Mobile Application Development: Graduate Assignment Report

Table of Contents

Title Page

CoreML:

CoreML was introduced in June 2017 by WWDC and is a framework that aid in creating machine learning models and their related functionality. It has support for Image processing (Image classification, object detection, style transfer, hand pose classification), Video processing (Style transfer, Action classification, hand action classification), Motion processing (Activity classification), Sound processing (Sound classification), Text processing (Text classification, word tagging), Table processing (Tabular classification, tabular regression, and recommendation).

CoreML has a dedicated ML API which we can use to boost the performance of the application by having smart features. It mainly has API in 4 categories Vision framework, Natural Language framework, Speech framework, and Sound Analysis framework. These are very powerful and internally uses and has support to Support Vector machines, Ensemble models, Neural networks, Convolutional neural networks, generalized linear model, pipeline models, and recurrent neural network.

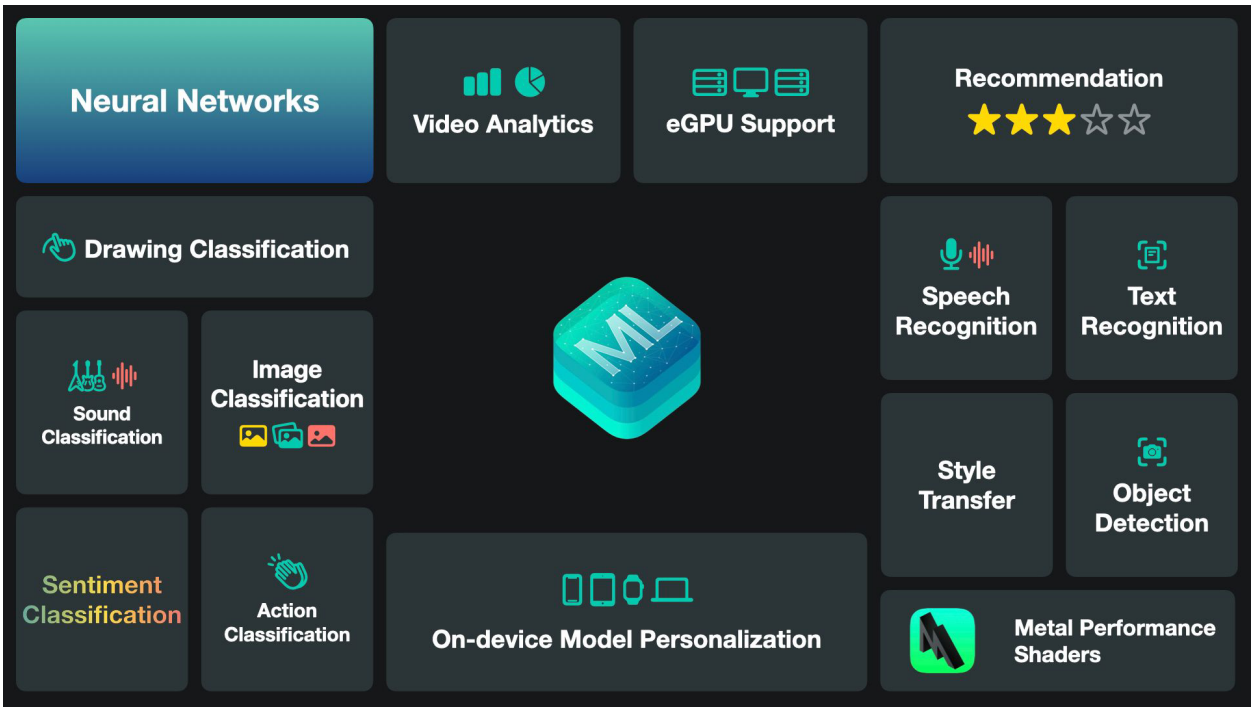

Fig 1.1

There has been the addition of new models with every release of CoreML such as with version2, Bayesian regressor, non-maximum suppression, and Custom models. Custom models are used when our requirements and needs differ from the original implementation with custom parameters like learning rate, the number of iteration models to be trained, etc vary.

Also, version3 brought in some new models like K-Nearest Neighbors (KNN), sound analyzer preprocessor, linked models, and word embedding and support for feature extractors. With CoreML3, we can also create tensors from a distribution such as RandomNormalStaticLayer, RandomUniformStaticLayer,

RandomBernoulliStaticLayer, etc. At later date, more layers were added.

A neural network works as our brain does, it has an input layer, hidden layer, and output layer. The input layer contains as many neurons as several inputs while the hidden layer has no bound and can contain as many neurons and layers as it needs without any bounds. The denser the model is more time the model takes to train and the more accurate it is. It is not always the case that the model will be having high accuracy as the number of hidden layers increases but it can lead to overfitting.

It is always good to introduce some noise in the model to avoid learning all the training data, because if it learns training data then the model will not be able to generalize well. To achieve this “Create ML” with XCode has provided a check box to add noise to the model along with cropping the image as required.

K-NN is a supervised machine learning model which works by checking the similarities between the classes. It can have multiple classes i.e., labels and there are techniques. For the calculating distance, we have Euclidian, Manhattan (for continuous values), and Hamming distance (for categorical data). It selects the points closest to the query point k and then for

classification, it checks for the most votes and in terms of regression it averages the labels.

Journal of Challenges:

The most important step to train any machine learning model is to get excellent training data along with labels. The more the data, the better it is. Finding the right set of images of railroad boxcars was a challenge as there was no available dataset of the railroad boxcar with classification as rusted, new, good, wrecked, or similar. Also, classifying images with the respective labels is very important as just one image mislabeled can have an impact on the accuracy of the model especially if the training data is less.

There were multiple instances where I had to relabel the images to get the desired level of accuracy. Also having 5 labels and classifying them based on corrosion level because a rusted railroad boxcar would look very similar to a new railroad boxcar with brown color. Also, there is a slight difference between the conditions as good and new of the railroad boxcar.

Integrating CoreML with the iOS App:

Another area of challenge was integrating the model with the application. But the apple documentation is very useful in guiding how to integrate the application and the methods available in the classes.

I noticed a bug with the latest version of XCode running on Macintosh with a new apple silicon chip (M1 chip), where the application would crash every time after uploading an image using the photos application. The application when ran on a physical device like iPhone 13 pro works fine.

Application Design:

The application is designed to display the accuracy with which the image is being classified by the labels. There are five labels i.e., new, good, bad, rusty, and wrecked. Once the image is uploaded then the application segues to the destination view controller with the class identified as the best suited for the image.

Along with the class identified, the destination view controller also shows the confidence percentage of each of the classes along with the image.

Please refer application guide for more information.

References:

1. Apple documentation [https://developer.apple.com/documentation/coreml]
2. Machine Think [https://machinethink.net/blog/new-in-coreml3/]
3. K-NN Model [https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4]
4. Integration document [https://developer.apple.com/documentation/coreml/integrating_a_core_ml_model_into_your_app]
5. Fig 1.1 is referenced from [https://www.createwithswift.com/core-ml-explained-apples-machine-learning-framework/]