Compression of Convolutional Layers by Variational Bayesian Matrix Factorization

Tomohiro Hayase
Doctoral student (Random Matrices),
Kawahigashi Lab, Math.Sci., University of Tokyo,

Motivation

Compress weights of CNN to speed up and to use without GPU (in particular for mobile applications).

Compression of deep convolutional neural networks for fast and low power mobile applications [KPYCYS 2016, ICLR]

Their technique consists of two steps:

1. Tucker decomposition
2. Stable rank selection by using the global analytic solution of variational Bayesian matrix factorization [NSBT 2013, JMLR]

Tucker Decomposition

Weights of Convolution Layers

A convolutional layer (kernel size = 3, number of output channels = S, number of input channels = T) has the weight: $K \in \mathbb{R}^S \otimes \mathbb{R}^T \otimes \mathbb{R}^3 \otimes \mathbb{R}^3$

The computation by the layer is given by

$$Y_{o,h,w} = \sum_{i,j,c} K_{o,c,i,j} X_{c,h_i,w_j},$$

$$h_i := (h-1)S + i - P, \;\; w_j := (w-1)S + j - P,$$

where S is stride, P is zero-padding,

Tucker-2 Decomposition

--- We have the following isomophisms as vector spaces:

$$K \in \mathbb{R}^S \otimes \mathbb{R}^T \otimes \mathbb{R}^3 \otimes \mathbb{R}^3 \simeq M_{S,9T}(\mathbb{R}) \simeq M_{T,9S}(\mathbb{R})$$

--- We can see K as two kinds of matrices. We apply SVD to each matrices:

$$K^{(1)} \in M_{S,9T}(\mathbb{R}),\ K^{(1)} = U^{(1)} D^{(1)} V^{(1)}$$

$$K^{(2)} \in M_{T,9S}(\mathbb{R}),\ K^{(2)} = U^{(2)} D^{(2)} V^{(2)}$$

--- Define new kernel $K' := (U^{(1)} \otimes U^{(2)} \otimes 1 \otimes 1)^T K$

Tucker-2 Decomposition

Equivalently,

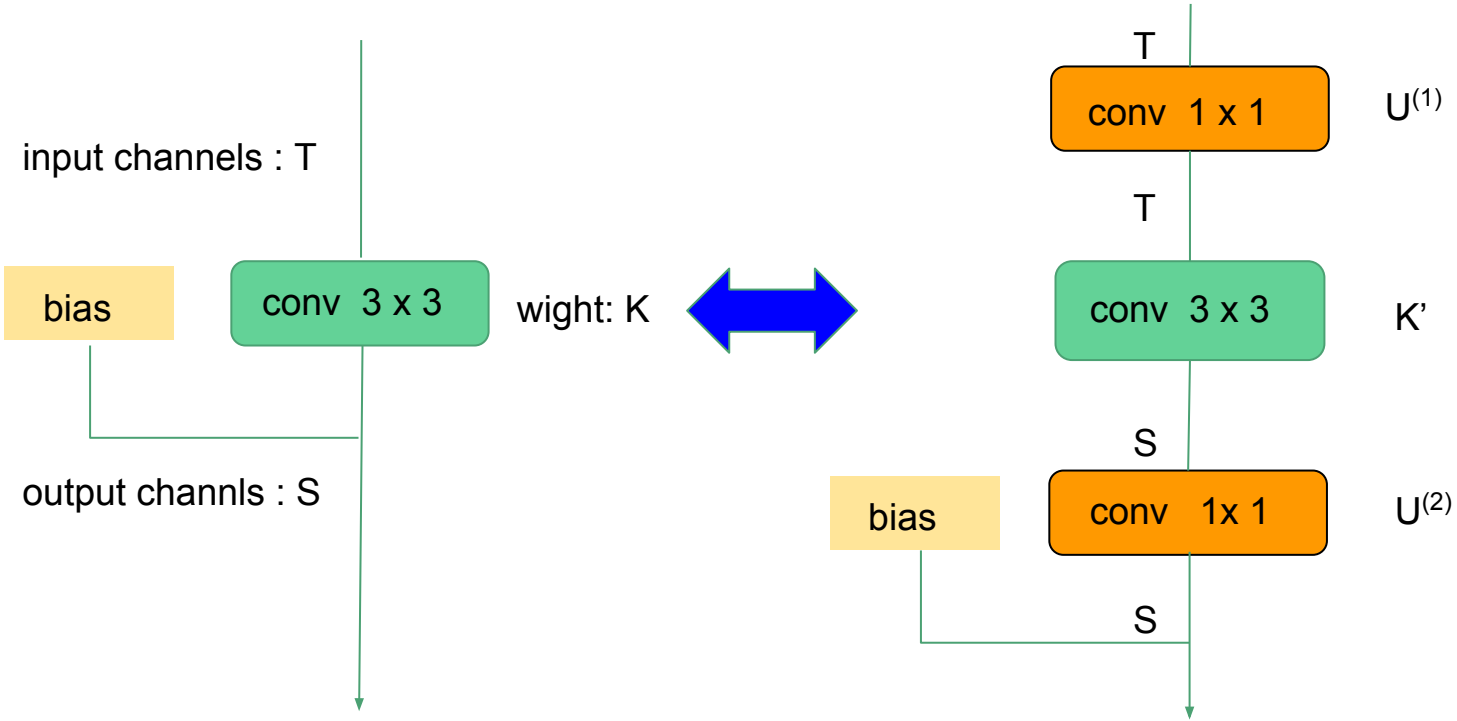
$$K'_{o',c',h,w} = \sum_{o,c} U^{(1)}_{o,o'} U^{(2)}_{c,c'} K_{o,c,h,w}, \;\; h, w = 0, 1, 2$$

Because of the orthogonality, it holds that

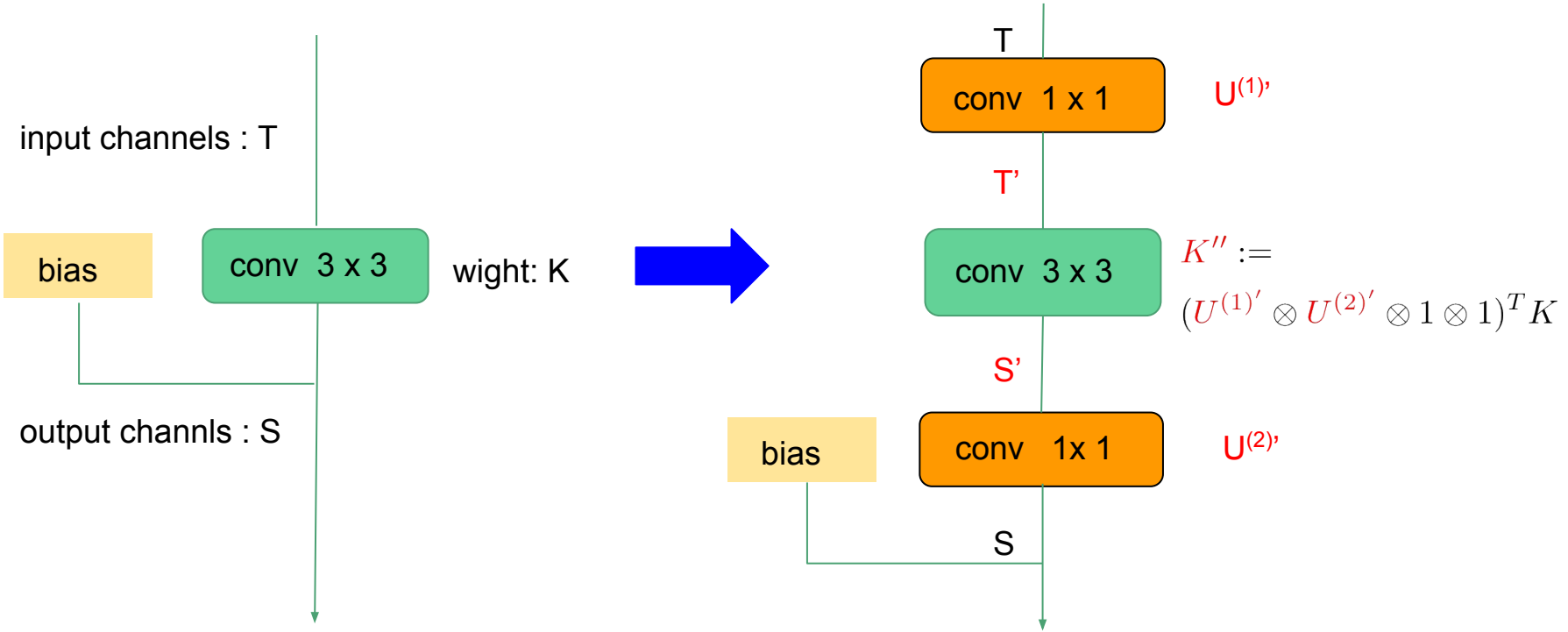
$$K = (U^{(1)} \otimes U^{(2)} \otimes 1 \otimes 1) K',$$

$$K_{o,c,h,w} = \sum_{c',o'} U^{(1)}_{o,o'} U^{(2)}_{c,c'} K_{o',c',h,w}, \;\; h, w = 0, 1, 2.$$

Tucker-2 Decomposition



Reduced Tucker-2 Decomposition



How to estimate ranks ?

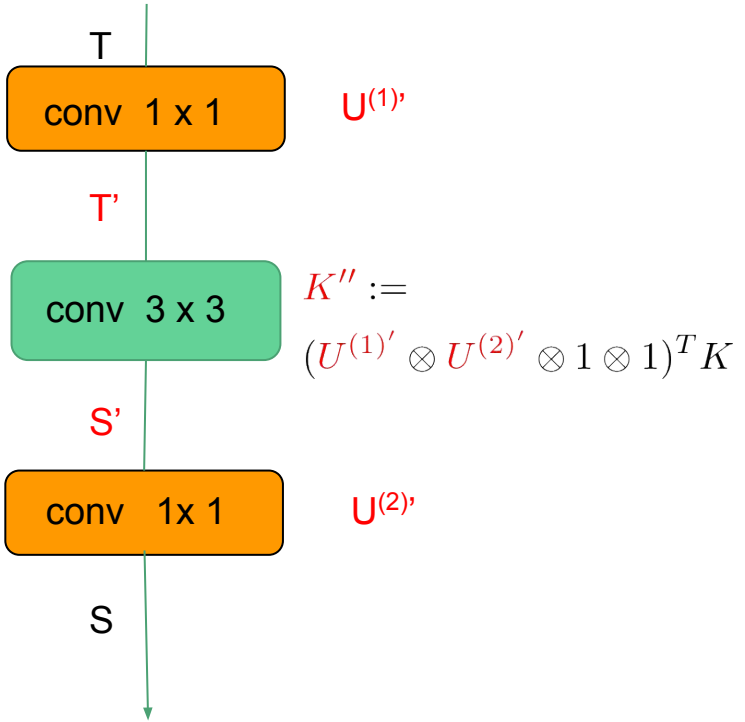
Consider reduced SVD (Pruning off some small singular values) of two representation matrices of K.

Q. How to estimate ranks T' and S' ?

A1. Do experimentally with monitoring loss:

for T', S' > 0 and test loss > thres:
Reduce T', S'.
Fine-tune whole network.
Compute test loss.
end for

A2. Estimate ranks by variational Bayesian matrix factorization and fine-tune.



Variational Bayesian Matrix Factorization

Bayesian Matrix Factorization

$$\mathbf{V} = \mathbf{B}\mathbf{A}^T + \sigma^2\mathbf{Z},$$

$$\mathbf{B} = B + \Sigma_B^{1/2}\mathbf{Z_B}, \mathbf{A} = A + \Sigma_A^{1/2}\mathbf{Z_A}.$$

$A, B, \Sigma_A, \Sigma_B \in M_{M,L}(\mathbb{R}), \Sigma_A, \Sigma_B \geq 0$, and $\{ \mathrm{Z}, \mathrm{Z}_A, Z_B\}$ is independent family of real Gaussian random matrices (i.e. entries are i.i.d. Gaussian N(0,1))

$$P(V|\mathbf{B}, \mathbf{A}) := \frac{1}{(2\pi\sigma^2)^{M/2}} \exp\left(-\frac{1}{2\sigma^2}||V - \mathbf{B}\mathbf{A}^T||_F^2\right), \; P(V) = \langle P(V|\mathbf{B}, \mathbf{A})\rangle$$

For given matrix V, the optimization task is formulated as minimizing the Bayes free energy -logP(V):

$$\underset{A,B,\Sigma_A,\Sigma_B,\sigma}{\operatorname{argmin}} \; [-\log P(V)]$$

The estimation is given as the expectation: $\hat{V} = \frac{1}{P(V)}\langle \mathbf{B}\mathbf{A}^T P(V|\mathbf{B}, \mathbf{A})\rangle$

Variational Bayes

Taking expectation (computing P(V)) is difficult...

Variational Bayes methods is one of the deterministic alternative of minimizing free energy (minimizing a lower bound of free energy).

Main idea:

Restrict posterior distributions.

(In Variational Bayesian MF, we assume that posterior distributions of **B** and **A** are independent.)

Global Analytic Solution of VBMF

Fortunately, VBMF has a global analytic solution. Assume that $L \leq M$. Set $\alpha = L/M$. Let τ be the unique zero of the decreasing function Ξ:

$$\Xi(\tau) := \Psi(\tau) + \Psi(\tau/\alpha), \ \Psi(t) := \frac{log(z+1)}{z} - \frac{1}{2}.$$

Let $(\gamma_h)_{h \leq L}$ be the singular values of observed matrix V.

The estimated variance $(\hat{\sigma}^{\mathrm{EVB}})^2$ by VBMF is the unique mimimizer of the following 1-dim function

$$\Omega(\sigma^{-2}) := \frac{1}{L} \sum_{h=1}^{L} \psi(\gamma_h^2 / M\sigma^2).$$

where

$$\begin{aligned}
\psi(x) &:= 1 - \log x + \mathbf{1}(x > \underline{x}) \left(\log(\tau(x;\alpha)+1) + \alpha \log(\tau(x;\alpha)/\alpha + 1) - \tau(x;\alpha),\right. \\
\tau(x;\alpha) &:= \frac{1}{2}\left(x - (1+\alpha) + \sqrt{(x-(1+\alpha))^2 - 4\alpha}\right), \\
\underline{x} &:= (1+\underline{\tau})(1+\alpha/\underline{\tau}).
\end{aligned}$$

Rank selection by VBMF

The rank H^{EVB} of estimated $\hat{V}^{\mathrm{EVB}}$ satisfies

$$\gamma_{H^{\mathrm{EVB}}+1} \leq \hat{\sigma}^{\mathrm{EVB}} \sqrt{M\underline{x}} \leq \gamma_{H^{\mathrm{EVB}}},$$

where $(\gamma_h)_{h \leq L}$ is the singular values of the observed matrix V.

Experiments

CNN : fully convolutional CNN (13+1 conv layers, similar to VGG16) :

weight : 137MB, top-5 accuracy: 87 % (on ILSVRC2012, 1000-class, 50000 test images)

⇒ (Tucker on VBMF) ⇒

weight : 13MB (i.e. 1/10)

Fine-tune by ILSVRC2012: The results of top-5 accuracy(on test sets) are

1 epoch: 80% ⇒ 10 epoch: 84%

Summary of Algorithm

1. Represent each kernel K as two matrices $K^{(1)}$, $K^{(2)}$.
2. For each matrix, calculate the singular values (γ_h) and solve the 1-dim optimizing problem $\min\Omega(\sigma^{-2})$ of σ.
3. Find the rank H^{EVB} . This gives us the reduced number of channels T', S'.
4. Reduce the orthogonal matrices: $U^{(1)'} = U[:, :S']$, $U^{(2)'} = U[:, :T']$
5. Calculate the new kernel K''.
6. Copy the weights of the undecomposed layers.
7. Fine-tune.

Related Works

Similar (or advanced) structure of Tucker decomposition:

-- The Fire Module : SqueezeNet https://arxiv.org/pdf/1602.07360.pdf

-- The Depth-wise Separable Convolution : MobileNets
https://arxiv.org/pdf/1704.04861.pdf

The other compression methods:

-- Network Pruning with quantization to 8bit and Huffman encoding:
Deep Compression https://arxiv.org/pdf/1510.00149.pdf