移动设备也能玩机器学习

杨鑫奇 @xinqiyang
億百国際株式会社

ecafe event
July 2017

Who am I?

杨鑫奇

wechat/weibo/twitter/facebook/github

@xinqiyang

- 全栈工程师 (PHP,Java,Object-C,Python,javascript,etc)
- 曾任职于百度，干过派遣，现在自己折腾创业
- 现在东京创办亿咖啡，一个华人技术交流社区
- Founder of 億百国際株式会社

事先说明

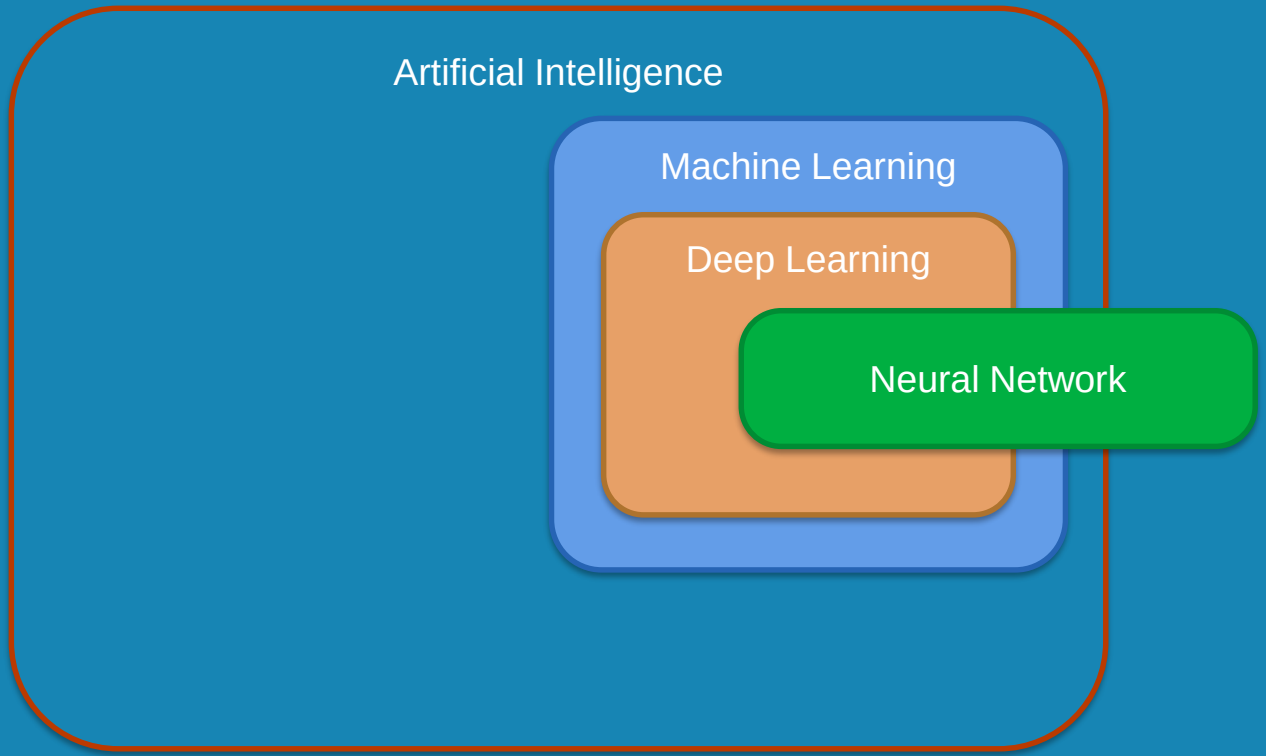
- 不是一个ML专家
- 不是资深Pythoner
- 也不太懂高深的算法
- 在ML领域涉世未深

那为什么还会来讲这一场？

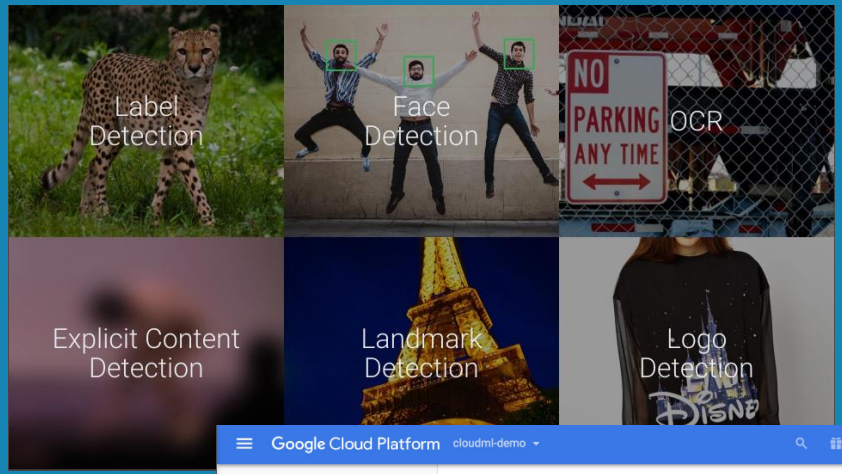
本次大纲

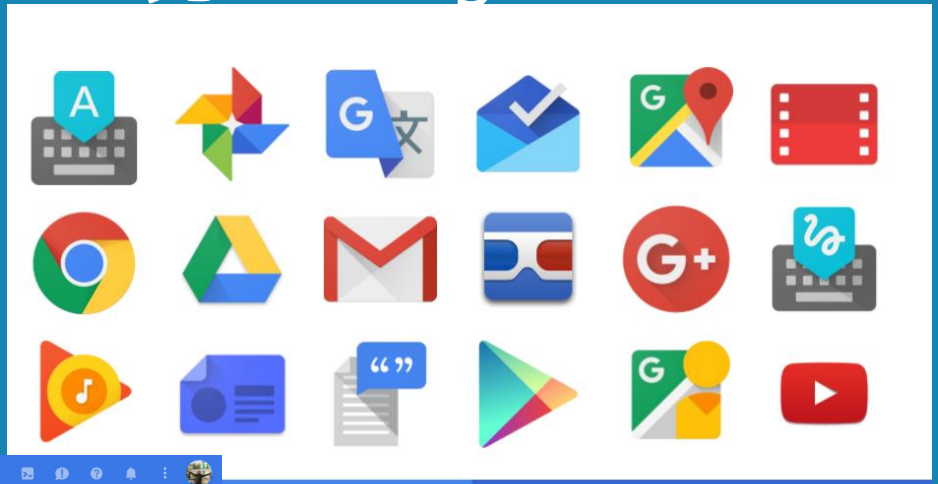
- 众观下各大厂商所提供的ML服务（ google,aws,microsoft, azure, aliyun,qcoud etc)
- 从移动方向看ML的应用场景具有极大的商业价值
- 从实际应用的角度来看Machine Learning
- 如何动手玩转ML
- 我们如何在APP中使用ML（演示）
- 如何用好ML工具来训练并使用自己的模型
- 移动设备上ML的训练和使用的未来会是什么样的？
- 提问回答 && 讨论

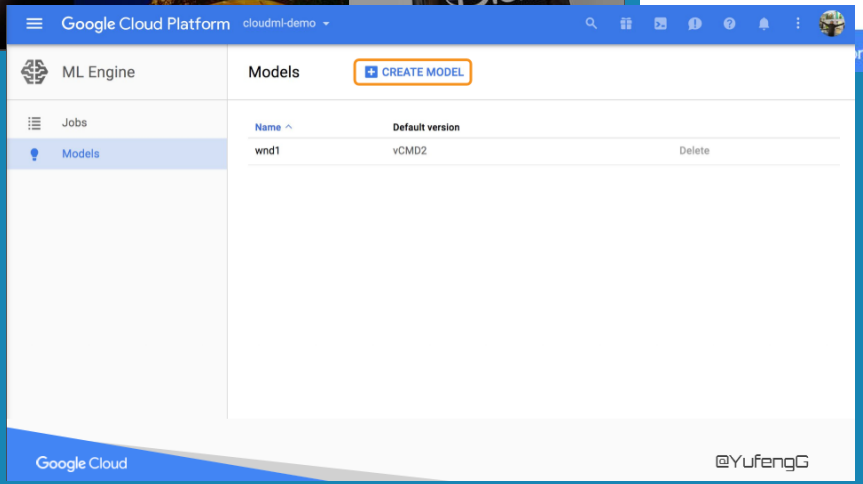
热门的热门，我们应该学什么？



各大厂商所提供的ML服务 - Google







TensorFlow

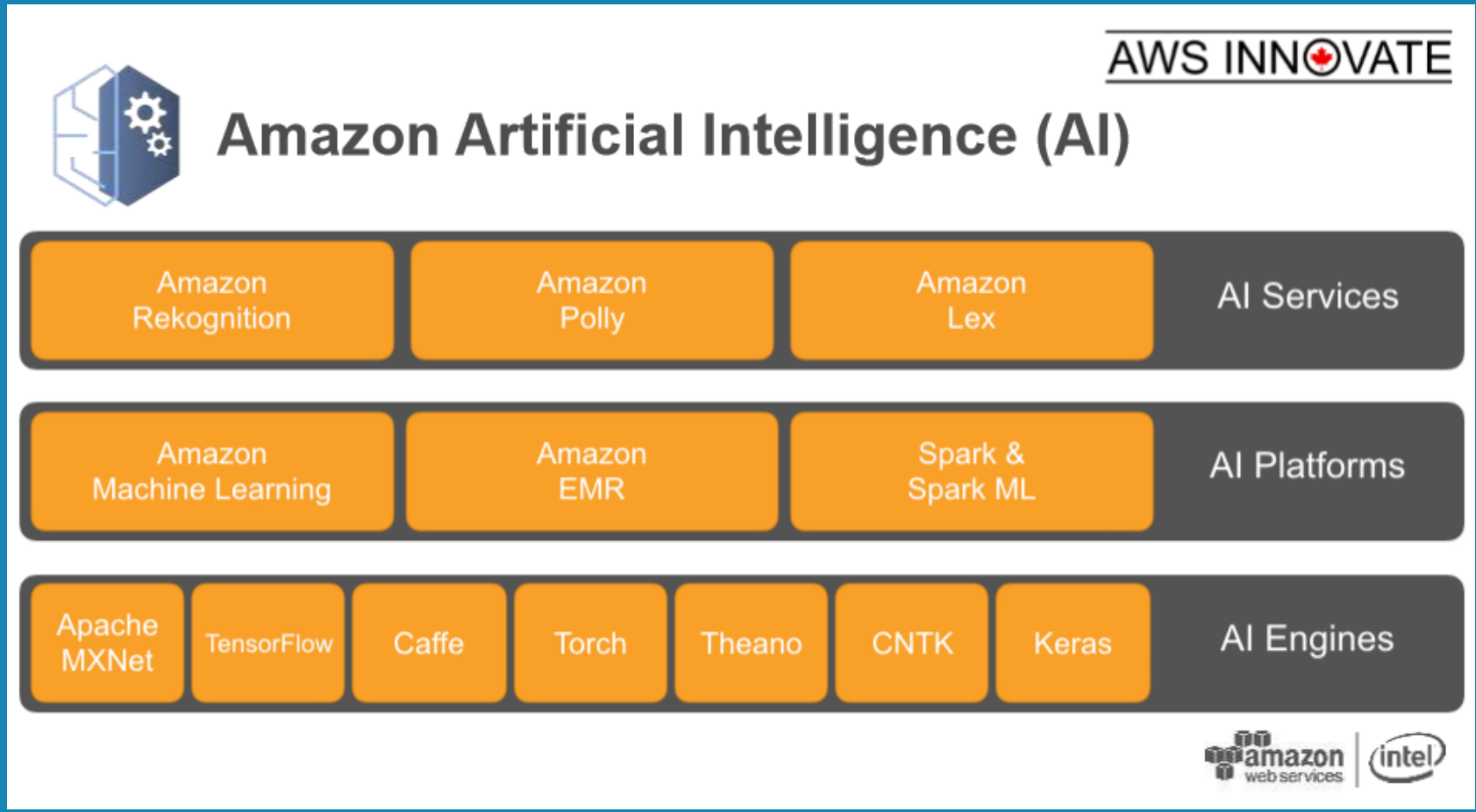
Open source Machine Learning library

Especially useful for **Deep Learning**

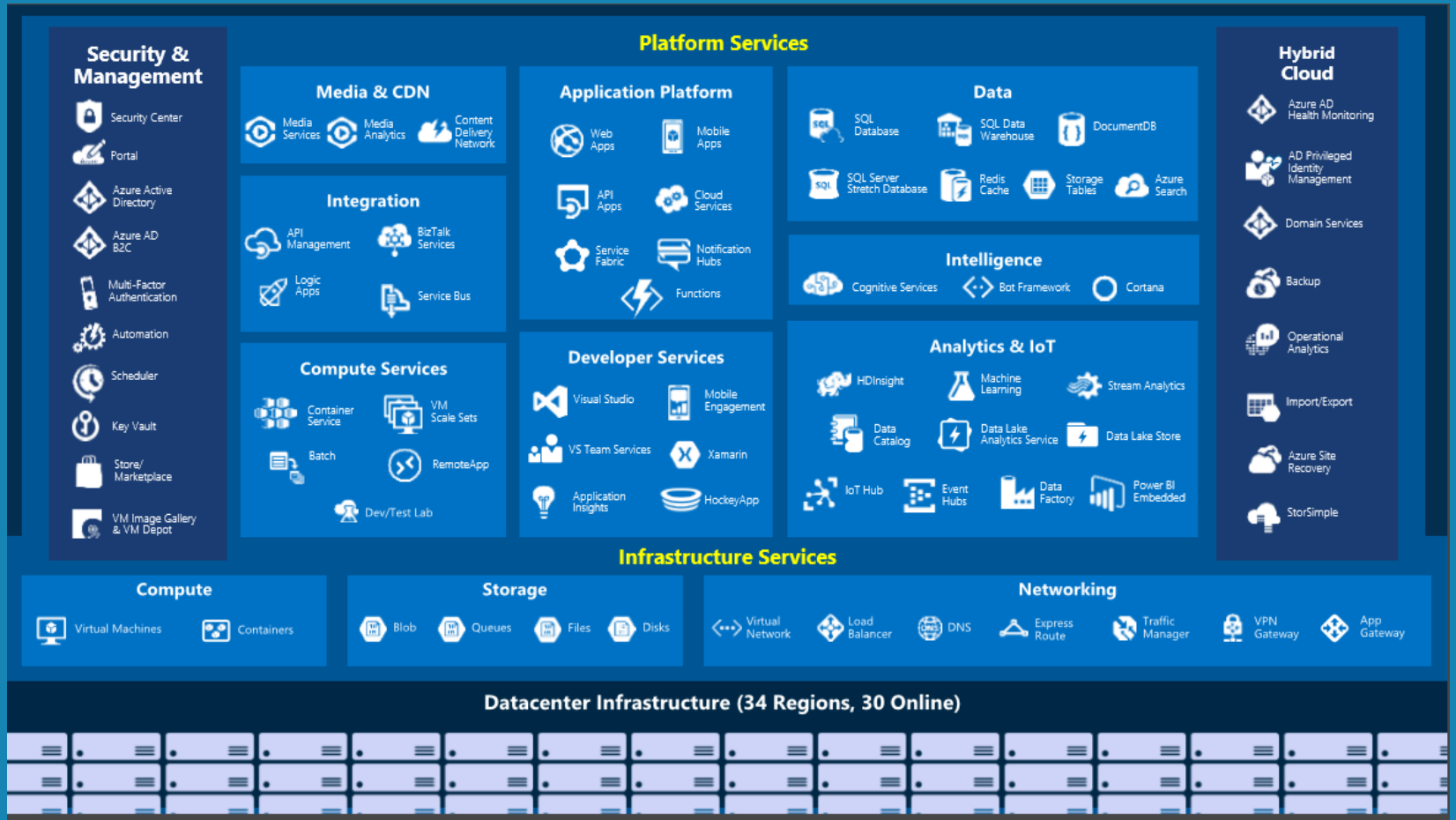
For research **and** production

Apache 2.0 license

各大厂商所提供的ML服务 - AWS



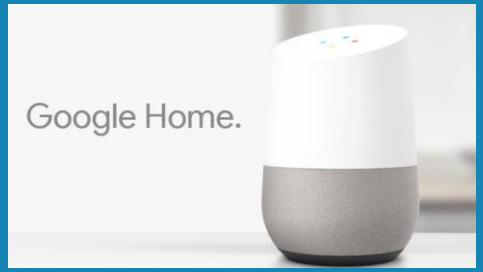
各大厂商所提供的ML服务– Microsoft Azure



各大厂商所提供的ML服务– AliYun

各大厂商所提供的ML服务 – QCloud

从移动方向看ML的应用场景具有极大的商业价值

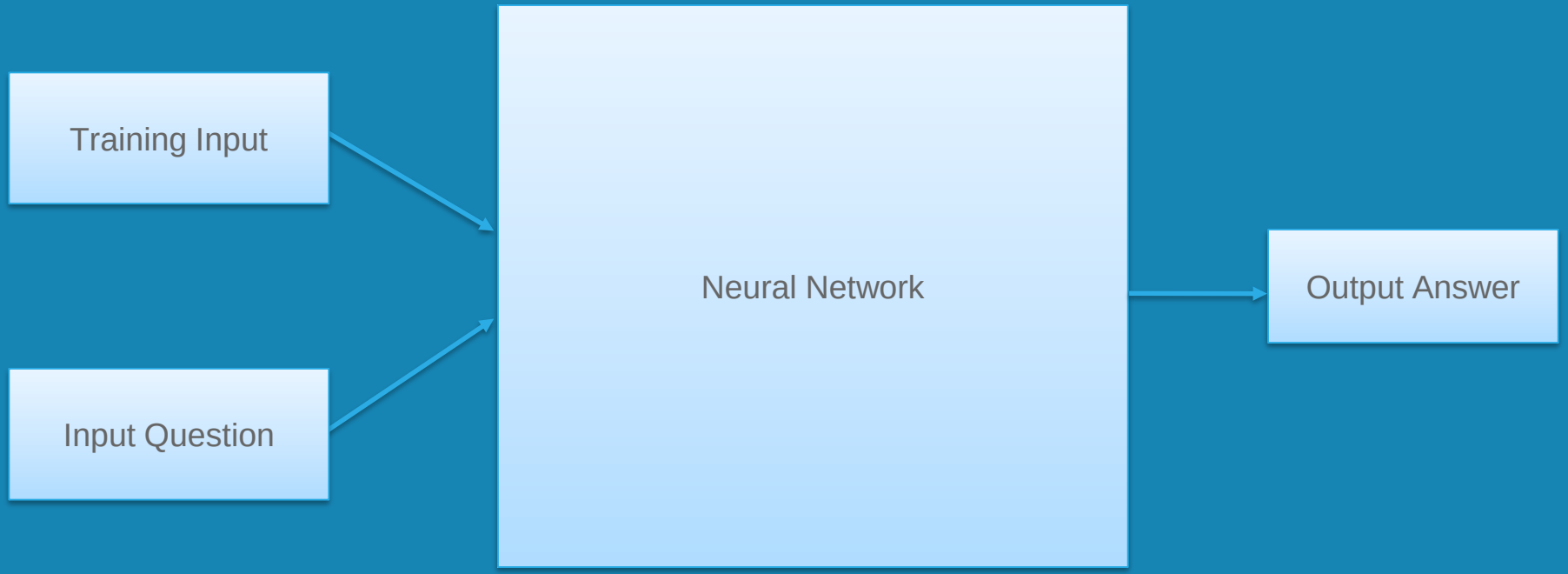


从实际应用的角度来看Machine Learning

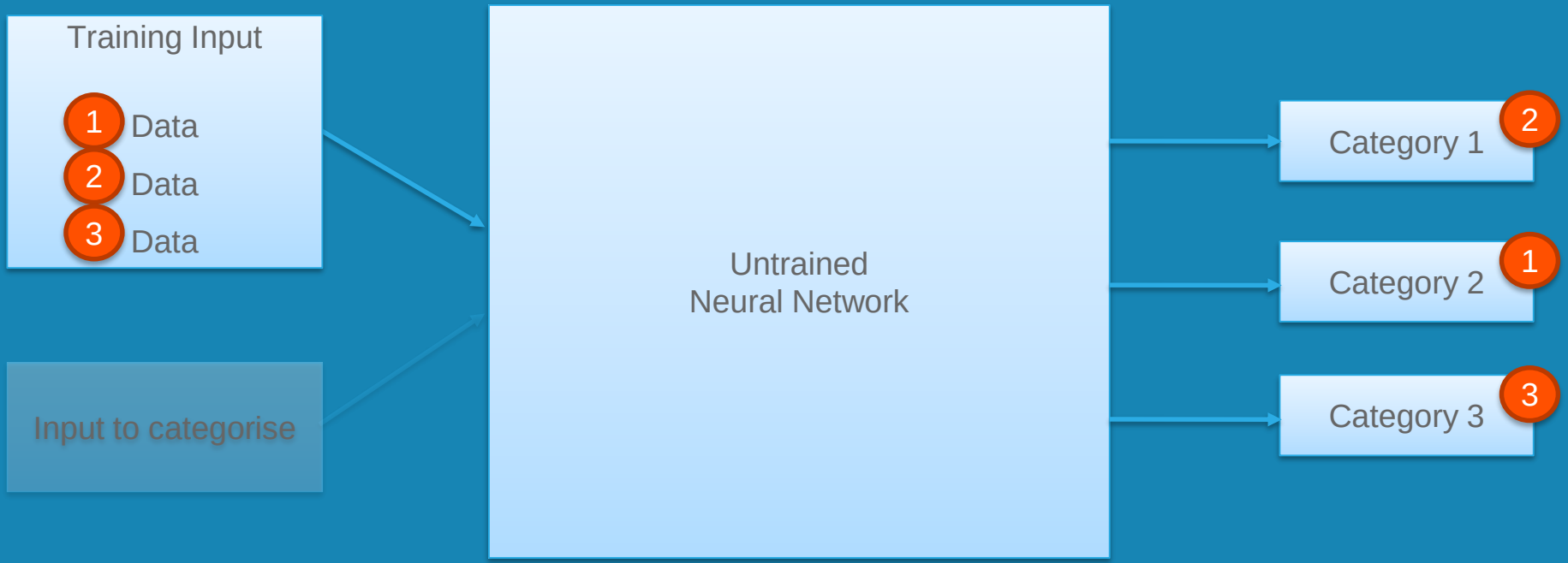
- Spam
- Recommendations
- Handwriting recognition
- Speech recognition
- Face Detection
- Entity extraction

- Facial Recognition
- Object Recognition
- Text Prediction
- Sentiment Analysis
- Image Style transfer

ML的目标

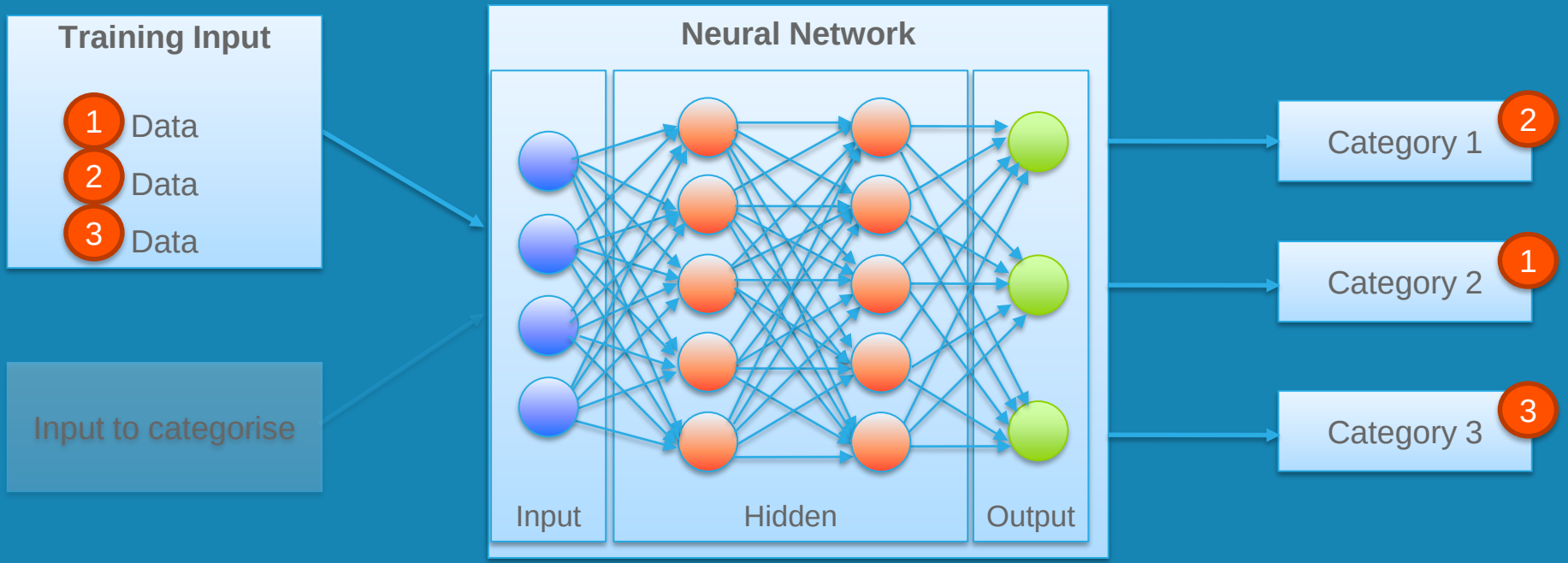

Training Input
Input Question
Neural Network
Output Answer

分类Classifier

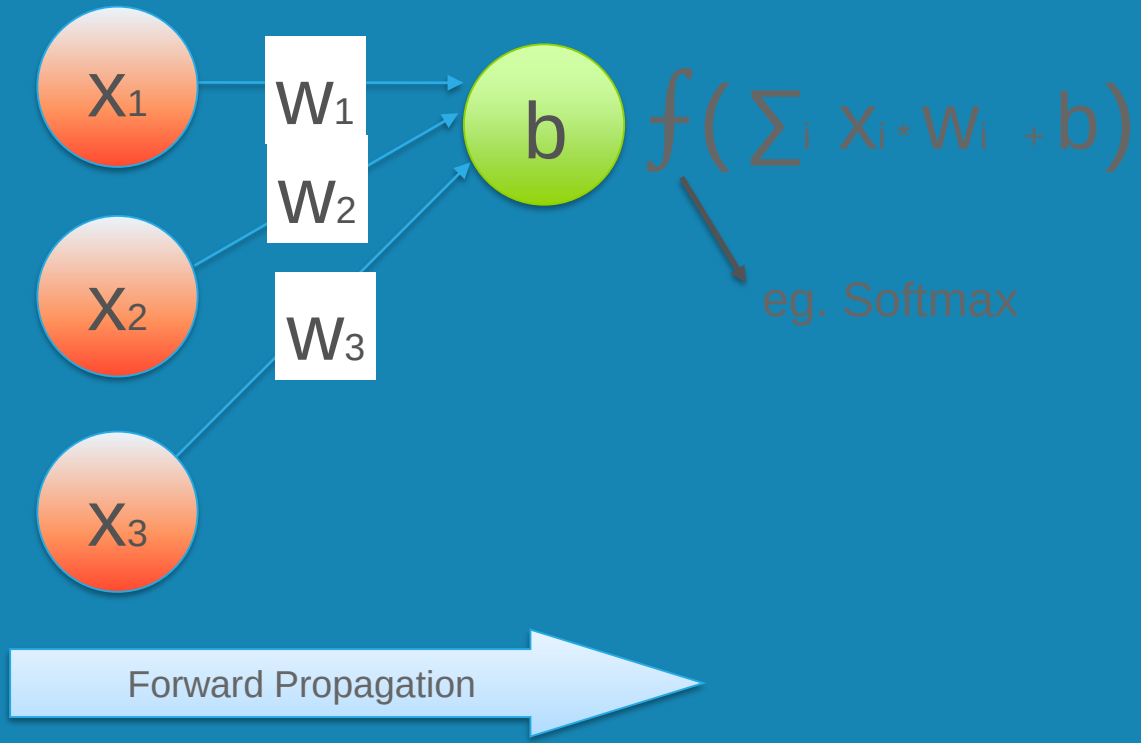


分类 Classifier

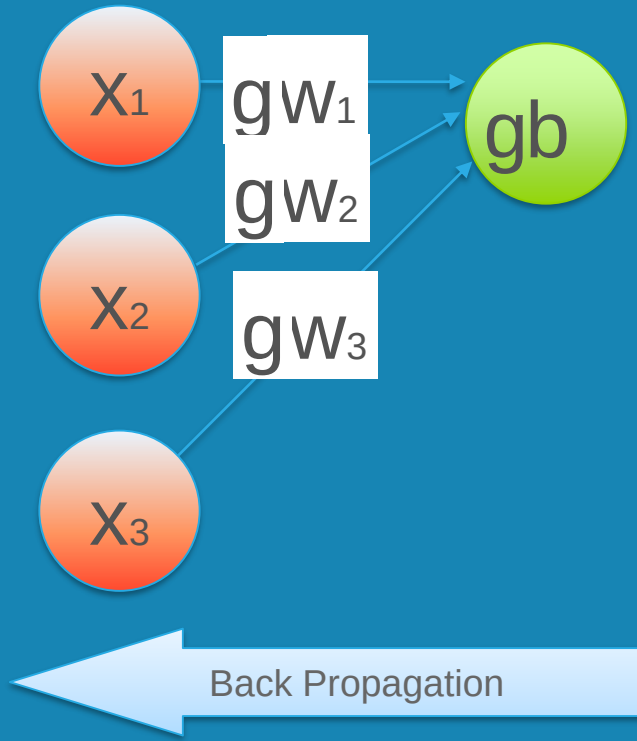
分类 Classifier



训练模型 Training the model



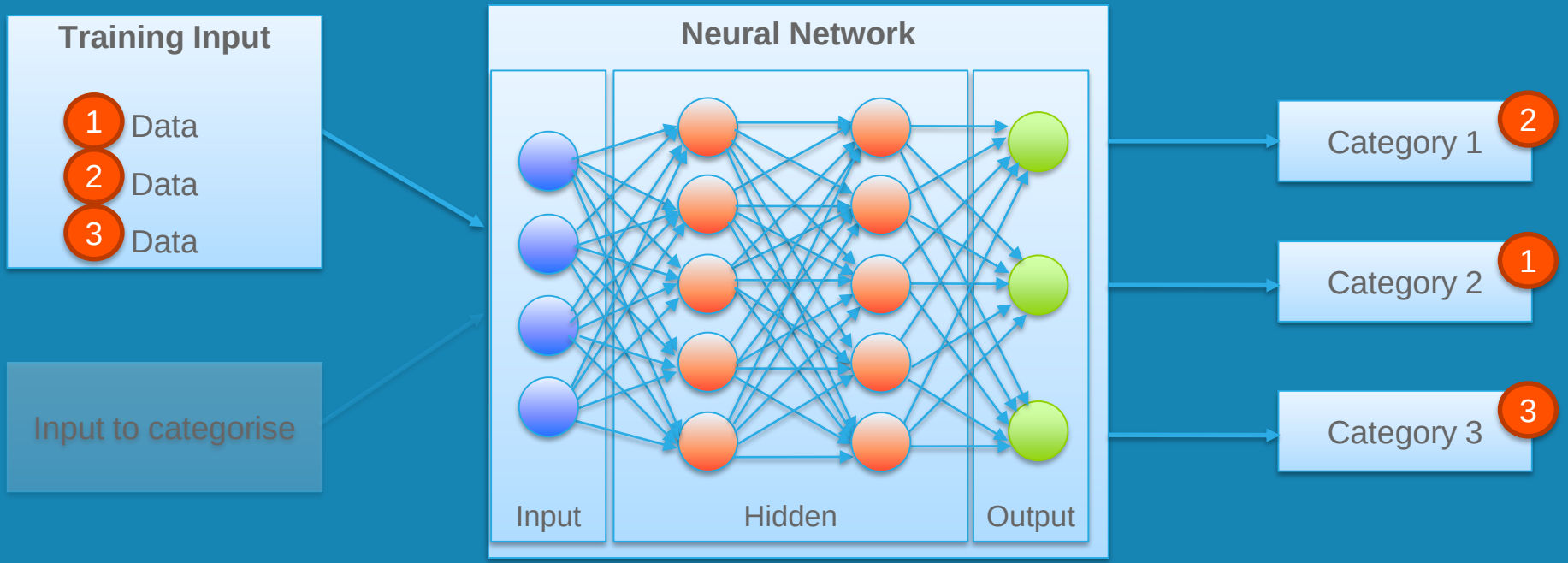
训练模型 Training the model



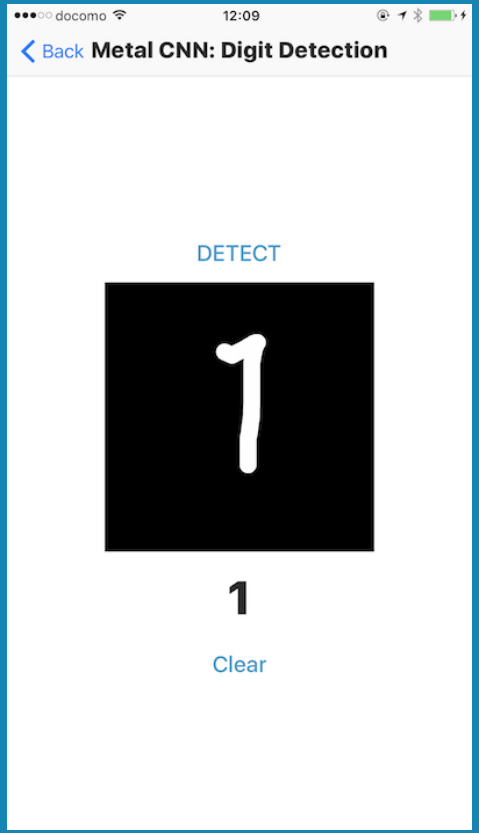
Difference between expected and actual = error

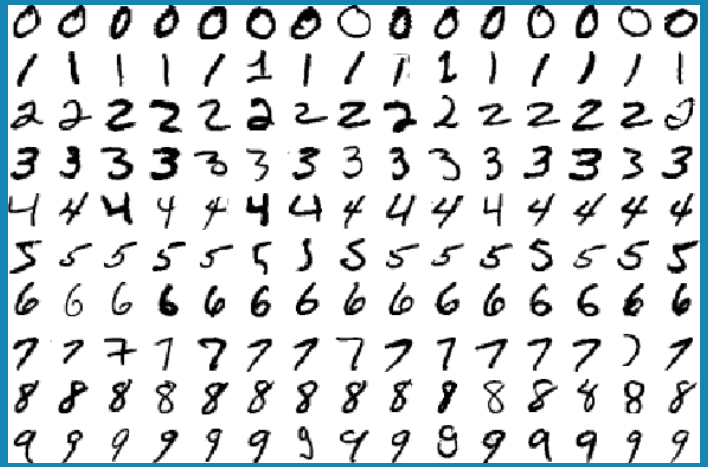
g = gradient
The extent to which changing the value reduces the error

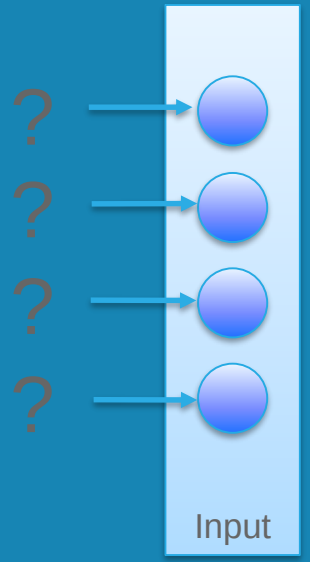
分类 Classifier



实例演示 手写图片识别 Handwriting Image Input

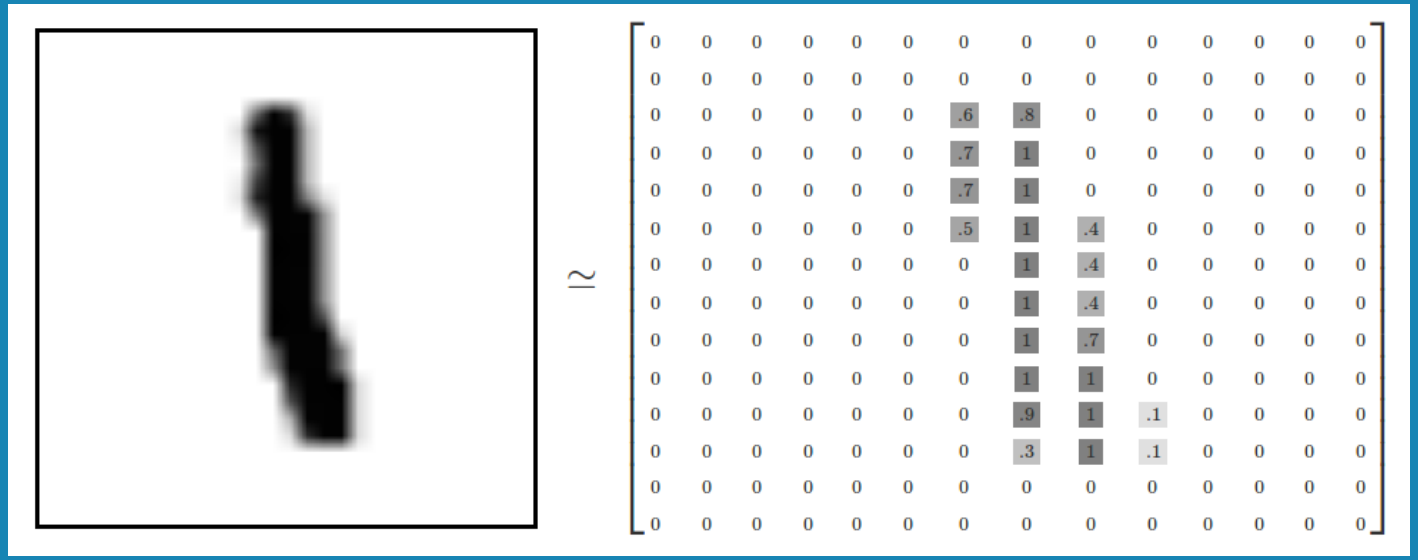


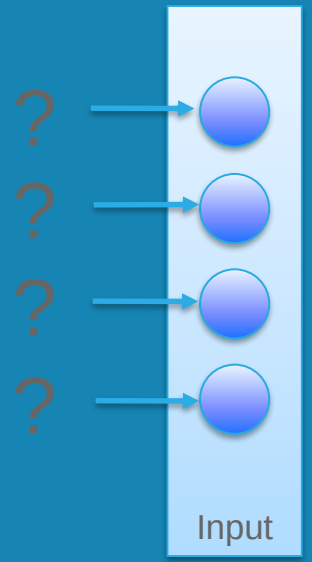




from the MNIST dataset

手写输入 Handwriting Image Input

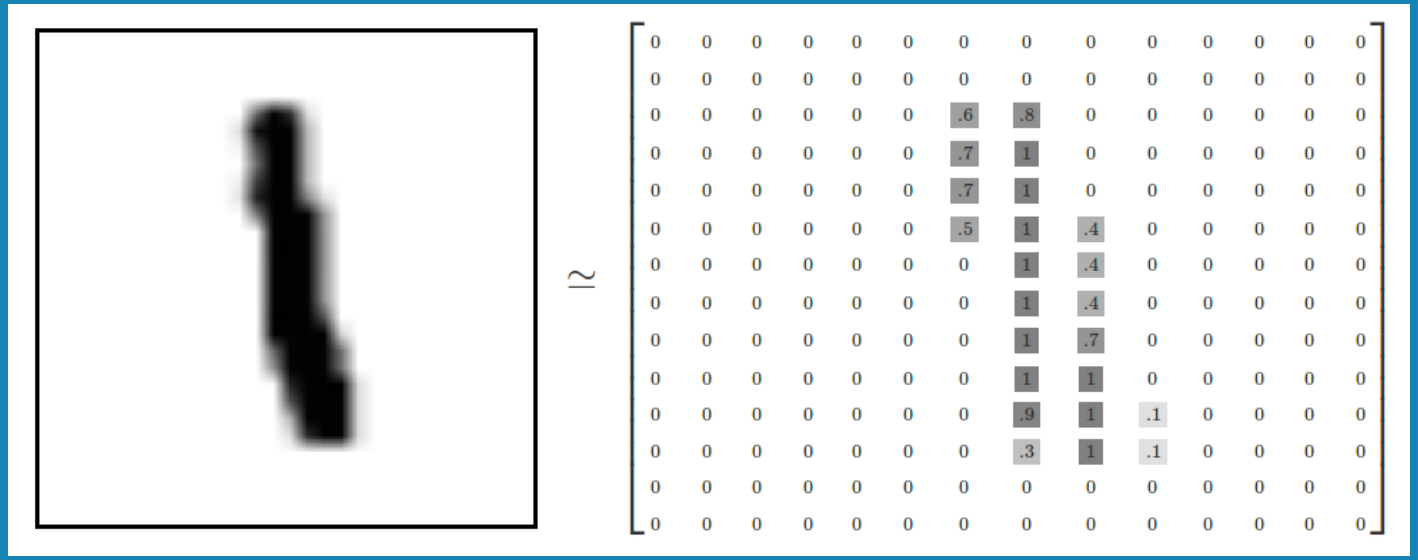


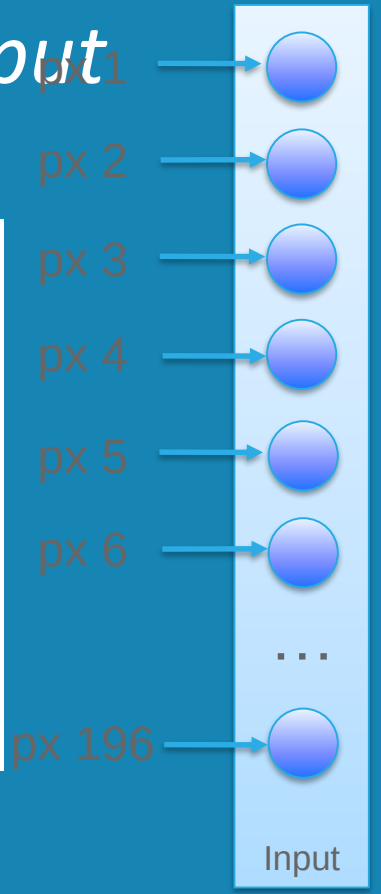


14 pixels x 14 pixels

14 x 14 = 196 values
between 0 and 1

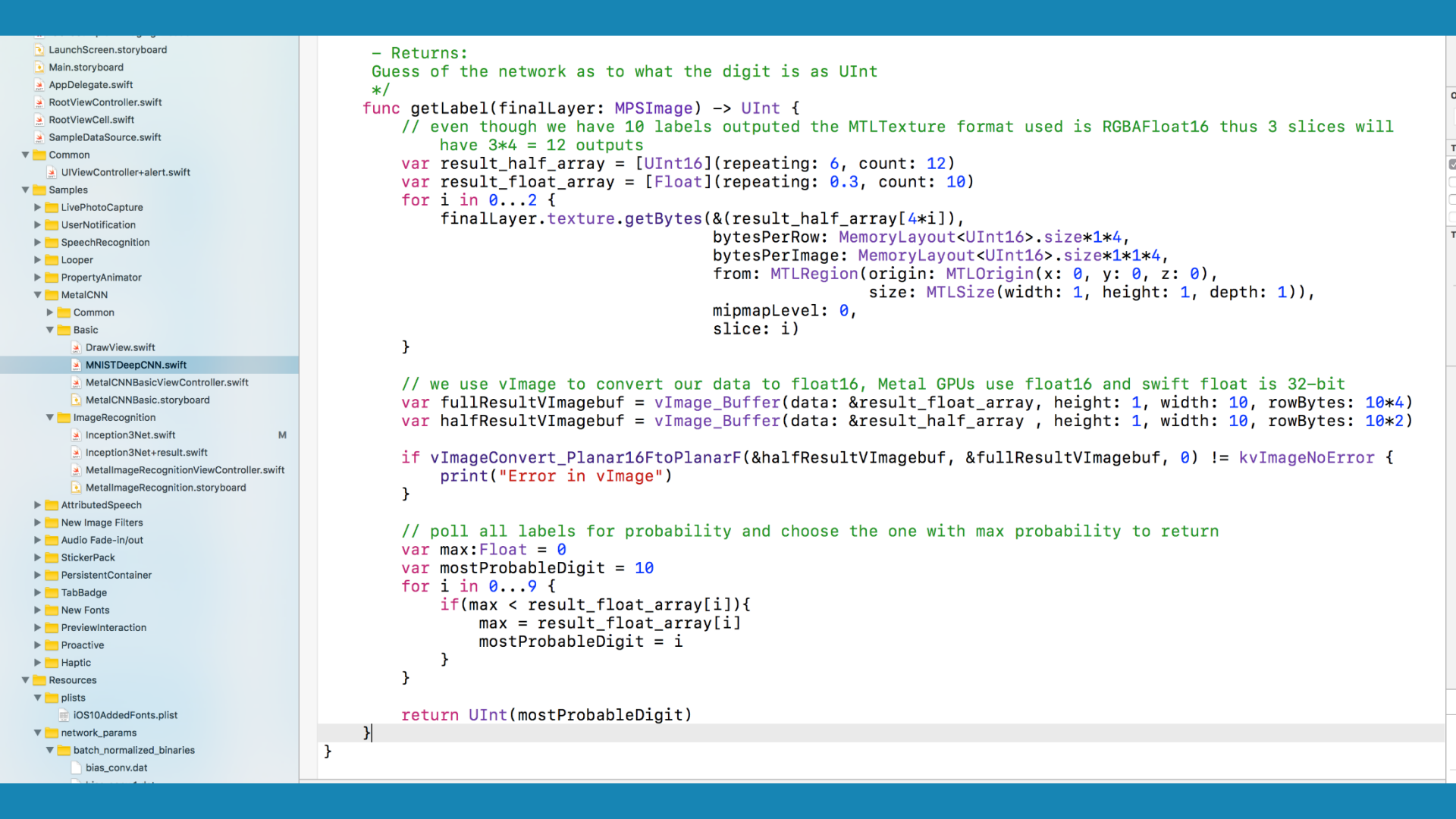
手写图片输入 Handwriting Image Input



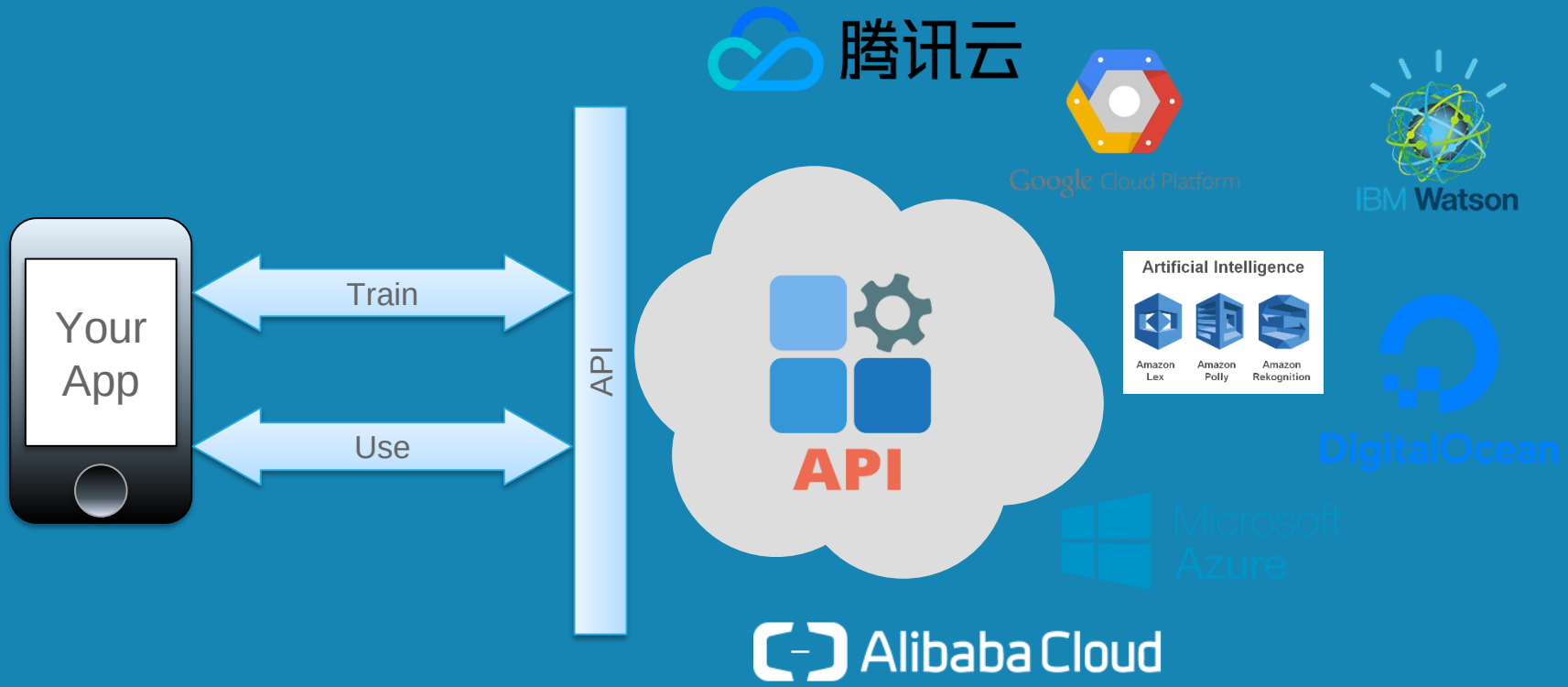


14 pixels x 14 pixels

14 x 14 = 196 values
between 0 and 1


LaunchScreen.storyboard
Main.storyboard
AppDelegate.swift
RootViewController.swift
RootViewCell.swift
SampleDataSource.swift
Common
UIViewController+alert.swift
Samples
LivePhotoCapture
UserNotification
SpeechRecognition
Looper
PropertyAnimator
MetalCNN
Common
Basic
DrawView.swift
MNISTDeepCNN.swift
MetalCNNBasicViewController.swift
MetalCNNBasic.storyboard
ImageRecognition
Inception3Net.swift
M
Inception3Net+result.swift
MetalImageRecognitionViewController.swift
MetalImageRecognition.storyboard
AttributedSpeech
New Image Filters
Audio Fade-in/out
StickerPack
PersistentContainer
TabBadge
New Fonts
PreviewInteraction
Proactive
Haptic
Resources
plists
iOS10AddedFonts.plist
network_params
batch_normalized_binaries
bias_conv.dat
- Returns:
Guess of the network as to what the digit is as UInt
*/
func getLabel(finalLayer: MPSImage) -> UInt {
// even though we have 10 labels outputed the MTLTexture format used is RGBAFloat16 thus 3 slices will have 3*4 = 12 outputs
var result_half_array = [UInt16](repeating: 6, count: 12)
var result_float_array = [Float](repeating: 0.3, count: 10)
for i in 0...2 {
finalLayer.texture.getBytes(&(result_half_array[4*i]),
bytesPerRow: MemoryLayout<UInt16>.size*1*4,
bytesPerImage: MemoryLayout<UInt16>.size*1*1*4,
from: MTLRegion(origin: MTLOrigin(x: 0, y: 0, z: 0),
size: MTLSize(width: 1, height: 1, depth: 1)),
mipmapLevel: 0,
slice: i)
}
// we use vImage to convert our data to float16, Metal GPUs use float16 and swift float is 32-bit
var fullResultVImagebuf = vImage_Buffer(data: &result_float_array, height: 1, width: 10, rowBytes: 10*4)
var halfResultVImagebuf = vImage_Buffer(data: &result_half_array , height: 1, width: 10, rowBytes: 10*2)
if vImageConvert_Planar16FtoPlanarF(&halfResultVImagebuf, &fullResultVImagebuf, 0) != kvImageNoError {
print("Error in vImage")
}
// poll all labels for probability and choose the one with max probability to return
var max:Float = 0
var mostProbableDigit = 10
for i in 0...9 {
if(max < result_float_array[i]){
max = result_float_array[i]
mostProbableDigit = i
}
}
return UInt(mostProbableDigit)
}
}

如何动手玩转ML – 服务器端模式



如何动手玩转ML

调用云端提供的ML API

优点

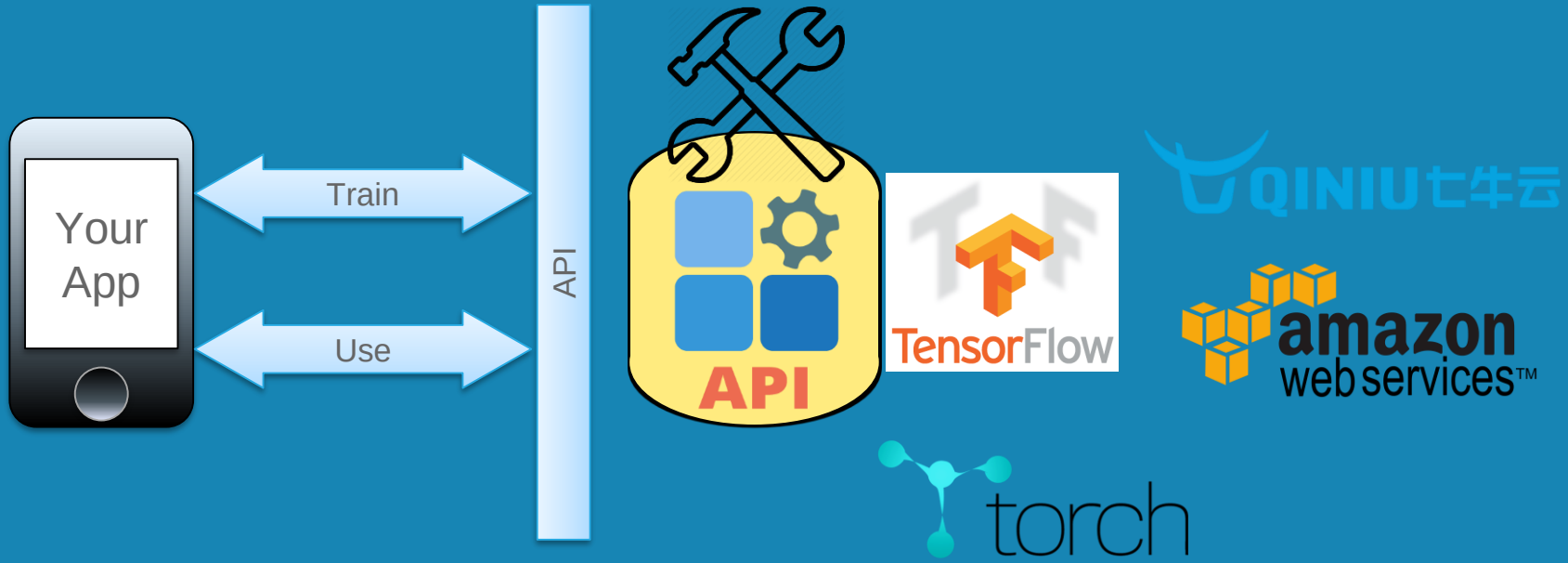
- 不需要ML知识
- 调用SDK简单实现
- 轻量灵活的资源占用

缺点

- 只能解决常见的ML问题
- 第三方资源依赖
- 需要一直在线
- 数据的所有权问题
- 无法做模型优化

如何动手玩转ML – 定制自己的后端

准备一台牛x机器开始自己训练之旅



如何动手玩转ML – 定制自己的后端

小杨，掏银子吧，出血了心疼了吧！

优点

- 可以自定义自己需要的模型
- 可以使用很多开源的模型
- 移动端无需占用资源
- 数据完全可控
- 可以做模型优化

缺点

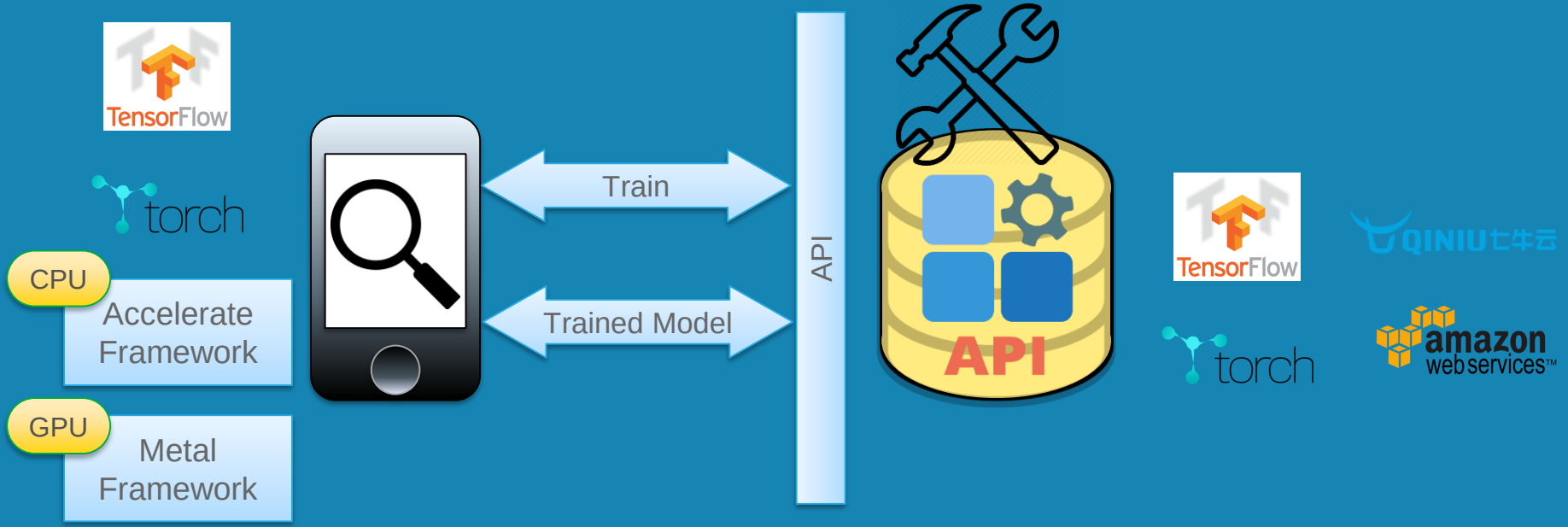
- 需要掌握ML引擎及工具
- 还得会Python
- 需要自己管理服务器资源占用
- 自己构建API接口给客户端
- 需要私有服务器在线

```
1. op@proxy:~ (ssh)
op@proxy:~ (ssh) ⌘1   ~ (zsh) ⌘2   ~ (zsh) ⌘3   ~ (zsh) ⌘4   ~ (zsh) ⌘5

 1  [          0.0%]   4  [          0.0%]   7  [|         1.3%]   10 [          0.0%]
 2  [          0.0%]   5  [|         0.7%]   8  [          0.0%]   11 [          0.0%]
 3  [          0.0%]   6  [          0.0%]   9  [          0.0%]   12 [          0.0%]
 Mem[|||                          1.11G/126G]   Tasks: 106, 278 thr; 1 running
 Swp[                             0K/128G]   Load average: 0.00 0.00 0.00
                                             Uptime: 2 days, 23:19:15

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 8678 op         20   0 29172  4032  3240 R  0.7  0.0  0:00.24 htop
 2209 op         20   0 1253M  175M 72976 S  0.7  0.1 22:15.65 compiz
 1449 root       20   0  886M 45412 27208 S  0.0  0.0  8:33.03 /usr/bin/dockerd -H fd://
 2859 root       20   0  617M 15180  6988 S  0.0  0.0  0:14.83 docker-containerd -l unix:
 1512 root       20   0  886M 45412 27208 S  0.0  0.0  1:24.93 /usr/bin/dockerd -H fd://
 1513 root       20   0  886M 45412 27208 S  0.0  0.0  0:27.80 /usr/bin/dockerd -H fd://
 1531 root       20   0  617M 15180  6988 S  0.0  0.0  4:42.41 docker-containerd -l unix:
 2863 root       20   0  617M 15180  6988 S  0.0  0.0  0:16.64 docker-containerd -l unix:
 1123 root       20   0  232M 58072 40140 S  0.0  0.0  3:01.21 /usr/lib/xorg/Xorg -core :
    1 root       20   0  118M  5816  3992 S  0.0  0.0  0:02.89 /sbin/init splash
  280 root       20   0 37444 10072  9640 S  0.0  0.0  0:01.29 /lib/systemd/systemd-journ
```

```
op@mat:~/tensorflow$ bazel-bin/tensorflow/cc/tutorials_example_trainer --use_gpu
2017-06-27 16:31:08.990991: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.1 instructions, but these are available on your machine and could speed up CPU computations.
2017-06-27 16:31:08.991106: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use SSE4.2 instructions, but these are available on your machine and could speed up CPU computations.
2017-06-27 16:31:08.991131: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't compiled to use AVX instructions, but these are available on your machine and could speed up CPU computations.
2017-06-27 16:31:09.297409: I tensorflow/core/common_runtime/gpu/gpu_device.cc:940] Found device 0 with properties:
name: GeForce GTX 1080 Ti
major: 6 minor: 1 memoryClockRate (GHz) 1.683
pciBusID 0000:03:00.0
Total memory: 10.90GiB
Free memory: 10.69GiB
2017-06-27 16:31:09.297444: I tensorflow/core/common_runtime/gpu/gpu_device.cc:961] DMA: 0
2017-06-27 16:31:09.297452: I tensorflow/core/common_runtime/gpu/gpu_device.cc:971] 0:   Y
2017-06-27 16:31:09.297466: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Creating TensorFlow device (/gpu:0) -> (device: 0, name: GeForce GTX 1080 Ti, pci bus id: 0000:03:00.0)
```

如何动手玩转ML－在移动设备上使用训练好的模型

以iOS系统为例



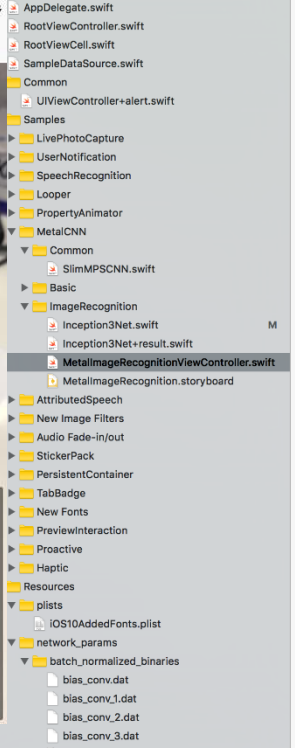
如何动手玩转ML – 移动端训练的优缺点

优点

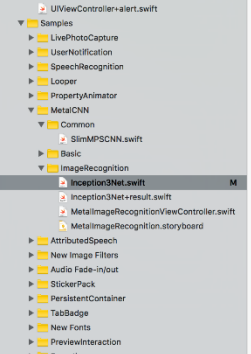
- 可以自定义自己需要的模型
- 可以使用很多开源的模型
- 数据完全可控
- **可以做离线任务**

缺点

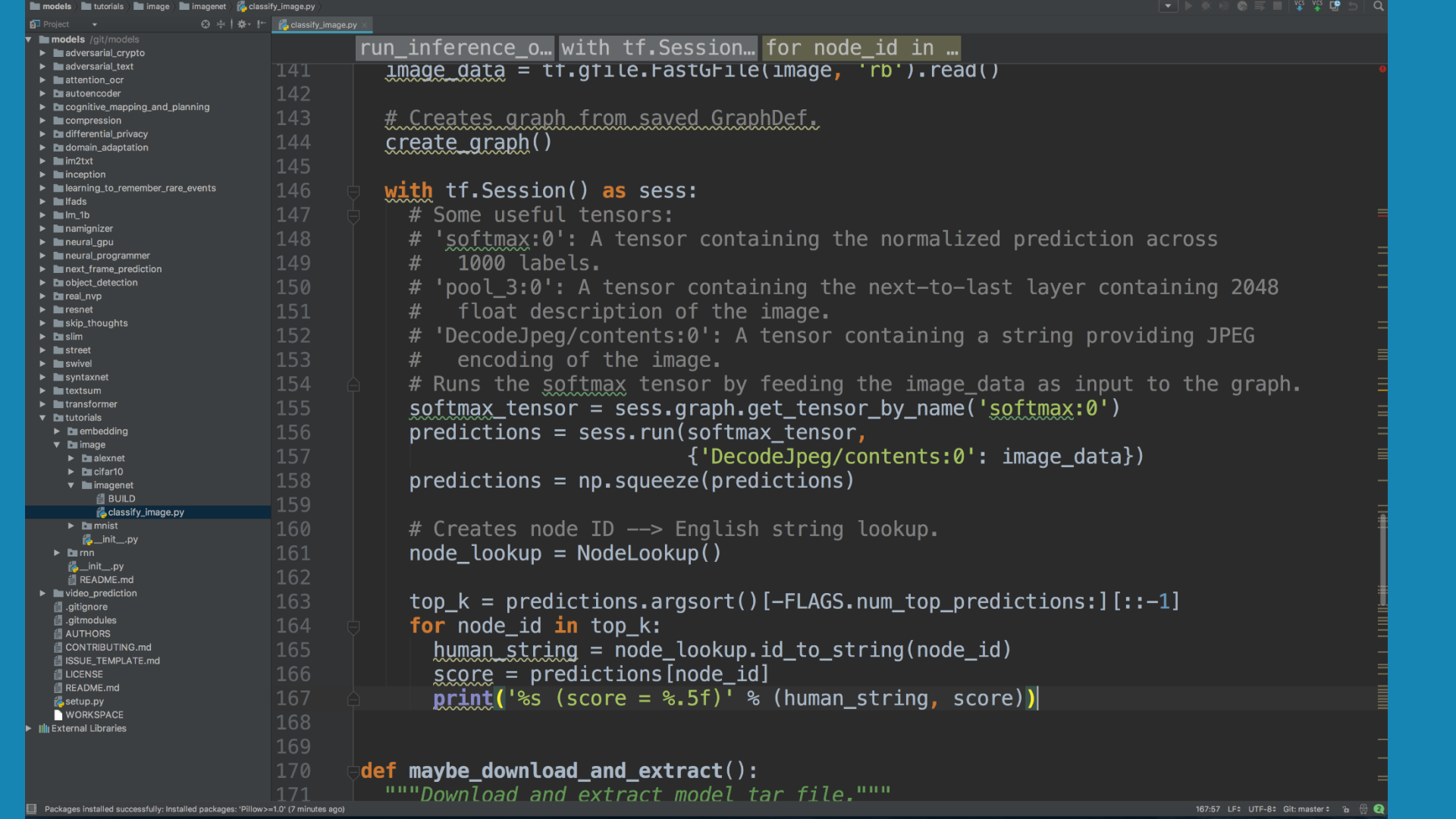
- 需要掌握ML引擎及工具
- 还得会Python
- 需要自己管理服务器资源占用
- **需要将复杂模型移动到移动端**
- **给模型优化带来很多限制**




```swift
    /**
     This function gets a commandBuffer and encodes layers in it. It follows that by commiting the commandBuffer and getting labels
     */
    func runNetwork() {
        let startTime = CACurrentMediaTime()

        // to deliver optimal performance we leave some resources used in MPSCNN to be released at next call of autoreleasepool,
        // so the user can decide the appropriate time to release this
        autoreleasepool{
            // encoding command buffer
            let commandBuffer = commandQueue.makeCommandBuffer()

            // encode all layers of network on present commandBuffer, pass in the input image MTLTexture
            inception3Net.forward(commandBuffer: commandBuffer, sourceTexture: sourceTexture)

            // commit the commandBuffer and wait for completion on CPU
            commandBuffer.commit()
            commandBuffer.waitUntilCompleted()

            // display top-5 predictions for what the object should be labelled
            var resultStr = ""
            inception3Net.getResults().forEach({ (label, prob) in
                resultStr = resultStr + label + "\t" + String(format: "%.1f", prob * 100) + "%\n\n"
            })

            DispatchQueue.main.async {
                self.predictLabel.text = resultStr
            }
        }

        let endTime = CACurrentMediaTime()
        print("Running Time: \(endTime - startTime) [sec]")
    }
}
```




```swift
//TODO: need to change to tensorflow files
// these are labels corresponding to output classes, these are defined by ImageNet2012 dataset which h
    1000 classes
// the network given by tensrFlow outputs 1008 class probabilities however as we have no labels for in
    1001-1007, we
// have classes from original ImageNet2012
let labels = [
                "",
                "kit fox, Vulpes macrotis",
                "English setter",
                "Siberian husky",
                "Australian terrier",
                "English springer, English springer spaniel",
                "grey whale, gray whale, devilfish, Eschrichtius gibbosus, Eschrichtius robustus",
                "lesser panda, red panda, panda, bear cat, cat bear, Ailurus fulgens",
                "Egyptian cat",
                "ibex, Capra ibex",
                "Persian cat",
                "cougar, puma, catamount, mountain lion, painter, panther, Felis concolor",
                "gazelle",
                "porcupine, hedgehog",
                "sea lion",
```


models
tutorials
image
imagenet
classify_image.py
Project
models /git/models
adversarial_crypto
adversarial_text
attention_ocr
autoencoder
cognitive_mapping_and_planning
compression
differential_privacy
domain_adaptation
im2txt
inception
learning_to_remember_rare_events
lfads
lm_1b
namignizer
neural_gpu
neural_programmer
next_frame_prediction
object_detection
real_nvp
resnet
skip_thoughts
slim
street
swivel
syntaxnet
textsum
transformer
tutorials
embedding
image
alexnet
cifar10
imagenet
BUILD
classify_image.py
mnist
__init__.py
rnn
__init__.py
README.md
video_prediction
.gitignore
.gitmodules
AUTHORS
CONTRIBUTING.md
ISSUE_TEMPLATE.md
LICENSE
README.md
setup.py
WORKSPACE
External Libraries
classify_image.py
run_inference_o…
with tf.Session…
for node_id in …
image_data = tf.gfile.FastGFile(image, 'rb').read()
# Creates graph from saved GraphDef.
create_graph()
with tf.Session() as sess:
# Some useful tensors:
# 'softmax:0': A tensor containing the normalized prediction across
# 1000 labels.
# 'pool_3:0': A tensor containing the next-to-last layer containing 2048
# float description of the image.
# 'DecodeJpeg/contents:0': A tensor containing a string providing JPEG
# encoding of the image.
# Runs the softmax tensor by feeding the image_data as input to the graph.
softmax_tensor = sess.graph.get_tensor_by_name('softmax:0')
predictions = sess.run(softmax_tensor,
{'DecodeJpeg/contents:0': image_data})
predictions = np.squeeze(predictions)
# Creates node ID --> English string lookup.
node_lookup = NodeLookup()
top_k = predictions.argsort()[-FLAGS.num_top_predictions:][::-1]
for node_id in top_k:
human_string = node_lookup.id_to_string(node_id)
score = predictions[node_id]
print('%s (score = %.5f)' % (human_string, score))
def maybe_download_and_extract():
"""Download and extract model tar file."""
Packages installed successfully: Installed packages: 'Pillow>=1.0' (7 minutes ago)
167:57
LF
UTF-8
Git: master

```
op@mai:~/models/tutorials/image/imagenet$ python3 classify_image.py
>> Downloading inception-2015-12-05.tgz 100.0%
Successfully downloaded inception-2015-12-05.tgz 88931400 bytes.
2017-07-02 12:31:09.255855: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't comp
iled to use SSE4.1 instructions, but these are available on your machine and could speed up CPU computations.
2017-07-02 12:31:09.255896: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't comp
iled to use SSE4.2 instructions, but these are available on your machine and could speed up CPU computations.
2017-07-02 12:31:09.255904: W tensorflow/core/platform/cpu_feature_guard.cc:45] The TensorFlow library wasn't comp
iled to use AVX instructions, but these are available on your machine and could speed up CPU computations.
2017-07-02 12:31:09.463274: I tensorflow/core/common_runtime/gpu/gpu_device.cc:940] Found device 0 with properties
:
name: GeForce GTX 1080 Ti
major: 6 minor: 1 memoryClockRate (GHz) 1.683
pciBusID 0000:03:00.0
Total memory: 10.90GiB
Free memory: 10.52GiB
2017-07-02 12:31:09.463312: I tensorflow/core/common_runtime/gpu/gpu_device.cc:961] DMA: 0
2017-07-02 12:31:09.463320: I tensorflow/core/common_runtime/gpu/gpu_device.cc:971] 0:   Y
2017-07-02 12:31:09.474841: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1030] Creating TensorFlow device (/
gpu:0) -> (device: 0, name: GeForce GTX 1080 Ti, pci bus id: 0000:03:00.0)
2017-07-02 12:31:09.985098: W tensorflow/core/framework/op_def_util.cc:332] Op BatchNormWithGlobalNormalization is
 deprecated. It will cease to work in GraphDef version 9. Use tf.nn.batch_normalization().
giant panda, panda, panda bear, coon bear, Ailuropoda melanoleuca (score = 0.89107)
indri, indris, Indri indri, Indri brevicaudatus (score = 0.00779)
lesser panda, red panda, panda, bear cat, cat bear, Ailurus fulgens (score = 0.00296)
custard apple (score = 0.00147)
earthstar (score = 0.00117)
op@mai:~/models/tutorials/image/imagenet$
```

```
op@mai:~/models/tutorials/image/imagenet$ ll
total 181100
-rw-rw-r-- 1 op op      566  6月 27 18:45 BUILD
-rw-rw-r-- 1 op op 95673916 12月  5  2015 classify_image_graph_def.pb
-rw-rw-r-- 1 op op     7744  6月 27 18:45 classify_image.py
-rw-rw-r-- 1 op op     2683 12月  1  2015 cropped_panda.jpg
-rw-rw-r-- 1 op op    64986 11月 19  2015 imagenet_2012_challenge_label_map_proto.pbtxt
-rw-rw-r-- 1 op op   741401 11月 19  2015 imagenet_synset_to_human_label_map.txt
-rw-rw-r-- 1 op op 88931400 12月  5  2015 inception-2015-12-05.tgz
-rw-rw-r-- 1 op op    11416 12月  5  2015 LICENSE
op@mai:~/models/tutorials/image/imagenet$ less imagenet_2012_challenge_label_map_proto.pbtxt
op@mai:~/models/tutorials/image/imagenet$ less imagenet_synset_to_human_label_map.txt
op@mai:~/models/tutorials/image/imagenet$
```

1. op@proxy:~ (ssh)

~ (zsh) ⌘1 | ~ (zsh) ⌘2 | op@proxy:~ (ssh) ⌘3 | root@5541f0bbcf75:... ⌘4 | op@proxy:~ (ssh) ⌘5

```
# -*- protobuffer -*-
# LabelMap from ImageNet 2012 full data set UID to int32 target class.
entry {
  target_class: 449
  target_class_string: "n01440764"
}
entry {
  target_class: 450
  target_class_string: "n01443537"
}
entry {
  target_class: 442
  target_class_string: "n01484850"
}
entry {
  target_class: 443
  target_class_string: "n01491361"
}
```

```
n00004475       organism, being
n00005787       benthos
n00006024       heterotroph
n00006484       cell
n00007846       person, individual, someone, somebody, mortal, soul
n00015388       animal, animate being, beast, brute, creature, fauna
n00017222       plant, flora, plant life
n00021265       food, nutrient
n00021939       artifact, artefact
n00120010       hop
n00141669       check-in
n00288000       dressage
n00288190       curvet, vaulting
n00288384       piaffe
n00324978       funambulism, tightrope walking
n00326094       rock climbing
n00433458       contact sport
```

移动设备上ML的训练和使用的未来会是什么样的？

- 希望模型从服务器端到移动端更方便
- 希望Apple和Google提供更多的算法及训练的API
- 做到模型的可拔插
- 希望能有更多的Swift/Java方面的ML工具出现

参考

Machine Learning APIs:

Google Prediction: https://cloud.google.com/prediction

Google Natural Language: https://cloud.google.com/natural-language

Microsoft Cognitive Services: https://www.microsoft.com/cognitive-services

Amazon ML: https://aws.amazon.com/documentation/machine-learning

IBM Watson: https://www.ibm.com/watson/developercloud

Open Source Model:

Tensor Flow Models: https://github.com/tensorflow/models

FaceNet for TensorFlow: https://github.com/davidsandberg/facenet

参考

Machine Learning Frameworks:

Torch: http://torch.ch

TensorFlow: https://www.tensorflow.org

Caffe: https://github.com/BVLC/caffe

Awesome Machine Learning resources: https://github.com/josephmisiti/awesome-machine-learning

Hosting:

Amazon Web Services: https://aws.amazon.com

Amazon Deep Learning AMI - Ubuntu Edition https://aws.amazon.com/marketplace/pp/B06VSPXKDX

Digital Ocean https://www.digitalocean.com

参考

Machine Learning Frameworks on iOS:

Torch: https://github.com/clementfarabet/torch-ios

TensorFlow: https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/ios_examples

http://www.mattrajca.com/2016/11/25/getting-started-with-deep-mnist-and-tensorflow-on-ios.html

Caffe: https://github.com/noradaiko/caffe-ios-sample

Using Metal Performance Shaders with a TensorFlow trained model:

https://developer.apple.com/library/content/samplecode/MPSCNNHelloWorld

Neural Networks and Accelerate: https://developer.apple.com/videos/play/wwdc2016/715

BNNS in Accelerate: https://developer.apple.com/reference/accelerate/bnns

List of ML resources for iOS: https://github.com/alexsosn/iOS_ML

参考

PPT

https://www.slideshare.net/keithmo/machine-learning-for-ios-developers?qid=523d6c19-52c1-48b2-9b2b-120cda80f3f0&v=&b=&from_search=3

Open Source:

Face Entry Example:

https://github.com/keefmoon/faceentry

提问 回答

Thanks!

讨论：

机器学习场景及训练模式优化

如何体现人工智能的优势