

# 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

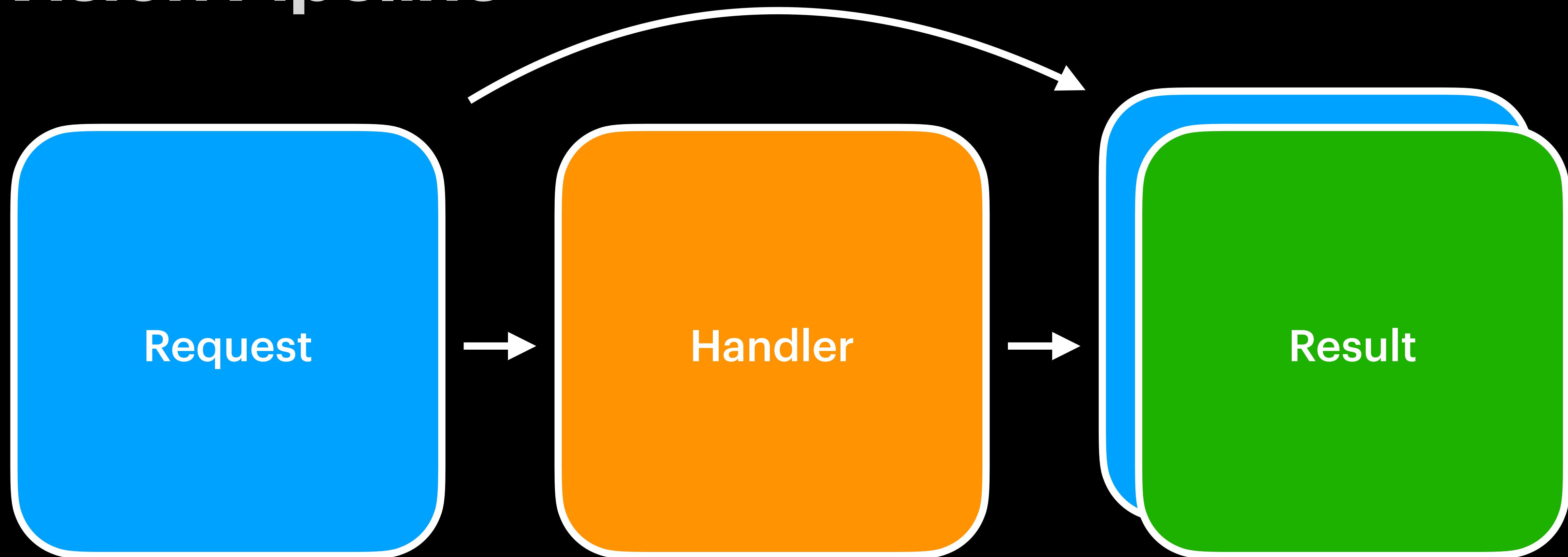
Common computer vision tasks

Optimized to run on iPhones and iPads

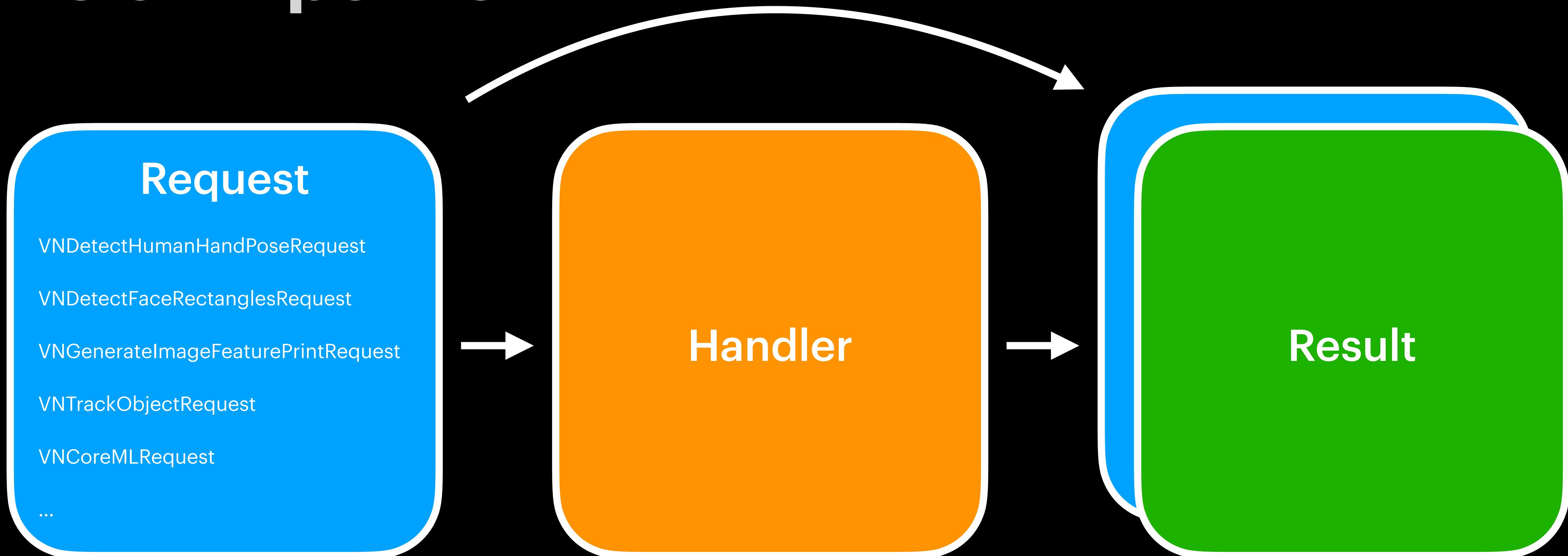
Updated yearly

# Vision Pipeline

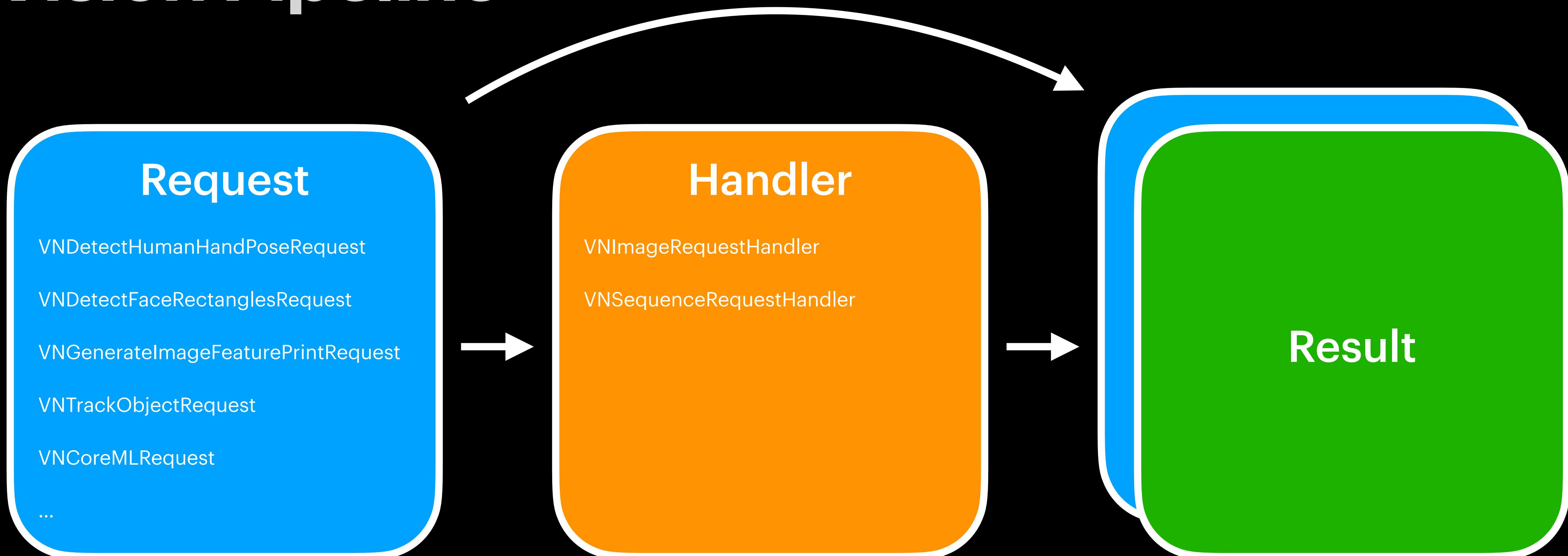
# Vision Pipeline



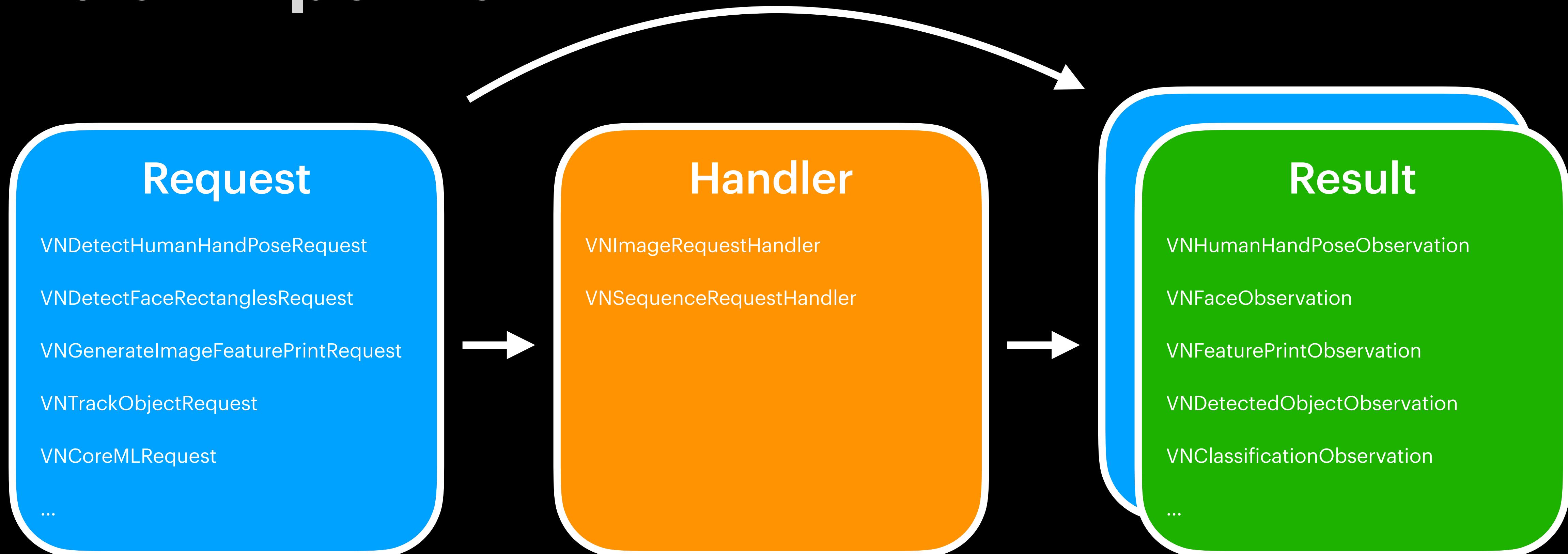
# Vision Pipeline



# Vision Pipeline



# Vision Pipeline



# Vision Tutorials

# Vision Tutorials

## Photo Stacking

[www.raywenderlich.com/3733151-photo-stacking-in-ios-with-vision-and-metal](http://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](http://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](https://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:

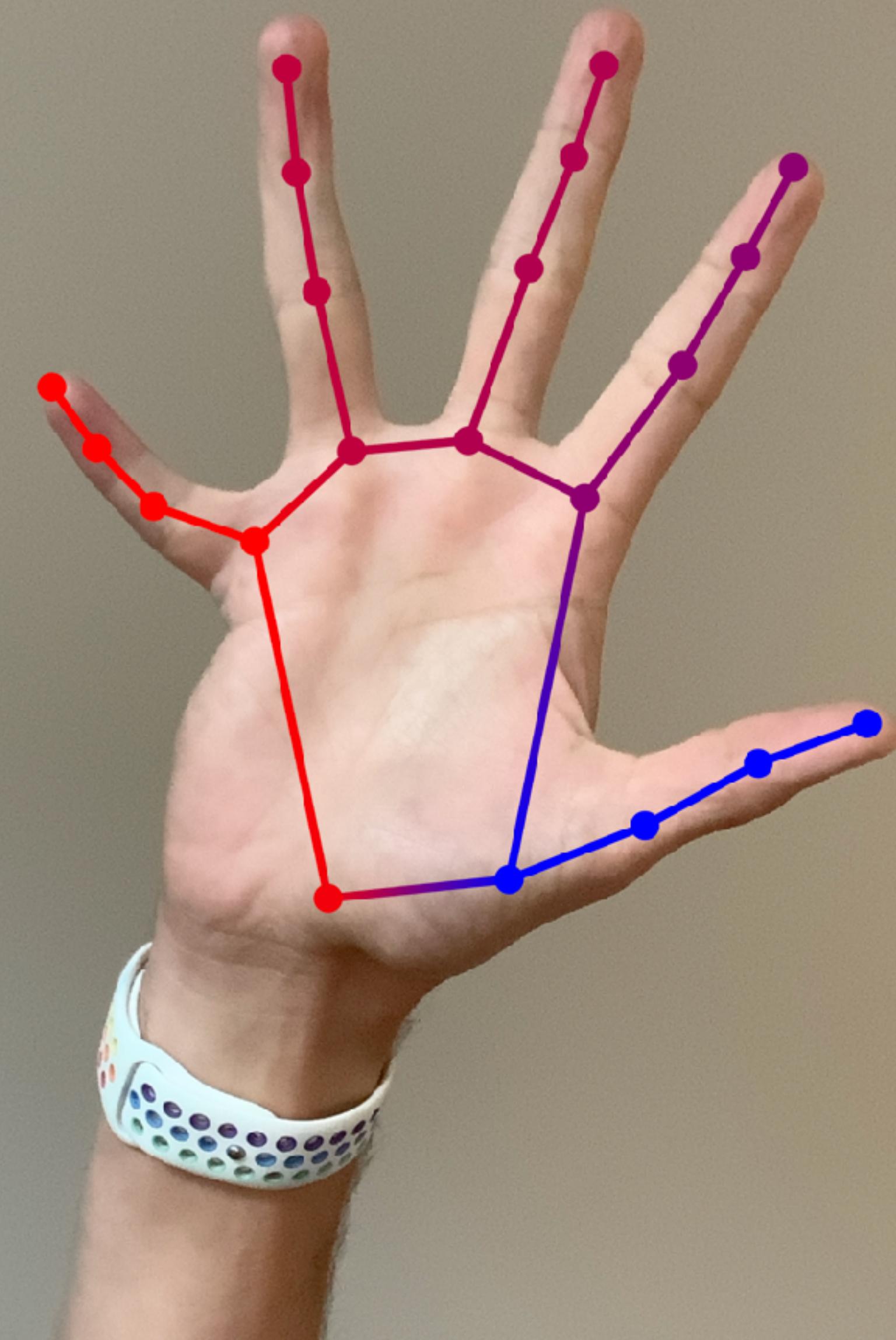
Custom Classifier

# Hand Pose Request

Detects up to 21 points on a hand

4 points per finger and thumb

1 point for the wrist



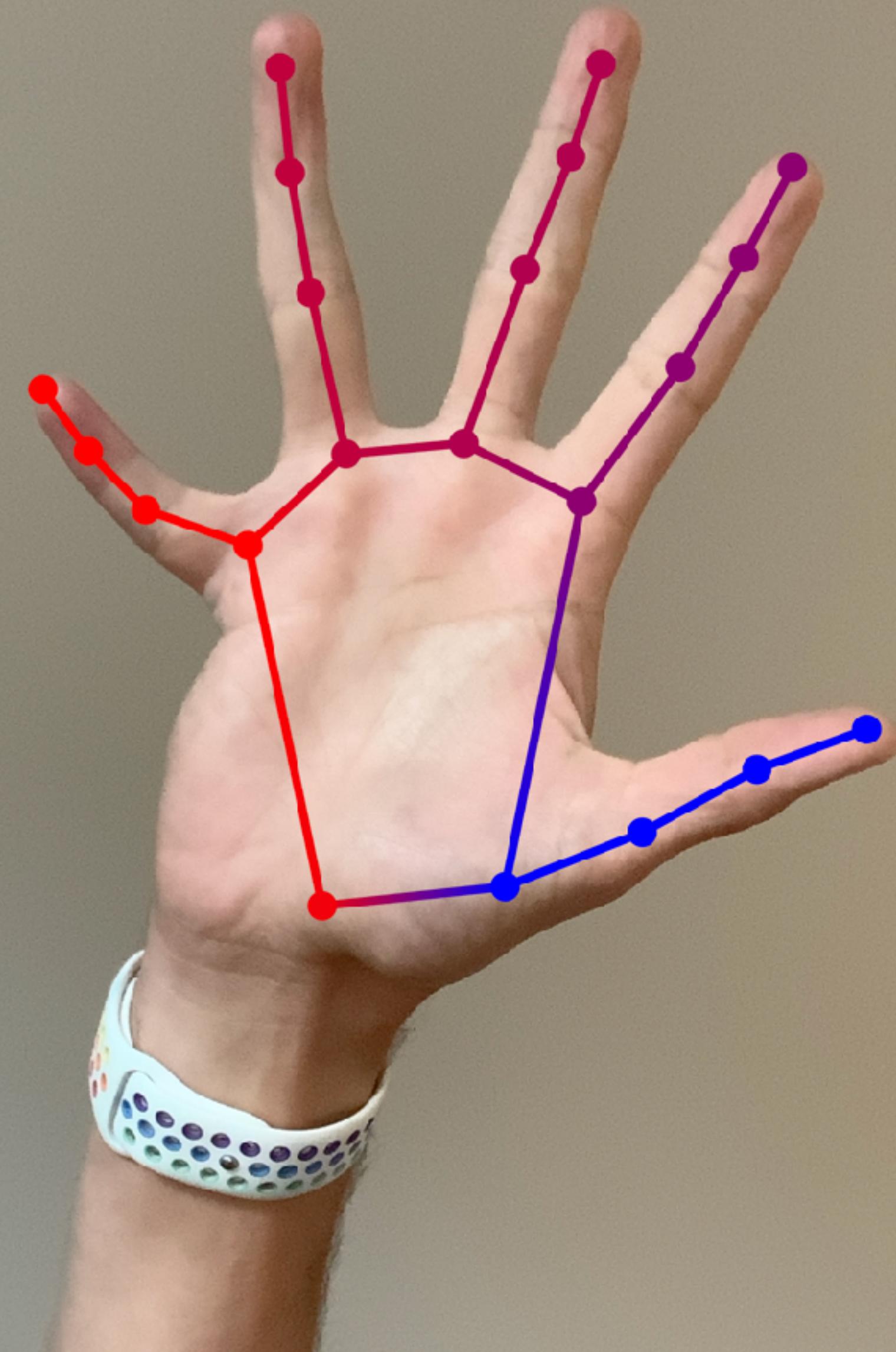
# Hand Pose Request\*

Detects up to 21 points on a hand

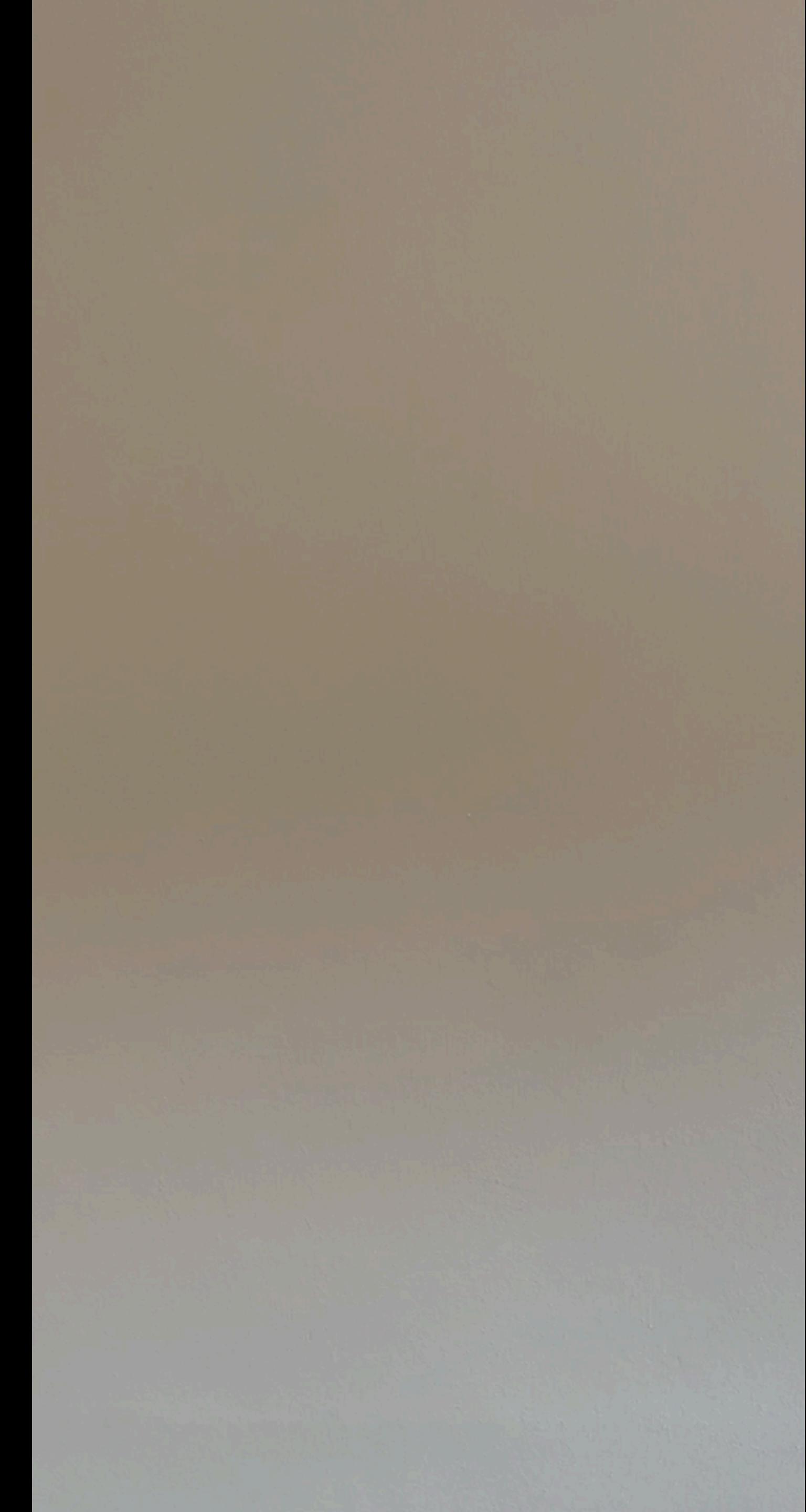
4 points per finger and thumb

1 point for the wrist

\* Human hands

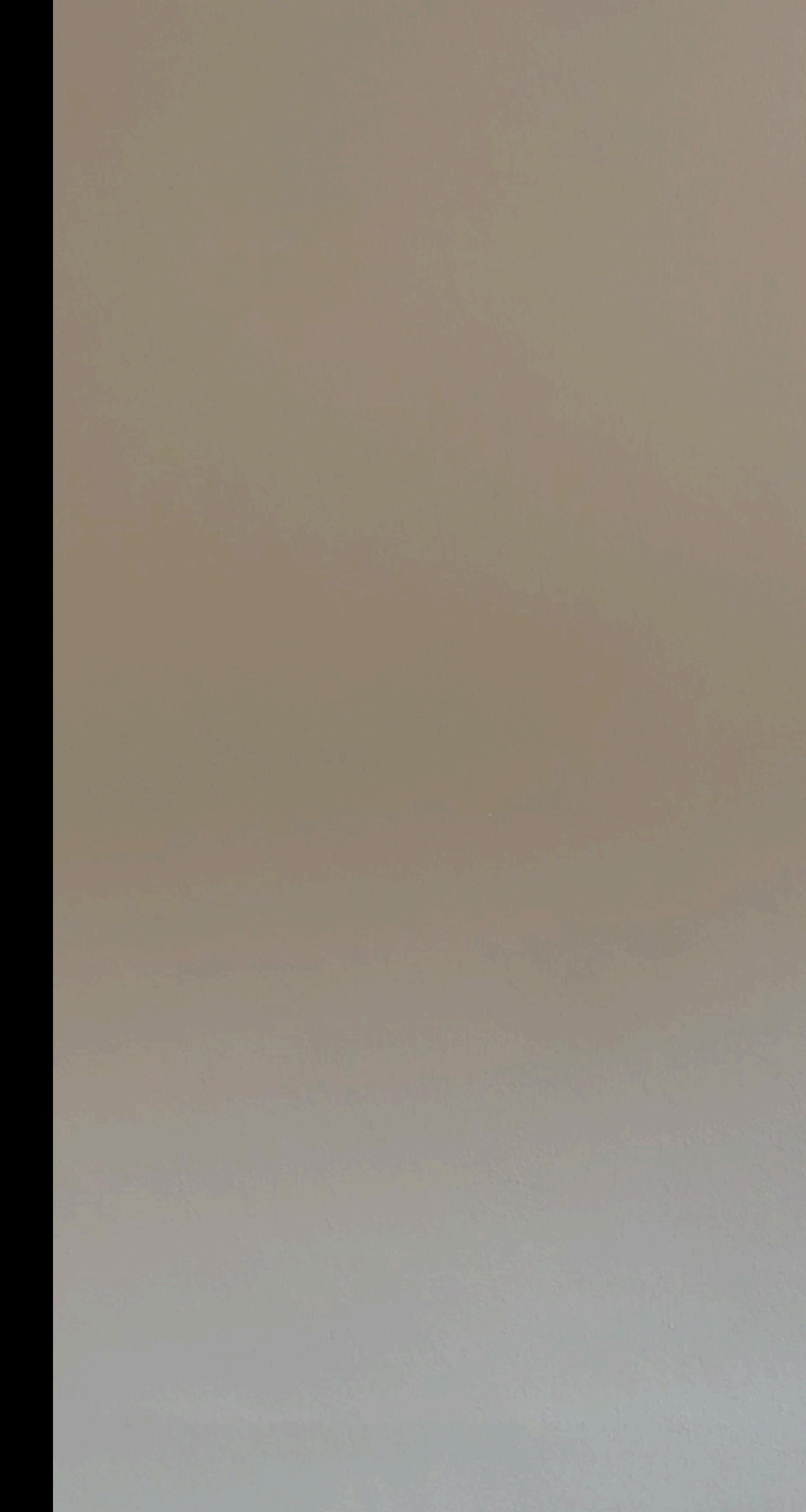


# Hand Pose in the App



# Hand Pose in the App

Unique game mechanic



# Hand Pose in the App

Unique game mechanic

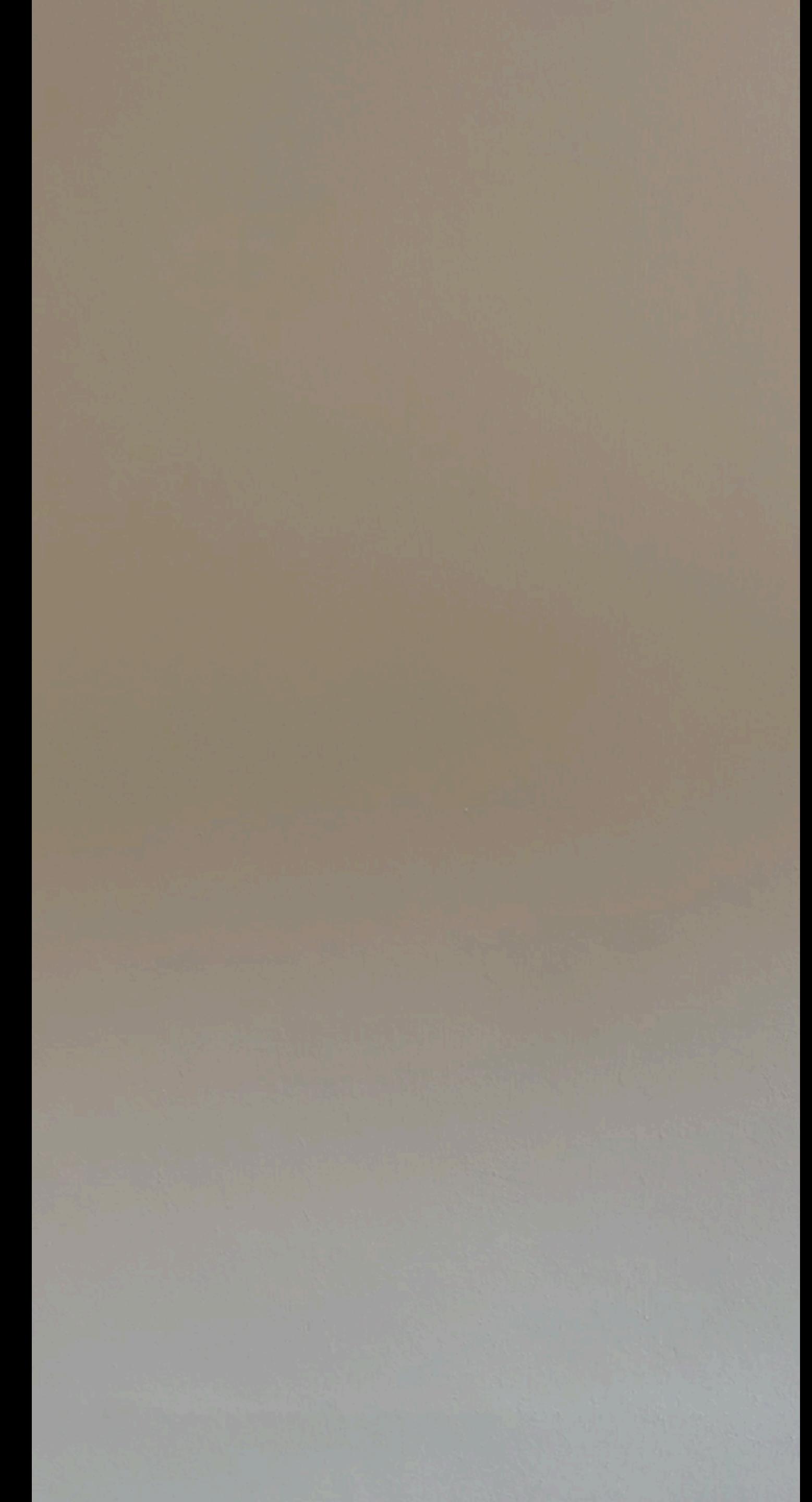
Simplify by just looking at the tip  
of the thumb and index finger

# Hand Pose in the App

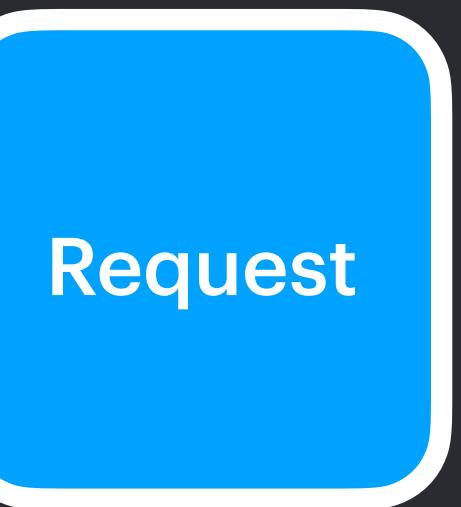
Unique game mechanic

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

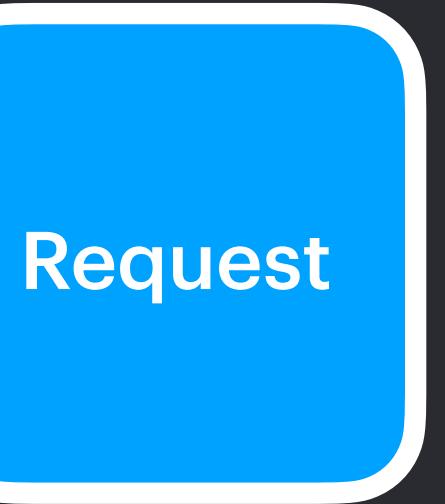
If the tips are close together, the  
user is pinching



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



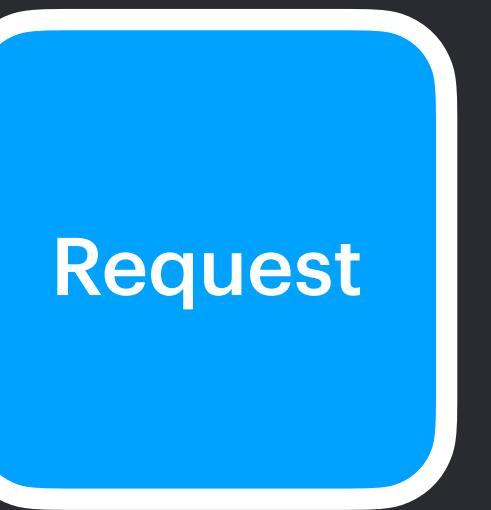
```
// Create the Vision request  
let request = VNRequest<VNHumanHandObservation>(  
    type: .hand,  
    features: [.pose])  
  
// Configure the request  
request.maximumHandCount = 1
```



```
// Create the Vision request
let request = VNRequest(for: .detectHumanHandPose)

// Configure the request
request.maximumHandCount = 1

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

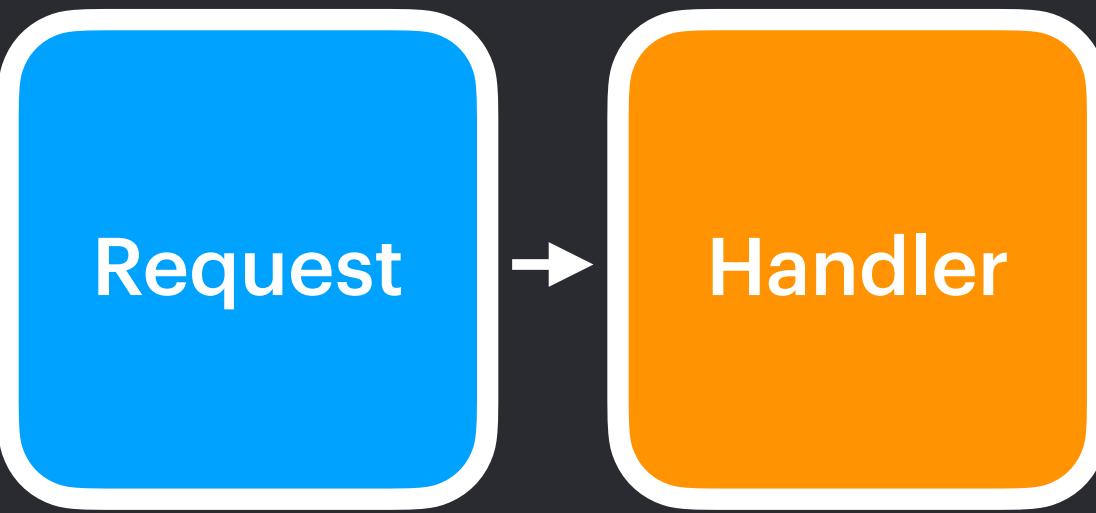


```
// Create the Vision request
let request = VNRequest()
request.name = "Human Hand Pose"
request.maximumHandCount = 1

// Configure the request
request.minimumDetectionConfidence = 0.5
request.minNumKeyPoints = 2
request.maxNumKeyPoints = 20

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

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



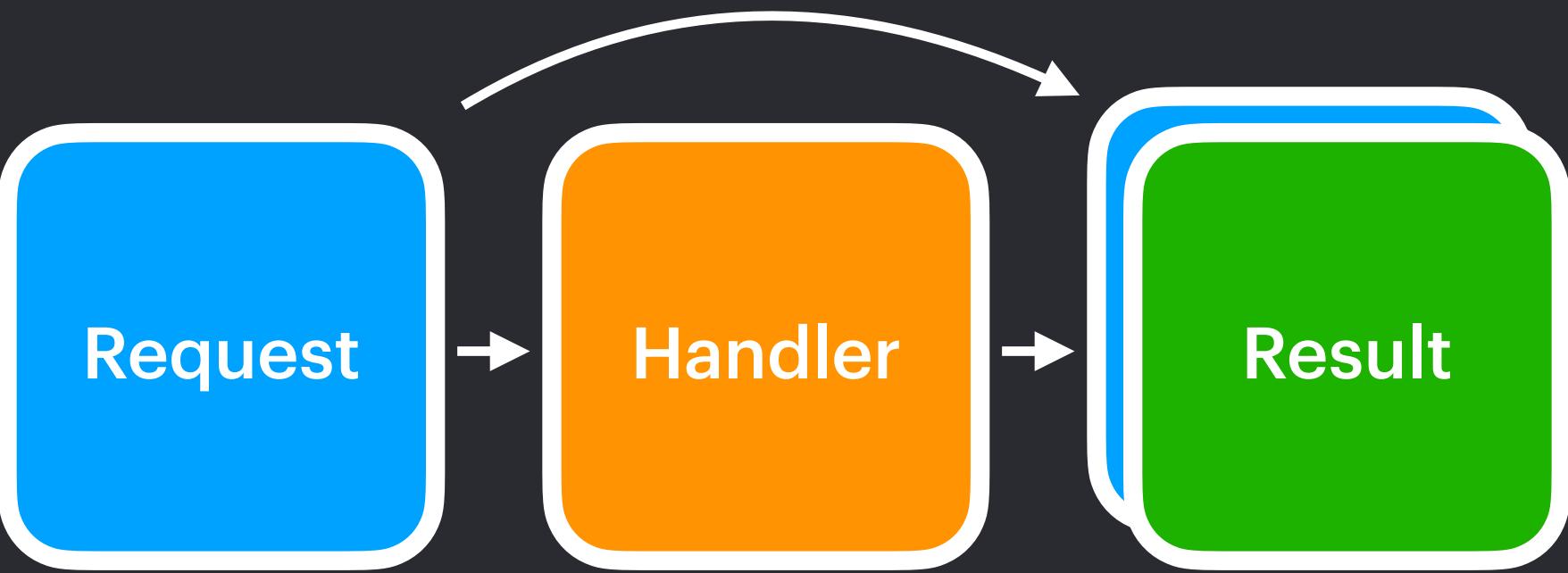
```
// Create the Vision request
let request = VNRequest()
request.name = "Human Hand Pose"
request.maximumHandCount = 1

// 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 {

}
```

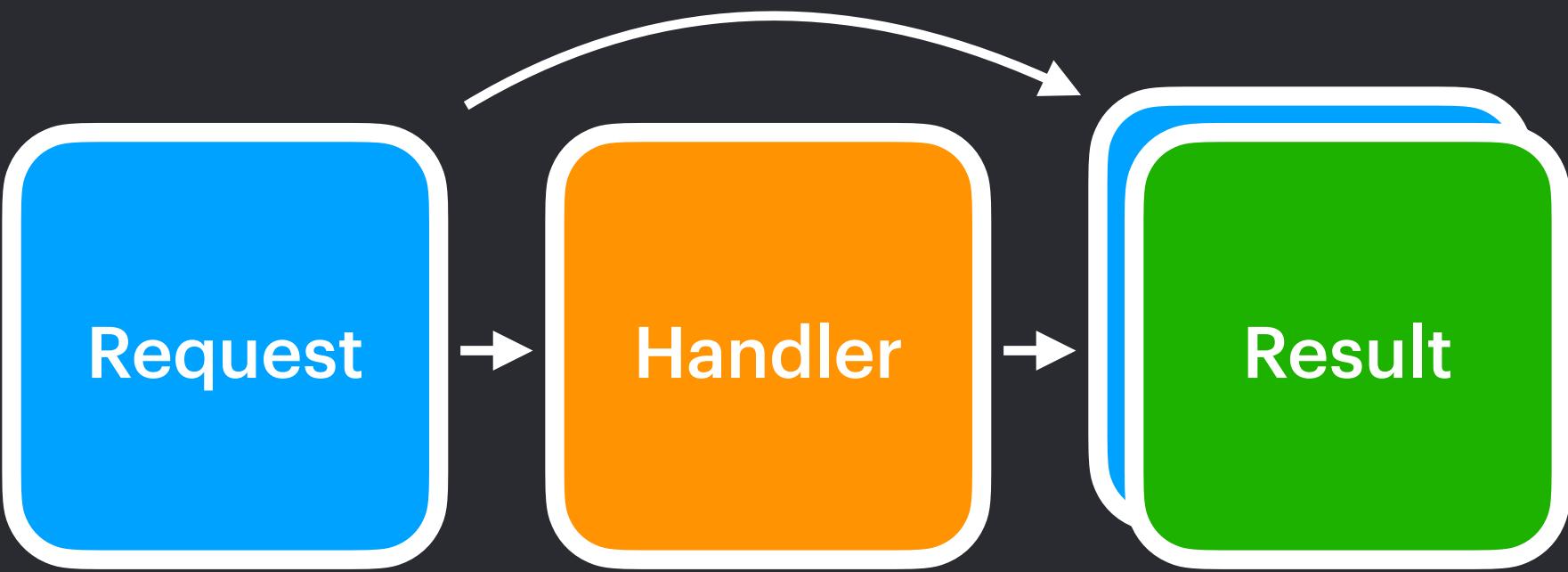


```
// Create the Vision request
let request = VNRequest()
request.name = "Human Hand Pose"
request.maximumHandCount = 1

// 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

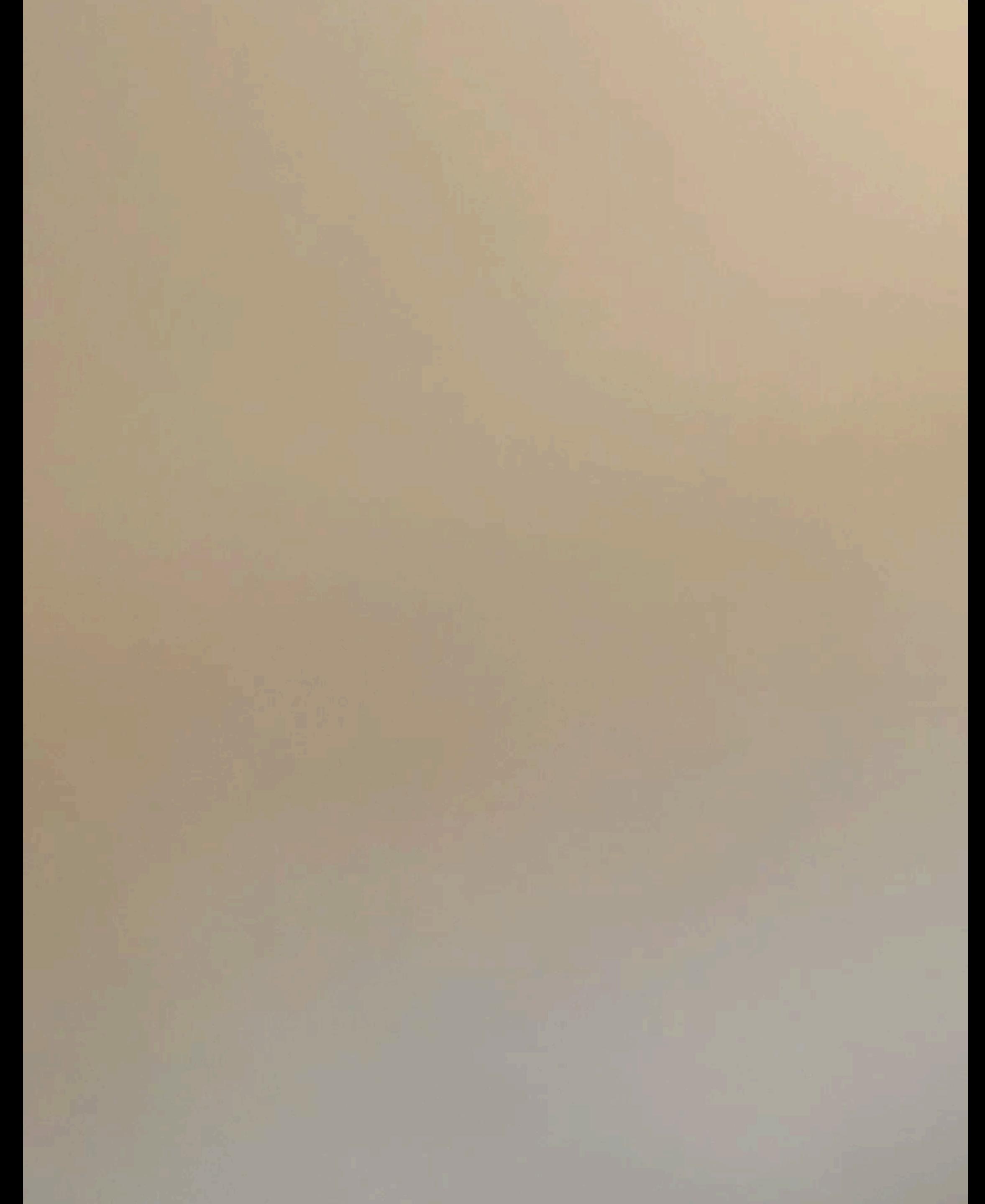
Comparing Feature Vectors

Object Tracking

BONUS:

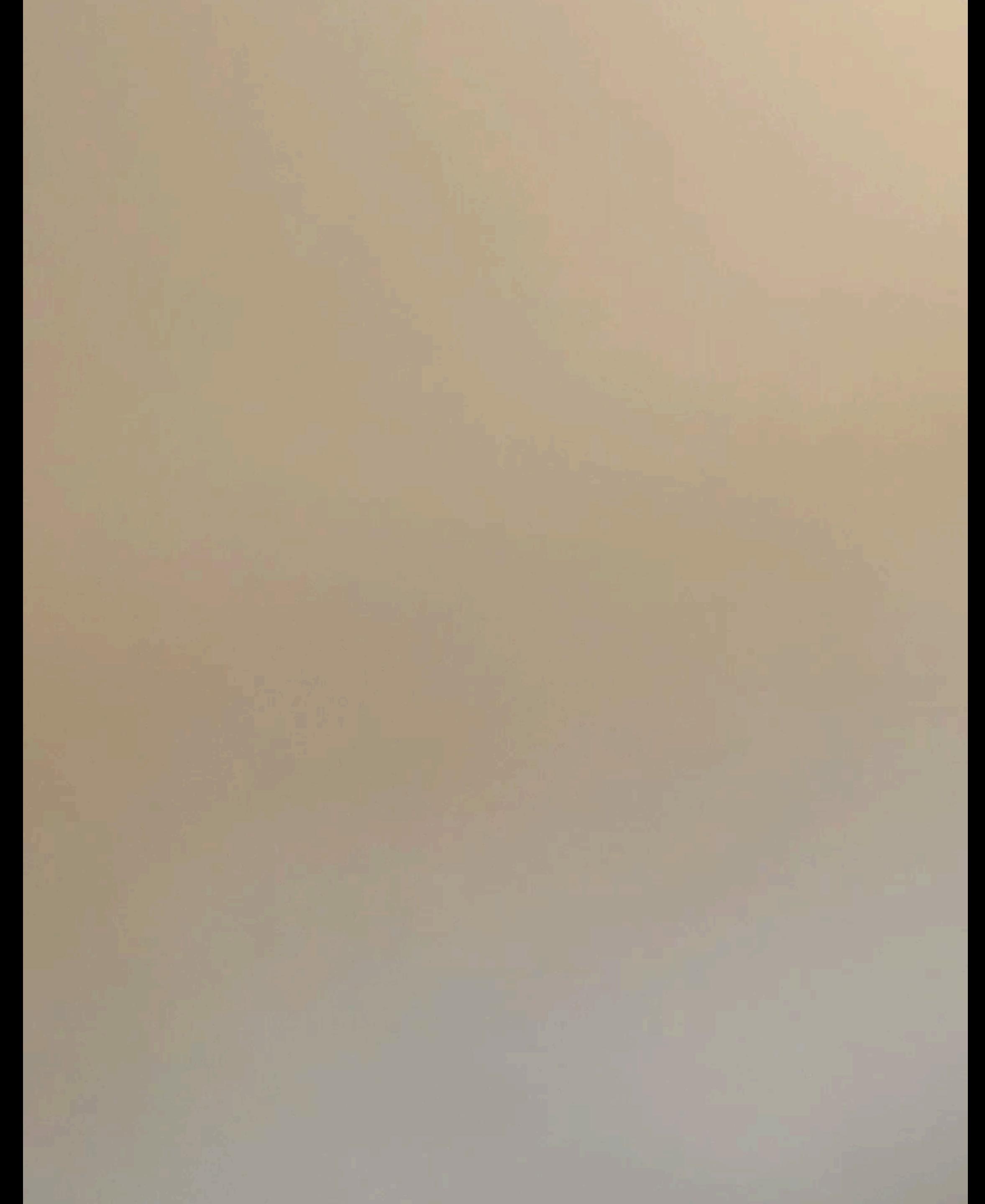
Custom Classifier

# 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

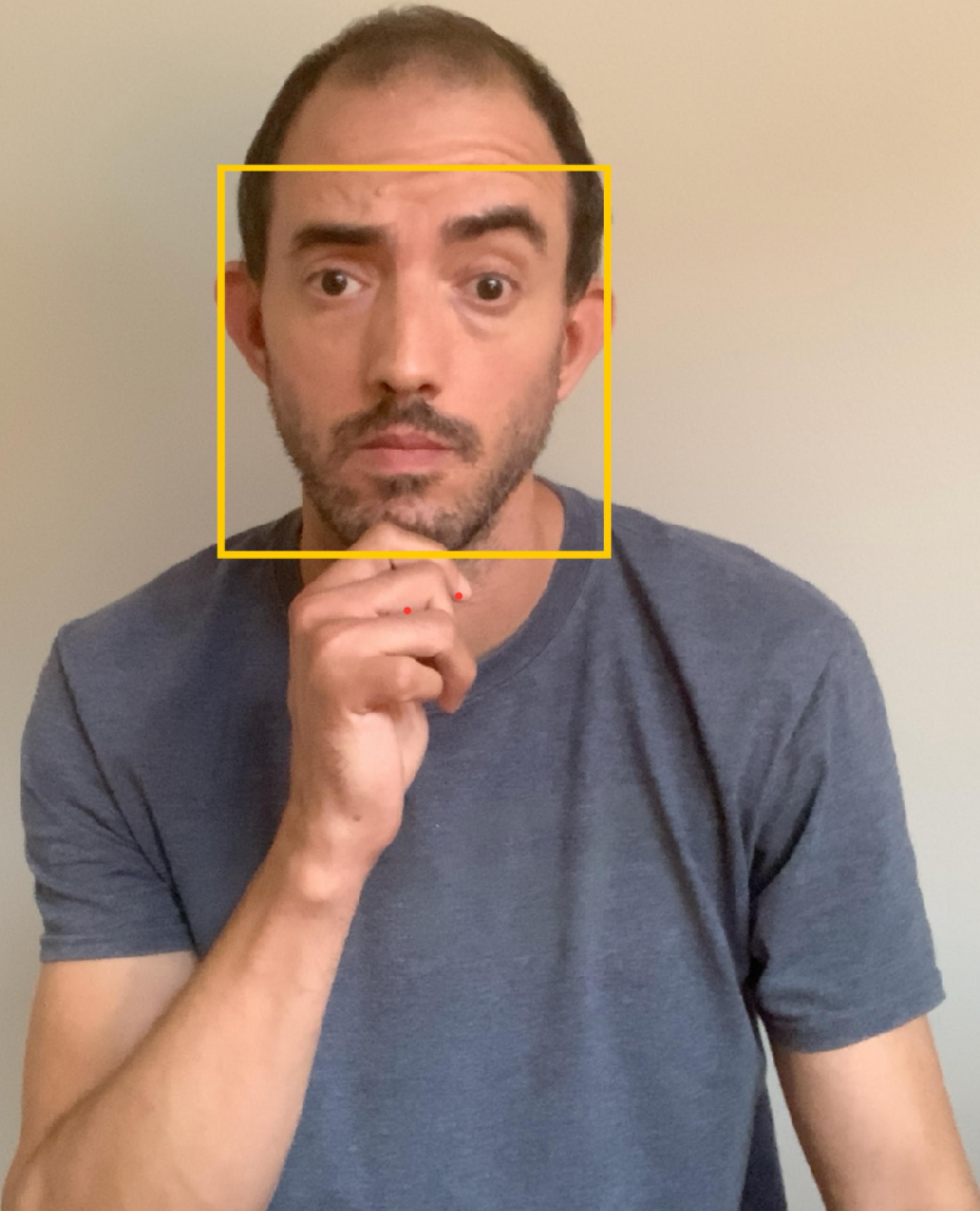
Faces are a good proxy for heads

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

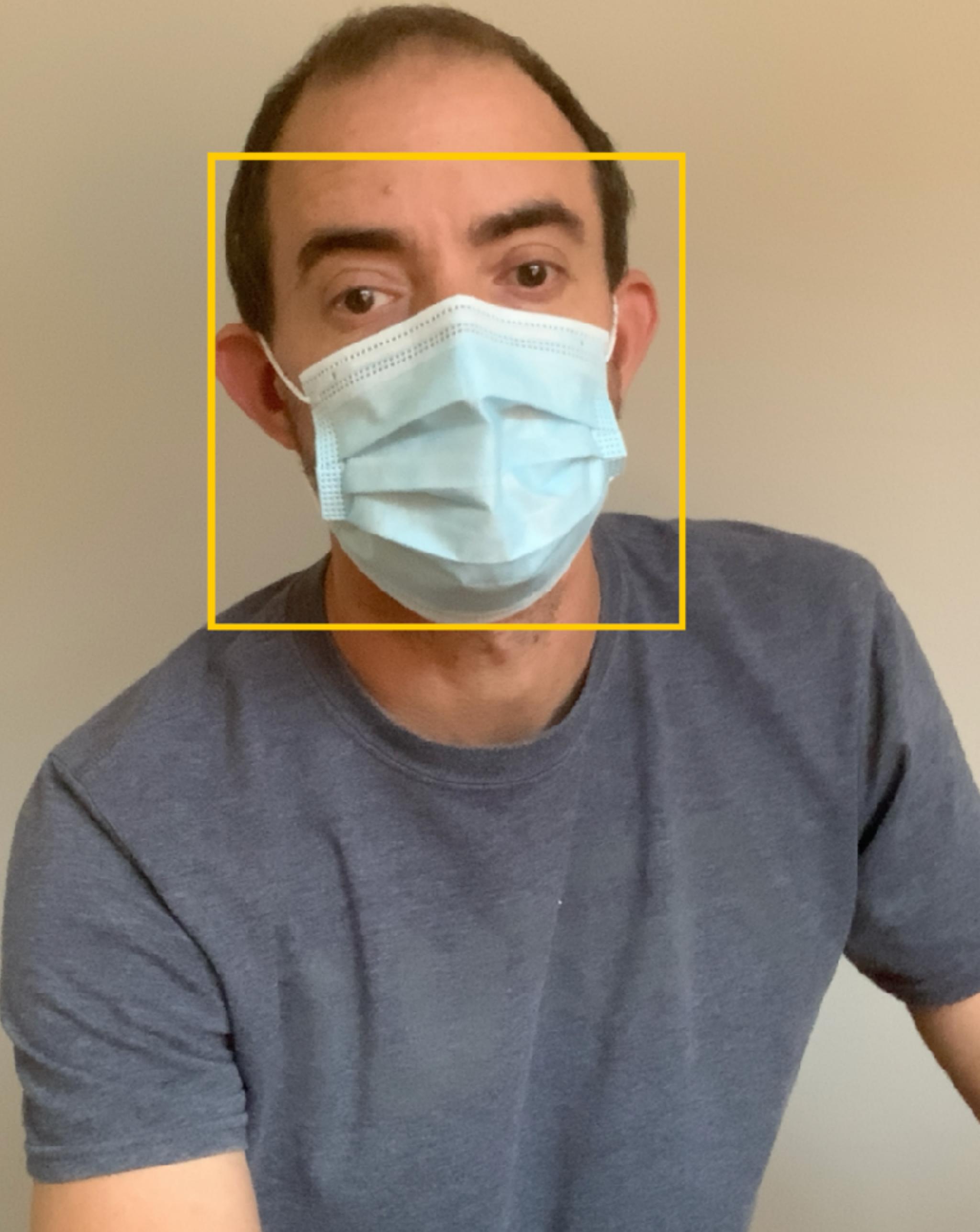


# Face Rectangles Request

Normal faces

Profile faces

Obscured faces



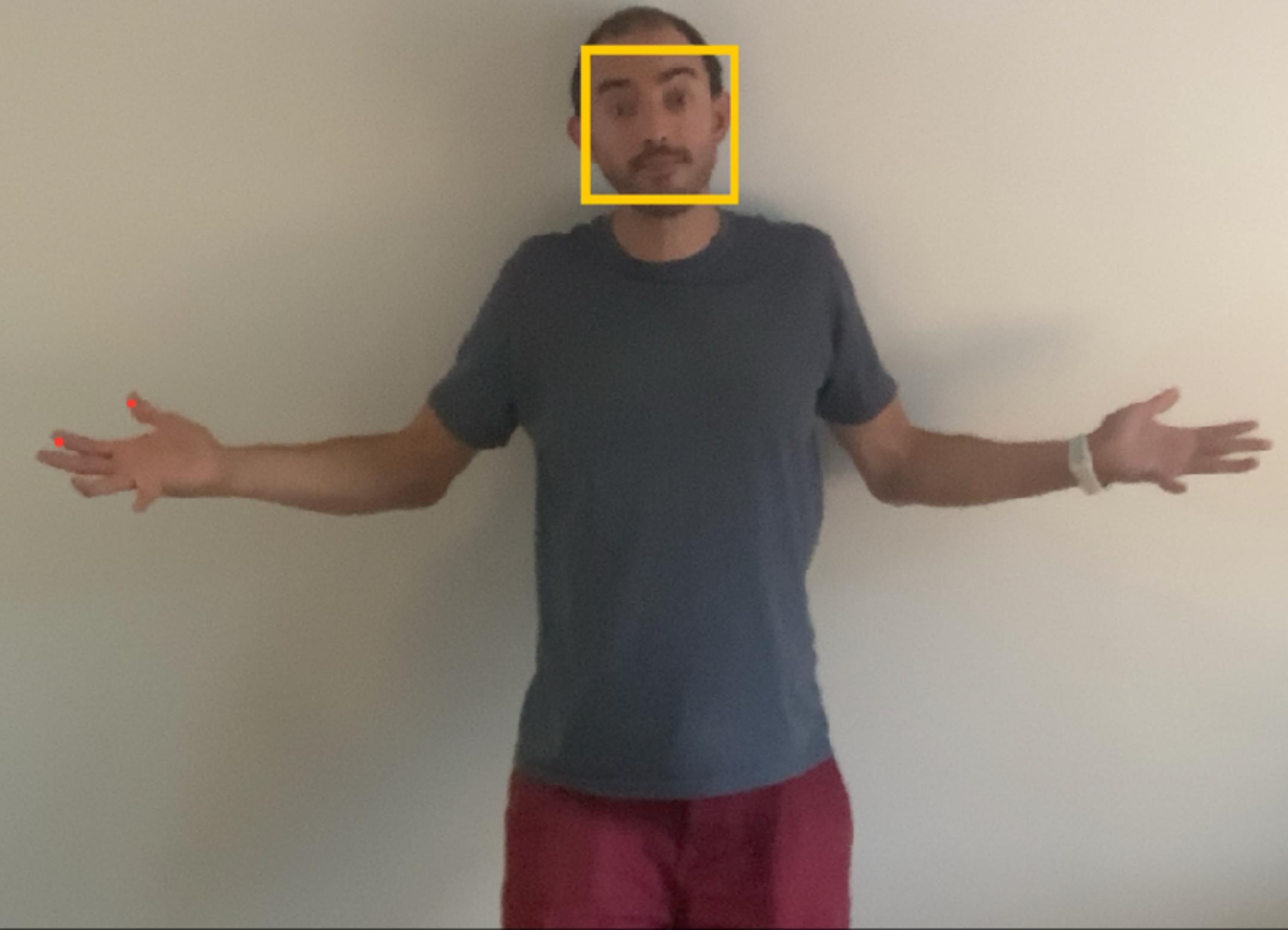
# Face Rectangles Request

Normal faces

Profile faces

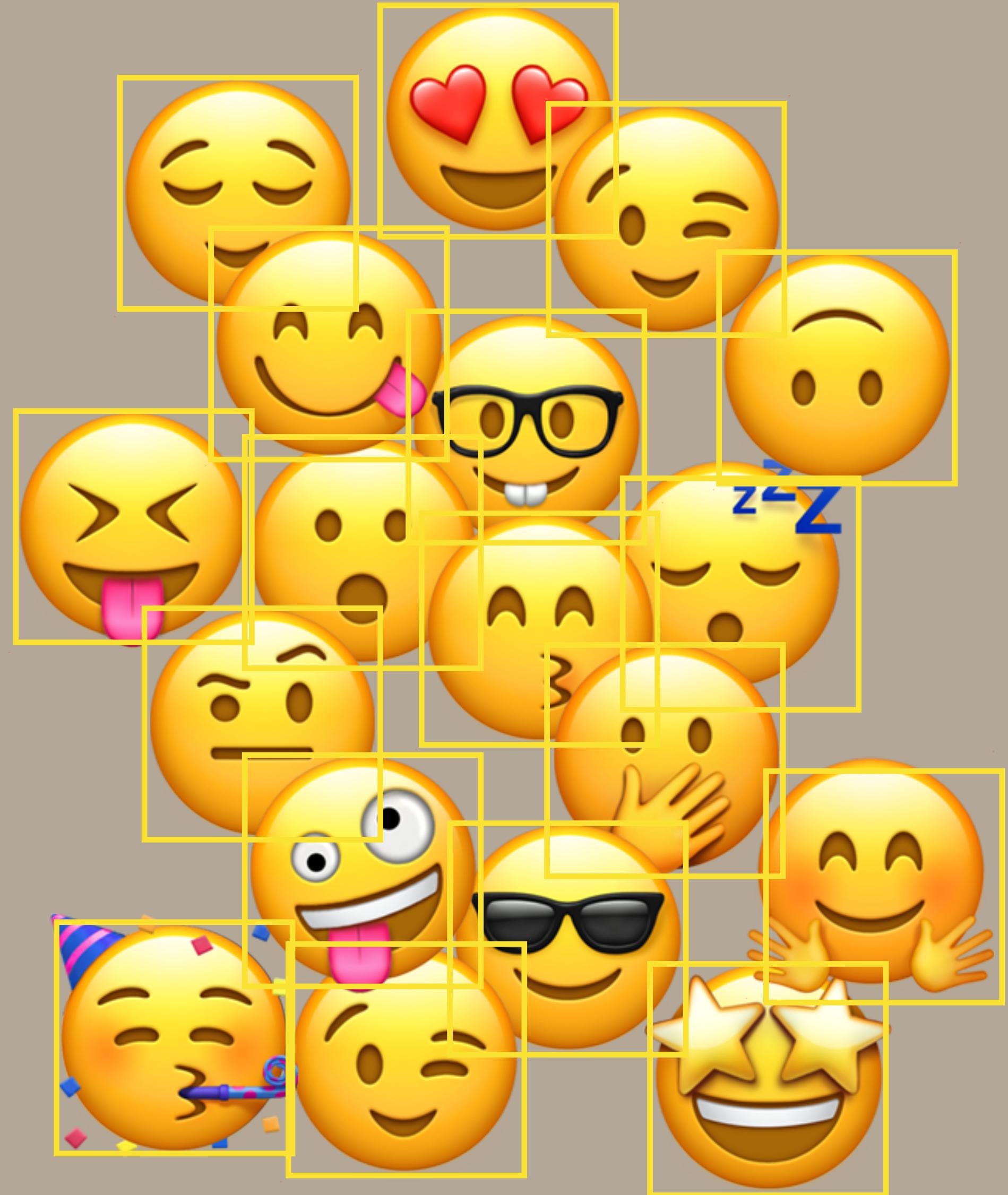
Obscured faces

Small faces

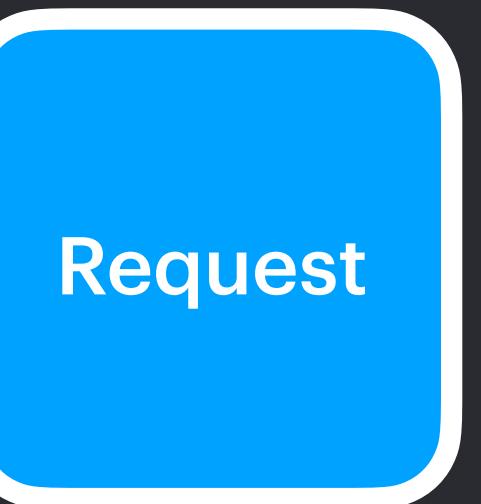


# Face Rectangles Request

- Normal faces
- Profile faces
- Obscured faces
- Small faces
- Multiple faces



```
// Create the Vision request  
let request = VNRequest(rectangle)
```



```
// Create the Vision request
let request = VNRequest(rectangle)

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



```
// Create the Vision request
let request = VNRequest(rectangle)

// 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: [:])
```

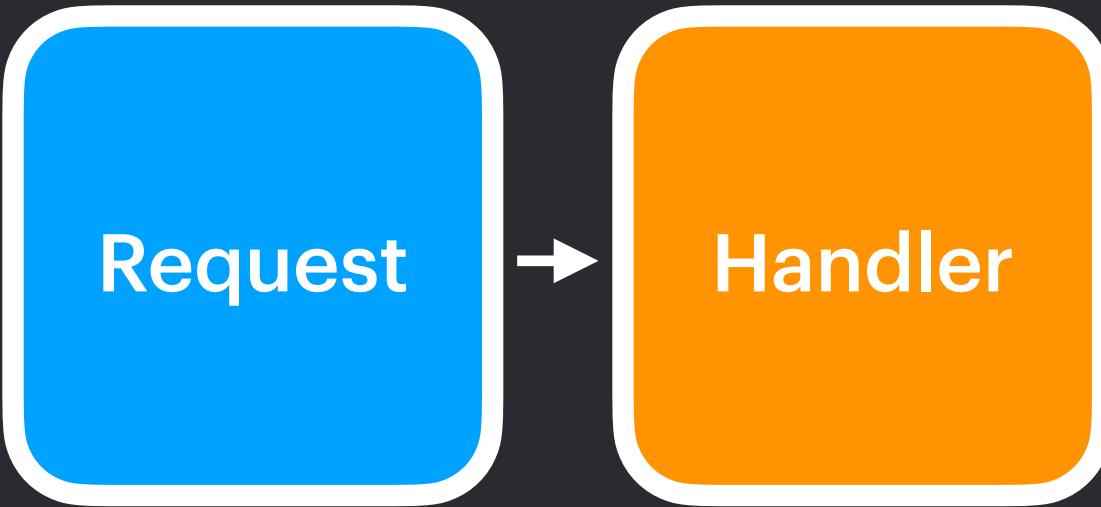


```
// Create the Vision request
let request = VNRequest(rectangle)

// 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 = VNRequest(rectangleFromImage: image)

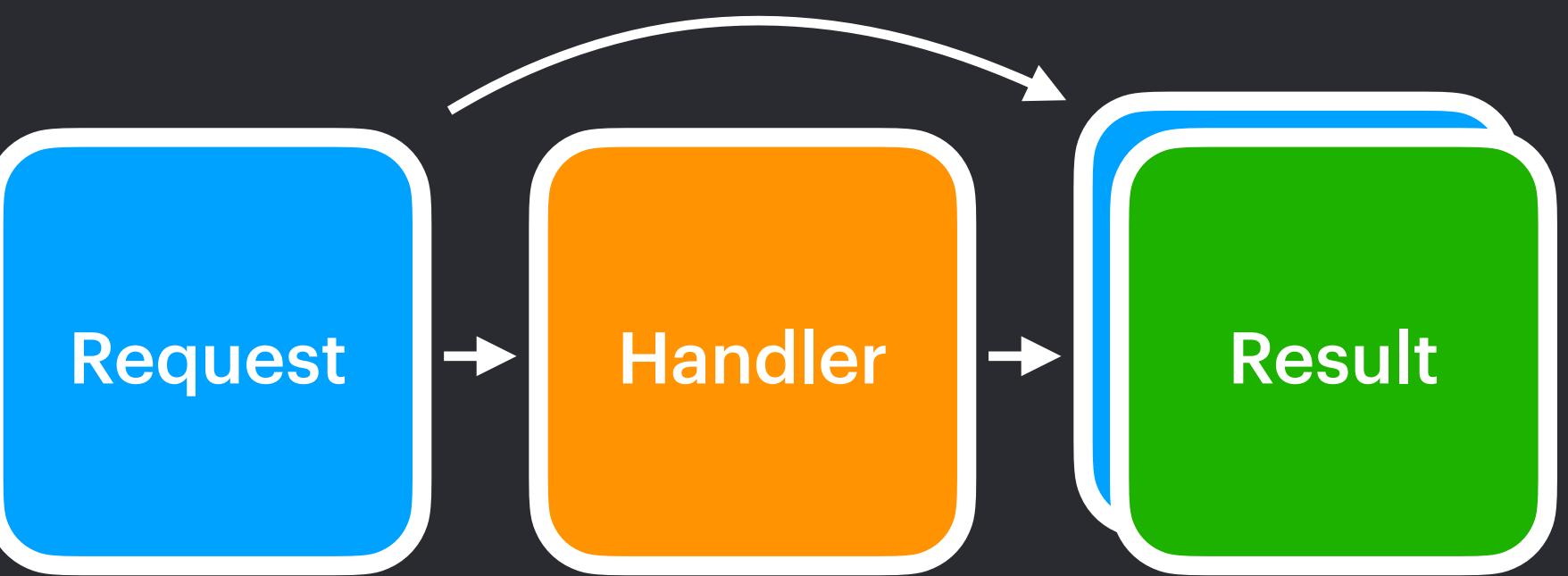
// Configure the request
request.regionOfInterest = CGRect(x: 0.2, y: 0.5, width: 0.6, height: 0.3)

// 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] {

}
```



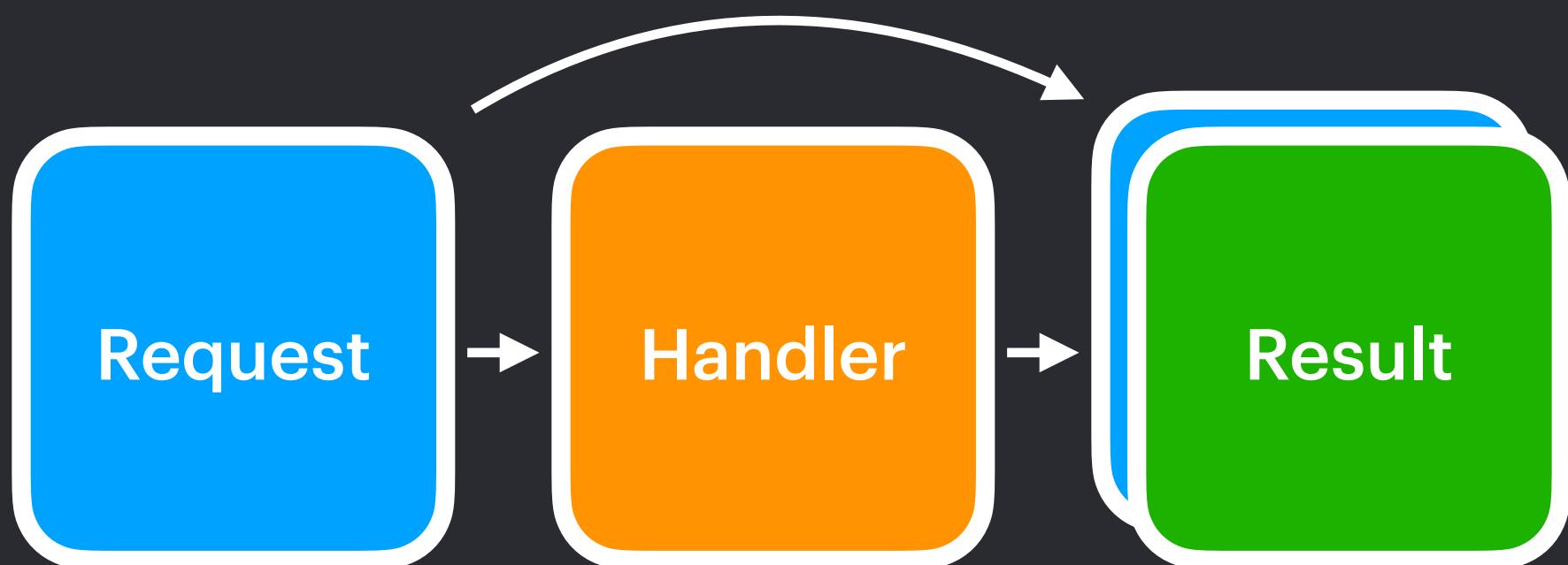
```
// Create the Vision request
let request = VNRequest(rectangleFromImage: image)

// 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)
```



(0.0, 0.0)

(1.0, 1.0)

```
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)
```



# Why specify a Region of Interest?

# Why specify a Region of Interest?



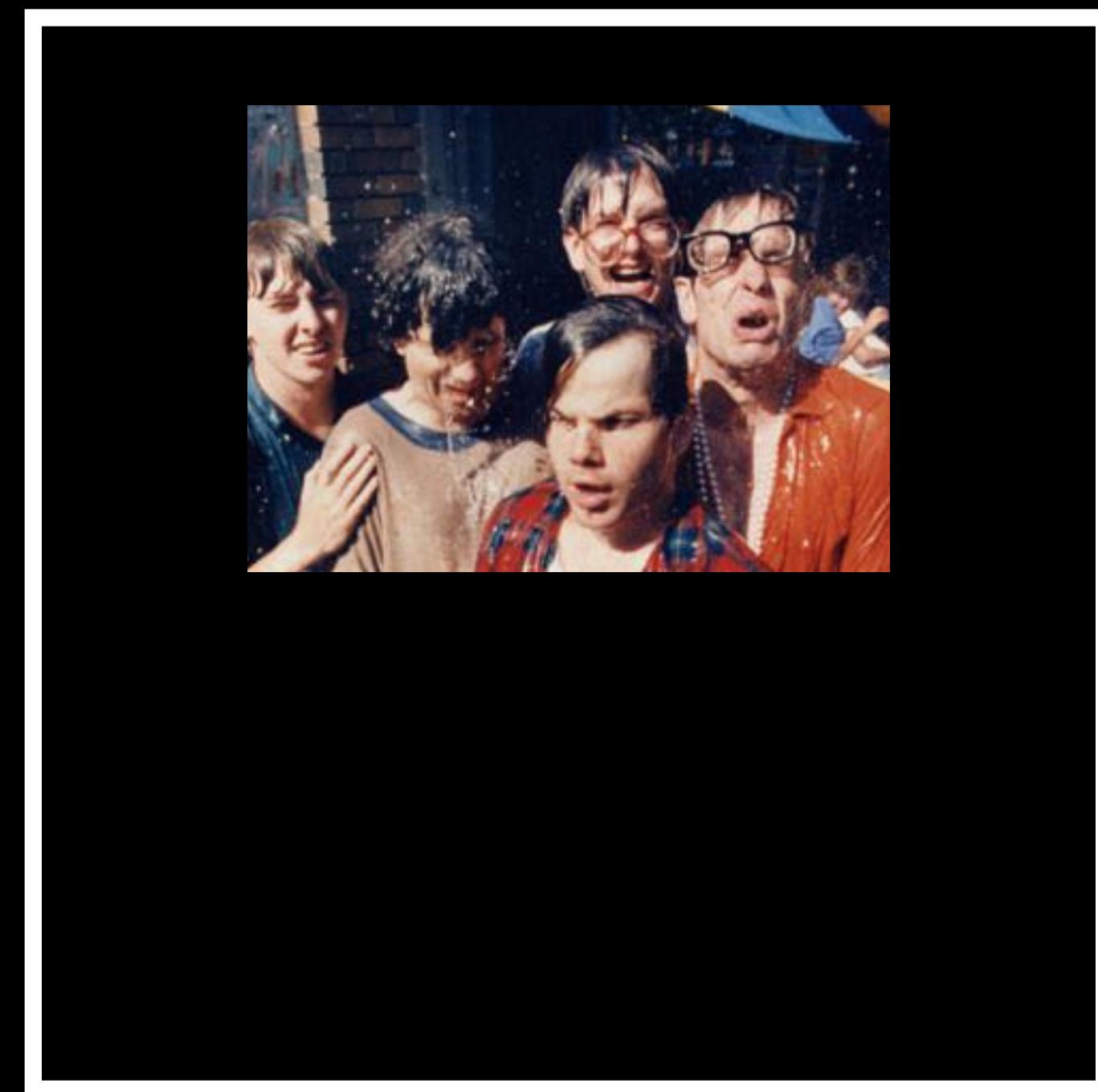
# Why specify a Region of Interest?



# Why specify a Region of Interest?



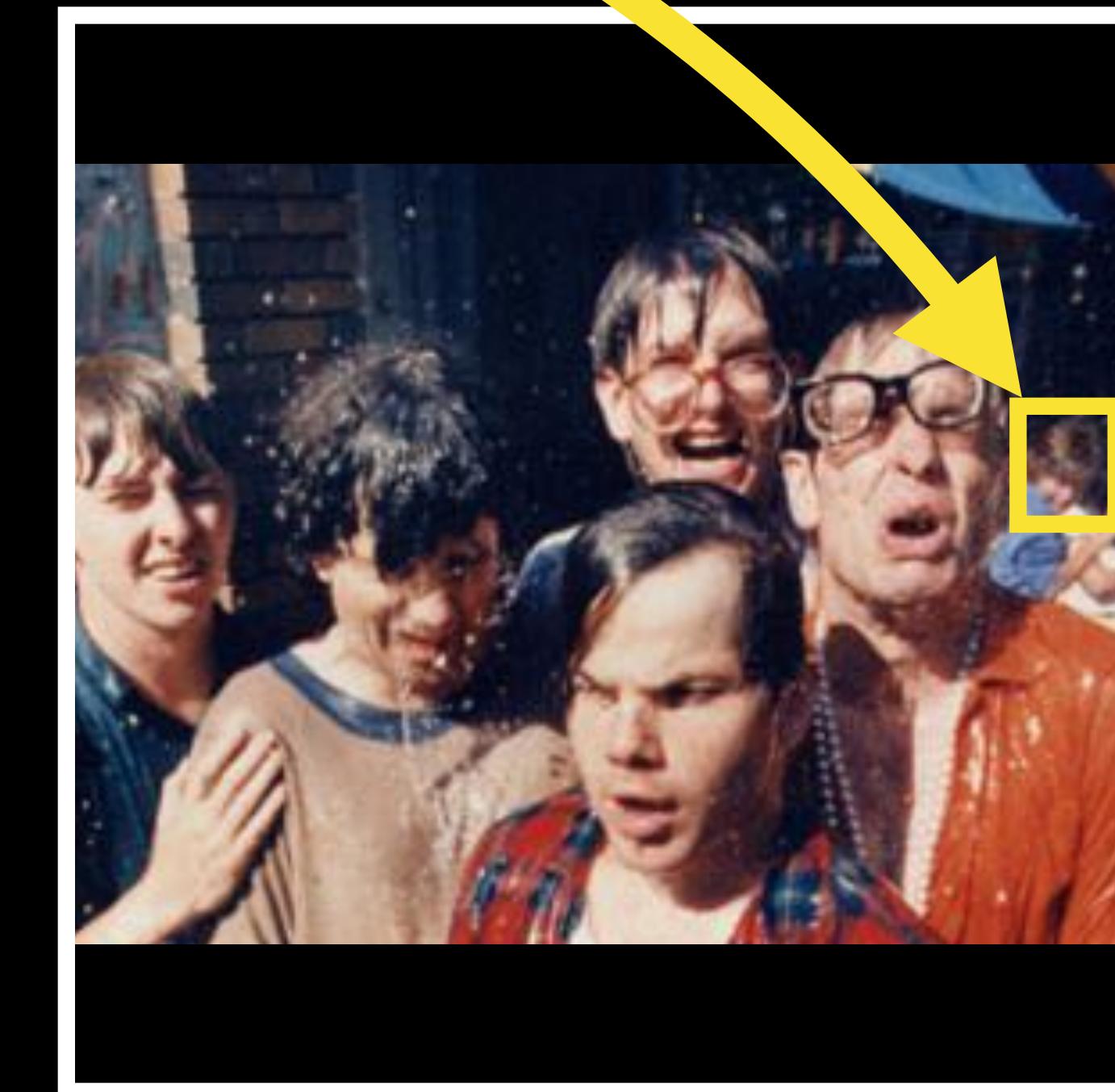
# Why specify a Region of Interest?



# Why specify a Region of Interest?



# Why specify a Region of Interest?



# One Problem

Players can game the system

SCORE:

0

# Crushing Your Head

## The App

Hand Pose Detector

Face Detector

Comparing Feature Vectors

Object Tracking

BONUS:

Custom Classifier

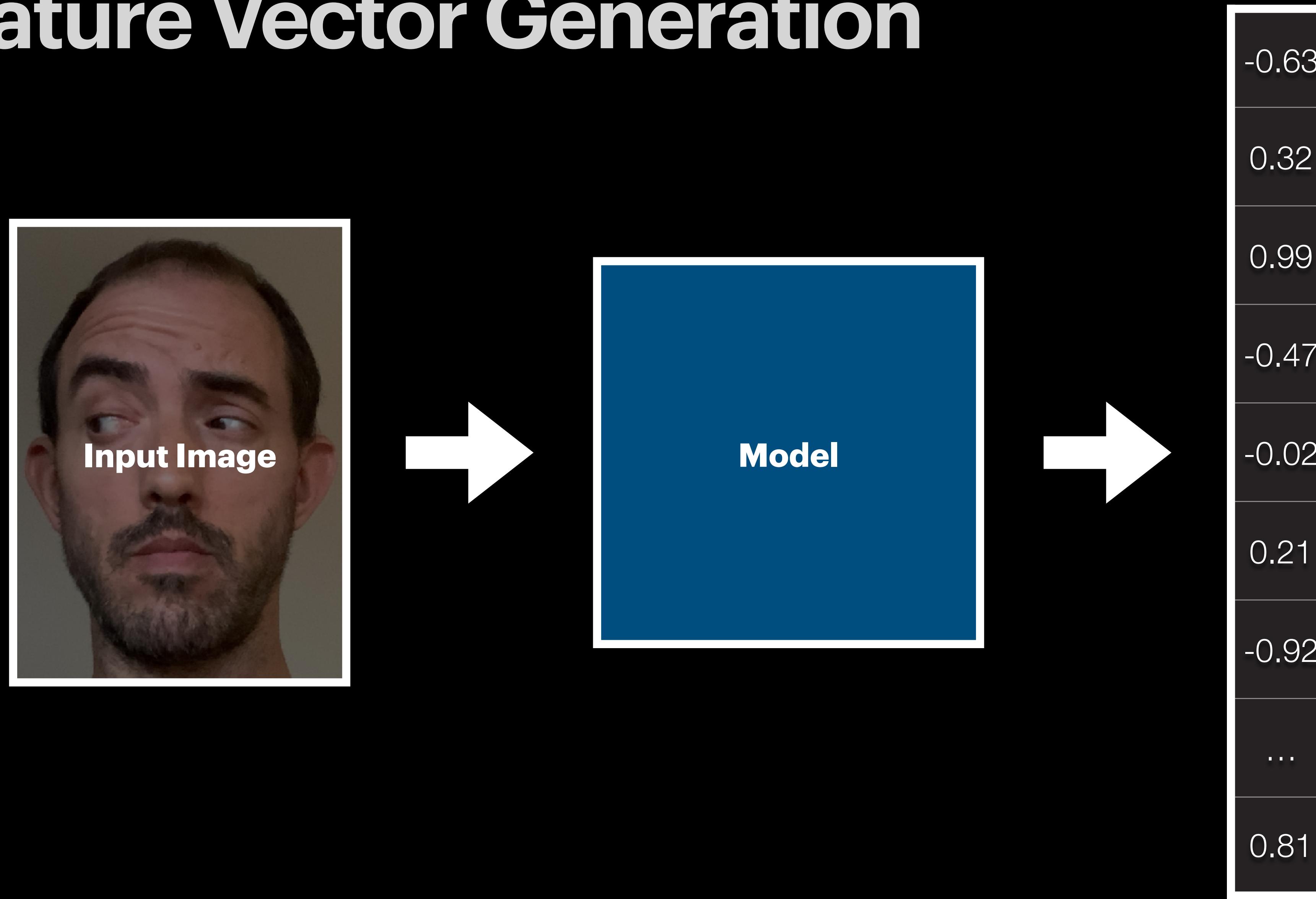
# Feature Vector

a.k.a. Feature Print

a.k.a. Feature Embedding

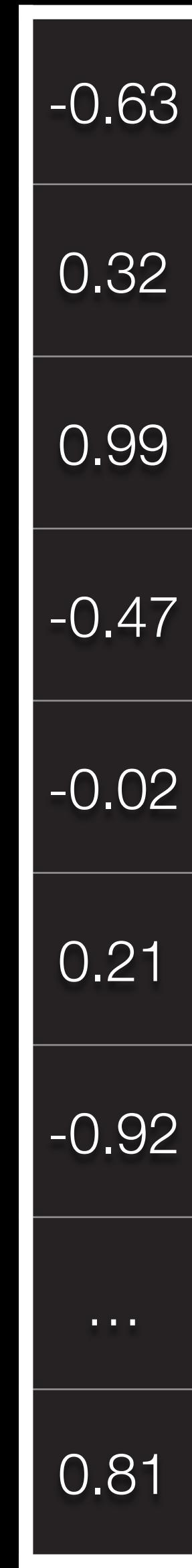
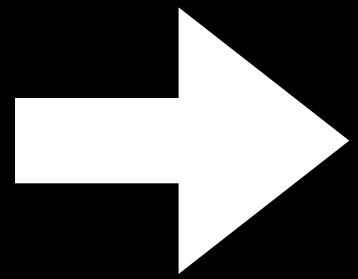
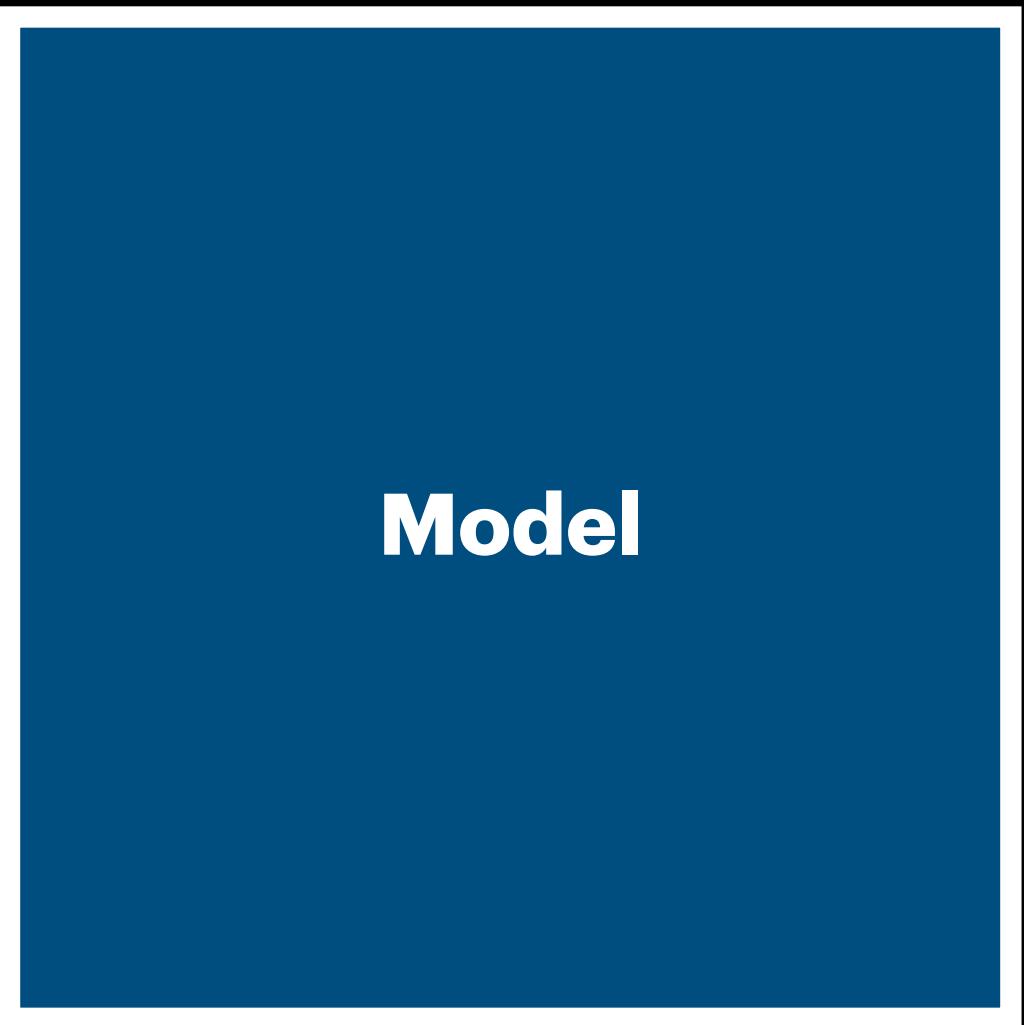
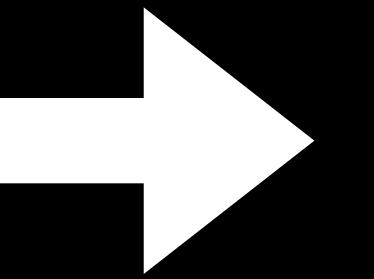
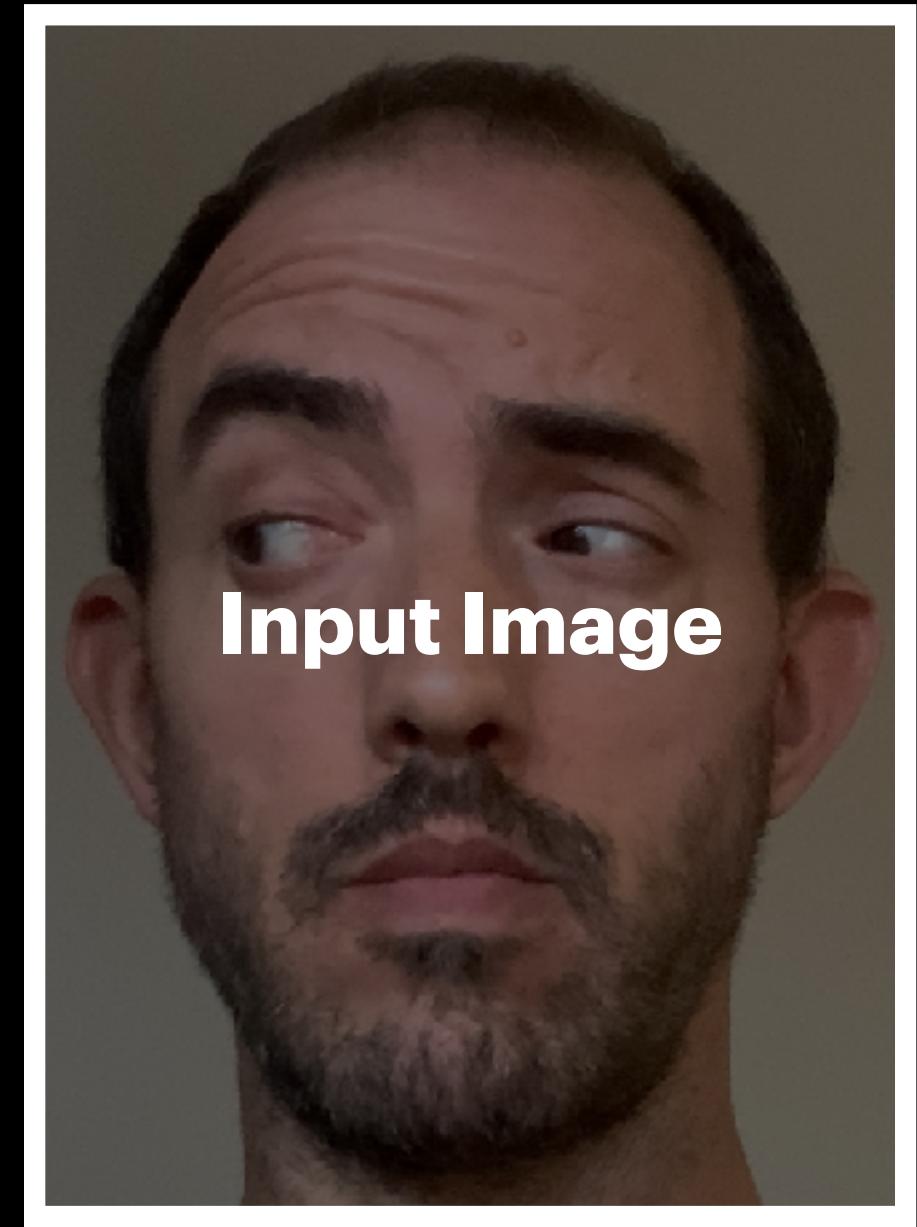
An array of numbers representing important characteristics of an image

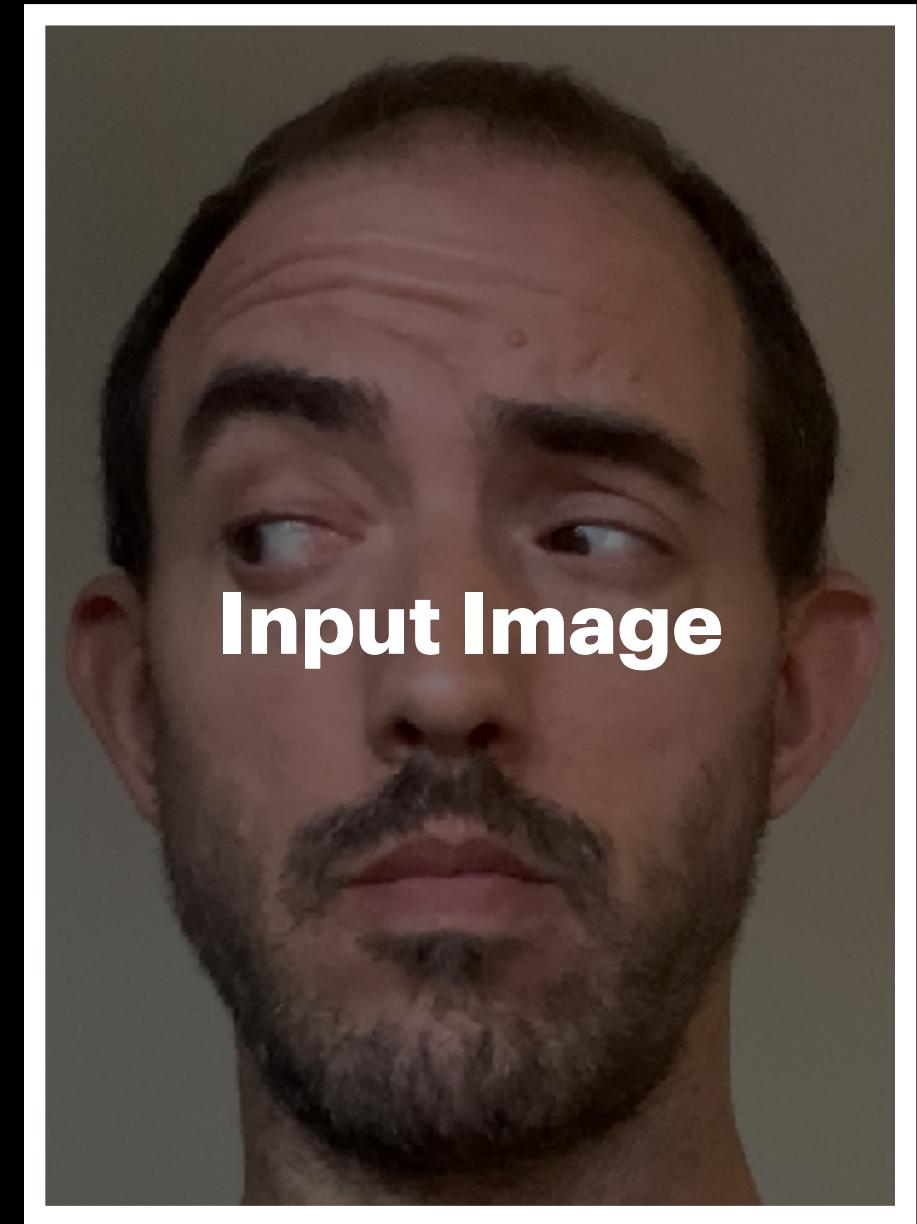
# Feature Vector Generation



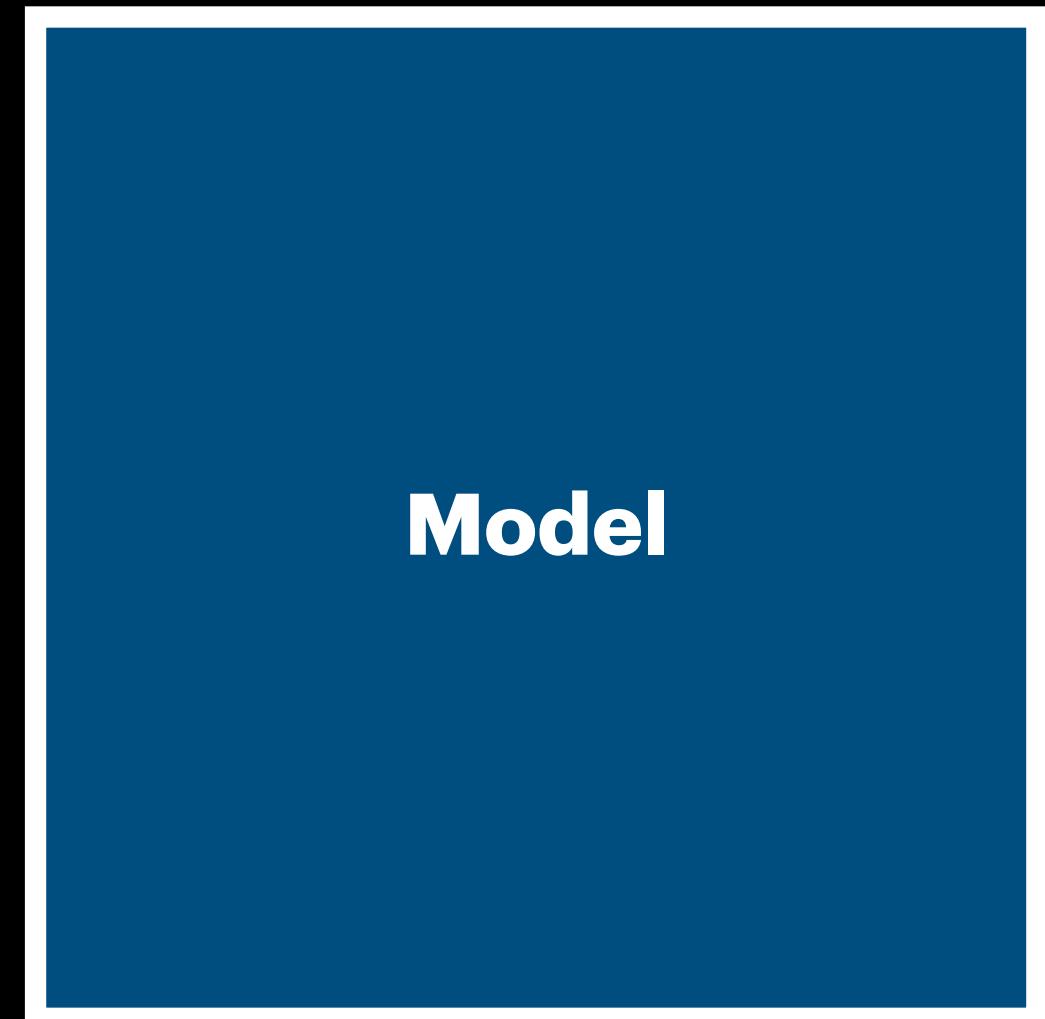
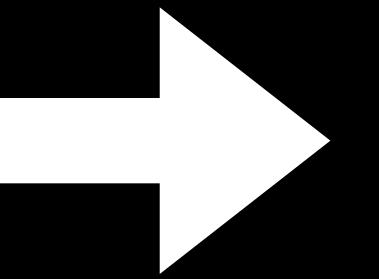
**Does this look familiar?**



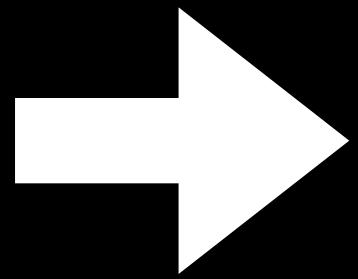




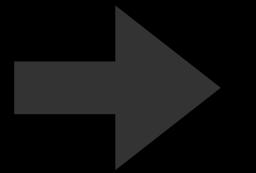
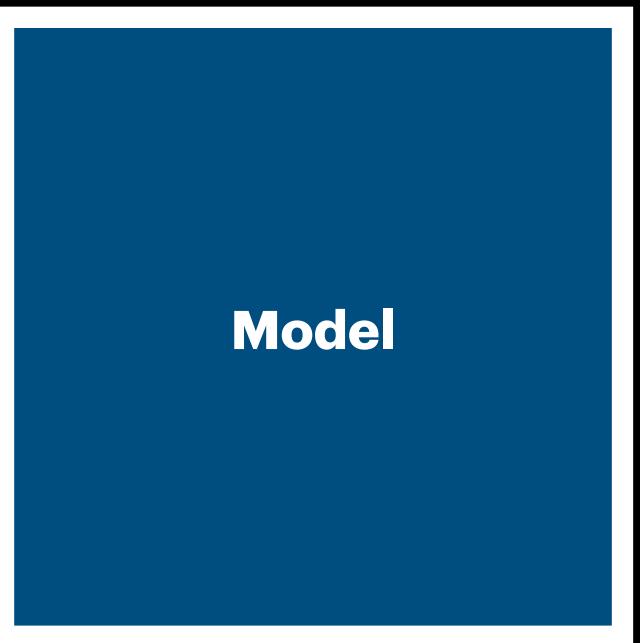
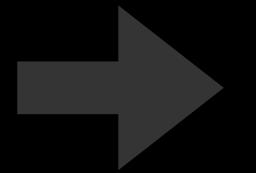
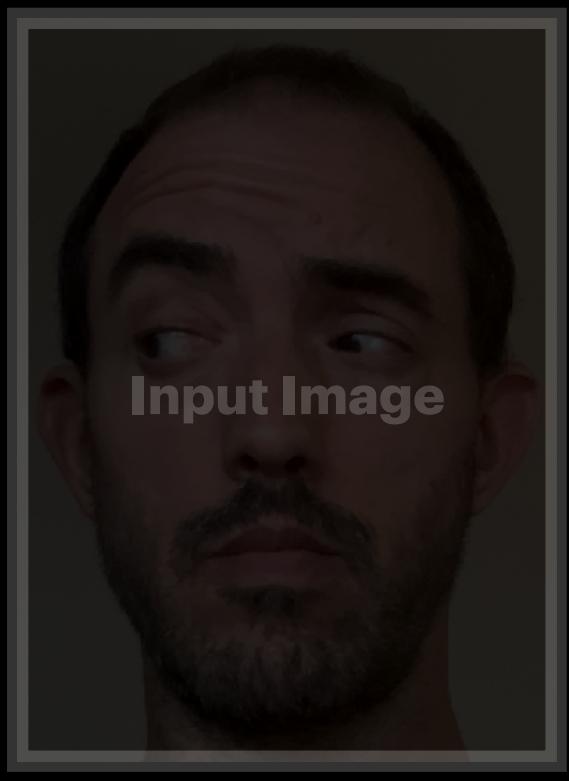
**Input Image**



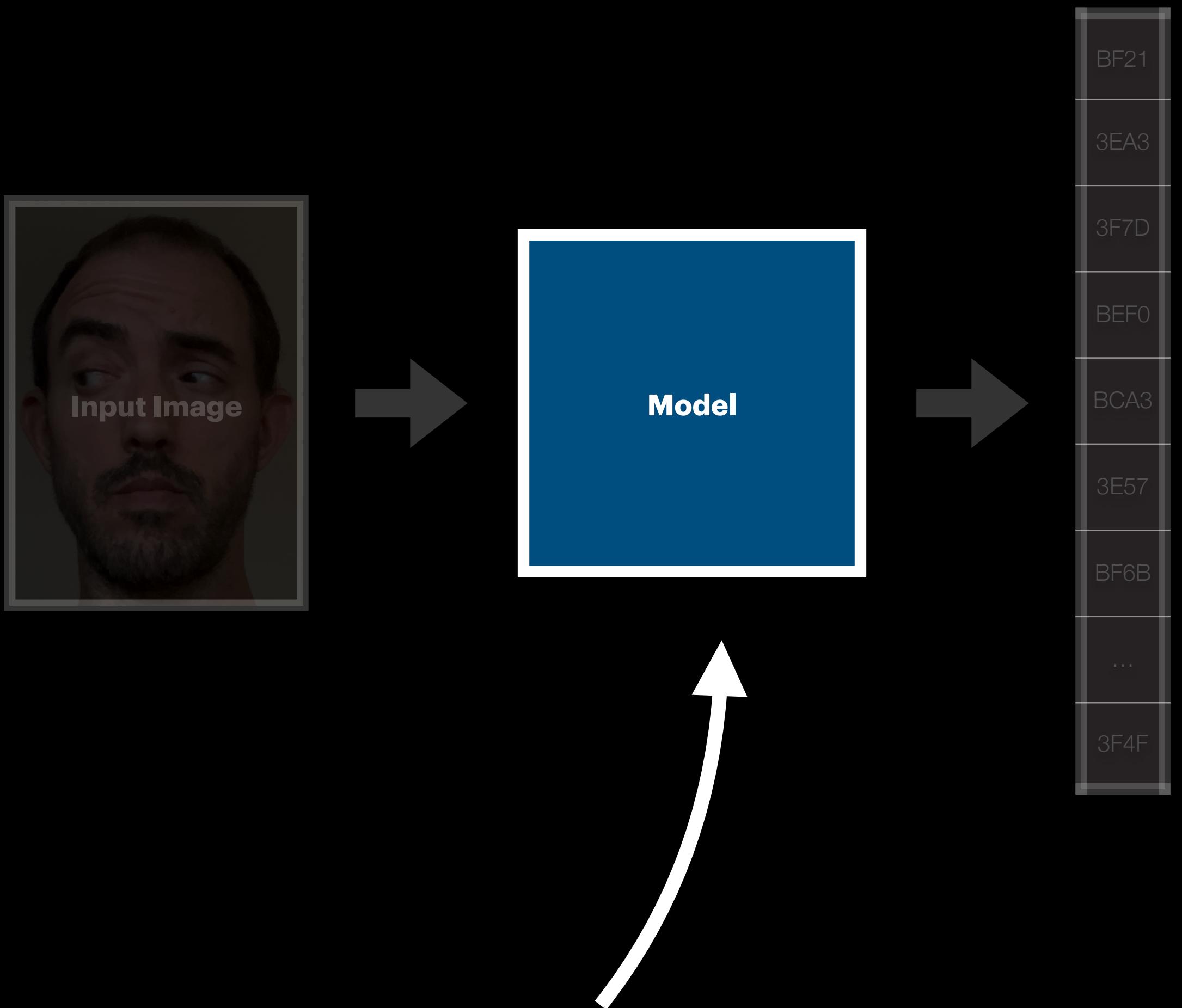
**Model**



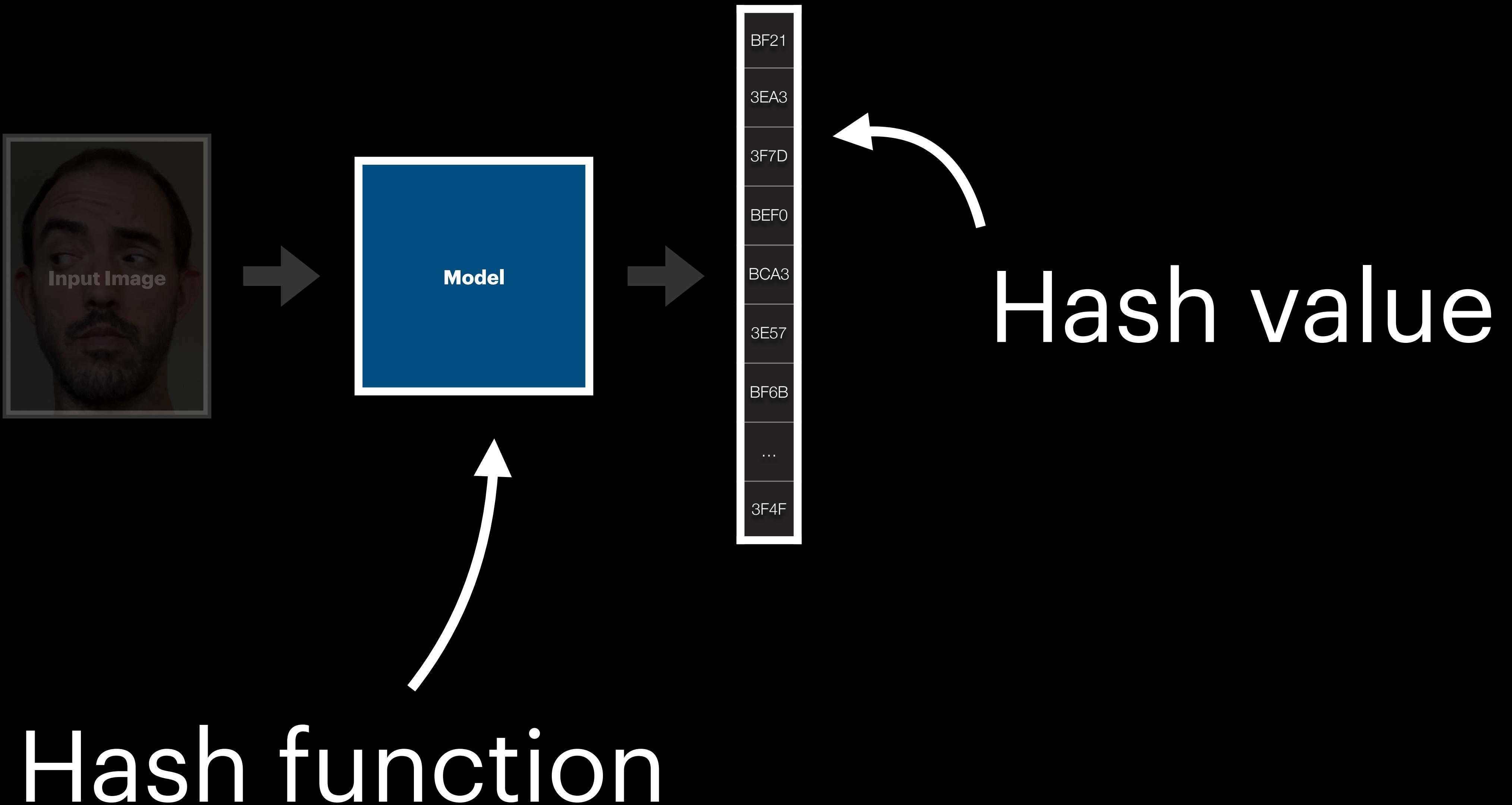
BF21
3EA3
3F7D
BEF0
BCA3
3E57
BF6B
...
3F4F



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



# Hash function

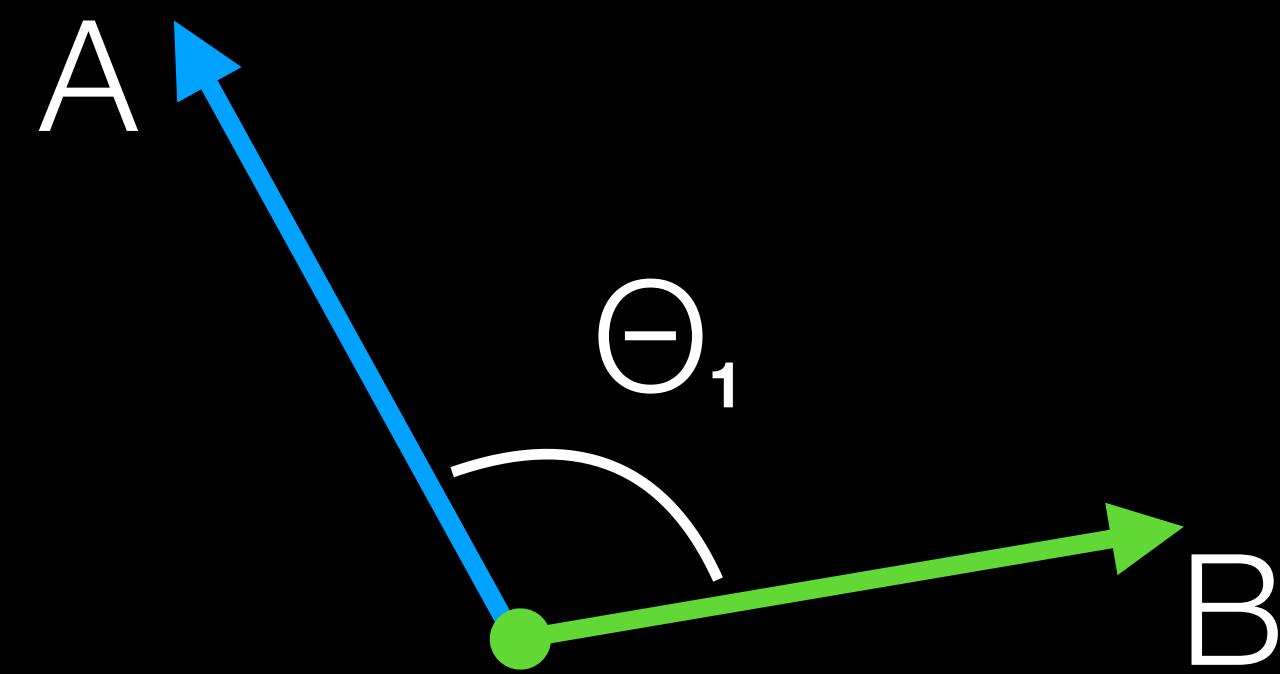


**WARNING: Math Ahead**

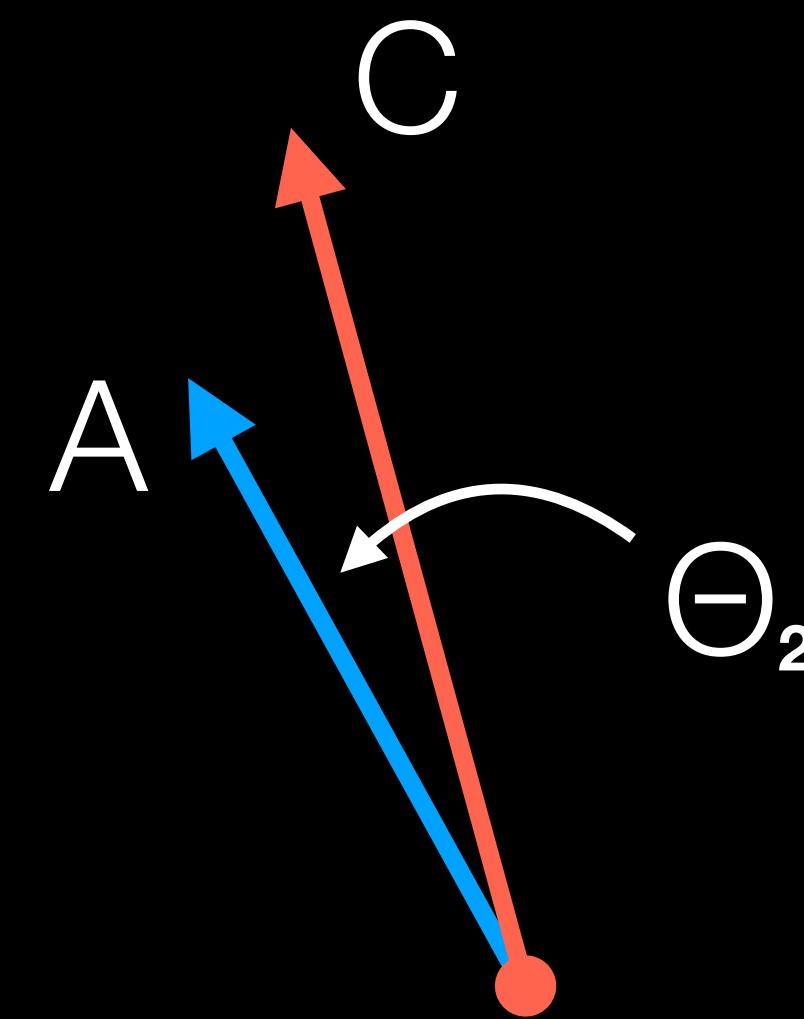
# Comparing 2D Vectors

# Comparing 2D Vectors

## The Dot Product



Less similar

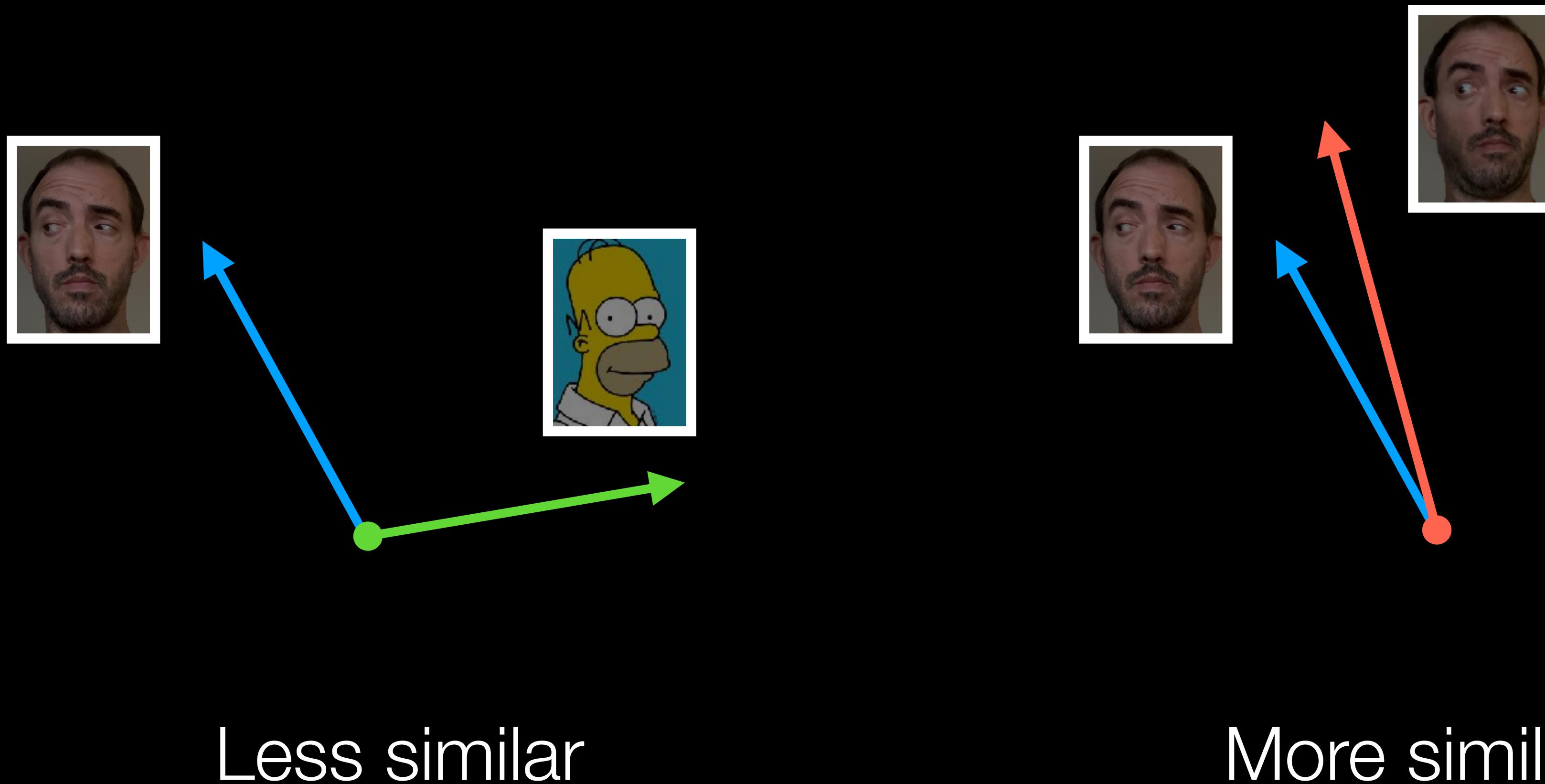


More similar

**Even works on 1000s of dimensions**  
i.e. Feature Vectors



# Comparing Feature Vectors



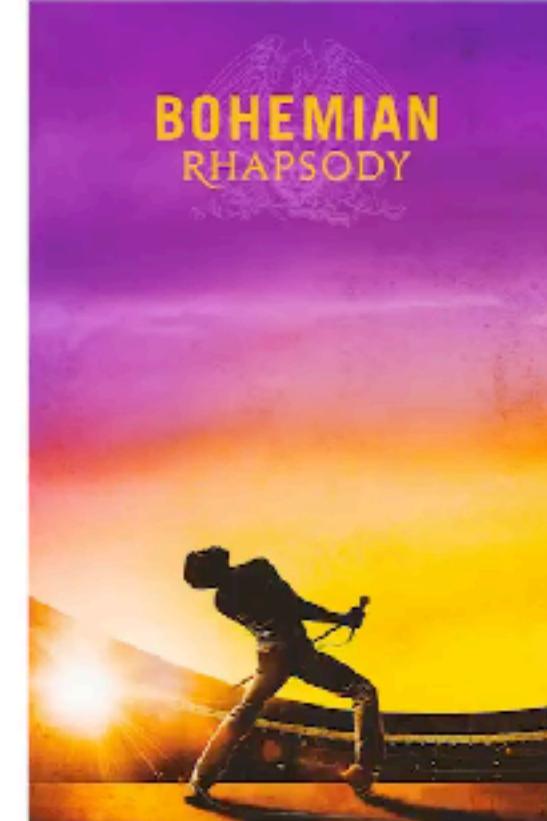
# Feature Vector Demo

Vuu  
Ålesund, NO

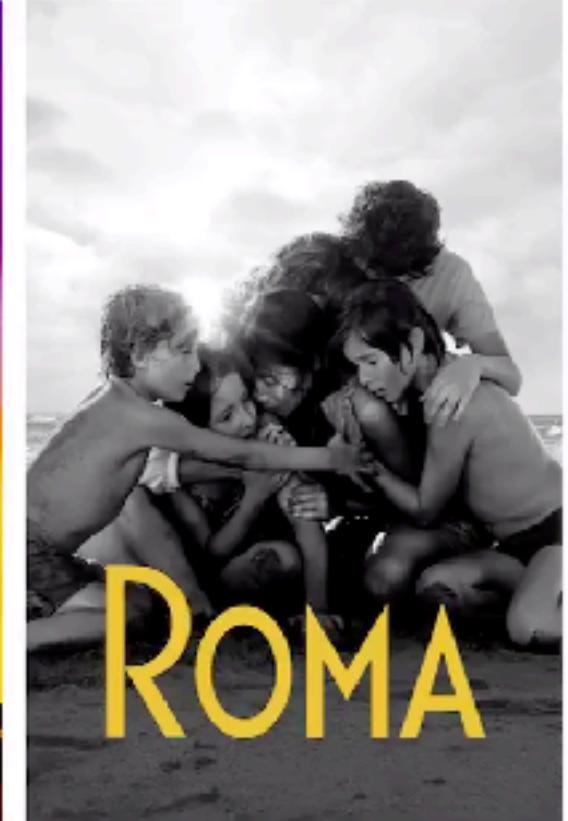
## Search

Movies, TV-Series, Cast or Crew

Featured Movies & TV Series  
Mrs. Robinson, you're trying to seduce me



Bohemian Rhapsody  
2018



Roma  
2018



Activity

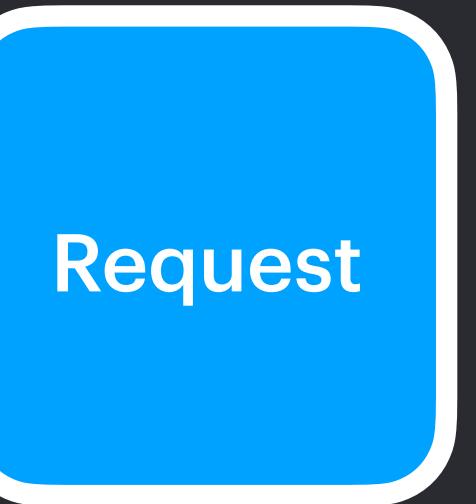
Search

Profile

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



```
// Create the Vision request  
let request = VNGenerateImageFeaturePrintRequest()  
  
// Configure the request  
request.imageCropAndScaleOption = .centerCrop
```



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

// Configure the request
request.imageCropAndScaleOption = .centerCrop

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

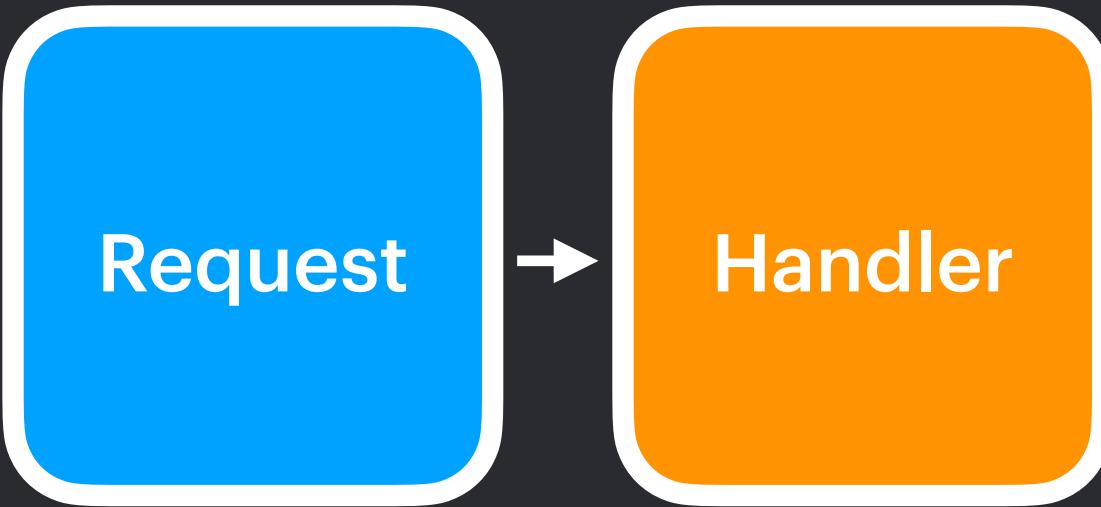


```
// Create the Vision request
let request = VNGenerateImageFeaturePrintRequest()

// Configure the request
request.imageCropAndScaleOption = .centerCrop

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

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



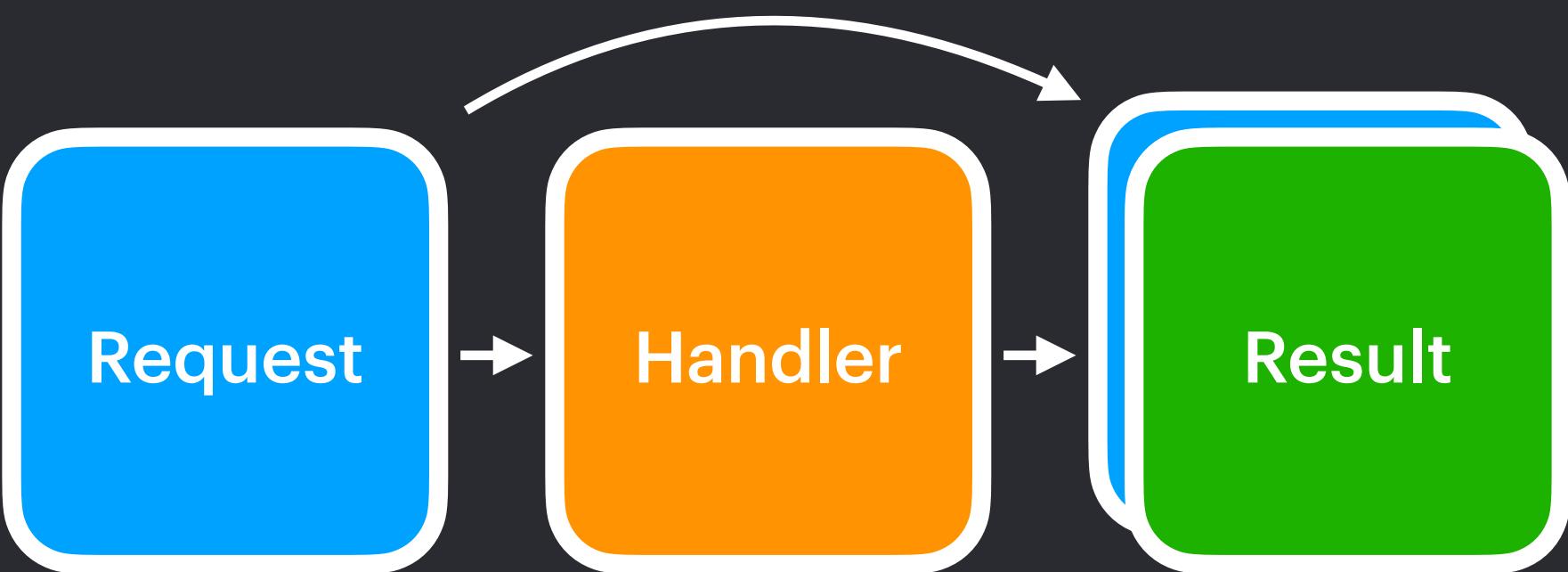
```
// Create the Vision request
let request = VNGenerateImageFeaturePrintRequest()

// 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

**SCORE:**

0

# Object Tracking in the App

SCORE:

0

# Object Tracking in the App

Each detected object has a UUID

SCORE:

0

# Object Tracking in the App

Each detected object has a UUID

Track heads across frames

SCORE:

0

# Object Tracking in the App

Each detected object has a UUID

Track heads across frames

Useful for preventing MHC

SCORE:

0

# Object Tracking in the App

Each detected object has a UUID

Track heads across frames

Useful for preventing MHC\*

\* Multiple Head Crushings

SCORE:

0

# 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]()

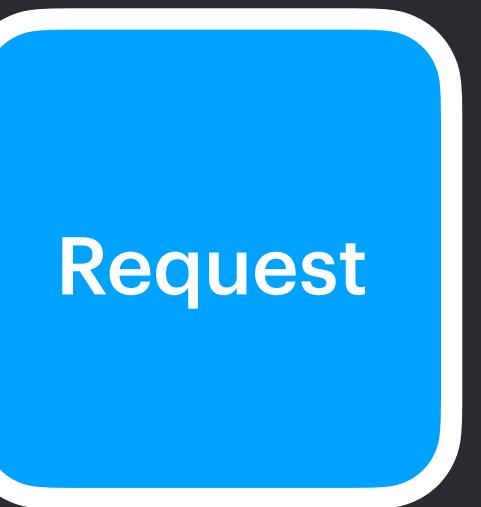
// create a request for every head to be tracked
for head in self.trackedHeads {
    requests.append(VNTrackObjectRequest(detectedObjectObservation: head.value))
}

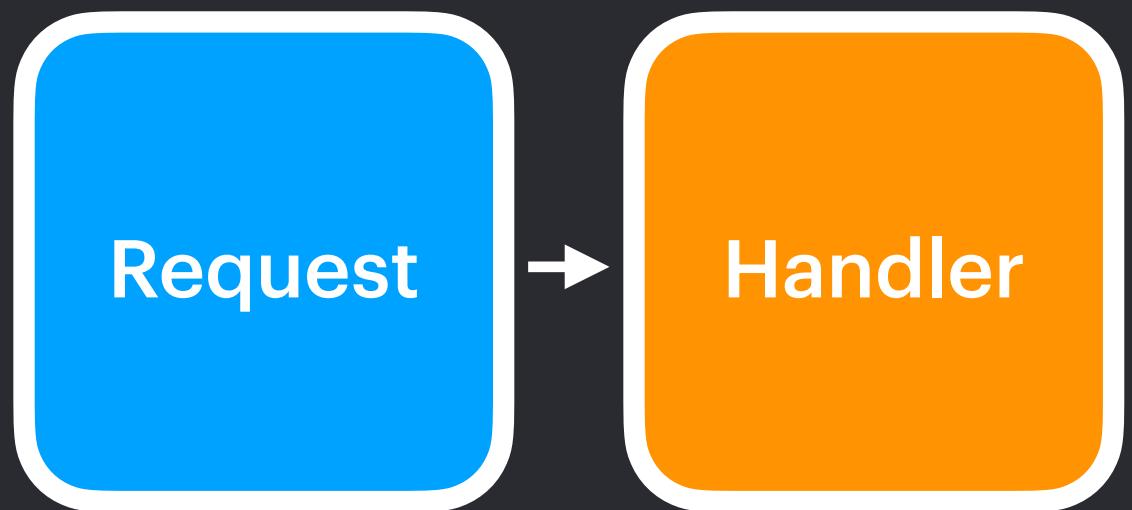
if requests.isEmpty() {
    // detect fresh faces
    requests.append(VNDetectFaceRectanglesRequest())

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



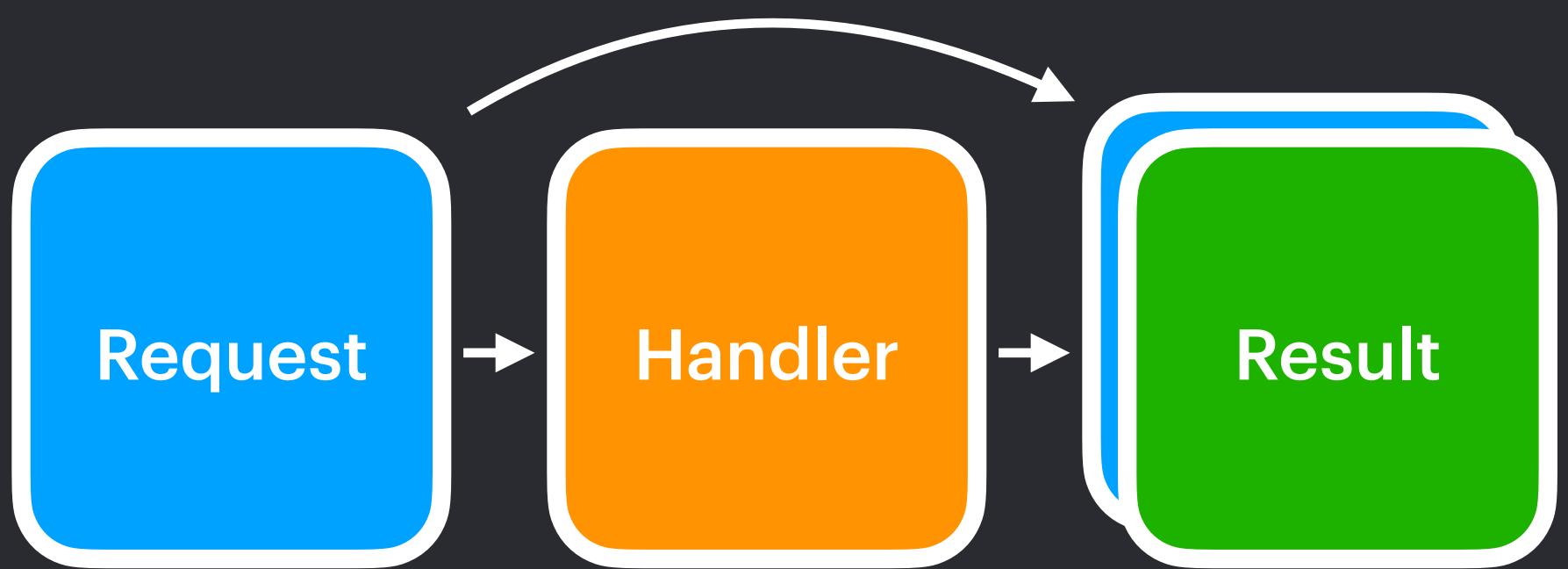
...  
...





...

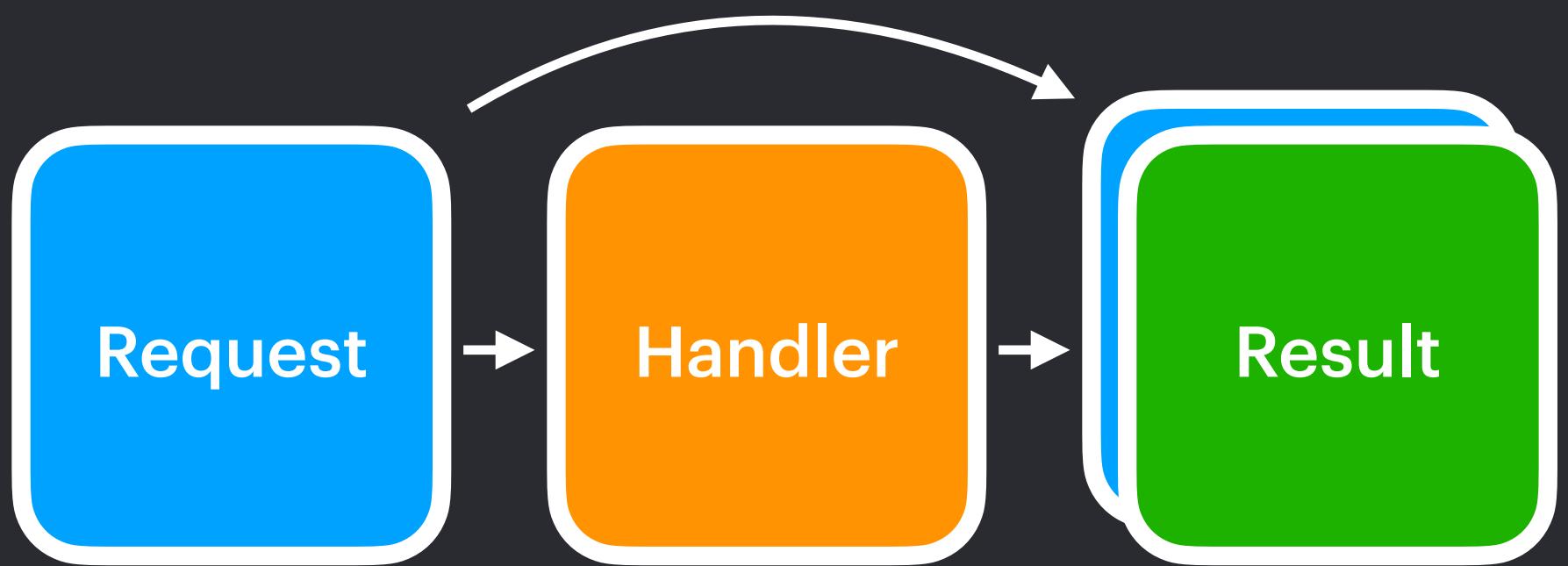
```
// Perform the request  
try? handler.perform(requests, on: image)
```



```
// 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

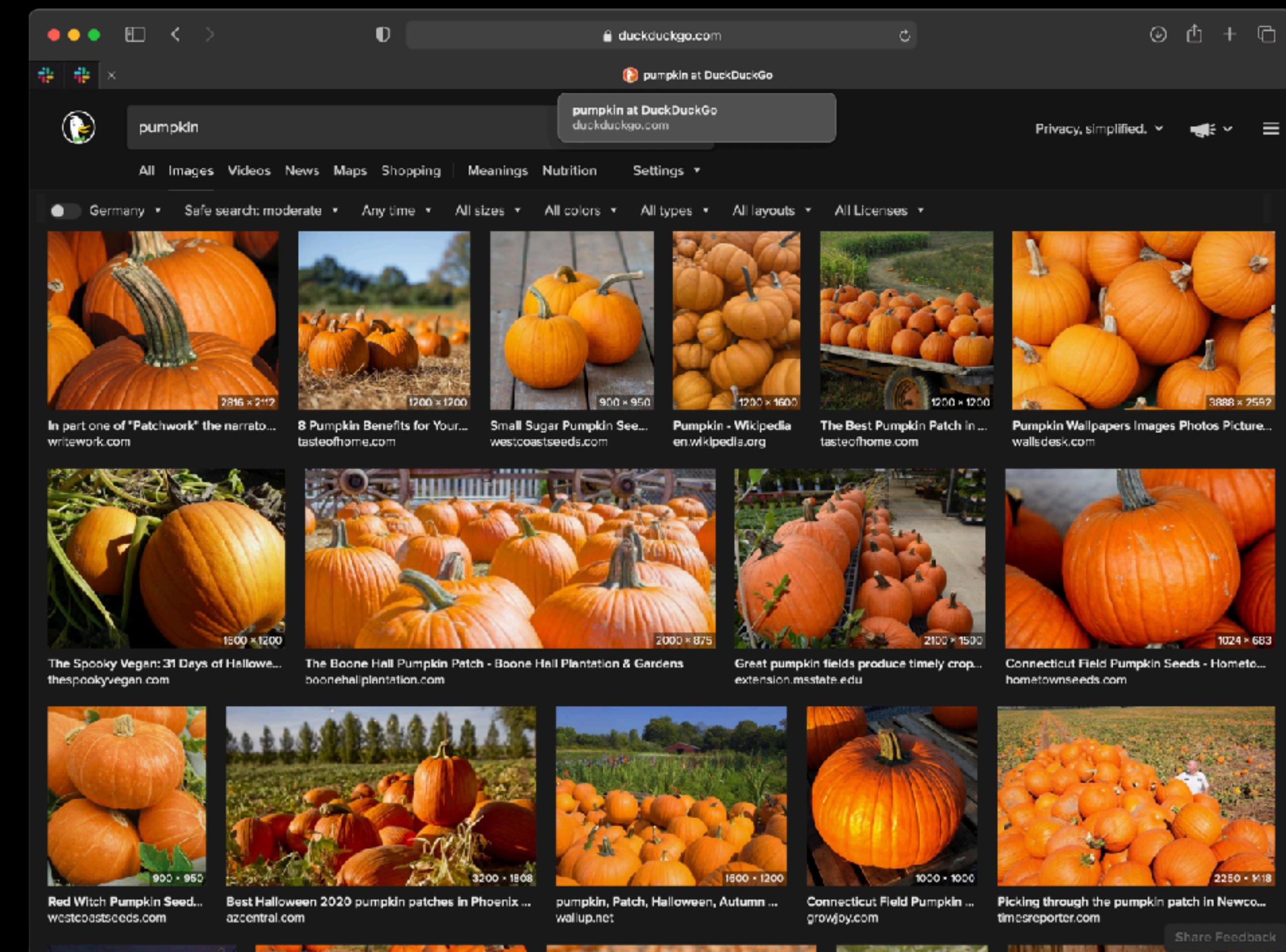
# Smashing Pumpkins

## Train a custom CoreML classifier

# Smashing Pumpkins

## Train a custom CoreML classifier

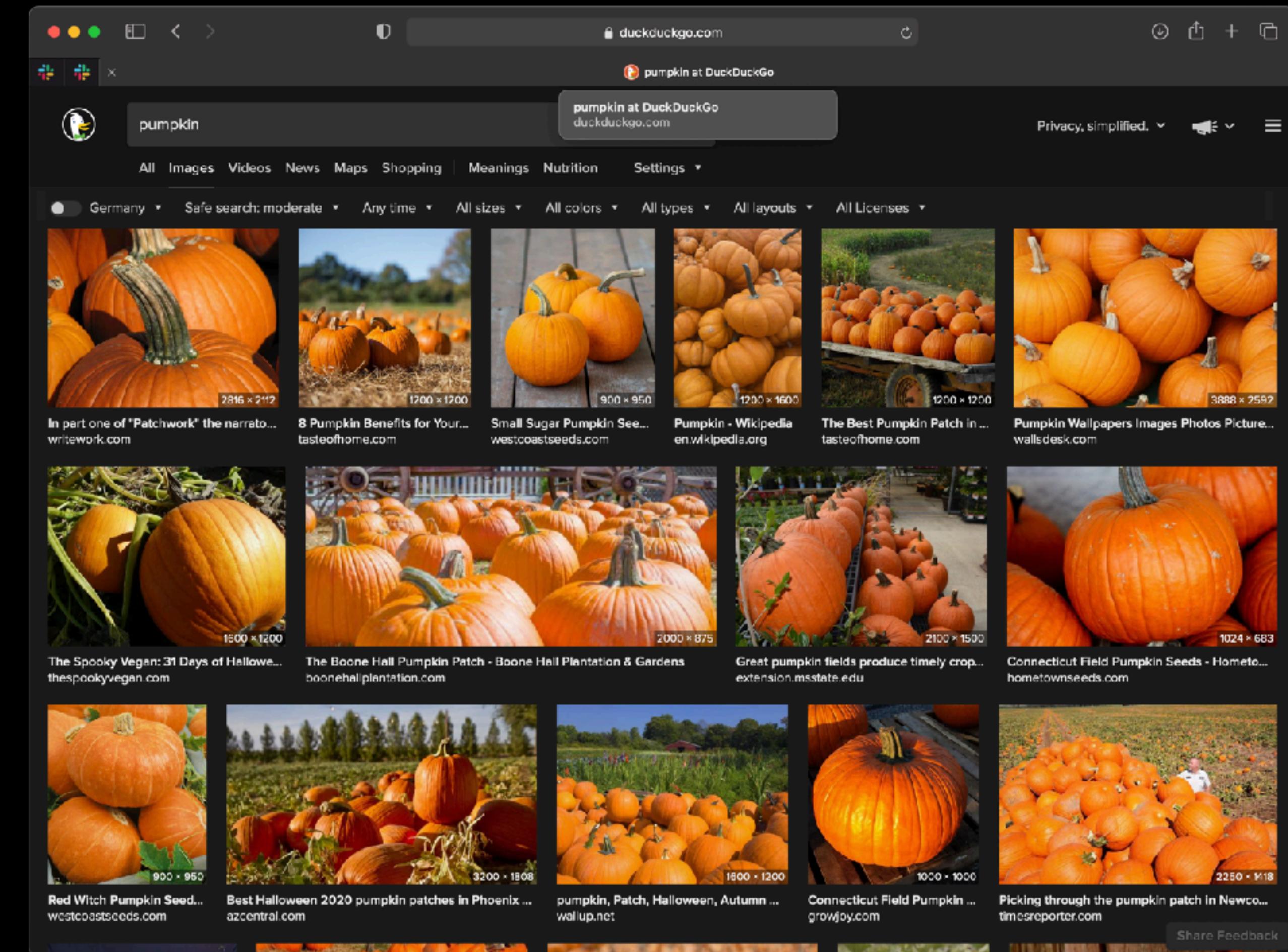
Collect images of pumpkins and not pumpkins



# Smashing Pumpkins

## Train a custom CoreML classifier

Collect images of pumpkins and  
not pumpkins\*



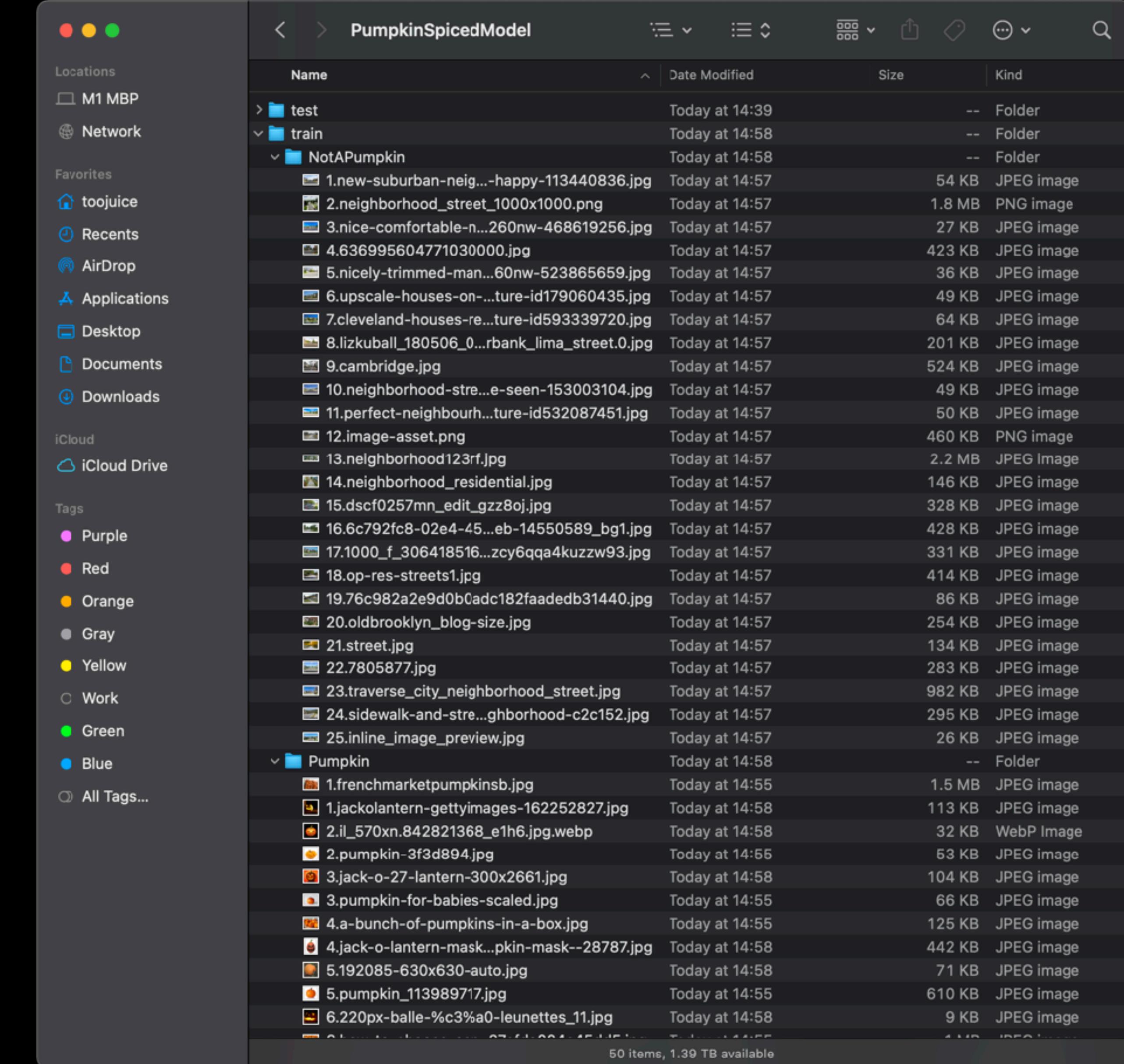
\* Be mindful of image licenses

# Smashing Pumpkins

## Train a custom CoreML classifier

Collect images of pumpkins and  
not pumpkins\*

Organize the folder structure



The screenshot shows a Mac OS X Finder window titled "PumpkinSpicedModel". The left sidebar lists "Locations" (M1 MBP, Network), "Favorites" (toojuice, Recents, AirDrop, Applications, Desktop, Documents, Downloads), and "Tags" (Purple, Red, Orange, Gray, Yellow, Work, Green, Blue, All Tags...). The main pane displays a file list with columns for Name, Date Modified, Size, and Kind. The structure is as follows:

- test (Folder, Today at 14:39)
- train (Folder, Today at 14:58)
  - NotAPumpkin (Folder, Today at 14:58)
    - 1.new-suburban-neig...-happy-113440836.jpg (JPEG image, 54 KB)
    - 2.neighborhood\_street\_1000x1000.png (PNG image, 1.8 MB)
    - 3.nice-comfortable-n...260nw-468619256.jpg (JPEG image, 27 KB)
    - 4.636995604771030000.jpg (JPEG image, 423 KB)
    - 5.nicely-trimmed-man...60nw-523865659.jpg (JPEG image, 36 KB)
    - 6.upscale-houses-on-...ture-id179060435.jpg (JPEG image, 49 KB)
    - 7.cleveland-houses-re...ture-id593339720.jpg (JPEG image, 64 KB)
    - 8.lizkuball\_180506\_0...rbanck\_lima\_street.0.jpg (JPEG image, 201 KB)
    - 9.cambridge.jpg (JPEG image, 524 KB)
    - 10.neighborhood-stre...e-seen-153003104.jpg (JPEG image, 49 KB)
    - 11.perfect-neighbour...ture-id532087451.jpg (JPEG image, 50 KB)
    - 12.image-asset.png (PNG image, 460 KB)
    - 13.neighborhood123rf.jpg (JPEG image, 2.2 MB)
    - 14.neighborhood\_residential.jpg (JPEG image, 146 KB)
    - 15.dscf0257mn\_edit\_gzz8oj.jpg (JPEG image, 328 KB)
    - 16.6c792fc8-02e4-45...eb-14550589\_bg1.jpg (JPEG image, 428 KB)
    - 17.1000\_f\_306418516...zcy6qqa4kuzzw93.jpg (JPEG image, 331 KB)
    - 18.op-res-streets1.jpg (JPEG image, 414 KB)
    - 19.76c982a2e9d0b0adc182faadedb31440.jpg (JPEG image, 86 KB)
    - 20.oldbrooklyn\_blog-size.jpg (JPEG image, 254 KB)
    - 21.street.jpg (JPEG image, 134 KB)
    - 22.7805877.jpg (JPEG image, 283 KB)
    - 23.traverse\_city\_neighborhood\_street.jpg (JPEG image, 982 KB)
    - 24.sidewalk-and-stre...ghborhood-c2c152.jpg (JPEG image, 295 KB)
    - 25.inline\_image\_preview.jpg (JPEG image, 26 KB)
  - Pumpkin (Folder, Today at 14:58)
    - 1.frenchmarketpumpkinsb.jpg (JPEG image, 1.5 MB)
    - 1.jackolantern-gettyimages-162252827.jpg (JPEG image, 113 KB)
    - 2.il\_570xn.842821368\_e1h6.jpg.webp (WebP Image, 32 KB)
    - 2.pumpkin-3f3d894.jpg (JPEG image, 53 KB)
    - 3.jack-o-27-lantern-300x2661.jpg (JPEG image, 104 KB)
    - 3.pumpkin-for-babies-scaled.jpg (JPEG image, 66 KB)
    - 4.a-bunch-of-pumpkins-in-a-box.jpg (JPEG image, 125 KB)
    - 4.jack-o-lantern-mask...pkin-mask--28787.jpg (JPEG image, 442 KB)
    - 5.192085-630x630-auto.jpg (JPEG image, 71 KB)
    - 5.pumpkin\_113989717.jpg (JPEG image, 610 KB)
    - 6.220px-balle-%c3%a0-leunettes\_11.jpg (JPEG image, 9 KB)

\* Be mindful of image licenses

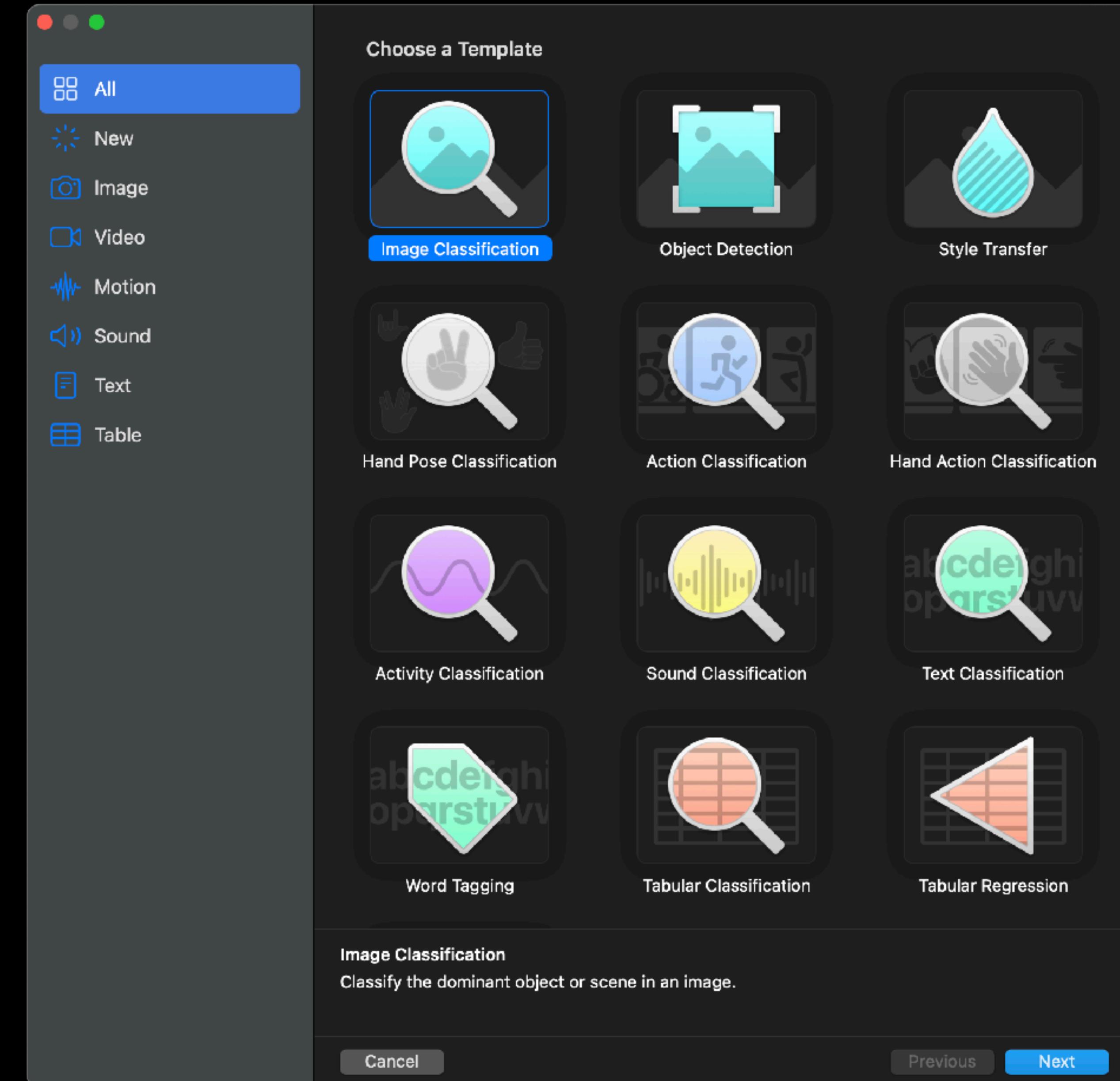
# Smashing Pumpkins

## Train a custom CoreML classifier

Collect images of pumpkins and  
not pumpkins\*

Organize the folder structure

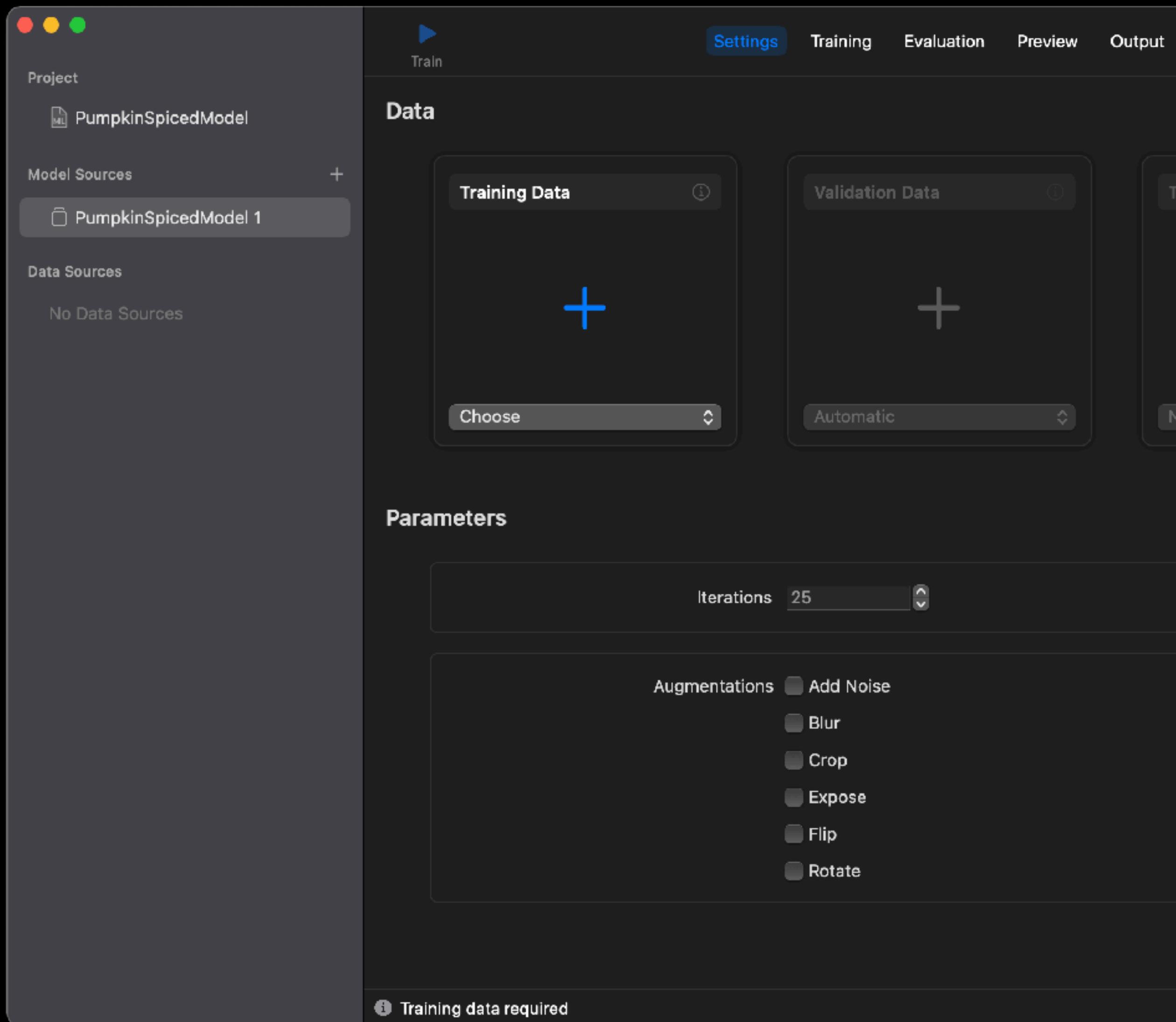
Create an image classification  
project in CreateML



\* Be mindful of image licenses

# Smashing Pumpkins

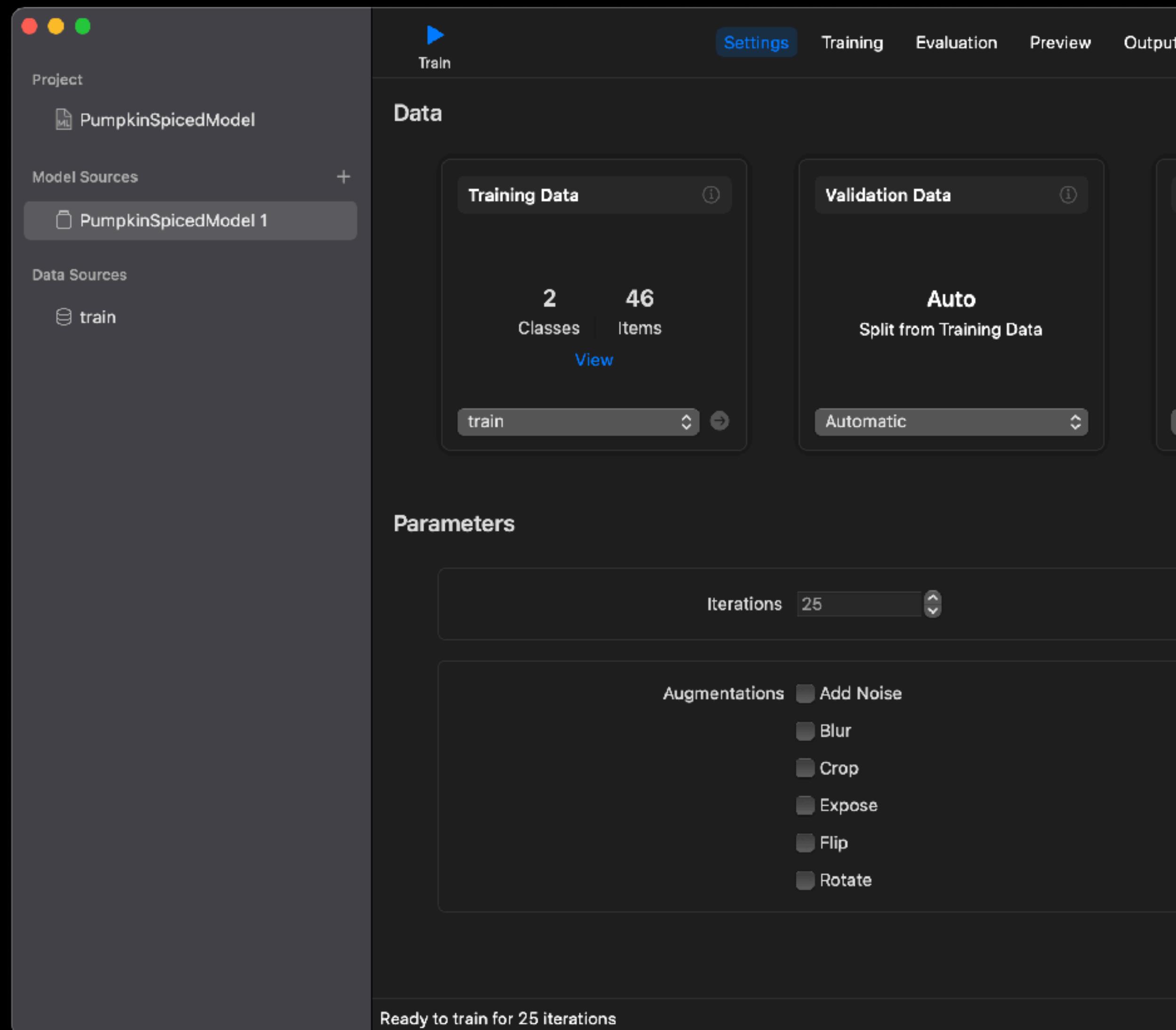
## Train a custom CoreML classifier



# Smashing Pumpkins

## Train a custom CoreML classifier

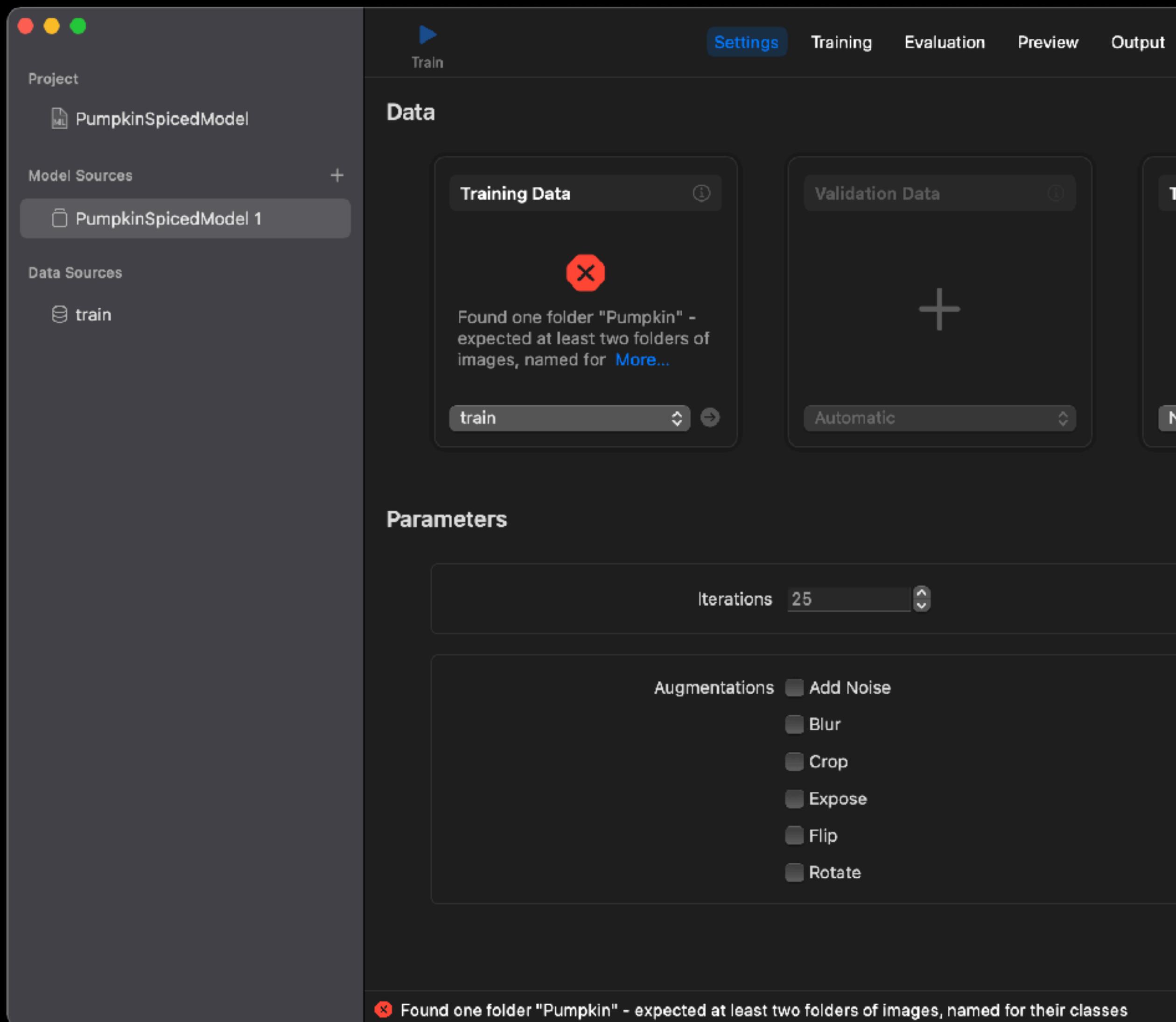
Add the training folder to your project



# Smashing Pumpkins

## Train a custom CoreML classifier

Add the training folder to your project

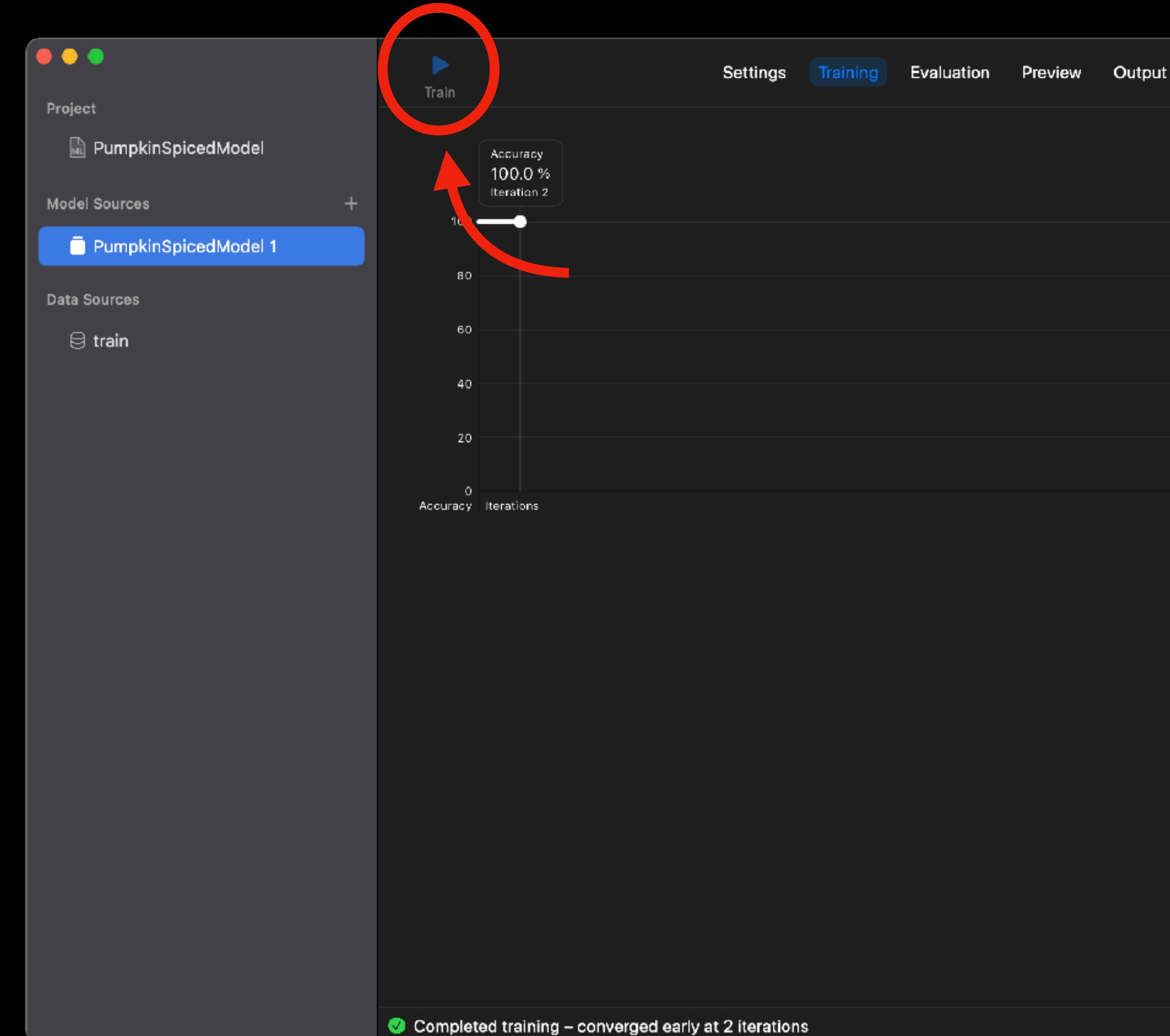


# Smashing Pumpkins

## Train a custom CoreML classifier

Add the training folder to your project

Train



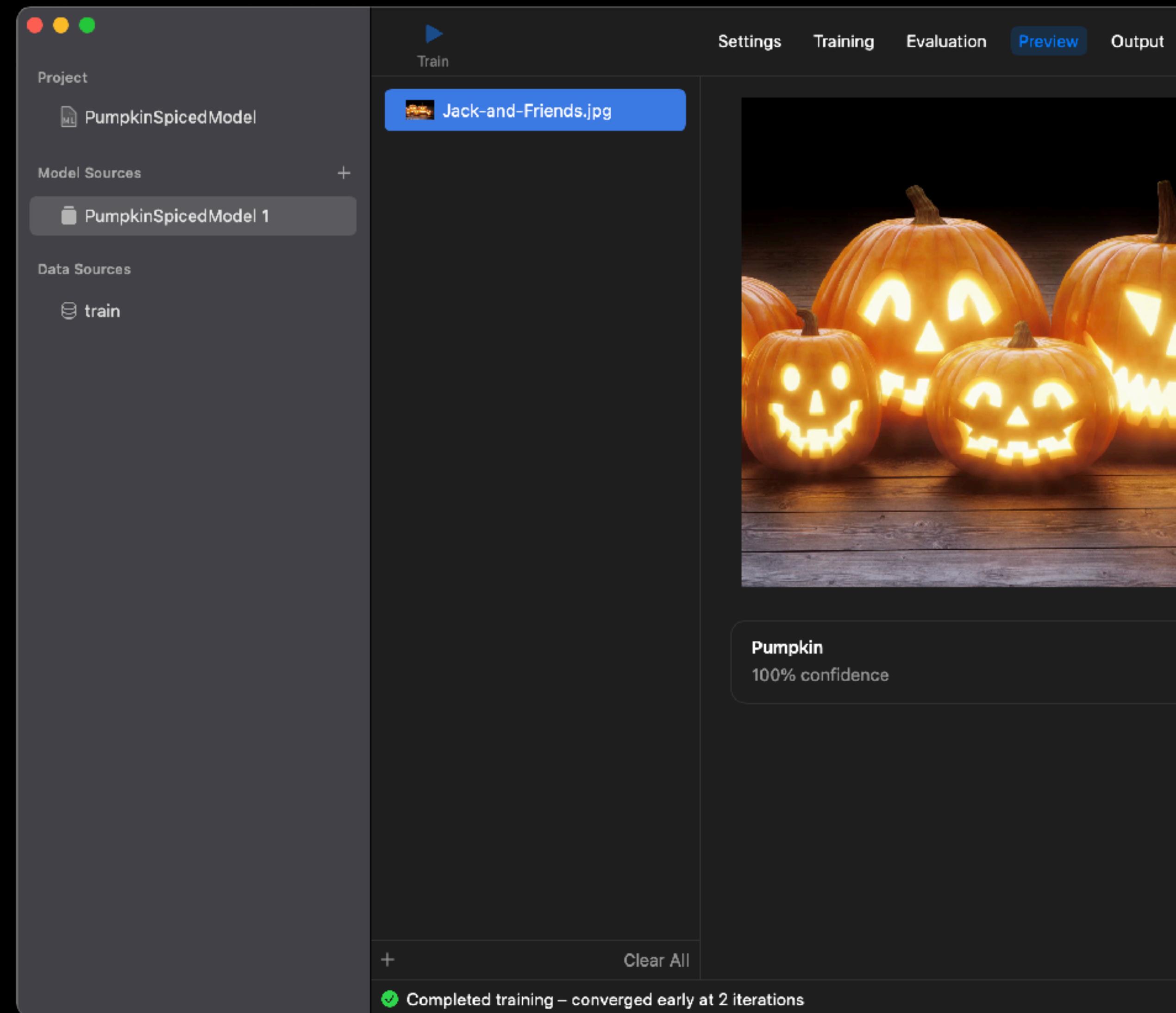
# Smashing Pumpkins

## Train a custom CoreML classifier

Add the training folder to your project

Train

Evaluate or spot check the model



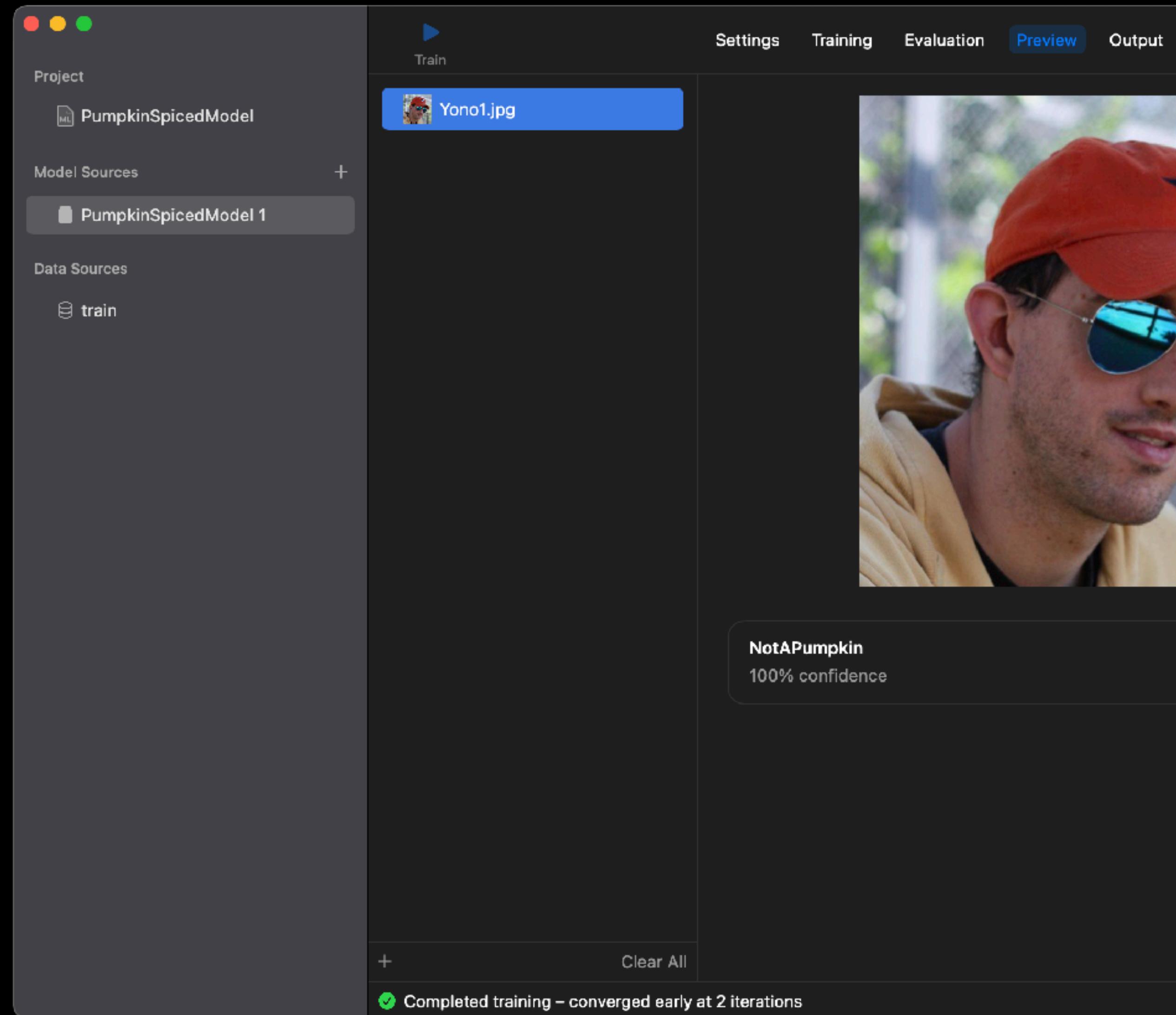
# Smashing Pumpkins

## Train a custom CoreML classifier

Add the training folder to your project

Train

Evaluate or spot check the model



# Smashing Pumpkins

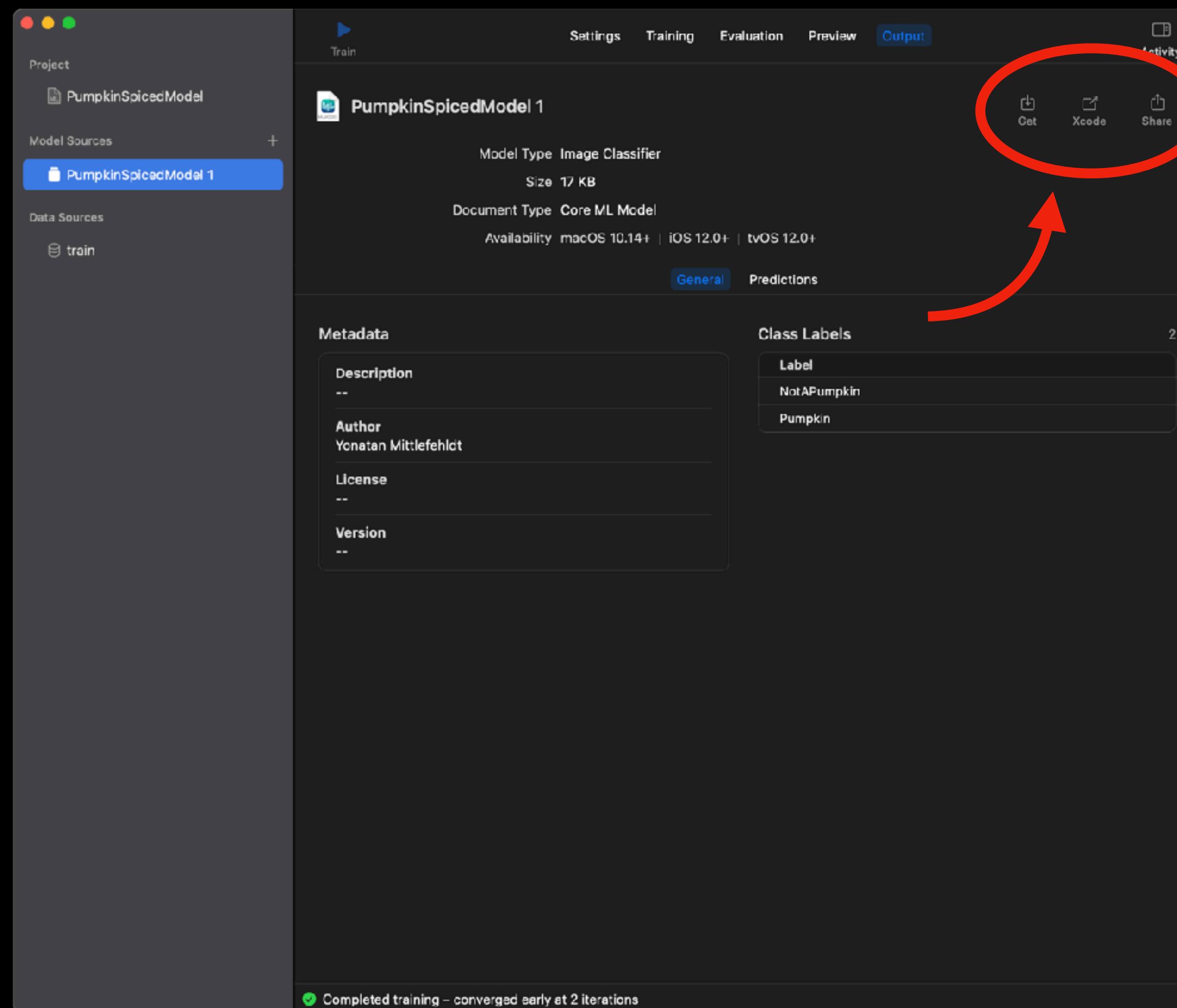
## Train a custom CoreML classifier

Add the training folder to your project

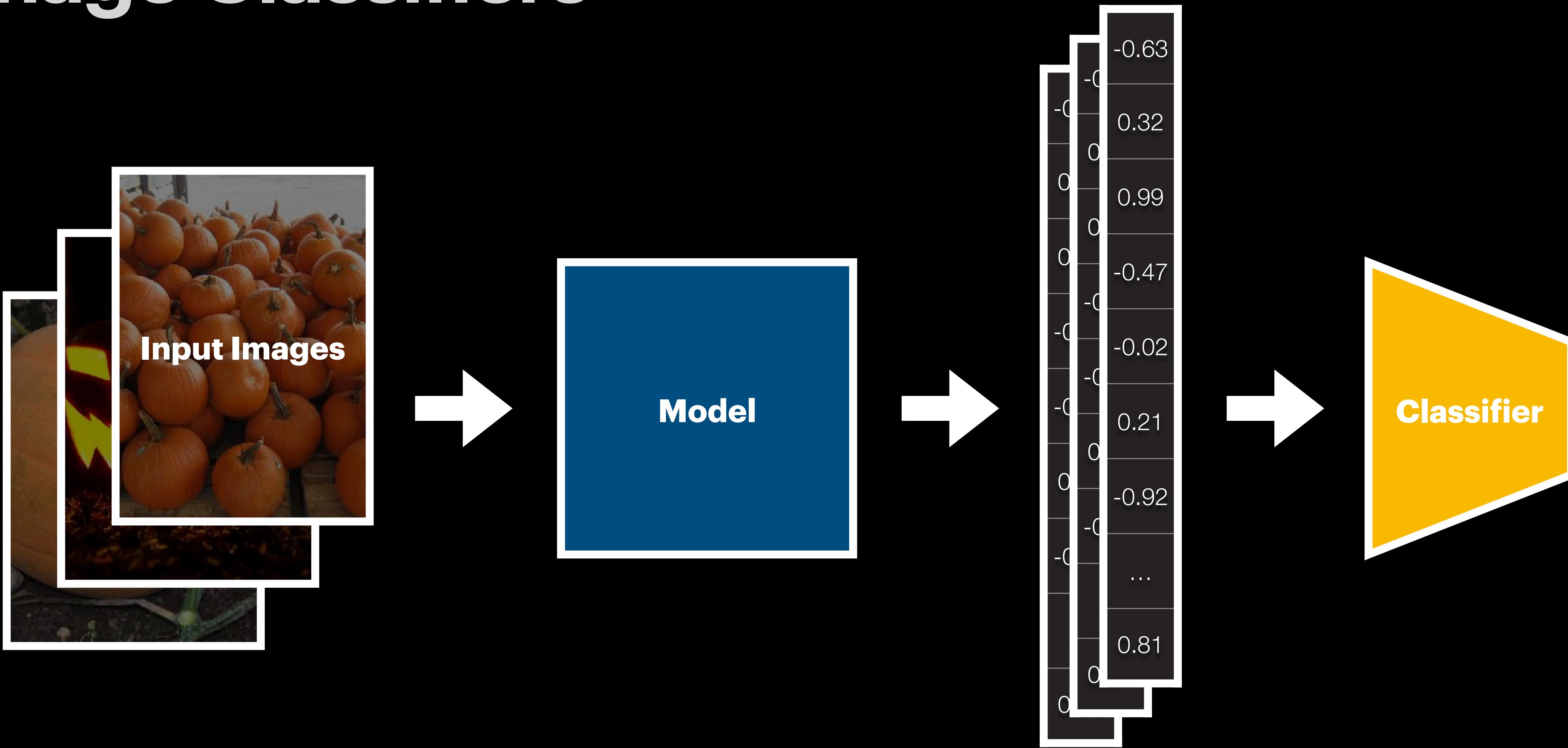
Train

Evaluate or spot check the model

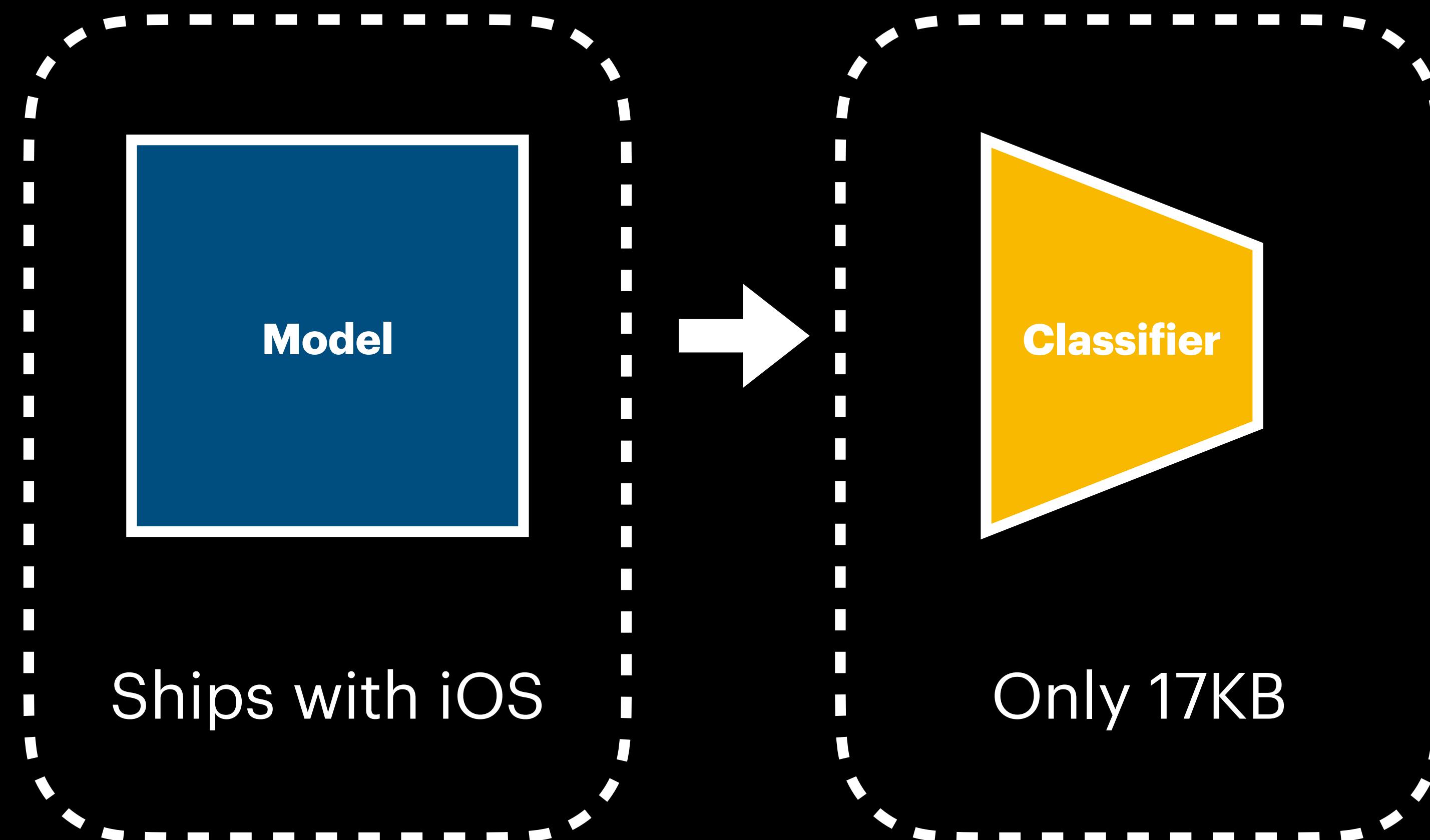
Export the CoreML model



# Image Classifiers



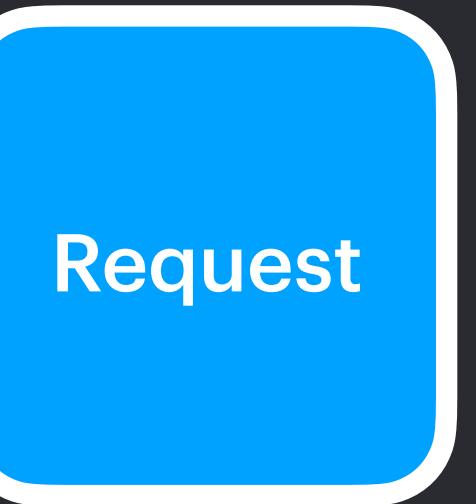
# Image Classifiers



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



```
// Create the Vision request  
let request = VNCoreMLRequest(model: model)  
  
// Configure the request  
request.regionOfInterest = potentialPumpkinRect
```



```
// 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: [:])
```

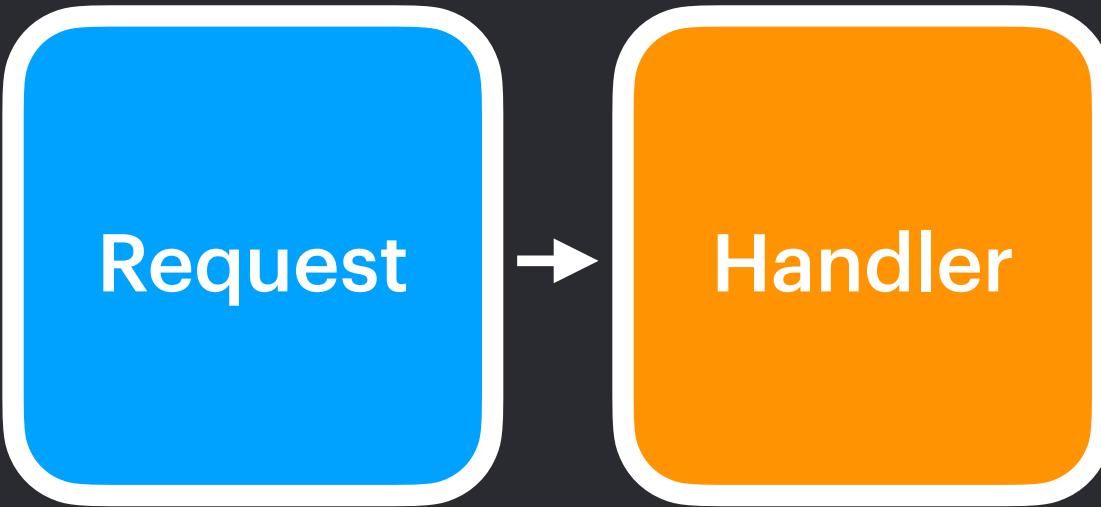


```
// Create the Vision request
let request = VNCoreMLRequest(model: model)

// 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

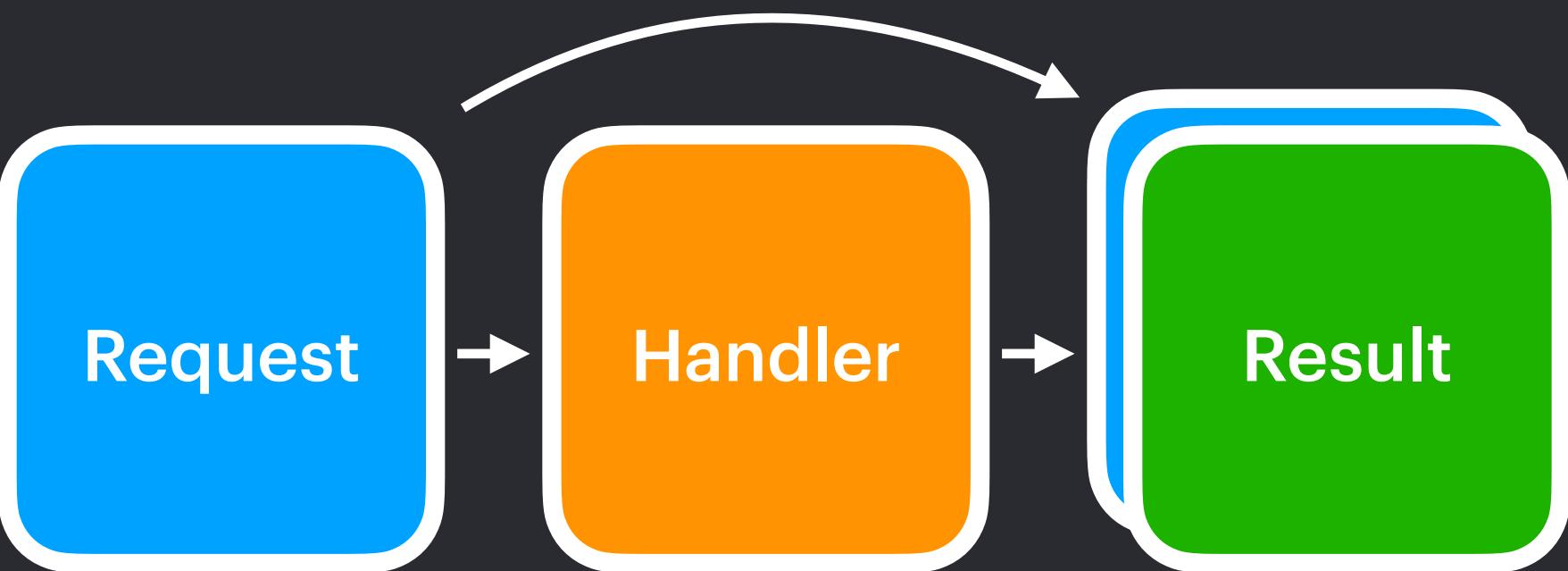
// Create the request handler
let requestHandler = VNImageRequestHandler(cvPixelBuffer: image, options: [:])

// 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)

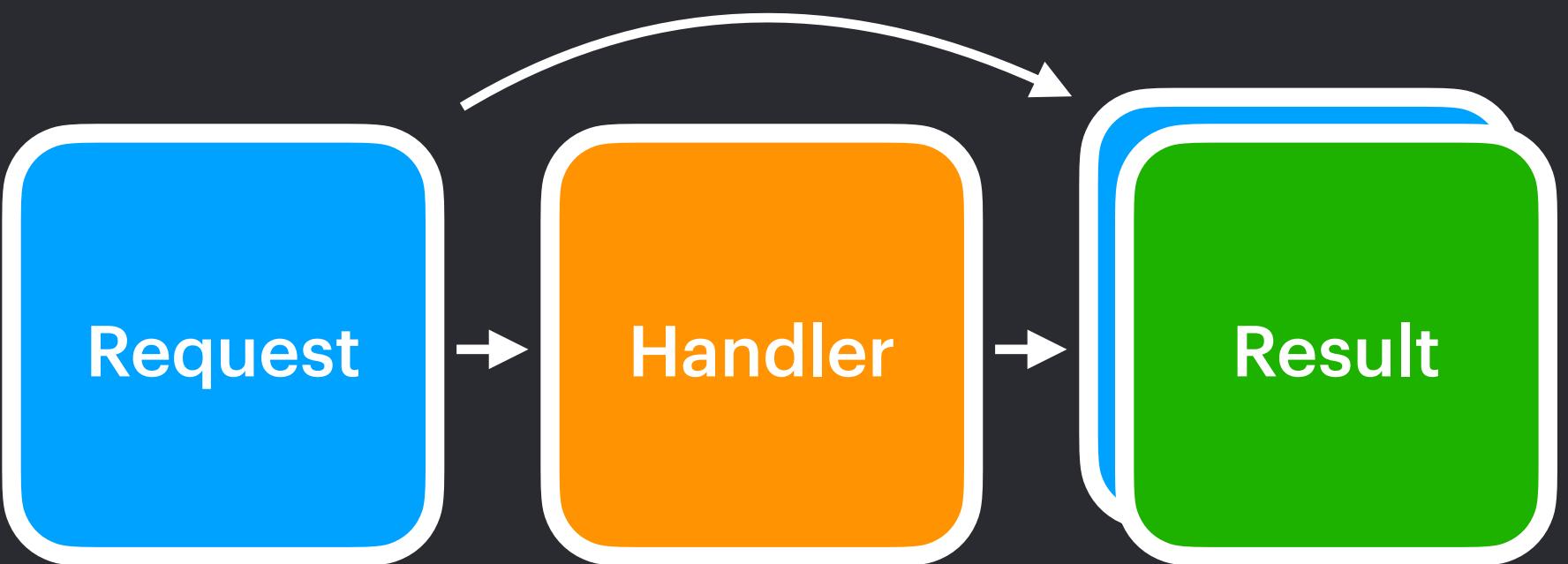
// Configure the request
request.regionOfInterest = potentialPumpkinRect

// 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 {

}
```



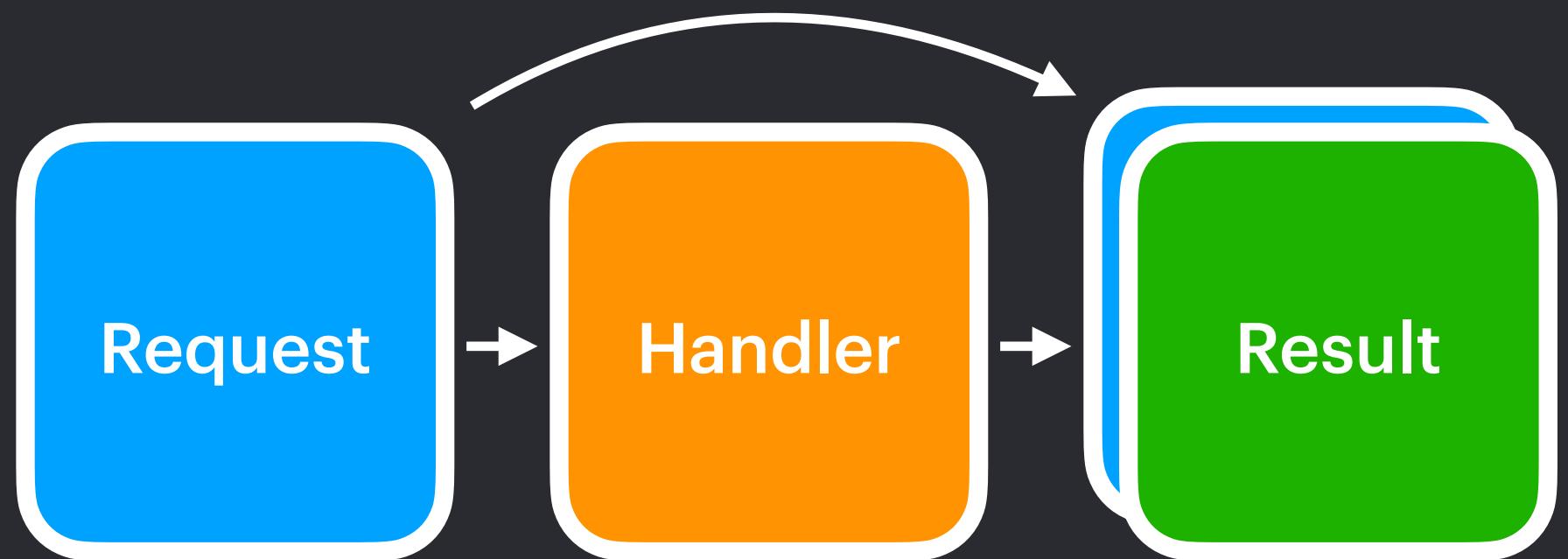
```
// Create the Vision request
let request = VNCoreMLRequest(model: model)

// Configure the request
request.regionOfInterest = potentialPumpkinRect

// Create the request handler
let requestHandler = VNIImageRequestHandler(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

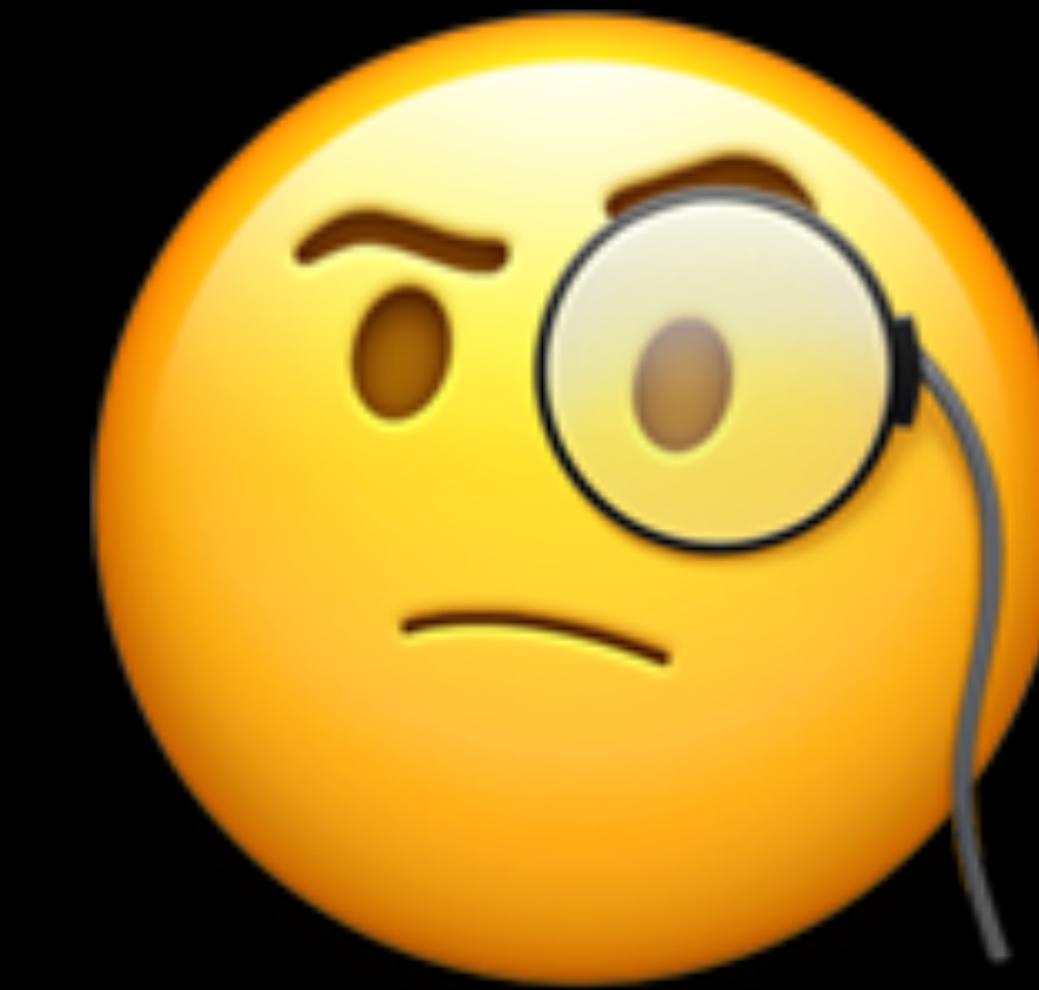
BONUS:

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>