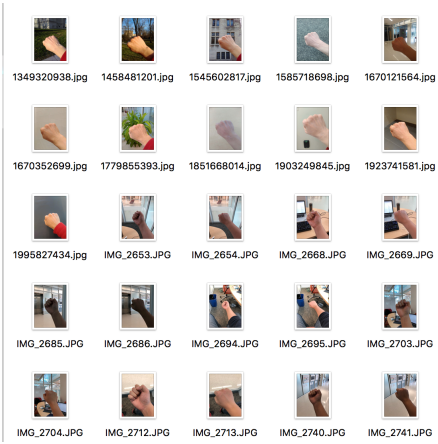

Figure 1: Fist-Hand Left

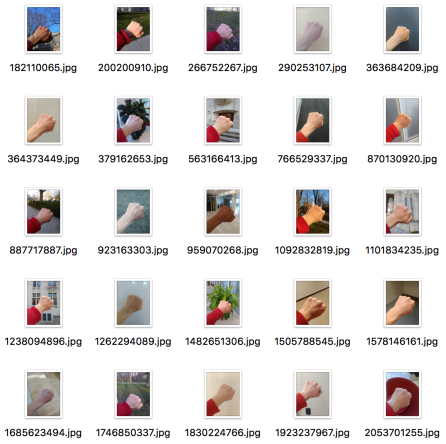

Figure 2: Fist-Hand Right

3.3 Core ML on iOS

With the Core ML framework, we can use our trained machine learning model to classify input data from the front-end camera. The Vision framework works with Core ML to apply classification models to images, and to preprocess those images to make machine learning tasks easier and more reliable. The pipeline is as follow:

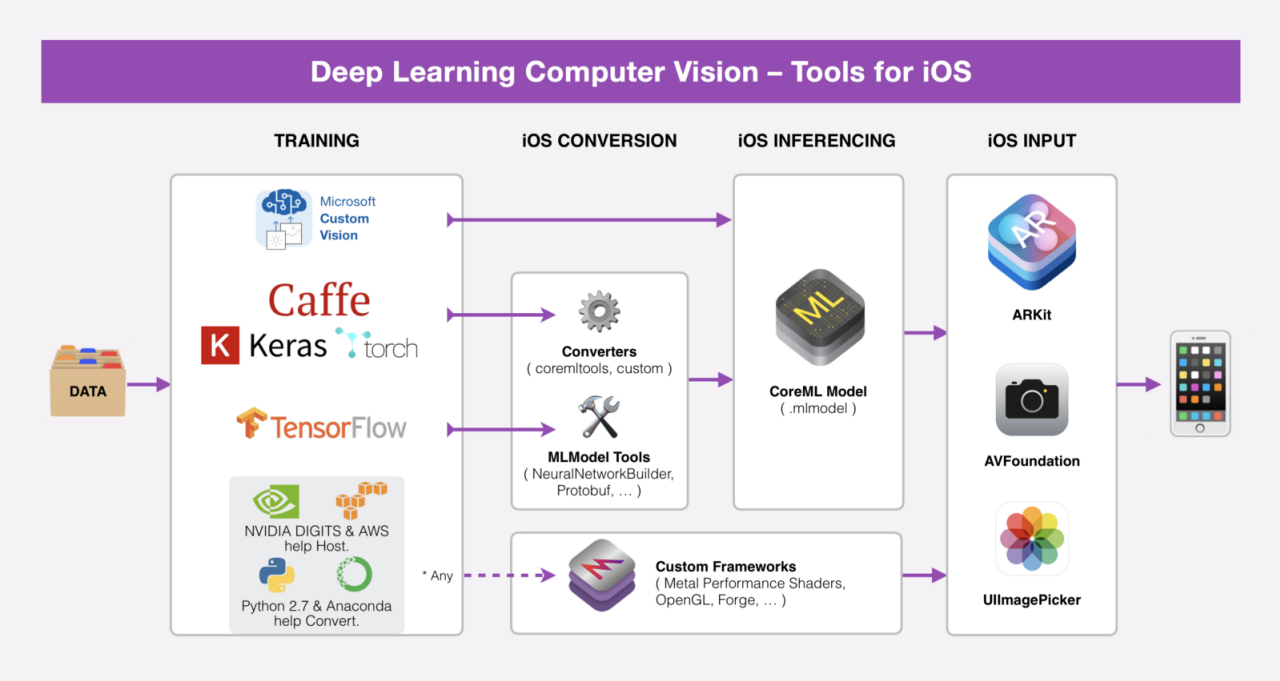

Figure 3: Machine Learning Pipeline for iOS

Core ML automatically generates a Swift class that provides easy access to our ML model; in

our project, Core ML automatically generates the gesture01 class from the gesture01 model. To set up a Vision request using the model, it creates an instance of that class and uses its model property to create a VNCoreMLRequest object. Then it uses completion handler of the request object to specify a method to receive results from the model after we run the request.

An ML model processes input images in a fixed aspect ratio, but input images may have arbitrary aspect ratios, so Vision must scale or crop the image to fit. For best results, set the request's imageCropAndScaleOption property to match the image layout that the model was trained with. For the available classification models, the centerCrop option is appropriate unless noted otherwise.

It creates a VNImageRequestHandler object with the image to be processed, and passes the requests to that object's perform(_:) method. This method runs synchronously — uses a background queue so that the main queue isn't blocked while your requests are executing.

Most models are trained on images that are already oriented correctly for display. To ensure proper handling of input images with arbitrary orientations, we need to pass the image's orientation to the image request handler. (This sample app adds an initializer, init(_:), to the CGImagePropertyOrientation type for converting from UIImageOrientation orientation values.)

The completion handler of the Vision request indicates whether the request succeeded or resulted in an error. If it succeeded, its results property contains a VNClassificationObservation object describing possible classification identified by the ML model.[6]

3.4 Application Control

When we get the prediction results, we can use these results to control other application. The hand gestures control music player. We get three hand gestures results which are "fist-hand", "five-hand" and "checkmark-hand". We use "fist-hand" to start playing the music, use "five-hand" to pause the music, and use "checkmark-hand" to cancel the music playing. When the music is playing, we can see the time of how long is the song playing, and when we pause the music, the time will also stop. When we cancel the playing, the time will be clear to zero which means if we play it again, the music will start from the beginning. So, follow these steps, we can use our hand to control the app without touching the screen.

4 Results

As a result, we can control the app to play, pause and cancel the music playing. However, it works not very precisely because we don't have enough training data (only 30 40 images for each label). Most time it works well especially for the "fist-hand" and "five-hand", because

these gestures are very unique from each other.

We also trained the model to recognize the “circle-hand”, “index-hand” and “thumb-hand”, but these three works not so well because they don’t have enough differences in such small training data. So, we drop these three results.

The hand gesture recognition model and it’s preformance is as follow:

Figure 4: Fist-Hand

Figure 5: Five-Hand

Figure 6: Checkmark-Hand

The model performance is as follow:

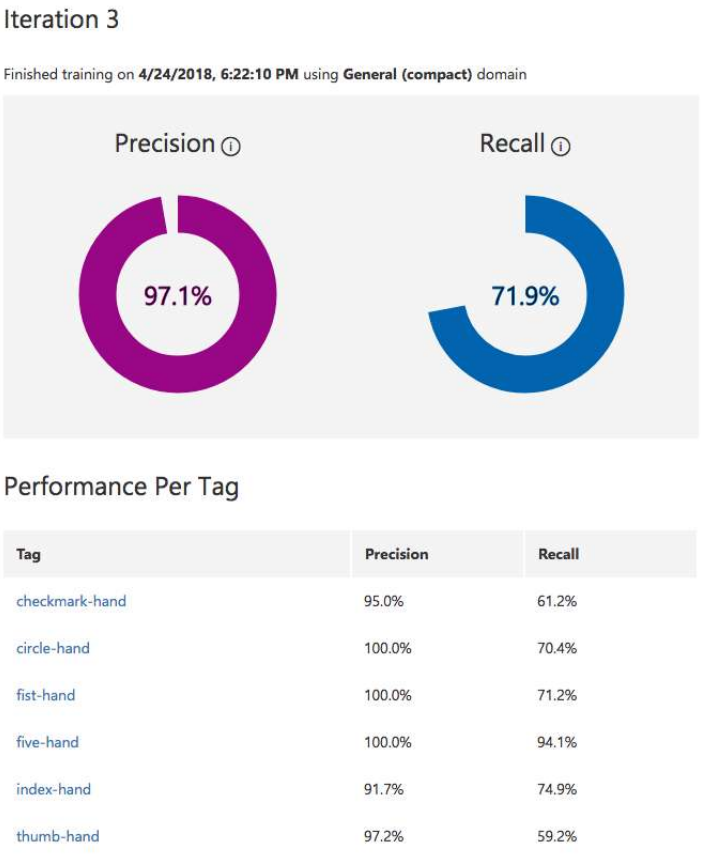

Figure 7: Model Performance

5 Discussion

In this project, we use our own model to make hand gestures recognition, it works well to use three different results to make different operations to control the music player app. So, we can use the iOS camera and Core ML to develop the image recognition apps.

However, there are still some problems need to be solved. In AR/VR apps, we also want to use our hands like a cursor on the screen, so we want to not only know which gesture it is but also need to know where it is so that we can use our hand more freely. But we tried several ways like sliding window, it works not so well because the computation of mobile devices cannot deal with complex calculation. And after class, we find there are YOLO and SSD can help us to solve this problem. So our next step is to use a more fast algorithm to detect hand gestures and tracking the hand.

6 Conclusion

Apple gives developer useful tools to make the development more convenient, so we can use them easily to build our own computer vision apps. But at the same time, we need to know the calculation power of the mobile device is not powerful as PC. And in the following steps, we need to focus more on how to use a faster algorithm to detect the hands on the image and tracking them. Then in our VR/AR apps, we can use our hands easily and freely to make interaction with the virtual objects in the virtual environment.

References

[1] Apple Inc. Arkit framework. https://developer.apple.com/documentation/arkit.

[2] Apple Inc. Avfoundation framework. https://developer.apple.com/documentation/avfoundation.

[3] Apple Inc. Core ml and vision demo. https://developer.apple.com/documentation/vision/classifying_images_with_vision_and_core_ml.

[4] Apple Inc. Core ml framework. https://developer.apple.com/documentation/coreml.

[5] Apple Inc. Custom vision framework. https://docs.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/home.

[6] Hunter Ward. Create your own object recognizer—ml on ios. https://medium.com/@hunter.ley.ward/create-your-own-object-recognizer-ml-on-ios-7f8c09b461a1. Nov 12, 2017.