Positional Normalization (Supplementary Material)

Boyi Li[1,2]*, **Felix Wu**[1]*, **Kilian Q. Weinberger**[1], **Serge Belongie**[1,2]
[1]Cornell University [2]Cornell Tech
{bl728, fw245, kilian, sjb344}@cornell.edu

Appendices

A Algorithm of PONO-MS

The implementation of PONO-MS in TensorFlow [1] an PyTorch[7] are shown in Listing 1 and 2 respectively.

```
# x is the features of shape [B, H, W, C]

# In the Encoder
def PONO(x, epsilon=1e-5):
    mean, var = tf.nn.moments(x, [3], keep_dims=True)
    std = tf.sqrt(var + epsilon)
    output = (x - mean) / std
    return output, mean, std

# In the Decoder
# one can call MS(x, mean, std)
# with the mean and std are from a PONO in the encoder
def MS(x, beta, gamma):
    return x * gamma + beta
```

Listing 1: PONO and MS in TensorFlow

```
# x is the features of shape [B, C, H, W]

# In the Encoder
def PONO(x, epsilon=1e-5):
    mean = x.mean(dim=1, keepdim=True)
    std = x.var(dim=1, keepdim=True).add(epsilon).sqrt()
    output = (x - mean) / std
    return output, mean, std

# In the Decoder
# one can call MS(x, mean, std)
# with the mean and std are from a PONO in the encoder
def MS(x, beta, gamma):
    return x * gamma + beta
```

Listing 2: PONO and MS in PyTorch

*: Equal contribution.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

B Equations of Existing Normalization

Batch Normalization (BN) computes the mean and std across B, H, and H dimensions, i.e.

$$\mu_c = \mathbb{E}_{b,h,w}[X_{b,c,h,w}], \quad \sigma_c = \sqrt{\mathbb{E}_{b,h,w}[X^2_{b,c,h,w} - \mu_c] + \epsilon},$$

where ϵ is a small constant applied to handle numerical issues.

Synchronized Batch Normalization views features of mini-batches across multiple GPUs as a single mini-batch.

Instance Normalization (IN) treats each instance in a mini-batch independently and computes the statistics across only spatial dimensions, i.e.

$$\mu_{b,c} = \mathbb{E}_{h,w}[X_{b,c,h,w}], \quad \sigma_{b,c} = \sqrt{\mathbb{E}_{h,w}[X^2_{b,c,h,w} - \mu_{b,c}] + \epsilon}.$$

Layer Normalization (LN) normalizes all features of an instance within a layer jointly, i.e.

$$\mu_b = \mathbb{E}_{c,h,w}[X_{b,c,h,w}], \quad \sigma_b = \sqrt{\mathbb{E}_{c,h,w}[X^2_{b,c,h,w} - \mu_b] + \epsilon}.$$

Finally, Group Normalization (GN) lies between IN and LN, it devides the channels into G groups and apply layer normalization within a group. When $G = 1$, GN becomes LN. Conversely, when the $G = C$, it is identical to IN. To define it formally, it computes

$$\mu_{b,g} = \mathbb{E}_{c\in S_g,h,w}[X_{b,c,h,w}], \quad \sigma_{b,g} = \sqrt{\mathbb{E}_{c\in S_g,h,w}[X^2_{b,c,h,w} - \mu_{b,g}] + \epsilon},$$

where $S_g = \{\lceil \frac{(g-1)C}{G} + 1\rceil, \ldots, \lceil \frac{gC}{G} \rceil\}$.

C PONO Statistics of Models Pretrained on ImageNet

Figure 1 shows the means and the standard deviations extracted by PONO based on the features generated by VGG-19 [8], ResNet-152 [3], and DenseNet-161 [4] pretrained on ImageNet [2].

D Implementation details

We add PONO to the encoder right after a convolution operation and before other normalization or nonlinear activation function. Figure 2 shows the model architecture of CycleGAN [9] with Positional Normalization. Pix2pix [5] uses the same architecture.

E Qualitative Results Based on CycleGAN and Pix2pix

We show some outputs of CycleGAN in Figure 3. The Pix2pix outputs are shown in Figure 4.

F Qualitative Results Based on DRIT and MUNIT.

We randomly sample 10 *cat and dog* image pairs and show the outputs of DRIT, DRIT + PONO-MS, MUNIT, and MUNIT' PONO-MS in Figure 5.

G PONO in Image Classification

To evaluate PONO on image classification task, we add PONO to the begining of each ResBlock of ResNet-18 [3] (also affects the shortcut). We followed the common training procedure base on Wei Yang's open sourced code [2] on ImageNet [6]. Figure 6 shows that with PONO, the training loss and error are reduced significantly and the validation error also drops slightly from 30.09 to 30.01. Admittedly, this is not a significant improvement. We believe that this result may inspire some future architecture design.

[2] https://github.com/bearpaw/pytorch-classification

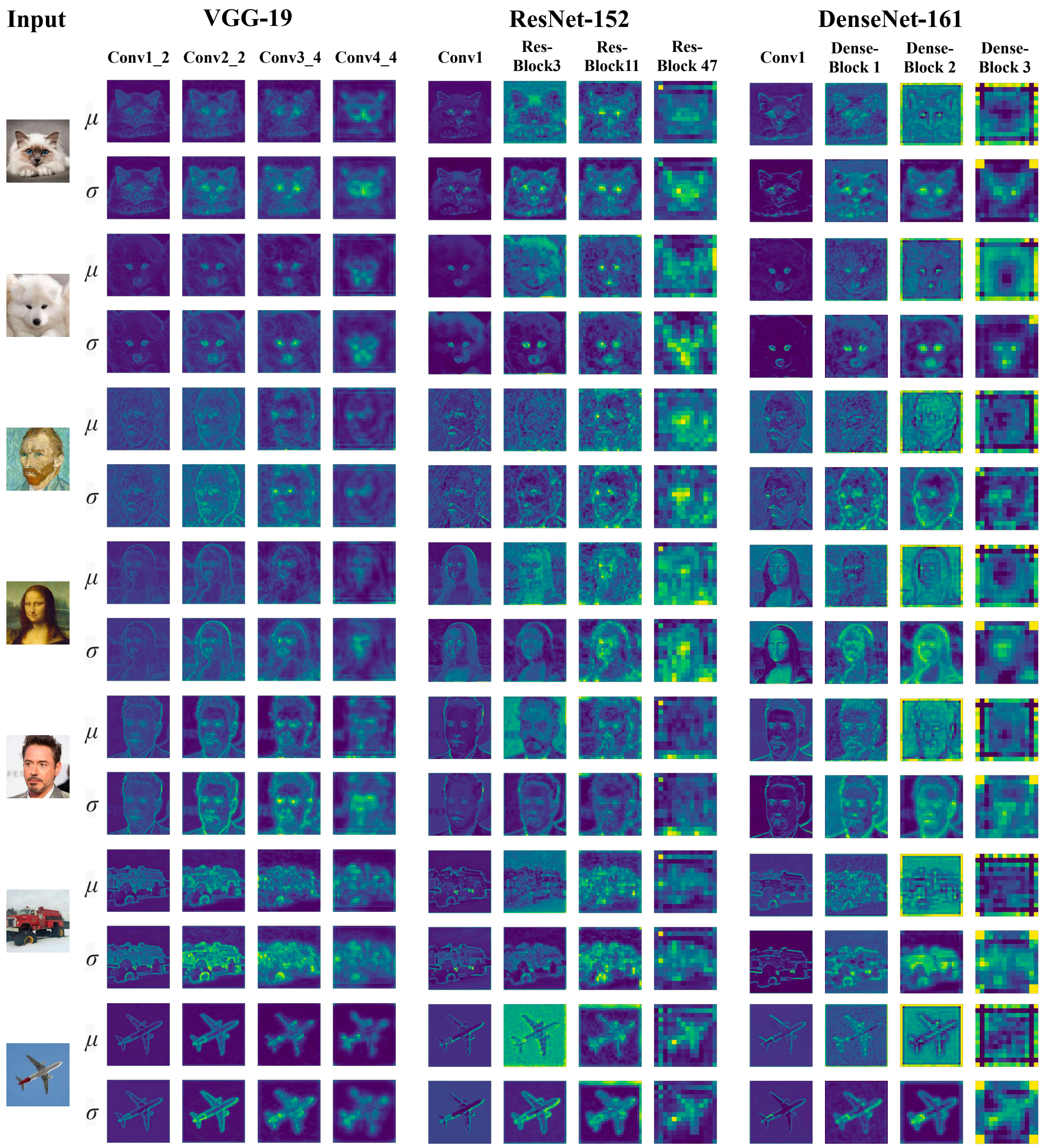

Figure 1: We extract the PONO statistics from VGG-19, ResNet-152, and Dense-161 at layers right before downsampling (max-pooling or strided convolution).

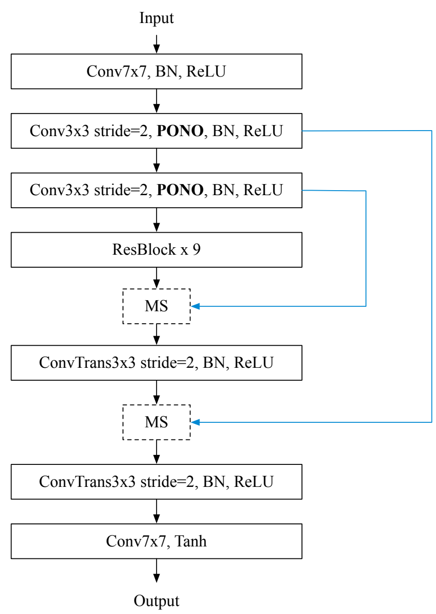

Figure 2: The generator of CycleGAN + PONO-MS. Pix2pix uses the same architecture. The operations in a block is applied from left to right sequentially. The **blue** lines show how the first two moments are passed. ConvTrans stands for transposed convolution. Each ResBlock has Conv3x3, BN, ReLU, Conv3x3, and BN.

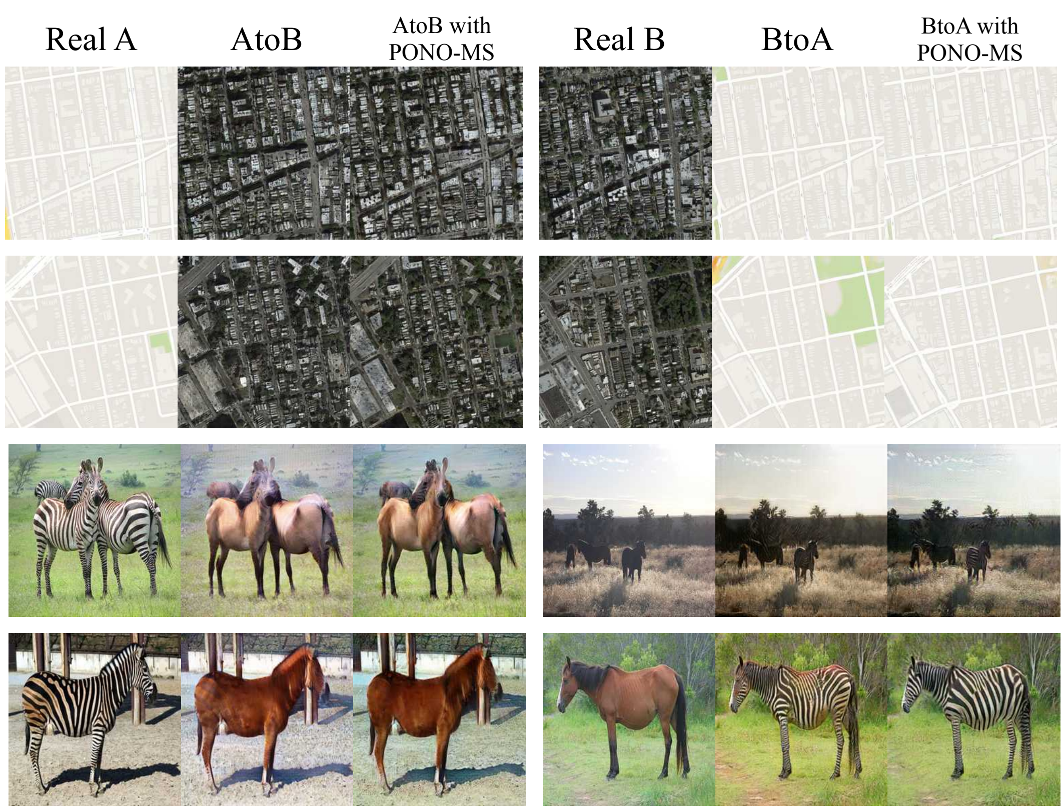

Figure 3: Qualitative results of CycleGAN (with/without PONO-MS) with randomly sampled inputs.

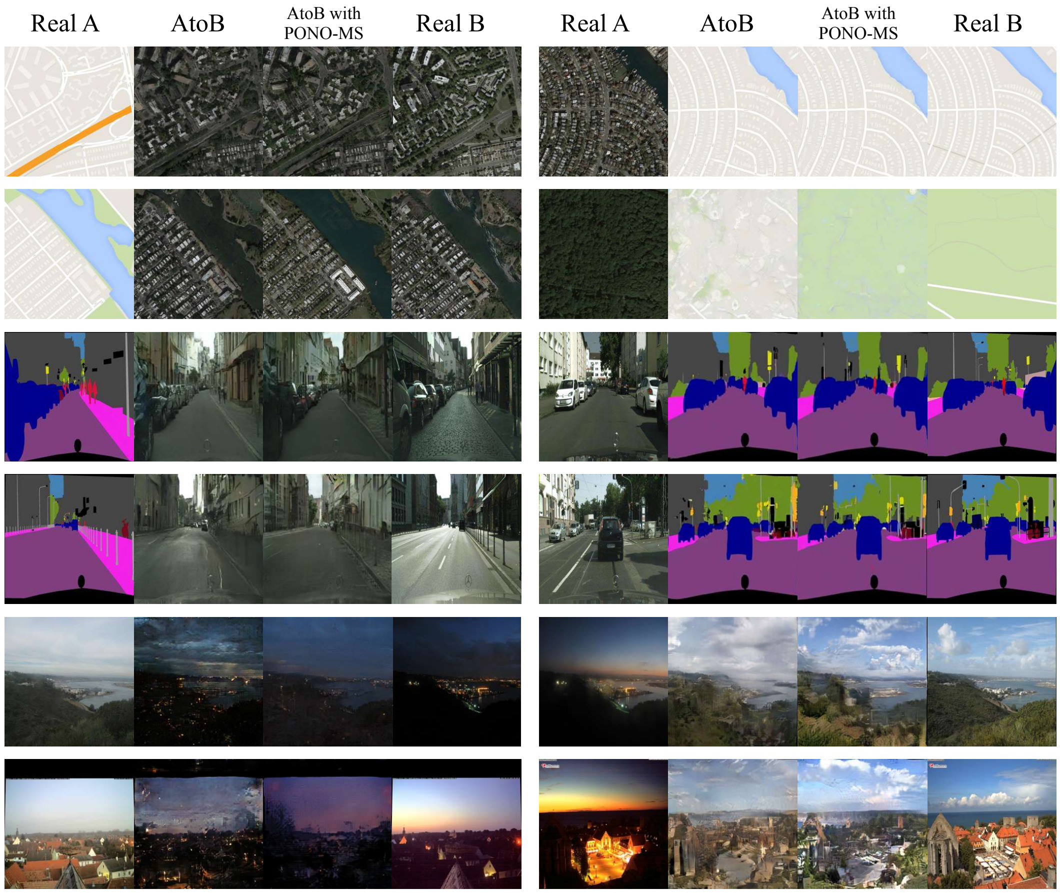

Figure 4: Qualitative results of Pix2pix (with/without PONO-MS) with randomly sampled inputs.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, and et al. M. Isard. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[5] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[7] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *Proc. of ICLR*, 2015.

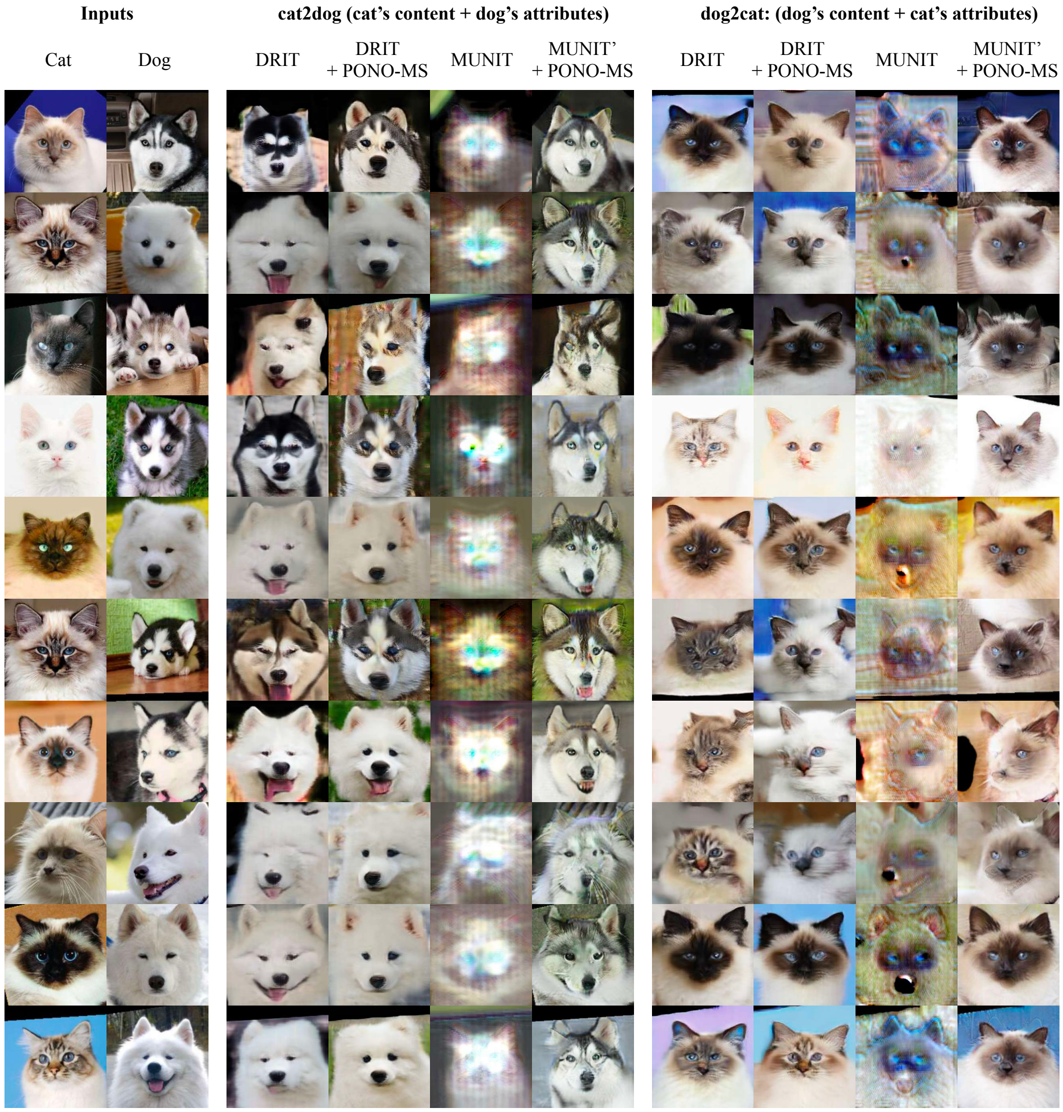

Figure 5: Qualitative results of DRIT and MUNIT (with/without PONO-MS) with randomly sampled inputs.

[9] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

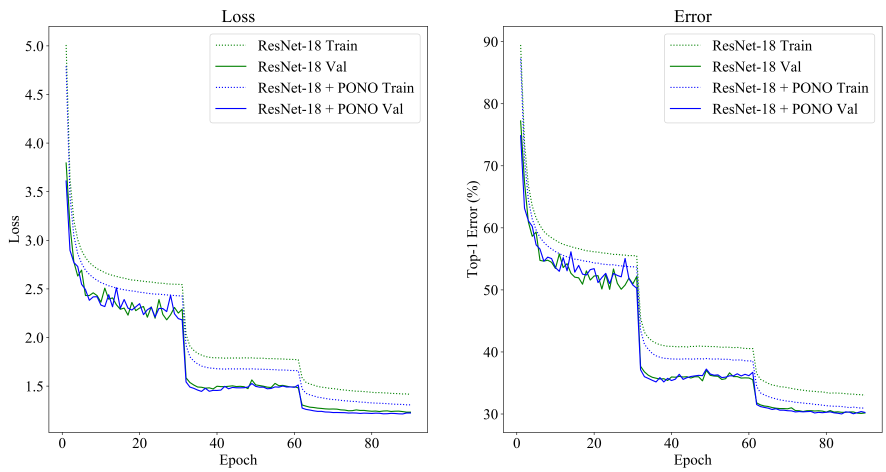

Figure 6: Training and validation curves of ResNet-18 and ResNet-18 + PONO on ImageNet.