CNN训练Cifar-10技巧

关于数据集

Cifar-10是由Hinton的两个大弟子Alex Krizhevsky、Ilya Sutskever收集的一个用于普适物体识别的数据集。Cifar是加拿大政府牵头投资的一个先进科学项目研究所。

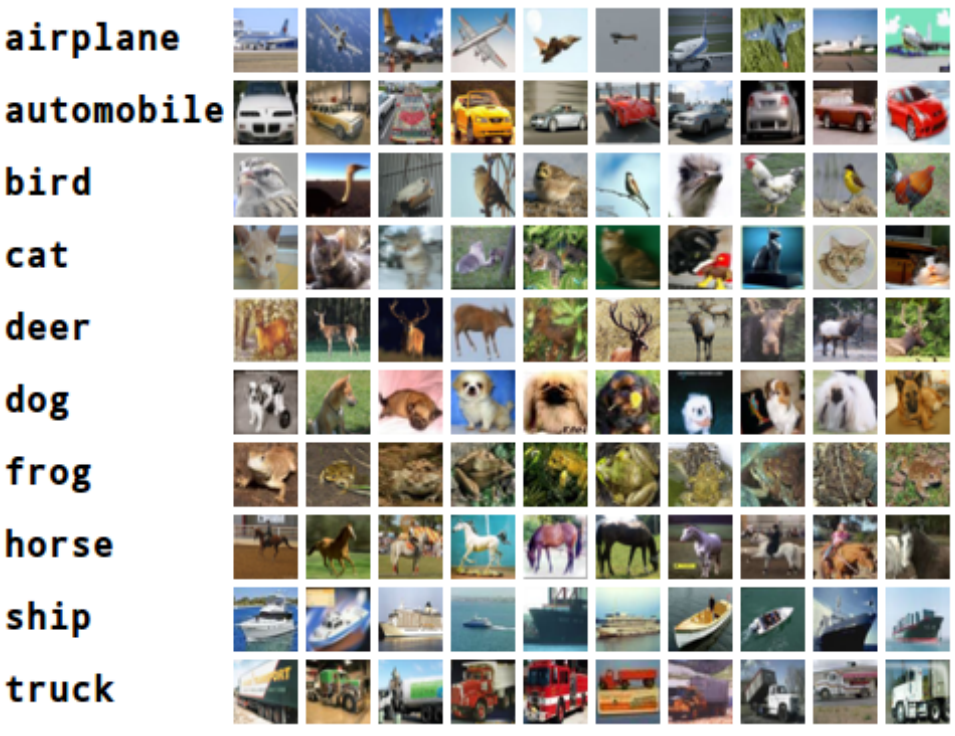
说白了，就是看你穷的没钱搞研究，就施舍给你。Hinton、Bengio和他的学生在2004年拿到了Cifar投资的少量资金，建立了神经计算和自适应感知项目。

这个项目结集了不少计算机科学家、生物学家、电气工程师、神经科学家、物理学家、心理学家，加速推动了DL的进程。从这个阵容来看，DL已经和ML系的数据挖掘分的很远了。

DL强调的是自适应感知和人工智能，是计算机与神经科学交叉。DM强调的是高速、大数据、统计数学分析，是计算机和数学的交叉。

Cifar-10由60000张32*32的RGB彩色图片构成，共10个分类。50000张训练，10000张测试（交叉验证）。这个数据集最大的特点在于将识别迁移到了普适物体，而且应用于

多分类（姊妹数据集Cifar-100达到100类，ILSVRC比赛则是1000类）。



可以看到，同已经成熟的人脸识别相比，普适物体识别挑战巨大，数据中含有大量特征、噪声，识别物体比例不一。而且分类庞大（SVM直接跪哭）。

因而，Cifar-10相对于传统图像识别数据集，是相当有挑战的。

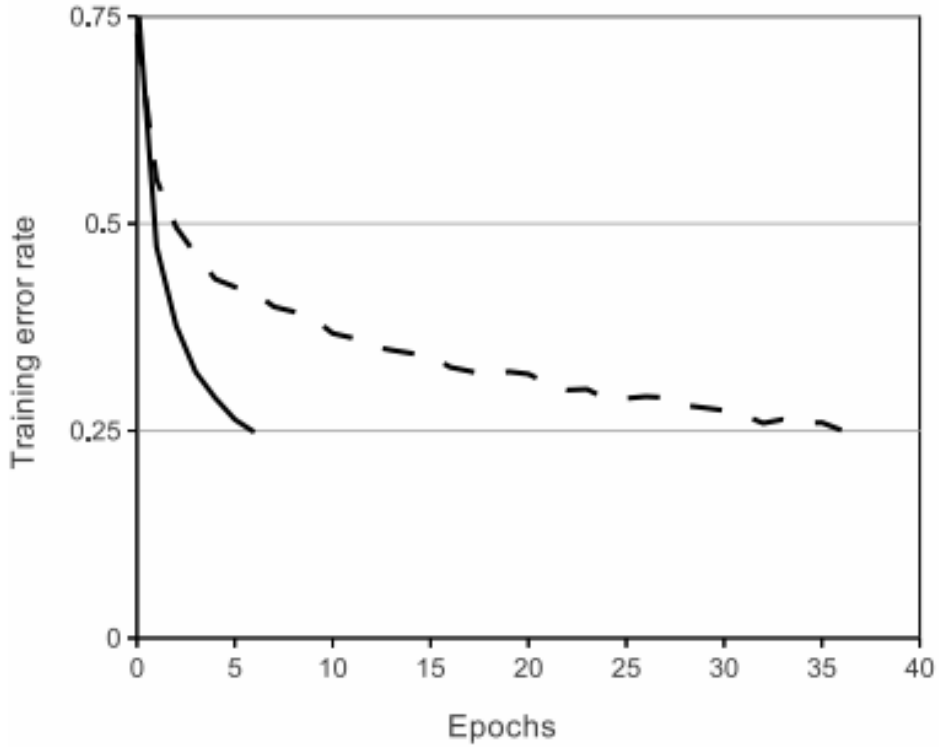
Alex Krizhevsky在2012年的论文ImageNet Classification with Deep Convolutional Neural Networks 使用了一些改良CNN方法去

解决普适物体识别难题，效果惊人。尤其是开创性地使用了CUDA来加速神经网络训练，并且开放了Cuda-Convnet和绝秘CNN结构，群众反响热烈。

于是就有了今年GTC 2015老黄手捧着TitanX大喊：NVIDIA显卡伴你更好深度学习！在通用计算上,CUDA确实多方面压制OpenCL。

Part I 总览

1.1 Cifar-10的训练走向



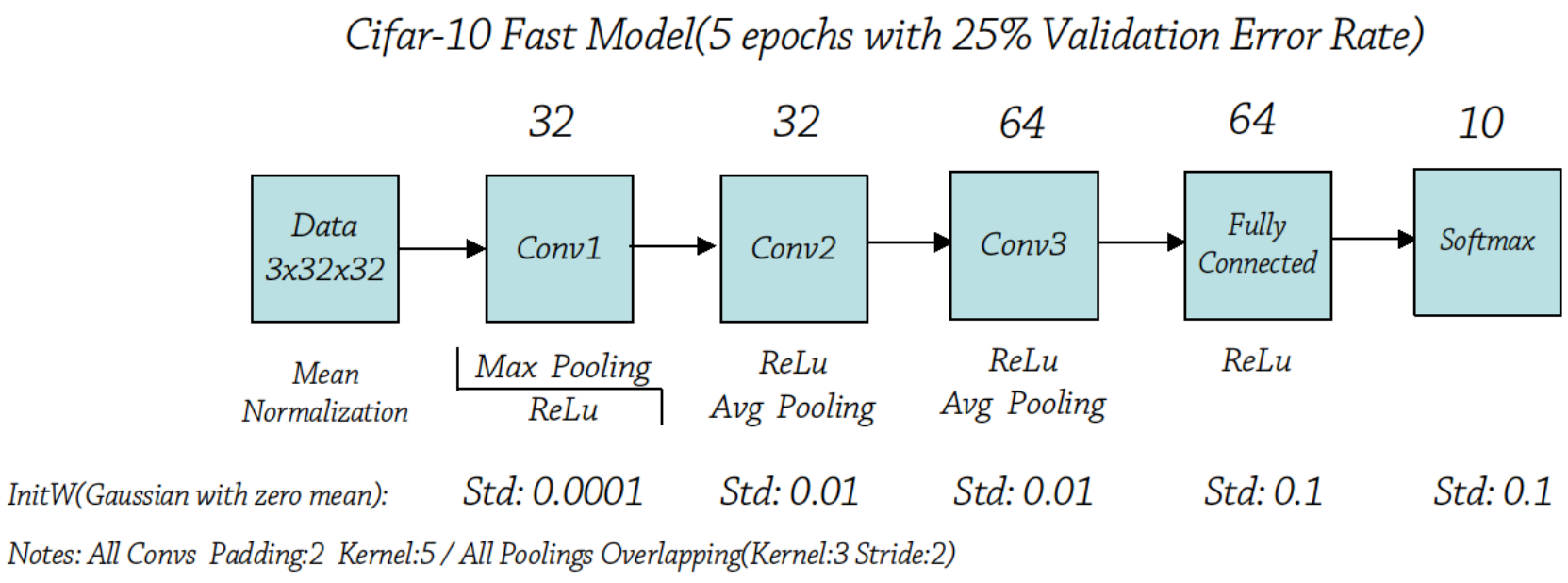
Alex的论文里可能困惑最大的就是这张奇葩的图。使用了ReLu的CNN在batchsize为100的训练中，epoch 5（2500次迭代，显卡好点只要80s）就把验证集错误率降到了25%。

而Tanh则要花上35个epoch。困惑点有两个:①居然下降那么快②我照着传统CNN模型改改，为什么只能降到30%，而且还在50+ epochs。

然后Alex又来了一句，我们的改良CNN已经把Cifar-10的错误率降到了11%。难怪LeCun会说：已解决CIFAR-10，目标 ImageNet(ILSVRC)。

1.2 快速简易结构

已有的测试推出了一个Cifar-10的CNN深度、广度基本结构，理论上这个网络容量能够支持把验证集错误率降到25%左右。结构如下:



Part II ReLu激活函数、初始化W、学习率、样本均值归一化

2.1 ReLu

Xavier刚在2011年的Deep Sparse Rectifier Neural Networks论文中给出了ReLu在非监督学习网络中的卓越成效。

Alex就在2012年的AlexNet中，把所有Tanh/Logistic全换成了ReLu（卷积+隐层，Softmax要取概率没办法），但是没有介绍细节。（在其源码中可以看到）

正如我的前一篇文章ReLu激活函数提到的，ReLu为网络引入了大量的稀疏性，加速了复杂特征解离。非饱和的宽广映射空间，加速了特征学习。

实际测Model的时候，可以对比一下tanh和ReLu。

2.2 初始化W

如果生搬硬套Tanh/Logistic的初始化W经验规则：

对于Log-Sigmoid： $[-4*\sqrt{6}/\sqrt{LayerInput+LayerOut}, 4*\sqrt{6}/\sqrt{LayerInput+LayerOut}]$

对于Tanh-Sigmoid：
$[\sqrt{6}/\sqrt{LayerInput+LayerOut}, \sqrt{6}/\sqrt{LayerInput+LayerOut}]$

那么你会死的很难看。情况①：NaN。情况②：似然2.3000，验证集错误率90%。情况③：不论怎么训练，验证集在70~80%波动，然后过拟合。

主要问题是：①经验规则适用于[-1,1]的输入，如果直接用，那么会爆网络。②ReLu的非饱和线性端是做回归用的激活函数，输出近似高斯分布。

基于以上两点，Alex选择了高斯分布生成零均值、小标准差的随机值作为初始化W，并且逐层加大标准差，使得W有弹性。（0.0001-0.01-0.01-0.1-0.1）

由于高斯分布的的不均匀性（大值有，超小值也有），使得经过ReLu激活，不会有很无解的爆数值问题.当然输入不要太大，[0,256]（原始）or

[-128,128](减均值)

2.3 学习率

ReLu的线性不饱和激活端着相对于Tanh的双饱和端（经验规则0.1），肯定要降量级。

Caffe和Alex给的Model基础都是0.001(W)/0.002(b)。

至于为什么Bias的学习率是2倍，个人猜测是更快抑制Wx加权结果，加速学习。

2.4 样本均值归一化（重点）

Alex 和 Caffe中的初始化参数都是基于均值归一化的，如果不做归一化，会因为输入大了一半，导致训练失败。

这也是为什么Caffe强制为样本计算图像均值的原因。

一个简单策略是：训练样本均值归一化。即对训练集所有样本计算各个维度的均值（比如32x32图像，就应该有32x32个均值）

并且将均值存储起来。训练网络时，训练集、验证集减去存起来的均值。测试网络时，测试集减去存起来的均值。（一定要全减去训练集的均值）

这样，像素值[0,255]被调整成了近似[-128,128]。

尽管图像数据格式规整，但是做一做归一化还是挺有用处的。

Part III 重叠降采样、均采样

3.1 重叠

传统的CNN中，Pooling是不重叠的。不重叠忽视了邻近像素的对特征的影响，会造成网络精度下降。(不做重叠Pooling，很难迅速达到25%)

Alex在paper中还提到，重叠Pooling一定程度上减轻了过拟合。

为了对比测试重叠/非重叠模型，我在Caffe中对默认的Fast Model进行了修改。

参数~Kernel Conv1：(3,3)、Conv2:(4,4)、Conv3:(5，5) Pooling大小都是(2,2)，在epoch 9开始，lr降一个量级。

Fast 10 epochs Validation Error Rate

Epoch/Model	1	2	3	4	5	6	7	8	9	10
重叠（默认）	44%	37%	34%	30%	30%	30%	30%	30%	25%	25%
非重叠（修改）	52%	44%	39%	37%	35%	34%	33%	32%	29%	29%

可以看到，重叠结构带来更高的精度和更快的特征学习。

3.2 均采样

值得注意的是，重叠结构在提高精度的同时，可能引入噪声。而Alex则在Conv2、Conv3全使用了Avg Pooling

我猜测可能和Pooling的重叠结构有关，如果对不断重叠的Pooling结果，依旧全使用Max Pooling，那么有更大可能把噪声放大了。

而此时使用Avg Pooling就能把引入的噪声给降噪。于是，我在Caffe中对默认重叠模型，进行了不同组合Pooling的测试。

Fast 10 epochs Validation Error Rate（Overlapping）

Epoch/Pooling	1	2	3	4	5	6	7	8	9
Max+Avg+Avg（默认)(最好)	44%	37%	34%	30%	30%	30%	30%	30%	25%
Max+Max+Max	47%	42%	37%	37%	34%	32%	31%	31%	27%

Max+Max+Avg（较好）	42%	37%	33%	32%	30%	30%	30%	30%	26%
Avg+Avg+Avg（略差）	50%	43%	40%	38%	36%	35%	34%	34%	31%
Avg+Max+Avg	47%	40%	37%	35%	34%	34%	32%	32%	31%
Avg+Max+Max（最差）	47%	42%	40%	38%	37%	36%	36%	36%	33%

【分析】

可以看到，Alex使用的Max+Avg+Avg无论在精度，还是在速度上，都远超其他组合。

Max+Max+Max在速度和精度上都不如Max+Max+Avg，说明Avg的引入效果还不错。

比较奇葩的下面3个组合，我们首先看到的是Avg+Avg+Avg，在错误率下降速度方面排倒数第一。网络训练速度也是倒一（Avg计算比Max慢）

然后是Avg+Max+Max，在精度方面排倒数第一，而在Conv3用Avg换掉Max就好很多。

【总结】

从Avg Pooling角度来看，这玩意不仅计算量大，而且引入噪声，比Max Pooling不知差到哪里去了。

所以，凡是在Conv1使用Avg Pooling的，都没有好果子吃，精度太差。所以，没有重叠的Conv1更应该考虑使用Max Pooling。

在Conv2、Conv3中，使用Avg反而效果很好了，可能原因是抵消了重叠结构带来的部分噪声。两个Avg效果好于一个Avg，说明Avg也不是那么坏嘛。

3.3 非重叠均采样

然而当我把上述组合用在非重叠结构里时，看起来不太对劲。

Fast 10 epochs Validation Error Rate（Non-Overlapping）

Epoch/Pooling	1	2	3	4	5	6	7	8	9
Max+Avg+Avg(较好)	52%	44%	39%	37%	35%	33%	32%	32%	29%
Max+Max+Max(最好)	50%	40%	37%	35%	34%	33%	33%	32%	29%
Avg+Avg+Avg(最差)	56%	48%	45%	41%	40%	38%	37%	36%	33%
Avg+Max+Max(较差)	53%	45%	41%	39%	37%	37%	37%	36%	33%

可以看到，此时Max+Max+Max在速度上领先了Max+Avg+Avg，这和重

叠结构的情况相反。

Avg+Avg+Avg依旧最差，Avg+Max+Max也不好过，因为它在Conv1用了Avg。

对比了两个结构，不难发现，Avg在重叠结构里面作用巨大，而在非重叠结构里面，如果用Avg，则会影响网络训练。

除此之外，不难发现，非重叠结构的精度明显不如重叠结构，这也是为什么Alex论文贴的那张图比较奇葩了。

Cifar-10想要快速达到25%错误率，传统LeNet修改版结构是很难满足要求的。

3.3 长时间完全训练

来自Caffe中的full_train,跑完vaild error大概可以优化到 19%。

这个数字的贡献有二：

①重叠结构：

非重叠结构在epoch 10左右达到临界29%，之后再训练，会衰退到32%～35%

而重叠结构加Max Pooling，最后能维持在28%左右，最后降低学习率，大概能提升到25%

②Avg Pooling

重叠结构+Avg Pooling的突破点在epoch 20~30，这时候，与Max Pooling不同的是，error，突的一下，

从28%掉到23%。然后在epcoh 40, lr降一个量级后，从23%飙到19%。

可以看到Avg Pooling和重叠结构带来的精度提升，在训练后期，真是厚积薄发。

3.4 基本没有用的LRN层

尽管Caffe中默认是加上去的，但是根据作者在知乎的说法，只是为了尊重前人的成果。

examples只是简单的复现了Alex的结构。