Vision with Core ML
Powerful Computer Vision made easy
Session 717

Frank Doepke, Flat Pixel Enthusiast
Custom image classification
Object recognition
Vision fundamentals
Custom Image classification

Object recognition

Vision fundamentals
Custom Image classification

Object recognition

Vision fundamentals
Custom Image Classification
Worth more than a thousand pictures
The Storyline

Create an app helping shoppers identify items
Create an app helping shoppers identify items
• Train a custom classifier
The Storyline

Create an app helping shoppers identify items

• Train a custom classifier
• Build an iOS app
The Storyline

Create an app helping shoppers identify items
- Train a custom classifier
- Build an iOS app
- Keep an eye on pitfalls
Our Training Regimen

Train using Create ML
Our Training Regimen
Train using Create ML

Take pictures
Our Training Regimen
Train using Create ML

Take pictures

Sort into folders—the folder names are used as labels
Our Training Regimen
Train using Create ML

Take pictures

Sort into folders—the folder names are used as labels

• How much data do I need
Our Training Regimen
Train using Create ML

Take pictures

Sort into folders—the folder names are used as labels
• How much data do I need
  - Minimum of 10 per category but more is better
Our Training Regimen
Train using Create ML

Take pictures

Sort into folders—the folder names are used as labels

• How much data do I need
  - Minimum of 10 per category but more is better
  - Highly imbalanced datasets are hard to train
Our Training Regimen
Train using Create ML

Take pictures

Sort into folders—the folder names are used as labels

• How much data do I need
  - Minimum of 10 per category but more is better
  - Highly imbalanced datasets are hard to train

• Augmentation adds some robustness on top of it but doesn’t replace variety
Our Training Regimen
Train using Create ML

Take pictures

Sort into folders—the folder names are used as labels

• How much data do I need
  - Minimum of 10 per category but more is better
  - Highly imbalanced datasets are hard to train

• Augmentation adds some robustness on top of it but doesn’t replace variety
Under the Hood
Under the Hood

Transfer Learning
Under the Hood

Transfer Learning

• Starts with a pre-trained model
  - This is the heavy load
Transfer Learning

• Starts with a pre-trained model
  - This is the heavy load
• Use that model as a Feature Extractor
Under the Hood

Transfer Learning
• Starts with a pre-trained model
  - This is the heavy load
• Use that model as a Feature Extractor
• Train a last layer as a classifier with your labeled data
Vision FeaturePrint.Scene
Vision Frameworks FeaturePrint for Image Classification
Vision FeaturePrint.Scene

Vision Frameworks FeaturePrint for Image Classification
• Available through ImageClassifier training in Create ML
Vision FeaturePrint.Scene

Vision Frameworks FeaturePrint for Image Classification
• Available through ImageClassifier training in Create ML
• Trained on a very large dataset
Vision FeaturePrint.Scene

Vision Frameworks FeaturePrint for Image Classification
• Available through ImageClassifier training in Create ML
• Trained on a very large dataset
• Capable of predicting over 1000 categories
Vision Frameworks FeaturePrint for Image Classification

- Available through ImageClassifier training in Create ML
- Trained on a very large dataset
- Capable of predicting over 1000 categories
- Powers user facing features in Photos
Vision FeaturePrint.Scene

Vision Frameworks FeaturePrint for Image Classification
• Available through ImageClassifier training in Create ML
• Trained on a very large dataset
• Capable of predicting over 1000 categories
• Powers user facing features in Photos
• Continuous improvement
Vision Frameworks FeaturePrint for Image Classification

- Available through ImageClassifier training in Create ML
- Trained on a very large dataset
- Capable of predicting over 1000 categories
- Powers user facing features in Photos
- Continuous improvement
  - You might want to retrain in the future
Vision FeaturePrint.Scene
Vision FeaturePrint.Scene

Already on device
Vision FeaturePrint.Scene

Already on device

• Smaller disk footprint for your custom model
Vision FeaturePrint.Scene

Already on device

• Smaller disk footprint for your custom model
Vision FeaturePrint.Scene

Already on device
- Smaller disk footprint for your custom model

98 MB
Vision FeaturePrint.Scene

Already on device

- Smaller disk footprint for your custom model

![Bar chart showing 98 MB for ResNet and 5 MB for SqueezeNet]
Vision FeaturePrint.Scene

Already on device

- Smaller disk footprint for your custom model

![Bar chart showing disk footprint for Resnet, Squeezenet, and Vision]

- Resnet: 98 MB
- Squeezenet: 5 MB
- Vision: <1MB
Vision FeaturePrint.Scene

Already on device
- Smaller disk footprint for your custom model

Optimized for Apple devices

Bar chart showing:
- Resnet: 98 MB
- Squeezenet: 5 MB
- Vision: < 1 MB
Vision FeaturePrint.Scene
Vision FeaturePrint.Scene

Training in Create ML

Labeled Images → Vision FeaturePrint.Scene → Trained Classifier
Vision FeaturePrint.Scene

Labeled Images → Training in Create ML → Trained Classifier → Core ML Model

JPEG
Refining the App
Only classify when needed
Refining the App
Only classify when needed

Don’t run expensive tasks when not needed
Refining the App
Only classify when needed

Don’t run expensive tasks when not needed

Am I holding still?
Refining the App
Only classify when needed

Don’t run expensive tasks when not needed

Am I holding still?

Using registration
• Cheap and fast
• Camera holds still
• Subject is not moving
Refining the App

Only classify when needed

Don’t run expensive tasks when not needed

Am I holding still?

Using registration
- Cheap and fast
- Camera holds still
- Subject is not moving

VNTranslationalImageRegistrationRequest
Refining the App

Only classify when needed

Don’t run expensive tasks when not needed

Am I holding still?

Using registration
• Cheap and fast
• Camera holds still
• Subject is not moving

VNTranslationalImageRegistrationRequest
Refining the App
Always have a backup plan
Refining the App
Always have a backup plan

Classifications can be wrong
Refining the App
Always have a backup plan

Classifications can be wrong

Even when confidence is high > plan for it
Refining the App

Always have a backup plan

Classifications can be wrong

Even when confidence is high > plan for it

Alternative identification

• Barcode reading
Demo
Build the RobotShop
Recap

Using Registration for Scene Stability
Recap

Using Registration for Scene Stability

Use the `VNSequenceRequestHandler` with `VNTranslationalImageRegistrationRequest`
Recap

Using Registration for Scene Stability

Use the `VNSequenceRequestHandler` with `VNTranslationalImageRegistrationRequest`

Compare against previous frame:

```swift
sequenceRequestHandler.perform([request], on: previousBuffer!)
```
Recap

Using Registration for Scene Stability

Use the `VNSequenceRequestHandler` with `VNTranslationalImageRegistrationRequest`.

Compare against previous frame:

```swift
sequenceRequestHandler.perform([request], on: previousBuffer!)
```

Registration is returned as pixels in the `alignmentObservation.alignmentTransform`.
Recap

Analyze only when scene is stable
Recap

Analyze only when scene is stable

Create an `VNImageRequestHandler` for the current frame and pass in the orientation
Recap

Analyze only when scene is stable

Create an `VNImageRequestHandler` for the current frame and pass in the orientation

Perform Barcode and Image Classification together

```swift
try imageRequestHandler.perform([barcodeDetection, imageClassification])
```
Recap

Manage your buffers
Recap

Manage your buffers

Some Vision requests can take longer
Recap

Manage your buffers

Some Vision requests can take longer

Perform longer task asynchronously
Recap

Manage your buffers

Some Vision requests can take longer

Perform longer task asynchronously

Do not queue up more buffers than the camera can provide

• We only operate with a one deep queue in this example
Recap
Recap

Why not use just Core ML?
Recap

Why not use just Core ML?

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Flexibility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>inputs</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>image</td>
<td>Image (Color 299 x 299)</td>
<td>299... x 299...</td>
<td>Input image to be classified</td>
</tr>
<tr>
<td><strong>outputs</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>classLabelProbs</td>
<td>Dictionary (String → Double)</td>
<td></td>
<td>Probability of each category</td>
</tr>
<tr>
<td>classLabel</td>
<td>String</td>
<td></td>
<td>Most likely image category</td>
</tr>
</tbody>
</table>
Why not use just Core ML?

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Flexibility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>image</td>
<td>Image (Color 299 x 299)</td>
<td>299... x 299...</td>
<td>Input image to be classified</td>
</tr>
<tr>
<td>classLabelProbs</td>
<td>Dictionary (String → Double)</td>
<td></td>
<td>Probability of each category</td>
</tr>
<tr>
<td>classLabel</td>
<td>String</td>
<td></td>
<td>Most likely image category</td>
</tr>
</tbody>
</table>
Recap

Why not use just Core ML?

<table>
<thead>
<tr>
<th>Model Evaluation Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td><strong>inputs</strong></td>
</tr>
<tr>
<td>image</td>
</tr>
<tr>
<td><strong>outputs</strong></td>
</tr>
<tr>
<td>classLabelProbs</td>
</tr>
<tr>
<td>classLabel</td>
</tr>
</tbody>
</table>

Vision does all the scaling and color conversion for you
Object Recognition
I spy with my little eye
What and How

YOLO (You Only Look Once)

Fast Object Detection and Classification

• Label and Bounding Box
• Finds multiple and different objects

Train for custom objects

• Training is more involved than ImageClassifier
What and How

YOLO (You Only Look Once)

Fast Object Detection and Classification
- Label and Bounding Box
- Finds multiple and different objects

Train for custom objects
- Training is more involved than ImageClassifier
Demo

Where is my breakfast
VNRecognizedObjectObservation

Result of a VNCoreMLModelRequest
VNRecognizedObjectObservation

Result of a VNCoreMLModelRequest

New observation subclass VNRecognizedObjectObservation
VNRecognizedObjectObservation

Result of a VNCoreMLModelRequest

New observation subclass VNRecognizedObjectObservation

YOLO based models made easy
let mlModel = try MLModel(contentsOf: modelURL)
let visionModel = try VNCoreMLModel(for: mlModel)
let objectRecognition = VNCoreMLRequest(model: visionModel,
                                        completionHandler: { (request, error) in
                                            guard let results = request.results else { return }

                                            for case let foundObject as VNRecognizedObjectObservation in results {
                                                let bestLabel = foundObject.labels.first! // Label with highest confidence
                                                let objectBounds = foundObject.boundingBox

                                                // Use the computed values.
                                                print(bestLabel.identifier, bestLabel.confidence, objectBounds)
                                            }
                                        })

}}
Tracking
Tracking

Tracking is faster and smoother than re-detection
Tracking

Tracking is faster and smoother than re-detection

• Use tracking to follow a detected object
Tracking

Tracking is faster and smoother than re-detection

• Use tracking to follow a detected object
• Tracking is a lighter algorithm
Tracking

Tracking is faster and smoother than re-detection

• Use tracking to follow a detected object
• Tracking is a lighter algorithm
• Applies temporal smoothing
Tracking

Tracking is faster and smoother than re-detection
• Use tracking to follow a detected object
• Tracking is a lighter algorithm
• Applies temporal smoothing
Vision Fundamentals
The tripod to computer vision
Not all algorithms are orientation agnostic
Image Orientation

Not all algorithms are orientation agnostic

Images are not always upright
  • EXIF orientation defines what is upright
  • When using a URL as input Vision reads the EXIF orientation from file
Image Orientation

Not all algorithms are orientation agnostic

Images are not always upright
• EXIF orientation defines what is upright
• When using a URL as input Vision reads the EXIF orientation from file

Live from a capture feed
Image Orientation

Not all algorithms are orientation agnostic

Images are not always upright
• EXIF orientation defines what is upright
• When using a URL as input Vision reads the EXIF orientation from file

Live from a capture feed
• Orientation has to be inferred from `UIDevice.current.orientation`
Image Orientation

Not all algorithms are orientation agnostic

Images are not always upright
• EXIF orientation defines what is upright
• When using a URL as input Vision reads the EXIF orientation from file

Live from a capture feed
• Orientation has to be inferred from `UIDevice.current.orientation`
• Needs to be mapped to a `CGImagePropertyOrientation`
Vision Coordinate System
Vision Coordinate System

Origin is at the lower-left corner
Vision Coordinate System

Origin is at the lower-left corner

All processing is in relation to the image in upright coordinates
Vision Coordinate System

Origin is at the lower-left corner

All processing is in relation to the image in upright coordinates

Normalized coordinates
Vision Coordinate System

Origin is at the lower-left corner

All processing is in relation to the image in upright coordinates

Normalized coordinates
• 0.0 to 1.0
Vision Coordinate System

Origin is at the lower-left corner

All processing is in relation to the image in upright coordinates

Normalized coordinates
• 0.0 to 1.0
• Landmarks are relative to the face rectangle
Vision Coordinate System

Origin is at the lower-left corner

All processing is in relation to the image in upright coordinates

Normalized coordinates

• 0.0 to 1.0

• Landmarks are relative to the face rectangle

• VNUtils.h provides conversion utils into image coordinates like VNImageRectForNormalizedRect
Confidence Score
Confidence Score

A lot of algorithm can express how certain they are about the results
Confidence Score

A lot of algorithms can express how certain they are about the results.

Confidence is expressed between 0.0 (low) and 1.0 (highest).
Confidence Score

A lot of algorithm can express how certain they are about the results.

Confidence is expressed between 0.0 (low) and 1.0 (highest).

The scale is not uniform across request types.
Classification Example
Classification Example
Classification Example

Classification:
- StepperMotor 1.000
- ServoMotor 0.000
- Beam 0.000
- Microcontroller 0.000
- StepperMotorDriver 0.000
Classification Example
Classification Example
Classification Example

Classification:
sandbar, sand bar 0.395
seashore, coast, seacoast, sea-coast 0.322
parachute, chute 0.118
airship, dirigible 0.067
swimming trunks, bathing trunks 0.012
Classification Example
Classification Example
Classification Example

Classification:
- ocarina, sweet potato 0.946
- harmonica, mouth organ, harp, mouth harp 0.020
- cellular telephone, cellular phone, mobile phone 0.008
- handkerchief, hankie, hanky, hankey 0.005
- flute, transverse flute 0.002
Classification Example
Classification Example

```swift
// // VisionObjectRecognitionViewController.swift
// // VisionCoreObjectRecognition-Swift
// // Created by Frank OneOne on 6/24/17.
// // Copyright © 2018 Apple. All rights reserved.
// //
// import UIKit
// import AVFoundation
// import Vision

class VisionObjectRecognitionViewController: ViewController {

private var detectionOverlay: CALayer = nil

private var requests = [VNRequest]()

@discardableResult
func setupVision() -> NSError? {
    // Setup Vision parts
    let error: NSError? = nil

    guard let modelURL = Bundle.main.url(forResource: "ObjectDetector", withExtension: "nnmodel") else {
        return NSError(domain: "VisionObjectRecognitionViewController", code: -1, userInfo: [NSLocalizedDescriptionKey: "Model file is missing"])
    }

do {
    let visionModel = try VNCoreMLModel(for: modelURL)
    let objectRecognition = VNCoreMLRequest(model: visionModel, completionHandler: {
        (request, error) in
        // perform all UI updates on the main queue
    })
    self.requestQueue = [objectRecognition]
}

  return error

```
Classification Example

Classification:
- website, internet site, site 0.994
- monitor 0.002
- screen, CRT screen 0.002
- desktop computer 0.000
- hand-held computer, hand-held microcomputer 0.000
Confidence Score Conclusions
Confidence Score Conclusions

Does 1.0 mean it is certainly correct?

- It fulfilled the criteria of the algorithm but our perception can differ
Confidence Score Conclusions

Does 1.0 mean it is certainly correct?

• It fulfilled the criteria of the algorithm but our perception can differ

Where the threshold is depends on the use case

• Labeling requires high confidence—observe how your classifier behaves
• Search might want to include lower confidence scores as they are probable
More Information

https://developer.apple.com/wwdc18/717