What’s New in Core ML
Part two

Aseem Wadhwa, Core ML
Sohaib Qureshi, Core ML
Core ML
Part One Recap

Model size

Performance

Customization

Float weights → Quantized weights

60 MB → 15 MB
Part One Recap

Model size
Performance
Customization
Part One Recap

Model size

Performance

Customization
Part One Recap

Model size
Performance
Customization
Part Two
Agenda
Agenda

Core ML Tools ecosystem
Agenda

Core ML Tools ecosystem

Quantization utilities
Agenda

Core ML Tools ecosystem
Quantization utilities
Custom conversion
Core ML Tools ecosystem
Quantization utilities
Custom conversion
Download
Models

MobileNet

MobileNets are based on a streamlined architecture that have depth-wise separable convolutions to build lightweight, deep neural networks.

Detected the dominant objects present in an image from a set of 1000 categories such as trees, animals, food, vehicles, people, and more.

- View original model details
- Download Core ML Model (171 MB)

Last Year

Core ML Tools

https://github.com/apple/coremltools
Last Year

Core ML Tools

- Caffe
- Keras
- LIBSVM
- turi
- dmlc
- XGBoost
- Python

MLMODEL
Today

Core ML Tools

- TensorFlow
- ONNX
- MXNet
- Caffe
- turi
- dmlc
- XGBoost
- LIBSVM
- Keras
- IBM Watson
- Yandex
- CatBoost

MLMODEL
TensorFlow Converter
https://github.com/tf-coreml/tf-coreml

In collaboration with Google
TensorFlow Converter
https://github.com/tf-coreml/tf-coreml

Support for custom layers
Support for Quantized TensorFlow models
(coming soon!)

TensorFlow Converter
https://github.com/tf-coreml/tf-coreml
TensorFlow Converter
https://github.com/onnx/onnx-coreml

In collaboration with Facebook and Prisma
TensorFlow Converter
https://github.com/onnx/onnx-coreml
Core ML Tools ecosystem
Quantization utilities
Custom conversion
Core ML Tools 2.0
Quantization utilities
Core ML Tools 2.0
https://github.com/apple/coremltools

Support for latest Core ML .mlmodel specification

Quantization utilities

Flexible shape utilities
Quantization Utilities

MLMODEL → Float32 weights → Core ML Tools → Quantize weights → 1-8 bit weights → MLMODEL

Post-training Quantization
Quantization
Peeking under the hood
Quantization
Peeking under the hood
Quantization

Peeking under the hood
Quantization
Peeking under the hood
Weight Quantization

Peeking under the hood

Min

Max

Float weights
Weight Quantization
Peeking under the hood

Min          Max

Float weights

Quantized weights
Linear
Three-bit example

Min Max

Float weights

Quantized weights
Linear
Three-bit example

Min | Max
---|---
0   | 7

Float weights
scale*quantized + bias
Quantized weights
## Lookup Table

### Three-bit example

<table>
<thead>
<tr>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7</td>
</tr>
</tbody>
</table>

#### Float weights

- 0
- 1
- 2
- 3
- 4
- 5
- 6
- 7

#### Quantized weights
Lookup Table
Three-bit example

Min Max

Float weights

lookupTable[quantized]

Quantized weights
Quantization Utilities

ML MODEL

Float32 weights

Core ML Tools

Quantize weights

1-8 bit weights

ML MODEL
Quantization Utilities

Float32 weights → Quantize weights → 1-8 bit weights

Decide on precision and algorithm
Quantization Utilities

1. Decide on precision and algorithm
2. Let Core ML Tools work its magic!
Demo
Quantization in Core ML Tools
quantized_model = quantize_weights(model, 8, 'kmeans')

compare_model(model, quantized_model, './sample_data/')
// Quantize model using KMeans Lookup Table
quantized_model = quantize_weights(model, 8, 'kmeans')

// Compare Quantized model with original
compare_model(model, quantized_model, './sample_data/')
// Quantize model using KMeans Lookup Table
quantized_model = quantize_weights(model, 8, 'kmeans')

// Compare Quantized model with original
compare_model(model, quantized_model, './sample_data/')
Model Size Versus Agreement
Model Size Versus Agreement

Graph showing the relationship between model size and agreement metric. The x-axis represents model size ranging from Float32 to One bit, and the y-axis represents the agreement metric. Two models, Model 1 (red dots) and Model 2 (blue dots), are plotted on the graph.
Model Size Versus Agreement

<table>
<thead>
<tr>
<th>Model Size</th>
<th>Agreement Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Float32</td>
<td></td>
</tr>
<tr>
<td>One bit</td>
<td></td>
</tr>
</tbody>
</table>

- Model 1
- Model 2
Model Size Versus Agreement

Metrics
- Accuracy
- Signal-to-noise ratio
- Visual inspection
- Model-specific metrics
Core ML tools ecosystem
Quantization utilities
Custom conversion
Custom Conversion

Aseem Wadhwa, Core ML
New Neural Network Layer
New Model Architecture
New Neural Network Layer → Customization → New Model Architecture
New Neural Network Layer → Customization → MLMODEL
Model Conversion

Converters: Simple API
Model Conversion

Converters: Simple API

```python
import coremltools
coremltools.converters.keras.convert(keras_model)
```
Model Conversion

Converters: Simple API

```python
import onnx_coreml
onnx_coreml.convert(onnx_model)
```
Model Conversion

Converters: Simple API

```python
import tfcoreml
tfcoreml.convert(tf_model_path=tf_model_path,
                 mlmodel_path=mlmodel_path,
                 output_feature_names=['output:0'])
```
Model Conversion

Converters: Simple API

```python
import tfcoreml

tfcoreml.convert(
    tf_model_path=tf_model_path,
    mlmodel_path=mlmodel_path,
    output_feature_names=['output:0'],
)
```

`NotImplementedError: Unsupported Ops of type: Tile`
Model Conversion

Converters: Simple API

```python
import tfcoreml
tfcoreml.convert(tf_model_path=tf_model_path,
                  m1model_path=m1model_path,
                  output_feature_names=['output:0'],
)
```

Use Custom Layers!
Custom Layer Examples
Image Classifier

Beach

Model Class

ImageClassifierModel

Model is not part of any target. Add the model to a target to enable generation of the model class.

Model Evaluation Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>input_image</td>
<td>Image (Color 320 x 240)</td>
<td>Input image</td>
</tr>
<tr>
<td>probability_vector</td>
<td>MultiArray (Double 10)</td>
<td>Output probability vector</td>
</tr>
</tbody>
</table>

Beach
Opening the Hood

Neural Network

Beach
Convertible to Core ML

Supported by Core ML
Not convertible to Core ML

- Convolution
- New Activation
- Convolution
- New Activation
- Convolution

Supported by Core ML

Not Supported by Core ML
Custom Layer

- Convolution
- New Activation
- Convolution
- New Activation
- Convolution

Supported by Core ML

Custom Layer in Core ML Model
Image Classifier

- **Model Class**: ImageClassifierModel
  - Model is not part of any target. Add the model to a target to enable generation of the model class.

- **Model Evaluation Parameters**
  - **Inputs**
    - input_image: Image (Color 320 x 240)
      - Input image
  - **Outputs**
    - probability_vector: MultiArray (Double 10)
      - Output probability vector

- **Dependencies**
  - **Custom Layers**
    - AAPl.MyNewActivation: A new activation function I want to try out.

With Custom Activation Layer
Image Classifier

- Model Class: ImageClassifierModel
  - Model is not part of any target. Add the model to a target to enable generation of the model class.

- Model Evaluation Parameters
  - **Name** | **Type** | **Description**
  - **inputs**
    - input_image: Image (Color 320 x 240) | Input image
  - **Outputs**
    - probability_vector: MultiArray (Double 10) | Output probability vector

- Dependencies
  - **Name** | **Description**
  - **Custom Layers**
    - AAAPL_MyNewActivation: A new activation function I want to try out.
## Image Classifier

### Dependencies

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Custom Layers</strong></td>
<td></td>
</tr>
<tr>
<td>AAPLMyNewActivation</td>
<td>A new activation function I want to try out.</td>
</tr>
</tbody>
</table>
A Simple Classifier

Classifier Net

6
Spatial Transformer Network

Localization Net $\theta$ Grid Sampler (Custom layer) Classifier Net

Demo
Conversion with Custom Layer
Custom Layer (Parameters)

gridSampler.swift
import CoreML

@objc(AAPLGridSampler) class AAPLGridSampler: NSObject, MLCustomLayer {

    var Wout: Int = 1
    var Hout: Int = 1

    required init(parameters: [String : Any]) throws {
        self.Wout = parameters["output_width"] as! Int
        self.Hout = parameters["output_height"] as! Int
        super.init()
    }

    func outputShapes(forInputShapes inputShapes: [[NSNumber]]) throws -> [[NSNumber]] {
        var outputShape: [[NSNumber]] = Array(repeating: Array(repeating: 1, count: 5), count: 1)
        outputShape[0][0] = inputShapes[0][0]
        outputShape[0][1] = inputShapes[0][1]
        outputShape[0][2] = inputShapes[0][2]
        outputShape[0][3] = NSNumber(value: self.Hout)
        outputShape[0][4] = NSNumber(value: self.Wout)
        return outputShape
    }

    func evaluate(inputs: [MLMultiArray], outputs: [MLMultiArray]) throws {
        let input_image = inputs[0]
        let theta = inputs[1]
        let output_image = outputs[0]
        let Hout: Int = output_image.shape[3].intValue
        let Wout: Int = output_image.shape[4].intValue
        let (w_grid, h_grid) = compute_grid(input_dims: [Hout, Wout],
                                            output_dims: [output_image.shape[0].intValue, output_image.shape[1].intValue],
                                            theta: theta)
        compute_output(w_grid: w_grid,
                       h_grid: h_grid,
                       input_image: input_image,
                       output_image: output_image)
    }
}
import CoreML

objc(AAPLGridSampler) class AAPLGridSampler: NSObject, MLCustomLayer {

    var Wout: Int = 1
    var Hout: Int = 1

    required init(parameters: [String: Any]) throws {
        self.Wout = parameters["output_width"] as! Int
        self.Hout = parameters["output_height"] as! Int
        super.init()
    }

    func outputShapes(forInputShapes inputShapes: [[NSNumber]]) throws -> [[NSNumber]] {
        var outputShape: [[NSNumber]] = Array(repeating: Array(repeating: 1, count: 5), count: 1)
        outputShape[0][0] = inputShapes[0][0]
        outputShape[0][1] = inputShapes[0][1]
        outputShape[0][2] = inputShapes[0][2]
        outputShape[0][3] = NSNumber(value: self.Hout)
        outputShape[0][4] = NSNumber(value: self.Wout)
        return outputShape
    }

    func evaluate(inputs: [MLMultiArray], outputs: [MLMultiArray]) throws {
        let input_image = inputs[0]
        let theta = inputs[1]
        let output_image = outputs[0]
        let Hout: Int = input_image.shape[3].intValue
        let Wout: Int = input_image.shape[4].intValue
        let (w_grid, h_grid) = compute_grid(input_dims: [Hout, Wout],
                                            output_dims: [self.Hout, self.Wout],
                                            theta: theta)

        compute_output(w_grid: w_grid,
                       h_grid: h_grid,
                       input_image: input_image,
                       output_image: output_image)
    }
}
import CoreML

@objc(AAPLGridSampler) class AAPLGridSampler: NSObject, MLCustomLayer {

    var Wout: Int = 1
    var Hout: Int = 1

    required init(parameters: [String : Any]) throws {
        self.Wout = parameters["output_width"] as! Int
        self.Hout = parameters["output_height"] as! Int
        super.init()
    }

    let Win: Int = input_image.shape[4].intValue
    let Hout: Int = output_image.shape[3].intValue
    let Wout: Int = output_image.shape[4].intValue

    let (w_grid, h_grid) = compute_grid(input_dims: [Win, Win],
                                         output_dims: [Hout, Wout],
                                         theta: theta)

    compute_output(w_grid: w_grid,
                   h_grid: h_grid,
                   input_image: input_image,
                   output_image: output_image)
}
import CoreML

@objc(AAPLGridSampler) class AAPLGridSampler: NSObject, MLCustomLayer {

    var Wout: Int = 1
    var Hout: Int = 1

    required init?(parameters: [String : Any]) throws {
        self.Wout = parameters["output_width"] as! Int
        self.Hout = parameters["output_height"] as! Int
        super.init()
    }

    func outputShapes(forInputShapes inputShapes: [NSNumber]) throws -> [NSNumber] {
        var outputShape: [NSNumber] = Array(repeating: Array(repeating: 1, count: 5), count: 1)
        outputShape[0][0] = inputShapes[0][0]
        outputShape[0][1] = inputShapes[0][1]
        outputShape[0][2] = inputShapes[0][2]
        outputShape[0][3] = NSNumber(value: self.Hout)
        outputShape[0][4] = NSNumber(value: self.Wout)
        return outputShape
    }

    func evaluate(inputs: [MLMultiArray], outputs: [MLMultiArray]) throws {
        let input_image = inputs[0]
        let theta = inputs[1]
        let output_image = outputs[0]
        let Hint: Int = input_image.shape[3].intValue
        let Wint: Int = input_image.shape[4].intValue
        let Hout: Int = output_image.shape[3].intValue
        let Wout: Int = output_image.shape[4].intValue
        let (w_grid, h_grid) = compute_grid(input_dims: [Win, Wint],
                                            output_dims: [Hout, Wout],
                                            theta: theta)

        compute_output(w_grid: w_grid,
                      h_grid: h_grid,
                      input_image: input_image,
                      output_image: output_image)
    }
func evaluate(inputs: [MLMultiArray], outputs: [MLMultiArray]) throws {
    let input_image = inputs[0]
    let theta = inputs[1]
    let output_image = outputs[0]
    let Hin:Int = input_image.shape[3].intValue
    let Win:Int = input_image.shape[4].intValue
    let Hout:Int = output_image.shape[3].intValue
    let Wout:Int = output_image.shape[4].intValue
    let (w_grid, h_grid) = compute_grid(input_dims: [Hin, Win],
                                         output_dims: [Hout, Wout],
                                         theta: theta)
    compute_output(w_grid: w_grid,
                  h_grid: h_grid,
                  input_image: input_image,
                  output_image: output_image)
}
```swift
import CoreML

@objc(AAPLGridSampler) class AAPLGridSampler: NSObject, MLCustomLayer {

    var Wout: Int = 1
    var Hout: Int = 1

    required init(parameters: [String : Any]) throws {
        self.Wout = parameters["output_width"] as! Int
        self.Hout = parameters["output_height"] as! Int
        super.init()
    }

    func outputShapes(forInputShapes inputShapes: [[NSNumber]] throws -> [[NSNumber]] {
        var outputShape: [[NSNumber]] = Array(repeating: Array(repeating: 1, count: 5), count: 1)
        outputShape[0][0] = inputShapes[0][0]
        outputShape[0][1] = inputShapes[0][1]
        outputShape[0][2] = inputShapes[0][2]
        outputShape[0][3] = NSNumber(value: self.Hout)
        outputShape[0][4] = NSNumber(value: self.Wout)
        return outputShape
    }

    func evaluate(inputs: [MLMultiArray], outputs: [MLMultiArray]) throws {
        let input_image = inputs[0]
        let theta = inputs[1]
        let output_image = outputs[0]
        let Hint: Int = input_image.shape[3].intValue
        let Wint: Int = input_image.shape[4].intValue
        let Wout: Int = output_image.shape[3].intValue
        let Hout: Int = output_image.shape[4].intValue
        let (w_grid, h_grid) = compute_grid(input_dims: [Wint, Hint], output_dims: [Wout, Hout], theta: theta)

        compute_output(w_grid: w_grid, h_grid: h_grid, input_image: input_image, output_image: output_image)
    }
```
func outputShapes(forInputShapes inputShapes: [[NSNumber]]) throws -> [[NSNumber]] {
    var outputShape: [[NSNumber]] = Array(repeating: Array(repeating: 1, count: 5), count: 1)
    outputShape[0][0] = inputShapes[0][0]
    outputShape[0][1] = inputShapes[0][1]
    outputShape[0][2] = inputShapes[0][2]
    outputShape[0][3] = NSNumber(value: self.Hout)
    outputShape[0][4] = NSNumber(value: self.Wout)
    return outputShape
}

let Win:Int = input_image.shape[4].intValue
let Hout:Int = output_image.shape[3].intValue
let Wout:Int = output_image.shape[4].intValue
let (w_grid, h_grid) = compute_grid(input_dims: [Win, Win],
    output_dims: [Hout, Wout],
    theta: theta)

compute_output(w_grid: w_grid,
    h_grid: h_grid,
    input_image: input_image,
    output_image: output_image)
Custom Layer

Layer in a Neural Network
Custom Model

New Model

Core ML  Custom Model  Core ML
Summary
Core ML Tools 2.0
Core ML Tools 2.0

Rich Core ML Tools ecosystem
Core ML Tools 2.0

Rich Core ML Tools ecosystem

Easy-to-use quantization utilities
Core ML Tools 2.0

Rich Core ML Tools ecosystem
Easy-to-use quantization utilities
Integrate new layers
## More Information


<table>
<thead>
<tr>
<th>Event</th>
<th>Lab or Lab 12</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine Learning Lab</td>
<td>Technology Lab 2</td>
<td>Wednesday 4:00PM</td>
</tr>
<tr>
<td>Machine Learning Lab</td>
<td>Technology Lab 12</td>
<td>Friday 2:00PM</td>
</tr>
</tbody>
</table>