#WWDC19

Understanding Images in Vision Framework

Brittany Weinert, Vision
Rohan Chandra, Vision
Sergey Kamensky, Vision

© 2019 Apple Inc. All rights reserved. Redistribution or public display not permitted without written permission from Apple.

Saliency

Image Classification

Image Similarity

New detectors

New revisions

Saliency

Brittany Weinert, Vision Team

What Catches Your Eye?

What Catches Your Eye?

Attention and Objectness Based Saliency

Attention and Objectness Based Saliency

Attention Based

- Human Aspected
- Trained on eye movements

Attention and Objectness Based Saliency

Attention Based

- Human Aspected
- Trained on eye movements

Objectness Based

- Foreground Objects
- Trained on object segmentation

Attention and Objectness Based Saliency

Attention and Objectness Based Saliency

Attention and Objectness Based Saliency

Attention and Objectness Based Saliency

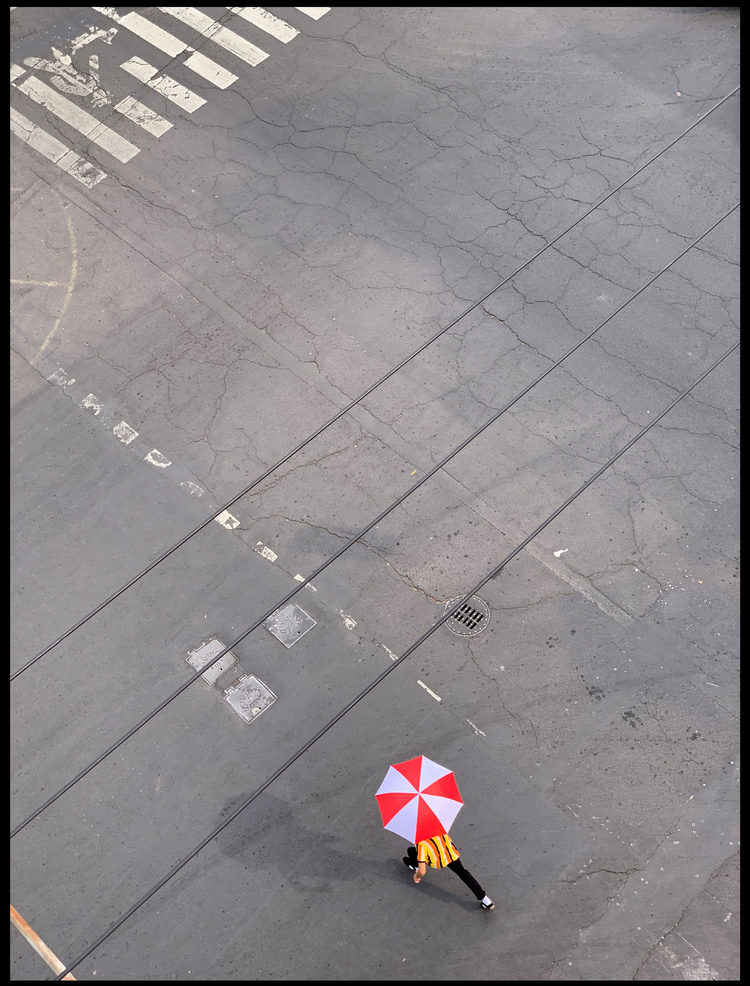
Attention and Objectness Based Saliency

Attention and Objectness Based Saliency

Attention and Objectness Based Saliency

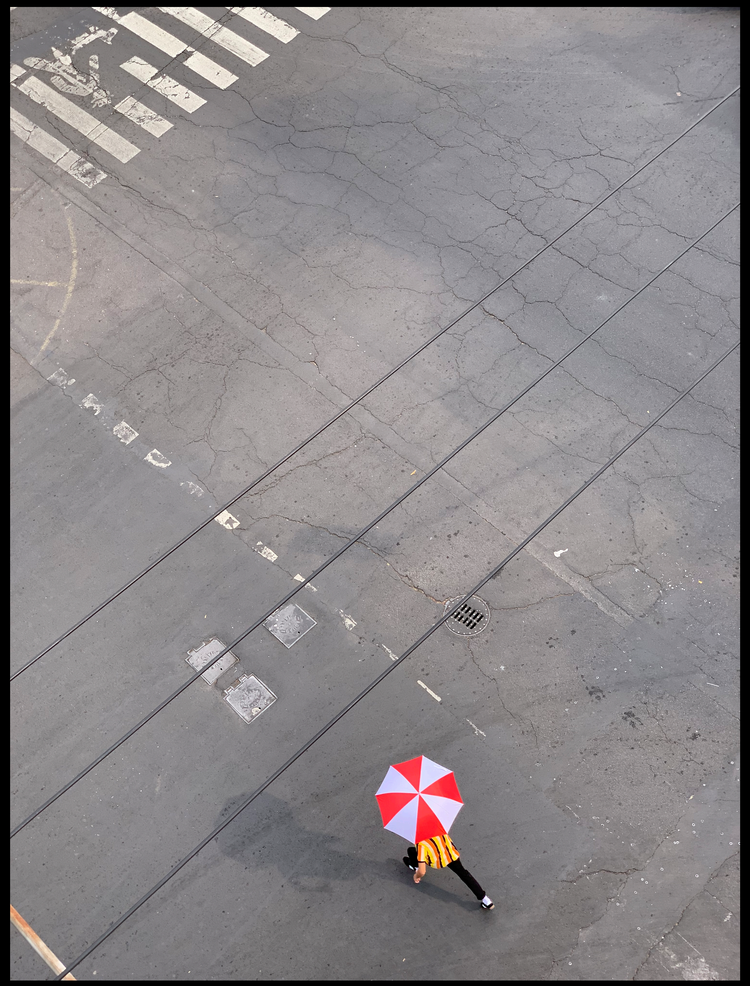
Attention and Objectness Based Saliency

Attention and Objectness Based Saliency

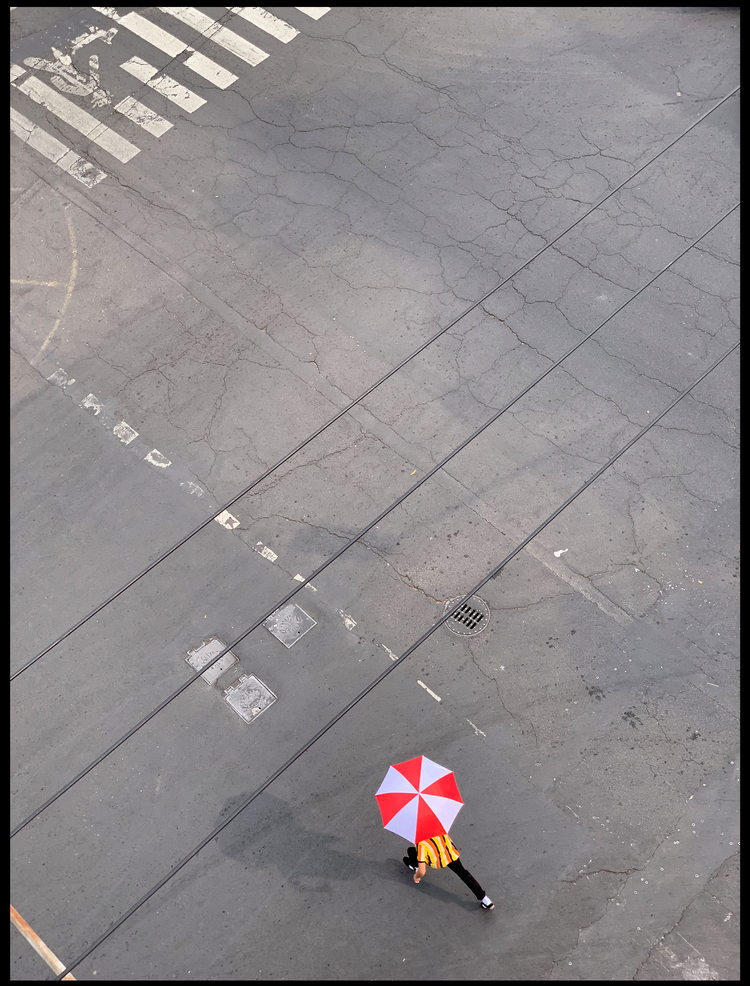


- Contrast
- Faces
- Subjects
- Horizons
- Light

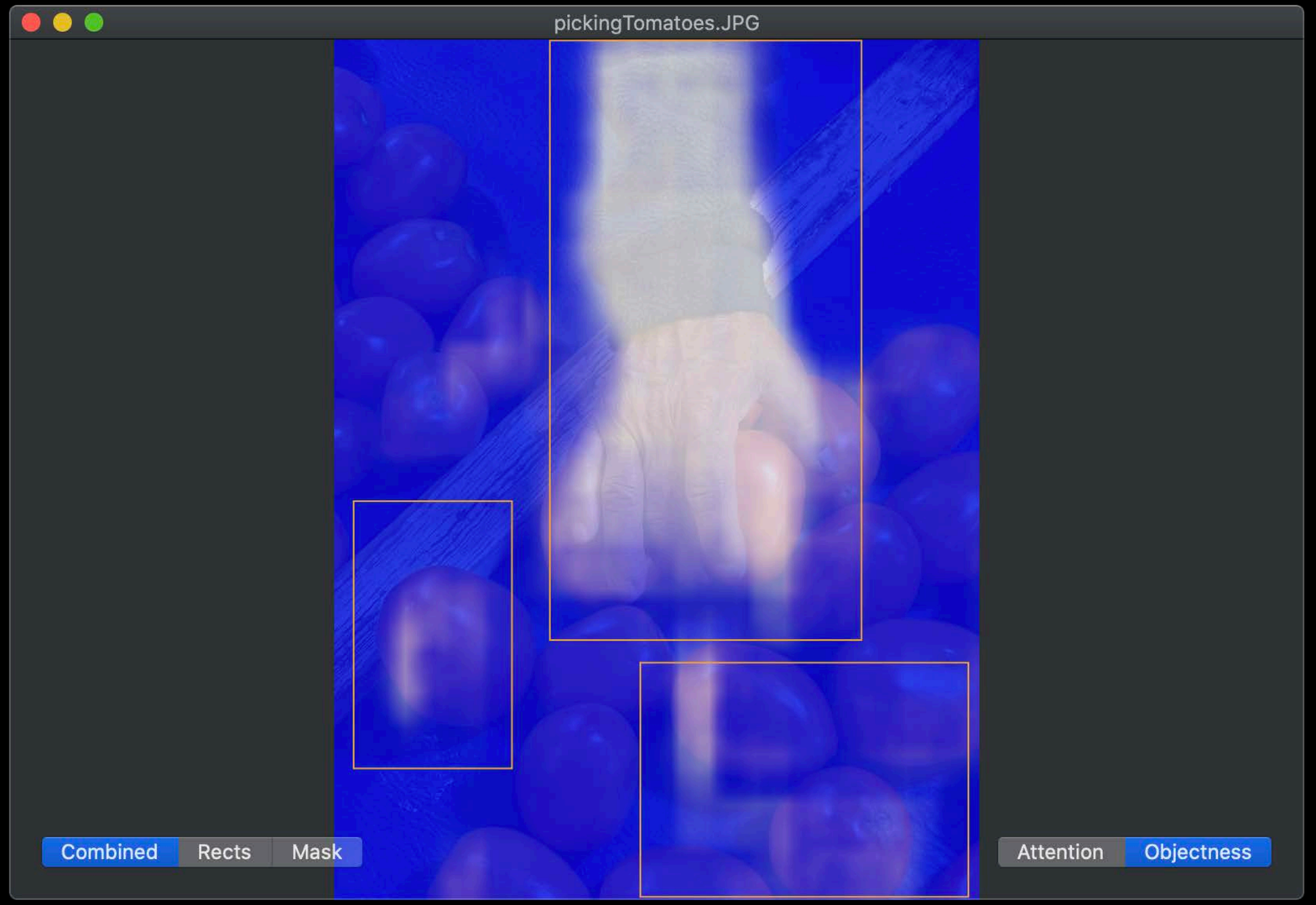
Attention and Objectness Based Saliency



Attention and Objectness Based Saliency



Saliency Sample App



The Heatmap

The Heatmap

The Heatmap

The Heatmap

```
// Making a Request
  let handler = VNImageRequestHandler(url: imageURL)
  let request: VNImageBasedRequest = VNGenerateAttentionBasedSaliencyImageRequest()
  request.revision = VNGenerateAttentionBasedSaliencyImageRequestRevision1

  try? handler.perform([request])
  guard let result = request.results?.first
  let observation = result as? VNSaliencyImageObservation
  else { fatalError("missing result") }

  let pixelBuffer = observation.pixelBuffer
```

```swift
// Making a Request
let handler = VNImageRequestHandler(url: imageURL)
let request: VNImageBasedRequest = VNGenerateAttentionBasedSaliencyImageRequest()
request.revision = VNGenerateAttentionBasedSaliencyImageRequestRevision1

try? handler.perform([request])
guard let result = request.results?.first
let observation = result as? VNSaliencyImageObservation
else { fatalError("missing result") }

let pixelBuffer = observation.pixelBuffer
```

```
// Making a Request
  let handler = VNImageRequestHandler(url: imageURL)
  let request: VNImageBasedRequest = VNGenerateAttentionBasedSaliencyImageRequest()
  request.revision = VNGenerateAttentionBasedSaliencyImageRequestRevision1

  try? handler.perform([request])
  guard let result = request.results?.first
  let observation = result as? VNSaliencyImageObservation
  else { fatalError("missing result") }

  let pixelBuffer = observation.pixelBuffer
```

```swift
// Making a Request
  let handler = VNImageRequestHandler(url: imageURL)
  let request: VNImageBasedRequest = VNGenerateAttentionBasedSaliencyImageRequest()
  request.revision = VNGenerateAttentionBasedSaliencyImageRequestRevision1

  try? handler.perform([request])
  guard let result = request.results?.first
  let observation = result as? VNSaliencyImageObservation
  else { fatalError("missing result") }

  let pixelBuffer = observation.pixelBuffer
```

```
// Making a Request
  let handler = VNImageRequestHandler(url: imageURL)
  let request: VNImageBasedRequest = VNGenerateAttentionBasedSaliencyImageRequest()
  request.revision = VNGenerateAttentionBasedSaliencyImageRequestRevision1

  try? handler.perform([request])
  guard let result = request.results?.first
  let observation = result as? VNSaliencyImageObservation
  else { fatalError("missing result") }

  let pixelBuffer = observation.pixelBuffer
```

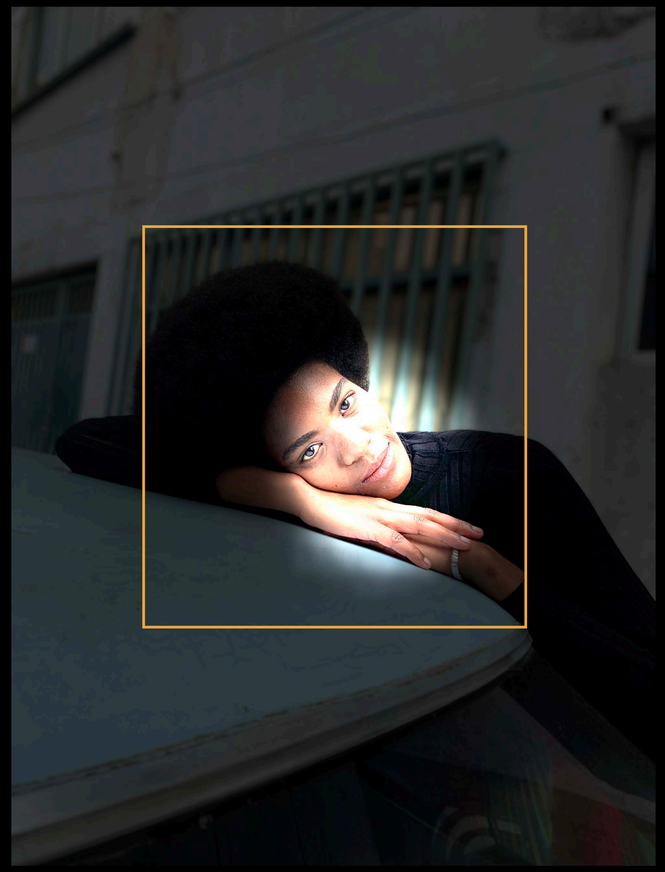
```
// Making a Request
  let handler = VNImageRequestHandler(url: imageURL)
  let request: VNImageBasedRequest = VNGenerateAttentionBasedSaliencyImageRequest()
  request.revision = VNGenerateAttentionBasedSaliencyImageRequestRevision1

  try? handler.perform([request])
  guard let result = request.results?.first
  let observation = result as? VNSaliencyImageObservation
  else { fatalError("missing result") }

  let pixelBuffer = observation.pixelBuffer
```

```swift
// Making a Request
  let handler = VNImageRequestHandler(url: imageURL)
  let request: VNImageBasedRequest = VNGenerateAttentionBasedSaliencyImageRequest()
  request.revision = VNGenerateAttentionBasedSaliencyImageRequestRevision1

  try? handler.perform([request])
  guard let result = request.results?.first
  let observation = result as? VNSaliencyImageObservation
  else { fatalError("missing result") }

  let pixelBuffer = observation.pixelBuffer
```

```swift
// Making a Request
  let handler = VNImageRequestHandler(url: imageURL)
  let request: VNImageBasedRequest = VNGenerateAttentionBasedSaliencyImageRequest()
  request.revision = VNGenerateAttentionBasedSaliencyImageRequestRevision1

  try? handler.perform([request])
  guard let result = request.results?.first
  let observation = result as? VNSaliencyImageObservation
  else { fatalError("missing result") }

  let pixelBuffer = observation.pixelBuffer
```

```
// Making a Request
  let handler = VNImageRequestHandler(url: imageURL)
  let request: VNImageBasedRequest = VNGenerateObjectnessBasedSaliencyImageRequest()
  request.revision = VNGenerateObjectnessBasedSaliencyImageRequestRevision1

  try? handler.perform([request])
  guard let result = request.results?.first
  let observation = result as? VNSaliencyImageObservation
  else { fatalError("missing result") }

  let pixelBuffer = observation.pixelBuffer
```

Bounding Boxes



```
func addSalientObjects(in observation: VNSaliencyImageObservation,
                                to path: CGMutablePath, transform:CGAffineTransform)
{
    guard let objects = observation.salientObjects else { return }
    for object in objects {
        path.addRect(object.boundingBox, transform:transform)
    }
}
```

```
func addSalientObjects(in observation: VNSaliencyImageObservation,
                                to path: CGMutablePath, transform:CGAffineTransform)
{
    guard let objects = observation.salientObjects else { return }
    for object in objects {
        path.addRect(object.boundingBox, transform:transform)
    }
}
```

```
func addSalientObjects(in observation: VNSaliencyImageObservation,
                                to path: CGMutablePath, transform:CGAffineTransform)
{
    guard let objects = observation.salientObjects else { return }
    for object in objects {
        path.addRect(object.boundingBox, transform:transform)
    }
}
```

Graphical Uses

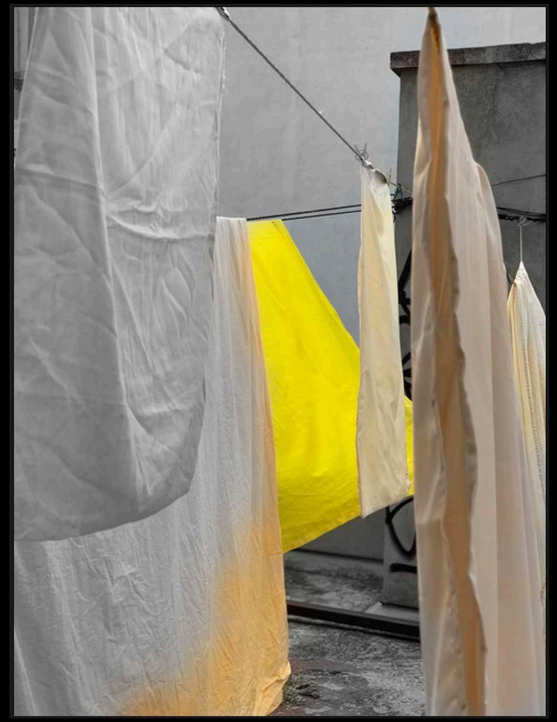
Experiment with a different type of filter or photo transition

Graphical Uses

Experiment with a different type of filter or photo transition

Graphical Uses

Experiment with a different type of filter or photo transition

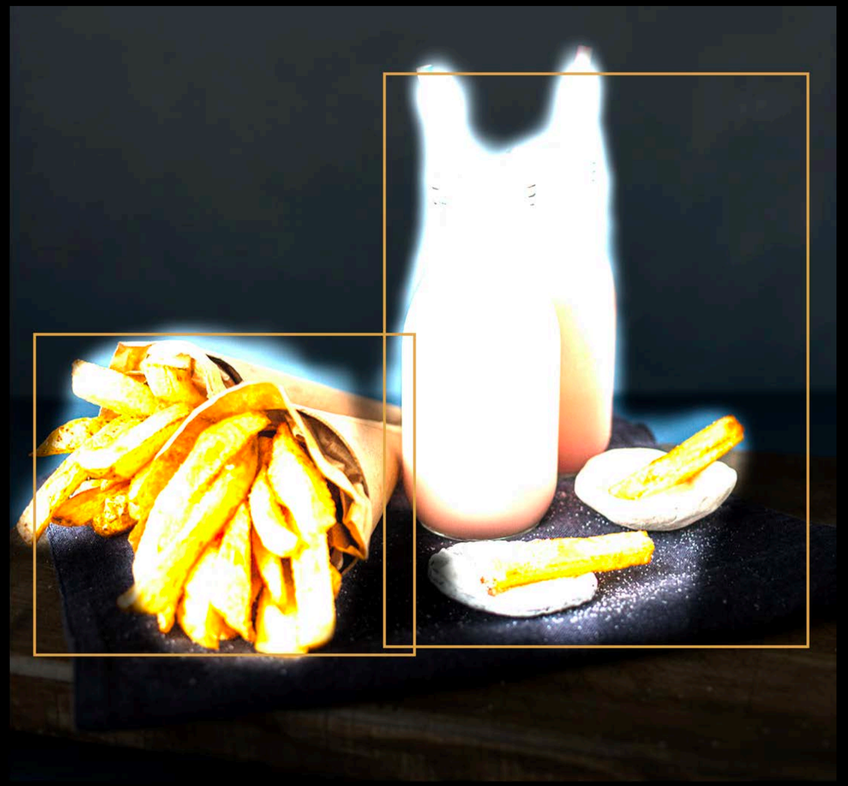



Working Together with Other Vision Algorithms

Use classification to recognize the salient objects

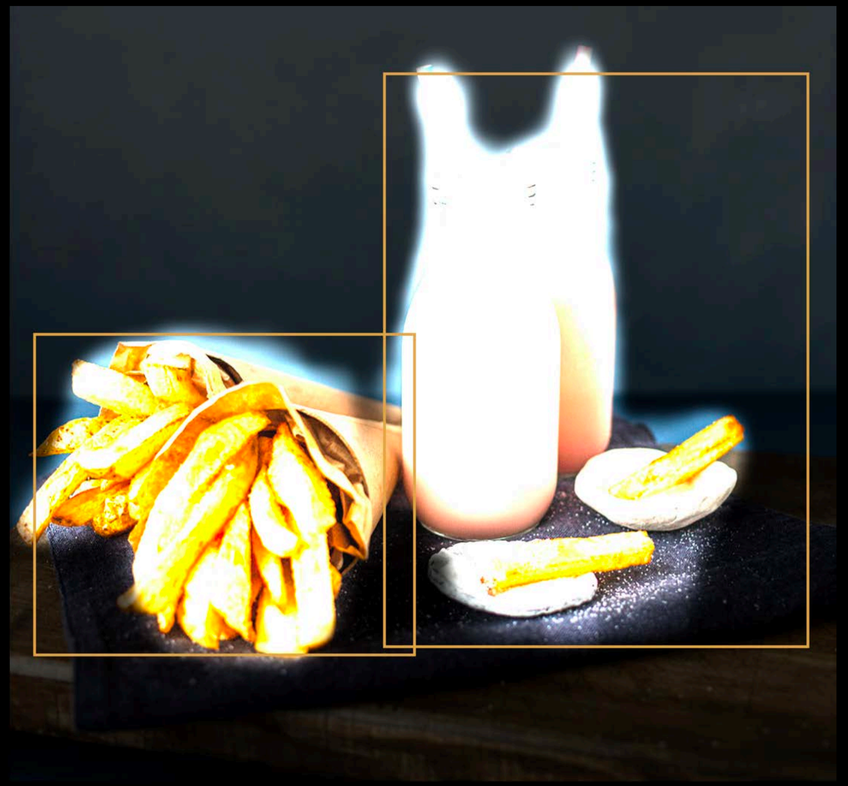
Working Together with Other Vision Algorithms

Use classification to recognize the salient objects



Working Together with Other Vision Algorithms

Use classification to recognize the salient objects



Classification and Image Similarity

Rohan Chandra, Vision Team

Classification basics

Taxonomy

Interpreting observations

Image Similarity

Classification basics

Taxonomy

Interpreting observations

Image Similarity

Classification in the Vision API

What is Classification

What objects are
in the image?

kitten blanket child

chair

table bowl textile

Difficulties in Large Scale Classification

Difficulties in Large Scale Classification

Annotated Data

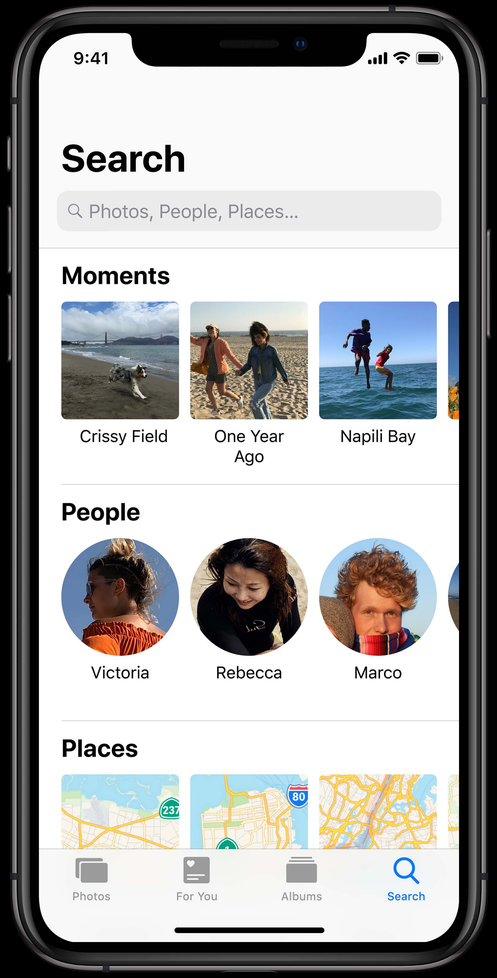
Difficulties in Large Scale Classification

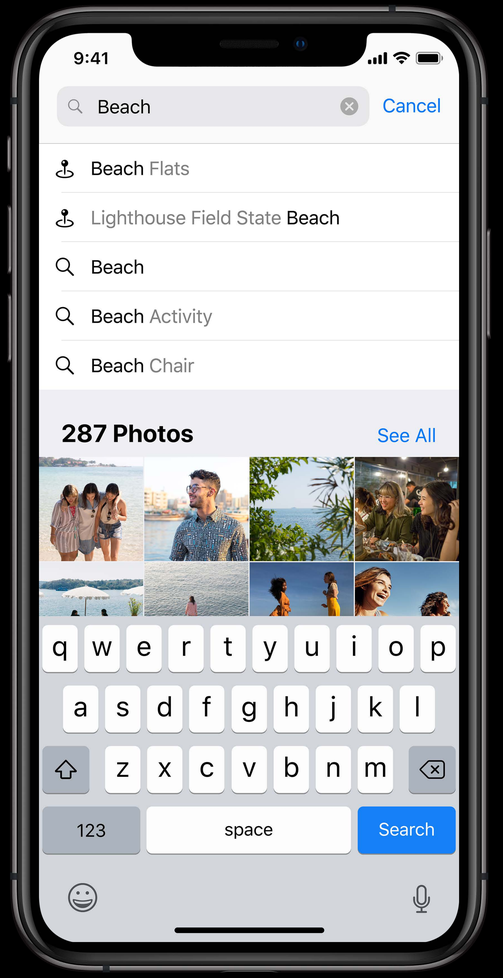
Annotated Data + Compute Power

Difficulties in Large Scale Classification

Annotated Data + Compute Power + Specialized Expertise

Classification Powers Photos Search





Multi-label Versus Mono-label Classification

Multi-label Versus Mono-label Classification

Mono-label model
predicts “cat”

Multi-label model predicts
“cat, adult, book”

Classification basics

Taxonomy

Interpreting observations

Image Similarity

Classification basics

Taxonomy

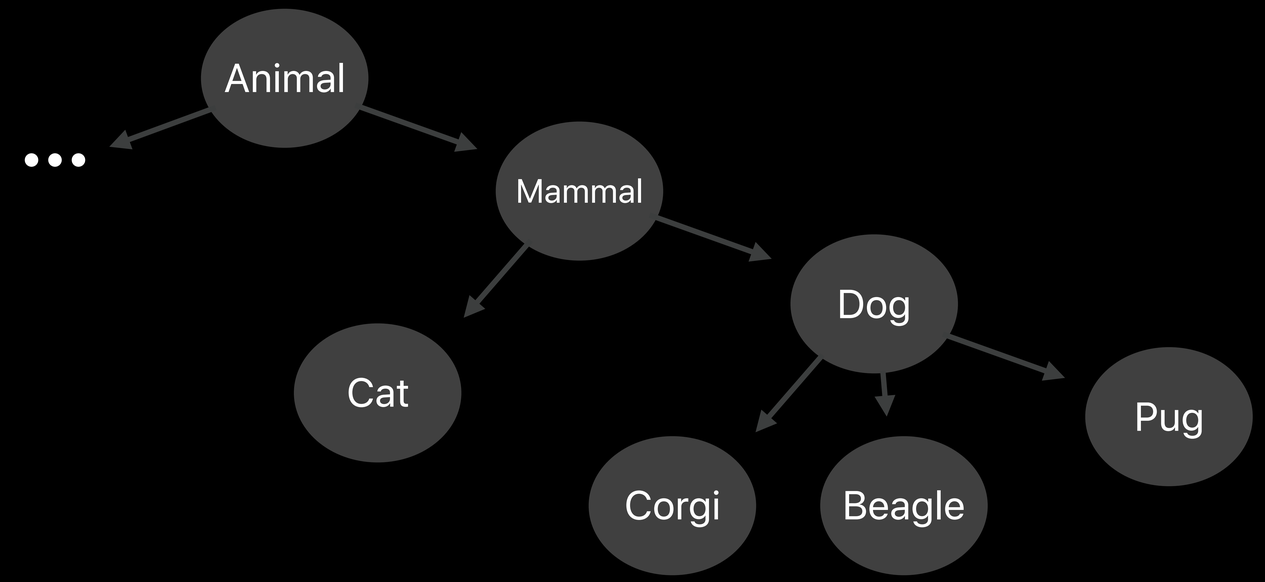
Interpreting observations

Image Similarity

Taxonomy

Taxonomy

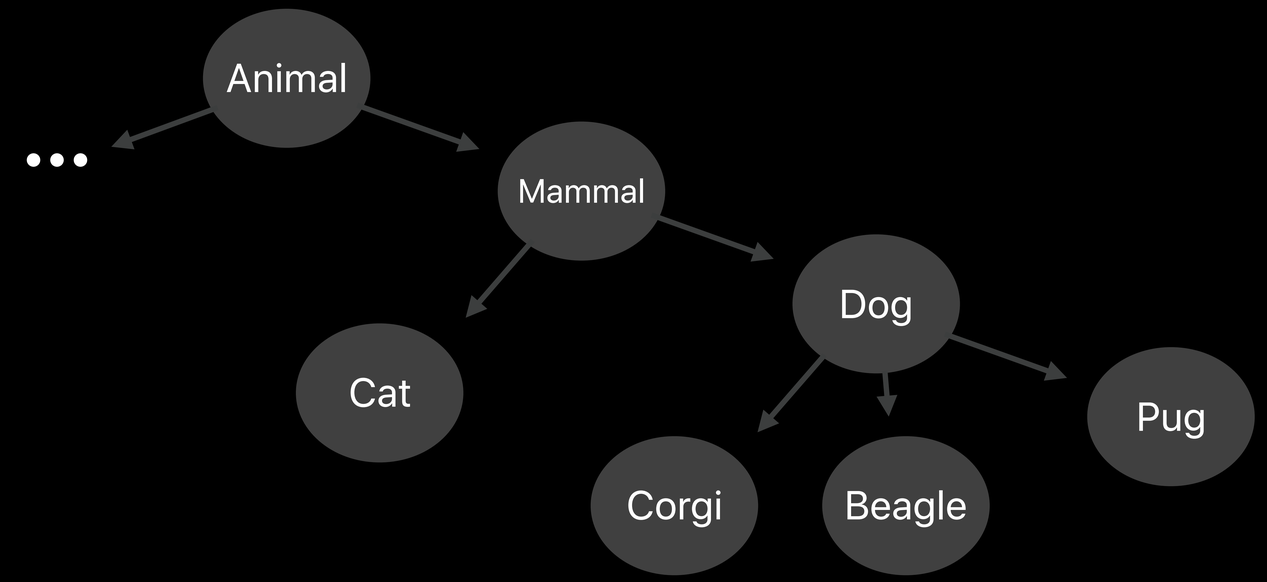
Hierarchical structure,
containing around 1000 classes



Taxonomy

Hierarchical structure,
containing around 1000 classes

Grouping based on shared
semantic meanings

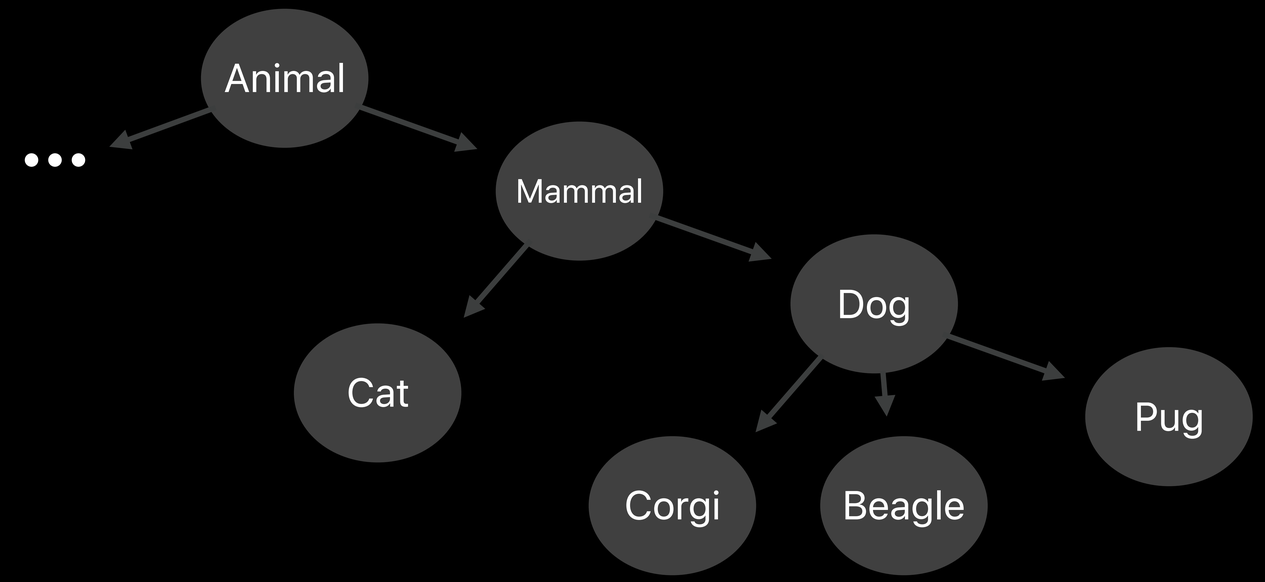


Taxonomy

Hierarchical structure,
containing around 1000 classes

Grouping based on shared
semantic meanings

Defines relationships between
classes of increasing specificity

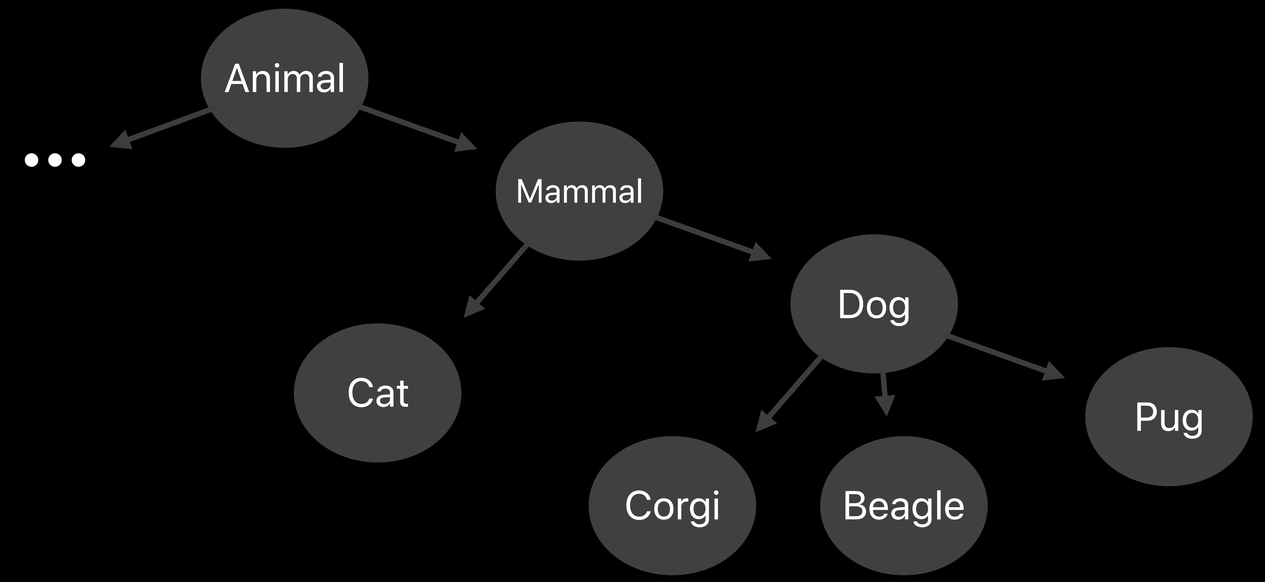


Taxonomy

Hierarchical structure,
containing around 1000 classes

Grouping based on shared
semantic meanings

Defines relationships between
classes of increasing specificity




```
// List full taxonomy with
VNClassifiyImageRequest.knownClassifications(forRevision: VNClassifyImageRequestRevision1 )
```

Taxonomy Construction

Taxonomy Construction

Include classes that are visually identifiable

Taxonomy Construction

Include classes that are visually identifiable

Avoid

- Abstract / controversial concepts

Taxonomy Construction

Include classes that are visually identifiable

Avoid

- Abstract / controversial concepts
- Proper nouns, adjectives, and basic shapes

Taxonomy Construction

Include classes that are visually identifiable

Avoid

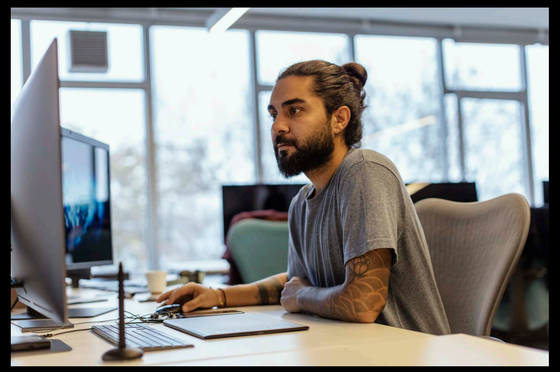
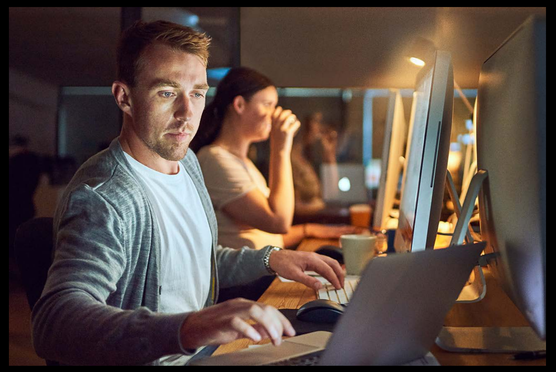
- Abstract / controversial concepts
- Proper nouns, adjectives, and basic shapes
- Occupations

Taxonomy Construction

Include classes that are visually identifiable

Avoid

- Abstract / controversial concepts
- Proper nouns, adjectives, and basic shapes
- Occupations


```
let handler = VNImageRequestHandler(url: imageUrl)
let request = VNClassifyImageRequest()
try? handler.perform([request])
let observations = request.results as? [VNClassificationObservation]
```

```
let handler = VNImageRequestHandler(url: imageUrl)
let request = VNClassifyImageRequest()
try? handler.perform([request])
let observations = request.results as? [VNClassificationObservation]
```

```
let handler = VNImageRequestHandler(url: imageUrl)
let request = VNClassifyImageRequest()
try? handler.perform([request])
let observations = request.results as? [VNClassificationObservation]
```

```
let handler = VNImageRequestHandler(url: imageUrl)
let request = VNClassifyImageRequest()
try? handler.perform([request])
let observations = request.results as? [VNClassificationObservation]
```

Observations output:
(<Label>, <Confidence>)

```
[
(animal, 0.848),
(cat, 0.848),
(mammal, 0.848),
(clothing, 0.676),
(beanie, 0.675),
(hat, 0.675),
(people, 0.616),
(adult, 0.616),
(snow, 0.445),
(jacket, 0.214),
(outdoor, 0.063),
(leash, 0.057),
(cord, 0.057),
…
```

Classification basics

Taxonomy

Interpreting observations

Image Similarity

Classification basics

Taxonomy

Interpreting observations

Image Similarity

Interpreting Observation, Terms

Interpreting Observation, Terms

Confidence > Threshold =>

Predicted motorcycle in image

Interpreting Observation, Terms

Confidence > Threshold =>

Predicted motorcycle in image

Confidence < Threshold =>

Predicted **no** motorcycle in image

Precision and Recall












Precision and Recall

Recall

Percentage of Target Class retrieved from entire library



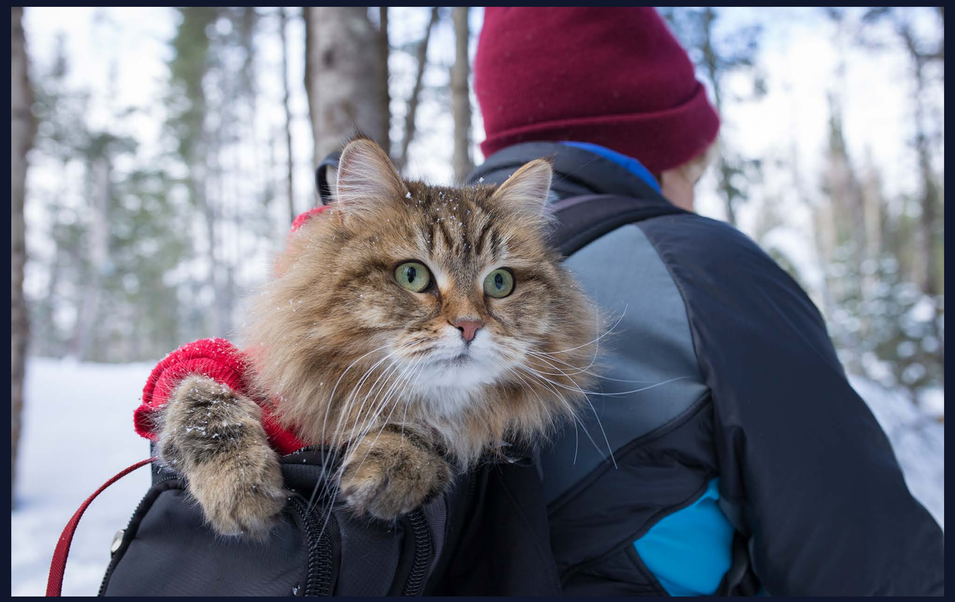








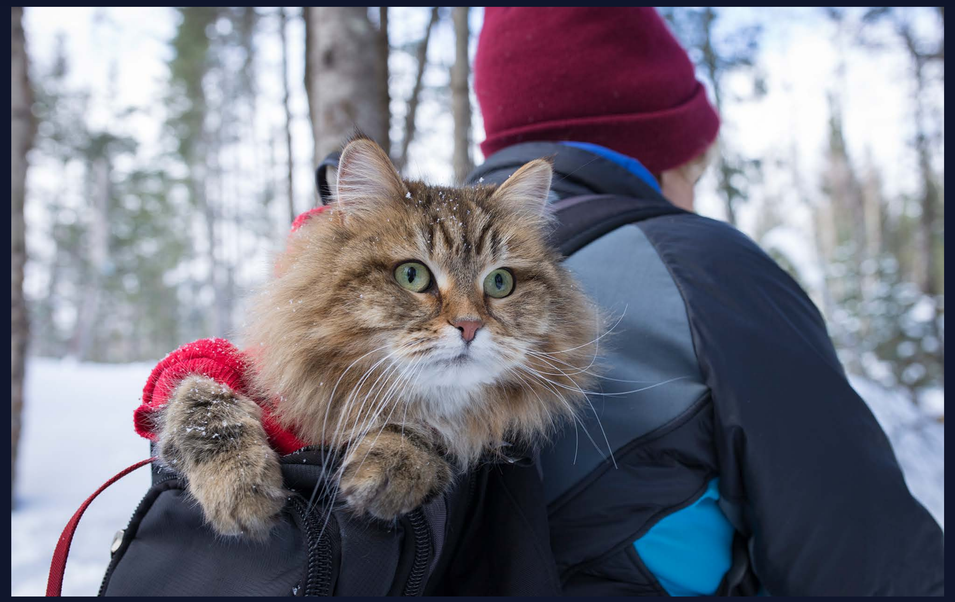

```
let handler = VNImageRequestHandler(url: imageUrl)
let request = VNClassifyImageRequest()
try? handler.perform([request])
let observations = request.results as? [VNClassificationObservation]
let searchObservations = observations?.filter { $0.hasMinimumPrecision(0.0,
forRecall: 0.7) }
```



Observations output:
(<Label>, <Confidence>)

```
[
(animal, 0.848),
(cat, 0.848),
(mammal, 0.848),
(clothing, 0.676),
(beanie, 0.675),
(hat, 0.675),
(people, 0.616),
(adult, 0.616),
(snow, 0.445),
(jacket, 0.214),
(outdoor, 0.063),
(leash, 0.057),
(cord, 0.057),
…
]
```

```
let handler = VNImageRequestHandler(url: imageUrl)
let request = VNClassifyImageRequest()
try? handler.perform([request])
let observations = request.results as? [VNClassificationObservation]
let searchObservations = observations?.filter { $0.hasMinimumPrecision(0.0,
forRecall: 0.7) }
```



```
[
(animal, 0.848),
(cat, 0.848),
(mammal, 0.848),
(clothing, 0.676),
(beanie, 0.675),
(hat, 0.675),
(people, 0.616),
(adult, 0.616),
(snow, 0.445),
(jacket, 0.214),
]
```

Precision and Recall












Precision and Recall

Precision

Percentage of returned results that are target class



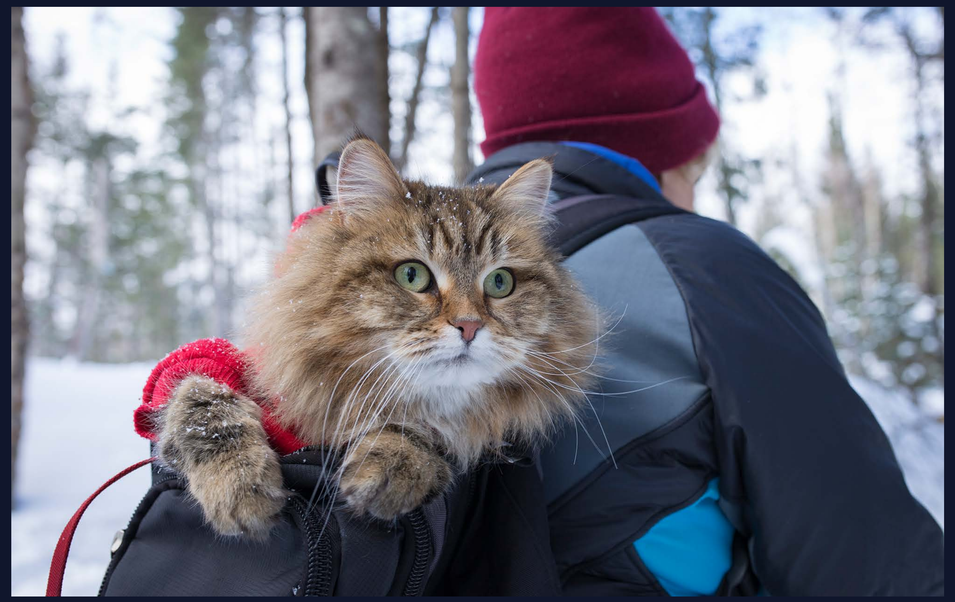








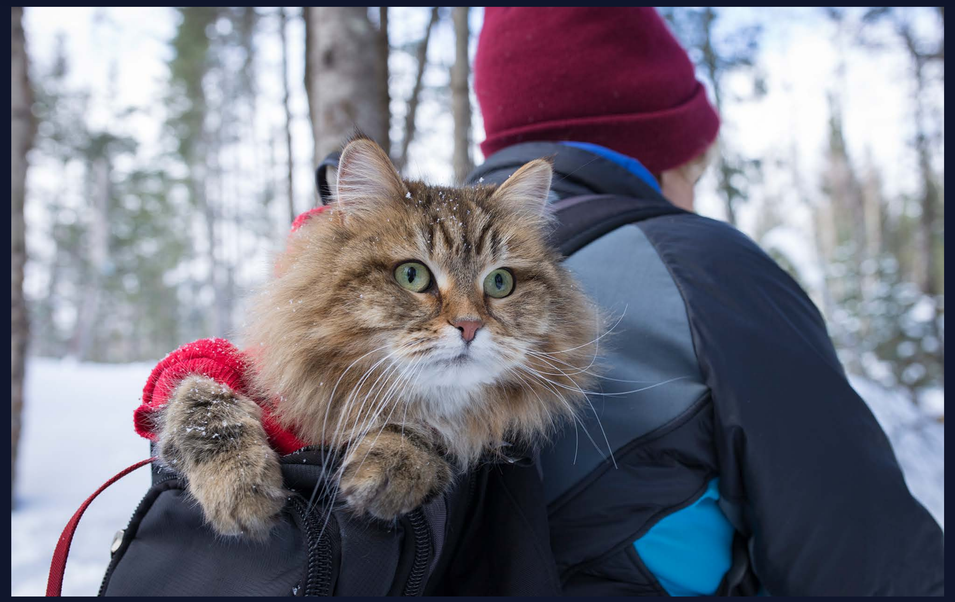

```
let handler = VNImageRequestHandler(url: imageUrl)
let request = VNClassifyImageRequest()
try? handler.perform([request])
let observations = request.results as? [VNClassificationObservation]
let searchObservations = observations?.filter { $0.hasMinimumRecall(0.0,
forPrecision: 0.7) }
```



Observations output:
(<Label>, <Confidence>)

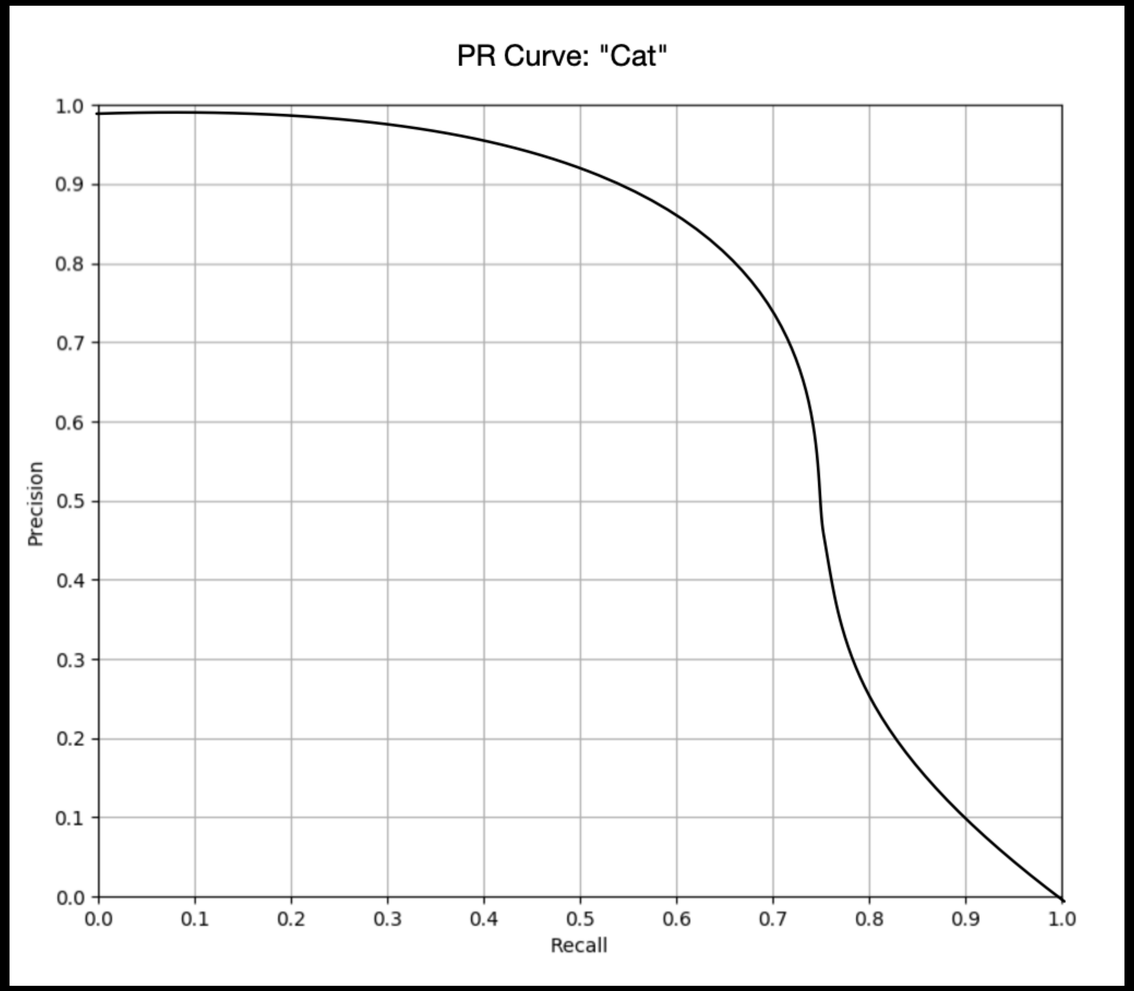
```
[
(animal, 0.848),
(cat, 0.848),
(mammal, 0.848),
(clothing, 0.676),
(beanie, 0.675),
(hat, 0.675),
(people, 0.616),
(adult, 0.616),
(snow, 0.445),
(jacket, 0.214),
(outdoor, 0.063),
(leash, 0.057),
(cord, 0.057),
…
]
```

```swift
let handler = VNImageRequestHandler(url: imageUrl)
let request = VNClassifyImageRequest()
try? handler.perform([request])
let observations = request.results as? [VNClassificationObservation]
let searchObservations = observations?.filter { $0.hasMinimumRecall(0.0,
forPrecision: 0.7) }
```

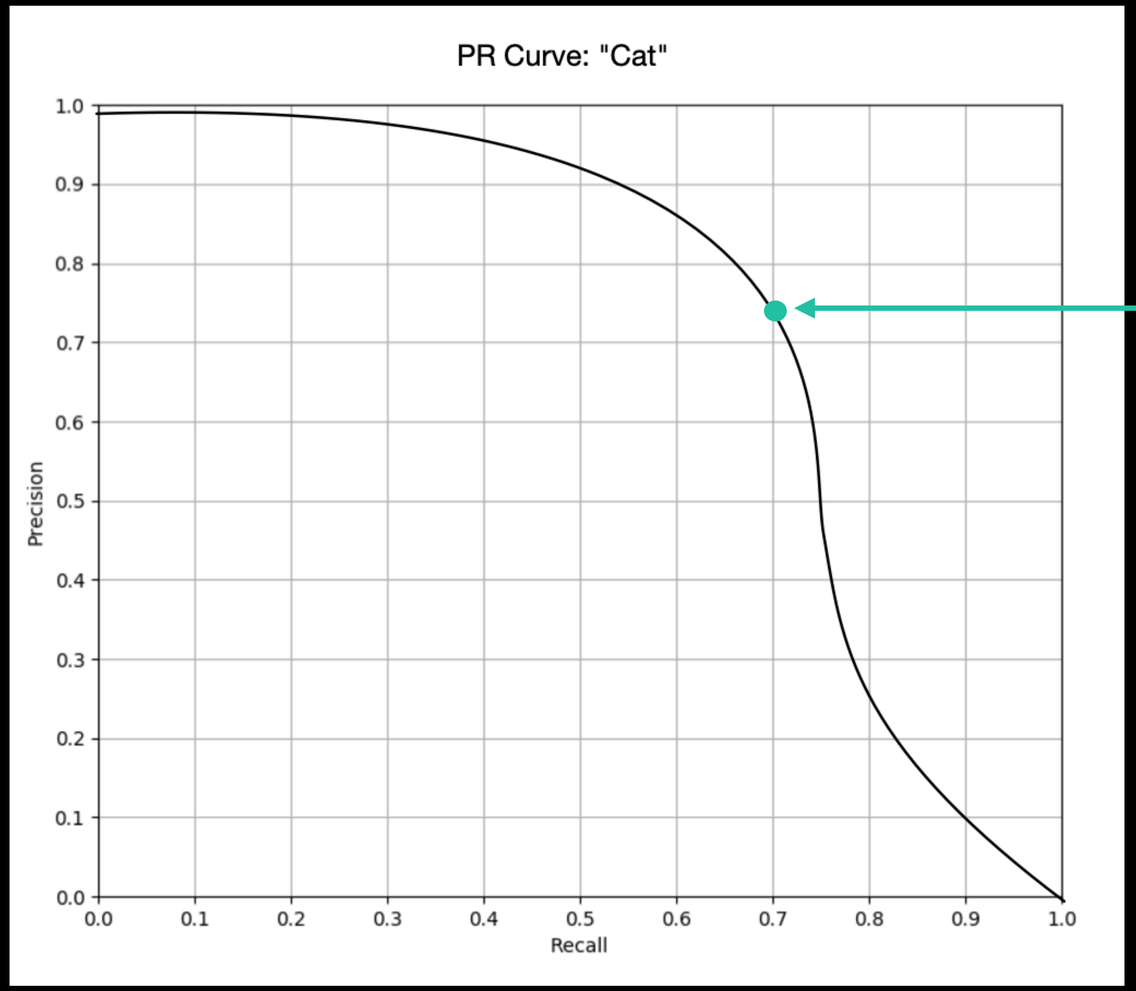


```
[
(animal, 0.848),
(cat, 0.848),
(mammal, 0.848),
(clothing, 0.676),
(beanie, 0.675),
(hat, 0.675),
(people, 0.616),
(adult, 0.616),
(snow, 0.445),
]
```

The PR Curve

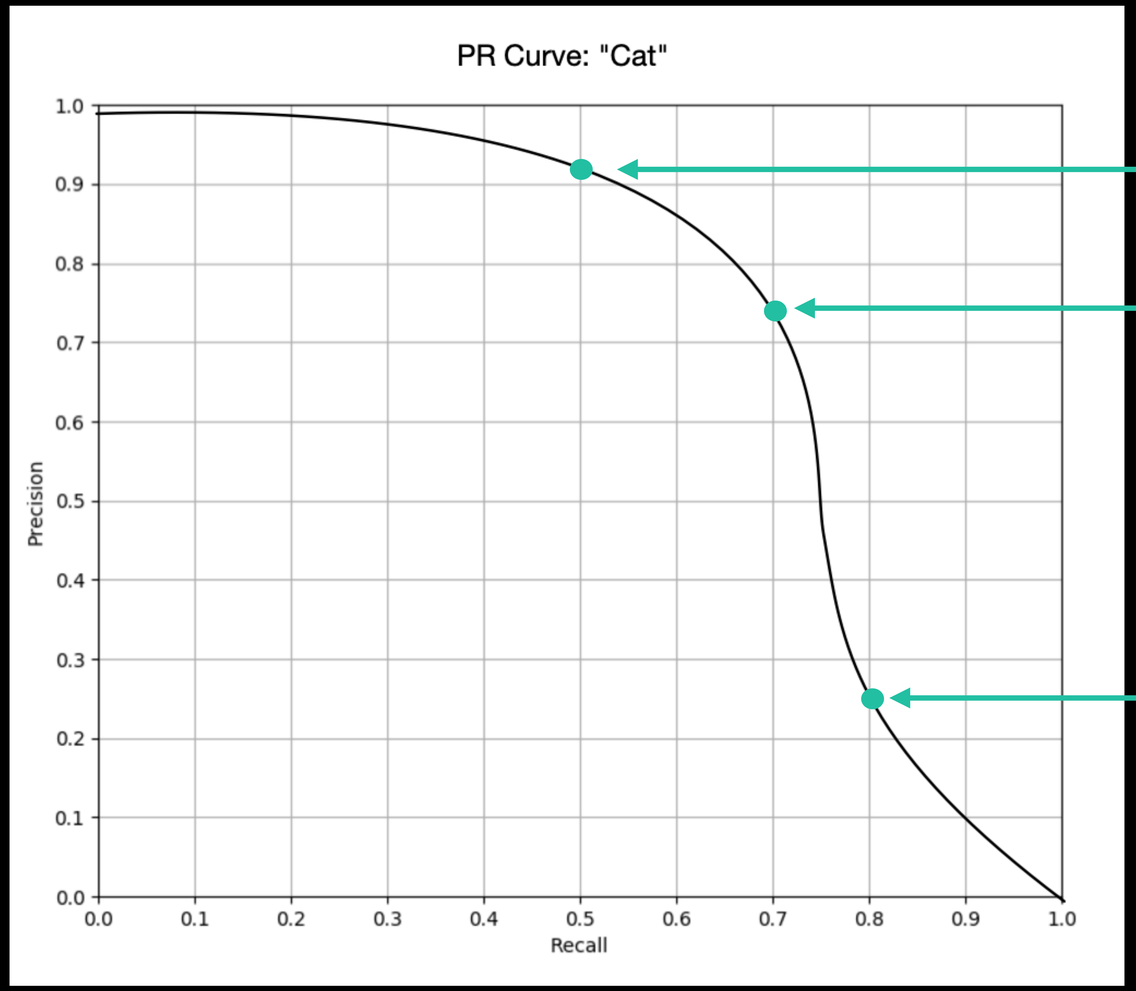

PR Curve: "Cat"
Precision
Recall
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0

The PR Curve



Operating point where
Recall = 0.7
Precision = 0.74

The PR Curve



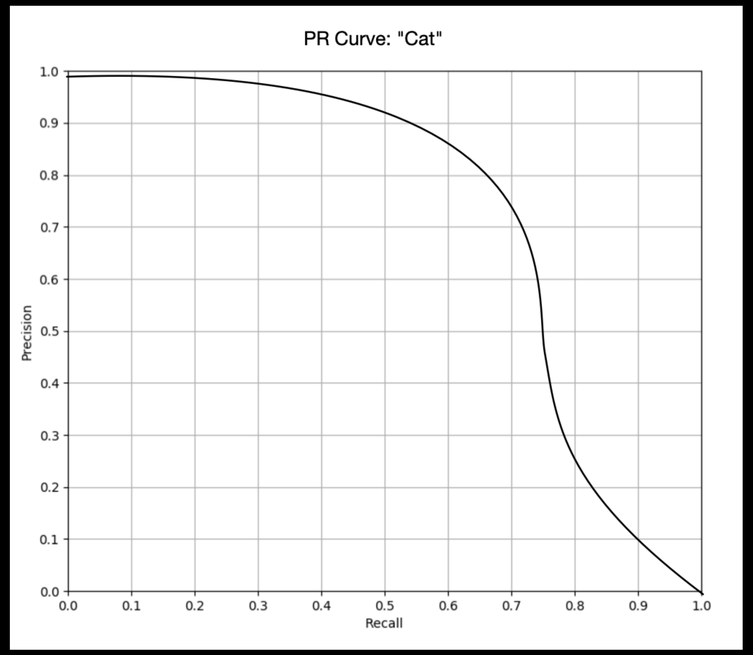
Operating point where
Recall = 0.5
Precision = 0.92

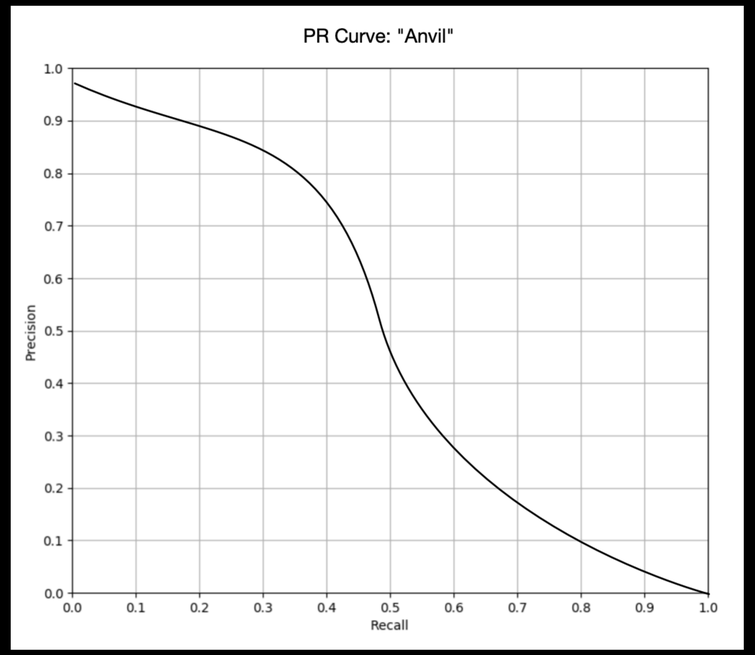
Operating point where
Recall = 0.7
Precision = 0.74

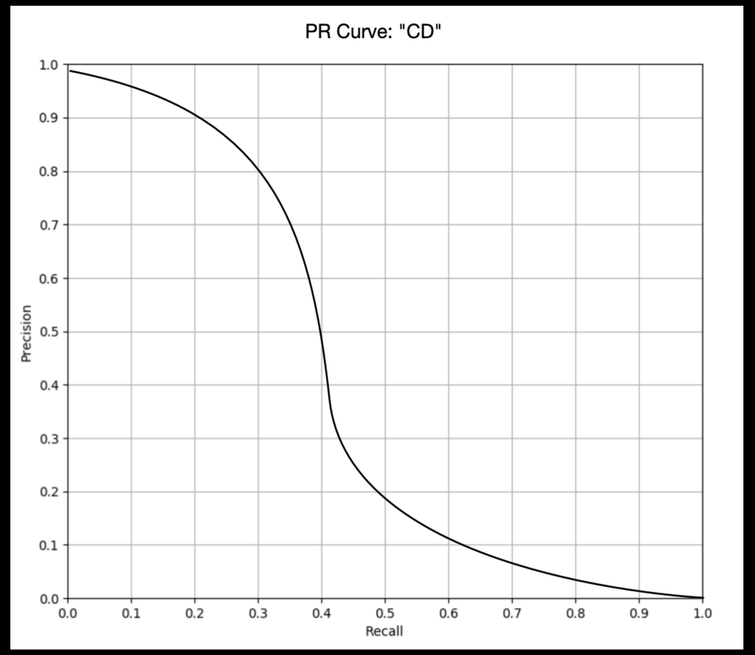
Operating point where
Recall = 0.8
Precision = 0.26

Filtering the PR Curve

```
let observations = request.results as? [VNClassificationObservation]
let searchObservations = observations?.filter { $0.hasMinimumPrecision(0.4, forRecall: 0.5) }
```

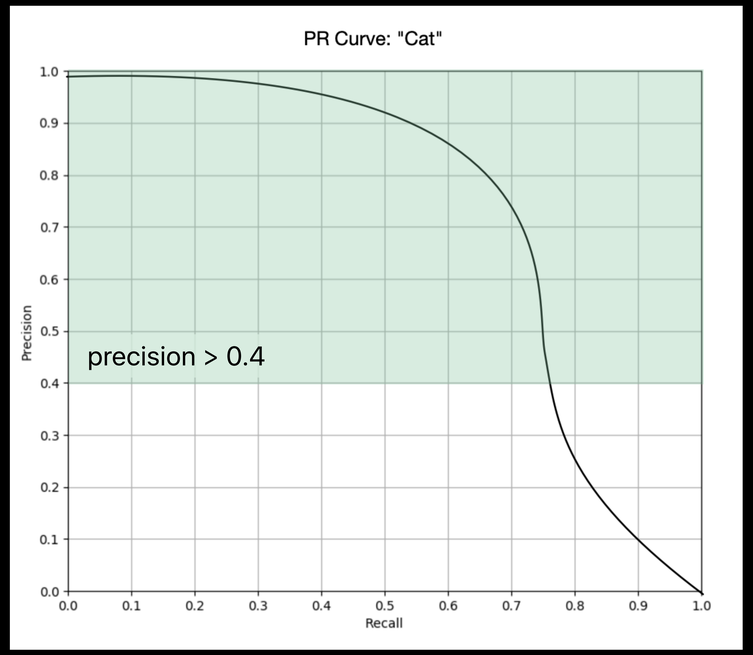


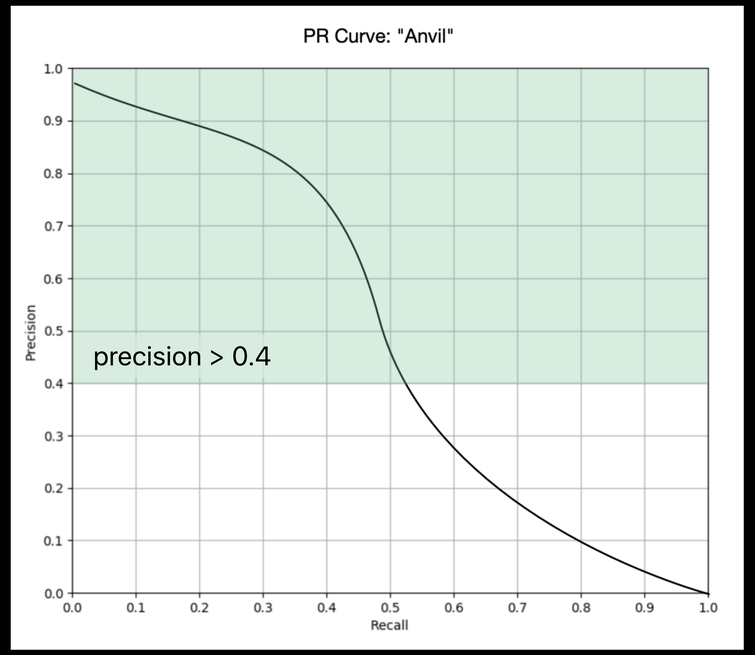


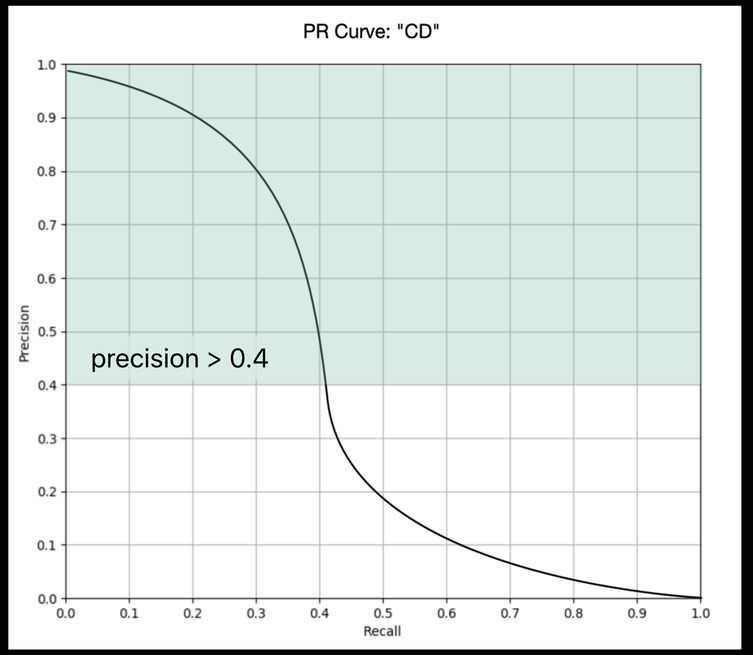


Filtering the PR Curve

```
let observations = request.results as? [VNClassificationObservation]
let searchObservations = observations?.filter { $0.hasMinimumPrecision(0.4, forRecall: 0.5) }
```

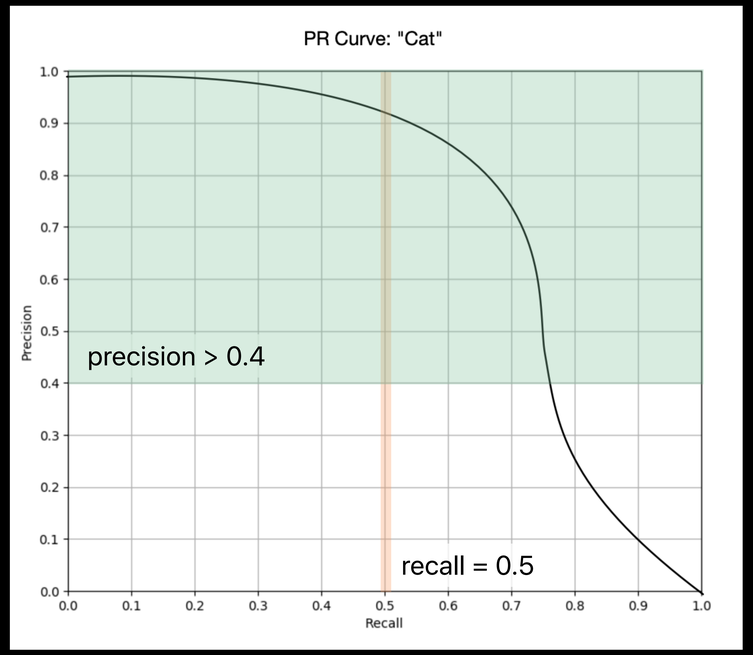


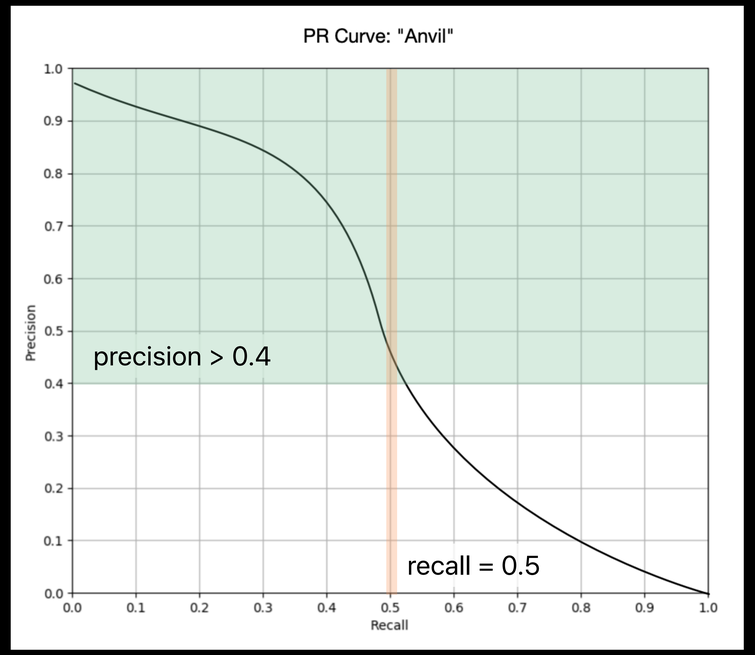


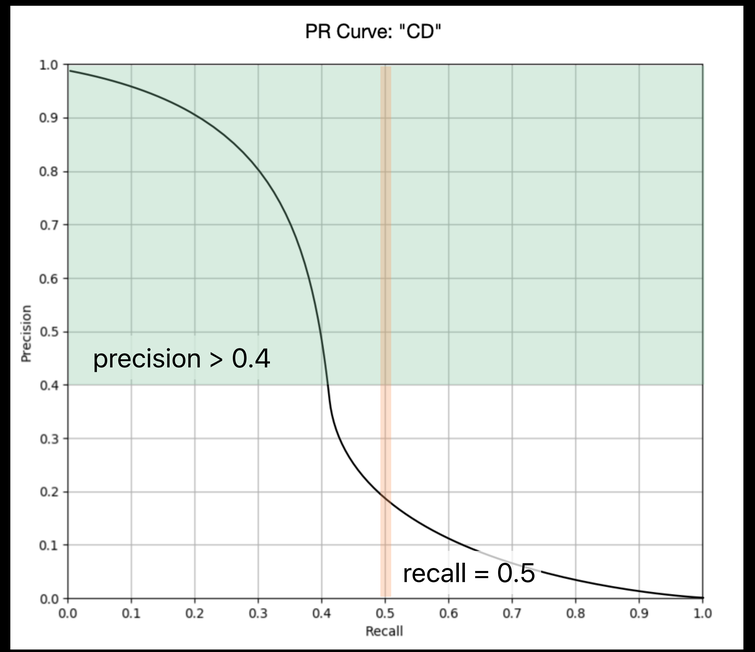


Filtering the PR Curve

```
let observations = request.results as? [VNClassificationObservation]
let searchObservations = observations?.filter { $0.hasMinimumPrecision(0.4, forRecall: 0.5) }
```

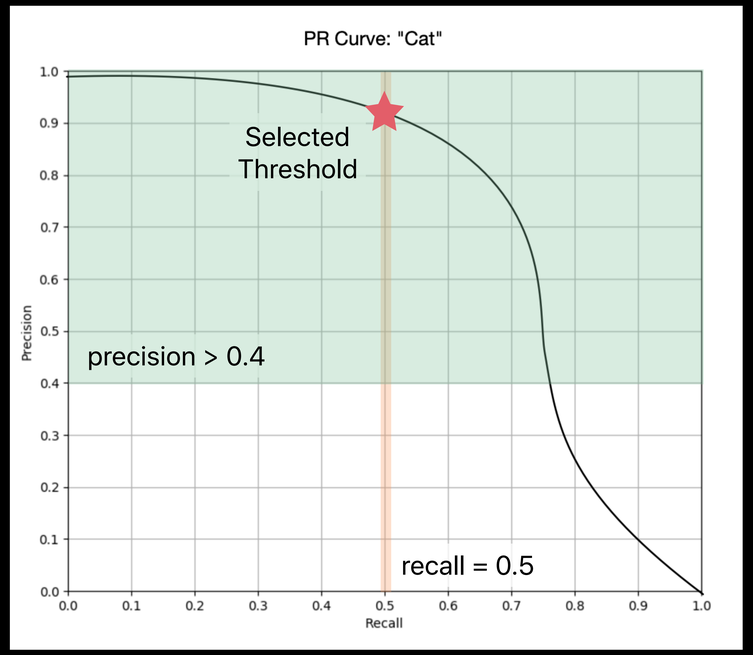


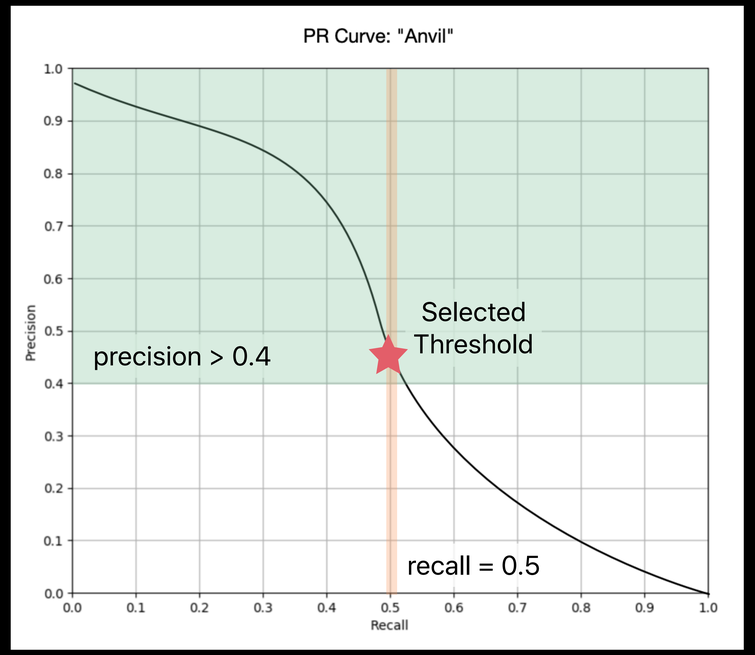


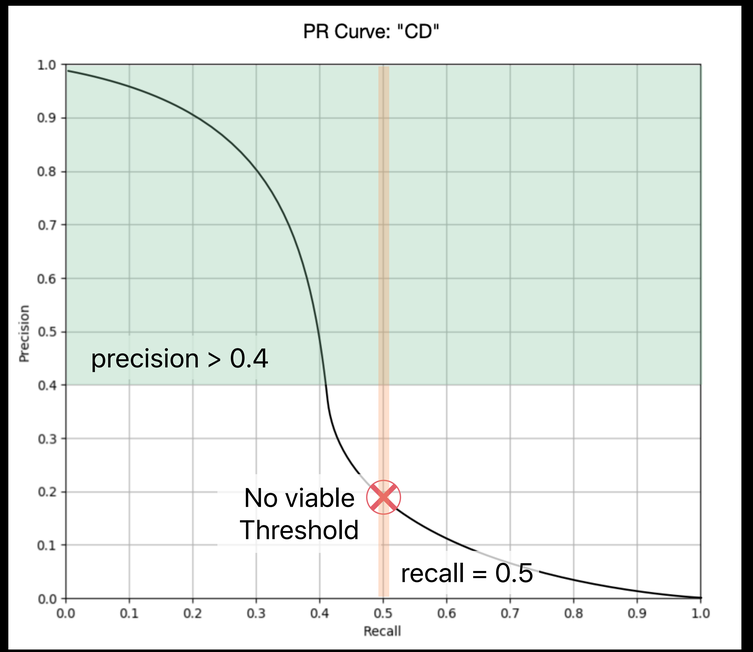


Filtering the PR Curve

```
let observations = request.results as? [VNClassificationObservation]
let searchObservations = observations?.filter { $0.hasMinimumPrecision(0.4, forRecall: 0.5) }
```







Classification: Summary

Returned Observation contains labels and an associated confidence

Choice of threshold is application specific

Can be determined by desired precision and recall

```
[
(wood_natural, 0.90),
(animal, 0.82),
(mammal, 0.82),
(feline, 0.82),
(cat, 0.82),
(snow, 0.51),
(outdoor, 0.37),
(land, 0.37),
…
]
```

Classification basics

Taxonomy

Interpreting observations

Image Similarity

Classification basics

Taxonomy

Interpreting observations

Image Similarity

Image Similarity

How can we describe an image?

Image Similarity

Describe an image using pixels?

Input Image

Image Similarity

Describe an image using pixels?

Input Image

Fails to retrieve

Image Similarity

Describe an image using words?

Input Image

Image Similarity

Describe an image using words?

Input Image

Semantically Similar
Word search

Kitten,
Bowl

Image Similarity

Describe an image using words?

Input Image

Semantically Similar
Word search

Kitten,
Bowl

Image Similarity

Image Similarity

Descriptor should describes image content, not just appearance

Image Similarity

Descriptor should describes image content, not just appearance

Classification network creates representations of images

Image Similarity

Descriptor should describes image content, not just appearance

Classification network creates representations of images

FeaturePrint — vector image descriptor similar to a word vector

Image Similarity

Descriptor should describes image content, not just appearance

Classification network creates representations of images

FeaturePrint — vector image descriptor similar to a word vector

Image Similarity

Image Similarity

Image Similarity

Image Similarity

Image Similarity

Image Similarity

Image Similarity

Demo

Image Similarity game

```
// Generate featureprints for copies and compute distances from original featureprint
for index in contestantImageURLs.indices {
    let contestantImageURL = contestantImageURLs[index]
    if let contestantFPO = featureprintObservationForImage(atURL: contestantImageURL) {
    do {
        var distance = Float(0)
        try contestantFPO.computeDistance(&distance, to: originalFPO)
            ranking.append((contestantIndex: index, featureprintDistance: distance))
        } catch {
            print("Error computing distance between featureprints.")
        }
   }
}
```

```
// Generate featureprints for copies and compute distances from original featureprint
for index in contestantImageURLs.indices {
    let contestantImageURL = contestantImageURLs[index]
    if let contestantFPO = featureprintObservationForImage(atURL: contestantImageURL) {
    do {
        var distance = Float(0)
        try contestantFPO.computeDistance(&distance, to: originalFPO)
            ranking.append((contestantIndex: index, featureprintDistance: distance))
        } catch {
            print("Error computing distance between featureprints.")
        }
   }
}
```

```
// Generate featureprints for copies and compute distances from original featureprint
for index in contestantImageURLs.indices {
    let contestantImageURL = contestantImageURLs[index]
    if let contestantFPO = featureprintObservationForImage(atURL: contestantImageURL) {
    do {
        var distance = Float(0)
        try contestantFPO.computeDistance(&distance, to: originalFPO)
            ranking.append((contestantIndex: index, featureprintDistance: distance))
        } catch {
            print("Error computing distance between featureprints.")
        }
   }
}
```

Face Technology, New Detectors, Tracking, and CoreML

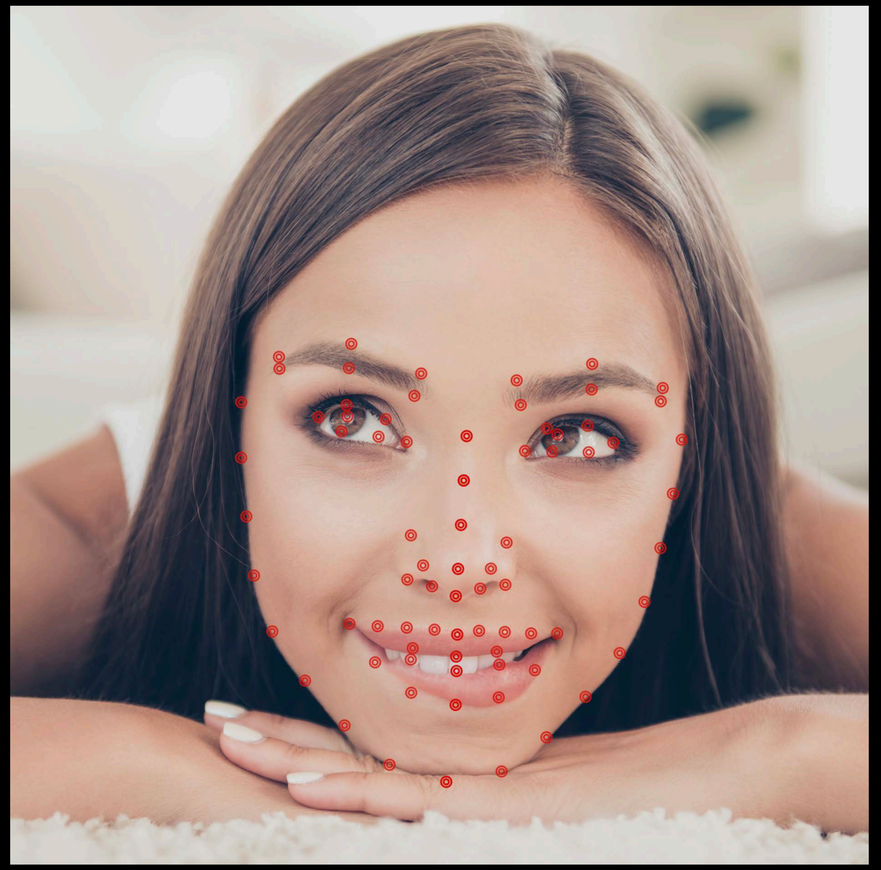
Sergey Kamensky, Vision Team

Face Technology

Face Landmarks

Face Landmarks

New



Face Landmarks

New

Face Landmarks

New

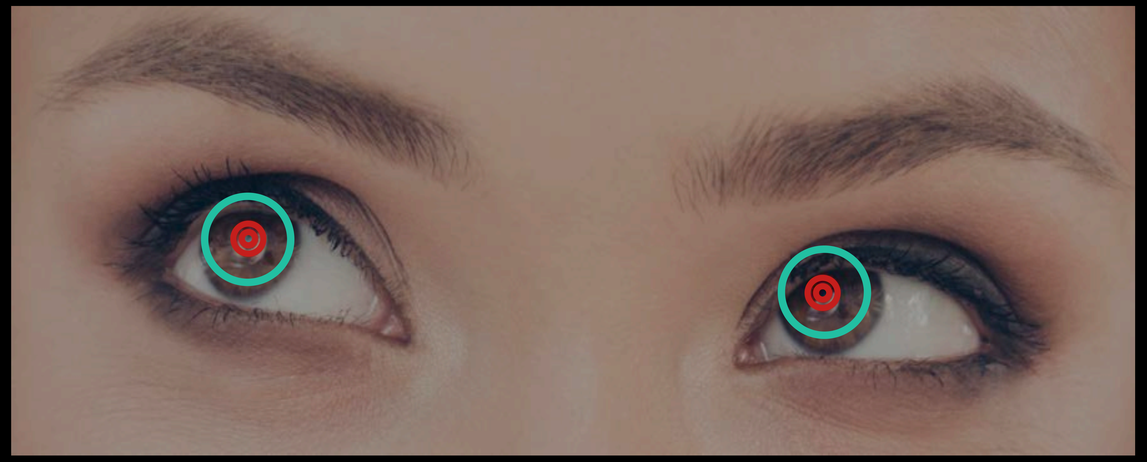
Single Confidence Score

Confidence Score per point

Face Landmarks

Pupils Detection

New



Face Landmarks

Example

Face Landmarks

Example

```
let request = VNDetectFaceLandmarksRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let faceObservation = request.results!.first! as! VNFaceObservation
let landmarks2D = faceObservation.landmarks!
```

Face Landmarks

Example

```
let request = VNDetectFaceLandmarksRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let faceObservation = request.results!.first! as! VNFaceObservation
let landmarks2D = faceObservation.landmarks!
```

Face Landmarks

Example

```
let request = VNDetectFaceLandmarksRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let faceObservation = request.results!.first! as! VNFaceObservation
let landmarks2D = faceObservation.landmarks!
```

Face Landmarks

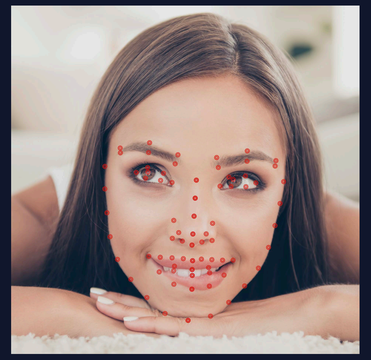
Example

```
let request = VNDetectFaceLandmarksRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let faceObservation = request.results!.first! as! VNFaceObservation
let landmarks2D = faceObservation.landmarks!
```

Face Landmarks

Example

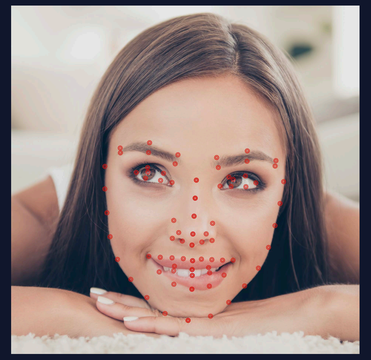
```
let request = VNDetectFaceLandmarksRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let faceObservation = request.results!.first! as! VNFaceObservation
let landmarks2D = faceObservation.landmarks!
```

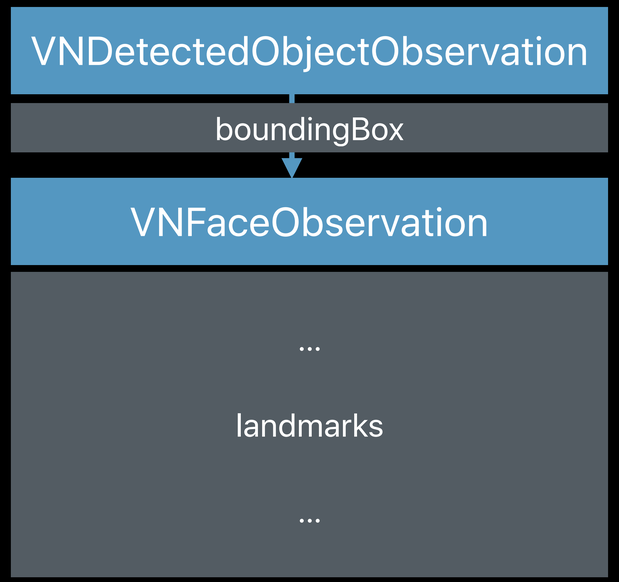


Face Landmarks

Example

```
let request = VNDetectFaceLandmarksRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let faceObservation = request.results!.first! as! VNFaceObservation
let landmarks2D = faceObservation.landmarks!
```

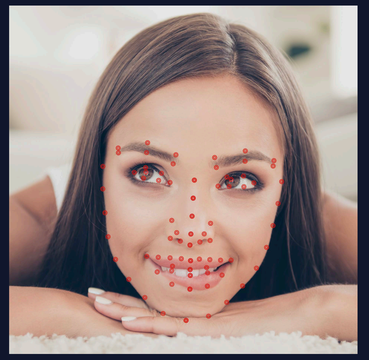


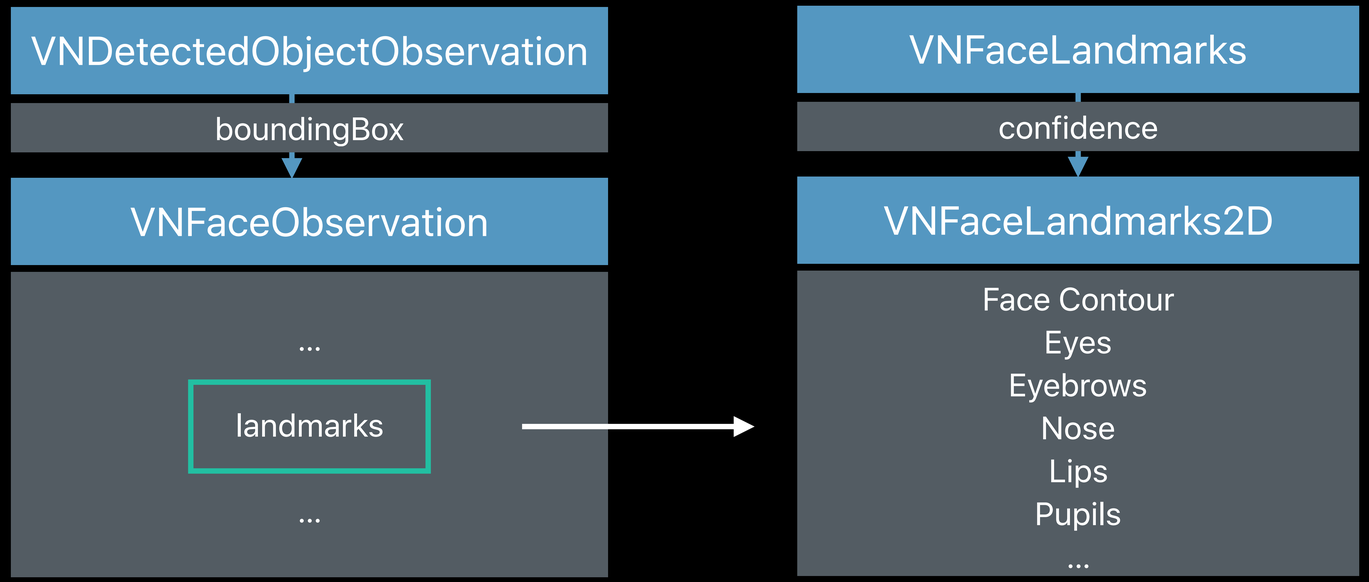


Face Landmarks

Example

```
let request = VNDetectFaceLandmarksRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let faceObservation = request.results!.first! as! VNFaceObservation
let landmarks2D = faceObservation.landmarks!
```

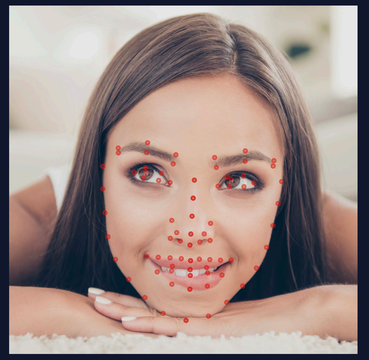


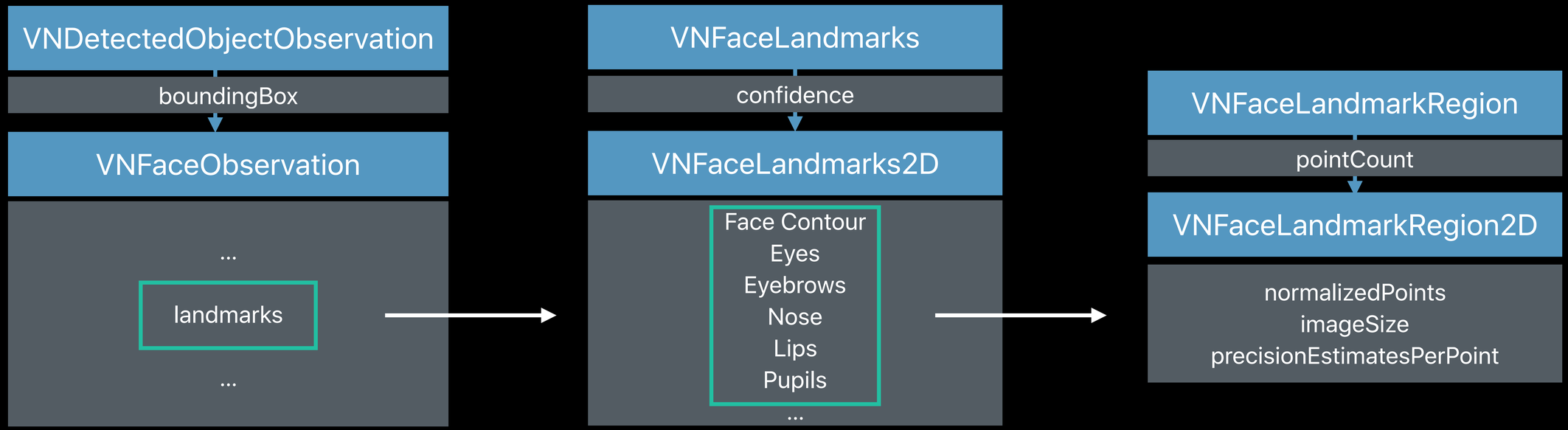


Face Landmarks

Example

```
let request = VNDetectFaceLandmarksRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let faceObservation = request.results!.first! as! VNFaceObservation
let landmarks2D = faceObservation.landmarks!
```

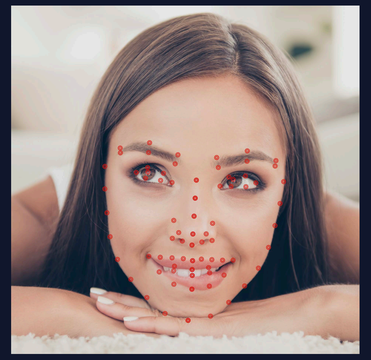


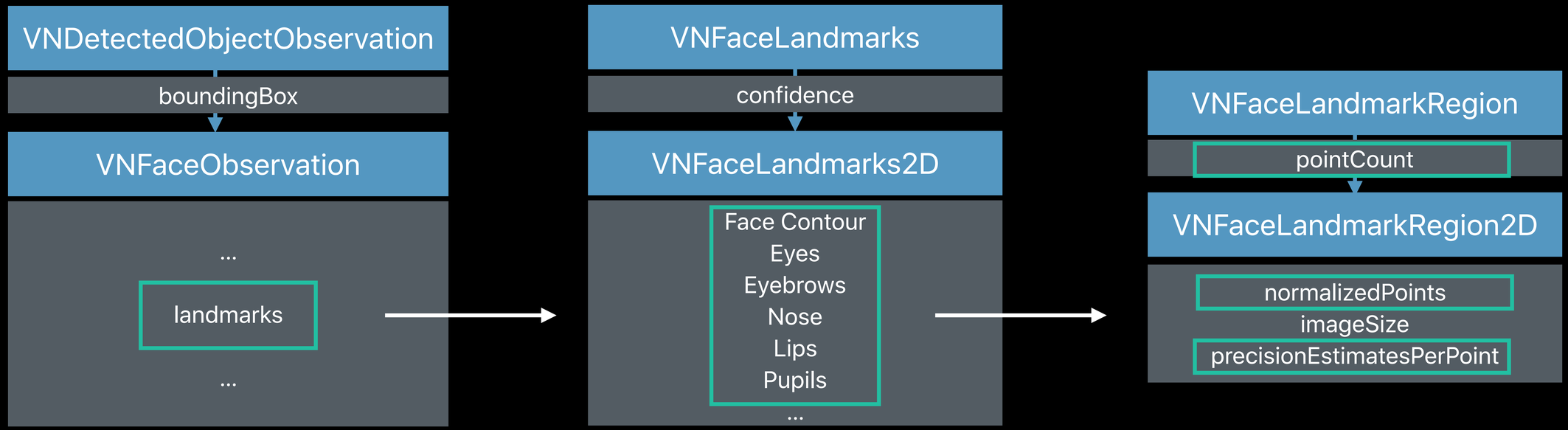


Face Landmarks

Example

```
let request = VNDetectFaceLandmarksRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let faceObservation = request.results!.first! as! VNFaceObservation
let landmarks2D = faceObservation.landmarks!
```





Face Landmarks

Revisioning example — default versus explicit

```
let request = VNDetectFaceLandmarksRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let faceObservation = request.results!.first! as! VNFaceObservation
let lm2D = faceObservation.landmarks!
```

Face Landmarks

Revisioning example — default versus explicit

```
let request = VNDetectFaceLandmarksRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let faceObservation = request.results!.first! as! VNFaceObservation
let lm2D = faceObservation.landmarks!
```

Face Landmarks

Revisioning example — default versus explicit

```
let request = VNDetectFaceLandmarksRequest()
request.revision = VNDetectFaceLandmarksRequestRevision2
// request.constellation = VNRequestFaceLandmarksConstellation65Points
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let faceObservation = request.results!.first! as! VNFaceObservation
let lm2D = faceObservation.landmarks!
```

2018 SDKs

Face Landmarks

Revisioning example — default versus explicit

```
let request = VNDetectFaceLandmarksRequest()
request.revision = VNDetectFaceLandmarksRequestRevision3
request.constellation = VNRequestFaceLandmarksConstellation76Points
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let faceObservation = request.results!.first! as! VNFaceObservation
let lm2D = faceObservation.landmarks!
```

2019 SDKs

Face Landmarks

Revisioning example — default versus explicit

```
let request = VNDetectFaceLandmarksRequest()
request.revision = VNDetectFaceLandmarksRequestRevision3
request.constellation = VNRequestFaceLandmarksConstellation76Points
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let faceObservation = request.results!.first! as! VNFaceObservation
let lm2D = faceObservation.landmarks!
```

2019 SDKs

Face Capture Quality

Face Capture Quality

Face Capture Quality is a holistic measure that considers:
lighting, blur, occlusion, expression, pose, etc.

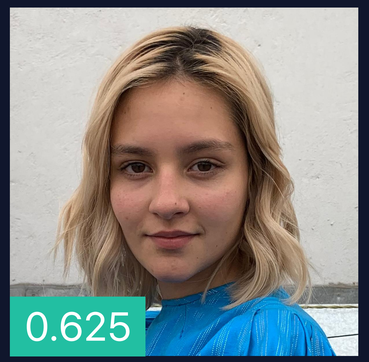
Face Capture Quality

Face Capture Quality is a holistic measure that considers:
lighting, blur, occlusion, expression, pose, etc.

Face Capture Quality

Example

```
let request = VNDetectFaceCaptureQualityRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let faceObservation = request.results!.first! as! VNFaceObservation
let faceCaptureQuality = faceObservation.faceCaptureQuality
```



Face Capture Quality

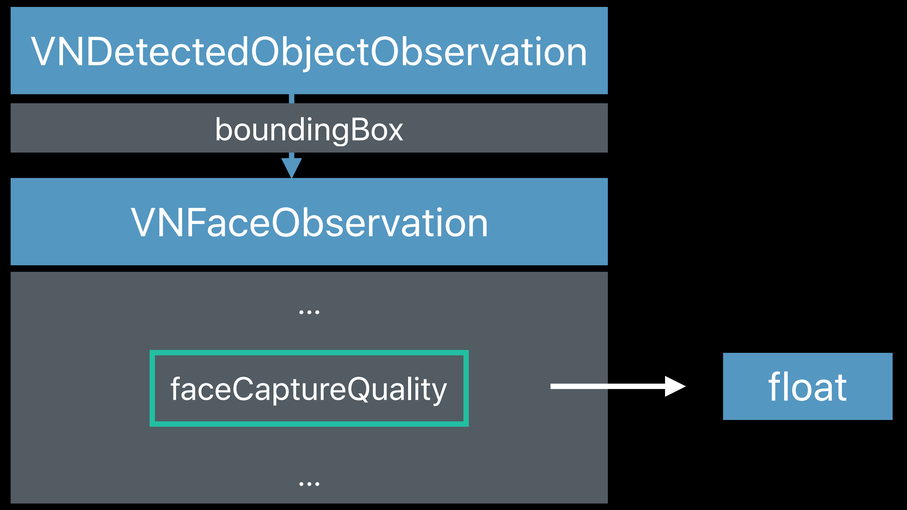
Example

```
let request = VNDetectFaceCaptureQualityRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let faceObservation = request.results!.first! as! VNFaceObservation
let faceCaptureQuality = faceObservation.faceCaptureQuality
```

Face Capture Quality

Example

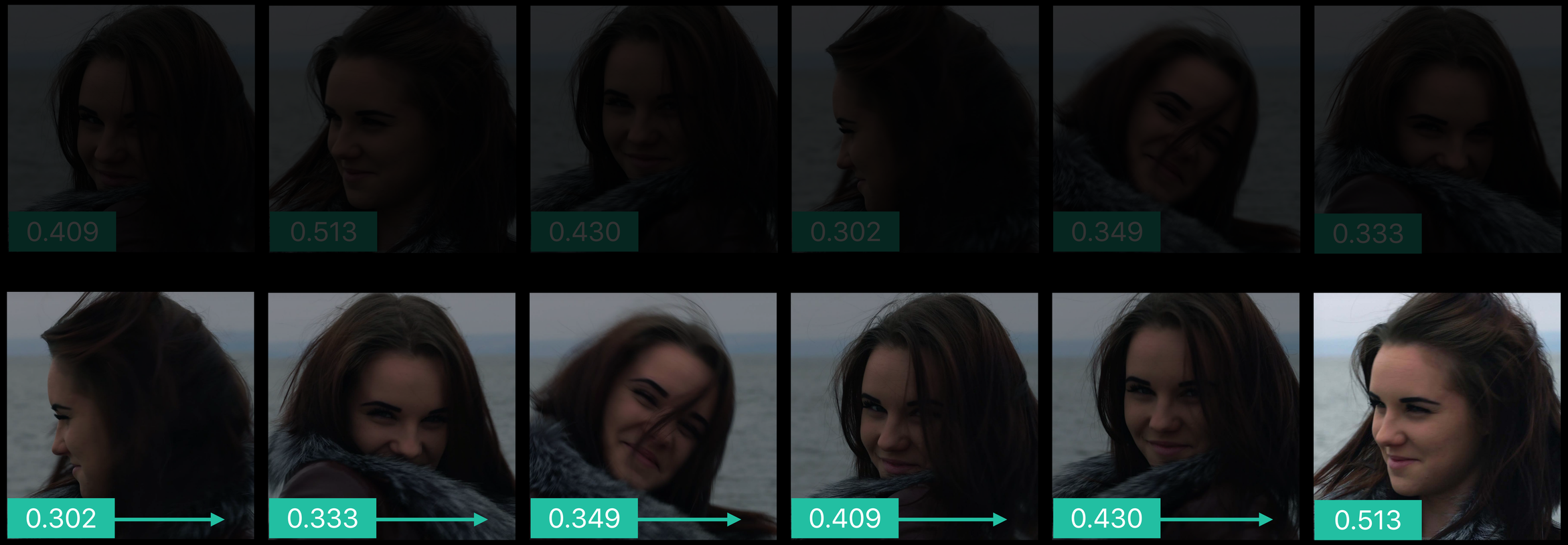
```
let request = VNDetectFaceCaptureQualityRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let faceObservation = request.results!.first! as! VNFaceObservation
let faceCaptureQuality = faceObservation.faceCaptureQuality
```



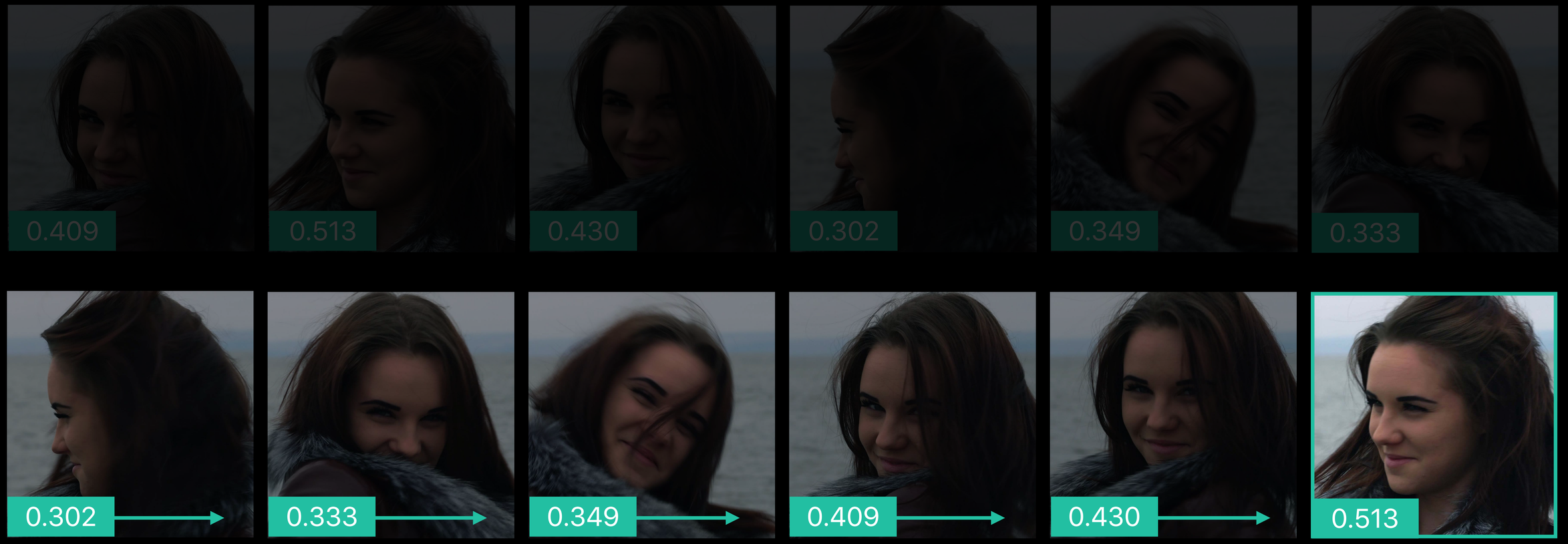
Face Capture Quality

Face Capture Quality

Face Capture Quality

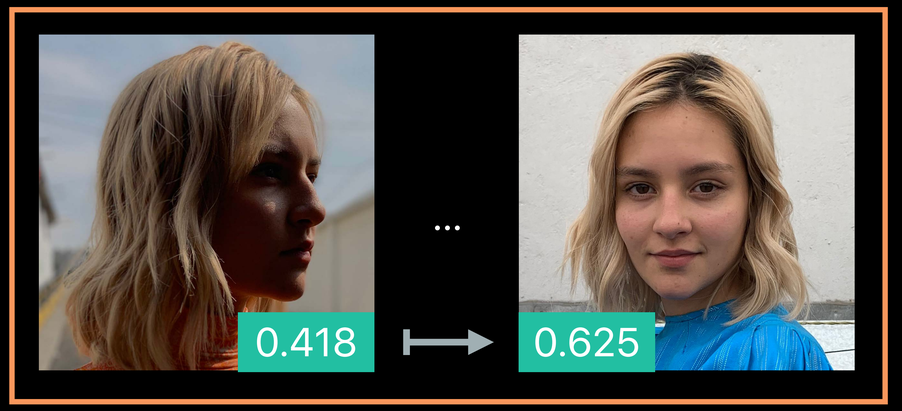


Face Capture Quality

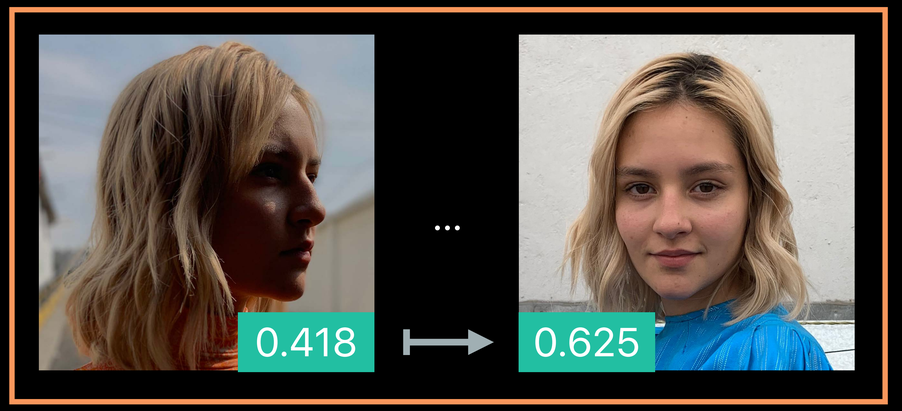


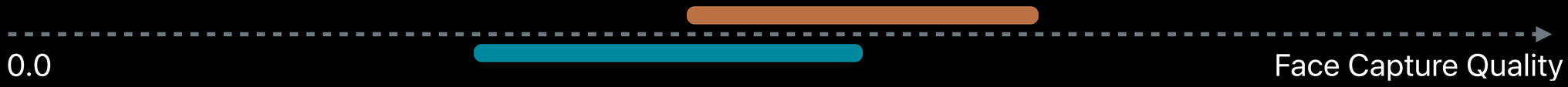
Face Capture Quality

Face Capture Quality



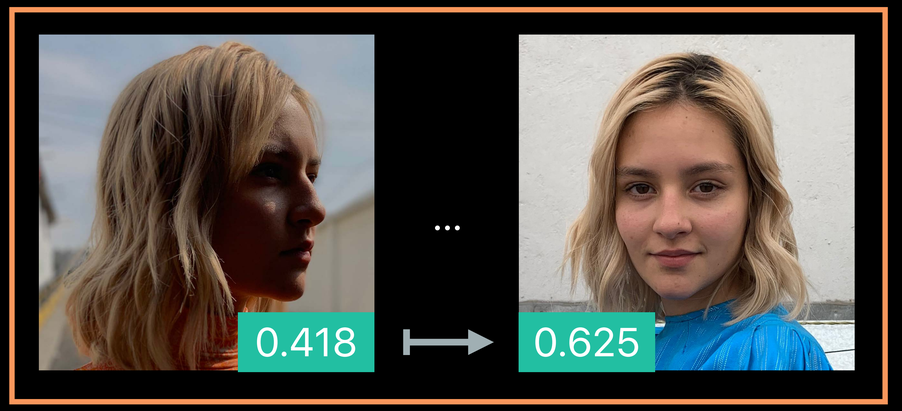
Face Capture Quality

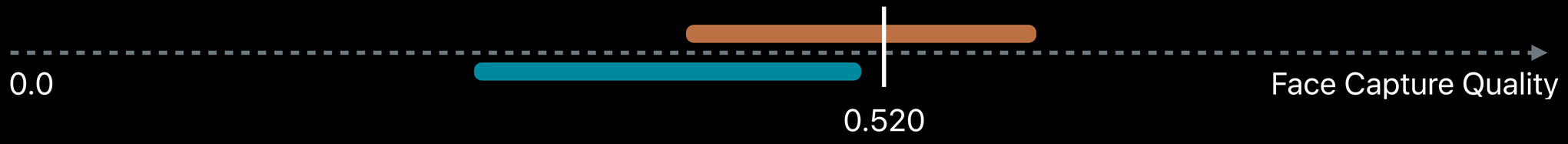




Face Capture Quality should not be compared against a threshold

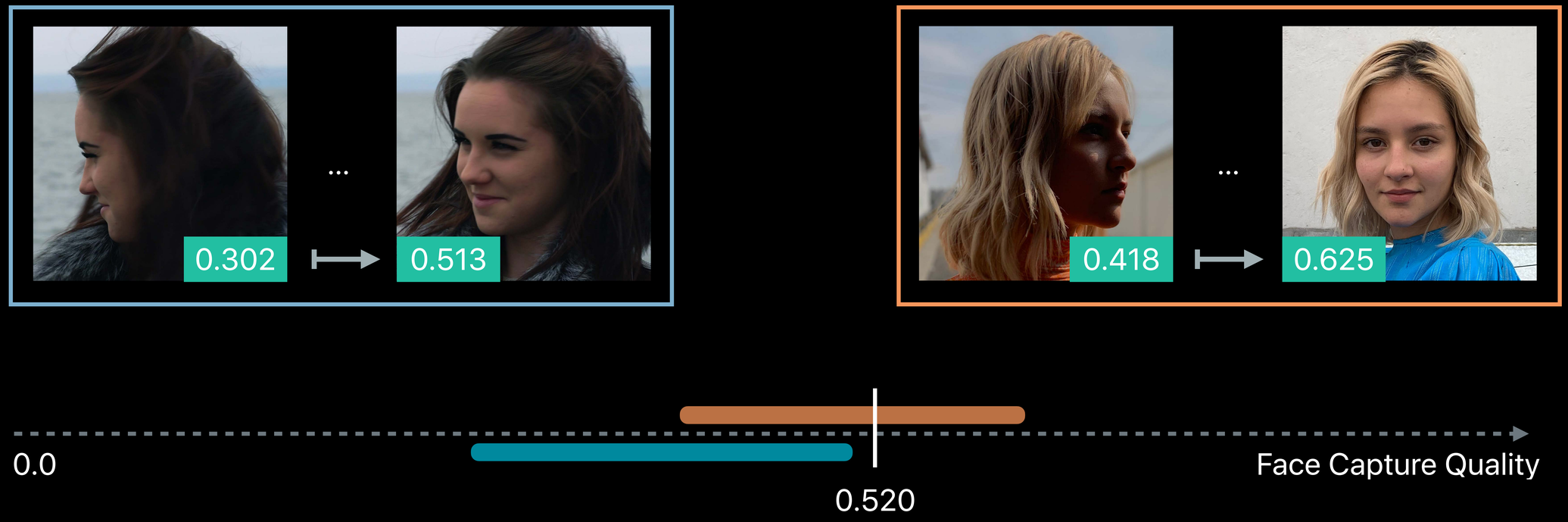
Face Capture Quality





Face Capture Quality should not be compared against a threshold

Face Capture Quality

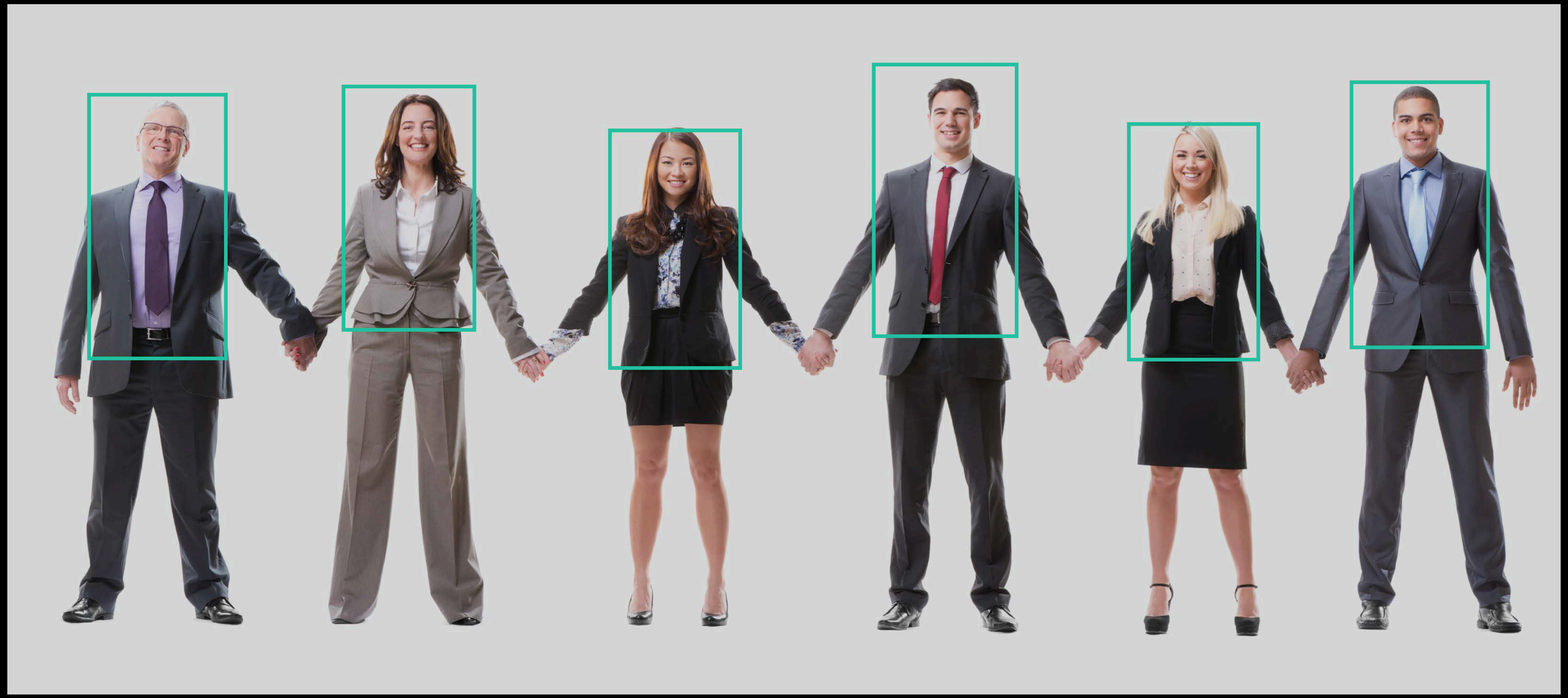


Face Capture Quality should not be compared against a threshold

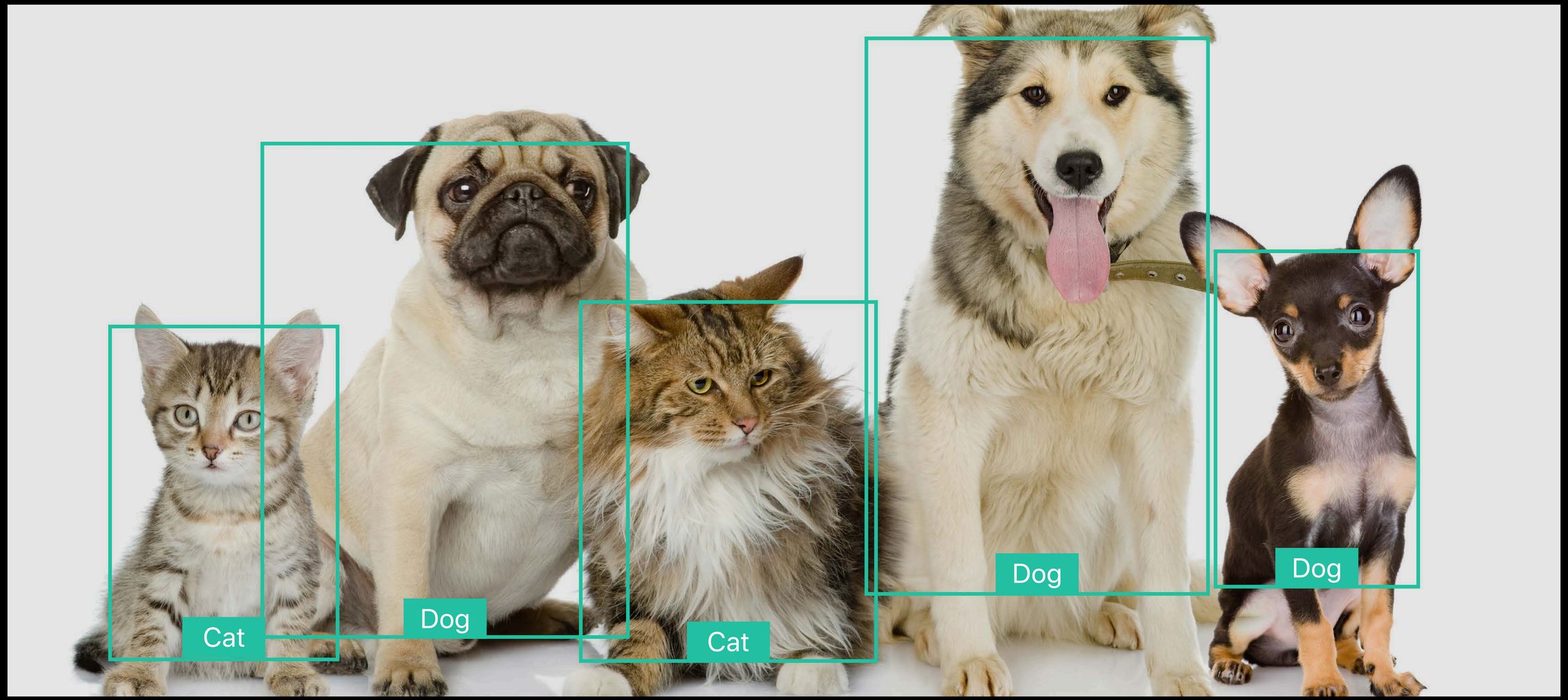
Face Capture Quality is a comparative measure of the same subject

New Detectors

New Detectors - Human Detector



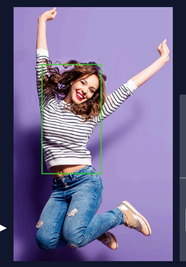
New Detectors - Cat and Dog Detectors



New Detectors

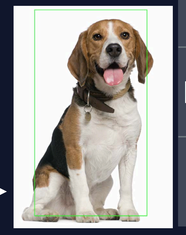
Examples

```
// Human Detector
let request = VNDetectHumanRectanglesRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let human = request.results!.first! as! VNDetectedObjectObservation
```



uuid	1234
boundingBox	[0.2, 0.3, 0.4, 0.8]

```
// Animal Detector
let request = VNDetectAnimalRectanglesRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let dog = request.results!.first! as! VNRecognizedObjectObservation
```

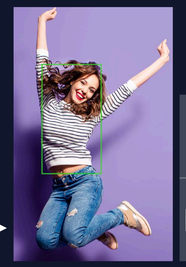


uuid	5678
boundingBox	[0.2, 0.4, 0.5, 0.8]
labels	{“dog”}

New Detectors

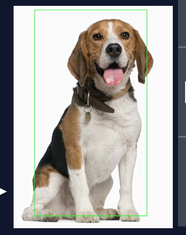
Examples

```
// Human Detector
let request = VNDetectHumanRectanglesRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let human = request.results!.first! as! VNDetectedObjectObservation
```



uuid	1234
boundingBox	[0.2, 0.3, 0.4, 0.8]

```
// Animal Detector
let request = VNDetectAnimalRectanglesRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let dog = request.results!.first! as! VNRecognizedObjectObservation
```

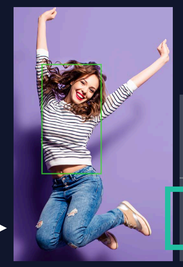


uuid	5678
boundingBox	[0.2, 0.4, 0.5, 0.8]
labels	{“dog”}

New Detectors

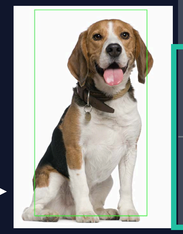
Examples

```
// Human Detector
let request = VNDetectHumanRectanglesRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let human = request.results!.first! as! VNDetectedObjectObservation
```



uuid	1234
boundingBox	[0.2, 0.3, 0.4, 0.8]

```
// Animal Detector
let request = VNDetectAnimalRectanglesRequest()
let requestHandler = VNImageRequestHandler(url: imageURL)
try requestHandler.perform([request])
let dog = request.results!.first! as! VNRecognizedObjectObservation
```



uuid	5678
boundingBox	[0.2, 0.4, 0.5, 0.8]
labels	{“dog”}

Tracking Enhancements

Tracking with Vision

Tracking with Vision

Less expansion into the background

Tracking with Vision

Less expansion into the background

Better handling of occlusions

Tracking with Vision

Less expansion into the background

Better handling of occlusions

Machine Learning based

Tracking with Vision

Less expansion into the background

Better handling of occlusions

Machine Learning based

Runs on CPU, GPU, and A12 Bionic with low power consumption

New Object Tracker

New Object Tracker

Tracking with Vision

Example

```
let requestHandler = VNSequenceRequestHandler()
var inputObservation = VNDetectedObjectObservation(boundingBox: objectBoundingBox)

for _ in 1...5 {
   let frame = frameFeeder.nextFrame()
   let request = VNTrackObjectRequest(detectedObjectObservation: inputObservation)
   try requestHandler.perform([request], on: frame)
   let observation = request.results!.first! as! VNDetectedObjectObservation
   // Process new object location returned in observation.boundingBox
   inputObservation = observation
}
```

Tracking with Vision

Example

```
let requestHandler = VNSequenceRequestHandler()
var inputObservation = VNDetectedObjectObservation(boundingBox: objectBoundingBox)

for _ in 1...5 {
    let frame = frameFeeder.nextFrame()
    let request = VNTrackObjectRequest(detectedObjectObservation: inputObservation)
    try requestHandler.perform([request], on: frame)
    let observation = request.results!.first! as! VNDetectedObjectObservation
    // Process new object location returned in observation.boundingBox
    inputObservation = observation
}
```

Tracking with Vision

Example

```
let requestHandler = VNSequenceRequestHandler()
var inputObservation = VNDetectedObjectObservation(boundingBox: objectBoundingBox)

for _ in 1...5 {
    let frame = frameFeeder.nextFrame()
    let request = VNTrackObjectRequest(detectedObjectObservation: inputObservation)
    try requestHandler.perform([request], on: frame)
    let observation = request.results!.first! as! VNDetectedObjectObservation
    // Process new object location returned in observation.boundingBox
    inputObservation = observation
}
```

Tracking with Vision

Example

```
let requestHandler = VNSequenceRequestHandler()
var inputObservation = VNDetectedObjectObservation(boundingBox: objectBoundingBox)

for _ in 1...5 {
    let frame = frameFeeder.nextFrame()
    let request = VNTrackObjectRequest(detectedObjectObservation: inputObservation)
    request.revision = VNTrackObjectRequestRevision2
    try requestHandler.perform([request], on: frame)
    let observation = request.results!.first! as! VNDetectedObjectObservation
    // Process new object location returned in observation.boundingBox
    inputObservation = observation
}
```

Vision and Core ML

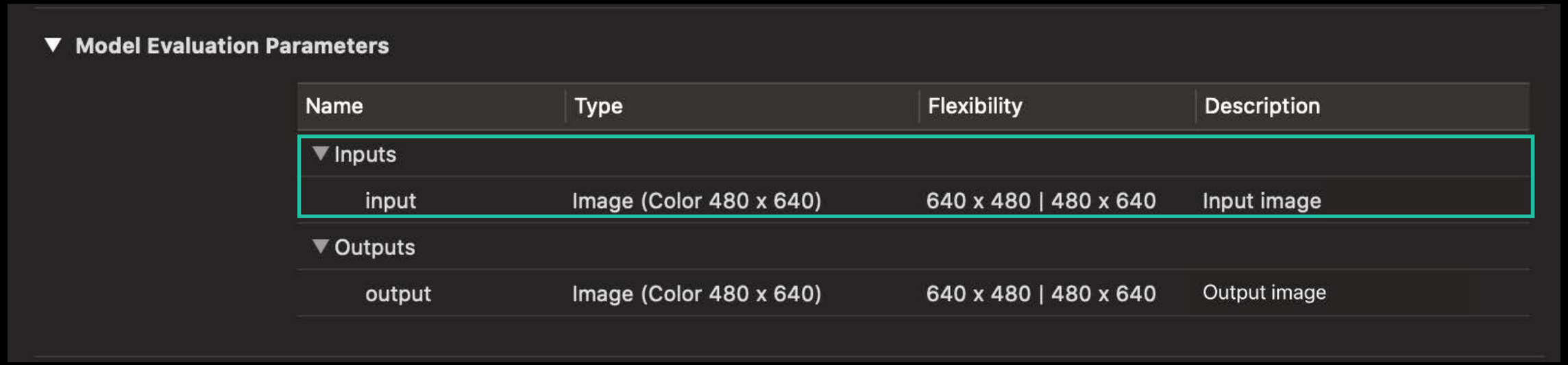
Vision and CoreML Integration Enhancements

▼ **Model Evaluation Parameters**

Name	Type	Flexibility	Description
▼ Inputs			
input	Image (Color 480 x 640)	640 x 480 | 480 x 640	Input image
▼ Outputs			
output	Image (Color 480 x 640)	640 x 480 | 480 x 640	Output image

Vision now works with Core ML models that have single input of image type

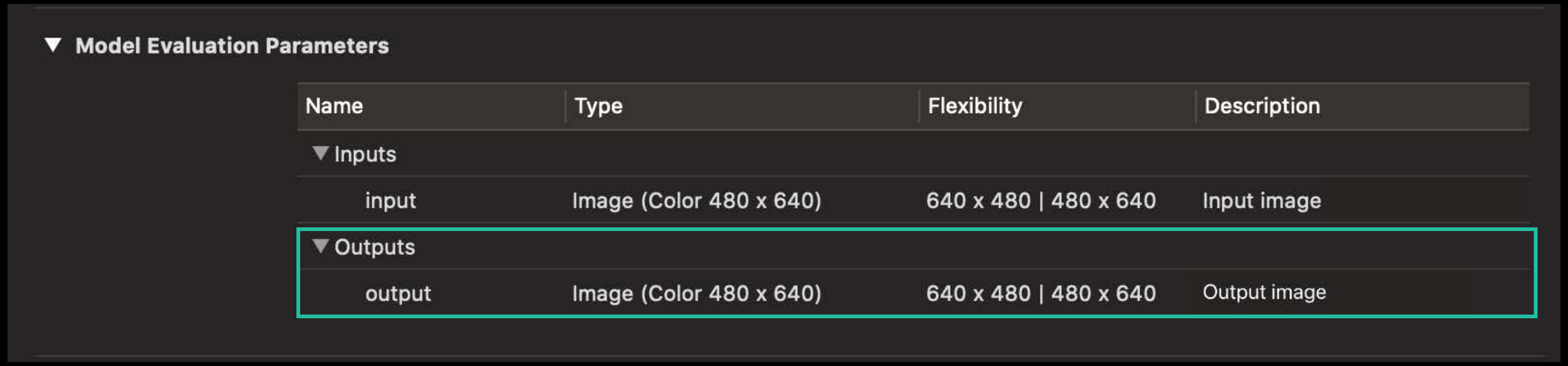
Vision and CoreML Integration Enhancements



Vision now works with Core ML models that have single input of image type

Vision converts '*Inputs*' image to Core ML required input size and color scheme

Vision and CoreML Integration Enhancements



Vision now works with Core ML models that have single input of image type

Vision converts '*Inputs*' image to Core ML required input size and color scheme

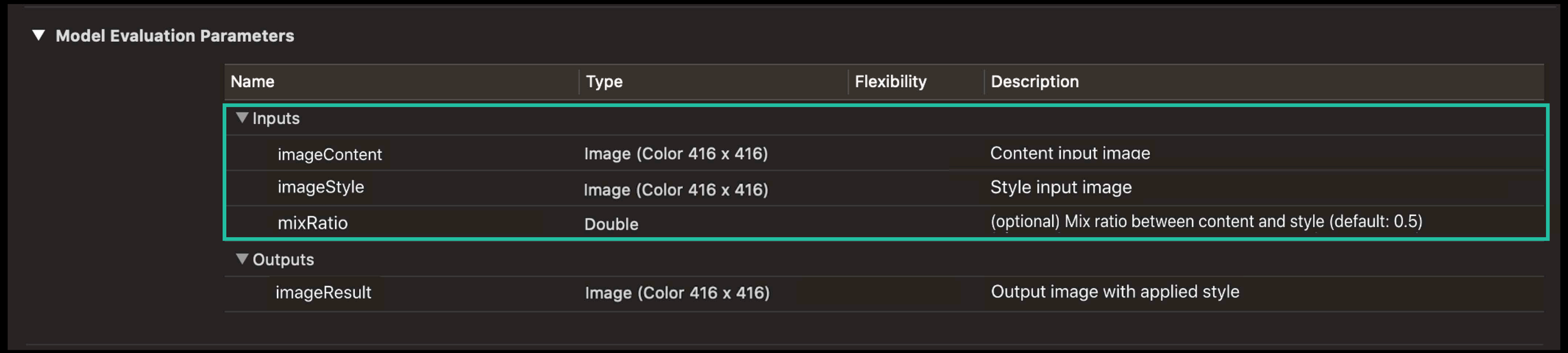
Vision wraps '*Outputs*' into appropriate Observation types

Vision and CoreML Integration Enhancements

▼ **Model Evaluation Parameters**

Name	Type	Flexibility	Description
▼ Inputs			
imageContent	Image (Color 416 x 416)		Content input image
imageStyle	Image (Color 416 x 416)		Style input image
mixRatio	Double		(optional) Mix ratio between content and style (default: 0.5)
▼ Outputs			
imageResult	Image (Color 416 x 416)		Output image with applied style

Vision and CoreML Integration Enhancements



Vision can now work with Core ML models that have one or more *'Inputs'*

- Including multi-image inputs

Vision and CoreML Integration Enhancements

▼ Model Evaluation Parameters

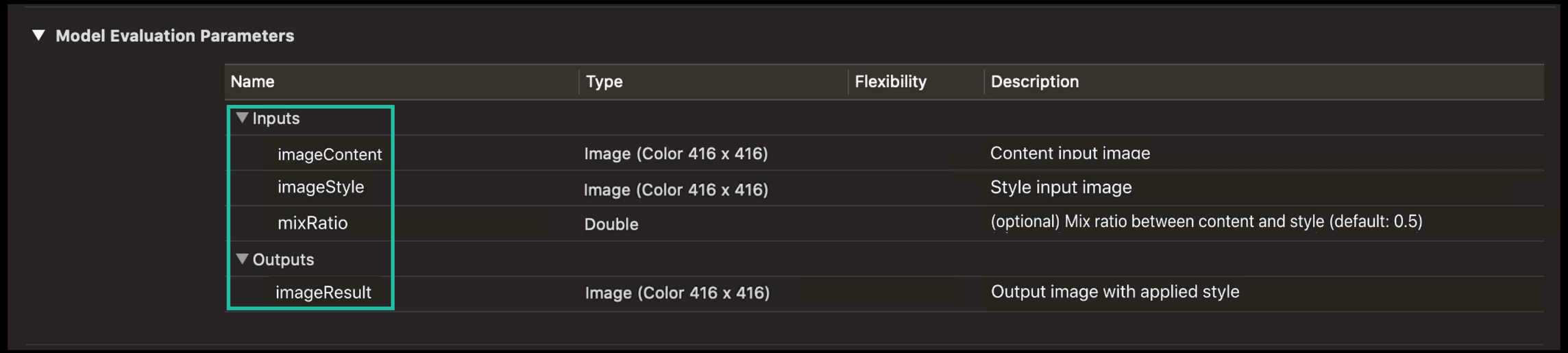
Name	Type	Flexibility	Description
▼ Inputs			
imageContent	Image (Color 416 x 416)		Content input image
imageStyle	Image (Color 416 x 416)		Style input image
mixRatio	Double		(optional) Mix ratio between content and style (default: 0.5)
▼ Outputs			
imageResult	Image (Color 416 x 416)		Output image with applied style

Vision can now work with Core ML models that have one or more *'Inputs'*

- Including multi-image inputs

Vision will use name-mapping of '*Output*' names to Observations

Vision and CoreML Integration Enhancements



▼ Model Evaluation Parameters

Name	Type	Flexibility	Description
▼ Inputs			
imageContent	Image (Color 416 x 416)		Content input image
imageStyle	Image (Color 416 x 416)		Style input image
mixRatio	Double		(optional) Mix ratio between content and style (default: 0.5)
▼ Outputs			
imageResult	Image (Color 416 x 416)		Output image with applied style

Vision can now work with Core ML models that have one or more *'Inputs'*

- Including multi-image inputs

Vision will use name-mapping of '*Output*' names to Observations

Vision and CoreML Integration Flow - New API

```
let mlModel = try MLModel(contentsOf: modelURL)
let visionModel = try VNCoreMLModel(for: mlModel)
visionModel.inputImageFeatureName = "imageContent"
visionModel.featureProvider =
    ["imageStyle" : MLFeatureValue.featureValueWithPixelBuffer(pixelBuffer)
     "mixRatio" : MLFeatureValue.featureValueWithDouble(0.3)]
let request = VNCoreMLRequest(model: visionModel)
let requestHandler = VNImageRequestHandler(url: imageContentURL) // "imageContent"

try requestHandler.perform([request])

let results = request.results!
for case let resultingImage as VNPixelBufferObservation in results {
   if resultingImage.featureName == "imageResult" { /* process results */ }
}
```

More Information

developer.apple.com/wwdc19/222

Text Recognition in Vision Framework	Thursday, 4:00
Machine Learning Labs	WWDC 2019


WWDC19