Training and Inference for Integer-Based Semantic Segmentation Network

Jiayi Yang[1], Lei Deng[2], Yukuan Yang[3], Yuan Xie[2], Guoqi Li[3*]
[1]International School, Beijing University of Posts and Telecommunications, Beijing 100876, China.
[2]Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106, USA.
[3]Department of Precision Instrument, Center for Brain Inspired Computing Research, Tsinghua University, Beijing 100084, China.

***Abstract*—Semantic segmentation has been a major topic in research and industry in recent years. However, due to the computation complexity of pixel-wise prediction and backpropagation algorithm, semantic segmentation has been demanding in computation resources, resulting in slow training and inference speed and large storage space to store models. Existing schemes that speed up segmentation network change the network structure and come with noticeable accuracy degradation. However, neural network quantization can be used to reduce computation load while maintaining comparable accuracy and original network structure. Semantic segmentation networks are different from traditional deep convolutional neural networks (DCNNs) in many ways, and this topic has not been thoroughly explored in existing works. In this paper, we propose a new quantization framework for training and inference of segmentation networks, where parameters and operations are constrained to 8-bit integer-based values for the first time. To maintain the ability to perform inference on fixed-point devices, we further replace traditional L2 batch normalization with L1 batch normalization. Our proposed framework is evaluated on mainstream semantic segmentation networks like FCN-VGG16 and DeepLabv3-ResNet50, achieving comparable accuracy against floating-point framework on ADE20K dataset and PASCAL VOC 2012 dataset.**

Keywords: Neural Network Quantization, Semantic Segmentation, Fully Convolutional Network

I. Introduction

Semantic segmentation has been a major topic of research in deep learning since the development of Fully Convolutional Network [12], and recent models such as PSPNet [24], DeepLab series [1] [2] [3] [4] have achieved considerably high accuracies on public datasets. However, semantic segmentation suffers from huge computation cost and storage space because of its pixel-wise prediction. For instance, consider a ResNet50 DCNN and a DeepLab-ResNet50 segmentation network: it takes around 2 ms to train one image in ResNet50 and 25 ms for DeepLab with same input size 224×224 on an Nvidia Titan V GPU. To generate a full-size semantic segmentation prediction, it takes much longer than to expect a single classification result in DCNN. This property makes shifting from traditional segmentation to real-time segmentation suffering.

Regarding this problem, recent work on real-time semantic segmentation networks [13] [23] [22] often design new network structure to trade off accuracy for inference speed. To achieve higher efficiency while maintaining similar accuracy calls for other methods than to design new network structures. Recent works on neural network quantization managed to lower the bit-width of dataflow while maintaining accuracy. The quantized networks restrict parameters and computation to lower bits, and simulate full precision training and inference with discrete dataflow.

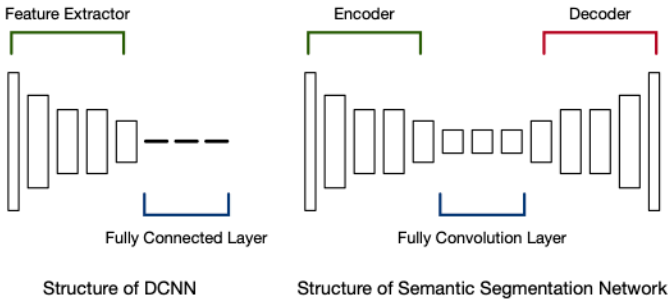


Fig. 1: Difference of general structure between DCNN and semantic segmentation networks.

However, major works on network quantization like BNN [5], WAGE [21] mainly explore quantization on DCNN designed for object classification. Due to the complexity of pixel-wise prediction, semantic segmentation tasks have to use deeper networks. This causes some quantization methods in DCNN with fewer layers not suitable for semantic segmentation. Furthermore, existing works of quantization on semantic segmentation task [17] [18] does not fully quantize all the parameters in the network, making it hard to be implemented on integer-based deep learning chip, or to generalize to other existing models that are not dedicatedly quantized.

The general structure comparison between DCNN and semantic segmentation network are shown in Figure 1. First, it removes the fully connected layers from the picture and replaces it with convolution layers. Second, as DCNN is the encoder of the semantic segmentation network, there also exists a decoder structure that recovers the feature map dimension. Third, segmentation network weights are initialized differently. The encoder is usually initialized with pre-trained weight from DCNN, and the decoder part is initialized in

*Corresponding author. Email addresses: markyang@bupt.edu.cn (J. Yang), leideng@ucsb.edu (L. Deng), yyk17@mails.tsinghua.edu.cn (Y. Yang), yuanxie@ece.ucsb.edu (Y. Xie), liguoqi@mail.tsinghua.edu.cn (G. Li).

differently in various networks. Therefore, to quantize semantic segmentation network is not so intuitive as implementing quantization frameworks on DCNN.

Hence, we propose a new framework that quantizes deep semantic segmentation network into integer-based dataflow, constraining nodes into low bit discrete space. In addition to weight (W), activation (A), error (E), gradient (G) and update (U), we address batch normalization [11] to train deeper models, which is often neglected or replaced in previous works on quantization. While batch normalization helps the training of deeper network and speeds up convergence, it also contributes to a large portion of computation on run time, and the nonlinearity introduced by square and root operations makes it difficult to be quantized. Fortunately, L1-norm batch normalization (L1BN) [20] is proven to be mathematically equivalent to L2-norm batch normalization (L2BN) but demonstrates stronger linearity. Consequently, we quantized L1BN in our network instead of L2BN, therefore endowing it the ability to run on integer-based hardware, which falls short on computing square root operations.

Our proposed framework is evaluated on ADE20K dataset [25] and PASCAL VOC 2012 datasets [8] on mainstream segmentation network FCN and DeepLabv3. We achieve comparable accuracy compared with full precision networks when constraining major dataflow into 8 bits integers, striking a balance between bit-width and performance. We conduct experiments and analysis on quantized semantic segmentation network from different perspectives (e.g. bit-width, quantization details), providing some insights for further adaptation on other segmentation networks.

In summary, our contributions are:

- We propose a framework for semantic segmentation network that constrains the major dataflow of training and inference to 8-bit integers.
- Our framework achieves comparable performance as the full-precision network. Designed for general segmentation network, our framework leaves room for application on other models for further work.
- We perform a thorough analysis and experiment of different factors impacting of performance of semantic segmentation network.

II. Related Work

In this paper, we focus on how to reduce the computation load on both training and inference through reducing precisions of operations and operands, further replacing Batch Normalization (BN) with its more efficient counterpart.

A. Semantic Segmentation

Many models have been developed since the development of Fully Convolutional Network. In this work, we experiment on two classic networks that can represent most of the mainstream networks used today.

Fully Convolutional Network: Fully convolutional network represents the broad class of network for semantic segmentation. It replaces fully connected layers in traditional CNN with convolution layers and appends a decoder that recovers feature map resolution to produce pixel-wise semantic segmentation prediction. Although it is more efficient to predict one label, as its output is a pixel-wise label, the computation overhead is huge. In this paper, we use the classic FCN8s with VGG16 [16] as it's encoder in the experiments.

DeepLab: DeepLab architecture has been one of the most accurate models for the task of semantic segmentation since the work of FCN. It mainly contains a DCNN backbone appended with an atrous spatial pyramid pooling(ASPP) module. DenseCRF was removed from the framework to maintain simplicity. Most recent DeepLabv3 [3] and DeepLabv3+ have achieved state-of-the-art accuracy on public datasets.

B. Network Quantization

Neural network quantization [6] [14] [7] is the approach of trying to reduce the model size and accelerate computation by reducing the bit-width of the operands in the network. In recent works, *Wu et al.* [21] proposed WAGE to discretize parameters in both training and inference. They identify weight (W), activation (A) error (E) and gradient (G) in forward and backward propagation, and constrain them to low bit-width integers. To simulate BN, they replaced it by a layer-wise constant scaling factor. However, WAGE's experiments were carried on shallow CNNs, and the performance on deeper networks are not satisfying. Other works like 8b Training [19] and FX Training [15] push quantization framework to deeper CNNs, though BN is rarely mentioned in these frameworks. To sum up, despite the thriving research of quantization in traditional CNN, there is still room to quantize in the dataflow of neural networks. Moreover, to quantize semantic segmentation network is not as intuitive due to the structural and detail differences with DCNN.

III. Network Quantization

Quantization method and quantization object are two important factors when considering network quantization. Different objects may use different methods depending on their data distribution and property. In this section, we first identify the quantization methods. Then, we introduce the entire framework in the order of forward pass and then back pass.

A. Quantization Methods and Distribution Analysis

1) Uniform Quantization: Uniform quantization is the basis of other quantization methods. The minimum quantization distance is governed by

$$\Delta(k) = 2^{1-k} \tag{1}$$

where k is the quantization bit-width. Uniform quantization is similar to ADCs used in signal processing, which deterministically map the floating-point values into nearest sdiscrete state. Uniform quantization is defined as

$$UQ(x,k) = Clip(\Delta * \lfloor \frac{x}{\Delta} + \frac{1}{2} \rfloor, -1+\Delta, 1-\Delta) \tag{2}$$

where x is the quantization target and Δ is the quantization distance with respect to parameter k. $Clip(\cdot)$ function clips

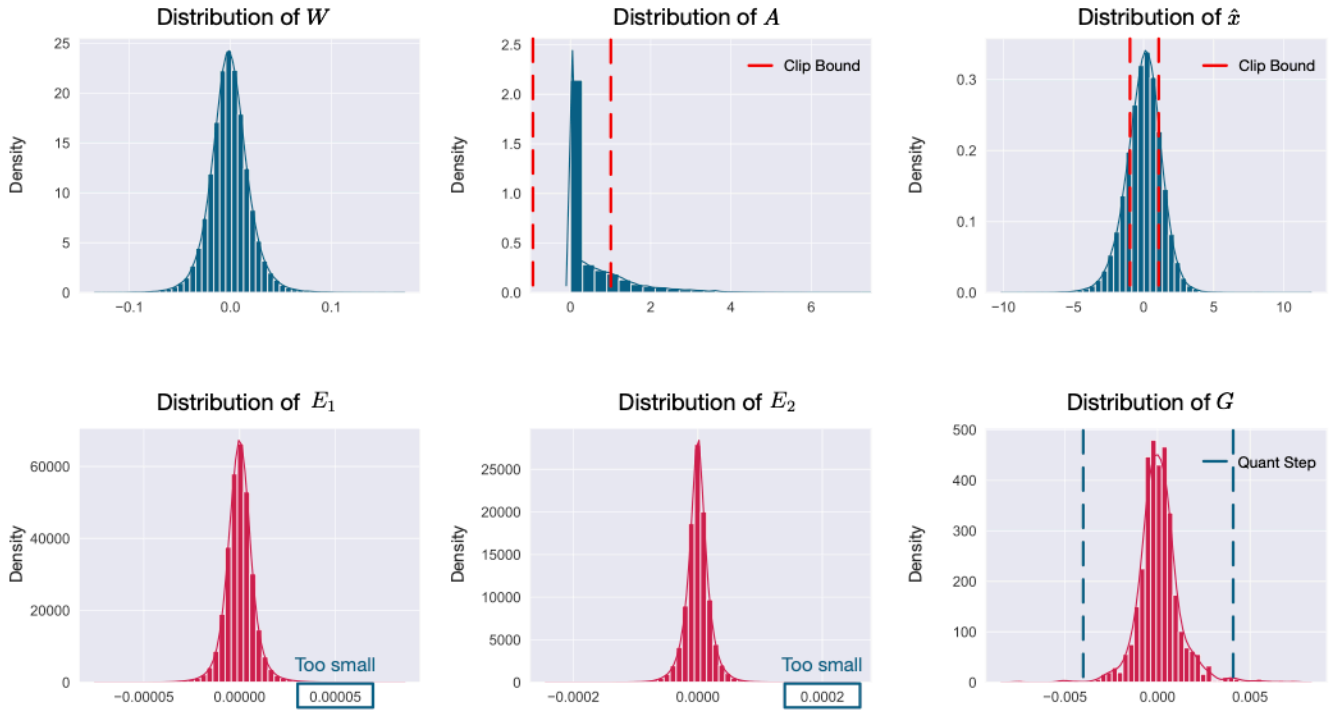


Fig. 3: Distribution comparison of various quantization objects. The distribution in blue represents forward propagation dataflow, and the distribution in red represents backward propagation.

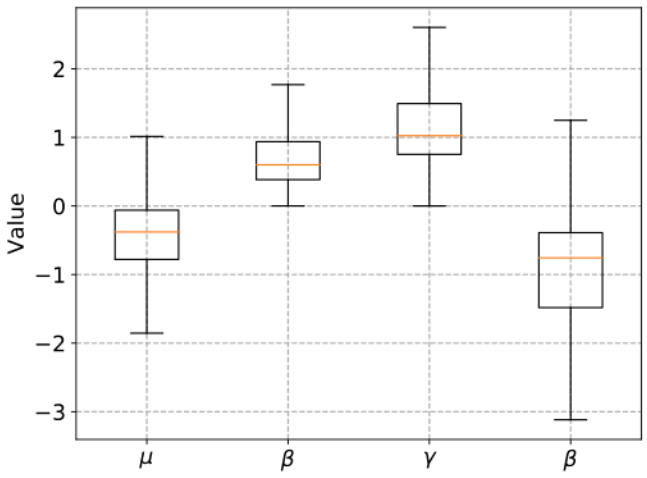


Fig. 2: Quartile box graph of σ, μ, γ and β. The majority of value lies between $[-2, 2]$.

the quantized value inside the quantization range between $[-1 + \Delta, 1 - \Delta]$.

Quantization objects for semantic segmentation networks without BN include weight (W), activation (A), error (E_1), gradient (G) and update (U). E_1 is defined as the gradient of activation, and G is the gradient of weight. For networks with BN, variance (σ), mean (μ), normalized output ($\hat{x}$), scale (γ), shift (β) and error (E_2) is included. The additional E_2 is defined as the gradient of normalized output in BN.

Figure 2 shows the distribution of σ, μ, γ and β. Experiments reveal that these values lie stably inside the range of$[-2, 2]$ throughout the training process, and the therefore we perform constant scaling of 2 and 2^{-1} before and after uniform quantization to fit the quantization range of $[-1 + \Delta, 1 - \Delta]$.

2) Scale Quantization: However, quantization for W, A, E_1, E_2, G and normalized output X is not as straightforward. Figure 3 illustrates the distribution of these objects in full-precision DeepLabv3-ResNet50 network during training. Here we identify two failure modes considering uniform quantization $UQ(\cdot)$:

- **First** failure mode is when the distribution is concentrated between the smallest quantization step $\Delta(k)$.
- **Second** failure mode is when considerable portion of weights are clipped by quantization boundary.

As all the quantization object distribution mentioned in Figure 3 except W belongs to one of these two modes, the framework suffers performance degradation when using uniform quantization.

Scale quantization can be applied to escape from these two failure modes. An important component in scale quantization is the dynamic sacaling factor. Various scaling factors have been used in previous works, but our experiments show that the naive dynamic scale factor

$$Scale(x) = max(|x|) \tag{3}$$

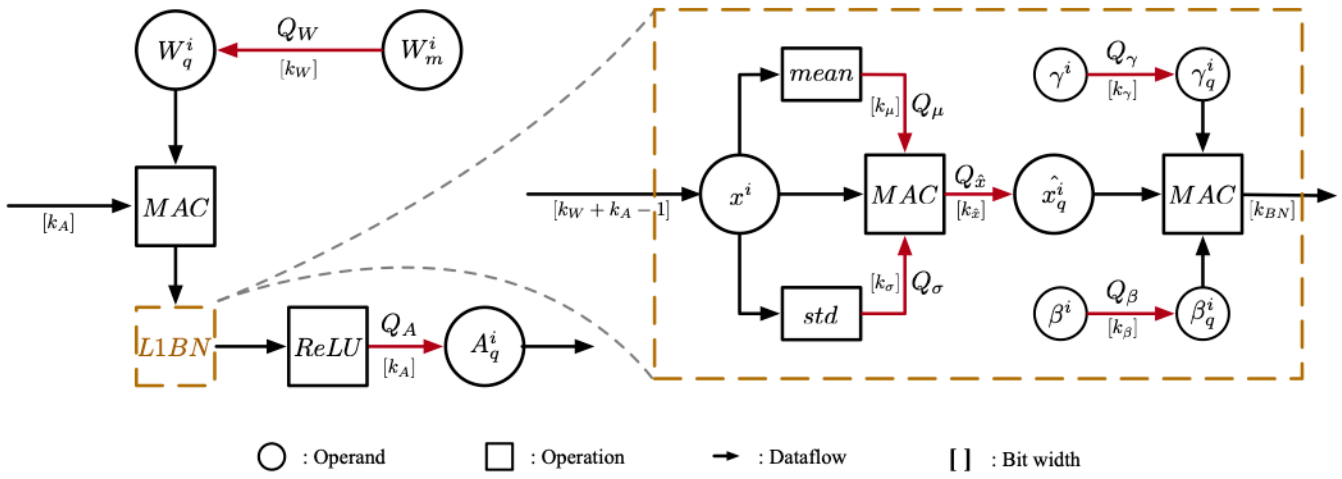


Fig. 4: Dataflow and quantization framework in forward propagation. The red arrows indicate the quantization and dataflow of discrete parameters. L1BN module is removable from the quantization framework.

does not differ with in terms of semantic segmentation network performance. For any object x, most of the normalized value in $\frac{x}{Scale(x)}$ will lie in the quantization range $[-1+\Delta, 1-\Delta]$. The scale quantization equation is given as

$$SQ(x,k) = Scale(x) * UQ(\frac{x}{Scale(x)}, k) \tag{4}$$

where x will be scaled by $Scale(x)$ before quantization to fit data range and scaled back after quantization to preserve its original order of magnitude. ***A* and $\hat{x}$ distribution**. The distribution of activation and normalized output is far different from weight. They both fall into the **second** mode by exceeding the quantization range. The values clipped by function $Clip(\cdot)$ will result in information loss that affects training during loss calculation. Hence, we use scale quantization for activation and normalized output in BN to ensure network convergence.

***E_1* and *E_2* distribution**. Both error objects fall into the **first** failure mode where the order of magnitude is smaller than the quantization step Δ. Suppose that bit-width $k_E = 8$, the quantization step $\Delta(k_E) \approx 0.008$ far exceeds the order of magnitude of error in $[10^{-5}, 10^{-4}]$ range. This will result in error propagation with all the value rounded to zero. Therefore we also apply scale quantization given in Equation (4) for error in the dataflow.

***G* distribution**. Gradient have similar distribution when compared to error, but through experiments, we found that network performance drops noticeably when rounding to around 8 bit. This suggests gradient is more sensitive to bit-width and preservation of small values. To solve this problem, we round gradients stochastically in place of $UQ(\cdot)$.

3) Stochastic Quantization: Suppose we have a properly scaled quantization object x, the definition of stochastic quantization is governed by

$$RQ(x,k) = Clip(\Delta * Round(\frac{x}{\Delta}), -1+\Delta, 1-\Delta) \tag{5}$$

where Δ denotes the quantization step according to k defined in Equation (1). $RQ(\cdot)$ largely resembles uniform quantization, except that we round the values stochastically using

$$Round(x) = \begin{cases} \lfloor x \rfloor & P = \lceil x \rceil - x \\ \lfloor x \rfloor + 1 & P = x - \lfloor x \rfloor \end{cases}. \tag{6}$$

The floating-point number x will be rounded to its two neighbor integers stochastically, and the probability P correlates to the distance to that integer linearly. The expectation of stochastic rounding $\mathbb{E}(Round(x)) = x$, resulting in no expected error compared to deterministic rounding. Further, stochastic rounding feeds additional probability information into the dataflow, and gradients inside the smallest quantization step can also be rounded to non-zero values. Details of gradient quantization will be covered in Section III-C2.

B. Forward Propagation

1) Weight Quantization: Initialization of weight in semantic segmentation network is different from DCNN. The encoder structure is initialized with full-precision weight trained from object classification tasks like ImageNet. Fully convolution layers are initialized with MSRA initialization [10], and decoder is initialized as bilinear upsampling filter [12]. Since we train the network with discrete dataflow, initialized weights must also be in discrete state by

$$W_m^i = UQ(W^i, k_U) \tag{7}$$

with weight update bit-width k_U. We also discover that the standard initialization approach converges well for fully convolution layers and decoder structure, and other initialization

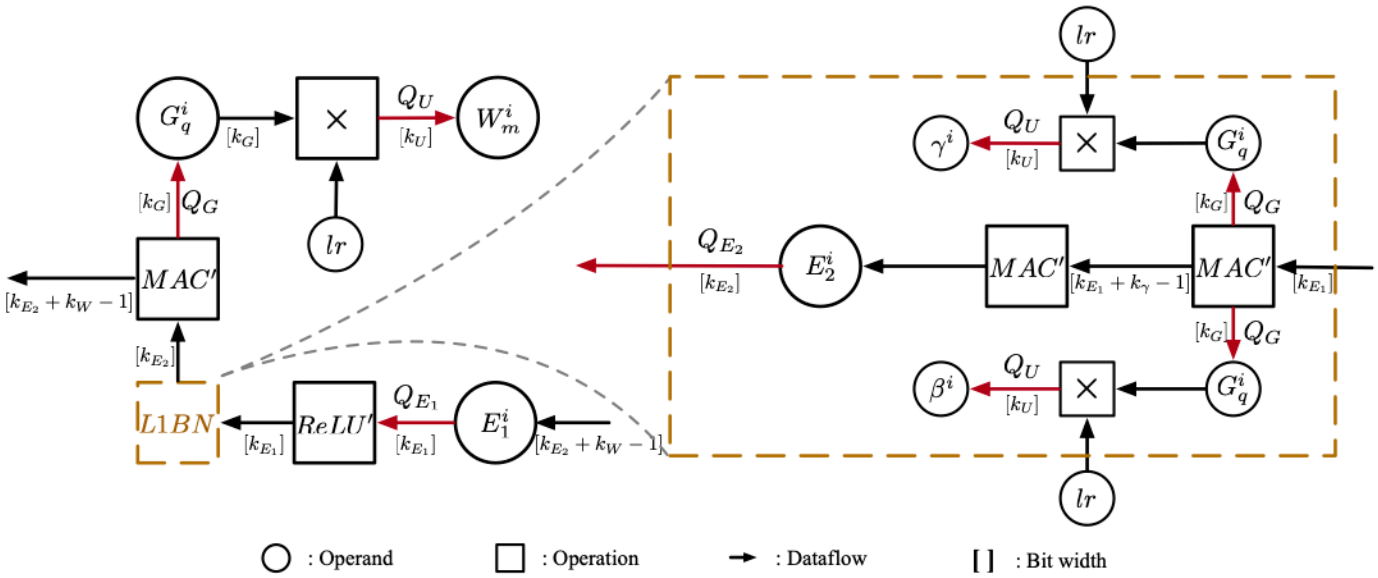


Fig. 5: Dataflow and quantization framework in back propagation. The red arrows indicates the quantization and dataflow of discrete parameters. L1BN module is removable from the quantization framework.

methods do not contribute to noticeable speed up in convergence nor higher accuracy.

During forward propagation, weights are directly quantized with uniform quantization

$$W_q^i = Q_W(W_m^i) = UQ(W_m^i, k_W) \tag{8}$$

where k_W denotes the quantization bit-width of weights, and W_m^i denotes the master weights stored in the network. Note that master weights are initialized and stored with update bit-width k_U, covered with details in Section III-C2. As shown in Figure 4, the master weights stored with k_U bit is first quantized to k_W bit by Q_W before feeding into the convolution layer.

2) L1BN Quantization: Batch normalization (BN), serving as an important normalizing method, are not often addressed in existing quantization frameworks. However, we cannot ignore BN if we want to quantize the entire dataflow. Traditional BN with L2-norm can be summarized by

$$\hat{x^i} = \frac{x^i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B} + \epsilon}}, \; y^i = \gamma \hat{x^i} + \beta \tag{9}$$

where the former normalizes the distribution, and the later recovers some representation ability lost from the normalization operation; ϵ demotes a small value added for numerical stability; $\mu_\mathcal{B}$ and $\sigma_\mathcal{B}$ represents the mean and variance calculated from the mini-batch $\mathcal{B}$,where

$$\mu_\mathcal{B} = \frac{1}{m}\sum_{i=1}^{m} x^i \tag{10}$$

and

$$\sigma_\mathcal{B} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x^i - \mu_\mathcal{B})^2} \tag{11}$$

respectively. However, the root and square operations in Equation (9) and (11) brings strong nonlinearity, making it hard to quantize the BN dataflow and implement on low bit-width hardware. Fortunately, while original BN introduces root and square from the L2-norm, it's low order counterpart L1-norm variance demonstrates better linearity [20]. Thus, $\hat{x^i}$ in Equation (9) is replaced with

$$\hat{x^i} = \frac{x^i - \mu_\mathcal{B}}{\sigma_\mathcal{B} + \epsilon} \tag{12}$$

where $\mu_\mathcal{B}$ remains the same, and $\sigma_\mathcal{B}$ replaced with

$$\sigma_\mathcal{B} = \frac{1}{m}\sum_{i=1}^{m} \left| x^i - \mu_\mathcal{B} \right| \tag{13}$$

representing L1-norm variance instead of L2-norm variance.

In L1BN quantization, we identify five quantization objects: $\mu_\mathcal{B}, \sigma_\mathcal{B}, \gamma, \beta$ and $\hat{x^i}$. The additional $\hat{x^i}$ has to be quantized due to the increase of bit-width in the normalization operation in Equation (9). The quantization of BN can be described as

$$\begin{aligned}
\mu_q &= Q_\mu(\mu_\mathcal{B}) = 2 * UQ(\frac{\mu_\mathcal{B}}{2}, k_\mu) \\
\sigma_q &= Q_\sigma(\sigma_\mathcal{B}) = 2 * UQ(\frac{\sigma_\mathcal{B}}{2}, k_\sigma) \\
\hat{x_q^i} &= Q_{\hat{x}}(\hat{x^i}) = SQ(\hat{x^i}, k_{\hat{x}}) \\
\gamma_q &= Q_\gamma(\gamma) = 2 * UQ(\frac{\gamma}{2}, k_\gamma) \\
\beta_q &= Q_\beta(\beta) = 2 * UQ(\frac{\beta}{2}, k_\beta)
\end{aligned} \tag{14}$$

where $k_\mu, k_\sigma, k_\gamma, k_\beta$ and $k_{\hat{x}}$ denotes the bit-width of the quantization objects, respectively. The dataflow of quantized BN is shown in the right part of Figure 4, the bit-width of output is k_{BN}. Depending on the network structure, the L1BN quantization block can be removed without changing other components in the framework.

3) Activation Quantization: After the MAC operations of convolution or scaling in BN (depending on if the network implements BN or not), the precision increases. As a result, the bit-width of activation has to be limited before the input of the next convolution layer. Activation is governed by scale equation where

$$A_q^i = Q_A(A^i) = SQ(A^i, k_A) \tag{15}$$

with k_A representing the bit-width of activations.

C. Backward Propagation

1) Error Quantization: As mentioned earlier, scale quantization is used for objects like error to prevent the values from being zeroed out, then we have

$$E_q^i = Q_E(E^i) = SQ(E^i, k_E) \tag{16}$$

where E^i defined as the gradient of activation in each convolution and deconvolution layer i, denoted as E_1^i in Figure 5. It will be quantized by Q_E with bit-width k_{E_1} and then used for further calculations in the chain rule. For networks with BN, an additional E_2^i should be quantized. The bit-width of error will increase according to the chain rule during backpropagation, so E_2^i is further restricted by Q_E to k_{E_2}, which is related to weight update.

2) Gradient Quantization: Gradients are quantized using the stochastic quantization method. Similar to uniform quantization, dynamic scaling has to be performed to avoid the **first** failure mode. However, while scale quantization scales back to the original order of magnitude after quantization, we identify that this does not necessarily apply to gradients. As the last step before weight update, gradients can be more sensitive to the order of magnitude. Hence, we give two versions of gradient quantization method depending on the network structure. For a network that implements BN, gradients are quantized by

$$G_q^i = Q_G(G^i) = Scale(G^i) * RQ(\frac{G^i}{Scale(G^i)}, k_G) \tag{17}$$

where the original order of magnitude is preserved by $Scale(G^i)$. For a network that does not implement BN, gradients are quantized by

$$G_q^i = Q_G(G^i) = RQ(\frac{G^i}{Scale(G^i)}, k_G) \tag{18}$$

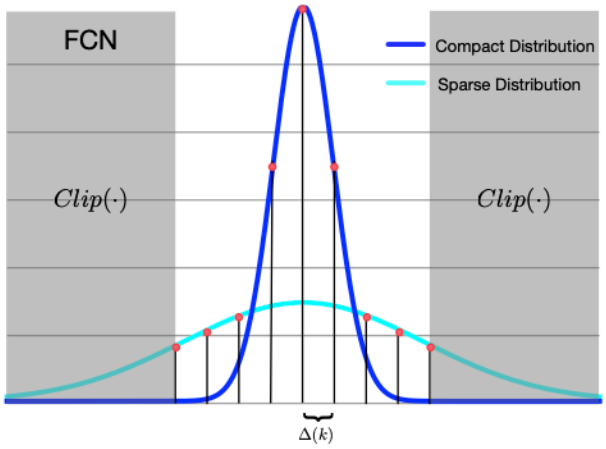
The difference between Equation (17) and (18) is that the latter abandons the original magnitude. We expand the detailed analysis in Section IV.

3) Update Quantization: Weight update directly reflects on master weights saved in the network, quantized by

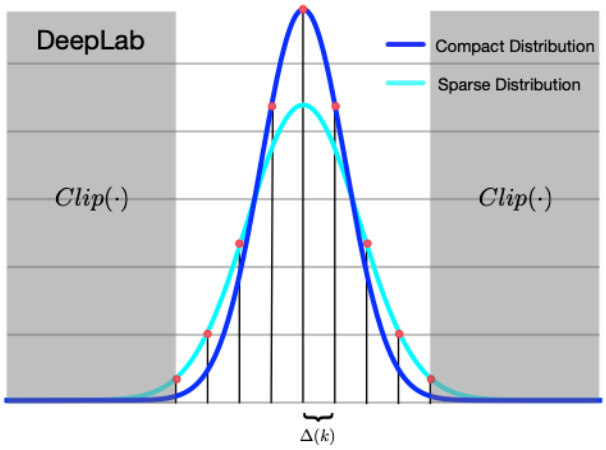
$$\Delta W^i = Q_U(G_q^i, k) = UQ(G_q^i * lr, k_U) \tag{19}$$

where gradient G_q^i is multiplied by quantized learning rate lr, and lr gradually decrease discretely during the entire training process.

Update takes the gradient and scale it by the learning rate. The product is quantized by $Q_U(\cdot)$ with bit-width k_U, returning the quantized weight update. It is responsible for the update of weights, and γ, β from the BN layer. Experiments show that the update quantization bit-width k_U should always be higher than k_W for the network to converge.



(a)



(b)

Fig. 6: Rough comparison of gradient distribution in FCN and DeepLab. Deep blue distribution describes the most compact distribution of a gradient layer, light blue describes the most scattered distribution of a gradient layer. (a) Gradient distribution of FCN, where gradients of different layers can have huge distribution differences. (b) Gradient distribution of DeepLab, where the distribution of different layers has a relatively smaller difference.

IV. Analysis on Back Propagation Quantization

Backpropagation is the key to the convergence of neural networks, and different quantization methods can dramatically influence the result of semantic segmentation networks. This section aims to provide insights and specify different quantization methods based on the properties brought by

semantic segmentation networks.

We have discussed previously the importance of dynamic scaling in objects that do not fit the uniform quantization range, but there still lies a problem on whether to keep the original order of magnitude of that layer, corresponding to Equation (17) and Equation (18). In previous works, DoReFa [26] scales the value back to its original magnitude after quantization, and some other frameworks do not. WAGE [21] omitted the scaling back based on the observation that it is the direction of error, rather than the order of magnitude leads the network towards convergence. This claim is partially valid depending on different contexts, since their gradient is also quantized with dynamic shifting, therefore canceling out the scale influence in error quantization.

Keep in mind that our quantization framework simulates full precision training, and full precision networks have different magnitude of gradient in different layers. When not scaling back to the original order of magnitude, we suffer from information loss of the original distribution. Our experiment results in Section V-C show that this information loss result in a noticeable degradation of accuracy in our DeepLab model.

However, this performance gain does not apply to all cases. For models that do not implement BN, the original order of magnitude should not be preserved. While the magnitude of gradient in these networks still varies, some extreme distribution in the decoder can have very large or small scale factors, resulting in a large dynamic range. Figure 6 roughly demonstrates the distribution of network with BN and network without BN in semantic segmentation. In a non-BN network, both failure modes are met mentioned in Section III-A2 due to the large dynamic range. This makes it hard to fit the quantization range of weight update quantization described in Equation (19). We discover that the extreme value in a non-BN network is introduced by the unique initialization method in semantic segmentation networks: First, the encoder uses pre-trained weights from DCNN on object classification tasks. Second, the full convolution layer and decoder layers are initialized into filters like bilinear upsampling. These two differences contribute to the large dynamic range of gradients in non-BN networks. Thus, dynamic scaling in these networks acts as a normalizing method in backpropagation. Although sacrificing the absolute magnitude of gradient in each layer, it helps to ensure valid weight updates in each training step.

V. Experiments & Results

To test the effectiveness of the quantization framework on semantic segmentation networks, we evaluate on two mainstream networks: FC8s-VGG16 [12] and DeepLabv3-ResNet-50 [3]. The proposed models are trained and evaluated on PASCAL VOC 2012 and ADE20K datasets. PASCAL VOC 2012 includes 20 foreground object classes and one background class. The proposed models in this paper are trained using the augmented dataset provided by [9], resulting in 10,582 training images, 1449 validation images, and 1456 test images. ADE20K consists of 150 foreground object classes and no background class. The dataset contains 20210 training images and 2000 validation images. Since we aim at comparing quantized network against full precision network, all performance reported are averaged over ten run on the validation set to avoid randomness.

The quality of the results are measured using mean Intersection over Union (mIoU). The computation formula of a single class IoU is governed by

$$IoU_i = \frac{TP_i}{TP_i + FP_i + FN_i} \tag{20}$$

where for any class i, TP_i, FP_i, FN_i denotes the pixels correctly predicted as class i, the pixels wrongly predicted as class i, and the pixels that are wrongly predicted as non-i class, respectively. Then the mIoU is average on all appeared class in the evaluated image, we have

$$mIoU_i = \frac{1}{m}\sum_{j=1}^{m} IoU_j. \tag{21}$$

A. Implementation Details

FCN8s: We use pre-trained weights for the VGG16 network backbone trained on Imagenet, and initialize transpose convolution decoder with bilinear upsampling filter with trainable weights. Every layer is fine-tuned with stochastic gradient descent. The network is trained with stochastic gradient descend optimizer with learning rate decaying to half for every 10 thousand steps. We use MSE loss function together with L2 weight decay of 5e-4. We crop the images into 512×512 with batch size of 8. Quantization bit-width are set to $k_W = k_A = k_{E_1} = k_G = 8$, and $k_U = 24$, and quantized gradients are scaled in each layer according to Equation (18).

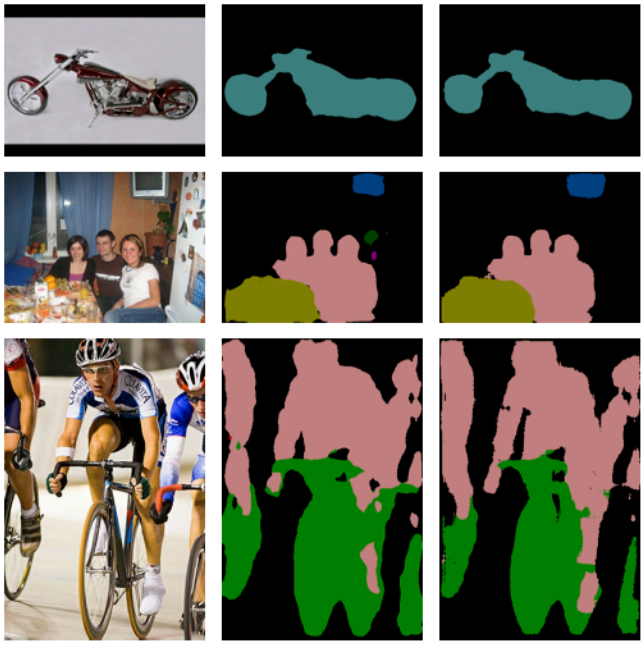
DeepLabv3: The network backbone is initialized with ResNet-50 pre-trained weights trained on ImageNet, and the rest of the network are initialized using the MSRA method [10]. Each layer is fine-tuned using softmax cross entropy with weight decay of 1e-5. We also use input images cropped at 512×512 and batch size of 8. Quantization bit-width are set to $k_W = k_A = k_{E_1} = k_G = 8, k_U = 24$ and no gradients scaling is performed during quantization, according to Equation (17). For BN quantization, $k_\mu = k_\sigma = k_{\hat{x}} = 16$ and $k_\gamma = k_\beta = 8$. For k_{E_2}, we report the results on both 8 bits and 16 bits.

B. Loss Curves and Accuracy

In this section, we demonstrate the consistency of our framework on semantic segmentation networks. The quantitive results on FCN and DeepLab are summarized in Table I, where mIoU are recorded on both datasets. All hyper parameters except learning rate are kept the same for full-precision network and quantization network, and no techniques to improve accuracy such as data augmentation or test time augmentation has been used. The results we have achieved on quantization network is comparable to its full-precision counterpart. FCN reports 2.05% and 1.77% mIoU degradation on the default 8-bit quantization scheme

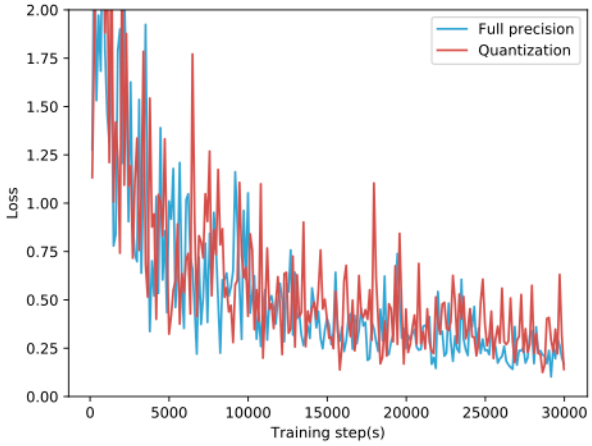
TABLE I: mIoU and pixel accuracy of different bit-width on FCN and DeepLab.

Model	W	A	G	E_1	E_2	U	BN	PASCAL	ADE20K
FCN	32	32	32	32	-	32	-	65.17%	27.85%
	8	8	8	8	-	24	-	**63.12%**	**26.08%**
DeepLab	32	32	32	32	32	32	$L2BN$	65.62%	26.14%
	8	8	8	8	16	24	$L2BN$	65.13%	25.92%
	8	8	8	8	16	24	$L1BN$	**65.21%**	**25.83%**
	8	8	8	8	8	24	$L1BN$	63.55%	24.16%

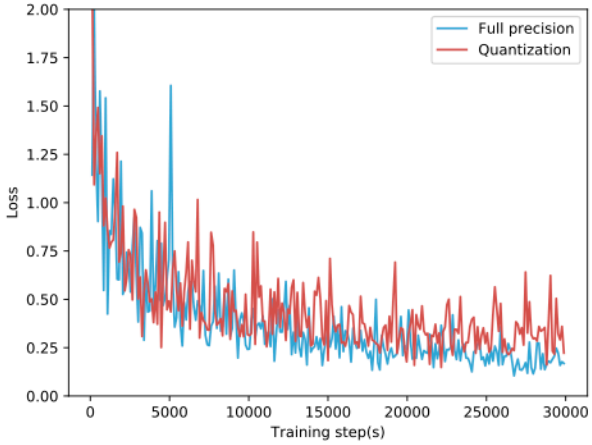


(a) Original Image (b) Full Precision (c) Quantized

Fig. 7: Comparison of visual semantic segmentation results on full precision network and quantized network.



(a)



(b)

Fig. 8: Loss curve comparison between full precision network and standard quantized network: (a) FCN, (b) DeepLab.

compared with its full-precision counterpart on PASCAL and ADE20K, respectively. Quantized DeepLabv3 achieved precision on PASCAL with only 0.41% degradation on the default scheme, and suffers 2.07% degradation with k_{E_2} also restricting to 8-bit quantization. Further, $L1BN$ proves to be equivalent to $L2BN$ in quantization network in terms of performance, but the linear property it preserves leaves potential for implementation on integer-based hardwares. Figure 7 demonstrates the comparison of qualitative results on full-precision network and quantized network.

Figure 8. shows the loss curve during training on PASCAL VOC 2012 data set. The quantized training curve follows the full-precision curve closely. This shows that our quantization framework effectively trained the network on discrete dataflow. FCN converges slower than DeepLab, and the curve tremors more seriously when quantized. This is due to the lack of normalization methods in FCN. The loss curve of quantized DeepLab converges similarly to the full precision training in early stages, but failed to close the gap at the final stages of training, and this gap is caused by the quantization step of weight update. With given quantization step, weight updates in the final training stage suffer from serious quantization noise, which can be resolved by increasing the bit-width of weight update.

C. Performance on different quantization methods

In Section IV, we theoretically analyzed the impact of preserving the original magnitude of gradients in backprop-

TABLE II: Performance comparison of different quantization methods. Using scale means that the original magnitude is not preserved.

Model	Scale	BN	mIoU
FCN	✓	-	63.12%
	✗	-	33.32%
DeepLab	✓	L1BN	64.13%
	✗	L1BN	65.21%

agation. Table II shows the accuracy of scaling the gradients or preserving the gradients. FCN suffers from huge mIoU degradation when preserving the original order of magnitude; Whereas DeepLab achieved 1.08% improvement. Although the performance improvement is not huge, DeepLab manages to close off some gap of mIoU between full-precision and quantization network. To further analysis the different behavior on FCN and DeepLab, we examined the performance of both networks during the training process, and discovered that the first 1000 steps are crucial to training the networks on PASCAL VOC 2012 dataset, and later training steps try to fine-tune the network to its performance limit. Therefore we extracted the gradient distribution of the first 1000 steps of the training process, and found different behavior on DeepLab and FCN.

Figure 9 shows the maximum and minimum order of magnitude extracted from the gradient distributions in the network. We observe that the order of magnitude of each layer can have a huge difference between layers. The range of the order of magnitude is wide for the layers in FCN. The maximum order of magnitude comes from the decoder gradients and minimum from the pre-trained encoder, covering the range of $[10^{-4}, 10^{1}]$. On the other hand, DeepLab gradients order of magnitude is constrained around $[10^{-2}, 10^{0}]$. While the extreme distribution in FCN does not affect the full precision network, the quantized network is restricted from the quantization range and minimum quantization step, as presented in Figure 6a. This suggests that despite the slower convergence of the decoder, by scaling the gradients to similar distribution in each layer ensures valid network updates in each step. Since DeepLab has a more compact distribution of gradients, by not scaling the gradients prevent the network from losing distribution information during training, resulting in a slight increase in performance.

D. Bit-width impact on weight update and weights

Bit-width of weight update k_W is strongly correlated to the performance of our network, and usually takes more bits compared to other quantization objects for the model to converge properly. As shown in Equation (19), weight update determines the bit-width of master weights. Figure 10 demonstrates the relationship between k_U and network performance, keeping other quantization parameters the same as the default bit-width. We found that as k_U approaches k_W, the network performance drops dramatically. Both DeepLab and FCN still generate acceptable results when k_U is above 16

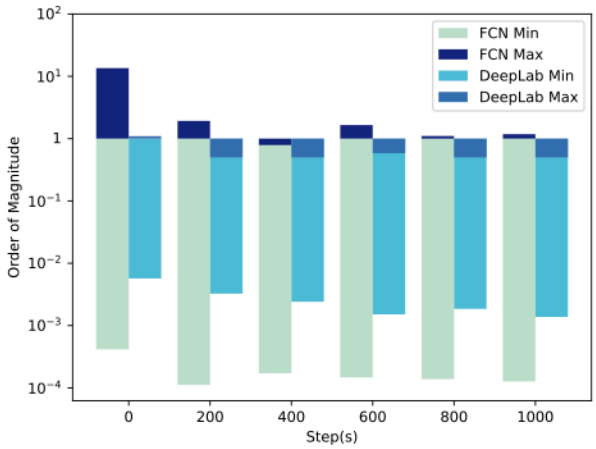


Fig. 9: Maximum scale and minimum scale on FCN and DeepLab on different training stages. FCN has a larger difference compared with DeepLab in concern of gradient magnitudes.

bits, thus weight update bit-width can be further compressed to save storage space. As k_U approaches the lower bound of k_W, the fluctuation of weights increases, and performance quickly degenerates to 1%. This is because when k_U is bigger, master weights create a buffer for quantized weights, so only the accumulated valid weight updates take effect finally in quantized weights.

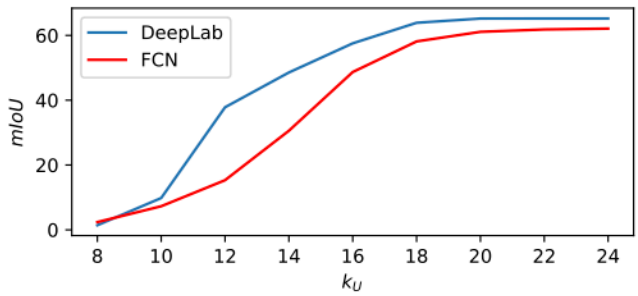


Fig. 10: mIoU performance curve of DeepLab and FCN on different k_U on PASCAL VOC 2012. Both models suffer from noticeable performance loss when k_U is lower than 16 bits.

E. Sensitivity of Decoder

Decoder is a critical part of semantic segmentation network that differs with traditional DCNN. While extracting features, feature maps are scaled to smaller sizes during pooling operation. Hence, the decoder recovers the resolution using layers like deconvolution layer and produces pixel-wise outputs. Here we use the FCN model to study the impact of decoder on our quantized framework (note that in experiments we divide the full convolution layer to the decoder structure for simplicity, since it does not belong to traditional DCNN).

To analyze the impact of quantizing decoder, we performed

an additional two set of experiments that solely quantizes either the encoder or decoder, and other parameters strictly follows the default quantization scheme. Further, we compare them with the results produced with full precision training and quantized training results acquired earlier, which is shown in Table III.

TABLE III: Performance comparison of quantizing different parts of dataflow. Decoder is more sensitive to quantization, responsible for a large portion of performance degradation.

Encoder	Decoder	mIoU
FP32	FP32	65.17%
INT8	FP32	64.74%
FP32	**INT8**	62.95%
INT8	**INT8**	63.12%

(a) Full Precision (b) Encoder Quantized

(c) Decoder Quantized (d) Quantized

Fig. 11: Visual results of segmentation prediction on quantizing different structures of the network.

The results suggest that the decoder seems to contribute more to the performance degradation compared to the decoder in quantization. To better understand this degradation, we extracted visual results from these four experiments and compared them. Figure 11 shows one set of the qualitative results, which represents the general failure mode of decoder well. While full precision and encoder quantized network produces results with a smooth boundary of segmentation mask, the boundary of decoder quantized network and entirely quantized network(default scheme) produces results with a rough edge when zooming the results. Moreover, we observe that a portion of the labelled pixels even separates from its boundary. As the overall visual segmentation results between these networks are similar, we suspect that the 2.05% performance largely comes from the rough edge.

Prior experience tells us that the quality of the decoder is closely related to the boundaries between different segmentation objects. Lead by this thought, we extracted the weights in the deconvolution layer in the early training stage and final training stage, shown in Figure 12. Compared with

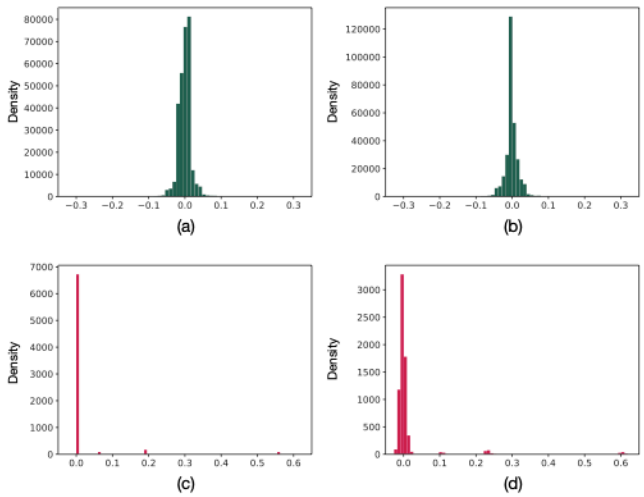


Fig. 12: Comparison of weight distribution on early and final training stages of the decoder in FCN. (a) Encoder early stage, (b) Encoder final stage, (c) Decoder early stage, (d) Decoder final stage.

the compact distribution of encoder variables, most values in the decoder are small values around zero, with sparse bigger values scattered in the range. This distribution suggests larger quantization noise compared to the distribution of traditional convolution layers. Due to this reason, quantization on the decoder cannot be quantized to lower than 8 bits, or the quantization noise will seriously affect backpropagation, leading the network to diverge. Although dynamic scaling quantization may not help much in quantizing the decoder, future work can try adaptive quantization to close the slight accuracy gap between full precision network and quantized network.

VI. Conclusion

In this paper, we thoroughly analyzed network quantization in semantic segmentation, and presented a quantization framework that achieves comparable performance on public datasets. It is promising that performance can be further improved on the quantization of decoder, by designing adaptive quantization methods to reduce the quantization error during decoder backpropagation. As it is not specifically designed for any specific network architecture, our quantization framework can be adapted to other semantic segmentation networks and provide acceleration in computation speed and save storage space on specific hardware. The quantized dataflow and L1BN promise the future for efficient online training and integer-based deep learning accelerators.

References

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[2] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[3] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017.

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[5] Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830, 2016.

[6] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131, 2015.

[7] Lei Deng, Peng Jiao, Jing Pei, Zhenzhi Wu, and Guoqi Li. Gxnor-net: Training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework. *Neural Networks*, 100:49–58, 2018.

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[9] Bharath Hariharan, Pablo Arbelaez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *International Conference on Computer Vision (ICCV)*, 2011.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[12] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[13] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *arXiv preprint arXiv:1606.02147*, 2016.

[14] 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.

[15] Charbel Sakr and Naresh Shanbhag. Per-tensor fixed-point quantization of the back-propagation algorithm. *arXiv preprint arXiv:1812.11732*, 2018.

[16] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[17] Zhiqiang Tang, Xi Peng, Shijie Geng, Lingfei Wu, Shaoting Zhang, and Dimitris Metaxas. Quantized densely connected u-nets for efficient landmark localization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 339–354, 2018.

[18] Zhiqiang Tang, Xi Peng, Kang Li, and Dimitris N Metaxas. Towards efficient u-nets: A coupled and quantized approach. *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[19] Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. In *Advances in neural information processing systems*, pages 7675–7684, 2018.

[20] S. Wu, G. Li, L. Deng, L. Liu, D. Wu, Y. Xie, and L. Shi. L1-norm batch normalization for efficient training of deep neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–9, 2018.

[21] Shuang Wu, Guoqi Li, Feng Chen, and Luping Shi. Training and inference with integers in deep neural networks. In *International Conference on Learning Representations*, 2018.

[22] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 325–341, 2018.

[23] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 405–420, 2018.

[24] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.

[25] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *arXiv preprint arXiv:1608.05442*, 2016.

[26] 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.