Summary of Deep Neural Network Quantization

Chen Shangyu

April 17, 2019

Contents

1 Training-based Quantization

1.1 XNOR-Net [4]

1.2 DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients [7]

1.2.1 Introduction

This paper trained neural network with quantized weights, activation (layer input) and gradients. It can train AlexNet from scratch to reach 46.1% top-1 accuracy.

1.2.2 Weight Quantization

Straight-Through Estimator

$$\begin{aligned} \textbf{Forward:} \quad & r_o = \text{sign}(r_i) \\ \textbf{Backward:} \quad & \frac{\partial c}{\partial r_i} = \frac{\partial c}{\partial r_o} \mathbb{I}_{|r_i| \leq 1} \end{aligned} \tag{1}$$

k-bit representation of Straight-Through Estimator

$$\begin{aligned} \textbf{Forward:} \quad & r_o = 2 \times \text{quantize}_k \left(\frac{\tanh(r_i)}{2\text{max}(|\tanh(r_i))} + \frac{1}{2} \right) - 1 \\ \textbf{Backward:} \quad & \frac{\partial c}{\partial r_i} = \frac{\partial r_o}{\partial r_i} \times \frac{\partial c}{\partial r_o} \end{aligned} \tag{2}$$

quantize_k is a function to map continuous values into k-bit fixed point ranging in $[0, 1]$.

1.2.3 Activation Quantization

$$f_\alpha^k(r) = \text{quantize}_k(r) \tag{3}$$

1.2.4 Gradient Quantization

$$\begin{aligned} \textbf{Forward:} \quad & r_o r_i \\ \textbf{Backward:} \quad & \frac{\partial c}{\partial r_i} = f_\gamma^k(\frac{\partial c}{\partial r_o}) \end{aligned} \tag{4}$$

where:

$$f_\gamma^k(\text{d}r) = 2\text{max}_0(|\text{d}r|) \left[\text{quantize}_k \left(\frac{\text{d}r}{2\text{max}_0(|\text{d}r|)} + \frac{1}{2} + \text{noise} \right) - \frac{1}{2} \right] \tag{5}$$

1.2.5 Algorithm

Algorithm 1 Training a L-layer DoReFa-Net with W-bit weights and A-bit activations using G-bit gradients. Weights, activations and gradients are quantized according to Eqn. 9, Eqn. 11, Eqn. 12, respectively.

Require: a minibatch of inputs and targets (a_0, a^*), previous weights W, learning rate η
Ensure: updated weights W^{t+1}

{1. Computing the parameter gradients:}
{1.1 Forward propagation:}
for $k = 1$ **to** L **do**
  $W_k^b \leftarrow f_\omega^W(W_k)$
  $\tilde{a}_k \leftarrow \texttt{forward}(a_{k-1}^b, W_k^b)$
  $a_k \leftarrow h(\tilde{a}_k)$
  if $k < L$ **then**
    $a_k^b \leftarrow f_\alpha^A(a_k)$
  end if
  Optionally apply pooling
end for
{1.2 Backward propagation:}
Compute $g_{a_L} = \frac{\partial C}{\partial a_L}$ knowing a_L and a^*.
for $k = L$ **to** 1 **do**
  Back-propagate g_{a_k} through activation function h
  $g_{a_k}^b \leftarrow f_\gamma^G(g_{a_k})$
  $g_{a_{k-1}} \leftarrow \texttt{backward\_input}(g_{a_k}^b, W_k^b)$
  $g_{W_k^b} \leftarrow \texttt{backward\_weight}(g_{a_k}^b, a_{k-1}^b)$
  Back-propagate gradients through pooling layer if there is one
end for
{2. Accumulating the parameters gradients:}
for $k = 1$ **to** L **do**
  $g_{W_k} = g_{W_k^b} \frac{\partial W_k^b}{\partial W_k}$
  $W_k^{t+1} \leftarrow Update(W_k, g_{W_k}, \eta)$
end for

1.3 Training and Inference with Integers in Deep Neural Network [5]

1.3.1 Introduction

This paper discretized both training and inference, where weights (W), activations (A), gradients (G) and errors (E) among layers are shifted and linearly constrained to low-bitwidth integers (not exactly "integers").

1.3.2 Quantization Intervals and Conversion

$$\sigma(k) = 2^{1-k}, k \in \mathbb{N}_+ \tag{6}$$

Given a float-point x, it is converted to k-bitwidth signed integer representation:

$$Q(x,k) = \text{clip}\left\{\sigma(k) \cdot \text{round}\left[\frac{x}{\sigma(k)}\right], -1 + \sigma(k), 1 - \sigma(k)\right\} \tag{7}$$

e.g. $Q(x, 2)$ quantizes $\{-1, 0.2, 0.6\}$ to $\{-0.5, 0, 0.5\}$

1.3.3 Training Algorithm

Algorithm 1 Training an I-layer net with WAGE method on floating-point-based or integer-based device. Weights, activations, gradients and errors are quantized according to Equations 6 - 12.

Require: a mini-batch of inputs and targets $(\boldsymbol{a}_q^0, \boldsymbol{a}^*)$ which are quantized to k_A-bit integers, shift-based α for each layer, learning rate scheduler η, previous weight $\boldsymbol{W}$ saved in k_G bits.
Ensure: updated weights $\boldsymbol{W}_{t+1}$

1. Forward propagation:
for $i = 1$ to I **do**
  $\boldsymbol{W}_q^i \leftarrow Q_W(\boldsymbol{W}^i)$ #Clip
  $\boldsymbol{a}^i \leftarrow ReLU(\boldsymbol{a}_q^{i-1}\boldsymbol{W}_q^i)$ #MAC, Clip
  $\boldsymbol{a}_q^i \leftarrow Q_A(\boldsymbol{a}^i)$ #Shift, Clip
end for
2. Back propagation:
Compute $\boldsymbol{e}^I \leftarrow \frac{\partial \mathcal{L}}{\partial \boldsymbol{a}^I}$ knowing $\boldsymbol{a}^I$ and $\boldsymbol{a}^*$ #Substrate
for $i = I$ to 1 **do**
  $\boldsymbol{e}_q^i \leftarrow Q_E(\boldsymbol{e}^i)$ #Max, Shift, Clip
  $\boldsymbol{e}^{i-1} \leftarrow \boldsymbol{e}_q^i \boldsymbol{W}_q^i$ #MAC, Clip
  $\boldsymbol{g}^i \leftarrow {\boldsymbol{e}_q^i}^{\mathrm{T}} \boldsymbol{a}_q^{i-1}$ #MAC, Clip
  $\Delta \boldsymbol{W}^i \leftarrow Q_G(\boldsymbol{g}^i)$ #Max, Shift, Random, Clip
  Update and Clip $\boldsymbol{W}^i$ according to Equation 12 #Update, Clip
end for

Figure 1:

2 Direct Quantization

2.1 Fixed Point Quantization of Deep Convolution Networks [3]

This paper converted a pre-trained floating point Deep Convolution Network (DCN) into a fixed point model, which deals with the scenario that users have no access to origin training data. The conversion is based on signal-to-quantization-noise-ratio (SQNR).

2.1.1 Optimal Uniform Quantizer

$$\text{Range} \approx \text{Stepsize} \times 2^{\text{Bitwidth}} \tag{8}$$

Previous work on minimizing SQNR showed that there exist optimal Stepsize under different Bitwidth for various input distribution:

It furthur shows that SQNR has an approximatly linear relationship with bitwidth:

$$\gamma_{dB} \approx \kappa \times \beta \tag{9}$$

2.1.2 Model Conversion

- Run a forward pass in floating point using a large set of typical inputs and record the activations.
- Collect the statistics of weights, biases and activations for each layer.

- Determine the fixed point formats of the weights, biases and activations for each layer:
  - Determine the effective standard deviation of the quantity being quantized:ξ.
  - Calculate the step size via Table: $s = \xi \times \text{Stepsize}(\beta)$
  - Compute the number of fractional bits: $n = -\lceil \log_2 s \rceil$

2.1.3 Analysis of Quantization on SQNR

In the absence of model fine-tuning, converting a floating point deep network into a fixed point deep network is essentially a process of introducing quantization noise into the neural network. The effect of quantization can be accurately captured in a single quantity, the SQNR, which can be approximated theoretically and analyzed layer-by-layer.

For quantized weights $\tilde{w} = w + n_w$ and quantized activation: $\tilde{a} = a + n_a$. When the noise $|n_w| << |w|$ and $|n_a| << |a|$, we can have:

$$\frac{1}{\gamma_{w \times a}} = \frac{1}{\gamma_w} + \frac{1}{\gamma_a} \tag{10}$$

Eq.10 means that introducing quantization noise to weights and activations independently is equivalent to adding the total noise after the product operation in a normalized system. Similarily, in one layer forward, we have:

$$\frac{1}{\gamma_{w_{i,j}^{l+1} a_j^l}} = \frac{1}{\gamma_{w^{l+1}}} + \frac{1}{\gamma_{a^l}} \tag{11}$$

Eq.11 can be generalized to all the layers in a DCN:

$$\frac{1}{\gamma_{\text{output}}} = \frac{1}{\gamma_{a^0}} + \frac{1}{\gamma_{w^1}} + \frac{1}{\gamma_{a^1}} + ... + \frac{1}{\gamma_{w^L}} + \frac{1}{\gamma_{a^L}} \tag{12}$$

In other word, the SQNR at the output of a layer in DCN is the *Harmonic Mean* of the SQNRs of all preceding quantization steps: The network performance will be dominated by the worst quantization step, since $\gamma_{\text{output}} \leq \gamma_{a^l}$ for all l.

It further shows that layers with more parameters should use relatively lower bit-width.

3 Incremental Quantization

3.1 Incremental Network Quantization: Towards Lossless CNNs with Low-Precision Weights [6]

3.1.1 Introduction

This paper converted weights to be either powers of two or zero using an incremental method: **repeatedly** converted a portion and retrain the rest.

3.1.2 Quantization Intervals

$$\hat{W} = \{\pm 2^{n_1}, ..., \pm 2^{n_2}, 0\} \tag{13}$$

e.g. $\{\pm 1, \pm 0.5, \pm 0.25, \pm 0.125, ..., 0\}$

3.1.3 Training Algorithm

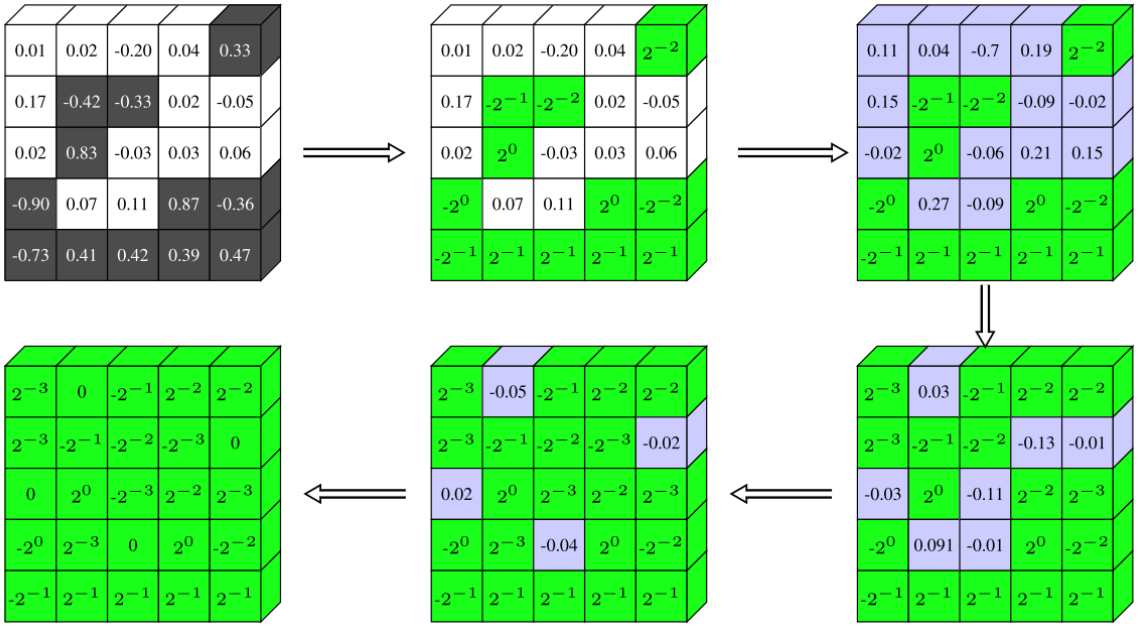

Figure 2: First row: results from the 1st iteration of the proposed three operations. The top left cube illustrates weight partition operation generating two disjoint groups, the middle image illustrates the quantization operation on the first weight group (green cells), and the top right cube illustrates the re-training operation on the second weight group (light blue cells). Second row: results from the 2nd, 3rd and 4th iterations of the INQ. In the figure, the accumulated portion of the weights which have been quantized undergoes from $50\% \rightarrow 75\% \rightarrow 87.5\% \rightarrow 100\%$

Algorithm 1 Incremental network quantization for lossless CNNs with low-precision weights.

Input: X: the training data, $\{\mathbf{W}_l : 1 \leq l \leq L\}$: the pre-trained full-precision CNN model, $\{\sigma_1, \sigma_2, \cdots, \sigma_N\}$: the accumulated portions of weights quantized at iterative steps

Output: $\{\widehat{\mathbf{W}}_l : 1 \leq l \leq L\}$: the final low-precision model with the weights constrained to be either powers of two or zero

1: Initialize $\mathbf{A}_l^{(1)} \leftarrow \emptyset$, $\mathbf{A}_l^{(2)} \leftarrow \{\mathbf{W}_l(i,j)\}$, $\mathbf{T}_l \leftarrow \mathbf{1}$, for $1 \leq l \leq L$
2: **for** $n = 1, 2, \ldots, N$ **do**
3: Reset the base learning rate and the learning policy
4: According to σ_n, perform layer-wise weight partition and update $\mathbf{A}_l^{(1)}$, $\mathbf{A}_l^{(2)}$ and $\mathbf{T}_l$
5: Based on $\mathbf{A}_l^{(1)}$, determine $\mathbf{P}_l$ layer-wisely
6: Quantize the weights in $\mathbf{A}_l^{(1)}$ by Equation (4) layer-wisely
7: Calculate feed-forward loss, and update weights in $\{\mathbf{A}_l^{(2)} : 1 \leq l \leq L\}$ by Equation (8)
8: **end for**

Figure 3:

4 Statistical Analysis of Weights and Activation

4.1 Deep Learning with Low Precision by Half-Wave Gaussian Quantization [1]

For weights quantization, it use

It considered quantizing the activation of deep neural network by exploiting the statistics of network activation: after batch normalization and dot-product of binary weights and quantized input, it approximately satisfies a Gaussian distribution with zeros mean and 1 std, as shown in Fig.4:

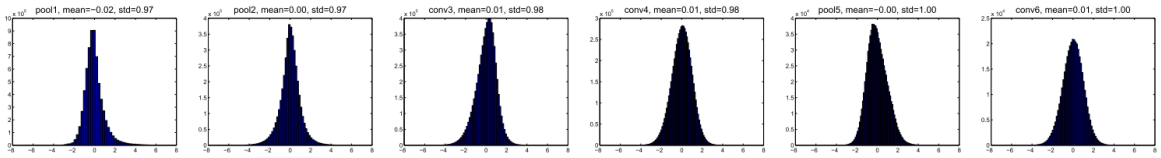

Figure 4: Dot-product distributions on different layers of AlexNet with binary weights and quantized activations (100 random images).

Therefore, it proposed the half-wave Gaussian quantizer for activation as ("Half" because of relu eliminate the negative response):

$$Q(x) = \begin{cases} q_i, & \text{if} \quad x \in (t_i, t_{i+1}) \\ 0, & x \leq 0 \end{cases} \tag{14}$$

t_is, q_is are the optimal quantization parameters which is attained by Lloyd's algorithm: it drew 10^6 samples from a standard Gaussian distribution of zero mean and unit variance, and obtained the optimal quantization parameters by Lloyd's algorithm. The resulting parameters t_i^* and q_i^* were used to parametrize a single HWGQ that was used **in all layers**, after batch normalization of dot-products.

It then proposed 3 types of backward functions to solve the problem of derivatives vanishing.

- Vanilla ReLU:

$$\tilde{Q}'(x) = \begin{cases} 1, & \text{if} \quad x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

- Clipped ReLU:

$$\tilde{Q}'(x) = \begin{cases} q_m, & x > q_m, \\ x, & x \in (0, q_m], \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

- Log-tailed ReLU:
  Forward:

$$\tilde{Q}(x) = \begin{cases} q_m + \log(x - \tau), & x > q_m, \\ x, & x \in (0, q_m], \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

Backward:

$$\tilde{Q}'(x) = \begin{cases} \dfrac{1}{x-\tau}, & x > q_m, \\ 1, & x \in (0, q_m], \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

where $\tau = q_m - 1$, the log-tailed ReLU is identical to the vanilla ReLU for dot products of amplitude smaller than q_m, but gives decreasing weight to amplitudes larger than this.

5 Theoretical Analysis of Quantization

5.1 Training quantized nets: A deeper understanding [2]

This work investigated training methods for quantized neural networks from a theoretical viewpoint:

5.1.1 Stochastic Rounding

$$w_b^{t+1} = Q_s(w_b^t - \alpha_t \nabla \tilde{f}(w_b^t)) \tag{19}$$

$$Q_s(w) = \Delta \times \begin{cases} \left\lfloor \frac{w}{\Delta} \right\rfloor + 1 & \text{for} \quad p \leq \frac{w}{\Delta} - \left\lfloor \frac{w}{\Delta} \right\rfloor \\ \left\lfloor \frac{w}{\Delta} \right\rfloor & \text{otherwise} \end{cases} \tag{20}$$

Quantization Error:

$$w^{t+1} = w^t - \alpha_t \nabla \tilde{f}(w^t) + r^t \tag{21}$$

where $r^t = Q_s(w^t - \alpha_t \nabla \tilde{f}(w^t)) - w^t + \alpha_t \nabla \tilde{f}(w^t)$, which is bounded by:

$$\mathbb{E}||r^t||^2 \leq \sqrt{d}\Delta\alpha_t G \tag{22}$$

G is a bound for: $\mathbb{E}||\nabla \tilde{f}(w^t)||^2 \leq G^2$, which is proven in the paper.

Convergence Analysis:
Assumption:

- Loss function F is μ-strongly convex: $f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\mu}{2}||y - x||_2^2$
- Gradient is bounded: $\mathbb{E}||\nabla f(w^t)|| \leq G^2$

Theorem 1. *Assume that F is μ-strongly convex and the learning rates are given by $\alpha_t = \frac{1}{\mu(t+1)}$. Consider the SR algorithm with updates of the form* (19)*. Then, we have:*

$$\mathbb{E}[F(\bar{w}^T) - F(w^*)] \leq \frac{(1 + log(T+1))G}{2\mu T} + \frac{\sqrt{d}\Delta G}{2} \tag{23}$$

where $\bar{w}^T = \frac{1}{T}\sum_{i=1}^{t} w^t$.

General Idea:

- Start from quantization error, which is related to quantization resolution, gradient etc.
- Then introduce F by μ-strongly convex.
- Finally telescope sum to reduce intermediate term.

Detailed Proof:
We start from weights update:

$$w^{t+1} = w^t - \alpha_t \nabla f(w^t) + r^t \rightarrow w^{t+1} - w^* = (w^t - w^*) - (\alpha_t \nabla f(w^t) - r^t)$$

Take expectation of both sides:

$$\begin{aligned}
&\mathbb{E}\|w^{t+1} - w^*\|^2 \\
&= \|w^t - w^*\|^2 - 2\underbrace{\mathbb{E}\langle w^t - w^*, \alpha_t \nabla \tilde{f}(w^t) - r^t\rangle}_{\mathbb{E}[r^t]=0} + \underbrace{\mathbb{E}\|\alpha_t \nabla \tilde{f}(w^t) - r^t\|^2}_{\mathbb{E}[r^t]=0} \\
&= \|w^t - w^*\|^2 - 2\alpha_t \langle w^t - w^*, \nabla F(w^t)\rangle + \alpha_t^2 \mathbb{E}\|\nabla \tilde{f}(w^t)\|^2 + \mathbb{E}\|r^t\|^2 \\
&\leq \|w^t - w^*\|^2 - 2\alpha_t \langle w^t - w^*, \nabla F(w^t)\rangle + \alpha_t^2 G^2 + \underbrace{\sqrt{d}\Delta\alpha_t G}_{\mathbb{E}\left\|r^t\right\|_2^2 \leq \sqrt{d}\Delta\alpha_t G.},
\end{aligned}$$

By μ-strongly convex: $F(w^*) - F(w^t) \geq \langle w^* - w^t, \nabla F(w^t)\rangle + \frac{\mu}{2}||w^* - w^t||^2 \rightarrow$

$$\begin{aligned}
\mathbb{E}\|w^{t+1} - w^*\|^2 \leq (1 - \alpha_t\mu)\|w^t - w^*\|^2 - 2\alpha_t(F(w^t) - F(w^*)) \\
+\alpha_t^2 G^2 + \sqrt{d}\Delta\alpha_t G.
\end{aligned}$$

Re-arranging the terms, taking expectation, and asssume that the stepsize decreases with the rate $\alpha_t = 1/\mu(t+1)$. Then we have:

$$\begin{aligned}
\mathbb{E}(F(w^t) - F(w^*)) \leq \frac{\mu t}{2}\mathbb{E}\|w^t - w^*\|^2 - \frac{\mu(t+1)}{2}\mathbb{E}\|w^{t+1} - w^*\|^2 \\
+\frac{1}{2\mu(t+1)}G^2 + \frac{\sqrt{d}\Delta G}{2}.
\end{aligned}$$

Averaging over $t = 0$ to T, we get a telescoping sum on the right hand side:

$$\begin{aligned}
\{t = t\} : &\underbrace{\frac{\mu t}{2}\mathbb{E}\|w^t - w^*\|^2}_{\text{eliminate}} - \frac{\mu(t+1)}{2}\mathbb{E}\|w^{t+1} - w^*\|^2 \\
\{t = t-1\} : &\underbrace{\frac{\mu(t-1)}{2}\mathbb{E}\|w^{t-1} - w^*\|^2}_{\text{eliminate when } t=1} - \underbrace{\frac{\mu t}{2}\mathbb{E}\|w^t - w^*\|^2}_{\text{eliminate}}
\end{aligned}$$

After elimination:

$$\begin{aligned}
\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}(F(w^t) - F(w^*)) &\leq \frac{G^2}{2\mu T}\sum_{t=0}^{T}\frac{1}{t+1} + \frac{\sqrt{d}\Delta G}{2} \\
&- \frac{\mu(T+1)}{2}\mathbb{E}\|w^{T+1} - w^*\|^2(\text{Get rid of}) \\
&\leq \frac{(1 + \log(T+1))G^2}{2\mu T} + \frac{\sqrt{d}\Delta G}{2}.
\end{aligned}$$

Using Jensen's inequality, we have:

$$\mathbb{E}(F(\bar{w}^T) - F(w^*)) \leq \frac{1}{T}\sum_{t=0}^{T}\mathbb{E}(F(w^t) - F(w^*))$$
$$\leq \frac{(1+\log(T+1))G^2}{2\mu T} + \frac{\sqrt{d}\Delta G}{2} \qquad \text{Q.E.D}$$

5.1.2 Binary Connect

$$\text{BinaryConnect:} \quad w_r^{t+1} = w_r^t - \alpha_t \nabla \tilde{f}\big(Q(w_r^t)\big). \tag{24}$$

Assumption:

- Hessian satisfies the Lipschitz bound: $\|\nabla^2 f_i(x) - \nabla^2 f_i(y)\| \leq L_2\|x-y\|$ for some $L_2 \geq 0$.

Theorem 2. *Assume F is L-Lipschitz smooth, the domain has finite diameter D, and learning rates are given by $\alpha_t = \frac{c}{\sqrt{t}}$. Consider the BC-SGD algorithm with updates of the form* (24)*. Then, we have:*

$$\mathbb{E}[F(\bar{w}^T) - F(w^*)] \leq \frac{1}{2c\sqrt{T}}D^2 + \frac{\sqrt{T+1}}{2T}cG^2 + \sqrt{d}\Delta LD.$$

Detailed Proof:
We start from weights update:

$$w^{t+1} = w^t - \alpha_t \nabla \tilde{f}\big(Q(w^t)\big) = w^t - \alpha_t \nabla \tilde{f}\big(w^t + r^t\big).$$

Taking expectation conditioned on w^t and r^t, we have

$$\begin{aligned}
&\mathbb{E}\|w^{t+1} - w^*\|^2 \\
&= \mathbb{E}\|w^t - \alpha_t \nabla \tilde{f}\big(w^t + r^t\big) - w^*\|^2 \\
&= \mathbb{E}\|w^t - \alpha_t \nabla \tilde{f}\big(w^t\big) + \alpha_t \nabla \tilde{f}\big(w^t\big) - \alpha_t \nabla \tilde{f}\big(w^t + r^t\big) - w^*\|^2 \\
&= \|w^t - w^*\|^2 - 2\alpha_t \mathbb{E}\langle w^t - w^*, \nabla \tilde{f}\big(w^t\big)\rangle + 2\alpha_t \mathbb{E}\langle w^t - w^*, \nabla \tilde{f}\big(w^t\big) - \nabla \tilde{f}\big(w^t + r^t\big)\rangle + \mathbb{E}\|\alpha_t \nabla \tilde{f}\big(w^t + r^t\big)\|^2 \\
&= \|w^t - w^*\|^2 - 2\alpha_t \langle w^t - w^*, \nabla F\big(w^t\big)\rangle + 2\alpha_t \langle w^t - w^*, \nabla F\big(w^t\big) - \nabla F\big(w^t + r^t\big)\rangle + \alpha_t^2 \mathbb{E}\|\nabla \tilde{f}\big(w^t + r^t\big)\|^2 \\
&\leq \|w^t - w^*\|^2 - 2\alpha_t \langle w^t - w^*, \nabla F\big(w^t\big)\rangle + 2\alpha_t \|w^t - w^*\| \|\nabla F\big(w^t\big) - \nabla F\big(w^t + r^t\big)\| + \alpha_t^2 G^2 \\
&\leq \|w^t - w^*\|^2 - 2\alpha_t \langle w^t - w^*, \nabla F\big(w^t\big)\rangle + 2\alpha_t L\|r^t\| \|w^t - w^*\| + \alpha_t^2 G^2.
\end{aligned}$$

6 Quantization Training

6.1 Deep Learning as a Mixed Convex Combinatorial Optimization Problem

Reference

[1] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5406–5414, 2017.

[2] Hao Li, Soham De, Zheng Xu, Christoph Studer, Hanan Samet, and Tom Goldstein. Training quantized nets: A deeper understanding. In *Advances in Neural Information Processing Systems*, pages 5811–5821, 2017.

[3] Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning*, pages 2849–2858, 2016.

[4] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[5] Feng Chen Luping Shi Shuang Wu, Guoqi Li. Training and inference with integers in deep neural network. *To be appeared in ICLR 2018*, 2018.

[6] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017.

[7] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.