精简卷积神经网络简介

以MobileNets和SkipNet为例

李振宇 桂林电子科技大学

October 28, 2019

大纲

1. 卷积
2. **MobileNets**
3. **SkipNet**
4. 更多工作

卷积

卷积 1: 卷积层的计算

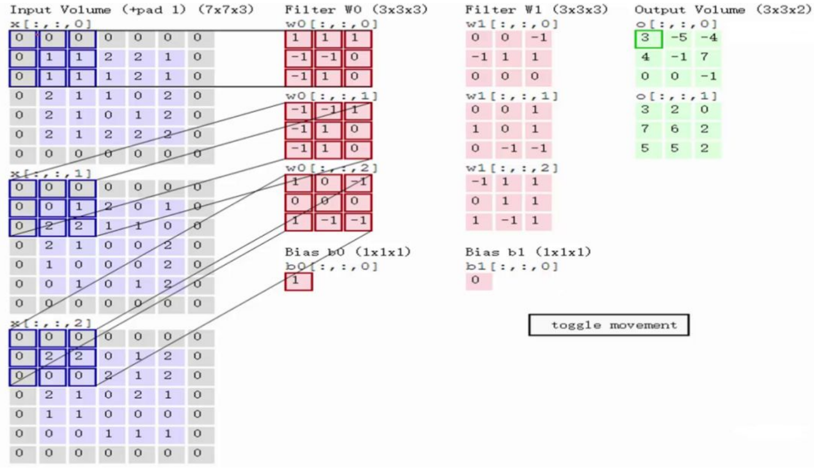


Figure. 卷积计算过程

卷积 2: 卷积层的参数

卷积计算的公式：

$$G_{k,l,n} = \sum_{i,j,m} K_{i,j,m,n} \cdot F_{k+i-1,l+j-1,m}$$

卷积层的参数：

Filter 滤波器数量

Kernel Size 卷积核大小

Strides 步长

Padding 填充

卷积 3: 池化层

虽然今天没有涉及池化的优化，但卷积总是和池化分不开的：

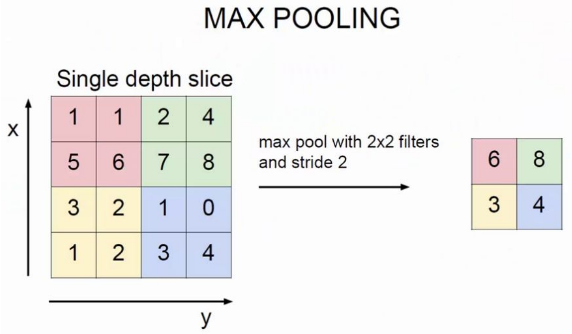


Figure. 最大池化

MobileNets

虽然AlexNet并非轻量化神经网络，但其中的两个方法，却是如今轻量化网络的重要基石：

Group Conv AlexNet中，为了在两个GPU上并行训练，其将卷积按照通道分解为两个Group，从而在两个GPU上完成。

ReLU AlexNet中，采用了ReLU激活函数，相比先前的tanh和sigmoid函数，ReLU函数的计算速度，和训练时的收敛速度都更快。

$$\mathrm{ReLU}(x) = \max(0, x)$$

MobileNets 3: Group Conv

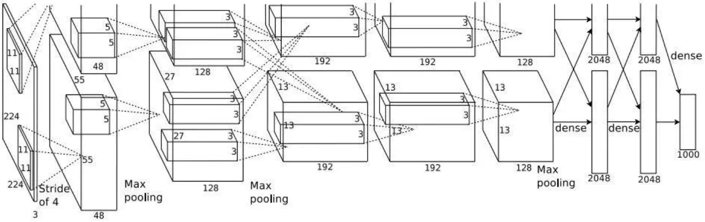


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Figure. AlexNet的Group Conv结构

MobileNets 4: Depthwise Conv

虽然AlexNet的Group Conv的目的是将计算分配给两个GPU并行运行，但是这也同样减少了计算量。

如果说将每个通道，都分到其对应的Group中，这种Group Conv的特殊情况，就将其运算量减少到了最少的情况。但这样做的代价是通道之间的互信息被忽略了。

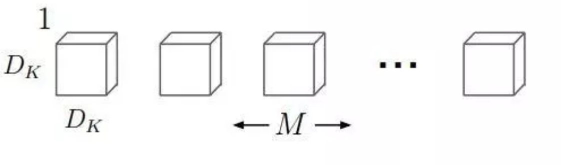


Figure. Depthwise Conv，其中M是通道数，也是滤波器数

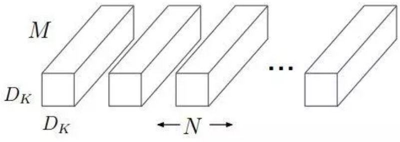


Figure. 标准的Conv（相对于 Depthwise Conv）

$$\mathbf{G}_{k,l,n} = \sum_{i,j,m} \mathbf{K}_{i,j,m,n} \cdot \mathbf{F}_{k+i-1,l+j-1,m} \text{普通卷积}$$

$$\hat{\mathbf{G}}_{k,l,m} = \sum_{i,j} \hat{\mathbf{K}}_{i,j,m} \cdot \mathbf{F}_{k+i-1,l+j-1,m} \text{Depthwise Conv}$$

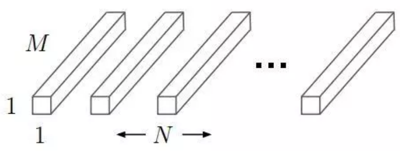


Figure. Pointwise Conv，即1 结构的普通卷积

计算复杂度：

$$D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F \text{ 普通卷积}$$

$$D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F \text{ Depthwise + Pointwise Conv}$$

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2}$$

MobileNets 7: 模型设计

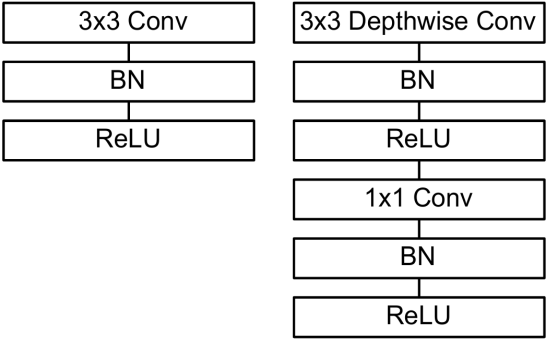


Figure. MobileNets模块

MobileNets 8: 模型设计

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
5× Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

MobileNets 9: 模型性能

Table 2. Resource Per Layer Type

Type	Mult-Adds	Parameters
Conv 1×1	94.86%	74.59%
Conv DW 3×3	3.06%	1.06%
Conv 3×3	1.19%	0.02%
Fully Connected	0.18%	24.33%

同时，为了进一步压缩模型大小，MobileNets设计了超参数宽度乘子α，使得MobileNets模块的输入通道数变为αM，输出通道数变为αN，此时计算复杂度变为：

$$D_K \cdot D_K \cdot \alpha M \cdot D_F \cdot D_F + \alpha M \cdot \alpha N \cdot D_F \cdot D_F$$

MobileNets 10: 模型性能

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

Table 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60

SkipNet

Gating units 在Highway Networks中，引入了这一概念。如果说原始的神经网络可以表示为：

$$\boldsymbol{y} = H(\mathbf{x}, \boldsymbol{W_H}) \tag{1}$$

引入$T(\mathbf{x}, \boldsymbol{W_T})$, $C(\mathbf{x}, \boldsymbol{W_C})$两个函数，作为变换gate和运载gate：

$$\boldsymbol{y} = H(\mathbf{x}, \boldsymbol{W_H}) \cdot T(\mathbf{x}, \boldsymbol{W_T}) + \mathbf{x} \cdot C(\mathbf{x}, \boldsymbol{W_C}) \tag{2}$$

设$C = 1 - T$，公式(2)变换为：

$$\boldsymbol{y} = H(\mathbf{x}, \boldsymbol{W_H}) \cdot T(\mathbf{x}, \boldsymbol{W_T}) + \mathbf{x} \cdot (1 - T(\mathbf{x}, \boldsymbol{W_T})) \tag{3}$$

又令$H' = H \cdot T$：

$$\boldsymbol{y} = H‘(\mathbf{x}, \boldsymbol{W_H}) + \mathbf{x} \cdot (1 - T(\mathbf{x}, \boldsymbol{W_T})) \tag{4}$$

可以看到，加入一个与原网络模块并列的gating unit 即可起到“加速信息流通”的作用。

Gating策略

实际上，ResNet的结构便符合这一标准：

$$x^{i+1}_{\text{ResNet}} = F^i\left(x^i_{\text{ResNet}}\right) + x^i_{\text{ResNet}} \quad (5)$$

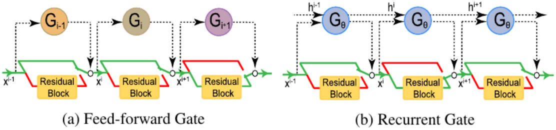


Figure. 两种不同的gating策略设计

作者针对ResNet设计了两种gating策略，一种是所有模块共用一个gate unit，还有一种是对每一个模块施加一个gate unit。

Gating units设计

针对不同的网络模块类型，作者设计了不同的Gating units：

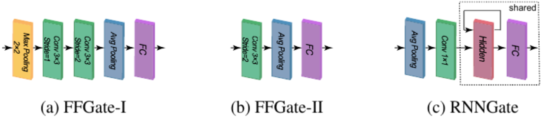


Figure. 三种不同的gating units

FFGate-I 大约消耗残差模块的19%的性能；
FFGate-II 大约消耗残差模块的12.5%的性能；
RNNGate 包含了单个LSTM模块，仅消耗0.04%的微不足道的性能。

对于模块的两个输出大小的估计，直接使用soft-max策略进行训练，随后在计算向后传播时恢复到硬判定使得模型的准确率非常的低，因此作者设计了混合强化学习策略，将跳过模块省略的计算作为“奖励”，因此得到的新的目标函数为：

$$
\begin{aligned}
\min \mathcal{J}(\theta) &= \min \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{g}} L_{\theta}(\mathbf{g}, \mathbf{x}) \\
&= \min \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{g}} \left[\mathcal{L}\left(\hat{y}\left(\mathbf{x}, F_{\theta}, \mathbf{g}\right), y\right) - \frac{\alpha}{N} \sum_{i=1}^{N} R_i \right]
\end{aligned} \tag{6}
$$

其中$R_i = (1 - g_i)\, C_i$作为跳过计算的奖励，常数C_i为计算消耗，而ResNet中，可以视为$C_i = 1$。

最终的训练策略

Algorithm 1: Hybrid Learning Algorithm (HRL+SP)

Input: A set of images $\mathbf{x}$ and labels $\mathbf{y}$

Output: Trained SkipNet

1. Supervised pre-training (Sec. 3.3)

$\theta_{SP} \leftarrow \mathrm{SGD}(L_{\text{Cross-Entropy}}, \text{SkipNet-}G_{\text{relax}}(\mathbf{x}))$

2. Hybrid reinforcement learning (Sec. 3.2)

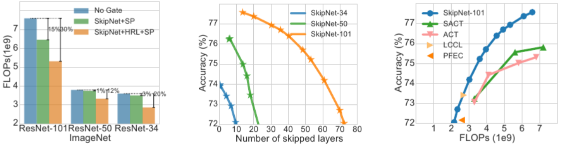
Initialize θ_{HRL+SP} with θ_{SP}

$\theta_{HRL+SP} \leftarrow \mathrm{REINFORCE}(\mathcal{J}, \text{SkipNet-}G(\mathbf{x}))$

Figure. 最终的训练策略

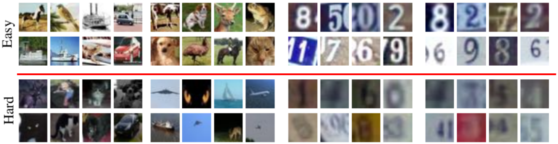
$$G_{\text{relax}}(\boldsymbol{x}) = \begin{cases} \mathbb{I}(S(\boldsymbol{x}) \geq 0.5), \text{ forward pass} \\ \qquad\qquad S(\boldsymbol{x}), \text{ backward pass} \end{cases}$$

模型性能



(a) Computation Reduction (b) Acc.-compt. Trade-off (c) Comparison with Others

Figure. 模型性能对比



(a) SkipNet +FFGate CIFAR-10 (b) SkipNet +RNNGate CIFAR-10 (c) SkipNet +FFGate SVHN (d) SkipNet +RNNGate SVHN

Figure. 跳过的模块数量与输入的复杂度有关

更多工作

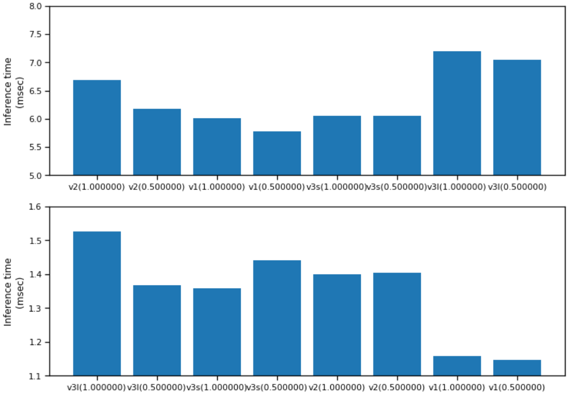
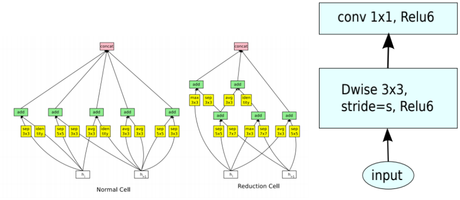


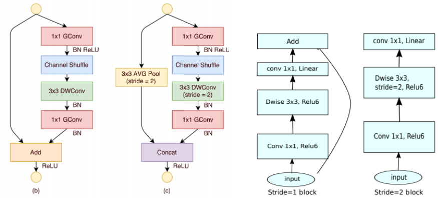
Figure. 各版本MobileNet在CPU和GPU上性能的表现

更多工作 3: 如何改进现有网络

根据今天讲到的网络特性，对于不同的实际使用环境，如何改进现有网络？



(a) NasNet[23] (b) MobileNet[27]



(c) ShuffleNet [20] (d) Mobilenet V2

Figure. MobileNet V2与其它网络的对比

更多工作 5: MobileNet V3

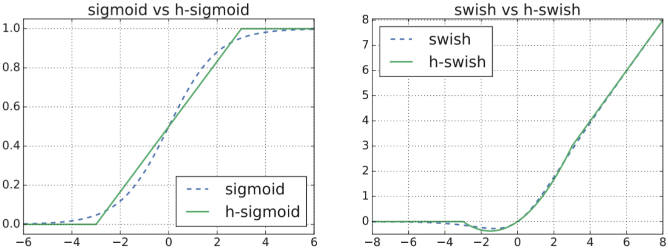


Figure. Sigmoid 和 Swish 的非线性形式及其hard近似

$$\text{swish } x = x \cdot \sigma(x)$$

$$\text{h} - \text{ swish } [x] = x\frac{\mathrm{Re\,LU6}(x+3)}{6}$$

今天的内容结束了，谢谢大家。