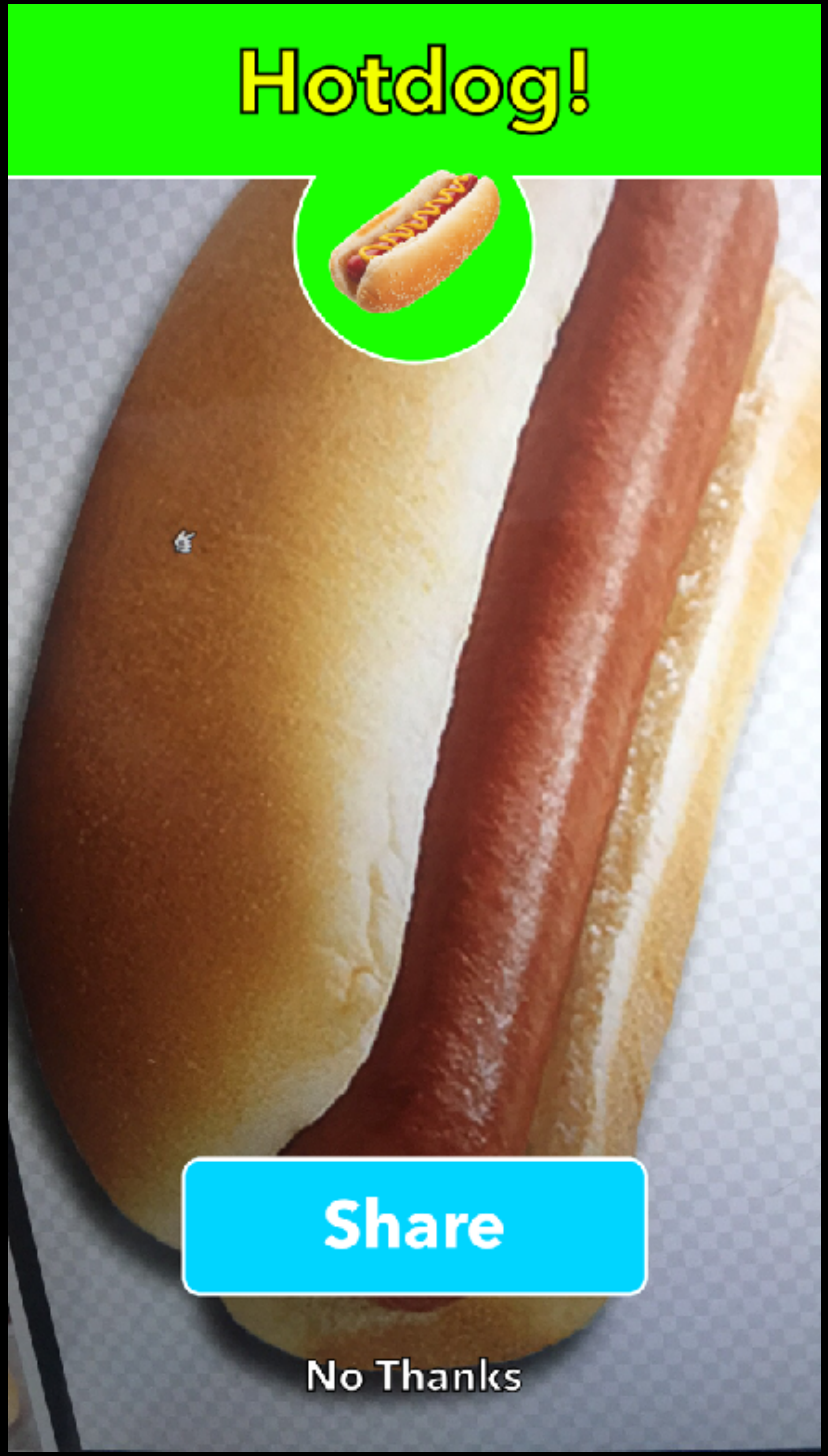
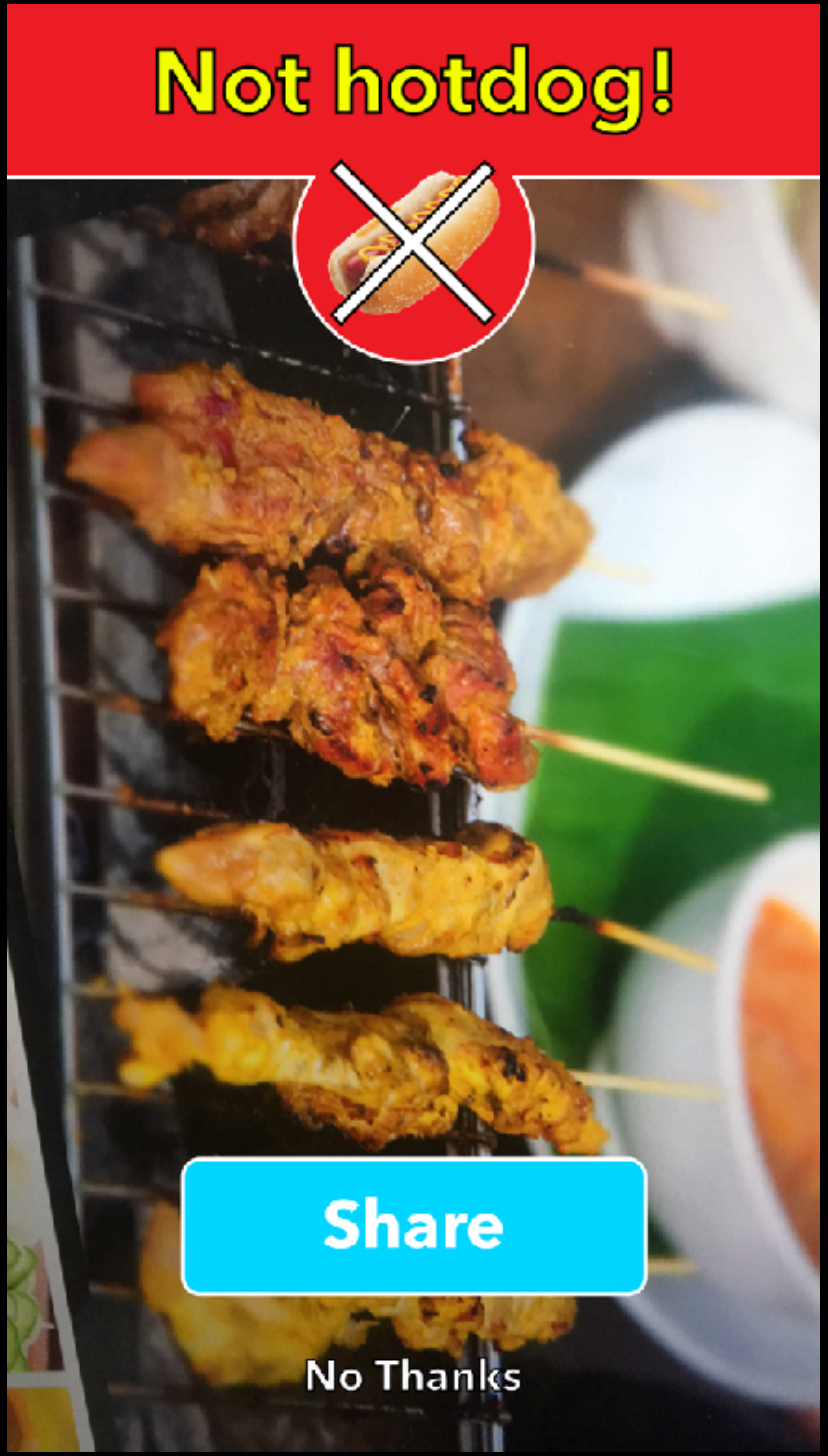
DL Dev Course: Week 08
Mobile

RED DRAGON AI

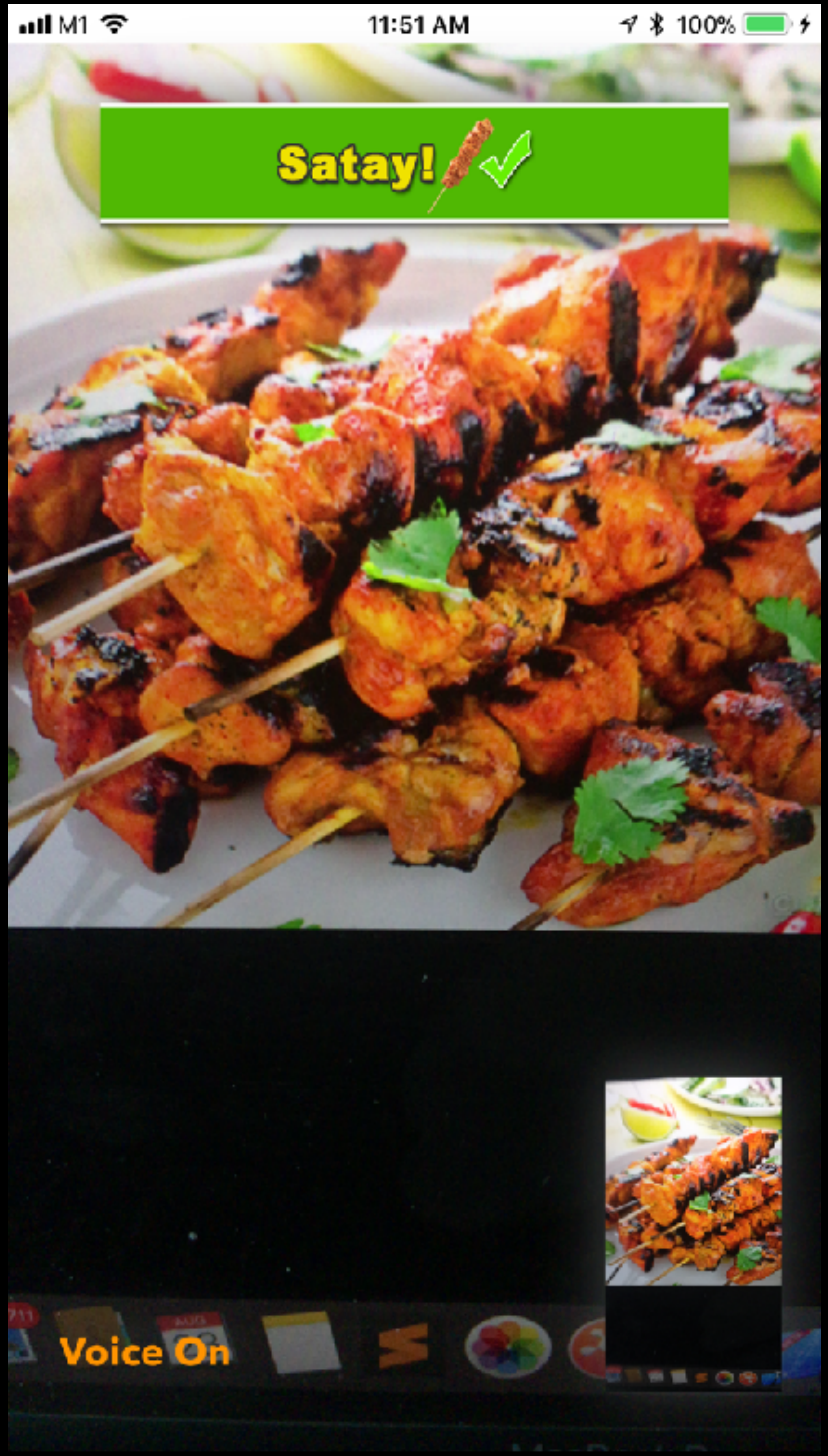
Silicon Valley

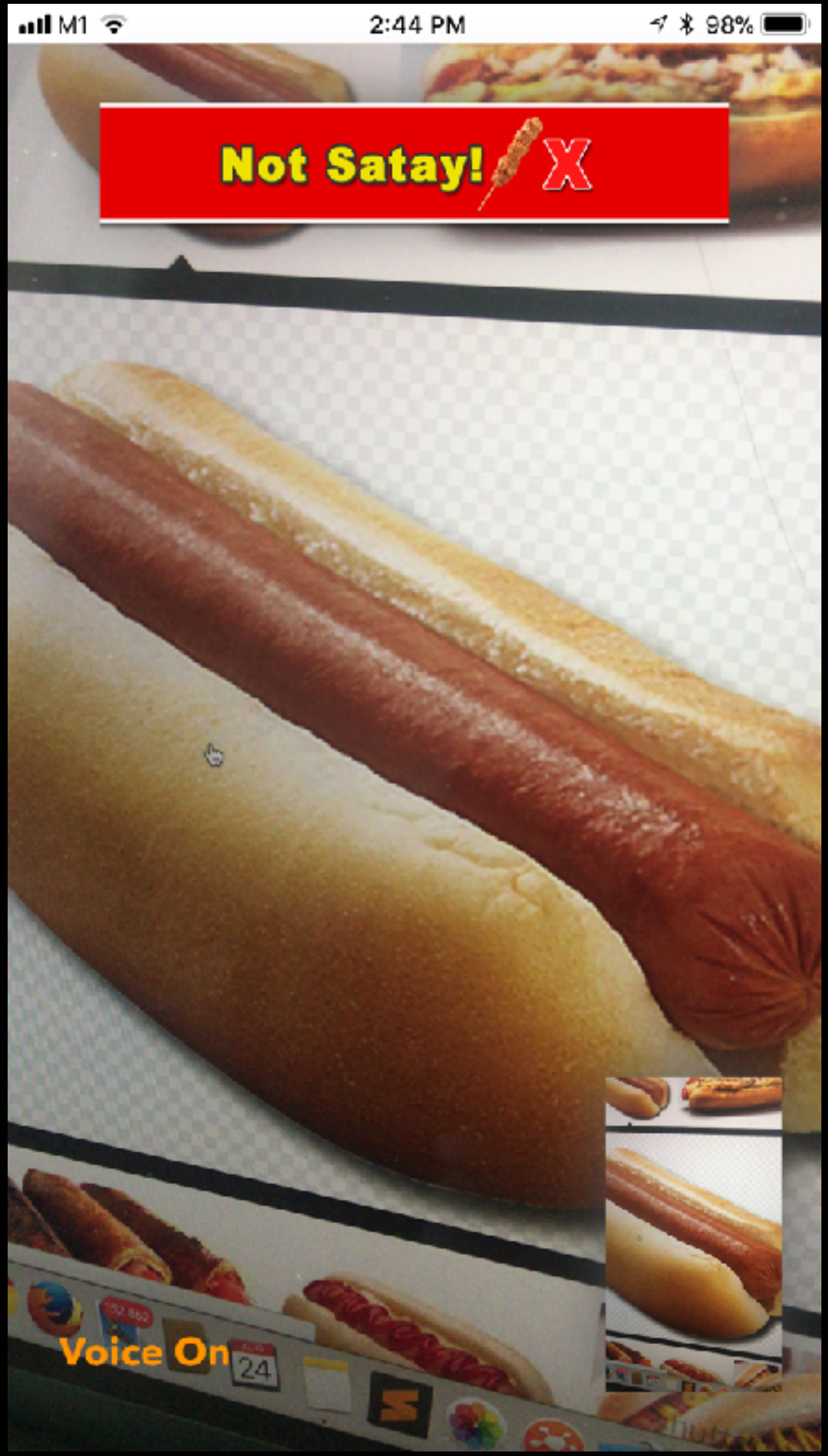




RED DRAGON AI

The goal





RED DRAGON AI

Topics

- 1. Mobile Models
- 2. Converting Keras Model to CoreML
- 3. iOS using CoreML
- 4 TF Lite

Data set

- 2k Satay Images
- Challenge is what is a picture of “Not Satay”
- 2.5k of other images (mostly other food)
- Trained with basic image augmentation

Image Augmentation

- Keras.ImageDataGenerator
- Rotation
- Rescale
- Zoom
- Image flipping

The Model

- Smaller the model the better
- Its about the number of parameters in your model
- Choosing model architectures
  - Squeezenet
  - Mobilenet
  - Inceptionv3
  - Resnet
  - VGG

RED DRAGON AI

Model Size

Model	Keras Size	Parameters	CoreML Size	Apple’s Size
VGG16	110.3 mb	14714688	84.6 mb	553.5 mb
InceptionV3	192.9 mb	34910754	192.9 mb	94.7 mb
Resnet	283.4 mb	23591816	94.4 mb	102.6 mb
Squeezenet	5.6mb	685816	Not Working	5 mb
Mobilenet		4253864	Not Working	None

Notebook

Setting up CoreML Tools

- Python 2.7
- TensorFlow 1.1
- Keras 2.04+?
- pip install coremltools

Converting to CoreML

- Inputs
- Outputs, Classes
- is_bgr - RGB , BGR
- Color biases

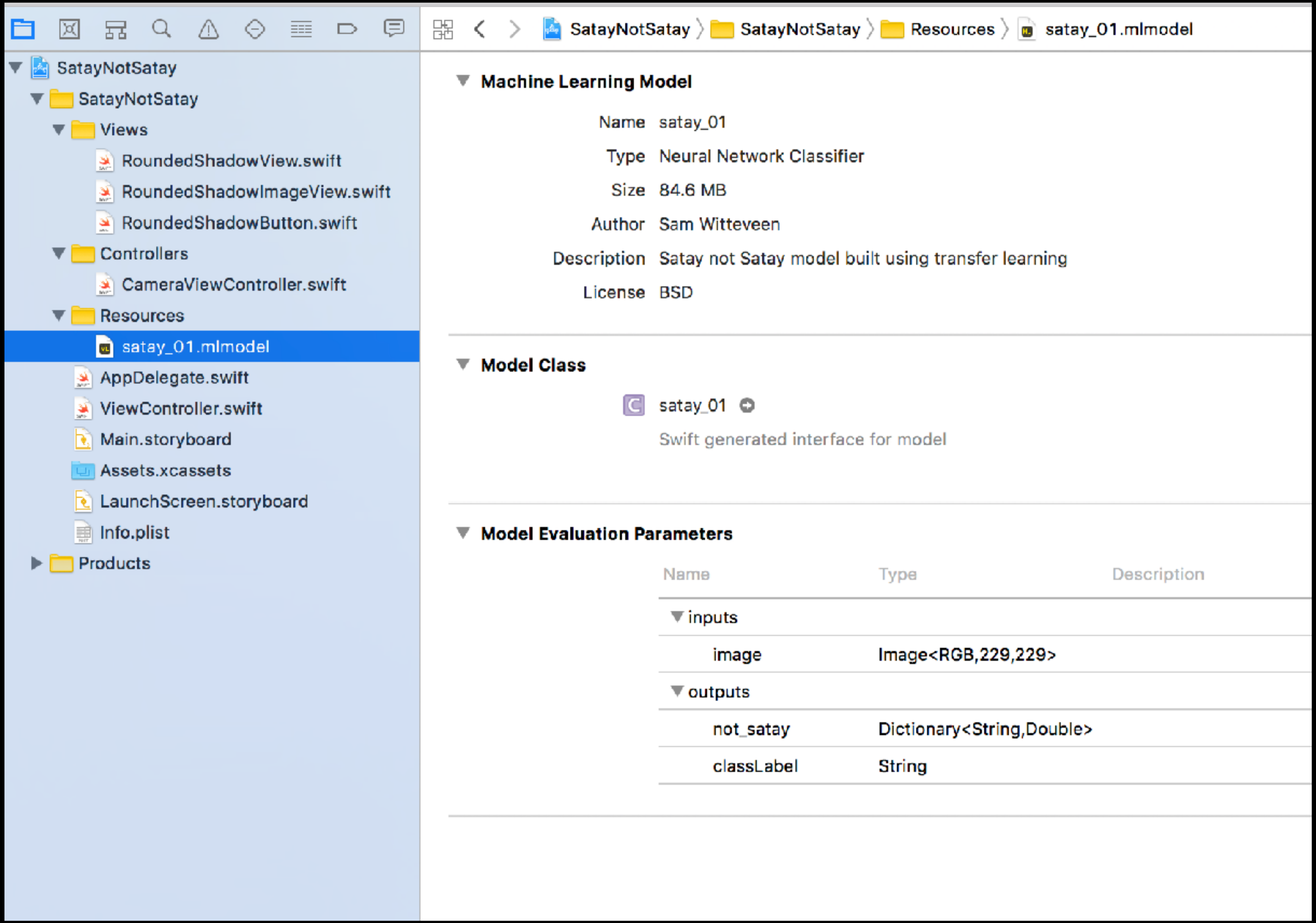
Notebook

RED DRAGON AI

iOS Xcode CoreML

RED DRAGON AI

Xcode



Machine Learning Model

Name satay_01

Type Neural Network Classifier

Size 84.6 MB

Author Sam Witteveen

Description Satay not Satay model built using transfer learning

License BSD

Model Class

satay_01

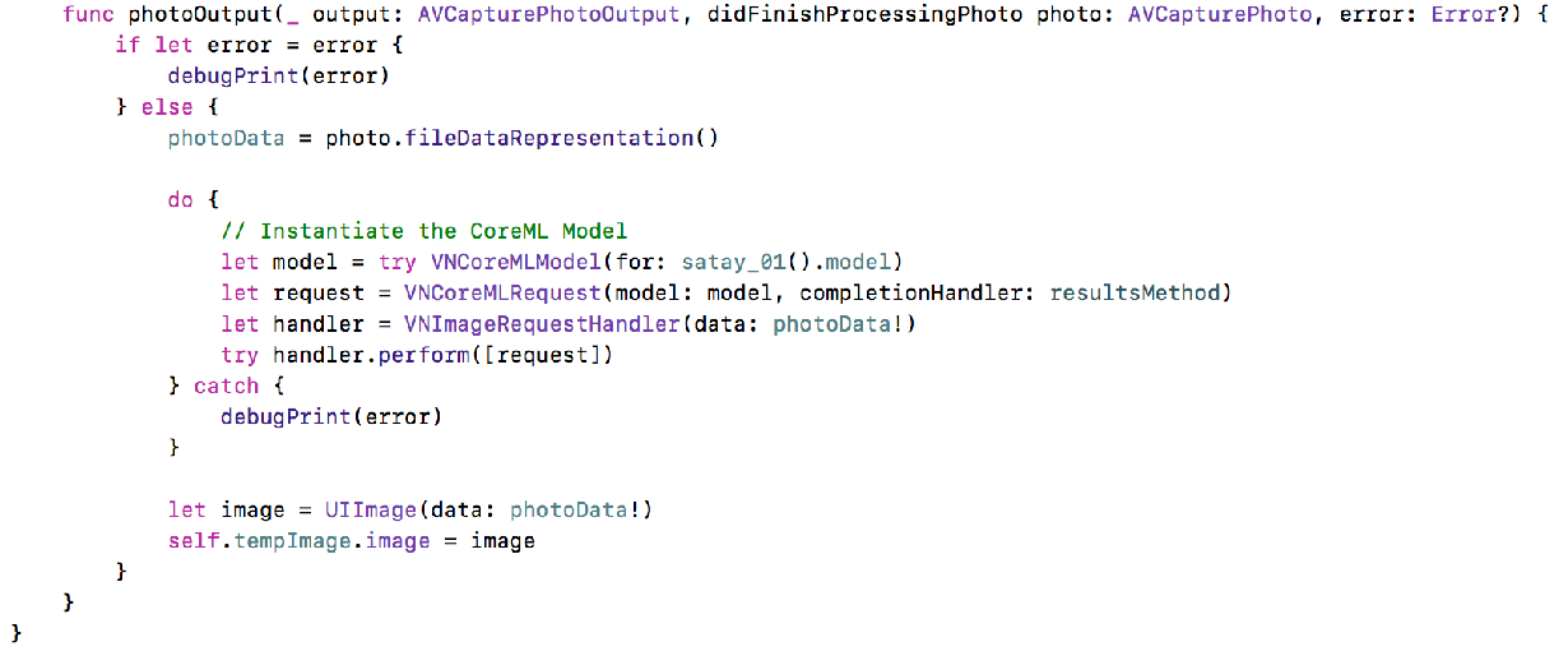
Swift generated interface for model

Model Evaluation Parameters

Name	Type	Description
▼ inputs		
image	Image<RGB,229,229>	
▼ outputs		
not_satay	Dictionary<String,Double>	
classLabel	String	

RED DRAGON AI

Xcode



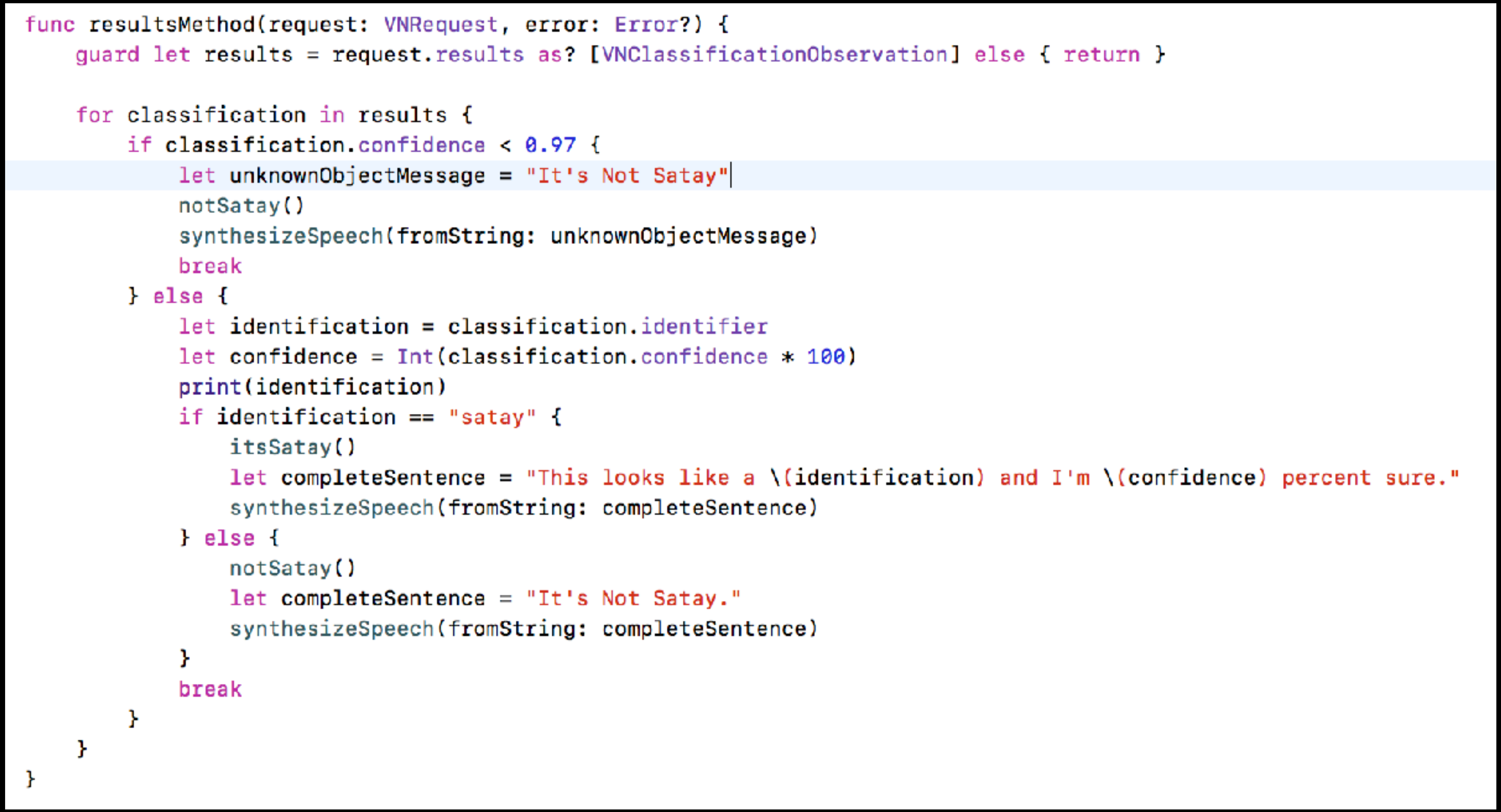
```
    func photoOutput(_ output: AVCapturePhotoOutput, didFinishProcessingPhoto photo: AVCapturePhoto, error: Error?) {
        if let error = error {
            debugPrint(error)
        } else {
            photoData = photo.fileDataRepresentation()

            do {
                // Instantiate the CoreML Model
                let model = try VNCoreMLModel(for: satay_01().model)
                let request = VNCoreMLRequest(model: model, completionHandler: resultsMethod)
                let handler = VNImageRequestHandler(data: photoData!)
                try handler.perform([request])
            } catch {
                debugPrint(error)
            }

            let image = UIImage(data: photoData!)
            self.tempImage.image = image
        }
    }
}
```

RED DRAGON AI

Xcode



```
func resultsMethod(request: VNRequest, error: Error?) {
    guard let results = request.results as? [VNClassificationObservation] else { return }

    for classification in results {
        if classification.confidence < 0.97 {
            let unknownObjectMessage = "It's Not Satay"
            notSatay()
            synthesizeSpeech(fromString: unknownObjectMessage)
            break
        } else {
            let identification = classification.identifier
            let confidence = Int(classification.confidence * 100)
            print(identification)
            if identification == "satay" {
                itsSatay()
                let completeSentence = "This looks like a \(identification) and I'm \(confidence) percent sure."
                synthesizeSpeech(fromString: completeSentence)
            } else {
                notSatay()
                let completeSentence = "It's Not Satay."
                synthesizeSpeech(fromString: completeSentence)
            }
            break
        }
    }
}
```

Example/Video

RED DRAGON AI

Conclusions

- CoreML makes getting a model on iOS very quick
- Still rather limited
- Does allow for using Metal2 to speed up the model
- TensorFlow native support maybe coming

Resources

- https://github.com/samwit/SatayNotSatay - Full iOS Source
- https://github.com/samwit/TensorFlowTalks - Keras & CoreMLToos notebooks
- https://medium.com/@timanglade/how-hbos-silicon-valley-built-not-hotdog-with-mobile-tensorflow-keras-react-native-ef03260747f3