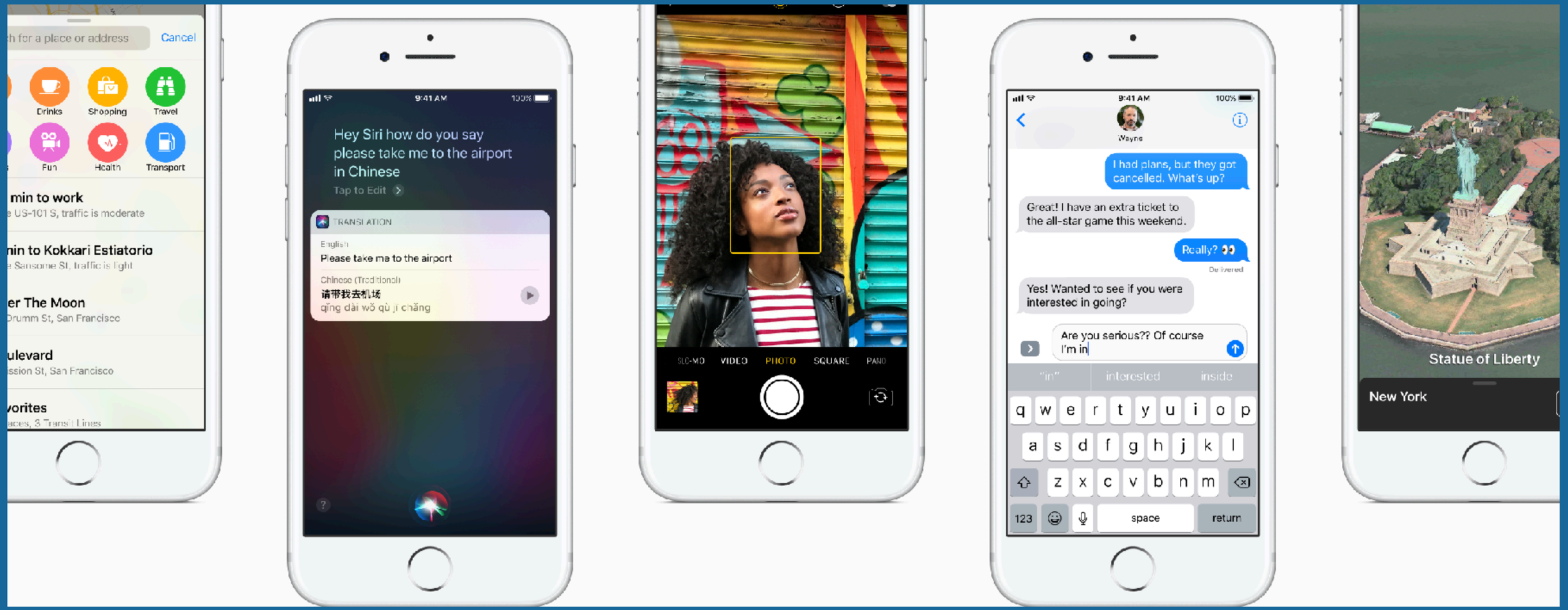
THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 4

VISION WITH COREML

COREML

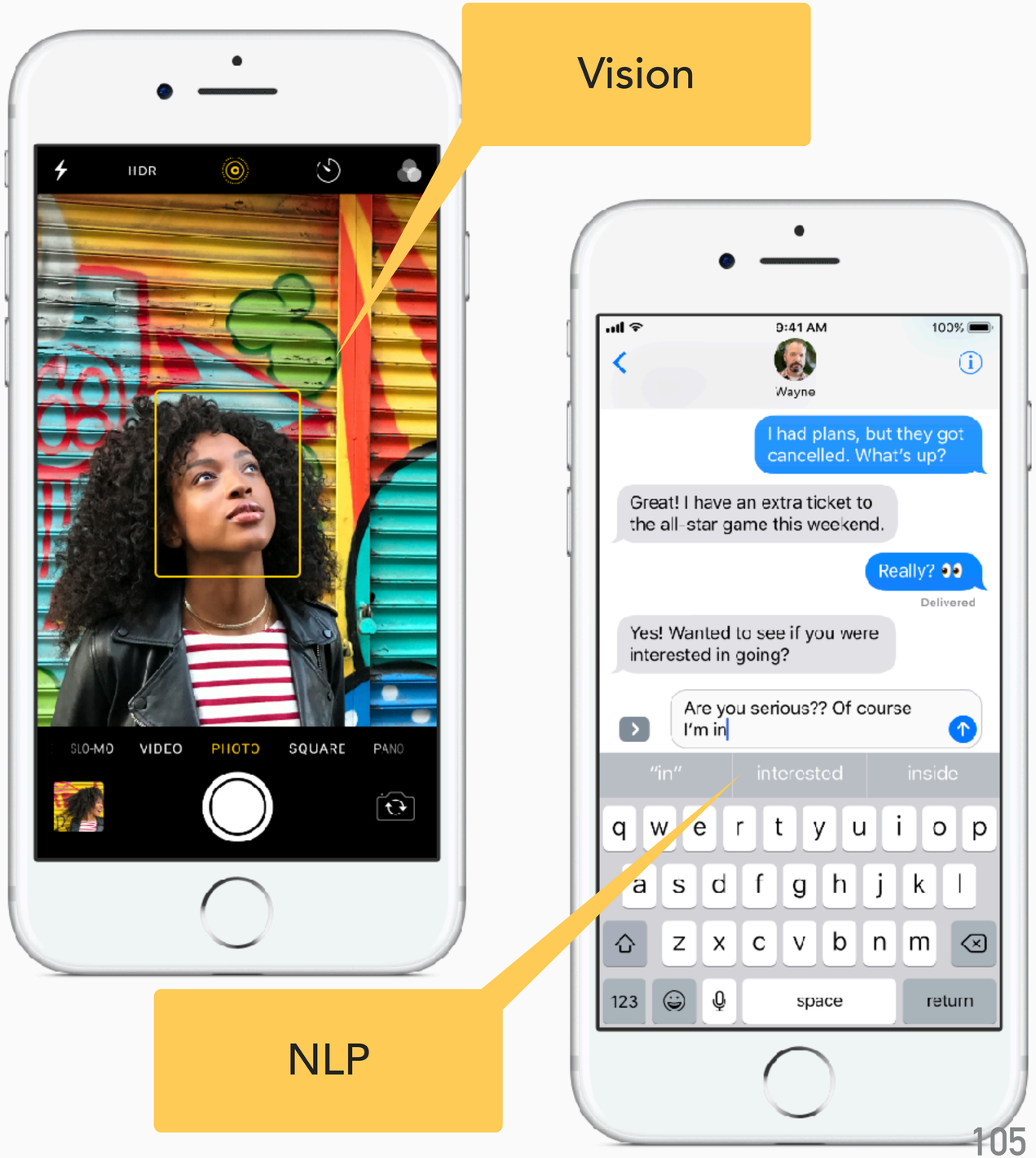


- Framework (iOS11) to integrate machine learning models into your app

COREML

SUBTITLE

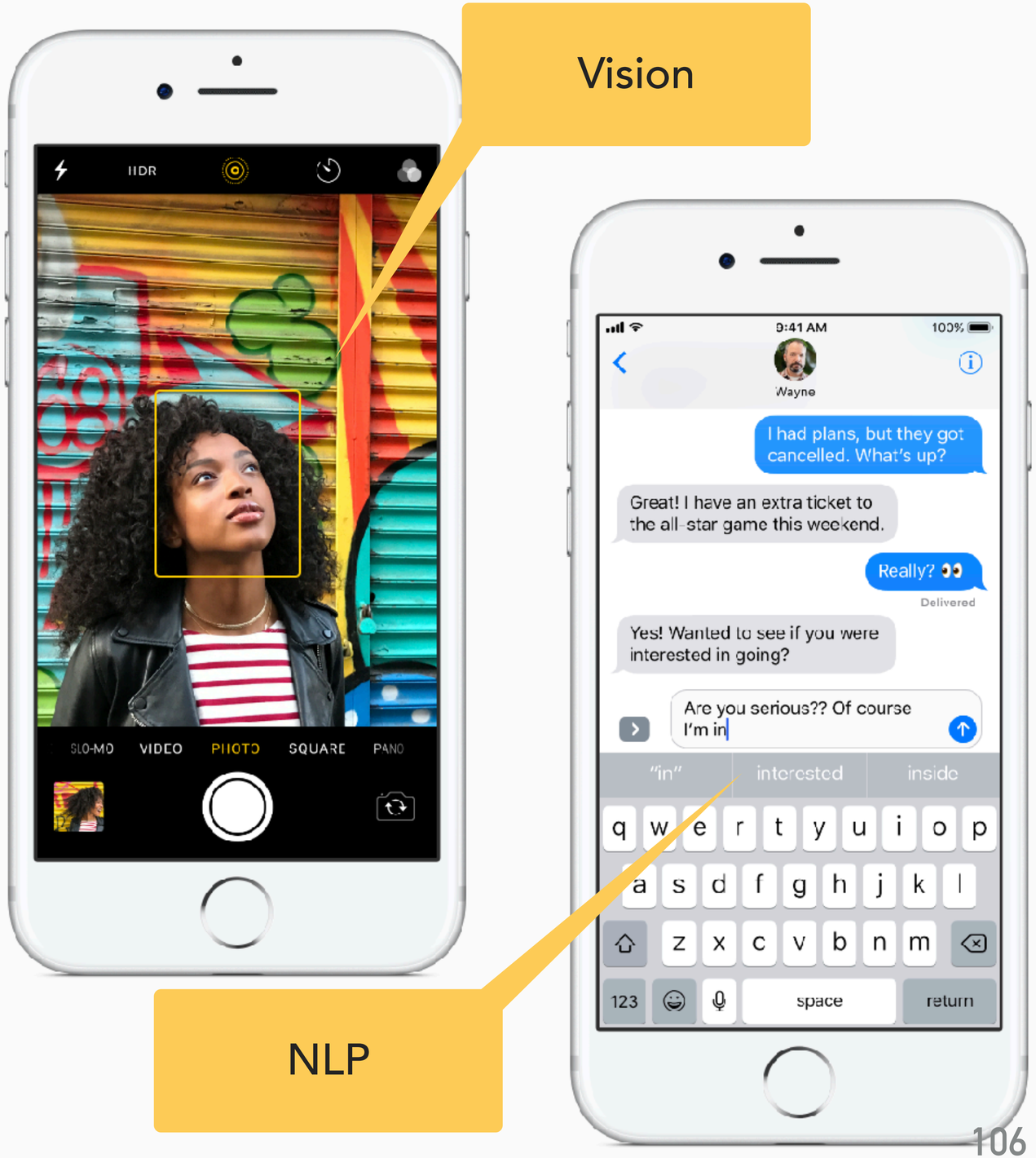
- Core ML is the foundation for domain-specific frameworks and functionality
  - Vision for image analysis
  - Foundation for natural language processing
  - GameplayKit for evaluating learned decision trees



COREML

SUBTITLE

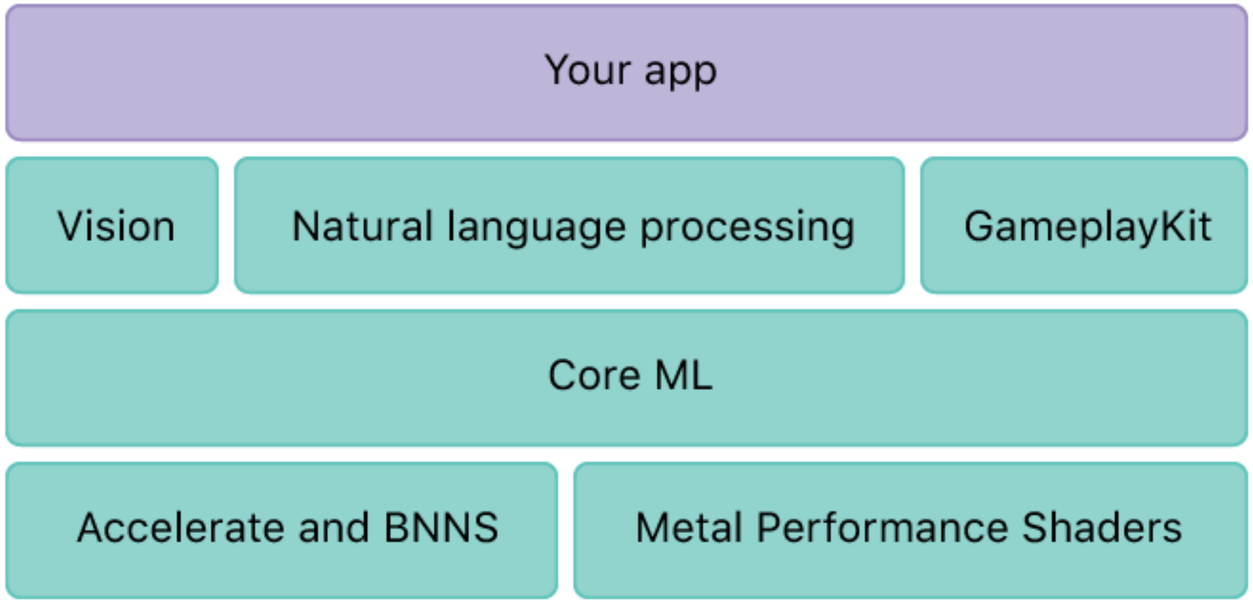
- Apple uses it
  - Siri
  - Camera
  - QuickType
  - Photos



COREML

SUBTITLE

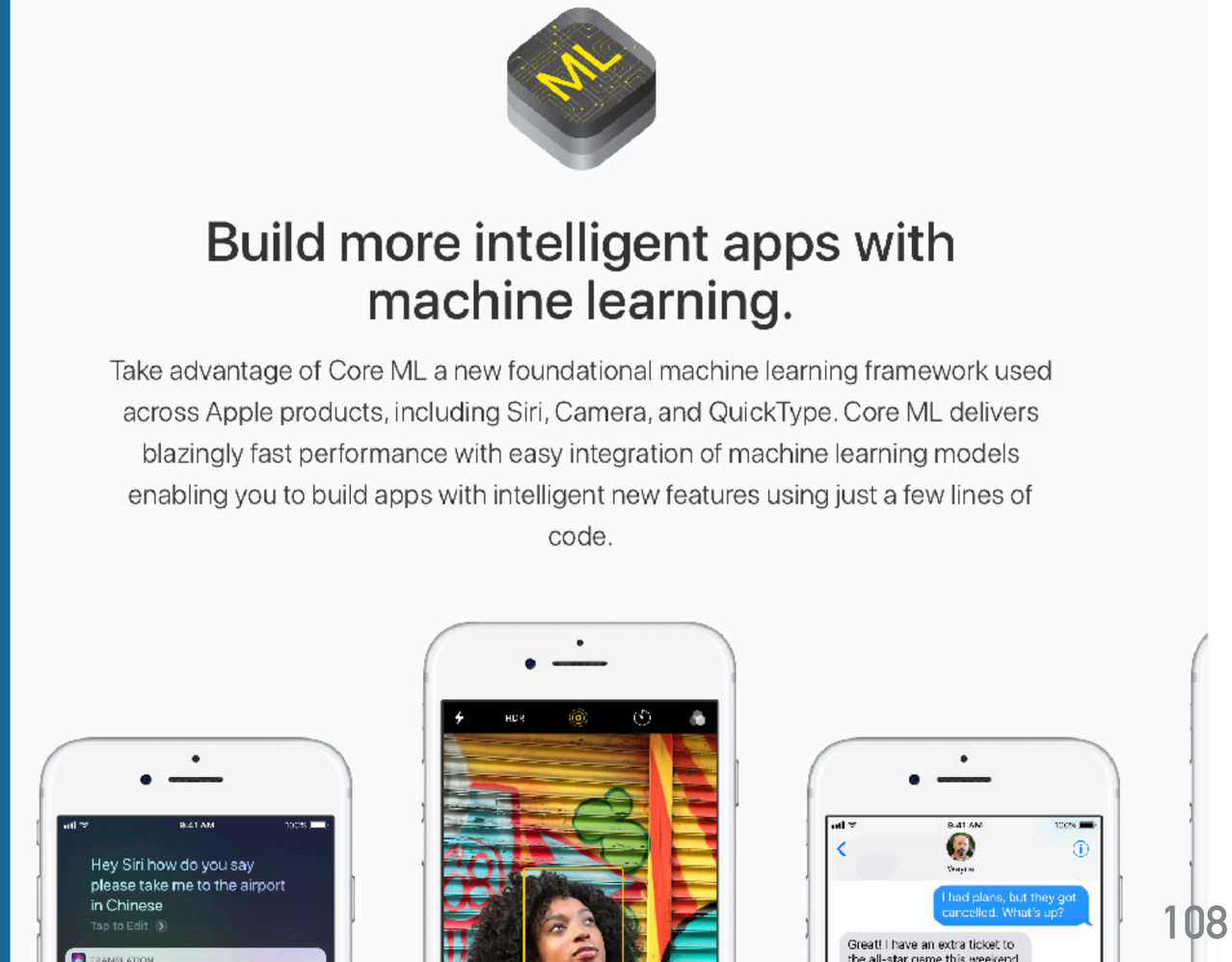
- Core ML itself builds on top of low-level primitives
  - Accelerate
  - BNNS (Basic Neural Network Subroutines)
  - Metal Performance Shaders
- Performed on device



COREML

SUBTITLE

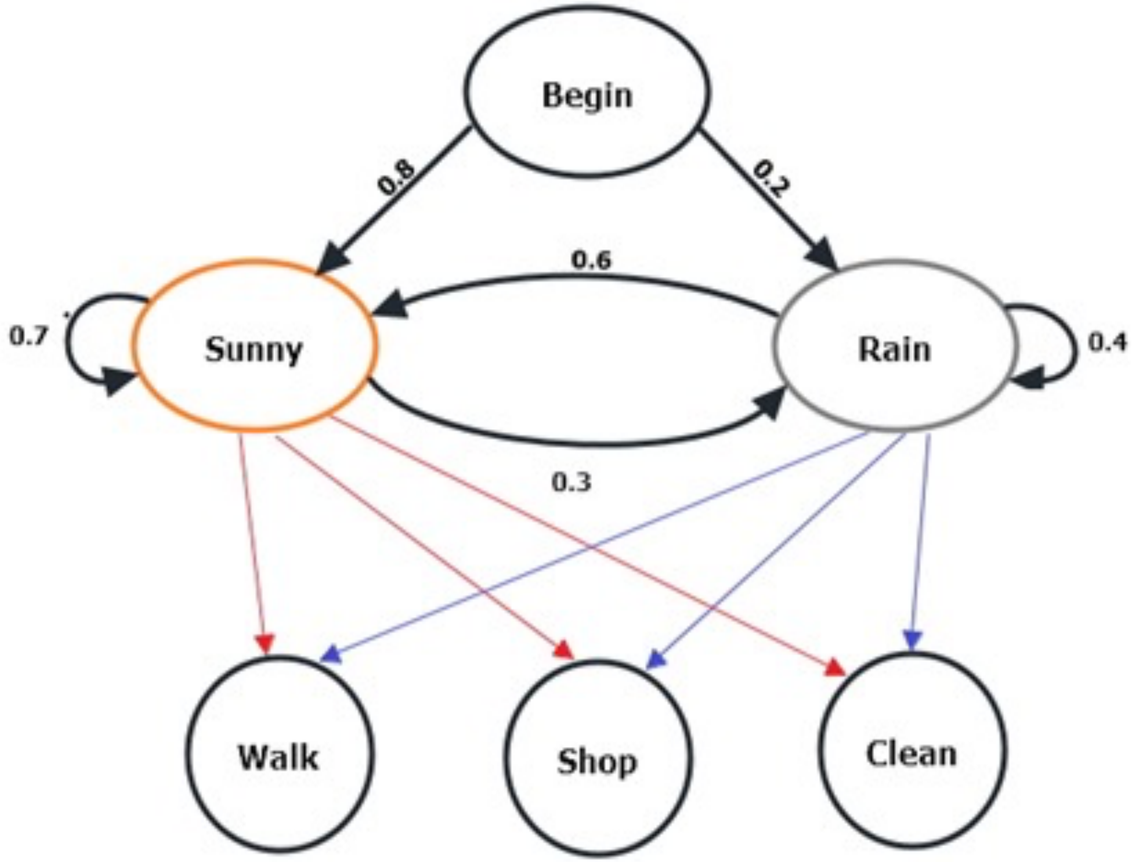
- Machine learning is a class of algorithms that make predictions base on training data
- Predictions can be refined without human intervention at run time



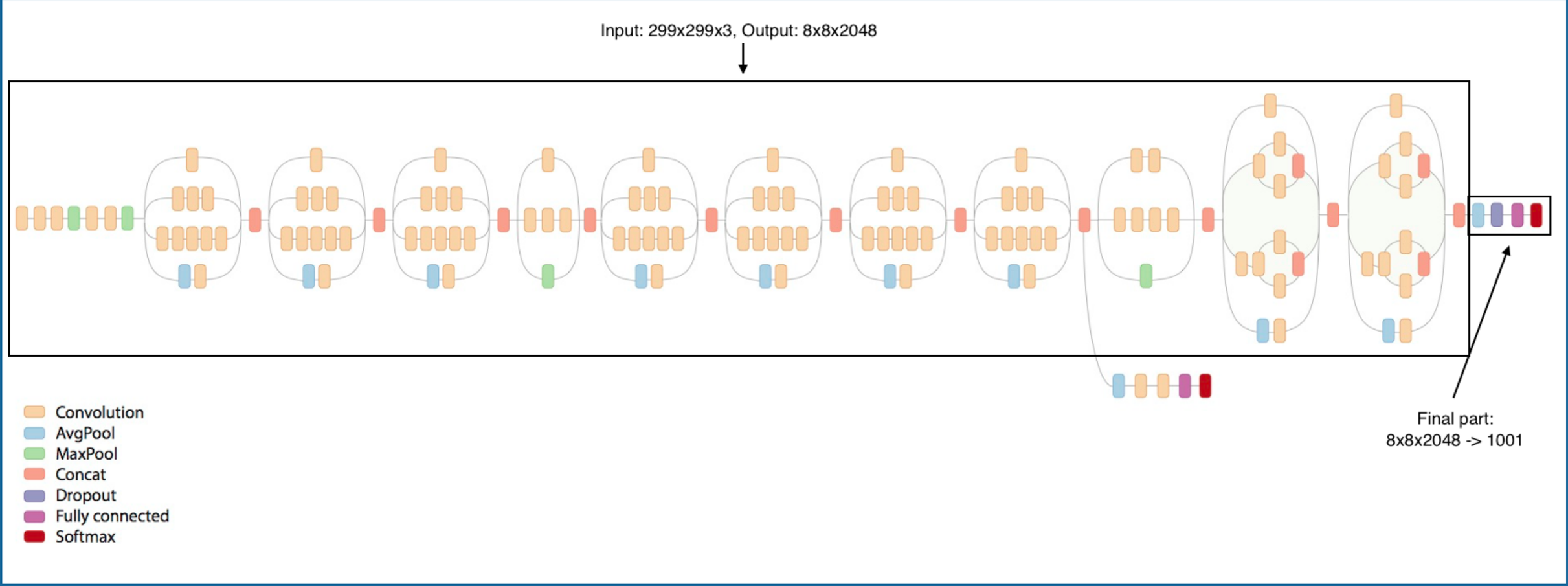
COREML

SUBTITLE

- Many different kinds of machine learning algorithms
  - Try to estimate probability of being in a state
  - Move through additional layers to refine
- Prediction and probability (confidence)



COREML

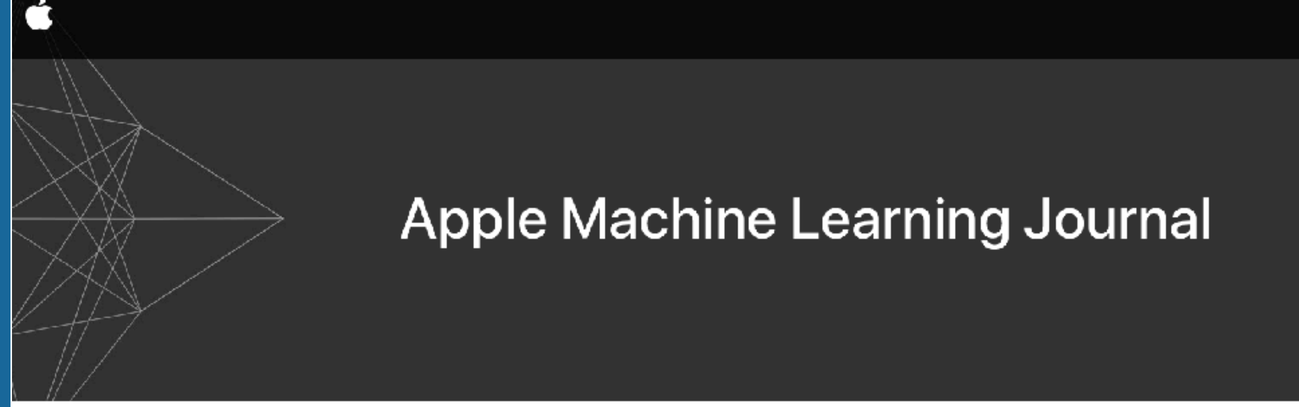


- Nueral network for image detection using Tensor Flow

COREML

SUBTITLE

- Apple publishes some of their machine learning research at https://machinelearning.apple.com



Personalized Hey Siri

Vol. 1, Issue 9 • April 2018
by Siri Team

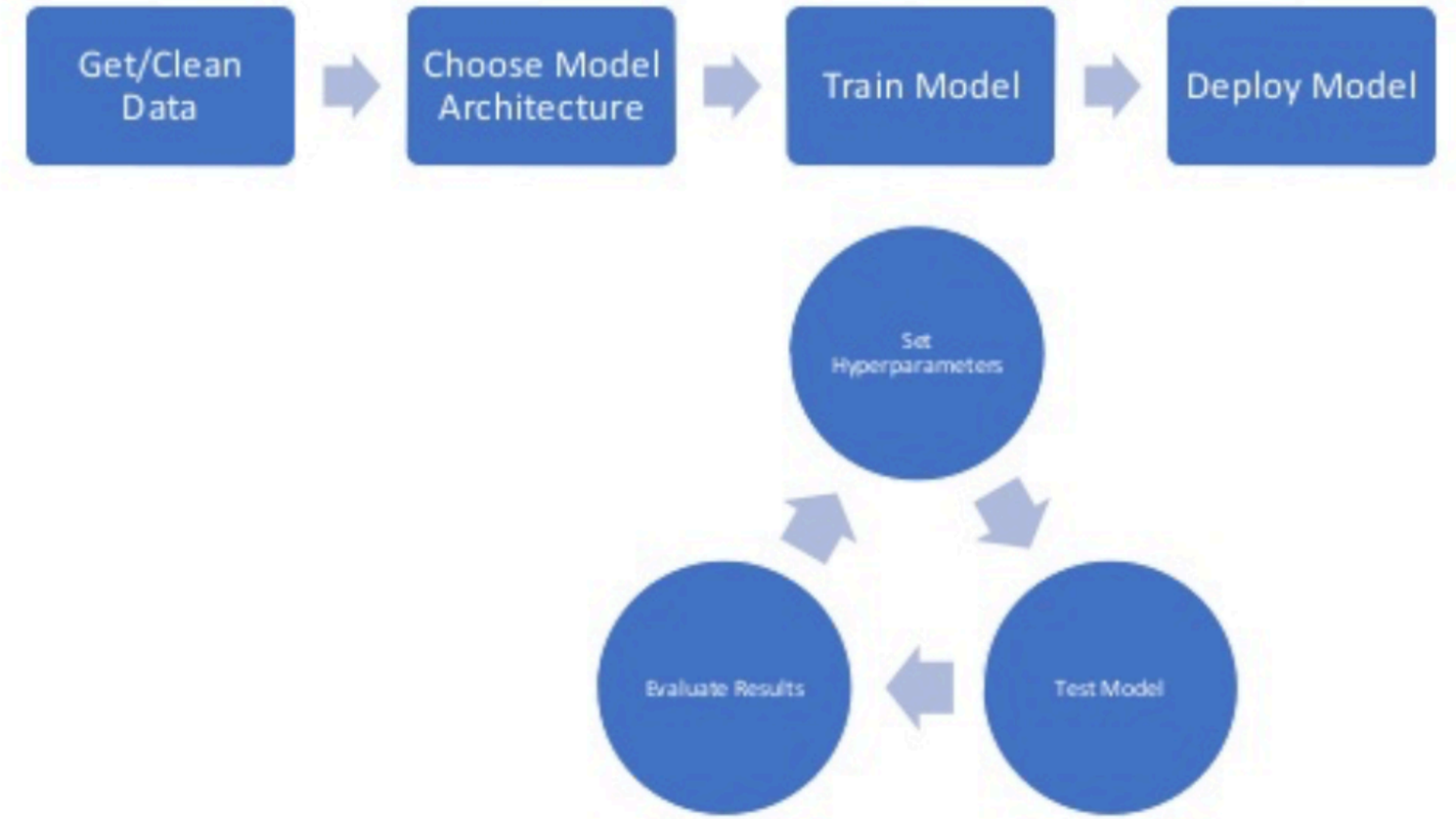
Apple introduced the “Hey Siri” feature with the iPhone 6 (iOS 8). This feature allows users to invoke Siri without having to press the home button. When a user says, “Hey Siri, how is the weather today?” the phone wakes up upon hearing “Hey Siri” and processes the rest of the utterance as a Siri request.

The feature’s ability to listen continuously for the “Hey Siri” trigger phrase lets users access Siri in situations where their hands might be otherwise occupied, such as while driving or cooking, as well as in situations when their respective devices are not within arm’s reach. Imagine a scenario where a user is asking his or her iPhone 6 on the kitchen counter to set a timer while putting a turkey into the oven.

Read the article ›

COREML

- ML workflow
- CoreML comes in after you have trained your model



COREML

SUBTITLE

- Apple provides popular modes that are drag-and-drop ready for using with CoreML
  - Object detection
  - Handwriting recognition
- Tools for converting your own models to work with Apple Frameworks

Working with Models

Build your apps with the ready-to-use Core ML models below, or use Core ML Tools to easily convert models into the Core ML format.

Models

MobileNet

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

Detects 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 (17.1 MB)

Model Converters

Core ML Tools

Core ML Tools is a python package that can be used to convert models from machine learning toolboxes into the Core ML format.

Get Core ML Tools ↗

Apache MXNet

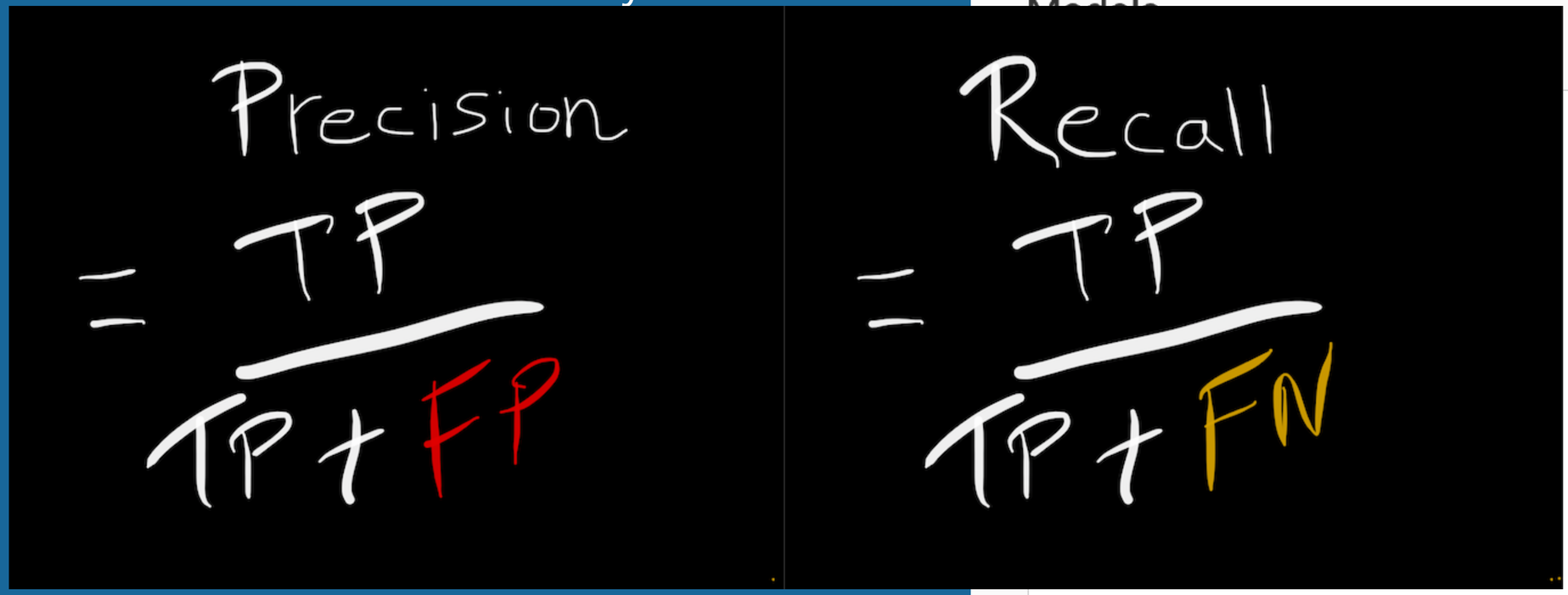
MXNet helps you train machine learning models and convert them into the Core ML format.

Get MXNet model converter ↗

COREML

SUBTITLE

- Some models are already



Working with Models

Build your apps with the ready-to-use Core ML models below, or use Core ML Tools to easily convert models into the Core ML format.

Model Converters

Core ML Tools

Core ML Tools is a python package that can be used to convert models from machine learning toolboxes into the Core ML format.

Get Core ML Tools ↗

Apache MXNet

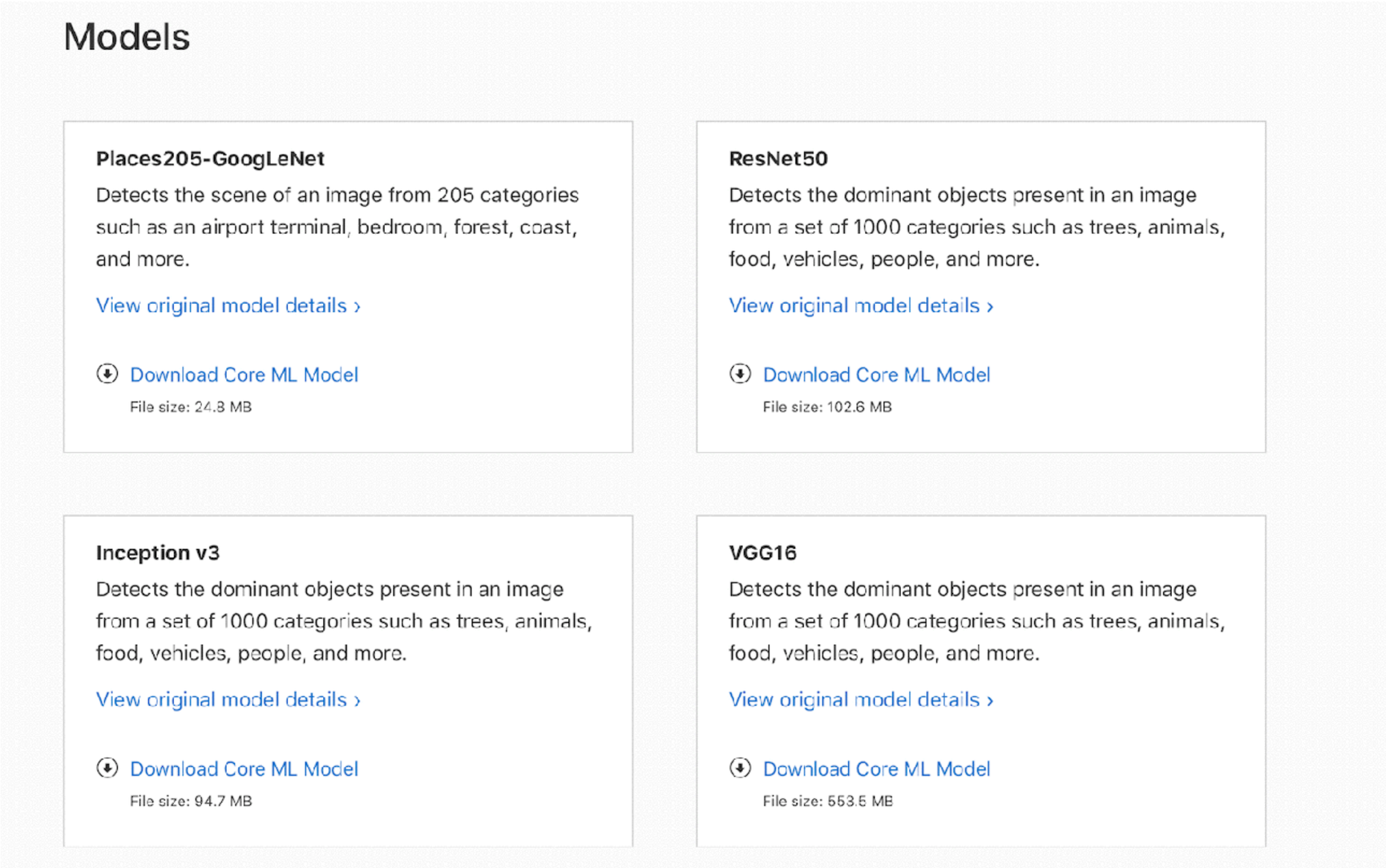
MXNet helps you train machine learning models and convert them into the Core ML format.

Get MXNet model converter ↗

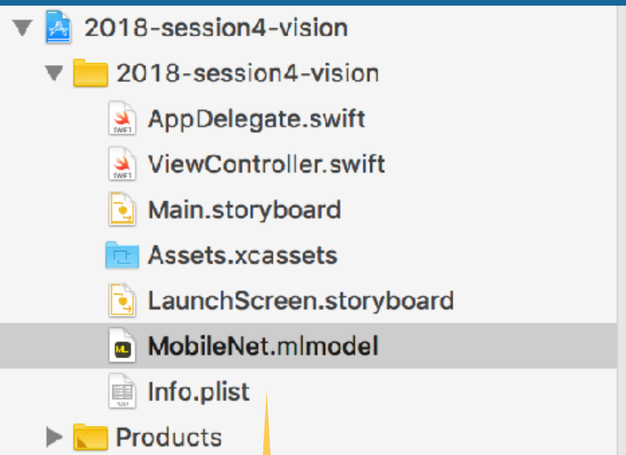
USING CORMEL FOR OBJECT DETECTION

OBJECT DETECTION

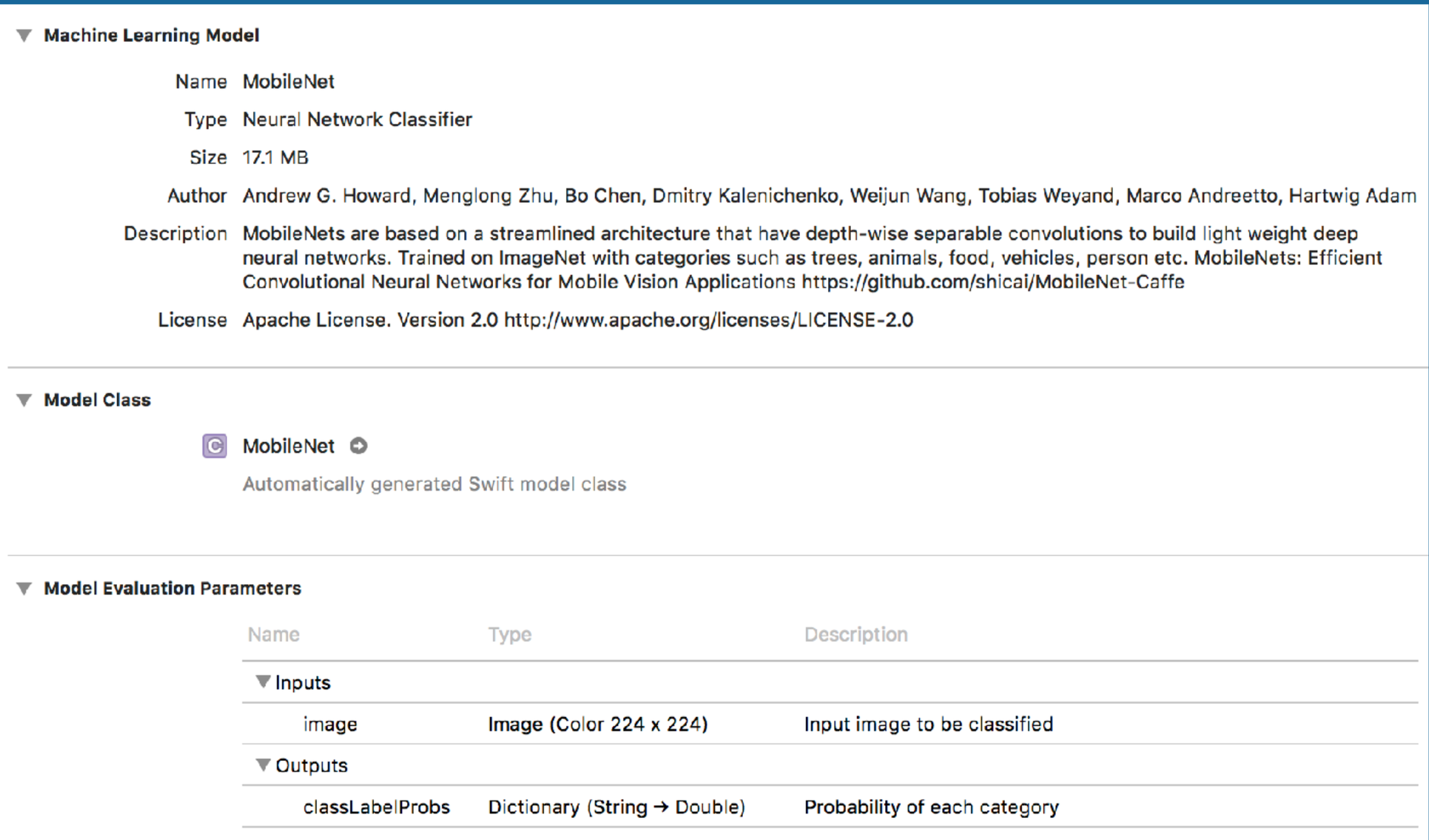
- Choose a model
- Factors
  - Size
  - Training data
  - Methodology



OBJECT DETECTION



Recognized by Xcode



OBJECT DETECTION

Model Class

MobileNet

Automatically generated Swift model class

Creates a class

Model Evaluation Parameters

Name	Type	Description
Inputs		
image	Image (Color 224 x 224)	Input image to be classified
Outputs		
classLabelProbs	Dictionary (String → Double)	Probability of each category
classLabel	String	Most likely image category

OBJECT DETECTION

- Choose an image from and prepare it for CoreML

```
override func viewDidLoad() {
  super.viewDidLoad()

  // The images in asset catalog
  let images = ["dog","swim","cat","pro","apple"]

  // Grab a random one
  let randomIndex = Int(arc4random_uniform(UInt32(images.count)))
  let image = UIImage(named: images[randomIndex] )
  imageView.image = image

  // Convert to CIImage (CoreImage)
  guard let ciImage = CIImage(image: image!) else {
    fatalError("Not able to convert UIImage to CIImage")
  }

  detectObjects(image: ciImage)
}
```

OBJECT DETECTION

- Set up vision request

```
func detectObjects(image: CIImage) {

  // Load the ML model through its generated class
  guard let model = try? VNCoreMLModel(for: MobileNet().model) else {
    fatalError("Can't load the ML model")
  }

  // Create a Vision request with completion handler
  let request = VNCoreMLRequest(model: model) { request, error in

    guard let results = request.results as? [VNClassificationObservation] else {
        fatalError("Unexpected result type from VNCoreMLRequest")
    }

    // Put the top 5 into a string to show on the screen
    var tags = ""
    for result in results[0...5] {
      tags += "🤔 \(result.identifier) (\(Int(result.confidence * 100))%)\n"
    }

    // Update UI on main queue
    DispatchQueue.main.async {
       self.textView.text = tags
    }
  }

  // Run the Core ML model classifier on global dispatch queue
  let handler = VNImageRequestHandler(ciImage: image)
  DispatchQueue.global(qos: .userInteractive).async {
    do {
      try handler.perform([request])
    } catch {
      print(error)
    }
  }
}
```

OBJECT DETECTION

```swift
// Load the ML model through its generated class
guard let model = try? VNCoreMLModel(for: MobileNet().model) else {
  fatalError("Can't load the ML model")
}

// Create a Vision request with completion handler
let request = VNCoreMLRequest(model: model) { request, error in

  guard let results = request.results as? [VNClassificationObservation] else {
      fatalError("Unexpected result type from VNCoreMLRequest")
  }

  // Put the top 5 into a string to show on the screen
  var tags = ""
  for result in results[0...5] {
    tags += "🤔 \(result.identifier) (\(Int(result.confidence * 100))%)\n"
  }

  // Update UI on main queue
  DispatchQueue.main.async {
```

Load model by name

OBJECT DETECTION

```
// Create a Vision request with completion handler
let request = VNCoreMLRequest(model: model) { request, error in

  guard let results = request.results as? [VNClassificationObservation] else {
      fatalError("Unexpected result type from VNCoreMLRequest")
  }

  // Put the top 5 into a string to show on the screen
  var tags = ""
  for result in results[0...5] {
    tags += "🤔 \(result.identifier) (\(Int(result.confidence * 100))%)\n"
  }

  // Update UI on main queue
  DispatchQueue.main.async {
     self.textView.text = tags
  }
}
```

Create the request with completion handler

OBJECT DETECTION

```
        // Update UI on main queue
        DispatchQueue.main.async {
            self.textView.text = tags
        }
    }

    // Run the Core ML model classifier on global dispatch queue
    let handler = VNImageRequestHandler(ciImage: image)
    DispatchQueue.global(qos: .userInteractive).async {
        do {
            try handler.perform([request])
        } catch {
            print(error)
        }
    }
}
```

Start it off

NUMBER RECOGNITION

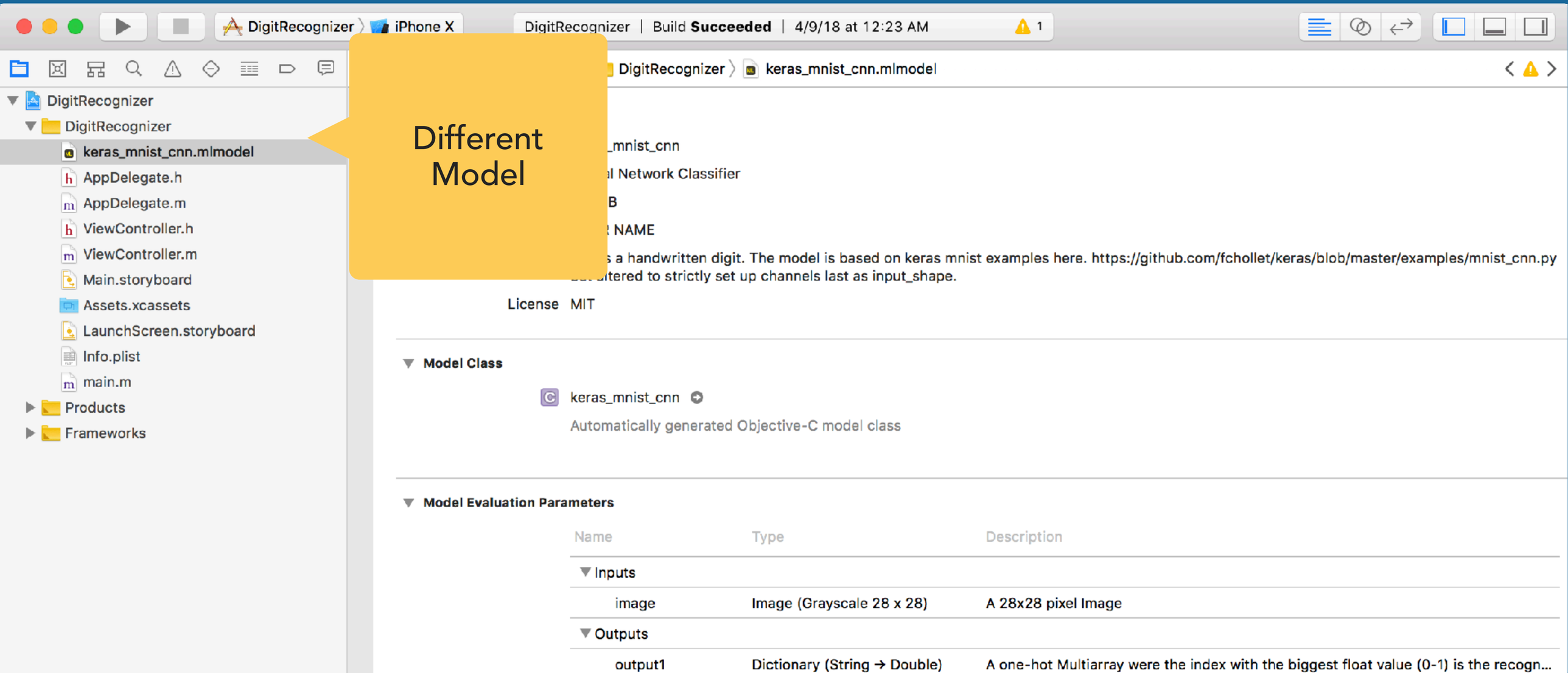
NUMBER DETECTION

- MNIST database of handwritten digits
- 60,000 training images
- 10,0000 testing images

NUMBER DETECTION

- Recognize a hand drawn image

NUMBER DETECTION



NUMBER DETECTION

Create Model

```
MLModel *ml_model = [[[keras_mnist_cnn alloc] init] model];
VNCoreMLModel *vnc_core_ml_model = [VNCoreMLModel modelForMLModel: ml_model error:nil];

VNCoreMLRequest *request = [[VNCoreMLRequest alloc] initWithModel: vnc_core_ml_model completionHandler:
    (VNRequestCompletionHandler) ^(VNRequest *request, NSError *error){
    NSArray *results = [request.results copy];

    VNClassificationObservation *res = ((VNClassificationObservation *)(results[0]));

    self.resultLabel.text = [NSString stringWithFormat: @"Digit may be: %@", res.identifier];
}];

NSDictionary *options_dict = [[NSDictionary alloc] init];
NSArray *request_array = @[request];

VNImageRequestHandler *handler = [[VNImageRequestHandler alloc] initWithCIImage:self.imageToDetect options:options_dict];
dispatch_queue_t myCustomQueue;
myCustomQueue = dispatch_queue_create("com.lukau.VNImageRequestHandlerQueue", NULL);

self.resultLabel.text = @"Predicting...";
dispatch_sync(myCustomQueue, ^{
    [handler performRequests:request_array error:nil];

});
```

NUMBER DETECTION

```
MLModel *ml_model = [[[keras_mnist_cnn alloc] init] model];
VNCoreMLModel *vnc_core_ml_model = [VNCoreMLModel modelForMLModel: ml_model error:nil];

VNCoreMLRequest *request = [[VNCoreMLRequest alloc] initWithModel: vnc_core_ml_model completionHandler:
    (VNRequestCompletionHandler) ^(VNRequest *request, NSError *error){
    NSArray *results = [request.results copy];

    VNClassificationObservation *res = ((VNClassificationObservation *)(results[0]));

    self.resultLabel.text = [NSString stringWithFormat: @"Digit may be: %@", res.identifier];
}];

NSDictionary *options_dict = [[NSDictionary alloc] init];
NSArray *request_array = @[request];

VNImageRequestHandler *handler = [[VNImageRequestHandler alloc] initWithCIImage:self.imageToDetect options:options_dict];
dispatch_queue_t myCustomQueue;
myCustomQueue = dispatch_queue_create("com.lukau.VNImageRequestHandlerQueue", NULL);

self.resultLabel.text = @"Predicting...";
dispatch_sync(myCustomQueue, ^{
    [handler performRequests:request_array error:nil];

});
```

Setup Vision Request

NUMBER DETECTION

```
MLModel *ml_model = [[[keras_mnist_cnn alloc] init] model];
VNCoreMLModel *vnc_core_ml_model = [VNCoreMLModel modelForMLModel: ml_model error:nil];

VNCoreMLRequest *request = [[VNCoreMLRequest alloc] initWithModel: vnc_core_ml_model completionHandler:
    (VNRequestCompletionHandler) ^(VNRequest *request, NSError *error){
    NSArray *results = [request.results copy];

    VNClassificationObservation *res = ((VNClassificationObservation *)(results[0]));

    self.resultLabel.text = [NSString stringWithFormat: @"Digit may be: %@", res.identifier];
}];

NSDictionary *options_dict = [[NSDictionary alloc] init];
NSArray *request_array = @[request];

VNImageRequestHandler *handler = [[VNImageRequestHandler alloc] initWithCIImage:self.imageToDetect options:options_dict];
dispatch_queue_t myCustomQueue;
myCustomQueue = dispatch_queue_create("com.lukau.VNImageRequestHandlerQueue", NULL);

self.resultLabel.text = @"Predicting...";
dispatch_sync(myCustomQueue, ^{
    [handler performRequests:request_array error:nil];

});
```

Start the Request

NUMBER DETECTION

- Demo

NUMBER DETECTION

Model trained on 28x28 pixels

```
// Original image
self.imageToDetect = [[CIImage alloc]initWithImage:self.drawingCanvas.image];

// You may get better performance by rescaling to match the database image size
UIImage *scaledCanvasImage = [self imageWithImage:self.drawingCanvas.image scaledToSize:CGSizeMake(28, 28)];
self.imageToDetect = [[CIImage alloc]initWithImage:scaledCanvasImage];
```

- You may get better performance by rescaling input size to training size

NUMBER DETECTION

Handwritten digit recognition with MNIST on iOS with Keras

Or: How to use a Keras deep learning neural network in iOS?

THE UNIVERSITY OF CHICAGO

ADVANCED iOS APPLICATION DEVELOPMENT

MPCS 51032 • SPRING 2020 • SESSION 4