Mobile ML Workshop

ML@B · Spring 2018

Installation and Setup

- `pip install numpy keras tensorflow`
- Download Xcode from the Mac App Store
- `git clone https://github.com/mlberkeley/mobile_ml_workshop`
- **Sign In Here**: www.tinyurl.com/mobile-ml

Who We Are

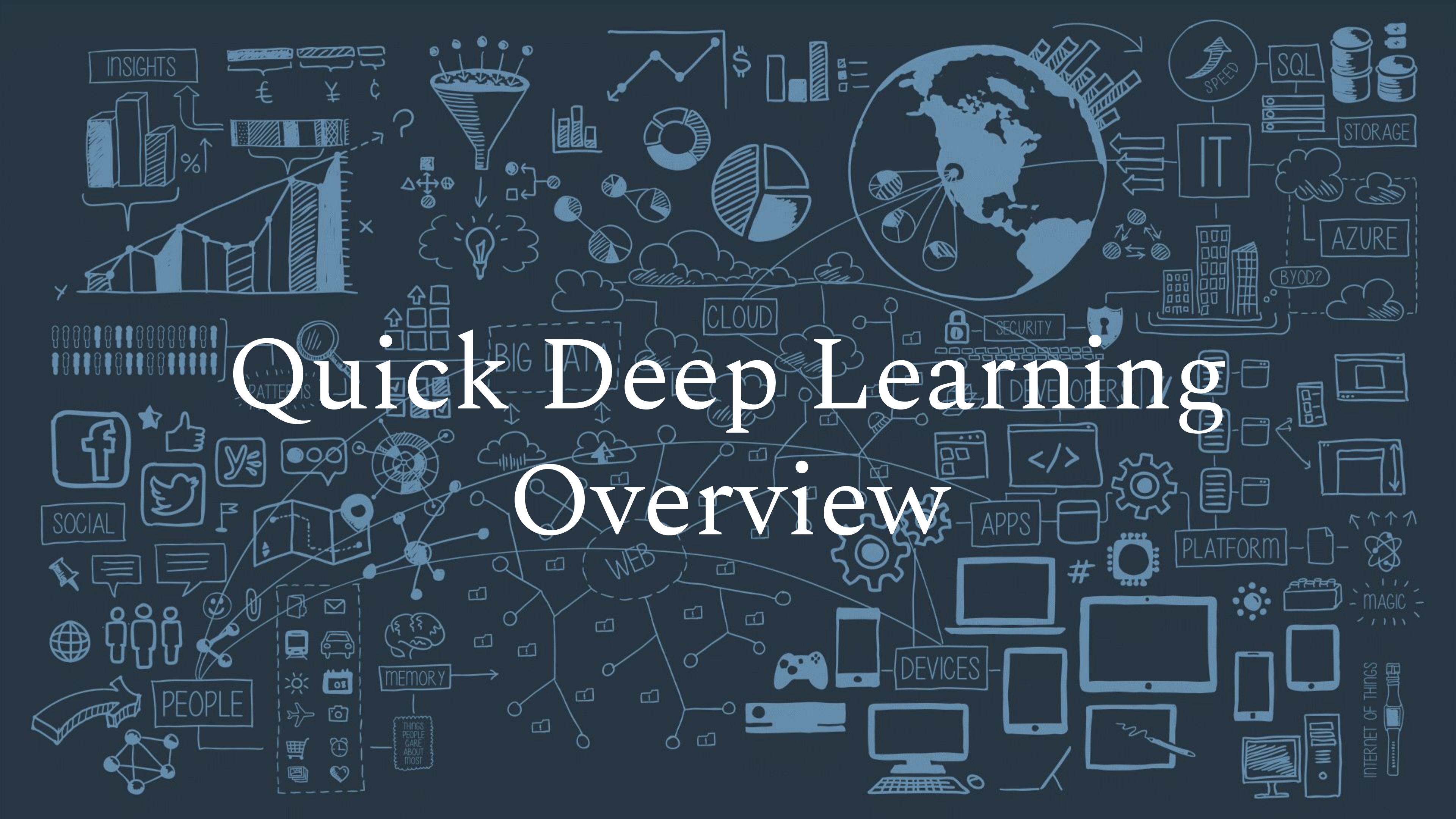
Naren Krishna
ML@B VP of Resources
EECS
narenk [at] berkeley.edu

Gokul Swamy
ML@B Project Manager
EECS
gokul.swamy [at] berkeley.edu

What we will be building

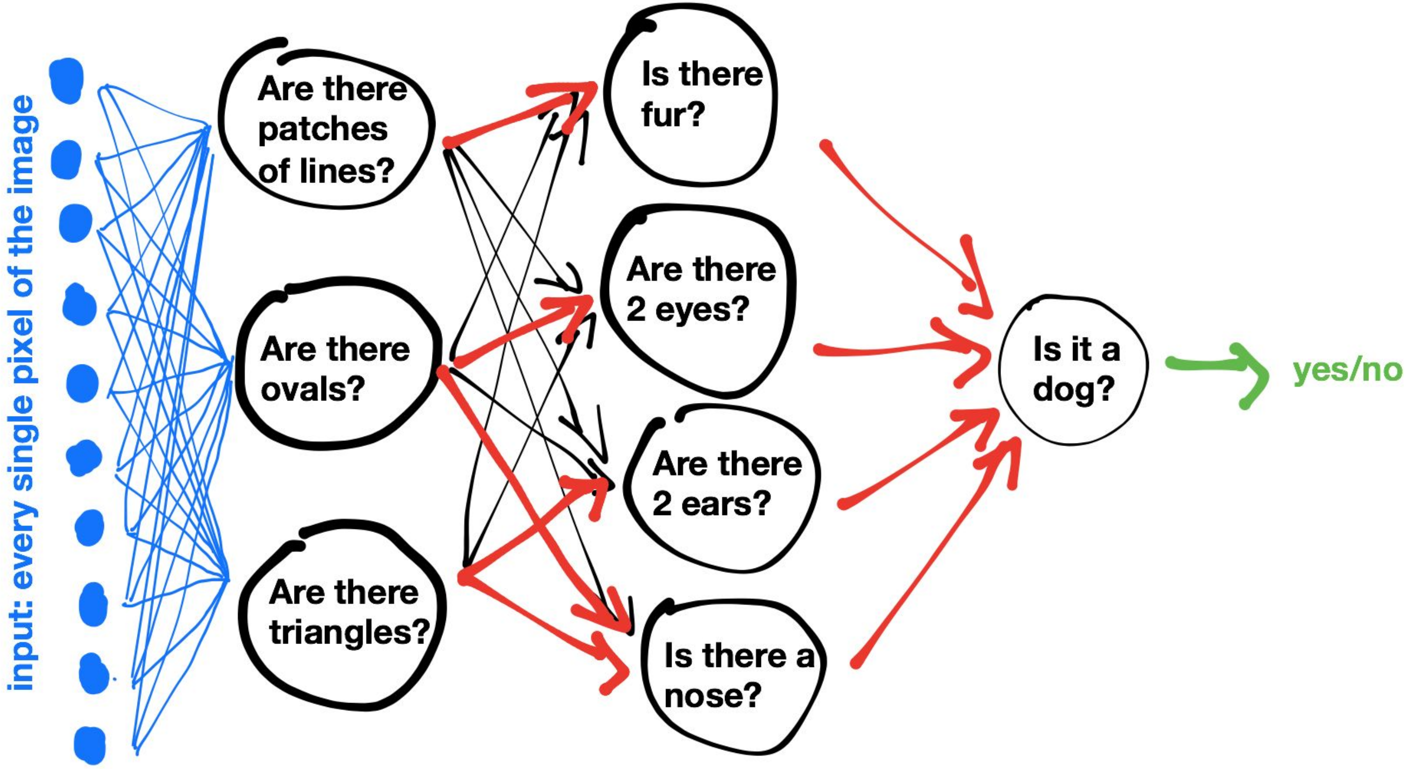
Agenda

- Deep Learning Overview
- Principles of Mobile ML
- Swift Overview
- CoreML
- Building a Mobile ML System

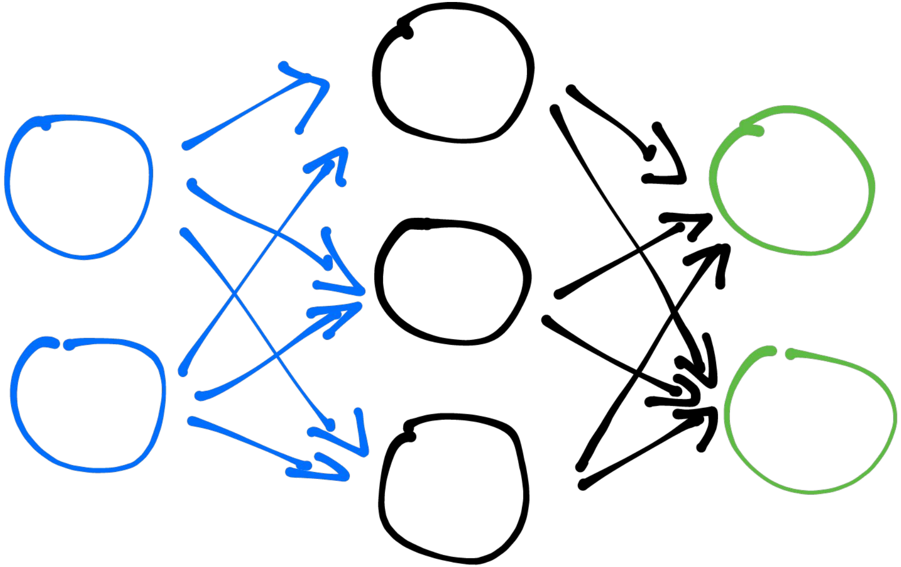


Quick Deep Learning Overview

Why are Neural Nets so Powerful?



Structure of Basic Feedforward Network

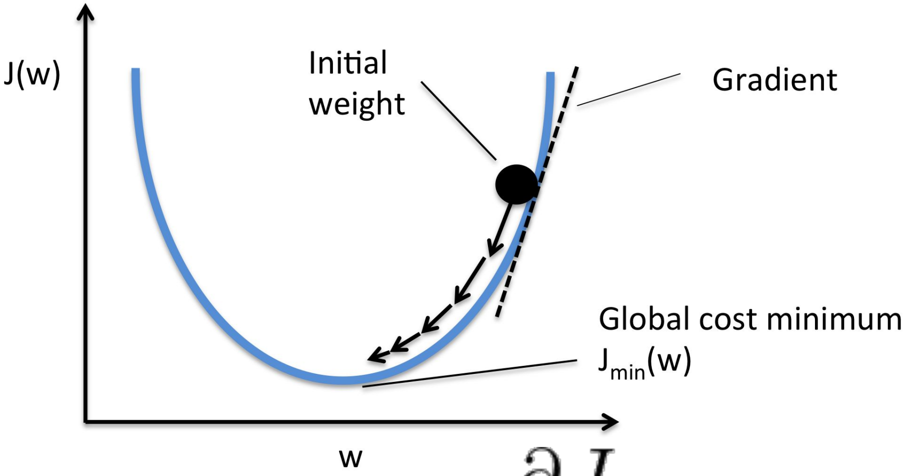


Input layer Hidden layer Output layer

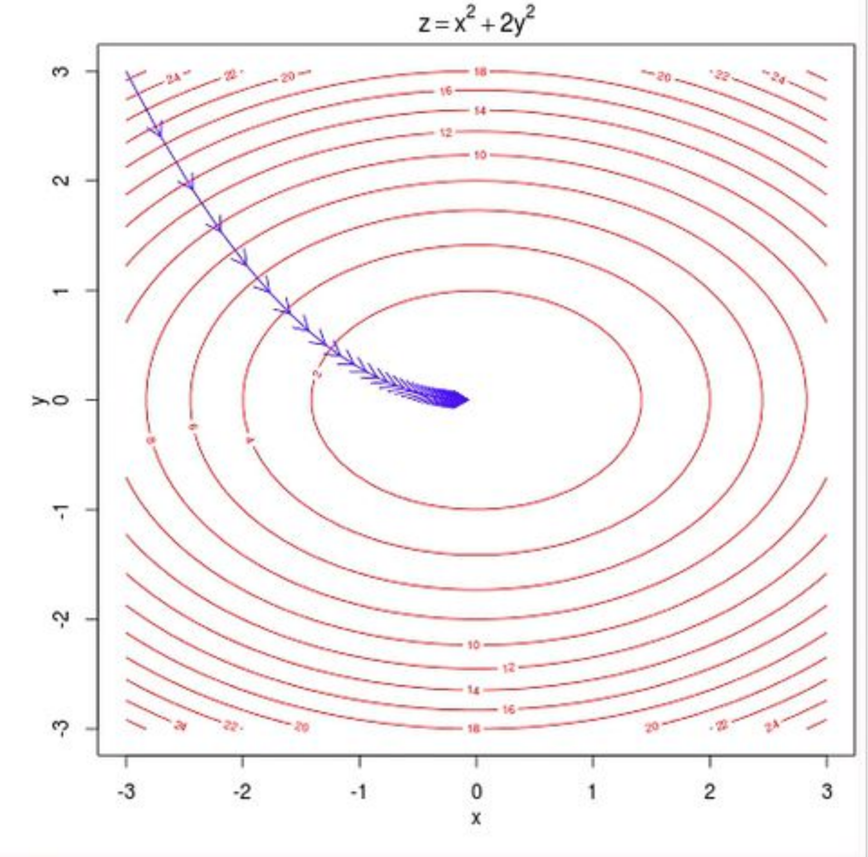
$$\hat{y} = f(\vec{x}) = \sigma\left(W^{(2)}\sigma\left(W^{(1)}\vec{x} + \beta^{(1))}\right) + \beta^{(2))}\right)$$

Gradient Descent

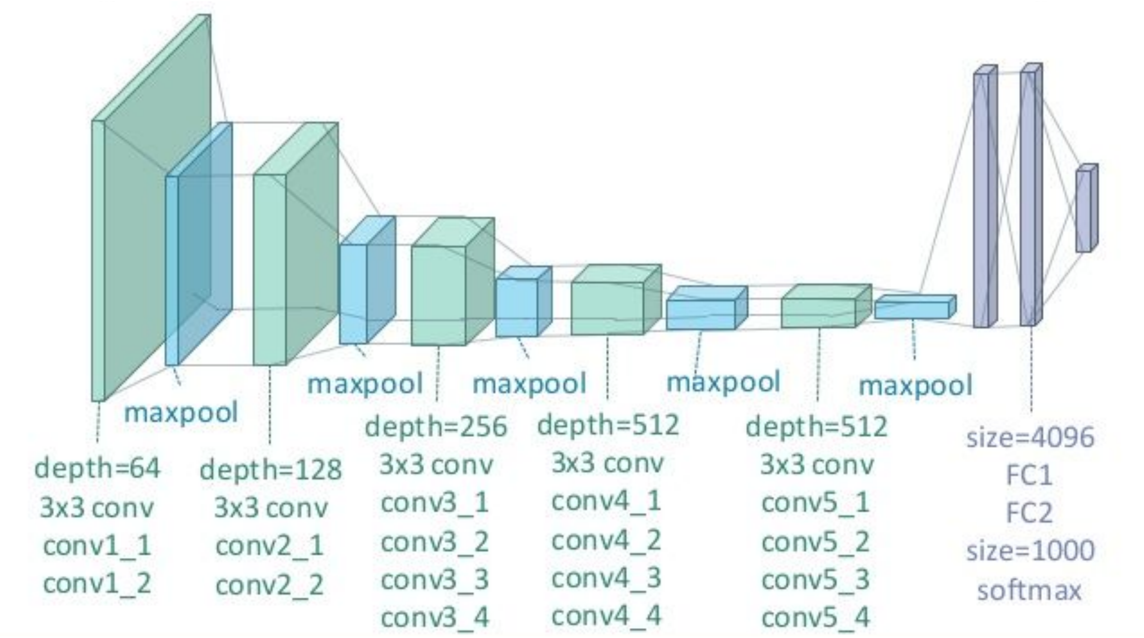
Define the cost function J(w) to be the average of the L(f(x), y) over all the training examples x. Note that the cost function J(w) is a function of the weights of the network.



$$\vec{w} \leftarrow \vec{w} - \eta \frac{\partial J}{\partial \vec{w}}$$



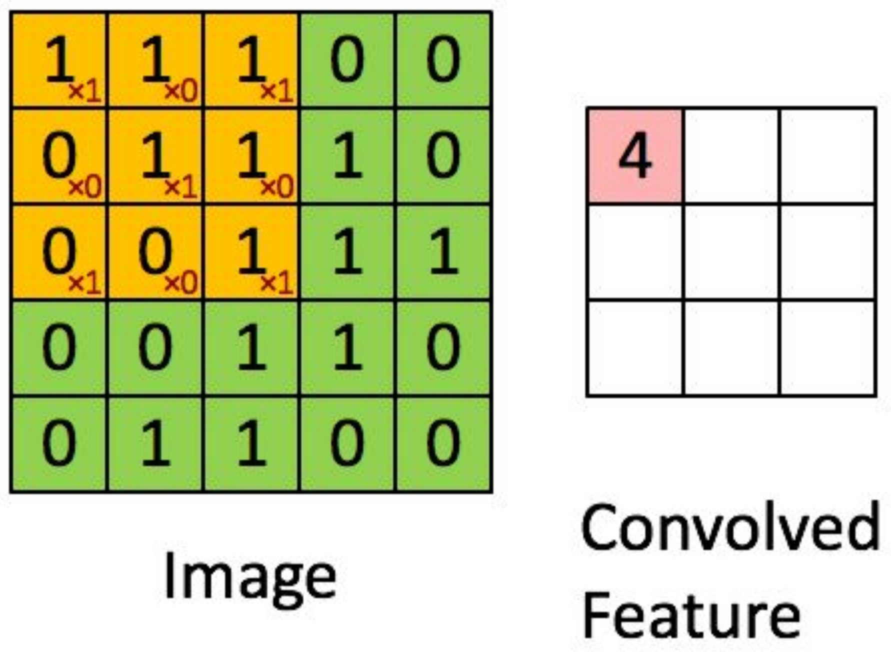
Convolutional Neural Networks



Convolutions

Weight Filter

1	0	1
0	1	0
1	0	1



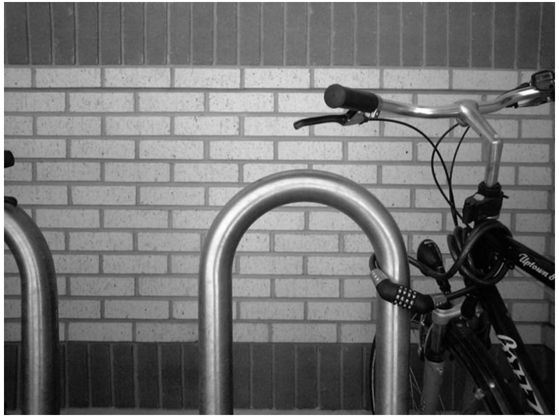
1 ×1	1 ×0	1 ×1	0	0
0 ×0	1 ×1	1 ×0	1	0
0 ×1	0 ×0	1 ×1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved Feature

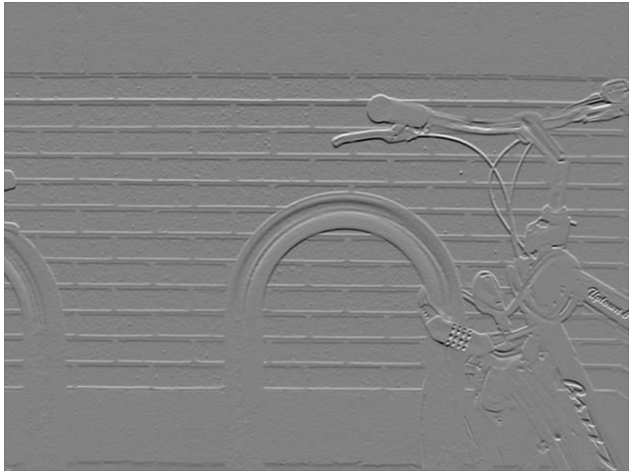
Convolutions



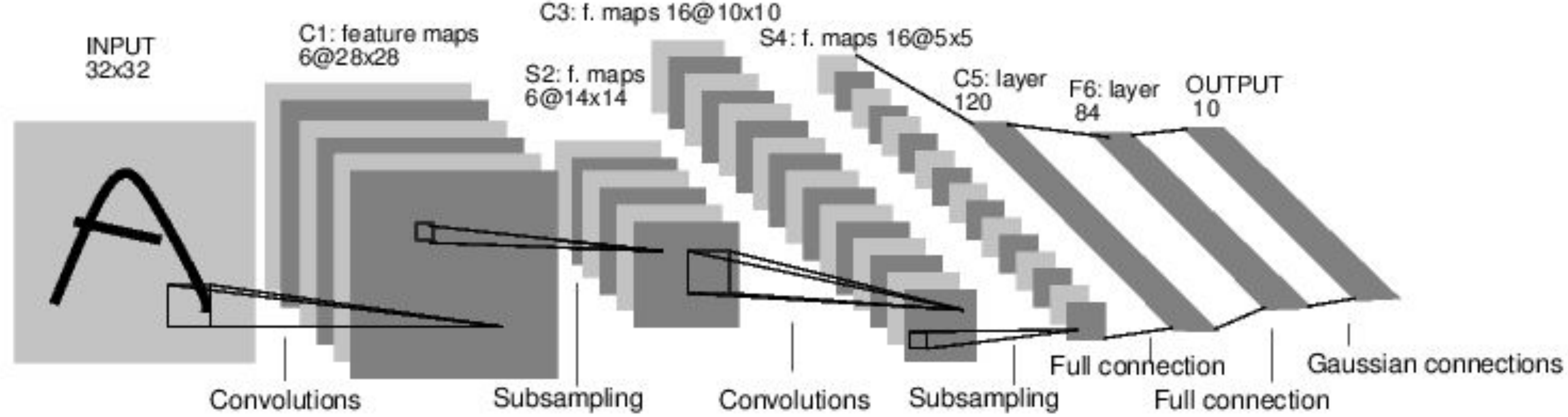
*

1	2	1
0	0	0
-1	-2	-1

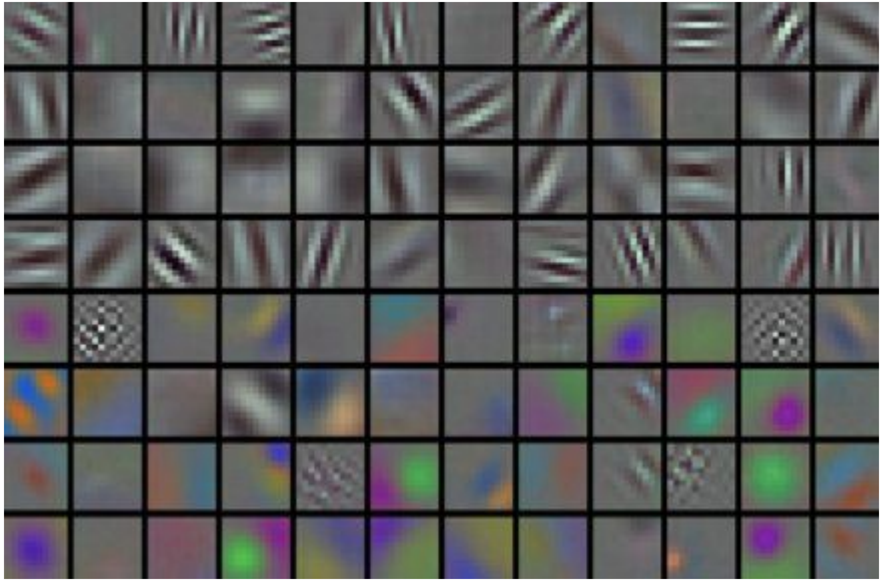
—



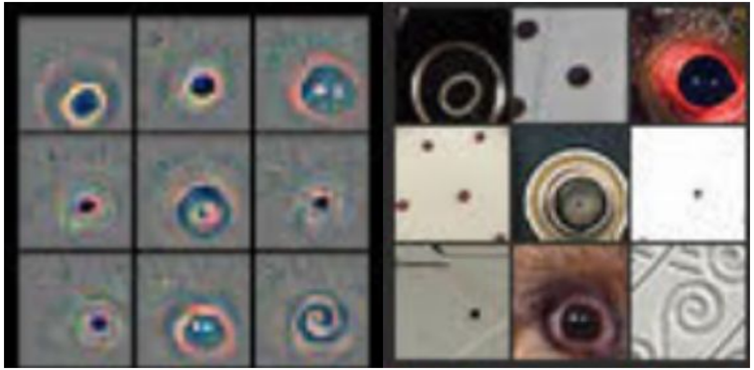
Convolutional Neural Networks



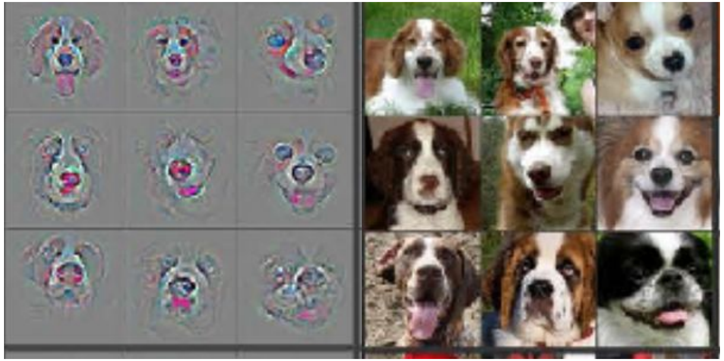
Early Layers Learn Edge Detectors



Later Layers Build Interesting Abstractions

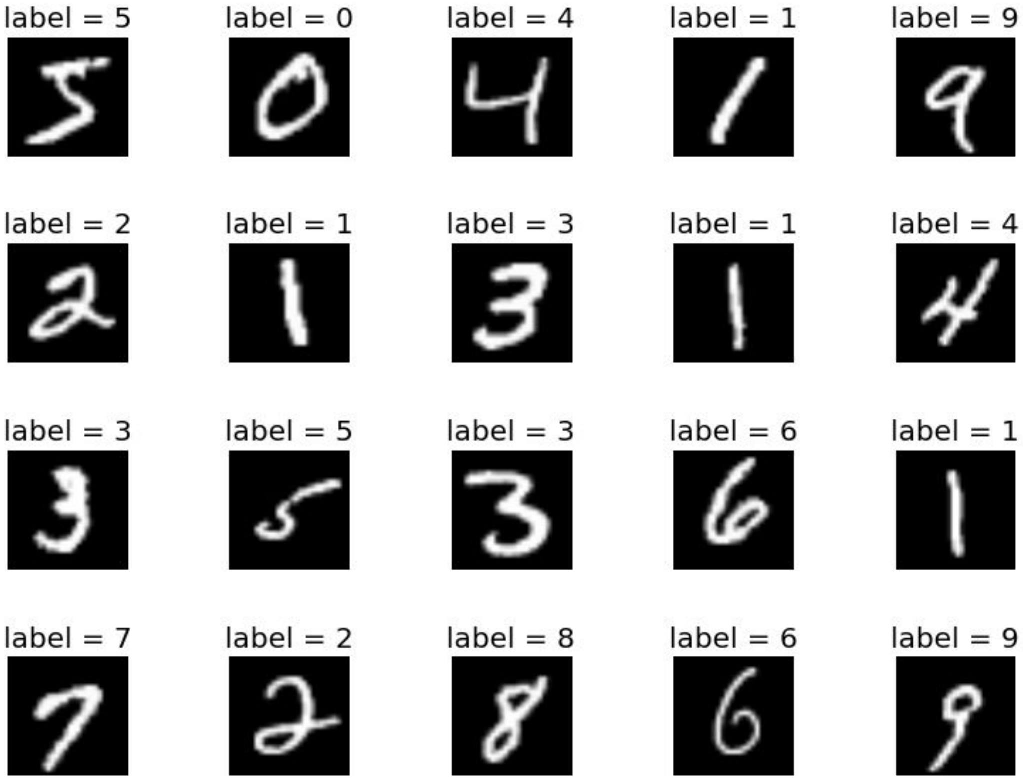


Layer 2 of AlexNet



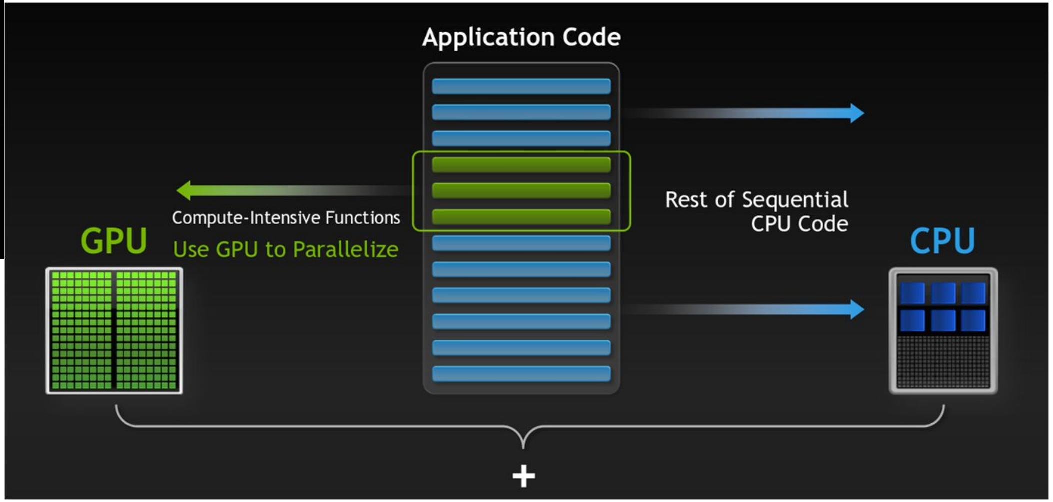
Layer 4 of AlexNet

Deep Learning for MNIST Digit Classification



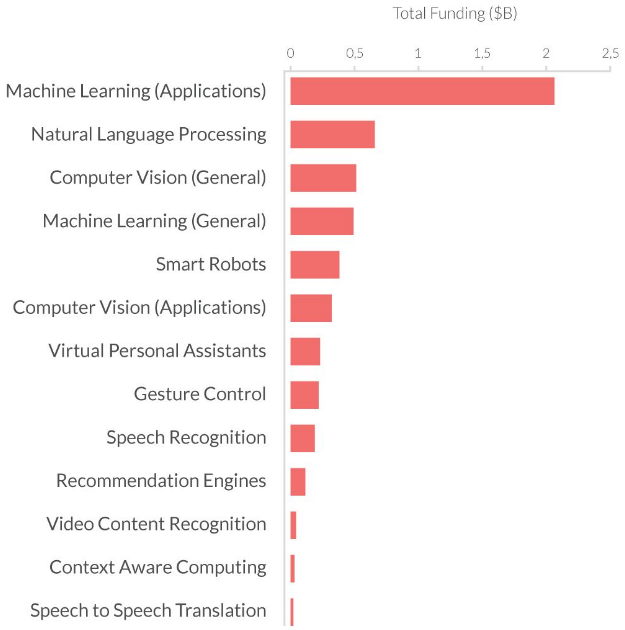
High Level Mobile Principles

How does Deep Learning leverage computational resources?



Why Mobile?

FUNDING BY ARTIFICIAL INTELLIGENCE CATEGORY

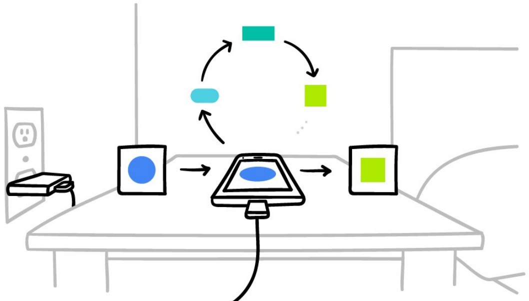


Challenges for Mobile ML

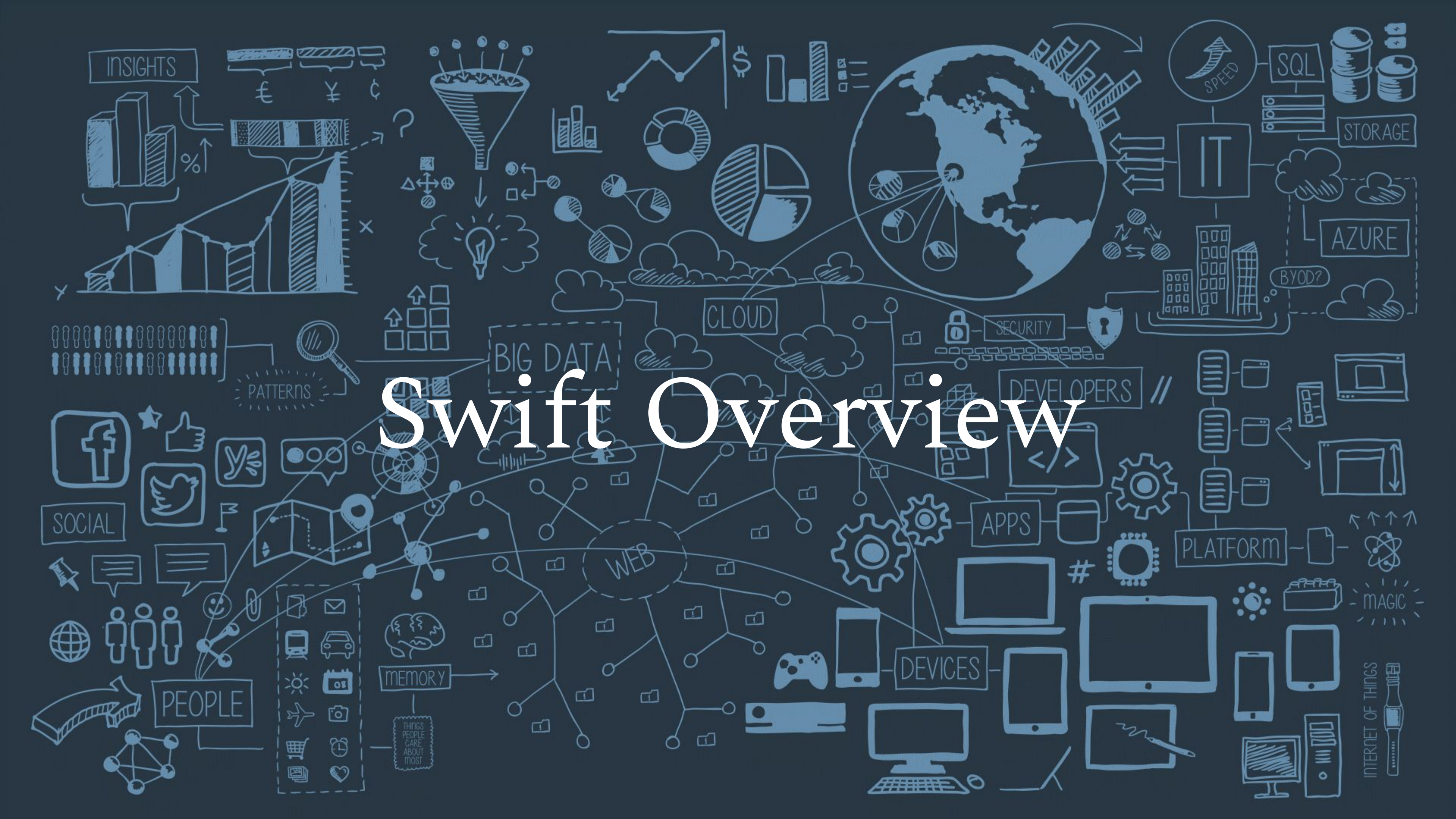
- Computational resource-limited environment
- Restrictions on amount of power we can consume
- Cannot assume connectivity to the internet or power source
- Very sensitive and personal data on people’s phones.
- Data is not i.i.d (different amounts and sources of data)

Strategies to Deal With Challenges

- Balance between training on device (better for privacy) and training on a server (better for aggregation and compute resources)
- If training on device, be sure to only do it when user has enough battery and computational resources (for example, plugged in overnight)
- If training on server, be sure to anonymize data in some way (look into differential privacy if curious) and not wait on disconnected users to deliver data
- Federated Learning: Training on mobile and using a weighted average to update model on server



Swift Overview



What is Swift?

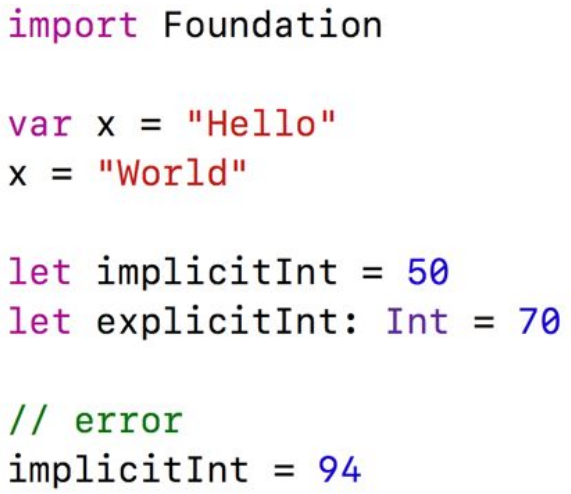
- Swift is the language currently used to develop iOS Apps
- ~10th most popular programming language
- Created by Apple, open-sourced
- Built on top of speedy LLVM (low-level virtual machine) compiler
- Supports Python-like REPL and runs on Linux

Constants/Variables

- Use **let** to declare a constant, **var** to declare a variable

- // to comment a line

- Swift will infer a type but you can explicitly define a type with a colon following the variable name



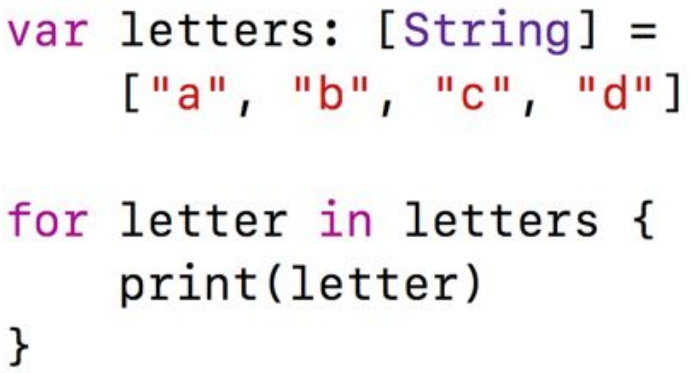
```
import Foundation

var x = "Hello"
x = "World"

let implicitInt = 50
let explicitInt: Int = 70

// error
implicitInt = 94
```

Arrays

- Specified using square brackets around type

- Do not have a fixed size

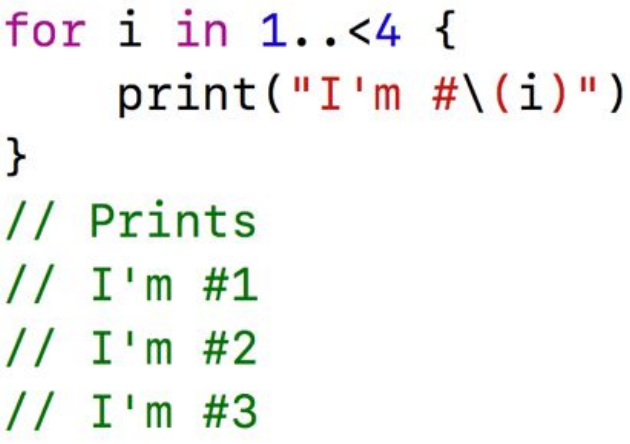
- Use .**append** or **+=** to add elements
  Use .**count** to get size of array
  Swift supports enhanced for loops



```
var letters: [String] =
    ["a", "b", "c", "d"]

for letter in letters {
    print(letter)
}
```

Control Flow

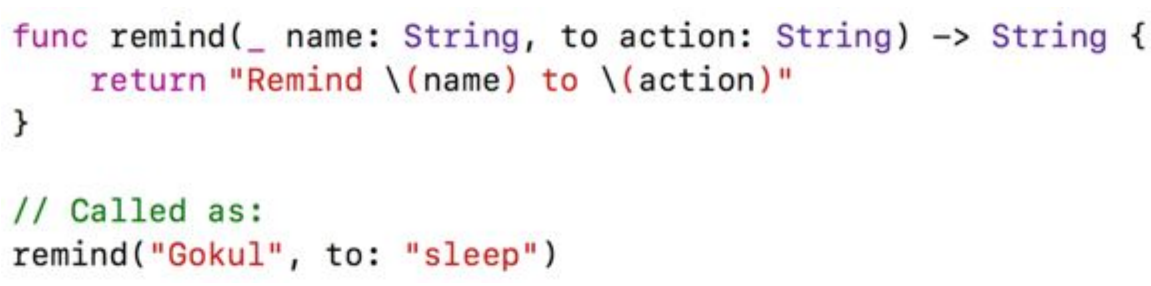
- Use **…** to specify an inclusive interval
- Use **..<** to specify an upper-bound exclusive interval
- Also, "…**\(var_name)**…" inserts variable into string



```
for i in 1..<4 {
    print("I'm #\(i)")
}
// Prints
// I'm #1
// I'm #2
// I'm #3
```

Functions

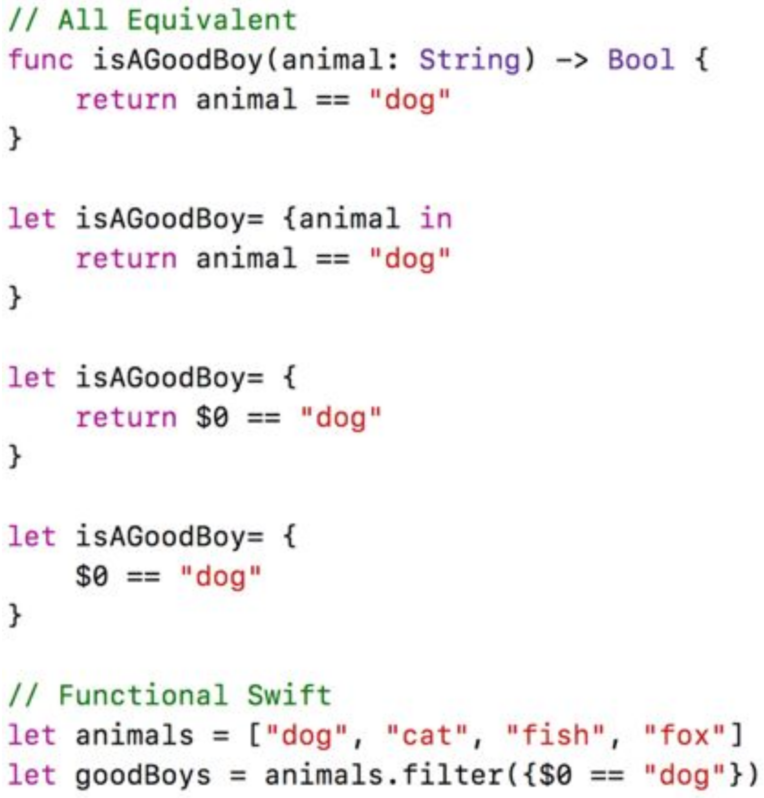
- Explicit parameter and return types
- Interior and exterior parameter names



```swift
func remind(_ name: String, to action: String) -> String {
    return "Remind \(name) to \(action)"
}

// Called as:
remind("Gokul", to: "sleep")
```

Closures

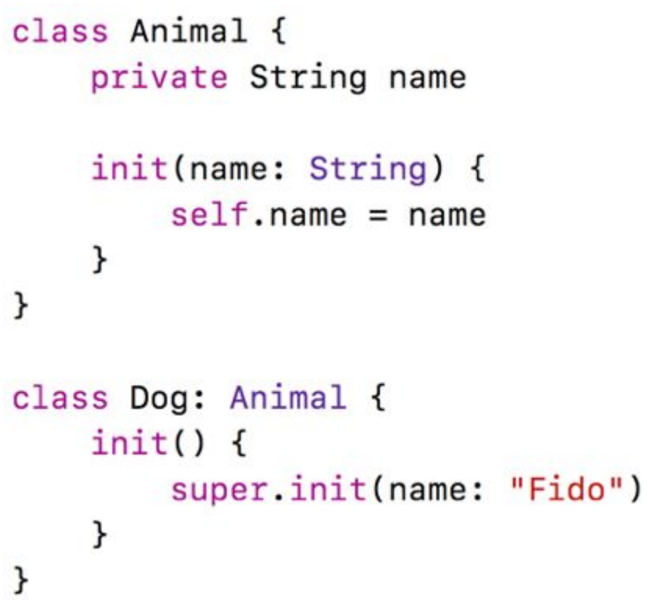
- Unnamed functions that can be passed as arguments to other functions
- Used for functional programming



```
// All Equivalent
func isAGoodBoy(animal: String) -> Bool {
    return animal == "dog"
}

let isAGoodBoy= {animal in
    return animal == "dog"
}

let isAGoodBoy= {
    return $0 == "dog"
}

let isAGoodBoy= {
    $0 == "dog"
}

// Functional Swift
let animals = ["dog", "cat", "fish", "fox"]
let goodBoys = animals.filter({$0 == "dog"})
```

Classes

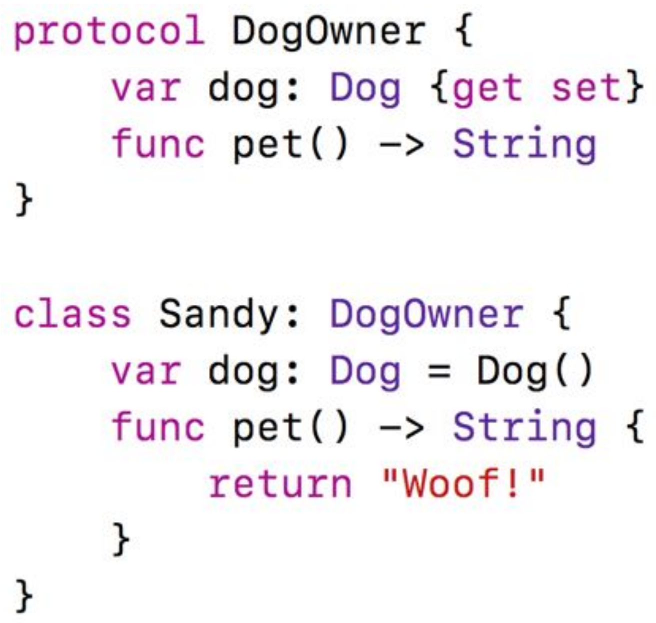
- Use **self** to refer to current object
- Use **super** to reference superclass
- Use a colon to specify subclassing



```
class Animal {
    private String name

    init(name: String) {
        self.name = name
    }
}

class Dog: Animal {
    init() {
        super.init(name: "Fido")
    }
}
```

Protocols

- Similar to interfaces in other languages
- Can specify variables and functions
- Can specify whether variable should be gettable or settable
- Same colon syntax as subclassing



```
protocol DogOwner {
    var dog: Dog {get set}
    func pet() -> String
}

class Sandy: DogOwner {
    var dog: Dog = Dog()
    func pet() -> String {
        return "Woof!"
    }
}
```

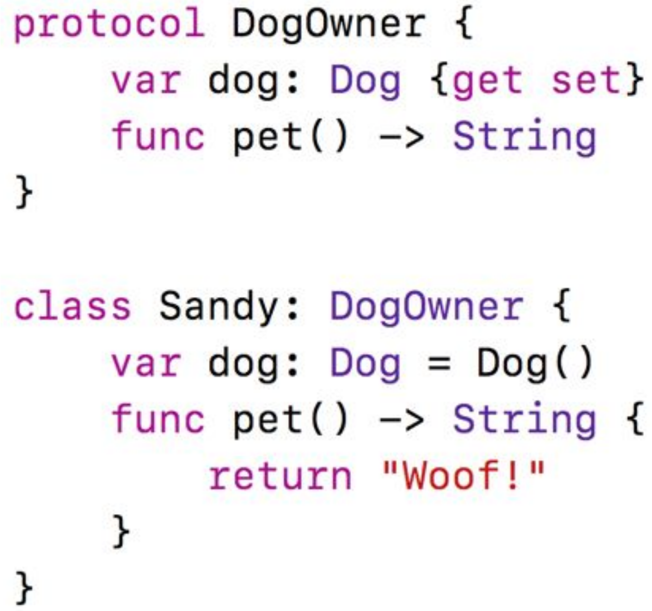
Protocols

- Similar to interfaces in other languages

- Can specify variables and functions

- Can specify whether variable should be gettable or settable

- Same colon syntax as subclassing

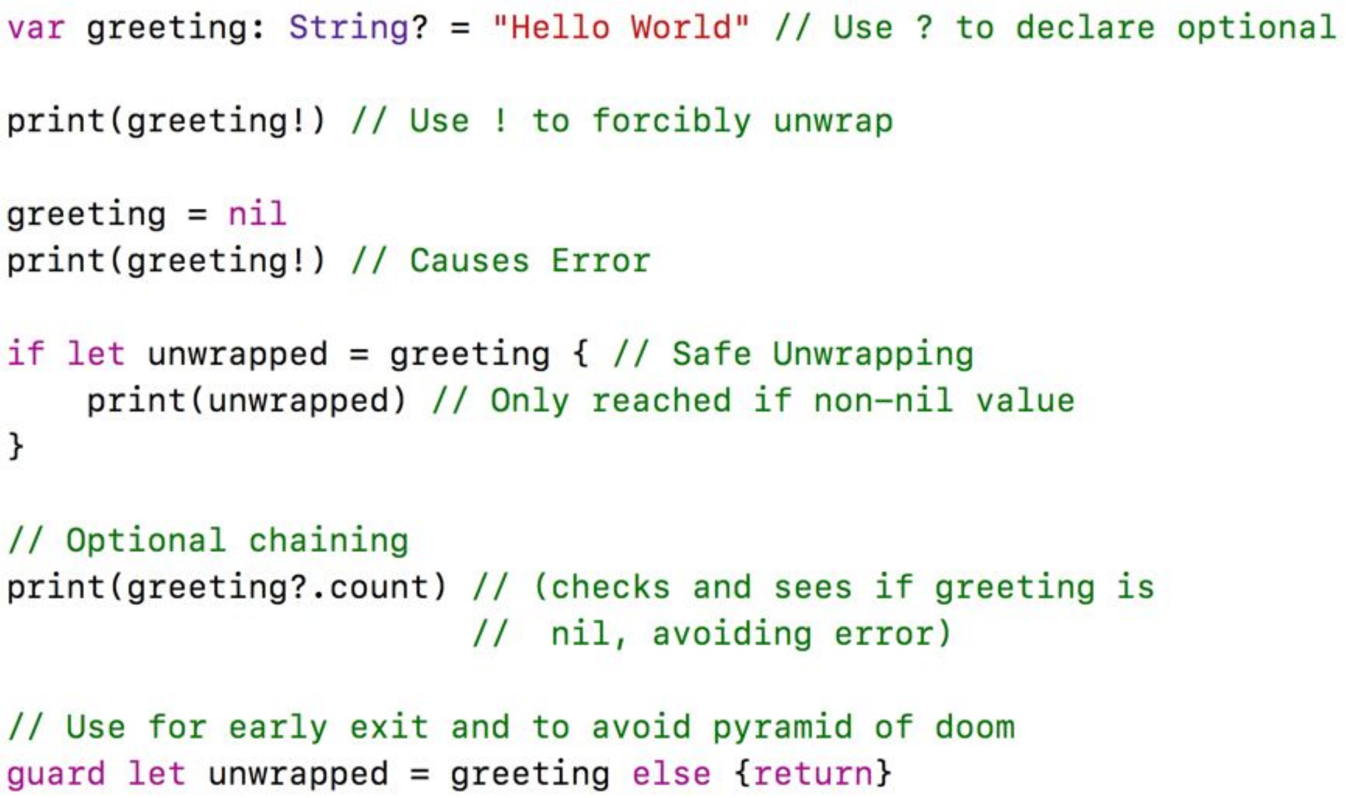


```
protocol DogOwner {
    var dog: Dog {get set}
    func pet() -> String
}

class Sandy: DogOwner {
    var dog: Dog = Dog()
    func pet() -> String {
        return "Woof!"
    }
}
```

Optionals

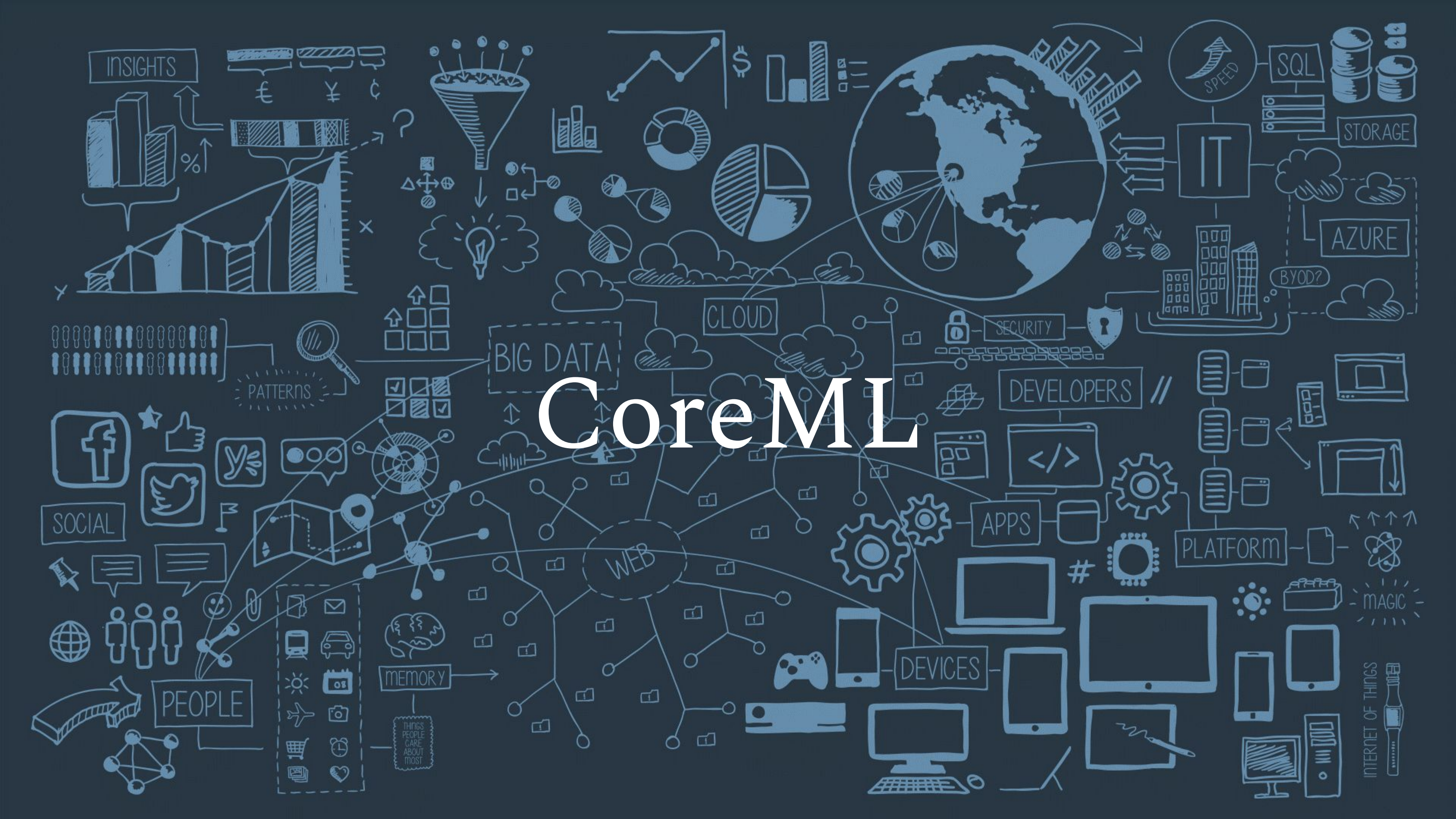
- One of the signature features (and sometimes headaches) of Swift
- A type that indicates that an object of another (non-optional) type exists or a nil value
- Unwrap to get at data that is potentially contained

Optionals: Syntax



```
var greeting: String? = "Hello World" // Use ? to declare optional

print(greeting!) // Use ! to forcibly unwrap

greeting = nil
print(greeting!) // Causes Error

if let unwrapped = greeting { // Safe Unwrapping
    print(unwrapped) // Only reached if non-nil value
}

// Optional chaining
print(greeting?.count) // (checks and sees if greeting is
                       //  nil, avoiding error)

// Use for early exit and to avoid pyramid of doom
guard let unwrapped = greeting else {return}
```

CoreML


INSIGHTS
%
€ ¥ ¢
$
SQL
SPEED
STORAGE
IT
AZURE
BYOD?
CLOUD
SECURITY
BIG DATA
PATTERNS
DEVELOPERS
SOCIAL
APPS
PLATFORM
WEB
MAGIC
MEMORY
THINGS PEOPLE CARE ABOUT MOST
PEOPLE
DEVICES
INTERNET OF THINGS

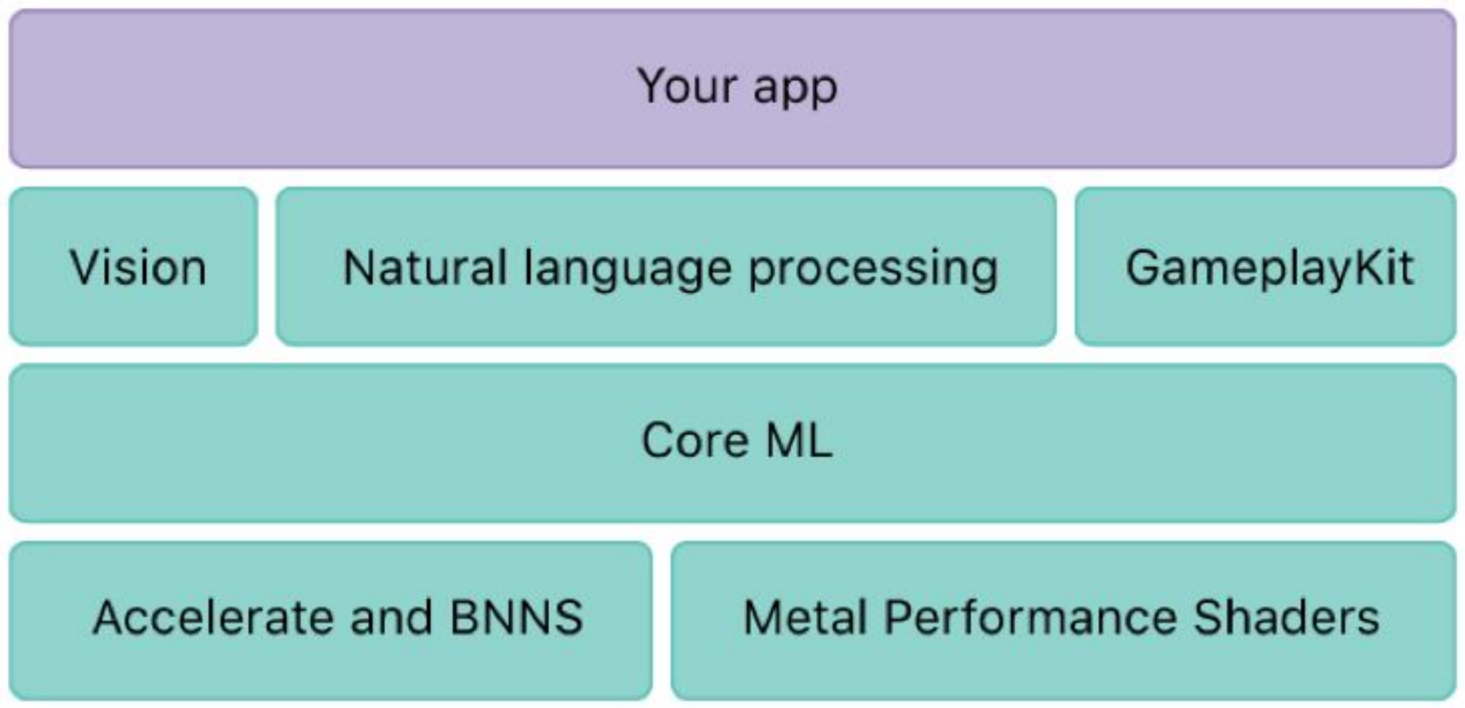
What is CoreML?

- CoreML released at WWDC 2017
- Represents a shift in traditional developer model
    - Researcher creates model
    - Apple takes care of hard part of implementation
    - You just have to connect the above

CoreML

- Lets you convert ML code written in Python into a **.mlmodel** file can be dropped directly into your app
- Hard part: optimizes code for iOS devices
- Some helpful preprocessing included
- Does not currently support on-device training

Architecture



Your app

Vision

Natural language processing

GameplayKit

Core ML

Accelerate and BNNS

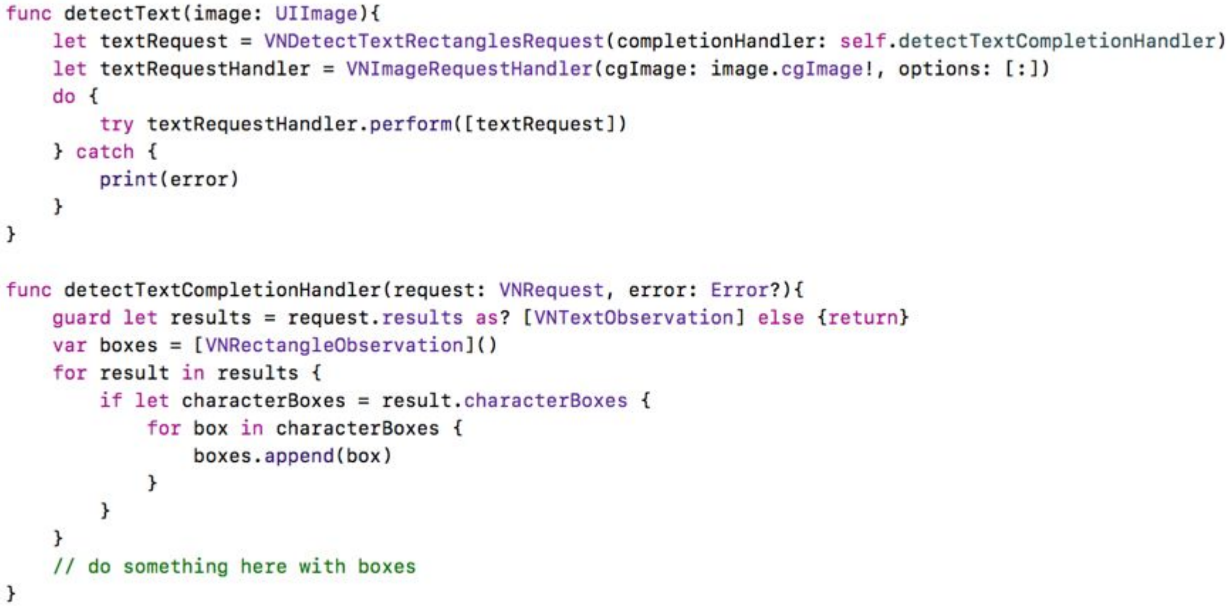
Metal Performance Shaders

Vision

- Allows you to detect face landmarks, perform OCR, scan barcodes, perform optical flow
- Slower but more accurate than CoreImage and AVFoundation
- Three Steps
  1. Call **VNRequestHandler**
  2. Which executes a **VNRequest**
  3. And returns some **VNObservation** for you to process

Vision: Code

- Be sure to add Camera Usage Description to Info.plist and import Vision



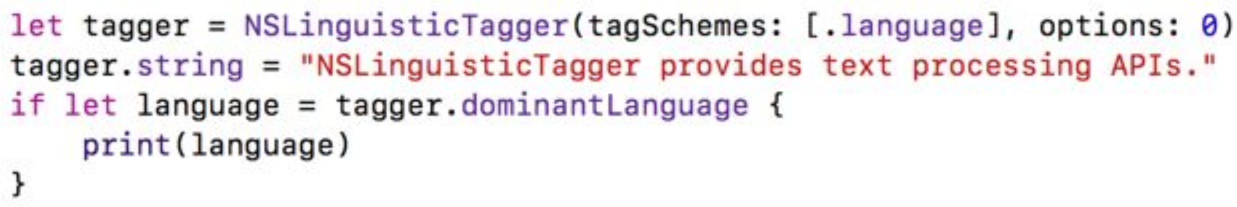
```swift
func detectText(image: UIImage){
    let textRequest = VNDetectTextRectanglesRequest(completionHandler: self.detectTextCompletionHandler)
    let textRequestHandler = VNImageRequestHandler(cgImage: image.cgImage!, options: [:])
    do {
        try textRequestHandler.perform([textRequest])
    } catch {
        print(error)
    }
}

func detectTextCompletionHandler(request: VNRequest, error: Error?){
    guard let results = request.results as? [VNTextObservation] else {return}
    var boxes = [VNRectangleObservation]()
    for result in results {
        if let characterBoxes = result.characterBoxes {
            for box in characterBoxes {
                boxes.append(box)
            }
        }
    }
    // do something here with boxes
}
```

NSLinguisticTagger

- Part of NS but revamped with deep learning for 2017
- On device processing so no privacy worries
- Can identify language, tokenize (split up into chunks), lemmatize (give root form of word), and detect named entities (rigid designators of concepts that may be important)
- Two Steps
  - Create **NSLinguisticTagger** with tagSchemes set to application
  - Set **tagger.string** to the text to be analyzed.

NSLinguisticTagger: Code

- For example, to detect dominant language:



```
let tagger = NSLinguisticTagger(tagSchemes: [.language], options: 0)
tagger.string = "NSLinguisticTagger provides text processing APIs."
if let language = tagger.dominantLanguage {
    print(language)
}
```

- For all use case sample code, check out this link

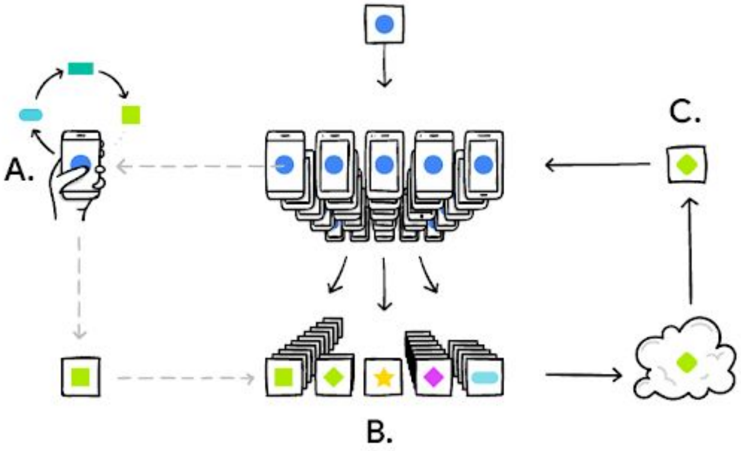
Building a Mobile ML System

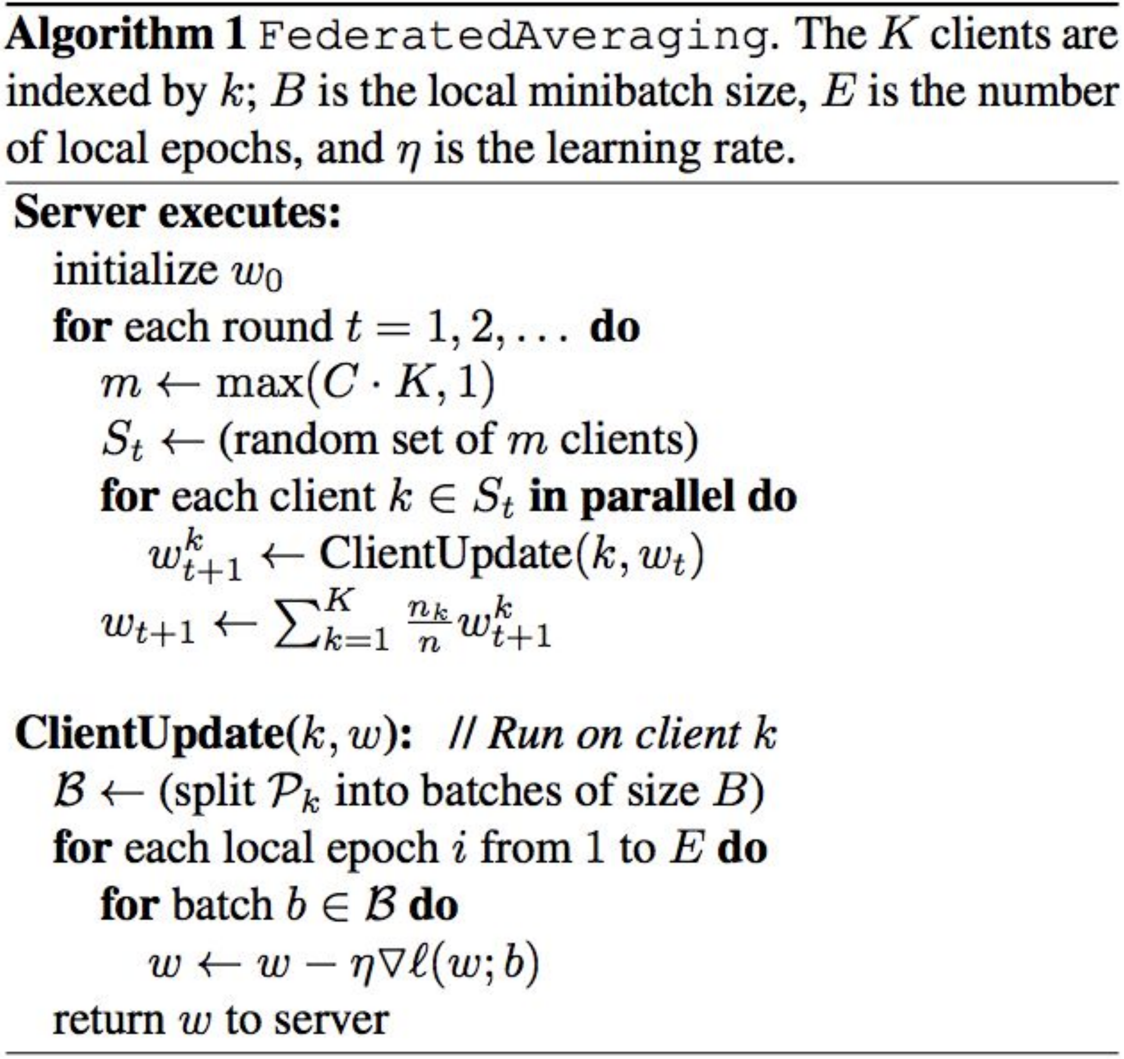
Setup

- We’ll be using Python 3.5
- `git clone https://github.com/mlberkeley/mobile_ml_workshop`
- `pip install numpy keras tensorflow`
- Download Xcode from the Mac App Store
- `pip install h5py coremltools`
- `xcode-select --install`

Federated Learning

- Compute gradients based on individual user data
- Send gradients to server and update model by averaging gradients across users
- Maintains user data privacy

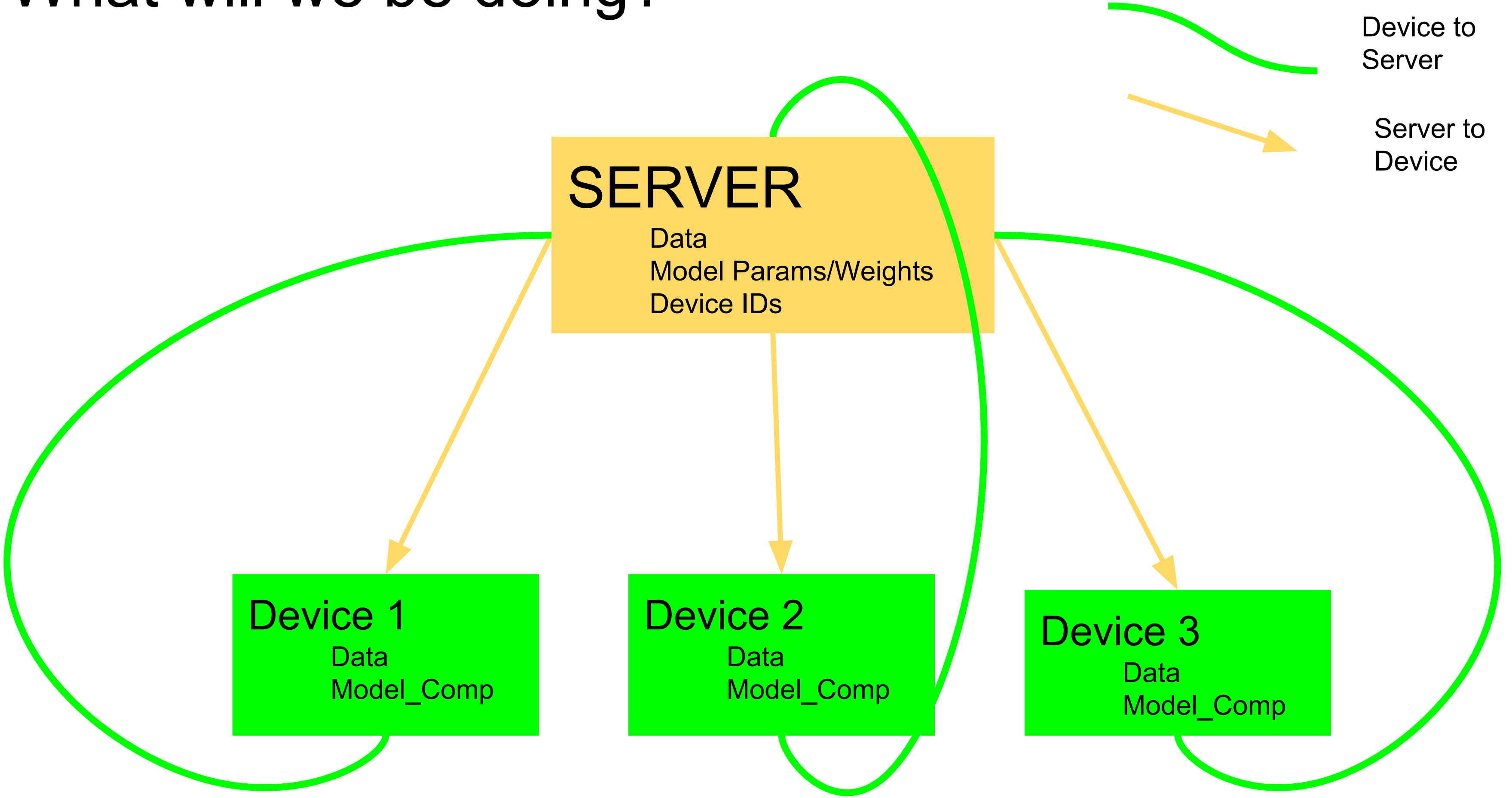




Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

```
Server executes:
  initialize $w_0$
  for each round $t = 1, 2, \dots$ do
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of $m$ clients)
    for each client $k \in S_t$ in parallel do
      $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate$(k, w)$:   // Run on client k
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)
  for each local epoch $i$ from 1 to $E$ do
    for batch $b \in \mathcal{B}$ do
      $w \leftarrow w - \eta \nabla \ell(w; b)$
  return $w$ to server
```

What will we be doing?



Device to Server

Server to Device

SERVER

Data

Model Params/Weights

Device IDs

Device 1

Data

Model_Comp

Device 2

Data

Model_Comp

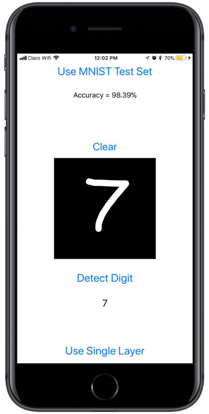
Device 3

Data

Model_Comp

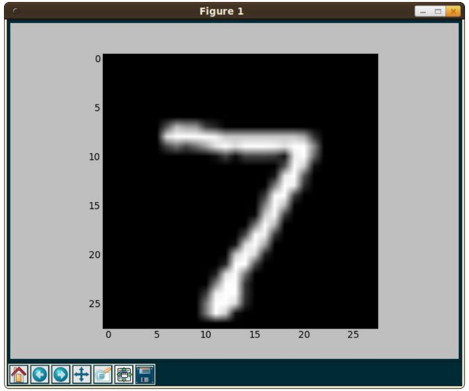
How can we make our model more accurate?

- Do preprocessing on MNIST images to make sure no pixels have gray values
    - Similar format as inputs from phone
- Change pen style on phone to sense pressure when writing
    - Makes digits written on phone more similar to MNIST images



Accuracy = 98.39%

7



Extra Resources

How to Deep Learn at Home

Run on CPU (really slow)

Buy a GPU (more expensive option)

Use AWS! - Here’s a Guide

(Jan. 2018) OR

Use Google Colab! (GPU guide)

Deep Learning Resources

ML@B’s Machine Learning Crash Course: Part 3

Stanford’s CS231n Convolutional Networks for Visual Recognition

Deep Learning Textbook by Goodfellow et. al.

ML@B’s Deep Learning Decal

Neural Networks and Deep Learning (Backprop section)