Computer Vision

Not Just For Breakfast Anymore

Yono Mittlefehldt — @yonomitt — 2021

Who is Yono?

Co-creator of Gus on the Go

Who is Yono?

Co-creator of Gus on the Go

Freelance software developer
specializing in Computer Vision and iOS

Who is Yono?

Co-creator of Gus on the Go

Freelance software developer
specializing in Computer Vision and iOS

Tutorial writer for raywenderlich.com

Who is Yono?

Co-creator of Gus on the Go

Freelance software developer
specializing in Computer Vision and iOS

Tutorial writer for raywenderlich.com

One-time chauffeur to Bruce Campbell

Will this be on the test?

Not at all

The Kids in the Hall

Canada's Monty Python

What does this have to do with Computer Vision?

— You, right about now

Crushing Your Head

The App

Crushing Your Head

The App

Crushing Your Head

The App

Hand Pose Detector

Crushing Your Head

The App

Hand Pose Detector

Face Detector

Crushing Your Head
The App

Hand Pose Detector

Face Detector

Comparing Feature Vectors

Crushing Your Head

The App

Hand Pose Detector

Face Detector

Comparing Feature Vectors

Object Tracking

Crushing Your Head
The App

Hand Pose Detector

Face Detector

Comparing Feature Vectors

Object Tracking

BONUS:

Custom Classifier

Apple's Vision Framework

Vision Framework

Vision Framework

Common computer vision tasks

Vision Framework

Common computer vision tasks

Optimized to run on iPhones and iPads

Vision Framework

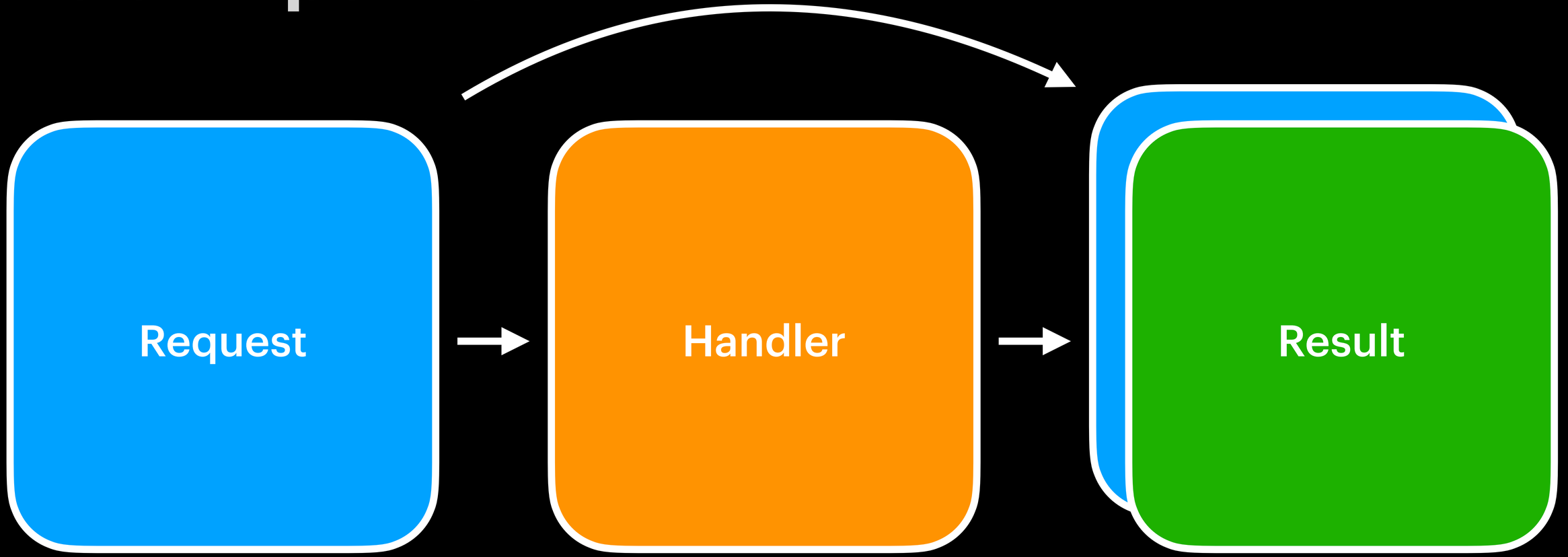
Common computer vision tasks

Optimized to run on iPhones and iPads

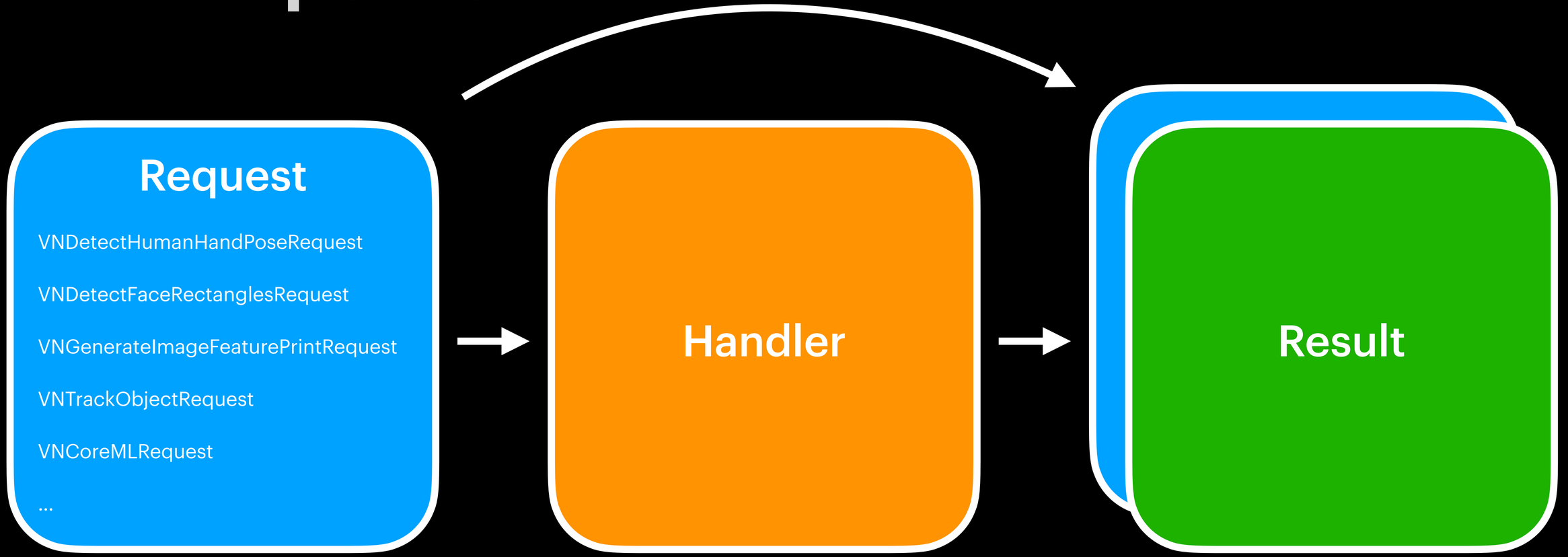
Updated yearly

Vision Pipeline

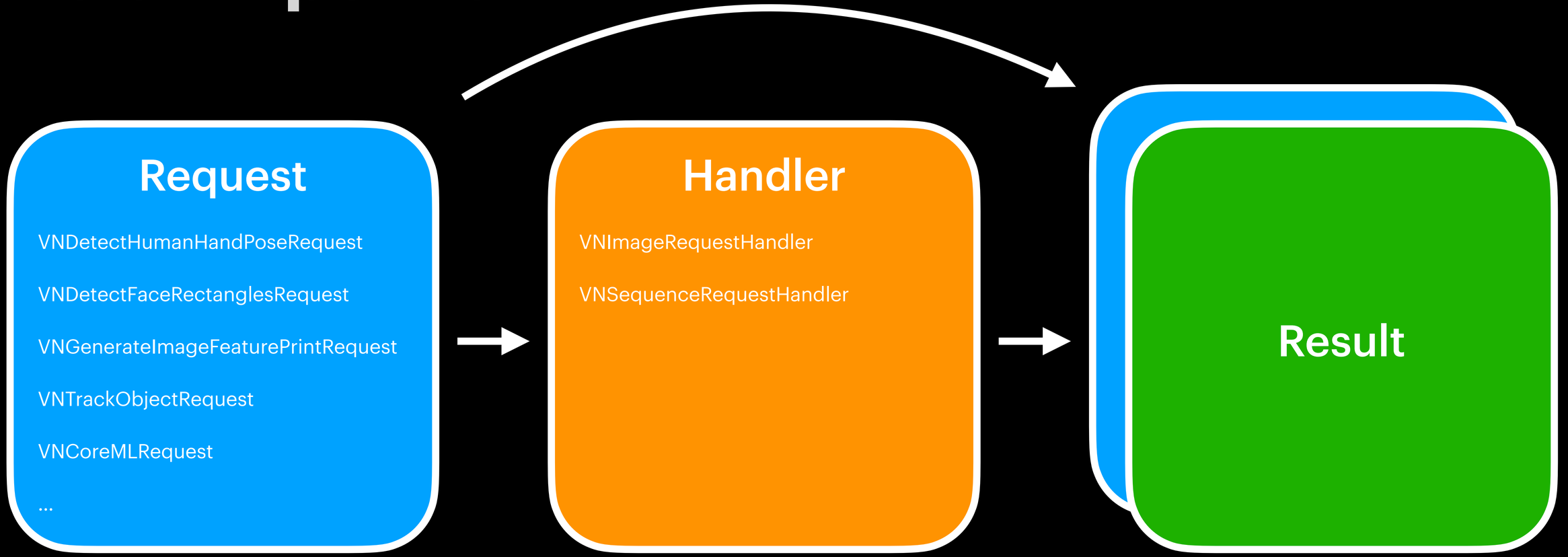
Vision Pipeline



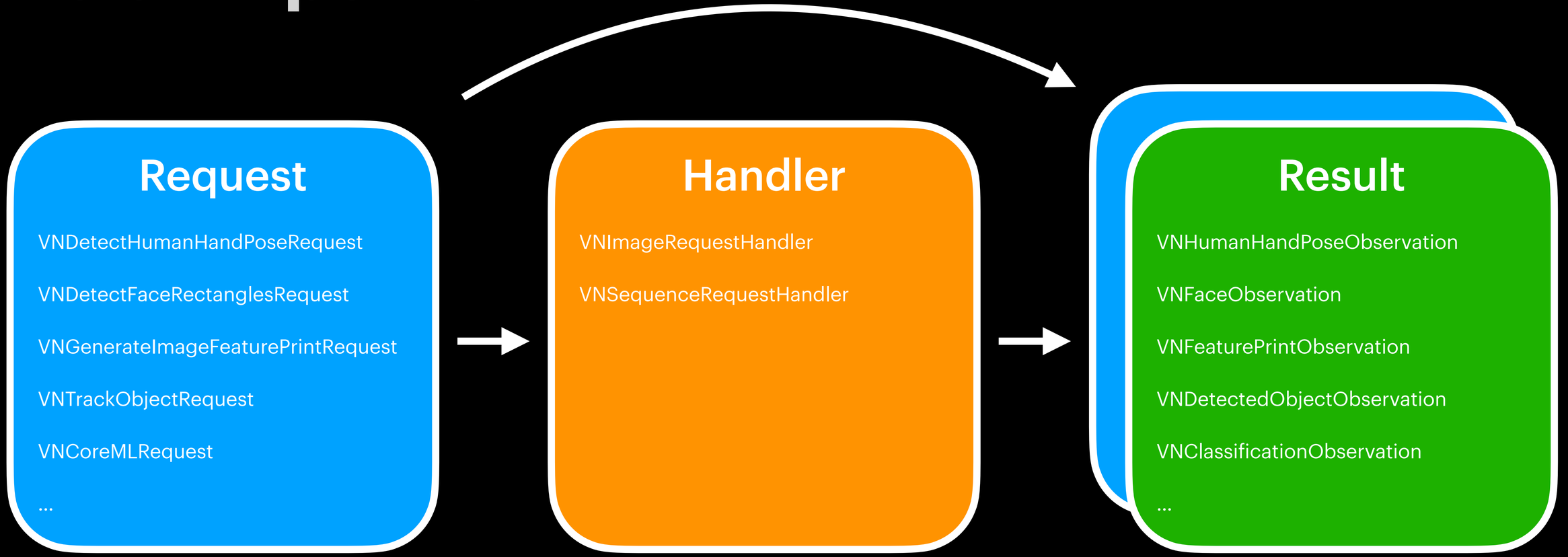
Vision Pipeline



Vision Pipeline



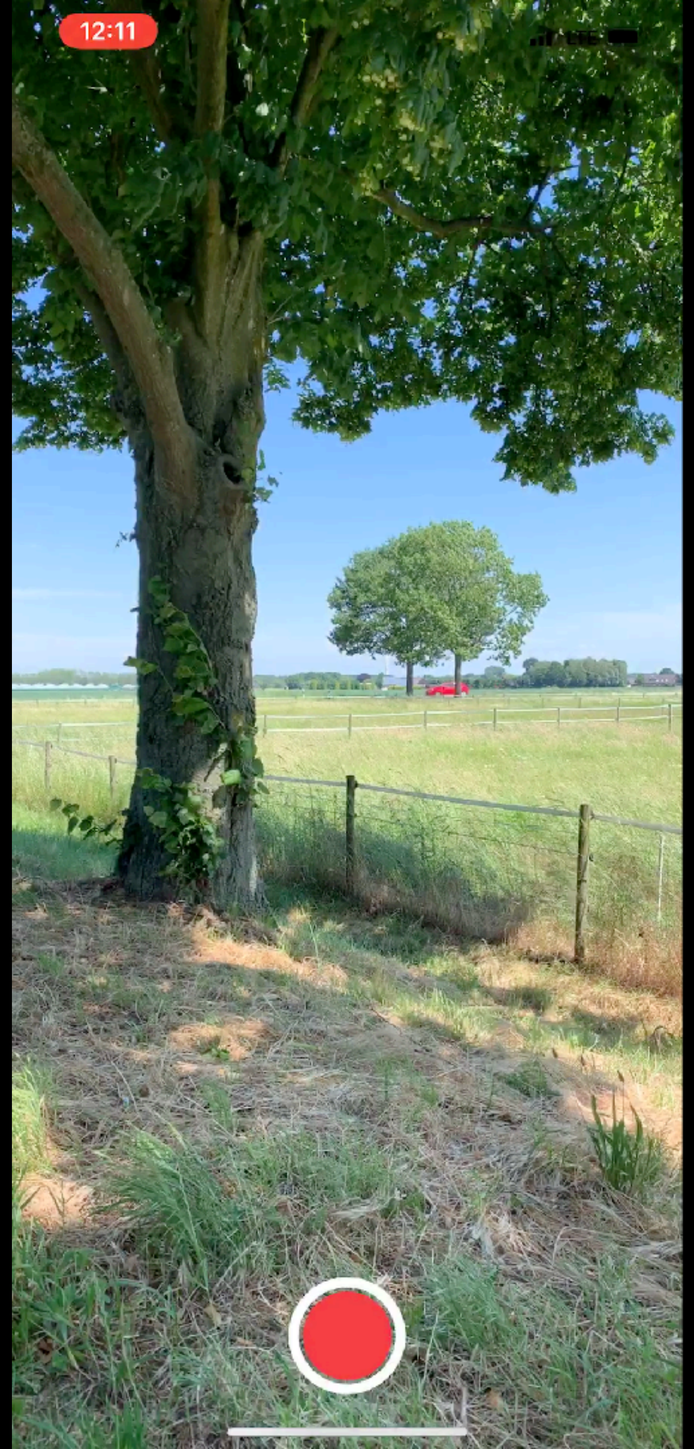
Vision Pipeline



Vision Tutorials

Vision Tutorials

Photo Stacking



www.raywenderlich.com/3733151-photo-stacking-in-ios-with-vision-and-metal

Vision Tutorials

Photo Stacking

Saliency Analysis

www.raywenderlich.com/5807038-saliency-analysis-in-ios-using-vision

Vision Tutorials

Photo Stacking

Saliency Analysis

Face Landmark Detection

www.raywenderlich.com/1163620-face-detection-tutorial-using-the-vision-framework-for-ios

Crushing Your Head
The App

Hand Pose Detector

Face Detector

Comparing Feature Vectors

Object Tracking

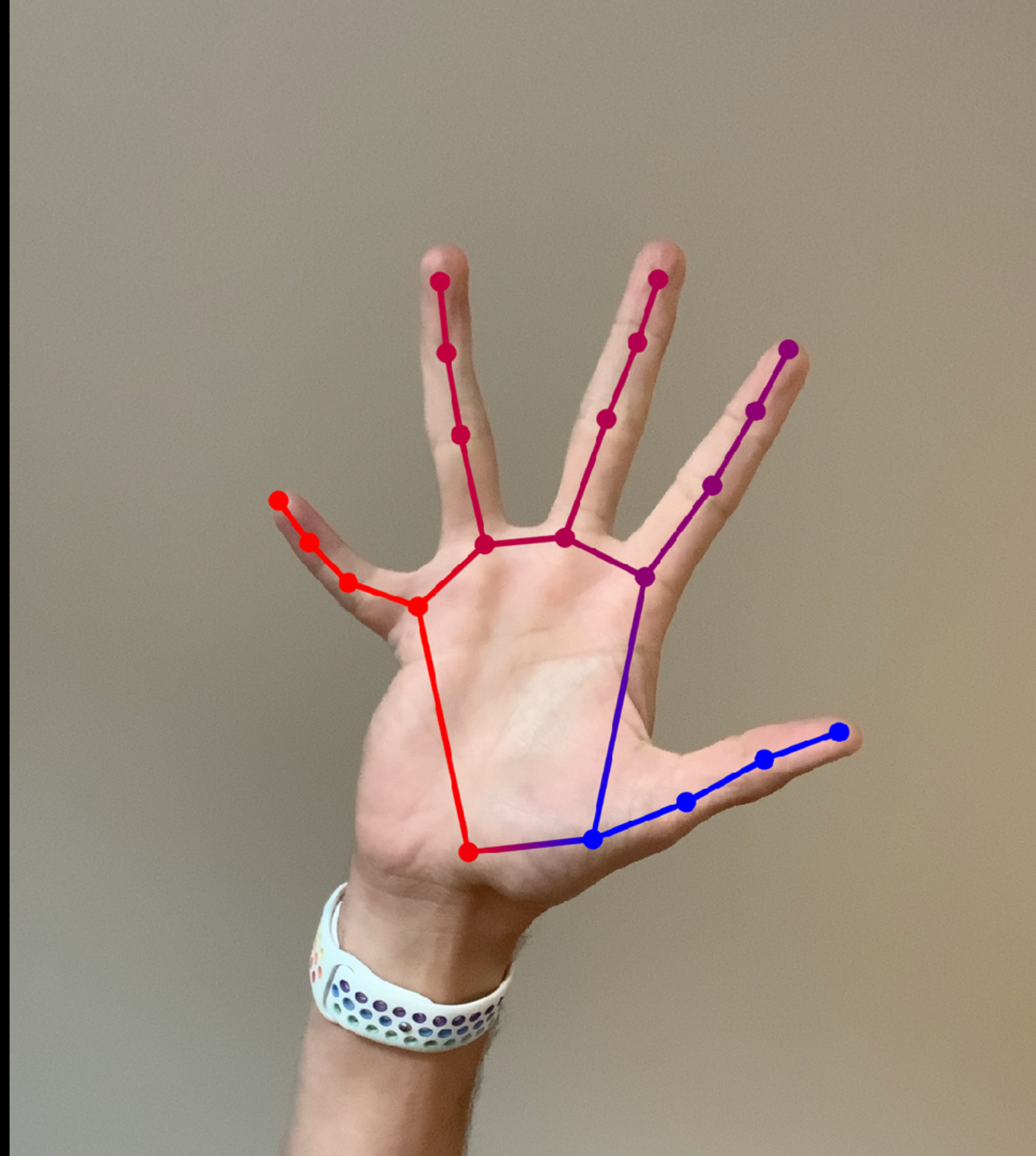
BONUS:

Custom Classifier

Hand Pose Request

Detects up to 21 points on a hand

- 4 points per finger and thumb
- 1 point for the wrist

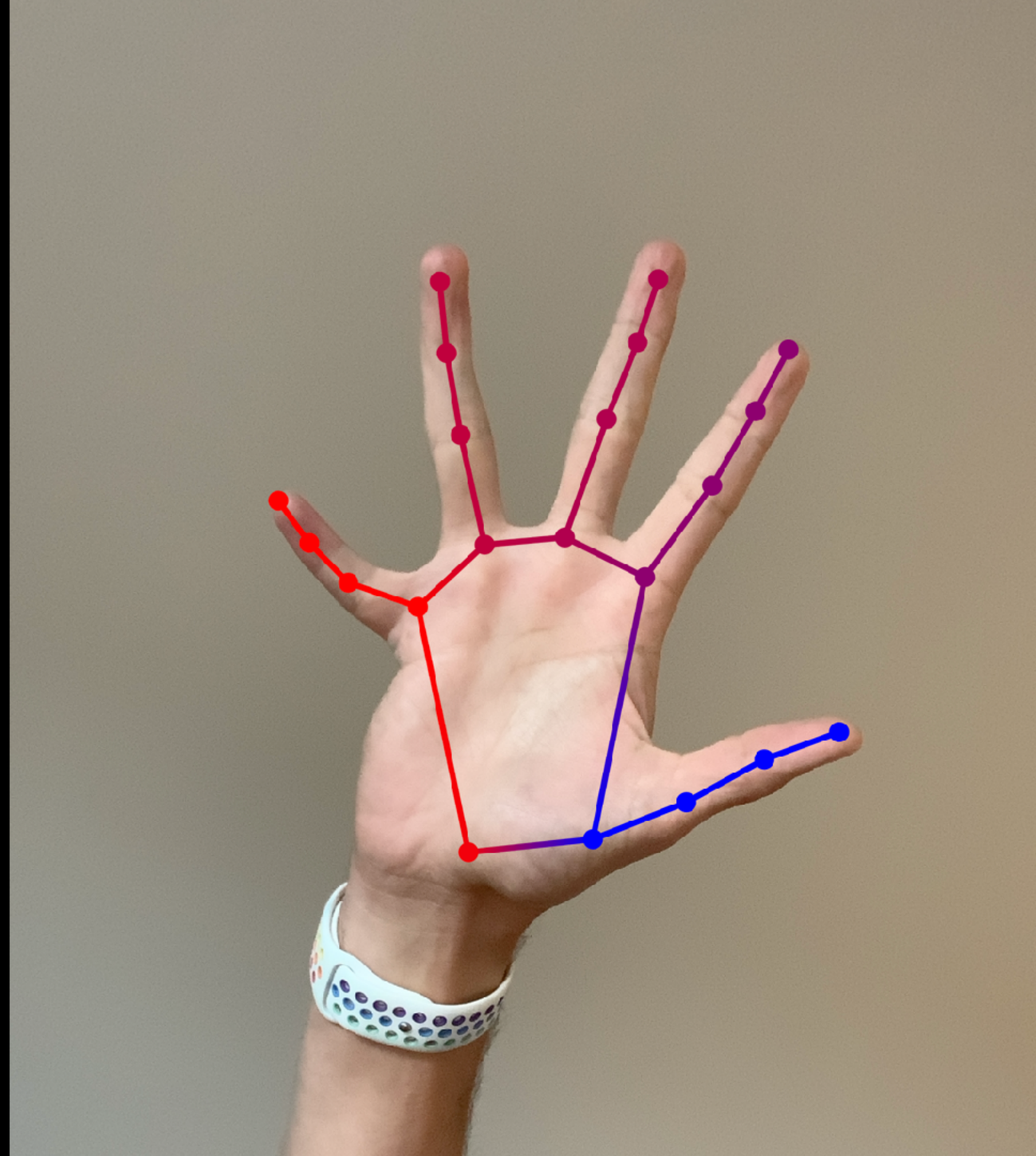


Hand Pose Request*

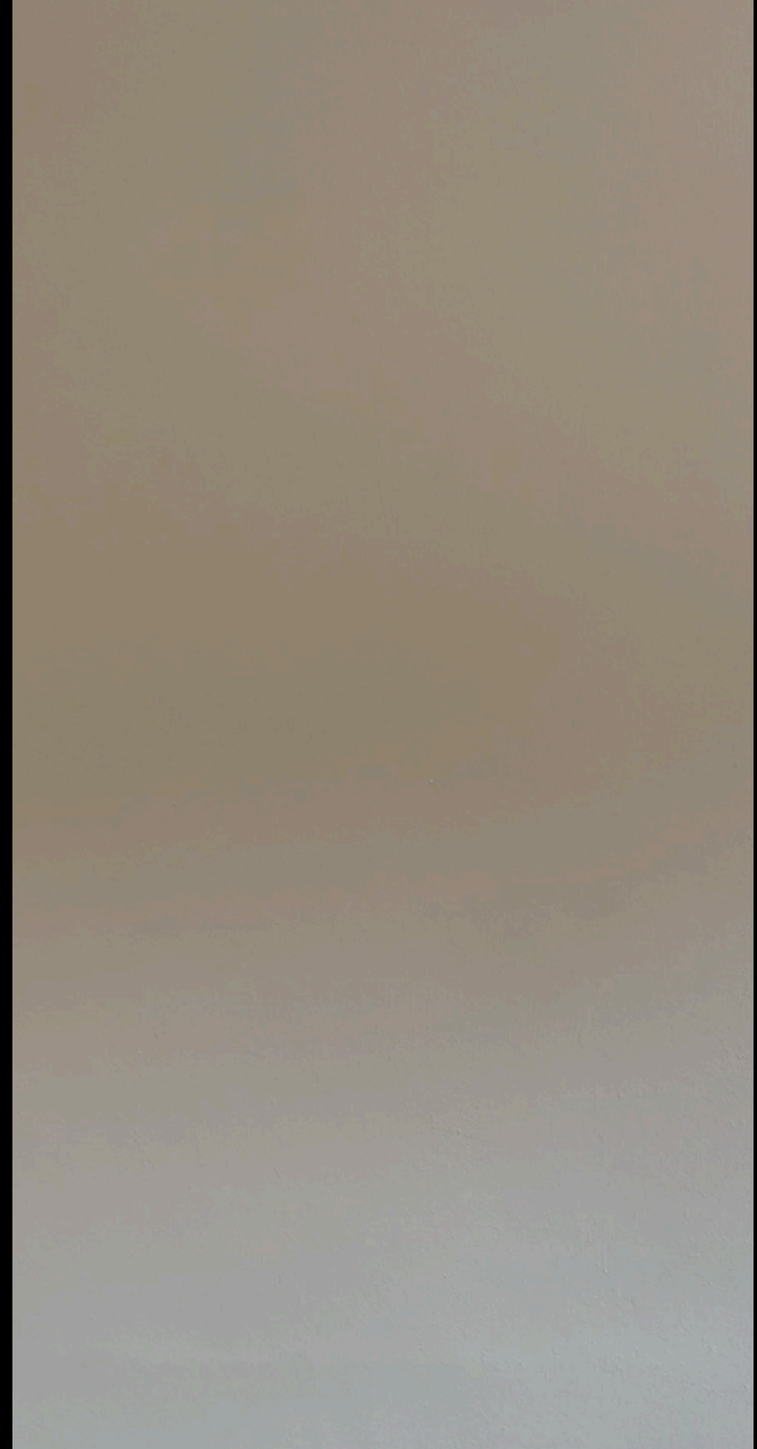
Detects up to 21 points on a hand

- 4 points per finger and thumb
- 1 point for the wrist

* Human hands

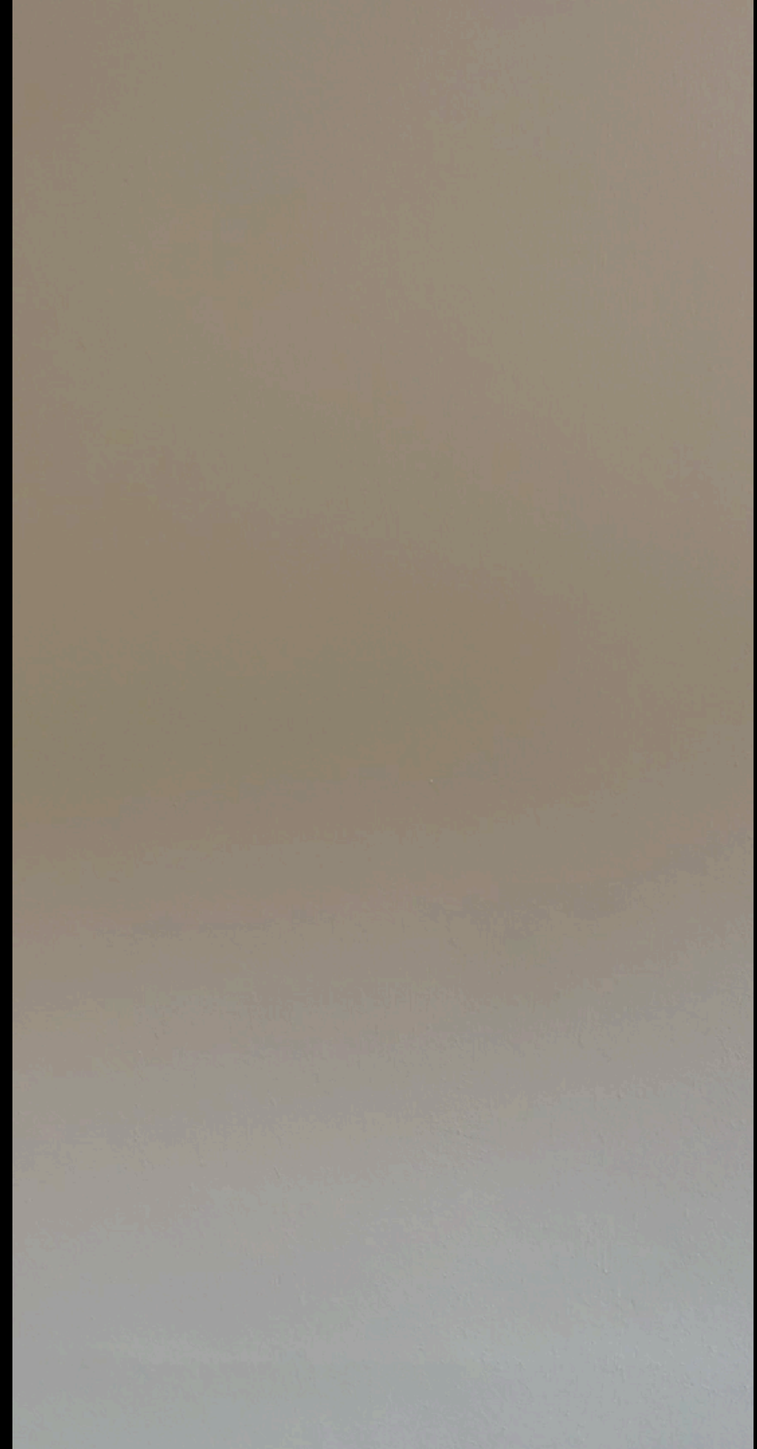


Hand Pose in the App



Hand Pose in the App

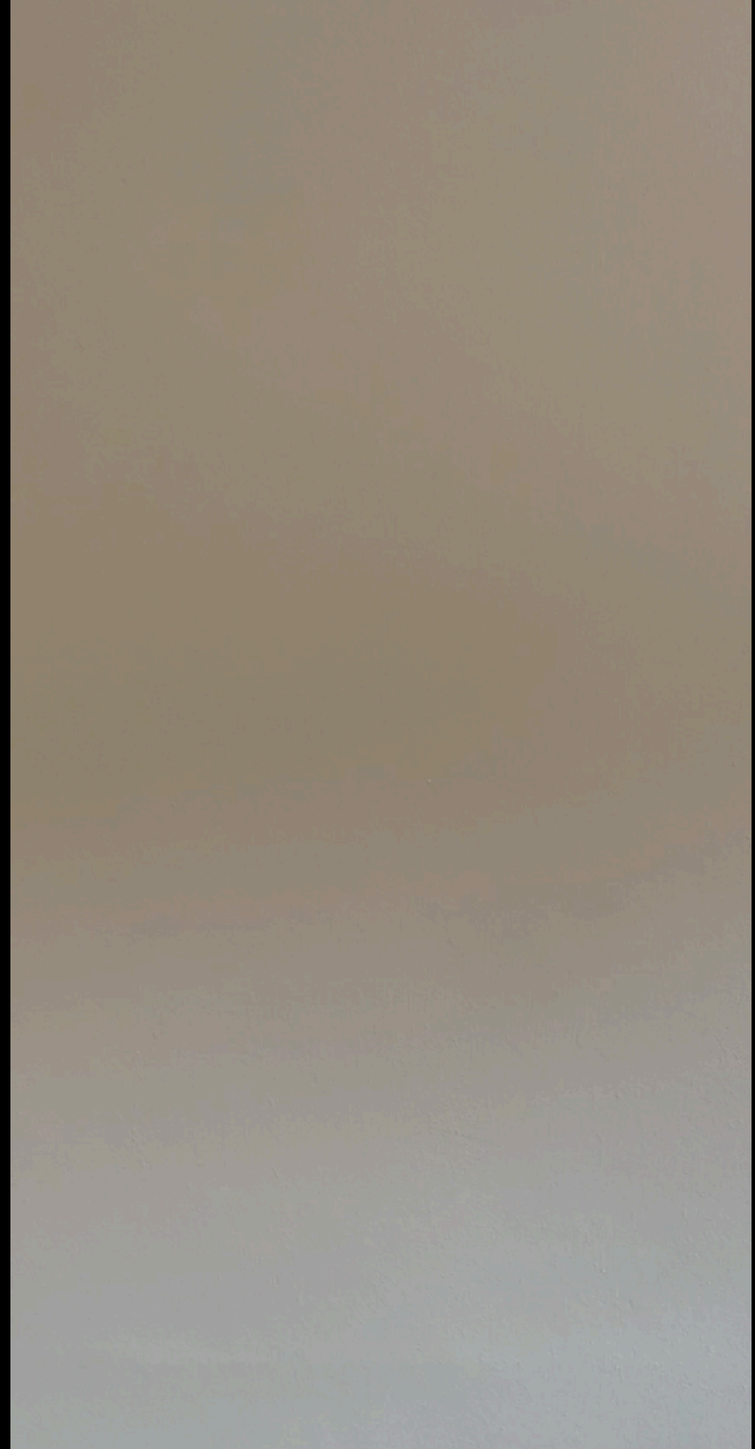
Unique game mechanic



Hand Pose in the App

Unique game mechanic

Simplify by just looking at the tip
of the thumb and index finger

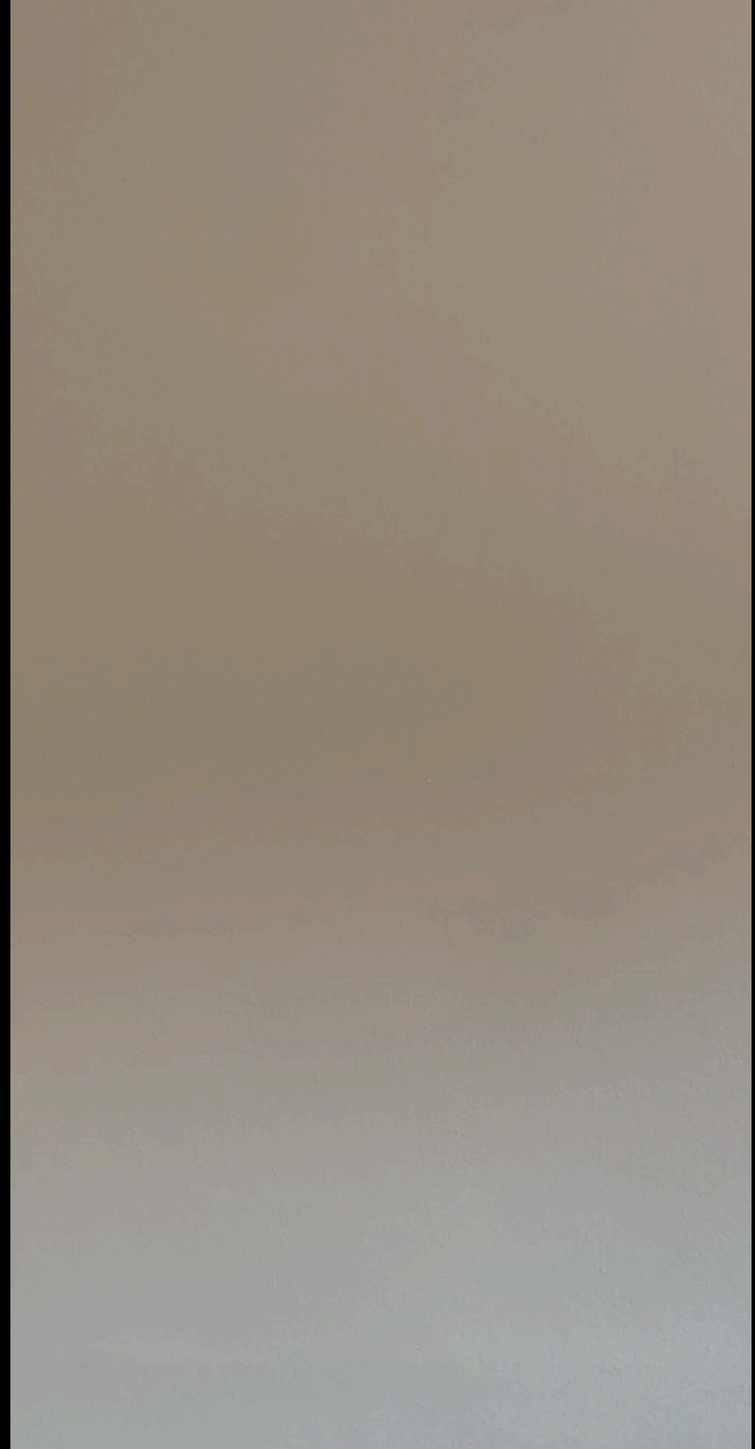


Hand Pose in the App

Unique game mechanic

Simplify by just looking at the tip of the thumb and index finger

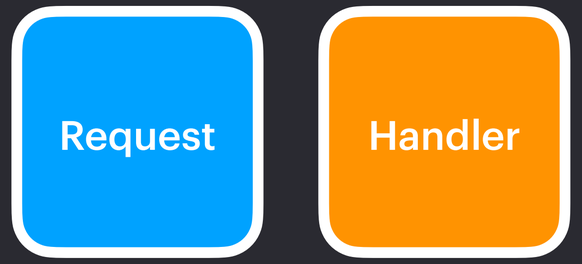
If the tips are close together, the user is pinching



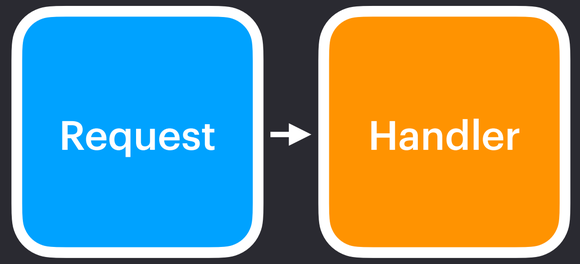
```
// Create the Vision request
let request = VNDetectHumanHandPoseRequest()
```

```
// Create the Vision request
let request = VNDetectHumanHandPoseRequest()

// Configure the request
request.maximumHandCount = 1
```

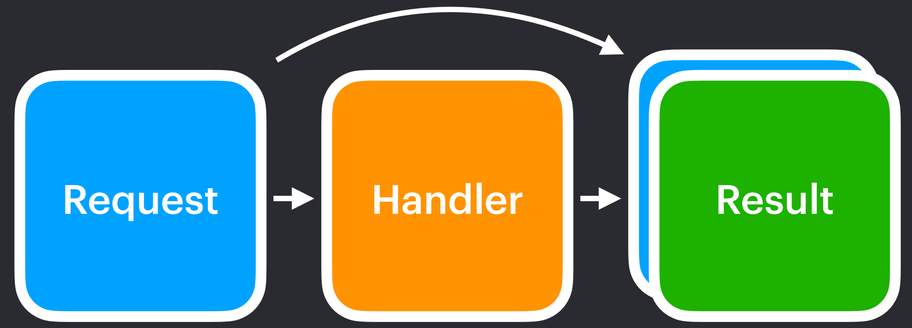
```swift
// Create the Vision request
let request = VNDetectHumanHandPoseRequest()

// Configure the request
request.maximumHandCount = 1

// Create the request handler
let requestHandler = VNImageRequestHandler(cvPixelBuffer: image, options: [:])
```



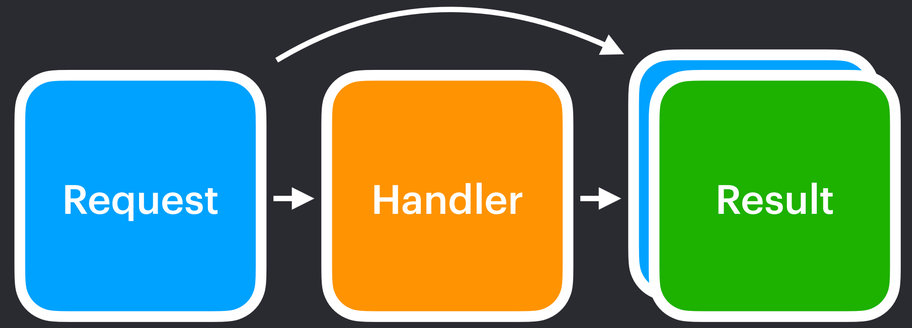
```swift
// Create the Vision request
let request = VNDetectHumanHandPoseRequest()

// Configure the request
request.maximumHandCount = 1

// Create the request handler
let requestHandler = VNImageRequestHandler(cvPixelBuffer: image, options: [:])

// Perform the request
try? requestHandler.perform([request])
```



```swift
// Create the Vision request
let request = VNDetectHumanHandPoseRequest()

// Configure the request
request.maximumHandCount = 1

// Create the request handler
let requestHandler = VNImageRequestHandler(cvPixelBuffer: image, options: [:])

// Perform the request
try? requestHandler.perform([request])

// Check results
if let results = request.results as? [VNHumanHandPoseObservation],
   let result = results.first {


}
```



```swift
// Create the Vision request
let request = VNDetectHumanHandPoseRequest()

// Configure the request
request.maximumHandCount = 1

// Create the request handler
let requestHandler = VNImageRequestHandler(cvPixelBuffer: image, options: [:])

// Perform the request
try? requestHandler.perform([request])

// Check results
if let results = request.results as? [VNHumanHandPoseObservation],
   let result = results.first {
    // Do awesome things with the result
    return Pinch(from: result)
}
```



Crushing Your Head
The App

Hand Pose Detector

Face Detector

Comparing Feature Vectors

Object Tracking

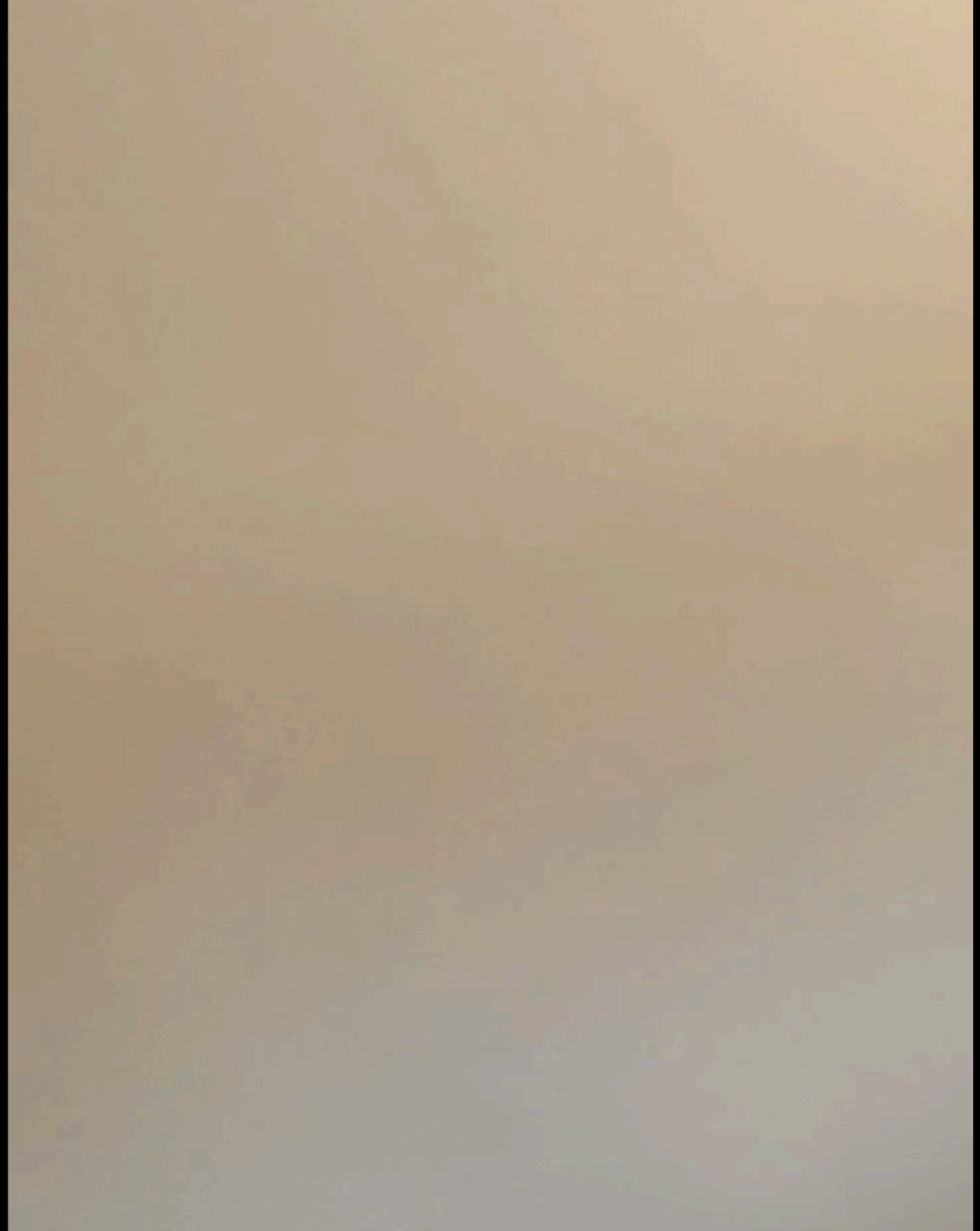
BONUS:

Custom Classifier

Face Detection in the App

Face Detection in the App

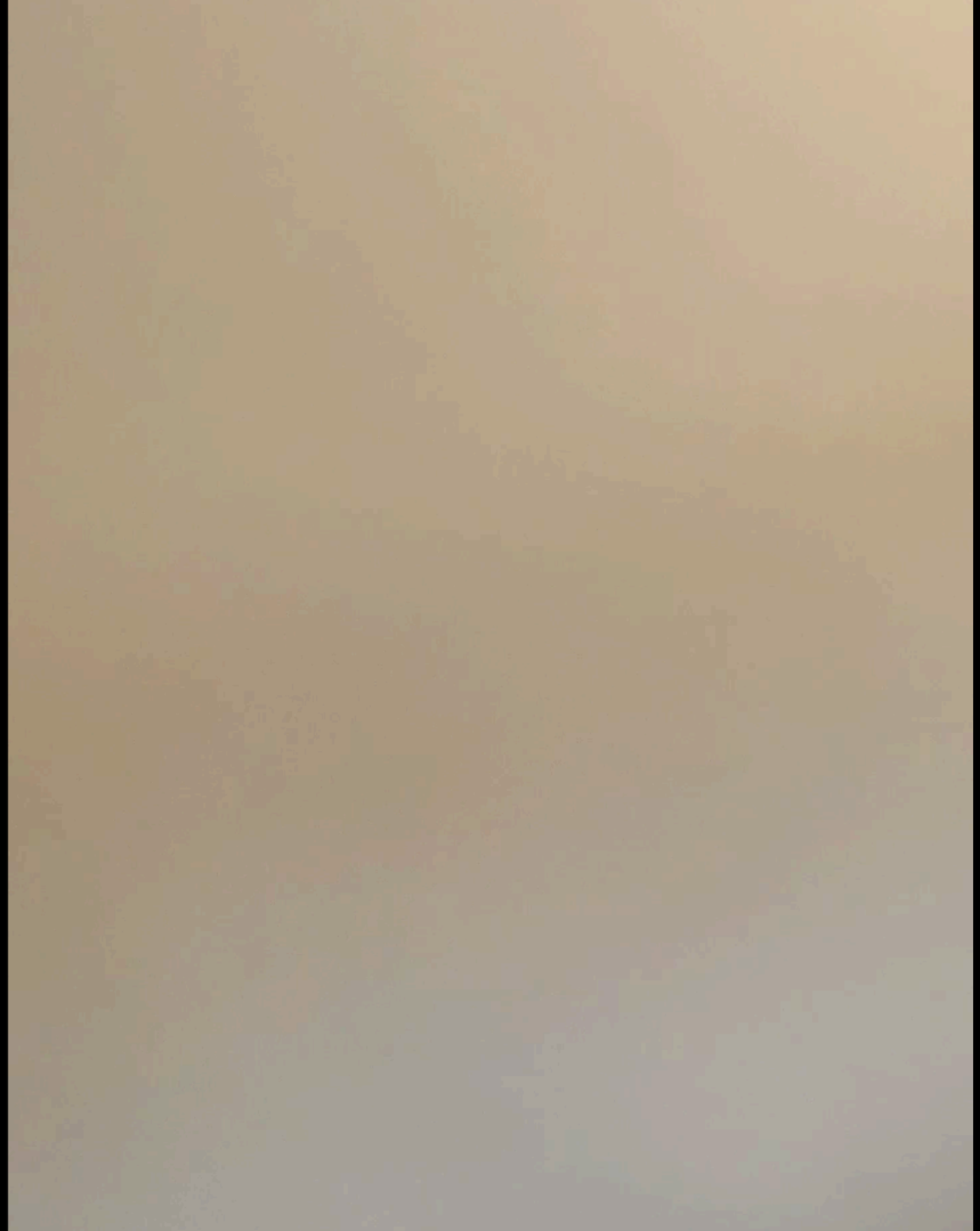
Goal of the app is to crush heads



Face Detection in the App

Goal of the app is to crush heads

Faces are a good proxy for heads

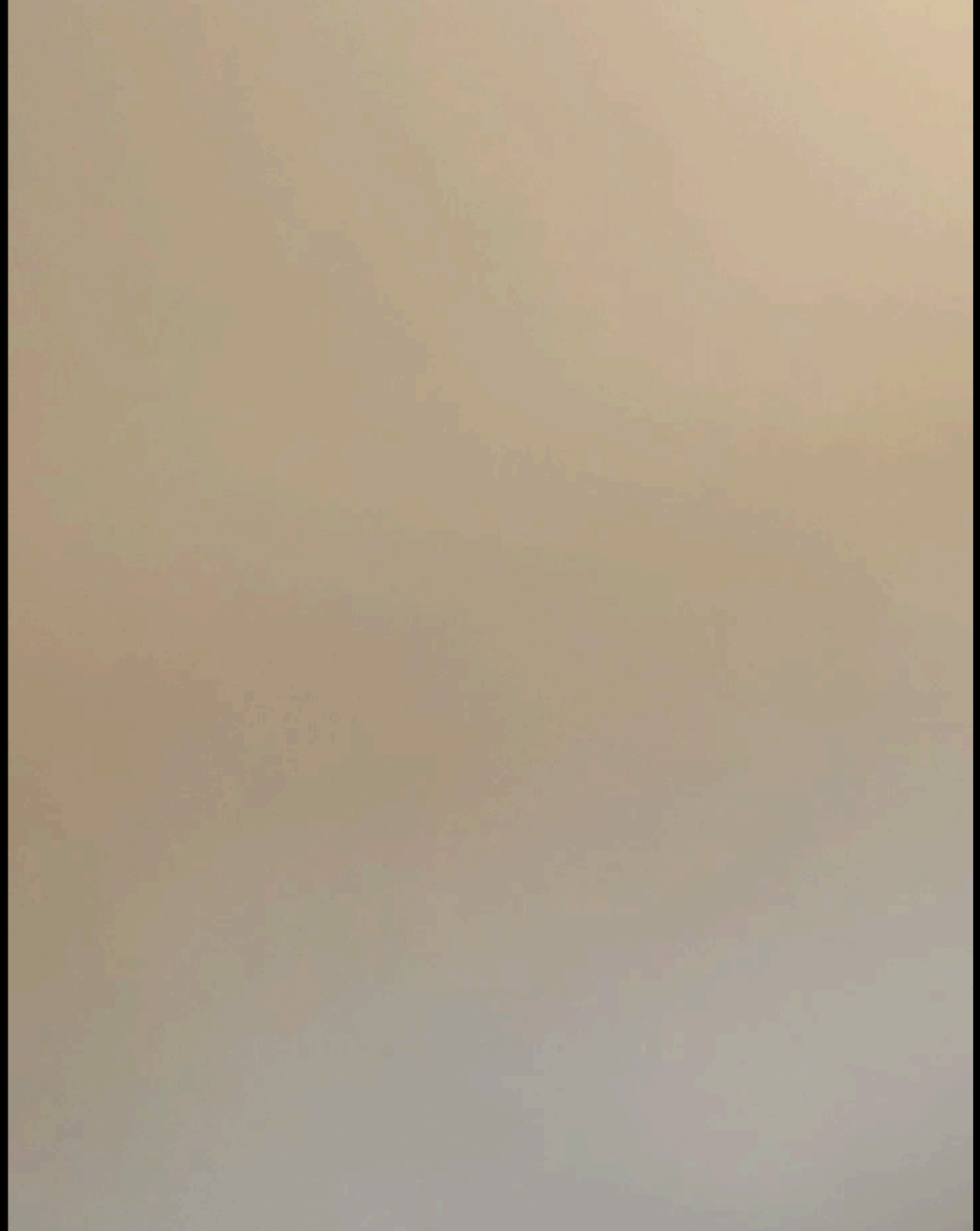


Face Detection in the App

Goal of the app is to crush heads

Faces are a good proxy for heads

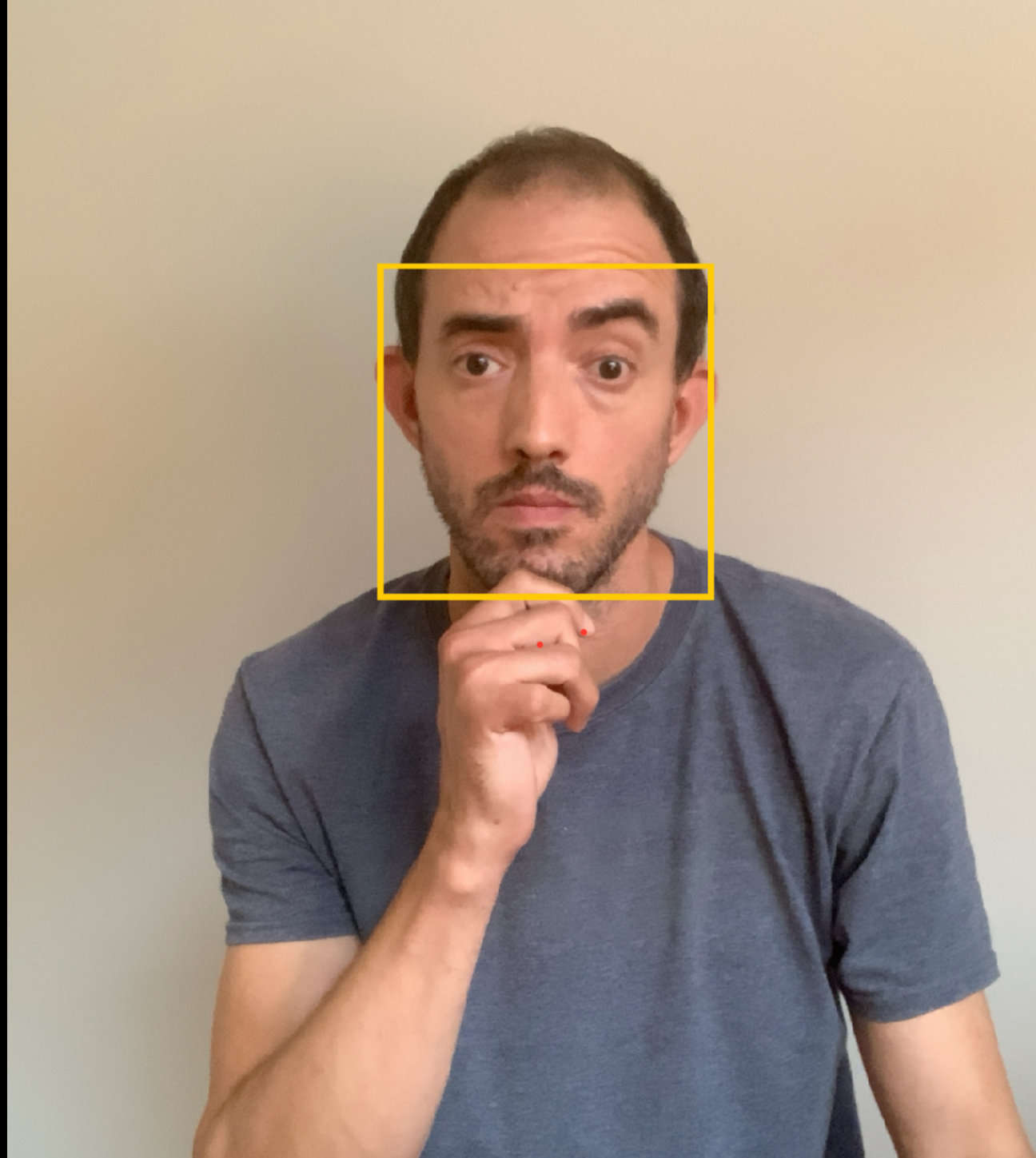
If a pinch occurs over a detected face, we can be reasonably sure the head was crushed



Face Rectangles Request

Face Rectangles Request

Normal faces



Face Rectangles Request

Normal faces

Profile faces

Face Rectangles Request

Normal faces

Profile faces

Obscured faces

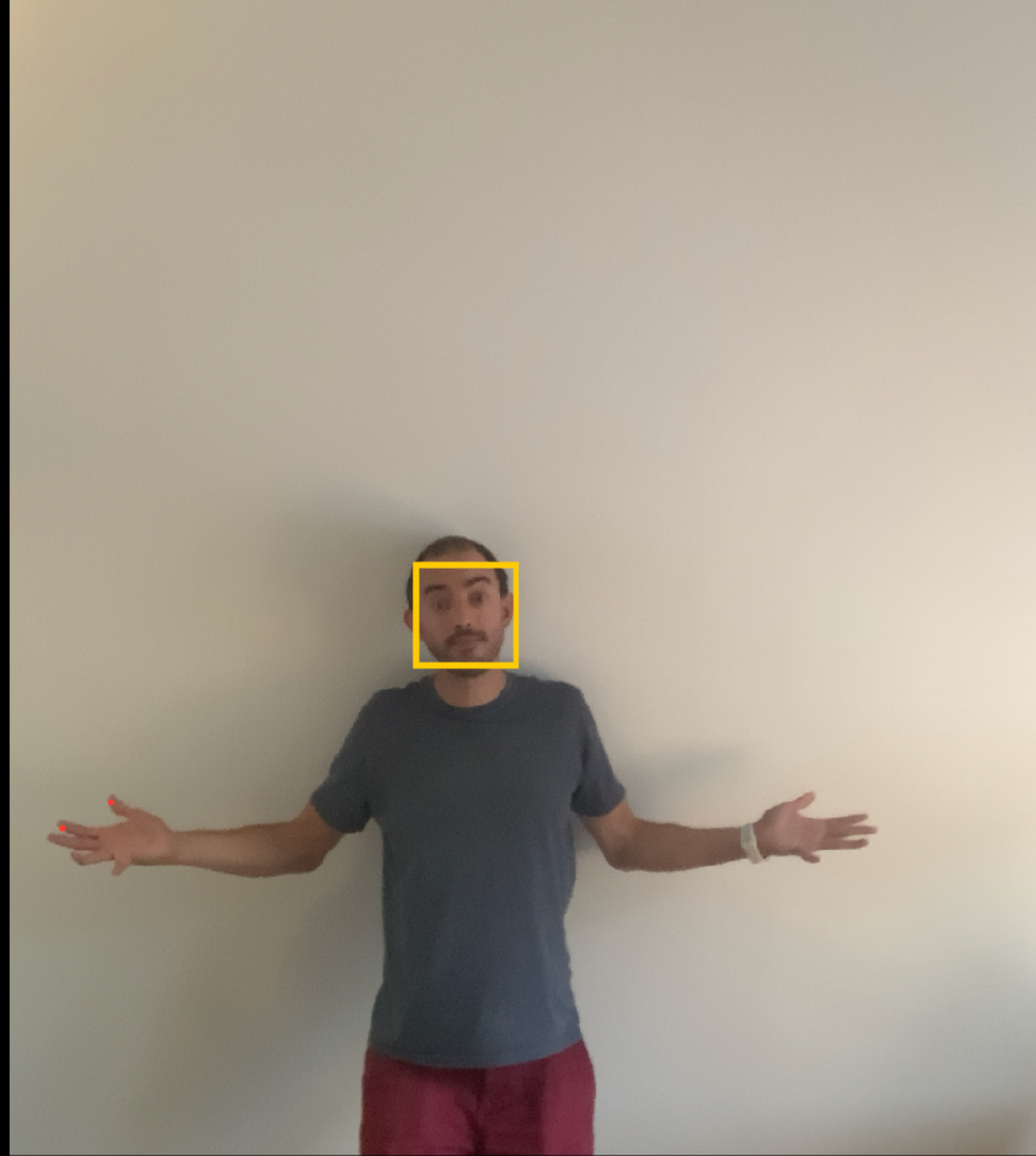
Face Rectangles Request

Normal faces

Profile faces

Obscured faces

Small faces



Face Rectangles Request

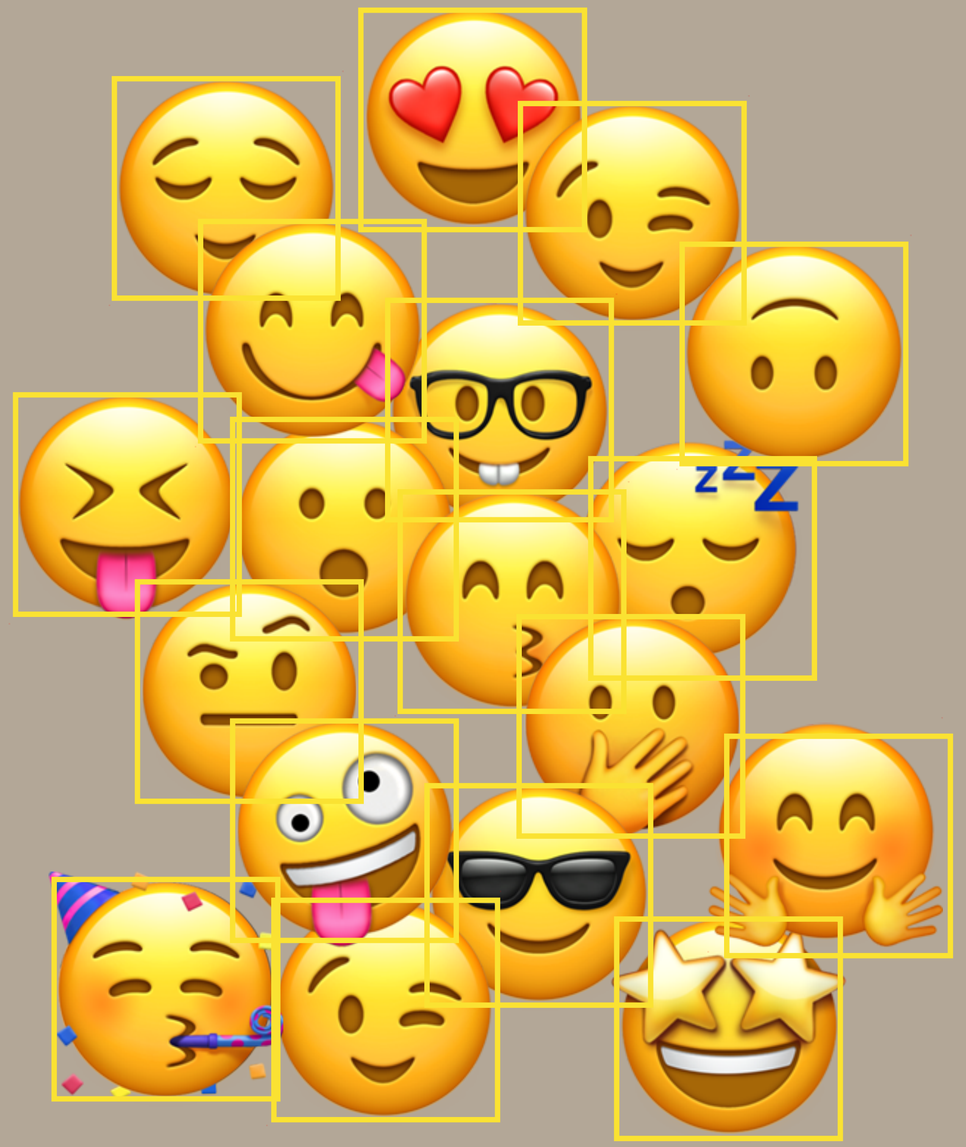
Normal faces

Profile faces

Obscured faces

Small faces

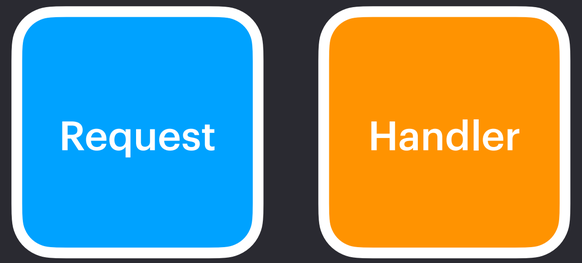
Multiple faces



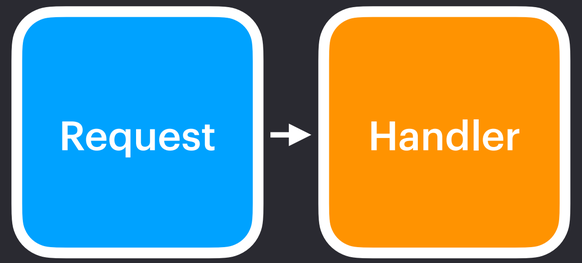
```
// Create the Vision request
let request = VNDetectFaceRectanglesRequest()
```

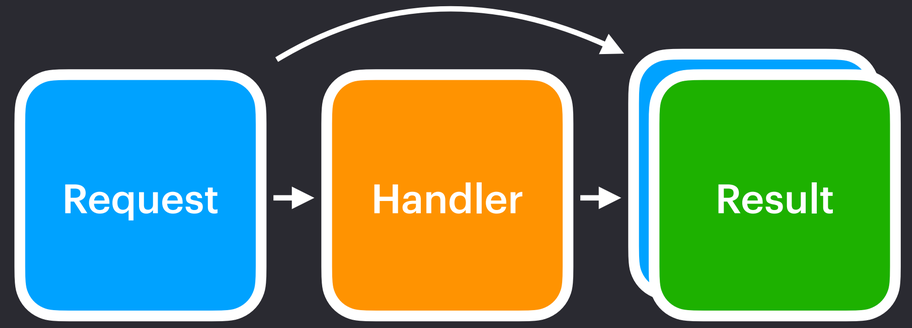
```swift
// Create the Vision request
let request = VNDetectFaceRectanglesRequest()

// Configure the request
request.regionOfInterest = CGRect(x: 0.2, y: 0.5, width: 0.6, height: 0.3)
```

```swift
// Create the Vision request
let request = VNDetectFaceRectanglesRequest()

// Configure the request
request.regionOfInterest = CGRect(x: 0.2, y: 0.5, width: 0.6, height: 0.3)

// Create the request handler
let requestHandler = VNImageRequestHandler(cvPixelBuffer: image, options: [:])
```

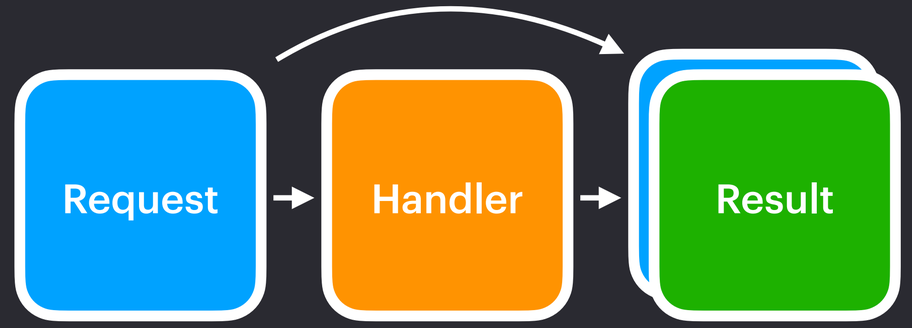


```swift
// Create the Vision request
let request = VNDetectFaceRectanglesRequest()

// Configure the request
request.regionOfInterest = CGRect(x: 0.2, y: 0.5, width: 0.6, height: 0.3)

// Create the request handler
let requestHandler = VNImageRequestHandler(cvPixelBuffer: image, options: [:])

// Perform the request
try? requestHandler.perform([request])
```






```
// Create the Vision request
let request = VNDetectFaceRectanglesRequest()

// Configure the request
request.regionOfInterest = CGRect(x: 0.2, y: 0.5, width: 0.6, height: 0.3)

// Create the request handler
let requestHandler = VNImageRequestHandler(cvPixelBuffer: image, options: [:])

// Perform the request
try? requestHandler.perform([request])

// Check results
if let results = request.results as? [VNFaceObservation] {


}
```




```
// Create the Vision request
let request = VNDetectFaceRectanglesRequest()

// Configure the request
request.regionOfInterest = CGRect(x: 0.2, y: 0.5, width: 0.6, height: 0.3)

// Create the request handler
let requestHandler = VNImageRequestHandler(cvPixelBuffer: image, options: [:])

// Perform the request
try? requestHandler.perform([request])

// Check results
if let results = request.results as? [VNFaceObservation] {
    // Do awesome things with the results
    return results.map { Head(id: $0.uuid, bbox: $0.boundingBox) }
}
```

```
request.regionOfInterest = CGRect(x: 0.2, y: 0.5, width: 0.6, height: 0.3)
```

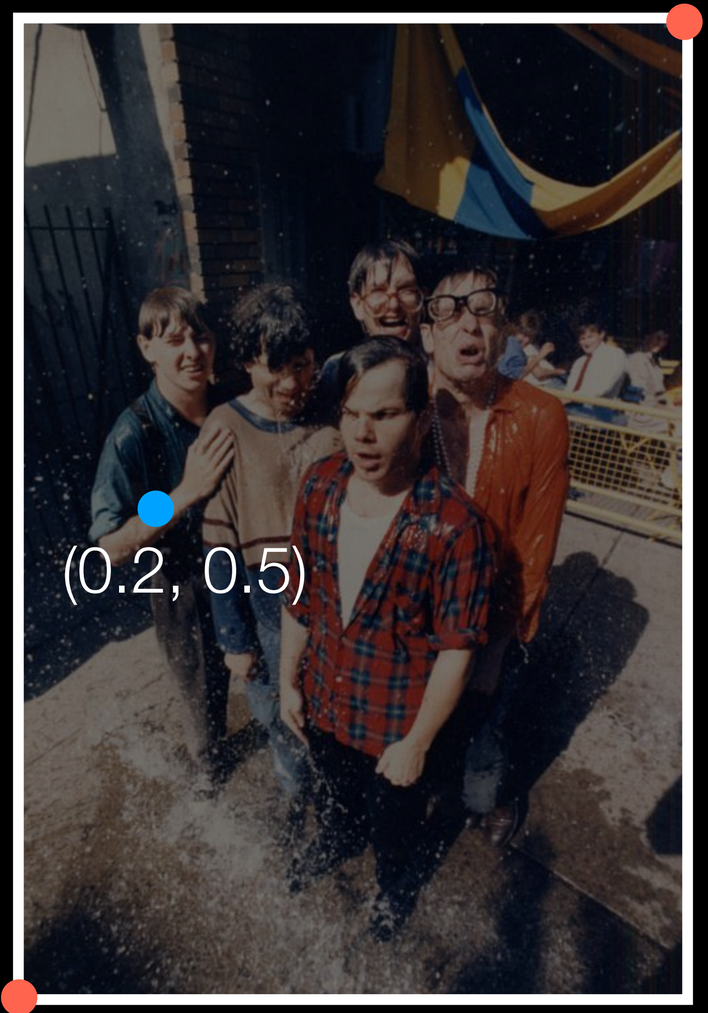
```
request.regionOfInterest = CGRect(x: 0.2, y: 0.5, width: 0.6, height: 0.3)
```

```
request.regionOfInterest = CGRect(x: 0.2, y: 0.5, width: 0.6, height: 0.3)
```

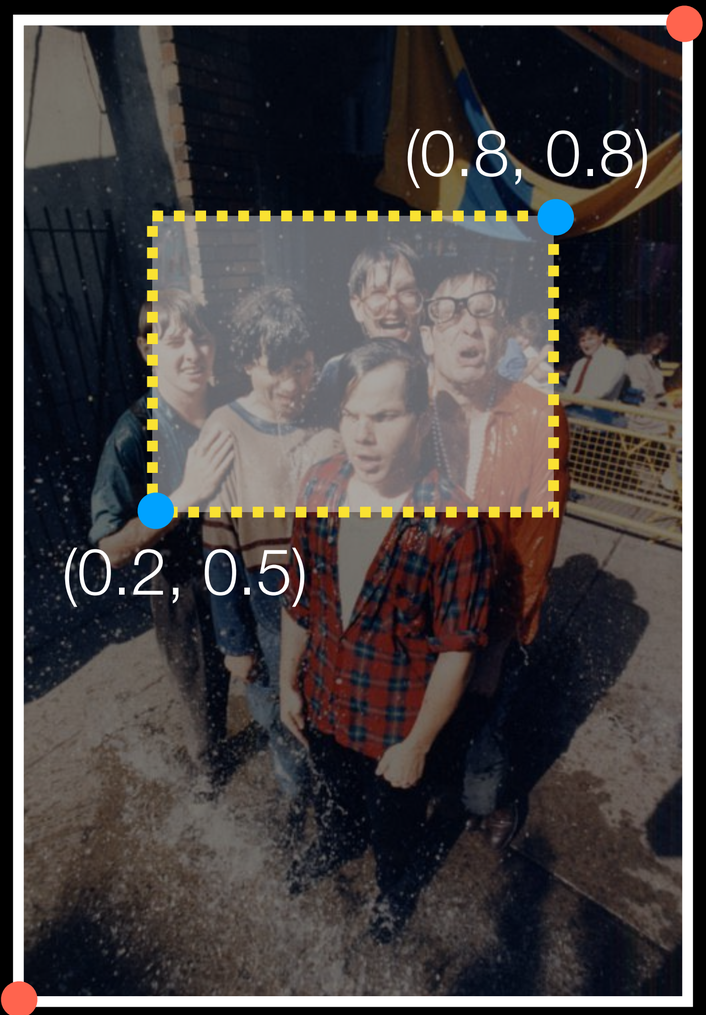
(0.0, 0.0)

```
request.regionOfInterest = CGRect(x: 0.2, y: 0.5, width: 0.6, height: 0.3)
```

```
request.regionOfInterest = CGRect(x: 0.2, y: 0.5, width: 0.6, height: 0.3)
```



```
request.regionOfInterest = CGRect(x: 0.2, y: 0.5, width: 0.6, height: 0.3)
```



Why specify a Region of Interest?

Why specify a Region of Interest?

Why specify a Region of Interest?

Why specify a Region of Interest?

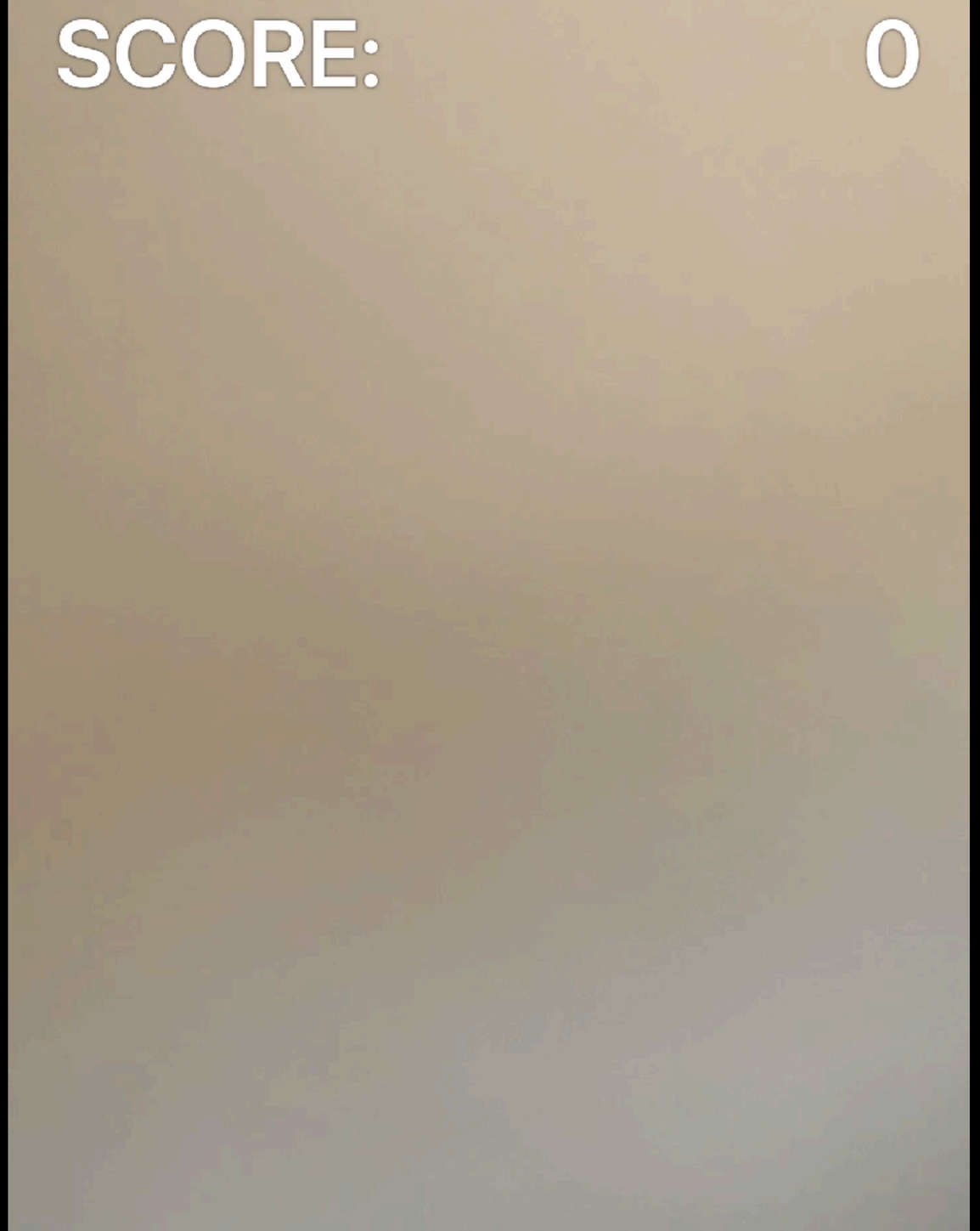
Why specify a Region of Interest?

Why specify a Region of Interest?

Why specify a Region of Interest?

One Problem

Players can game the system



Crushing Your Head
The App

Hand Pose Detector

Face Detector

Comparing Feature Vectors

Object Tracking

BONUS:

Custom Classifier

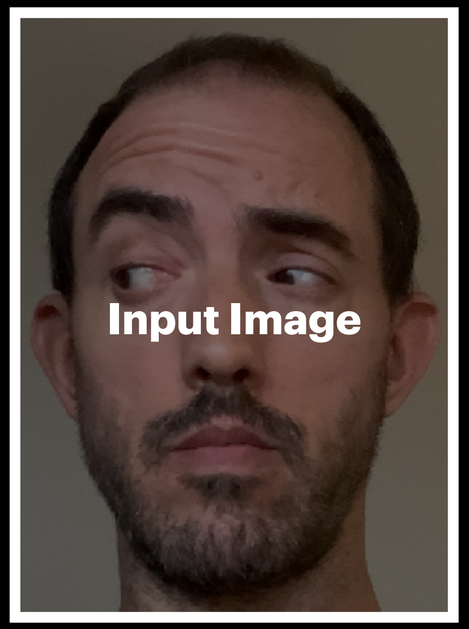
Feature Vector

a.k.a. Feature Print

a.k.a. Feature Embedding

An array of numbers representing important characteristics of an image

Feature Vector Generation



Model

-0.63

0.32

0.99

-0.47

-0.02

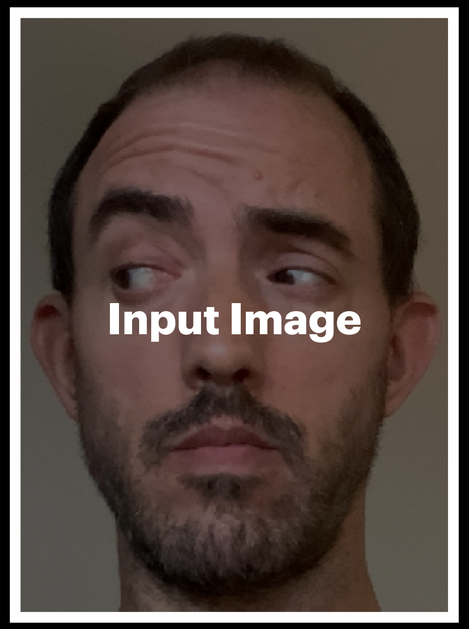
0.21

-0.92

...

0.81

Does this look familiar?



Model

-0.63

0.32

0.99

-0.47

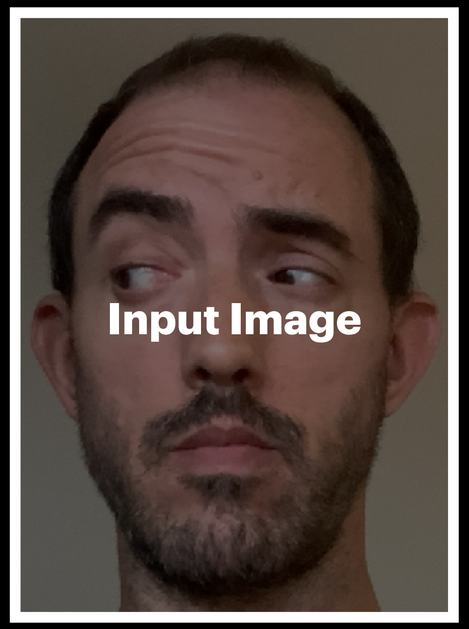
-0.02

0.21

-0.92

...

0.81



Model

BF21

3EA3

3F7D

BEF0

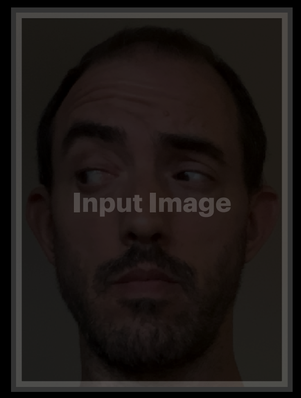
BCA3

3E57

BF6B

...

3F4F



Model

BF21

3EA3

3F7D

BEF0

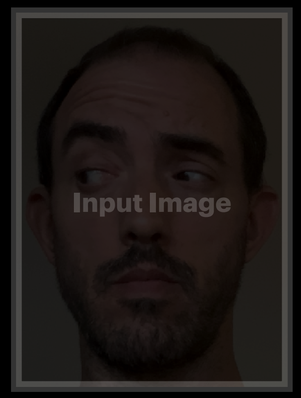
BCA3

3E57

BF6B

...

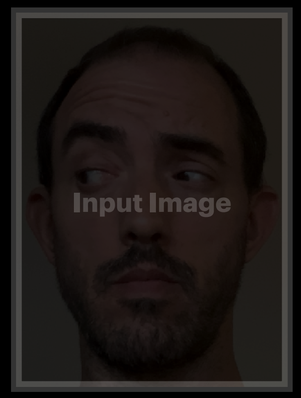
3F4F



Model

BF21
3EA3
3F7D
BEF0
BCA3
3E57
BF6B
...
3F4F

Hash function



Model

BF21
3EA3
3F7D
BEF0
BCA3
3E57
BF6B
...
3F4F

Hash value

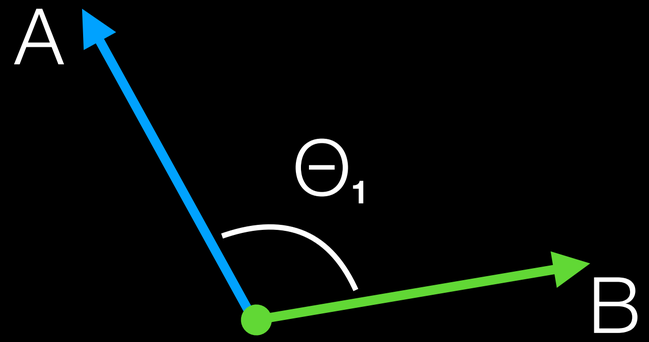
Hash function

WARNING: Math Ahead

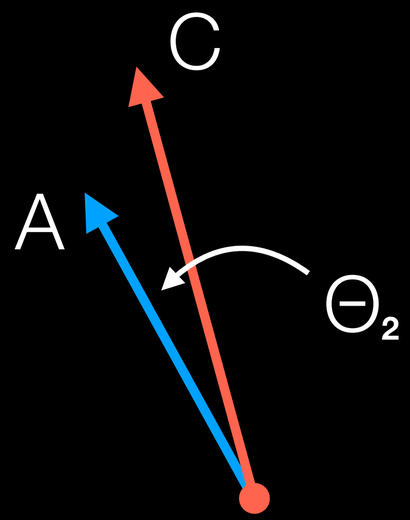
Comparing 2D Vectors

Comparing 2D Vectors

The Dot Product



Less similar

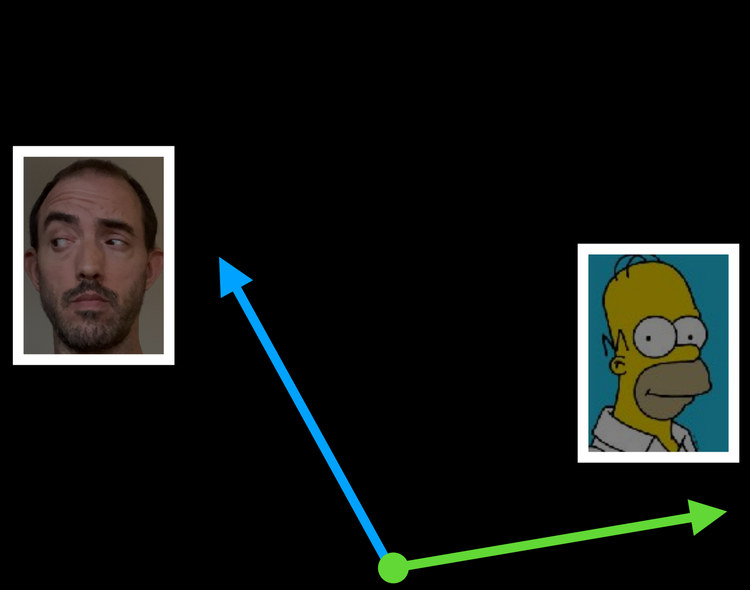


More similar

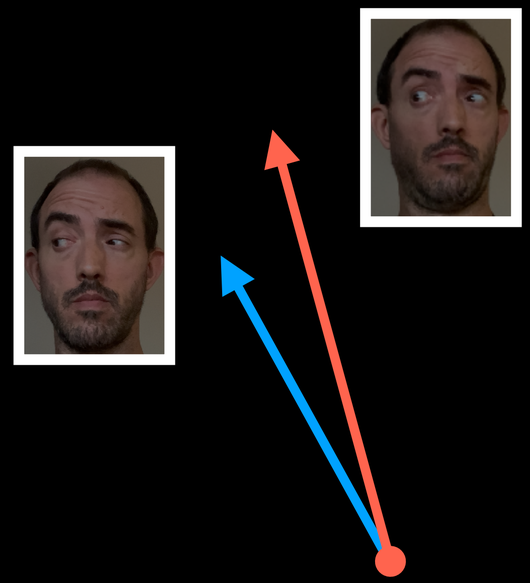
Even works on 1000s of dimensions

i.e. Feature Vectors

Comparing Feature Vectors



Less similar

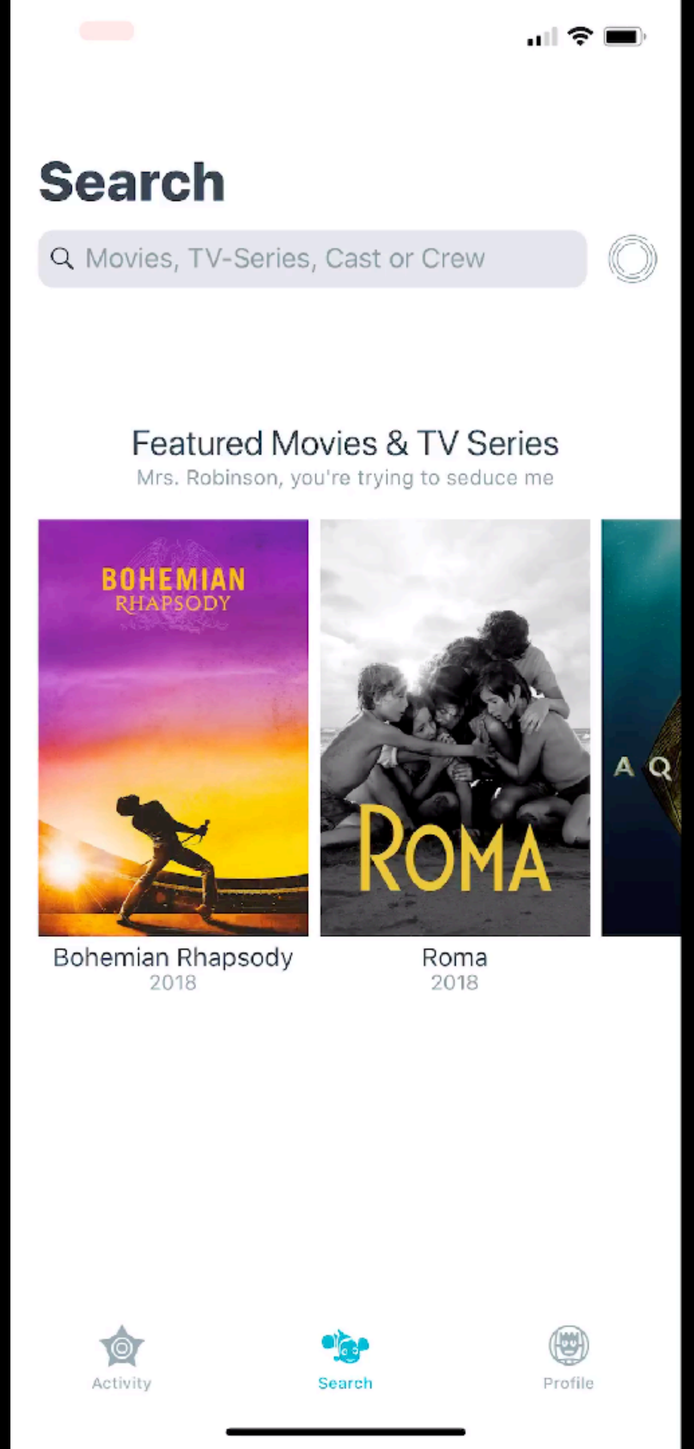


More similar

Feature Vector Demo

Vuu

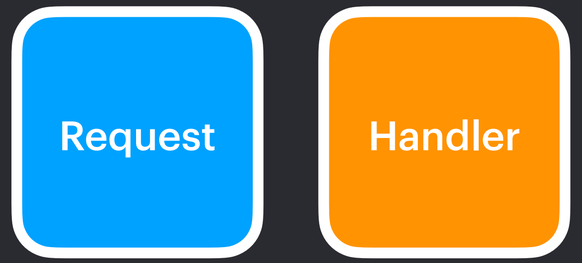
Ålesund, NO



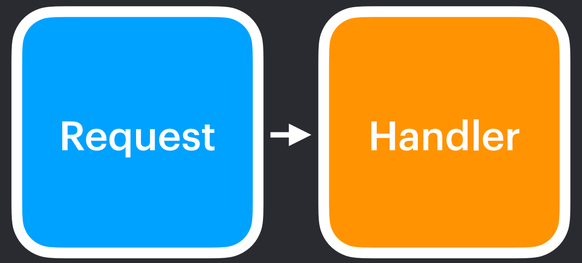
```
// Create the Vision request
let request = VNGenerateImageFeaturePrintRequest()
```

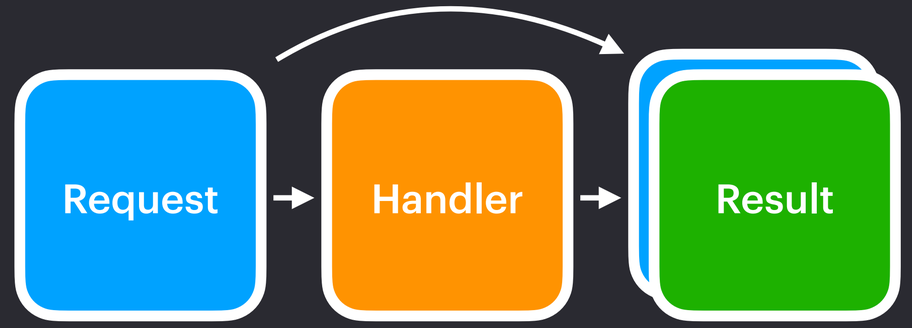
```
// Create the Vision request
let request = VNGenerateImageFeaturePrintRequest()

// Configure the request
request.imageCropAndScaleOption = .centerCrop
```

```swift
// Create the Vision request
let request = VNGenerateImageFeaturePrintRequest()

// Configure the request
request.imageCropAndScaleOption = .centerCrop

// Create the request handler
let requestHandler = VNImageRequestHandler(cvPixelBuffer: image, options: [:])
```



```swift
// Create the Vision request
let request = VNGenerateImageFeaturePrintRequest()

// Configure the request
request.imageCropAndScaleOption = .centerCrop

// Create the request handler
let requestHandler = VNImageRequestHandler(cvPixelBuffer: image, options: [:])

// Perform the request
try? requestHandler.perform([request])
```






```
// Create the Vision request
let request = VNGenerateImageFeaturePrintRequest()

// Configure the request
request.imageCropAndScaleOption = .centerCrop

// Create the request handler
let requestHandler = VNImageRequestHandler(cvPixelBuffer: image, options: [:])

// Perform the request
try? requestHandler.perform([request])

// Check results
if let results = request.results as? [VNFeaturePrintObservation],
   let result = results.first {
    // result.type will be set to .float or .double
    // result.data will be the feature vector represented as raw Data bytes
}
```

```
request.imageCropAndScaleOption
```

```
request.imageCropAndScaleOption = .scaleFill
```

```
request.imageCropAndScaleOption = .centerCrop
```

```
request.imageCropAndScaleOption = .scaleFit
```

Feature Vectors Comparison in the App

Feature Vectors Comparison in the App

It’s not in the app

Feature Vectors Comparison in the App

It’s not in the app because:

1. Requires determining a similarity threshold

Feature Vectors Comparison in the App

It’s not in the app because:

1. Requires determining a similarity threshold
2. Depends on whether Apple’s model can discriminate well between faces

Feature Vectors Comparison in the App

It’s not in the app because:

1. Requires determining a similarity threshold
2. Depends on whether Apple’s model can discriminate well between faces
3. There’s a simpler solution

Crushing Your Head
The App

Hand Pose Detector

Face Detector

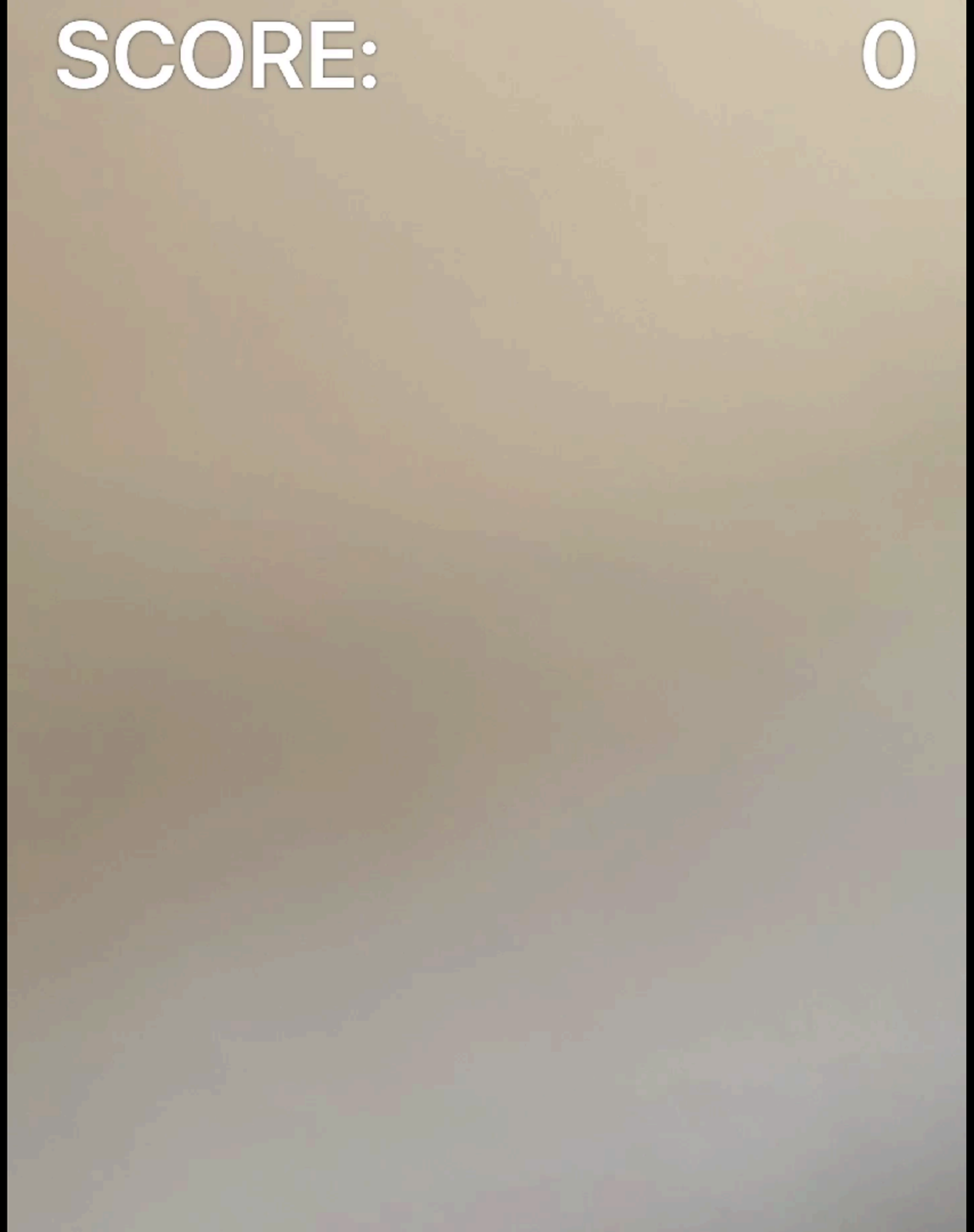
Comparing Feature Vectors

Object Tracking

BONUS:

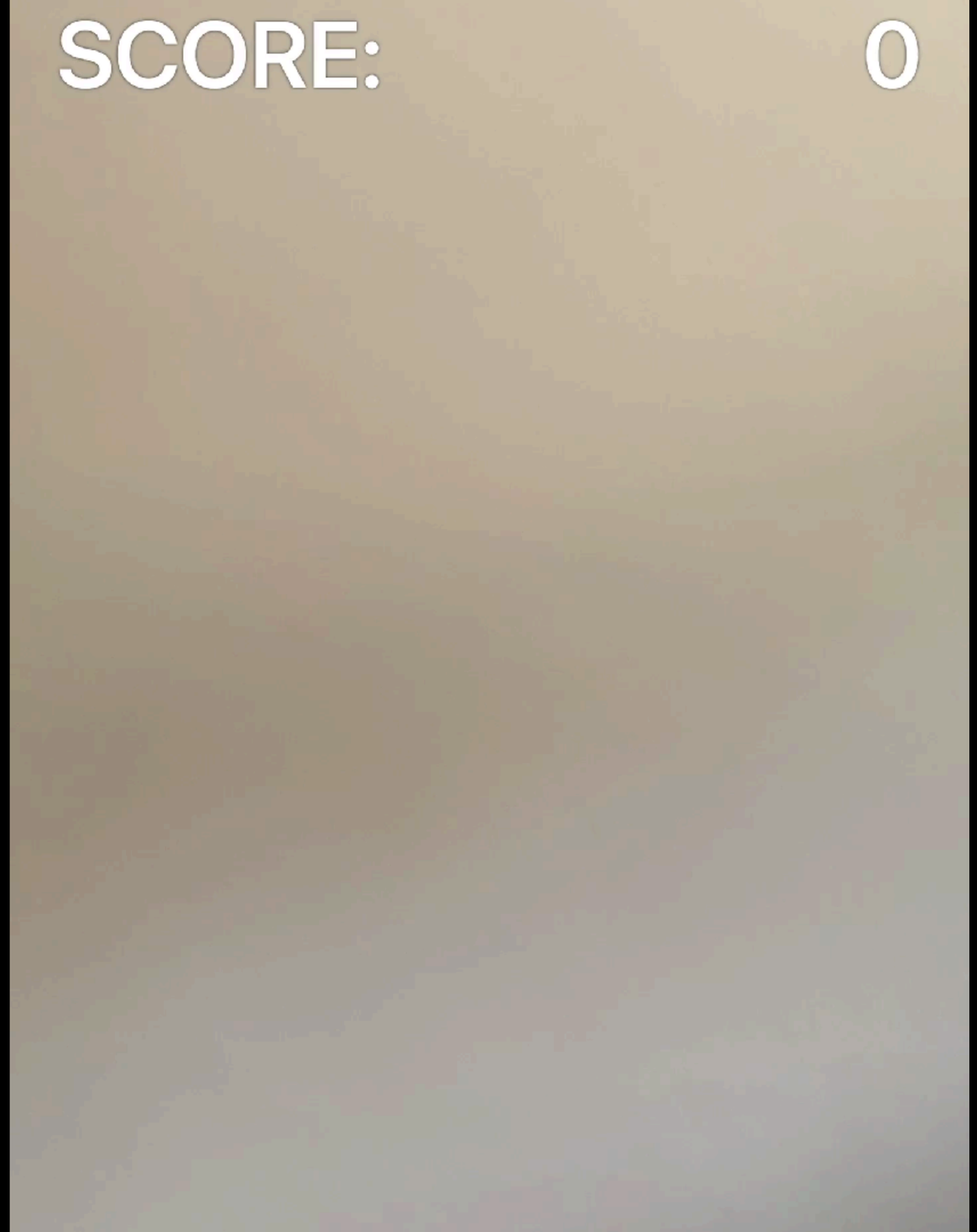
Custom Classifier

Object Tracking in the App



Object Tracking in the App

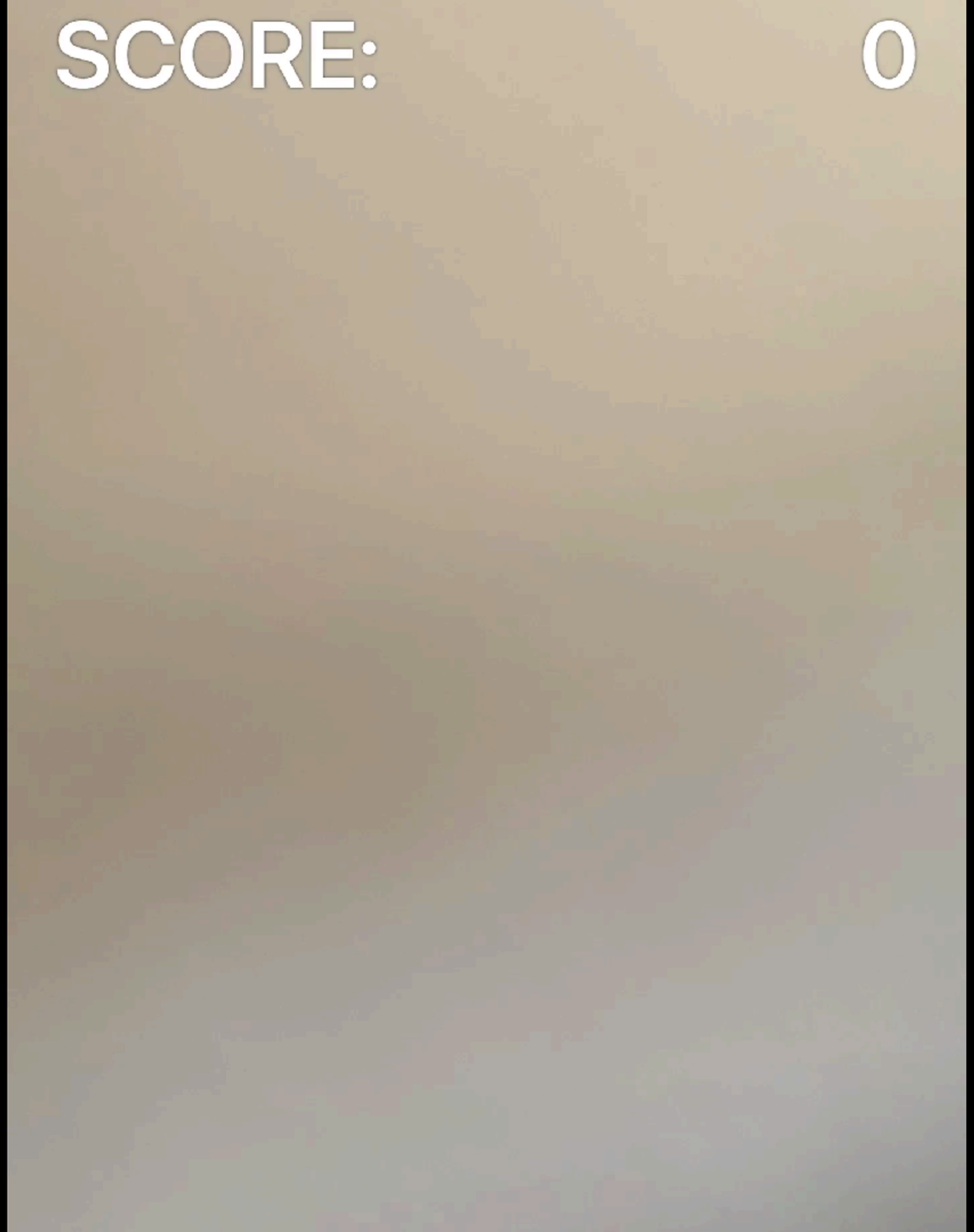
Each detected object has a UUID



Object Tracking in the App

Each detected object has a UUID

Track heads across frames

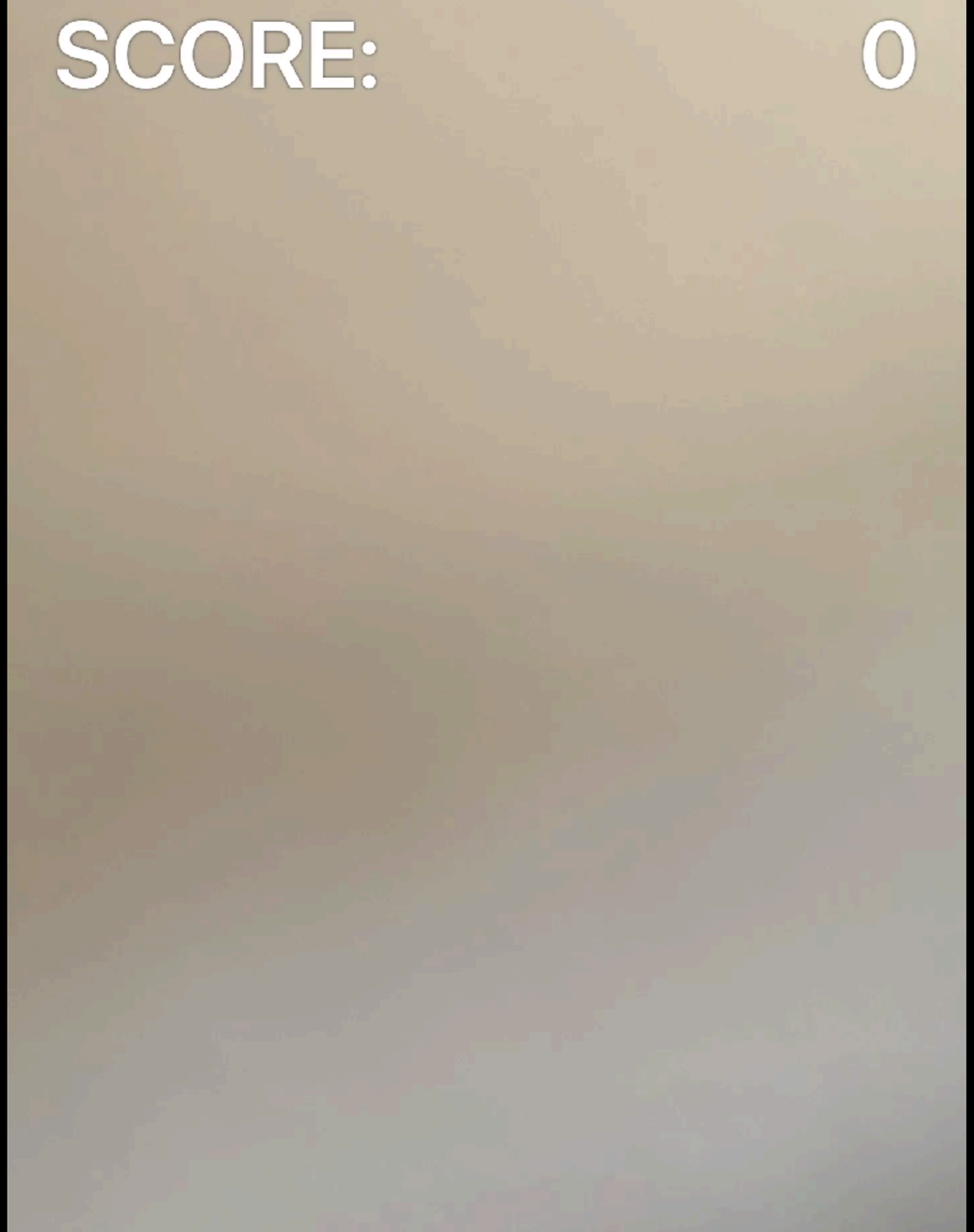


Object Tracking in the App

Each detected object has a UUID

Track heads across frames

Useful for preventing MHC



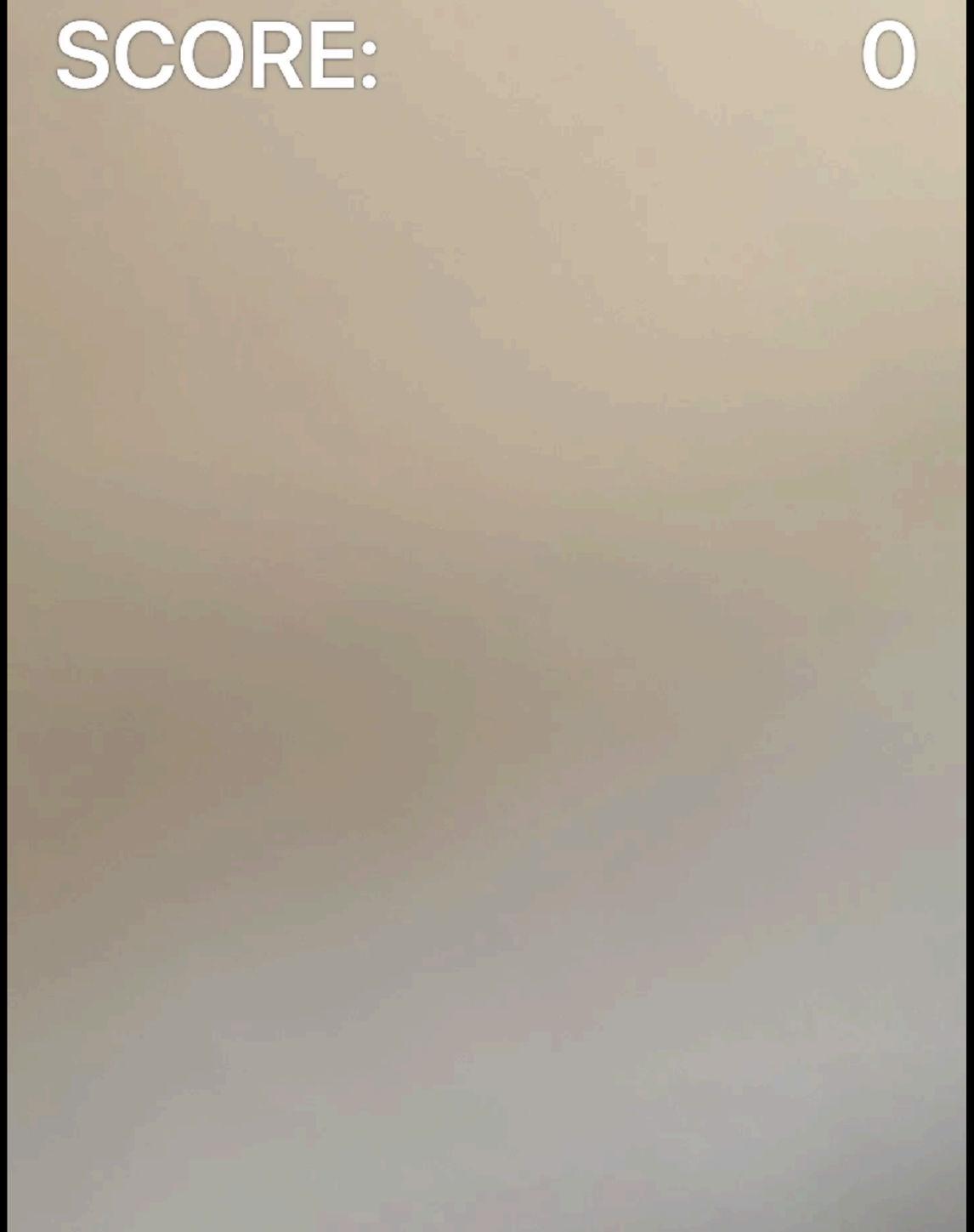
Object Tracking in the App

Each detected object has a UUID

Track heads across frames

Useful for preventing MHC*

* Multiple Head Crushings



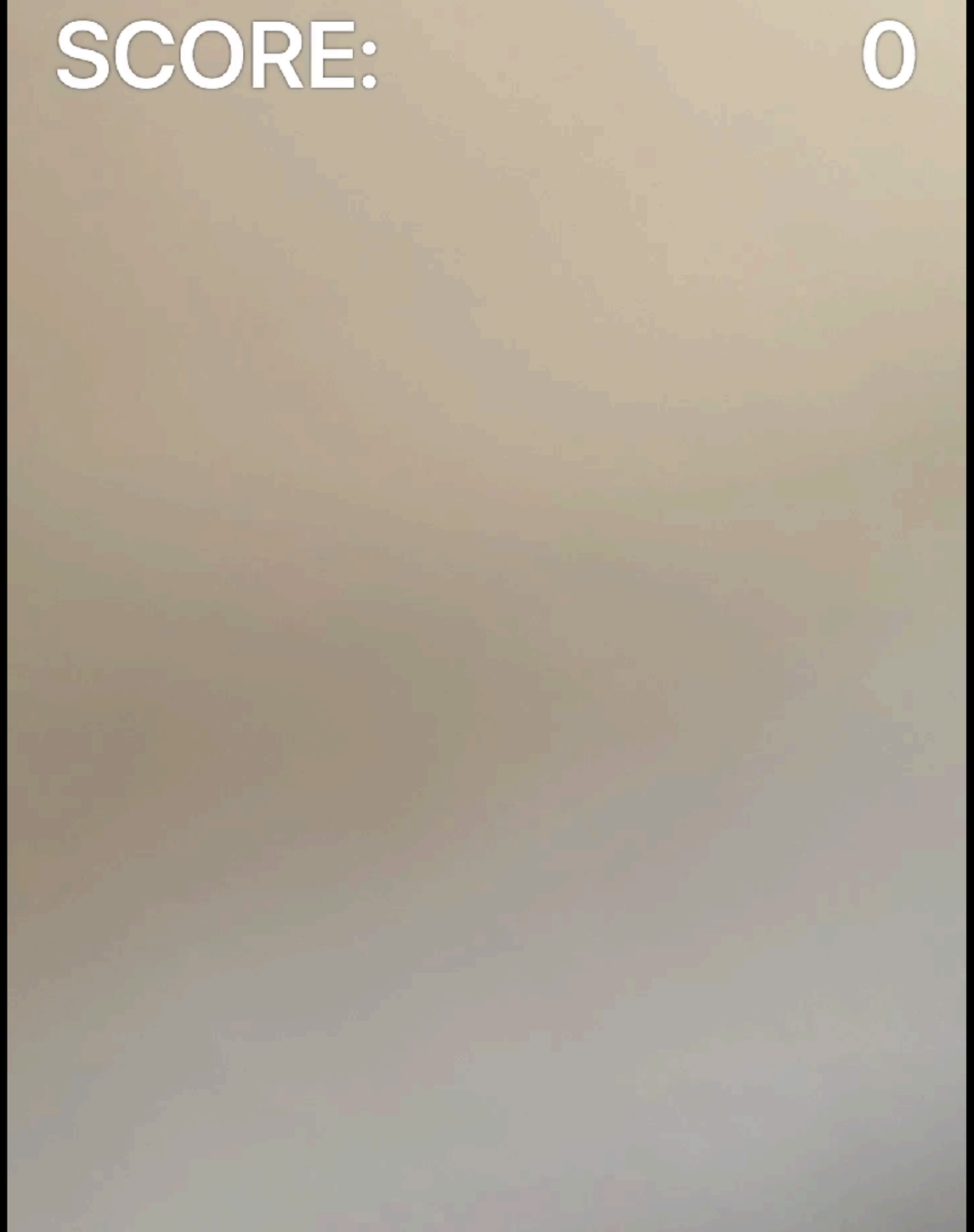
Object Tracking in the App

Each detected object has a UUID

Track heads across frames

Useful for preventing MHC* 🎉

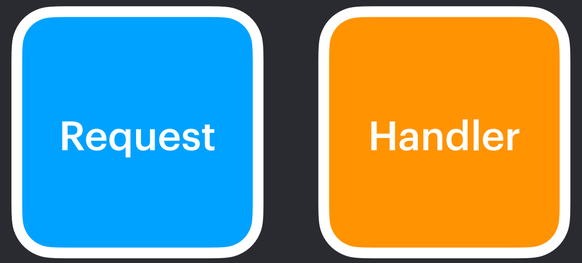
* Multiple Head Crushings



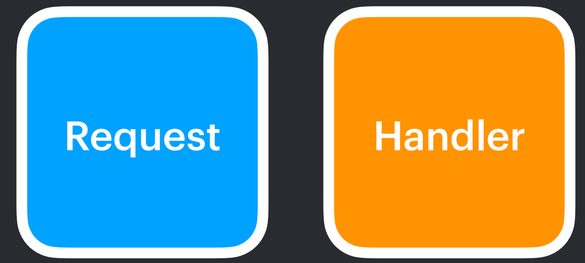
```
// Create array of Vision requests
var requests = [VNRequest]()
```

```
// Create array of Vision requests
var requests = [VNRequest]()

// create a request for every head to be tracked
for head in self.trackedHeads {
    requests.append(VNTrackObjectRequest(detectedObjectObservation: head.value))
}
```

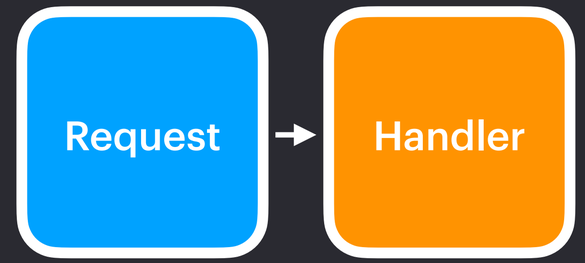

```
// Create array of Vision requests
var requests = [VNRequest]()

// create a request for every head to be tracked
for head in self.trackedHeads {
    requests.append(VNTrackObjectRequest(detectedObjectObservation: head.value))
}

if requests.isEmpty() {
    // detect fresh faces
    requests.append(VNDetectFaceRectanglesRequest())



}
```

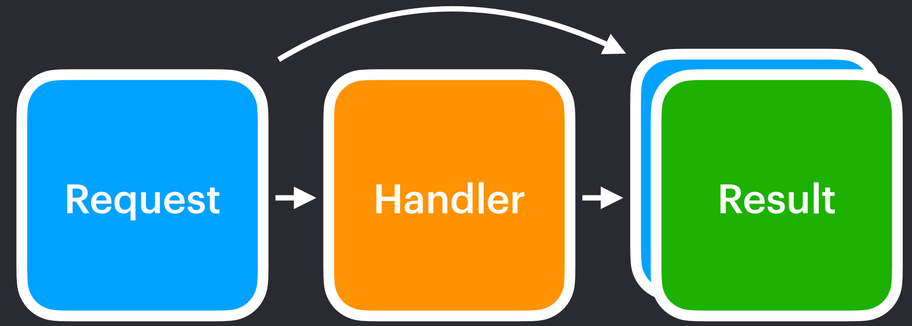



```
// Create array of Vision requests
var requests = [VNRequest]()

// create a request for every head to be tracked
for head in self.trackedHeads {
    requests.append(VNTrackObjectRequest(detectedObjectObservation: head.value))
}

if requests.isEmpty() {
    // detect fresh faces
    requests.append(VNDetectFaceRectanglesRequest())

    // reset the sequence handler
    self.handler = VNSequenceRequestHandler()
}
```

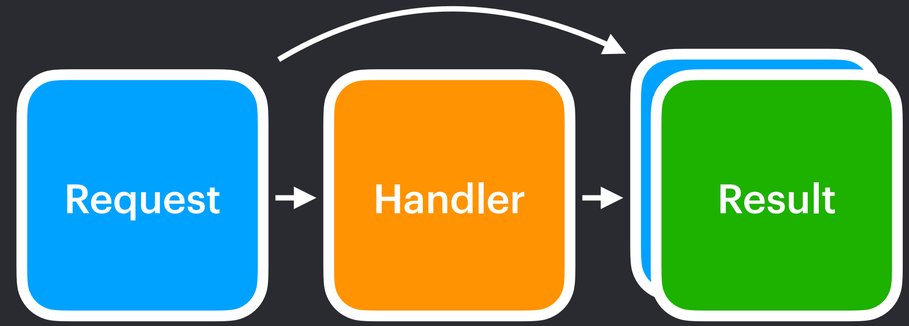

Request
Handler

```
...


// Perform the request
try? handler.perform(requests, on: image)
```






```
...


// Perform the request
try? handler.perform(requests, on: image)


// iterate through the completed requests
for request in requests {
    if let results = request.results as? [VNRecognizedObjectObservation] {


    }
}
```




```
...


// Perform the request
try? handler.perform(requests, on: image)


// iterate through the completed requests
for request in requests {
    if let results = request.results as? [VNRecognizedObjectObservation] {
        // handle a detected/tracked object
        self.updateTrackedHeads(with: results)
    }
}
```

What if Crushing Heads is too gruesome for you?

Smashing Pumpkins?

Crushing Your Head
The App

Hand Pose Detector

Face Detector

Comparing Feature Vectors

Object Tracking

BONUS:

Custom Classifier

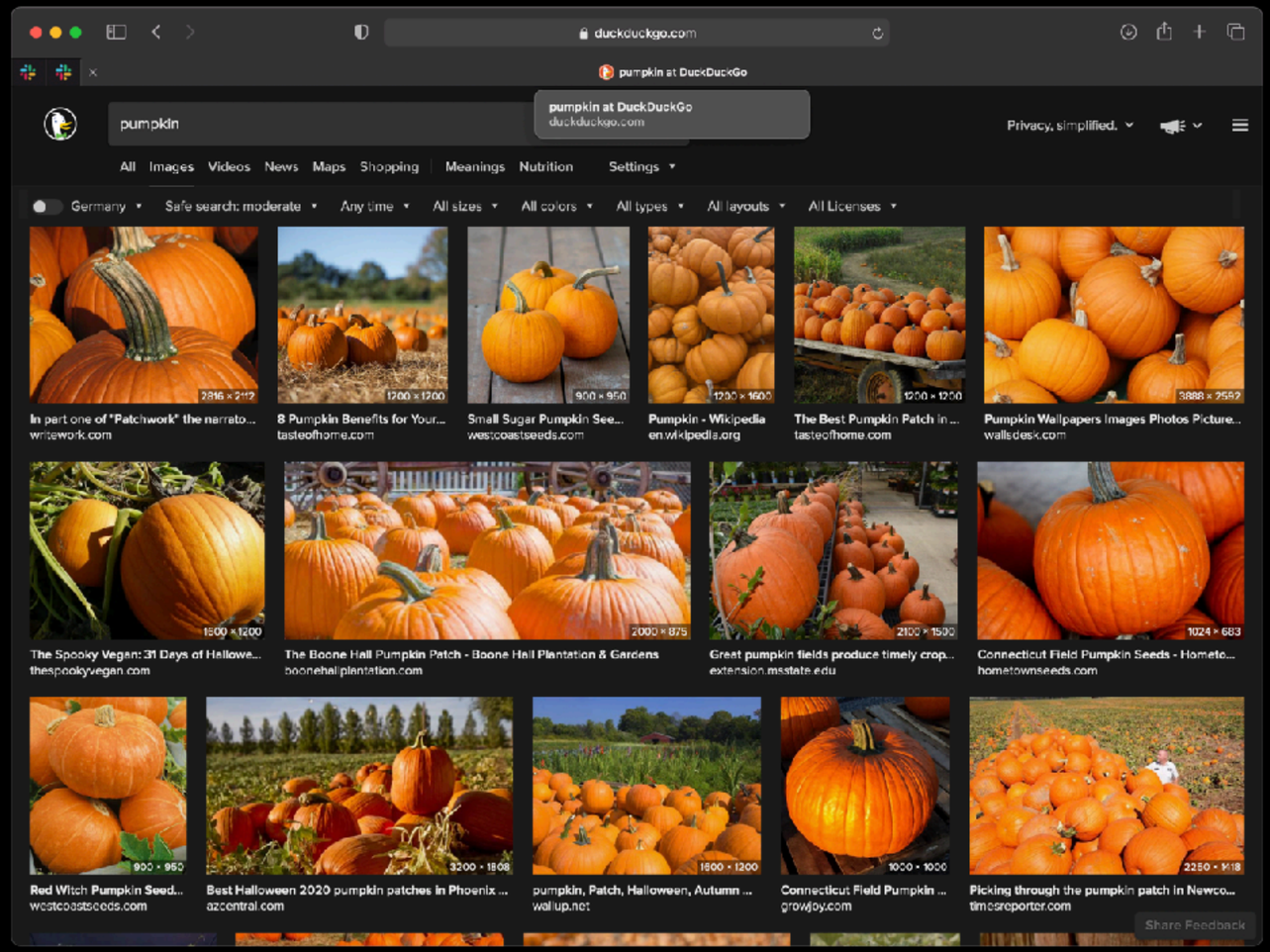
Smashing Pumpkins

Train a custom CoreML classifier

Smashing Pumpkins

Train a custom CoreML classifier

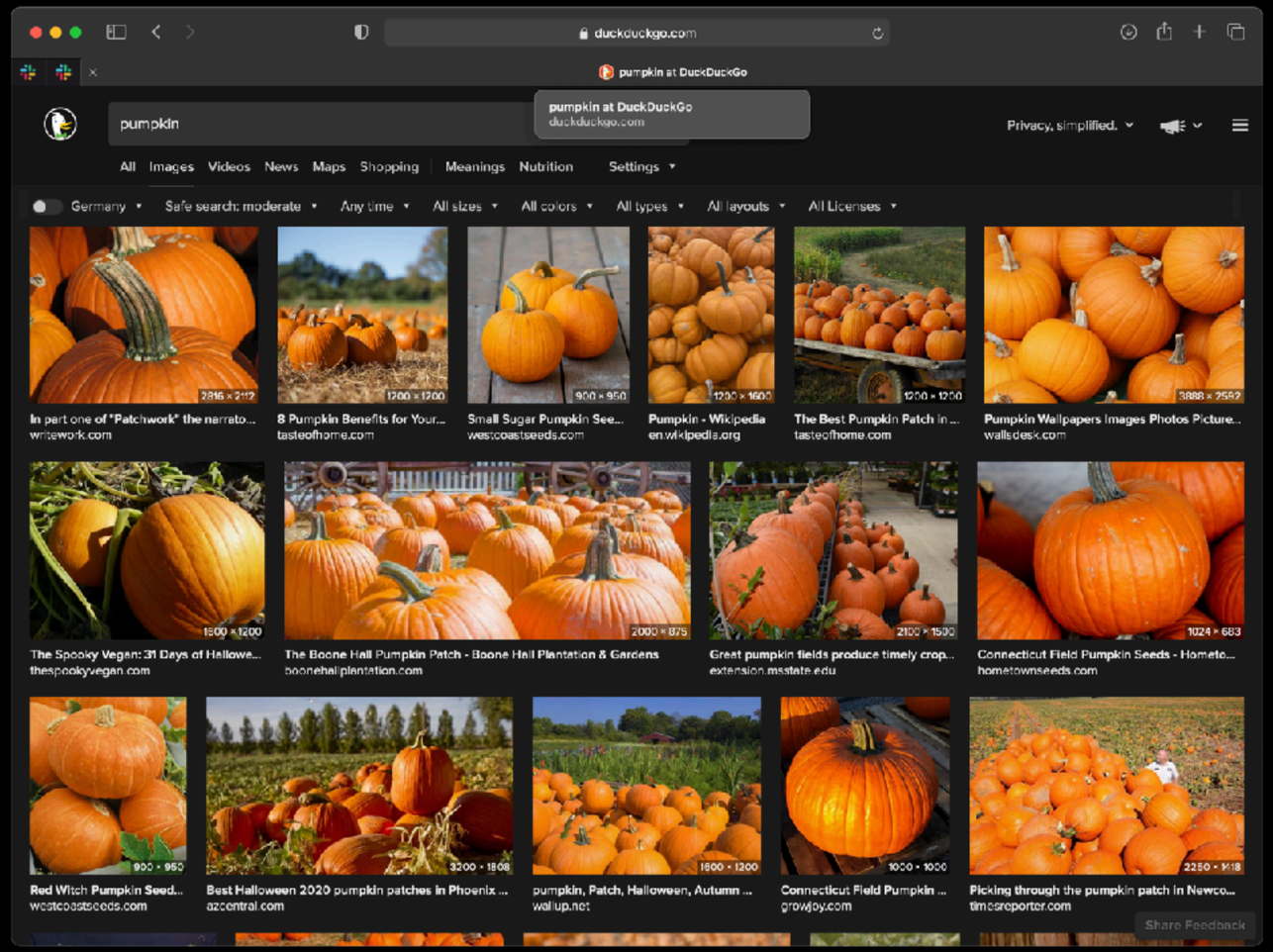
Collect images of pumpkins and not pumpkins



Smashing Pumpkins

Train a custom CoreML classifier

Collect images of pumpkins and not pumpkins*



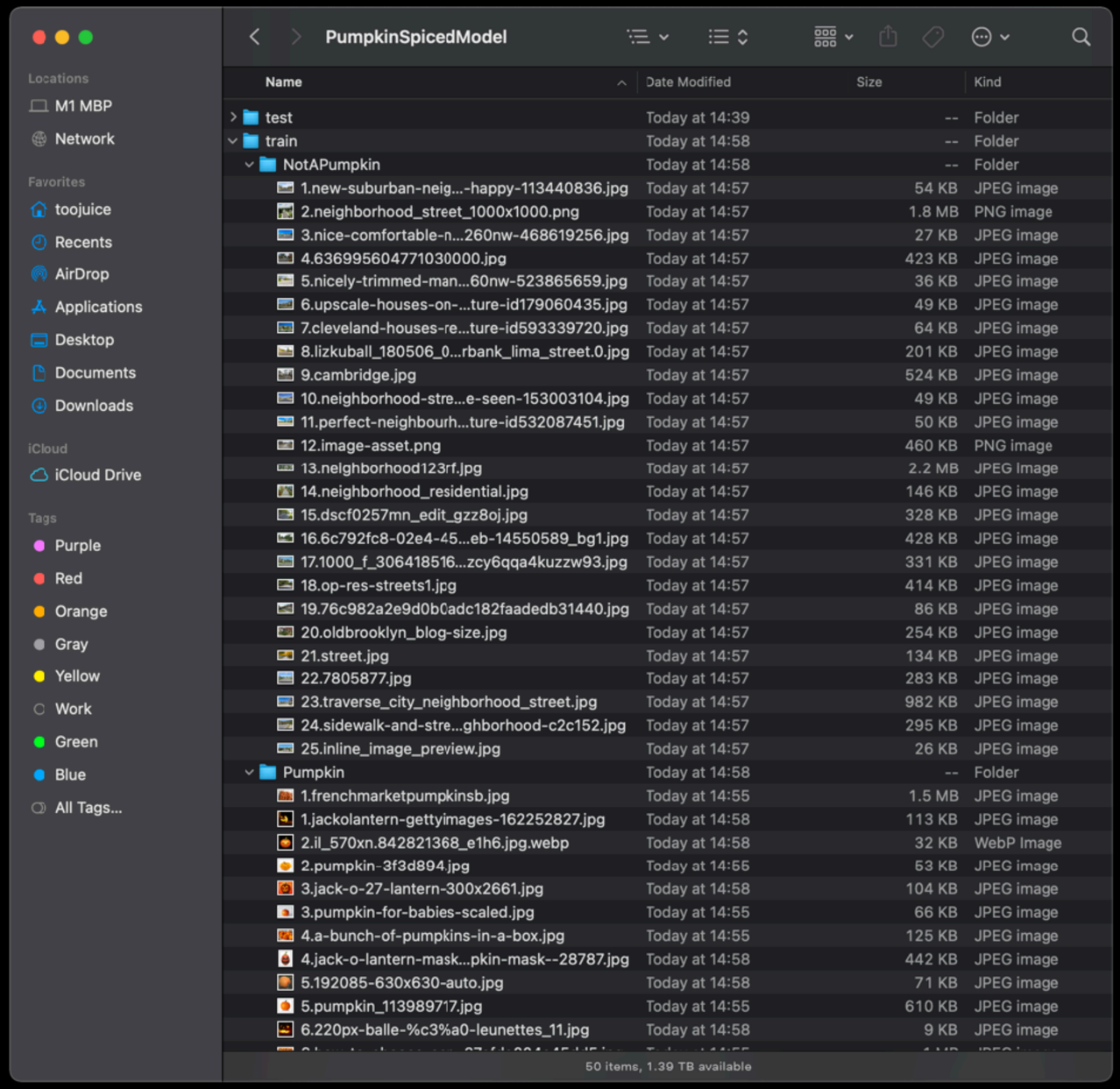
* Be mindful of image licenses

Smashing Pumpkins

Train a custom CoreML classifier

Collect images of pumpkins and not pumpkins*

Organize the folder structure



* Be mindful of image licenses

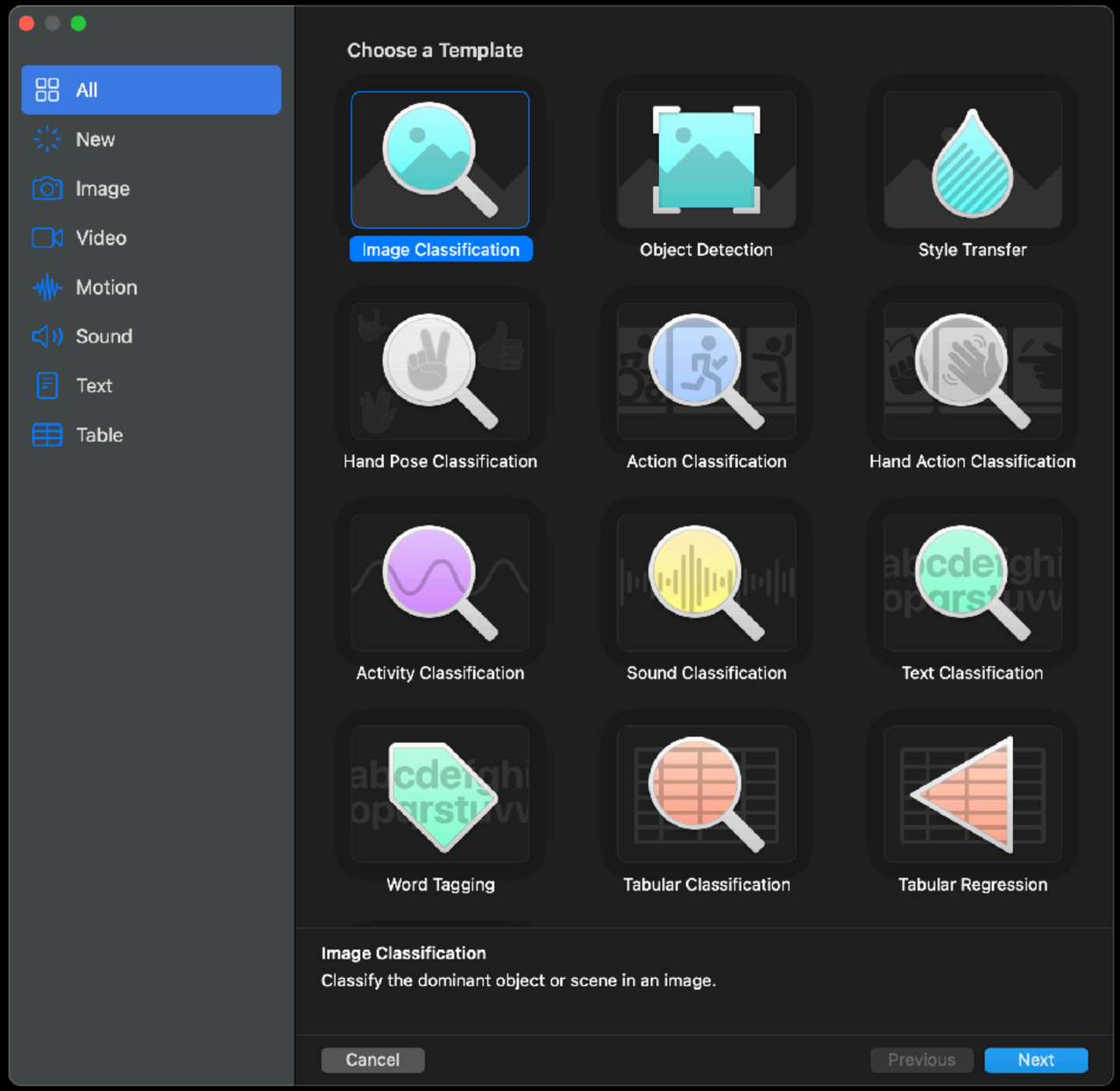
Smashing Pumpkins

Train a custom CoreML classifier

Collect images of pumpkins and not pumpkins*

Organize the folder structure

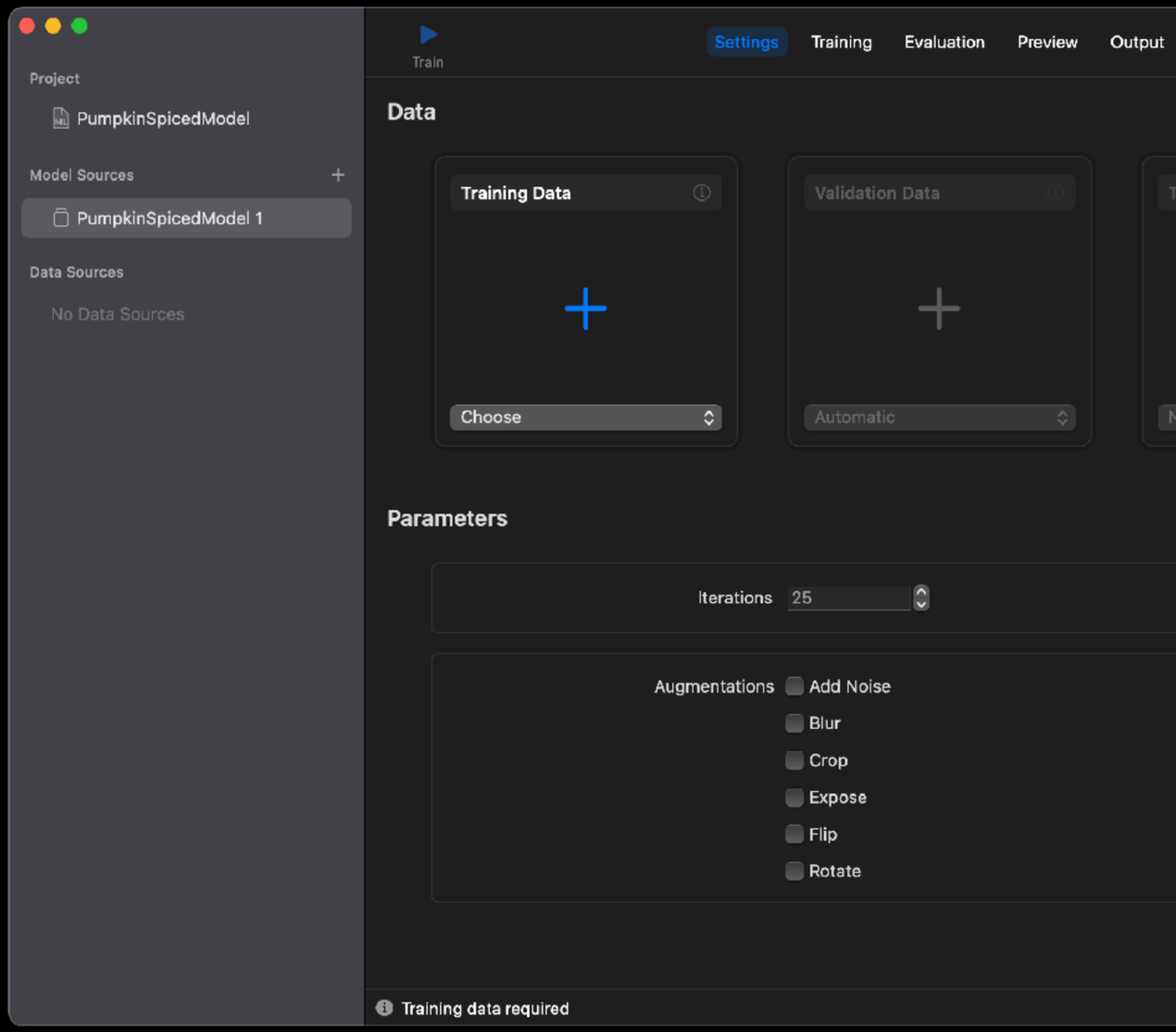
Create an image classification project in CreateML



* Be mindful of image licenses

Smashing Pumpkins

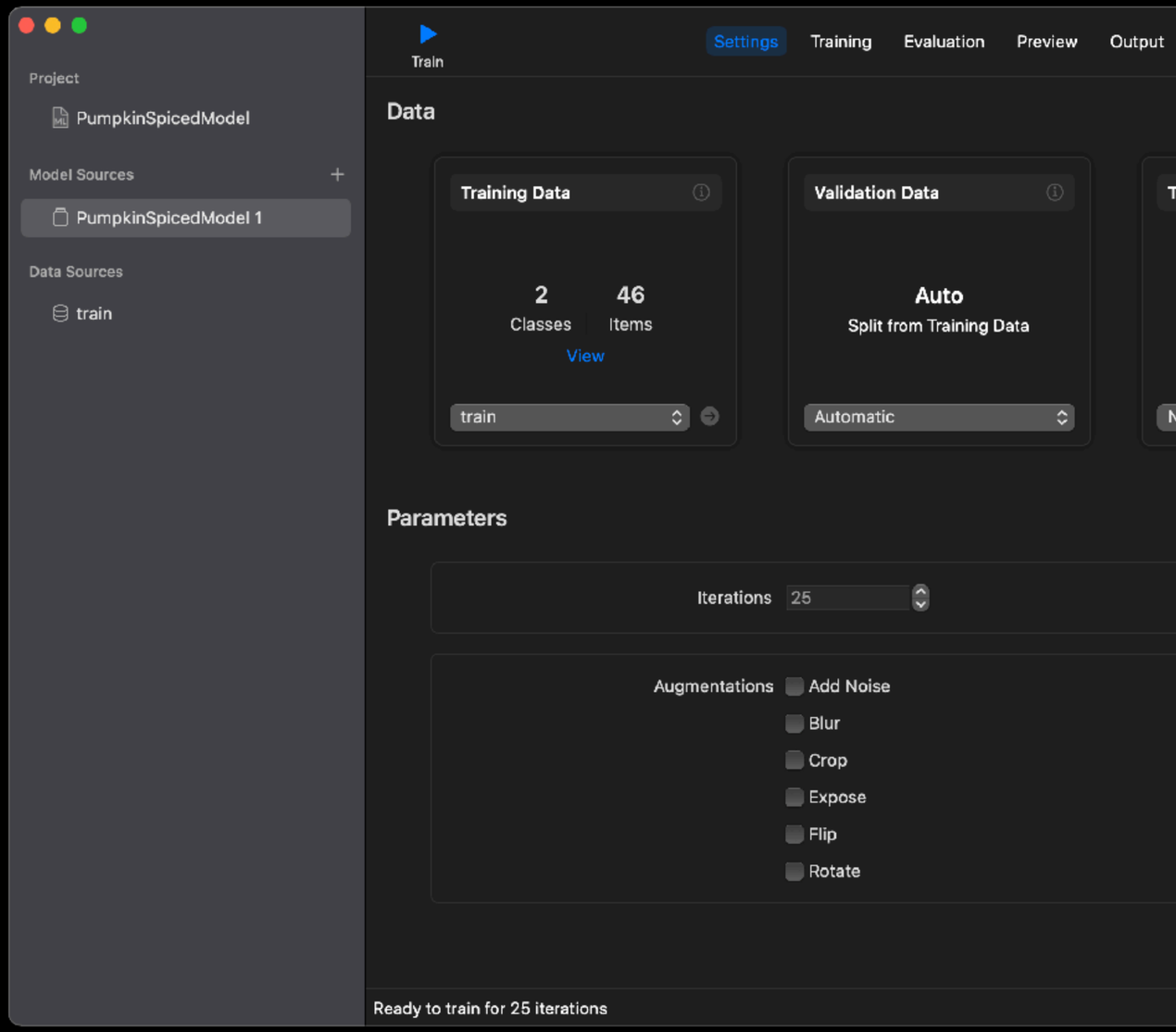
Train a custom CoreML classifier



Smashing Pumpkins

Train a custom CoreML classifier

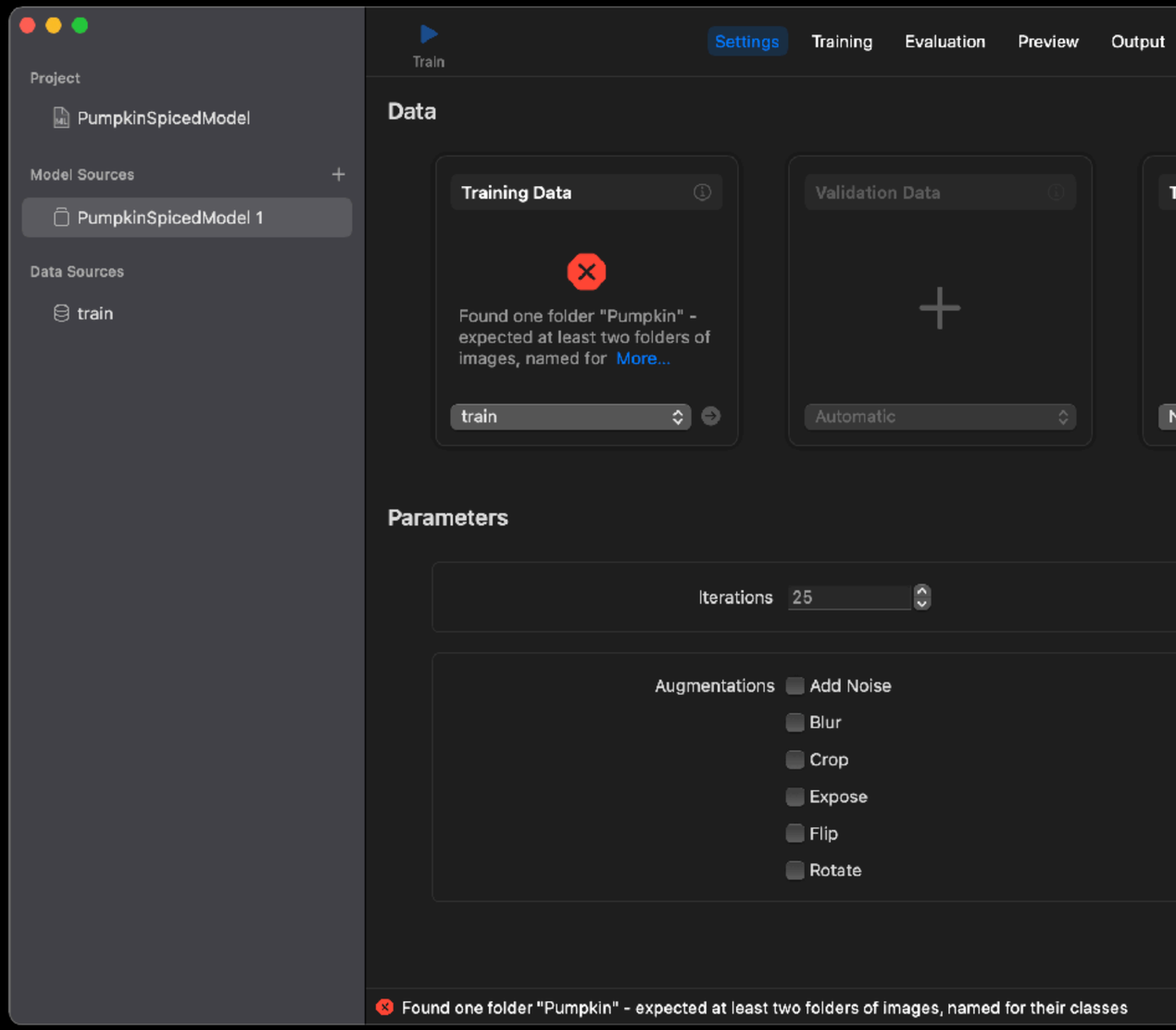
Add the training folder to your project



Smashing Pumpkins

Train a custom CoreML classifier

Add the training folder to your project

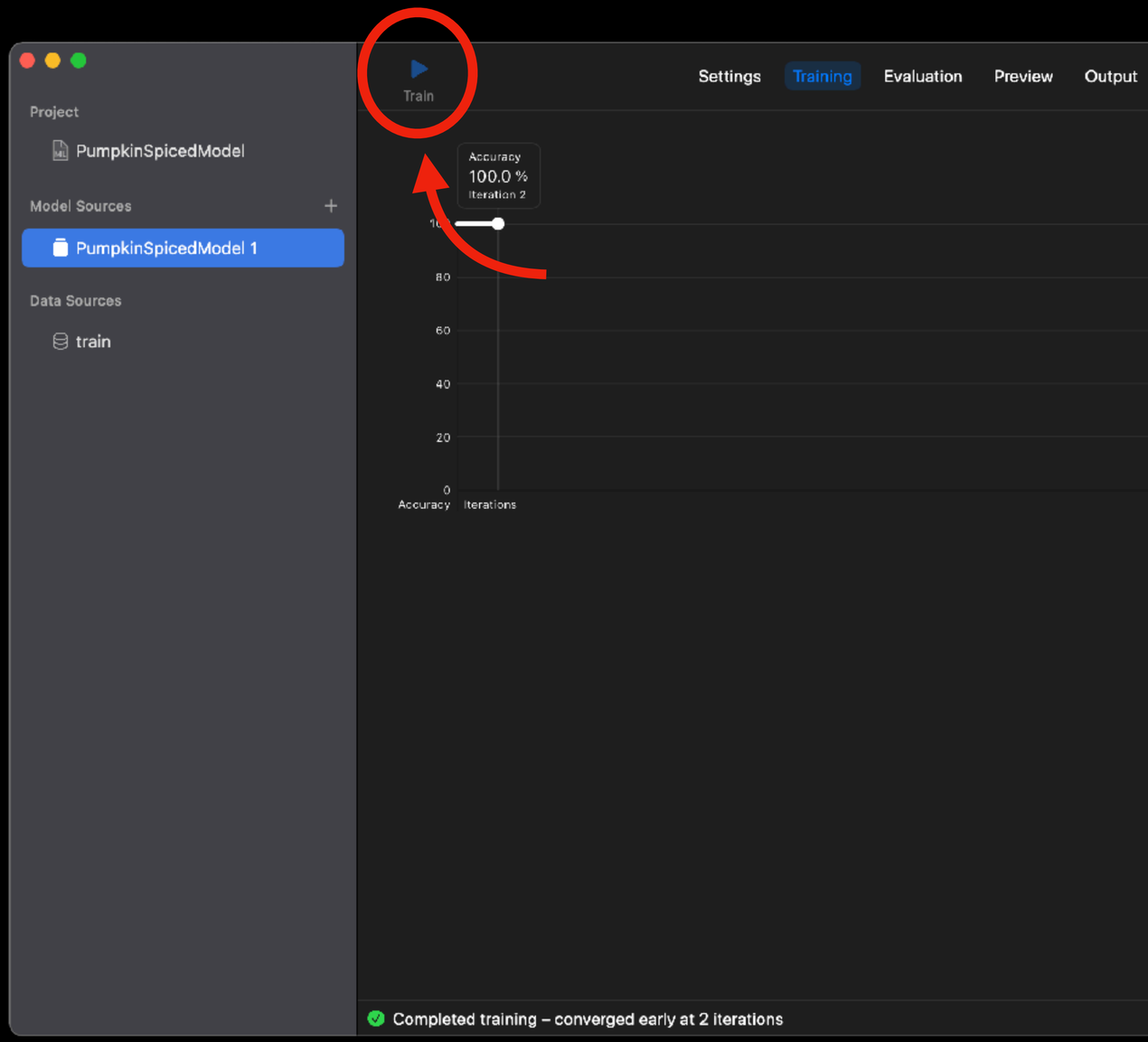


Smashing Pumpkins

Train a custom CoreML classifier

Add the training folder to your project

Train



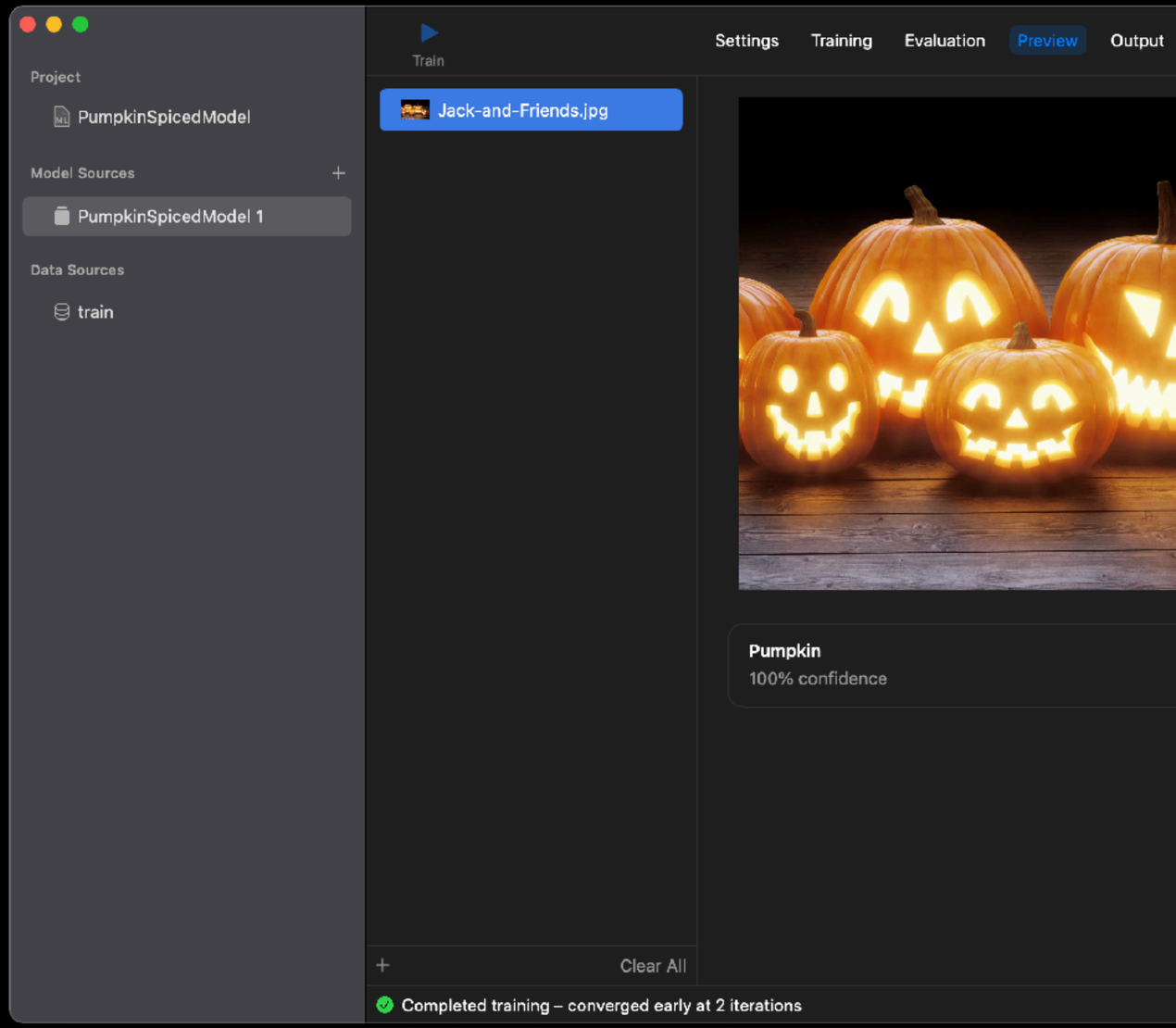
Smashing Pumpkins

Train a custom CoreML classifier

Add the training folder to your project

Train

Evaluate or spot check the model



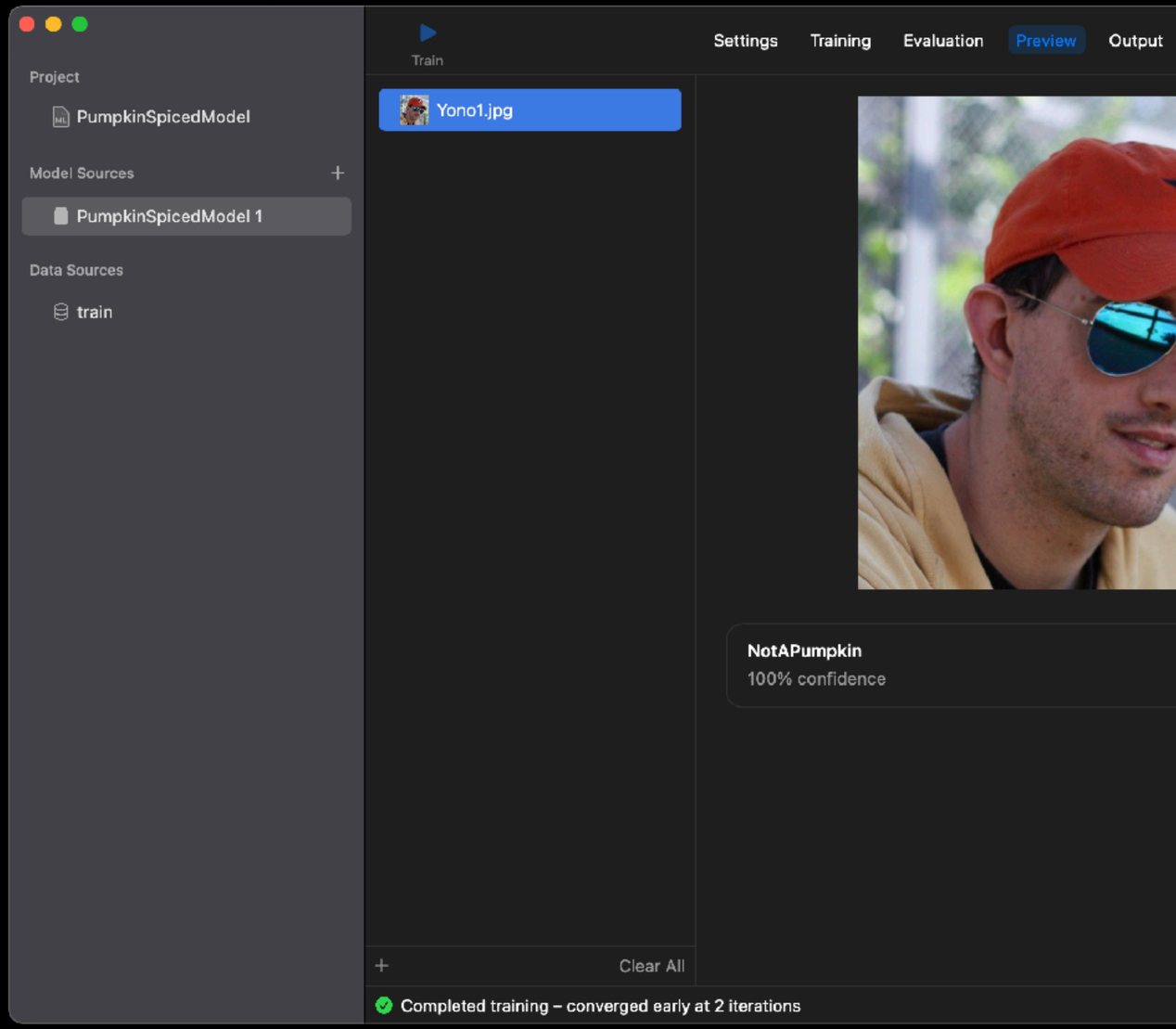
Smashing Pumpkins

Train a custom CoreML classifier

Add the training folder to your project

Train

Evaluate or spot check the model



Smashing Pumpkins

Train a custom CoreML classifier

Add the training folder to your project

Train

Evaluate or spot check the model

Export the CoreML model

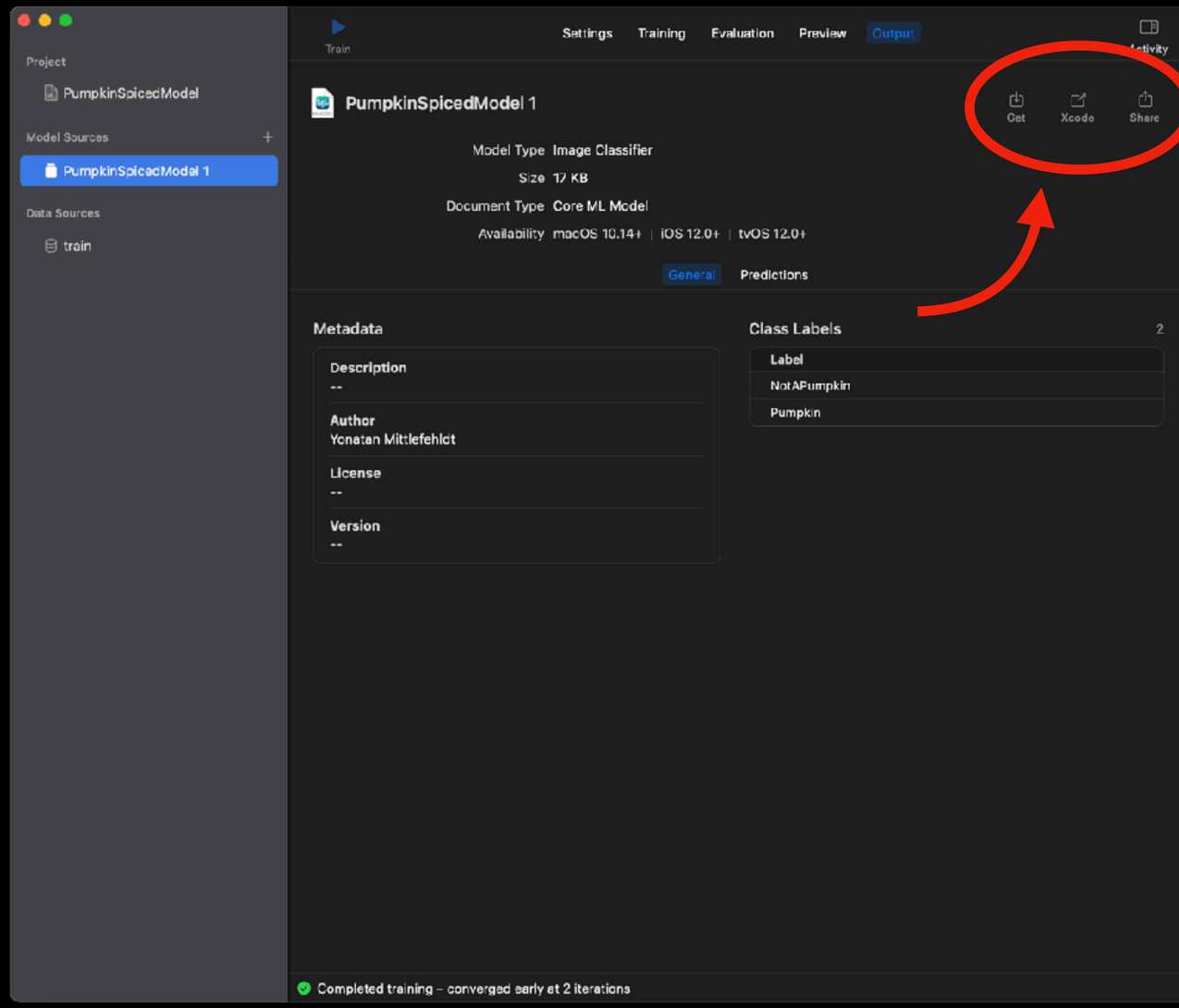


Image Classifiers

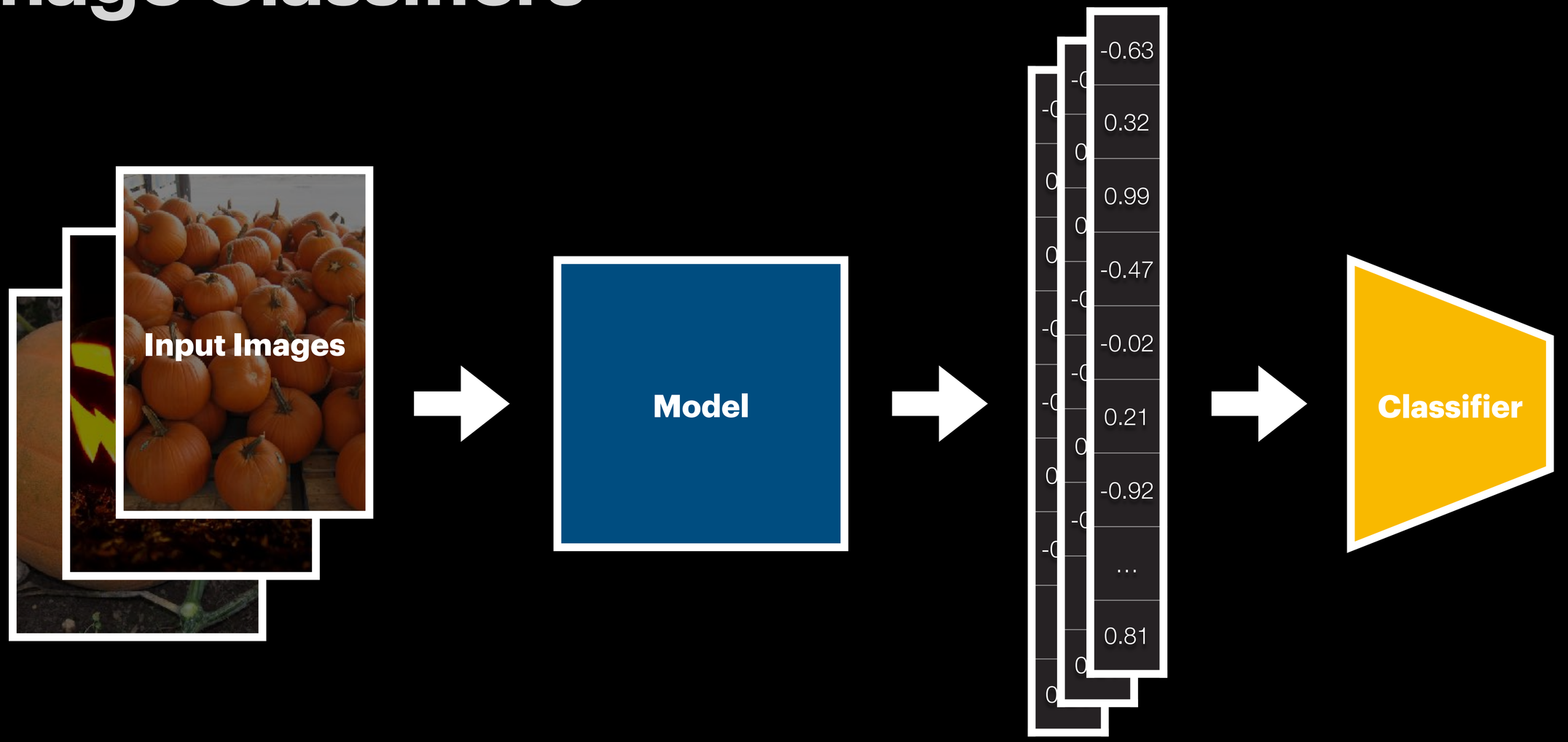
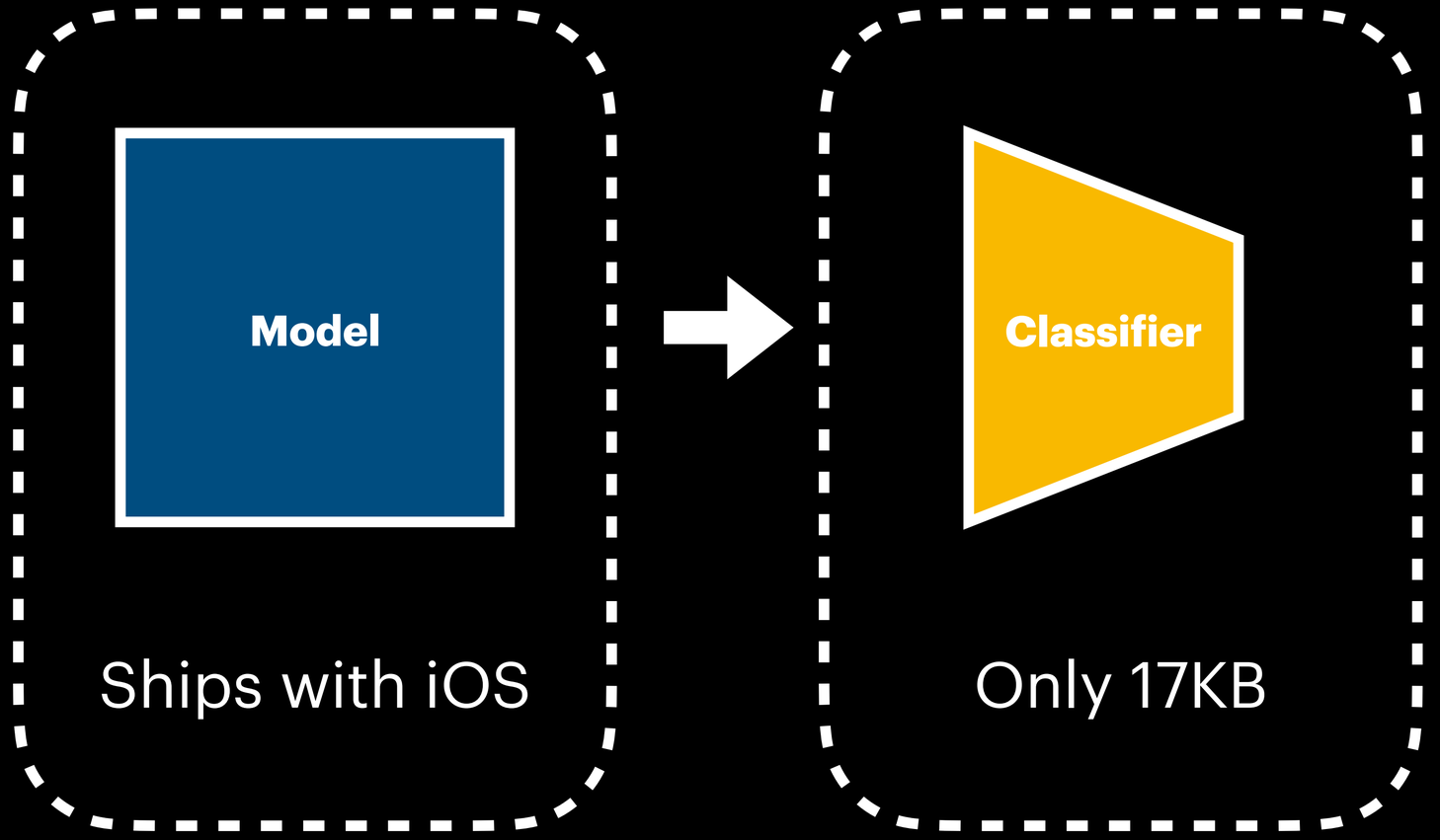


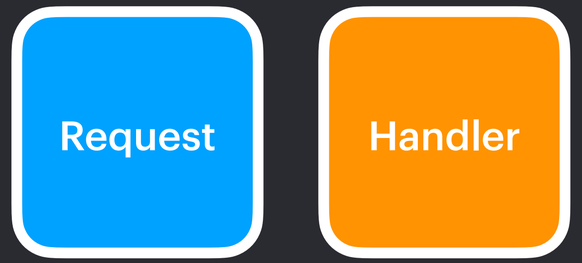
Image Classifiers



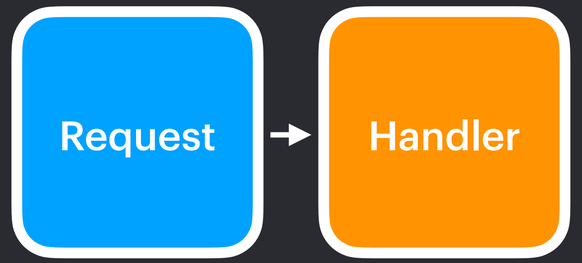
```swift
// Create the Vision request
let request = VNCoreMLRequest(model: model)
```

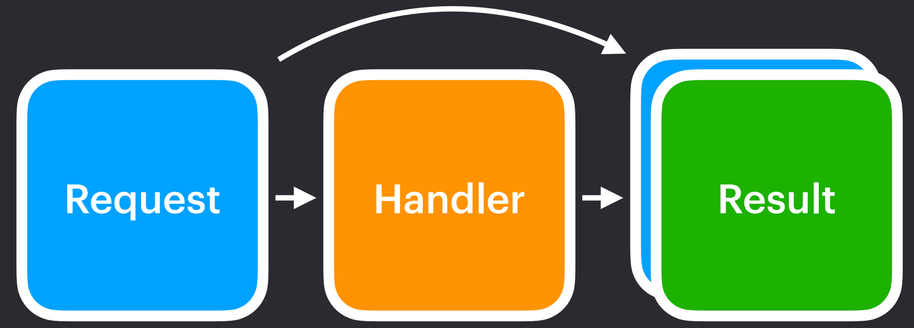
```
// Create the Vision request
let request = VNCoreMLRequest(model: model)

// Configure the request
request.regionOfInterest = potentialPumpkinRect
```

```swift
// Create the Vision request
let request = VNCoreMLRequest(model: model)

// Configure the request
request.regionOfInterest = potentialPumpkinRect

// Create the request handler
let requestHandler = VNImageRequestHandler(cvPixelBuffer: image, options: [:])
```

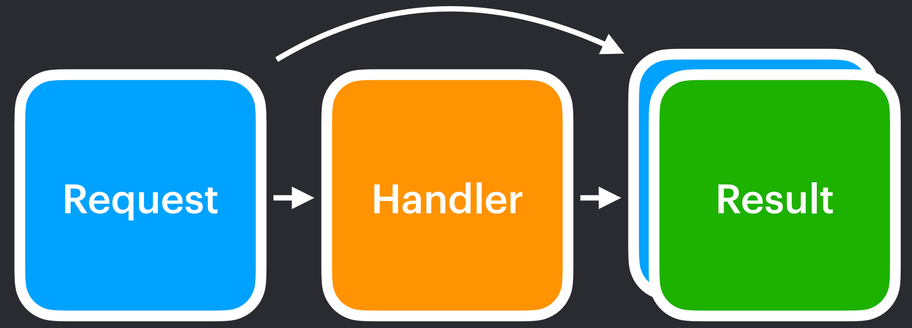


```swift
// Create the Vision request
let request = VNCoreMLRequest(model: model)

// Configure the request
request.regionOfInterest = potentialPumpkinRect

// Create the request handler
let requestHandler = VNImageRequestHandler(cvPixelBuffer: image, options: [:])

// Perform the request
try? requestHandler.perform([request])
```

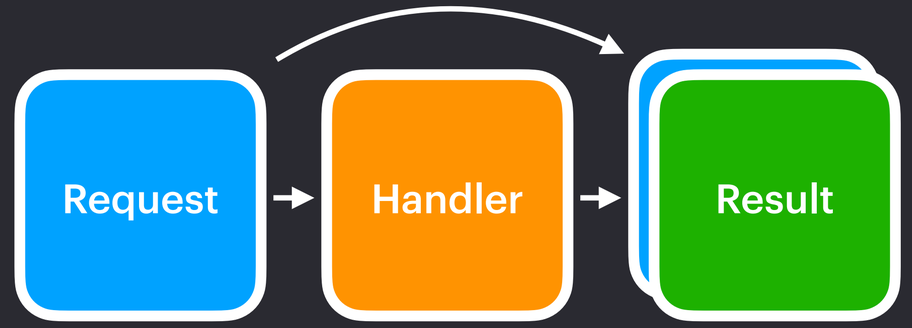





```
// Create the Vision request
let request = VNCoreMLRequest(model: model)

// Configure the request
request.regionOfInterest = potentialPumpkinRect

// Create the request handler
let requestHandler = VNImageRequestHandler(cvPixelBuffer: image, options: [:])

// Perform the request
try? requestHandler.perform([request])

// Check results
if let results = request.results as? [VNClassificationObservation],
   let result = results.first {


}
```




```
// Create the Vision request
let request = VNCoreMLRequest(model: model)

// Configure the request
request.regionOfInterest = potentialPumpkinRect

// Create the request handler
let requestHandler = VNImageRequestHandler(cvPixelBuffer: image, options: [:])

// Perform the request
try? requestHandler.perform([request])

// Check results
if let results = request.results as? [VNClassificationObservation],
   let result = results.first,
   result.identifier == "Pumpkin" && result.confidence > minConfidence {

}
```




```
// Create the Vision request
let request = VNCoreMLRequest(model: model)

// Configure the request
request.regionOfInterest = potentialPumpkinRect

// Create the request handler
let requestHandler = VNImageRequestHandler(cvPixelBuffer: image, options: [:])

// Perform the request
try? requestHandler.perform([request])

// Check results
if let results = request.results as? [VNClassificationObservation],
   let result = results.first,
   result.identifier == "Pumpkin" && result.confidence > minConfidence {
    return Pumpkin(id: result.uuid)
}
```

Crushing Your Head
The App

Hand Pose Detector

Face Detector

Comparing Feature Vectors

Object Tracking

BONUS:

Custom Classifier

Wouldn’t it be great if this could be run on a hands free device...

...some sort of glasses maybe?

https://github.com/yonomitt/Crushing-Your-Head

Thank You

@yonomitt

https://github.com/yonomitt/Crushing-Your-Head