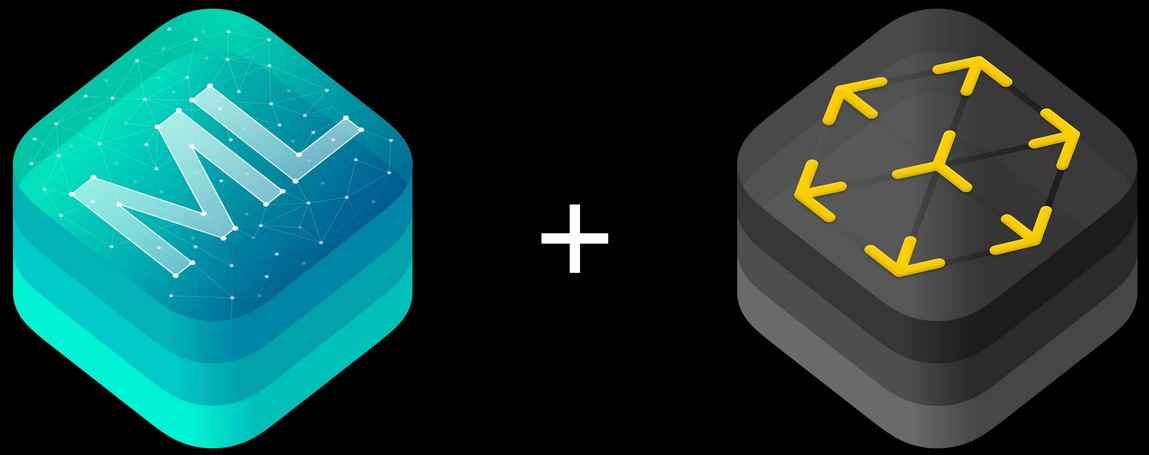
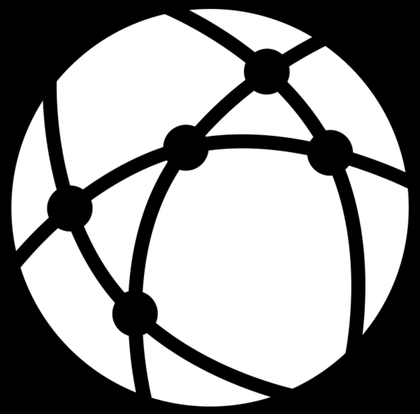
#WWDC19

Creating Great Apps Using Core ML and ARKit

Brent Dimick, Core ML
Scott Gauthreaux, Core ML

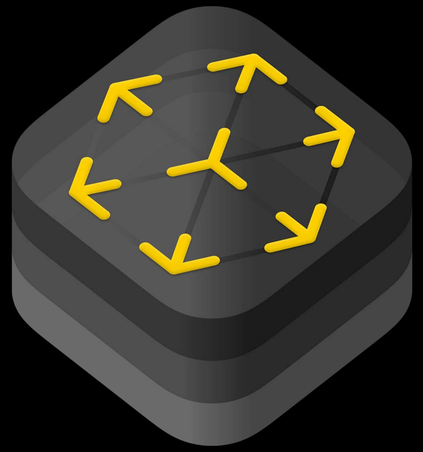



ML



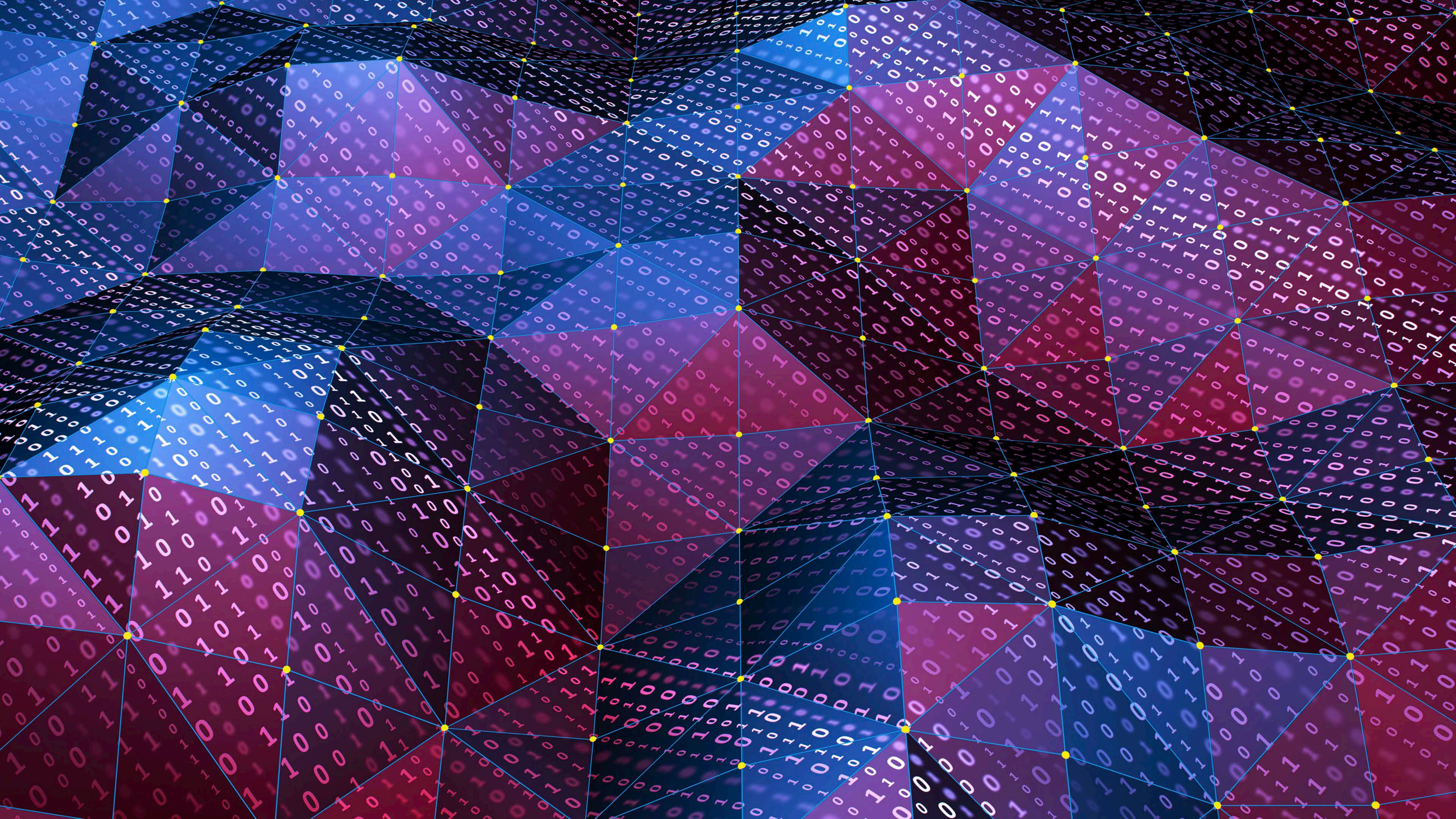

ML


ML

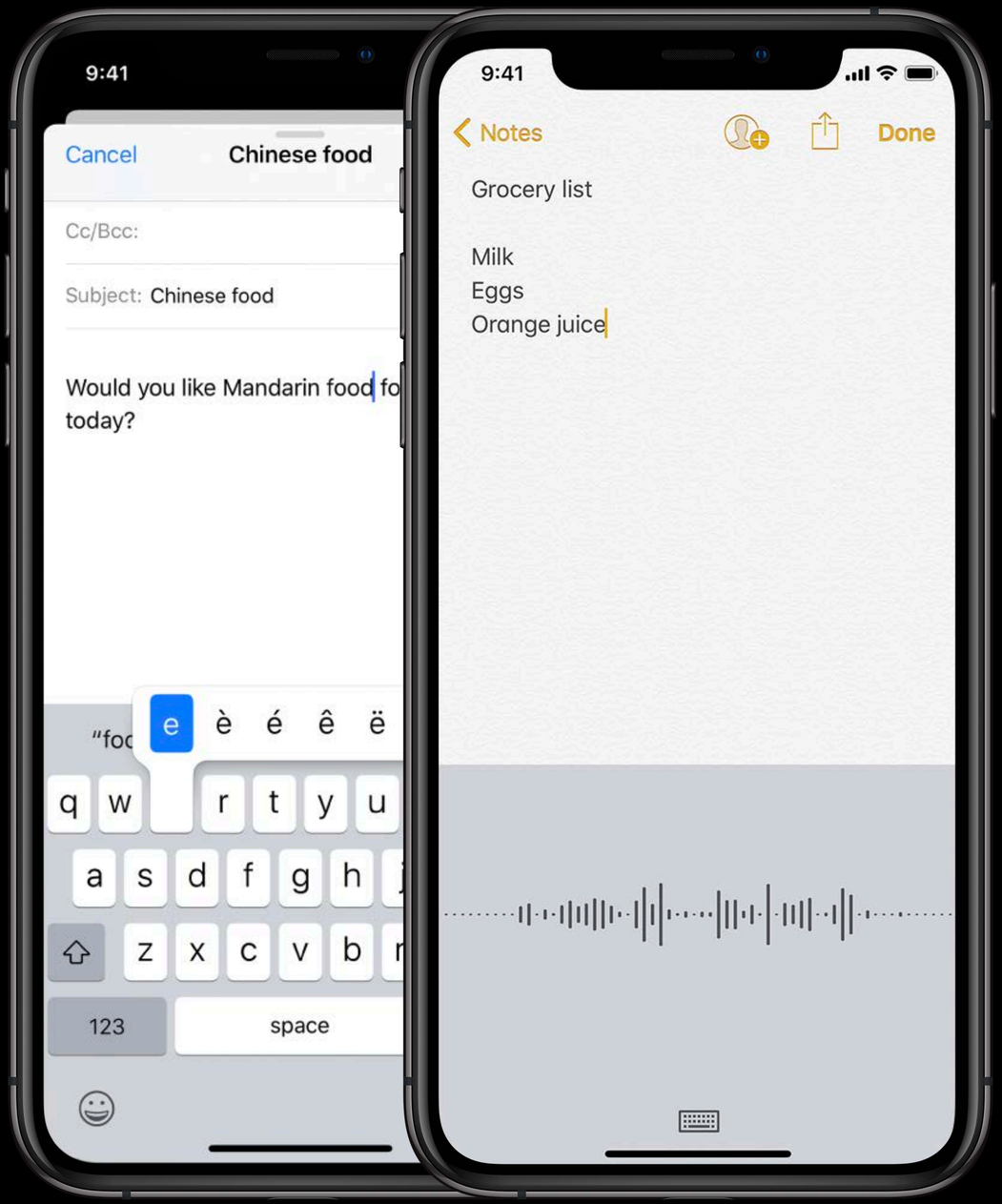



ML

Can Machine Learning help?

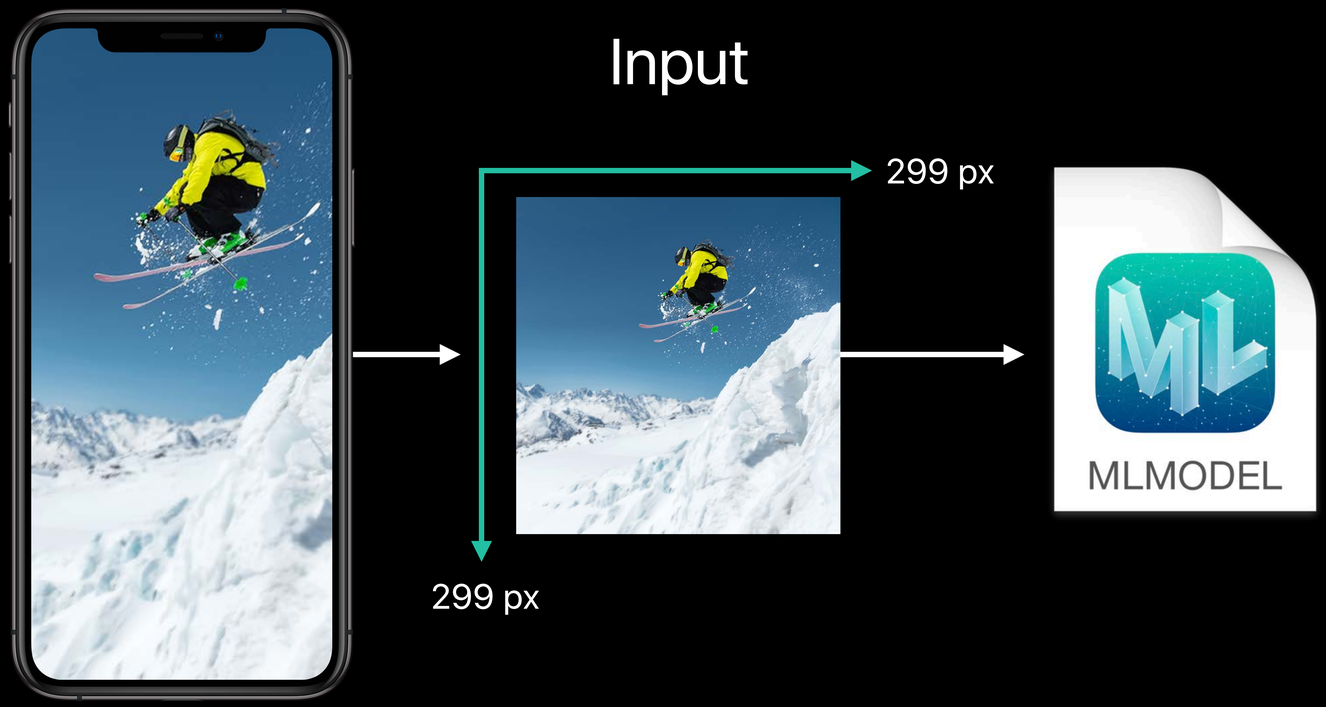


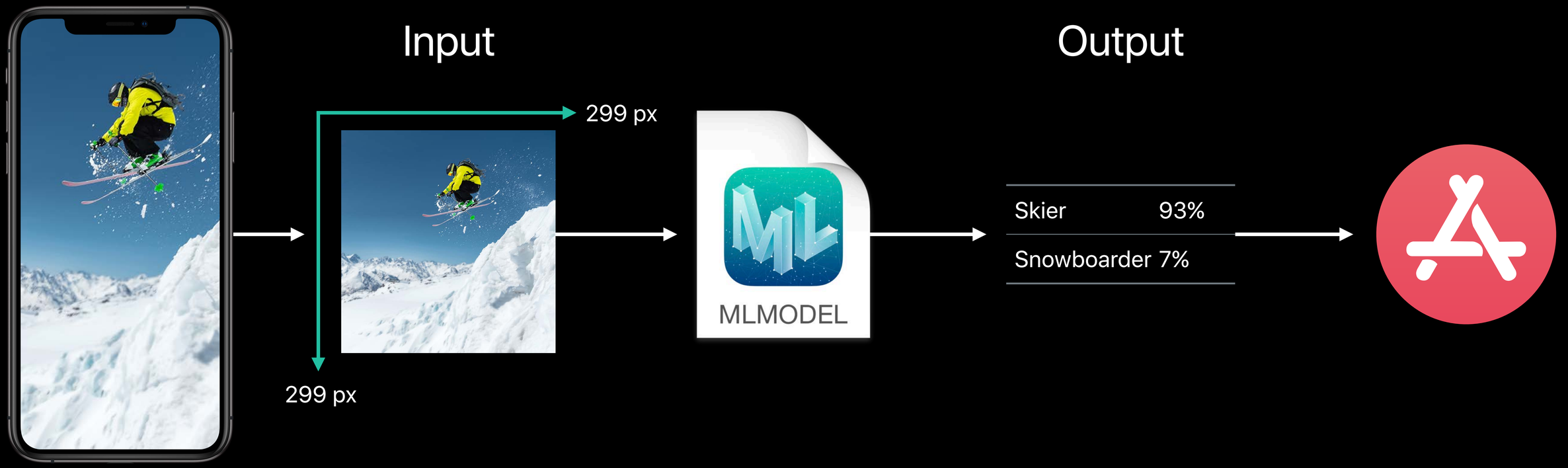

9:41
Cancel
Chinese food
Send
Cc/Bcc:
Subject: Chinese food
Would you like Mandarin food for lunch today?
e è é ê ë ē ė ę
q w r t y u i o p
a s d f g h j k l
z x c v b n m
123
space
return


9:41
Cancel
Chinese food
Cc/Bcc:
Subject: Chinese food
Would you like Mandarin food fo
today?
e è é ê ë
q w r t y u
a s d f g h
z x c v b
123
space
9:41
Notes
Done
Grocery list
Milk
Eggs
Orange juice


9:41
Cancel
Chinese food
Cc/Bcc:
Subject: Chinese food
Would you like Mandarin food fo
today?
e è é ê ë
q w r t y u
a s d f g h
z x c v b
123
space
9:41
Notes
Grocery list
Milk
Eggs
Orange juice
SLO-MO
VIDEO
PHOTO
PORTRAIT
SQUARE

Understanding your model


Input
299 px
299 px
MLMODEL

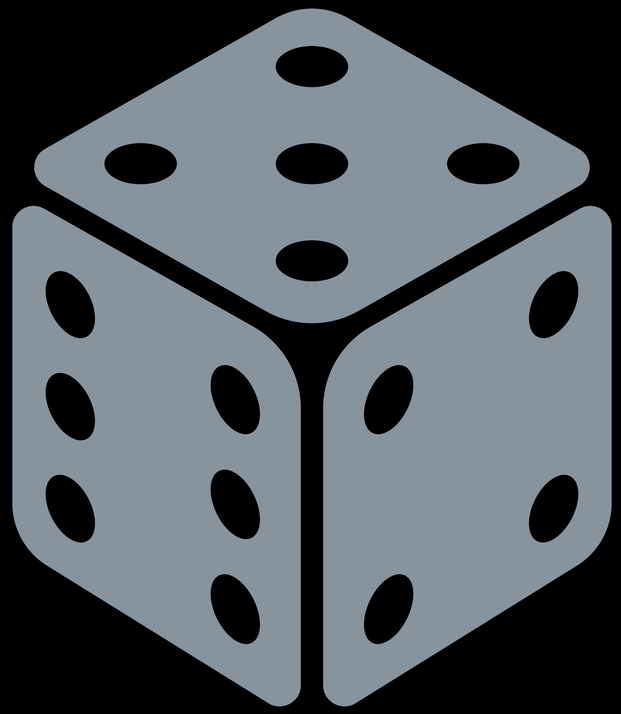

Input
299 px
299 px
MLMODEL
Output
Skier 93%
Snowboarder 7%

Our Journey

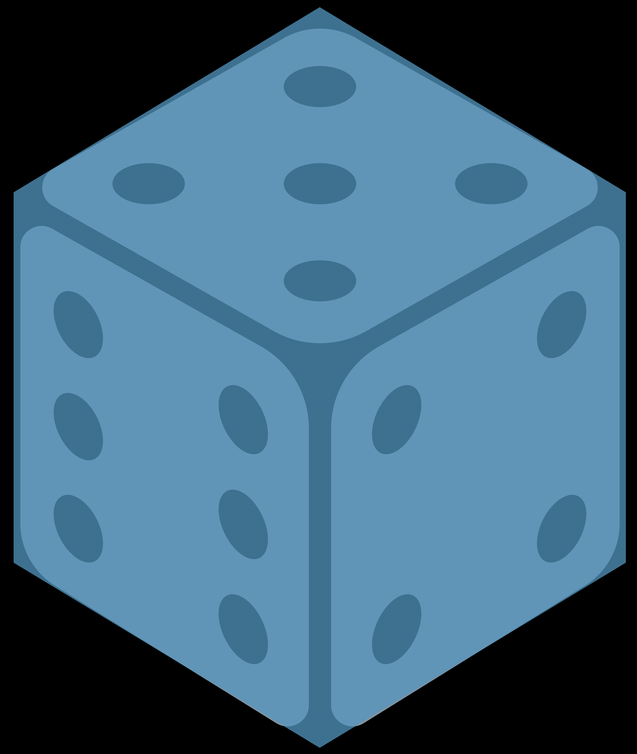
Recognizing dice

Programmatically

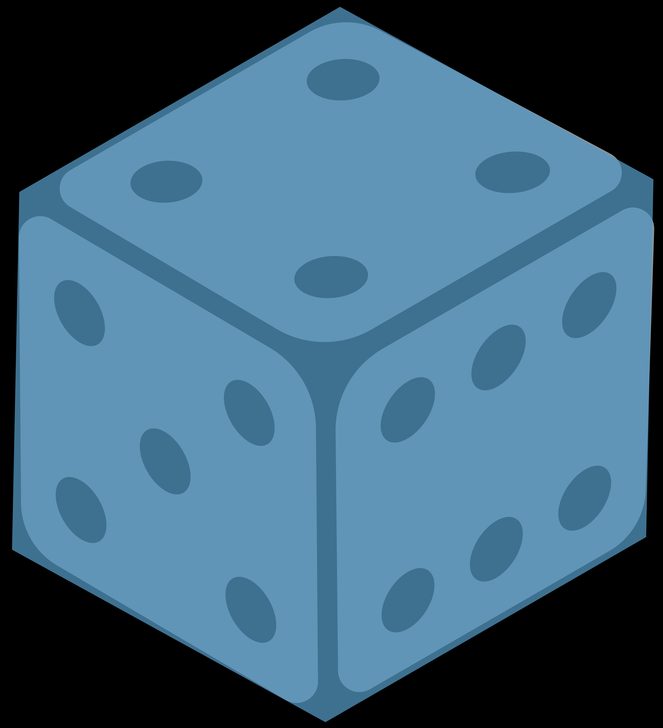
Recognizing Dice



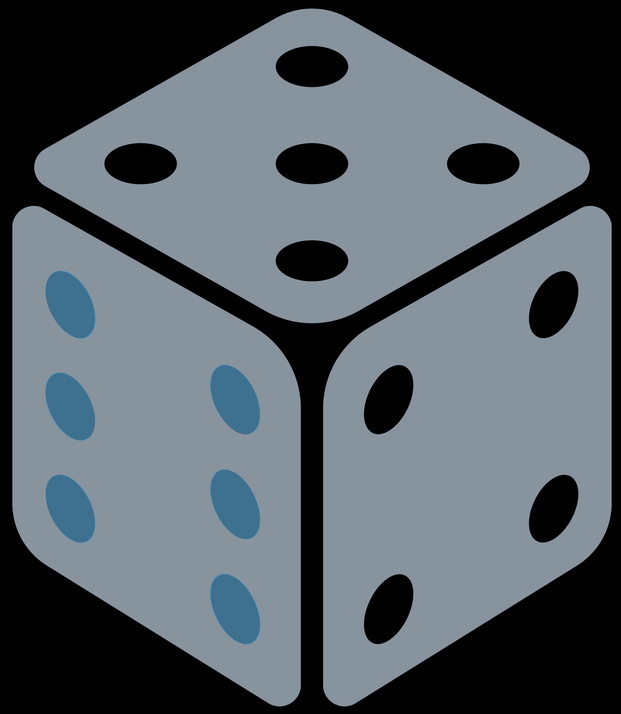
Recognizing Dice



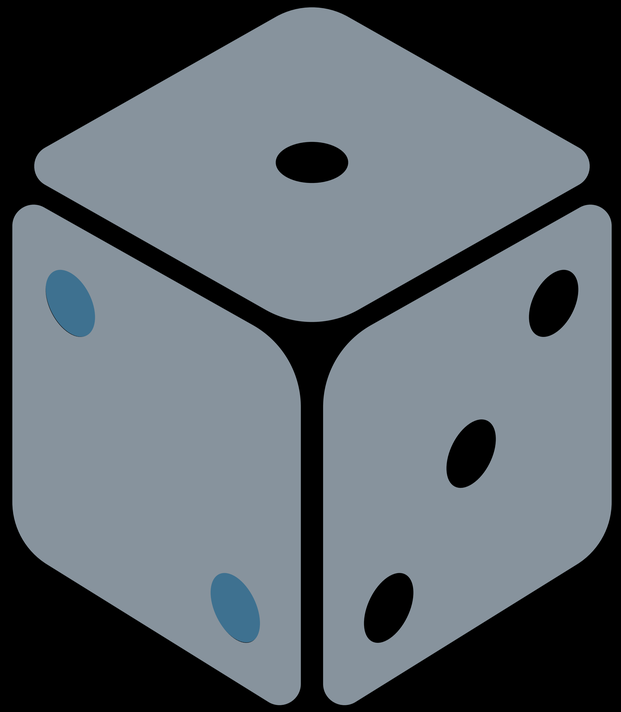
Recognizing Dice



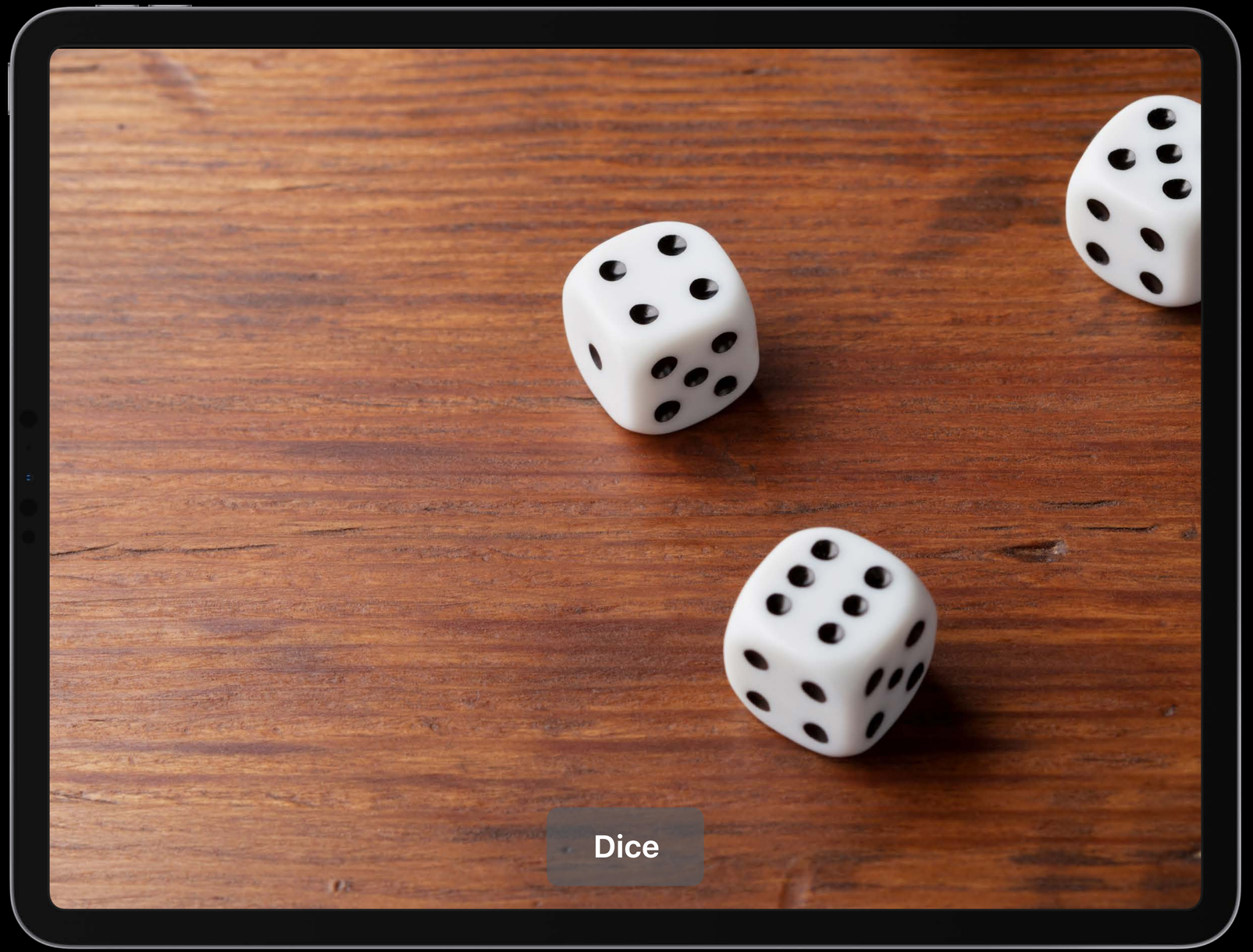
Recognizing Dice

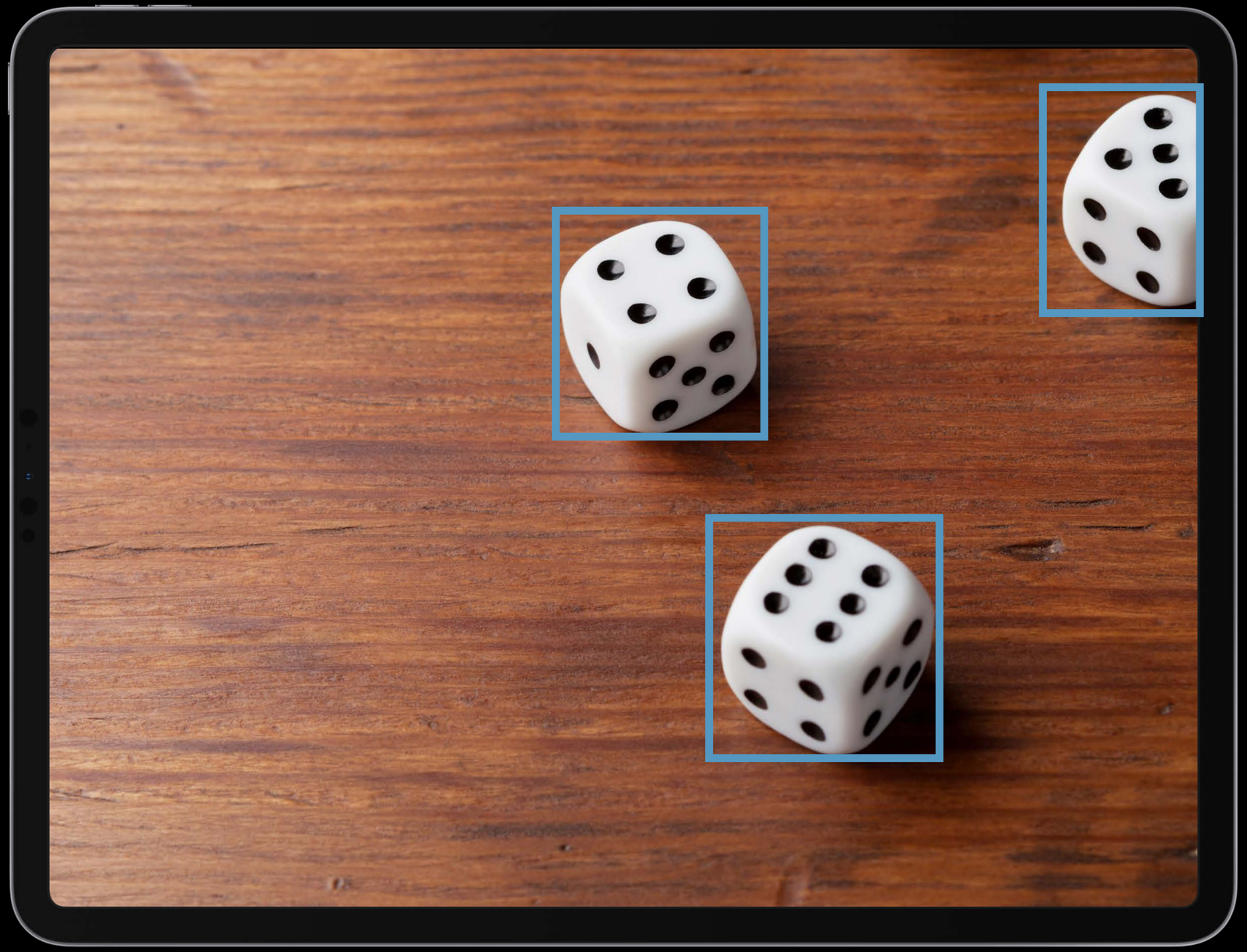


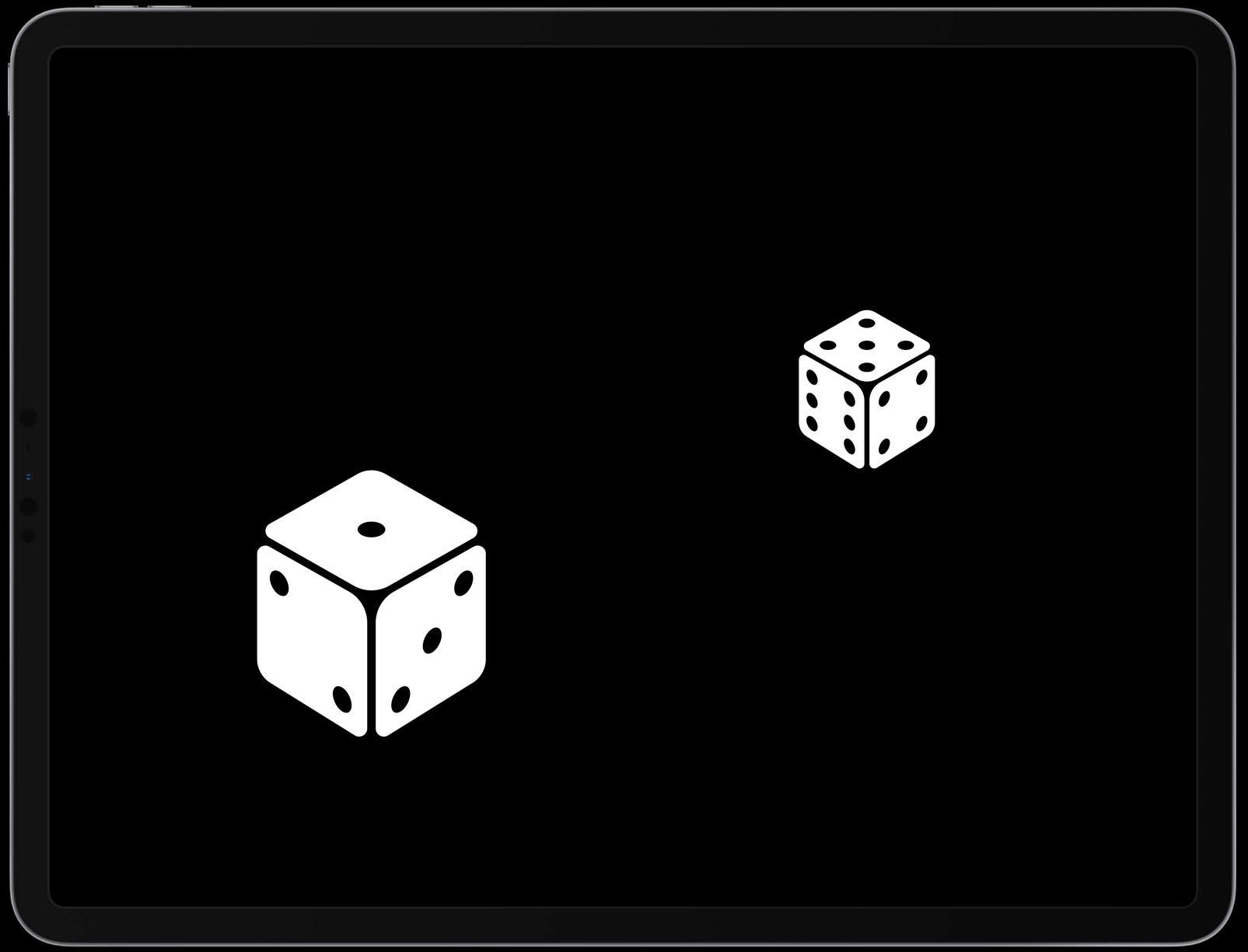
Recognizing Dice

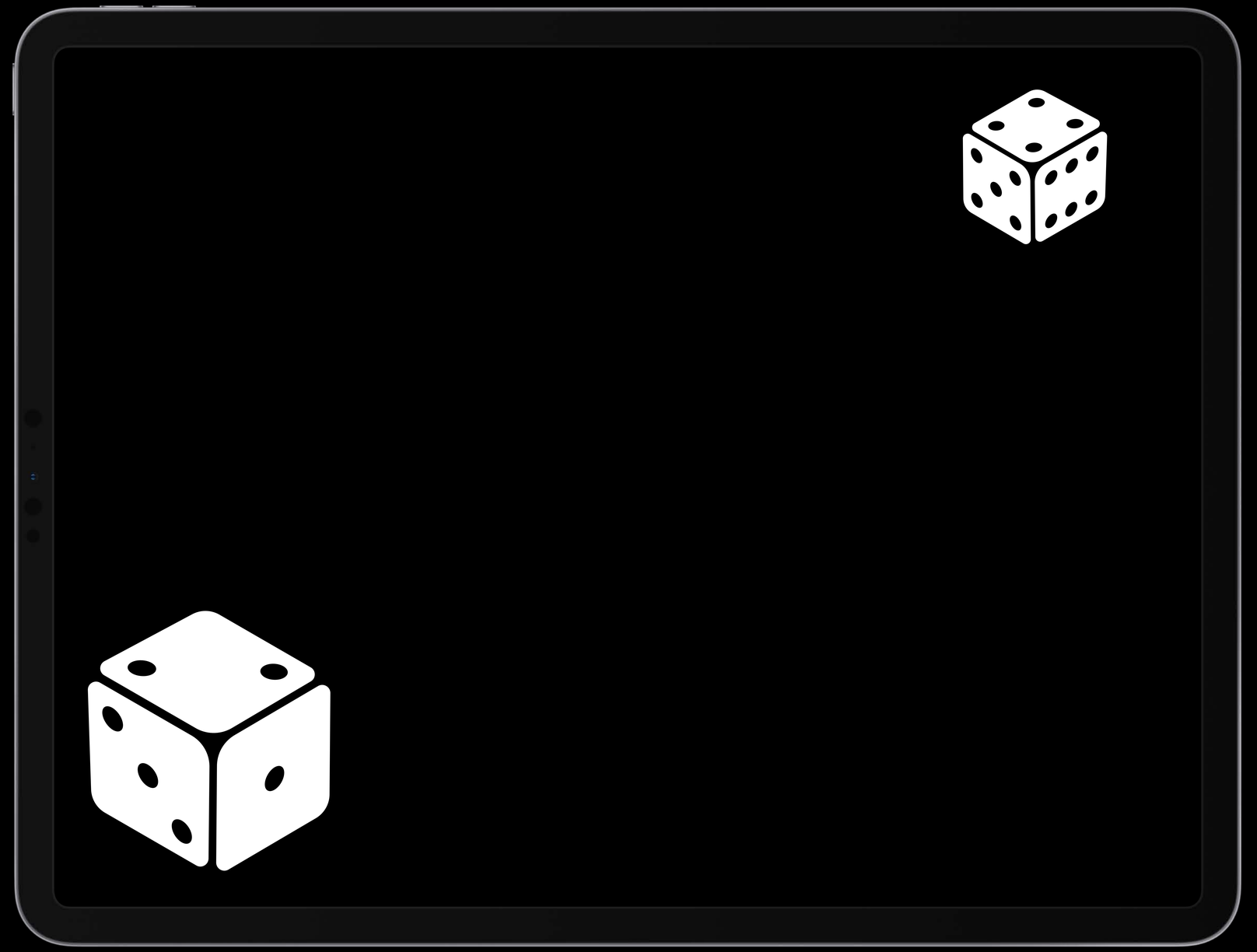


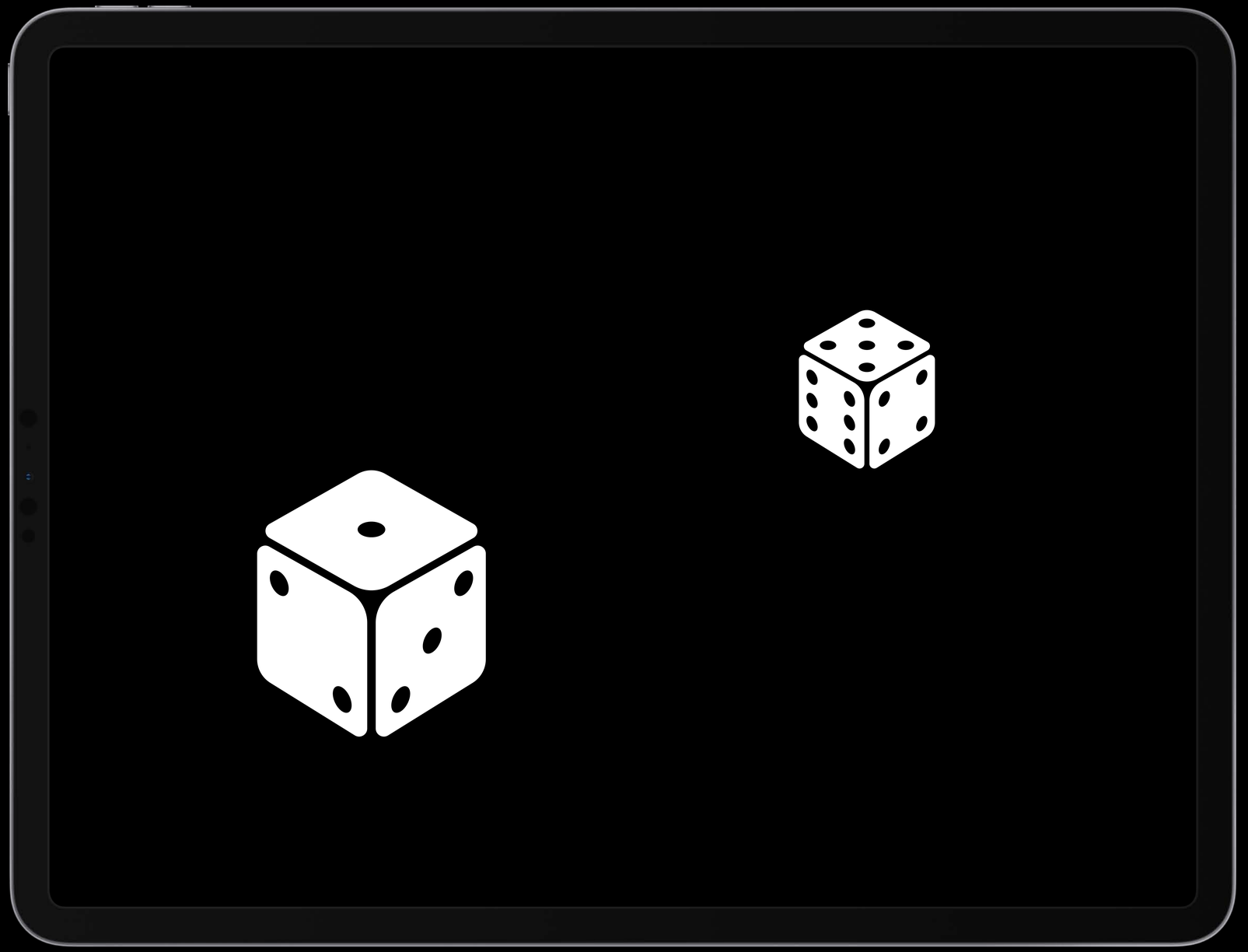
Using Machine Learning

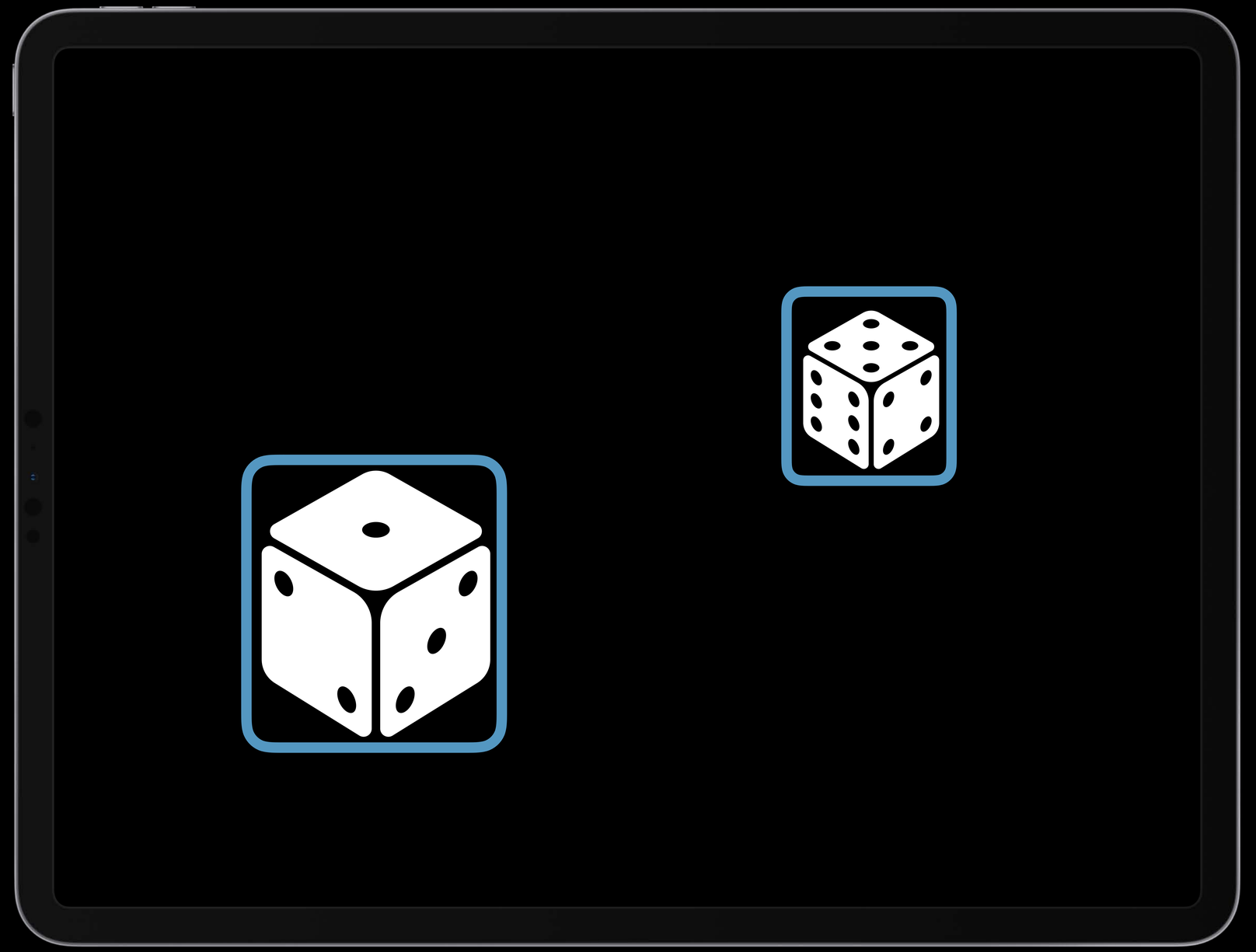

Dice



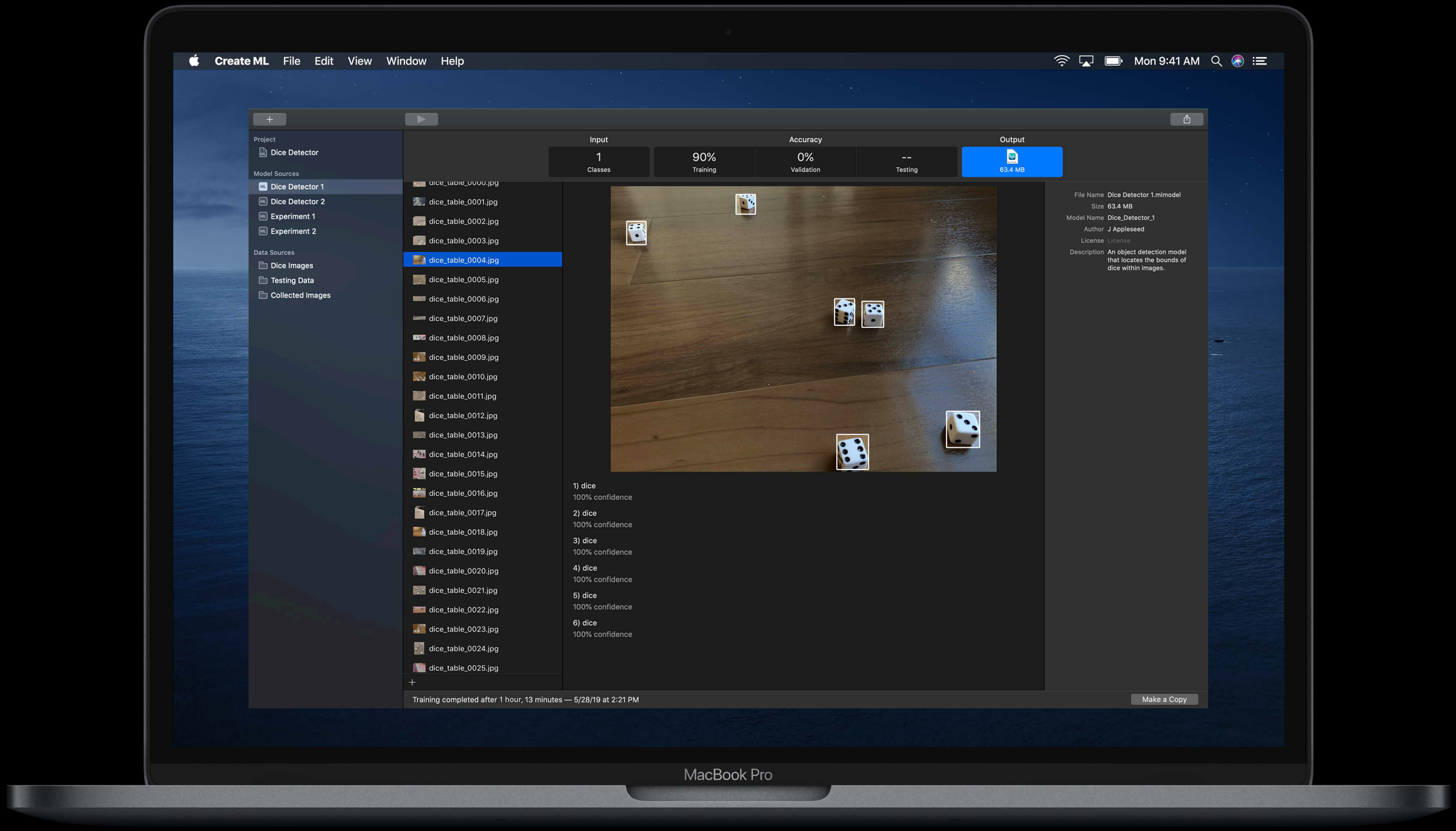









Create ML File Edit View Window Help
Mon 9:41 AM
Choose a Template
All
Image
Sound
Activity
Text
Table
Image
Image Classifier
Object Detector
Sound
Object Detector
A machine learning model that has been trained to detect objects in images.
Cancel
Previous
Next
MacBook Pro


Create ML File Edit View Window Help
Mon 9:41 AM
Project
Dice Detector
Model Sources
Dice Detector 1
Dice Detector 2
Experiment 1
Experiment 2
Data Sources
Dice Images
Testing Data
Collected Images
Input
1
Classes
Accuracy
90%
Training
0%
Validation
--
Testing
Output
63.4 MB
dice_table_0001.jpg
dice_table_0002.jpg
dice_table_0003.jpg
dice_table_0004.jpg
dice_table_0005.jpg
dice_table_0006.jpg
dice_table_0007.jpg
dice_table_0008.jpg
dice_table_0009.jpg
dice_table_0010.jpg
dice_table_0011.jpg
dice_table_0012.jpg
dice_table_0013.jpg
dice_table_0014.jpg
dice_table_0015.jpg
dice_table_0016.jpg
dice_table_0017.jpg
dice_table_0018.jpg
dice_table_0019.jpg
dice_table_0020.jpg
dice_table_0021.jpg
dice_table_0022.jpg
dice_table_0023.jpg
dice_table_0024.jpg
dice_table_0025.jpg
1) dice
100% confidence
2) dice
100% confidence
3) dice
100% confidence
4) dice
100% confidence
5) dice
100% confidence
6) dice
100% confidence
File Name Dice Detector 1.mlmodel
Size 63.4 MB
Model Name Dice_Detector_1
Author J Appleseed
License License
Description An object detection model that locates the bounds of dice within images.
Training completed after 1 hour, 13 minutes — 5/28/19 at 2:21 PM
Make a Copy
MacBook Pro

Demo

Scott Gauthreaux, Core ML

```
// Handling object detection observations

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

        let shapeLayer = self.createRoundedRectLayerWithBounds(objectBounds)
        self.detectionOverlay.addSublayer(shapeLayer)
    }
}
```

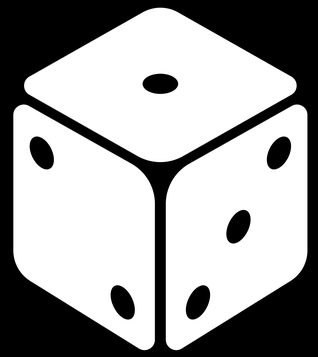
```
// Handling object detection observations

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

        let shapeLayer = self.createRoundedRectLayerWithBounds(objectBounds)
        self.detectionOverlay.addSublayer(shapeLayer)
    }
}
```

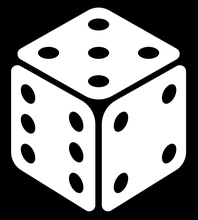
```swift
// Handling object detection observations

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

        let shapeLayer = self.createRoundedRectLayerWithBounds(objectBounds)
        self.detectionOverlay.addSublayer(shapeLayer)
    }
}
```

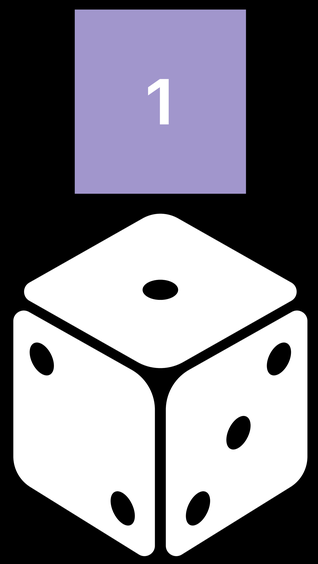
```
// Handling object detection observations

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

        let shapeLayer = self.createRoundedRectLayerWithBounds(objectBounds)
        self.detectionOverlay.addSublayer(shapeLayer)
    }
}
```

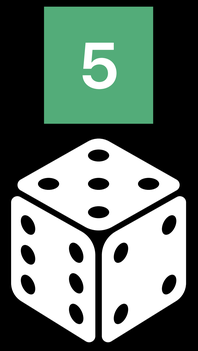
```
// Handling object detection observations

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

        let shapeLayer = self.createRoundedRectLayerWithBounds(objectBounds)
        self.detectionOverlay.addSublayer(shapeLayer)
    }
}
```

One step further.
Recognizing the values on the dice.

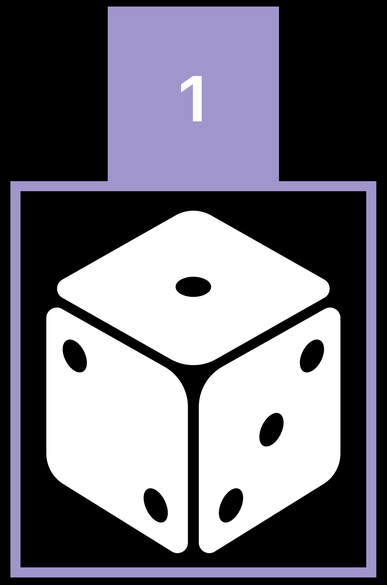


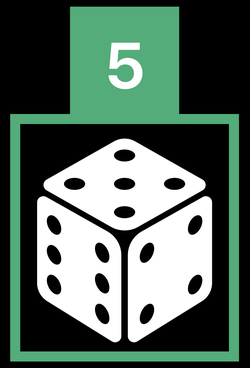



1


5

Data Collection and Annotation

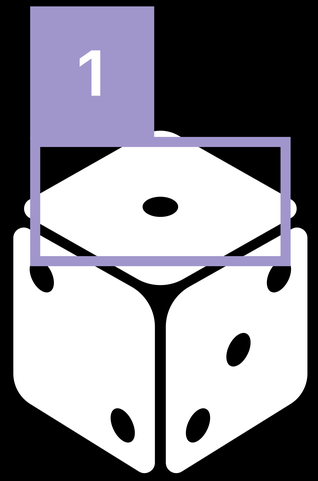





4
5
5


4

Focusing on Tops




4
5
5


4
5
4
5
6


1
6
3

1
6
3

Demo

When Does a Roll End?

When Does a Roll End?

What we observe

When Does a Roll End?

What we observe

Interpreting model output

```swift
// Function to tell if a roll has ended with the given values

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

```swift
// Function to tell if a roll has ended with the given values

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

```
// Function to tell if a roll has ended with the given values

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

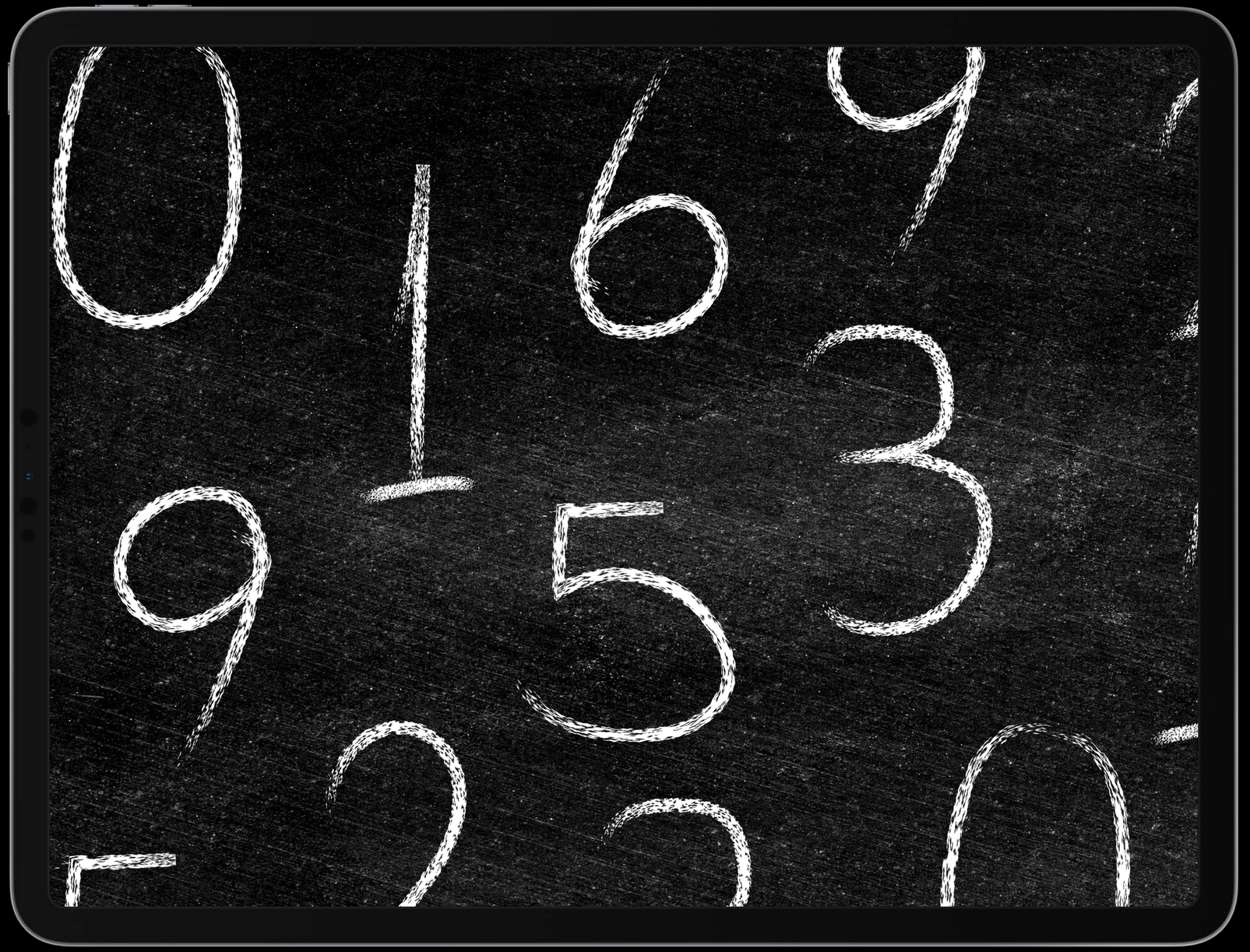
```
// Function to tell if a roll has ended with the given values

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

```swift
// Function to tell if a roll has ended with the given values

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

Recognizing dice

Recognizing dice

Handling input


0 1 6
9 5 3

Safari File Edit View History Bookmarks Window Help

Mon 9:41 AM

developer.apple.com

Developer Discover Design Develop Distribute Support Account

Machine Learning

Overview Core ML Create ML Models Resources

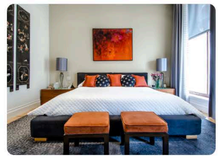
Core ML Models

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

Images Text

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

Images



FCRN-DepthPrediction

Depth Estimation

Predict the depth from a single image.

View models

MNIST

Drawing Classification

Classify a single handwritten digit (supports digits 0-9)

View models

MobileNetV2

Image Classification

The MobileNetv2 architecture trained to classify the dominant object in a camera frame or image.

View models

MacBook Pro

```swift
// Turning a drawing into a number with PencilKit and Vision

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

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

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

try requestHandler.perform([request])
let results = request.results as? [VNClassificationObservation]
```

```swift
// Turning a drawing into a number with PencilKit and Vision

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

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

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

try requestHandler.perform([request])
let results = request.results as? [VNClassificationObservation]
```

NEW

```
// Turning a drawing into a number with PencilKit and Vision

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

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

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

try requestHandler.perform([request])
let results = request.results as? [VNClassificationObservation]
```

```
// Turning a drawing into a number with PencilKit and Vision

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

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

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

try requestHandler.perform([request])
let results = request.results as? [VNClassificationObservation]
```

```swift
// Turning a drawing into a number with PencilKit and Vision

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

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

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

try requestHandler.perform([request])
let results = request.results as? [VNClassificationObservation]
```

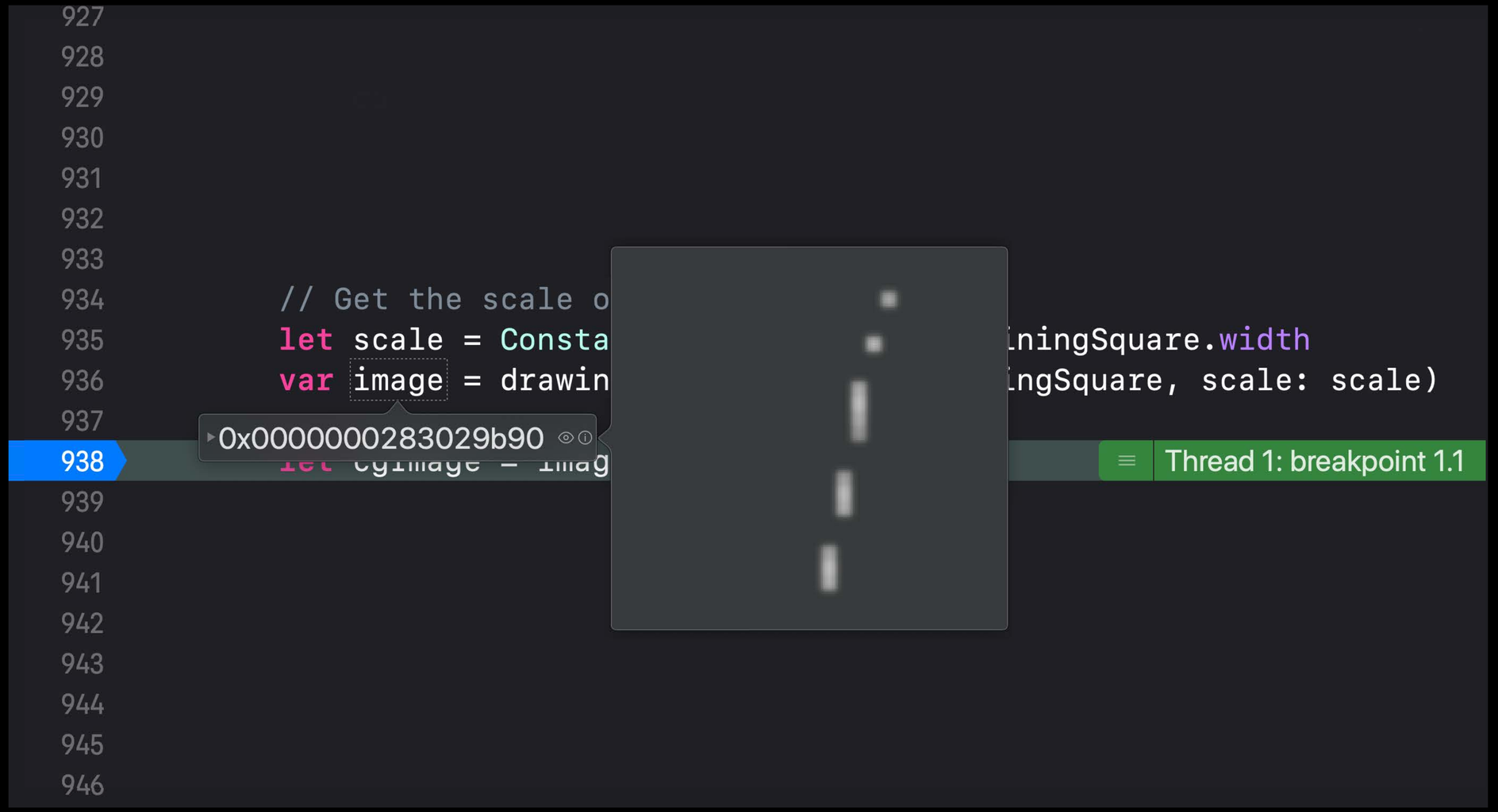
1 2 3 5 9

Predicted

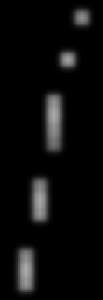
Predicted

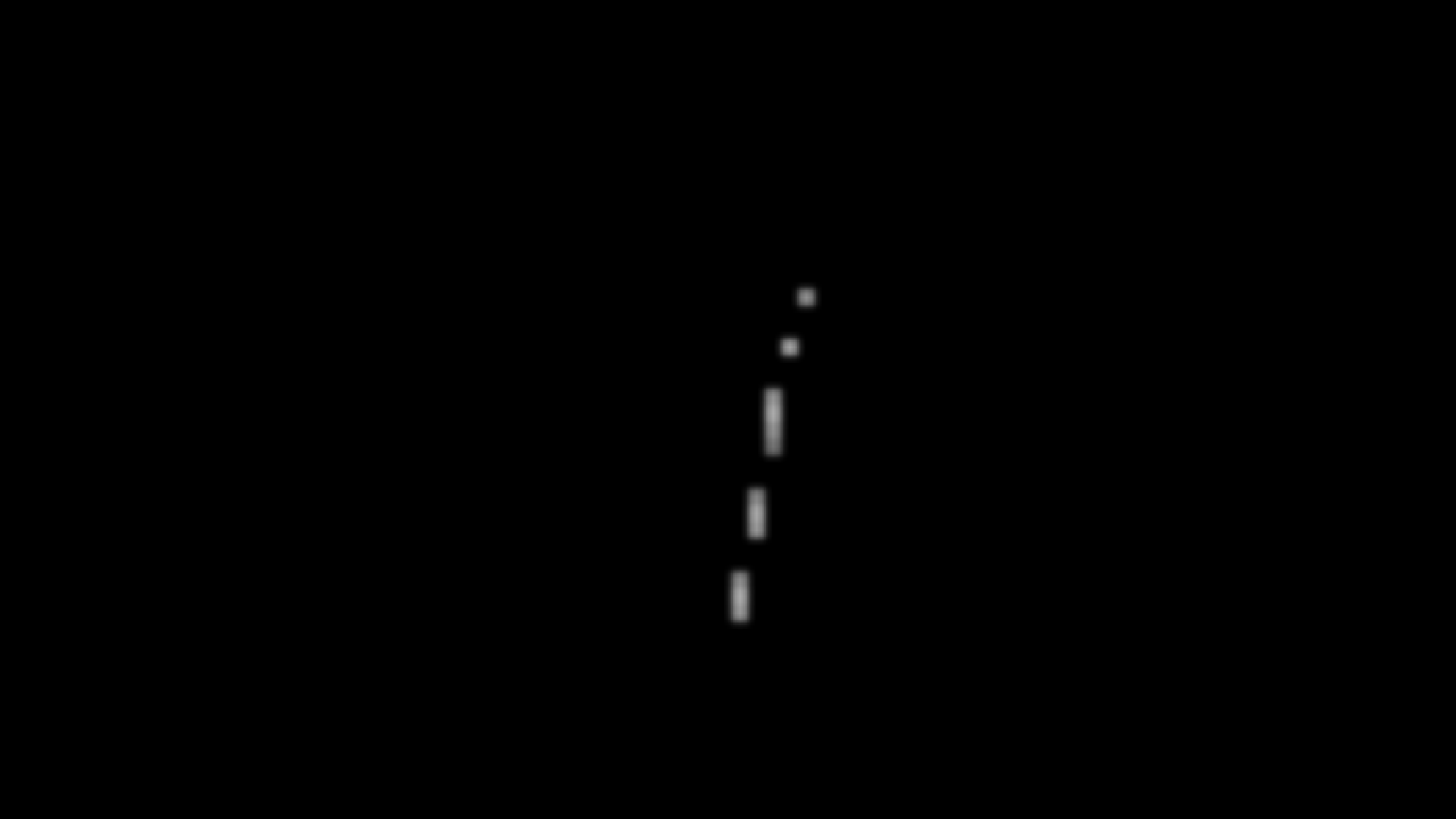
1

```
927
928
929
930
931
932
933
934         // Get the scale of our drawing
935         let scale = Constants.mnistSize / containingSquare.width
936         var image = drawing.image(from: containingSquare, scale: scale)
937
938         let cgImage = image.cgImage
939
940
941
942
943
944
945
946
```


927
928
929
930
931
932
933
934
// Get the scale o
935
let scale = Consta
iningSquare.width
936
var image = drawin
ingSquare, scale: scale)
937
0x0000000283029b90
938
Thread 1: breakpoint 1.1
939
940
941
942
943
944
945
946

Downscaled





Downscaled

Predicted

7

NEW

```
canvasView.allowsFingerDrawing = true
canvasView.tool = PKInkingTool(.marker, color: .white, width: 60.0)
```

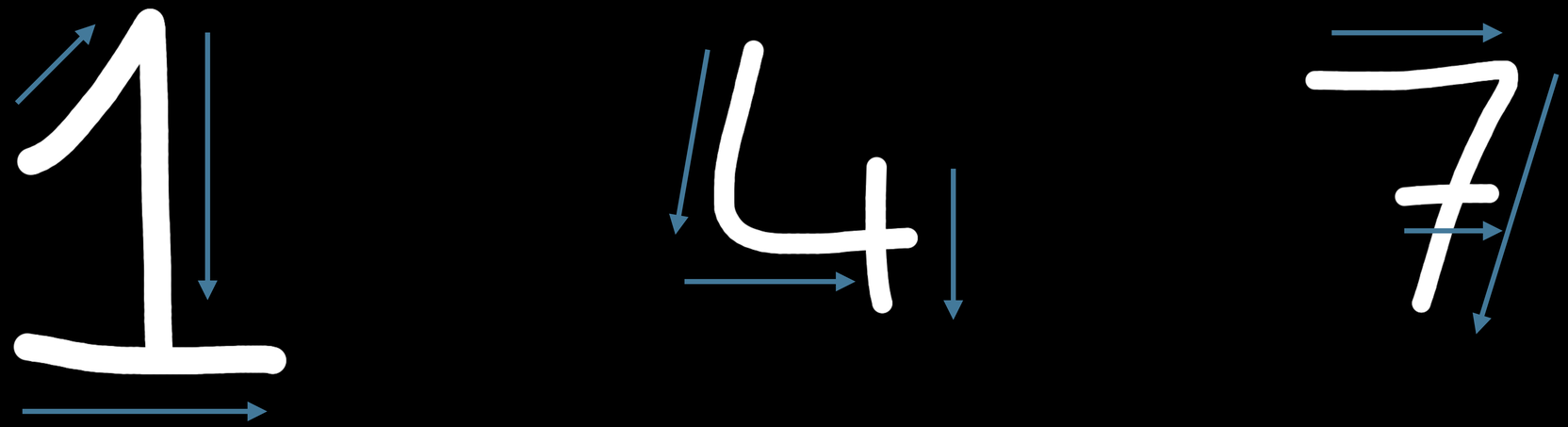
NEW

```
canvasView.allowsFingerDrawing = true
canvasView.tool = PKInkingTool(.marker, color: .white, width: 60.0)
```

NEW

```
canvasView.allowsFingerDrawing = true
canvasView.tool = PKInkingTool(.marker, color: .white, width: 60.0)
```

1 4 7



12

1

12

Demo

NEW

```
// How to enable offline Speech to Text

let speechRecognitionRequest = SFSpeechAudioBufferRecognitionRequest()

speechRecognitionRequest.requiresOnDeviceRecognition = true
```

NEW

```
// How to enable offline Speech to Text

let speechRecognitionRequest = SFSpeechAudioBufferRecognitionRequest()

speechRecognitionRequest.requiresOnDeviceRecognition = true
```

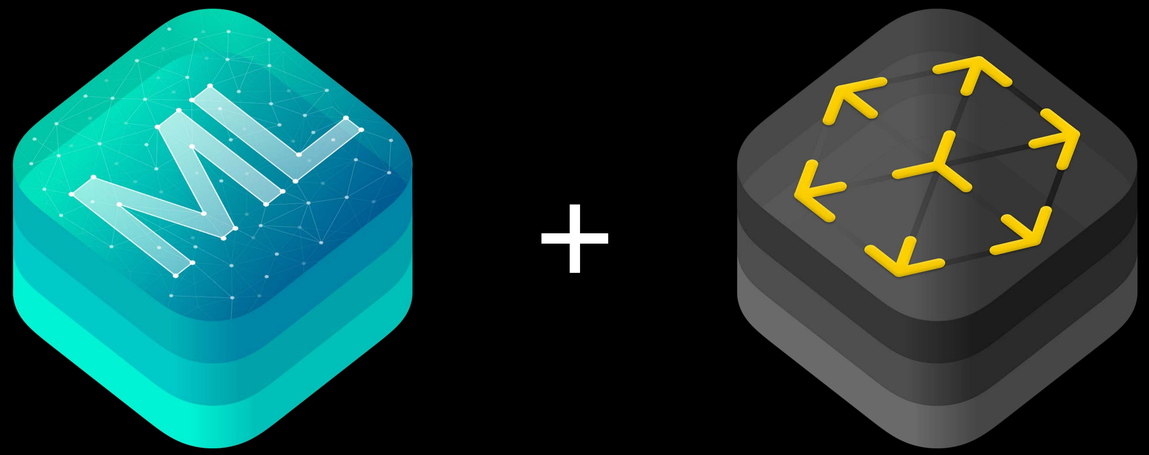
Recognizing dice

Handling input

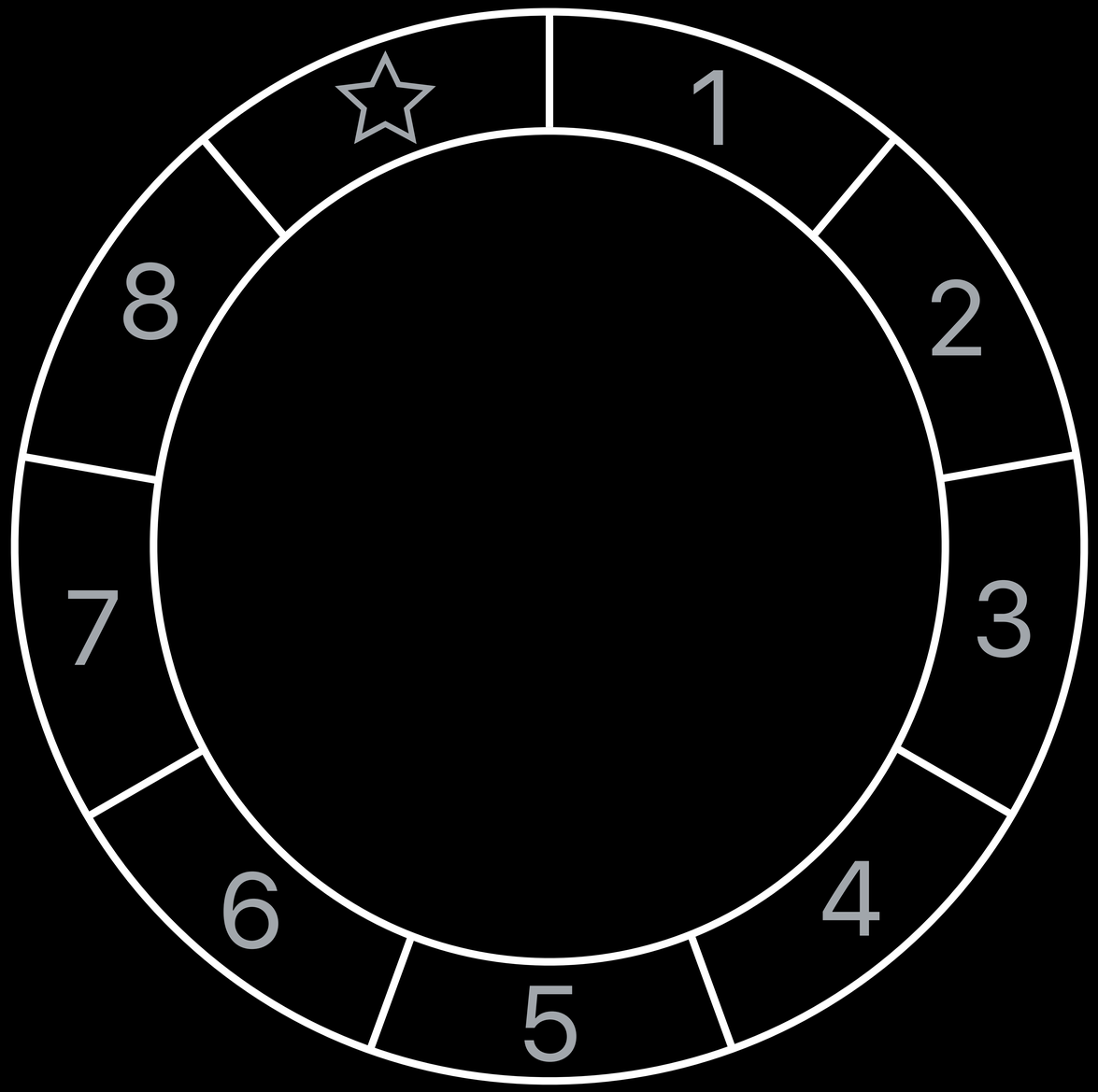
Recognizing dice

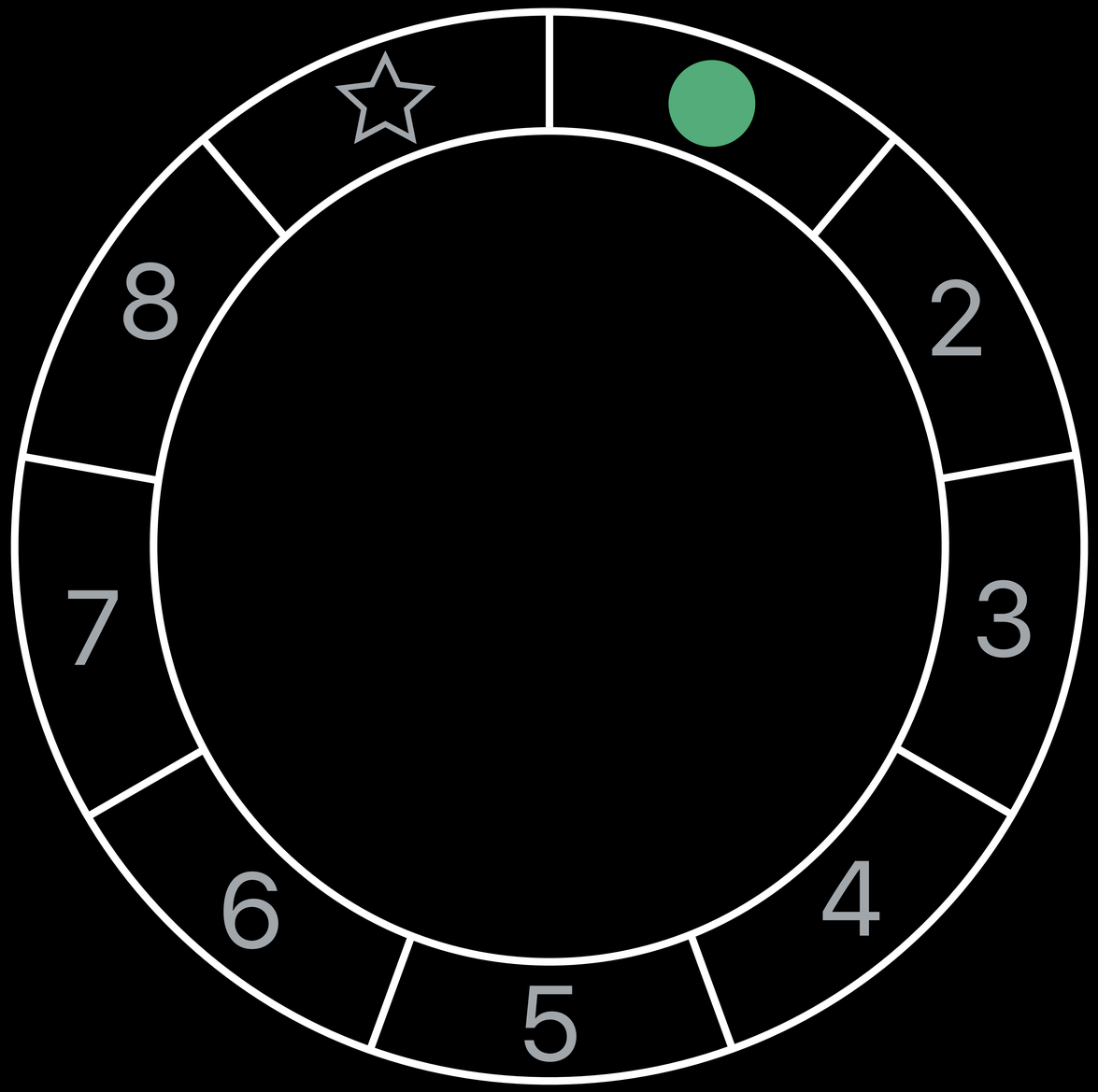
Handling input

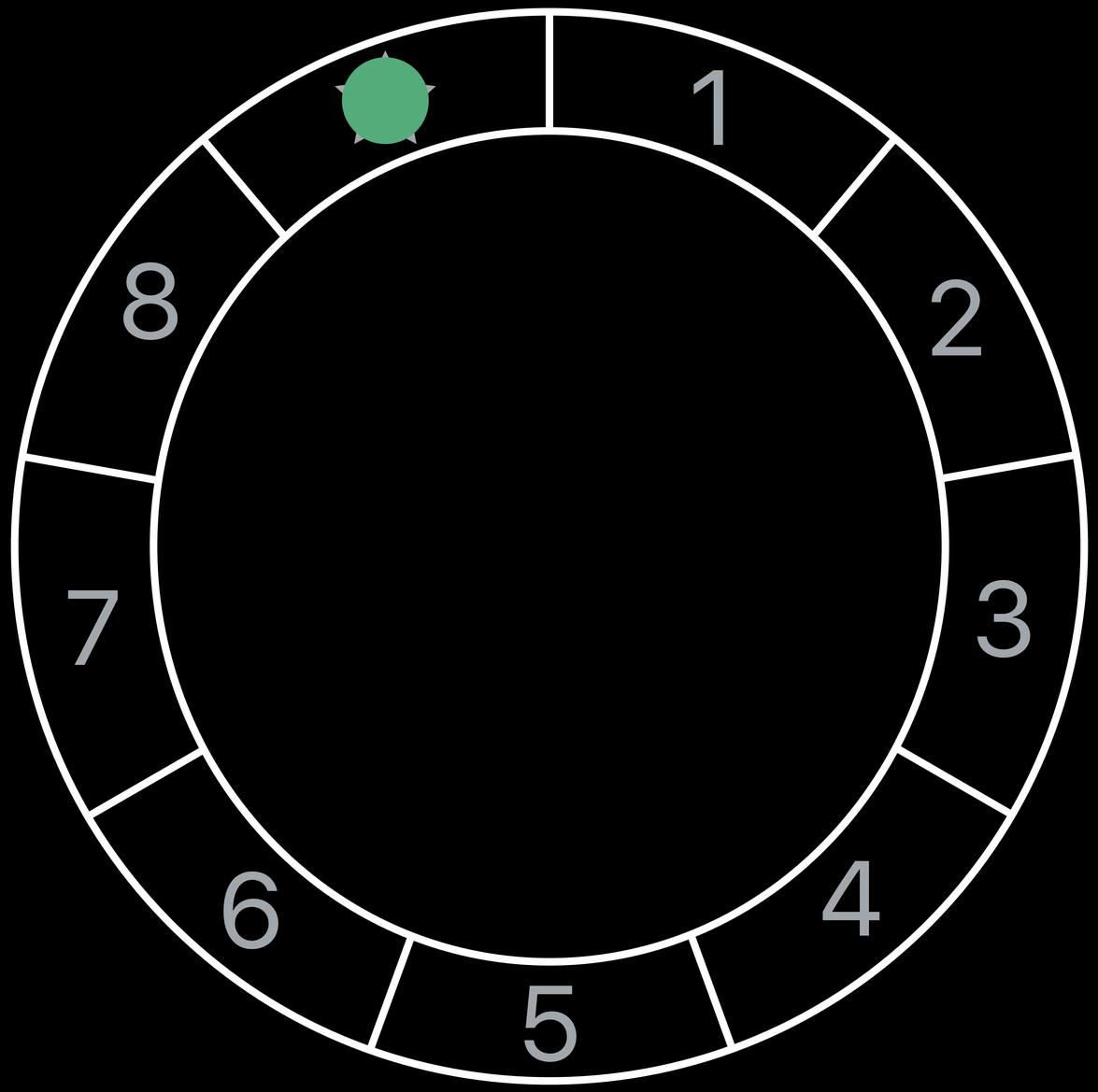
Finalizing the game

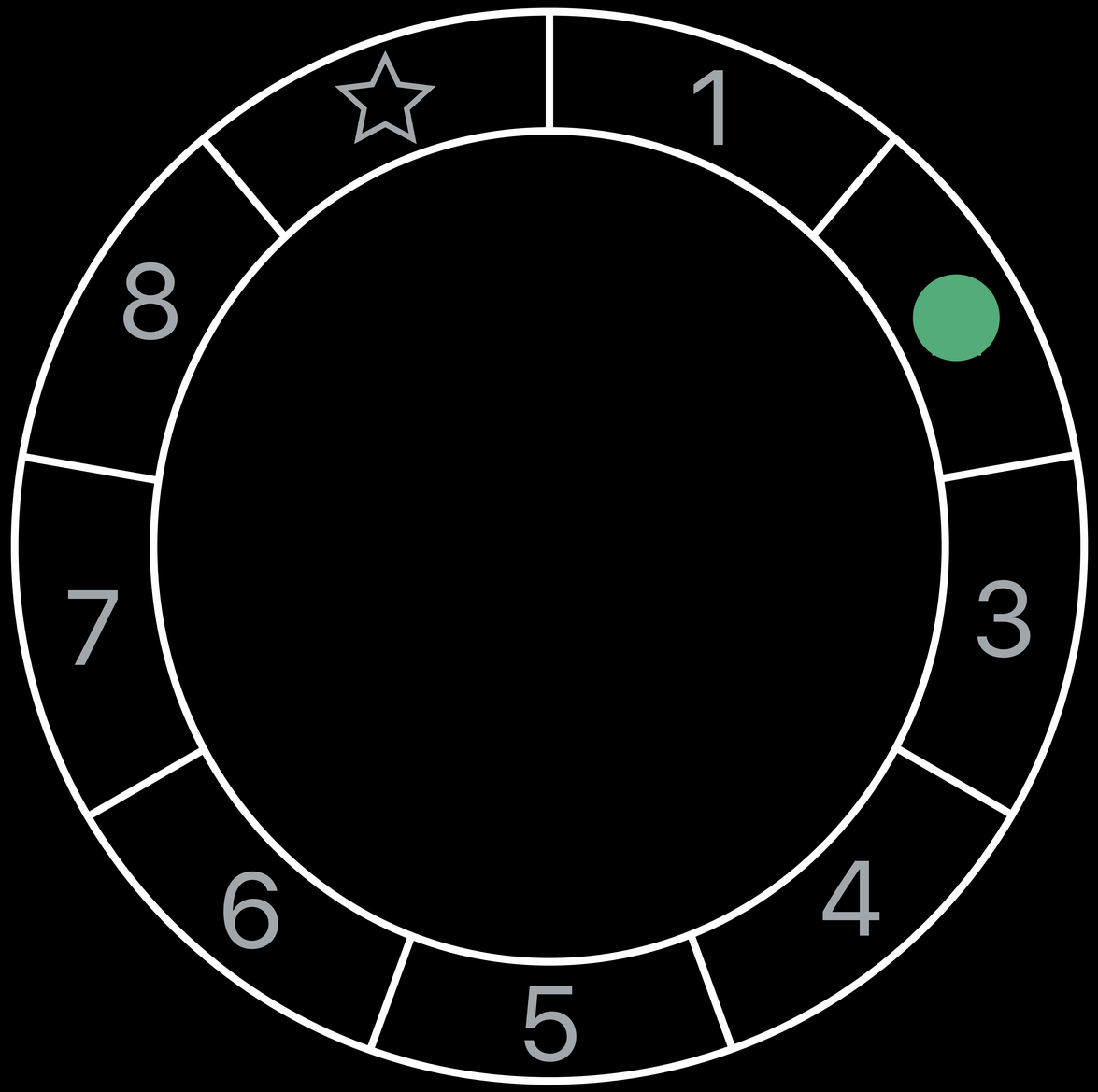


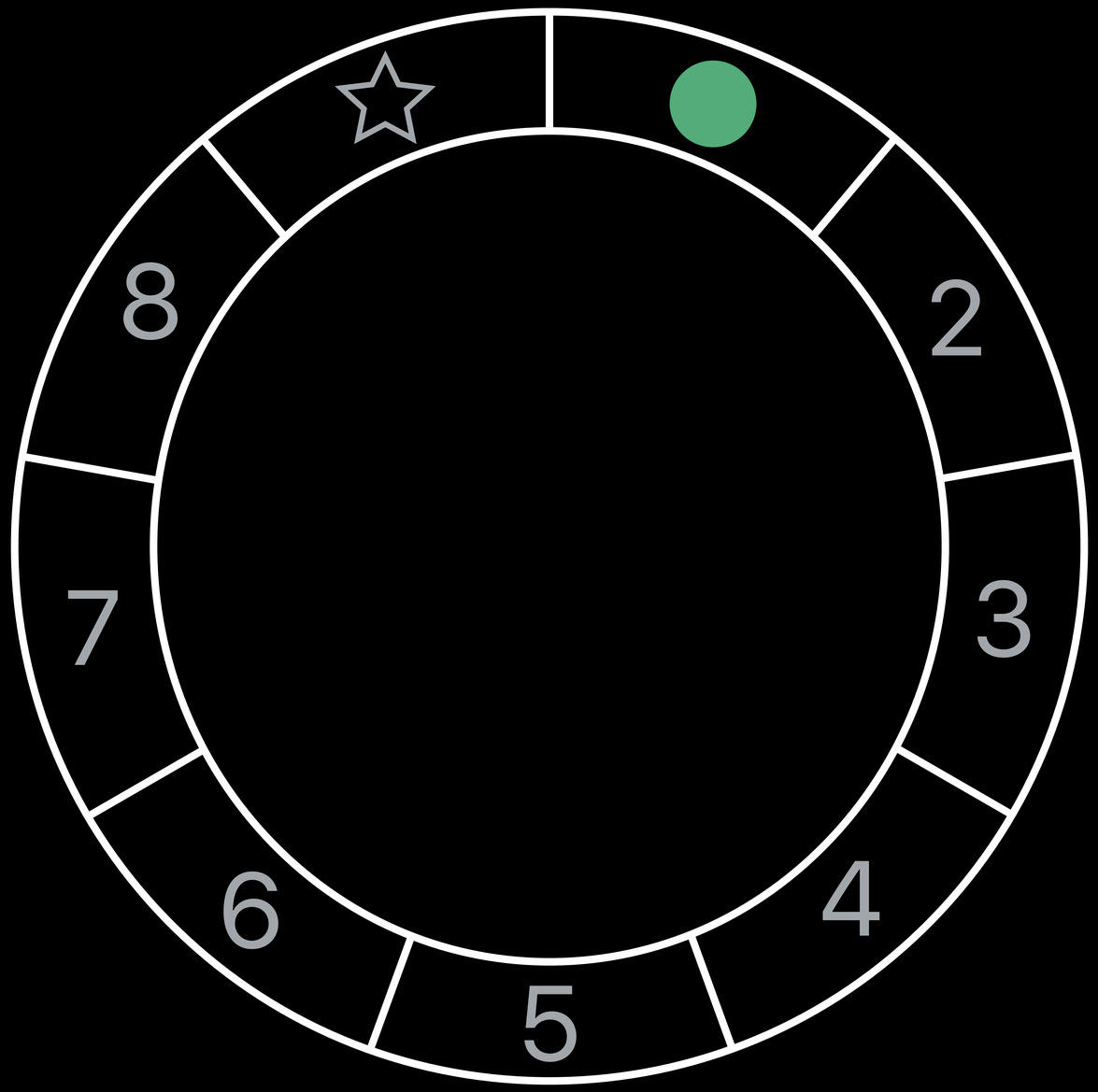
Rules of the Game

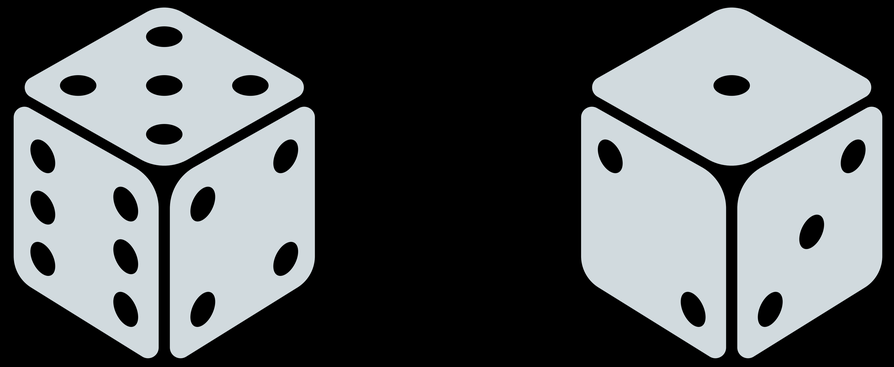

1
2
3
4
5
6
7
8

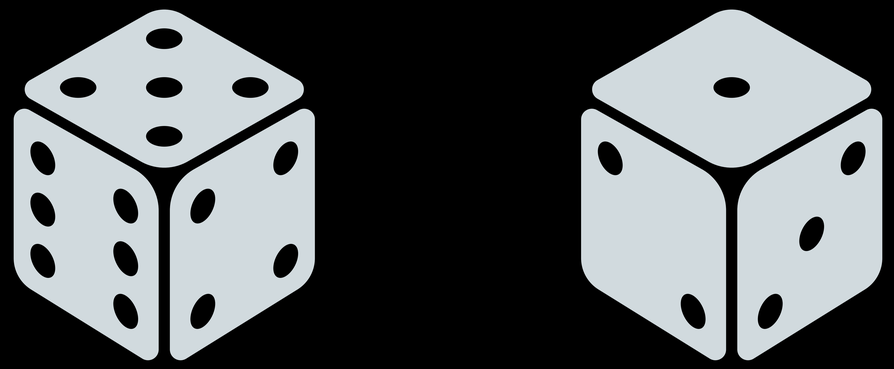

2
3
4
5
6
7
8


1
2
3
4
5
6
7
8

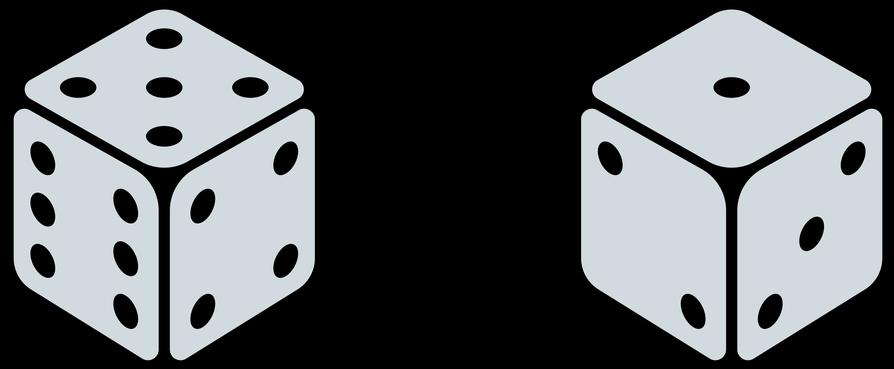

1
3
4
5
6
7
8


2
3
4
5
6
7
8





5 + 1 = 6



5 − 1 = 4

Demo

Summary

Recognizing dice

Handling input

Finalizing the game

More Information

developer.apple.com/wwdc19/228

Machine Learning Lab

Friday, 2:00


WWDC19