# AI AND IOS

Aileen Nielsen

aileen.a.nielsen@gmail.com

# WHAT I WAS GOING TO TALK ABOUT

Pythonic aspects of Swift

Machine learning with Swift

Building neural networks for your apps

**But then…WWDC 2017 & CoreML**

# OUTLINE

Core ML & related

Machine learning concepts

Final bits and bobs

**Goal: empowerment**

# CORE ML & RELATED

# BEFORE WWDC 2016

- **Data scientists don’t always know much about the implementation details** of the techniques they use
  - Especially bad/poorly trained data scientists
  - They may know about ‘neural networks’ but don’t know how to break their models up into component pieces for implementation
- **But** iOS engineers are iOS engineers, **not data engineers or data scientists**
- **But** many interesting new applications are going to use some form of intelligence...and this trend will only accelerate
- **Open source projects** to
  - Translate existing ML models into code
  - Train models
  - Facilitate data-sciencey stuff

Swift A.I.

A.I. LIBRARY FOR SWIFT

**MAChineLearning**

Machine Learning for the Mac

**Intro**

*MAChineLearning* (pron. 'mækʃiːn 'lɜːnɪŋ) is framework that provides a quick and easy way to experiment Machine Learning with native code on the Mac, with some specific support for Natural Language Processing. It is written in Objective-C, but it is compatible by Swift.

Currently the framework supports:

- [Neural Networks](#Neural Networks)
- [Bag of Words](#Bag of Words)
- [Word Vectors](#Word Vectors)

(swix)

**Swift Matrix and Machine Learning Library**

Apple's Swift is a high level language that's *asking* for some numerical library to perform computation *fast* or at the very least *easily*. This is a bare-bones wrapper for that library.

A way to have iOS run high-level code similar to Python or Matlab is something I've been waiting for, and am incredibly excited to see the results. This will make porting complex signal processing algorithms to C *much* easier. Porting from Python/MATLAB to C was (and is) a pain in the butt, and this library aims to make the conversion between a Python/Matlab algorithm and a mobile app *simple*.

# WWDC 2016

# WWDC 2016

**BNNS** introduced a set of routines in the **Accelerate (CPU) framework** designed to implement neural networks, introduced in iOS 10

- Supports only 3 types of layers: Convolution, Fully Connected, Pooling

**MPSCNN** introduced a set of routines with **Metal (GPU) framework** to implement neural networks, introduced in iOS 10

- Also build neural networks **layer by layer**
- **More opportunities for customization** than BNNS
- Similar but not identical API

**WWDC Emphasis on AI**: speech recognition, user intent, SiriKit, machine learning, and NNs

# WWDC 2017

**CoreML**

- Close to a ‘**set it and forget it**’ style API
- Uses Metal & Accelerate under the hood

Neural-network-related enhancements to **Metal**

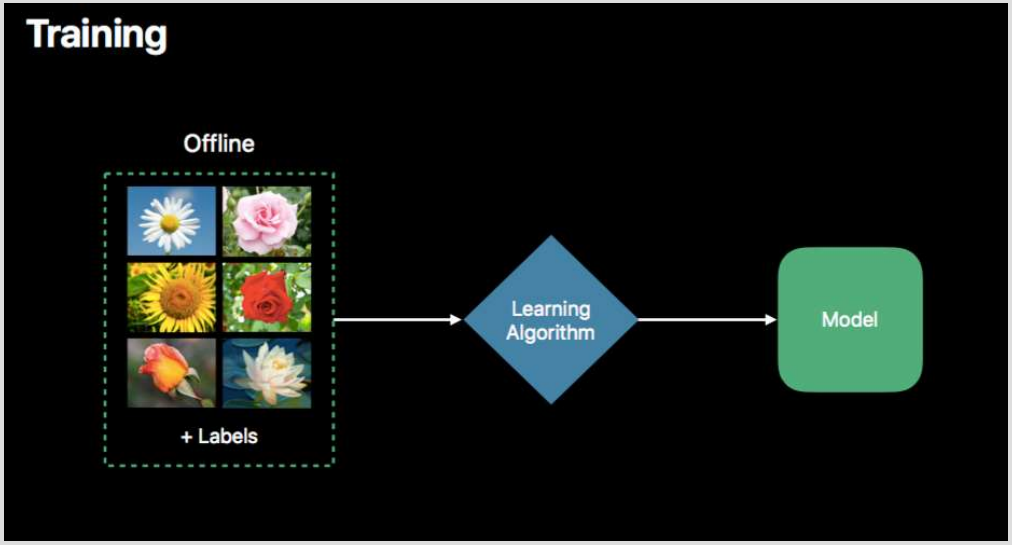
- **More kinds of kernels**
- API for **building graphs** makes this **more Keras like**
- Kernels are now **serializable**
- **More datatypes, in particular the preferred 16-bit float is now supported**

Seems **BNNS did not get much updating**

# CORE ML IN 5 MINUTES

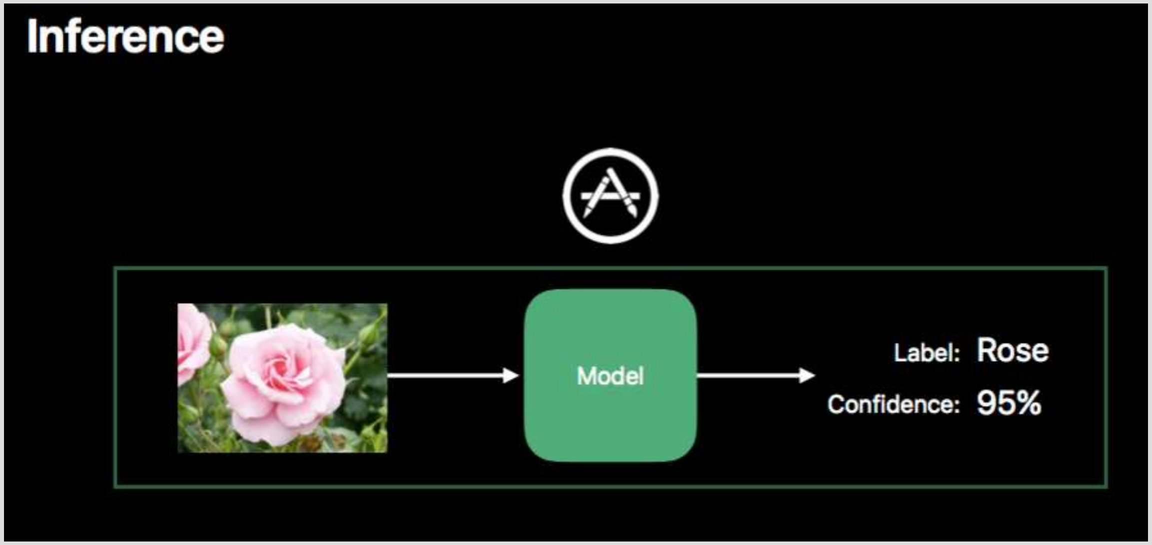
https://devstreaming-cdn.apple.com/videos/wwdc/2017/710vxa4hl8hyb72/710/710_core_ml_in_depth.pdf

# CORE ML IN 5 MINUTES



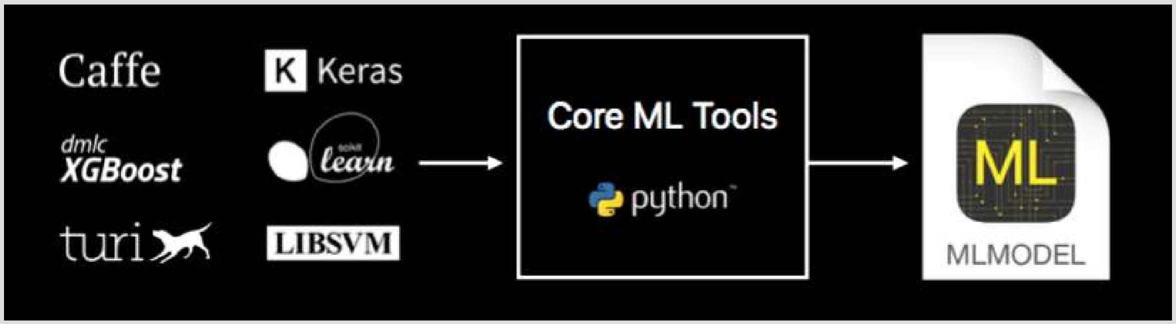

https://devstreaming-cdn.apple.com/videos/wwdc/2017/710vxa4hl8hyb72/710/710_core_ml_in_depth.pdf

# CORE ML IN 5 MINUTES




https://devstreaming-cdn.apple.com/videos/wwdc/2017/710vxa4hl8hyb72/710/710_core_ml_in_depth.pdf

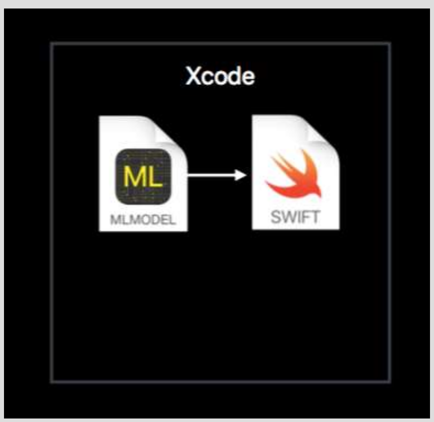
# CORE ML IN 5 MINUTES

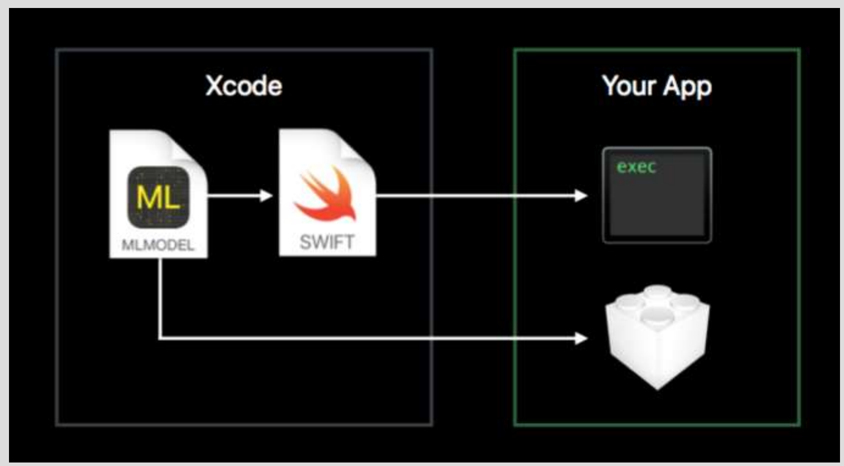



```python
import coremltools

keras_model = 'flowers.h5'
model = coremltools.converters.keras.convert(
                keras_model,
                image_input_names = 'data',
                class_labels = 'labels.txt')

model.save('FlowerClassifier.mlmodel')
```



# CORE ML IN 5 MINUTES

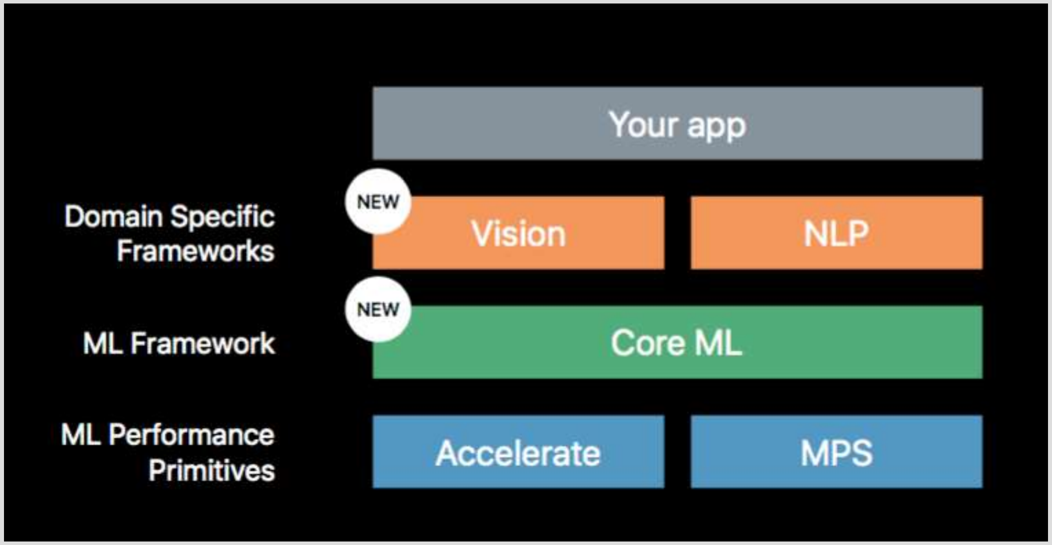

```swift
class FlowerClassifierInput {
  var flowerImage: CVPixelBuffer
}

class FlowerClassifierOutput {
  let flowerType: String
  let flowerTypeProbs: [String: Double]
}

class FlowerClassifier {
  convenience init()
  func prediction(flowerImage: CVPixelBuffer) throws -> FlowerClassifierOutput
}
```

```swift
let flowerModel = FlowerClassifier()
if let prediction = try? flowerModel.prediction(flowerImage: image) {
   return prediction.flowerType
}
```

# CORE ML IN 5 MINUTES



# COOL FEATURES

CoreML **decides which parts of a model to run on CPU vs. GPU** (so even the conscientious iOS engineer does not need to run CPU vs. GPU variants to see which is more performant)

- GPU is best for many parallel identical operations
- CPU is good for memory intensive or computationally serialized processes
- In general GPU tends to be better

Can run it from the **simulator**

- (unlike Metal)

In subsequent beta versions, got possibility to download and compile **mlmodel** so you don't have to release a new app to update your model

- This is **amazing** if your app basically is your model

# CURRENTLY SUPPORTED MODELS

- support vector machines (SVM)
- tree ensembles such as random forests and boosted trees
- linear regression and logistic regression
- neural networks: feed-forward, convolutional, recurrent

**More on these later…**

# DOWNSIDES TO COREML (OR OPPORTUNITIES TO LEARN)

- You probably need to **learn some Python**
  - But it's very Swifty
  - Or rather, Swift is very Pythonic
- **You will be responsible for understanding what is going wrong in models on device**
  - So you'll need to understand the basics of models
- Ultimately this will emerge as a **specialization** for iOS engineers
  - Model translation
  - Model implementation
  - Model performance enhancements and pitfalls
- **Not (yet?) open source**, so it will always be behind the data science tools

# CORE ML GOTCHAS

Models can **run fine on simulator but crash on device**

- E.g. on **simulator model ran on the CPU** and **on device ran on GPU** and did something impermissible

Many early **’bugs’** center around **not giving the right kind of input**

- There are **many ways to describe an image**, but only one that the model expects
- Unexpected model behavior is **usually the result of not matching expected input**

# PREDICTIONS FOR CORE ML

- **Fine tuning**
- **R** support
  - Finance – various packages but **glmnet** and **elasticnet** for starters
  - Sociology/politics – various packages, but **glmnet** and **survey** are possibly most important
  - Biostats – **Bioconductor** package
- **Unsupervised learning**
  - But this is **computationally taxing** so just as likely as on-device learning
  - **Everything is a lot less demanding than training neural networks**

# ML CONCEPTS

# REGULARIZATION

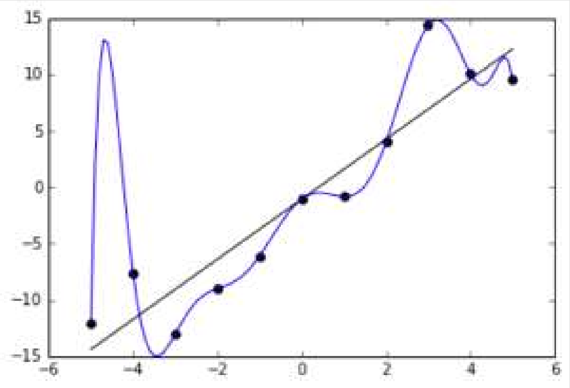
$$Obj(\Theta) = L(\Theta) + \Omega(\Theta)$$

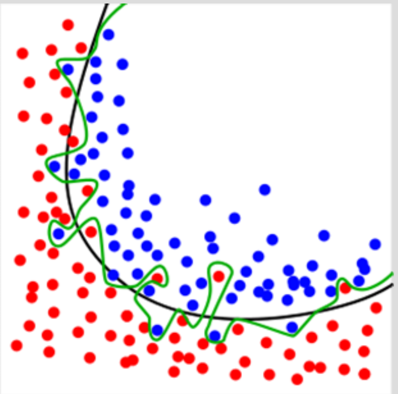
**Training Loss** measures how well model fit on training data

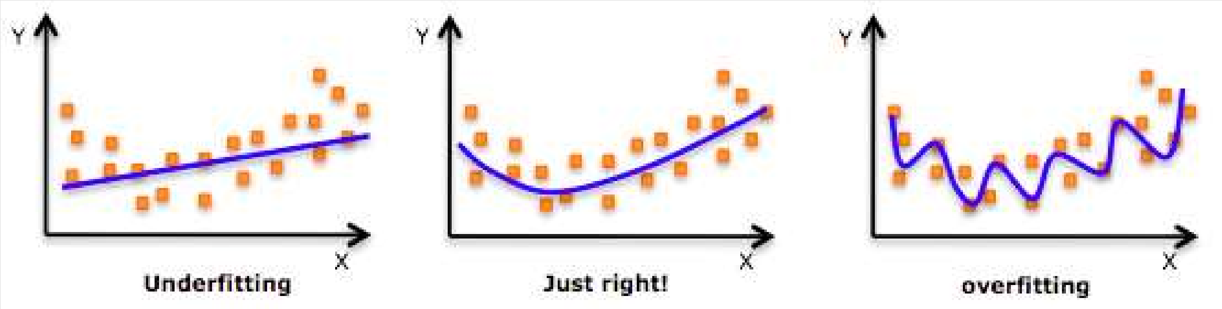
**Regularization**, measures complexity of model

https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf

# OVERFITTING (AND UNDERFITTING)



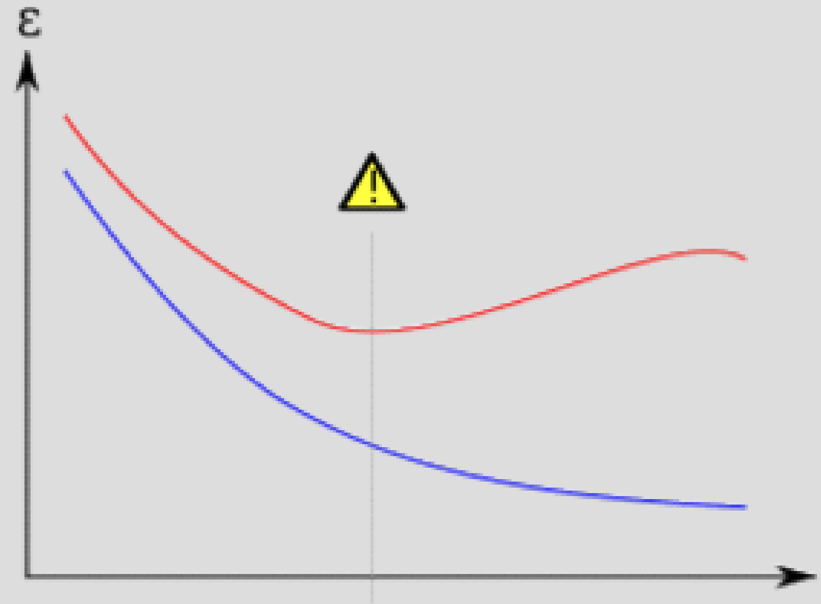



https://en.wikipedia.org/wiki/Overfitting
http://pingax.com/regularization-implementation-r/

# OVERFITTING DIAGNOSIS

Error

Cross-validation error




https://en.wikipedia.org/wiki/Overfitting

# OTHER TERMS YOU'LL ENCOUNTER

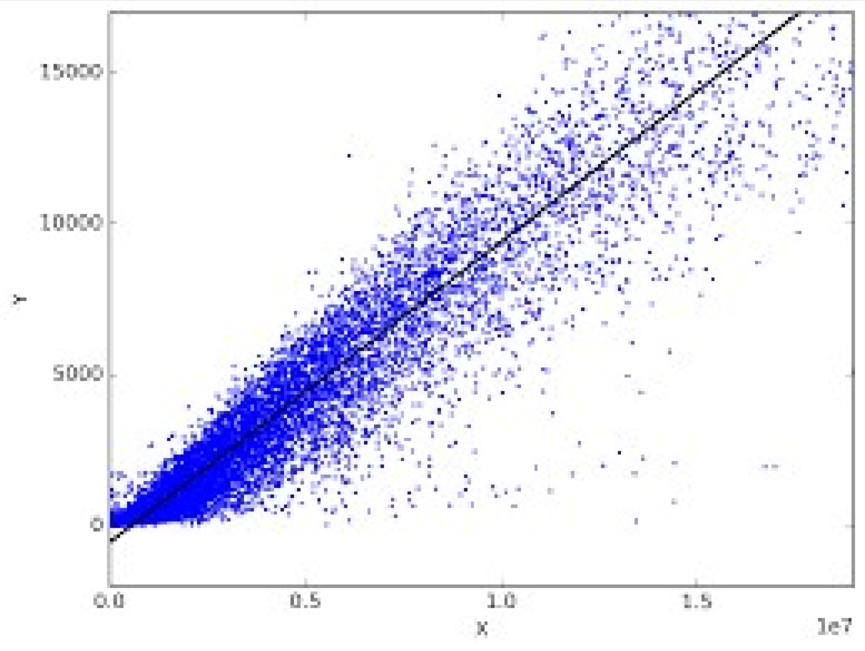
- Preprocessing
- Cross-validation
- Test & training sets
- Wide data vs. long data
- Normalized data
- Model compression

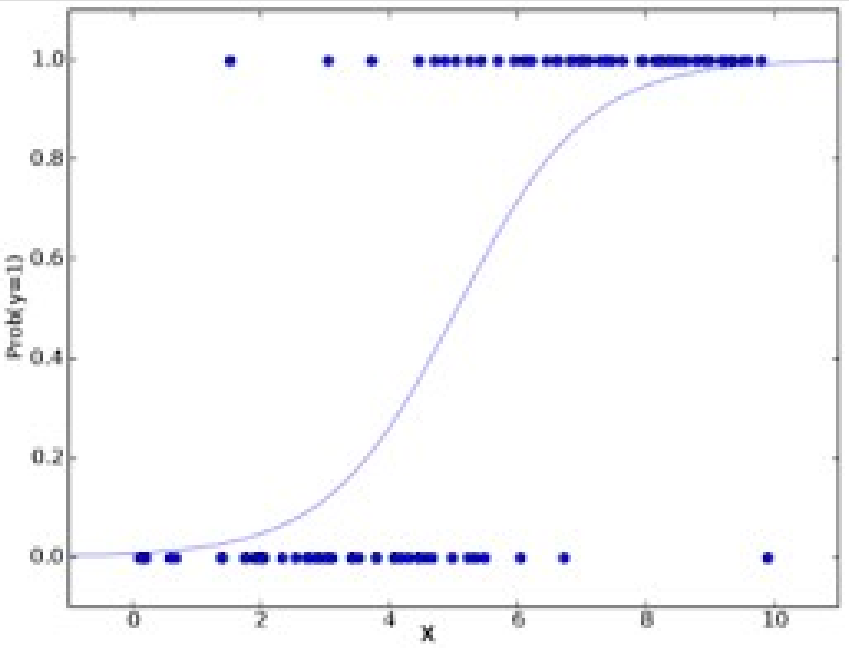
# CURRENTLY SUPPORTED MODELS

- **linear regression** and logistic regression
- support vector machines (**SVM**)
- **tree ensembles** such as random forests and boosted trees
- **neural networks**: feed-forward, convolutional, recurrent

# REGRESSION MODELS

- Linear regression models try to **predict a dependent value via a linear relationship with a set of predictors**




https://stats.stackexchange.com/questions/203872/what-to-do-when-a-linear-regression-gives-negative-estimates-which-are-not-possi

# REGRESSION MODELS

- Linear regression models try to **predict a dependent value via a linear relationship** with a set of predictors
  - Normal methodology makes vanilla **assumptions about input data**
  - However **assumptions aren't always true** in the real world but there is **usually a workaround**
  - **Computationally cheap** relative to many other methods
  - Remains workhorse of prediction
- **Logistic regression** uses linear methods but **seeks to model a probability,** which is bounded, and hence requires a linear separation
- Most real-world models will use some form of **regularization**
  - 'wide' datasts

# PROS AND CONS OF REGRESSION MODELS

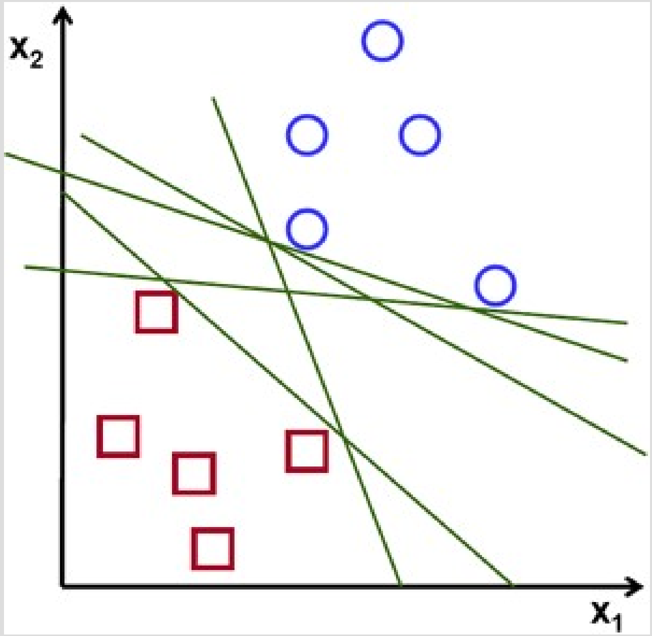
**Pros**

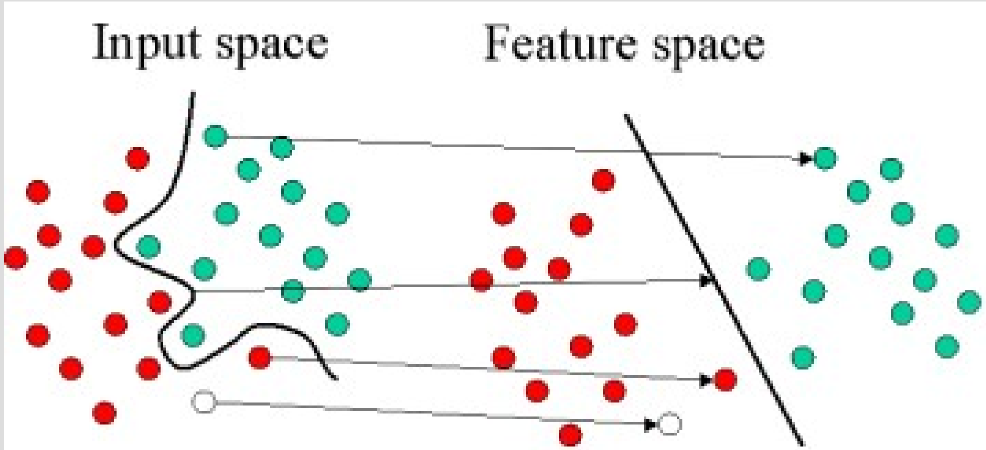
- Straightforward statistical models for generating confidence intervals Logistic regression provides probabilities, so we get classification + probability instead of just classification
- Some closed form solutions
- Well developed refinements for tricky data problems

**Cons**

- We don't always have a good reason to think linear relationships are linear
- Input data is not always independent even though it is assumed to be so
- Sensitive to outliers
- Only looks at the mean of a dependent variable without looking at the distribution

# SUPPORT VECTOR MACHINES




http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html
http://www.statsoft.com/Textbook/Support-Vector-Machines

# SUPPORT VECTOR MACHINES

- The support vector machine is a **separating plane** which best separates groups
- SVM selects a hyper plane which
  - First **maximizes classification accuracy**
  - Second **maximizes minimum distance from hyperplane** to members of each group (**maximizing the margin**)
- Possible to have a **non-linear hyperplane**
  - But usually want to **justify decision** to have a non-linear hyperplane
  - You can fit any set of points with enough parameters, that's a danger called **overfitting**

# PROS AND CONS OF SVM

**Pros**

- Relatively **memory efficient** so can be used in high dimensional spaces
- Works when **number of dimensions is greater than number of samples**
- Works really well with **strong signal (large margin)**
- More **robust to outliers** than regression

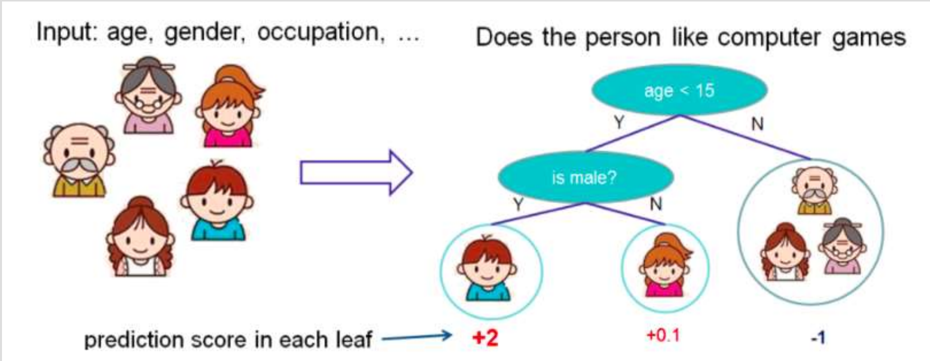
**Cons**

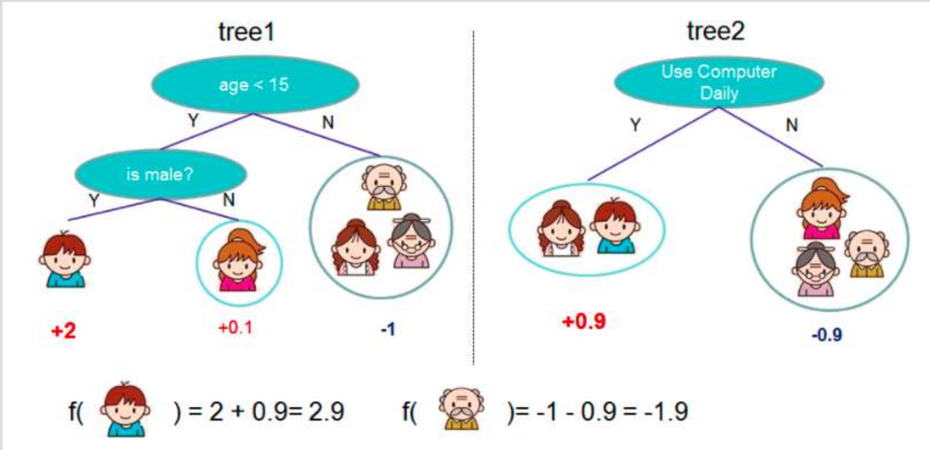
- Does not work so well with **noisy sampling** (small margin)
- With larger data set, very **long training times**
- Does **not provide probability estimates**

# TREE ENSEMBLES

- **Trees** are a fairly intuitive way of making both categorical and numerical predictions

- **Random forests:** It turns out that a bunch of 'dumb' trees together can do a lot better than a single tree




http://xgboost.readthedocs.io/en/latest/model.html

# PROS AND CONS OF TREE ENSEMBLES

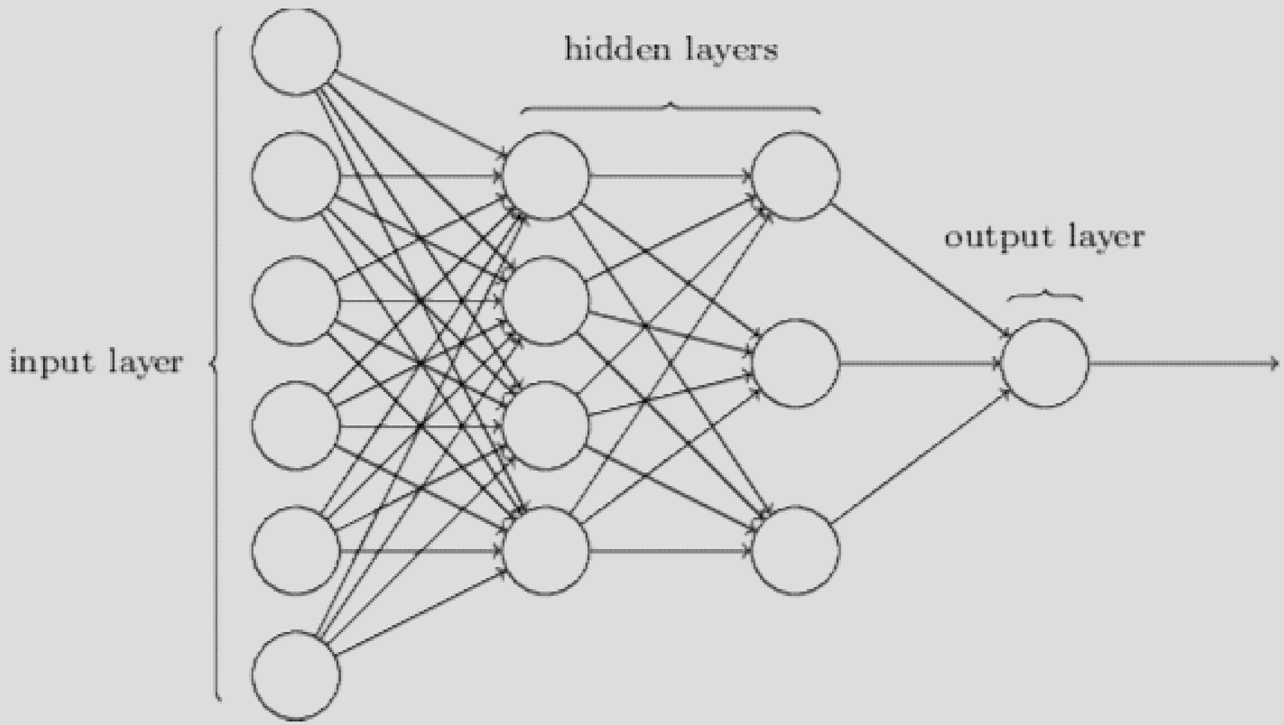
**Pros**

- Random ensemble methods are robust to highly correlated input variables
- High accuracy with uncomplicated methodology

**Cons**

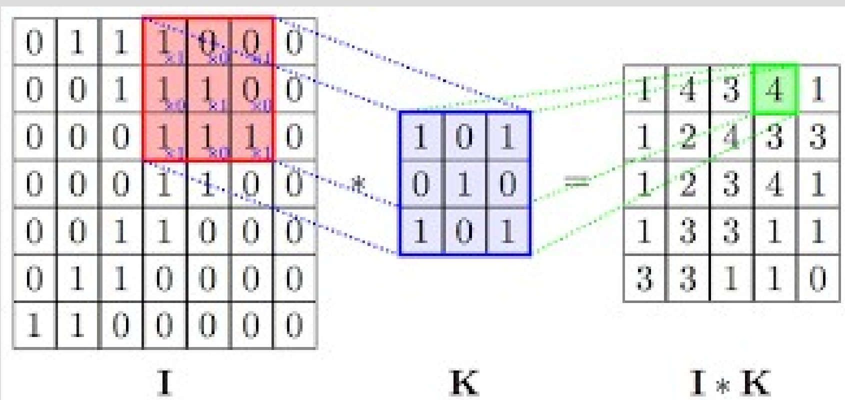
- Ensembles are not easy to visually interpret

# NEURAL NETWORKS




http://neuralnetworksanddeeplearning.com/chap1.html

# CONVOLUTIONAL NEURAL NETWORKS

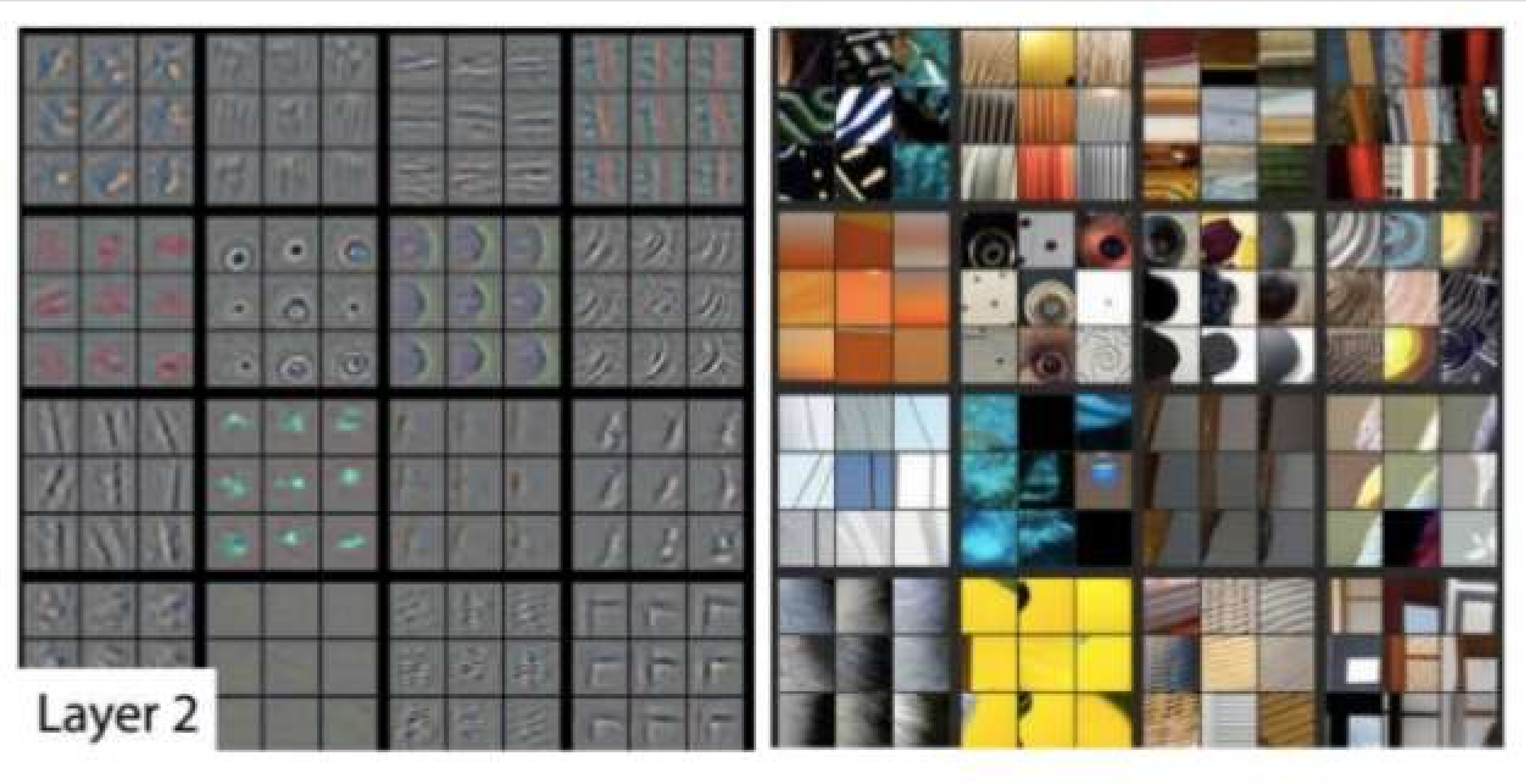



Original image

Center of Circle Blur on the right side of crab

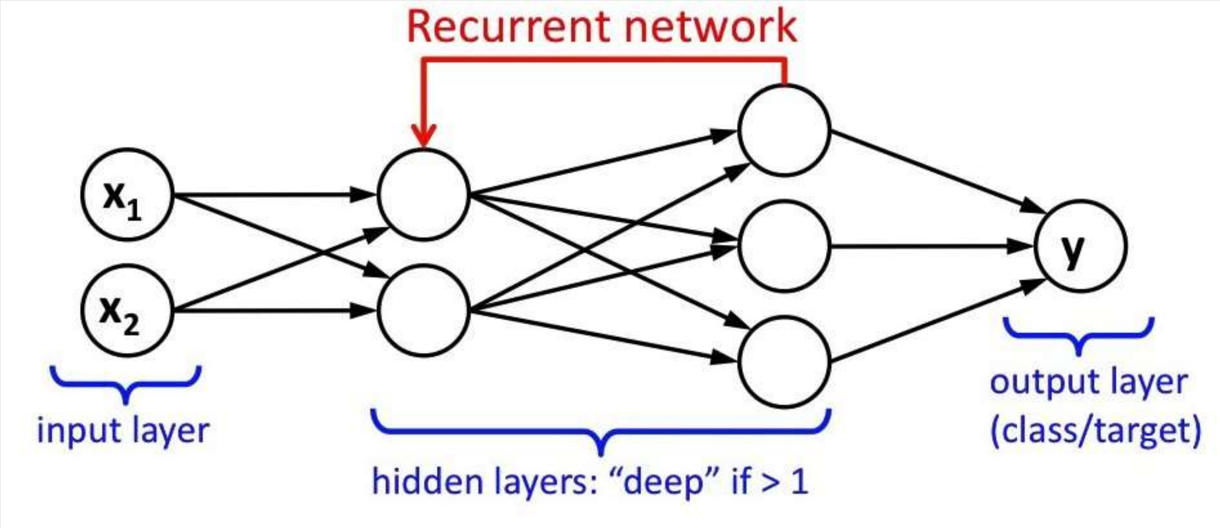
https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html
https://www.cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf
https://documentation.apple.com/en/motion/usermanual/index.html#chapter=16%26section=9%26tasks=true

# CONVOLUTIONAL NEURAL NETWORKS



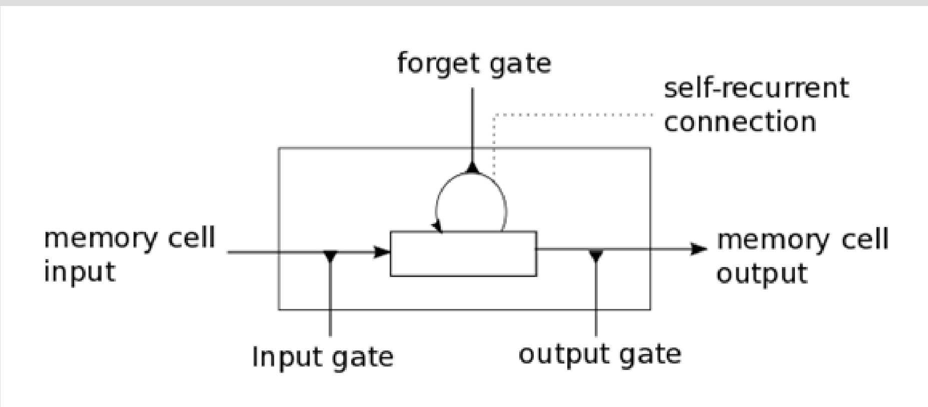

https://cambridgespark.com/content/tutorials/convolutional-neural-networks-with-keras/index.html
https://www.cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf

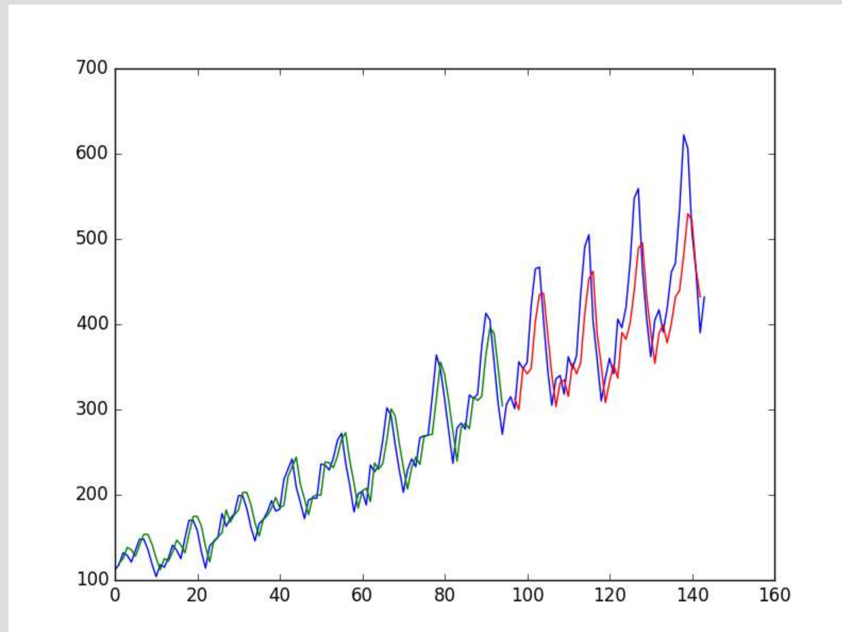
# RECURRENT NEURAL NETWORK




https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/recurrent_neural_networks.html

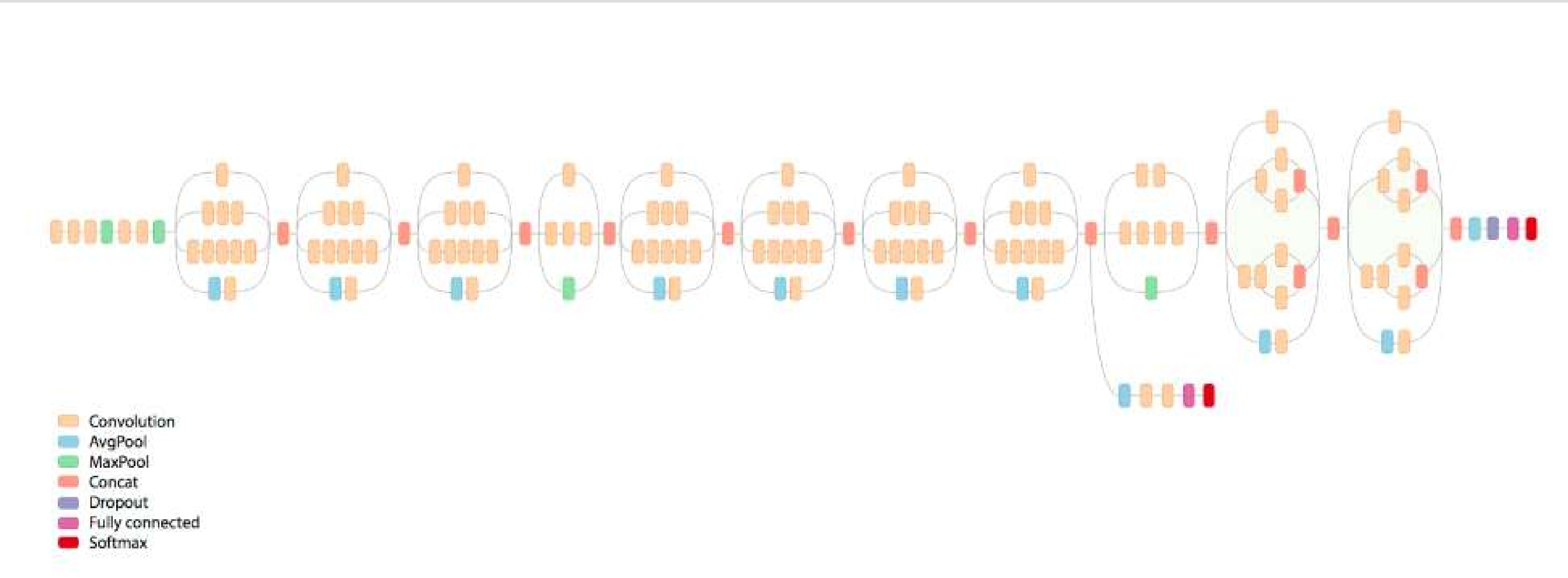
# RECURRENT NEURAL NETWORK




http://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/
http://deeplearning.net/tutorial/lstm.html

# STATE OF THE ART …ISH
# INCEPTION V. 3




https://academy.realm.io/posts/swift-developer-on-machine-learning-try-swift-2017-gallagher/

# FINAL BITS AND BOBS

# UPSIDES OF CORE ML ACCORDING TO APPLE

Apple listed the benefits of running on device as

- User **privacy**
- **Data** cost
- **Server** cost

But…

- **Larger footprint** on the device
- **Can correct models** on the server-side
- Server-side can **keep the model builders closer to the production** version of the model
- Apart from image-related applications, many models run on **small input data**

# AND SOME DOWNSIDES

- For some organizations, the model is the **core IP** and **core worth** of the enterprise…so organizations may not want to include more engineers on the list of people familiar with the model

- **Model performance/correctness checking may get forgotten**

# ETHICS

- Ethics in AI and machine learning is an **increasingly fraught area**

- Possibility that there is **unintentional racial/gender discrimination**
  - usually done by proxy such as using zip codes which can be a strong proxy for race

- This **may become an increasingly regulated area**

- You have the o**bligation to make sure you are putting out correct models**, even if you didn't build them

- Some tension **between privacy and fairness** because how can you check downstream effects?

# PRIVACY

- **Apple is emphasizing privacy** in all its AI implementations
- But at some point **bug reporting** is going to clash with privacy
- The **initial (and corrective) data** has to come from somewhere

# WHAT DOES THIS MEAN FOR YOUR CAREER?

- You should get to talk to **more kinds of people** at your organizations
- As always, the value play is what can you **add** to the APIs Apple is providing?
- New demands on **QA & bug searches**
- **New frontiers**

# THERE ARE STILL INTERESTING THINGS GOING ON BEYOND APPLE API

## DSLR-Quality Photos on Mobile Devices with Deep Convolutional Networks

Andrey Ignatov, Nikolay Kobyshev, Kenneth Vanhoey, Radu Timofte, Luc Van Gool

ETH Zurich

andrey.ignatoff@gmail.com, {nk, vanhoey, timofter, vangool}@vision.ee.ethz.ch



### Abstract

Despite a rapid rise in the quality of built-in smartphone cameras, their physical limitations — small sensor size, compact lenses and the lack of specific hardware, — impede them to achieve the quality results of DSLR cameras. In this work we present an end-to-end deep learning approach that bridges this gap by translating ordinary photos into DSLR-produced images. We propose learning the translation function using a residual convolutional neural network that improves both color rendition and image sharpness.

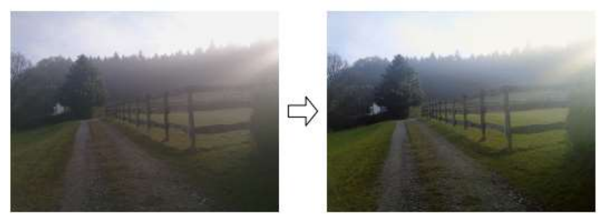


Figure 1: Sony smartphone image enhanced to DSLR-quality by our method. Best zoomed on screen.

http://alexsosn.github.io/ml/2017/05/02/mobile-ml-papers.html

# THERE ARE STILL INTERESTING THINGS GOING ON BEYOND APPLE API

**Real-Time Visual Place Recognition for Personal Localization on a Mobile Device**

Michał Nowicki · Jan Wietrzykowski · Piotr Skrzypczyński


**Fig. 4** The trajectories of two PUT MC test sequences: `PUT MC short`, marked by orange dots, and `PUT MC long`, marked by blue dashes

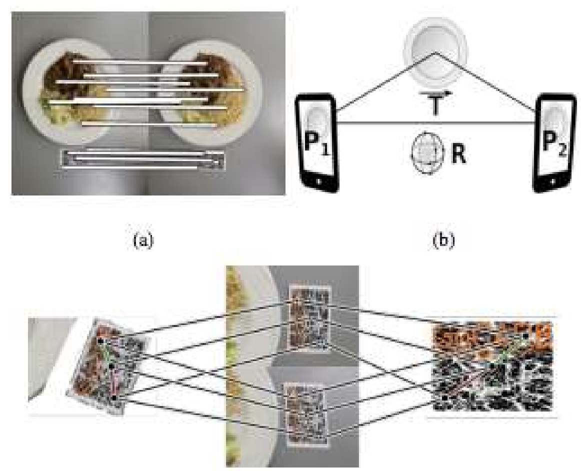

**Fig. 5** OpenFABMAP incorrectly recognized locations: A) due to many features placed on windows, B) due to people occluding the view

http://alexsosn.github.io/ml/2017/05/02/mobile-ml-papers.html

# THERE ARE STILL INTERESTING THINGS GOING ON BEYOND APPLE API

## Two-view 3D Reconstruction for Food Volume Estimation

Joachim Dehais, *Student Member, IEEE*, Marios Anthimopoulos, *Member, IEEE*, Sergey Shevchik, Stavroula Mougiakakou, *Member, IEEE*



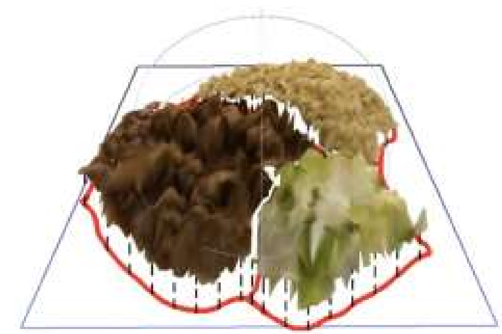



Fig. 6. Volume estimation: food surface (separated using the segmentation map), plate surface and vertical projection on the dish plane.

THE HARD THINGS ARE EASY,
AND **THE EASY THINGS ARE HARD**