Machine Learning course
advanced track

Practice 14: Model Compression

Radoslav Neychev
Iurii Efimov

MIPT
07.12.2019, Moscow, Russia

References

These slides are mostly based on paper [“A Survey on Methods and Theories of Quantized Neural Networks” by Yunhui Guo]

Model Compression Methods

- Weight Pruning
- Low-rank Approximation
- Knowledge Distillation
- Quantization

Model Compression Methods

- ~~Weight Pruning~~
- ~~Low-rank Approximation~~
- ~~Knowledge Distillation~~ (whiteboard driven)
- Quantization

State-of-the-art CNN Architectures

	# of parameters	Layers	flops	Top-1 error rate on ImageNet
AlexNet	60M	8	725M	43.45%
VGGNet-16 (BN)	138M	16	15484M	26.63%
GoogleNet	6.9M	22	1566M	31.30%
ResNet-152	60.2M	152	11300M	22.16%

State-o

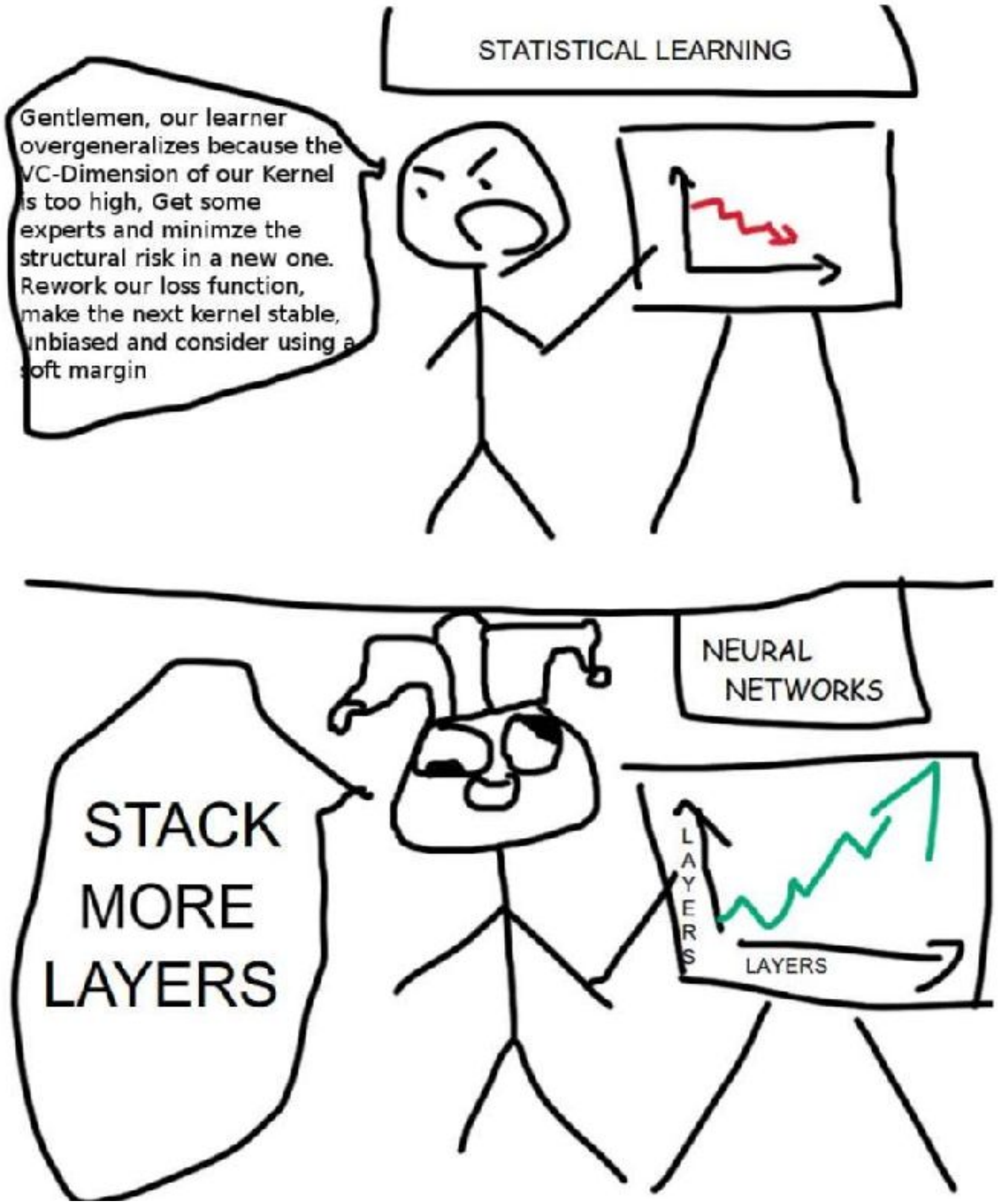
ImageNet

AlexNet

VGGNet-
(BN)

GoogleN

ResNet-1



Motivation

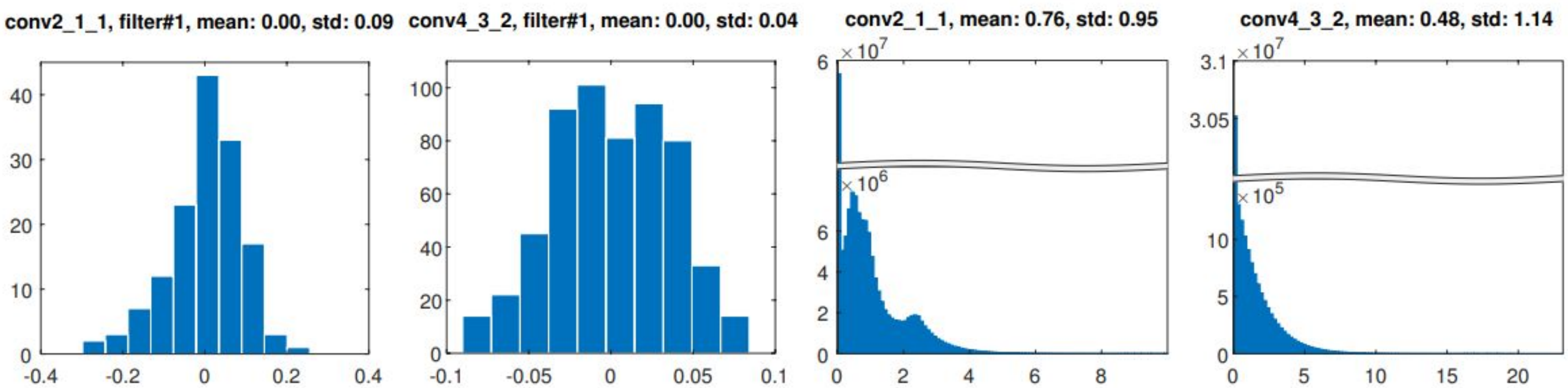


Fig. 1: Distributions of weights (left two columns) and activations (right two columns) at different layers of the ResNet-20 network trained on CIFAR-10. All the test-set images are used to get the activation statistics.

[1] LQ-Nets: Learned Quantization for Highly Accurate and Compact DNNs, https://arxiv.org/pdf/1807.10029.pdf

Motivation

- DL in low-power devices: smartphones, etc.
- Reduce memory consumption
- Reduce processing time
- Reduce energy consumption
- Reduce training time

Quantization

- Make model more compact without performance degradation
- Use fixed-point operations instead of floating-point
- Given a,b - binary vectors, a dot product is calculated as:

$$a\dot{b} = bitcount(a \ \ and \ b)$$

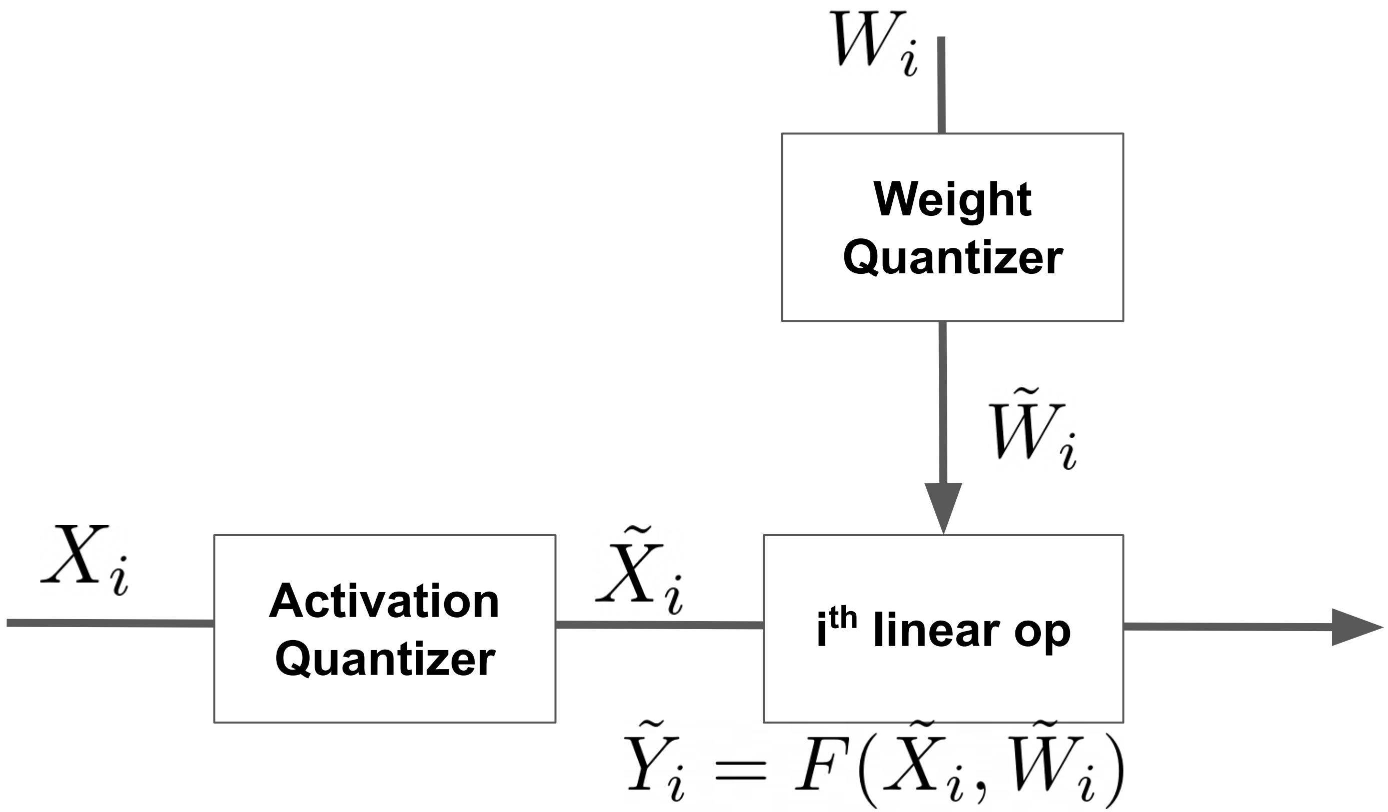
Quantization

- **How?**
  - Deterministic: Discrete mapping between floating-point and fixed-point values
  - Stochastic: quantized values are sampled from discrete distributions
- **What?**
  - Weights
  - Activations
  - Gradients
- **When?**
  - Training phase
  - Post-training phase

Quantization

- **How?**
  - Deterministic: Discrete mapping between floating-point and fixed-point values
  - ~~Stochastic: quantized values are sampled from discrete distributions~~ (not today)
- **What?**
  - Weights
  - Activations
  - ~~Gradients~~ (not today)
- **When?**
  - Training phase
  - ~~Post-training phase~~ (not today)

- **Pipeline**



W_i
Weight Quantizer
$\tilde{W}_i$
X_i
Activation Quantizer
$\tilde{X}_i$
i^{th} linear op
$\tilde{Y}_i = F(\tilde{X}_i, \tilde{W}_i)$

- **Forward pass:**[1]

$$x^b = sign(x) = \begin{cases} +1, & \text{if } x \geq 1 \\ 0, & \text{otherwise} \end{cases}$$

- **Backward pass: gradients are 0 almost everywhere? Solution: Straight Through Estimator (STE)**[2]

A simple example is the STE defined for Bernoulli sampling with probability $p \in [0, 1]$:

$$\textbf{Forward: } q \sim Bernoulli(p)$$

$$\textbf{Backward: } \frac{\partial c}{\partial p} = \frac{\partial c}{\partial q}.$$

[1] BinaryConnect: Training Deep Neural Networks with binary weights during propagations, https://arxiv.org/abs/1511.00363
[2] Deep Neural Networks for Acoustic Modeling in Speech Recognition, https://static.googleusercontent.com/media/research.google.com/ru//pubs/archive/38131.pdf

- **Basic approach[1]: BinaryConnect**

$$\textbf{Forward:} \quad x^b = \mathrm{Sign}(x)$$

$$\textbf{Backward:} \quad \frac{\partial E}{\partial x} = \frac{\partial E}{\partial x^b} \mathrm{I}_{|x| \leq 1}$$

where $\mathrm{I}_{|x| \leq 1}$ is an indicator function defined as,

$$\mathrm{I}_{|x| \leq 1} = \begin{cases} 1 & |x| \leq 1, \\ 0 & \text{otherwise} \end{cases}$$

[1] BinaryConnect: Training Deep Neural Networks with binary weights during propagations, https://arxiv.org/abs/1511.00363

- **Improved approach[1]: XNOR-Net**

$$\textbf{Forward:} \quad x^b = \mathrm{Sign}(x) \times \mathrm{E}_F(|x|)$$

$$\textbf{Backward:} \quad \frac{\partial E}{\partial x} = \frac{\partial E}{\partial x^b}$$

where $EF(|x|)$ is the mean of absolute weight values of each output channel

[1] XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks, https://arxiv.org/abs/1603.05279

- **General scheme:**

$$Q(x) = sc^{-1}(x)(\hat{Q}(sc(x)))$$

$sc(x) : \mathbb{R} \to [0, 1]$ - scaling function

$\hat{Q}(x) : [0, 1] \to \{a_1, .., a_k\}$ - quantization for k quantization levels

Quantization: Rounding

Drawbacks:

- Performance drop may occur due to rounding
- Keep real values during training → memory overhead
- Harder to converge

Quantization: Vector-based

- Cluster the weights and use centroids to replace actual weights

- Given weight matrix

$$W \in R_{m \times n}$$

- can perform k-means clustering

$$min \sum_{i}^{m} \sum_{j}^{n} \sum_{l}^{k} ||w_{ij} - c_k||^2$$

[1] Compressing Deep Convolutional Networks using Vector Quantization, https://arxiv.org/abs/1412.6115

[2] Quantized Convolutional Neural Networks for Mobile Devices, https://arxiv.org/abs/1512.06473

Quantization: Vector-based

Drawbacks:

- Expensive computation (K-means)
- Hard to achieve binary weights
- Hard to train from scratch
- Ignores local information in CNN

Find optimal approximation for real value weights

- Binary weights:

$$J(B) = ||W - \alpha B||^2 \to \min_B$$

$$B^\star = sign(W) \quad \alpha^\star = \frac{1}{n}||W||_{l1}$$

- Ternary weights:[2]

$$\alpha^*, W^{t^*} = \begin{cases} \text{argmin}_{\alpha, W^t} J(\alpha, W^t) = \|W - \alpha W^t\|_2^2 \\ \text{s.t.} \quad \alpha \geq 0, W_i^t \in -1, 0, 1, i = 1, 2, ..., n. \end{cases}$$

- Etc.

[1] XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks, https://arxiv.org/abs/1603.05279

[2] Ternary Weights Networks, https://arxiv.org/abs/1605.04711

Quantization: Optimization

Drawbacks:

- Convergence relies on weak assumptions
- Harder to implement
- Some methods require second-order derivatives

- **XNOR-net**

Notation: L is the number of layers in the network. $\mathbf{w}_{t-1}$ is the weight at time $t-1$. b_{t-1} is the bias at time $t-1$. a_k is the activation of layer k. n is the number of elements in a filter. E is the loss function.
Input: a minibatch of data (inputs, labels),a learning rate η_t.
Forward pass:

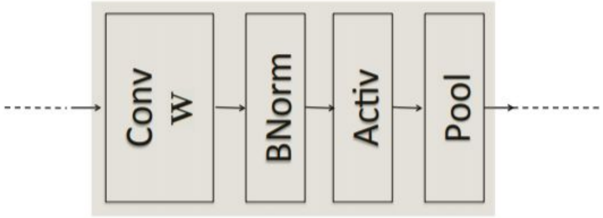
1. for $l = 1$ to L do:
2. for k^{th} filter in l^{th} layer do:
3. $\mathbf{a}^{lk} = \frac{1}{n}\|\mathbf{w}^{lk}_{t-1}\|_{l_1}$
4. $\mathbf{b}^{lk} = \text{Sign}(\mathbf{w}^{lk}_{t-1})$
5. $\mathbf{w}^{lk}_{b} = \mathbf{a}^{lk}\mathbf{b}^{lk}$
6. Compute activations based on binarized filters

Backward pass:

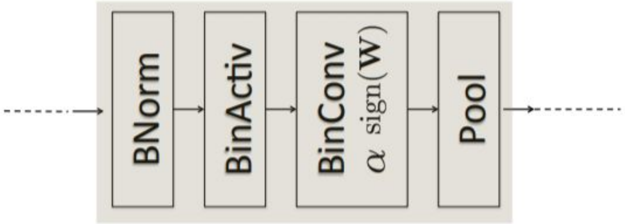
1. Compute backward gradient $\frac{\partial E}{\mathbf{w}_b}$ based on $\mathbf{w}_b$

Parameter update:

1. Update $\mathbf{w}_t$ using $\mathbf{w}_{t-1}$ and $\frac{\partial E}{\mathbf{w}_b}$



A typical block in CNN



A block in XNOR-Net

[1] XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks, https://arxiv.org/abs/1603.05279

- **DoReFa-Net**

A simple example is the STE defined for Bernoulli sampling with probability $p \in [0, 1]$:

$$\textbf{Forward: } q \sim Bernoulli(p)$$

$$\textbf{Backward: } \frac{\partial c}{\partial p} = \frac{\partial c}{\partial q}.$$

An STE we will use extensively in this work is $\mathbf{quantize_k}$ that quantizes a real number input $r_i \in [0, 1]$ to a k-bit number output $r_o \in [0, 1]$. This STE is defined as below:

$$\textbf{Forward: } r_o = \frac{1}{2^k - 1} \operatorname{round}((2^k - 1) r_i) \tag{5}$$

$$\textbf{Backward: } \frac{\partial c}{\partial r_i} = \frac{\partial c}{\partial r_o}. \tag{6}$$

[1] XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks, https://arxiv.org/abs/1603.05279

- **DoReFa-Net**

An STE we will use extensively in this work is $\mathbf{quantize_k}$ that quantizes a real number input $r_i \in [0, 1]$ to a k-bit number output $r_o \in [0, 1]$. This STE is defined as below:

$$\textbf{Forward: } r_o = \frac{1}{2^k - 1} \operatorname{round}((2^k - 1) r_i) \tag{5}$$

$$\textbf{Backward: } \frac{\partial c}{\partial r_i} = \frac{\partial c}{\partial r_o}. \tag{6}$$

In case we use k-bit representation of the weights with $k > 1$, we apply the STE f_ω^k to weights as follows:

$$\textbf{Forward: } r_o = f_\omega^k(r_i) = 2 \operatorname{quantize_k}\left(\frac{\tanh(r_i)}{2\max(|\tanh(r_i)|)} + \frac{1}{2}\right) - 1. \tag{9}$$

$$\textbf{Backward: } \frac{\partial c}{\partial r_i} = \frac{\partial r_o}{\partial r_i} \frac{\partial c}{\partial r_o}\ ^{4} \tag{10}$$

[1] XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks, https://arxiv.org/abs/1603.05279

- **DoReFa-Net: Let's code!**

[1] XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks, https://arxiv.org/abs/1603.05279