Efficient Training of Very Deep Neural Networks for Supervised Hashing

Ziming Zhang 1,2,3  Yuting Chen 1  Venkatesh Saligrama 1

1 Data Science & Machine Learning Lab, College of Engineering, Boston University, Boston, USA
2 School of Automation, Huazhong University of Science and Technology, Wuhan, China
3 MOE Key Laboratory of Image Information Processing and Intelligent Control, Huazhong University of Science and Technology, Wuhan, China

Motivation:
Deep Neural Networks
• Excellent performance
• Slow training speed:
  • VGG [Simonyan and Zisserman, 2014] on ILSVRC2014 with 16 conv layers takes approximately one month using 4 GPUs
• Vanishing gradient problem in back propagation (using the chain rule)
• Can we train DNNs faster?
  • Our network: 64 fully connected layers, 1024 nodes per layer, 1 GPU, training time ~3 hours
  • State-of-the-art for supervised hashing.

Optimization:
• Introduce auxiliary variables:

\[
\min_{\Omega, W} \Omega(\Theta, W) + \sum_i \ell(W^T x_i, y_i)
\]

\[
\min_{\Omega, W} \Omega(\Theta, W) + \sum_i \ell(W^T x_i, y_i) + \beta \sum_i \left\| f_{\Theta}(z_{i,m}) - f_{\Theta}(z_{i,m-1}) \right\|^2_2 + \mu \sum_i \left\| f_{\Theta}(z_{i,m}) - \bar{y}^{(m)} \right\|^2_2
\]

\[
\min_{\Omega, W} \Omega(\Theta, W) + \sum_i \ell(W^T x_i, y_i) + \beta \sum_i \left\| f_{\Theta}(z_{i,m}) - f_{\Theta}(z_{i,m-1}) \right\|^2_2 + \mu \sum_i \left\| f_{\Theta}(z_{i,m}) - \bar{y}^{(m)} \right\|^2_2
\]

• Augmented Lagrangian (or Adaptive Direction Method of Multipliers):

\[
\min_{\Omega, W} \Omega(\Theta, W) + \sum_i \ell(W^T x_i, y_i) + \beta \sum_i \left\| f_{\Theta}(z_{i,m}) - f_{\Theta}(z_{i,m-1}) \right\|^2_2 + \mu \sum_i \left\| f_{\Theta}(z_{i,m}) - \bar{y}^{(m)} \right\|^2_2
\]

• Gradients w.r.t. variables

\[
\frac{\partial \Omega}{\partial \theta} = \beta \sum f_{\Theta}(z_{i,m}) - f_{\Theta}(z_{i,m-1}) \frac{\partial f_{\Theta}(z_{i,m})}{\partial \theta}
\]

\[
\frac{\partial \Omega}{\partial \omega} = \beta \sum f_{\Theta}(z_{i,m}) - f_{\Theta}(z_{i,m-1}) \frac{\partial f_{\Theta}(z_{i,m})}{\partial \omega}
\]

Supervised hashing
• Goal: learn hash codes for linear classification [Shen et al.]
• General problem:

\[
\min_{\Omega, W} \Omega(\Theta, W) + \sum_i \ell(W^T b_i, y_i), s.t., b_i = \text{sgn}(F(x_i, \Theta), y_i)
\]

Method
Train DNN with optimization problem:

\[
\min_{\Omega, W} \Omega(\Theta, W) + \sum_i \ell(W^T f_{\Theta}(z_{i,m}), y_i)
\]

s.t. \(F(x_i) = f_{\Theta}(z_{i,m}) = f_{\Theta}(\text{concat}(x_i, \Theta^{(m)})), 1 \leq m \leq M, \forall i\)

\[
\Theta = \{\Theta^{(m)}\}_{m=1}^{M}, \text{ network weights}
\]

\[
\omega \in \mathbb{R}^{\text{model size}}, \text{ layer weights}
\]

\[
f_{\Theta}(\cdot) \in \mathbb{R}^\text{model size}, \text{ activation function}
\]

\[
e.g., f_{\Theta}(x; \Theta^{(m)}) = \max(0, \Theta^{(m)} x)
\]

Decomposes the training process
• Break long term dependency: \(z_{i,m} = f_{\Theta}(x_{i,m}), \forall i, m\)
• Dependency between loss \(\ell\) and regularizer \(\Omega\):

\[
\frac{\partial \ell}{\partial \theta} \quad \text{and} \quad \frac{\partial \Omega}{\partial \theta}
\]

\[
\frac{\partial \ell}{\partial \omega} \quad \text{and} \quad \frac{\partial \Omega}{\partial \omega}
\]

Opt
\[
\frac{\partial \Omega}{\partial \theta} = \beta \sum f_{\Theta}(z_{i,m}) - f_{\Theta}(z_{i,m-1}) \frac{\partial f_{\Theta}(z_{i,m})}{\partial \theta}
\]

\[
\frac{\partial \Omega}{\partial \omega} = \beta \sum f_{\Theta}(z_{i,m}) - f_{\Theta}(z_{i,m-1}) \frac{\partial f_{\Theta}(z_{i,m})}{\partial \omega}
\]

\[
\frac{\partial \ell}{\partial \theta} = \frac{\partial \ell}{\partial f_{\Theta}(z_{i,m})} \frac{\partial f_{\Theta}(z_{i,m})}{\partial \theta}
\]

\[
\frac{\partial \ell}{\partial \omega} = \frac{\partial \ell}{\partial f_{\Theta}(z_{i,m})} \frac{\partial f_{\Theta}(z_{i,m})}{\partial \omega}
\]

Supervised hashing
• Goal: learn hash codes for linear classification [Shen et al.]
• General problem:

\[
\min_{\Omega, W} \Omega(\Theta, W) + \sum_i \ell(W^T b_i, y_i), s.t., b_i = \text{sgn}(F(x_i, \Theta), y_i)
\]

Opt

\[
\frac{\partial \Omega}{\partial \theta} = \beta \sum f_{\Theta}(z_{i,m}) - f_{\Theta}(z_{i,m-1}) \frac{\partial f_{\Theta}(z_{i,m})}{\partial \theta}
\]

\[
\frac{\partial \Omega}{\partial \omega} = \beta \sum f_{\Theta}(z_{i,m}) - f_{\Theta}(z_{i,m-1}) \frac{\partial f_{\Theta}(z_{i,m})}{\partial \omega}
\]

\[
\frac{\partial \ell}{\partial \theta} = \frac{\partial \ell}{\partial f_{\Theta}(z_{i,m})} \frac{\partial f_{\Theta}(z_{i,m})}{\partial \theta}
\]

\[
\frac{\partial \ell}{\partial \omega} = \frac{\partial \ell}{\partial f_{\Theta}(z_{i,m})} \frac{\partial f_{\Theta}(z_{i,m})}{\partial \omega}
\]

Alternate minimization

• Independent substructures
• Local updates

Application: Hashing

Very deep supervised hashing

Input features  Hidden layers  Hash codes  Class labels

Postdoc positions & code available: https://zimingzhang.wordpress.com/