Tensorflow：

基于 GoogLeNet 的图像处理

郑志超，田亚雷，江志东，张昊阳，黄进

摘要：

由于深度学习和卷积网络的发展，图像目标分类和检测能力得到了显著提高。过去几年深度学习神经元网络的大部分的进步不仅仅是更强大硬件、更大数据集、更大模型的结果，而主要是新的想法、算法和网络结构改进的结果，比较明显的就是 Google 推出的 Google Inception Net,大大提高了深度神经元网络的层级。

Googe Inception Net 首次出现在 ILSVRC2014 的比赛中(和 VGGNet 同年)，以较大的优势获得冠军。那一届的 GoogleNet 通常被称为 Inception V1，Inception V1 的特点是控制了计算量的参数量的同时，获得了非常好的性能-top5 错误率 6.67%，这主要归功于 GoogleNet 中引入一个新的网络结构 Inception 模块，所以 GoogleNet 又被称为 Inception V1(后面还有改进版 V2、V3、V4)架构中有 22 层深，V1 比 VGGNet 和 AlexNet 都深，但是它只有 500 万的参数量,计算量也只有 15 亿次浮点运算,在参数量和计算量下降的同时保证了准确率，可以说是非常优秀并且实用的模型。

1. 引言

Google Inception Net 发展至今共经历了 4 个版本的迭代，它们分别是：

Inception V1:

Inception V1 中精心设计的 Inception Module 提高了参数的利用率；Inception V1 去除了模型最后的全连接层，用全局平均池化层(将图片尺寸变为 1x1)，在先前的网络中，全连接层占据了网络的大部分参数，很容易产生过拟合现象；[1]

Inception V2:

Inception V2 学习了 VGGNet，用两个 3*3 的卷积代替 5*5 的大卷积核(降低参数量的同时减轻了过拟合)，同时还提出了注明的 Batch Normalization(简称 BN)方法。BN 是一个非常有效

的正则化方法，可以让大型卷积网络的训练速度加快很多倍，同时收敛后的分类准确率可以的到大幅度提高。

BN 在用于神经网络某层时，会对每一个 mini-batch 数据的内部进行标准化处理，使输出规范化到(0,1)的正态分布，减少了 Internal Covariate Shift(内部神经元分布的改变)。BN 论文指出，传统的深度神经网络在训练时，每一层的输入的分布都在变化，导致训练变得困难，我们只能使用一个很小的学习速率解决这个问题。而对每一层使用 BN 之后，我们可以有效的解决这个问题，学习速率可以增大很多倍，达到之间的准确率需要的迭代次数有需要 1/14，训练时间大大缩短，并且在达到之间准确率后，可以继续训练。以为 BN 某种意义上还起到了正则化的作用，所有可以减少或取消 Dropout，简化网络结构。[7]

当然，在使用 BN 时，需要一些调整：

增大学习率并加快学习衰减速度以适应 BN 规范化后的数据
去除 Dropout 并减轻 L2 正则(BN 已起到正则化的作用)
去除 LRN
更彻底地对训练样本进行 shuffle
减少数据增强过程中对数据的光学畸变(BN 训练更快，每个样本被训练的次数更少，因此真实的样本对训练更有帮助)

Inception V3：

Inception V3 在两个方面优化了原有 Inception V2,首先，它引入了 Factorization into small convolutions 的思想，将一个较大的二维卷积拆成两个较小的一位卷积，比如将 7*7 卷积拆成 1*7 卷积和 7*1 卷积（下图是 3*3 拆分为 1*3 和 3*1 的示意图）。 一方面节约了大量参数，加速运算并减去过拟合，同时增加了一层非线性扩展模型表达能力。论文中指出，这样非对称的卷积结构拆分，结果比对称地拆分为几个相同的小卷积核效果更明显，可以处理更多、更丰富的空间特征、增加特征多样性。[8]

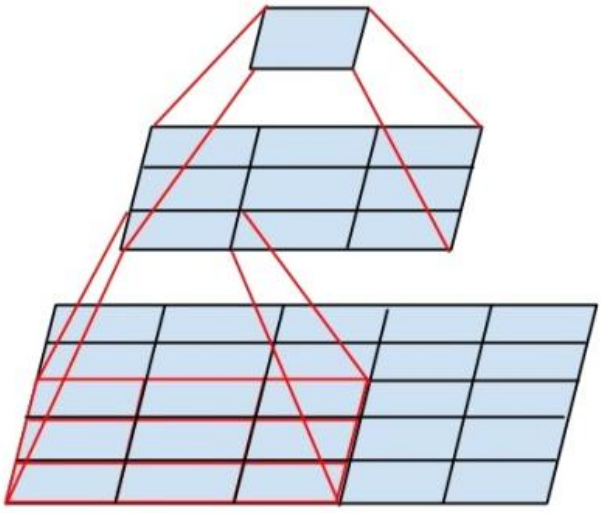

图 1：迷你网络替换 5x5 卷积[8]

Inception V3 优化了 Inception Module 的结构，现在 Inception Module 有 35*35、17*17 和 8*8 三种不同的结构。这些 Inception Module 只在网络的后部出现，前部还是普通的卷积层。并且还在 Inception Module 的分支中还使用了分支。

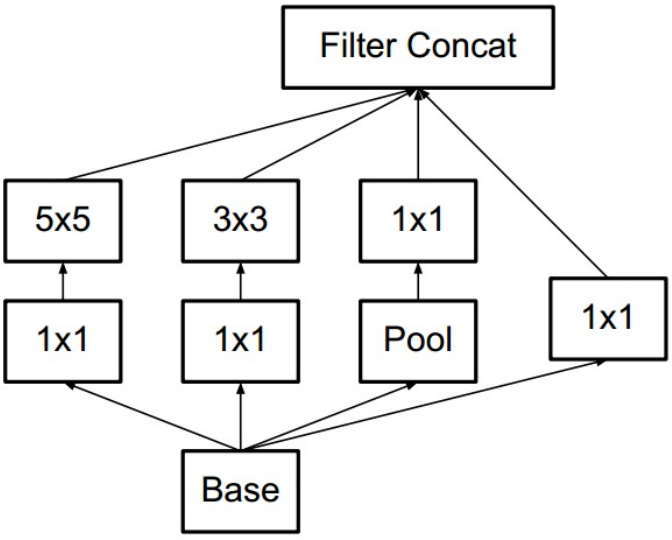

图 2：Inception 原结构图

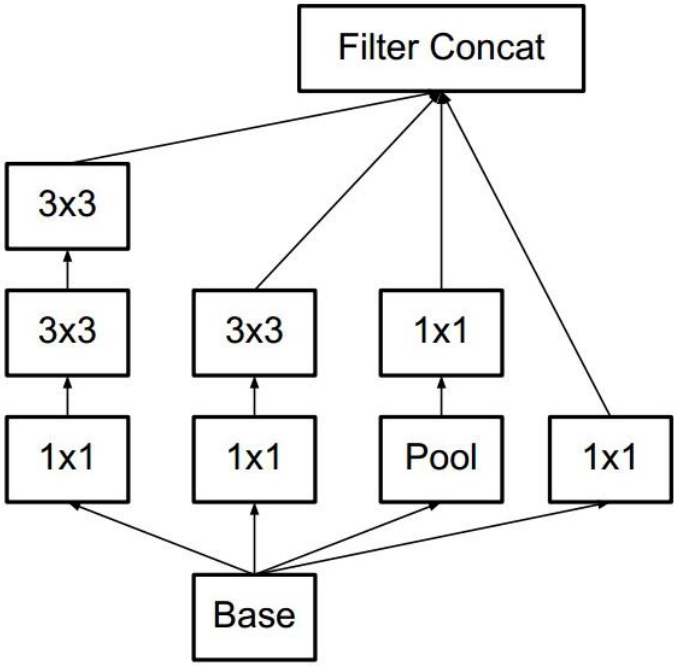

图 3：使用 2 个 2x3 替换 5x5 后的 Inception 结构

Inception V4:

Ineption v4 研究了 Inception 模块结合 Residual Connection 能不能有改进？发现 ResNet 的结构可以极大地加速训练，同时性能也有提升，得到一个 Inception-ResNet v2 网络，同时还设计了一个更深更优化的 Inception v4 模型，能达到与 Inception-ResNet v2 相媲美的性能。

Inception-v4 共有四种结构，一种不包含 Residual connection 结构的；包含 Residual connection 结构的，根据包含 Inception 模块的不同又分为 2 种：Inception-ResNet-v1 和 Inception-ResNet-v2。[9]

2.细节分析：

Inception module 是将 Hebbian 原理应用在神经网络上，如果数据集的概率分布可以被一个很大很稀疏的神经网络表达，那么构筑这个网络的最佳方法是逐层构筑网络：将上一层高度相关的节点聚类，并将聚类出来的每一个小簇连接到一起。

Hebbian 原理描述了突触可塑性的基本原理，即神经反射活动的持续与重复会导致神经元连续稳定性持久提升，当两个神经元细胞 A 和 B 距离很近，并且 A 参与了对 B 重复、持续的兴奋，那么某些代谢会导致 A 将作为使 B 兴奋的细胞。总结一下:“一起发射的神经元会连在一起”,学习过程中的刺激会使神经元间突触强度增加。[10] 在符合 Hebbian 原理的基础上，我们应该把相关性高的一簇神经元节点连接在一起。在普通的数据集中，这可能需要对神经元节点做聚类，但是在图片数据中，天然的就是临近区域的数据相关性高，因此相邻的像素点被卷积操作连接在一起（符合 Hebbian 原理），而卷积操作就是在做稀疏连接。在 CNN 模型中，我们可能有多个卷积核，在同一空间位置但在不同通道的卷积核的输出结果相关性极高。我们可以使用 1*1 的卷积很自然的把这些相关性很高的、在同一空间位置但是不同通道的特征连接在一起。

一般来说，提升网络性能最直接的方式就是增加网络的大小:
1.增加网络的深度
2.增加网络的宽度

这样简单的解决办法有两个主要的缺点:
1.网络参数的增多，网络容易陷入过拟合中，这需要大量的训练数据，而在解决高粒度分类的问题上，高质量的训练数据成本太高;
2.简单的增加网络的大小，会让网络计算量增大，而增大计算量得不到充分的利用，从而造成计算资源的浪费

解决上面的两个缺点的思路：

将全连接的结构转换为稀疏结构(即使是内部卷积)
如果数据集的概率分布可以可以有大型的稀疏的深度神经网络表示，则优化网络的方法可以

是逐层的分析层输出的相关性，对相关的输出做聚类操作。

当对非均匀稀疏数据结构计算时，计算效率非常低，这需要在底层的计算库做优化;不均匀稀疏模型需要复杂的计算工程实现和设备,大多数面向视觉的机器学习系统只是利用了卷积在空间域中稀疏特性。稀疏矩阵乘法有一个良好的实践办法是将稀疏矩阵聚类成相对密集的子矩阵。Inception 算法试图逼近隐含在视觉网络中的稀疏结构，并利用密集、易实现的组件来实现这样的假设(隐含的稀疏结构)。[1]

Inception 架构的主要思想是建立在找到可以逼近的卷积视觉网络内的最优局部稀疏结构，并可以通过易实现的模块实现这种结构; 使用大的卷积核在空间上会扩散更多的区域，而对应的聚类就会变少，聚类的数目随着卷积核增大而减少，为了避免这个问题，inception 架构当前只使用 1*1,3*3,5*5 的滤波器大小，这个决策更多的是为了方便而不是必须的。[1]

将池化单元组合到一起就会面临更加明显的问题:他们输出滤波器数量等于上一阶段滤波器数量，每个阶段的跨越这就不可避免的会增加输出数量。在计算力需求急速增长的部分我们明智的应用降维和 projections 技术。1*1 卷积做 compute reductions, 而不是计算 3*3 或 5*5 的卷积;除此之外，1*1 卷积还可以用来矫正线性激活。

在 Inception Module 中，通常 1*1 卷积的比例(输出通道数占比)最高，3*3 卷积和 5*5 卷积稍低。整个模型中，会有多个堆叠的 Inception Module，我们希望靠后的 Inception Module 可以捕捉更高阶的抽象特征，因此靠后的 Inception Module 的卷积的空间集中度应该逐渐降低，这样可以捕获更大面积的特征。因此，越靠后的 Inception Module 中，3*3 和 5*5 这两个大面积的卷积核的占比(输出通道数)应该更多。

3．GoogLeNet:

始于 LeNet-5，一个有着标准的堆叠式卷积层冰带有一个或多个全连接层的结构的卷积神经网络。通常使用 dropout 来针对过拟合问题。 为了提出一个更深的网络，GoogLeNet 做到了 22 层，利用 inception 结构，这个结构很好地利用了网络中的计算资源，并且在不增加计算负载的情况下，增加网络的宽度和深度。同时，为了优化网络质量，采用了 Hebbian 原理和多尺度处理。GoogLeNet 在分类和检测上都取得了不错的效果。最近深度学习的发展，大多来源于新的想法，算法以及网络结构的改善，而不是依赖于硬件，新的数据集，更深的网络，并且深度学习的研究不应该完全专注于精确度的问题上，而更应该关注与网络结构的改善方面的工作。

GoogLeNet 使用了平均池化代替 FC 层可以提高 top-1 精度 0.6%,需要注意的是即使是移除掉了 FC 层，仍需要使用 dropout 技术 。网络的中间层应该是很有判别力的(考虑到网络深度有 22 层，且有一些浅层的模型表现的很好) 我们期望分类器可以在较低阶段就可以区分，故在网络的中间层添加了辅助分类器，这不仅可以在 BP 中增加传播的梯度信号，而且可以提供额外的正则化 。在训练过程中，辅助分类器的 loss 也计算到总的 loss 中，loss 以不同比例的权重计算(占比为 0.3)，在 inference 阶段，辅助分类器不使用。[1]

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

表 1：基于 Inception module 的 GoogleLeNet 架构图

4.使用 Tensorflow 实现 GoogLeNet：

由于 Google Inception Net 相对比较复杂，我们模仿着在 TensorFlow 的 GitHub 代码库上开源的 Inception-V3 模型的源码，V3 模型中使用的 Inception 模型相比 V1 和 V2 有着更加复杂多样的网络结构；

Inception V3 模型共有 46 层，有三个 Inception 模型组(三层、五层、三层)，共有 96 的卷积层，可以想象 V3 的代码会很长，这里使用 tf.contrib.slim 工具辅助设计网络， 简化代码。

类型	kernel 尺寸/步长(或注释)	输出尺寸
conv0	3 * 3 / 2	149 * 149 * 32
conv1	3 * 3 / 1	147 * 147 * 32
conv2	3 * 3 / 1	147 * 147 * 64
pool1	3 * 3 / 2	73 * 73 * 64
conv3	3 * 3 / 1	73 * 73 * 80

conv4	3 * 3 / 1	71 * 71 * 192
pool2	3 * 3 / 2	35 * 35 * 192
Inception 模块	mixed_b	35 * 35 * 256
	mixed_c	35 * 35 * 288
	mixed_d	35 * 35 * 288
Inception 模块	mixed_a	17 * 17 * 768
	mixed_b	17 * 17 * 768
	mixed_c	17 * 17 * 768
	mixed_d	17 * 17 * 768
	mixed_e	17 * 17 * 768
Inception 模块	mixed_a	8 * 8 * 1280
	mixed_b	8 * 8 * 2048
	mixed_c	8 * 8 * 2048
池化	8 * 8	1 * 1 * 2048
logits	logits	1 * 1 * 1000
Softmax	分类输出	1 * 1 * 1000
辅助分类器		
avg_pool	5 * 5 / 3	5 * 5 * 768
conv0	1 * 1 / 1	5 * 5 * 128
conv1	5 * 5 / 1	1 * 1 * 768
logits	logits	1 * 1 * 1000

表 2：使用 Tensorflow 实现 GoogLeNet 网络总览

输出（前向传播时间）：

```
2018-06-23 11:43:30.350918: step 0, duration = 0.167
2018-06-23 11:43:32.019900: step 10, duration = 0.166
2018-06-23 11:43:33.689330: step 20, duration = 0.166
2018-06-23 11:43:35.360135: step 30, duration = 0.167
2018-06-23 11:43:37.034390: step 40, duration = 0.166
2018-06-23 11:43:38.715208: step 50, duration = 0.167
2018-06-23 11:43:40.395576: step 60, duration = 0.166
2018-06-23 11:43:42.072846: step 70, duration = 0.166
2018-06-23 11:43:43.751294: step 80, duration = 0.167
2018-06-23 11:43:45.428440: step 90, duration = 0.166
2018-06-23 11:43:46.939761: Forward across 100 steps, 0.168 +/- 0.002 sec / batch
```

从结果上来看，V3 网络的性能还不错，虽然输入图片比 VGGNet 的 224*224 大了 78%，但是 forward 速度却比 VGGNet 更快。 这主要归功于其较小的参数量，inception V3 参数量比 inception V1 的 700 万多了很多，不过仍然不到 AlexNet 的 6000 万参数量的一半。相比 VGGNet 的 1.4 亿参数量就更少了。整个网络的浮点计算量为 50 亿次，比 inception V1 的 15 亿次大了不少，但是相比 VGGNet 来说不算大。因此较少的计算量让 inception V3 网络变得非常实用，可以轻松地移植到普通服务器上提供快速响应服务，甚至移植到手机上进行实时的图像识别。

代码 Github 地址：
https://github.com/jiangzhidong2/MachineLearningG4/blob/master/GoogleNet.py

总结

V3 作为一个极深的网络，拥有巧妙的设计和构造，整个网络结构分支非常复杂。我们平时可能不必设计这么复杂的网络，但 V3 中有许多思想和技巧值得借鉴。

1.Factorization into small convolutions 很有效，可以降低参数量、减轻过拟合，增加网络的非线性的表达能力。
2.卷积网络从输入到输出，应该让图片尺寸逐渐减小，输出通道数逐渐增多，即让空间结构简化，将空间信息转化为高阶抽象的特征信息。
3.Inception Module 用多个分支提取不同抽象程度的高阶特征的思路很有效，可以丰富网络的表达能力。

由于时间以及小组成员分散关系，没有成功的对网络结构进行修改并打印出运行的差异，也没有找到足够的数据集进行训练。仅对 InceptionV3 进行了模拟运行。

引用:

[1] *Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich.*
Going Deeper with Convolutions.

[2] *Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, Xiaoqiang Zheng.*
TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems.

[3] *Yuan Tang*
TF.Learn: TensorFlow's High-level Module for Distributed Machine Learning

[4] *Soheil Bahrampour, Naveen Ramakrishnan, Lukas Schott, Mohak Shah*
Comparative Study of Deep Learning Software Frameworks

[5] *Abhinav Vishnu, Charles Siegel, Jeffrey Daily*
Distributed TensorFlow with MPI

[6] *Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, Michael Collins*
Globally Normalized Transition-Based Neural Networks

[7] *Sergey Ioffe, Christian Szegedy*
Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

[8] *Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna*
Rethinking the Inception Architecture for Computer Vision

[9] *Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi*
Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning

[10] 赫布理论
https://baike.baidu.com/item/%E8%B5%AB%E5%B8%83%E7%90%86%E8%AE%BA/8347084?fr=aladdin