2020 Fall Comp 576 Final Project: Quantized Fast Neural Style Transfer

Han Guo [*1] **Qixuan Yu** [*1]

1. Introduction

Deep Neural networks (DNNs) are notorious for their large computational costs and energy consumptions. Speed and power become a major challenge when the models are deployed to edge devices for real-time applications. A promising approach to address these problems is quantization, or Quantized Nerual Networks (QNNs).

Quantization has been well-studied and experimented on task-specific networks, such as those for image classification and object detection. Although many manuscripts on network quantization exist, we find little, if any, works that attempted image generation tasks with QNNs. When we learned neural style transfer as an interesting topic in class, the idea comes to our mind: can we use model quantization on neural style transfer?

In our final project, we try to explore how model quantization affect neural style transfer in the following aspects:

- How does quantization affect stylization quality of images?
- How does quantization affect computational costs of the transformation?

2. Background

The seminal work on neural style transfer was initially brought up by (Gatys et al., 2016) where the Gram matrix is introduced to measure the pixel-wise covariance for style matching from style image and content image. However, pixel-wise covariance Gram matrix calculation is extremely slow and energy consuming. Later, (Johnson et al., 2016) proposed another approach: instead of measuring pixel-wise covariance, focusing on high-level representation, i.e. perceptual loss, is more favorable for fast real-time neural style transfer.

Model quantization has attract a lot of attentions since 2015 (Guo, 2018). With larger and deeper networks appear, quantization is used to create both with high accuracy and low computational cost model. The quantization technique has been well studied with some very famous works (Courbariaux et al., 2016; Zhou et al., 2018; Banner et al., 2018).

In our project, we aim to adapt and evaluate quantization with neural style transfer. We choose (Johnson et al., 2016) as our neural style transfer baseline due to the following reasons: it has a high performance; it has been adopted by PyTorch official as the neural style transfer toutorial example; and its open-source codebase is well-written and well-documented. We follow (Banner et al., 2018) as our quantization method due to its high scalability, modularized design, and high accuracy in general applications. For its recency and good performance, we follow (Qin et al., 2020) for network binarization, a distinct type of quantization where values are quantized to only 1 bit.

2.1. Neural Style

We have adopted fast neural transfer by (Johnson et al., 2016). Specifically, this work differ from (Gatys et al., 2016) by introducing perceptual loss on per-pixel relations. Perceptual loss consists two loss calculations: feature reconstruction loss and style reconstruction loss.

The feature reconstruction loss is Euclidean distance between feature representations

$$l_{feat}^{\Phi,j}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\Phi_j(\hat{y}) - \Phi_j(y)\|_2^2 \tag{1}$$

where $\Phi_j(x)$ is the activation of the jth layer, C_j, H_j, W_j denotes number of channels, heights, and weights of jth layer. This is calculated in the third ReLU of VGG16 backbone network.

The style reconstruction loss is defined as squared Frobenius norm of difference between the Gram matrices of input and output images,

$$l_{style}^{\Phi,j}(\hat{y}, y) = \|G_j^{\Phi}(\hat{y}) - G_j^{\Phi}(y)\|_F^2 \tag{2}$$

Here we define Gram matrix $G_j^{\Phi}(y)$ as to be $C_j \times C_j$ matrix whose elements are given by

$$G_j^{\Phi}(x)_{c,c'} = \frac{1}{C_j H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \Phi_j(x)_{h,w,c} \Phi_j(x)_{h,w,c'} \tag{3}$$

[*]Equal contribution [1]Department of Computer Science, Rice University. Correspondence to: Han Guo <hg31@rice.edu>, Qixuan Yu <qy12@rice.edu>.

Follow ICML 2020 format Copyright 2020 by the author(s).

where $\Phi_j(x)$ is the activation at jth layer. The style reconstruction loss is calculated in the all four ReLUs of VGG16 backbone network.

With all given loss components, the expression of the total loss is finalized as,

$$\hat{y} = \text{argmin}_y \lambda_c l_{feat}^{\Phi,j}(y, y_c) + \lambda_s l_{style}^{\Phi,j}(y, y_s) + \lambda_{TV} l_{TV}(y) \tag{4}$$

here $\lambda_s, \lambda_c, \lambda_{TV}$ are scalars, y_c is content image and y_s is style image.

2.2. Quantization

Our quantization scheme is directly implemented based on (Banner et al., 2018). In this work, the quantization happens both in feed-forward pass and back propagation. For forward pass, (Banner et al., 2018) uses range BN to facilitate quantization. mathematically, it has following expressions,

$$\hat{x}^{(d)} = \frac{x^{(d)} - \mu^{(d)}}{C(n) \cdot \text{range}(x^{(d)} - \mu^{(d)})} \tag{5}$$

where $\mu^{(d)}$ is the expectation over $x^{(d)}$, n is batch size, $C(n) = \frac{1}{\sqrt{2 \cdot \ln(n)}}$ is scale factor, and $\text{range}(x) = \max(x) - \min(x)$. The idea is to use scale factor and range to estimate standard deviation. As described by (Banner et al., 2018), the rest of quantization scheme follows GEMMLOWP. Given an input tensor x, clamping values $[v_{min}, v_{max}]$,and number of bits$M$ the output is,

$$\text{scale} = (v_{max} - v_{min})/2^M$$

$$\text{zero} - \text{point} = \text{round}(\min(\max(-v_{min}/\text{scale}, 0), 2^M))$$

$$\text{output} = \text{round}(x/\text{scale} + \text{zero} - \text{point})$$

For back propagation, we use STE approach to approximate differentiation through discrete variables.

2.3. Binarization

An extreme quantization is to set the value representation to be binary. This is also called binarization. Binarization usually saves more computational cost because the values are limited to only one bit. In general binarized neural network achieves lower accuracy than quantized neural network and full precision network under the same network architecture. The mechanism of binarization follows the same logic as multi-bit quantization, therefore we do not expand detail on this. We choose (Guo et al., 2019) as our implementation. The reason why we choose another implementation is because BNN specific code simplified computational complexity since it does not need to vary the bit representation.

3. Quantized Style Transfer Architecture

We follow the same transformation network defined by (Johnson et al., 2016). Specifically, the network consists of two part: an image transformation network that transform input image x to stylized output image y, and a loss network for perceptual loss function as in figure 1. The image transformation network consists of 3 set of conv + instance norm + ReLU. Instance norm is defined in (Ulyanov et al., 2017), and we will further discuss this in Section 4. It follows by 5 residual block, each one has two conv layers, two instance norm layers, and one ReLU. It then goes to deconvolutional layers – conv + instance norm + ReLU. The difference between conv layer and deconv layer is that in deconv layer (upsampling layer), the input data will be upsampled by using nearest-neighbor interpolation method.

By default, we use VGG16 defined by (Simonyan & Zisserman, 2015) as our perceptual loss backbone network. In comparison with the PyTorch implementation of VGG16, (Johnson et al., 2016) only uses the first 4 ReLU layers for style and content losses calculation, therefore, only first 23 layers are of our interest.

We apply quantization based on (Banner et al., 2018) and (Qin et al., 2020) to every conv layer in image transformation network. Transformation network parameter details are in figure 2, and the residual architecture is in figure 3.

We use the ImageNet (Deng et al., 2009) pretrained VGG16 weight provided by PyTorch. Due to the limitation of GPU access, we do not, however, use quantized VGG16. Further justification and attempts are provided in Appendix A

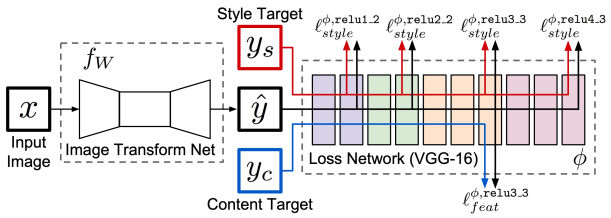

Figure 1. This is taken from (Johnson et al., 2016) as we use their codebase for experiments

4. Experiment

In this section, we present and analysis several quantization experiments on the architecture descibed in Section 3. We use ImageNet pretrained VGG16 backbone model. We train our transformation network on Microsoft COCO2014 dataset (Lin et al., 2014). We train the transformation network 2 epochs for each trial with batch size=4 with MSE loss calculation. We use Adam optimizer and a constant learning rate of 1e-3.

Layer	Activation size
Input	$3 \times 256 \times 256$
$32 \times 9 \times 9$ conv, stride 1	$32 \times 256 \times 256$
$64 \times 3 \times 3$ conv, stride 2	$64 \times 128 \times 128$
$128 \times 3 \times 3$ conv, stride 2	$128 \times 64 \times 64$
Residual block, 128 filters	$128 \times 64 \times 64$
Residual block, 128 filters	$128 \times 64 \times 64$
Residual block, 128 filters	$128 \times 64 \times 64$
Residual block, 128 filters	$128 \times 64 \times 64$
Residual block, 128 filters	$128 \times 64 \times 64$
$64 \times 3 \times 3$ conv, stride $1/2$	$64 \times 128 \times 128$
$32 \times 3 \times 3$ conv, stride $1/2$	$32 \times 256 \times 256$
$3 \times 9 \times 9$ conv, stride 1	$3 \times 256 \times 256$

Figure 2. Detailed parameters by (Johnson et al., 2016)

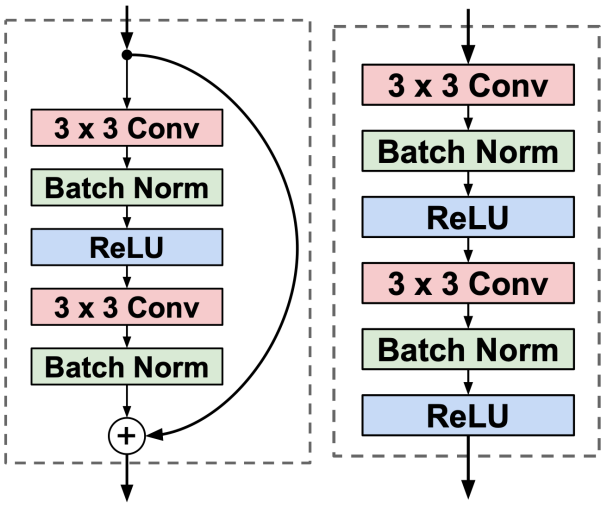

Figure 3. Residual connections by (Johnson et al., 2016)

4.1. Stylization Quality

To investigate how level of quantization affect the quality of neural style transfer, we experimented with image styling under multiple quantization bit and settings.

4.1.1. Complete Quantization

In our first experiment, we quantize all of weight, activation, and gradient by (Banner et al., 2018). We use figure 6 as our content image, and figure 4 as our style image. Figure 7 contains following of experiments: full precision (floating point 32 bit baseline), 16 bit, 8 bit, and 4 bit. As we can see from the resulting graph, there is not significant difference among full precision, 16 bit, and 8 bit setting; when bit goes as low as 4 bit, the details in content image is losing. A zoom-in comparison between full precision and 4 bit is available in figure 7. As we can see, the creases on the wooden door is clearly visible in full precision setting, but less detail revealed in 4 bit quantization setting.

Table 1. Total Loss with Complete Quantization

Quantization bits weight / gradient bits	Total Loss
full/full	440607
16/16	441345
8/8	441941
6/6	450690
4/4	571748

However, this does not significantly affect the styling performance. All three quantized trial show competitive style transfer performance. One notable difference is that the low bit trail appears to have lighter coloration comparing to full precision. One explanation is that, since the quantization scheme we used involving calculatation of the weight statistics (range and bias), the heavy coloration lies outside of the weight distribution and is being clipped out.

Quantitatively, with quantization in all of weight, activation, and gradient, the training total losses of the transformation are in Table 1. As described in Section 2.1, the loss contains both style and content losses. The numerical values of the losses are high, but the difference between quantized and full-precision transform networks are relatively small except 4-bit quantization. This is consistent with visual observation in Figure 7 where output from 4-bit network has the worst quality.

4.1.2. Weight and Activation Quantization

Since our quantization can both apply to gradient and to weight and activaiton, it is natural to think that what if we only quantize weight and activation while leaving gradient as full precision? This heuristic comes from the empirical results that low precision gradient usually leads to bad performance and gradient explosion. Figure 8 shows comparison among full precision, 4 bit weight and gradient, and 4 bit weight with full gradient. From the graph, we can see that with full precision of gradient calculation, the detail preservation increases drastically as the creases are almost entirely preserved from full precision counterpart.

The training total losses of quantizing only weight and activation are in Table 2. The difference between full-precision and quantized transform network losses are again very small, in the 4-bit weight full precision gradient case, the total loss is even slightly smaller than that of the full precision network.

Table 2. Total Loss with Quantized Weight and Activation

QUANTIZATION BITS WEIGHT / GRADIENT BITS	TOTAL LOSS
FULL/FULL	440607
16/FULL	441395
8/FULL	441463
4/FULL	440488

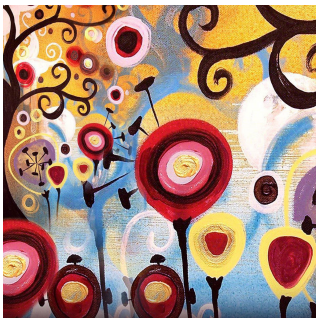

Figure 4. style image: candy.jpg

Figure 5. style image: mosaic.jpg

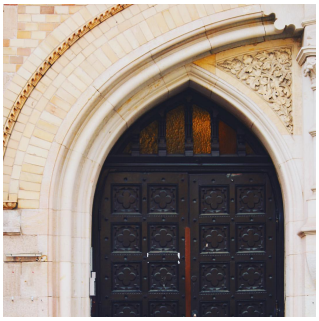

Figure 6. content image: amber.jpg

We have also conducted experiments other style images with similar results. For example, in figure 9, we apply different quantization bit on mosaic style image. We can also see that in 4 bit setting, detail is mildly lost and color spectrum is less varied as full precision model. Similarly, we have tried setting when gradient does not quantize. Figure 10 shows that when gradient remains full precision, the resulting output image is almost identical as full precision baseline. This

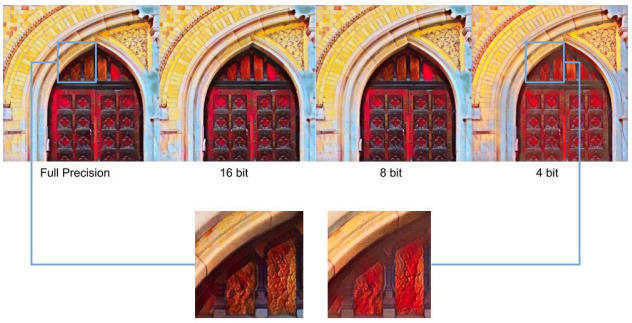

Figure 7. Different quantization bit on transformation network

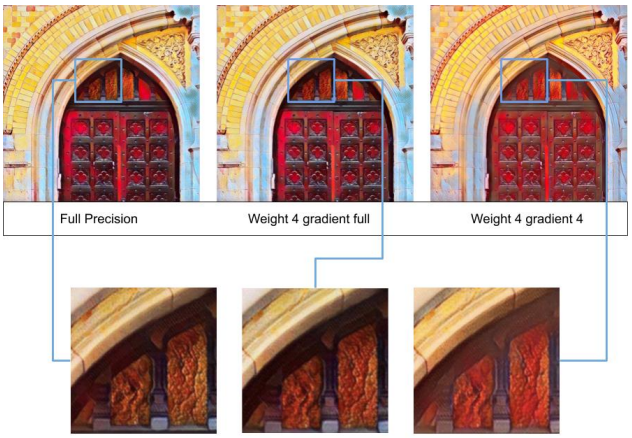

Figure 8. Comparison between quantized gradient and full gradient

is not surprising since VGG16 is large network and typically over-parameterized. Therefore we can reduce model size while achieving comparable results.

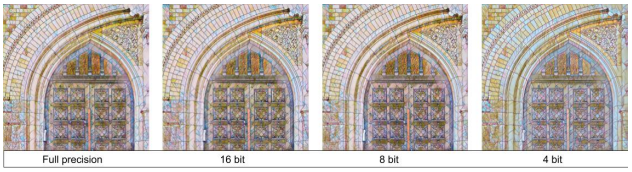

Figure 9. Different quantization bit on transformation network on mosaic style

4.1.3. BINARIZATION

Our third experiment binarizes the transform network to use either 1-bit weight 1-bit activation (1w1a) or 1-bit weight 1-bit activation (1w1a). Initial results with 2 epochs of training using constant 1e-3 learning rate produced very bad image quality as shown in Figure 11. Training loss on 1w32a

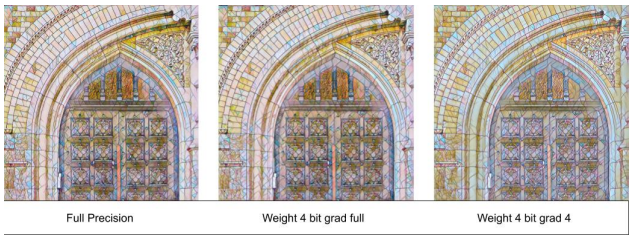

Figure 10. Comparison between quantized gradient and full gradient on mosaic style

and 1w1a are 1278227 and 5118470, respectively. The losses are an order of magnitude higher that all quantization experiments in Section 4.1.1 and 4.1.2. In 1w32a, styling appears inconsistent across regions with common hightlight and shadow regions. In 1w1a, neither contents nor styles were identifiable.

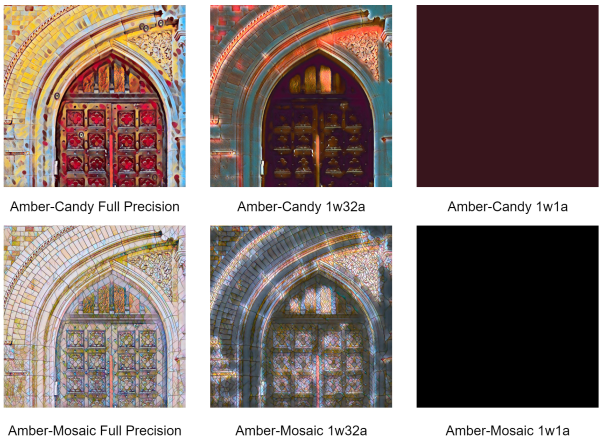

Figure 11. Poor Quality with BNN under Default Training Schedule

The training schedule clearly needs improvements for the binarized transform network. While increasing the number of epochs or the batch size can certainly improve the result, we keep them the same to make fair comparisons across our experiments. Adam is already an effective optimizer, but the constant learning rate of 1e-3 is too high for the binary transformer. Observation of the training logs shows that the loss drops quickly in the beginning and comes to a halt only a few 10000 batches into the first epoch. Further training often increase the loss. In a refined schedule, we adopt an aggressive learning rate decay schedule. The learning rate starts at 1e-3 and decays down with a 0.1 multiplier every a few 10000 batches within each epoch. Under this schedule, we observe that the loss drops continuously all throughout the training process. Although the final results are still much worse that the full-precision or the quantized versions, they do show significant improvements over the default schedule. The total loss values are shown in Table 3 and representative comparisons are shown in Figure 12. 1w1a still fails to show any meaningful result but 1w32a displays significant improvements.

Table 3. Total Loss with Binarization

Configuration	Total Loss
Full precision	440607
1w32a default schedule	1278227
1w32a refined schedule	954718
1w1a default schedule	5118470
1w1a refined schedule	5043898

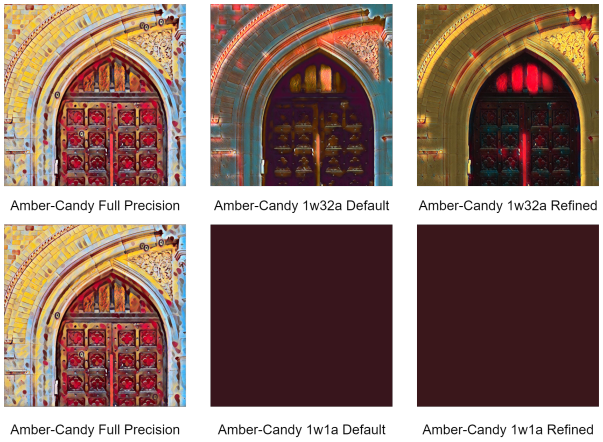

Figure 12. Better Quality with BNN under Refined Training Schedule

4.2. Computational Cost

We expect the computational cost of our quantized neural style transfer network to be significantly smaller than that of the original, full precision network. To asses this computational advantage quantitatively, we calculate the effective number of floating point operations (FLOPs) of our quantized transformer network.

We follow THOP, the widely accepted pytorch Ops counter as the backbone tool for our calculation. In particular, we create a distinction between quantized operations (q) and full precision operations (f) in our calculation. The final effective FLOPs of a quantized network is calculated as,

$$\text{Effective FLOPs} = q \times \frac{\text{Quantized bits}}{\text{Full Precision bits}} + f \tag{6}$$

Table 4 compares FLOPs of various quantized transform

Table 4. FLOPs for quantized and full precision transform network.

QUANTIZATION BITS WEIGHT / GRADIENT BITS	QUANTIZED OPS(M)	FULL PRECISION OPS(M)	TOTAL OPS(M)	REDUCTION(%)
0/0	0	157.86	157.86	-
16/16	78.74	0.38	79.12	49.88
16/0	118.11	0.38	118.49	24.94
8/8	39.37	0.38	39.75	**74.82**
8/0	98.43	0.38	98.80	37.41
6/6	29.53	0.38	29.90	81.06
4/4	19.69	0.38	20.06	87.29
4/0	88.58	0.38	88.96	43.65
3/0	86.12	0.38	86.50	45.20
2/0	83.66	0.38	84.04	16.76
1/0	81.20	0.38	81.58	48.32

networks with the full precision one (0 bits means full precision). The quantized transform networks generally observe significant reductions in FLOPs, with the extreme of 87.29% reduction when the weight and the gradient are both quantized to 4 bits. Prior observations show that a weight 4 gradient 4 quantization introduces notable degradation in the style transferring quality, but the near 90% reduction in computation justifies its usage in mobile and edge devices with extreme constrains in computational power. A balance of cost and quality can be achieved with 8 bit weight and 8 bit gradients. Such configuration achieves 74.82% reduction in FLOPs, yet keeps style transfer quality nearly identical as a full precision network.

4.3. Cost-Quality Balance

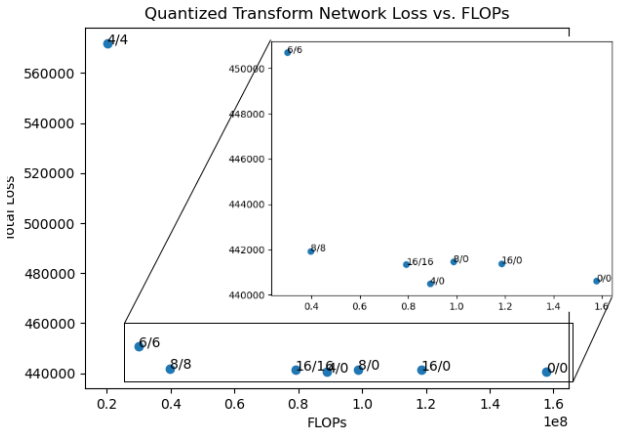

Figure 13. Quantized Transform Network Loss vs. FLOPs

Synthesizing the experimental results and calculated FLOPs, Figure 13 shows the transform network total loss ploted against the amount of FLOPs. All quantized neural networks, expect 4/4 showed losses very close to the full precision counterpart while displaying various amount of significant FLOPs reductions. 8/8 demonstrates the power of quantization in its 74% FLOPs reduction while keeping nearly identical loss. 6/6 is approach the lower bound of the quantization bits while 4/4 shows the limitation where precision reduction begin to increase the loss significantly. This result is consistent with the visual examination of the stylized images in Section 4.

4.4. BatchNorm vs InstanceNorm

In this section, we explore the differences between using instance normalization and batch normalization. Batch normalization was first brought up by (Ioffe & Szegedy, 2015), and hence widely used in neural network architecture design. Batch normalization captures and transforms batch statistic in an affine manner. Although it is not theoretically well understood, batch normalization has been widely used in neural architecture design. (Ulyanov et al., 2017) argues a slightly change of batch normalization, instance normalization, better serves the purpose of neural style transfer. Instead of manipulating the entire batch statistics, instance normalization works on each instance's statistics. The rationale behind this change is that the result of stylization should not depend on the contrast of the content image. The generator network should discard contrast information in the content image. To further understand whether this change is necessary for low bit setting, we compare the use of batch normalization and instance normalization in transformation network under three settings – full precision and 4 bit. We use nn.BatchNorm2d and nn.InstanceNorm respectively. The resulting pictures are shown in figure 14. Contrary to (Ulyanov et al., 2017), we did not observe significant improvement by replacing BatchNorm with InstanceNorm in both full precision setting and low bit setting. One possible reasoning for this phenomenon is that the batch size we are using (4 images per batch) is small, therefore it does not appear to have visible improvement.

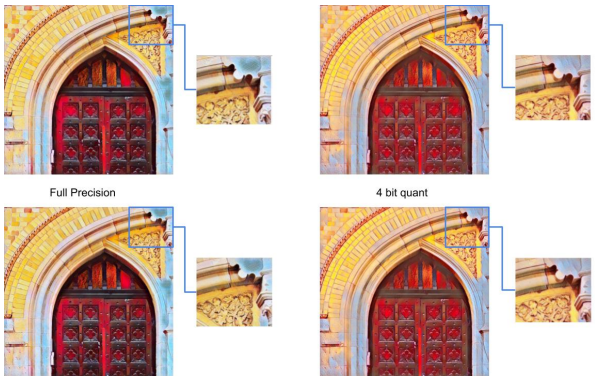

Figure 14. The upper row are pictures using InstanceNorm. The lower row of pictures are using BatchNorm.

5. Conclusion

We present a novel application of network quantization to neural style transfer image generation. We demonstrate that with a clever choice of quantizaiton bits, a quantized transform network can produce high quality output images with over 70% savings in computational costs. We also compare the difference between using instance normalization an batch normalization under both full precision and 4 bit precision setting.

References

Banner, R., Hubara, I., Hoffer, E., and Soudry, D. Scalable methods for 8-bit training of neural networks, 2018.

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1, 2016.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Gatys, L. A., Ecker, A. S., and Bethge, M. Image style transfer using convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2414–2423, 2016. doi: 10.1109/CVPR.2016.265.

Guo, J., Zhan, Z., Gao, Y., Xiao, Y., Lou, J.-G., Liu, T., and Zhang, D. Towards complex text-to-sql in cross-domain database with intermediate representation, 2019.

Guo, Y. A survey on methods and theories of quantized neural networks, 2018.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

Johnson, J., Alahi, A., and Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution, 2016.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. Microsoft coco: Common objects in context. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T. (eds.), *Computer Vision – ECCV 2014*, pp. 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1.

Qin, H., Gong, R., Liu, X., Shen, M., Wei, Z., Yu, F., and Song, J. Forward and backward information retention for accurate binary neural networks. In *IEEE CVPR*, 2020.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition, 2015.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Instance normalization: The missing ingredient for fast stylization, 2017.

Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., and Zou, Y. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients, 2018.

A. Attempts At Quantizing Loss Network Over Cifar

As detailed in Section 3, the transform network has ImageNet-pretrained VGG-16 as its loss network. While our limited computing power prohibited us from quantized training on ImageNet, we were in the hope to use Cifar to obtain comparable results. We planned to train quantized and full-precision VGG-16 on Cifar, append them to quantized and full-precision transform networks, and compare the stylization results.

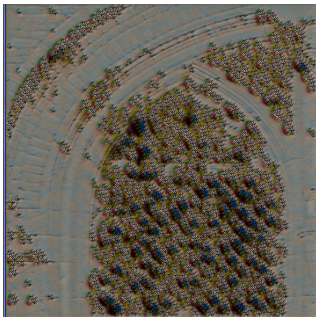

Figure 15. Output of Full Precision Transform Network with Full Precision Cifar100-Pretrained Loss Network

We realized even with full-precision transform network and loss network, results on Cifar was disappointing. Figure 15 shows the stylizing result of content image amber (Figure 6) and style image mosaic (Figure 5) using full-precision transform network and full-precision Cifar100-pretrained Loss Network. We believe the small size of Cifar limited the learning of VGG-16 so that the feature extraction power of the first few convolution layers are much inferior than the ImageNet-pretrained counterpart.