

## Survey on DL@UAV vision

### CONTENT

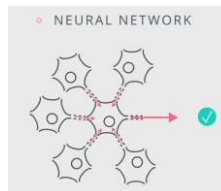


# 01

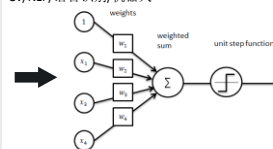
## What is DL?

### 1\_ What is DL

From Neurons to Perceptron



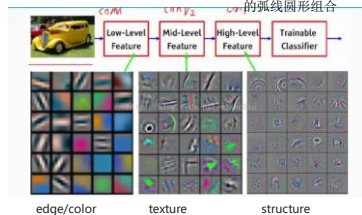
1957 年美国康乃尔大学的心理学家罗森布拉特 (Rosenblatt) 提出了感知器或感知机 (Perceptron) 的概念, 并试图用它来模拟动物和人脑的感知与学习能力, 而且用一个电子线路设计了著名的感知器神经网络模型, 称为仿脑机, 能够识别英文字母印刷体。深度学习应用的四个大的领域: CV/NLP/语音识别/机器人。





## 1\_ What is DL

CNN as Feature Extractor







CNN Visualization

CNN就是一个特征提取的工具,通过特征可视化的手段,我们可以还原出对某一层激活函数激活值最大的一些图像,可以看到从低到高,CNN在逐渐抽取越来越抽象的特征,开始时边缘/颜色块等,然后就是纹理,然后就是些结构,比如下面的蜂巢,上面的像眼睛一样的弧线圆形组合

02

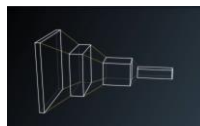
## Application

## 2\_ Application

-  Object Recognition
-  Object Detection
-  Visual Object Tracking(VOT)
-  Visual Slam

## 2\_ Application

Object Recognition



Classification

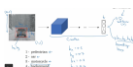


Dogs vs. Cats in Kaggle

1. CNN用于OR的一般架构就是在空间上越来越小,深度上越来越深,就是把图像的信息从空间上抽象到深度上的过程,所以我们可以看到,CNN逐层空间大小在变小,深度在增加
2. 在网络的结尾会用一个分类器结束
3. 在分类任务的比赛中,我们通常会看到大家用集成学习的方法,将好几个模型的结果来投票,或是直接contact在一起再分类,可以大幅提高分类的精度

- ## 2\_ Application
1. 用CNN做定位相当于在之前的分类问题上加了一个回归问题,就是要估计 $x,y,w,h$ 这四个连续变量
  2. 两个流派,要么就是先proposal再做分类和回归,要么是滑动窗,当然后来不管是proposal还是sliding window大家都不会在输入图像上去做,而是在提取出的特征图上去做
  3. 现在Object detection的意义更广了,不止是要把物体定位出来,还要估计图像的轮廓,有代表性的就是ICCV 2017的best paper Mask r-cnn,这是从r-cnn这条路上一路过来的一个成果

### Object Detection



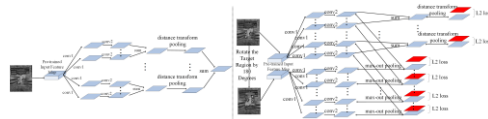
Classification + multi-object localization



VEDAI UAV Object Detection Dataset

- ## 2\_ Application
- 本文旨在将CNN和KRR结合起来,其中CNN用于用于关注目标的局部信息, KRR用于关注目标的整体信息

### VOT

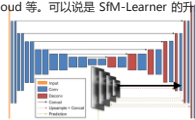
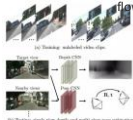


### LSART – VOT 2017 Championship

- Exploit both the KRR and CNN as two complementary regressions for visual tracking
- CNN focuses on the small localized region
- KRR focuses on the holistic target

- ## 2\_ Application
- 它用两个网络分别/独自无监督地估计单帧的深度, 和视频序列中的camera的pose变化
  - 光度一致性, 就是对于同一个物体的点,在不同两帧图像上投影点,图像灰度应该是一样的
  - SfM-Net论文的核心思想也是利用 photometric constancy 来计算 pose, depth. 除此之外, 作者还计算了光流, scene flow, 3D point cloud 等. 可以说是 SfM-Learner 的升级版

### Visual Slam



### SfM-Learner – CVPR 2017 from Google

- Use two isolated network (Depth CNN & Pose CNN) to predict depth & camera pose
- Base on photometric consistency principle
- See more on SfM-Net (also compute optic flow/scene flow/3D point cloud)

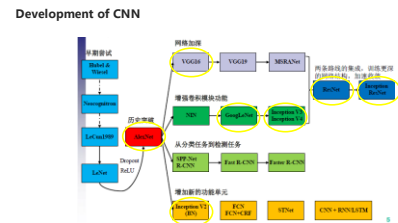
03

## Road Map

### 3\_ Road Map

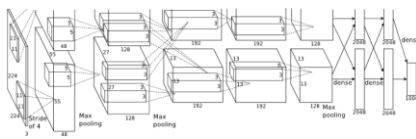


### 3\_ Road Map



### 3\_ Road Map

### AlexNet – 2012 ImageNet Championship (Top-5 Error 16.4%)



- First time, CNN won the ImageNet Challenge
- First time, Group Convolution Concept proposed
- First time, Use Relu as activation function(used in the following networks till now)

### 3\_ Road Map

**VGGNet – 2014 ImageNet (Top-5 Error 9.9%)**

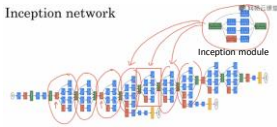


- Deep, 16/19 layers
- Conv+Pooling+FC
- Too much Parameters ~138,000,000

### 3\_ Road Map

Inception(GoogLeNet) – 2014 ImageNet Championship (Top-5 Error 9.2%)

Inception network



- Much less Parameters ~5,000,000(1/12 of AlexNet, 1/27 of VGGNet)
- 1x1 convolution(Network in Network)
- Group Convolution(Inception Module)
- Replace FC with Average Pooling

### 3\_ Road Map

数据标准

Single-crop, Single-model experimental results(一个样本截取一张图的, 单一模型不用集成学习的)

On ILSVRC 2012 dataset validation set

目前在ImageNet数据集上人眼能达到的错误率大概在5.1%, 这还是经过了大量训练的专家能达到的成绩, 一般人要区分1000种类型的图片是比较困难的

Inception Family

- Inception-V2 (Top-5 Error 7.8%)
  - Batch Normalization/two 3x3 replace 5x5
- Inception-V3 (Top-5 Error 5.6%)
  - Conv Factorization 3x3 -> 1x3+3x1
- Xception (Top-5 Error 5.5%)
  - Depth-wise separable convolution(separate depth conv & spatial conv)

### 3\_ Road Map

ResNet– 2015 ImageNet Championship (Top-5 Error 4.49%)  
2016 CVPR best paper

梯度弥散产生的原因在于靠近输入端的梯度是后面层的累加的乘积, 如果weight值在1以内(我们通常会这样初始化, 因为最后的sigmoid或者softmax在1以内的梯度最大, 更新快), 如果层数比较深, 基本就是指数效果( $0.9^n$ ), 到了最后面的隐藏层的时候, 梯度基本就没更新了, 这就是梯度弥散, 那反过来, 如果weight值超过1, 那又会出现梯度爆炸( $1.1^n$ ), 这本质上是多层层叠带来的梯度不稳定问题, 唯一可能稳定的情况是这些层连续乘积刚好平衡大约等于1, 但是几率很小

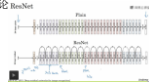
假如模型的层数越深, 这种梯度弥散的情况就更加严重, 导致深层部分的网络权重参数得不到很好的训练, 这就是为什么在ResNet出现之前, CNN网络都不超过二十几层的原因

数据来自ResNet论文(Single-Model)

从此能够限制层数的只有内存大小了 – KM He

KM He 2017年还有两篇ICCV best paper, 与object detection有关, 下个专题再讨论

- Shortcut connection/solve Gradient vanishing
- Break the depth limitation



### 3\_ Road Map

Inception-ResNet & ResNeXt(Top-5 Error 4.8% & 4.4%)



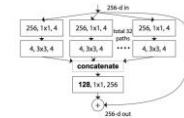
Inception Module + shortcut connection

天下大势, 分久必合, 合久必分

Cardinality和Group Conv的区别在于Cardinality是把一个Conv操作分成了拓扑结构完全相同的几个path, 这样方便加速

2017年最后一届ImageNet分类的冠军, Momenta的SENet也要研究一下

以及2017 CVPR best Paper, DenseNet, 可以看做是对ResNet的又一次大升级



Residual block + Group conv(cardinality)

### 3\_ Road Map

#### Cost Comparison

CNN模型在不断逼近计算机视觉任务的精度极限的同时，其深度和尺寸也在成倍增长。巨大的模型无法在嵌入式平台上落地，就算想通过网络传输，较高的带宽也让很多用户望而生畏。另一方面，大尺寸的模型也对设备功耗和运行速度带来了巨大的挑战。所以，模型的小型化和加速成了亟待解决的问题。

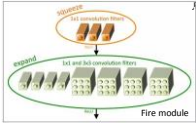
Model	Model Size(MB)	Mult-Adds	Million Para
AlexNet	200	720	60
VGG16	500	15300	138
GoLeNet	~50	1550	6.8
Inception-V3	90-100	5000	23.2

### 3\_ Road Map

- Introduction of CNN Development
- Small CNN for mobile application
- Network Compression

### 3\_ Road Map

#### SqueezeNet(ICLR-2017, Stanford)



1. 用1x1替换部分的3x3,减少参数
2. 减少进入3x3 filter的通道数,也就是先用1x1去压缩通道,就是squeeze
3. 少用pooling或者是conv stride,保留比较大的特征图,信息保留的多,最后的准确性更高,但这一点使模型解释的成本更高,运算代价更大了

- Replace part of 3x3 filters with 1x1 filters(small network)
- Decrease the number of input channels to 3x3 filters(small network)
- Downsample late in the network, conv layers has large activation map(high accuracy)

### 3\_ Road Map

#### SqueezeNet(ICLR-2017, Stanford)

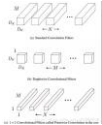
Table 2: Comparing SqueezeNet to model compression approaches. By model size, we mean the number of bytes required to store all of the parameters in the trained model.

CNN Architecture	Compressive Approach	Top-1 Accuracy (%)	Model Size (MB)	Top-1 Accuracy (%)	Model Size (MB)
AlexNet	None (Baseline)	56.7	248MB	56.7	248MB
AlexNet	WVD (SqueezeNet)	56.7	248MB	56.7	248MB
AlexNet	Network Pruning (Net2Net)	56.7	248MB	56.7	248MB
AlexNet	Deep Compression (Hao et al. 2016)	56.7	248MB	56.7	248MB
SqueezeNet (ours)	None	56.7	1.0MB	56.7	1.0MB
SqueezeNet (ours)	Deep Compression	56.7	1.0MB	56.7	1.0MB

- Small Network, but high compute cost(large activation map)
- Nothing Fresh in architecture

### 3\_ Road Map

MobileNet(CVPR-2017, Google)



- 1 conv filter for 1 channel(totally separate spatial conv & depth conv)
- Low compute cost(compare with SqueezeNet)
- Almost equivalent accuracy level(SqueezeNet, AlexNet)

1. 将空间conv和深度conv完全分离开,提速
2. 相比SqueezeNet,计算量方面有明显提升(SqueezeNet似乎并没有考虑这个)
3. Accuracy上, MobileNet其实和AlexNet差不了太多,稍有提升

Table 3. Smaller MobileNet Compared to Popular Models				
Model	ImageNet Accuracy	Params	Multi-Add	FLOPs
MobileNet V1	71.0%	5.3M	1.1B	1.1B
SqueezeNet	71.0%	1.0M	1.2B	1.2B
AlexNet	57.0%	237M	1.1B	1.1B

### 3\_ Road Map

ShuffleNet(CVPR-2017, Face++)

ShuffleNet的几个创新点

1. 提出了一个类似于ResNet的BottleNeck单元。借鉴ResNet的旁路分支思想，ShuffleNet也引入了类似的网络单元。不同的是，在stride=2的单元中，用concat操作代替了add操作，用average pooling代替了1x1stride=2的卷积操作，有效地减少了计算量和参数。
2. 提出将1x1卷积采用group操作会得到更好的分类性能。在MobileNet中提出，1x1卷积的操作占据了约95%的计算量，所以作者将1x1也更改为group卷积，使得相比MobileNet的计算量大大减少。
3. 提出了核心的shuffle操作将不同group中的通道进行打散，从而保证不同输入通道之间的信息传递。解决多个group conv叠加出现的边界效应问题。Channels Shuffle操作会导致内存不连续这个影响有待评估。

- Channel shuffle for less cross-talk between channels(group conv issue)
- 1x1 group conv to reduce compute cost
- Use Residual block to raise accuracy

### 3\_ Road Map

ShuffleNet(CVPR-2017, Face++)

Model	Complexity(MFLOPs)	Ch. err. (%)	$\Delta$ err. (%)
1.0 MobileNet-224	569	29.4	-
ShuffleNet 2x (g=3)	524	26.3	3.1
ShuffleNet 2x (with SF[3], g=3)	527	24.7	4.7
0.75 MobileNet-224	328	31.6	-
ShuffleNet 1.5x (g=3)	292	28.5	3.1
0.5 MobileNet-224	149	36.3	-
ShuffleNet 1x (g=8)	140	32.4	3.9
0.25 MobileNet-224	41	49.4	-
ShuffleNet 0.5x (g=4)	38	44.6	7.8
ShuffleNet 0.5x (shallow, g=2)	40	42.8	6.6

- Lower Complexity than MobileNet
- Higher Accuracy than MobileNet

MFLOPs(Million Floating-point Operations per Second) 衡量计算复杂度的指标

### 3\_ Road Map

Small CNN Conclusion

- Separate convolution thoroughly (X-conv + Y-conv + Depth-conv)
- Raise utilization of channel information
- Residual block + Group Conv would be standard architecture



### 3\_ Road Map



### 3\_ Road Map

#### Network Compression Conclusion

- Distilling Knowledge(Teacher-student Framework)
- **Structure Pruning**(Deep Compression/Channel Pruning)
- **Quantizing parameters**(Deep Compression/Hashed-Net)
- **Low-bit Representation**(XOR-net, DOREFA-net, Squeeze the Last Bit)
- Low-rank Decomposition(Singular Value Decomposition/Tucker Decomposition)

- 五个大方向
1. 知识蒸馏, 用老师网络训练学生网络
  2. 结构剪枝, 去掉不重要的连接
  3. 量化参数, 就是用整数存, 不用浮点存
  4. 低bit表达, 把Weight, Activation, Gradient用低bit形式表达, 让卷积的点乘运算优化成移位甚至是同或运算
  5. 低秩分解, 先训练一个模型, 然后用矩阵的低秩分解将一个大的Weight矩阵分解成几个小的, 降低计算量

### 3\_ Road Map

#### Distilling Knowledge

老师网络是一个甚至好几个复杂网络, 学生网络是一个结构相对简单的网络

Distillation是先训练老师网络, 得到logits层的输出, 在softmax函数中加T参数计算Soft Target, 用Soft Target结合Hard Target(就是原来训练数据的标签)作为学生网络训练数据的标签

1. 首先, Distilling Knowledge是迁移学习的一种发展, 一般的迁移学习是直接利用学到的权重, 而蒸馏模型所迁移的是标签
2. Soft Target是给Softmax函数加一个T参数, 让学生网络不止是学最终的分, 也要学老师网络中给这张图预测了哪几种类型, 概率各是多少:这样, 把原先的有监督分类问题, 变成了一个有监督回归问题  
Soft Target的作用就是作为Regularizers, 它弥补了分类问题中监督信号不足的问题, 降低了搜索空间, 从而提高了generalization
3. 在竞赛中, 常用集成学习的方法来提高准确率, 就是用好几个网络训练的结果来投票, 而这里用集成学习方法训练老师网络, 我们可以直观理解为不止一个老师, 要好几个老师来教一个学生

- Teacher-student Framework(Transfer learning)
- Soft Target -> Change Classification to Regression
- Use Ensemble Learning to get high performance(More than 1 teacher)

Geoffrey Hinton et al.[Google], "Distilling the Knowledge in a Neural Network", NIPS 2014

### 3\_ Road Map

#### Accompanying Readings

- Lei Jimmy Ba et al(Toronto), "Do Deep Nets Really Need to be Deep?", NIPS 2014
- Ping Luo et al(CUHK), "Face Model Compression by Distilling Knowledge from Neurons", AAAI 2016
- Guobin Chen(NEC & Missouri), "Learning Efficient Object Detection Models with Knowledge Distillation", NIPS 2017
- Junho Yim et al(KAIST), "A Gift from Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning", CVPR 2017

### 3\_ Road Map

#### Deep Compression (ICLR-2016 best paper, Stanford)

1. 剪枝, 去掉一些不重要的连接(训练出来几乎为0权重的连接), Han Song在ICLR领奖的演讲这儿说的很有意思, 剪枝相当于一个人喝大了, 但又没有丧失意识, 可以做一些简单的任务, 所以对简单任务, 我们不需要那么多的冗余连接
2. 权值共享是用少的bit数来存之前都是浮点数的权重, 而且只存权重的shift和code book, 就相当于训练这边编密码, 解释的时候用code book来解码
3. Huffman按照符号出现的概率来进行变长编码, 可以进一步减轻存储参数的Memory压力, 过去图像压缩经常会用到

- Pruning/Sparse Connection
- Quantization/Weight sharing
- Huffman Encoding

### 3\_ Road Map

#### Deep Compression (ICLR-2016 best paper, Stanford)

1. 几乎没有准确率的损失
2. 如果和小模型搭配使用, 将更有利于在移动端部署
3. 虽然模型被压缩了, 但模型的解释成本增加了, 这一点还要double confirmed一下  
因为模型的稀疏连接, 需要特殊的库来处理, 所以这种压缩方法在模型解释的时候一定是增加了成本

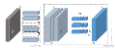
Model	Original Channels	Sparsity	Original Accuracy	Compressed Accuracy
ResNet-50	256	40%	80.5%	80.5%
VGG-16	128	30%	73.7%	73.7%
GoogLeNet	96	75%	69.0%	69.0%
MobileNet	96	75%	55.8%	55.8%
Inception-v3	256	40%	73.3%	73.3%
EfficientNet	128	30%	73.7%	73.7%

- Almost no accuracy lost
- Combining with small network would be more useful in embedded platform
- High model interpretation cost? Need to be verified

### 3\_ Road Map

#### Channel Pruning (ICCV-2017, Face++)

1. 结构化剪枝, 不会造成稀疏连接, 所以解释的成本要更低
2. 把求解mask  $\beta$  当做一个优化问题求解
3. 对于ResNet进行通道剪枝的时候, 增加一个sampler block, 使两个input的通道一样



- Structure Pruning. Lower model interpretation cost
- Compute mask  $\beta$  by solving an optimization problem
- Add *sampler* block for multi-branch networks(e.g. ResNet)

$$\min_{\beta} \sum_{i=1}^n \sum_{j=1}^m \beta_{ij} \left( \sum_{k=1}^n \beta_{ik} \right) \quad \text{s.t.} \quad \beta_{ij} \in \{0, 1\}$$



### 3\_ Road Map

#### Channel Pruning (ICCV-2017, Face++)

Model	Original Channels	Sparsity	Original Accuracy	Compressed Accuracy
ResNet-50	256	40%	80.5%	80.5%
VGG-16	128	30%	73.7%	73.7%
GoogLeNet	96	75%	69.0%	69.0%
MobileNet	96	75%	55.8%	55.8%
Inception-v3	256	40%	73.3%	73.3%
EfficientNet	128	30%	73.7%	73.7%

Note: 1. Accuracy is the mean of 100 trials. 2. Sparsity is the ratio of zero channels.

1. 和Deep Compression不一样, channel pruning会造成精度损失, 但是结构化剪枝将来模型解释的成本低
2. 剪枝完fine-tune一下效果会好很多
3. 增加了很多hyper-para, 比如正则化系数 $\lambda$ , 还有压缩的比例

- Accuracy Lost
- Need fine-tune after pruning
- More hyper-para(regular coefficient  $\lambda$ , compress ratio) need be manually set, increase tuning difficult

### 3\_ Road Map

#### Accompanying Readings

- Yuhui He et al(Face++), "Channel Pruning for Accelerating Very Deep Neural Networks", ICCV 2017
- W. Chen et al(Washington), "Compressing Neural Networks with the Hashing Trick", ICML 2015

### 3\_ Road Map Low Bit Representation

**XNOR-Net**的目的有两点一是降低模型大小,二是简化计算  
分两个层次来实现  
一是Binary Weight,就是Weight每次更新完用符号位作为二值和该通道所有Weight的L1范数作为scale把所有参数都变成-1和1  
原先的训练框架没有变,只是在计算Weight和Weight导数的时候,要增加二值过程和二值的导数(二值是没有导数的,近似成了 $\text{HTanh}(x)$ )。这样确实省了存储空间,而且性能也没降低,但是Compute cost没降多少  
二是XNOR-Net,就是不光要把Weights都变成-1和1,还要把激活值都变成-1和1,这样 $A \cdot W$ 原先要做的点乘运算就变成了XNOR(同或)运算,大大降低计算代价。但是feature map的表现力下降了(这相当于给feature map做了个dither),所以Acc就降的比较狠

二值是low bit representation最极端的形式,用三值或者bit位更高一点效果会好一些

XNOR-Net最重要的贡献在于把最耗资源的点乘运算换成了同或运算,而且这个方法已经商业化了,公司叫XNOR.ai,这个团队把之前的YOLO二值化了,现在位图A表示的Raspberry Pi Zero,效果跟标准二值网络给在芯片上实现神经网络提供了绝对的方向  
What to do to reduce weights Activation, low Memory cost

- Use scaling factor to improve accuracy
- Change dot function to Xnor calculation(low Compute cost)

Muhammad Rastegari et al(Washington), "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks", ECCV 2016

### 3\_ Road Map

#### Accompanying Readings

- Yoshua Bengio et al.(Montreal), "Binarized Neural Networks: Training Neural Networks with Weights and Activations Constrained to +1 or -1", NIPS 2016
- Shuchang Zhou et al.(Face++), "DOREFA-NET: TRAINING LOW BITWIDTH CONVOLUTIONAL NEURAL NETWORKS WITH LOW BITWIDTH GRADIENTS", CVPR 2016
- Song Han et al.(Stanford), "Trained Ternary Quantization", ICLR 2017
- Cong Leng et al.(Alibaba), "Extremely Low Bit Neural Network: Squeeze the Last Bit Out with ADMM", AAAI 2018

### 3\_ Road Map

#### Low-rank Decomposition

1. 基本的思路是在通道维度上进行PCA把维度降下来,通过SVD的手段
2. 如果每一层单纯降秩会产生累加误差,浅层网络还好,深层网络误差就会比较大,所以这里创新了一个非对称重构的方法,在每层低秩逼近的时候,优化的目标不是上一层低秩近似的输入产生的结果,而是原始输入的结果  
这样把误差控制在当前层,不会向后累加,但是这就变成了一个GSVD解决的问题,就是最终要近似的目标不是本身
3. 吸收了前人的做法,不止在通道维度上压缩,还在空间维度上压缩,那这个就和Inception-V3的做法很像了,但是V3是先设计网络直接就用这个网络训练,而降秩是先训练再分解网络

- SVD for conv filter low-rank decomposition
- Not only use SVD on channel dim but also spatial dim(3x3 -> 3x1+1x3)
- Asymmetric Reconstruction for Multi-Layer/Solve accumulated error in deep layer

Xingyao Zhang et al(Face++), "Accelerating Very Deep Convolutional Networks for Classification and Detection", TPAMI 2016

### 3\_ Road Map

#### Accompanying Readings

- Max Jaderberg et al.(Oxford), "Speeding up Convolutional Neural Networks with Low Rank Expansions", BMVC 2014
- Shaohui Lin et al.(Tencent), "Towards Convolutional Neural Networks Compression via Global Error Reconstruction", IJCAI 2016
- Yong-Deok Kim et al.(Samsung), "NETWORKS FOR FAST AND LOW POWER MOBILE APPLICATIONS", ICLR 2016
- P. Wang et al.(CAS), "Accelerating Convolutional Neural Networks for Mobile Applications", ACM MM 2016

## 04 Start a project

### 4\_ Start a project



Deep learning Toolkits



Distribution Learning



Deployment on Mobile



Issues when Training

### 4\_ Start a project

#### Deep Learning Toolkits Comparison



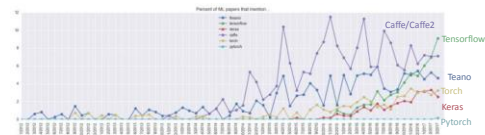
1. Caffe从UC Berkeley的实验室出来,最开始的作者是贾扬清,后来被Facebook收购,因为Caffe灵活性不足,贾扬清在Facebook内部又搞了Caffe2,估计Caffe用户一两年内都要迁移到其他框架上去
2. PyTorch目前也由Facebook团队维护
3. Theano开创了深度学习框架符号编程的先河(相比于caffe的命令编程),17年停止维护
4. Tensorflow由Google团队维护,所以我们看到前面Google发的paper框架一定都是tensorflow的
5. MXNet之前是一帮大学的研究人员维护(Xgboost作者华盛顿大学陈天奇在内,美国,加拿大,新加坡的多所高校),17年被Amazon接盘

## 4\_ Start a project

### Deep Learning Toolkits Comparison

% of projects	Frameworks	has been around for (years)
0-1	TensorFlow	16
1-2	Caffe	17
2-3	PyTorch	17
3-4	PyTorch	17
4-5	PyTorch	17
5-6	PyTorch	17
6-7	PyTorch	17
7-8	PyTorch	17
8-9	PyTorch	17
9-10	PyTorch	17

Tensorflow用户数量这两年稳定增长,到2017.3已经处于不可撼动的地位  
Caffe2/Pytorch都开源时间不长,数量比较少  
MXNet虽然性能很优,但是一直没有火起来,  
Amazon加入后不知道是否会有改观



## 4\_ Start a project

1. TensorFlow的优势在于社区庞大,对移动端支持较好,可以快速地开发一个部署在Android系统上的应用,缺点是性能比较差,据说google内部对于内存和GPU优化是做了后门的,而且调试困难
2. Facebook同时支持Caffe2和Pytorch两个框架,意图很明显,Pytorch用于研究,架构优雅简介,方便快速的验证想法,Caffe2主攻工业应用,性能已经可以和MXNet比肩了,并且重视移动端的支持
3. MXNet最大的优势在于对内存的优化比较极致,相同的内存可以训练更大的网络,但这点估计很快要被Caffe2超越

### Deep Learning Toolkits Comparison

Toolkit	Modeling Capability	Interfaces	Model Deployment	Architecture Complexity	Training Speed (VGG-style CNN on CIFAR-10)	Distribution Training
TensorFlow	CNN/RNN/LSTM/etc.	Python/C++/Go/Java	OS/Android/iOS	★★★★★	173s	
Caffe2	CNN/RNN/LSTM/etc.	C++/Python/Matlab	OS/Android/iOS/ARM	★★★★☆	149s	
MXNet	CNN/RNN/LSTM/etc.	Python/C++/Java/Julia	OS/Android/iOS	★★★★★	149s	
Pytorch	CNN/RNN/LSTM/etc.	Python/Lua	OS/Android	★★★★★	168s	

## 4\_ Start a project

还有一些针对特定任务的框架,比如纯C写的darknet,还有Facebook新出的针对目标检测的detection

### Deep Learning Toolkits Conclusion

- **TensorFlow** is a safe bet for most projects, Not perfect but has huge community, wide usage.
  - **Pytorch** is best for research. However still new, there can be rough patches
  - Use **TensorFlow** for one graph over many machines
  - Consider **Caffe2** or **TensorFlow** for production deployment
  - Consider **TensorFlow** or **Caffe2** for mobile
1. TensorFlow虽然不完美,但是利于部署,有稳定社群,同时上层API如Keras支持Tensorflow,便于开发,在Android上部署有天然的优势,比如Tensorflow Lite支持用Android 8.1神经网络API利用手机GPU进行加速,得益于Google的这点好处还是别人没法比的
  2. PyTorch虽适合研究,但是它很新,因此用的时候有很多就要自己填
  3. 除了Tensorflow, Caffe2也可以考虑用于产品部署
  4. 手机端可以考虑TensorFlow或Caffe2
- 总结: 框架相当于刻刀, 雕塑的好坏还是要取决于匠人, 框架就是框架, 关键是要出活儿

## 4\_ Start a project



Deep learning Toolkits



Distribution Learning



Deployment on Mobile



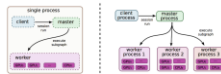
Issues when Training

## 4\_ Start a project

### Basic Principle(Single-device & Multi-GPU)

1. 假设你的机器上有3个GPU;
2. 在单机单GPU的训练中, 数据是一个batch一个batch的训练。在单机多GPU中, 数据一次处理3个batch(假设是3个GPU训练), 每个GPU处理一个batch的数据计算。
3. 变量, 或者说参数, 保存在CPU上
4. 刚开始的时候数据由CPU分发给3个GPU, 在GPU上完成了计算, 得到每个batch要更新的梯度。
5. 然后在CPU上收集完了3个GPU上的要更新的梯度, 计算一下平均梯度, 然后更新参数。
6. 然后继续循环这个过程。

[https://github.com/TonyChang/Distribute\\_MNIST](https://github.com/TonyChang/Distribute_MNIST)



## 4 Start a project



- Cluster -> The whole system for training assignment
- Job -> parameter server/worker(for calculation)
- Task -> single process on one device

Cluster, Job, Task 三者可以简单的看成是层次关系, Task可以看成是机器上的一个进程, 多个task组成job; job又有: ps, worker两种, 分别用于参数服务、计算服务, 所有job组成cluster

同步更新指各个用于并行计算的电脑, 计算完各自的batch后, 求取梯度值, 把梯度值统一送到ps服务器机器中, 由ps服务器求取梯度平均值, 更新ps服务器上的参数。

如下图所示, 可以看成四台电脑, 第一台电脑用于存储参数、共享参数、共享计算, 可以简单的理解成内存、计算共享专用的区域, 也就是ps job; 另外三台电脑用于并行计算的, 也就是worker job

这跟刚才介绍的单机多GPU的过程一样, 只是我们专门用一台device来完成参数服务的工作。同步更新的问题就是速度取决于最慢的那台机器, 所以最好三台机器的性能差不多, 否则性能高的机器完全没有体现出来优势

异步更新是指ps服务器收到只要收到一台机器的梯度值, 就直接进行参数更新, 无需等待其它机器。这种迭代方法比较不稳定, 收敛曲线波动比较厉害, 因为当A机器计算完更新了ps中的参数, 可能B机器还是在用上一次迭代的旧版参数值

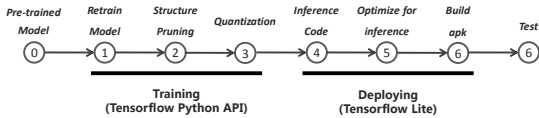
<https://www.tensorflow.org/deploy/distributed>

## 4\_ Start a project

- Deep learning Toolkits
- Distribution Learning
- Deployment on Mobile
- Issues when Training

## 4\_ Start a project

### Deployment on Mobile



<https://www.tensorflow.org/tutorials/deploy-on-android/w/>  
<https://www.oreilay.com/learning/tensorflow-for-mobile-ports>

## 4\_ Start a project



Deep learning Toolkits



Distribution Learning



Deployment on Mobile



Issues when Training

## 4\_ Start a project

### Issues when Training

- **Lack of Training Data** -> Data Augmentation/Transfer Learning
- **Low Accuracy** -> Ensemble learning/Feature stacking
- **Model convergence Issue** -> Gradient Descent Optimization/Activation function
- **Local Optima** -> Weight Initialization/Learning rate auto-tuning
- **Overfitting** -> Regularization

