

U-Net Fixed-Point Quantization for Medical Image Segmentation

MohammadHossein AskariHemmat¹, Sina Honari², Lucas Rouhier³, Christian S. Perone³, Julien Cohen-Adad³, Yvon Savaria¹, and Jean-Pierre David¹

¹ Ecole Polytechnique Montreal, Canada
{mohammadhossein.askari-hemmat, yvon.savaria,
jean-pierre.david}@polymtl.ca
² Mila-University of Montreal, Canada
sina.honari@umontreal.ca

³ NeuroPoly Lab, Institute of Biomedical Engineering, Polytechnique Montreal, Montreal
christian.perone@gmail.com, {lucas.rouhier, julien.cohen-adad}@polymtl.ca

Abstract. Model quantization is leveraged to reduce the memory consumption and the computation time of deep neural networks. This is achieved by representing weights and activations with a lower bit resolution when compared to their high precision floating point counterparts. The suitable level of quantization is directly related to the model performance. Lowering the quantization precision (e.g. 2 bits), reduces the amount of memory required to store model parameters and the amount of logic required to implement computational blocks, which contributes to reducing the power consumption of the entire system. These benefits typically come at the cost of reduced accuracy. The main challenge is to quantize a network as much as possible, while maintaining the performance accuracy. In this work, we present a quantization method for the U-Net architecture, a popular model in medical image segmentation. We then apply our quantization algorithm to three datasets: (1) the Spinal Cord Gray Matter Segmentation (GM), (2) the ISBI challenge for segmentation of neuronal structures in Electron Microscopic (EM), and (3) the public National Institute of Health (NIH) dataset for pancreas segmentation in abdominal CT scans. The reported results demonstrate that with only 4 bits for weights and 6 bits for activations, we obtain 8 fold reduction in memory requirements while losing only 2.21%, 0.57% and 2.09% dice overlap score for EM, GM and NIH datasets respectively. Our fixed point quantization provides a flexible trade off between accuracy and memory requirement which is not provided by previous quantization methods for U-Net such as TernaryNet. ⁴

Keywords: U-Net · Quantization · Deep Learning

1 Introduction

Image segmentation, the task of specifying the class of each pixel in an image, is one of the active research areas in the medical imaging domain. In particular, image segmentation for biomedical imaging allows identifying different tissues, biomedical

⁴ Our code will be released at <https://github.com/hossein1387/U-Net-Fixed-Point-Quantization-for-Medical-Image-Segmentation>

structures, and organs from images to help medical doctors diagnose diseases. However, manual image segmentation is a laborious task. Deep learning methods have been used to automate the process and alleviate the burden of segmenting images manually.

The rise of Deep Learning has enabled patients to have direct access to personal health analysis [1]. Health monitoring apps on smart phones are now capable of monitoring medical risk factors. Medical health centers and hospitals are equipped with pre-trained models used in medical CADs to analyse MRI images [2]. However, developing a high precision model often comes with various costs, such as a higher computational burden and a large model size. The latter requires many parameters to be stored in floating point precision, which demands high hardware resources to store and process images at test time. In medical domains, images typically have high resolution and can also be volumetric (the data has a depth in addition to width and height). Quantizing the neural networks can reduce the feedforward computation time and most importantly the memory burden at inference. After quantization, a high precision (floating point) model is approximated with a lower bit resolution model. The goal is to leverage the advantages of the quantization techniques while maintaining the accuracy of the full precision floating point models. Quantized models can then be deployed on devices with limited memory such as cell-phones, or facilitate processing higher resolution images or bigger volumes of 3D data with the same memory budget. Developing such methods can reduce the required memory to save model parameters potentially up to 32x in memory footprint. In addition, the amount of hardware resources (the number of logic gates) required to perform low precision computing, is much less than a full precision model [3]. In this paper, we propose a fixed point quantization of U-Net [4], a popular segmentation architecture in the medical imaging domain. We provide a comprehensive quantization results on the Spinal Cord Gray Matter Segmentation Challenge [5], the ISBI challenge for segmentation of neuronal structures in electron microscopic stacks [6], and the public National Institute of Health (NIH) dataset for pancreas segmentation in abdominal CT scans [7]. In summary, this work makes the following contributions: Developing such methods can reduce the required memory to save model parameters potentially up to 32x in memory footprint. In addition, the amount of hardware resources (the number of logic gates) required to perform low precision computing, is much less than a full precision model [3].

In this paper, we propose a fixed point quantization of U-Net [4], a popular segmentation architecture in the medical imaging domain. We provide a comprehensive quantization results on the Spinal Cord Gray Matter Segmentation Challenge [5], the ISBI challenge for segmentation of neuronal structures in electron microscopic stacks [6], and the public National Institute of Health (NIH) dataset for pancreas segmentation in abdominal CT scans [7]. In summary, this work makes the following contributions:

- We report the first fixed point quantization results on the U-Net architecture for the medical image segmentation task and show that the current quantization methods available for U-Net are not efficient for the common hardware in the industry.
- We quantify the impact of quantizing the weights and activations on the performance of the U-Net model on three different medical imaging datasets.

- We report results comparable to a full precision segmentation model by using only 6 bits for activation and 4 bits for weights, effectively reducing the weights size by a factor of $8\times$ and the activation size by a factor of $5\times$.

2 Related Works

2.1 Image Segmentation

Image segmentation is one of the central problems in medical imaging [8], commonly used to detect regions of interest such as tumors. Deep learning approaches have obtained the state-of-the-art results in medical image segmentation [9,10]. One of the favorite architectures used for image segmentation is U-Net [4] or its equivalent architectures proposed around the same time; ReCombinator Networks [11], SegNet [12], and DeconvNet [13], all proposed to maintain pixel level information that is usually lost due to pooling layers. These models use an encoder-decoder architecture with skip connections, where the information in the encoder path is reintroduced by skip connections in the decoder path. This architecture has proved to be quite successful for many applications that require full image reconstruction while changing the modality of the data, as in the image-to-image translation [14], semantic segmentation [4,12,13] or landmark localization [11,15]. While all the aforementioned models propose the same architecture, for simplicity we refer to them as U-Net models. U-Net type models have been very popular in the medical imaging domain and have been also applied to the 3 dimensional (3D) segmentation task [16]. One problem with U-Net is its high usage of memory due to full image reconstruction. All encoded features are required to be kept in memory and then used while reconstructing the final output. This approach can be quite demanding, especially for high resolution or 3D images. Quantization of weights and activations can reduce the required memory for this model, allowing to process images with a higher resolution or with a bigger 3D volume at test time.

2.2 Quantization for Medical Imaging Segmentation

There are two approaches to quantize a neural network, namely deterministic quantization and stochastic quantization [3]. Although DNN quantization has been thoroughly studied [3,17,18], little effort has been done on developing quantization methods for medical image segmentation. In the following, we review recent works in this field.

Quantization in Fully Convolutional Networks: Quantization has been applied to Fully Convolutional Networks (FCN) in biomedical image segmentation [19]. First, a quantization module is added to the suggestive annotation in FCN. In suggestive annotation, instead of using the original dataset, a representative training dataset is used, which in turn increases the accuracy. Next, FCN segmentations are quantized using Incremental Quantization (INQ). Authors report that suggestive annotation with INQ using 7 bits results in accuracy close to or better than those obtained with a full precision model. In FCN, features of different resolutions are upsampled back to the image resolution and merged together right before the final output predictions. This approach is sub-optimal compared to the U-Net which upsamples features only to one higher resolution, allowing the model to process them before they are passed to higher resolution layers. This gradual resolution increase in reconstruction acts as a

conditional computation, where the features of higher resolution are computed using the lower resolution features. As reported in [11], this process of conditional computation results in faster convergence time and increased accuracy in the U-Net type architectures compared to the FCN type architectures. Considering the aforementioned advantages of U-Net, in this paper we pursue the quantization of this model.

U-Net Quantization: In [20], authors propose the first quantization for U-Net. They introduce 1) a parameterized ternary hyperbolic tangent to be used as the activation function, 2) a ternary convolutional method that calculates matrix multiplication very efficiently in the hamming space. They report 15-fold decrease in the memory requirement as well as 10x speed-up at inference compared to the full precision model. Although this method shows significant performance boost, in Section 4 we demonstrate that this is not an efficient method for the currently available CPUs and GPUs.

3 Method

We propose fixed point quantization for U-Net. We start with a full precision (32 bit floating point) model as our baseline. We then use the following fixed point quantization function to quantize the parameters (weights and activation) in the inference path:

$$\text{quantize}(x, n) = (\text{round}(\text{clamp}(x, n) \ll n) \gg n) \quad (1)$$

where *round* function projects its input to the nearest integer, \ll and \gg are shift left and right operators, respectively. In our simulation, shift left and right are implemented by multiplication and division in powers of 2. The *clamp* function is defined as:

$$\text{clamp}(x, n) = \begin{cases} 2^n - 1 & \text{when } x \geq 2^n - 1 \\ x & \text{when } 0 < x < 2^n - 1 \\ 0 & \text{when } x \leq 0 \end{cases} \quad (2)$$

Equation (1) quantizes an input $x \in \mathbb{R}$ to the closest value that can be represented by n bits. To map any given number x to its fixed point value we first split the number into its floating and integer parts using:

$$x_f = \text{abs}(x) - \text{floor}(\text{abs}(x)), x_i = \text{floor}(\text{abs}(x)) \quad (3)$$

and then use the following equation to convert x to its fixed point representation using the specified number of bits for the integer (*ibits*) and fractional (*fbits*) parts:

$$\begin{aligned} \text{to_fixed_point}(x, \text{ibits}, \text{fbits}) &= \text{sign}(x) * \text{quantize}(x_i, \text{ibits}) \\ &+ \text{sign}(x) * \text{quantize}(x_f, \text{fbits}) \end{aligned} \quad (4)$$

Equation (4) is a fixed point quantization function that maps a floating point number x to the closest fixed point value with *ibits* integer and *fbits* fractional bits. Throughout this paper, we use $Q^{p.i.f}$ notation to denote that we are using a fixed point quantization of parameter p by using i bits to represent the integer part and f bits to represent the fractional part. Based on our experiments, we did not benefit from an incremental quantization (INQ) as explained in [17]. Although this method could work for higher precision models, for instance when using fixed point $Q^w8.8$ (Quantizing weights with

8 bits integer and 8 bits fractional parts), for extreme quantization as in $Q^w0.4$, learning from scratch gave us the best accuracy with the shortest learning time. As shown in Figure S1, in the full precision case the weights of all U-Net layers are in $[-1, 1]$, hence the integer part for the weight quantization is not required.

3.1 Training

For numerical stability and to verify the gradients can propagate in training, we demonstrate that our quantization is differentiable. Starting from Equation (2), the derivative is:

$$\forall x \in \mathbb{R}, \forall n \in \mathbb{Z}^+, \quad \frac{\partial}{\partial x} \text{clamp}(x, n) = \begin{cases} 0 & \text{when } x \geq 2^n - 1 \\ 1 & \text{when } 0 < x < 2^n - 1 \\ 0 & \text{when } x \leq 0 \end{cases} \quad (5)$$

which is differentiable except on the thresholds. To make it completely differentiable, a straight-through estimator (STE), introduced in [21], is used which passes gradients over the thresholds and also over the *round* function in Equation (1).

3.2 Observations on U-Net Quantization

Dropout Dropout [22] is a regularization technique to prevent over-fitting of DNNs. Although it is used in the original implementation of U-Net, we found that when this technique is applied along with quantization, the accuracy drops a lot. Hence, in our implementation we removed dropout from all layers. This is due to the fact that quantization acts as a strong regularizer, as reported in [3], hence further regularization with dropout is not required. As shown in Figure S2 for each quantized precision, dropout reduces the accuracy, with the gap being even higher for lower precision quantizations.

Full Precision Layers It is well accepted to keep the first and the last layers in full precision, when applying quantization [3,23]. However, we found that in the segmentation task, keeping the last layer in full precision has much more impact than keeping the first layer in full precision.

Batch Normalization Batch normalization is a technique that improves the training speed and accuracy of DNN. We used the Pytorch implementation of batchnorm. In training, we use the quantization block after the batchnorm block in each layer (S.2 lists all the layers in our U-Net implementation) such that the batchnorm is first applied using the floating point calculations and then the quantized value is sent to the next layer (hence not quantizing the batchnorm block during training). However, at inference, Pytorch folds the batchnorm parameters into the weights, effectively including batchnorm parameters in the quantized model as part of the quantized weights.

4 Results and Discussion

We implemented the U-Net model and our fixed-point quantizer in Pytorch. We trained our model over 200 epochs with a batch size of 4. We applied our fixed point quantization along with TernaryNet [20] and Binary [18] quantization on three different datasets: GM

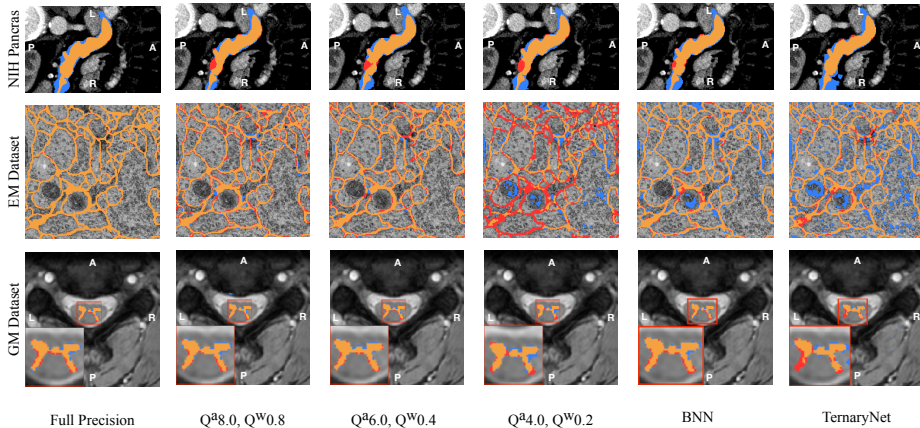


Fig. 1: Sample prediction versus ground truth segmentation results for NIH Pancreas (top), EM (middle) and GM (bottom) datasets. From left to right, the result of different quantization methods and precisions are reported. Segments in ■ show false positive, segments in ■ show false negative and segments in ■ show true positive. Check S.4 for more results.

[5], EM [6], and NIH [7]. For GM and EM datasets, we used an initial learning rate of $1e - 3$ and for NIH we used initial learning rate of 0.0025. For all datasets we used Glorot for weight initialization and cosine annealing scheduler to reduce learning rate in training. Please check our repository for the model and training details.

NIH pancreas [7] dataset is composed of 82 3D abdominal CT scan and their corresponding pancreas segmentation images. Unfortunately, we did not had access to the pre-processed dataset described in [20], nevertheless, we extracted 512x512 2-D slices from the original dataset and applied a region of interest cropping to get 7059 images of size 176x112. The final dataset contains 7059 176x112 2-D images which are separated into training and testing dataset (respectively 80% and 20%). For GM and EM datasets, we used the provided dataset as described in [5] and [6] respectively. For both EM and GM datasets, we did not used any region of interest cropping and we used images of size 200x200.

The task of image segmentation for GM and NIH pancreas datasets is imbalanced. As suggested in [5], instead of weighted cross-entropy, we used a surrogate loss for the dice similarity coefficient. This loss is referred to as the dice loss and is formulated as $\mathcal{L}_{dice} = \frac{2 \sum_{n=1}^N p_n r_n + \epsilon}{\sum_{n=1}^N p_n + \sum_{n=1}^N r_n + \epsilon}$, where $p_n \in [0, 1]$ and $r_n \in \{0, 1\}$ are prediction and ground truth pixels respectively (with 0 indicating not belonging and 1 indicating belonging to the class of interest) and ϵ is the added noise for numerical stability. For the EM dataset, using a weighted sum of cross entropy and dice loss produced the best results.

Figure 1 along with Table 1 show different quantization methods on the aforementioned datasets. Considering NIH dataset, Figure 1(top) and Table 1 show that despite using only 1 and 2 bits to represent network parameters, Binary and TernaryNet quantizations produce results that are close to the full precision model. However, for other datasets, our fixed point $Q^a6.0$, $Q^w0.4$ quantization surpasses Binary and TernaryNet

Table 1: Dice scores of the quantized U-Net models on EM (left) GM (middle) and NIH (right) datasets. The last two row shows results for Binary and TernaryNet quantizations. Other rows report results obtained for different weight and activation quantization precisions. For GM and EM datasets, we also report results when Tanh is used instead of ReLU as the activation function.

Quantization		Parameter Size	EM Dataset		GM Dataset		NIH Panceas
Activation	Weight		Dice Score ReLU	Dice Score Tanh	Dice Score ReLU	Dice Score Tanh	Dice Score
Full Precision		18.48 MBytes	94.05	93.02	56.32	56.26	75.69
Q8.8	Q8.8	9.23 MBytes	92.02	91.08	56.11	56.01	74.61
Q8.0	Q0.8	4.61 MBytes	92.21	88.42	56.10	53.78	73.05
Q6.0	Q0.4	2.31 MBytes	91.03	90.93	55.85	52.34	73.48
Q4.0	Q0.2	1.15 MBytes	79.80	54.23	51.80	48.23	71.77
BNN [18]		0.56 MBytes	78.53	-	31.44	-	72.56
TernaryNet [20]		1.15 MBytes	-	82.66	-	43.02	73.9

quantization. The other important factor here is how efficient these quantization techniques can be implemented using the current CPU and GPU hardware. At the time of writing this paper, there is no commercially available CPU or GPU that can efficiently store and load sub-8bit parameters of a neural network, which leaves us to use custom functions to do bit manipulation to make sub-8bit quantization more efficient. Moreover, in the case of TernaryNet, to apply floating point scaling factor after ternary convolution, floating point operations are required. Our fixed point quantization uses only integer operations, which requires less hardware footprint and use less power compared to floating point operations. Finally, TernaryNet uses Tanh instead of ReLU for the activations. Using hyperbolic tangent as an activation function increases training time [24] and execution time at inference. To verify it, we evaluated the performance of ReLU and Tanh in a simple neural network with 3 fully connected layers. We used the Intel’s OpenVino [25] inference engine together with high performance `gemm_blas` and `avx2` instructions. Table 2 illustrates that using ReLU instead of Tanh at training and inference can increase performance by up to 8 times. These results can be extended to U-Net, since activation inference time is only a function of the input size. To compensate for the computation time, TernaryNet implements an efficient ternary convolution that gains up to 8 times in performance. At inference, an efficient Tanh function can be implemented that uses only two comparators to perform Tanh for ternary values. Considering accuracy, when Tanh is used as an activation function, the full precision accuracy is lower compared to ReLU [20]. We observe similar behavior in the results reported in Table 1. Our fixed point quantizer provides a flexible trade-off between accuracy and memory, which makes it a practical solution for the current CPUs and GPUs, does not require floating-point operations, and leverages the more efficient ReLU function. As opposed to BNN and TernaryNet quantizations, Table 1 shows that our approach for quantization of U-Net provides consistent results over 3 different datasets.

5 Conclusion

In this work, we proposed a fixed point quantization method for the U-Net architecture and evaluated it on the medical image segmentation task. We report quantization results

Table 2: Comparing Relu and Tanh run time using Intel’s OpenVino [25]. Each row illustrates the execution time for a layer of a neural network in micro seconds. It demonstrates that using Tanh as activation can increase execution time by up to 8 times compared to ReLU.

Layer Type	Instruction Type	Execution time	Execution time	Performance Gain of	Tensor Dimension
		in μ s Tanh	in μ s ReLU	using ReLU over Tanh	
Activation	jit_avx2_FP32	30	5	6	[100, 100]
FullyConnected	gemm_blas_FP32	20	19	-	-
Reshape	unknown_FP32	0	0	-	-
FullyConnected	gemm_blas_FP32	860	527	-	-
Activation	jit_avx2_FP32	77	9	8.6	[100, 300]

on three different semantic segmentation datasets and show that our fixed point quantization produces more accurate and also more consistent results over all these datasets compared to other quantization techniques. We also demonstrate that Tanh, as the activation function, reduces the base line accuracy and also adds a computational complexity in both training and inference. Our proposed fixed point quantization technique provides a trade-off between accuracy and the required memory, does not require floating point computation and is more suitable for the currently available CPU and GPU hardware.

References

1. Miotto, R., et al.: Deep learning for healthcare: review, opportunities and challenges. *Briefings in Bioinformatics* (2017)
2. S., T., et al.: The role of deep learning in improving healthcare. *Data Science for Healthcare* (2019)
3. Hubara, I., et al.: Quantized neural networks: Training neural networks with low precision weights and activations. *JMLR* (2018)
4. Ronneberger, O., et al.: U-net: Convolutional networks for biomedical image segmentation. In: *MICCAI*, Springer (2015)
5. Prados, F., et al.: Spinal cord grey matter segmentation challenge. *NeuroImage* (2017)
6. Cardona, A., et al.: An integrated micro-and macroarchitectural analysis of the drosophila brain by computer-assisted serial section electron microscopy. *PLoS Biology* (2010)
7. Roth., H.R., et al.: Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation. *MICCAI* (2015)
8. Pham, D.L., et al.: Current methods in medical image segmentation. *Annual review of biomedical engineering* (2000)
9. Litjens, G., et al.: A survey on deep learning in medical image analysis. *Medical Image Analysis* (2017)
10. Shen, D., et al.: Deep learning in medical image analysis. *Annual review of biomedical engineering* (2017)
11. Honari, S., et al.: Recombinator networks: Learning coarse-to-fine feature aggregation. In: *CVPR* (2016)
12. Badrinarayanan, V., et al.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *TPAMI* (2017)
13. Noh, H., et al.: Learning deconvolution network for semantic segmentation. In: *ICCV* (2015)
14. Isola, P., et al.: Image-to-image translation with conditional adversarial networks. In: *CVPR* (2017)

15. Newell, A., et al.: Stacked hourglass networks for human pose estimation. In: ECCV, Springer (2016)
16. Çiçek, Ö., et al.: 3d u-net: learning dense volumetric segmentation from sparse annotation. In: MICCAI, Springer (2016)
17. Zhou, A., et al.: Incremental network quantization: Towards lossless cnns with low-precision weights. CoRR (2017)
18. Courbariaux, M., et al.: Binaryconnect: Training deep neural networks with binary weights during propagations. In: NeurIPS (2015)
19. Xu, X., et al.: Quantization of fully convolutional networks for accurate biomedical image segmentation. In: CVPR (2018)
20. Heinrich, M.P., et al.: Ternarynet: Faster deep model inference without gpus for medical 3d segmentation using sparse and binary convolutions. CoRR (2018)
21. Hinton, G., et al.: Neural networks for machine learning, video lectures. Coursera (2012)
22. Srivastava, N., et al.: Dropout: a simple way to prevent neural networks from overfitting. JMLR (2014)
23. Tang, W., et al.: How to train a compact binary neural network with high accuracy? In: AAAI (2017)
24. Krizhevsky, A., et al.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25. Curran Associates, Inc. (2012)
25. Deanne Deuermeyer, Andrey Z., A.R.F.B.: Release notes for intel® distribution of openvino™ toolkit 2019 accessed on Jun 13th 2019.

Supplementary Information for U-Net Fixed-Point Quantization for Medical Image Segmentation

S.1 Weight Visualization of Full-Precision U-Net

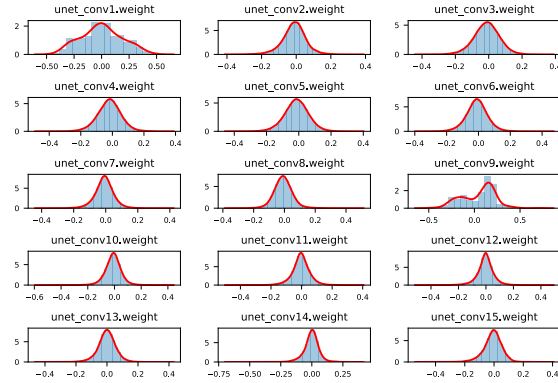


Fig. S1: The weight distribution of U-Net when no quantization is used (full precision). The red curve shows the kernel density estimate (KDE) of the weights and the blue bins show the actual weight values of the distribution. Each plot shows the KDE of one convolutional layer in U-Net. As observed in this figure, in the full precision model almost all weights are distributed in $[-1, +1]$. Hence, no integer part is required in the quantized weights and using higher number of bits for the fractional part of the weights will result in a higher precision.

S.2 Model Architecture

Layer (type)	Output Shape	Param #
Conv2d-1	[64 , 200, 200]	640
BatchNorm2d-2	[64 , 200, 200]	128
QuantLayer-3	[64 , 200, 200]	0
Conv2d-4	[64 , 200, 200]	36,928
BatchNorm2d-5	[64 , 200, 200]	128
QuantLayer-6	[64 , 200, 200]	0
DownConv-7	[64 , 200, 200]	0
MaxPool2d-8	[64 , 100, 100]	0

Conv2dQuant-9	[128, 100, 100]	73,856
BatchNorm2d-10	[128, 100, 100]	256
QuantLayer-11	[128, 100, 100]	0
Conv2dQuant-12	[128, 100, 100]	147,584
BatchNorm2d-13	[128, 100, 100]	256
QuantLayer-14	[128, 100, 100]	0
DownConv-15	[128, 100, 100]	0
MaxPool2d-16	[128, 50, 50]	0
Conv2dQuant-17	[256, 50, 50]	295,168
BatchNorm2d-18	[256, 50, 50]	512
QuantLayer-19	[256, 50, 50]	0
Conv2dQuant-20	[256, 50, 50]	590,080
BatchNorm2d-21	[256, 50, 50]	512
QuantLayer-22	[256, 50, 50]	0
DownConv-23	[256, 50, 50]	0
MaxPool2d-24	[256, 25, 25]	0
Conv2dQuant-25	[256, 25, 25]	590,080
BatchNorm2d-26	[256, 25, 25]	512
QuantLayer-27	[256, 25, 25]	0
Conv2dQuant-28	[256, 25, 25]	590,080
BatchNorm2d-29	[256, 25, 25]	512
QuantLayer-30	[256, 25, 25]	0
DownConv-31	[256, 25, 25]	0
Upsample-32	[256, 50, 50]	0
Conv2dQuant-33	[256, 50, 50]	1,179,904
BatchNorm2d-34	[256, 50, 50]	512
QuantLayer-35	[256, 50, 50]	0
Conv2dQuant-36	[256, 50, 50]	590,080
BatchNorm2d-37	[256, 50, 50]	512
QuantLayer-38	[256, 50, 50]	0
DownConv-39	[256, 50, 50]	0
UpConv-40	[256, 50, 50]	0
Upsample-41	[256, 100, 100]	0
Conv2dQuant-42	[128, 100, 100]	442,496
BatchNorm2d-43	[128, 100, 100]	256
QuantLayer-44	[128, 100, 100]	0
Conv2dQuant-45	[128, 100, 100]	147,584
BatchNorm2d-46	[128, 100, 100]	256
QuantLayer-47	[128, 100, 100]	0
DownConv-48	[128, 100, 100]	0
UpConv-49	[128, 100, 100]	0
Upsample-50	[128, 200, 200]	0
Conv2dQuant-51	[64, 200, 200]	110,656
BatchNorm2d-52	[64, 200, 200]	128
QuantLayer-53	[64, 200, 200]	0
Conv2dQuant-54	[64, 200, 200]	36,928
BatchNorm2d-55	[64, 200, 200]	128
QuantLayer-56	[64, 200, 200]	0
DownConv-57	[64, 200, 200]	0
UpConv-58	[64, 200, 200]	0

Conv2d-59

[1 , 200, 200]

577

```

=====
Total params: 4,837,249
Trainable params: 4,837,249
Non-trainable params: 0
-----

```

```

-----
Input size (MB): 0.15
Forward/backward pass size (MB): 593.57
Params size (MB): 18.45
Estimated Total Size (MB): 612.17
-----

```

S.3 Dropout In Quantization

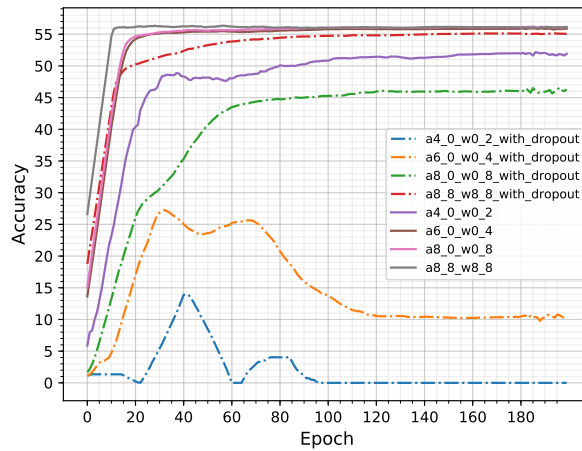


Fig. S2: Dice score of different U-Net quantization precisions over 200 epochs on the Spinal Cord Gray Matter Segmentation data set. We used our fixed point quantizer to show that using drop out while applying quantization on the model can drastically reduce the dice score. This phenomenon has more severe impact when the model precision is reduced. Curves shown with -- use dropout and quantization.

S.4 More Experimental Results

S.4.1 More Experimental Results For EM dataset

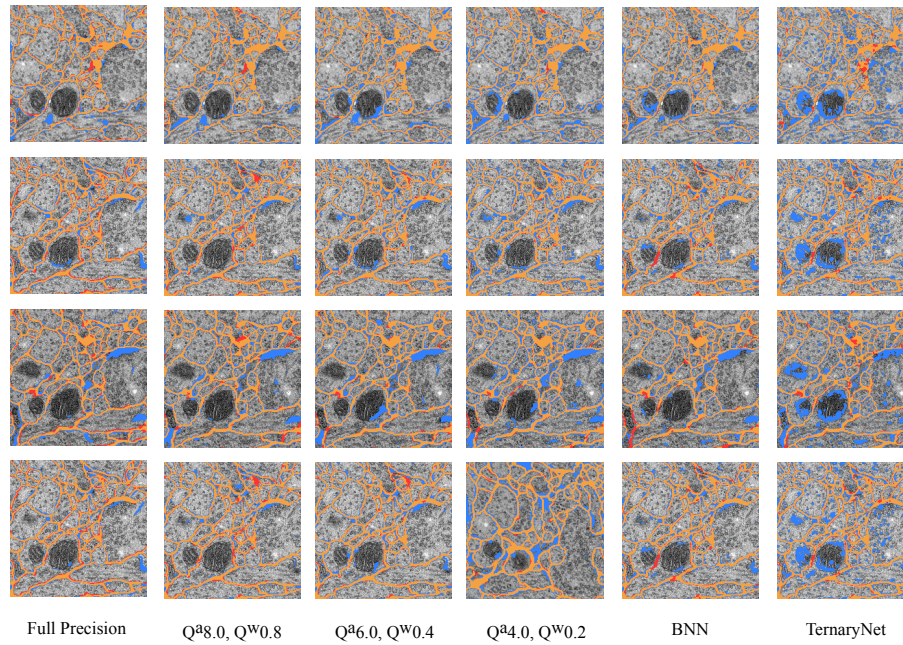


Fig. S3: Sample prediction versus ground truth segmentation results for EM data set. From left to right, results for different quantization precision are reported. Segments in ■ show false positive, segments in ■ show false negative and segments in ■ show true positive. The quantized model of $Q^{a6.0}, Q^{w0.4}$ obtains close results compared to the full precision model.

S.4.2 More Experimental Results For NIH Pancreas dataset

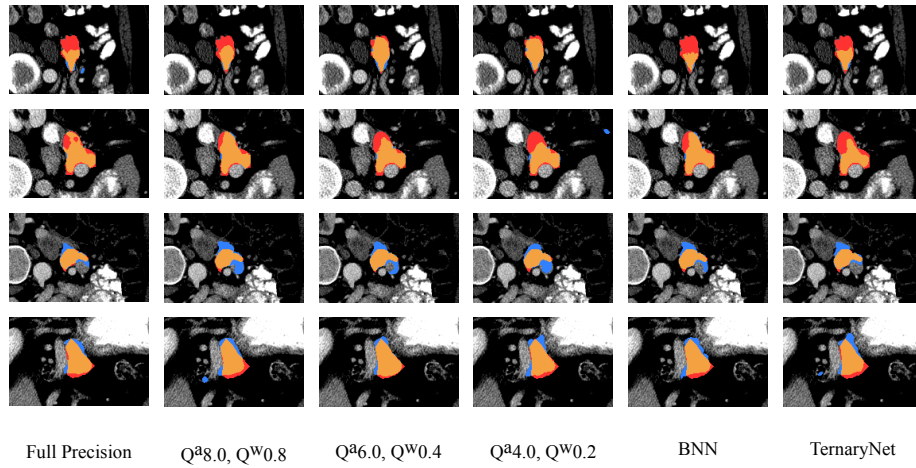


Fig. S4: Sample prediction versus ground truth segmentation results for NIH pancreas dataset. From left to right, results for different quantization methods. Segments in ■ show false positive, segments in ■ show false negative and segments in ■ show true positive. Note that in contrary to GM dataset, NIH pancreas segmentation is much harder since the pancreas has a very high anatomical variability.

S.4.3 More Experimental Results For GM dataset

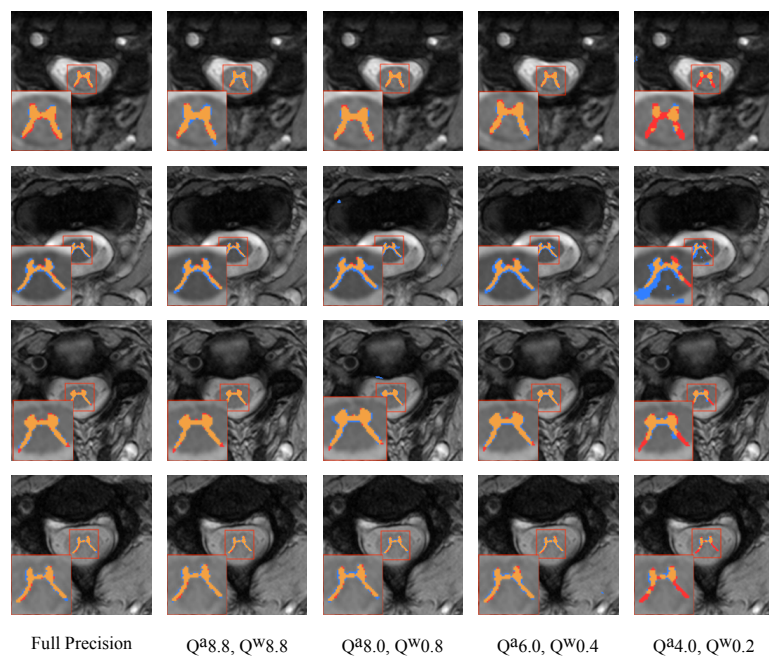


Fig. S5: Sample prediction versus ground truth segmentation results for GM data set. From left to right, results for different quantization precision are reported. Segments in ■ show false positive, segments in ■ show false negative and segments in ■ show true positive. The quantized model of $Q^{a6.0}, Q^{w0.4}$ obtains close results compared to the full precision model.