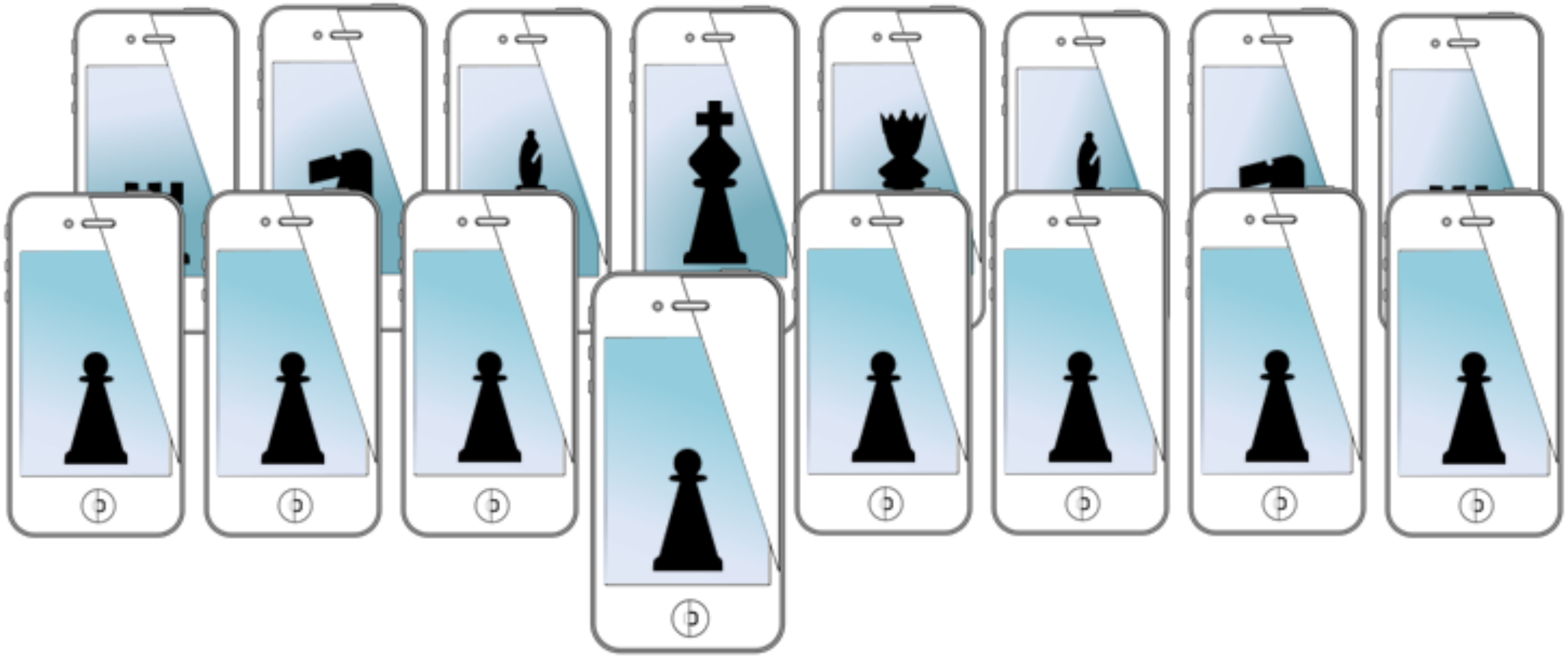
MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

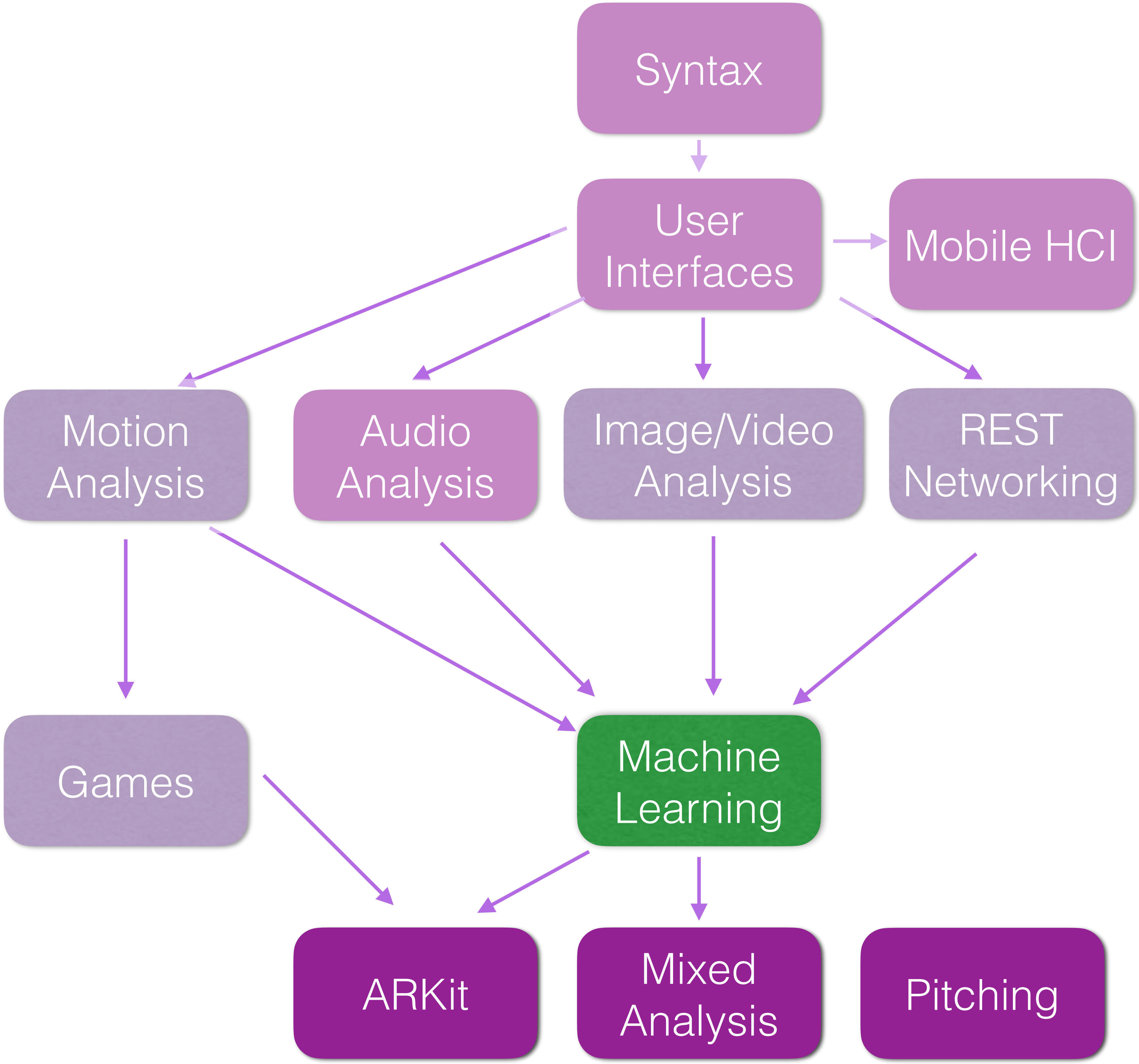
machine learning with Apple

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

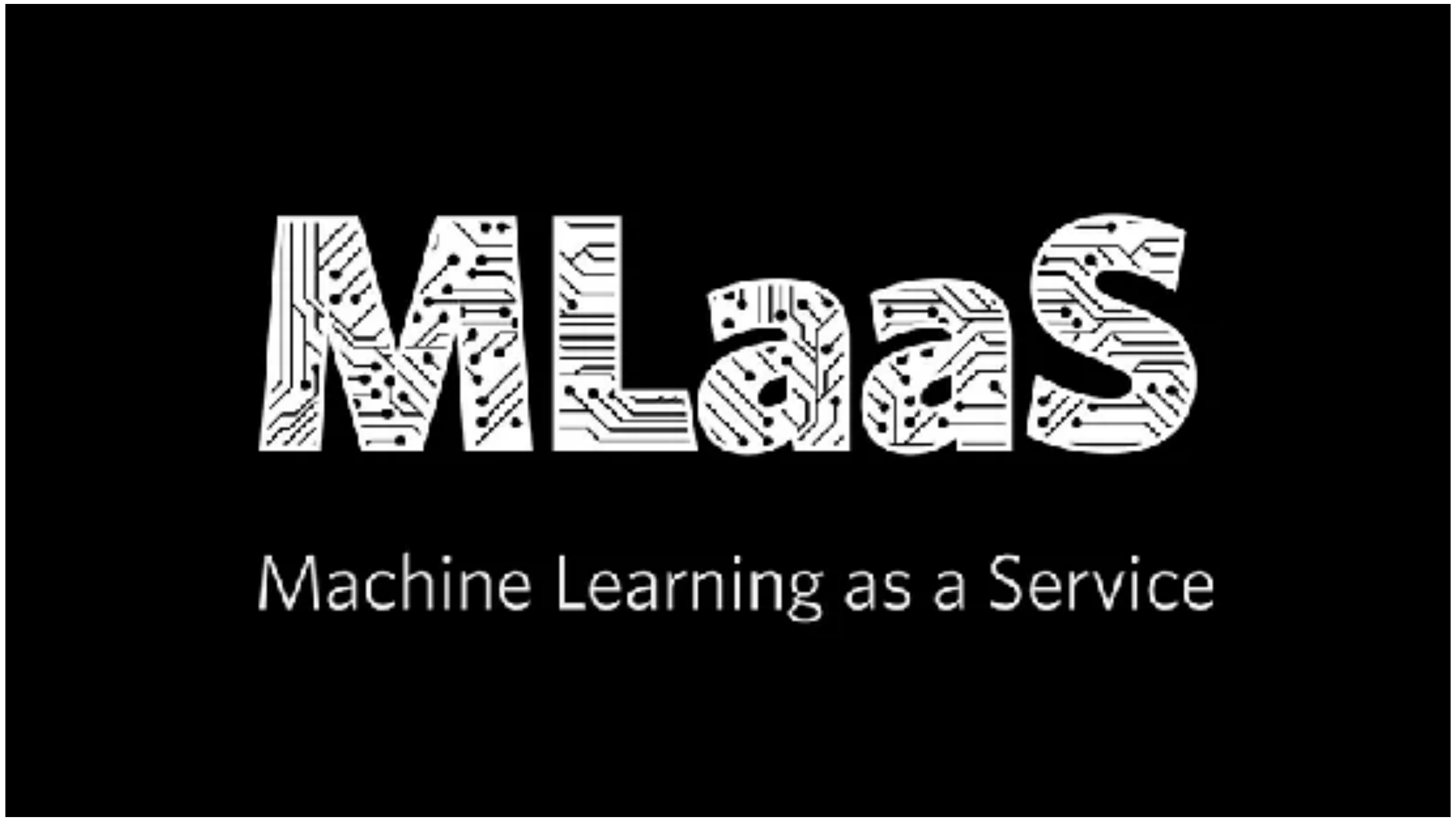
course logistics and agenda

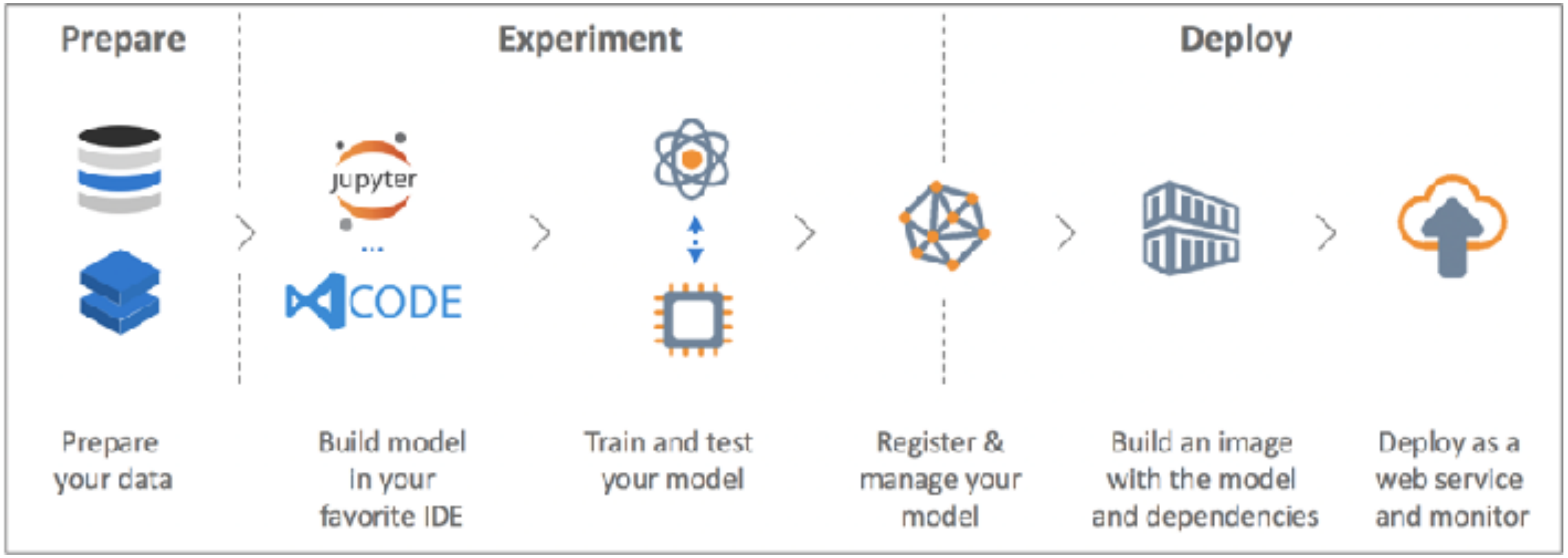
- logistics:
    - grading update
    - **A5** is due soon-ish, try to make it a first draft of your final project!
- Agenda
    - CoreML
    - **final project proposal** is also due at same time
        - today: talk about old projects!

class overview

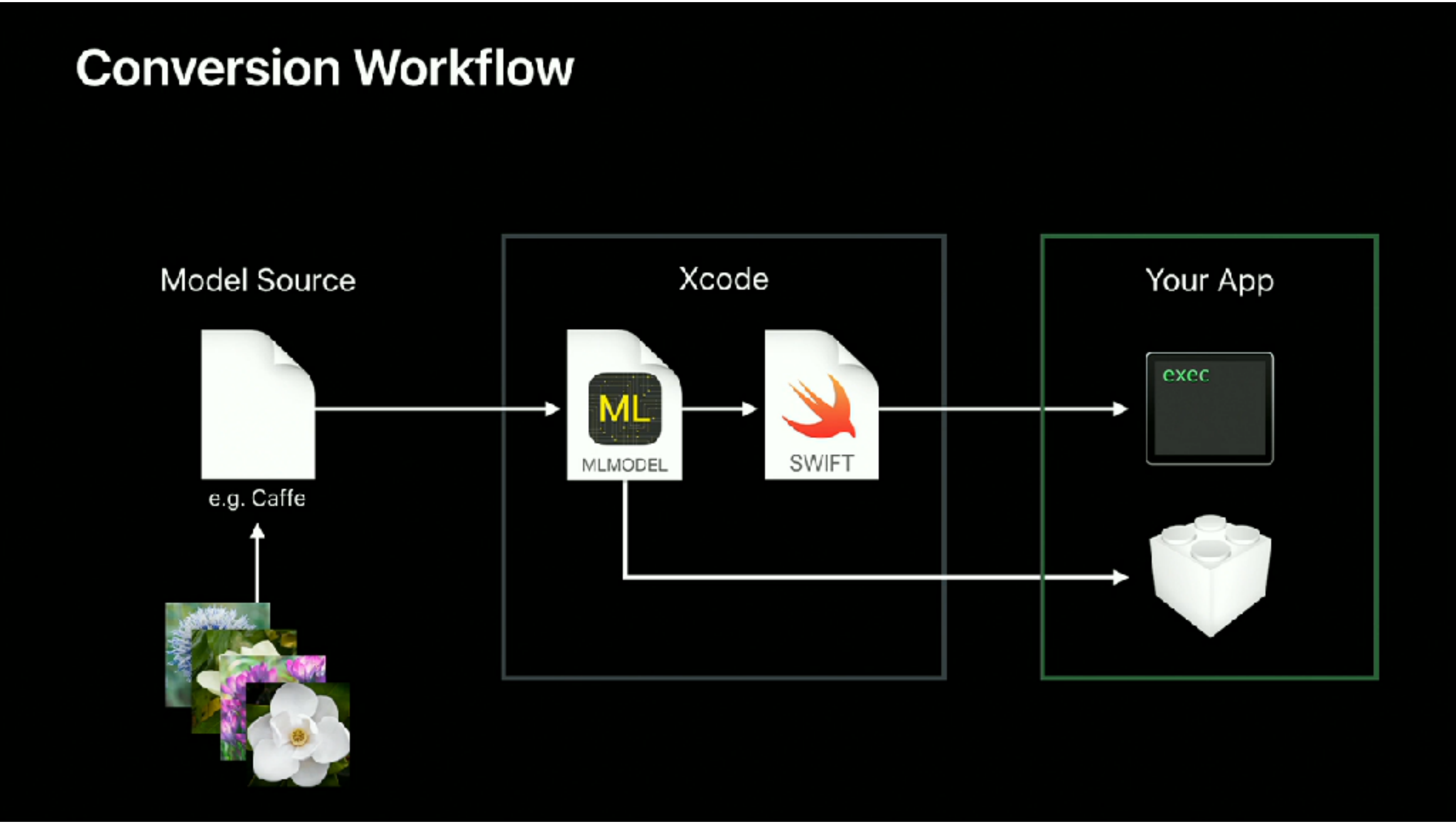


lab five town hall

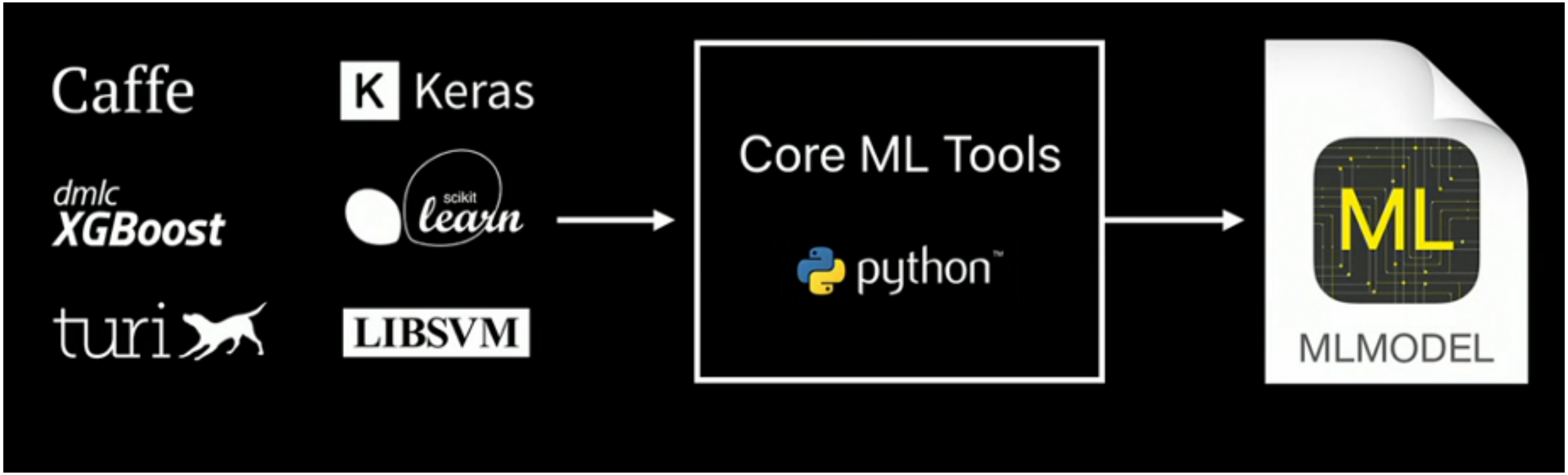



Prepare
Experiment
Deploy
jupyter
CODE
Prepare your data
Build model In your favorite IDE
Train and test your model
Register & manage your model
Build an image with the model and dependencies
Deploy as a web service and monitor

CoreML


Conversion Workflow
Model Source
e.g. Caffe
Xcode
ML
MLMODEL
SWIFT
Your App
exec

CoreML

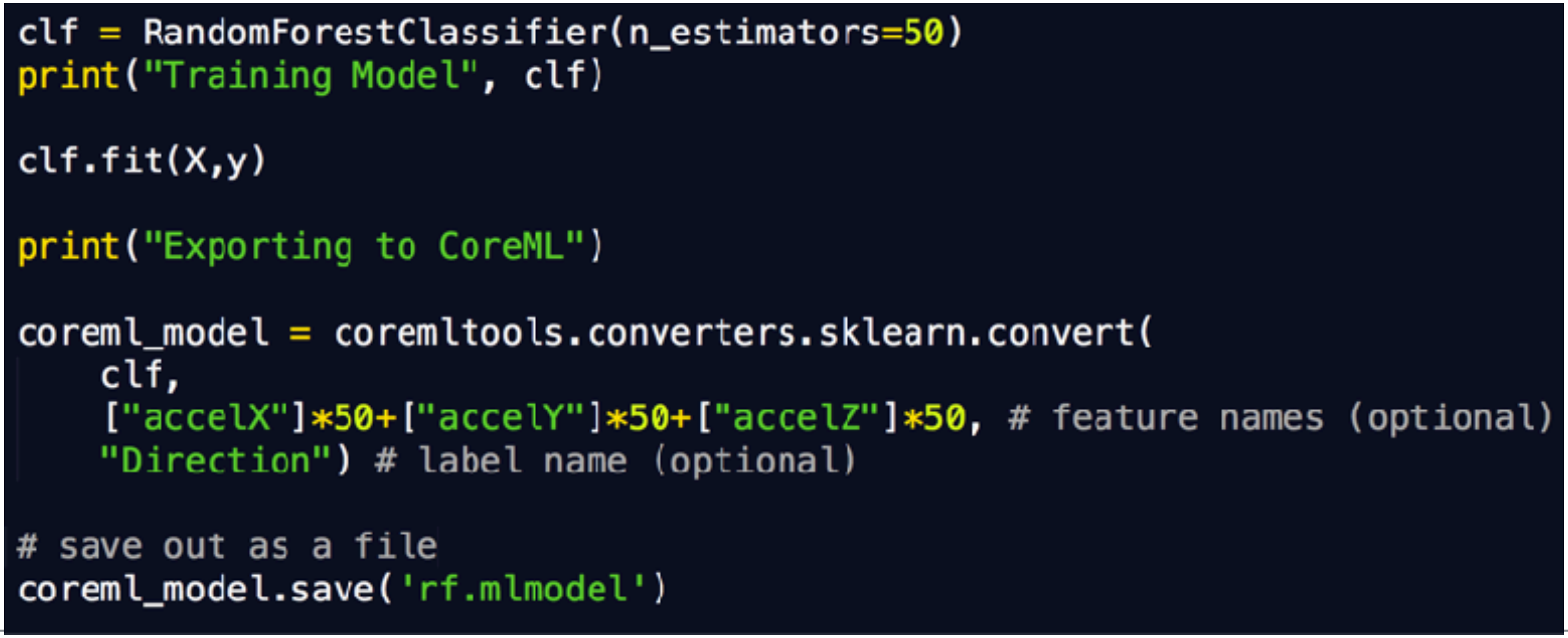

Caffe
Keras
dmlc XGBoost
scikit learn
turi
LIBSVM
Core ML Tools
python
ML
MLMODEL

but… we want more than **pre-trained** models

installing CoreMLTools

- built into TuriCreate
  `model.export_coreml("MyModel.mlmodel")`

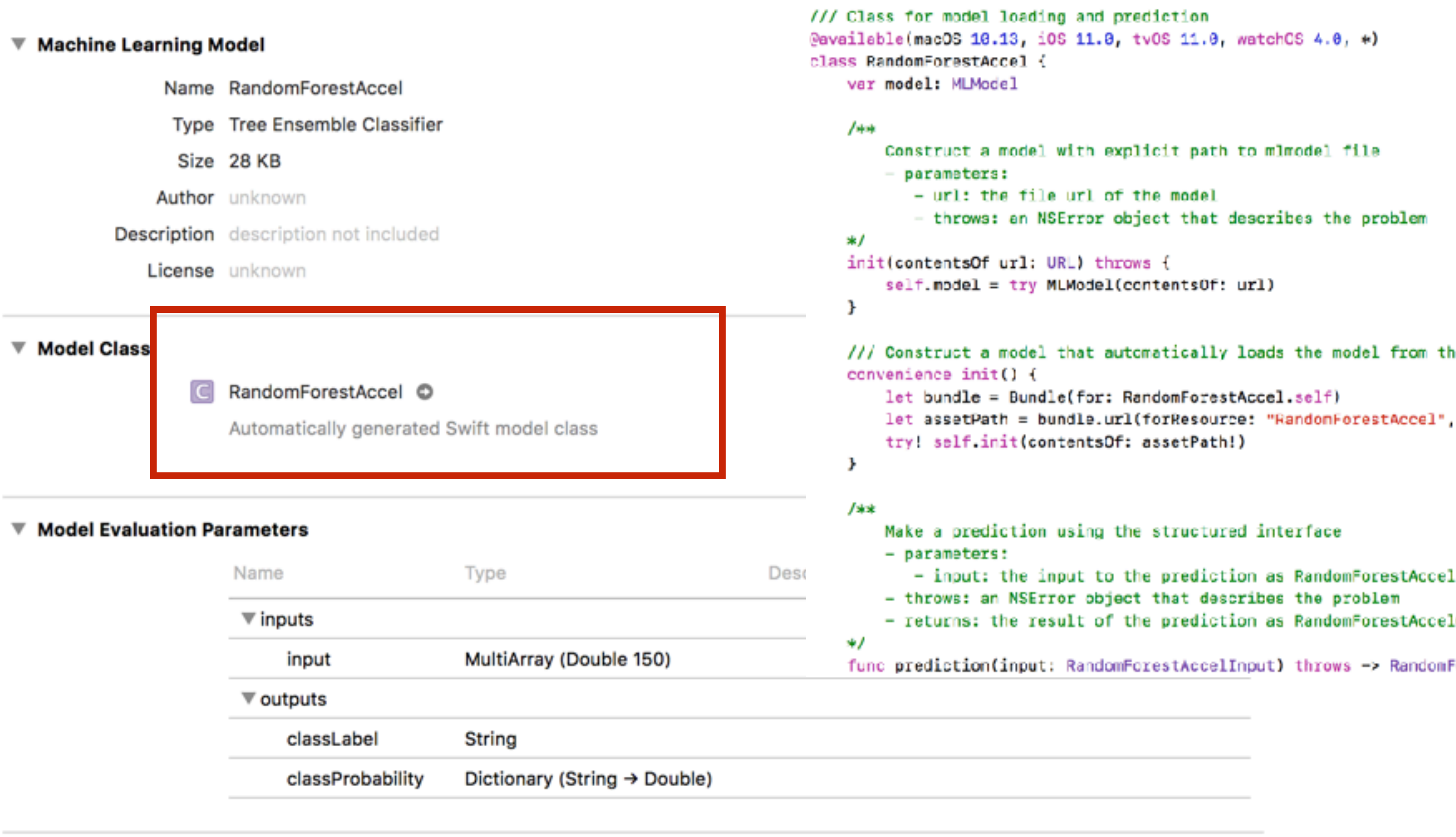
- also available for other libraries: so create a conda environment and install coremltools
  - `conda install sklearn numpy (+others) …`
  - `pip install coremltools`



```
clf = RandomForestClassifier(n_estimators=50)
print("Training Model", clf)

clf.fit(X,y)

print("Exporting to CoreML")

coreml_model = coremltools.converters.sklearn.convert(
    clf,
    ["accelX"]*50+["accelY"]*50+["accelZ"]*50, # feature names (optional)
    "Direction") # label name (optional)

# save out as a file
coreml_model.save('rf.mlmodel')
```

using CoreML

- drag into project



▼ **Machine Learning Model**

Name	RandomForestAccel
Type	Tree Ensemble Classifier
Size	28 KB
Author	unknown
Description	description not included
License	unknown

▼ **Model Class**

RandomForestAccel

Automatically generated Swift model class

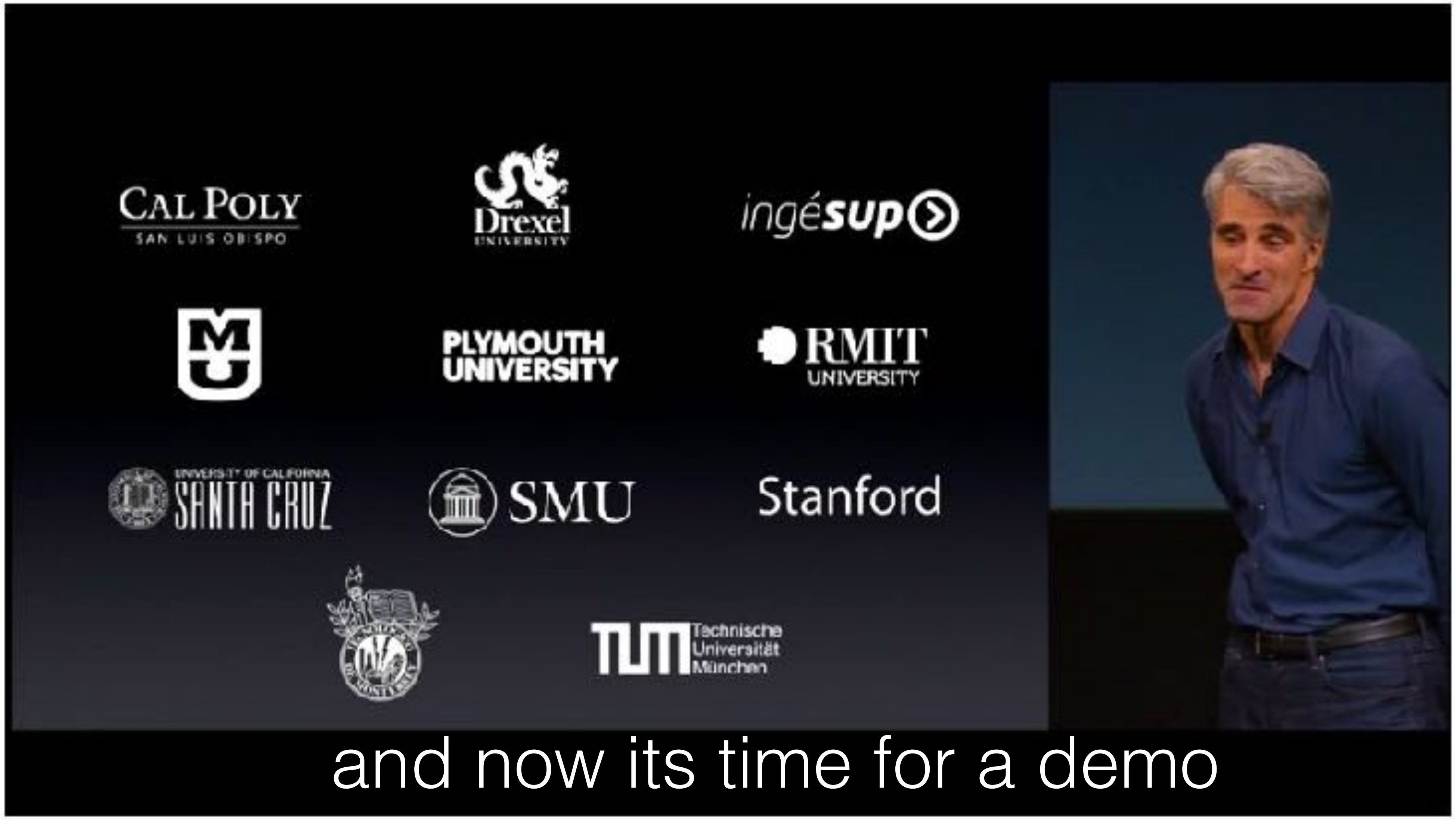
▼ **Model Evaluation Parameters**

Name	Type	Desc
▼ inputs		
input	MultiArray (Double 150)	
▼ outputs		
classLabel	String	
classProbability	Dictionary (String → Double)	

```
/// Class for model loading and prediction
@available(macOS 10.13, iOS 11.0, tvOS 11.0, watchOS 4.0, *)
class RandomForestAccel {
    var model: MLModel

    /**
        Construct a model with explicit path to mlmodel file
        - parameters:
          - url: the file url of the model
          - throws: an NSError object that describes the problem
    */
    init(contentsOf url: URL) throws {
        self.model = try MLModel(contentsOf: url)
    }

    /// Construct a model that automatically loads the model from th
    convenience init() {
        let bundle = Bundle(for: RandomForestAccel.self)
        let assetPath = bundle.url(forResource: "RandomForestAccel",
        try! self.init(contentsOf: assetPath!)
    }

    /**
        Make a prediction using the structured interface
        - parameters:
          - input: the input to the prediction as RandomForestAccel
        - throws: an NSError object that describes the problem
        - returns: the result of the prediction as RandomForestAccel
    */
    func prediction(input: RandomForestAccelInput) throws -> RandomF
```

CoreML

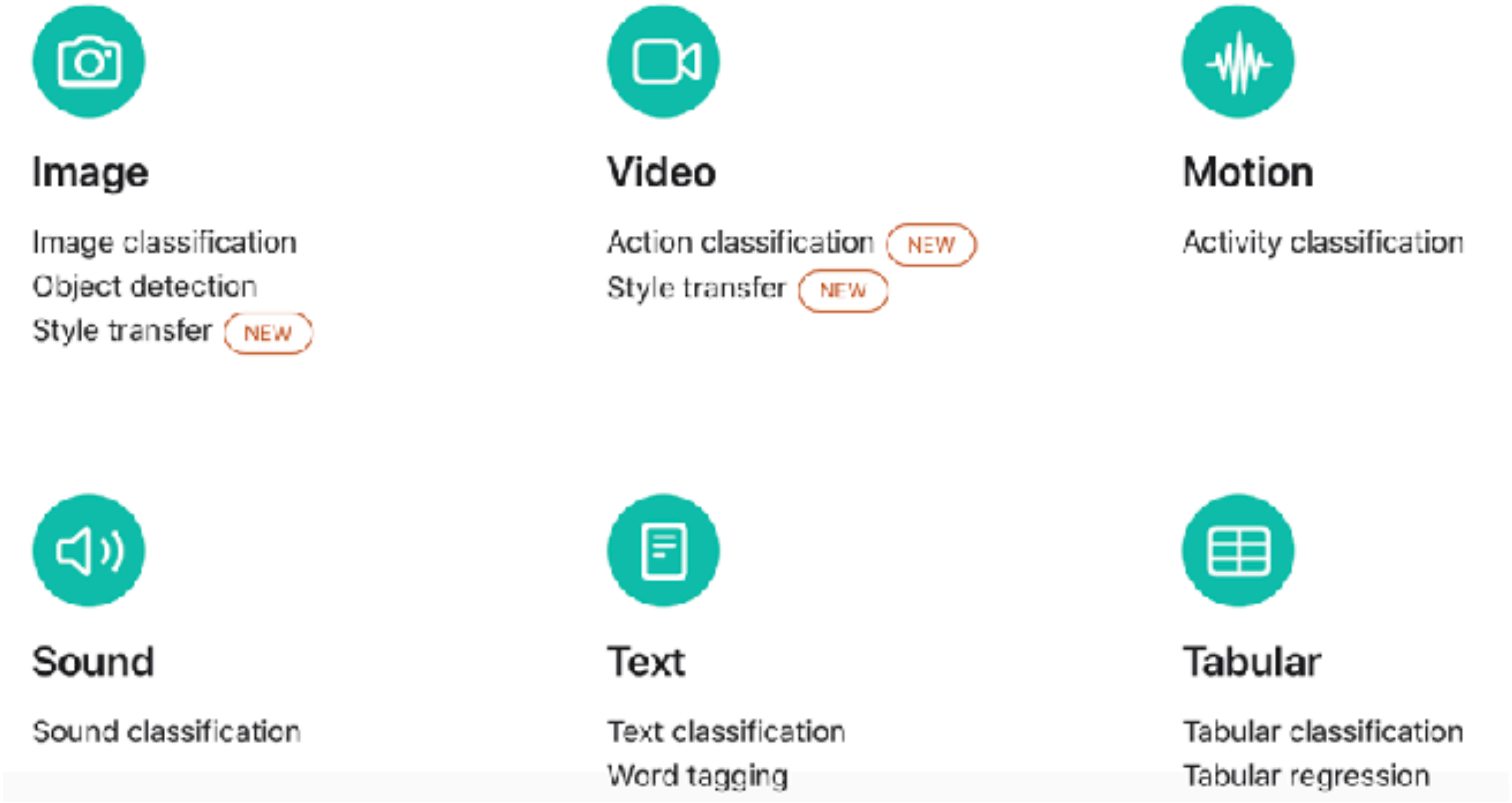
- extended demo from beginning to end
- combining our tornado, mongo, iOS, CoreML



and now its time for a demo

Create ML

making machine learning easy to use for developers that are not data scientists



Image

Image classification
Object detection
Style transfer NEW

Video

Action classification NEW
Style transfer NEW

Motion

Activity classification

Sound

Sound classification

Text

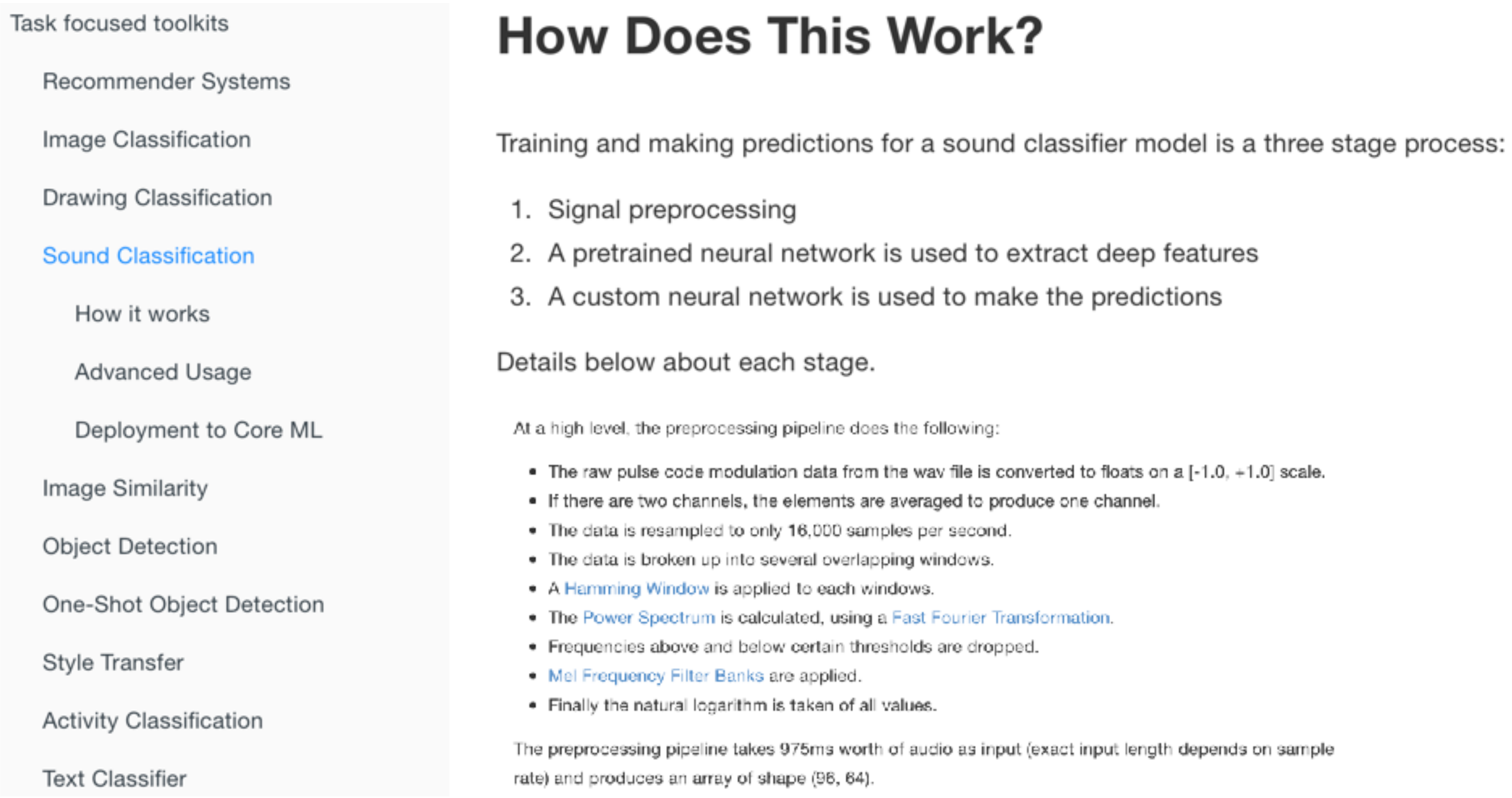
Text classification
Word tagging

Tabular

Tabular classification
Tabular regression

these are also built into TuriCreate!

could be good for final projects!!!



Task focused toolkits

- Recommender Systems
- Image Classification
- Drawing Classification
- Sound Classification
  - How it works
  - Advanced Usage
  - Deployment to Core ML
- Image Similarity
- Object Detection
- One-Shot Object Detection
- Style Transfer
- Activity Classification
- Text Classifier

How Does This Work?

Training and making predictions for a sound classifier model is a three stage process:

1. Signal preprocessing
2. A pretrained neural network is used to extract deep features
3. A custom neural network is used to make the predictions

Details below about each stage.

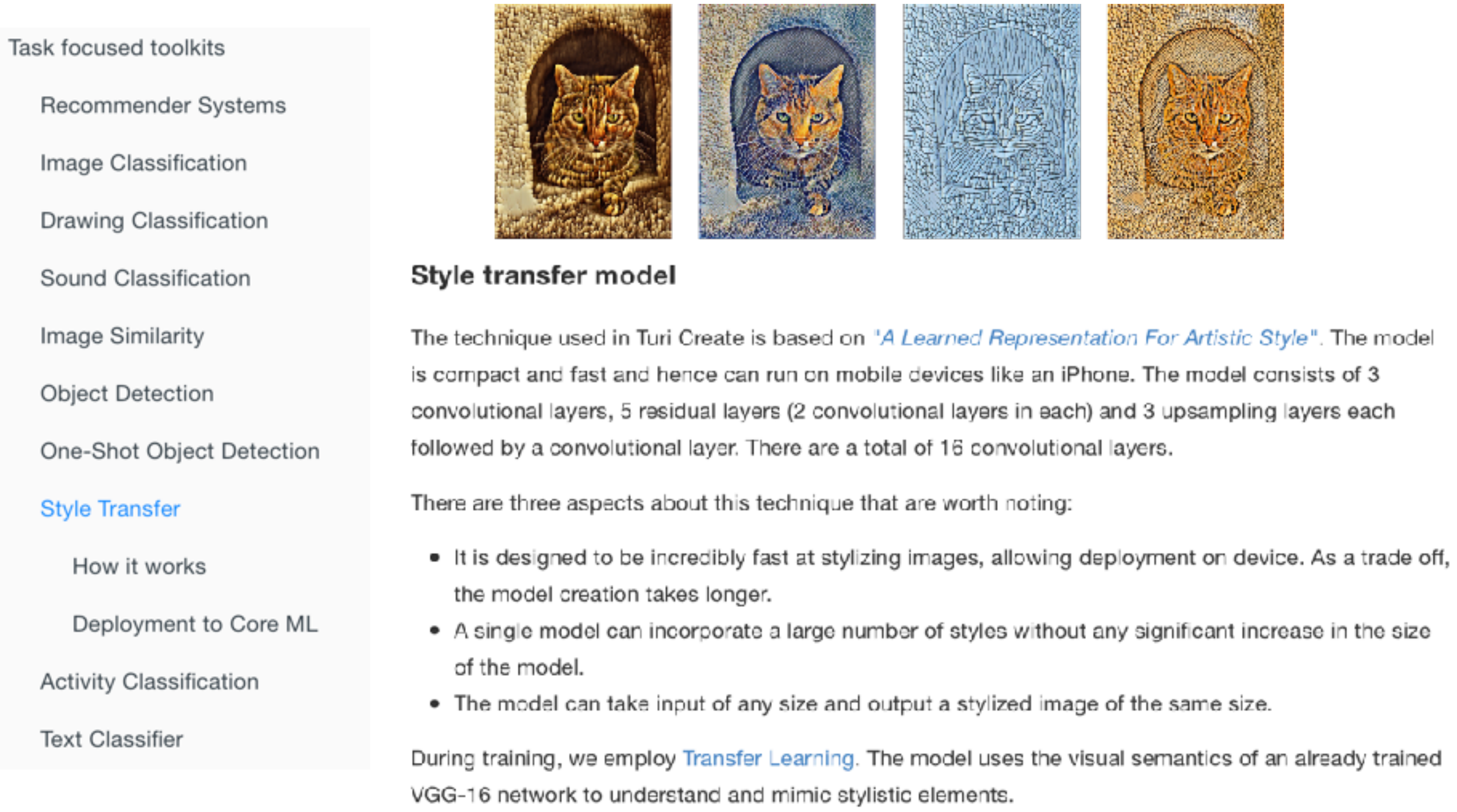
At a high level, the preprocessing pipeline does the following:

- The raw pulse code modulation data from the wav file is converted to floats on a [-1.0, +1.0] scale.
- If there are two channels, the elements are averaged to produce one channel.
- The data is resampled to only 16,000 samples per second.
- The data is broken up into several overlapping windows.
- A Hamming Window is applied to each windows.
- The Power Spectrum is calculated, using a Fast Fourier Transformation.
- Frequencies above and below certain thresholds are dropped.
- Mel Frequency Filter Banks are applied.
- Finally the natural logarithm is taken of all values.

The preprocessing pipeline takes 975ms worth of audio as input (exact input length depends on sample rate) and produces an array of shape (96, 64).

https://apple.github.io/turicreate/docs/userguide/sound_classifier/

could be good for final projects!!!



Style transfer model

The technique used in Turi Create is based on "*A Learned Representation For Artistic Style*". The model is compact and fast and hence can run on mobile devices like an iPhone. The model consists of 3 convolutional layers, 5 residual layers (2 convolutional layers in each) and 3 upsampling layers each followed by a convolutional layer. There are a total of 16 convolutional layers.

There are three aspects about this technique that are worth noting:

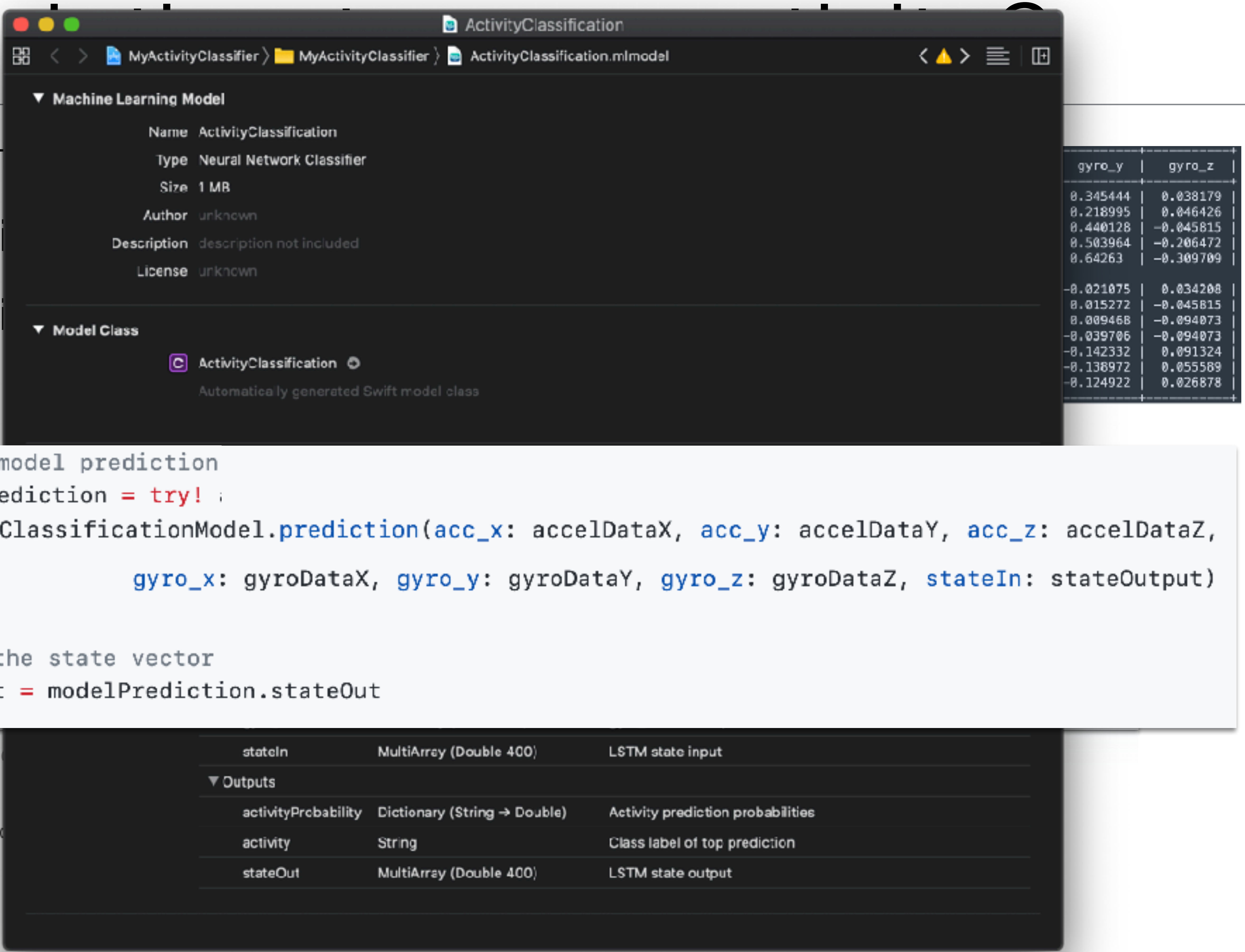
- It is designed to be incredibly fast at stylizing images, allowing deployment on device. As a trade off, the model creation takes longer.
- A single model can incorporate a large number of styles without any significant increase in the size of the model.
- The model can take input of any size and output a stylized image of the same size.

During training, we employ Transfer Learning. The model uses the visual semantics of an already trained VGG-16 network to understand and mimic stylistic elements.

https://apple.github.io/turicreate/docs/userguide/sound_classifier/

up

- forma
- organi
- organi
- train



ActivityClassification

MyActivityClassifier 〉 MyActivityClassifier 〉 ActivityClassification.mlmodel

▼ Machine Learning Model

Name	ActivityClassification
Type	Neural Network Classifier
Size	1 MB
Author	unknown
Description	description not included
License	unknown

▼ Model Class

ActivityClassification

Automatically generated Swift model class

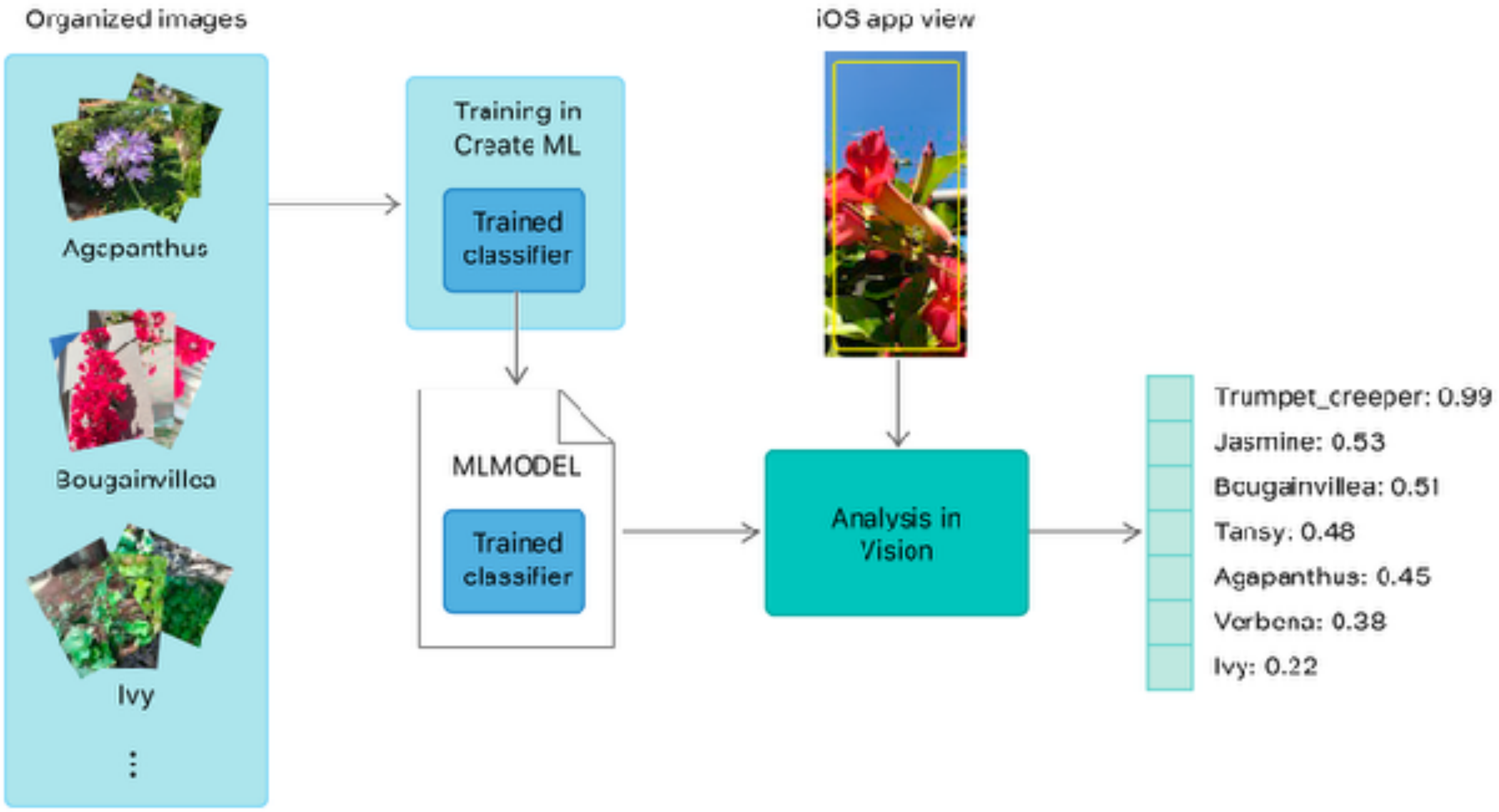
```
 gyro_y   |  gyro_z   |
0.345444  |  0.038179 |
0.218995  |  0.046426 |
0.440128  | -0.045815 |
0.503964  | -0.206472 |
0.64263   | -0.309709 |

-0.021075 |  0.034208 |
0.015272  | -0.045815 |
0.009468  | -0.094073 |
-0.039706 | -0.094073 |
-0.142332 |  0.091324 |
-0.138972 |  0.055589 |
-0.124922 |  0.026878 |
```

```
// Perform model prediction
let modelPrediction = try! 
   activityClassificationModel.prediction(acc_x: accelDataX, acc_y: accelDataY, acc_z: accelDataZ,

                        gyro_x: gyroDataX, gyro_y: gyroDataY, gyro_z: gyroDataZ, stateIn: stateOutput)


 // Update the state vector
 stateOutput = modelPrediction.stateOut
```

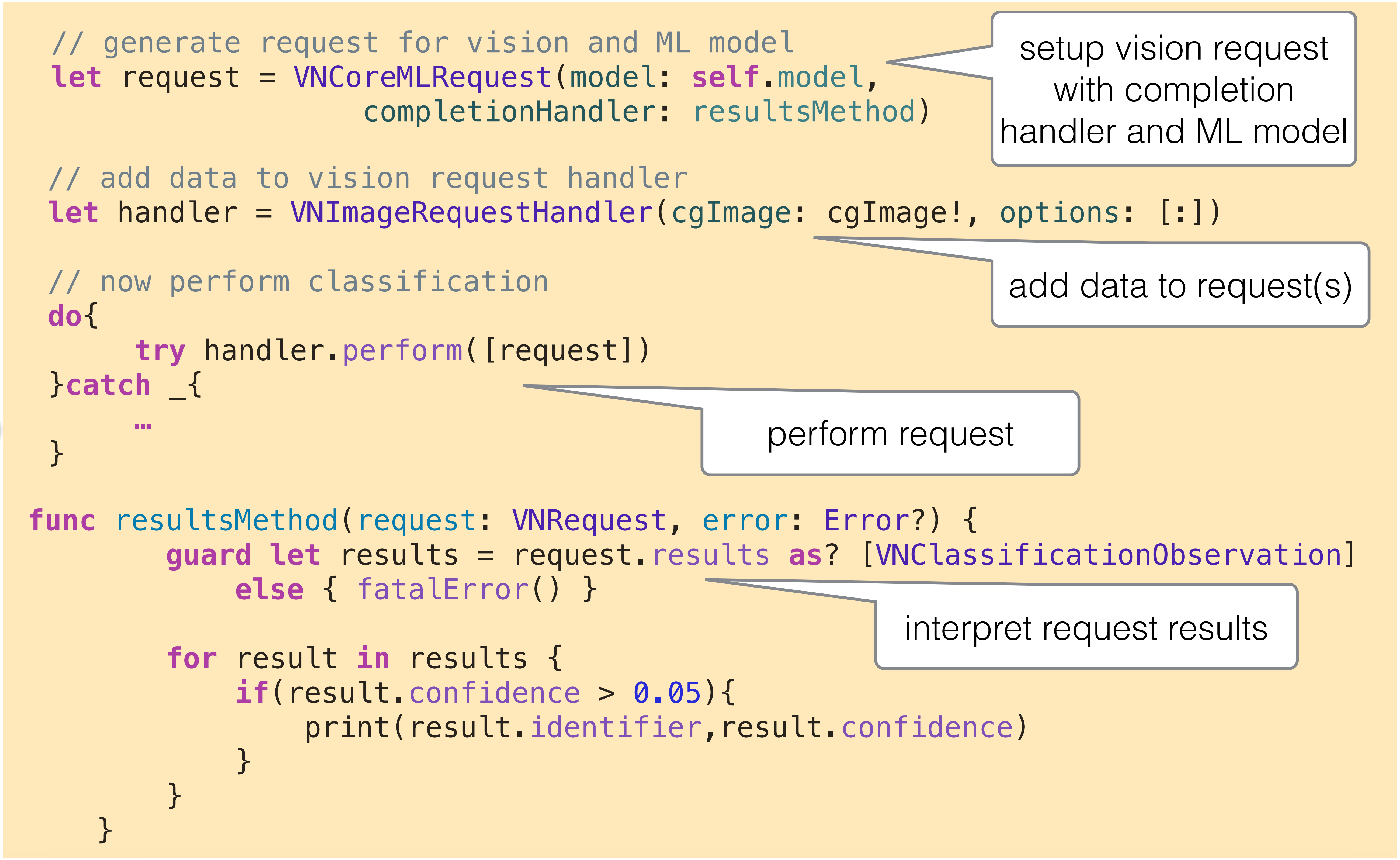
stateIn	MultiArray (Double 400)	LSTM state input
▼ Outputs		
activityProbability	Dictionary (String → Double)	Activity prediction probabilities
activity	String	Class label of top prediction
stateOut	MultiArray (Double 400)	LSTM state output

CoreML with vision API

- load ml model in Xcode
- wrap model
- create vision request
- wait for result in completion handler



the vision API



```
// generate request for vision and ML model
let request = VNCoreMLRequest(model: self.model,
                      completionHandler: resultsMethod)

// add data to vision request handler
let handler = VNImageRequestHandler(cgImage: cgImage!, options: [:])

// now perform classification
do{
     try handler.perform([request])
}catch _{
     …
}

func resultsMethod(request: VNRequest, error: Error?) {
        guard let results = request.results as? [VNClassificationObservation]
            else { fatalError() }

        for result in results {
            if(result.confidence > 0.05){
                print(result.identifier,result.confidence)
            }
        }
    }
```

setup vision request with completion handler and ML model

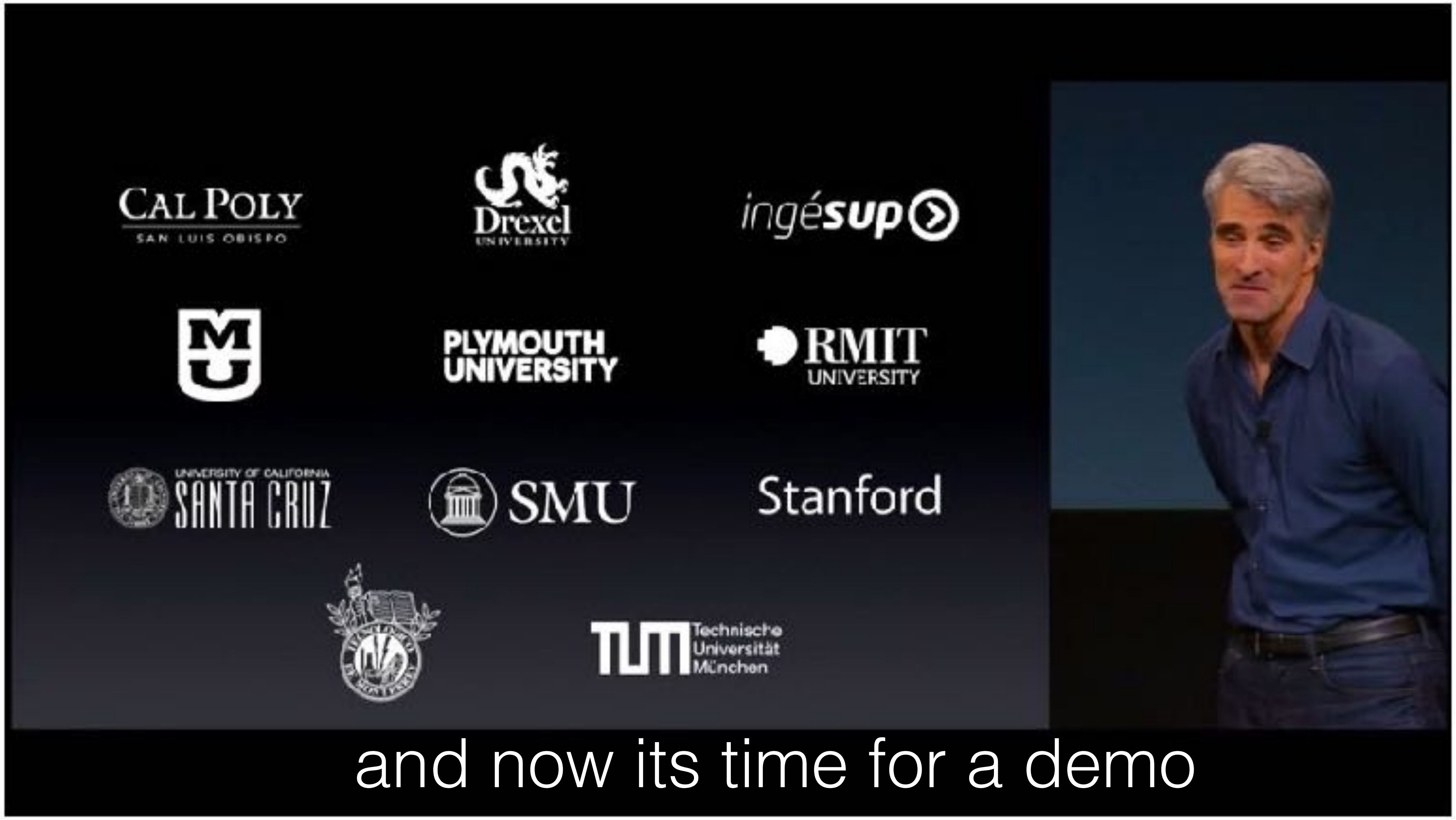
add data to request(s)

perform request

interpret request results

CoreML, a taste (if time)

- demo using vision API (if time)



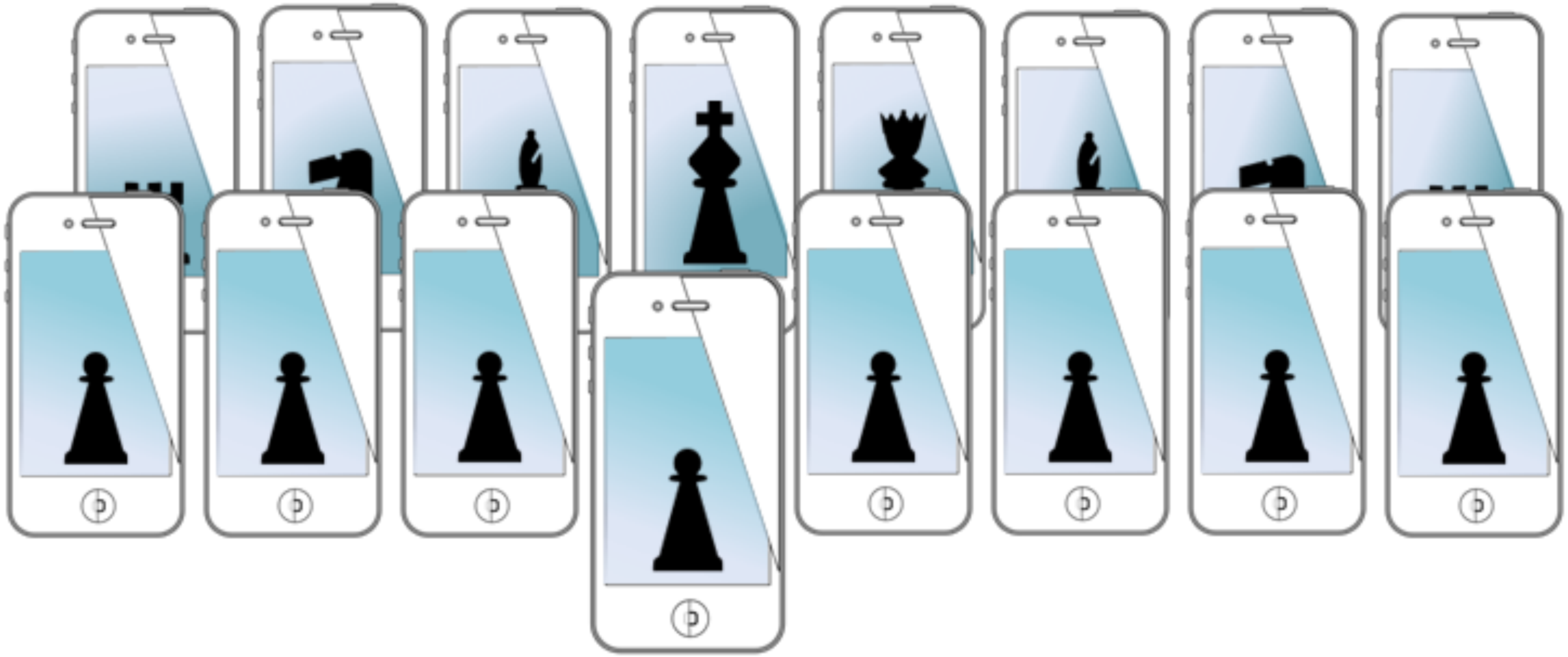
and now its time for a demo

https://github.com/SMU-MSLC/CoreMLVision

And now…

- final project discussion
- and proposal!!

MOBILE SENSING LEARNING



CS5323 & 7323

Mobile Sensing and Learning

machine learning crash course

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