

Mobile Application Development



Oklahoma State University, Stillwater

Faisal Jaffri

MAD-CS 5143 - Online

Dr. Blayne Mayfield

April 29th, 2022

Introduction

When it comes to integrating machine learning models into your iOS app, apple provides a very intelligent framework known as CoreML. With the help of CoreML, you can build, train, and deploy machine learning models very easily into your apple applications.

It provides two important features:

- Load Pre-Trained Model

CoreML provides a very easy and robust way to convert your pre-trained ML model into a class that could be easily used in the XCode.

- Make Predictions

Once the users download the app, all the models run locally on the user's device and remove any dependency on the network. Depending on your model, this helps your app make predictions like image classifications, speech recognition, video, etc.

Apple also provides a list of ready-to-use plug-and-play pre-trained models which you can easily use in the app. It has two broad categories of models, Images, and text. For instance MobileNetV2, Resnet50 image classification models, and BERT-Squad a text-based Question Answering model.

CoreML brings a lot of cool ML features on the plate, but essentially it boils down to only classification and regression. But with that being said, one of the best things about CoreML is that even if you are not from a machine learning background, you can use it easily use it without knowing any of the complexity in the background.

Let's talk about Machine learning modes



Machine learning models are the core of any machine learning application. Different tools available in the market like TensorFlow, PyTorch, Scikit-learn, Catalyst, etc, can train your model. Apple also provides a framework known as CreatML that can be used to train models with the help of tools like Swift and macOS playground. You can train your custom models to perform image classification, extract meaning from text, recognize patterns, and use them in the app. For instance, you can train a model to recognize cats by showing it images of different apps. Once the model is trained, you can easily test it out on the new data and evaluate its performance. If satisfied, you can easily integrate it into your app using Core ML.

App functionality

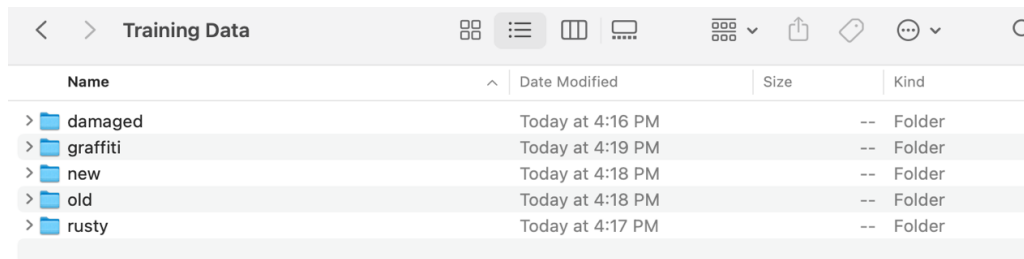
With the help of CoreML, the app distinguishes the condition of railroad boxcars based on the amount of rust, their overall structure, and how it looks.

The app classifies the railroad boxcars into 5 different conditions:

- 1- New, railroad boxcars which have no rust and are brand new
- 2- Rusty, railroad boxcars which contain a lot of rust in them
- 3- Old, railroad boxcars that are too old.
- 4- Damaged/worn-out, railroad boxcars which are completely damaged.
- 5- graffiti, railroad boxcars that contain graffiti on them.

Image Classifier model:

To classify different railroad boxes, I used an Image classifier ML model. I train the model with different railroad boxcar images and labeled them accordingly. For my training data, I created a directory structure that looks like this:

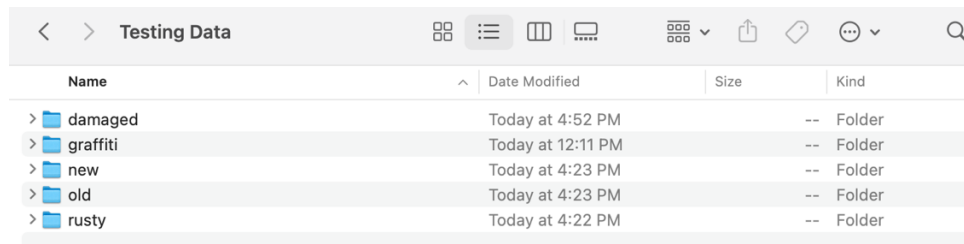


The screenshot shows a file explorer window titled 'Training Data'. It contains a table with columns: Name, Date Modified, Size, and Kind. The table lists five folders: 'damaged', 'graffiti', 'new', 'old', and 'rusty'. Each folder is marked as 'Folder' and has a date and time listed under 'Date Modified'.

Name	Date Modified	Size	Kind
> damaged	Today at 4:16 PM	--	Folder
> graffiti	Today at 4:19 PM	--	Folder
> new	Today at 4:18 PM	--	Folder
> old	Today at 4:18 PM	--	Folder
> rusty	Today at 4:17 PM	--	Folder

Each of the subfolders present in the Training data contains images of different railroad boxes. Each subfolder name represents a label. For example, the label damaged will contain all the railroad boxcar images which are damaged, met with an accident, or worn out.

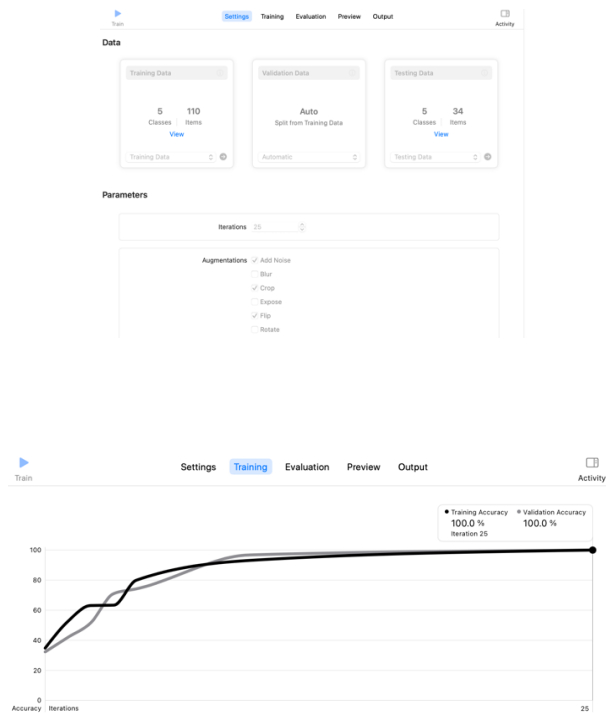
In the same way, we will have Testing data which we can feed into the CreateML framework to test our input data.



The screenshot shows a file explorer window titled 'Testing Data'. It contains a table with columns: Name, Date Modified, Size, and Kind. The table lists five folders: 'damaged', 'graffiti', 'new', 'old', and 'rusty'. Each folder is marked as 'Folder' and has a date and time listed under 'Date Modified'.

Name	Date Modified	Size	Kind
> damaged	Today at 4:52 PM	--	Folder
> graffiti	Today at 12:11 PM	--	Folder
> new	Today at 4:23 PM	--	Folder
> old	Today at 4:23 PM	--	Folder
> rusty	Today at 4:22 PM	--	Folder

Once our dataset is prepared, we can feed it into the CreateML framework, which provides crucial information on our data. Below is the snapshot of the framework:



MyBoxcarImageClassifier 1

Model Type: Image Classifier
Size: 66 KB
Document Type: Core ML Model
Availability: macOS 10.14+ | iOS 12.0+ | tvOS 12.0+

Get Xcode Share

General Predictions

Metadata

Description
Classifier to distinguish boxcar on the way it looks i.e. rusty, old, new, damaged, and contains graffiti

Author
Faisal Jaffri

License
--

Version
--

Class Labels 5

Label
damaged
graffiti
new
old
rusty



new
100% confidence

Once satisfied with the output, I downloaded the “.mlmodel” file which gets integrated into the Xcode with the help of CoreML.

How the app is created:

The main storyboard contains a navigation controller, and view controller with UIImageView, and the navigation bar button item “camera”.

The app uses “UIImagePickerControllerDelegate” and “UINavigationControllerDelegate” to capture image from the user device.

The app uses CoreML framework to interact with the custom image classifier model which we created in the previous step.

It uses:

- VNCoreMLModel
- VNCoreMLRequest
- VNImageRequestHandler

Once the image is fed into the CoreML engine, with the help of our trained “.mlmodel”, It gives “VNClassificationObservation” as result.

This result contains different confidence value for the given input image, We take the image which have the highest confidence value.

```
<VNClassificationObservation: 0x2807266a0>
F16B04F4-19DC-4FAF-B05F-903C9B896AA2 VNCoreMLRequestRevision1
confidence=0.990721 "rusty", <VNClassificationObservation: 0x280726760>
30E6E725-4F7E-434C-B999-111625F4D312 VNCoreMLRequestRevision1
confidence=0.009259 "damaged", <VNClassificationObservation:
0x280726700> 0D2424A4-EAC6-42B4-91A0-CC05A4E76F2E
VNCoreMLRequestRevision1 confidence=0.000016 "new",
<VNClassificationObservation: 0x280726820>
8501F17A-3355-4403-A87E-5E314A7D6244 VNCoreMLRequestRevision1
confidence=0.000005 "old", <VNClassificationObservation: 0x2807267c0>
633930E2-C40A-4439-9B65-4C903F6AF49F VNCoreMLRequestRevision1
confidence=0.000000 "graphiti "]
```

Core ML limitations:

Although CoreML is best for the IOS app developer community, it has its own limitations. The pre-trained model remains as it is, it becomes static. It means that you cannot use user-generated data to train your application further, the model only learns what it has learned when it was created. User-generated data cannot be used in any way to make the application better.

REFERENCES:

- <https://developer.apple.com/machine-learning/models/#image>
- <https://developer.apple.com/documentation/coreml>
- <https://developer.apple.com/machine-learning/>