模型压缩文献调研总结

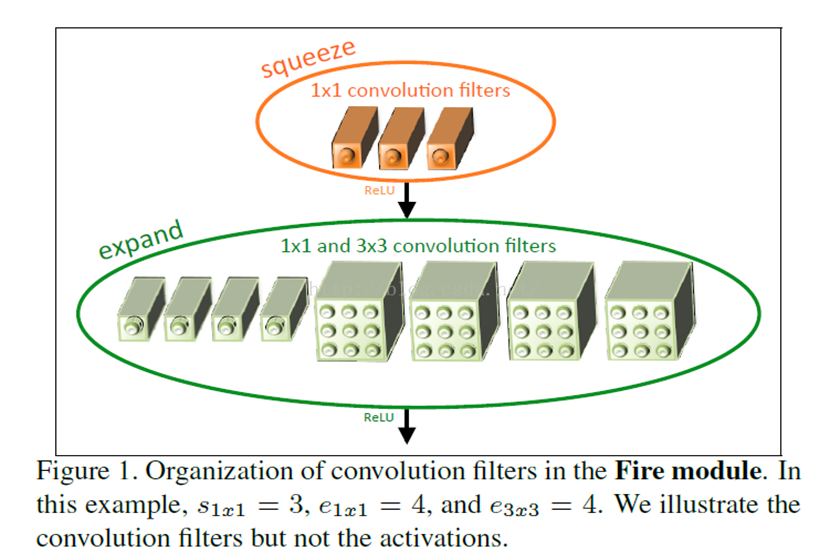
根据当前的调研情况，模型压缩相关工作可以分成模型结构优化设计、模型裁剪、模型量化、low-rank模型和knowledge distillation等五个类型。

1 模型结构优化设计

模型结构优化设计是通过设计compact模型的方式，在不损失性能的情况下减小模型大小，提升inference速度。本文主要介绍squeezeNet[1]、mobileNet[2]和shuffleNet[3]。

1) squeezeNet

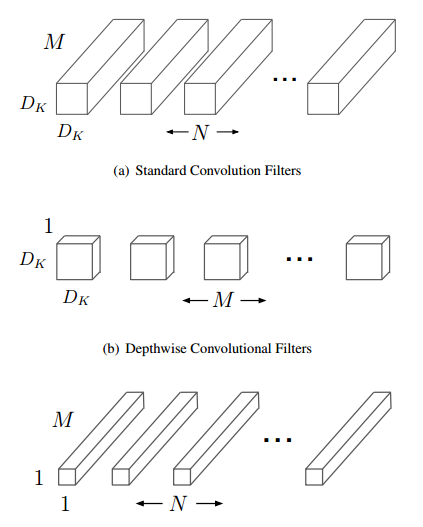
squeezeNet通过引入FireModule，实现模型参数压缩。如下图所示，FireModule包含squeeze（1\*1卷积）和expand（1\*1卷积和3\*3卷积的组合）两个部分。



squeezeNet可以在参数压缩50倍的情况下实现AlexNet的精度（从240M变为4.8M）。

2) mobileNet

mobileNet将原来的卷积分解为depthwise和1\*1 pointwise的方法，如下图所示。



计算量从

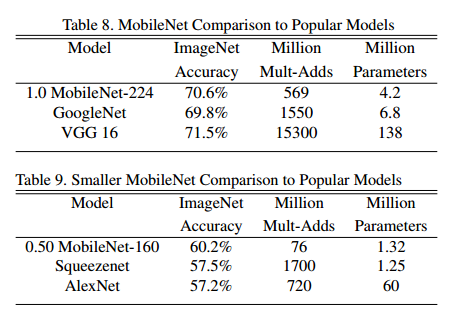


变为



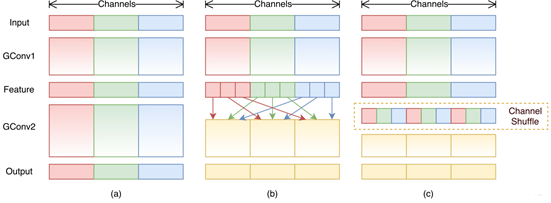
其中DF为feature map的size。参数量从DK∙DK∙M∙N，变为DK∙DK∙M+M∙N。

图片分类结果如下：（0.5 mobileNet-160表示，各层channel数在base模型基础上乘以0.5，输入图片大小为160\*160）

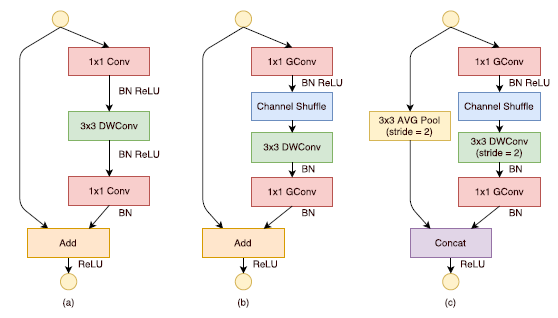


3）shuffleNet

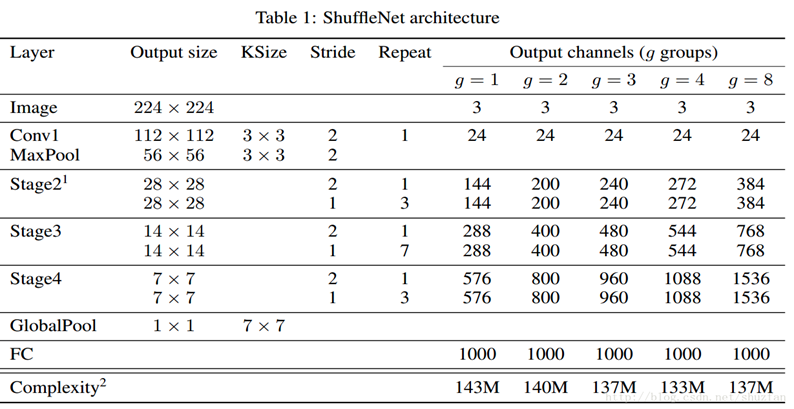
shuffleNet是在group卷积的基础上，通过shuffle使不同组之间信息可以流通，如下图所示。



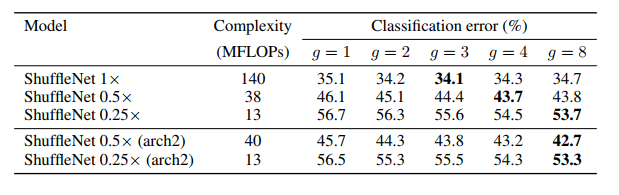
典型的ShuffleNet unit如下（在resNet单元基础上，(a)用通道卷积代替原始卷积，(b)加入channel shuffle，(c)为针对下采样的情况）：



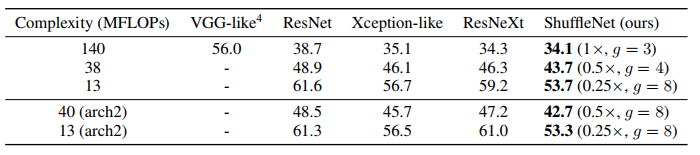
网络结构如下图所示（为了保证参数量基本一致，当使用不同group时调整了feature map的数量）。

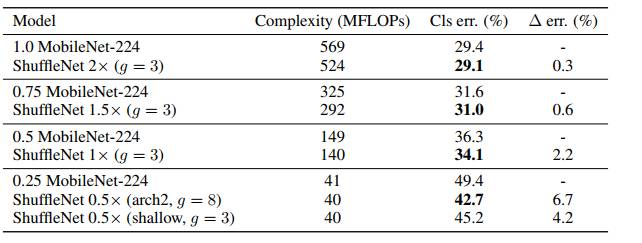


图片分类结果如下图所示。其中“s x”表示feature map的channels数量与原始结构的比例；arch2表示将stage3中的两个单元移除，同时增加整体的网络宽度（实现同样复杂度；对于小网络，feture map的宽度至关重要）。



与其他网络结构对比如下图所示。相比mobileNet，同样运算量可以得到更好的准确率，特别是对于小模型；Xception和ResNext是用group convolution（depthwise+pointwise）重新设计Inception和ResNet而得到的网络结构。





2 模型裁剪

模型裁剪可以分为Inference-time pruning和training-based的方法。前者是对已经训练好的模型进行prune；后者则在目标函数中加入相关正则项使得weight易于裁剪。

2.1 Inference-time pruning

本文主要介绍基于权重大小裁剪[4]、LASSO regression based channel selection+least square reconstruction[5]、基于APoZ[6]的方法。

2.1.1 基于权重大小进行裁剪

有很多基于权重大小进行裁剪的方法[4，7，8]，以文献[4]的方法为例：

a）一层层对weight进行裁剪。

b）按绝对值之和裁剪：裁剪掉m个最小的filters以及对应的输出；同时它又是下一层的输入，所以也得去掉下层中对应的输入filters，剩余的保留。

c）每裁剪一层均重新训练的效果优于裁剪所有层后一次性训练网络。

d) 对每一层单独剪枝，查看在validation set上准确度的变化，从而得到敏感度，根据敏感度确定不同层的剪枝比例。

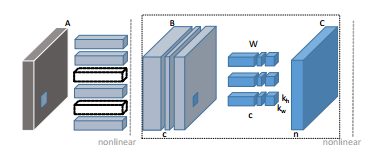
e)一般约deeper的层，对裁剪越敏感。

f) VGG-16能实现约34.2%的加速；ResNet-110能实现约38.6%的加速。

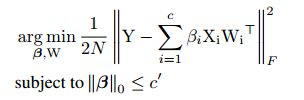
2.2.2 LASSO regression based channel selection+least square reconstruction

文章[5]采用data-driven的方式，选取channel并修改weight使得其输出的feature map能够重建。

针对单层的prune如下图所示，feature map B经过卷积W得到feature map C；prune的目标是去除B中的部分channel，并调整W使得到的C’能够与C尽量一致。去除B中的channel，会同时影响B前后的相关层。

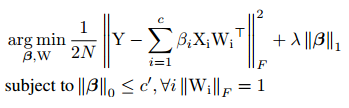


优化目标如下所示



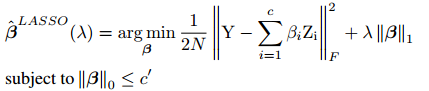
其中W为n\*c\*kh\*kw的卷积矩阵; X为从feature map B中采样得到的N\*c\*kh\*kw的input volume（N为样本数）;Y为W与X卷积得到的原始输出，是N\*n的矩阵;Wi为W的第i个channel且reshape为n\*khkw的矩阵；Xi为X中的第i个channel且reshape为N\*khkw的矩阵。

上述目标为NP难的，所以简化为优化下式：



优化时先固定W，得到β；再固定β，得到W。

1）固定W，得到β。优化目标如下图，可用LASSO回归求解。

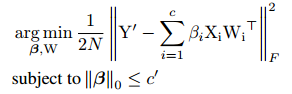


2）固定β，得到W。优化目标如下图，其中，可以用least square求解。

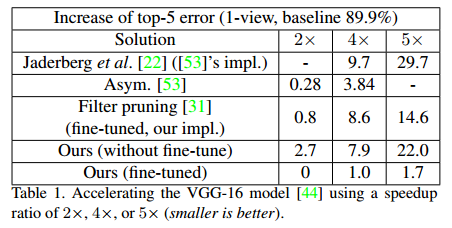


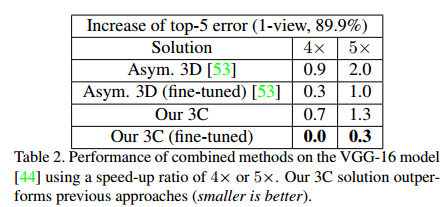
上述两步迭代进行即可得到结果。

当对整个模型进行prune时，采用的逐层prune的方式，但Y都是用原始feature map，以防止误差累积：

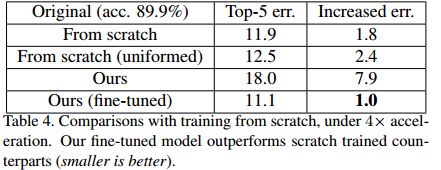


结果如下图所示

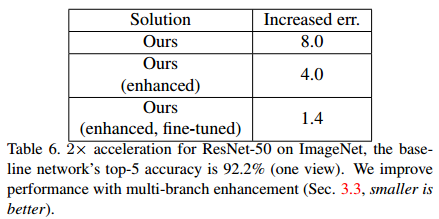


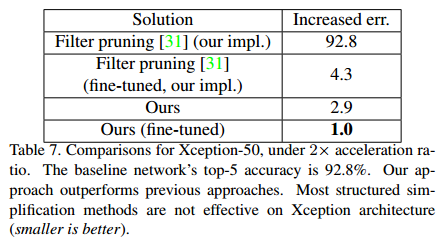


与Training from scratch对比结果如下图



ResNet和Xception结果如下图：





2.2.3 基于APoZ的裁剪

APoZ定义如下：

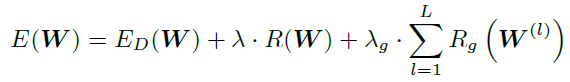


反映输出的feature map中某个channel中激活为0的神经元所占的比重。其中为第i层第c个feature map，N为样本数，M为输出 feature map的维度。

对VGG，在无损精度下可实现2倍压缩。

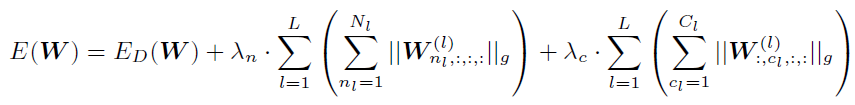
2.2 training-based approach

文献[9]通过在目标函数中加入group lasso的正则项，可以实现filter级、channel级以及shape级稀疏化：

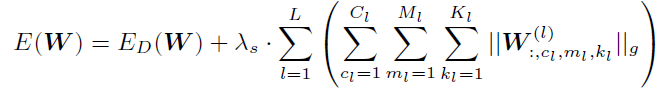


第三项为group lasso的正则项，可以为：

1）filter-channel wise：（每个filter/channel是一组）



2）shape wise：（每个channel+位置是一组）



训练完后通过剔除全为0（接近0）的值来提升效率。

结果：对于AlexNet，若不剔除参数，精度可提升约1%；在不改变精度的情况下，AlexNet上提速30%~40%。

3模型量化

主要包括low-precison model和vector quantization两种类别。前者是指用更少的位数来编码weight和/或中间结果（如activation、梯度等）；后者是利用k-means、product quantization等方法对weight进行聚类，通过存取codebook（聚类中心）和index的方式实现对模型的压缩以实现压缩[7, 10]。

3.1 low-precison model

3.1.1 BinaryConnect [14]

BinaryConnect的几个特点：

1）只量化weight：weight被量化为+1 和-1

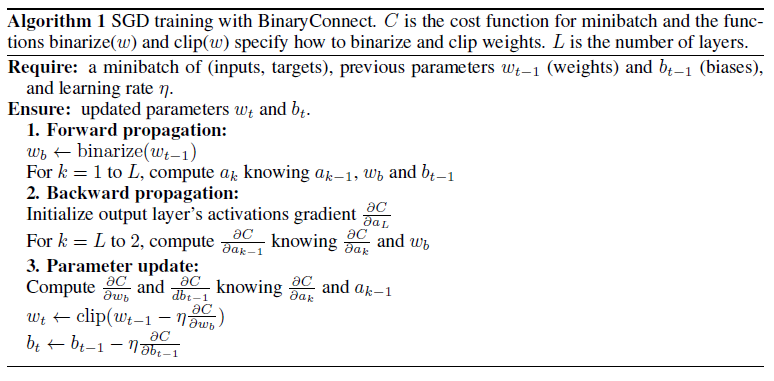
2）两种量化方法：确定性量化方法和随机量化方法





其中为hard sigmoid函数

3）在前向和后向过程中使用量化的wb，但会维持一个高精度的wt，更新都是作用于wt；且会将wt clip为[-1，1]范围内。



4）随机量化方法可视为一种正则化方法（类似dropout）。

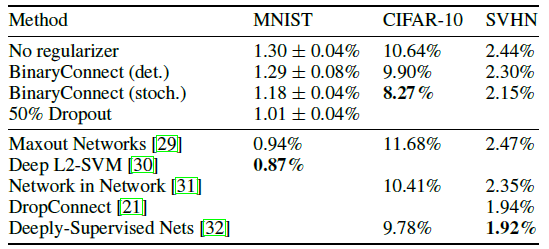
5）inference时3种方法：

a) 使用量化的wb（确定性量化时）

b) 使用wt（随机量化是为一种正则化手段）

c) 随机采样得到一组量化的wb，然后ensemble（实际中未采用）

6) 结果



3.1.2 BNN[15]

BNN的几个特点：

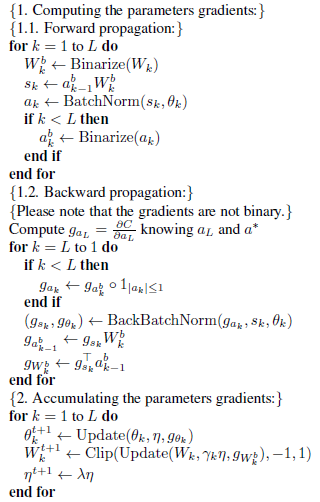
1）同时量化weight和activation；与3.1.1相同，更新过程会维持一个高精度的weight，更新都是作用于高精度weight；高精度weight在更新过程中会clip。

2）涉及到符号函数的求导：

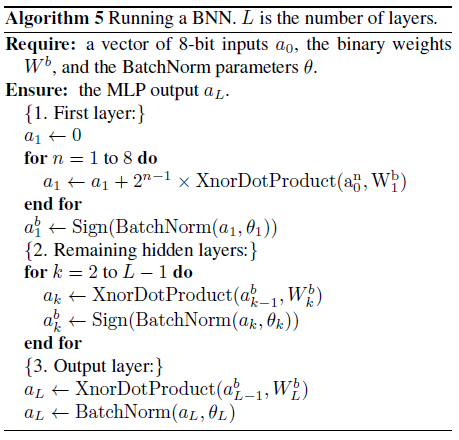




3）训练过程如下图所示：

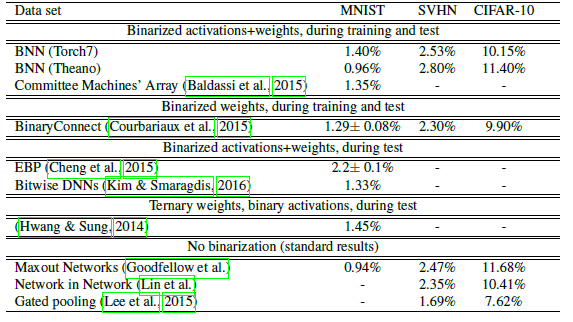


4）乘加操作操作可以变为XNOR-count。



5）结果

两个版本BNN在训练时的量化方式（确定性VS随机）等上不同。



3.1.3 BWN和Xnor-net[16]

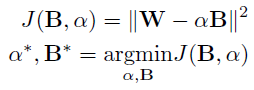
BWN的特点：

1）只量化weight，但与BinaryConnect不同的是会有scaling factor：

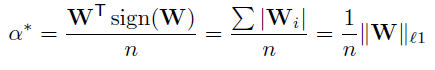


，，

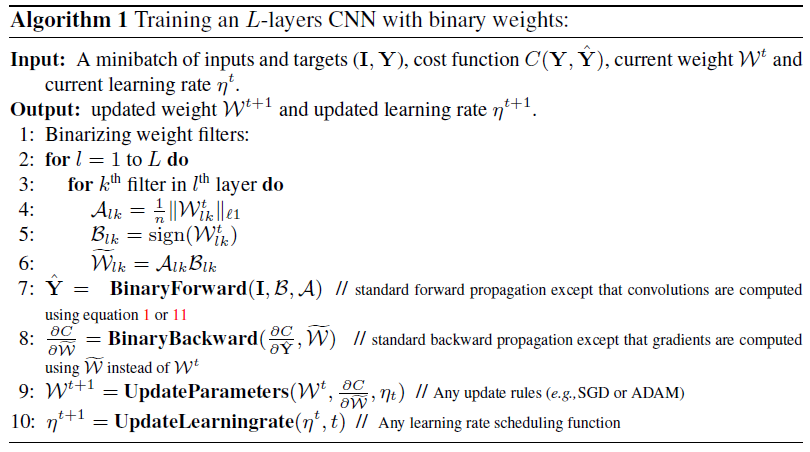
2）通过优化下面的目标



可以得到B中元素为W中对应元素的符号，



3）训练流程



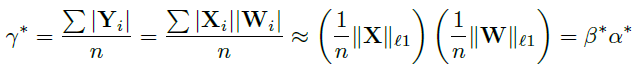
Xnor-net的特点：

1）量化weight和activation，会引入scaling factor：



其中代表点积，X是input的volume，与W具有同样的维度。





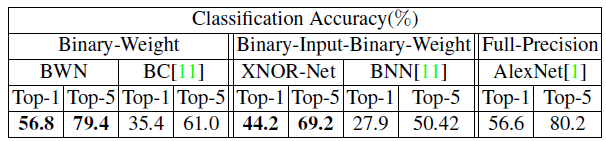
计算时首先计算出K使得Kij为对应位置处的β。

2）典型的XNOR-Net的block



3）训练过程与BWN相同，第1层和最后1层不量化（与BNN相似）

BWN和Xnor-net的识别结果（ImageNet）：





3.1.4 其他

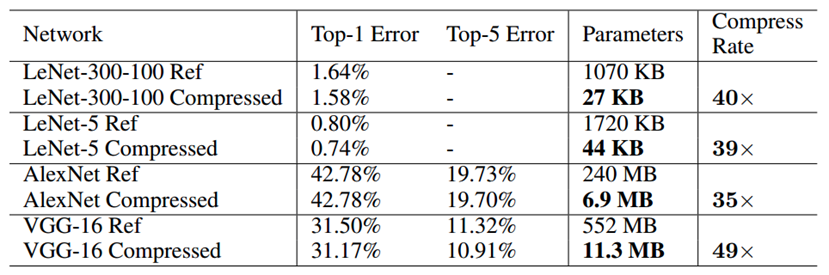
Ternay Weight networks[17]则用{-1，0，1}+scaling factor对weight进行量化。

DoReFaNet [18] 则用1bit量化weight，2bit量化activations，6bit量化梯度。QNN[19]也进行了类似的尝试。

INQ[20]则通过参数分组、分组量化、retrain的渐进式方法对网络weight进行量化，且均量化为0或二次幂。

3.2 vector quantization

以文献[7]为例，其综合利用prune、聚类和huffman编码等实现压缩。 prune是裁剪掉权重小于一定阈值的连接；聚类利用kmeans。结果如下:（该方法主要用于压缩，但在没有特殊库或硬件的支持下不利于inference速度）。



4 low-rank model

low-rank model 是通过将网络参数投影到低维空间，以实现模型压缩和计算加速。本文主要介绍文献[11]和[12]。

4.1 Speeding up Convolutional Neural Networks with Low Rank Expansions

weight矩阵存在冗余，可以投影到低维空间。常用方法包括：

1）filter维度：通过以N个filters为基，将原始的M组filters表示为基的线性组合，这样便可以用N次滤波计算和一定数量的线性求和来替代M次滤波计算。

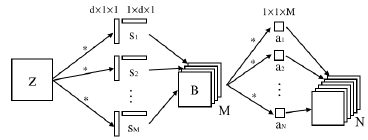
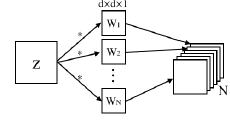
2）spatial维度：利用separable filter，将d×d的filter分解为d×1的filter和1×d的filter。

文章提出两种方法：

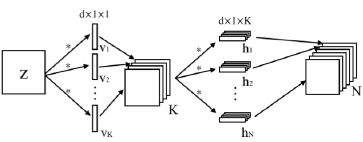
1) 2D filter维度。设为某卷积层中第n个3D filter，为中第c个channel对应的2D filter。设，即可近似为M个filter的线性组合。



同时将分解为seprable filters（d\*1 filter+1\*d filter），在M<d min{d,N}时可提速，如下图所示（channel数为1，上半部分图为原始，下半部分图为压缩后）：



2）3D fitlter维度。filter拆分成3D vertical+3D horizontal filter。即先用K个c\*d\*1的vertical filter，然后再用N个k×1×d的horizontal filter。在K、N、C为same order的情况下，提速约d倍。

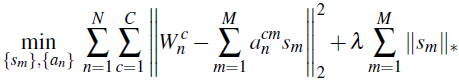


优化方法：

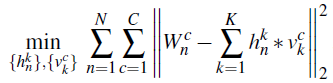
1）filter reconstruction optimization

通过优化使得filter尽量一致。

2D filter维度：

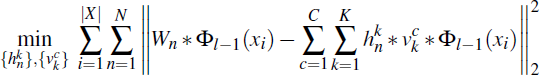


3D fitler维度：



2）data reconstruction optimization

通过优化使得卷积后得数据尽量一致。以3D filter维度为例：



结果：在ICDAR 2003 character 数据集上；网络为自己设计的four layer CNN。在无精度损失的情况下可以实现约2.5倍加速；在1%精度损失的情况下可以实现约4.5倍加速。

4.2 Accelerating very deep convolutional networks for classification and detection

4.2.1 单层的压缩

文献[12]认为在一个位置处的滤波结果是处在low-rank subspace的。记



其中x为k\*k\*c+1维，是layer input的一个volume（+1是为处理bias）；W为维，d是filter的数目；y是d维的。因为假设滤波结果y是在low-rank subspace上，则



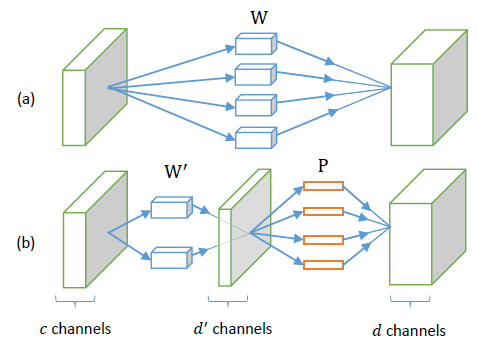
其中M为d×d的矩阵，rank为d'<d，从而可以分解为

M=PQT

P和Q均为d\*d'维，设W'=QTW，则有

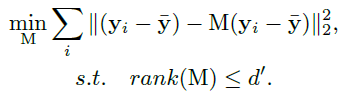


W'实际为一组（d'个）新filters，P实际为1\*1\*d'的fiter，如下图（（a）为原始卷积，(b)为压缩后）。



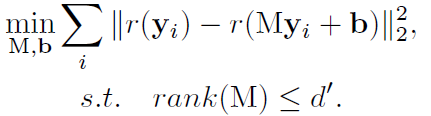
剩下的问题是要得到M，考虑两种情况：

1）线性情况

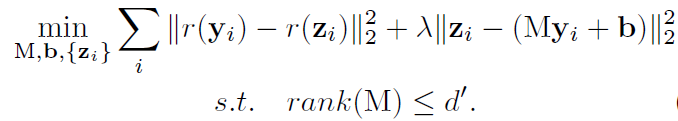


可以用SVD（实际为PCA方法）求解，即对YYT进行特征值分解。

2）非线性情况

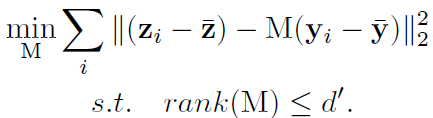


可以近似为



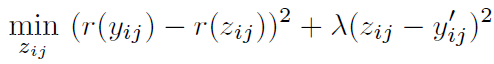
从而可以分成两步：

1）固定{zi}，优化M和b。



可以用GSVD进行求解。

2）固定M和b，优化{zi}。



为一维优化问题。

上述两步迭代一定次数后得到M，用SVD分解便可以得到P和Q。

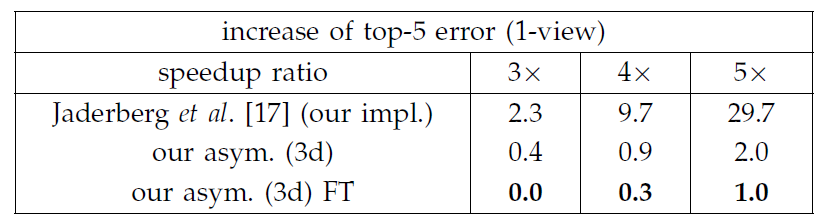
4.2.2 整个模型的压缩

通过Layer by layer的方式对整个模型进行压缩；和prune那篇文章一样，都是用原始输出进行优化，防止误差累积。

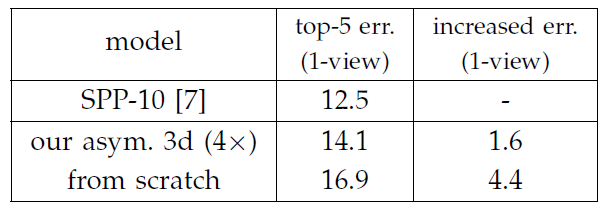
此外，可以利用4.1中的方法进一步将k×k的filter拆分成k×1和1×k。

4.2.3结果

1）VGG16的结果如下图：



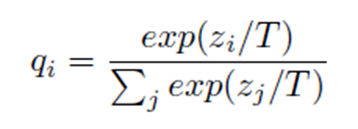
2）通过对比发现压缩后finetune的方法比直接对压缩模型从头开始训练好。



5 knowledge distillation

knowledge distillation是指利用预先训练好的复杂模型（teacher model）的输出作为监督信号去训练另外一个简单的网络（student model）[13]。

文献[13]在训练小模型时，用到的监督信号为soft target：



在一般模型的训练过程中，T取1；但在soft target中，T往往取较大的值，从而使得qi的大小在不同类中更加均匀，即更加soft。

训练小模型时，目标函数是hard target和soft target的加权平均。hard target为我们平时训练过程中所使用的。以MNIST数据集为例，对于用来表示数字3的图片，它的label就是[0,0,1,0,0...0] ，从概率的角度上即“图片上的数字是3的概率是100%，是其他数字的概率是0%”，这就是hard target。

在MNIST上的结果如下图:

(1) 复杂模型会有67个错误；

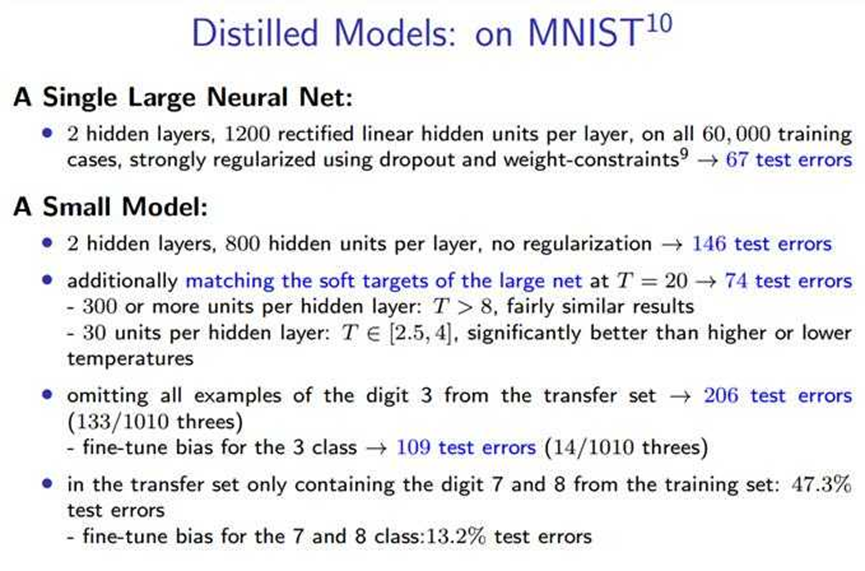
(2) 简单模型只用hard target会有146个错误；

(3) 简单模型用hard target+soft target会有74个错误；

(4) 模型简化后可能会对T敏感；

(5) 训练简单模型的数据中没有3，通过soft target训练出的模型仍可识别出3；

(6) 训练简单模型的数据中只有7和8，通过soft target训练出的模型仍可识别出其他数字。



参考文献：

[1] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, SQUEEZENET : ALEXNET-LEVEL ACCURACY WITH 50X FEWER PARAMETERS AND <0.5MB MODEL SIZE.

[2] Howard, Andrew G., et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications.

[3] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, Jian Sun. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices

[4] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, Hans Peter Graf. Pruning Filters for Efficient ConvNets.

[5] Yihui He, Xiangyu Zhang, Jian Sun. Channel Pruning for Accelerating Very Deep Neural Networks.

[6] H. Hu, R. Peng, Y.-W. Tai, and C.-K. Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures.

[7] Song Han, Huizi Mao, William J. Dally. DEEP COMPRESSION : COMPRESSING DEEP NEURAL NETWORKS WITH PRUNING , TRAINED QUANTIZATION AND HUFFMAN CODING.

[8] Yiwen Guo, Anbang Yao, Yurong Chen. Dynamic Network Surgery for Efficient DNNs.

[9] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, Hai Li. Learning Structured Sparsity in Deep Neural Networks.

[10] Yunchao Gong, Liu Liu, Ming Yang, Lubomir Bourdev. COMPRESSING DEEP CONVOLUTIONAL NETWORKS USING VECTOR QUANTIZATION.

[11] Max Jaderberg, Andrea Vedaldi, Andrew Zisserman. Speeding up Convolutional Neural Networks with Low Rank Expansions.

[12] X Zhang, J Zou, K He, J Sun. Accelerating Very Deep Convolutional Networks for Classification and Detection.

[13] Geoffrey Hinton, Oriol Vinyals, Jeff Dean. Distilling the Knowledge in a Neural Network.

[14] Matthieu Courbariaux, Bengio Yoshua, and David Jean-Pierre. Binaryconnect: Training deep neural networks with binary weights during propagations.

[15] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1.

[16] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks.

[17] Fengfu Li and Bin Liu. Ternary weight networks.

[18] Shuchang Zhou, Zekun Ni, Xinyu Zhou, He Wen, Yuxin Wu, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients.

[19] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations.

[20] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, Yurong Chen. Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights.