Creating Great Apps Using Core ML and ARKit

Brent Dimick, Core ML
Scott Gauthreaux, Core ML
Can Machine Learning help?
Subject: Chinese food

Would you like Mandarin food for lunch today?
Subject: Chinese food

Would you like Mandarin food for today?

Grocery list

Milk
Eggs
Orange juice
Understanding your model
Our Journey
Recognizing dice
Programmatically
Recognizing Dice
Recognizing Dice
Recognizing Dice
Recognizing Dice
Recognizing Dice
Using Machine Learning
Object Detector

A machine learning model that has been trained to detect objects in images.
// Handling object detection observations

func handleObservations(_ observations: [VNRecognizedObjectObservation]) {
    self.diceCount = observations.count
    for observation in observations {
        let objectBounds = self.bounds(for: observation)

        let shapeLayer = self.createRoundedRectLayerWithBounds(objectBounds)
        self.detectionOverlay.addSublayer(shapeLayer)
    }
}
// Handling object detection observations

func handleObservations(_ observations: [VNRecognizedObjectObservation]) {
    self.diceCount = observations.count
    for observation in observations {
        let objectBounds = self.bounds(for: observation)

        let shapeLayer = self.createRoundedRectLayerWithBounds(objectBounds)
        self.detectionOverlay.addSublayer(shapeLayer)
    }
}
func handleObservations(_: observations: [VNRecognizedObjectObservation]) {
    self.diceCount = observations.count
    for observation in observations {
        let objectBounds = self.bounds(for: observation)

        let shapeLayer = self.createRoundedRectLayerWithBounds(objectBounds)
        self.detectionOverlay.addSublayer(shapeLayer)
    }
}
func handleObservations(_ observations: [VNRecognizedObjectObservation]) {
    self.diceCount = observations.count
    for observation in observations {
        let objectBounds = self.bounds(for: observation)

        let shapeLayer = self.createRoundedRectLayerWithBounds(objectBounds)
        self.detectionOverlay.addSublayer(shapeLayer)
    }
}
// Handling object detection observations

func handleObservations(_ observations: [VNRecognizedObjectObservation]) {
    self.diceCount = observations.count
    for observation in observations {
        let objectBounds = self.bounds(for: observation)

        let shapeLayer = self.createRoundedRectLayerWithBounds(objectBounds)
        self.detectionOverlay.addSublayer(shapeLayer)
    }
}
One step further.
Recognizing the values on the dice.
Data Collection and Annotation
Focusing on Tops

1

5
Demo
When Does a Roll End?
When Does a Roll End?

What we observe
When Does a Roll End?

What we observe

Interpreting model output
/ Function to tell if a roll has ended with the given values

```swift
func hasRollEnded(newObservations: [VNRecognizedObjectObservation],
                    oldObservations: [VNRecognizedObjectObservation]) -> Bool {
    if oldObservations.count != newObservations.count {
        return false
    }
    var matches = 0
    for new in newObservations {
        for old in oldObservations {
            if new.labels.first?.identifier == old.labels.first?.identifier &&
                overlapRatio(rectA: old.boundingBox, rectB: new.boundingBox) > 0.85 {
                matches += 1
            }
        }
    }
    return matches == newObservations.count
}
```
// Function to tell if a roll has ended with the given values

func hasRollEnded(newObservations: [VNRecognizedObjectObservation],
                    oldObservations: [VNRecognizedObjectObservation]) -> Bool {
    if oldObservations.count != newObservations.count {
        return false
    }
    var matches = 0
    for new in newObservations {
        for old in oldObservations {
            if new.labels.first?.identifier == old.labels.first?.identifier &&
                overlapRatio(rectA: old.boundingBox, rectB: new.boundingBox) > 0.85 {
                matches += 1
            }
        }
    }
    return matches == newObservations.count
}
// Function to tell if a roll has ended with the given values

func hasRollEnded(newObservations: [VNRecognizedObjectObservation],
                   oldObservations: [VNRecognizedObjectObservation]) -> Bool {
    if oldObservations.count != newObservations.count {
        return false
    }
    var matches = 0
    for new in newObservations {
        for old in oldObservations {
            if new.labels.first?.identifier == old.labels.first?.identifier &&
                overlapRatio(rectA: old.boundingBox, rectB: new.boundingBox) > 0.85 {
                matches += 1
            }
        }
    }
    return matches == newObservations.count
}
// Function to tell if a roll has ended with the given values

func hasRollEnded(newObservations: [VNRecognizedObjectObservation],
                   oldObservations: [VNRecognizedObjectObservation]) -> Bool {
    if oldObservations.count != newObservations.count {
        return false
    }
    var matches = 0
    for new in newObservations {
        for old in oldObservations {
            if new.labels.first?.identifier == old.labels.first?.identifier &&
               overlapRatio(rectA: old.boundingBox, rectB: new.boundingBox) > 0.85 {
                matches += 1
            }
        }
    }
    return matches == newObservations.count
}
// Function to tell if a roll has ended with the given values

func hasRollEnded(newObservations: [VNRecognizedObjectObservation],
    oldObservations: [VNRecognizedObjectObservation]) -> Bool {
    if oldObservations.count != newObservations.count {
        return false
    }
    var matches = 0
    for new in newObservations {
        for old in oldObservations {
            if new.labels.first?.identifier == old.labels.first?.identifier &&
                overlapRatio(rectA: old.boundingBox, rectB: new.boundingBox) > 0.85 {
                matches += 1
            }
        }
    }
    return matches == newObservations.count
}
Recognizing dice
Recognizing dice

Handling input
Core ML Models

Build intelligence into your apps using machine learning models from the research community designed for Core ML.

Images

Models can be used with Core ML, Create ML, Xcode, and are available in a number of sizes and architecture formats. Refer to the model's associated Xcode project for guidance on how to load and use the model in your app.

Images

- FCN8s-DepthPrediction
  - Depth Estimation
  - Predict the depth from a single image.
  - Download

- MININET
  - Image Classification
  - Classify a single handwritten digit (e.g., digit 0-9)
  - Download

- MobileNetV2
  - Image Classification
  - The MobileNetV2 architecture trained to classify the dominant object in a scene frame or image
  - Download
// Turning a drawing into a number with PencilKit and Vision

let model = try VNCoreMLModel(for: MNISTClassifier().model)
let request = VNCoreMLRequest(model: model)
request.imageCropAndScaleOption = .aspectFit

let image = canvasView.drawing.image(from: canvasView.drawing.bounds, scale: 1.0)

guard let cgImage = image.cgImage,
    requestHandler = VNImageRequestHandler(cgImage: cgImage, options: [:]) else {
    return
}

try requestHandler.perform([request])
let results = request.results as? [VNClassificationObservation]
let model = try VNCoreMLModel(for: MNISTClassifier().model)
let request = VNCoreMLRequest(model: model)
request.imageCropAndScaleOption = .aspectFit

let image = canvasView.drawing.image(from: canvasView.drawing.bounds, scale: 1.0)

guard let cgImage = image.cgImage, requestHandler = VNImageRequestHandler(cgImage: cgImage, options: [:]) else {
    return
}

try requestHandler.perform([request])
let results = request.results as? [VNClassificationObservation]
// Turning a drawing into a number with PencilKit and Vision

let model = try VNCoreMLModel(for: MNISTClassifier().model)
let request = VNCoreMLRequest(model: model)
request.imageCropAndScaleOption = .aspectFit

let image = canvasView.drawing.image(from: canvasView.drawing.bounds, scale: 1.0)

guard let cgImage = image.cgImage, requestHandler = VNImageRequestHandler(cgImage: cgImage, options: [:]) else {
    return
}

try requestHandler.perform([request])
let results = request.results as? [VNClassificationObservation]
// Turning a drawing into a number with PencilKit and Vision

let model = try VNCoreMLModel(for: MNISTClassifier().model)
let request = VNCoreMLRequest(model: model)
request.imageCropAndScaleOption = .aspectFit

let image = canvasView.drawing.image(from: canvasView.drawing.bounds, scale: 1.0)

guard let cgImage = image.cgImage,
    requestHandler = VNImageRequestHandler(cgImage: cgImage, options: [:]) else {
    return
}

try requestHandler.perform([request])
let results = request.results as? [VNClassificationObservation]
let model = try VNCoreMLModel(for: MNISTClassifier().model)
let request = VNCoreMLRequest(model: model)
request.imageCropAndScaleOption = .aspectFit

let image = canvasView.drawing.image(from: canvasView.drawing.bounds, scale: 1.0)

guard let cgImage = image.cgImage, requestHandler = VNImageRequestHandler(cgImage: cgImage, options: [:]) else {
    return
}

try requestHandler.perform([request])
let results = request.results as? [VNClassificationObservation]
Predicted 1
// Get the scale of our drawing
let scale = Constants.mnistSize / containingSquare.width
var image = drawing.image(from: containingSquare, scale: scale)

let cgImage = image.cgImage
// Get the scale of the image
let scale = ConstantDrawingSquare.width

var image = drawingContext.getImageData(0, 0, 0, 0, 1, 1)
canvasView.allowsFingerDrawing = true
canvasView.tool = PKInkingTool(.marker, color: .white, width: 60.0)
canvasView.allowsFingerDrawing = true
canvasView.tool = PKInkingTool(.marker, color: .white, width: 60.0)
canvasView.allowsFingerDrawing = true

canvasView.tool = PKInkingTool(.marker, color: .white, width: 60.0)
Demo
// How to enable offline Speech to Text

let speechRecognitionRequest = SFSpeechAudioBufferRecognitionRequest()

speechRecognitionRequest\requiresOnDeviceRecognition = true
// How to enable offline Speech to Text

let speechRecognitionRequest = SFSpeechAudioBufferRecognitionRequest()
speechRecognitionRequest.requiresOnDeviceRecognition = true
Recognizing dice

Handling input
Recognizing dice
Handling input
Finalizing the game
Rules of the Game
5 + 1 = 6
5 - 1 = 4
Demo
Summary

Recognizing dice
Handling input
Finalizing the game
More Information

developer.apple.com/wwdc19/228