1.1 亮点

（1） 提出两个学习 pooling 的方法（结合 max pooling 和 average pooling）

（2） 构建一种树状的自学习池化滤波器

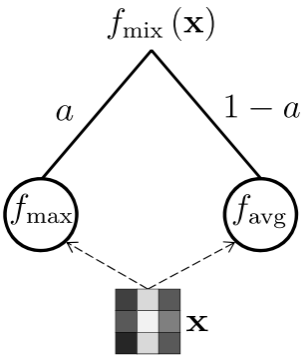
1.2 初衷

在 CNN 网络中主要有两个操作，一个是卷积，另一个是池化，然后在 CNN 中，卷积的参数是学习而来的，但是 pool 中，无论是 max pooling 和 average pooling 均不是学习而来，因此作者提出一种方法使得 CNN 的 pooling 也是学习得到的，不仅没有改变 pooling 不变性，仅仅增加一点计算量即可令模型的效果刷到了 state of art。

1.3 原理

作者提出了两种学习 pooling 的算法，第一种是把传统的 pooling 方法组合起来，第二种是组合一些学习到的 pooling 方法（tree pooling），其中第一种又分为两种实现，第一种实现为单纯地把 max pooling 和 average pooling 组合起来(mixed pooling)，第二种实现是在第一种的基础上加一个门限（gate pooling）。

1.3.1 mixed max-average pooling

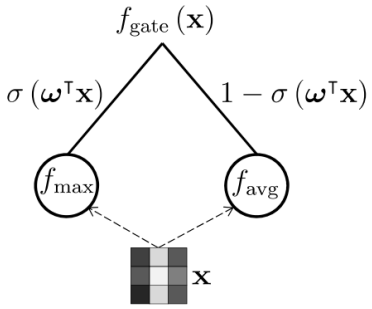


average pooling 是把池化区域中的所有像素取平均值，max pooling 是取所有像素的最大值，然后两种方法都在特定的场合里有非常有的作用，因此我们希望能把他们放在一起，根据不同的情况去选择用哪一种 pooling 方法，因此有：

$$f_{\text{mix}}(\mathbf{x}) = a_\ell \cdot f_{\text{max}}(\mathbf{x}) + (1 - a_\ell) \cdot f_{\text{avg}}(\mathbf{x}),$$

其中参数α_l是学习得到的，但是这也有一个问题，在训练结束之后，α_l一旦保持不变，也就意味着 pooling 的方法与被 pooling 的区域的特性无关，这样不太好，最好是有一个类似自适应的 pooling 方法，根据被 pooling 区域的特征确定 pooling 方法。这就产生了 gate max-average pooling。

1.3.2 gate max-average pooling

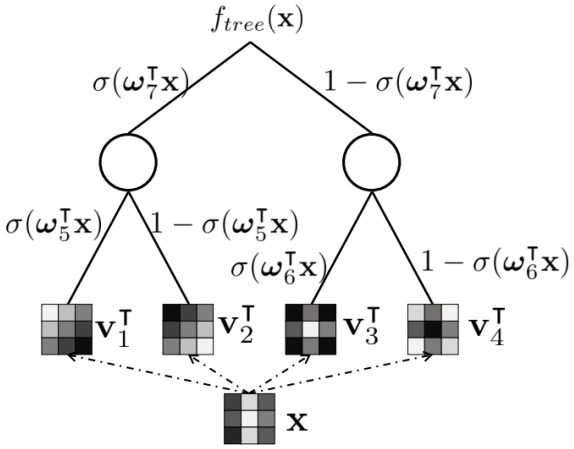


gate max-average pooling 把第一种方法的权值变成了门限权值矩阵与被 pooling 区域的像素值的内积，这样子学习的并不是α_l参数，而是权值矩阵 W：

$$f_{\text{gate}}(\mathbf{x}) = \sigma(\boldsymbol{\omega}^{\mathsf{T}}\mathbf{x}) f_{\max}(\mathbf{x}) + (1-\sigma(\boldsymbol{\omega}^{\mathsf{T}}\mathbf{x})) f_{\text{avg}}(\mathbf{x})$$

在第二种方法中，我们可以看到原来的权值换成了权值矩阵与被 pooling 区域像素点的内积，这样被 pooling 区域矩阵 x 携带了这张特定图片自身的信息，导入了图片信息。

1.3.3 tree pooling



首先，用一个池化滤波器与被池化区域做内积，比如我们有四个池化滤波器，分别做完内积就是四个叶子节点。然后，用 gated pooling 的思路，四个叶子节点的数值分别乘以 gated pooling 的权重再相加，得到两个节点。最后，这两个节点再做一次 gated pooling，得到根节点，根节点的值就是这个区域池化得到的结果。

1.4 实验

Method	MNIST	CIFAR10	CIFAR10+	CIFAR100
Baseline	0.39 ± 0.031	9.01 ± 0.096	7.22 ± 0.099	34.38 ± 0.096
w/ **Stochastic** no learning	0.38 ± 0.04	8.50 ± 0.05	7.30 ± 0.07	33.48 ± 0.27
w/ **50/50 mix** no learning	0.34 ± 0.012	8.11 ± 0.10	6.78 ± 0.17	33.53 ± 0.16
w/ **Mixed** 1 per pool layer 2 extra params	0.33 ± 0.018	8.09 ± 0.19	6.62 ± 0.21	33.51 ± 0.11
w/ **Mixed** 1 per layer/ch/rg >40k extra params	0.30 ± 0.012	8.05 ± 0.16	6.58 ± 0.30	33.35 ± 0.19
w/ **Gated** 1 per pool layer 18 extra params	**0.29** ± 0.016	**7.90** ± 0.07	**6.36** ± 0.28	**33.22** ± 0.16

Method	MNIST	CIFAR10	CIFAR10$^+$	CIFAR100	SVHN
Our baseline	0.39 ± 0.031	9.01 ± 0.096	7.22 ± 0.099	34.38 ± 0.096	1.89 ± 0.069
Tree 2 level; 1 per pool layer	0.34 ± 0.028	8.52 ± 0.175	6.54 ± 0.156	33.64 ± 0.285	1.81 ± 0.047
Tree 3 level; 1 per pool layer	0.38 ± 0.032	8.43 ± 0.091	6.38 ± 0.165	32.85 ± 0.181	1.73 ± 0.096
Tree+Max-Avg 1 per pool layer	**0.31** ± 0.031	**7.61** ± 0.121	**6.02** ± 0.047	**32.87** ± 0.278	**1.70** ± 0.069

实验结果还是非常不错的，有时间才 caffe 或者 TensorFlow 下可以实现一下。