From Hashing to CNNs: Training Binary Weight Networks via Hashing

2018-10-21

Overview

- 2018

- 本文揭示了保持内积哈希与BWN之间的紧密关联。为了减轻用哈希方法所带来的loss，本文将binary codes乘以了一个scaling factor并用交替优化的策略来更新binary codes以及factor。在Cifar10,Cifar100以及ImageNet上实验，本文提出的BWNH方法比之前方法要好。

- 参考博客：https://blog.csdn.net/ajj15120321/article/details/80571748

Our Method

Inner-product preserving hashing

Given two sets of points $\mathbf{X} \in \mathbb{R}^{S\times M}$ and $\mathbf{W} \in \mathbb{R}^{S\times N}$ where $\mathbf{X_i} \in \mathbb{R}^{S\times 1}$ and $\mathbf{W_i} \in \mathbb{R}^{S\times 1}$ represents i^{th} point of $\mathbf{X}$ and $\mathbf{W}$ respectively, we denote the inner-product similarity of $\mathbf{X}$ and $\mathbf{W}$ as $\mathbf{S} \in \mathbb{R}^{M\times N}$. (Shen et al. 2015a) proposed the inner-product preserving hashing by solving the following objective function:

$$\min \ \|\mathbf{S} - h(\mathbf{X})^{\mathrm{T}} g(\mathbf{W})\|_F^2 \tag{1}$$

where $h(\cdot)$ and $g(\cdot)$ are hash functions for $\mathbf{X}$ and $\mathbf{W}$ respectively.

Connection between hashing and binary weights

$$\begin{aligned} \min \ & L(\mathbf{B}) = \|\mathbf{W} - \mathbf{B}\|_F^2 \\ s.t. \ & \mathbf{B} \in \{+1, -1\}^{S\times N} \end{aligned} \tag{2}$$

$$\begin{aligned} \min \ & L(\mathbf{B}) = \|\mathbf{X}^{\mathrm{T}}\mathbf{W} - \mathbf{X}^{\mathrm{T}}\mathbf{B}\|_F^2 \\ s.t. \ & \mathbf{B} \in \{+1, -1\}^{S\times N} \end{aligned} \tag{3}$$

2 3区别是，由尽量减少WB之间的差距改为减少内积之间的差距

However, solving Eq. (3) still can cause somewhat accuracy drops. Inspired by (Rastegari et al. 2016), we multiply a scaling factor to each hashing codes $\mathbf{B_i}$:

$$g(\mathbf{W}) = \mathbf{BA} \tag{4}$$

where $\mathbf{A}$ is a diagonal matrix and $\alpha_i = \mathbf{A_{ii}}$ is the scaling factor for $\mathbf{B_i}$. Finally, our objective function is:

$$\begin{aligned} \min \ L(\mathbf{A}, \mathbf{B}) &= \|\mathbf{S} - \mathbf{X}^{\mathrm{T}}\mathbf{BA}\|_F^2 \\ &= \sum_i^N \|\mathbf{S_i} - \alpha_i \cdot \mathbf{X}^{\mathrm{T}}\mathbf{B_i}\|_F^2 \end{aligned} \tag{5}$$

where $\mathbf{S} = \mathbf{X}^{\mathrm{T}}\mathbf{W}$ and $\mathbf{S_i} \in \mathbb{R}^{M\times 1}$ is i^{th} column vector of $\mathbf{S}$. The Eq. (5) can be easily divided into N independent sub-problems:

$$\begin{aligned} \min \ & L_i(a_i, \mathbf{B_i}) = \|\mathbf{S_i} - \alpha_i \cdot \mathbf{X}^{\mathrm{T}}\mathbf{B_i}\|_F^2 \\ s.t. \ & \mathbf{B_i} \in \{+1, -1\}^{S\times 1} \end{aligned} \tag{6}$$

本文加入了一个B的Scaling Factor

Our Method

之后是如何对alpha和B进行求解，固定一个求另一个；之后消除层与层之间的量化误差累积；

和two-step思路差不多

$$\begin{aligned}\min L(\mathbf{A},\mathbf{B}) &= \|(\mathbf{X}^{l+1})^{\mathrm{T}}\mathbf{W}^{l+1} - (\tilde{\mathbf{X}}^{l+1})^{\mathrm{T}}\mathbf{B}^{l+1}\mathbf{A}^{l+1}\|_F^2 \\ &= \|\mathbf{S}^{l+1} - (\tilde{\mathbf{X}}^{l+1})^{\mathrm{T}}\mathbf{B}^{l+1}\mathbf{A}^{l+1}\|_F^2\end{aligned} \tag{13}$$

消除累积误差

Initialization of $\mathbf{B_i}$ and α_i At the beginning of alternating optimization method, we initialize $\mathbf{B_i}$ with $sign(\mathbf{W_i})$. For α_i, we take the average $L1$ norm of $\mathbf{W_i}$ as initialization.

Update α_i with $\mathbf{B_i}$ fixed By expanding Eq.(6), we have

$$\min\ L_i(\alpha_\mathbf{i}) = const + \alpha_i^2\|\mathbf{X}^{\mathrm{T}}\mathbf{B_i}\|_F^2 - 2\alpha_i\mathbf{S_i}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{B_i}. \tag{7}$$

Then the derivative of $L_i(\alpha_\mathbf{i}, \mathbf{B_i})$ w.r.t α_i is:

$$\frac{\partial L_i(\alpha_\mathbf{i})}{\partial \alpha_i} = 2\alpha_i\|\mathbf{X}^{\mathrm{T}}\mathbf{B_i}\|_F^2 - 2\mathbf{S_i}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{B_i} \tag{8}$$

By setting it to zero, we get the solution of α_i:

$$\alpha_i = \frac{\mathbf{S_i}^{\mathrm{T}}\mathbf{X}^{\mathrm{T}}\mathbf{B_i}}{\|\mathbf{X}^{\mathrm{T}}\mathbf{B_i}\|_F^2} \tag{9}$$

Solving $\mathbf{B_i}$ with α_i fixed By expanding Equation (6), we can get:

$$\begin{aligned}\min\ & L_i(\mathbf{B_i}) = const + \|\mathbf{Z}^{\mathrm{T}}\mathbf{B_i}\|_F^2 - 2\,\mathrm{Tr}\,(\mathbf{B_i}^{\mathrm{T}}\mathbf{q}) \\ s.t.\quad & \mathbf{B_i} \in \{+1,-1\}^{S\times 1}\end{aligned} \tag{10}$$

where $\mathbf{Z} = \alpha \cdot \mathbf{X}$,$\mathrm{Tr}()$ is the trace norm, and $\mathbf{q} = \alpha \cdot \mathbf{XS_i}$. Eq. (10) can be solved by *discrete cyclic coordinate descent* (DCC) method which is proposed in (Shen et al. 2015b) for solving hashing codes. Let b be the j^{th} element of $\mathbf{B_i}$, and $\mathbf{B_i}'$ the column vector of $\mathbf{B_i}$ excluding b. Similarly we denote the j^{th} element of $\mathbf{q}$ as $\mathbf{q_j}$, and let $\mathbf{q}'$ as the $\mathbf{q}$ excluding $\mathbf{q_j}$. Let $\mathbf{v}^{\mathrm{T}}$ be the j^{th} row of matrix $\mathbf{Z}$ and $\mathbf{Z}'$ be matrix $\mathbf{Z}$ excluding $\mathbf{v}^{\mathrm{T}}$. Then problem (10) can be written as:

$$\begin{aligned}\min\ & (\mathbf{B_i}'^{\mathrm{T}}\mathbf{Z}'\mathbf{v} - \mathbf{q_j})b \\ s.t.\quad & b \in \{+1,-1\}\end{aligned} \tag{11}$$

Then we can get the solution for the j^{th} element of $\mathbf{B_i}$:

$$b = sign(\mathbf{q_j} - \mathbf{B_i}'^{\mathrm{T}}\mathbf{Z}'\mathbf{v}) \tag{12}$$

By using this method, each element of $\mathbf{B_i}$ can be iteratively updated with other $S-1$ elements of $\mathbf{B_i}$ fixed.

Our Method

Algorithm 1: Training Binary weight Convolutional Neural Networks via Hashing

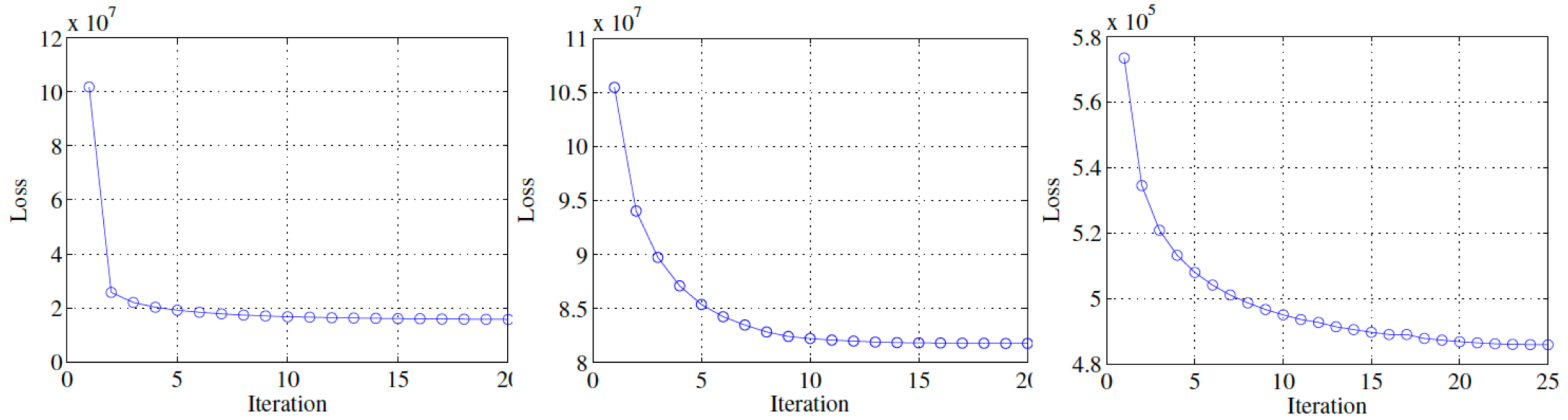
Input: Pre-trained convolutional neural networks weights $\{\mathbf{W}^l\}_{l=1}^{L}$ and Max_Iter

Output: Learned binary weights $\{\mathbf{B}^l\}_{l=1}^{L}$ and scaling factors $\{\mathbf{A}^l\}_{l=1}^{L}$

for $l = 1; l \leq L$ **do**
  Sampling a mini-batch images from database
  Forward propagation to get $\tilde{\mathbf{X}}^l$ and $\mathbf{X}^l$
  Calculate S with $\tilde{\mathbf{X}}^l$
  for $i = 1; i \leq N$ **do**
    Initialize $\mathbf{B_i}$ with $sign(\mathbf{W_i})$
    Initialize α_i with mean $L1$ norm of $\mathbf{W_i}$
    while *iter* $\leq$ *Max_Iters* **do**
      Update α_i with Eq.(9)
      for $j = 1; j \leq S$ **do**
        Update j^{th} element of $\mathbf{B_i}$ with Eq.(12)
      end
    end
  end
end
for $l = 1; l \leq L$ **do**
  Initialize l^{th} layer of binarized CNN model with $\mathbf{B}^l$
  Add a scale layer right after the l^{th} layer
  Initialize weights of the scale layer with $\mathbf{A}^l$
end
Fine-tune the binarized CNN model
return $\{\mathbf{B}^l\}_{l=1}^{L}$ and $\{\mathbf{A}^l\}_{l=1}^{L}$;

Experiments

Figure 1: The optimization loss vs Iteration using the proposed alternating optimization method on different networks



(a) optimization loss of VGG9 on CIFAR10 (b) optimization loss of AlexNet (c) optimization loss of ResNet-18

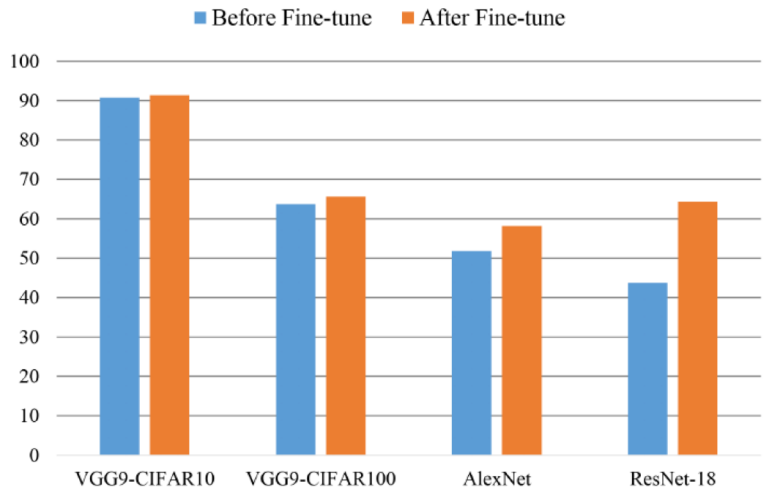


Figure 2: Top1 Accuracy of VGG9, AlexNet, and ResNet-18 with or without fine-tuning

Table 1: Test error rate of VGG9 on CIFAR10 and CIFAR100

Method	Test error rate	
	CIFAR10	CIFAR100
Full-Precision	9.01	30.45
BinaryConnect	11.15	37.70
BWN	10.67	37.68
SQ-BWN	9.40	35.25
BWNH (Ours)	**9.21**	**34.35**

Table 2: Classification Accuracy of AlexNet for different methods

Method	Classification Accuracy	
	Top1	Top5
BinaryConnect	35.4	61.0
BWN	56.8	79.4
SQ-BWN	51.2	75.1
HWGQ-BWN	52.4	75.9
BWNH (Ours)	**58.5**	**80.9**

Table 3: Classification Accuracy of ResNet-18 for different methods

Method	Classification Accuracy	
	Top1	Top5
Full-Precision	69.3	89.2
BWN	60.8	83.0
SQ-BWN	58.3	81.6
HWGQ-BWN	61.3	83.9
BWNH (Ours)	**64.3**	**85.9**

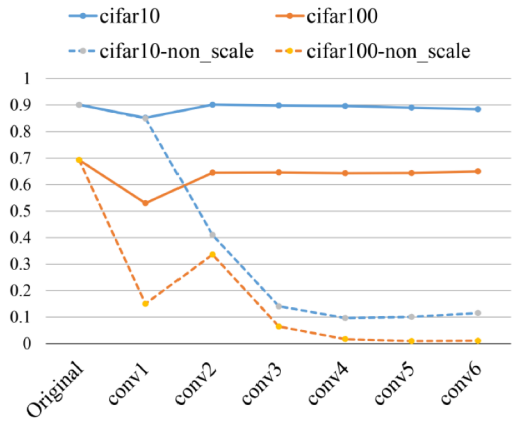


Figure 3: Accuracy of VGG9 Network on CIFAR10 and CIFAR100 after layer-wise optimization from *conv1* to *conv6*