Article

Converting Trained Models to Core ML

Convert trained models created with third-party machine learning tools to the Core ML model format.

Overview

If your model is created and trained using a supported third-party machine learning tool, you can use Core ML Tools to convert it to the Core ML model format. Table 1 lists the supported models and third-party tools.

Note

Core ML Tools is a Python package (`coremltools`), hosted at the Python Package Index (PyPI). For information about Python packages, see Python Packaging User Guide.

Table 1 Models and third-party tools supported by Core ML Tools

Model type	Supported models	Supported tools
Neural networks	Feedforward, convolutional, recurrent	Caffe v1 Keras 1.2.2+
Tree ensembles	Random forests, boosted trees, decision trees	scikit-learn 0.18 XGBoost 0.6
Support vector machines	Scalar regression, multiclass classification	scikit-learn 0.18 LIBSVM 3.22
Generalized linear models	Linear regression, logistic regression	scikit-learn 0.18
Feature engineering	Sparse vectorization, dense vectorization, categorical processing	scikit-learn 0.18
Pipeline models	Sequentially chained models	scikit-learn 0.18

Convert Your Model

Convert your model using the Core ML converter that corresponds to your model's third-party tool. Call the converter's `convert` method and save the resulting model to the Core ML model format (`.mlmodel`).

For example, if your model was created using Caffe, pass the Caffe model (`.caffemodel`) to the `coremltools.converters.caffe.convert` method.

```
import coremltools
coreml_model = coremltools.converters.caffe.convert('my_caffe_model.caffemodel')
```

Now save the resulting model in the Core ML model format.

```
coremltools.utils.save_spec(coreml_model, 'my_model.mlmodel')
```

Depending on your model, you might need to update inputs, outputs, and labels, or you might need to declare image names, types, and formats. The conversion tools are bundled with more documentation, as the options available vary by tool. For more information about Core ML Tools, see the Package Documentation.

Alternatively, Write a Custom Conversion Tool

It's possible to create your own conversion tool when you need to convert a model that isn't in a format supported by the tools listed in Table 1.

Writing your own conversion tool involves translating the representation of your model's input, output, and architecture into the Core ML model format. You do this by defining each layer of

Note

The Core ML model format is defined by a set of protocol buffer files and is described in detail in the Core ML Model Specification.

See Also

First Steps

Getting a Core ML Model

Obtain a Core ML model to use in your app.

{} Integrating a Core ML Model into Your App

Add a simple model to an app, pass input data to the model, and process the model's predictions.