Gerardo Lopez Falcón Follow
I'm from Costa Rica. I love to learn new technologies and share time with my family and friends...
Oct 3 · 7 min read

Creating an IOS app with Core ML from scratch!

> *The World* ***Requires Choice and Resolution*** — Re Creators anime.

In machine learning, everything starts with the model, the system that makes predictions or identifications. Teaching computer to learn involves a machine learning algorithm with training data to learn from. The output generated from training is usually known as machine learning models. There are different types of machine learning models that solve the same problem (e.g. object recognition) but with different algorithms. Neural Networks, Tree Ensembles, SVMs are some of these machine learning algorithms.

Machine learning like a iterative process

At first, we experiment with a public model but to bring a unique market value and advantage we want our model to outperform the others. What we are looking for is called an **ML feedback loop**. What Google does with its ML features follows this pattern:

- Get init data (once)
- ————————
- Label data
- Train model
- Test model
- Run model in production
- Get new data (and repeat)

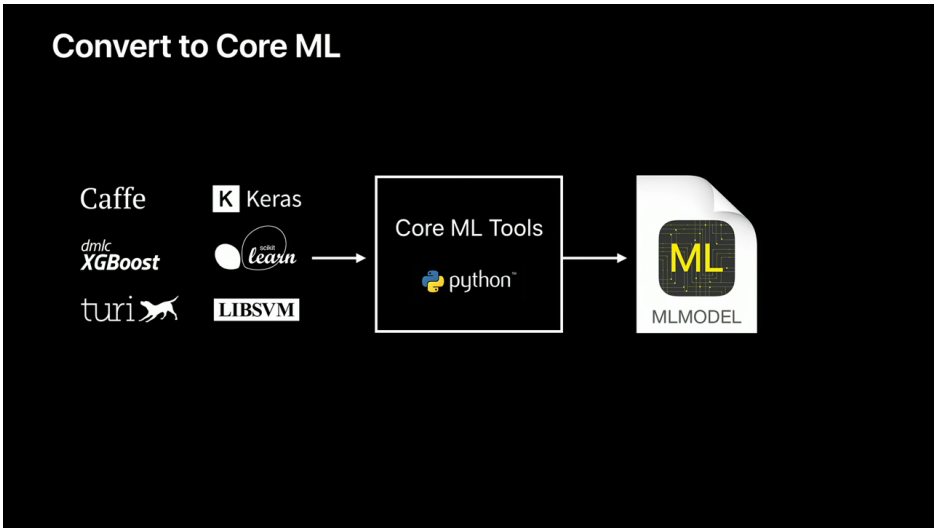
Now, for an mobile app, the process looks like:

In the above image, it seems that the mobile app uses the model created by ML but how does it work? Yeah, here is where Core ML comes in action.

How does CoreML use?

Core ML is new machine learning framework from Apple. It brings machine learning models to Apple devices and makes it easy for developers to take an advantage of ML. We can use a dozen of models prepared by Apple or convert the open-sourced model from popular ML frameworks like Keras, Caffe or scikit-learn.

The workflow for creating an IOS app using CoreML looks like:



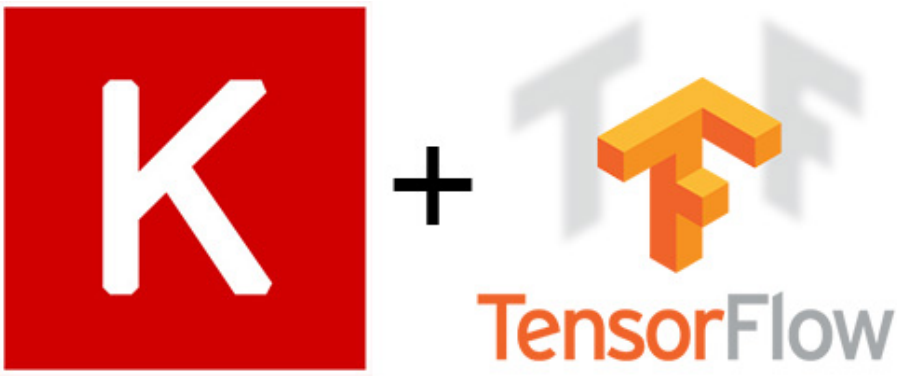
1- You need to create a data model using some ML Framework like Caffe, turi, Keras, etc.

2- To install the Python framework called Core ML Tools, it converts the data model to Core ML format. The result of the conversion is a file with a ***mlmodel*** extension.

3- That is it, you take the model created by Core ML Tools and use it withit your mobile app.

Software requirements

To train the model we need an ML framework. The most popular is Tensorflow developed by Google. It has the best support from community, tutorials and a lot of attention of developers. However, when you get a little deeper you will probably end up on Github issue page or stack overflow dealing with some mathematical issues or undocumented code. Compared to web applications or mobile development ML is still a little baby and as a developer, you need to count on it. Schedule some extra time for being lost in ML mystery. It is also easier to start with a high-level library like Keras. You can check links to some training tutorial at the end of the article.



Tensorflow and Keras are the most common ML libraries

Hardware

Many people say that we need a GPU to train a model. This is true for projects that require high accuracy or some network architecture tuning. If we need an image classifier for 10 classes then we can take advantage of transfer learning and fine-tune our model in 10 minutes on standard CPU. However, for real production applications, we will usually need a GPU performance. We have tried a couple of cloud providers and Amazon AWS g2.2xlarge instance is a good choice

Get Started!!

At this point, you know the necessaries tools that you need to create an IOS app using Machine learning, let's go!

Pipeline Setup:

To use the Core ML tools, the first step is to install Python on your Mac. First, download Anaconda (Choose Python 2.7 version). Anaconda is a super easy way to run Python on your Mac without causing any problems. Once you have Anaconda installed, head over to terminal and type the following:

```
conda install python=2.7.13

conda update python
```

The next step is to create a virtual environment. In a virtual environment, you can write programs with different versions of Python or packages in them. To create a new virtual environment, type the following lines of code.

```
conda create --name handwriting
```

When Terminal prompts you,

```
proceed ([y]/n)?
```

Type "y" for yes. Congrats! Now, you have a virtual environment named handwriting!

Lastly, type the following commands to install the Core ML Tools and Kera's dependencies:

```
pip install -U coremltools
pip install tensorflow==1.1
pip install keras==2.0.4
pip install h5py
```

Designing & Training the network

For this post like you noticed I'll use Kera like the ML framework for our example.

Ok, you should create a python file called `train.py` . Within the `train.py` insert the following code:

- First let us import some necessary libraries, and make sure that keras backend in TensorFlow:

```
import numpy as np

import keras

from keras.datasets import mnist

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten

from keras.layers.convolutional import Conv2D, MaxPooling2D

from keras.utils import np_utils

# (Making sure) Set backend as tensorflow

from keras import backend as K

K.set_image_dim_ordering('tf')
```

- Now let us prepare the dataset for training and testing.

```
# Define some variables

num_rows = 28

num_cols = 28

num_channels = 1

num_classes = 10

# Import data

(X_train, y_train), (X_test, y_test) = mnist.load_data()

X_train = X_train.reshape(X_train.shape[0], num_rows,
num_cols, num_channels).astype(np.float32) / 255

X_test = X_test.reshape(X_test.shape[0], num_rows,
num_cols, num_channels).astype(np.float32) / 255

y_train = np_utils.to_categorical(y_train)

y_test = np_utils.to_categorical(y_test)
```

- Design the model for training.

```
# Model

model = Sequential()

model.add(Conv2D(32, (5, 5), input_shape=(28, 28, 1),
activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.5))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.2))
```

```
model.add(Conv2D(128, (1, 1), activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.2))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dense(num_classes, activation='softmax'))

model.compile(loss='categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
```

- Train the model.

```
# Training

model.fit(X_train, y_train, validation_data=(X_test,
y_test), epochs=10, batch_size=200, verbose=2)
```

- Prepare model for inference by removing dropout layers.

```
# Prepare model for inference

for k in model.layers:

if type(k) is keras.layers.Dropout:

model.layers.remove(k)
```

- Finally save the model.

```
model.save('handWritten.h5')
```

Then, go to the terminal that you opened early, type the following:

```
python train.py
```

It'll create a data model trained called ***handWritten.h5***.

Keras to CoreML

To convert your model from Keras to CoreML, we need to do few more additional steps. Our deep learning model expects a 28×28 normalized grayscale image, and gives probabilities for the class predictions as output. Also, let us add little more information to our model such as license, author etc. Let's create a file called `convert.py` and insert the following code:

```
import coremltools

output_labels = ['0', '1', '2', '3', '4', '5', '6', '7',
'8', '9']

scale = 1/255.

coreml_model =
coremltools.converters.keras.convert('./handWritten.h5',

input_names='image',

image_input_names='image',

output_names='output',

class_labels= output_labels,

image_scale=scale)

coreml_model.author = 'Gerardo Lopez Falcon'
```

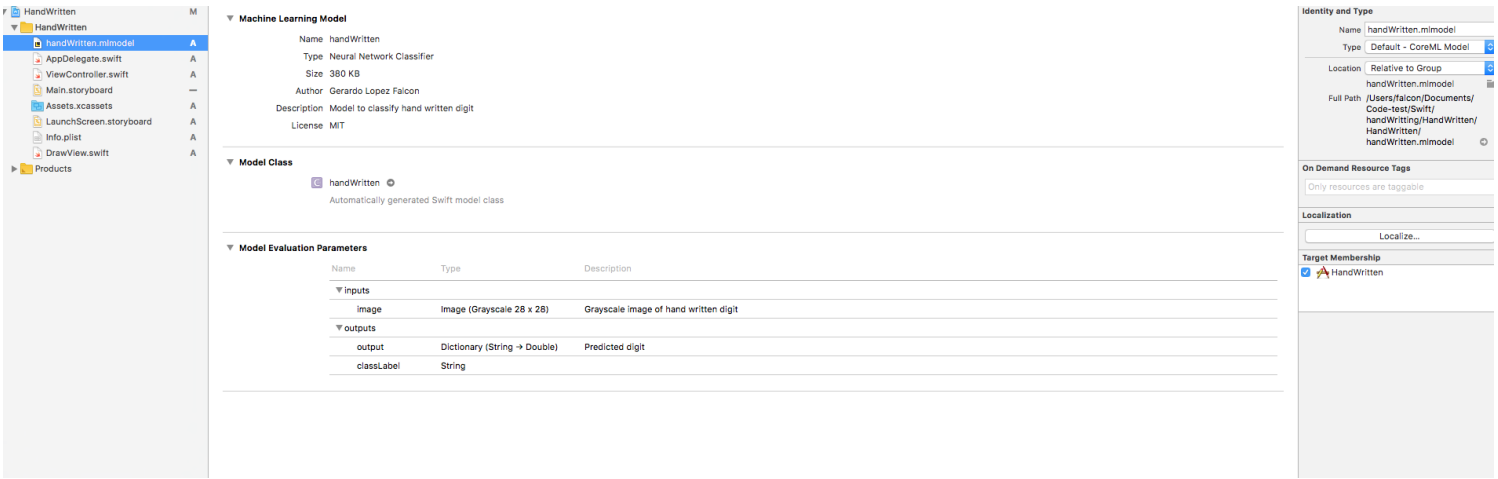
```
coreml_model.license = 'MIT'

coreml_model.short_description = 'Model to classify hand
written digit'

coreml_model.input_description['image'] = 'Grayscale image
of hand written digit'

coreml_model.output_description['output'] = 'Predicted
digit'


coreml_model.save('handWritten.mlmodel')
```

Come back to the terminal and type the following: `python convert.py` . It create us a file called handWritten.mlmodel (Now, we can use this file within an IOS app).

Integrating the Model into Xcode

Now we come to the final step to integrate the model we just converted into the Xcode project. Open the starter project and based on what you have learned so far, I'm challenging you to integrate the Core ML model into the app.

The first step is to drag and drop the `handWritten.mlmodel` into our Xcode project. Make sure that the box for Target Membership is checked.



Now, go to the ViewController below where are the IBOutlet instantiated and create one object from handWritten type:

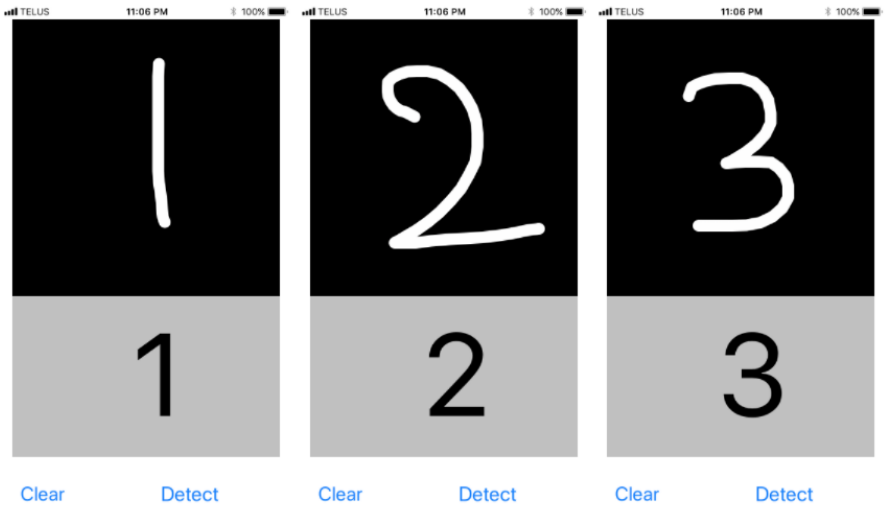
```
let model = handWritten()
```

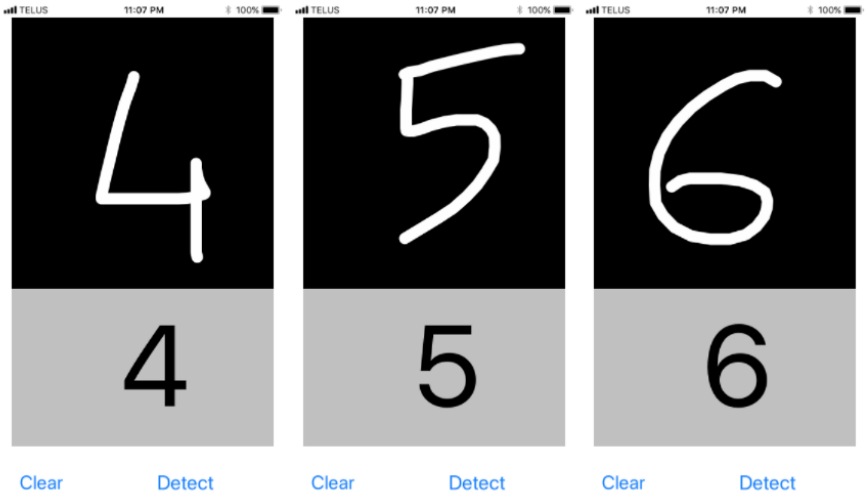
After, go to tappedDetect function and below of the `let pixelBuffer = .....` insert:

```
let output = try? model.prediction(image: pixelBuffer!)
```

we just have defined the constant prediction equal to the data from the model's prediction.

That's all! Build and run the app. These are the results of the app when tested on iPhone 7.





Conclusion

Now that you know how to convert data models, you are probably wondering where you can find data models. A simple Google search will give you tons of results. You can find data models for almost any category such as different types of cars, plants, animals, and even a model that can tell which celebrity you look like the most. Here are a couple of places to get you started!

- Caffe Model Zoo
- UCI Machine Learning Repository
- Deep Kearning Datasets

If you can't find a model that supports your needs, you are probably wondering if you can create your own data model. It can be done, but it will be difficult to do so. If you feel like you are up to the challenge, I would recommend you start with either Scikit-Learn or TensorFlow by visiting their homepage.

Thanks for sharing, if you can share this post with others, I'll appreciate it. And, I hope your claps :D