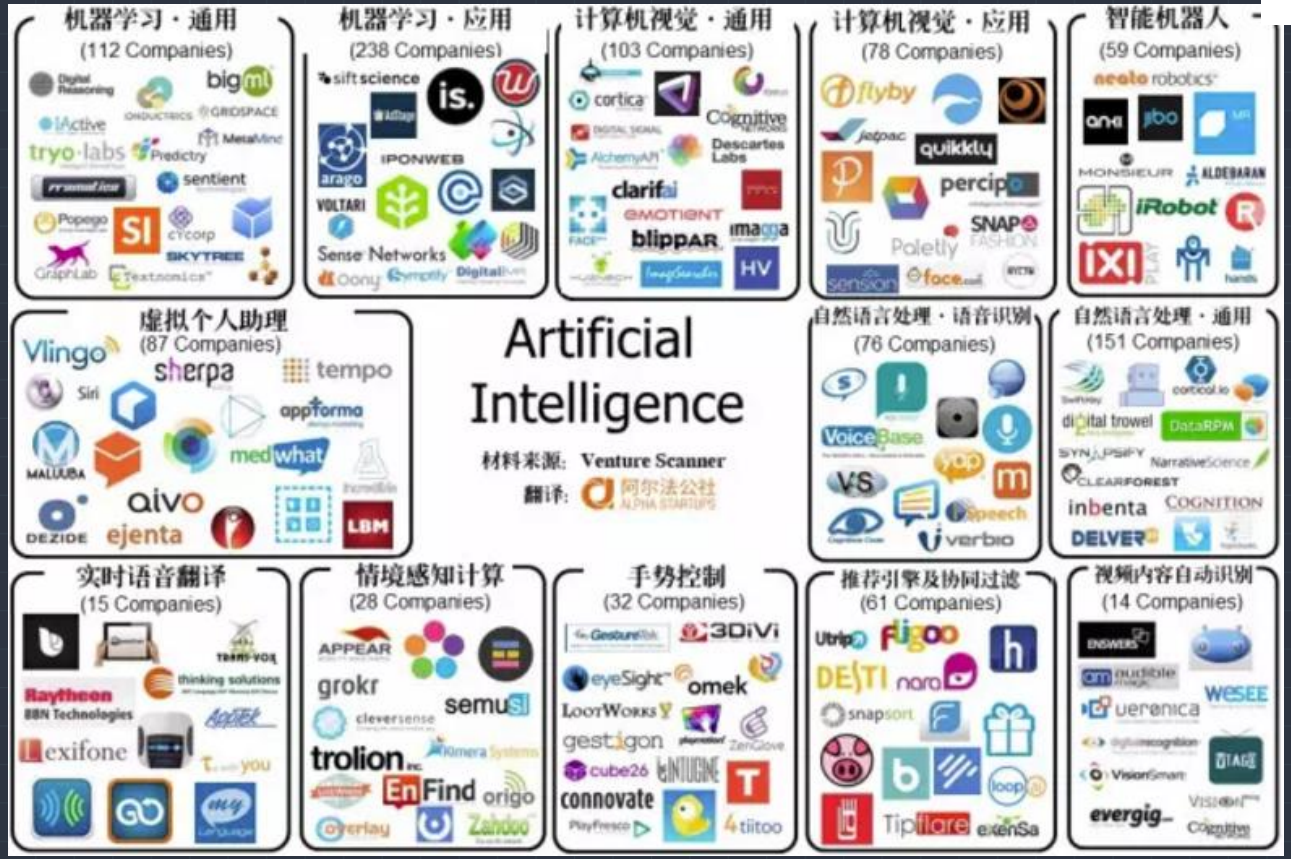
轻量级DNN网络在Android上的视觉应用

- 北京正安维视科技股份有限公司　张政



> 计算机视觉

face、person...

> 自然语言处理

翻译、搜索、Siri...

> 数据挖掘

消费习惯、知识库...

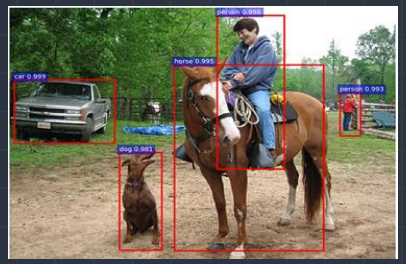
> 游戏

角色仿真、强化学习...

> 复合应用

无人驾驶、机器人...

植物识别



物体检测

瞳孔识别

语音识别

用户画像

自动驾驶

机器人

VR|AR


哥就不刮胡子？？


我是卸妆识别
还是化妆识别？

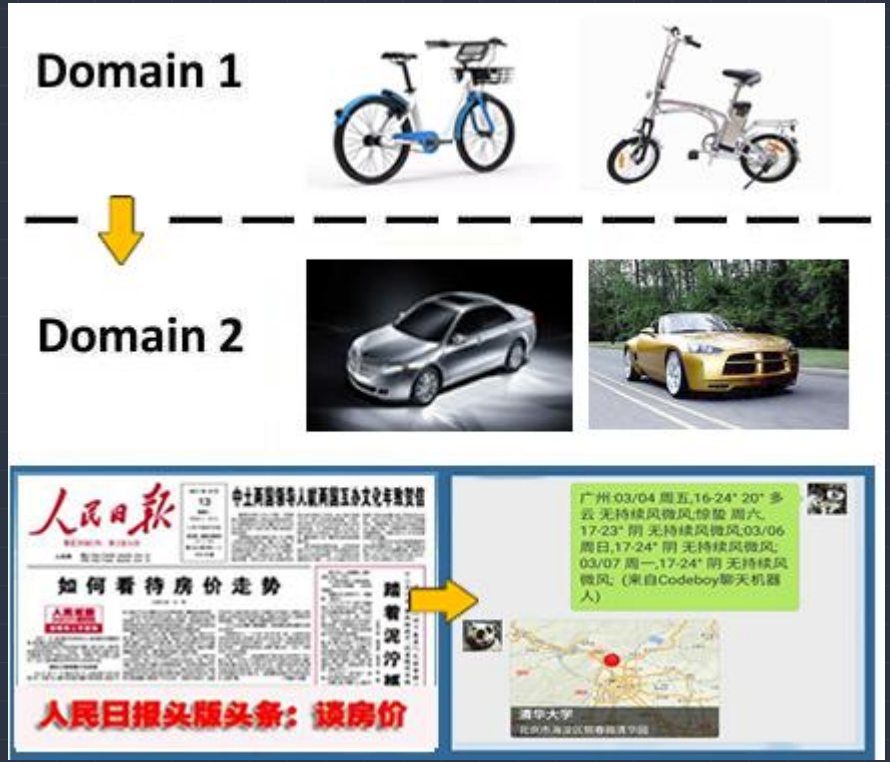

想刷脸，门都没有！！

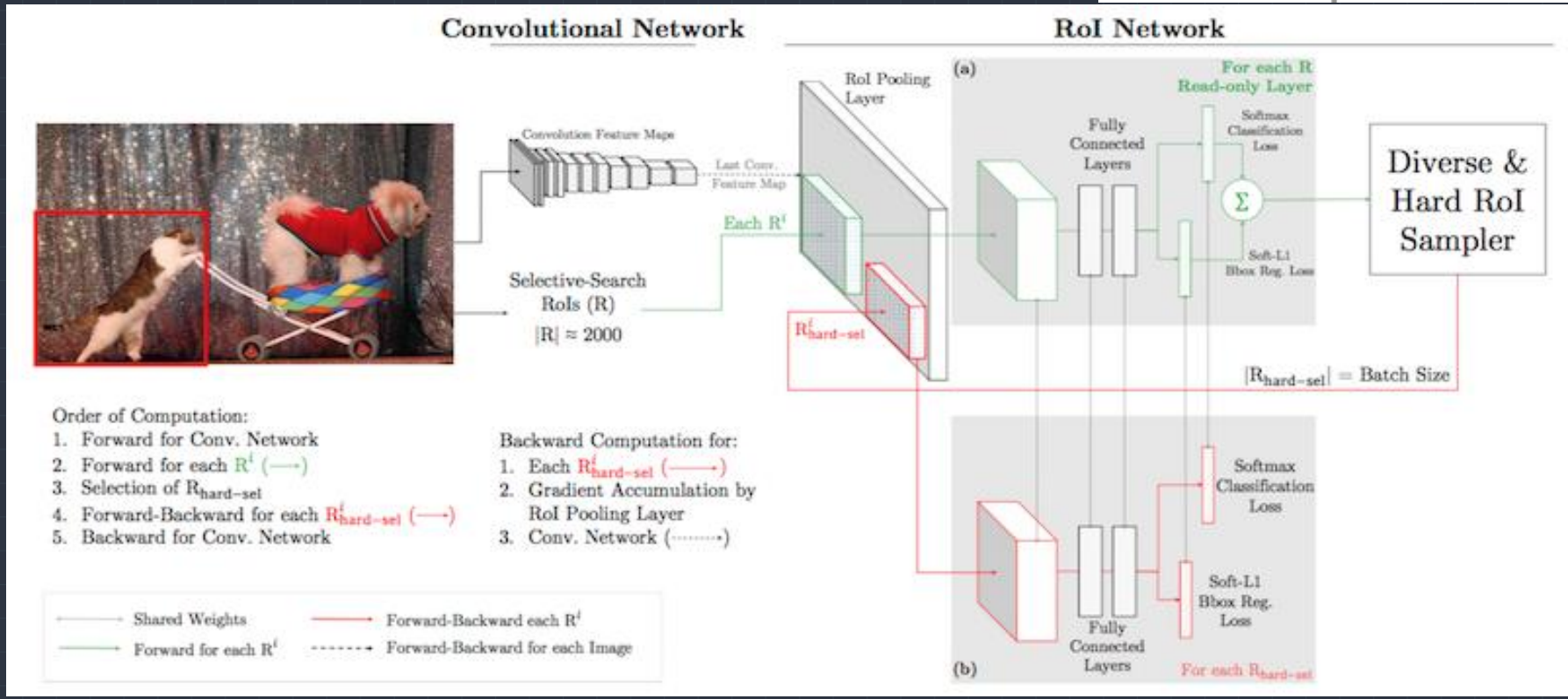
@ 常用CNN 框架

@ Base 模型的 accuracy 对比

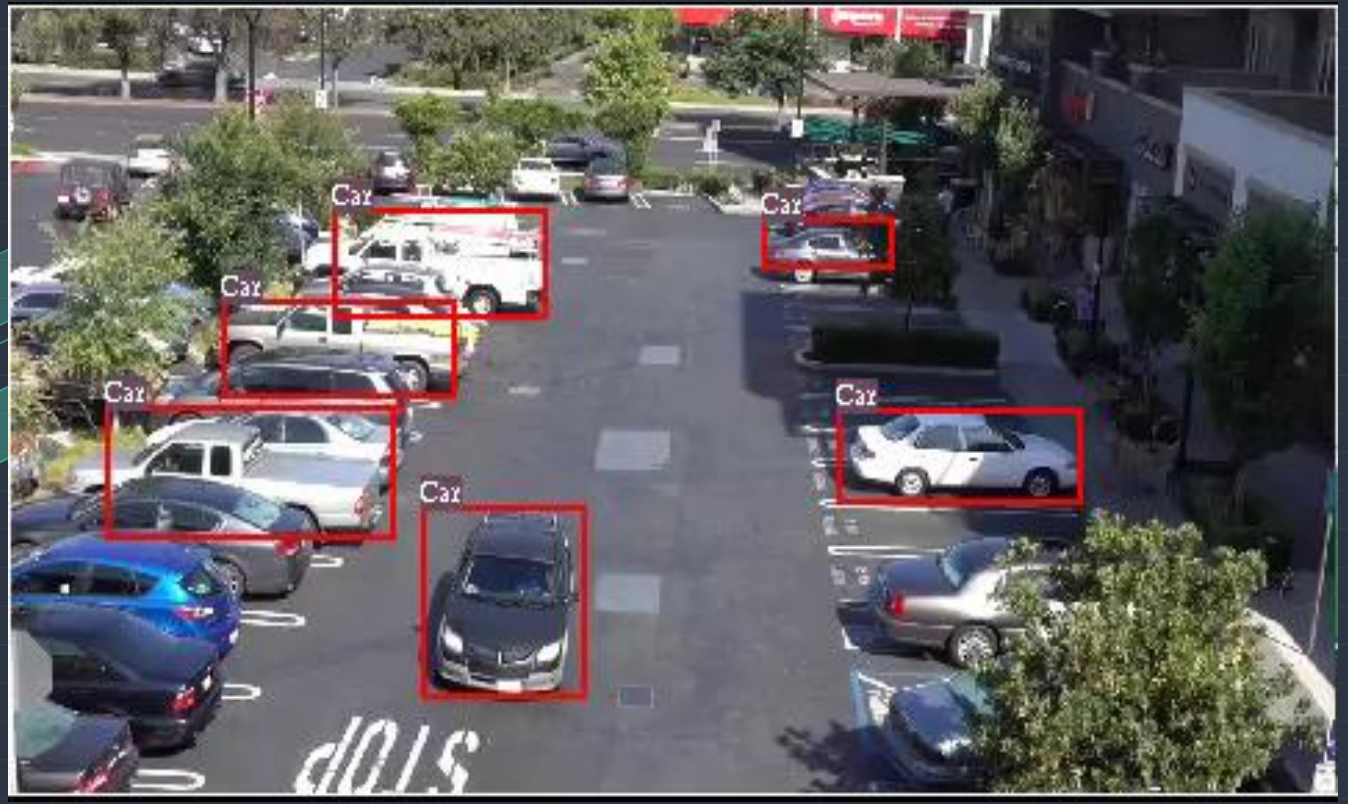
Model	Top-1 accuracy	Num. Params.
VGG-16	71.0	14,714,688
MobileNet	71.1	3,191,072
Inception V2	73.9	10,173,112
ResNet-101	76.4	42,605,504
Inception V3	78.0	21,802,784
Inception Resnet V2	80.4	54,336,736

- Learning学习 - learning to learn
- 终身学习 - life-long learning
- 知识转移 - knowledge transfer
- 归纳迁移 - inductive transfer
- 多任务学习 - multi-task learning
- 知识的巩固 - knowledge consolidation
- 上下文相关学习 - context sensitive learning
- 基于知识的归纳偏差 - knowledge-based inductive bias
- 元学习 - meta learning
- 增量学习 - incremental/cumulative learning

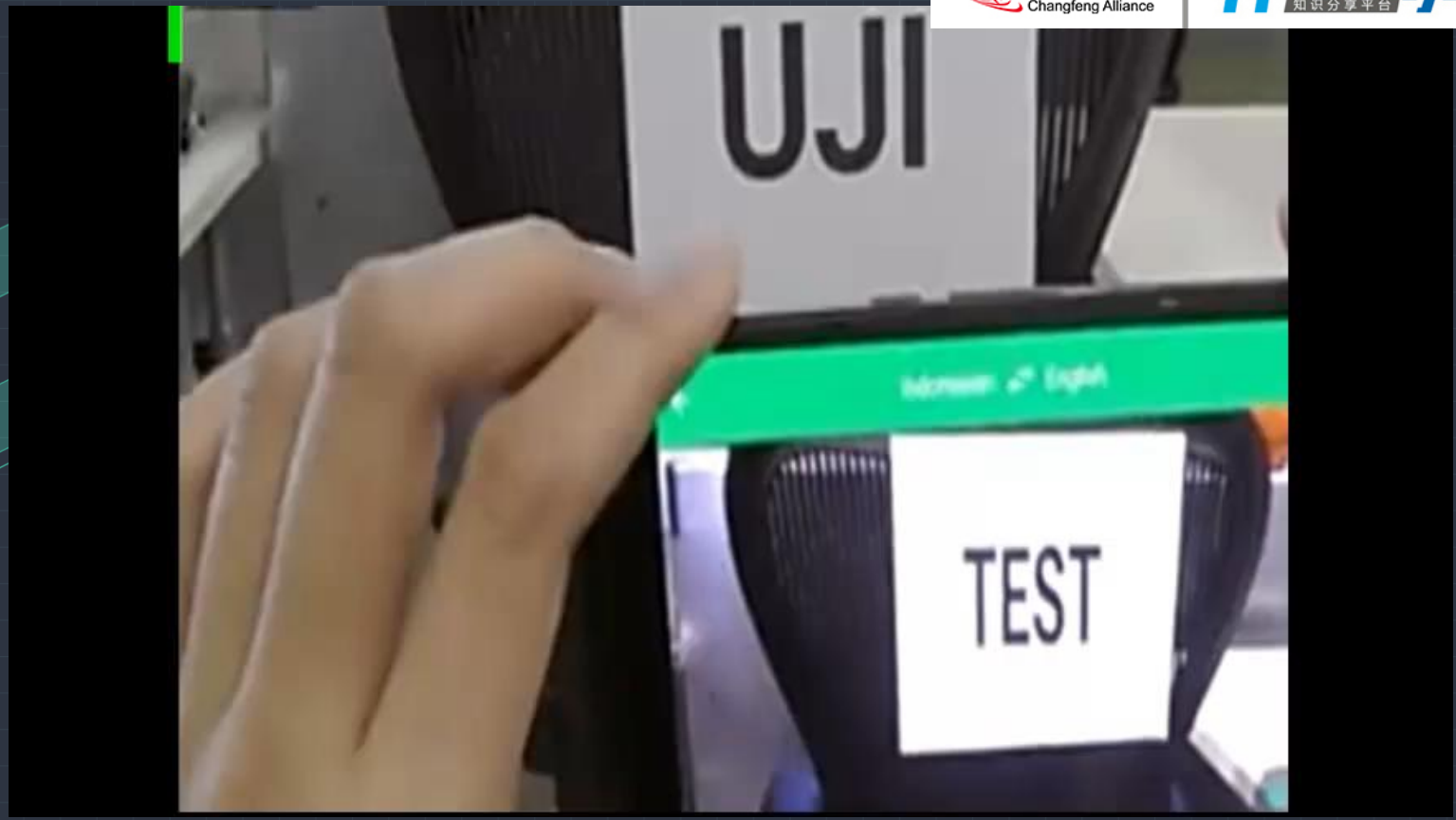



Convolutional Network
RoI Network
Convolution Feature Maps
Last Conv. Feature Map
Selective-Search RoIs (R)
|R| ≈ 2000
Each R^i
RoI Pooling Layer
(a)
For each R
Read-only Layer
Fully Connected Layers
Softmax Classification Loss
Soft-L1 Bbox Reg. Loss
Diverse & Hard RoI Sampler
R^i_hard-sel
|R_hard-sel| = Batch Size
Order of Computation:
1. Forward for Conv. Network
2. Forward for each R^i (——→)
3. Selection of R_hard-sel
4. Forward-Backward for each R^i_hard-sel (——→)
5. Backward for Conv. Network
Backward Computation for:
1. Each R^i_hard-sel (——→)
2. Gradient Accumulation by RoI Pooling Layer
3. Conv. Network (·····→)
Shared Weights
Forward for each R^i
Forward-Backward each R^i
Forward-Backward for each Image
(b)
Softmax Classification Loss
Soft-L1 Bbox Reg. Loss
Fully Connected Layers
For each R_hard-sel

> Deep Stream



> Translate

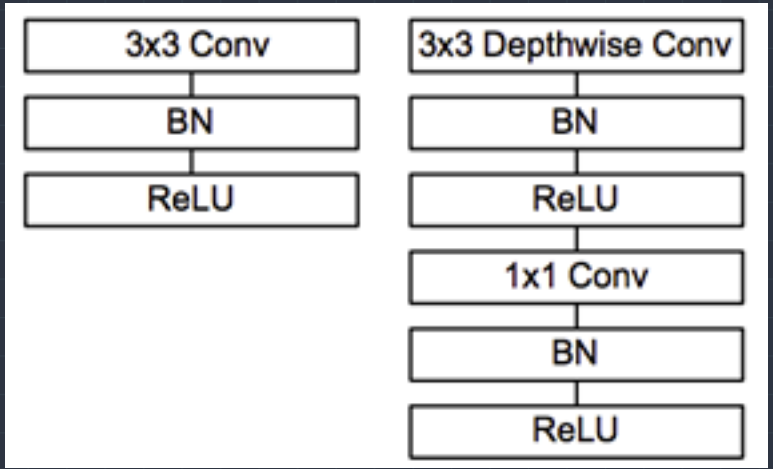


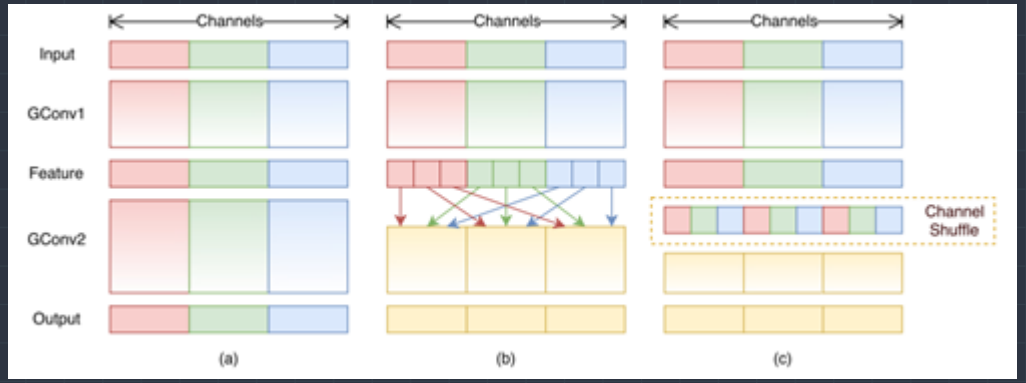
> ADAS

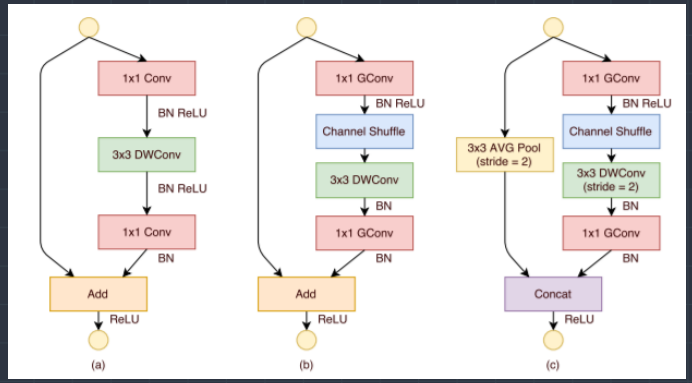
@ MobileNet

> Depthwise convolution

> Pointwise convolution





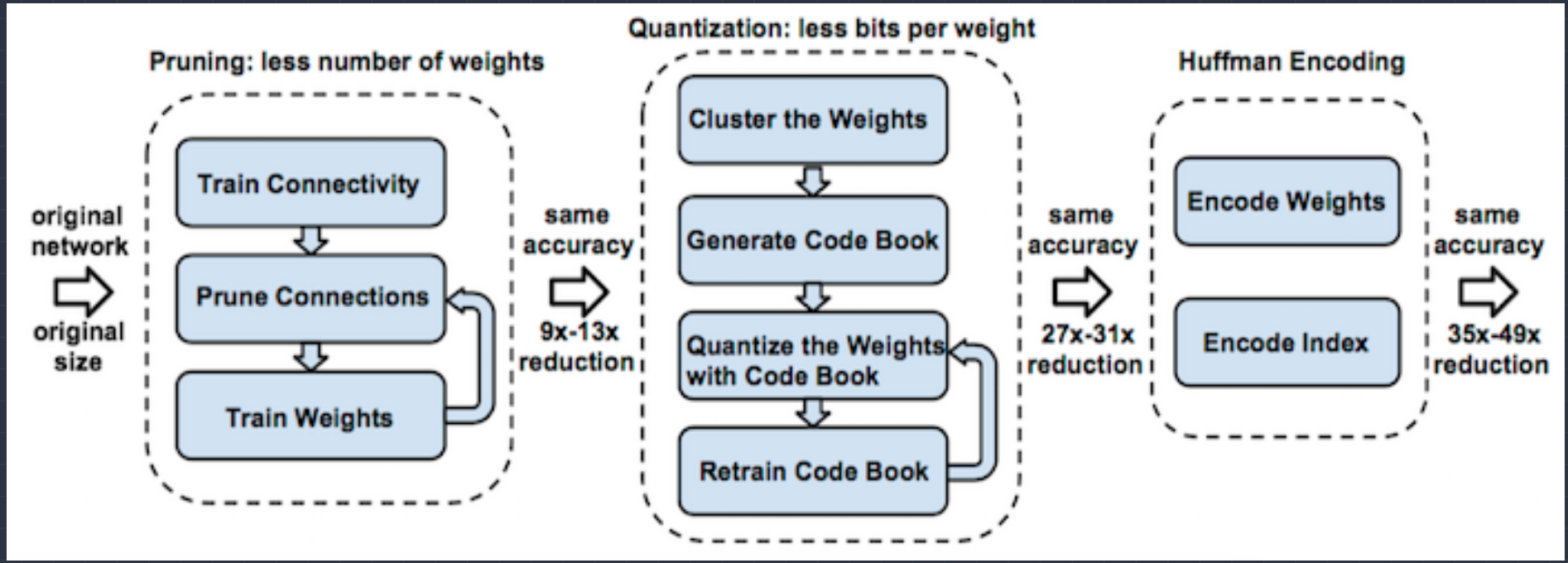


@ ShuffleNet

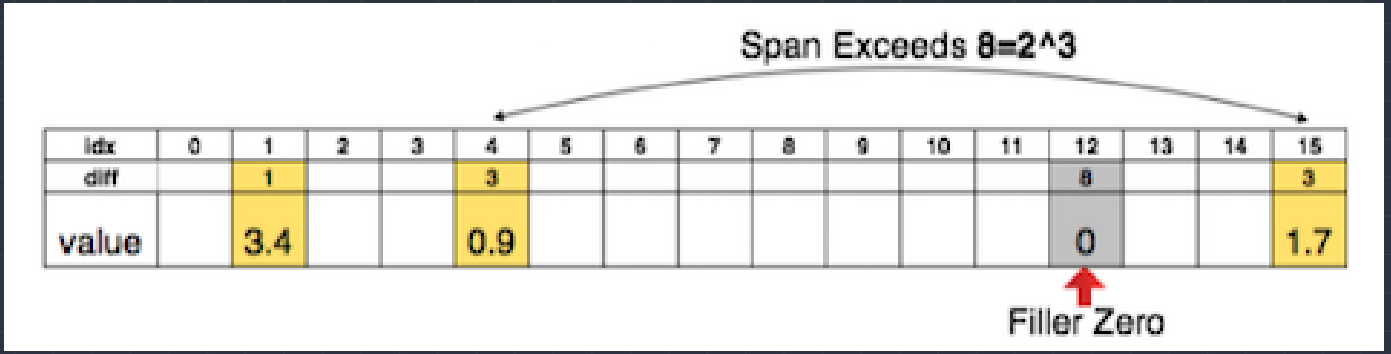
> Depthwise convolution

> Pointwise convolution

Model	Complexity (MFLOPs)	Cls err. (%)	Δ err. (%)
1.0 MobileNet-224	569	29.4	-
ShuffleNet 2× ($g = 3$)	524	**29.1**	0.3
0.75 MobileNet-224	325	31.6	-
ShuffleNet 1.5× ($g = 3$)	292	**31.0**	0.6
0.5 MobileNet-224	149	36.3	-
ShuffleNet 1× ($g = 3$)	140	**34.1**	2.2
0.25 MobileNet-224	41	49.4	-
ShuffleNet 0.5× (arch2, $g = 8$)	40	**42.7**	6.7
ShuffleNet 0.5× (shallow, $g = 3$)	40	45.2	4.2


Pruning: less number of weights
Quantization: less bits per weight
Huffman Encoding
original network
original size
Train Connectivity
Prune Connections
Train Weights
same accuracy
9x-13x reduction
Cluster the Weights
Generate Code Book
Quantize the Weights with Code Book
Retrain Code Book
same accuracy
27x-31x reduction
Encode Weights
Encode Index
same accuracy
35x-49x reduction

1. Pruning



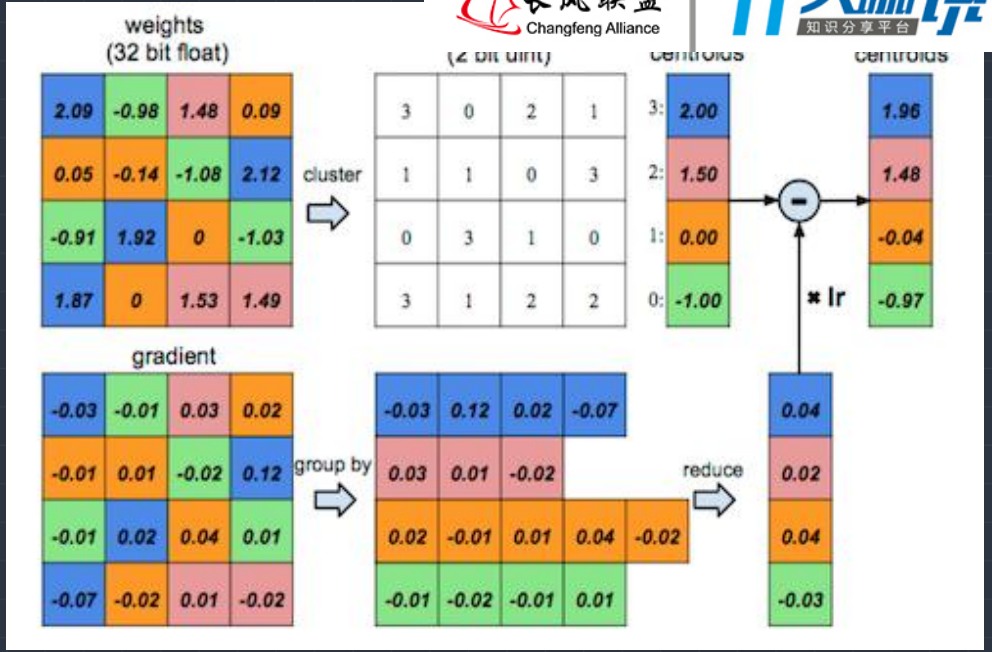
最常用的剪枝方法，即剔除对于网络贡献比较低的权值连接，这个比较好理解，通过Pruning使得网络变得稀疏，带来更少的计算量。

上图是基于稀疏矩阵的索引表示，用3bits 来表示索引的相对位置，其中黄色部分为有效权值区域，当相对位置的diff 超过8（3bits）的时候，中间插入了一个0权值来防止溢出。

事实上，**稀疏矩阵对于计算量来讲并没有太多的效率优化，远不如将整个的卷积和剔除来得更实在**，这也是 Pruning 接下来聚焦的方向。

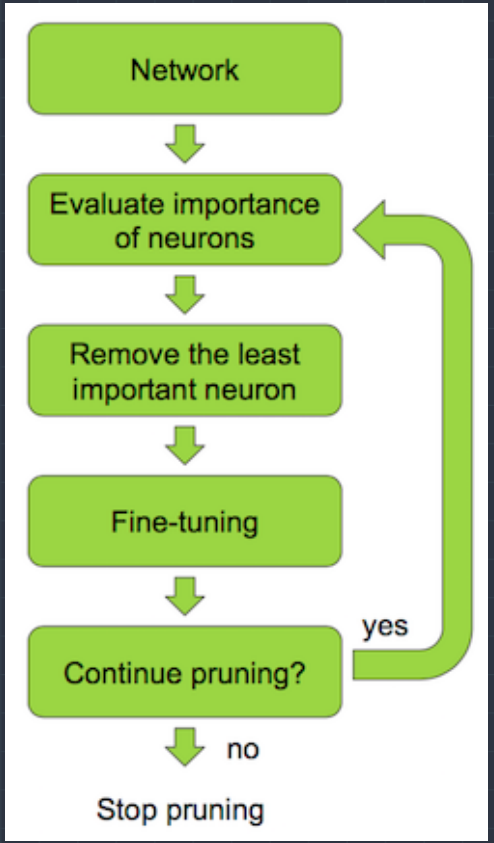
2. Quantization

量化，包括两个方向：
a）通过聚类的方式实现权值共享；
　这种方法误差很大，后续研究的也并不是很多知道下就好了。
b）采用更少的字节表示权值，比如 16bit、8bit；
　这里面还有 著名的 BinaryNet，XNOR-Net，不过对于效果影响比较大，通常是 Float16 or int8 。



3. Huffman 编码

通过对权值、索引进行编码，减少字节占用；
其中 (2)(3) 对系统的优化主要聚焦在节约内存占用，应用相对较少。



NVIDIA: Iteration Pruning

该方法的核心思路：

1）基于Kernel Filter进行 Pruning，不考虑更细的Weight 层面；

文中给出了原因，Weight 层面的 Pruning 能减少计算量，但稀疏的卷积核并不能带来效率的提升（缺少专用硬件）。

2）迭代删除 Least important Kernel，并进行 FineTuning；

3）提供了一个 Pruning 的准则；

判断依据：

1）Minimum weights

2）Activation

3）Mutual information

4）Taylor expansion

5）Relation To optimal Brain Damage

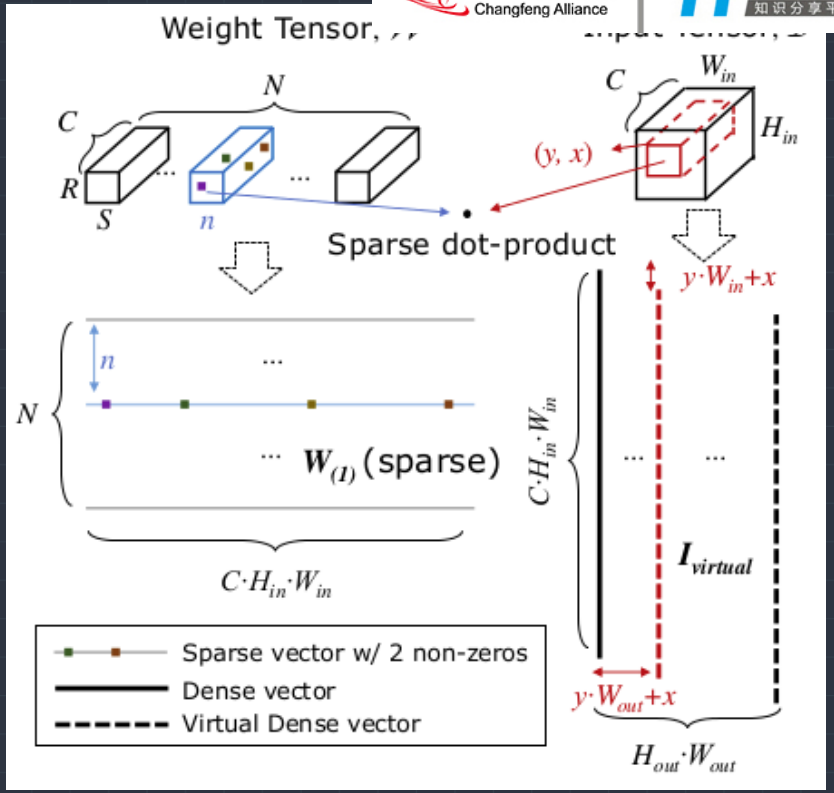
6）Average Percentage of Zeros（APoZ）

Intel: Direct Sparse Conv

Conv 层对于CNN网络的效率影响很大，与前面 Pruning 整个 Kernel 的方法不同，这篇文章主要是 Pruning 连接。

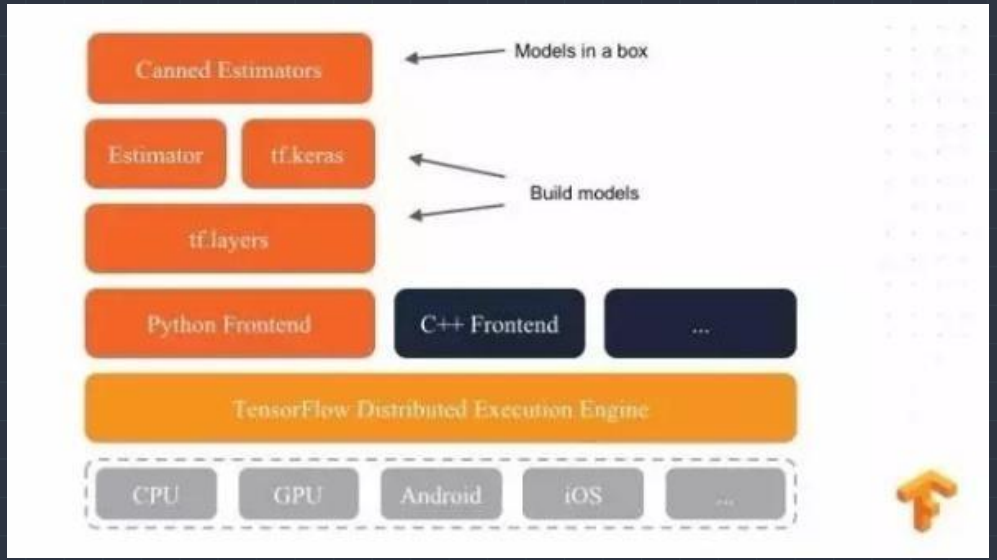
核心思想是 通过一种乘法（向量展开），实现 Dense Matrix（特征）和 Sparse Matrix（Kernel）之间的高效计算。同时提出了一种评价模型，预测不同网络下 稀疏程度的最佳值。

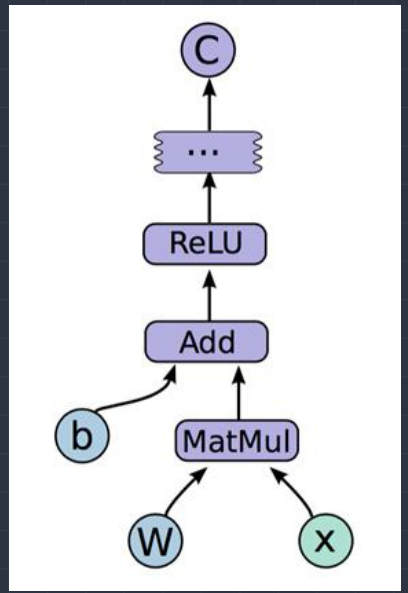
作者在 AlexNet 上进行 Conv 层压缩，获得了3.1~7.3倍的加速。

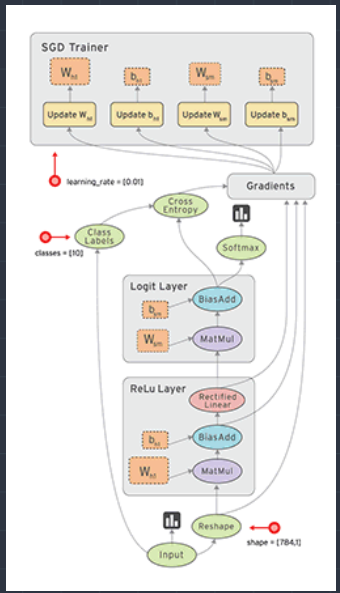


TensorFlow™

机器学习领域的下一个Android！







Thanks For Attention !