CS 5143 - MOBILE APPLICATION DEVELOPMENT

SPRING 2022

GRADUATE ASSIGNMENT REPORT

CORE ML

BHAVYA BHIMIREDDY

DEPARTMENT OF COMPUTER SCIENCE

OKLAHOMA STATE UNIVERSITY

INTRODUCTION :

Core ML applies machine learning algorithms to a set of training data to create a model. Use the model to make predictions based on the new input data. Models can perform a variety of tasks that are difficult or unrealistic to write in code. We use these Core ML models for real life predictions and classifications like image classifications, image predictions, action classifications, recommendations etc. These are used across apple products such as macOS, iOS, watchOS, and tvOS

As the Core ML is applied to pre-trained models, we have to be able to train a model in order to get the model from the trained data. These models can be trained using createML which is integrated in Xcode. We can directly train models in our Xcode using createML and can get the model immediately after training the data and can be easily downloaded. This is the easiest way of training to get the models, as they are in CoreML model format. The other way of training the data is by using the Core ML tools which can convert the models into CoreML model formats. We can also find many pre-trained models in “developer.apple.com”.

The models that are trained are downloaded in the CoreML model format, the file extension is ".mlmodel”. These models are dragged into the Xcode project and the algorithms are applied to these models to make the prediction when the new data is shown. Core ML is the foundation for domain-specific frameworks and functionality. Core ML supports Vision to analyze images, Natural Language for text processing, Speech to convert audio to text, and Audio Analysis to identify sounds within audio.

We do not use any network API calls to pass the data, we just use the training data model and create an application with the integration of the coreml model. Once the .mlmodel model is integrated we can give the input of our voice and the analysis report based on the training and testing data in the .mlmodel.

ADVANTAGES OF USING CORE ML :

Low Latency and Near Real-Time Results :
There is no need to create the network API calls where we have to send the data and wait for their responses. We can see everything which includes real-time results from our device application itself. This can be critical when the application is related to video processing from a device camera where our application resides.

Availability (Offline), Privacy, and Compelling Cost :
These applications can run in the devices without any internet connection that is they are available offline. As we are not using any third party api calls the data never leaves the device and at any time we can use the device for the analysis of real time objects.

DISADVANTAGES OF USING CORE ML :

Application Size:
The application size can be larger than expected as we are integrating the models in the application. The application size can be even larger for the models which are more accurate to make the analysis.

System Utilization:
For the prediction analysis there may be a heavy need for computation to make them work on mobile devices which may result in battery drains. Sometimes there may be some problems with the older versions of mobile devices where we can find it difficult to make accurate predictions.

Model Training:
As discussed above we have to train models and integrate them in the application to make predictions. But, we have to train these models from time to time with the updated data. This may result in a change of model in the application and now the data is larger and results again in the larger application size. This can be explained as one of the major drawbacks of using CoreML.

CORE-ML AND MACHINE LEARNING APIS :

There are 4 important machine learning APIs that are leveraged by the capability of CoreML, they are:

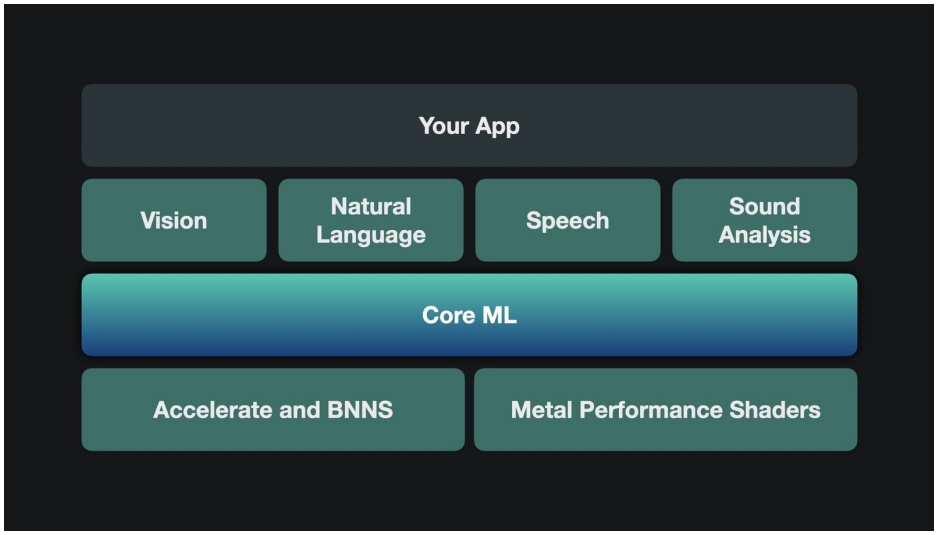
1)Vision Framework

2)Natural Language Framework

3)Speech Framework

4)Sound Analysis

This is how the internal architecture/structure looks like:



→The vision framework is used to analyze the images and videos in a system. They include image classification, object detection and classification of actions.

→ The natural language is used to analyze the text, this provides a NLP functionality to support many different languages and scripts.

→ The speech framework is used to recognize the speech and even can recognize the pre-recorded audio.

→ The sound analysis is used to classify the difference in sounds like the birds singing and the dogs barking.

In this way the above frameworks analyze the data with the leverage of underlying capabilities of the CoreML.

HOW CORE ML WORKS FOR IMAGE CLASSIFICATION :

We can describe the working of a Core ML application using an example
Let the problem statement be to distinguish between the car and bicycle.
First we have to prepare a training and testing dataset with both car and bicycle images. There has to be at least 50 images to get the appropriate results in the training dataset and at least a reasonable amount of images in the testing dataset.
This is how my dataset looks like :

Dataset → training dataset → car(50),bicycle(50)

Dataset → testing dataset → car(10),bicycle(10)

After we are prepared with our dataset we then train our model using createml which resides in xcode. In CreateML we give our testing and training data and then the images are trained. We can directly download the .mlmodel which is our trained data model. Let our model file name be ImageClassifierVehicle.mlmodel. Now we import this model in our xcode project and develop the application to classify the images of car and bicycle based on the input image given by the user.

Now there are three main steps that we need to classify the images in our application.

1) To set up the Model:
   This is the first where we have to set up our model which classifies the images and then VNCoreMLModel(which classifies the images based on our trained model) is processed. Then after classification of the model the request for the result of processing the images is made. The sample code is as shown below

```
lazy var classificationRequest: VNCoreMLRequest = {
do {
  let model = try VNCoreMLModel(for: ImageClassifierVehicle().model)
   let request = VNCoreMLRequest(model: model, completionHandler: {   [weak self] request,
error in
       self?.processClassifications(for: request, error: error)
})
request.imageCropAndScaleOption = .centerCrop
  return request
} catch {
  fatalError("Failed to load Vision ML model: \(error)")
}}()
```

2) To process the classification for the results:
   After the classification is made on the model, the images in the model are converted into the core images for processing. The input image is given by the user, then the VNImageRequestHandler is made to process the image and give the results of the classification based on the trained model.

```
func createClassificationsRequest(for image: UIImage) {
predictionLabel.text = "Classifying..."
let orientation = CGImagePropertyOrientation(image.imageOrientation)
guard let ciImage = CIImage(image: image)
else {
 fatalError("Unable to create \(CIImage.self) from \(image).")
}
DispatchQueue.global(qos: .userInitiated).async {
  let handler = VNImageRequestHandler(ciImage: ciImage, orientation: orientation)
  do {
   try handler.perform([self.classificationRequest])
  }catch {
   print("Failed to perform \n\(error.localizedDescription)")
  }
 }
}
```

3) To update the results with the classification and the confidence:
   Now after the request is made it identifies that the model is VNClassificationObservation as it belongs to the classifier model. Once the request is made for the results, the result is obtained with a label of classification with confidence and is displayed in the UI.

```
func processClassifications(for request: VNRequest, error: Error?) {
DispatchQueue.main.async {
  guard let results = request.results
  else {
    self.predictionLabel.text = "Unable to classify image.\n\(error!.localizedDescription)"
    return
  }
  let classifications = results as! [VNClassificationObservation]
  if classifications.isEmpty {
    self.predictionLabel.text = "Nothing recognized."
  } else {
    let topClassifications = classifications.prefix(2)
    let descriptions = topClassifications.map { classification in
      return String(format: "(%.2f) %@", classification.confidence, classification.identifier)
  }
  self.predictionLabel.text = descriptions.joined(separator: " |")
 }
}
```

REFERENCES :

The links below are referred to for the content of this report.

https://developer.apple.com/documentation/coreml

https://medium.com/datadriveninvestor/image-classifier-using-create-ml-core-ml-and-vision-framework-in-swift-345557960786

https://www.createwithswift.com/core-ml-explained-apples-machine-learning-framework/