Low-bit Quantization of Neural Networks for Efficient Inference

Yoni Choukroun Eli Kravchik Fan Yang Pavel Kisilev
Huawei Technologies Co.
{yoni.choukroun,eli.kravchik,yangfan74,pavel.kisilev}@huawei.com

Abstract

Recent machine learning methods use increasingly large deep neural networks to achieve state of the art results in various tasks. The gains in performance come at the cost of a substantial increase in computation and storage requirements. This makes real-time implementations on limited resources hardware a challenging task. One popular approach to address this challenge is to perform low-bit precision computations via neural network quantization. However, aggressive quantization generally entails a severe penalty in terms of accuracy, and often requires retraining of the network, or resorting to higher bit precision quantization. In this paper, we formalize the linear quantization task as a Minimum Mean Squared Error (MMSE) problem for both weights and activations, allowing low-bit precision inference without the need for full network retraining. We propose the analysis and the optimization of constrained MSE problems for performant hardware aware quantization. The proposed approach allows 4 bits integer (INT4) quantization for deployment of pretrained models on limited hardware resources. Multiple experiments on various network architectures show that the suggested method yields state of the art results with minimal loss of tasks accuracy.

1. Introduction

Neural networks (NNs) proved to be extremely effective in solving a broad variety of problems in computer vision, speech recognition and natural language processing [22, 14, 36]. Deep learning methods are usually evaluated only according to their accuracy over a given task. This criterion leads to the development of architectures with constantly increasing computational complexity and memory requirements. Thus, performing inference on low power System on a Chip (SoCs) used in smartphones or IoT devices is a significant challenge, due to the limited available memory and computational resources.

Several approaches have been proposed in order to make deep NNs less resource demanding. Network pruning of redundant and non-informative weights allows significant reduction of the network size [12, 11]. Matrix factorization via low-rank approximation exploits the redundancy property of the NN parameters in order to increase speed up [7]. Distillation of NNs aims to transfer the knowledge contained in a pretrained large network to a compressed model via adapted training process [15]. Also, new architectures (i.e.[19, 16, 17]) with more efficient operations such as point/depth-wise or grouped convolutions allow the reduction of the model size, compared to existing over-parameterized architectures.

Another popular direction, which we focus on in this paper, is the quantization of NN. Quantization methods attempt to reduce the precision of the NN parameters and/or activations from single precision (32 bit floating point, or FP32) to lower bit representations. Several benefits of low-bit precision can be exploited by deep learning accelerators. The storage requirement for a low-bit precision model can be diminished substantially, as well as the power consumption. Similarly, the memory bandwidth requirements can be significantly reduced. Since the multiply accumulate (MAC) operations are performed on low-bit processing engines, the computational complexity can be reduced as well. Perhaps the most important benefit of low bit representation is the saving of chip area. For instance, 8 bits integer (INT8) operations can save up to 30x energy and up to 116x area compared to FP32 operations [6], allowing significantly better computational throughput. However, low-bit precision inference often causes loss of the task accuracy, which is usually compensated with the help of heavy full retraining, mixed precision or non-uniform quantization

Trained quantization is a powerful but time-consuming approach, challenging to implement. It requires access to the full training dataset which might not always be available either due to privacy reasons, or if an off-the-shelf pretrained model is used. In this paper we address the *post-training* quantization problem for weights and/or activations of a pretrained FP32 NN on highly constrained hardware, wherein complete retraining, non-uniform quantization or mixed precision calculations cannot be tolerated. To the best of our knowledge, this is the first time *INT4 only* deployment of a pretrained NN via efficient *linear quantization* is performed with minimal loss of accuracy and data requirement. We propose a simple yet efficient optimization framework to find the *optimal* quantization parameters in the MMSE sense at each layer separately. The proposed MMSE reconstruction gives better accuracy than other existing MSE based optimization methods. NN parameters are quantized in a kernel-wise fashion that does not violate the linearity of the dot product operations, enabling efficient deployment on any common deep learning hardware. We identify key layers that are most sensitive to quantization errors, and provide an adaptive quantization scheme that makes use of multiple low precision tensors. Finally, we propose a refinement procedure for the scaling factors of the quantized tensors, which is performed on a small unlabelled calibration set. In a similar manner, the NN activations quantization coefficients are obtained offline via MMSE criterion. More sensitive activations are better approximated according to their reconstruction residual. The main contributions of this paper can be summarized as follows:

- We propose a hardware compliant low-bit precision linear quantization framework, for fast deployment of pretrained models on *low power* accelerators.
- We analyze the non-convexity of the constrained MMSE quantization problem and consequently propose optimal optimization that outperforms uniform and existing MMSE based quantizations. The quantization parameters are further differentiably refined for better performance.
- We explore the MMSE quantization process and enlighten the need for precise quantization of some layers. In contrast to training methods that widen all of the model's layers for improved accuracy, we widen only few detected *key* layers in order to keep the compression ratio low.

Extensive experiments on ImageNet demonstrate that our INT4 linear quantization method for both weights and activations, performs inference with only 3% top-1 and 1.7% top-5 mean accuracy degradation, as compared to the FP32 models, reaching state-of-art results. The above degradation can be further reduced according to the complexity-accuracy trade-off inherent to the proposed method. The remainder of the paper is organized as follows. Section 2 reviews related works. In section 3, after analyzing the MMSE quantization challenges, we develop our MMSE based quantization for accurate approximation of original models. In this section we also provide experimental results to demonstrate the usefulness of the steps in the proposed quantization pipeline. At the end of the section, we describe the modification of the presented algorithm components for the problem of activation quantization. Finally, quantization results from our experiments on several popular NNs are presented in Section 4.

2. Related Work

Neural network acceleration received increasing attention in the deep learning community, where the need for accurate yet fast and efficient frameworks is crucial for real-world applications. A powerful approach is quantization of NNs to low-bit representation. There are two main quantization scenarios. The first one is the full training of a given model to a desired lower bit precision. With this approach, the weights, the activations and even the gradients can be quantized to very low precision, enabling potentially fast training and inference [38]. The major problem with the training approach above arises from the discritness of the parameters, wherein the backpropagation approach is not well defined. The "straight-through estimator" [3] has been used in [4, 32, 5] in order to estimate the gradient of a stochastic neuron. [4] proposed to use stochastic quantization of the weights via random rounding, in order to inject regularizing noise to the training process. [35] suggests to approximate solution using variational Bayes method where the weights can be restricted to discrete values assuming Gaussian distribution. Instead of seeking for appropriate derivatives, [33] assumed smooth approximation of parameters with defined gradients. Non-uniform quantization of NN parameters has been proposed in [10] where the parameters are approximated using k-means algorithm. [24] proposed a high order quantization scheme of weights where the approximation residual is further processed allowing better refinement of full precision input. Estimation of the quantization parameters by solving constrained optimization problem has been proposed for binary [32] and ternary weights [23].

We focus on the second quantization scenario that targets direct quantization of a pretrained FP32 network to a lower bit-depth precision without full training. INT8 quantization of parameters has been proven to be relatively robust to quantization noise even with simple uniform quantization of weights [20]. [18] proposed L_2 error minimization of weights via alternating optimization in order to obtain a generalizing ability during the training. Nevertheless, INT8 quantization of the network activations is more challenging because of real time constraints. Nvidia proposed TensorRT [29], a quantization framework that searches for saturation threshold of the activations, based on the Kullback-Leibler divergence measure between the quantized activations and their full precision counterpart. Recently, [2] proposed to approximate activations, as if they were sampled from a known distribution in order to obtain, under some assumptions, analytically optimal threshold in the L_2 sense. However, quantization of full precision weights and activations to less than 8-bits usually causes significant loss of accuracy, a problem that has not been solved yet. In order to overcome such degradation in performance, quantization frameworks resort to retraining procedures, mixed precision solutions or non-uniform quantization. These solutions make fast and easy deployment of quantized NNs impossible, especially on highly constrained HW such as mobiles or IoT devices.

3. Proposed MMSE Quantization Method

Conversion of a full precision NN into its fixed point version introduces quantization noise (error) into the network. Since significant noise may deteriorate the model performance, in the absence of training capabilities, efforts should be invested in minimizing the noise power, in order to approximate the original model as accurately as possible. Minimization of the noise power of weights and of activations via MSE optimization is a natural criterion for quantization quality, even though no direct relationship can be easily established between the noise of the output and the model accuracy. In the following, we investigate how quantization affects the NN output, and propose optimal solution to the quantization problem, via MSE optimization with fixed precision constraints.

3.1. Quantization Process

Consider an L-layer NN architecture defined by $\{W_l, X_l\}_{l=1}^{L}$ with W_l and X_l being the l^{th} layer weight and input tensors respectively. Each weight is defined according to the set $W_l = \{W_{lk}\}_{k=1}^{k_l}$, where $W_{lk} \in \mathbb{R}^{c_l \times w_{kl} \times h_{kl}}$ denotes the k^{th} kernel (neuron), with c_l, w_{kl}, h_{kl} being the number of channels, the kernel's width and height, respectively. Similarly, $X_l \in \mathbb{R}^{c_l \times w_l \times h_l}$ represent the layer activations (feature maps), with c_l, w_l, h_l being the number of channels, width and height of the layer input, respectively.

In our setting we constrain a tensor $T \in \mathbb{R}^{c \times w \times h}$ to be approximated *linearly* as a quantized tensor $\tilde{T} \in \mathbb{Z}_p^{c \times w \times h}$ coupled with a scaling factor $\alpha \in \mathbb{R}$ such that $T \approx \hat{T} = \alpha \tilde{T}$. Here p denotes the desired integer bit precision, and $\mathbb{Z}_p$ the corresponding quantization range. Linear approximations are particularly appealing from hardware perspective, wherein the tensor multiplication unit can handle low precision tensors only. In this setting, the tensor multiplication is approximated as

$$T_3 = T_1 T_2 \approx (\alpha_1 \tilde{T}_1)(\alpha_2 \tilde{T}_2) = (\alpha_1 \alpha_2)(\tilde{T}_1 \tilde{T}_2) = \alpha_3 \tilde{T}_3. \quad (1)$$

A popular uniform quantization scheme is given by

$$\begin{aligned} \hat{T} &= \alpha \left[\frac{T - \delta}{\alpha} \right]_{\mathbb{Z}_p} + \delta \\ \delta &= \min_i (T_i) \\ \alpha &= \frac{\max_i (T_i) - \delta}{2^p - 1} \end{aligned} \quad (2)$$

where $[T]_{\mathbb{Z}_p} = \min(\max(\lfloor T \rceil, \min(\mathbb{Z}_p)), \max(\mathbb{Z}_p))$ denotes the rounding operator $\lfloor \cdot \rceil$, followed by saturation to the $\mathbb{Z}_p$ domain. The quantization offset δ can be of major importance for non symmetric distributions (e.g ReLU [9] activations), where *unsigned* representations $\mathbb{Z}_p = \mathbb{Z}_p^+ \in \{0, \dots, 2^p - 1\}$ allow an additional discrete level of representation. However, the use of offset obviously increases the computational complexity of tensor multiplications. In the case quantization offset is not allowed, we assume $\delta = 0$, $\alpha = \max_i(|T_i|)/(2^{p-1} - 1)$ and the *signed* range $\mathbb{Z}_p \in \{-2^{p-1}, \dots, 2^{p-1} - 1\}$.
Unless stated otherwise the results presented are obtained with signed quantization (no offset). Even though many quantization methods do not quantize the first and last layer of the model [12, 38, 32], unless stated otherwise here we quantize *all* the network parameters and activations (including the network input).

3.2. Mean Squared Error Analysis of Quantization

The relation between the full precision tensor weights W and activations X and their respective approximations $\hat{W}$ and $\hat{X}$ can be obtained as follows

$$\begin{aligned} \hat{Y} &= \hat{W}\hat{X} = (W + n_W)(X + n_X) \\ &= WX + Wn_X + n_W X + n_W n_X \\ &\approx WX + Wn_X + n_W X = Y + Wn_X + n_W X, \end{aligned} \quad (3)$$

where n_W, n_X denote the quantization noise of the weights and activations respectively and where the approximation is obtained by neglecting second order noise term. Let us consider the case where the NN is composed of linear layers only. In such setting the NN output Y is defined as

$$Y = W_L(W_{L-1} \dots (W_1 X_1)) = W_L X_L. \quad (4)$$

For ease of notation we will omit the mean factor of the MSE. Defining e_L^2, the MSE between the original model output and the quantized model output, we obtain in expectation that

$$
\begin{aligned}
\mathbb{E}(e_L^2) &= \mathbb{E}\,\|Y-\hat{Y}\|_F^2 = \mathbb{E}\,\|W_L X_L - \hat{W}_L \hat{X}_L\|_F^2 \\
&= \mathbb{E}\,\|W_L n_{X_L} + n_{W_L} X_L\|_F^2 \\
&= \mathbb{E}\,\|W_L n_{X_L}\|_F^2 + 2\,\mathbb{E}\,trace(n_{X_L}^T W_L^T X_L n_{W_L}) \\
&+ \mathbb{E}\,\|n_{W_L} X_L\|_F^2 \\
&\approx \mathbb{E}\,\|W_L n_{X_L}\|_F^2 + \mathbb{E}\,\|n_{W_L} X_L\|_F^2 \\
&\leq \mathbb{E}\,\|n_{W_L}\|_F^2\,\mathbb{E}\,\|X_L\|_F^2 + \mathbb{E}\,\|W_L\|_F^2\,\mathbb{E}\,\|n_{X_L}\|_F^2 \\
&= \mathbb{E}\,\|n_{W_L}\|_F^2\,\mathbb{E}\,\|X_L\|_F^2 + \mathbb{E}\,\|W_L\|_F^2\,\mathbb{E}(e_{L-1}^2),
\end{aligned}
\tag{5}
$$

where the approximation is obtained by assuming zero mean noise [31] and where $\|\cdot\|_F$ denotes the Frobenius norm. The inequality is obtained using Cauchy Schwarz inequality and by assuming the weights and the activations quantization noise are statistically independent, as well as the activations and the weights noise. We obtain here a recursive expression of the network output MSE. It is obvious that *first* layers may have significant impact on the output quality because of the recursion factor. Also, special attention should be given to the minimization of the *weights* noise which is coupled with possibly unbounded activations (ReLU). In real configurations, the non linear functions that play major role in the discriminative power of deep networks are much harder to model and analyze. Nevertheless, the analysis presented above is supported empirically in our non-linear experiments, wherein the first layers always impact model accuracy the most, in contrast to the analysis of [25]. Also, in our setting the approximation quality of the weights has much more influence on the performance than the activations, in contrast to the analysis of [38].

3.3. Kernel-Wise Quantization

In the low precision set up, such as 4 bit, *global* quantization where all the quantized kernels are coupled with a *single* scaling factor, such that $\hat{W}_l = \{\alpha_l \tilde{W}_{lk}\}_k$, can lead to poor performance, due to the high variance of all the accumulated kernel elements. This problem does not arise in the INT8 deployment setting that is robust enough even under global quantization [29, 20]. Applying different scaling factors to different buckets (partitions) of contiguous values allows better approximation of the original tensor [1]. However, the quantization framework should preserve the linearity of the dot product, so that for every two vectors $x, y \in \mathbb{R}^n$ we get

$$
\langle x, y\rangle \approx \langle \alpha\tilde{x}, \beta\tilde{y}\rangle = \sum_i^n \alpha\beta\tilde{x}_i\tilde{y}_i = \alpha\beta\langle\tilde{x},\tilde{y}\rangle_{\mathbb{Z}_p}. \tag{6}
$$

Maintaining the linearity is not so popular because of its restrictiveness and sub-optimality. For example, with the efficient group-wise [1, 28], channel-wise or pixel-wise quantization settings [8], the dot product must be split into several dot products to be accumulated together, such that $\langle x, y\rangle \approx \beta\sum_j^J \alpha_j\langle\tilde{x}^j,\tilde{y}^j\rangle_{\mathbb{Z}_p}$. Such procedures cannot be efficiently implemented on deep learning accelerators with dedicated matrix multiplication units such as in systolic arrays [21]. The only viable bucketing approach is the *kernel wise* approach where each kernel is coupled to its own scaling factor such that $\hat{W}_l = \{\alpha_{lk}\tilde{W}_{lk}\}_k$. Such setting maintains the dot product linearity at the kernel level and allows efficient tensors multiplication in low precision accelerators. Each discrete feature map is then multiplied with its corresponding floating point scaling factor. Thus, without violating the dot product linearity and with negligible storage addition (as the number of kernels), a significant improvement in performance can be obtained as summarized in Table 1. In this experiment, all convolutional layers are quantized *kernel-wise*, while all the fully connected layers are *globally* quantized. Also, we only allow *global* quantization of activations since every possible partitioning yields separation of the dot product induced by the convolutions.

Architecture	Original	Global	Kernel-wise
Alexnet [22]	56.624%	0.694%	**46.796%**
VGG16bn [34]	73.476%	3.034%	**65.23%**
Inception v3 [37]	76.226%	0.106%	**12.564%**
Resnet18 [13]	69.644%	1.83%	**44.082%**
Resnet50 [13]	76.012%	0.172%	**62.242%**
Resnet101 [13]	77.314%	0.148%	**64.434%**
SqueezeNet [19]	58.0%	1.528%	**29.908%**
DenseNet [17]	74.472%	0.58%	**57.072%**

Table 1. Top1 accuracy using INT4 *uniform* quantization of weights (FP32 activations) with global scaling factor [29, 20] for all kernels and the proposed kernel-wise approach on the ImageNet validation set. Global quantization simply collapses.

3.4. Minimum MSE Quantization

We aim at finding optimal approximation $\hat{T}$ of a given tensor $T \in \mathbb{R}^{c\times w\times h}$ by solving the constrained MSE problem as follows

$$
\begin{aligned}
\min_{\alpha,\tilde{T}} \quad & \|T-\alpha\tilde{T}\|_F^2. \\
\text{s.t} \quad & \alpha\in\mathbb{R}, \tilde{T}\in\mathbb{Z}_p^{c\times w\times h}
\end{aligned}
\tag{7}
$$

For any arbitrary precision p this optimization problem does not have an analytical solution. Assuming optimal $\alpha \neq 0$ is given, and denoting T_i the i^{th} element of the tensor T, we remain with a one constraint optimization problem where we can rewrite eq. (7) objective as

$$
\|T-\alpha\tilde{T}\|_F^2 = \sum_i (T_i - \alpha\tilde{T}_i)^2 = \alpha^2\sum_i\left(\frac{T_i}{\alpha} - \tilde{T}_i\right)^2. \tag{8}
$$

Thus, the *optimal* quantization is uniform and the quantized values are given according to $\tilde{T}_i = \lceil T_i/\alpha \rfloor_{\mathbb{Z}_p}\ \forall i$, where the scaling factor α also defines *saturation* of the the tensor values. Typical MSE as a function of α is shown in Figure 1.

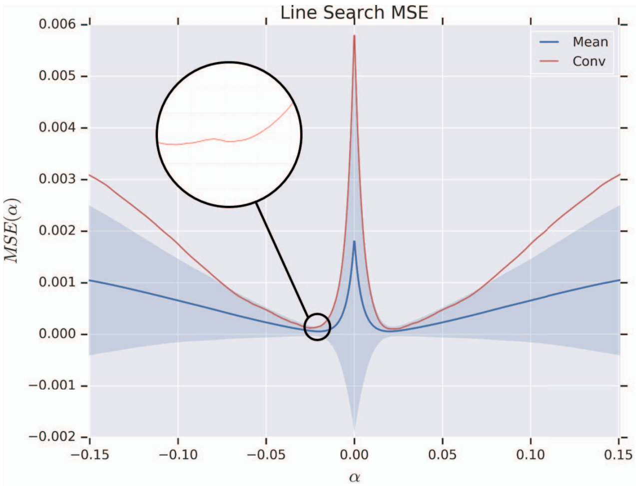

Figure 1. MSE as a function of the scaling factor α averaged for all the Alexnet convolutional kernels (Mean). Typical kernel MSE is presented in red (Conv) and zoomed to show the non-convexity of the function.

Architecture	Uniform	Altern.	Golden	OMSE
Alexnet	46.796%	40.962%	46.070%	**46.892%**
VGG16bn	65.23%	55.936%	62.250%	**65.414%**
Inception v3	12.564%	4.368%	7.408%	**22.028%**
Resnet18	44.082%	52.646%	52.398%	**56.688%**
Resnet50	62.242%	60.186%	63.178%	**67.356%**
Resnet101	64.434%	56.282%	63.052%	**65.066%**
SqueezeNet	29.908%	24.040%	**34.036%**	32.262%
DenseNet	57.072%	45.668%	50.058%	**59.012%**

Table 2. Top1 accuracy for INT4 quantized weights and FP32 activations using kernel-wise uniform quantization [29, 20], Alternating optimization (Altern.)[18], Golden Section search (Golden) and the proposed optimal MSE (OMSE) quantization.

Several methods for finding optimal α can be considered. Alternating optimization [18], even on one half of the domain, is not robust due to the non-convexity of the quantization optimization (8) where small ripples make convex optimization methods fail (Figure 1). Golden Section Search gives fast and better results because of the low resolution optimization at the early stage of the algorithm, that avoids attraction to bad local minima, but fails to reach optimum. In this work, we propose to use one dimensional exact line-search, that can be implemented very efficiently over the grid defined by the tensor range (few hundred points) using parallel computation. This approach allows *optimal* solution (up to the grid density) of the non-convex MMSE quantization problem. A comparison of different quantization methods is presented in Table 2 where we show the importance of optimal MSE solution.

3.5. Multiple Tensor Quantizations

Some layers can be harder to approximate than others, and have larger impact on the model accuracy loss, as shown in (5). In order to better reconstruct those layers, some works use mixed precision [25], where some layers use more bits for representation. Due to power and chip area considerations, low power devices generally do not allow mixed precision inference, because of the constrained low precision engines. We therefore propose to better approximate *key* layers by using multiple low precision quantized tensors. We use the term key layers to describe quantized layers with *high MSE*, or MSE bigger than some threshold τ. In contrary to other training methods where kernels are blindly added over all the network layers [26, 30], our approach allows *minimal* computational overhead. We formalize this setting as follows. Given n integers $\{p_i\}_{i=1}^n$ denoting the desired precision of the quantized tensors and a given tensor T, we want to find the n scalars $\{\alpha_i\}_{i=1}^n$ and the quantized tensors $\{\tilde{T}^i\}_{i=1}^n$ that best approximate the original tensor in the L_2 sense, such that

$$\min_{\alpha_i,\tilde{T}^i}, \quad \|T - \sum_{i=1}^{n} \alpha_i \tilde{T}^i\|_F^2 \qquad \text{s.t} \quad \alpha_i \in \mathbb{R}, \tilde{T}^i \in \mathbb{Z}_{p_i}^{c\times w\times h} \tag{9}$$

From the computational perspective, the proposed framework approximates a given kernel with quantized filter bank for better approximation. Thus, for a given tensor X approximated as $X \approx \hat{X} = \beta\tilde{X}$ we have

$$TX \approx \sum_{i=1}^{n} \alpha_i \tilde{T}^i X \approx \sum_{i=1}^{n} \alpha_i \beta(\tilde{T}^i \tilde{X}). \tag{10}$$

Here we opt for a nested optimization approach, where the solution is obtained iteratively until convergence. Then, the optimization can be written in the nested form as

$$\min_{\alpha_1,\tilde{T}^1} \{ \min_{\alpha_2,\tilde{T}^2} ...\{ \min_{\alpha_n,\tilde{T}^n} \{ \quad \|T - \sum_{i=1}^{n} \alpha_i \tilde{T}^i\|_F^2 \}\}\} \tag{11}$$

Alternating optimization is used until convergence, while at each iteration the quantized tensor and its scaling factor are obtained using the MSE quantization mapping suggested in Section 3.4. The proposed quantization approach for layers with high MSE is summarized in Algorithm 1. In the algorithm below, input T refers to one convolutional kernel of a given key layer.

Algorithm 1 Alternating Optimization for Multiple Quantization of high MSE layer

Input: Tensor T, desired precision $\{p_i\}_{i=1}^n$ and quantization mapping ϕ.
Output: $\{\alpha_i\}_{i=1}^n$ and $\{\tilde{T}_i\}_{i=1}^n$

```
while convergence rate > ε do
    for j ∈ [1, ..., n] do
        (α_j, T̃^j) = φ(T − Σ_{i=1,i≠j}^n α_i T̃^i, p_j)
    end for
end while
```

For the special *dual* case where $n = 2$, optimum can be obtained efficiently by resorting to dual line search. Since each element in $\tilde{T}^1$ can only have discrete values in $\mathbb{Z}_{p_1}$, one can evaluate $\tilde{T}^2$ for all the few 2^{p_1} allowed quantized values and later select the integer value giving minimum of the objective. Assuming α_1, α_2 are given via grid search, we want to find $\forall j$

$$\begin{aligned} &\min_{\tilde{T}^1_j, \tilde{T}^2_j} && (T_j - \alpha_1 \tilde{T}^1_j - \alpha_2 \tilde{T}^2_j)^2 \\ \Longleftrightarrow &\min_{\tilde{t} \in \mathbb{Z}_{p_1}} \min_{\tilde{T}^2_j} && (T_j - \alpha_1 \tilde{t} - \alpha_2 \tilde{T}^2_j)^2 \\ \Longleftrightarrow &\min_{\tilde{t} \in \mathbb{Z}_{p_1}} && \left(T_j - \alpha_1 \tilde{t} - \alpha_2 \left[\frac{T_j - \alpha_1 \tilde{t}}{\alpha_2}\right]_{\mathbb{Z}_{p_2}}\right)^2 \end{aligned} \tag{12}$$

Because of its high computational cost, this method should be reserved to small tensors only (convolutional kernels). As in Section 3.4 the proposed dual line search procedure is preferred to the alternating approach due to the high non-convexity of the optimization problem (9). In our experiments, dual line-search approach improves the MSE by 5x in average over the tested dual layers.

Layers with high MSE that are approximated using multiple quantized tensors obviously require more parameters and computations, so a trade-off between accuracy and performance can be established. Defining speedup is not trivial since it is highly dependent on the hardware design. Modern GPUs can double the TOPS performance at INT4 precision, since multiple *small* low precision matrix multiplications can be performed in a distributed fashion. In order to measure the increase in storage and computations we define the compression ratio for analysis of the framework. The compression ratio $0 < \text{CR} \leq 1$ is defined as the ratio of the quantized model (weights and scaling factors) size to the FP32 model size. In this work, difficult layers in the INT4 setting are approximated with the dual method only ($n = 2$). Performance of the proposed algorithm as well as the corresponding compression ratios are summarized in Table 3.

3.6. Scaling Factors Refinement

One issue with the quantization method proposed above is the rigidity of the quantization mapping performed axiomatically according to a given metric. In order to tackle this problem without the need for full retraining and without requiring gradients of non differentiable functions, we propose a post quantization adjustment of the scaling factors of the quantized NN weights. Given calibration data (unlabeled), we refine the scaling factors to better approximate the full precision model. We seek to optimize the (re)scaling factor γ defined as $\hat{T}(\gamma) = \gamma \alpha \tilde{T}$, with $\tilde{T}$ being the tensor quantized axiomatically. Consequently, the saturation threshold α is approximated for optimal reconstruction separately, using L_2 metric. The rescaling factor γ is optimized afterwards in a data driven way such that

$$\hat{T}(\gamma) = \gamma \left(\alpha \left[\frac{T}{\alpha}\right]_{\mathbb{Z}_p}\right). \tag{13}$$

Assuming $f(X, \{W_l\}_l)$ is the NN mapping function, we seek to minimize the regression (MSE) problem

$$\min_{\gamma_l = \{\gamma_{lk}\}_{lk}} \sum_i^M \| f(X_i, \{W_l\}_l) - f(X_i, \{\hat{W}_l(\gamma_l)\}_l) \|_F^2, \tag{14}$$

where M is the size of the calibration set. The advantage of this approach is that the number of optimized values is small and equal to the number of convolutional kernels. At contrary to training methods, this approach is fully *differentiable* avoiding sub-gradient definitions [3]. Thus, the optimization can be conducted very efficiently using popular stochastic gradient descent methods on a *small* calibration set as in [29]. Also, only a few optimization steps are required for both fast deployment and better *generalization*. Results of the refinement procedure, and its influence on

Architecture	Original	OMSE	Dual	CR
Alexnet	56.624%	46.892%	**54.408%**	0.1256
VGG16bn	73.476%	65.414%	**66.932%**	0.125
Inception v3	76.226%	22.028%	**51.642%**	0.1263
Resnet18	69.644%	56.688%	**64.034%**	0.1318
Resnet50	76.012%	67.356%	**70.060%**	0.1261
Resnet101	77.314%	65.066%	**71.492%**	0.1261
SqueezeNet	58.0%	32.262%	**53.466%**	0.1493
DenseNet	74.472%	59.012%	**64.400%**	0.1432

Table 3. Top1 accuracy using INT4 optimal MSE quantization of weights (FP32 activations) and INT4 dual MSE (Dual) for $\tau = 8 \cdot 10^{-5}$. The last column defines the compression ratio (CR) induced by the dual method. CR of regular (non-dual) INT4 linear quantization is ~ 0.125.

Architecture	Dual	OMSE+Opt.	Dual+Opt.
Alexnet	54.408%	53.306%	**55.226 %**
VGG16bn	66.932%	72.294%	**72.576%**
Inception v3	51.642%	73.656%	**74.790%**
Resnet18	64.034%	67.120%	**68.806%**
Resnet50	70.060%	74.672%	**74.976%**
Resnet101	71.492%	76.226%	**76.402%**
SqueezeNet	53.466%	54.514%	**56.248%**
DenseNet	64.400%	71.730%	**73.600%**

Table 4. Top1 accuracy using FP32 activations and INT4 dual MSE quantization of weights (Dual) and dual MSE + refined scaling factors (Dual+Opt.) with $\tau = 8 \cdot 10^{-5}$. As ablation study we present the refinement of the OMSE method without dual quantization (OMSE+Opt). The calibration set contains five hundred images from the ImageNet validation set and 25 epochs.

Architecture	Original	Baseline [20]	KLD [29]	OMSE	Dual (CR)	ACIQ* [2]	OMSE*	Dual*
Alexnet	56.624%	38.616%	46.002%	**48.908%**	**54.552%** (0.195)	-	**49.122%**	**54.994%**
Alexnet (offset)	56.624%	51.774%	53.198%	**53.286%**	**55.522%** (0.195)	52.304%	**53.998%**	**55.508%**
VGG16bn	73.476%	33.276%	59.026%	**62.168%**	**68.120%** (0.127)	-	**67.726%**	**68.334%**
VGG16bn (offset)	73.476%	58.832%	65.558%	**67.198%**	**71.478%** (0.127)	67.716%	**71.260%**	**71.260%**
Inception v3	76.226%	4.042 %	32.516 %	**40.916%**	**66.176%** (0.154)	-	**43.184%**	**68.608%**
Inception v3 (offset)	76.226%	57.516 %	67.640%	**67.964%**	**73.060%** (0.151)	59.826%	**69.528%**	**74.486%**
Resnet18	69.644%	48.37%	59.002%	**61.268%**	**66.522%** (0.150)	-	**63.744%**	**66.628%**
Resnet18 (offset)	69.644%	63.106%	64.952%	**64.992%**	**68.380%** (0.148)	65.694%	**67.508%**	**68.316%**
Resnet50	76.012%	40.786%	60.326%	**64.878%**	**70.368%** (0.129)	-	**66.562%**	**70.202%**
Resnet50 (offset)	76.012%	65.338%	69.160%	**71.274%**	**73.252%** (0.126)	71.362%	**73.392%**	**73.392%**
Resnet101	77.314%	36.494%	59.002%	**65.316%**	**70.770%** (0.129)	-	**65.350%**	**70.806%**
Resnet101 (offset)	77.314%	65.552%	69.746%	**72.750%**	**74.266%** (0.126)	69.544%	**74.332%**	**74.332%**
SqueezeNet	58.0%	3.282%	12.220%	**18.630%**	**52.382%** (0.216)	-	**20.448%**	**52.430%**
SqueezeNet (offset)	58.0%	24.97%	37.080%	**39.820%**	**56.150%** (0.203)	-	**42.026%**	**56.284%**
DenseNet	74.472%	47.808%	64.990%	**65.032%**	**67.952%** (0.132)	-	**67.558%**	**68.048%**
DenseNet (offset)	74.472%	67.676%	69.640%	**70.118%**	**72.282%** (0.132)	-	**72.304%**	**72.310%**

Table 5. Top1 accuracy using INT4 MSE quantization of activations, INT8 weights uniformly quantized and dual MSE ($\tau = 8 \cdot 10^{-5}$) compared with several methods. The baseline is defined as in [20] with clipping to the mean maximum and minimum range of the calibration set. KLD denotes the saturation technique based on Kullback-Leibler divergence metric proposed in [29]. In ACIQ framework [2] the first layer of the models is not quantized and unsigned representation is used. ACIQ results are taken from the paper. We provide similar to ACIQ experiment with our methods (OMSE* and Dual*) wherein we do not quantize the input layer (if first layer is the only key layer, OMSE* and Dual* yield same results). The calibration set is composed of 250 images for all the proposed methods.

accuracy improvement are presented in Table 4. These experiments show that the refinement procedure can improve the accuracy by up to 23%. We also provide ablation study to demonstrate the usefulness of the refinement stage in low compression ratio tasks where no dual layers are allowed.

3.7. Activations Quantization

In contrast to weights quantization, the activations quantization should be performed for each image on the fly. Obviously, optimal scaling factors of activations cannot be computed for every image. Thus, the quantization parameters are generally approximated using relatively small calibration set [29, 2] where saturating the activation values to a given threshold helps in improving accuracy. The saturation problem is usually solved by approximating the activations *distribution* [29, 2]. These methods are less sensitive to *outliers* with potential important impact on the underlying dot product. Here, we follow the same MSE minimization approach, wherein the parameters are obtained as described in Section 3.4. Thereby, given a small calibration set (few hundred data samples) we collect activations at different layers of the network, and seek for optimal saturation factor. Given $X_l \in \mathbb{R}^{d,c_l,w_l,h_l}$ the l^{th} layer activations with d the size of the calibration set, we solve

$$\min_{\beta_1} \quad \|X_l - \beta_1 \left[\frac{X_l}{\beta_1}\right]_{\mathbb{Z}_p} \|_F^2. \tag{15}$$

For key layers, optimal approximation using multiple quantizations as described in Section 3.5 suffers from strong overfitting. In order to approximate better, and to generalize at the same time, we obtain the optimal parameters by quantizing the *residual* from the first approximation, such that for a given activation X we have

$$\begin{aligned} X \approx \hat{X} &= \beta_1 \tilde{X}_1 + \beta_2 \tilde{X}_2 \\ &= \beta_1 \left[\frac{X}{\beta_1}\right]_{\mathbb{Z}_p} + \beta_2 \left[\frac{X - \beta_1 \tilde{X}_1}{\beta_2}\right]_{\mathbb{Z}_p}. \end{aligned} \tag{16}$$

We obtain β_1 from eq.(15), followed by obtaining β_2 from eq.(16). This procedure is similar to the alternating approach described in Algorithm 1 with one iteration only. Similar to the network weight compression ratio, we define the activations compression ratio as the ratio between the size of all the compressed activations and the size of all the original activations.

Table 5 presents the comparison of the variants of the pro-

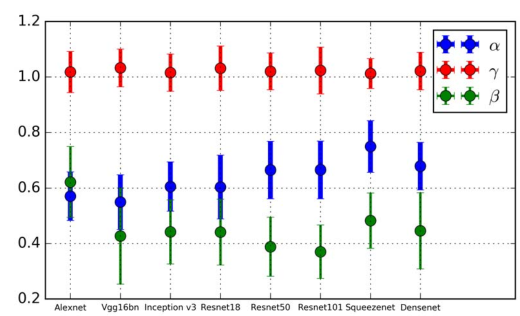

Figure 2. Mean and standard deviation of the set of scaling factors of the weights (α), the activations (β) on the calibration set (not dual) and the refined factors (γ), all normalized by the maximum value of the tensor they approximate.

Architecture	Original		KLD [29]		Our		Our+offset		CR(W,A)	% Dual
Alexnet	56.624%	79.056%	35.590%	59.326%	53.632%	77.244%	**54.476%**	**77.846%**	(0.125,0.195)	25%
VGG16bn	73.476%	91.536%	41.530%	66.008%	67.492%	88.016%	**70.658%**	**90.136%**	(0.125,0.127)	6.25%
Resnet18	69.644%	88.982%	31.934%	55.510%	65.456%	86.630%	**67.416%**	**87.716%**	(0.126,0.148)	23.80%
Resnet50	76.012%	92.934%	46.190%	70.162%	69.370%	89.204%	**72.602%**	**90.852%**	(0.126,0.129)	3.70%
Resnet101	77.314%	93.556%	49.948%	73.034%	69.700%	89.686%	**73.602%**	**91.526%**	(0.126,0.128)	1.90%
Inception v3	76.226%	92.984%	1.84%	4.848%	64.572%	85.852%	**71.606%**	**90.470%**	(0.126,0.154)	11.57%
SqueezeNet	58.184%	80.514%	8.224%	19.348%	50.722%	74.634%	**55.358%**	**78.482%**	(0.149,0.216)	68%
DenseNet	74.472%	91.974%	44.05%	68.52%	66.832%	87.518%	**71.558%**	**90.532%**	(0.143,0.132)	2.47%
SSD300	77.43%		47.38%		73.94%		**75.77%**		(0.125,0.126)	2.85%

Table 6. Performance of the proposed framework for INT4 weights and activations obtained using $\tau = 8 \cdot 10^{-5}$. Compression ratios of both the weights and the activations and the percentage of dual layers are provided in the last two columns, respectively. The mean top-1 decay is of 6.7% and 3% for the regular and offset (unsigned) versions, respectively. Similarly, mean top-5 decay is of 4% and 1.7% respectively. We present in the last row the quantization of the SSD object detector [27], with mAP performance on PASCAL VOC2007 dataset.

posed activation quantization method with several existing methods. The line-search method iterates over 50 samples only, compared to more than thousand for other methods [29, 20]. Statistics of thresholding factors for different models are presented in Figure 2 where we can observe the severe saturation induced by the MMSE metric.

4. Experiments

We evaluate the proposed framework on several popular architectures in order to demonstrate its robustness to aggressive quantization. We also evaluate quantization of non over-parameterized models such as SqueezeNet or DenseNet that are much more sensitive to quantization and are usually not analyzed in the NN quantization literature. We consider different INT4 quantization scenarios such as unsigned (with offset) and signed representations, and we present the accuracy-compression ratio trade-off analysis.

The framework has been implemented using the Pytorch library with its pretrained models. The MSE grid search iterates over 500 sampling points for the weights and 50 for the activations. The scaling factors refinement stage requires 25 epochs over 500 randomly sampled images from the validation set. The calibration set used for the quantization of activations is comprised of 250 images. *All* the layers are quantized, including the network input. Also, it is important to notice that there is currently no existing framework for linear INT4 only quantization of weights and activations for comparison in the inference setting. TensorRT [29] is currently the best known framework for post-training quantization and it sets a baseline for our method. The summary of the accuracy loss and the compression ratios is presented in Table 6, where we show the proposed method achieves a new state-of-the-art performance in post-training quantization by a large margin. In Table 7 we show how modification of the MSE threshold τ influences the accuracy-compression ratio trade-off.

Architecture	Original	Our	CR(W,A)	Our +offset
Alexnet	56.624%	49.570%	(0.125,0.178)	52.274%
		54.186%	(0.127,0.206)	54.936%
		54.720%	(0.130,0.245)	56.068%
Resnet50	76.012%	69.472%	(0.126,0.129)	72.530%
		73.078%	(0.137,0.156)	74.826%
		74.336%	(0.154,0.203)	75.198%
Inception v3	76.226%	61.856%	(0.126,0.146)	71.662%
		74.036%	(0.150,0.215)	75.354%
		74.766%	(0.186,0.238)	75.870%
DenseNet	74.472%	65.526%	(0.126,0.132)	70.730%
		70.716%	(0.155,0.154)	73.114%
		73.382%	(0.226,0.241)	74.116%

Table 7. Trade-off analysis of the proposed method without and with offset for three thresholding values $\tau = \{10, 2, 0.9\} \cdot 10^{-5}$. The compression ratio refers to the method without offset.

5. Conclusion

In this paper we introduced an efficient and accurate MSE-based low-bit precision quantization framework for neural networks. We analyze the MMSE quantization problem and propose optimal quantization for minimal loss of accuracy. The hardware-aware partitioning of the network parameters, and the refinement of high MSE layers using quantized filter banks, enables improved performance while remaining compliant with modern low power hardware. Given a small calibration set, we further refine the quantization scaling factors for better approximation of the original model. We also provide a framework for the quantization of the network activations, wherein we propose a method of residual quantization for improved approximation of the most sensitive layers. The proposed approach can be adjusted to any desired precision for constrained hardware deployment, according to the inherent compression-complexity trade-off of the method. The framework allows fast and efficient deployment of pretrained models, producing a new state-of-the-art INT4 inference quantization results.

References

[1] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720, 2017. 4

[2] R. Banner, Y. Nahshan, E. Hoffer, and D. Soudry. Aciq: Analytical clipping for integer quantization of neural networks. *arXiv preprint arXiv:1810.05723*, 2018. 2, 7

[3] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 2, 6

[4] M. Courbariaux, Y. Bengio, and J.-P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015. 2

[5] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016. 2

[6] W. Dally. High-performance hardware for machine learning. In *Tutorial in Advances in Neural Information Processing Systems*, 2015. 1

[7] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pages 1269–1277, 2014. 1

[8] J. Faraone, N. Fraser, M. Blott, and P. H. Leong. Syq: Learning symmetric quantization for efficient deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4300–4309, 2018. 4

[9] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011. 3

[10] Y. Gong, L. Liu, M. Yang, and L. Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014. 2

[11] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015. 1

[12] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015. 1, 3

[13] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[14] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012. 1

[15] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1

[16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1

[17] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269. IEEE, 2017. 1, 4

[18] K. Hwang and W. Sung. Fixed-point feedforward deep neural network design using weights+ 1, 0, and- 1. In *Signal Processing Systems (SiPS), 2014 IEEE Workshop on*, pages 1–6. IEEE, 2014. 2, 5

[19] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 1, 4

[20] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. *arXiv preprint arXiv:1712.05877*, 2017. 2, 4, 5, 7, 8

[21] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Computer Architecture (ISCA), 2017 ACM/IEEE 44th Annual International Symposium on*, pages 1–12. IEEE, 2017. 4

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1, 4

[23] F. Li and B. Liu. Ternary weight networks. *CoRR*, abs/1605.04711, 2016. 2

[24] Z. Li, B. Ni, W. Zhang, X. Yang, and W. Gao. Performance guaranteed network acceleration via high-order residual quantization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2603–2611. IEEE, 2017. 2

[25] D. Lin, S. Talathi, and S. Annapureddy. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning*, pages 2849–2858, 2016. 4, 5

[26] X. Lin, C. Zhao, and W. Pan. Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems*, pages 345–353, 2017. 5

[27] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 8

[28] N. Mellempudi, A. Kundu, D. Mudigere, D. Das, B. Kaul, and P. Dubey. Ternary neural networks with fine-grained quantization. *arXiv preprint arXiv:1705.01462*, 2017. 4

[29] S. Migacz. 8-bits inference with tensorrt. In *GPU Technology Conference*, 2017. 2, 4, 5, 6, 7, 8

[30] A. Mishra, E. Nurvitadhi, J. J. Cook, and D. Marr. WRPN: Wide reduced-precision networks. In *International Conference on Learning Representations*, 2018. 5

[31] A. Polino, R. Pascanu, and D. Alistarh. Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*, 2018. 4

[32] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016. 2, 3

[33] O. Shayar, D. Levi, and E. Fetaya. Learning discrete weights using the local reparameterization trick. *arXiv preprint arXiv:1710.07739*, 2017. 2

[34] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 4

[35] D. Soudry, I. Hubara, and R. Meir. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In *Advances in Neural Information Processing Systems*, pages 963–971, 2014. 2

[36] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014. 1

[37] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 4

[38] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016. 2, 3, 4