SqueezeNet

SqueezeNet 一种轻量化卷积神经网络。

原文地址：https://arxiv.org/pdf/1602.07360.pdf

SqueezeNet 是 Han 等提出的一种轻量且高效的 CNN 模型，它参数比 AlexNet 少 50x，但模型性能(accuracy)与 AlexNet 接近。在可接受的性能下，小模型相比大模型，具有很多优势：(1)更高效的分布式训练，小模型参数小，网络通信量减少；(2)便于模型更新，模型小，客户端程序容易更新；(3)利于部署在特定硬件如 FPGA，因为其内存受限。

1.1 相关工作

1.1.1 模型压缩

目的：确定一个参数很少的模型并保持准确性

常用的模型压缩技术有：奇异值分解(SVD)、网络剪枝、深度压缩、硬件压缩(EIE).

1.1.2 CNN 微观结构

在设计深度网络架构的过程中，如果手动选择每一层的滤波器显得过于繁复。通常先构建由几个卷积层组成的小模块，再将模块堆叠形成完整的网络。定义这种模块的网络为 CNN microarchitecture.指对单个卷积层进行优化设计，如采用 1x1 的小卷积核，还有很多采用可分解卷积(factorized convolution)结构或者模块化的结构(blocks, modules)。

1.1.3 CNN 宏观结构

与模块相对应，定义完整的网络架构为 CNN microarchitecture。指网络架构层面上的优化设计，如网路深度(层数)，还有像 ResNet 那样采用“短路”连接(bypass connection)。

1.1.4 神经网络设计空间

由于超参数繁多，深度神经网络具有很大的设计空间(Design Space)。通常

进行设计空间探索的方法有：

(1)贝叶斯优化(bayesian optimization)

(2)模拟退火(simulated annealing)

(3)随机搜索(randomized search)

(4)遗传算法(genetic algorithms)

1.2 SqueezeNet：在更少的参数情况下保持模型的精度

1.2.1 结构设计策略

对于一个 CONV layer，它的参数数量计算应该是：K×K×M×N。其中，K 是 filter 的 spatial size，M 和 N 分别是输入 feature map 和输出 activation 的 channel size。这篇文章主要目的是寻找保持高精度同时参数更少的网络结构，因此使用以下三个主要策略来减少 SqueezeNet 设计参数：

(1)使用 1x1 的卷积代替 3x3 的卷积：参数量减少为原来的 1/9，这相当于减少上述所说的 K；

(2)减少 3x3 卷积的输入通道数，相当于减少上述所说的 M；

(3)将下采样操作延后，以获得更大的激活图：更大的激活特征图能够保留更多的信息，可以提供更高的分类准确率。

(1)和(2)可以显著减少模型的参数量，(3)可以在参数量一定时提高正确率。

1.2.2 Fire 模块

Fire 模块是构建 SqueezeNet 的基本模块，以下定义 Fire 模块：

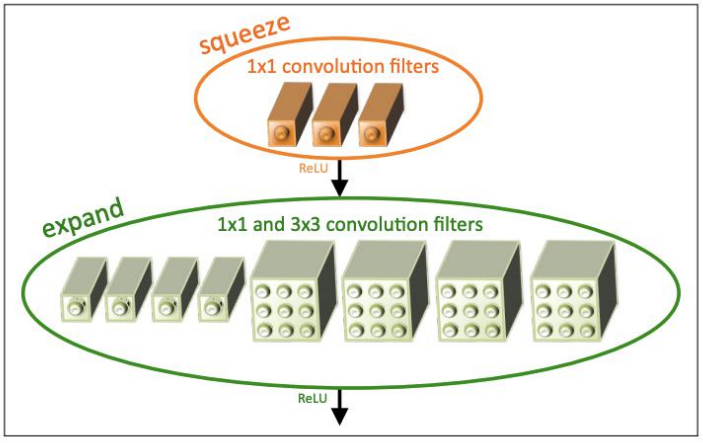
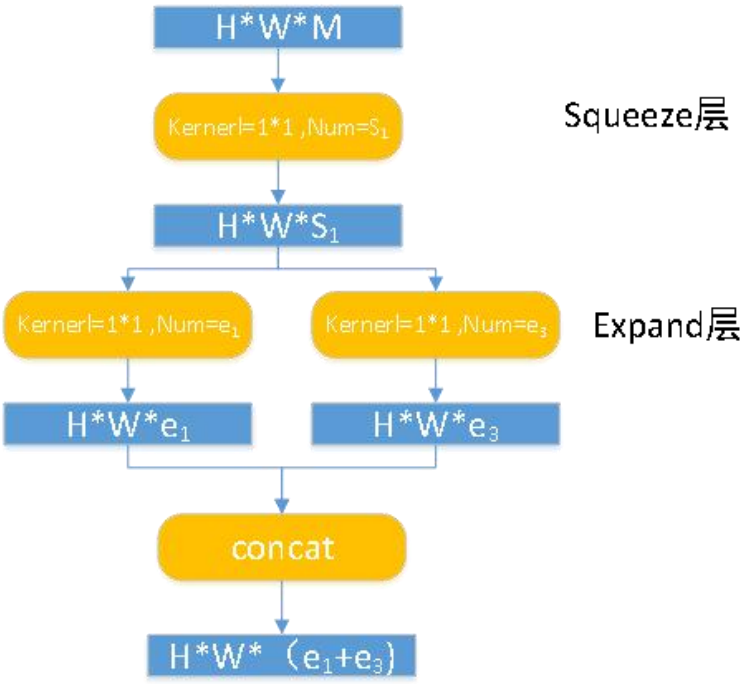


Figure 1: Microarchitectural view: Organization of convolution filters in the **Fire module**. In this example, $s_{1x1} = 3$, $e_{1x1} = 4$, and $e_{3x3} = 4$. We illustrate the convolution filters but not the activations.

- Squeeze 卷积层：只使用 1×1 卷积，即使用了策略(1)
- expand 层：使用 1×1 和 3×3 的卷积核组合
- Fire Module 中使用了三个可以调节的超参数：$s_{1\times1}$(squeeze 卷积层中 1x1 卷积核的个数)、$e_{1\times1}$(expand layer 中 1x1 卷积核的个数)、$e_{3\times3}$(expand layer 中 3x3 卷积核的个数)。
- 使用 Fire Module 的过程中，令$s_{1\times1} < e_{1\times1} + e_{3\times3}$，这样 Squeeze layer 可以限制 3x3 卷积核输入的通道数量，即使用了策略(2)。



1.2.3 SqueezeNet 架构

SqueezeNet 以卷积层(conv1)开始，接着使用了 8 个 Fire Modules，最后以卷积层(conv10)结束。每个 Fire module 中的卷积数量逐渐增加，并且在 conv1，fire4，fire8 和 conv10 这几层后使用步长为 2 的 max-pooling，即将池化层放在相对靠后的位置，这是使用了以上的策略(3)。总结构如下图所示：左边是原始的 SqueezeNet，中间是包含 simple bypass 的改进版本，右边是使用了 complex bypass 的改进版本。

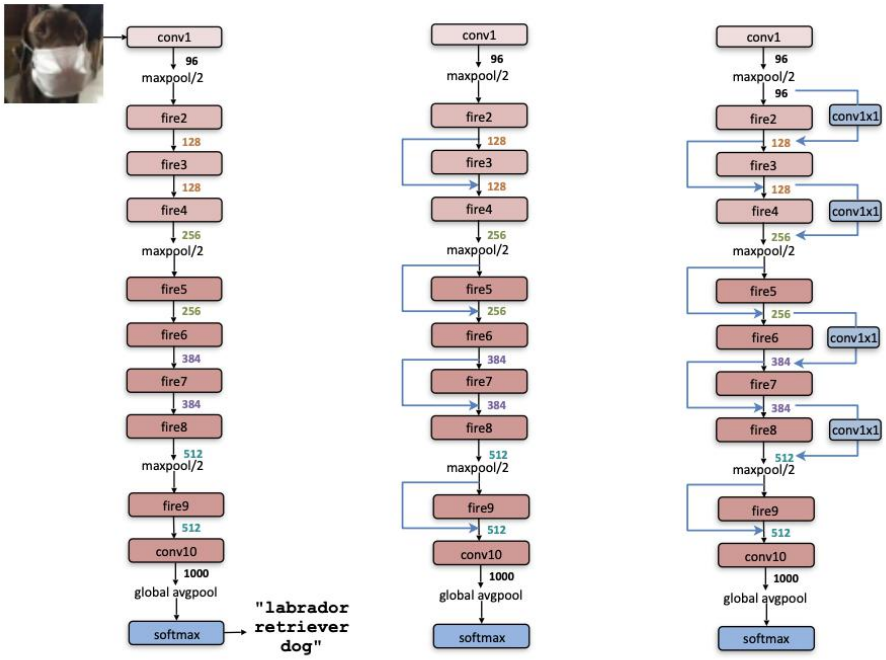


Figure 2: Macroarchitectural view of our SqueezeNet architecture. Left: SqueezeNet (Section 3.3); Middle: SqueezeNet with simple bypass (Section 6); Right: SqueezeNet with complex bypass (Section 6).

1.2.4 其他细节

- 为了使和卷积核输出的特征图有相同的尺寸，在 expand module 中，给中的原始输入添加一个像素的边界(zero padding)。
- squeeze 和 expand layers 两者都使用 RELU 作为激活函数。
- 在 fire 9 module 之后，使用 Dropout，随即丢弃的概率设置为 50%。
- 在训练过程中，初始学习率设置为 0.04，在训练过程中线性降低学习率。
- 由于 Caffe 框架中不支持使用两个不同尺寸的卷积核，在 expand layer 中实际是使用了两个单独的卷积层(和)，最后将这两层的输出连接起来，这在数值上等价于使用单层但是包含不同尺寸的卷积核。