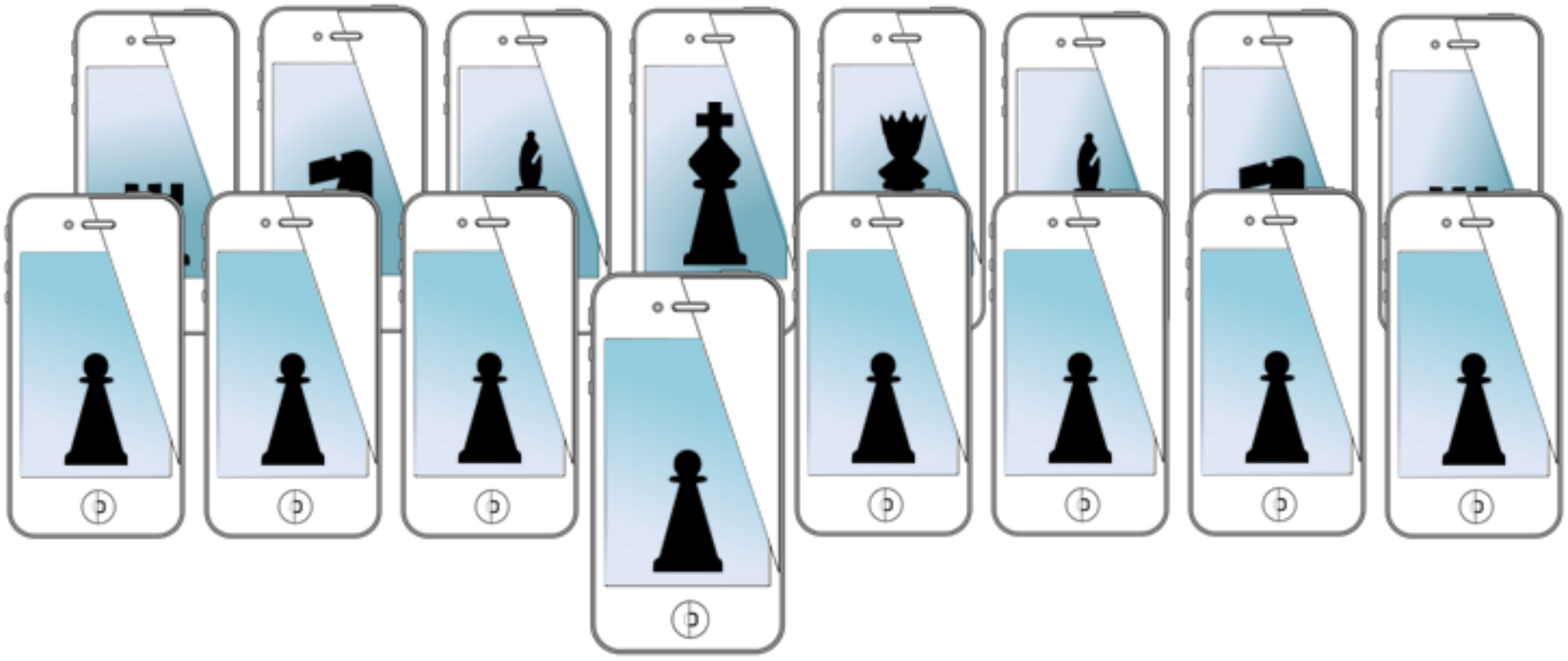
MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

Speech and Sound Classification

Eric C. Larson, Lyle School of Engineering,
Computer Science, Southern Methodist University

course logistics

- Speech and Sound Classification

speech overview

- introduced in 2016, same technology underpinning Siri
  - user must provide explicit authorization
  - free, but limited to certain number of recognitions per day
  - only allowed to dictate about 1 minute of audio (siri)
  - supports streaming audio and file I/O

general usage

- uses API similar to REST (like Core Vision)
  - create a task
  - configure it (options)
  - start task
    - use completion handler (for updates and final text)
- best practices:
  - signify to user that the app is recording
  - show dictation as it happens

tradeoff server/on-device

- 2016: speech is translated via cloud services
- 2019: also available on device

	Server	On-device
Accuracy	Best	Good
Limits	1 minute max audio duration Limited requests per day	None
Languages	50+	10+

using SFSpeechRecognizer

- callback model
- needs AVFoundation for adding audio chunks

```
private let speechRecogniser = SFSpeechRecognizer(locale: Locale(identifier: “en-US”))!
private var recognitionRequest: SFSpeechAudioBufferRecognitionRequest?
private var recognitionTask: SFSpeechRecognitionTask?
private let audioEngine = AVAudioEngine()

let audioSession = AVAudioSession.sharedInstance()
audioSession.setCategory(AVAudioSession.Category.record)
audioSession.setMode(AVAudioSession.Mode.measurement)
audioSession.setActive(true, options: .notifyOthersOnDeactivation)

let inputNode = audioEngine.inputNode
let recordingFormat = inputNode.outputFormat(forBus: 0)

inputNode.installTap(onBus: 0, bufferSize: 1024, format: recordingFormat)
{ (buffer: AVAudioPCMBuffer, when: AVAudioTime) in
            self.recognitionRequest?.append(buffer)
}

audioEngine.prepare()
audioEngine.start()
```

much of the code also has **guards** for **error checking**

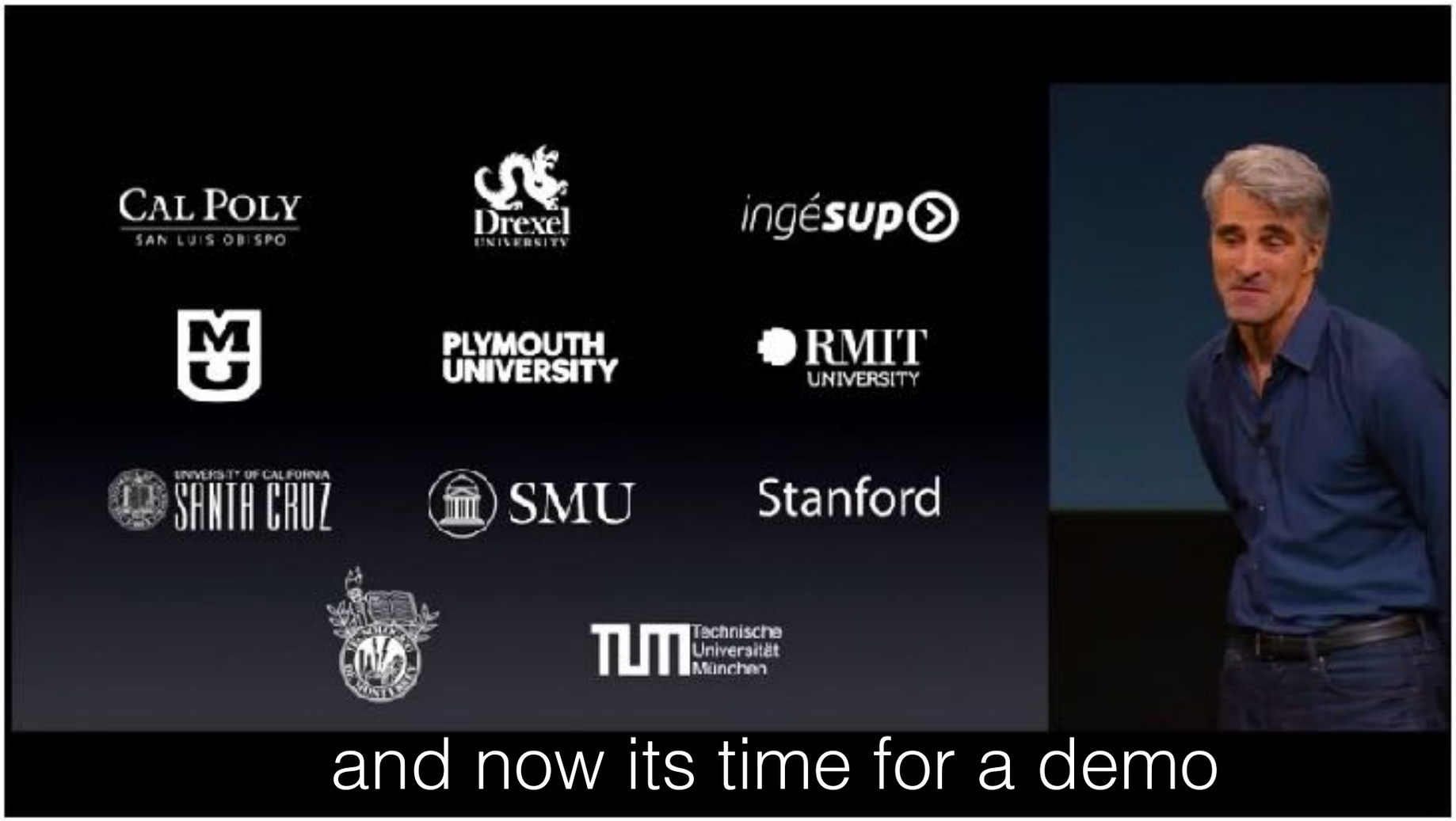
using SFSpeechRecognizer

```
let inputNode = audioEngine.inputNode
let recordingFormat = inputNode.outputFormat(forBus: 0)

inputNode.installTap(onBus: 0, bufferSize: 1024, format: recordingFormat)
{ (buffer: AVAudioPCMBuffer, when: AVAudioTime) in
            self.recognitionRequest?.append(buffer)
}

 // perform on device, if possible
 if speechRecogniser.supportsOnDeviceRecognition {
      recognitionRequest?.requiresOnDeviceRecognition = true
 }
```

```
recognitionTask = speechRecogniser.recognitionTask(with: recognitionRequest)
{ [unowned self] result, error in
        if let result = result {
           let transcribedText = result.bestTranscription.formattedString
            // do something with text
           }
        }

        if result?.isFinal ?? (error != nil) {
            // this will remove the listening tap
            // so that the transcription stops
            inputNode.removeTap(onBus: 0)
        }
    }
```

SFSpeechRecognizer

- adding audio blocks from input buffer



more advanced speech processing

- recognition result contains many aspects of the voice, including:
  - Transcribed text (as we have seen)
  - Alternate transcriptions
  - Confidence in result
  - Timing
  - Speaking rate
  - Pause duration
  - Voice analytics

more advanced speech processing

```
bestTranscription=<SFTranscription>,
formattedString=I helped Apple recognize speech,

speakingRate=0.000000, averagePauseDuration=0.000000,

segments=(
    <SFTranscriptionSegment>, substringRange={0, 1}, timestamp=0.54, duration=0.24,
    confidence=0.966,
    substring=I, alternativeSubstrings=(\n),
    phoneSequence=AY,
    ipaPhoneSequence=\U02c8a\U0361\U026a, voiceAnalytics=(null),

    <SFTranscriptionSegment>, substringRange={2, 6}, timestamp=0.78,
    duration=0.360000000000001, confidence=0.966,
    substring=helped, alternativeSubstrings=(\n),
    phoneSequence=h EH l p t,
    ipaPhoneSequence=h.\U02c8\U025b.l.p.t,
    voiceAnalytics=(null),

     …

    <SFTranscriptionSegment>, substringRange={25, 31}, timestamp=2.49,
    duration=0.569999999999998, confidence=0.966,
    substring=speech, alternativeSubstrings=(\n),
    phoneSequence=s p EE ch,
    ipaPhoneSequence=s.p.\U02c8i.t\U0361\U0283, voiceAnalytics=(null)
),
```

running on device will limit the available features!

more advanced speech processing

Each segment now has incredible amount of information

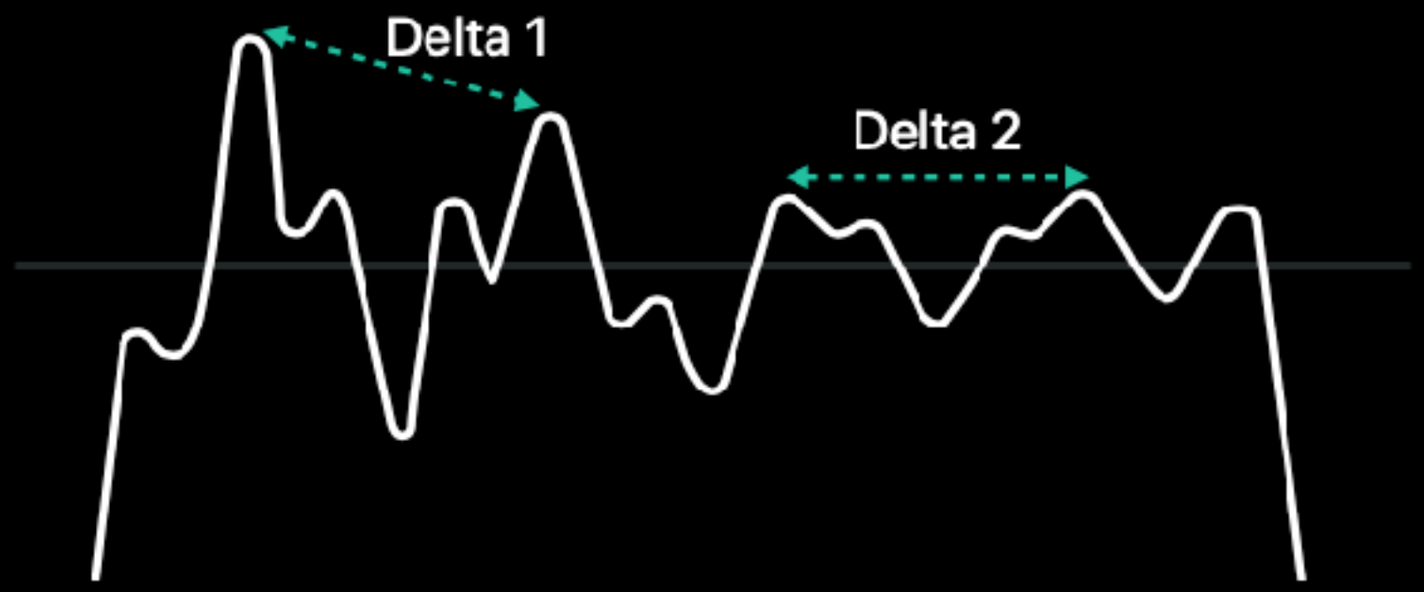
```
<SFTranscriptionSegment>,
substringRange={0, 10}, timestamp=0.27, duration=0.65, confidence=0.911,
substring=Performing,
alternativeSubstrings=(\n),
phoneSequence=(null),
ipaPhoneSequence=(null),

voiceAnalytics=<SFVoiceAnalytics>,
jitter=<SFAcousticFeature>, featureValues=(12.53122 … 0.6218916),
frameDuration=0.010000,
shimmer=<SFAcousticFeature>, featureValues=(0.7158176 … 2.518468),
frameDuration=0.010000,
pitch=<SFAcousticFeature>, featureValues=(0.8526305,   …. 0.04258926),
frameDuration=0.010000,
voicing=<SFAcousticFeature>, featureValues=(0.07444749 …   0.4056852),
frameDuration=0.010000",
```

each featureValues array is length of audio frames, could be **hundreds** of values

analytics

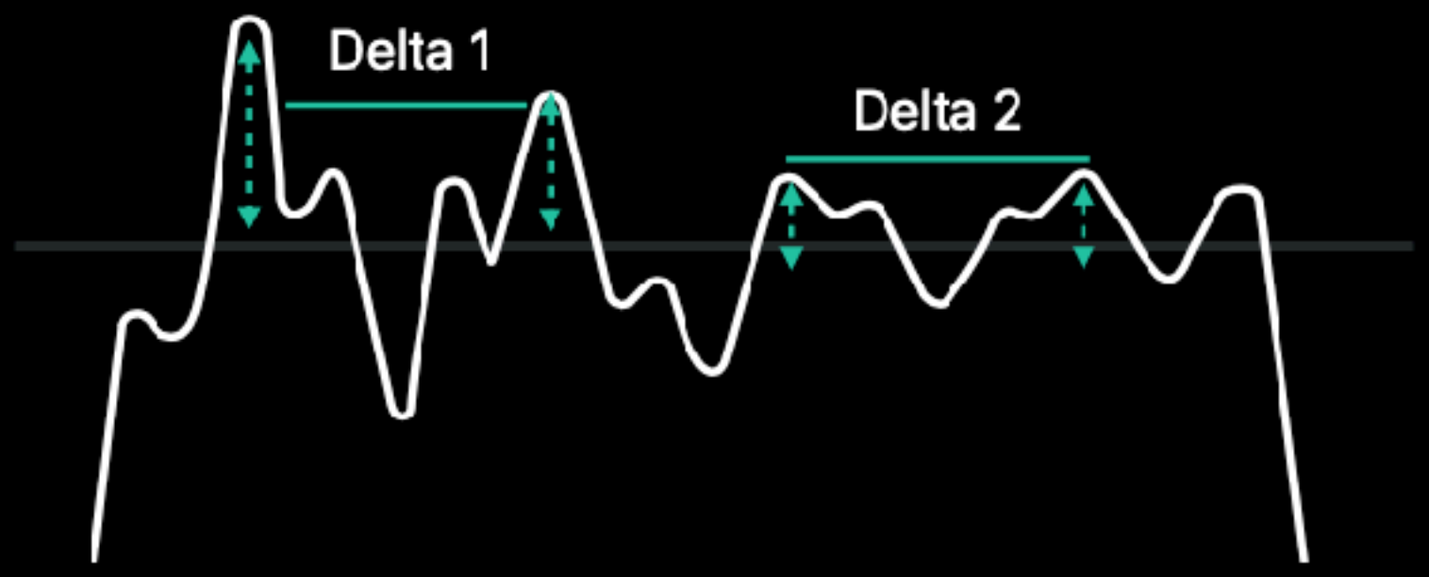
Jitter

Measures variation in pitch

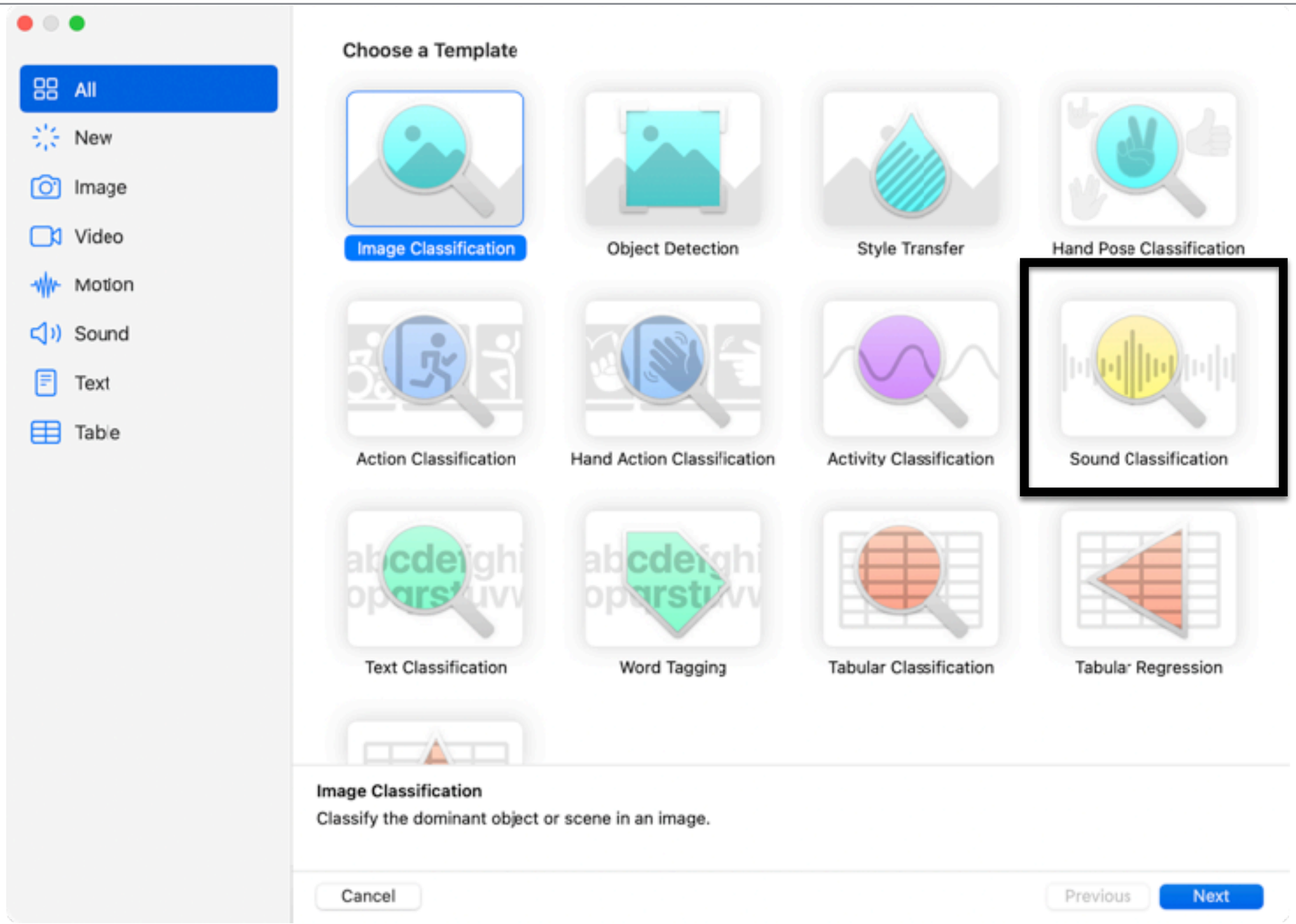


Shimmer

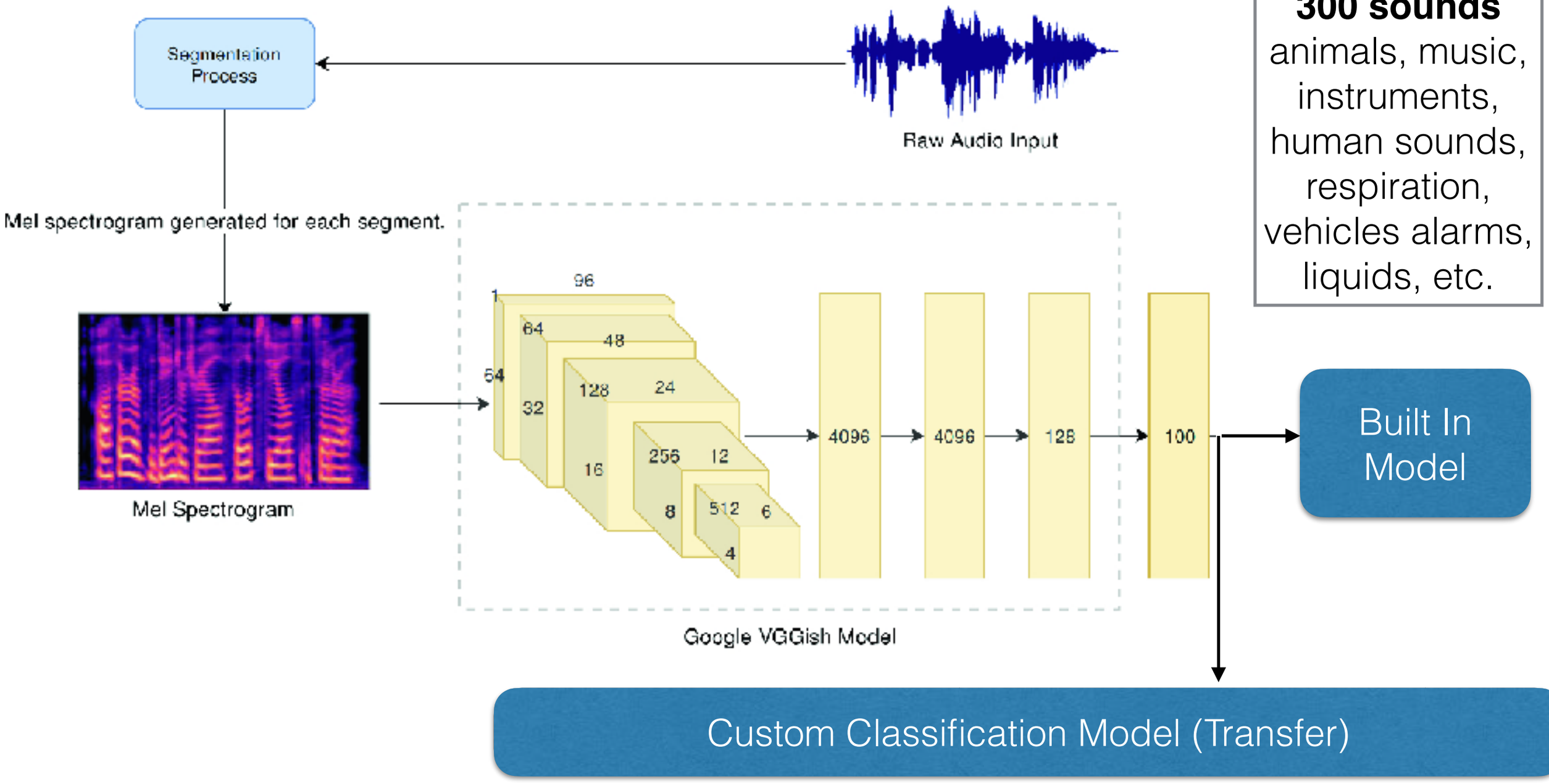
Measures variation in amplitude



Bonus: Create ML (iOS 15.0+)

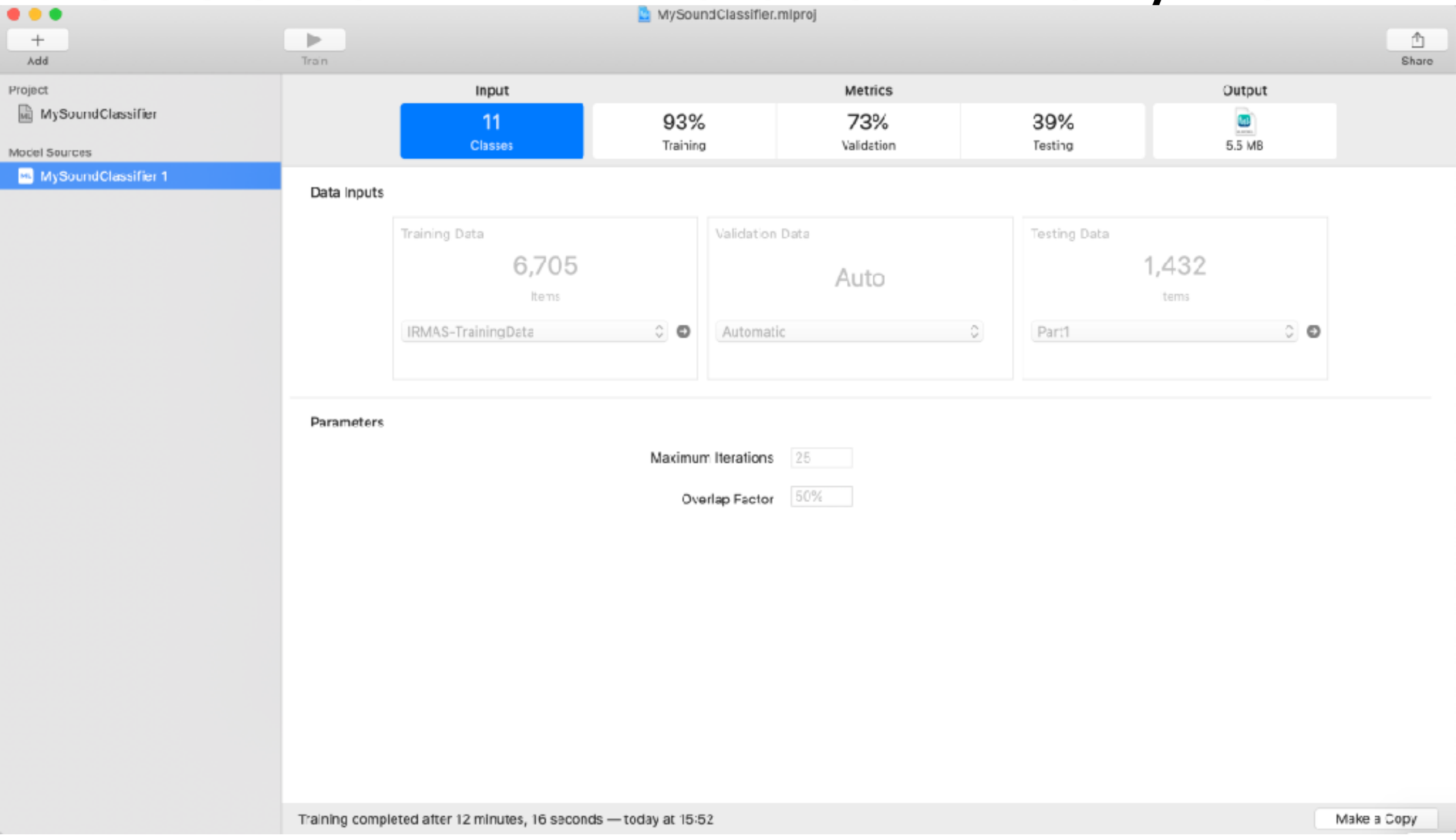


Create ML Audio Analyzer



300 sounds
animals, music, instruments, human sounds, respiration, vehicles alarms, liquids, etc.

Create ML Audio Analyzer



https://martinmitrevski.com/2019/12/09/sound-classification-with-create-ml-on-ios-13/

Create ML Audio Analyzer

```
let version1 = SNClassifierIdentifier.version1
self.soundRequest = SNClassifySoundRequest(classifierIdentifier: version1)

self.streamAnalyzer = SNAudioStreamAnalyzer(format: inputNode.inputFormat(forBus: 0))
streamAnalyzer?.add(self.soundRequest!, withObserver: self)

func startRecording() {
   let inputNode = audioEngine.inputNode
   let recordingFormat = inputNode.outputFormat(forBus: 0)

    inputNode.installTap(onBus: 0, bufferSize: 1024, format: recordingFormat) {
        (buffer: AVAudioPCMBuffer, when: AVAudioTime) in
           self.streamAnalyzer!.analyze(buffer, atAudioFramePosition: when.sampleTime)
    }

    audioEngine.prepare()
    do{
        try audioEngine.start()
    }
    catch {…}
}


func request(_ request: SNRequest, didProduce result: SNResult) {
    // Downcast the result to a classification result.
    guard let result = result as? SNClassificationResult else  { return }
    guard let classification = result.classifications.first else { return }

    self.classifierOutput.text = "\(classification.identifier).\n"
}
```

setup default classifier

audio stream

add audio to analysis buffer

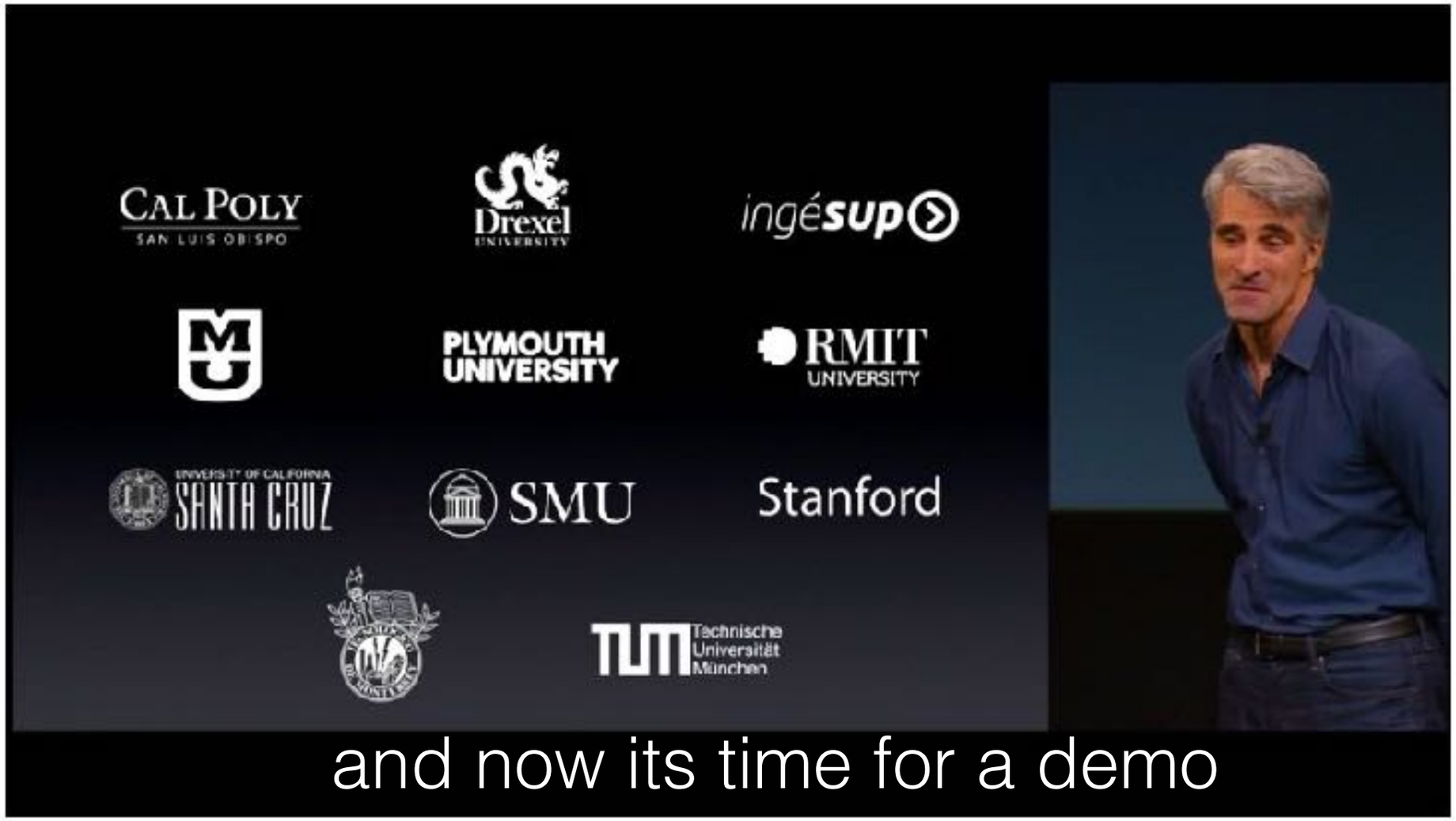
ViewController: SNResultsObserving

delegation handler

https://martinmitrevski.com/2019/12/09/sound-classification-with-create-ml-on-ios-13/

Sound Analysis

- add sound classification to our project

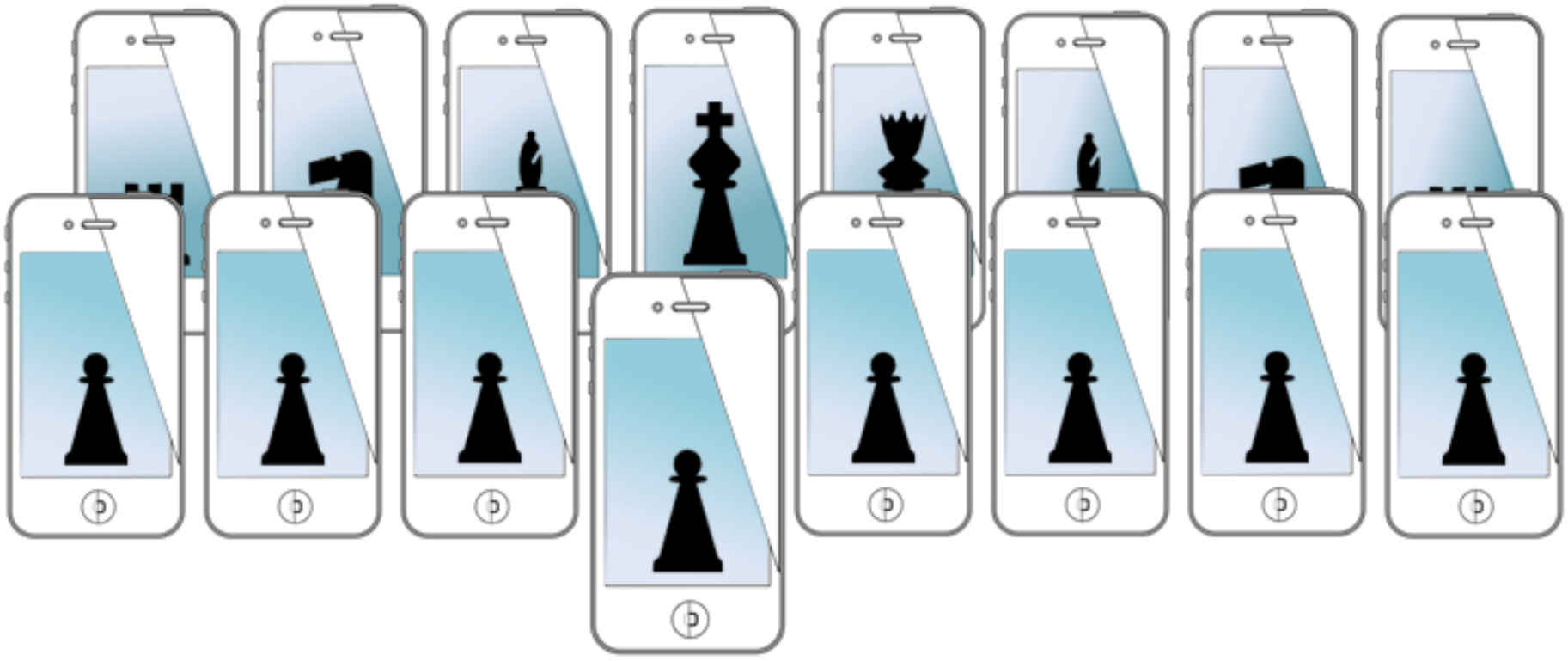


clock 🕰️
bowling_impact 🎳
elk_bugle 🐑
bird_vocalization 🦜🎶
percussion 🥁
door_slam 🚪
foghorn 🌫️📯
horse_neigh 🐴🗣️
applause 👏
writing ✍️
train_horn 🚆📯
fireworks 🎇
person_running 🏃
ukulele 🪕🌺
playing_hockey 🏒
mosquito_buzz 🦟
whoosh_swoosh_swish 💥💨
organ 🪗🎹
typewriter ⌨️
alarm_clock ⏰
chopping_wood 🪵🪓
bird_chirp_tweet 🐦🎵
theremin 🎶
ambulance_siren 🚑🚨
water_pump 💦⛽
cow_moo 🐄🗣️
engine_idling 🚗📣

for next time…

- Pitching
- ~Fin~

MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

Adding Vision Object Detection

Eric C. Larson, Lyle School of Engineering,
Computer Science, Southern Methodist University

Back Up Slides

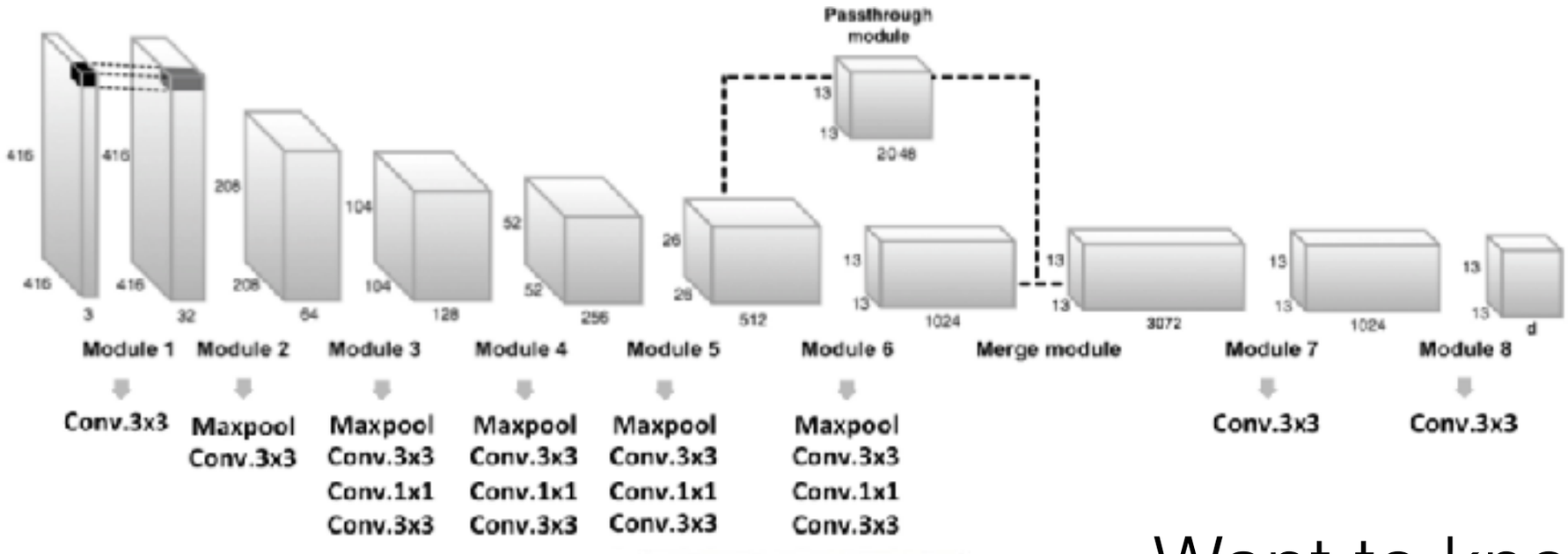
Detecting Objects in Images

- Using Turi Create
  - SFrames
  - Simple Machine Learning
- Object Detection with YOLO
  - Creation
  - Exporting to CoreML
  - Incorporating into ARKit

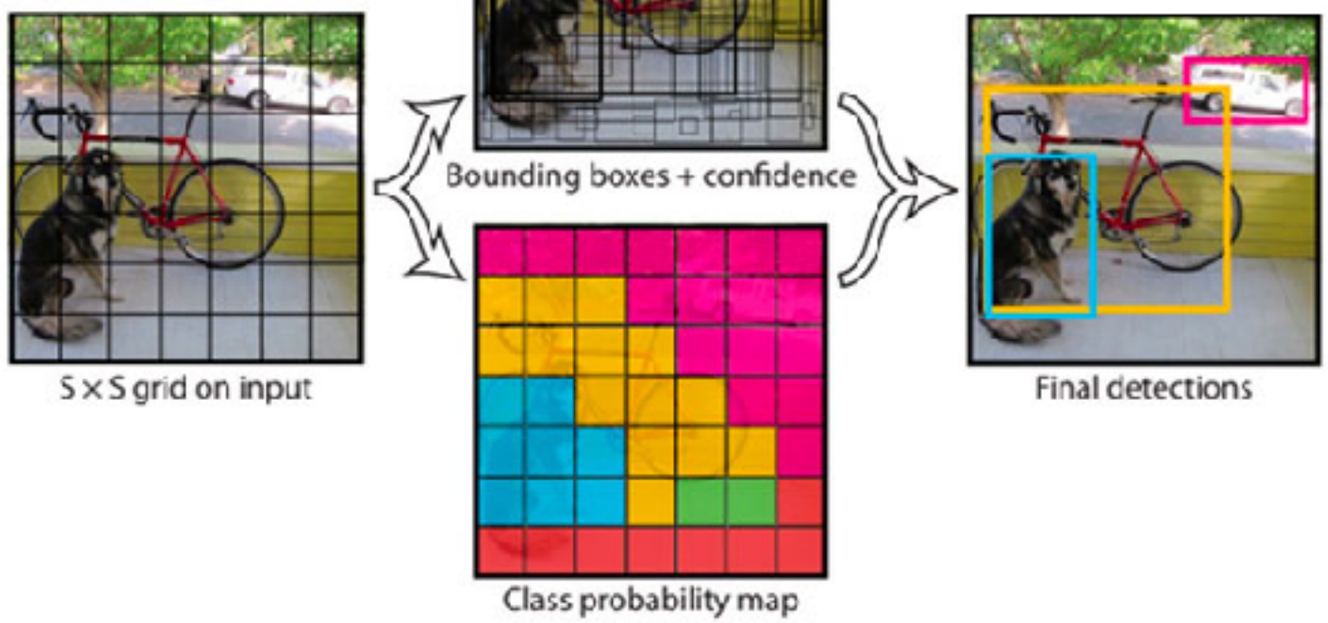
didn’t we already detect objects?

- with ARKit, yes
- but what if we want to detect all sorts of objects
- and we need to do it very fast (multiple times per second)
- and we only have examples of the types of objects we want, not a scan of the exact object
- enter… Turi Create

YOLO



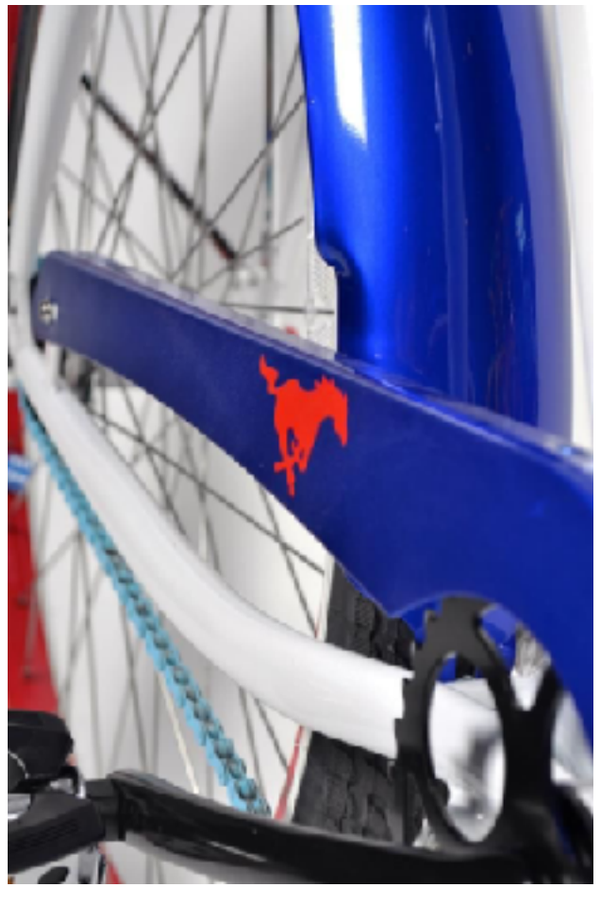
Want to know more?
Take CS8321…



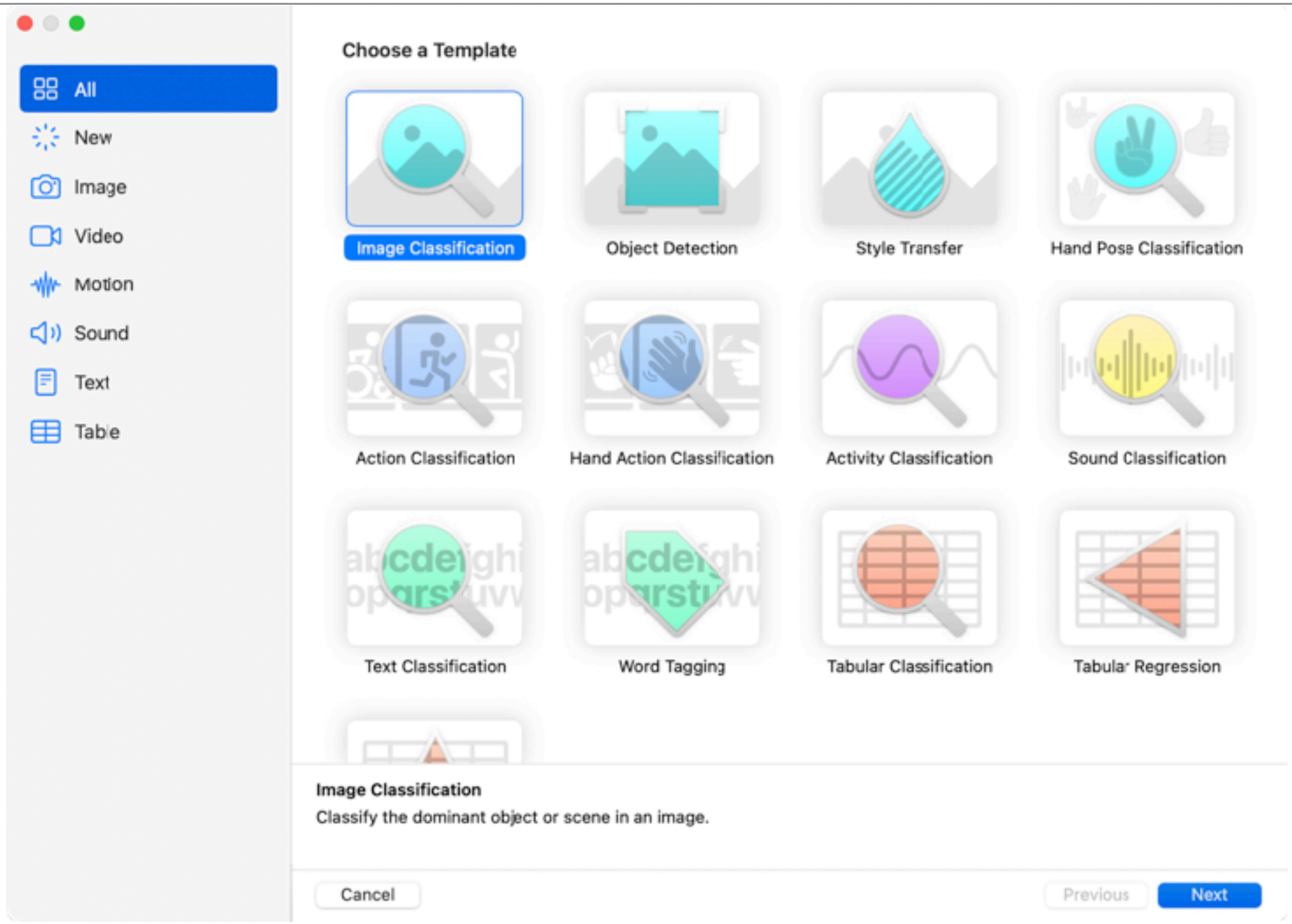
https://datascience.stackexchange.com/questions/42509/yolo-algorithm-understanding-training-data

create an object detector with Turi

- Turi Create, high level overview
  - SFrames and Object Detectors

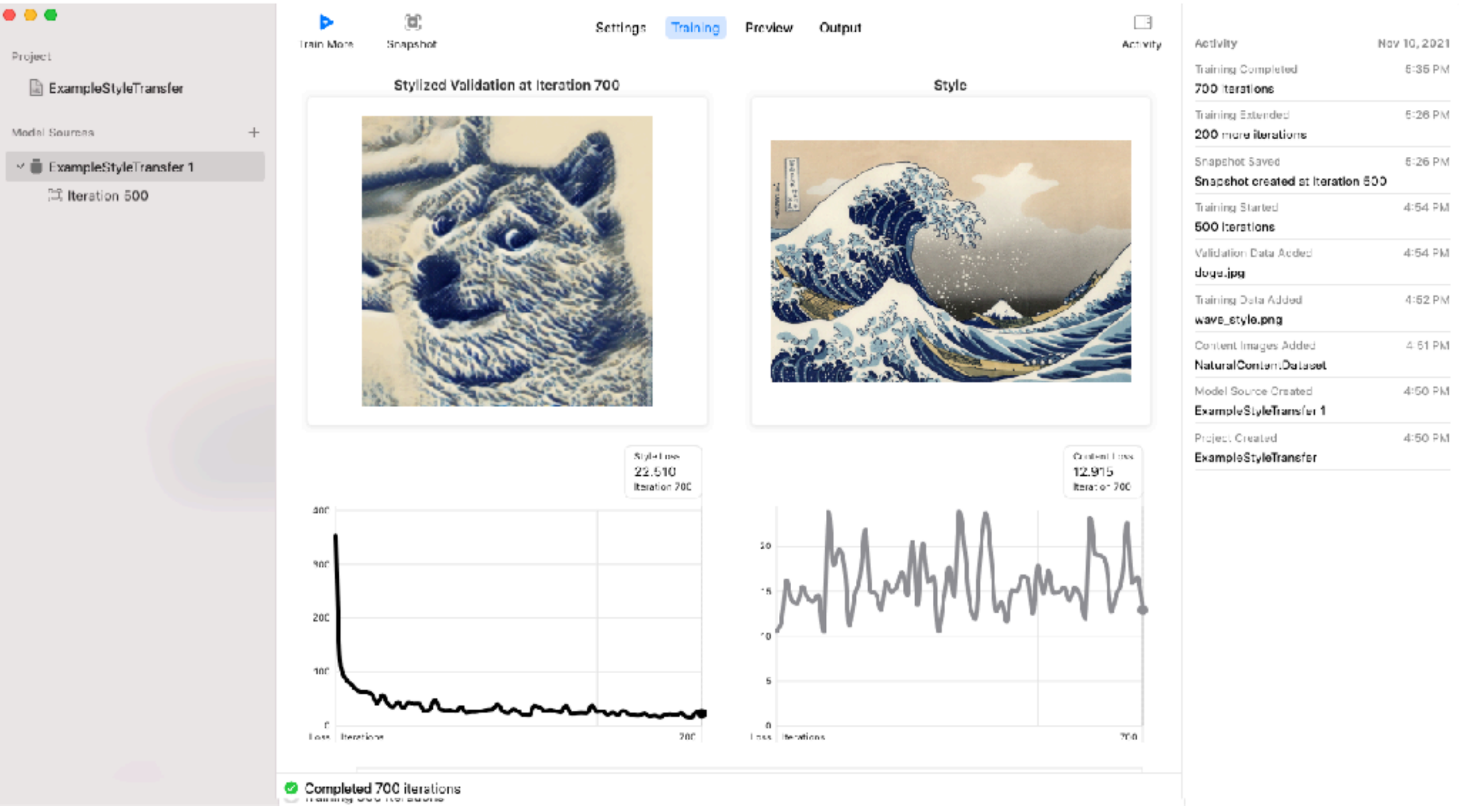


Bonus: Create ML (iOS 15.0+)



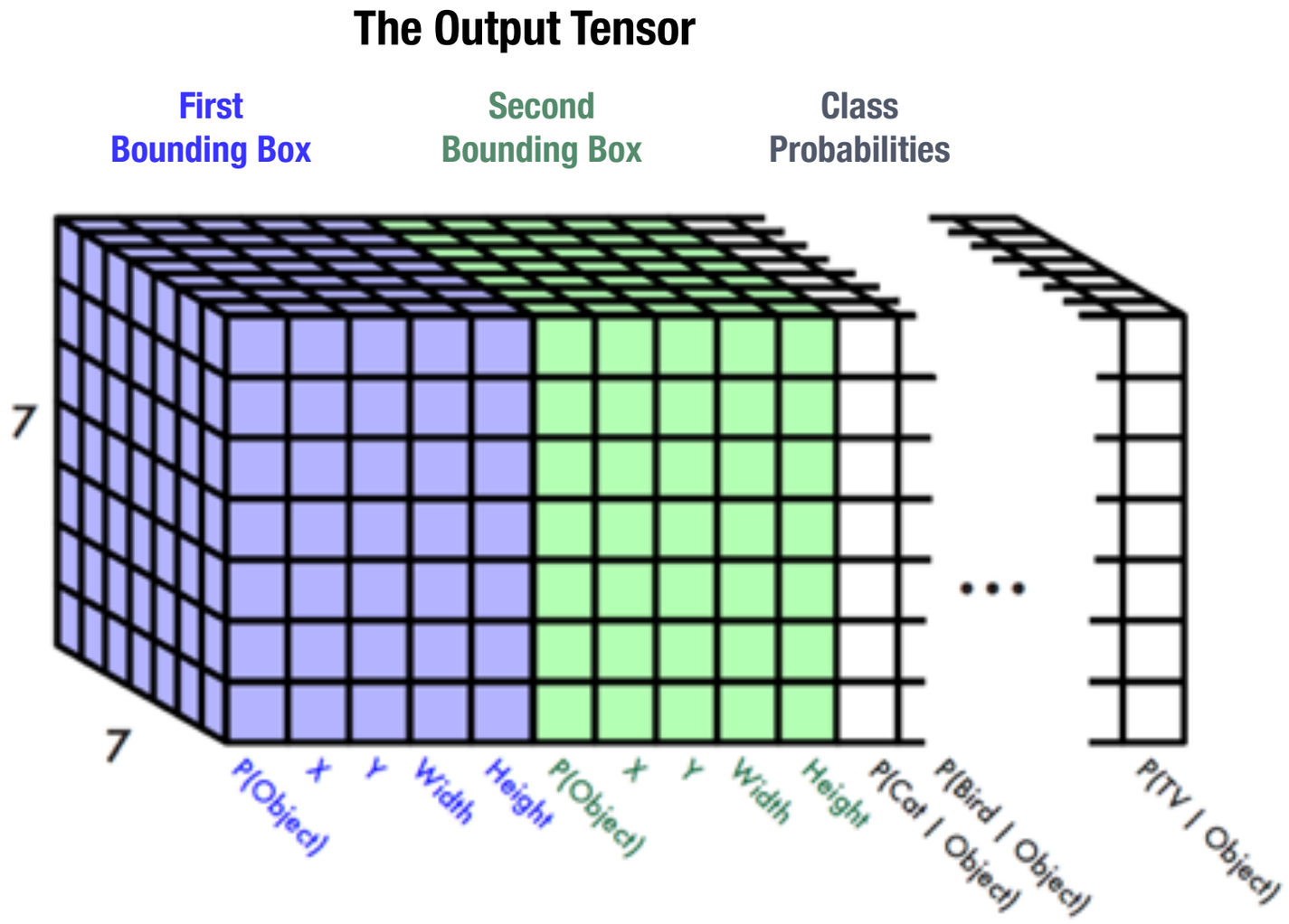
https://developer.apple.com/documentation/createml/

Create ML Stylizer



YOLO output tensor

- is there an object?
- where is the object?
- how large is the Object?
- what is the object?
- if competing, which
- object is most likely?



using the vision API

- Overview:
  - streaming video will require constant images from the AR session, **use delegation**
  - inside delegate, need to **grab** ARFrame, **convert** to pixel buffer, and **process** on a **background** thread
  - **handle vision** results in a new function
  - **update** the UI on the **main thread**

setting up vision

```
let model = PersonBike()
private var requests = [VNRequest]()

@discardableResult
func setupVision(useCPUOnly:Bool) {

  //MARK: One, Setup Vision
  let visionModel = try VNCoreMLModel(for: model.model)

  // use this request to setup the object recognition with Vision
  let objectRecognition = VNCoreMLRequest(model: visionModel,
                                          completionHandler:
                                              self.handleObjectRecognitionResult)

      objectRecognition.imageCropAndScaleOption = .scaleFill
      objectRecognition.usesCPUOnly = [True / False]
      self.requests = [objectRecognition]

  }
```

load model from Turi

setup vision wrapper

save this request for later

using vision with AR

ARDelegate function, called at 60 FPS

```
func renderer(_ renderer: SCNSceneRenderer, updateAtTime time: TimeInterval) {
        //MARK: Two, Get Image Frame from AR
       guard let frame = sceneView.session.currentFrame else { return }

       guard self.currentBuffer == nil, else { return }
      self.currentBuffer = frame.capturedImage // the pixels to process

       // run in the background so that AR doesn't suffer performance
       DispatchQueue.global(qos: .background).async {

           // setup input image for the request
           let imageRequestHandler = VNImageRequestHandler(
                                     cvPixelBuffer: self.currentBuffer,
                                       orientation: ORIENTATION,
                                           options: [:])

           imageRequestHandler.perform(self.requests)

       }
   }
```

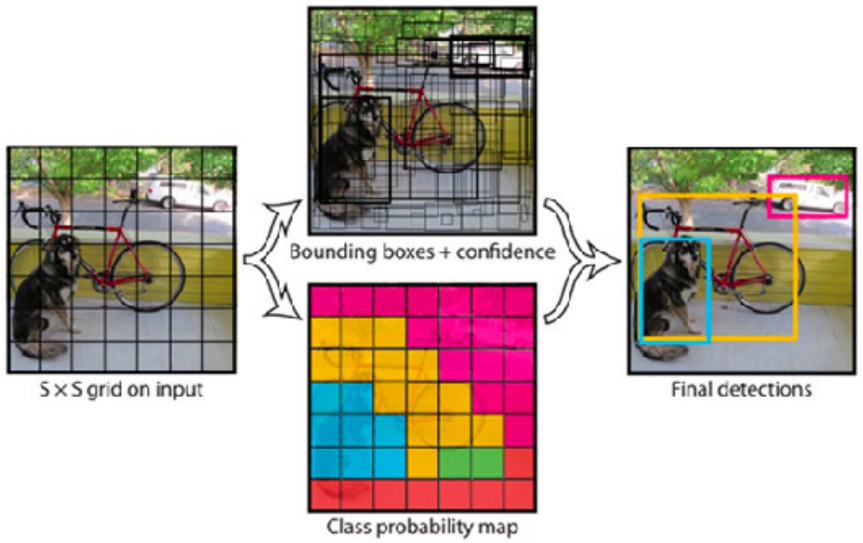
limit FPS of detection

setup image buffer

tell request what pixels to process

start processing request that we setup previously

handling output request



YOLO Output:

- list of bounding boxes for every square in grid
- list of all possible classes (bike, person)

can have multiple objects in the scene

```
func handleObjectRecognitionResult(_ request:VNRequest, error:Error?){
        // perform all the UI updates on the main queue
        if let results = request.results { // if we have valid results, else its nil
            DispatchQueue.main.async(execute: {

                self.drawVisionRequestResults(results)
                self.updateOverlay()

                // set as nil so we can process next ARFrame Image
                self.currentBuffer = nil
            })
        }
    }
```

go over, if time!

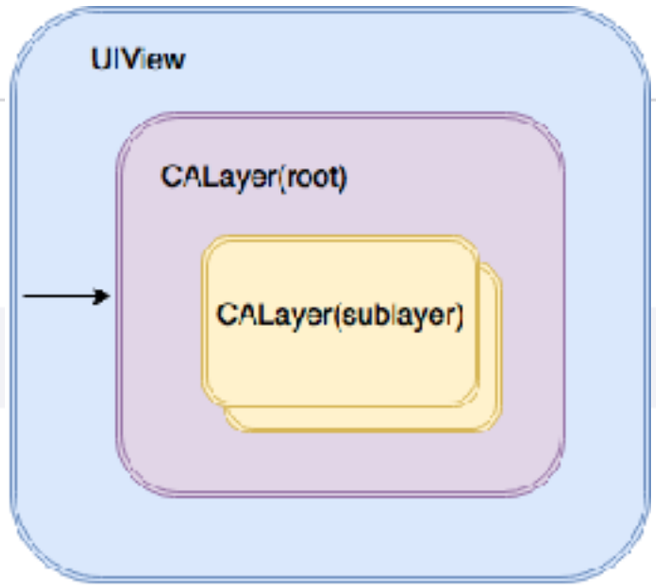
process the next captured frame now

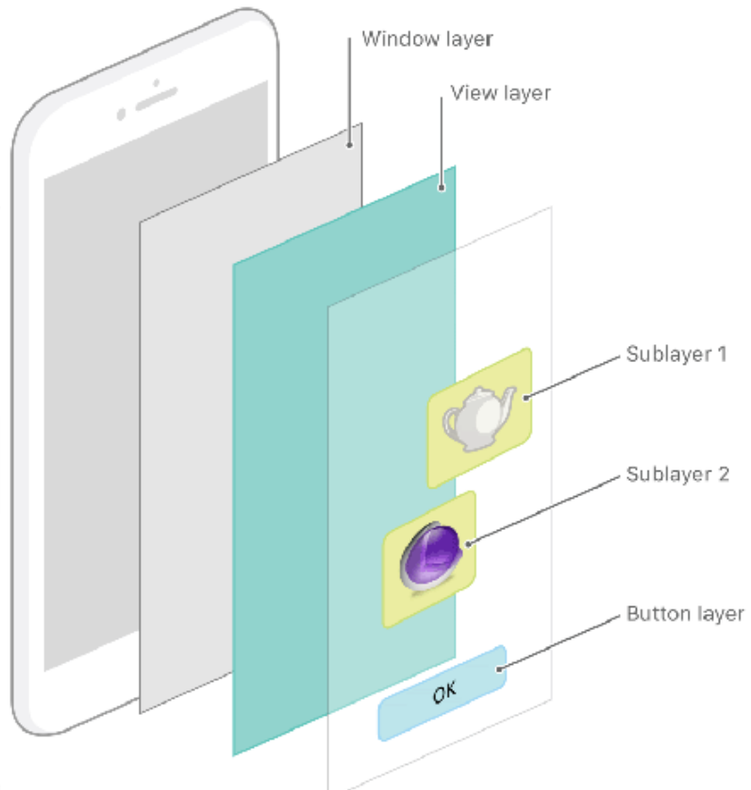
object detection

- YOLO demo
- If time, go over creating overlays with CATransactions

CATransactions

A mechanism for grouping multiple layer-tree operations into atomic updates to the render tree.





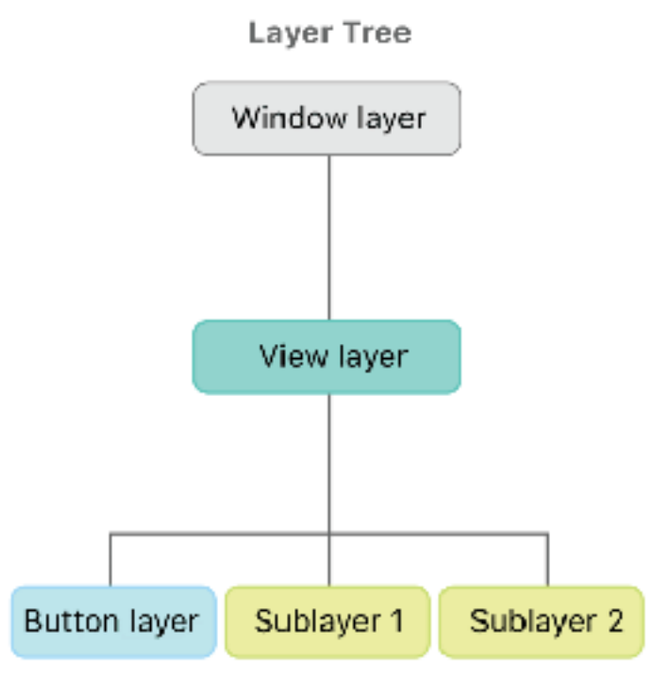
Declaration

```
class CATransaction : NSObject
```

Overview

`CATransaction` is the Core Animation mechanism for batching multiple layer-tree operations into atomic updates to the render tree. Every modification to a layer tree must be part of a transaction. Nested transactions are supported.

Core Animation supports two types of transactions: *implicit* transactions and *explicit* transactions. Implicit transactions are created automatically when the layer tree is modified by a thread without an active transaction and are committed automatically when the thread's runloop next iterates. Explicit transactions occur when the the application sends the `CATransaction` class a `begin()` message before modifying the layer tree, and a `commit()` message afterwards.

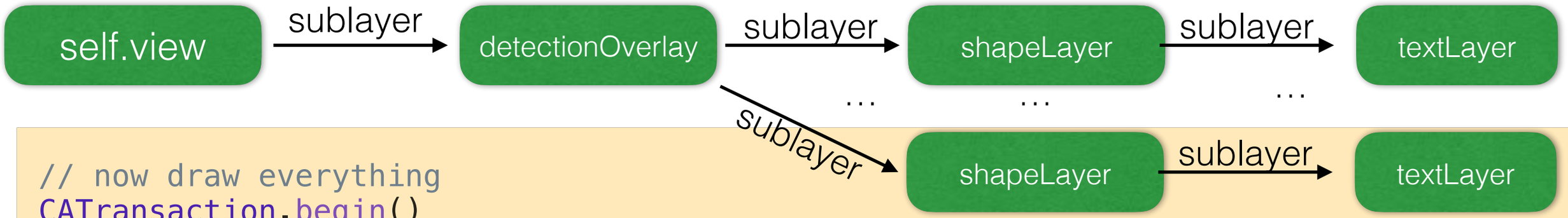


Overlays from YOLO

```
detectionOverlay = CALayer() // container layer that has all the renderings
detectionOverlay.name = "DetectionOverlay"

// set the initial bounds, will transform when we know more about the image
detectionOverlay.bounds = CGRect(x: 0.0, y: 0.0,
                                 width: self.view.bounds.width,
                                height: self.view.bounds.height )

self.sceneView.layer.addSublayer(detectionOverlay)
```

```
func updateOverlay() {

   let bounds = self.view.bounds

   // … Some magic transforms to the view  …
   // this tries to get the best mapping we can from the cropping that
   // Core Vision used, but It may not be 100% perfect
   // Scales, and adds magic numbers to the X and Y positions

   detectionOverlay.setNeedsDisplay() // sets display for all subviews
}
```

Overlays from YOLO




```
// now draw everything
CATransaction.begin()
detectionOverlay.sublayers = nil // remove all the old recognized objects

// for each result, create an overlay layer to display on it
for observation in results where observation is VNRecognizedObjectObservation {

   // Select the label with the highest confidence and get bounds
   let topLabelObservation = objectObservation.labels[0]
   let objectBounds = VNImageRectForNormalizedRect(objectObservation.boundingBox,
                        Int(captureImageSize.width),Int(captureImageSize.height))

   // get UI box for object
   let shapeLayer = self.createRoundedRectLayerWithBounds(objectBounds, {…})

   // show the label and confidence in the UI Box
   let textLayer = self.createTextSubLayerInBounds(objectBounds,  {…})

   shapeLayer.addSublayer(textLayer)// add text to box
   detectionOverlay.addSublayer(shapeLayer) // add box to the UI
}
CATransaction.commit() // now commit everything so that it displays
```