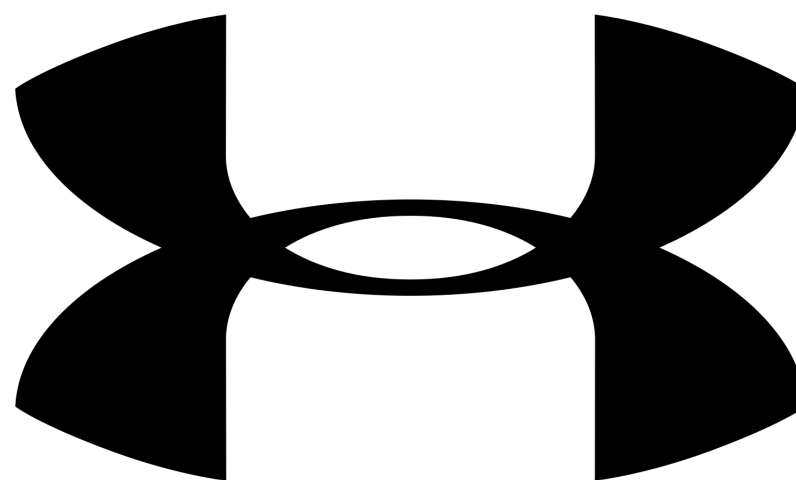


Machine Learning on Apple Watch

By: Conrad Stoll



UNDER ARMOUR



Agenda

- Apple Watch History
- Training a Model for CoreML
- Building an Apple Watch Game with CoreML

The Apple Watch was
released on April 24,
2015.

Every year, watchOS is
getting better 😊

How will watchOS 4 help
developers build better
Apple Watch apps?



Apple Watch Developers

CoreML

and Accelerate

Most apps are designed to
assume there is an iPhone...

We need Apple Watch apps
to work without an iPhone

Goal for this year: Ship an app using CoreML



Conrad Stoll

@conradstoll



Goal for this year: find a helpful use-case to add to an app, and use CoreML to implement it. [#WWDC17](#)

5:50 PM - 7 Jun 2017 from [Downtown, Austin](#)

Game: Snowman



Draw letters to guess a word or phrase



Apple Watches don't have
keyboards... 🙄



Uses Machine Learning for
Handwriting Recognition

Public MNIST dataset for digits

Lots of tutorials online

Recently includes letters!!!

Tutorials work for digits and letters!!!

A model needs to learn how to recognize letters a user might draw.

Training the model needs to happen
before the game is shipped.

Training does not happen on the watch.

The watch uses the model by itself,
without the iPhone.

Training does not change no matter
how many letters a user draws.



Training is generic for all users.*

Download a customized MLMModel

[illegible]

Tools

- Python
- scikit-learn
- Keras
- Tensorflow
- coremltools

Resources

→ Machine Learning Guide Podcast



<http://ocdevel.com/podcasts/machine-learning>

Dataset

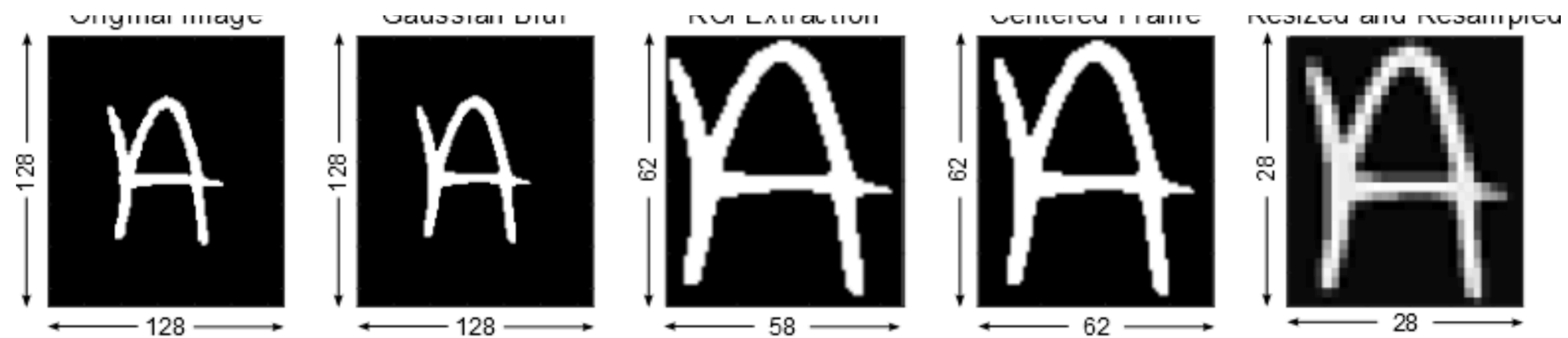
Extended MNIST

Extended MNIST

387,361 training letter images²

23,941 testing images

28x28 pixels



² <https://arxiv.org/pdf/1702.05373.pdf>

Model 1

Support Vector Machine

SVM models are easy to
get started with

Simple application of Supervised Learning

Model 1

```
num_samples = 10000
```

```
images = all_images[0:num_samples]
```

```
labels = all_labels[0:num_samples]
```

```
from sklearn import svm
```

```
clf = svm.SVC()
```

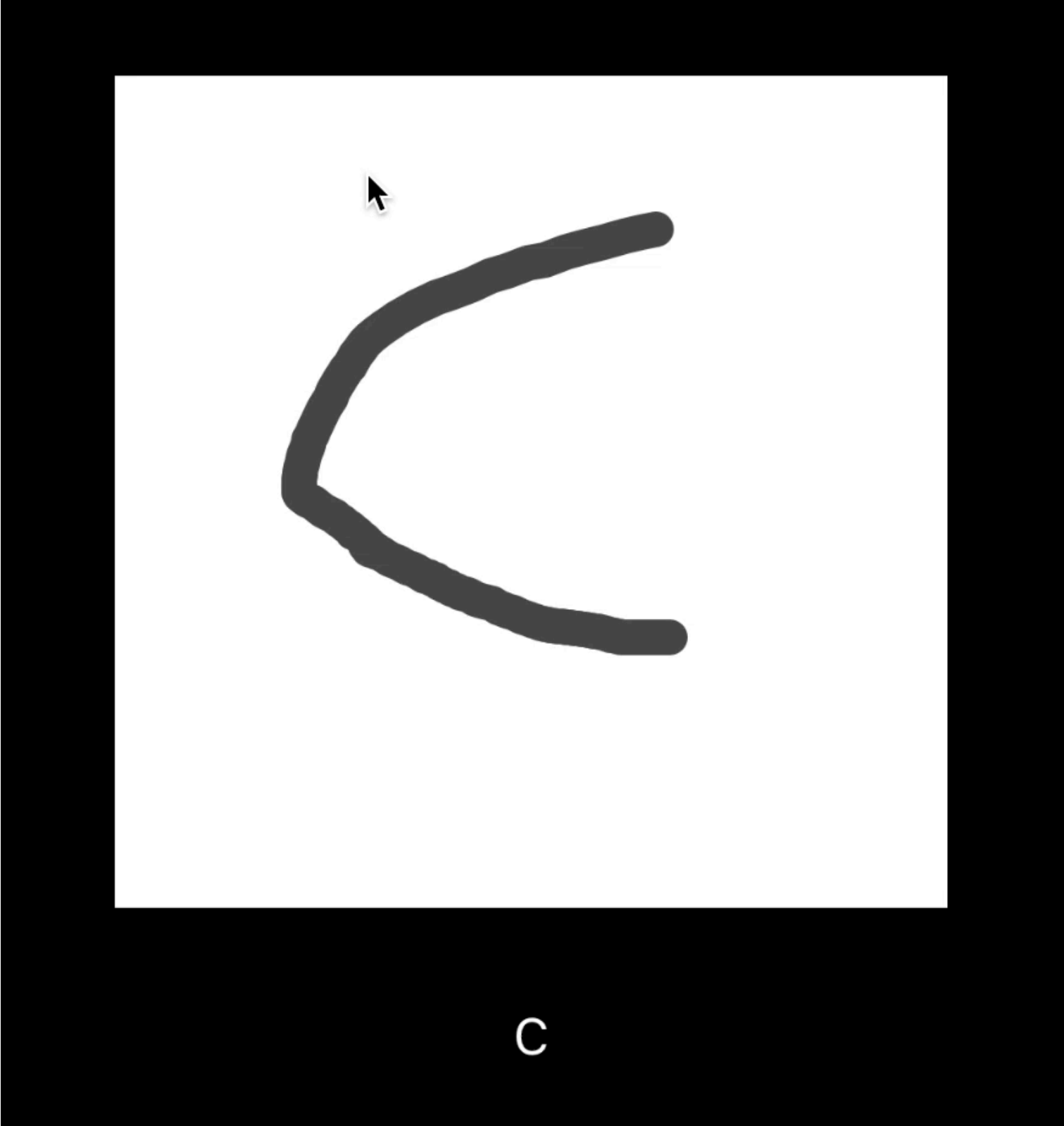
```
clf.fit(images, labels)
```

Accuracy: 69%

Size: 50 MB

Too big for Apple Watch 🥲

Poor user experience



Model 2

SVM, Split Alphabet

Model 2

```
half_letters = [1, 2, 3, 6, 7, 9, 10, 13, 16, 19, 21, 23, 24]
```

```
ind = [val in half_letters for val in labels]
```

```
labels_2=labels[ind]
```

```
images_2=images[ind][:]
```

```
from sklearn import svm
```

```
clf = svm.SVC()
```

```
clf.fit(images_2, labels_2)
```

Accuracy: 78%

Size: 8 MB, x2

Works on Apple Watch!



Accuracy could be better

Rule of thumb:

If you can achieve ~80% accuracy with SVM, then the information you're trying to get at is in your data.

Model 3

Dimensionality Reduction with PCA

PCA =



Principal Component
Analysis

Reduce the number of variables the
model needs to consider

Convert 784 variables into
25 variables

784 Variables

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0.96	0.86	0.82	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.96	0.72	0	0	0	0.74	0.71	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.73	0	0	0	0	0	0.69	0.78	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.78	0	0	0	0	0	0	0.62	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.87	0	0	0	0	0	0	0.26	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.98	0	0	0	0	0	0	0.97	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.81	0	0	0	0	0	0	0.89	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.82	0	0	0	0	0	0	0.82	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.78	0	0	0	0	0	0	0.28	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.24	0.75	0	0	0	0.62	0.72	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.26	0.67	0.56	0.76	0.92	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.35	0.63	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.62	0.56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.65	0.46	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.72	0.76	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.8	0.65	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.72	0.25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.7	0.37	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.58	0.45	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.62	0.25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.6	0.56	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.5	0.26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0.3	0.62	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25 Variables

0.75
0.7
0.65
0.72
0.82
0.62
0.98
0.74
0.77
0.65
0.55
0.35
0.79
0.76
0.81
0.65
0.71
0.95
0.84
0.56
0.15
0.24
0.68
0.57
0.42

PCA

```
from sklearn.decomposition import PCA  
pca = PCA(n_components=components);  
pca.fit(images)
```

```
# PCA matrix...Save this for later  
mat = pca.components_
```

Model 3

```
# Saved from previous step
```

```
mat = pca.components_
```

```
import numpy as np
```

```
images_pca = np.matmul(images, mat.transpose())
```

```
from sklearn import svm
```

```
clf = svm.SVC()
```

```
clf.fit(images_pca, labels)
```

Using PCA in watchOS

```
import Accelerate
```

```
// Input: 784
```

```
// Output: 25
```

```
func transform(from input: [NSNumber]) -> [NSNumber] {
```

```
    let image = input.map { $0.floatValue }           // 784
```

```
    let mat = pcaMatrix                               // Saved from Python
```

```
    var result = [Float](repeating: 0.0, count: 25)    // 25
```

```
    vDSP_mmul(image, 1, mat, 1, &result, 1, 1, 25, 784)
```

```
    return result.map { NSNumber(value: $0) }
```

```
}
```

Accuracy: 87%



Size: 5MB

Ok, now we're 🔥

Model 4

Convolutional Neural Network

Tutorials for Keras and Tensorflow

- Simple Convolutional Neural Network for MNIST³
- Deep Learning in Python⁴

³ <http://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras/>

⁴ <https://elitedatascience.com/keras-tutorial-deep-learning-in-python>

Model 4

```
# Build the model
```

```
model = baseline_model()
```

```
# Fit the model
```

```
model.fit(X_train, y_train,  
          validation_split=.1,  
          epochs=10,  
          batch_size=200,  
          verbose=True)
```

Go get some

Train on 112320 samples, validate on 12480 samples

Epoch 1/10

31000/112320 [=====>.....] - ETA: 53s - loss: 1.3668 - acc: 0.5894

Accuracy: 95%



Size: 245 KB

3:52



Start Drawing

Exporting Model to CoreML

coremltools

```
import coremltools
coreml_model = coremltools.converters.keras.convert(model, input_names = ['imageAlpha'], output_names = ['letterConfidence'])

coreml_model.author = 'Kate Bonnen and Conrad Stoll'
coreml_model.license = 'MIT'
coreml_model.short_description = "Recognize the hand-drawn letter from an input image."

coreml_model.input_description['imageAlpha'] = 'The input image alpha values, from top down, left to right.'
coreml_model.output_description['letterConfidence'] = 'Confidence for each letter ranging from index 1 to 26. Ignore index 0.'

coreml_model.save('letters_keras.mlmodel')
```

Importing a CoreML Model

    Snowman >  Machine Learning >  Models >  letters_keras.mlmodel

▼ Machine Learning Model

Name	letters_keras
Type	Neural Network
Size	245 KB
Author	Kate Bonnen and Conrad Stoll
Description	Recognize the hand-drawn letter from an input image.
License	MIT

▼ Model Class

 letters_keras ➔
Swift generated interface for model

▼ Model Evaluation Parameters

Name	Type	Description
▼ inputs		
imageAlpha	MultiArray<Double,1,28,2...	The input image alpha values, from top down, left to right.
▼ outputs		
letterConfidence	MultiArray<Double,27>	Confidence for each letter ranging from index 1 to 26. Ignore index 0.

Code Generation for your Model

// Generated by CoreML

```
class letters_keras {  
    var model: MLModel  
  
    convenience init() {  
        let bundle = Bundle(for: letters_keras.self)  
        let assetPath = bundle.url(forResource: "letters_keras", withExtension: "mlmodelc")  
        try! self.init(contentsOf: assetPath!)  
    }  
  
    func prediction(imageAlpha: MLMultiArray) throws -> letters_kerasOutput {  
        let input_ = letters_kerasInput(imageAlpha: imageAlpha)  
        return try self.prediction(input: input_)  
    }  
}
```

Building the Game

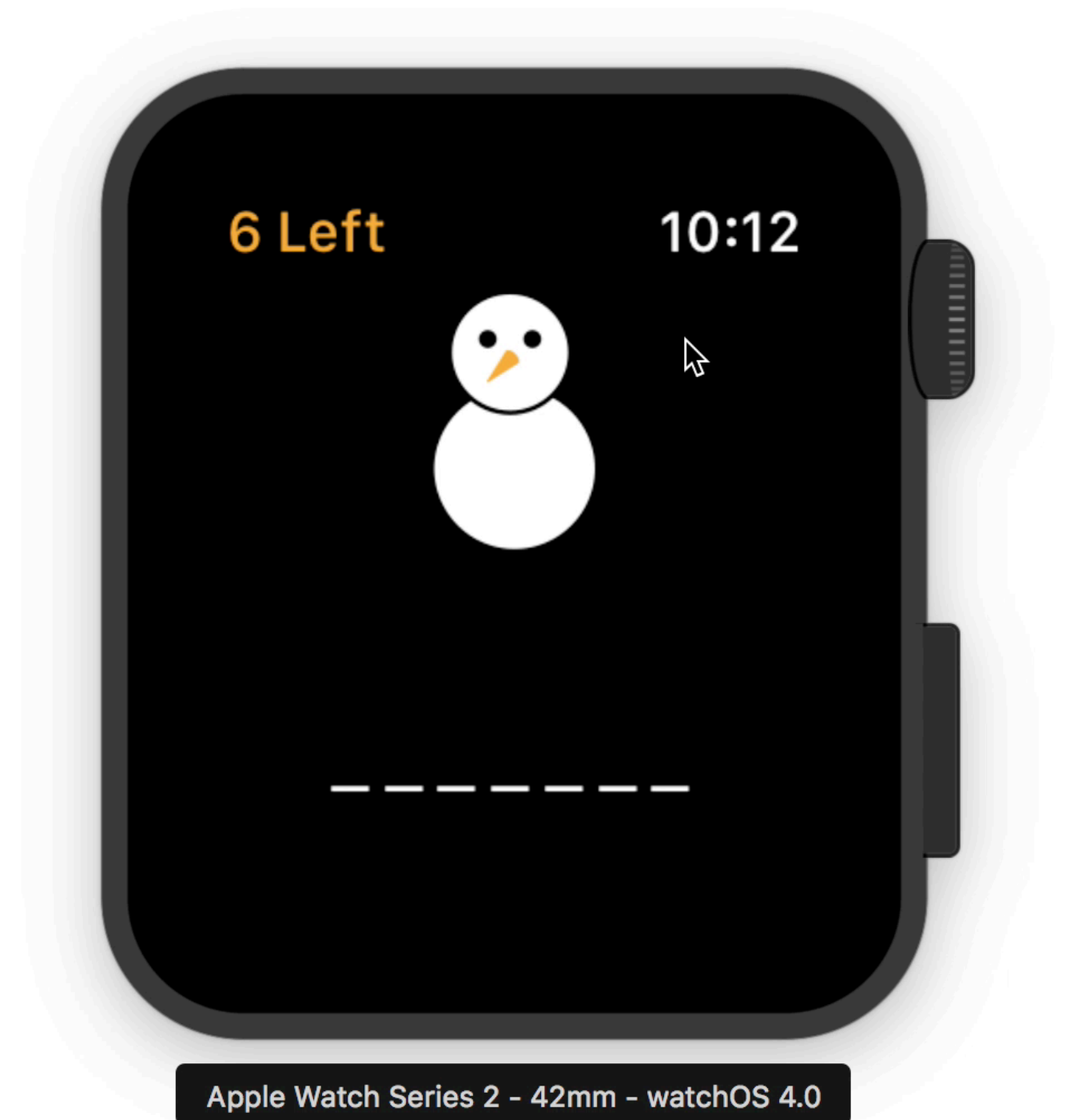
Snowman



Drawing a Letter
Anywhere

Overlapping Interface Layers

- Game Interface
- SpriteKit Drawing Path
- Pan Gesture
- Confirmation Interface



Pan Gesture Captures Path

```
@IBAction func didPan(sender : WKPanGestureRecognizer) {  
    let location = sender.locationInObject()  
  
    updateRecognitionLine(for: location, currentRecognizer: currentRecognizer)  
  
    if sender.state == .ended {  
        addSegmentAndWaitForConfirmation(with: currentRecognizer)  
    }  
}
```

Setting up a Path Shape Node

```
let line = SKShapeNode()
line.fillColor = SKColor.clear
line.isAntialiased = false
line.lineWidth = strokeWidth
line.lineCap = .round
line.strokeColor = UIColor.white
lineNode = line

let scene = SKScene(size: size)
scene.addChild(line)
scene.backgroundColor = UIColor.clear

drawScene.presentScene(scene)
```

Updating the Shape Node's Path

```
func updateRecognitionLine(for location: CGPoint, currentRecognizer: Recognizer) {  
    // Add the point to our path  
    let path = currentRecognizer.addPoint(location)  
  
    // Update the node's path  
    lineNode?.path = path  
}
```

Making a Prediction

Prediction Steps

- Stroke Path to Image
- Center and Crop Image
- Get Pixel Alpha Values Between 0 and 1
- Convert to Vector
- Send to CoreML

Stroke Path to Image

```
UIGraphicsBeginImageContextWithOptions(CGSize(width: drawingWidth,  
                                             height: drawingHeight), false, 0.0)
```

```
let context = UIGraphicsGetCurrentContext()!
```

```
context.setStrokeColor(UIColor.black.cgColor)
```

```
path.lineJoinStyle = .round
```

```
path.lineCapStyle = .round
```

```
path.lineWidth = strokeWidth
```

```
path.stroke(with: .normal, alpha: 1)
```

Compute Center and Crop Image

- Letter centered
- 2px padding on every side
- Square aspect ratio
- Must be 28x28 pixels

Get Image Alpha Values

```
extension UIImage {  
    func getPixelAlphaValue(at point: CGPoint) -> CGFloat {  
        guard let cgImage = cgImage, let pixelData = cgImage.dataProvider?.data else { return 0.0 }  
  
        let data: UnsafePointer<UInt8> = CFDataGetBytePtr(pixelData)  
  
        let bytesPerPixel = cgImage.bitsPerPixel / 8  
  
        let pixelInfo: Int = ((cgImage.bytesPerRow * Int(point.y)) + (Int(point.x) * bytesPerPixel))  
  
        // We don't need to know about color for this  
        //         let b = CGFloat(data[pixelInfo]) / CGFloat(255.0)  
        //         let g = CGFloat(data[pixelInfo+1]) / CGFloat(255.0)  
        //         let r = CGFloat(data[pixelInfo+2]) / CGFloat(255.0)  
  
        // All we need is the alpha values  
        let a = CGFloat(data[pixelInfo+3]) / CGFloat(255.0)  
  
        return a  
    }  
}
```

Matching Training and Input Data Structure

```
let a = CGFloat(data[pixelInfo+3]) / CGFloat(255.0)
```

a is between 0 and 1

not between 0 and 255

The order of the points in your image
also matter!

Get Every Pixel's Alpha Value

```
extension UIImage {  
    func pixelAlpha() -> [NSNumber] {  
        var pixels = [NSNumber]()  
  
        for w in 0...Int(self.size.width) - 1 {  
            for h in 0...Int(self.size.height) - 1 {  
                let point = CGPoint(x: w, y: h)  
                let alpha = getPixelAlphaValue(at: point)  
                let number = NSNumber(value: Float(alpha))  
  
                pixels.append(number)  
            }  
        }  
  
        return pixels  
    }  
}
```

Convert to MLMultiArray

```
import CoreML

let alphaValues = drawing.generateImageVectorForAlphaChannel()

let multiArray = try! MLMultiArray(shape: [1,28,28], dataType: MLMultiArrayDataType.double)

for (index, number) in alphaValues.enumerated() {
    multiArray[index] = number
}
```

Here's what an
MLMultiArray looks like



Make a Prediction with CoreML

```
import CoreML
```

```
let model = letters_keras()
```

```
let prediction = try! model.prediction(imageAlpha: multiArray)
```

1 Demo

Thank You

Conrad Stoll

@conradstoll

<https://github.com/cnstoll/Snowman>

<https://speakerdeck.com/cnstoll/machine-learning-on-apple-watch>