Beyond Filters: Compact Feature Map for Portable Deep Model

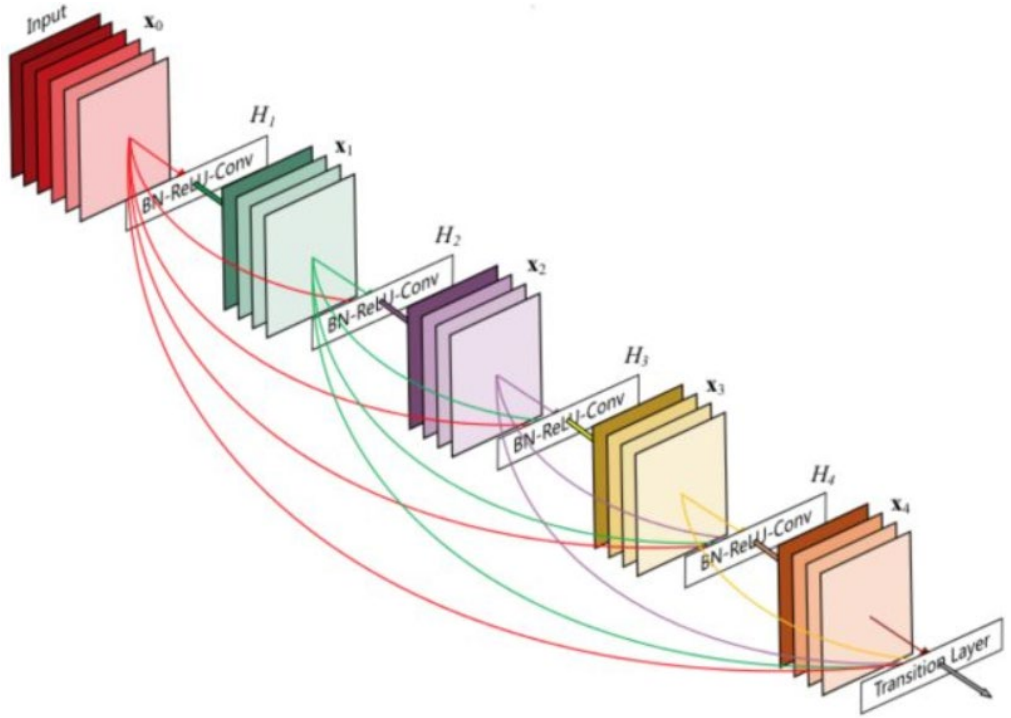
ICML2017

Framework

Feature Map Y

Dimensionality Reduction

New Feature Map $\tilde{Y}$ ⟷ Input $\tilde{X}$

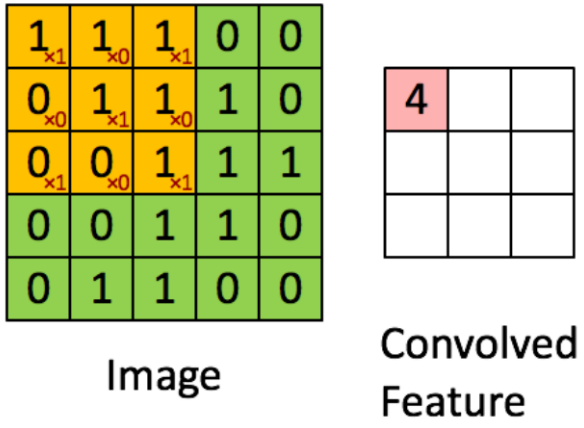


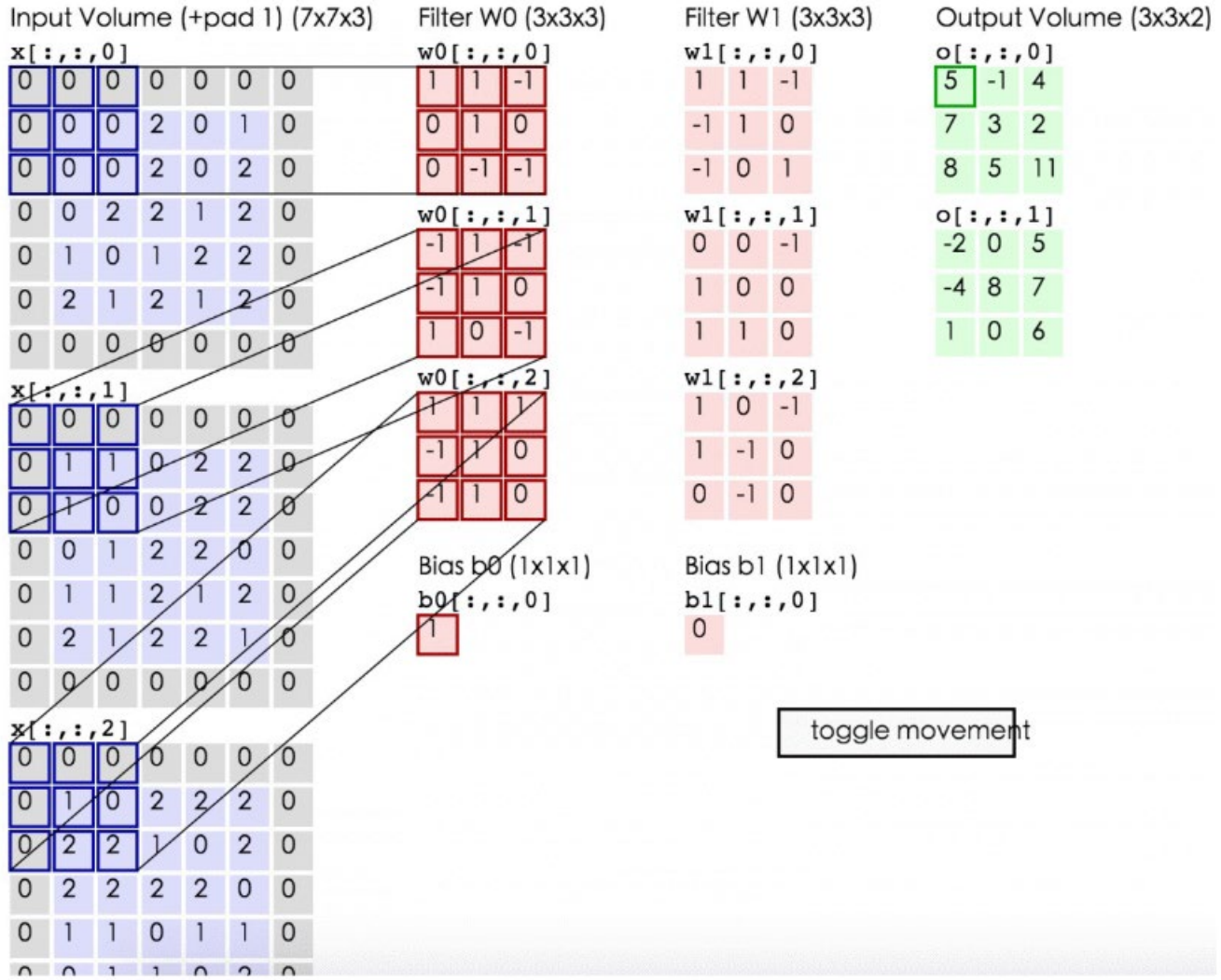
CNN Layer Reconstruction → Save memory/computation resource

Motivation

$$X \in \mathbb{R}^{H \times W \times c} \xrightarrow{F \in \mathbb{R}^{s_1 \times s_2 \times c \times d}} Y \in \mathbb{R}^{H' \times W' \times d}$$

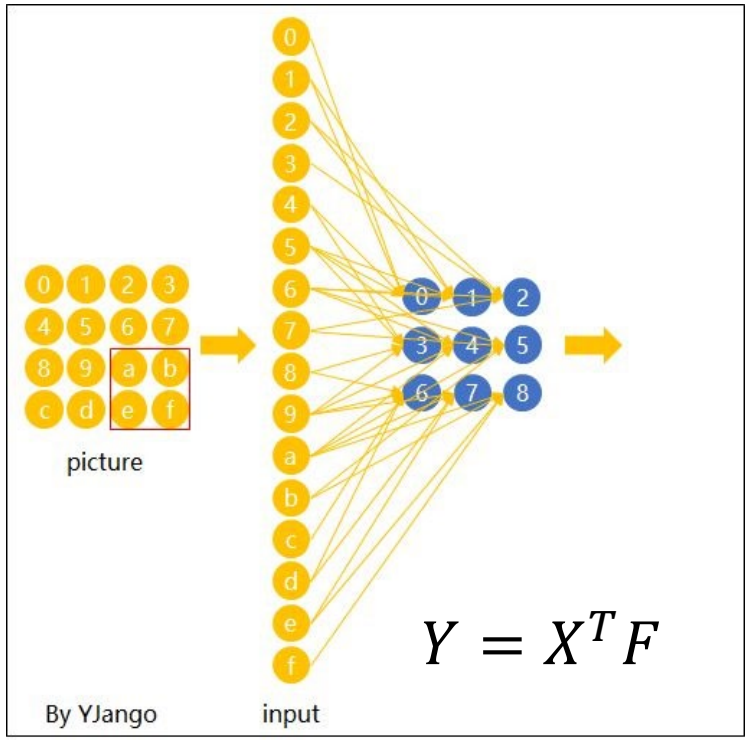
d is the number of convolution filters in this layer and d is much larger than $s = s_1 \times s_2$.





$X \in \mathbb{R}^{H \times W \times c}$ $\quad X = [\mathrm{vec}(X_1), \ldots, \mathrm{vec}(X_c)] \in \mathbb{R}^{HW \times c}$

$Y \in \mathbb{R}^{H' \times W' \times d}$ $\quad Y = [\mathrm{vec}(Y_1), \ldots, \mathrm{vec}(Y_d)] \in \mathbb{R}^{H'W' \times d}$



Model

Feature maps from different filters should be independent of each other as far as possible

$$\theta(Y) = \|Y^T Y \circ (1 - I)\|_F^2$$

$\tilde{Y} = \phi(Y) = YP^T \quad P \in R^{d \times d}$ is the projection matrix

$$\min_{P,C} \|PY^TYP^T - C\|_F^2, s.t.\ C = \mathrm{diag}(c)$$

An optimal projection should thus not only remove redundancy between feature maps, but also preserve the discriminability of these features.

$$\min_{P,C} \|PY^TYP^T - C\|_F^2 + \lambda \left\| D\left(\tilde{Y}\right) - D(Y) \right\|$$
$$s.t. \tilde{Y} = YP^T, C = \mathrm{diag}(c)$$

D_{ij} is the Euclidean distance between the i-th column and the j-th column of Y

Distances between feature maps can be easily preserved if P is orthogonal, i.e. $PP^T = I$

$$\|y_1P^T\|_2 = \|y_1\|_2 \quad \|y_1P^T - y_2P^T\|_2^2 = \|y_1 - y_2\|_2^2$$

$$\min_{P,c} \|PY^TYP^T - C\|_F^2 \ s.t. C = \mathrm{diag}(c), PP^T = I$$

The orthogonal transformation P learned is able to extract the intrinsic representation and preserve distances between feature maps, but the dimensionality has not been reduced since P is a square matrix.

Propose to use a sparse matrix to approximate the representation generated by P.

$$\min_{\tilde{Y}} \left\|\tilde{Y} - YP^T\right\|_F^2 + \lambda\left\|\tilde{Y}\right\|_{2,1} \quad \left\|\tilde{Y}\right\|_{2,1} = \sum_i \left\|\tilde{Y}^i\right\| \qquad \tilde{Y}^i \text{ is the } i\text{-th column in } \tilde{Y}$$

$$\tilde{Y}^i = \begin{cases} \dfrac{\|u_i\| - \lambda}{\|u_i\|} u_i & if\ \lambda < \|u_i\| \\ 0 & otherwise \end{cases} \qquad u_i = YP_i^T$$

Robust Subspace Segmentation by Low-Rank Representation(ICML2010)

$$\min_{P,C} \|PY^TYP^T - C\|_F^2 + \beta\|c\|_1$$
$$s.t. C = \mathrm{diag}(c), PP^T = I$$

Some small valued columns in $\tilde{Y} = YP^T$ will be discarded.

$$\min_{P,C} \|PY^TYP^T - C\|_F^2 + \alpha\|PP^T - I\|_F^2 + \beta\|c\|_1$$
$$s.t. P = \mathrm{circ}(p), C = \mathrm{diag}(c)$$

$$p = (p_1, \dots, p_d)^T$$

$$\mathrm{circ}(p) := \begin{bmatrix} p_1 & p_d & \cdots & p_3 & p_2 \\ p_2 & p_1 & p_d & & p_3 \\ \vdots & p_2 & p_1 & \ddots & \vdots \\ & & \ddots & \ddots & \\ p_{d-1} & & \ddots & \ddots & p_d \\ p_d & p_{d-1} & \cdots & p_2 & p_1 \end{bmatrix}$$

…, circulant matrices have complex internal structures and strong diversity thus can be utilized for approximating huge matrices

An exploration of parameter redundancy in deep networks with circulant projections(CVPR2015)

$$\min_{P,c}\|PY^TYP^T - C\|_F^2 + \alpha\|PP^T - I\|_F^2 + \beta\|c\|_1$$
$$s.t.\, P = \mathrm{circ}(p), C = \mathrm{diag}(c)$$

Solve c for a fixed projection P

$$\min_{P,c}\|PY^TYP^T - C\|_F^2 + \beta\|c\|_1$$
$$s.t.\, C = \mathrm{diag}(c)$$

$$\tilde{c} = \mathrm{diag}(PY^TYP^T)$$

$$\min_{c}\|\tilde{c} - c\|_2^2 + \beta\|c\|_1 \qquad c = \mathrm{sign}(\tilde{c}) \circ \max\{|\tilde{c}| - \beta/2, 0\}$$

机器学习11.4

Solve P for a fixed projection c

循环矩阵

$$\mathbf{x} = [x_0, x_1, x_2, \cdots, x_{N-1}]$$

$$\mathbf{X} = C(\mathbf{x}) = \begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_{N-1} \\ x_{N-1} & x_0 & x_1 & \cdots & x_{N-2} \\ x_{N-2} & x_{N-1} & x_0 & \cdots & x_{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1 & x_2 & x_3 & \cdots & x_0 \end{bmatrix}$$

$$\mathbf{X} = C(\mathbf{x}) = \mathbf{F} \cdot diag\left(\mathcal{F}(\mathbf{x})\right) \cdot \mathbf{F}^{\mathrm{H}}$$

$$\hat{x} = \mathcal{F}(x) = Fx$$

$$w = e^{-j2\pi/N}$$

$$\mathbf{F} = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{N-1} \\ 1 & w^2 & w^4 & \cdots & w^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & w^{N-1} & w^{2(N-1)} & \cdots & w^{(N-1)^2} \end{bmatrix}$$

$$\mathbf{F}^{\mathrm{H}}\mathbf{F} = \mathbf{F}\mathbf{F}^{\mathrm{H}} = \mathbf{I}$$

循环矩阵

$$\mathbf{A}+\mathbf{B}=C(\mathbf{a}+\mathbf{b})$$

$$\mathbf{A}\cdot\mathbf{B}=C\left(\mathcal{F}^{-1}\left(\mathcal{F}(\mathbf{a})\odot\mathcal{F}(\mathbf{b})\right)\right)$$

$$\mathbf{X}=C(\mathbf{x})=\begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_{N-1} \\ x_{N-1} & x_0 & x_1 & \cdots & x_{N-2} \\ x_{N-2} & x_{N-1} & x_0 & \cdots & x_{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_1 & x_2 & x_3 & \cdots & x_0 \end{bmatrix}$$

$$\mathbf{X}^{\mathrm{T}}=\mathbf{F}\cdot diag\left(\mathcal{F}^{*}(\mathbf{x})\right)\cdot\mathbf{F}^{\mathrm{H}}$$

$$\mathbf{X}^{-1}=\mathbf{F}\cdot diag(\mathcal{F}(\mathbf{x}))^{-1}\cdot\mathbf{F}^{\mathrm{H}}$$

$$\mathcal{F}(\mathbf{X}\mathbf{y})=\mathcal{F}^{*}(\mathbf{x})\odot\mathcal{F}(\mathbf{y})$$

$$\min_P \|PY^TYP^T - C\|_F^2 + \alpha\|PP^T - I\|_F^2$$
$$s.t. P = \mathrm{circ}(p)$$

Since P and its relevant terms can be efficient calculated in the frequency domain, transfer the into the Fourier domain for having a higher training speed.

$$\min_P L(P) = L_1(P) + \alpha L_2(P), s.t. P = \mathrm{circ}(p)$$

矩阵分析与应用（第2版）
3.矩阵微分

$$\frac{\partial L_1(P)}{\partial P} = 4HPP^THP - 4HPC = \frac{4}{d^2}HS^H\hat{P}\hat{P}^TSHS^H\hat{P}S - \frac{4}{d}HS^H\hat{P}S^HC \qquad \hat{P} = \mathrm{diag}(\hat{p})$$

$$\frac{\partial L_1(P)}{\partial \hat{P}} == \frac{1}{d}S\frac{\partial L_1(P)}{\partial P}S^H = \frac{4}{d^2}\left(\hat{H}\hat{P}\hat{P}^H\hat{H}\hat{P} - \hat{H}\hat{P}\hat{C}\right) \quad \Rightarrow \quad \frac{\partial L_1(P)}{\partial \hat{p}} = diag\left(\frac{\partial L_1(P)}{\partial \hat{P}}\right)$$

$$\hat{H} = SHS^H \quad \hat{C} = SCS^H$$

$$\min_{P,c} \|PY^TYP^T - C\|_F^2 + \alpha\|PP^T - I\|_F^2 + \beta\|c\|_1$$
$$s.t. P = \text{circ}(p), C = \text{diag}(c)$$

$$L_2(P) = \|PP^T - I\|_F^2 = \|\frac{1}{d}S^H\hat{P}^H\hat{P}S - \frac{1}{d}S^HIS\|_F^2$$
$$= \text{Tr}\left[\frac{1}{d}S^H\left(\hat{P}^H\hat{P} - I\right)^H\left(\hat{P}^H\hat{P} - I\right)S\right]$$
$$= \text{Tr}\left[\left(\hat{P}^H\hat{P} - I\right)^H\left(\hat{P}^H\hat{P} - I\right)\right] = \|\hat{p}^H \circ \hat{p} - 1\|_2^2$$

$$\frac{\partial L_2(P)}{\partial \hat{p}} = 4(\hat{p}\hat{p}^H \circ I)\hat{p} - 4\hat{p}$$

$$\hat{p} = \hat{p} - \eta\left(\frac{\partial L_1(P)}{\partial \hat{p}} + \alpha\frac{\partial L_2(P)}{\partial \hat{p}}\right)$$

$$P = \frac{1}{d}\text{diag}(\bar{c})S^H\hat{P}S$$

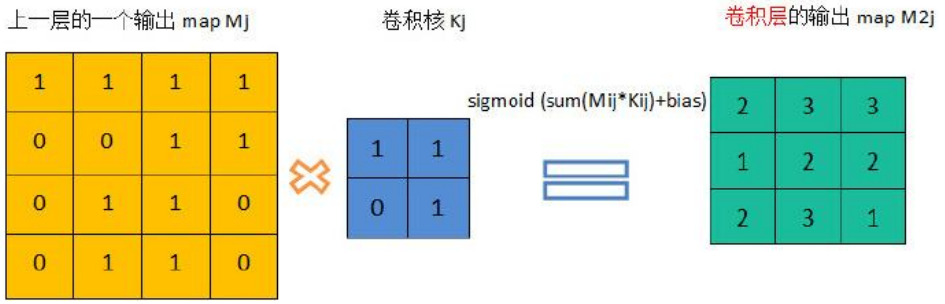
$$\bar{c}_i = 0, if\ c_i = 0\ and\ \bar{c}_i = 1\ otherwise$$

$$\tilde{Y} = YP_{\tilde{d}}^T = X^TFP_{\tilde{d}}^T = X^T\tilde{F}$$

CNN Layer Reconstruction

$$\min_{\tilde{F}} \left\| \tilde{Y} - \tilde{X}^T \tilde{F} \right\|_F^2 + \gamma \left\| \tilde{F} \right\|_F^2$$

$$\tilde{F} = \left(\tilde{X}\tilde{X}^T + \gamma \mathrm{I} \right)^{-1} \tilde{X}\tilde{Y}$$



$$L\left(\tilde{F}\right) = \mathrm{Tr}\left(\tilde{F}^T \tilde{X}^T \tilde{X} \tilde{F}\right) - 2Tr\left(\tilde{F}^T \tilde{X}^T \tilde{Y}\right) + \gamma Tr\left(\tilde{F}^T \tilde{F}\right)$$

$$\tilde{F} = \tilde{F} - \eta \frac{\partial L(\tilde{F})}{\partial \tilde{F}}$$

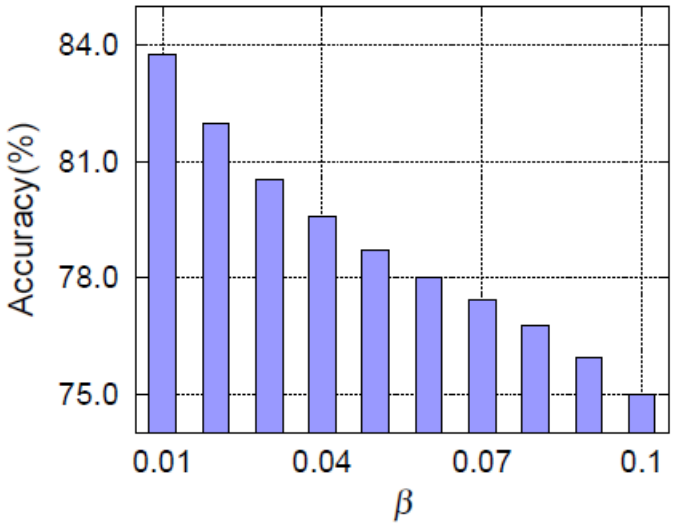
Experiments

Speed-up ratio

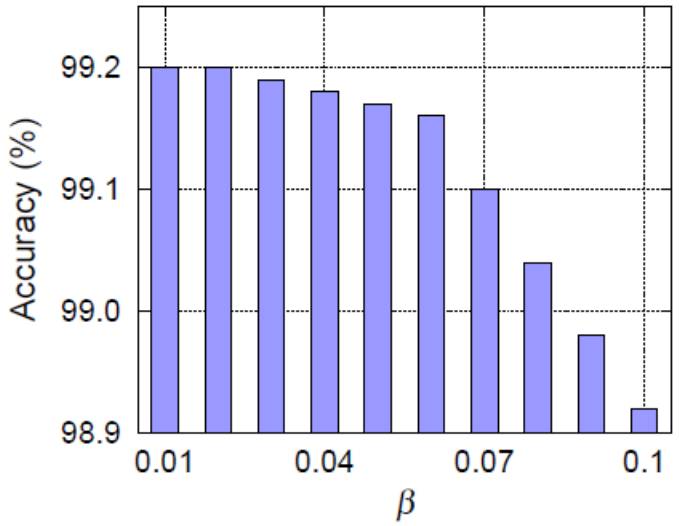
$$r_s = \frac{s_i^2 d_{i-1} d_i H_i' W_i'}{s_i^2 \tilde{d}_{i-1} \tilde{d}_i H_i' W_i'} = \frac{d_{i-1} d_i}{\tilde{d}_{i-1} \tilde{d}_i}$$

Compression ratio

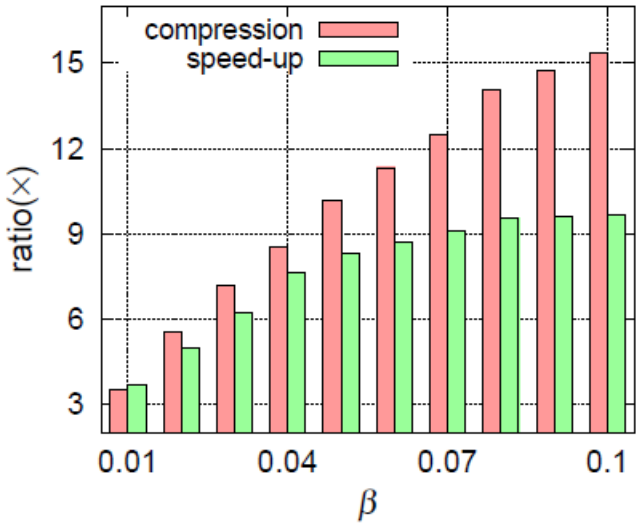
$$r_c = \frac{s_i^2 d_{i-1} d_i + d_i H_i' W_i'}{s_i^2 \tilde{d}_{i-1} \tilde{d}_i + \tilde{d}_i H_i' W_i'}$$



(a) Accuracy of the reconstructed model.



(b) Model accuracy after fine-tuning.



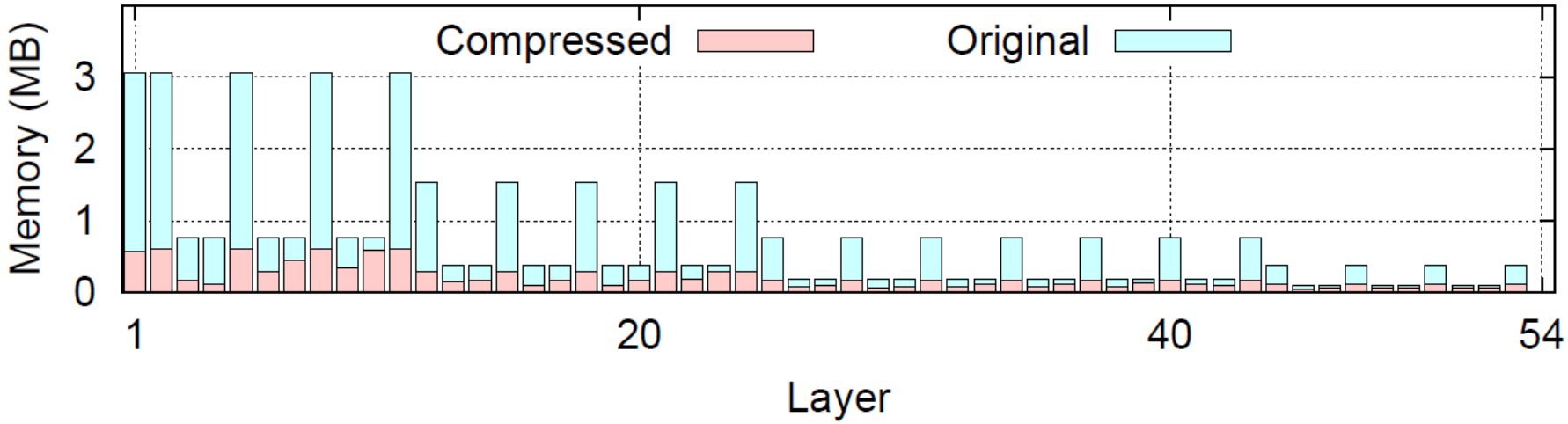
(c) Compression and speed-up ratios.

Table 1. Compression statistics for AlexNet.

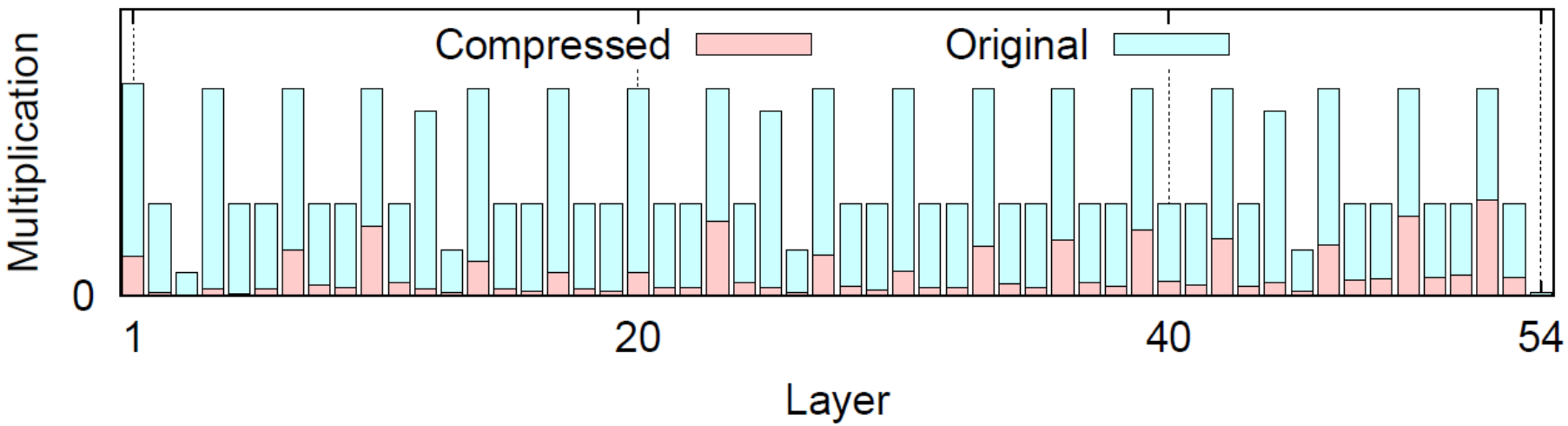
Layer	Memory	r_c	Mult.	r_s
conv1	1.24MB	2.4×	1.05×10^8	2.4×
conv2	1.88MB	10.0×	2.23×10^8	15.4×
conv3	3.62MB	15.5×	1.49×10^8	21.2×
conv4	2.78MB	3.0×	1.12×10^8	3.6×
conv5	1.85MB	1.2×	0.74×10^8	1.2×
fc6	144MB	3.5×	0.37×10^8	3.5×
fc7	64.02MB	72.0×	0.16×10^8	72.1×
fc8	15.63MB	23.4×	0.04×10^8	23.5×
Total	235.03MB	5.06×	7.24×10^8	4.31×

Table 2. Compression statistics for VGG-16 Net.

Layer	Memory	r_c	Mult.	r_s
conv1_1	12.26MB	10.7×	0.11×10^9	10.7×
conv1_2	12.39MB	1.8×	2.41×10^9	19.0×
conv2_1	6.41MB	3.3×	1.20×10^9	5.7×
conv2_2	6.69MB	4.0×	2.41×10^9	12.0×
conv3_1	4.19MB	7.2×	1.20×10^9	21.9×
conv3_2	5.31MB	2.7×	2.41×10^9	10.2×
conv3_3	5.31MB	2.9×	2.41×10^9	4.2×
conv4_1	6.03MB	41.3×	1.20×10^9	56.2×
conv4_2	10.53MB	16.5×	2.41×10^9	70.1×
conv4_3	10.53MB	3.9×	2.41×10^9	5.1×
conv5_1	9.38MB	7.8×	0.60×10^9	8.0×
conv5_2	9.38MB	18.5×	0.60×10^9	21.3×
conv5_3	9.38MB	23.4×	0.60×10^9	26.7×
fc6	392MB	8.6×	0.41×10^9	8.6×
fc7	64MB	3.3×	0.16×10^8	3.3×
fc8	15.62MB	2.2×	0.41×10^7	2.2×
Total	579.46MB	6.19×	2.04×10^{10}	9.63×



(b) Compression ratio of feature maps of convolutional layers (r_{c_2}).



(c) Speed-up ratio of all convolutional layers (r_s).

Input: A pre-trained convolutional neural network $\mathcal{N}$ learned through a dataset $\mathcal{X}$ with k convolutional layers: $\mathcal{L}_1, ..., \mathcal{L}_k$, weight parameter γ, learning rate η.

Calculate feature maps of each layer by using the original network: $\{\mathbf{Y}_1, ...\mathbf{Y}_k\} \leftarrow \mathcal{N}(\mathcal{X})$;
for $i = 1$ to $k - 1$ **do**
  Learn the projection P_i by solving Fcn. 9;
  Calculate new feature maps: $\tilde{\mathbf{Y}}_i \leftarrow \mathbf{Y}_i P_i^T$;
end for
Keep feature maps of the k-th layer: $\tilde{\mathbf{Y}}_k \leftarrow \mathbf{Y}_k$;
Construct a new network $\tilde{\mathcal{N}}$ according to $\{\tilde{\mathbf{Y}}_1, ...\tilde{\mathbf{Y}}_k\}$ and initialize convolution filters $\{\tilde{\mathbf{F}}_1, ...\tilde{\mathbf{F}}_k\}$ by random values from the standard normal distribution;
repeat
  Randomly select a batch $\mathcal{X}_j$ from $\mathcal{X}$;
  for $i = 1$ to k **do**
    Generate input data $\tilde{\mathbf{X}}_i$ of $\mathcal{L}_i$ exploiting $\tilde{\mathcal{N}}$;
    Estimate the new filter matrix (Fcn. 20): $\tilde{\mathbf{F}}_i \leftarrow \tilde{\mathbf{F}}_i - \eta \partial L(\tilde{\mathbf{F}}_i)/\partial \tilde{\mathbf{F}}_i$;
    Convert $\tilde{\mathbf{F}}_i$ into filter data and fill it in $\tilde{\mathcal{N}}$;
  end for
until Convergence;

Output: The new convolutional neural network $\tilde{\mathcal{N}}$.

It is worth mentioning that input data of the first layer of the original network N and feature map of the last layer (closely related to the number of classification labels) are kept unchanged.

As for other intermediate convolutional layers and fully connected layers, we can generate compact feature maps $\tilde{Y}$ from the original feature maps Y.

Then, calculate the input data $\tilde{X}$ using the compressed network $\tilde{N}$ and estimate the filter matrix $\tilde{N}$.