Survey on DL@UAV vision

CONTENT

What is DL?

Application

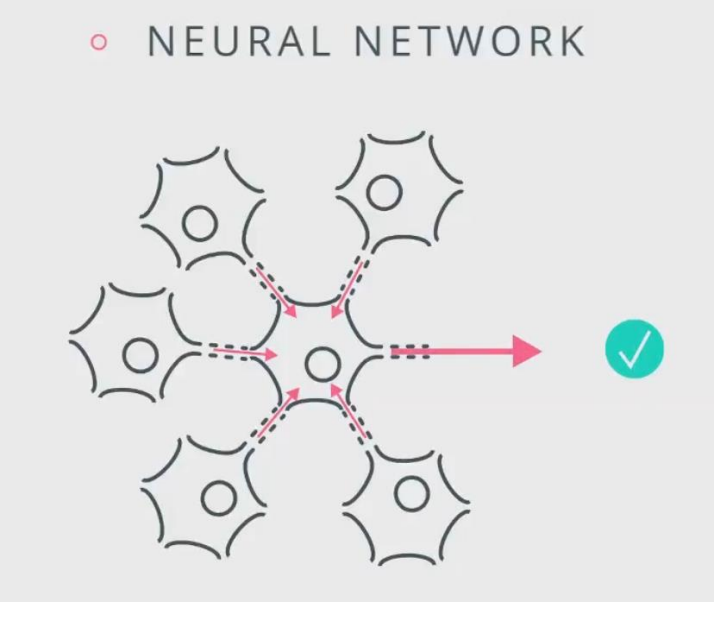
Road Map

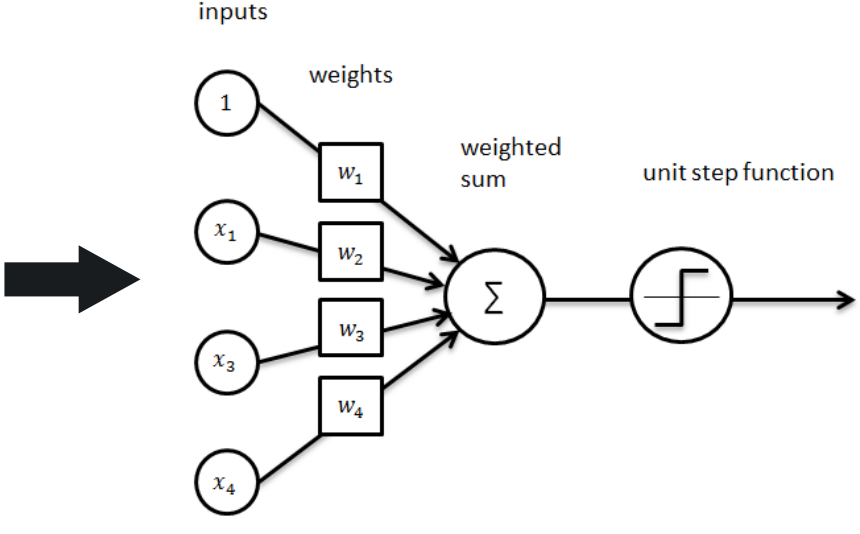
Start a Project

What is DL?

1_ What is DL

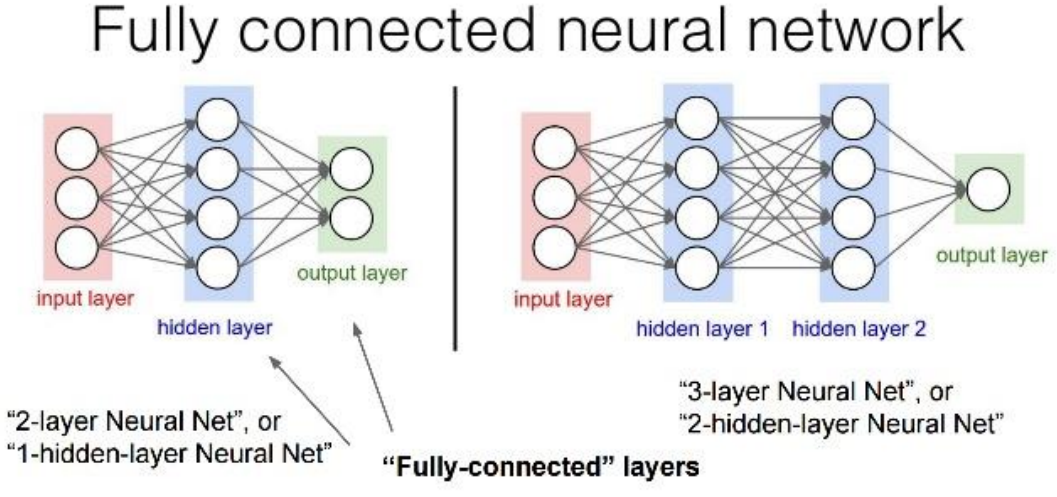
From **Neurons** to **Perceptron**





1_ What is DL

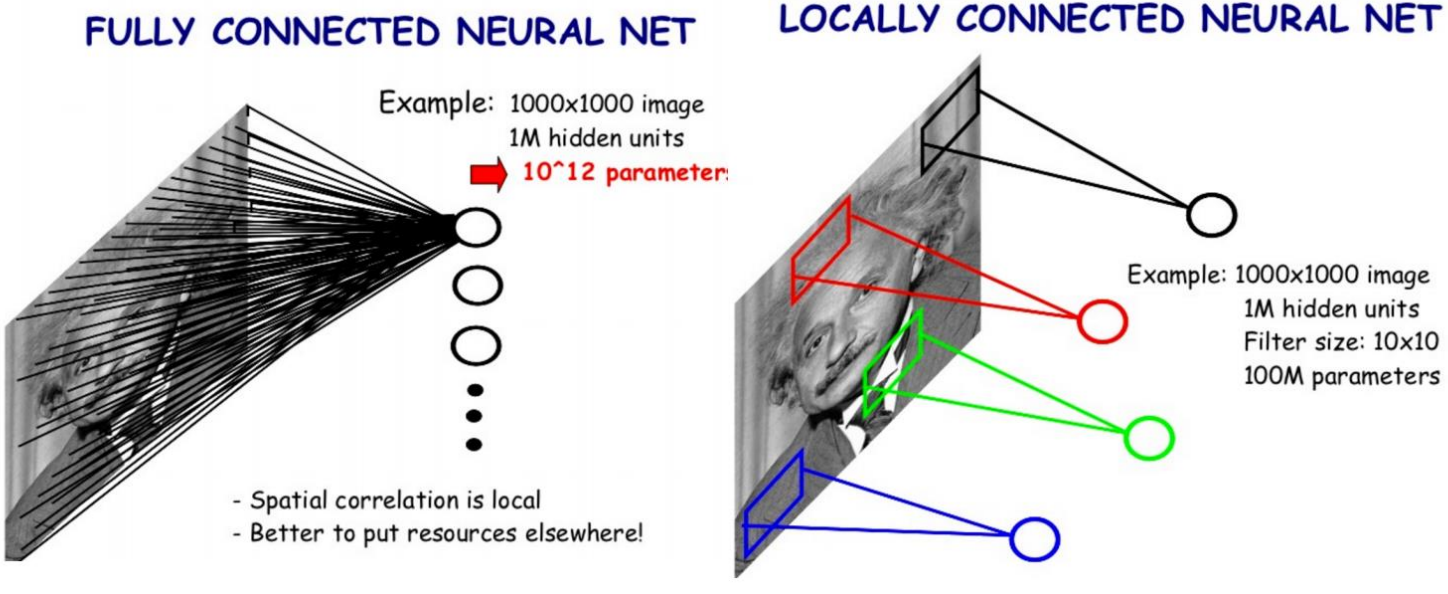
FC – Fully Connection



- Hidden Layer
- Nonlinear Activation

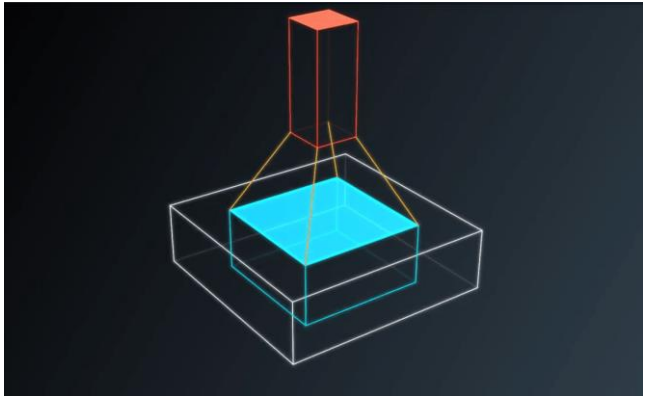
1_ What is DL

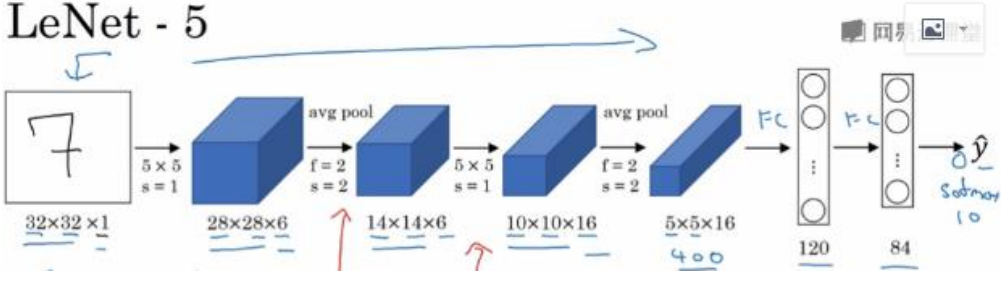
CNN – Convolutional Neural Network



1_ What is DL

CNN – Convolutional Neural Network



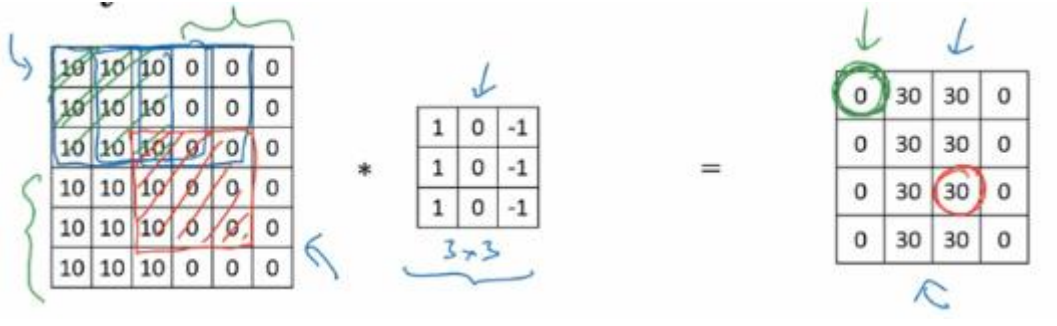


- Convolutional Kernel dimension

width * height * input channel * filter number

1_ What is DL

Why Convolution(compare with FC)



Vertical edge detector

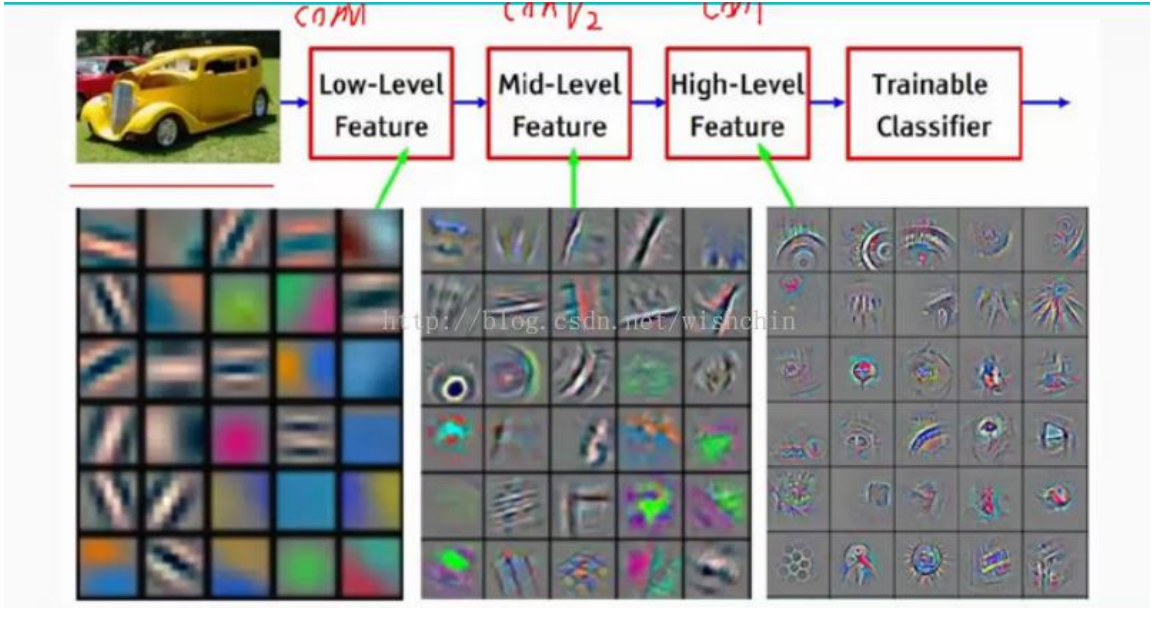
- **Parameter sharing:**
  A feature detector(such as a vertical edge detector) that is useful in one prat of the image is probably useful in another part of the image

- **Sparse connection**
  In each layer, each output value depends only on a small number of inputs

- **Less Parameters, Low Overfitting Risk**

1_ What is DL

CNN as Feature Extractor



edge/color texture structure

CNN Visualization

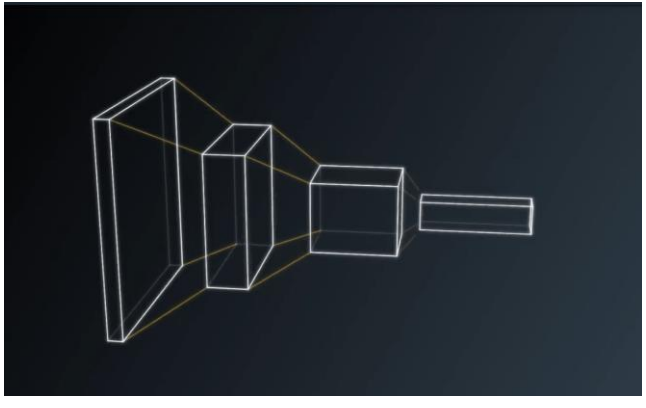
Application

2_ Application

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

2_ Application

Object Recognition

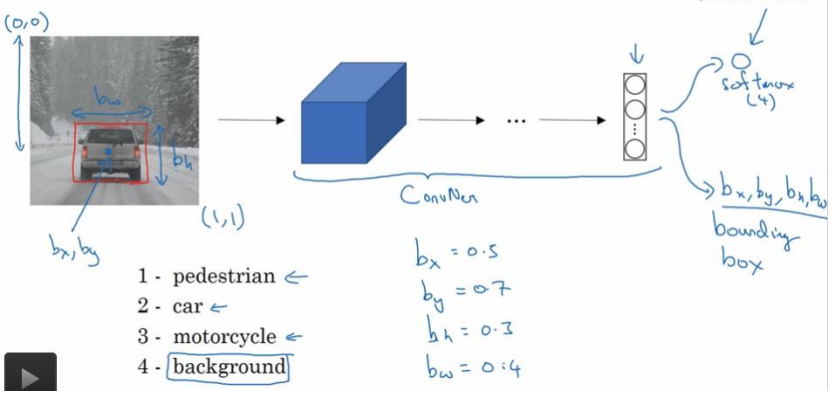


Classification

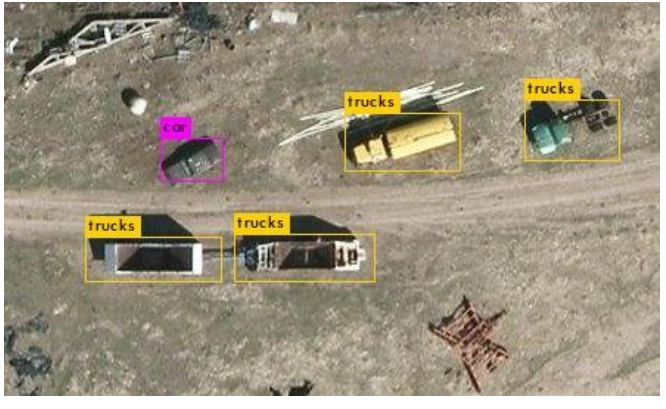
Dogs vs. Cats in Kaggle

2_ Application

Object Detection



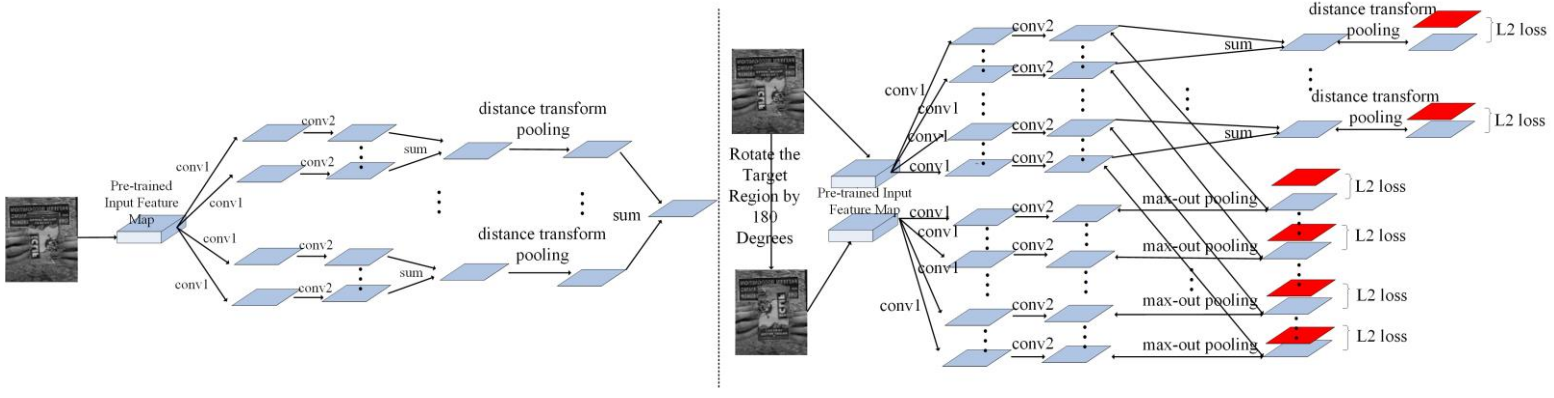
Classification + multi-object localization



VEDAI UAV Object Detection Dataset

2_ Application

VOT

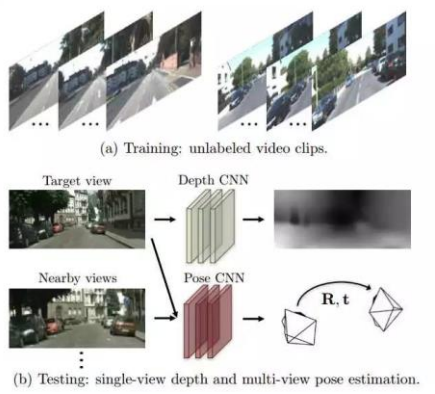


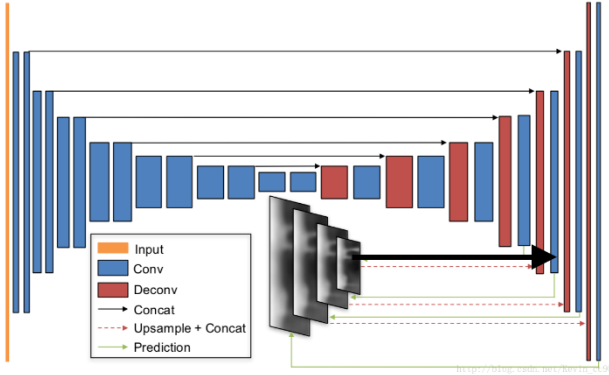
LSART – VOT 2017 Championship

- 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

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)

Road Map

3_ Road Map

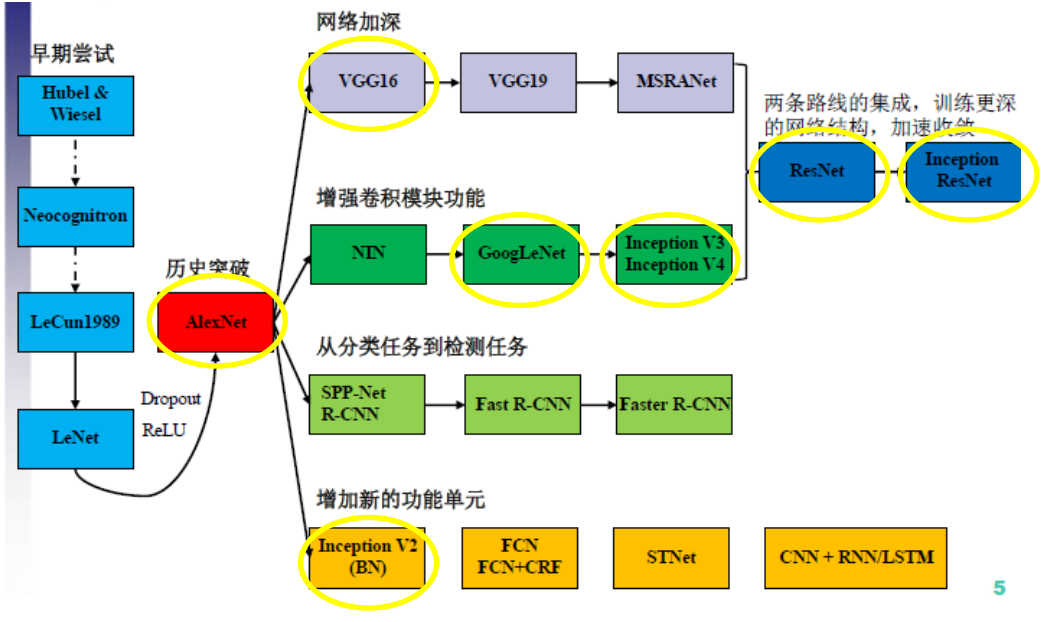
Introduction of CNN Development

Small CNN for mobile application

Network Compression

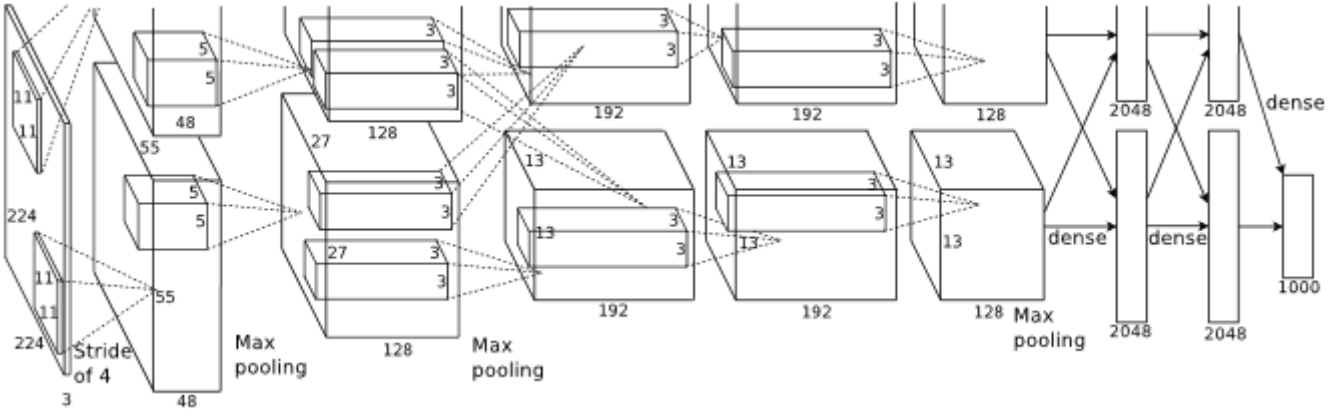
3_ Road Map

Development of CNN



3_ Road Map

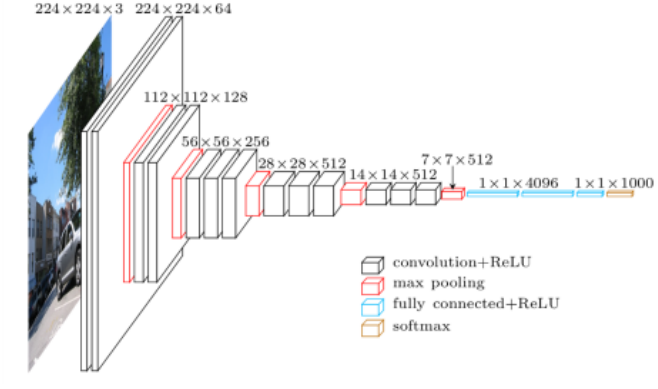
AlexNet – 2012 ImageNet Championship (Top-5 Error 16.4%)



- 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

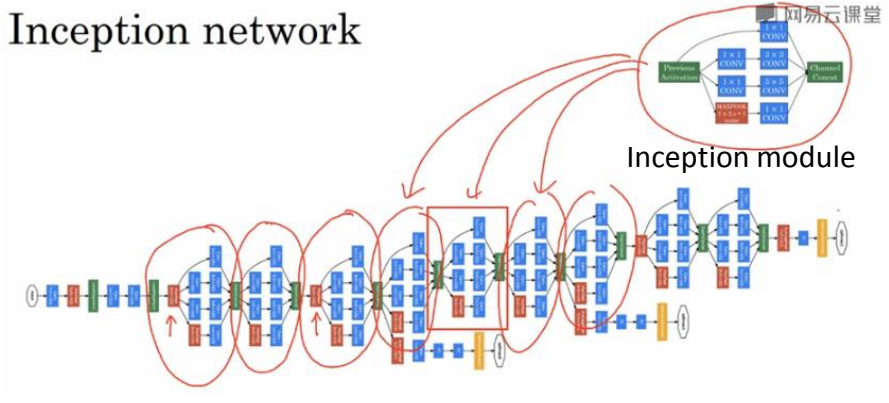
VGGNet – 2014 ImageNet (Top-5 Error 9.9%)



- 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%)



- 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

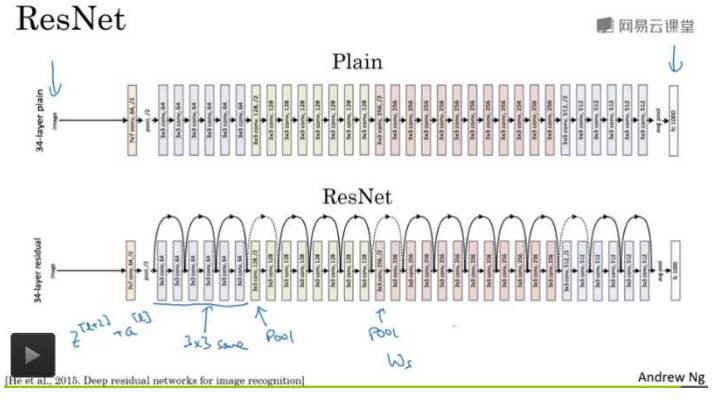
3_ Road Map

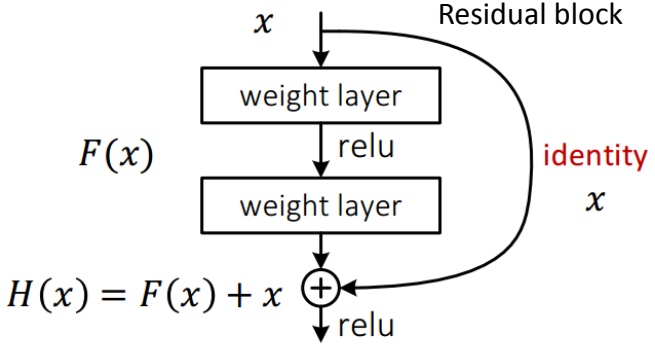
Inception Family

- 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

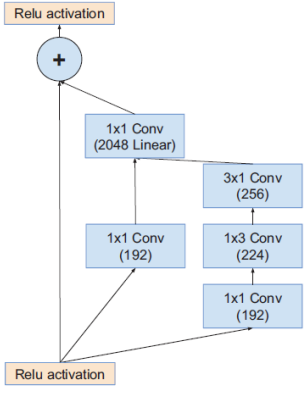


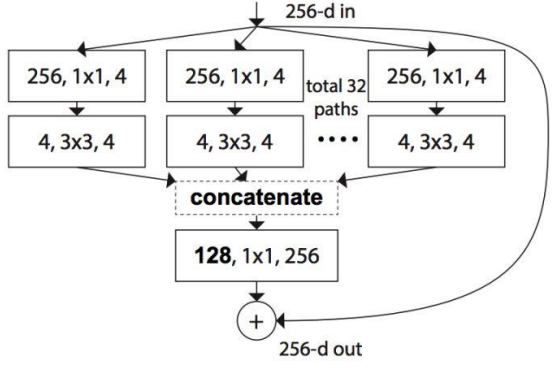


- Shortcut connection/solve Gradient vanishing
- Break the depth limitation

3_ Road Map

Inception-ResNet & ResNeXt(Top-5 Error 4.9% & 4.4%)





Inception Module + shortcut connection

Residual block + Group conv(**cardinality**)

3_ Road Map

Cost Comparison

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

3_ Road Map

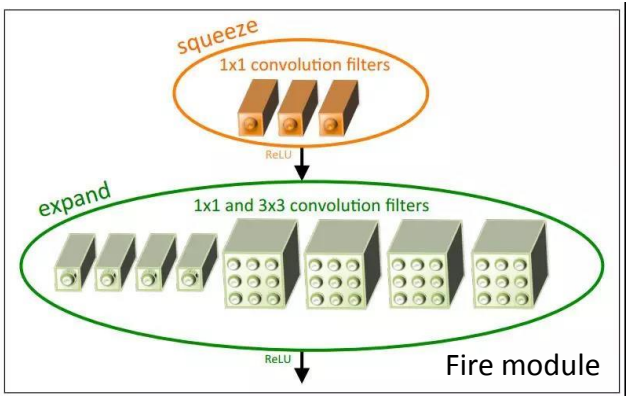
Introduction of CNN Development

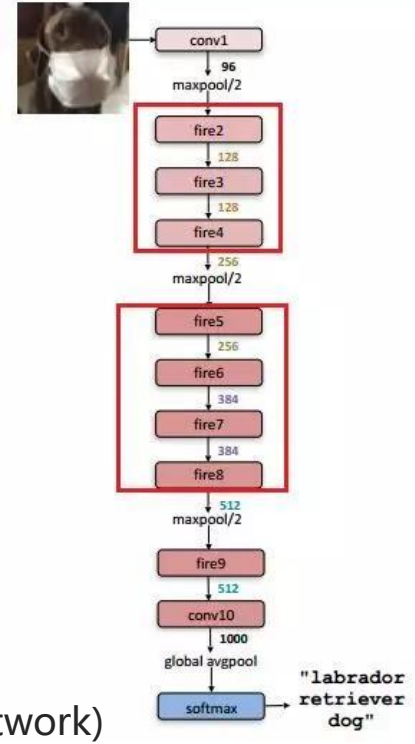
Small CNN for mobile application

Network Compression

3_ Road Map

SqueezeNet(ICLR-2017, Stanford)





- 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	Compression Approach	Data Type	Original → Compressed Model Size	Reduction in Model Size vs. AlexNet	Top-1 ImageNet Accuracy	Top-5 ImageNet Accuracy
AlexNet	None (baseline)	32 bit	240MB	1x	57.2%	80.3%
AlexNet	SVD (Denton et al., 2014)	32 bit	240MB → 48MB	5x	56.0%	79.4%
AlexNet	Network Pruning (Han et al., 2015b)	32 bit	240MB → 27MB	9x	57.2%	80.3%
AlexNet	Deep Compression (Han et al., 2015a)	5-8 bit	240MB → 6.9MB	35x	57.2%	80.3%
SqueezeNet (ours)	None	32 bit	4.8MB	**50x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	8 bit	4.8MB → 0.66MB	**363x**	57.5%	80.3%
SqueezeNet (ours)	Deep Compression	6 bit	4.8MB → 0.47MB	**510x**	57.5%	80.3%

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

3_ Road Map

MobileNet(CVPR-2017, Google)

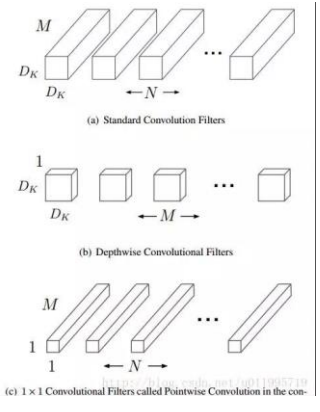


Table 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60

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

3_ Road Map

ShuffleNet (CVPR-2017, Face++)

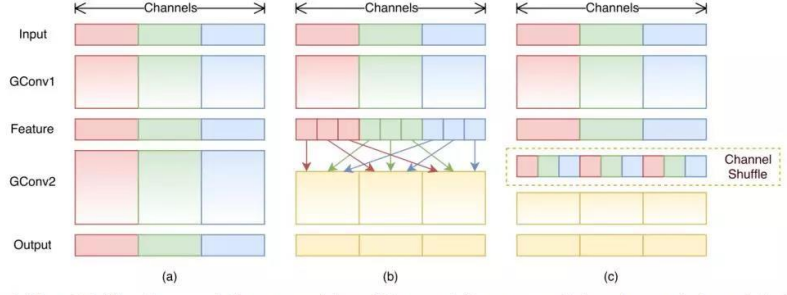
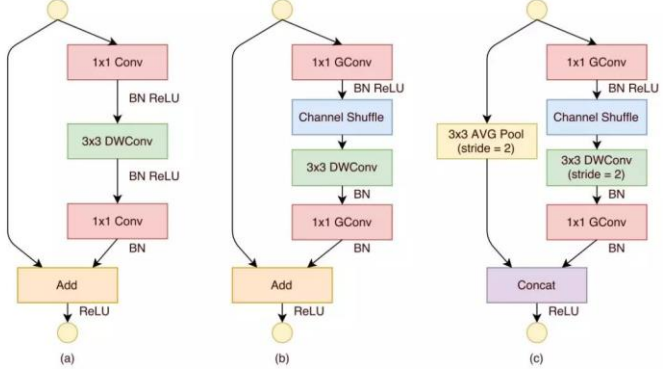


Figure 1. Channel shuffle with two stacked group convolutions. GConv stands for group convolution. a) two stacked convolution layers with the same number of groups. Each output channel only relates to the input channels within the group. No cross talk; b) input and output channels are fully related when GConv2 takes data from different groups after GConv1; c) an equivalent implementation to b) using channel shuffle.



- 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)	Cls err. (%)	Δ err. (%)
1.0 MobileNet-224	569	29.4	-
ShuffleNet 2× ($g = 3$)	524	**26.3**	3.1
ShuffleNet 2× (with *SE*[13], $g = 3$)	527	**24.7**	4.7
0.75 MobileNet-224	325	31.6	-
ShuffleNet 1.5× ($g = 3$)	292	**28.5**	3.1
0.5 MobileNet-224	149	36.3	-
ShuffleNet 1× ($g = 8$)	140	**32.4**	3.9
0.25 MobileNet-224	41	49.4	-
ShuffleNet 0.5× ($g = 4$)	38	**41.6**	7.8
ShuffleNet 0.5× (shallow, $g = 3$)	40	42.8	6.6

Table 5. ShuffleNet vs. MobileNet [12] on ImageNet Classification

- Lower Complexity than MobileNet
- Higher Accuracy than MobileNet

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

Introduction of CNN Development

Small CNN for mobile application

Network Compression

3_ Road Map

Deep Compression (ICLR-2016 best paper, Stanford)

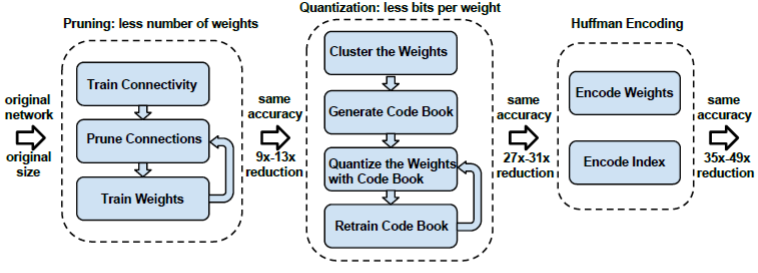
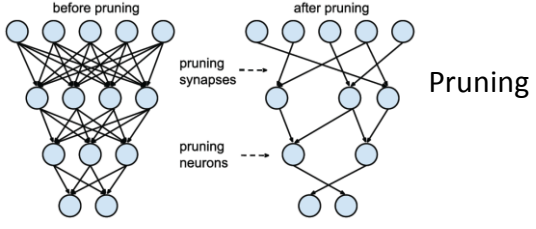
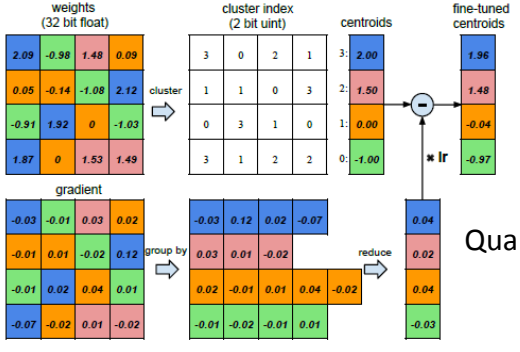


Figure 1: The three stage compression pipeline: pruning, quantization and Huffman coding. Pruning reduces the number of weights by 10×, while quantization further improves the compression rate: between 27× and 31×. Huffman coding gives more compression: between 35× and 49×. The compression rate already included the meta-data for sparse representation. The compression scheme doesn't incur any accuracy loss.



Pruning



Quantization

Figure 3: Weight sharing by scalar quantization (top) and centroids fine-tuning (bottom).

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

3_ Road Map

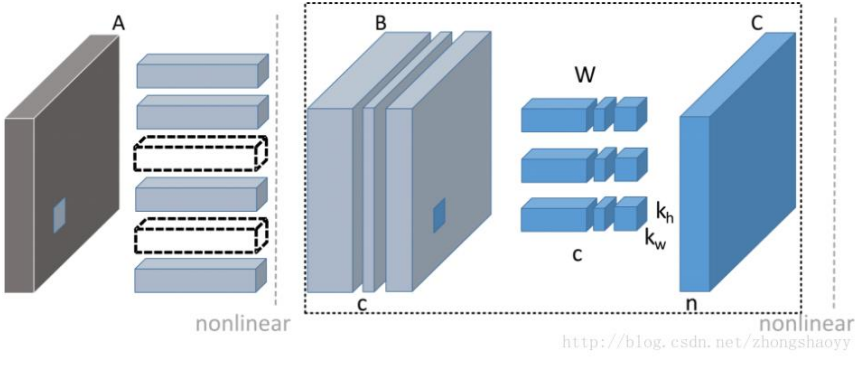
Deep Compression (ICLR-2016 best paper, Stanford)

Network	Original Size	Compressed Size	Compression Ratio	Original Accuracy	Compressed Accuracy
Lenet-300-100	1070KB	27KB	**40x**	98.36%	98.42%
Lenet-5	1720KB	44Kb	**39x**	99.20%	99.26%
AlexNet	240MB	6.9MB	**35x**	80.27%	80.30%
VGGNet	550MB	11.3MB	**49x**	88.68%	89.09%
GoogleNet	28MB	2.8MB	**10x**	88.90%	88.92%
SqueezeNet	4.8MB	0.47MB	**10x**	80.32%	80.35%

- 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++)



$$\arg\min_{\beta,W} \frac{1}{2N} \left\| Y - \sum_{i=1}^{c} \beta_i X_i W_i^{\top} \right\|_F^2 + \lambda \|\beta\|_1 \quad (2)$$

$$\text{subject to } \|\beta\|_0 \le c', \forall i \|W_i\|_F = 1$$

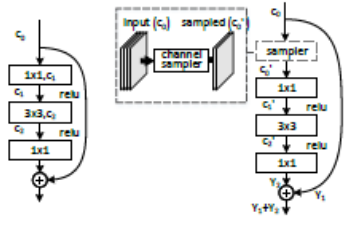


Figure 3. Illustration of multi-branch enhancement for residual block. **Left**: original residual block. **Right**: pruned residual block with enhancement, c_n denotes the feature map width. Input channels of the first convolutional layer are sampled, so that the large input feature map width could be reduced. As for the last layer, rather than approximate Y_2, we try to approximate $Y_1 + Y_2$ directly (Sec. 3.3 Last layer of residual branch).

- Structure Pruning, Lower model interpretation cost
- Compute mask β by solving a optimization problem
- Add *sampler* block for multi-branch networks(e.g. ResNet)

3_ Road Map

Channel Pruning (ICCV-2017, Face++)

Increase of top-5 error (1-view, baseline 89.9%)			
Solution	2×	4×	5×
Jaderberg *et al.* [22] ([52]'s impl.)	-	9.7	29.7
Asym. [52]	0.28	3.84	-
Filter pruning [31] (fine-tuned, our impl.)	0.8	8.6	14.6
Ours (without fine-tune)	2.7	7.9	22.0
Ours (fine-tuned)	0	1.0	1.7

Table 1. Accelerating the VGG-16 model [43] using a speedup ratio of 2×, 4×, or 5× (*smaller is better*).

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

3_ Road Map

Squeeze the Last Bit Out (AAAI-2018, Alibaba)

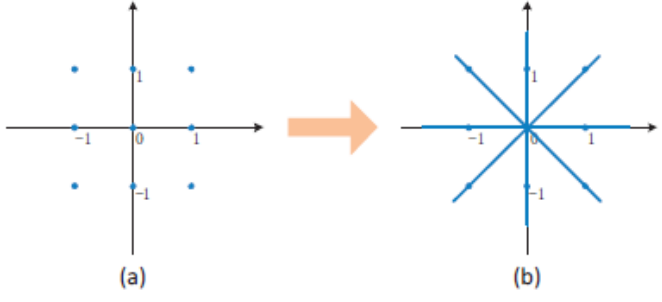


Figure 1: In ternary neural network, scaling factor expands the constrained space from (a) nice discrete points to (b) four lines in the space (two dimensional space as an example).

$$W^{k+1} := \arg\min_{W} L_\rho(W, G^k, \lambda^k) \quad (8)$$

$$G^{k+1} := \arg\min_{G} L_\rho(W^{k+1}, G, \lambda^k) \quad (9)$$

$$\lambda^{k+1} := \lambda^k + W^{k+1} - G^{k+1} \quad (10)$$

	Accuracy	Binary	BWN	Ternary	TWN	{-2, +2}	{-4, +4}	Full Precision
AlexNet	Top-1	**0.570**	0.568	**0.582**	0.575	0.592	0.600	0.600
	Top-5	**0.797**	0.794	**0.806**	0.798	0.818	0.822	0.824
VGG-16	Top-1	**0.689**	0.678	**0.700**	0.691	0.717	0.722	0.711
	Top-5	**0.887**	0.881	**0.896**	0.890	0.90?	0.909	0.899

Table 1: Accuracy of AlexNet and VGG-16 on ImageNet classification.

	Accuracy	Binary	BWN	Ternary	TWN	{-2, +2}	{-4, +4}	Full Precision
Resnet-18	Top-1	**0.648**	0.608	**0.670**	0.618	0.675	0.680	0.691
	Top-5	**0.862**	0.830	**0.875**	0.842	0.879	0.883	0.890
Resnet-50	Top-1	**0.687**	0.639	**0.725**	0.656	0.739	0.740	0.753
	Top-5	**0.886**	0.851	**0.907**	0.865	0.915	0.916	0.922

Table 2: Accuracy of ResNet-18 and ResNet-50 on ImageNet classification.

- Low Bit Networks
- ADMM algorithm
- Even higher accuracy in VGG-16, Compression has regularization effect

3_ Road Map

Network Compression Conclusion

- Pruning
- Quantization
- Low-Bit Network(Binary Network, Ternary Network)
- Low rank decomposition
- Teacher-student Framework

Start a project

4_ Start a project

Deep learning Toolkits

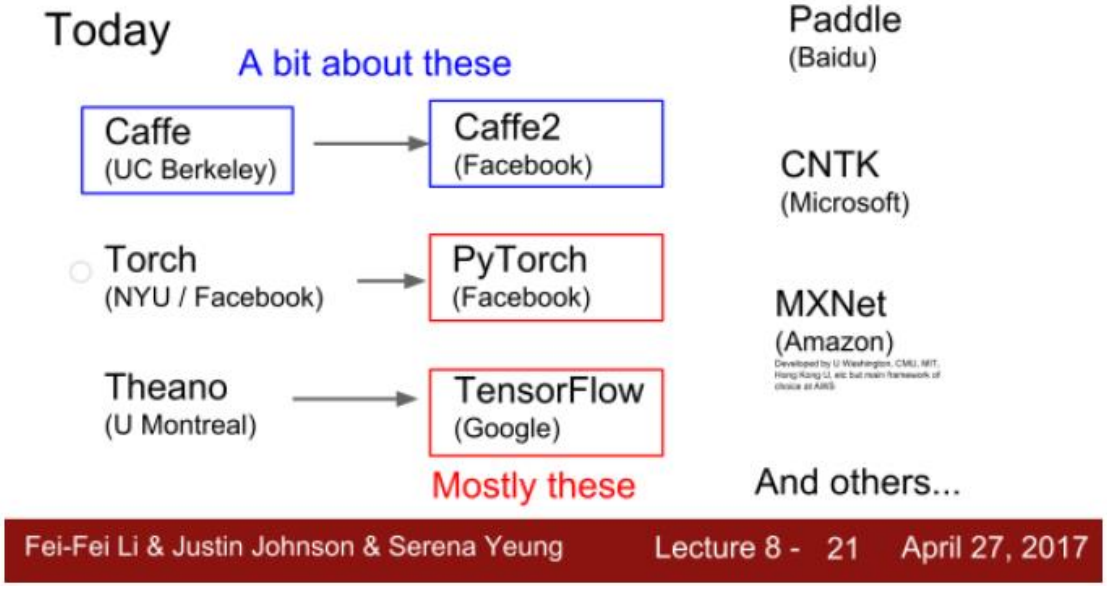
Distribution Learning

Deployment on Mobile

Issues when Training

4_ Start a project

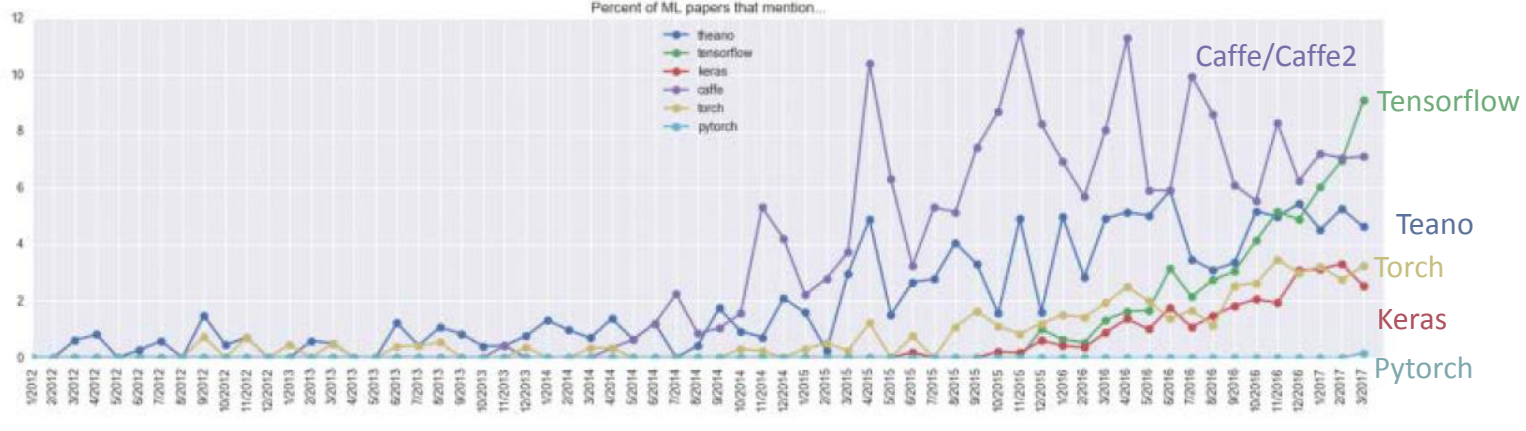
Deep Learning Toolkits Comparison



4_ Start a project

Deep Learning Toolkits Comparison

% of papers	framework	has been around for (months)
9.1	tensorflow	16
7.1	caffe	37
4.6	theano	54
3.3	torch	37
2.5	keras	19
1.7	matconvnet	26
1.2	lasagne	23
0.5	chainer	16
0.3	mxnet	17
0.3	cntk	13
0.2	pytorch	1
0.1	deeplearning4j	14



4_ Start a project

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…	iOS/Android/ARM	★★★★☆	173s	
Caffe2	CNN/RNN/LSTM/etc	C++/Python/Matlab	iOS/Android/ARM	★★★☆☆	149s	
MXNet	CNN/RNN/LSTM/etc.	Python/C++/R/Java/Julia…	iOS/Android/ARM	★★★★☆	149s	
Pytorch	CNN/RNN/LSTM/etc.	Python/Lua	iOS/Android	★★★★★	168s	

4_ Start a project

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

4_ Start a project

Deep learning Toolkits

Distribution Learning

Deployment on Mobile

Issues when Training

4_ Start a project

Deep learning Toolkits

Distribution Learning

Deployment on Mobile

Issues when Training

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

Thanks