Kernel Pooling for Convolutional Neural Networks

Yin Cui[1,2] Feng Zhou[3] Jiang Wang[4] Xiao Liu[3] Yuanqing Lin[3] Serge Belongie[1,2]

[1] Department of Computer Science, Cornell University [2] Cornell Tech

[3] Baidu Research [4] Google Research

CVPR July 21-26 HONOLULU 2017

Motivation

- ➢ **A Convolutional Neural Network (CNN) can be regarded as a feature extractor.**
- ➢ **However, most CNN uses a simple linear classifier.**
- ➢ **Recent work have demonstrated the power of 2nd order information on a wide range of visual tasks.**
- ➢ **Can we bring kernel classifier into CNNs?**

Possible Solutions

- Implicit feature mapping via kernel function. ✗
  - Both time and space complexity is O(n) w.r.t. the number of training data n.
  - Need to construct a kernel matrix, hard to be trained with SGD.
- Explicit feature mapping. ✓
  - High dimension: $O(d^p)$ for p^{th} order w.r.t. original feature dimension d. **But can be compactly approximated.**

Application

- Small dataset. CNN learned feature is not good enough.

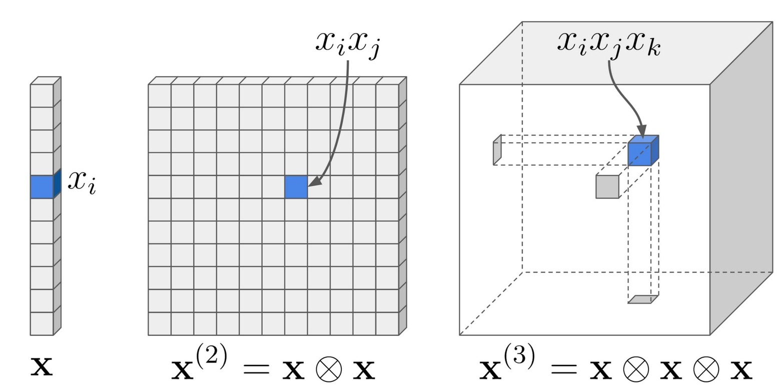
Kernel Pooling

Tensor Product

- Tensor product (p-level), a generalization of outer product.

$$\mathbf{x}^{(2)} = \mathbf{x} \otimes \mathbf{x} = \begin{bmatrix} x_1x_1 & x_1x_2 & \cdots & x_1x_c \\ x_2x_1 & x_2x_2 & \cdots & x_2x_c \\ \vdots & \vdots & \ddots & \vdots \\ x_cx_1 & x_cx_2 & \cdots & x_cx_c \end{bmatrix} \in \mathbb{R}^{c^2}$$

$$\mathbf{x}^{(p)} = \underbrace{\mathbf{x} \otimes \cdots \otimes \mathbf{x}}_{\text{p times}} \in \mathbb{R}^{c^p}$$



Taylor Series Kernel

- Taylor series kernel of order p: $\mathcal{K}_{\text{Taylor}}(\mathbf{x}, \mathbf{y}) = \sum_{i=0}^{p} \alpha_i^2 (\mathbf{x}^\top \mathbf{y})^i$
- Gaussian RBF (via approximation) and Polynomial are special cases of Taylor series kernel.
- The explicit feature map of Taylor series kernel can be represented by tensor product:

$$\phi_{\text{Taylor}}(\mathbf{x}) = [\alpha_0 (\mathbf{x}^{(0)})^\top, \ldots, \alpha_p (\mathbf{x}^{(p)})^\top]^\top \qquad (\mathbf{x}^\top \mathbf{y})^p = (\mathbf{x}^{(p)})^\top (\mathbf{y}^{(p)})$$

Compact Approximation

- Approximating tensor product by Count Sketch:

$$\tilde{\mathbf{x}}^{(p)} = \text{FFT}^{-1}(\text{FFT}(\mathcal{C}_1(\mathbf{x})) \circ \cdots \circ \text{FFT}(\mathcal{C}_p(\mathbf{x})))$$

$$\mathcal{C}(\mathbf{x}) = [c_1, c_2, \ldots, c_d]^\top, \text{where } c_i = \sum_{i:h(i)=j} s(i)\mathbf{x}_i$$

h(.) and s(.) are two hash functions. Their outputs are uniformly drawn from {1,2, … ,d} and {+1, -1}

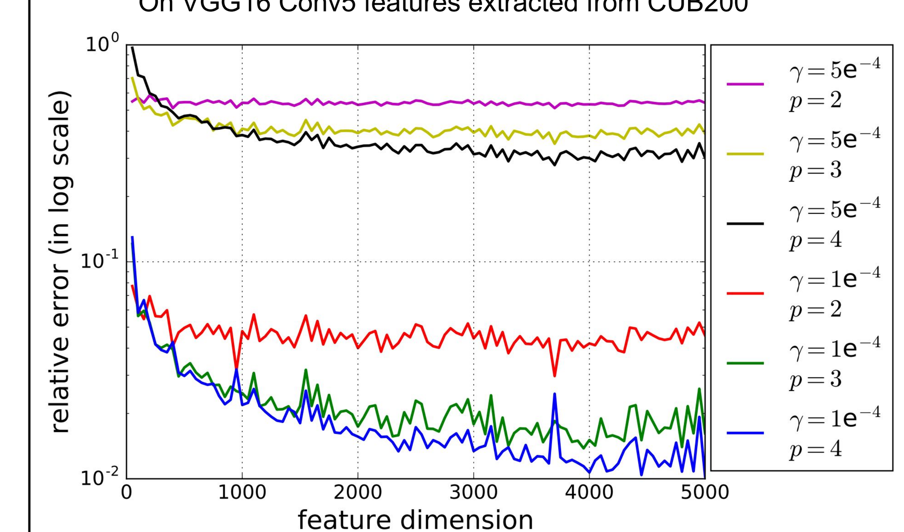
Experiments

Summary of Polling Strategies

	AlexNet / VGG	Inception / ResNet	Bilinear	Compact Bilinear	**Ours**
Strategy	$\sigma(W_2\sigma(W_1X))$	$\frac{1}{hw}\sum_{i,j} X_{ij}$	$\frac{1}{hw}\sum_{i,j} X_{ij}X_{ij}^\top$	$\frac{1}{hw}\sum_{i,j} TS(X_{ij})$	$\frac{1}{hw}\sum_{i,j} \phi(X_{ij})$
Dimension	d	c	c^2	d	d
Time	$\mathcal{O}(hwcd)$	$\mathcal{O}(hwc)$	$\mathcal{O}(hwc^2)$	$\mathcal{O}(hw(c+d\log d))$	$\mathcal{O}(hwp(c+d\log d))$
Space	$\mathcal{O}(hwcd)$	0	0	$2c$	pc
Parameters	$\mathcal{O}(hwcd)$	0	0	0	0 or p

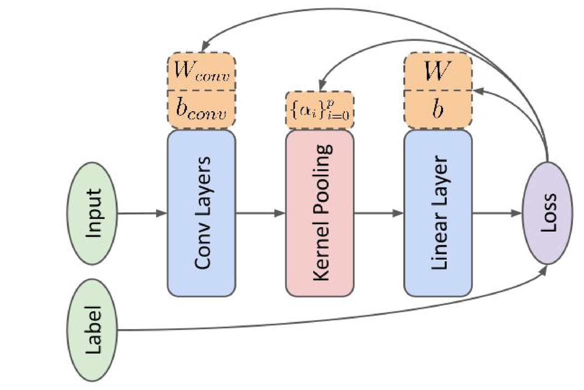
RBF Kernel Approximation Error

On VGG16 Conv5 features extracted from CUB200



Learning kernel compositions

Back-propagating loss to learn the coefficient for each term in Taylor series kernel:



Visual Recognition Accuracy

Dataset	CNN	Original	BP[23]	CBP[11]	KP	Others	
CUB [43]	VGG-16 [38]	73.1*	84.1	84.3	**86.2**	82.0	84.1
	ResNet-50 [15]	78.4	N/A	81.6	84.7	[18]	[16]
Stanford Car [19]	VGG-16	79.8*	91.3	91.2	**92.4**	**92.6**	82.7
	ResNet-50	84.7	N/A	88.6	91.1	[18]	[14]
Aircraft [27]	VGG-16	74.1*	84.1	84.1	**86.9**	80.7	
	ResNet-50	79.2	N/A	81.6	85.7	[14]	
Food-101 [4]	VGG-16	81.2	82.4	82.4	84.2	50.76	
	ResNet-50	82.1	N/A	83.2	**85.5**	[4]	

Image size of 224x224 is marked by *, otherwise 448x448 (except 224x224 for Food-101).
BP: Bilinear Pooling; CBP: Compact Bilinear Pooling; KP: Kernel Pooling

End-to-end Training with Kernel Pooling