Deep Neural Compression for Adaptive Instance Normalization Style Transfer

Ishank Agrawal
MIT
ishank@mit.edu

Nithin Kavi
Harvard College
nithinkavi@college.harvard.edu

Abstract

Style transfer is the task of transfering the style of one image onto the content of another image. Neural algorithms can perform this for arbitray styles and contents. However, conventionally such algorithms are slow. In this work, we apply neural compression techniques including pruning and quantization to optimize existing algorithms, and we show a significant speed up without losing quality.

Demo: `https://youtu.be/2DIhZrFUk9k`
Code: `https://github.com/mathletema/adain-style-mobile`

1 Introduction

Over the past decade, neural style transfer has become increasingly common. In this paper, we define neural style transfer as follows: given a style image and a content image, output a new image that represents the content image in the style of the style image. Typically, the style image is a famous painting such as Van Gogh's *Starry Night* or Dali's *The Persistence of Memory* but it can be any image.

There are many different types of Neural Style Transfer. The most general form is arbitrairy neural style transfer, but this is hard to do well because it is so general and enforces a “one size fits all" setting. Some other examples are included in the following flowchart:

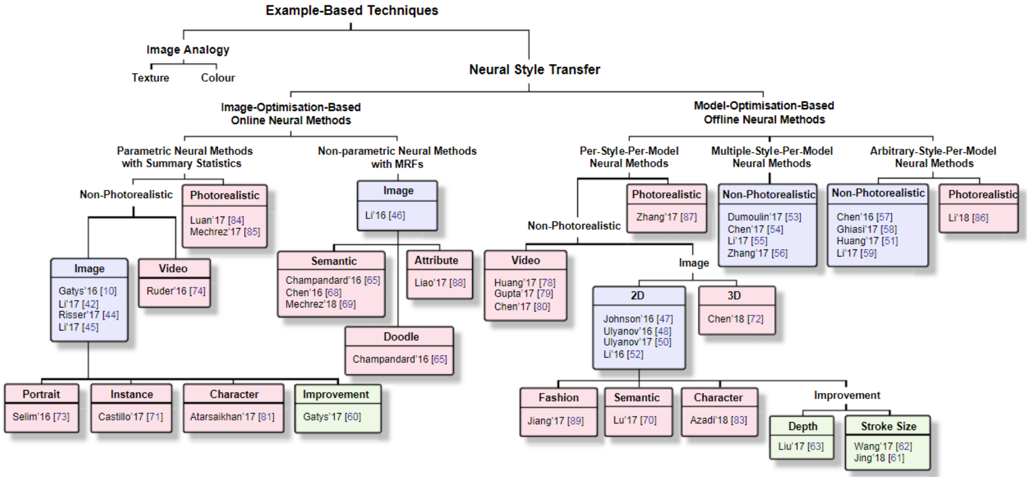

Figure 1: Various neural style transfer algorithms proposed [8]

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

In this work, we extend the work of Huang and Belongie in [1]. They showed that the means and variances of the features learned by a VGG-19 network is highly predictive of the style. Thus by applying an affine transformation of the features of the content image to match the mans and variances of the features of the style image and learning a decoder can perform arbitrary neural stye transfer with satisfactory performance.

The original architecture by [4] used VGG-19 as a fixed encoder, and only learned a decoder. Here we generalize their architecture to support arbitrary architecture encoders and decoders, which allow us to compress both of them. We utilize pruning and quantization to achieve the compression [2].

This paper is divided into multiple sections. In section 2, we summarize Adaptive Instance Normalization as described in [4]. In section 2, we discuss our modified architecture. In section 3 we describe our optimizations incluing our appraoch to pruning and quantization. In section 4, we compare performance of the optimized network versus the original one. Finally, we conclude and discuss potential future directions for this work.

2 Overview of Adaptive Instance Normalization

As mentioned earlier, Adaptive Instance Normalization, or AdaIN, is a fast neural algorithm for real time style transfer. Let $\textsc{vgg-19}$ refer to a VGG-19 neural network pretrained on the ImageNet dataset. Then AdaIN creates a generated image x_G from any content image x_C and ant style image x_S through:

$$\mathrm{y}_C = \textsc{vgg-19}(\mathrm{x}_C), \quad \mathrm{y}_S = \textsc{vgg-19}(\mathrm{x}_S) \tag{1}$$

$$\mathrm{y}_G = \textsc{adain}(\mathrm{y}_C, \mathrm{y}_S) \tag{2}$$

$$\mathrm{x}_G = \textsc{decoder}(\mathrm{y}_G) \tag{3}$$

where $\textsc{adain}$ refers to the affine transform (assuming $\mu(x)$ and $\sigma(x)$ are the means and variances of x):

$$\textsc{adain}(x, y) = \sigma(y)\left(\frac{x - \mu(x)}{\sigma(x)}\right) + \mu(y) \tag{4}$$

The decoder is then trained to minimize

$$\mathcal{L} = \mathcal{L}_c + \lambda\mathcal{L}_s \tag{5}$$

where

$$\mathcal{L}_c = \|\textsc{vgg-19}(\mathrm{x}_G) - \mathrm{y}_G\|_2 \tag{6}$$

$$\mathcal{L}_s = \sum_{i=1}^{L} \|\mu(\textsc{vgg-19}^{(i)}(\mathrm{x}_G)) - \mu(\textsc{vgg-19}^{(i)}(\mathrm{x}_S))\|_2 \quad + \tag{7}$$

$$\sum_{i=1}^{L} \|\sigma(\textsc{vgg-19}^{(i)}(\mathrm{x}_G)) - \sigma(\textsc{vgg-19}^{(i)}(\mathrm{x}_S))\|_2 \tag{8}$$

Note that since the $\textsc{vgg} - 19$ encoder is fixed, it will remain a computational bottleneck even after compressing the decoder. Several possible methods exist to compress the encoder. Firstly, it is possible to train a smaller model such as MobileNetV2 [3] to lean VGG-19 representations through knowledge distillation methods similar to [1]. However, this imposes unnecessary contraints on the feature space.

A second approach is to simply prune the existing encoder. However the VGG-19 encoder is also used to score the neural network's performance. Thus an encoder that outputs 0 for any input image trivially minimizes the loss without actually performing the task. This behavior is obviously unwanted.

Thus we propose separating the scoring encoder from the encoder used in the style transfer network by using pretrained VGG-19 solely for scoring.

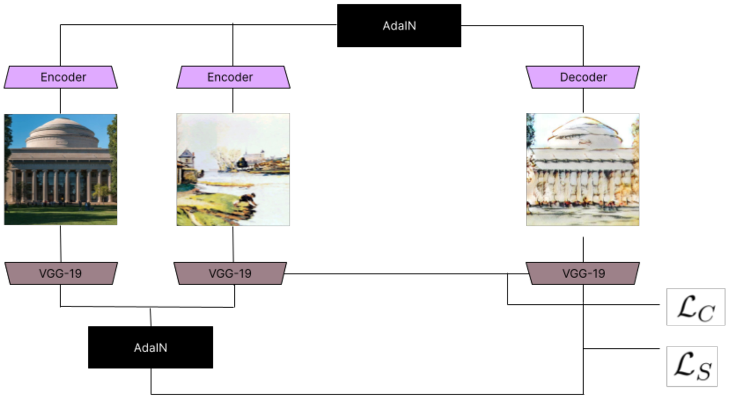

Figure 2: Our proposed architecture: AdaIN neural style transfer with an arbitrary encoder

3 Proposed Architecture

For an arbitrary encoder ENCODER and decoder DECODER, we compute the output image as

$$\mathrm{y}_C = \text{ENCODER}(\mathrm{x}_C), \quad \mathrm{y}_S = \text{ENCODER}(\mathrm{x}_S) \tag{9}$$

$$\mathrm{y}_G = \text{ADAIN}(\mathrm{y}_C, \mathrm{y}_S) \tag{10}$$

$$\mathrm{x}_G = \text{DECODER}(\mathrm{y}_G) \tag{11}$$

For scoring, we first compute the VGG-19 features through

$$\mathrm{y}_C^{\text{vgg}} = \text{VGG-19}(\mathrm{x}_C) \tag{12}$$

$$\mathrm{y}_S^{\text{vgg}} = \text{VGG-19}(\mathrm{x}_S) \tag{13}$$

$$\mathrm{y}_G^{\text{vgg}} = \text{VGG-19}(\mathrm{x}_G) \tag{14}$$

$$\tag{15}$$

We then define our target loss to be

$$\mathcal{L} = \mathcal{L}_c + \lambda \mathcal{L}_s \tag{16}$$

where

$$\mathcal{L}_c = \|\mathrm{y}_G^{\text{vgg}} - \text{ADAIN}(\mathrm{y}_C^{\text{vgg}}, \mathrm{y}_S^{\text{vgg}})\|_2 \tag{17}$$

$$\mathcal{L}_s = \sum_{i=1}^{L} \|\mu(\mathrm{y}_G^{\text{vgg}}) - \mu(\mathrm{y}_S^{\text{vgg}})\|_2 \quad + \tag{18}$$

$$\sum_{i=1}^{L} \|\sigma(\mathrm{y}_G^{\text{vgg,i}}) - \sigma(\mathrm{y}_S^{\text{vgg,i}})\|_2 \tag{19}$$

Here $\mathrm{y}^{\text{vgg,i}}$ refers to the ith layer of the $\text{VGG} - 19$ representation. An summary of this architecture is given in Figure 2.

Note that this architecture reduces to the original one proposed by [4] when ENCODER is frozen to VGG-19. Thus this architecture generalizes to support arbitrary encoders. In our optimization process, we start with the pretrained encoder (simply VGG-19) and decoder (from original paper), and iteratively compress and retrain against this target.

Finally, we trained this architecture using content images from the MS-COCO [6] dataset and style images from ArtBench [5] which is a subset of the WikiArt [7] dataset.

4 Compression

We first prune our model. We iteratively pick symmetric layers from the encoder and decoder stack and prune off the least important channels. Channel importance is determined by the L2 norm of the respective weight matrix. We then retrained the entire model against the target described earlier. We only pruned the later layers of the encoder (and corresponding earlier layers of the decoder), as the first few layers were not a bottleneck, and important for the quality of the resulting image. The resulting compression and speedup are given below.

Model	# of Encoder Params	# of Decoder Params	Compression	Time [1]	Speedup
AdaIN (original)	3.5 M	3.5 M		7.3s	
AdaIN pruned	1.4 M	1.4 M	2.5x	3.9s	1.87x

Table 1: Compression after pruning

Following this, we quantized the weights of the convolution layers, and retrained the model. The results of this step are given below:

Model	# Model Size	Compression	Time [2]	Speedup
AdaIN (original)	26.8MB	1x	7.3s	1x
AdaIN pruned	10.7MB	2.5x	3.9s	1.87x
AdaIN pruned + quantized	2.73MB	9.8x	2.4s	3.04x

Table 2: Compression after pruning + quantization

Qualitative outputs of the resulting images from the original network and our network are given below.

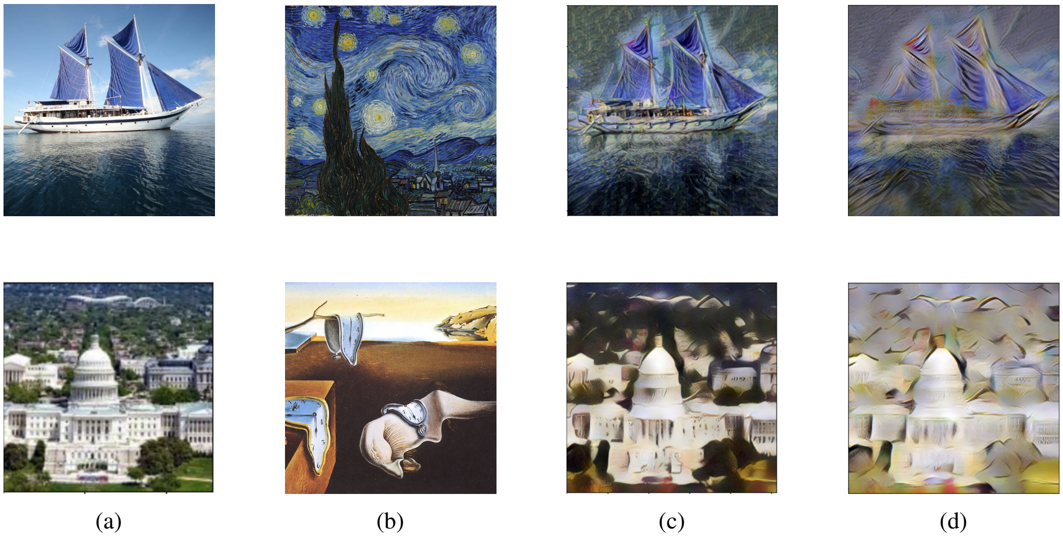
(a) (b) (c) (d)

Figure 3: Qualitative comparison of outputs. Here (a) content image, (b) style image, (c) original network, (d) our compressed network with pruning + quantization

We can see in both cases that we are able to mostly replicate the performance with the original VGG weights, with slight loss in quality.

5 Conclusions and Future Research

Our analysis shows that existing neural style transfer networks can be significantly compressed with minimal degradation in image quality. This allows us to perform fast neural style transfer on CPU machines.

Arbitrary neural style transfers have lower image quality as a result of their flexibility. It would be interesting to see how this approach generalizes to other neural style transfer algorithms.

We also mentioned the possibility of using a different architecture such as MobileNetV2. It would be interesting to combine knowledge distillation and our proposed architecture for even faster neural style transfer.

Acknowledgments

We would like to thank Professor Song Han and graduate students Han Cai and Ji Lin for running the course MIT 6.5940: Tiny ML and Efficient Deep Learning in Fall 2023 and teaching us the skills to conduct this research project.

References

[1] Porting arbitrary style transfer to the browser. `https://magenta.tensorflow.org/blog/2018/12/20/style-transfer-js/`.

[2] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016.

[3] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017.

[4] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization, 2017.

[5] Peiyuan Liao, Xiuyu Li, Xihui Liu, and Kurt Keutzer. The artbench dataset: Benchmarking generative models with artworks, 2022.

[6] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[7] Saif Mohammad and Svetlana Kiritchenko. WikiArt emotions: An annotated dataset of emotions evoked by art. In Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA).

[8] Xin Wang, Geoffrey Oxholm, Da Zhang, and Yuan-Fang Wang. Multimodal transfer: A hierarchical deep convolutional neural network for fast artistic style transfer, 2017.