Positional Normalization

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

Abstract

A popular method to reduce the training time of deep neural networks is to normalize activations at each layer. Although various normalization schemes have been proposed, they all follow a common theme: normalize across spatial dimensions and discard the extracted statistics. In this paper, we propose an alternative normalization method that noticeably departs from this convention and normalizes exclusively across channels. We argue that the channel dimension is naturally appealing as it allows us to extract the first and second moments of features extracted at a particular image position. These moments capture structural information about the input image and extracted features, which opens a new avenue along which a network can benefit from feature normalization: Instead of disregarding the normalization constants, we propose to re-inject them into later layers to preserve or transfer structural information in generative networks. Codes are available at https://github.com/Boyiliee/PONO.

1 Introduction

A key innovation that enabled the undeniable success of deep learning is the internal normalization of activations. Although normalizing inputs had always been one of the "tricks of the trade" for training neural networks [39], batch normalization (BN) [29] extended this practice to every layer, which turned out to have crucial benefits for deep networks. While the success of normalization methods was initially attributed to "reducing internal covariate shift" in hidden layers [29, 41], an array of recent studies [2, 3, 5, 25, 48, 60, 69, 77] has provided evidence that BN changes the loss surface and prevents divergence even with large step sizes [5], which accelerates training [29].

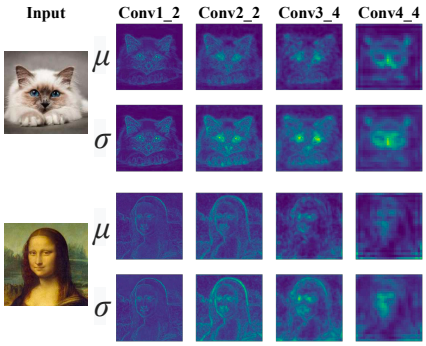

Figure 1: The mean μ and standard deviation σ extracted by PONO at different layers of VGG-19 capture structural information from the input images.

Multiple normalization schemes have been proposed, each with its own set of advantages: Batch normalization [29] benefits training of deep networks primarily in computer vision tasks. Group normalization [74] is often the first choice for small mini-batch settings such as object detection and instance segmentation tasks. Layer Normalization [41] is well suited to sequence models, common in natural language processing. Instance normalization [68] is widely used in image synthesis owing to its apparent ability to remove style information from the inputs. However, all aforementioned normalization schemes follow a common theme: they normalize across spatial dimensions and discard the extracted statistics. The philosophy behind their design is that the first two moments are considered expendable and should be removed.

*: Equal contribution.

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

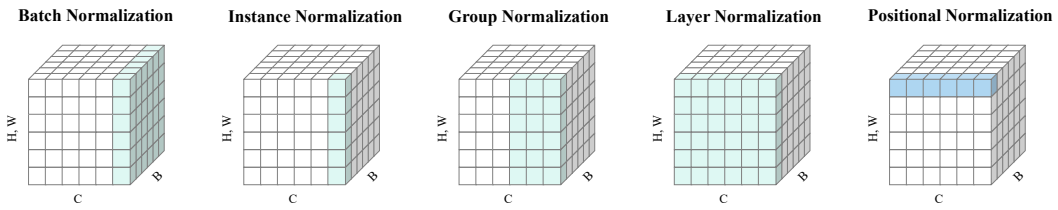

Figure 2: Positional Normalization together with previous normalization methods. In the figure, each subplot shows a feature map tensor, with B as the batch axis, C as the channel axis, and (H, W) as the spatial axis. The entries colored in **green** or **blue** (ours) are normalized by the same mean and standard deviation. Unlike previous methods, our method processes each position independently, and compute both statistics across the channels.

In this paper, we introduce Positional Normalization (PONO), which normalizes the activations at each position independently across the channels. The extracted mean and standard deviation capture the coarse structural information of an input image (see Figure 1). Although removing the first two moments does benefit training, it also eliminates important information about the image, which — in the case of a generative model — would have to be painfully relearned in the decoder. Instead, we propose to bypass and inject the two moments into a later layer of the network, which we refer to as Moment Shortcut (MS) connection.

PONO is complementary to previously proposed normalization methods (such as BN) and as such can and should be applied jointly. We provide evidence that PONO has the potential to substantially enhance the performance of generative models and can exhibit favorable stability throughout the training procedure in comparison with other methods. PONO is designed to deal with spatial information, primarily targeted at generative [20, 30] and sequential models [24, 33, 58, 65]. We explore the benefits of PONO with MS in several initial experiments across different model architectures and image generation tasks and provide code online at https://github.com/Boyiliee/PONO.

2 Related Work

Normalization is generally applied to improve convergence speed during training [51]. Normalization methods for neural networks can be roughly categorized into two regimes: *normalization of weights* [50, 55, 59, 73] and *normalization of activations* [29, 31, 37, 41, 47, 49, 61, 68, 74]. In this work, we focus on the latter.

Given the activations $X \in \mathbb{R}^{B\times C\times H\times W}$ (where B denotes the batch size, C the number of channels, H the height, and W the width) in a given layer of a neural net, the normalization methods differ in the dimensions over which they compute the mean and variance, see Figure 2. In general, activation normalization methods compute the mean μ and standard deviation (std) σ of the features in their own manner, normalize the features with these statistics, and optionally apply an affine transformation with parameters β (new mean) and γ (new std). This can be written as

$$X'_{b,c,h,w} = \gamma\left(\frac{X_{b,c,h,w} - \mu}{\sigma}\right) + \beta. \tag{1}$$

Batch Normalization (BN) [29] computes μ and σ across the B, H, and W dimensions. BN increases the robustness of the network with respect to high learning rates and weight initializations [5], which in turn drastically improves the convergence rate. Synchronized Batch Normalization treats features of mini-batches across multiple GPUs like a single mini-batch. Instance Normalization (IN) [68] treats each instance in a mini-batch independently and computes the statistics across only spatial dimensions (H and W). IN aims to make a small change in the stylization architecture results in a significant qualitative improvement in the generated images. Layer Normalization (LN) normalizes all features of an instance within a layer jointly, i.e., calculating the statistics over the C, H, and W dimensions. LN is beneficial in natural language processing applications [41, 70]. Notably, none of the aforementioned methods normalize the information at different spatial position independently. This limitation gives rise to our proposed Positional Normalization.

Batch Normalization introduces two learned parameters β and γ to allow the model to adjust the mean and std of the post-normalized features. Specifically, $\beta, \gamma \in \mathbb{R}^C$ are channel-wise parameters. Condi-

tional instance normalization (CIN) [16] keeps a set parameter of pairs $\{(\beta_i, \gamma_i)|i \in \{1, \ldots, N\}\}$ which enables the model to have N different behaviors conditioned on a style class label i. Adaptive instance normalization (AdaIN) [27] generalizes this to an infinite number of styles by using the μ and σ of IN borrowed from another image as the β and γ. Dynamic Layer Normalization (DLN) [36] relies on a neural network to generate the β and γ. Later works [28, 34] refine AdaIN and generate the β and γ of AdaIN dynamically using a dedicated neural network. Conditional batch normalization (CBN) [11] follows a similar spirit and uses a neural network that takes text as input to predict the residual of β and γ, which is shown to be beneficial to visual question answering models.

Notably, all aforementioned methods generate β and γ as vectors, shared across spatial positions. In contrast, Spatially Adaptive Denormalization (SPADE) [53], an extension of Synchronized Batch Normalization with dynamically predicted weights, generates the spatially dependent $\beta, \gamma \in \mathbb{R}^{B \times C \times H \times W}$ using a two-layer ConvNet with raw images as inputs.

Finally, we introduce shortcut connections to transfer the first and second moment from early to later layers. Similar skip connections (with add, concat operations) have been introduced in ResNets [21] and DenseNets [26] and earlier works [4, 24, 35, 56, 64], and are highly effective at improving network optimization and convergence properties [44].

3 Positional Normalization and Moment Shortcut

Prior work has shown that feature normalization has a strong beneficial effect on the convergence behavior of neural networks [5]. Although we agree with these findings, in this paper we claim that removing the first and second order information at multiple stages throughout the network may also deprive the deep net of potentially useful information — particularly in the context of generative models, where a plausible image needs to be generated.

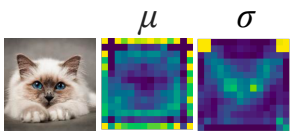

Figure 3: PONO statistics of DenseBlock-3 of a pretrained DenseNet-161.

PONO. Our normalization scheme, which we refer to as *Positional Normalization (PONO)*, differs from prior work in that we normalize exclusively over the channels at any given fixed pixel location (see Figure 2). Consequently, the extracted statistics are position dependent and reveal structural information at this particular layer of the deep net. The mean μ can be considered itself an "image", where the intensity of pixel i, j represents the average activation at this particular image location in this layer. The standard deviation σ is the natural second order extension. Formally, PONO computes

$$\mu_{b,h,w} = \frac{1}{C}\sum_{c=1}^{C} X_{b,c,h,w}, \quad \sigma_{b,h,w} = \sqrt{\frac{1}{C}\sum_{c=1}^{C}\left(X_{b,c,h,w} - \mu_{b,h,w}\right)^2 + \epsilon}, \tag{2}$$

where ϵ is a small stability constant (*e.g.*, $\epsilon = 10^{-5}$) to avoid divisions by zero and imaginary values due to numerical inaccuracies.

Properties. As PONO computes the normalization statistics at all spatial positions independently from each other (unlike BN, LN, CN, and GN) it is translation, scaling, and rotation invariant. Further, it is complementary to existing normalization methods and, as such, can be readily applied in combination with e.g. BN.

Visualization. As the extracted mean and standard deviations are themselves images, we can visualize them to obtain information about the extract features at the various layers of a convolutional network. Such visualizations can be revealing and could potentially be used to debug or improve network architectures. Figure 1 shows heat-maps of the μ and σ captured by PONO at several layers (Conv1_2, Conv2_2, Conv3_4, and Conv4_4) of VGG-19 [62]. The figure reveals that the features in lower layers capture the silhouette of a cat while higher layers locate the position of noses, eyes, and the end points of ears —- suggesting that later layers may focus on higher level concepts corresponding to essential facial features (eyes, nose, mouth), whereas earlier layers predominantly extract generic low level features like edges. We also observe a similar phenomenon from the features of ResNets [21] and DenseNets [26] (see Figure 3 and Appendix). The resulting images are reminiscent of related statistics captured in texture synthesis [15, 17–19, 22, 52, 72]. We observe that

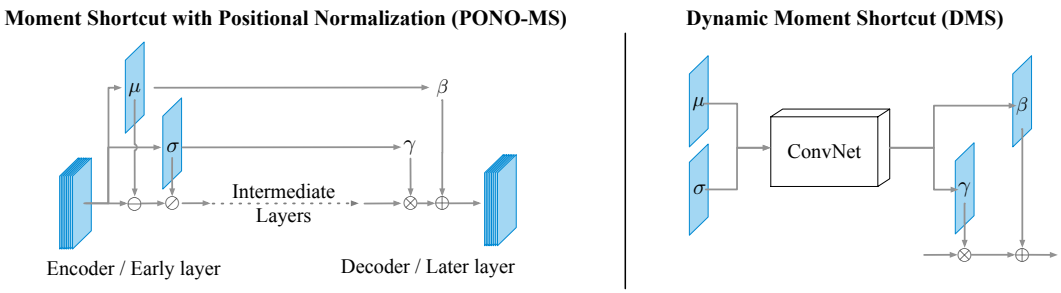

Figure 4: Left: PONO-MS directly uses the extracted mean and standard deviation as β and γ. Right: Optionally, one may use a (shallow) ConvNet to predict β and γ dynamically based on μ and σ.

unlike VGG and ResNet, DenseNet exhibits strange behavior on corners and boundaries which may degrade performance when fine-tuned on tasks requiring spatial information such as object detection or segmentation. This suggests that the padding and downsampling procedure of DenseNet should be revisited and may lead to improvements if fixed, see Figure 3. The visualizations of the PONO statistics support our hypothesis that the mean μ and the standard deviation σ may indeed capture structural information of the image and extracted features, similar to the way statistics computed by IN have the tendency to capture aspects of the style of the input image [27, 68]. This extraction of valuable information motivates the Moment Shortcut described in the subsequent section.

3.1 Moment Shortcut

In generative models, a deep net is trained to generate an output image from some inputs (images). Typically, generative models follow an encoder-decoder architecture, where the encoder digests an image into a condensed form and the decoder recovers a plausible image with some desired properties. For example, Huang et al. [27] try to transfer the style from an image A to an image B, Zhu et al. [79] "translate" an image from an input distribution (e.g., images of zebras) to an output distribution (e.g., images of horses), Choi et al. [9] use a shared encoder-decoder with a classification loss in the encoded latent space to enable translation across multiple distributions, and [28, 40] combine the structural information of an image with the attributes from another image to generate a fused output.

U-Nets [57] famously achieve strong results and compelling optimization properties in generative models through the introduction of skip connections from the encoder to the decoder. PONO gives rise to an interesting variant of such skip connections. Instead of connecting all channels, we only "fast-forward" the positional moment information μ and σ extracted from earlier layers. We refer to this approach as Moment Shortcut (MS).

Autoencoders. Figure 4 (left) illustrates the use of MS in the context of an autoencoder. Here, we extract the first two moments of the activations (μ, σ) in an encoder layer, and send them to a corresponding decoder layer. Importantly, the mean is *added* in the encoder, and the std is *multiplied*, similar to (β, γ) in the standard BN layer. To be specific, $\mathrm{MS}(\mathbf{x}) = \gamma F(\mathbf{x}) + \beta$, where F is modeled by the intermediate layers, and the β and γ are the μ and σ extracted from the input $\mathbf{x}$. MS biases the decoder explicitly so that the activations in the decoder layers give rise to similar statistics than corresponding layers in the encoder. As MS shortcut connections can be used with and without normalization, we refer to the combination of PONO with MS as **PONO-MS** throughout.

Provided PONO does capture essential structural signatures from the input images, we can use the extracted moments to transfer this information from a source to a target image. This opens an opportunity to go beyond autoencoders and use PONO-MS in image-to-image translation settings, for example in the context of CycleGAN [79] and Pix2Pix [30]. Here, we transfer the structure (through μ and σ) of one image from the encoder to the decoder of another image.

Dynamic Moment Shortcut. Inspired by Dynamic Layer Normalization and similar works [7, 28, 34, 36, 53], we propose a natural extension called Dynamic Moment Shortcut (DMS): instead of re-injecting μ and σ as is, we use a convolutional neural network that takes μ and σ as inputs to generate the β and γ for MS. This network can either generate one-channel outputs $\beta, \gamma \in \mathcal{R}^{B\times 1\times H\times W}$ or multi-channel outputs $\beta, \gamma \in \mathcal{R}^{B\times C\times H\times W}$ (like [53]). The right part of Figure 4 illustrates DMS

with one-channel output. DMS is particularly helpful when the task involves shape deformation or distortion. We refer to this approach as **PONO-DMS** in the following sections. In our experiments, we explore using a ConvNet with either one or two layers.

4 Experiments and Analysis

We conduct our experiments on unpaired and paired image translation tasks using CycleGAN [79] and Pix2pix [30] as baselines, respectively. Our code is available at https://github.com/Boyiliee/PONO.

4.1 Experimental Setup

We follow the same setup as CycleGAN [79] and Pix2pix [30] using their official code base.[2] We use four datasets: 1) **Maps** (Maps $\leftrightarrow$ aerial photograph) including 1096 training images scraped from Google Maps and 1098 images in each domain for testing. 2) **Horse** $\leftrightarrow$ **Zebra** including 1067 horse images and 1334 zebra images downloaded from ImageNet [12] using keywords wild horse and zebra, and 120 horse images and 140 zebra images for testing. 3) **Cityscapes** (Semantic labels $\leftrightarrow$ photos) [10] including 2975 images from the Cityscapes training set for training and 500 images in each domain for testing. 4) **Day** $\leftrightarrow$ **Night** including 17,823 natural scene images from Transient Attributes dataset [38] for training, and 2,287 images for testing. The first, third, and fourth are paired image datasets; the second is an unpaired image dataset. We use the first and second for CycleGAN, and all the paired-image datasets for Pix2pix.

Evaluation metrics. We use two evaluation metrics, as follows. (1) Fréchet Inception Distance [23] between the output images and all test images in the target domain. FID uses an Inception [66] model pretrained on ImageNet [12] to extract image features. Based on the means and covariance matrices of the two sets of extracted features, FID is able to estimate how different two distributions are. (2) Average Learned Perceptual Image Patch Similarity distance [78] of all output and target image pairs. LPIPS is based on pretrained AlexNet [37] features[3], which has been shown [78] to be highly correlated to human judgment.

Baselines. We include four baseline approaches: (1) CycleGAN or Pix2pix baselines; (2) these baselines with SPADE [53], which passes the input image through a 2-layer ConvNet and generates the β and γ for BN in the decoder. (3) the baseline with *additive skip connections* where encoder activations are added to decoder activations; (4) the baseline with *concatenated skip connections*, where encoder activations are concatenated to decoder activations as additional channels (similar to U-Nets [57]). For all models, we follow the same setup as CycleGAN [79] and Pix2pix [30] using their implementations. Throughout we use the hyper-parameters suggested by the original authors.

4.2 Comparison against Baselines

We add PONO-MS and PONO-DMS to the CycleGAN generator; see the Appendix for the model architecture. Table 1 shows that both cases outperform all baselines at transforming maps into photos, with the only exception of SPADE (which however performs worse in the other direction).

Although skip connections could help make up for the lost information, we postulate that directly adding the intermediate features back may introduce too much unnecessary information and might distract the model. Unlike the skip connections, SPADE uses the input to predict the parameters for normalization. However, on Photo $\rightarrow$ Map, the model has to learn to compress the input photos and extract structural information from it. A re-introduction of the original raw input may disturb this process and explain the worse performance. In contrast, PONO-MS normalizes exclusively across channels which allows us to capture structural information of a particular input image and re-inject/transfer it to later layers.

The Pix2pix model [30] is a conditional adversarial network introduced as a general-purpose solution for image-to-image translation problems. Here we conduct experiments on whether PONO-MS helps Pix2pix [30] with Maps [79], Cityscapes [10] and Day $\leftrightarrow$ Night [38]. We train for 200 epochs and

[2] https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix

[3] https://github.com/richzhang/PerceptualSimilarity, version 0.1.

Method	# of param.	**Map → Photo** FID	**Photo → Map** FID	**Horse → Zebra** FID	**Zebra → Horse** FID
CycleGAN (Baseline)	2×11.378M	57.9	58.3	86.3	155.9
+Skip Connections	**+0M**	83.7	56.0	75.9	145.5
+Concatenation	+0.74M	58.9	61.2	85.0	145.9
+SPADE	+0.456M	**48.2**	59.8	71.2	159.9
+PONO-MS	**+0M**	52.8	**53.2**	71.2	142.2
+PONO-DMS	+0.018M	53.7	54.1	**65.7**	**140.6**

Table 1: FID of CycleGAN and its variants on Map ↔ Photo and Zebra ↔ Horse datasets. CycleGAN is trained with two directions together, it is essential to have good performance in both directions.

	Maps [79]		**Cityscapes** [10]		**Day ↔ Night** [38]	
	Map → Photo	**Photo → Map**	**SL → Photo**	**Photo → SL**	**Day → Night**	**Night →Day**
Pix2pix (Baseline)	60.07 / **0.333**	68.73 / 0.169	71.24 / 0.422	102.38 / **0.223**	196.58 / 0.608	131.94 / **0.531**
+PONO-MS	**56.88 / 0.333**	**68.57 / 0.166**	**60.40 / 0.331**	**97.78** / 0.224	**191.10 / 0.588**	**131.83** / 0.534

Table 2: Comparison based on Pix2pix by FID / LPIPS on Maps [79], Cityscapes [10], and Day2Night. Note: for all scores, the lower the better (SL is short for *Semantic labels*).

compare the results with/without PONO-MS, under similar conditions with matching number of parameters. Results are summarized in Table 2.

4.3 Ablation Study

Table 3 contains the results of several experiments to evaluate the sensitivities and design choices of PONO-MS and PONO-DMS. Further, we evaluate *Moment Shortcut (MS)* without PONO, where we bypass both statistics, μ and σ, without normalizing the features. The results indicate that PONO-MS outperforms MS alone, which suggests that normalizing activations with PONO is beneficial. PONO-DMS can lead to further improvements, and some settings (e.g. *1 conv 3 × 3, multi-channel*) consistently outperform PONO-MS. Here, multi-channel predictions are clearly superior over single-channel predictions but we do not observe consistent improvements from a 5×5 rather than a 3×3 kernel size.

Normalizations. Unlike previous normalization methods such as BN and GN that emphasize on accelerating and stabilizing the training of networks, PONO is used to split off part of the spatial information and re-inject it later. Therefore, PONO-MS can be applied jointly with other normalization methods. In Table 4 we evaluate four normalization approaches (BN, IN, LN, GN) with and without PONO-MS, and PONO-MS without any additional normalization (bottom row). In detail, *BN + PONO-MS* is simply applying *PONO-MS* to the baseline model and keep the original BN modules which have a different purpose: to stabilize and speed up the training. We also show the models where BN is replaced by LN/IN/GN as well as these models with PONO-MS. The last row shows PONO-MS can work independently when we remove the original BN in the model. Each table entry displays the FID score without and with PONO-MS (the lower score is in **bold**). The final column (very right) contains the average improvement across all four tasks, relative to the default architecture, BN without PONO-MS. Two clear trends emerge: 1. All four normalization methods

Method	**Map → Photo**	**Photo → Map**	**Horse → Zebra**	**Zebra → Horse**
CycleGAN (Baseline)	57.9	58.3	86.3	155.9
+Moment Shortcut (MS)	54.5	56.6	79.8	146.1
+PONO-MS	52.8	53.2	71.2	142.2
+PONO-DMS (1 conv 3×3, one-channel)	55.1	53.8	74.1	147.2
+PONO-DMS (2 conv 3×3, one-channel)	56.0	53.3	81.6	144.8
+PONO-DMS (1 conv 3×3, multi-channel)	53.7	54.1	65.7	**140.6**
+PONO-DMS (2 conv 5×5, multi-channel)	52.7	54.7	**64.9**	155.2
+PONO-DMS (2 conv $3 \times 3, 5 \times 5$, multi-channel)	**48.9**	57.3	74.3	148.4
+PONO-DMS (2 conv 3×3, multi-channel)	50.3	**51.4**	72.2	146.1

Table 3: Comparisons of ablation study on FID (lower is better). PONO-MS outperforms MS alone. PONO-DMS can help obtain better performance than PONO-MS.

Method	**Map → Photo**	**Photo → Map**	**Horse → Zebra**	**Zebra → Horse**	Avg. Improvement
BN (Default) / BN + PONO-MS	57.92 / **52.81**	58.32 / **53.23**	86.28 / **71.18**	155.91 / **142.21**	1 / **0.890**
IN / IN + PONO-MS	67.87 / **47.14**	57.93 / **54.18**	**67.85** / 69.21	154.15 / **153.61**	0.985 / **0.883**
LN / LN + PONO-MS	54.84 / **49.81**	53.00 / **50.08**	87.26 / **67.63**	154.49 / **142.05**	0.964 / **0.853**
GN / GN + PONO-MS	51.31 / **50.12**	50.62 / **50.50**	93.58 / **63.53**	**143.56** / 144.99	0.940 / **0.849**
PONO-MS	**49.59**	**52.21**	**84.68**	**143.47**	**0.913**

Table 4: FID scores (lower is better) of CycleGAN with different normalization methods.

improve with PONO-MS on average and on almost all individual tasks; 2. additional normalization is clearly beneficial over pure PONO-MS (bottom row).

5 Further Analysis and Explorations

In this section, we apply PONO-MS to two state-of-the-art unsupervised image-to-image translation models: MUNIT [28] and DRIT [40]. Both approaches may arguably be considered concurrent works and share a similar design philosophy. Both aim to translate an image from a source to a target domain, while imposing the attributes (or the style) of another target domain image.

As task, we are provided with an image $\mathbf{x}_A$ in source domain A and an image $\mathbf{x}_B$ in target domain B. DRIT uses two encoders, one to extract content features $\mathbf{c}_A$ from $\mathbf{x}_A$, and the other to extract attribute features $\mathbf{a}_B$ from $\mathbf{x}_B$. A decoder then takes $\mathbf{c}_A$ and $\mathbf{a}_B$ as inputs to generate the output image $\mathbf{x}_{A\to B}$. MUNIT follows a similar pipeline. Both approaches are trained on the two directions, $A \to B$ and $B \to A$, simultaneously. We apply PONO to DRIT or MUNIT immediately after the first three convolution layers (convolution layers before the residual blocks) of the content encoders. We then use MS before the last three transposed convolution layers with matching decoder sizes. We follow the DRIT and MUNIT frameworks and consider the extracted statistics (μ's and σ's) as part of the content tensors.

5.1 Experimental Setup

We consider two datasets provided by the authors of DRIT: 1) **Portrait ↔ Photo** [40, 45] with 1714 painting images and 6352 human photos for training, and 100 images in each domain for testing and 2) **Cat ↔ Dog** [40] containing 771 cat images and 1264 dog images for training, and 100 images in each domain for testing.

In the following experiments, we use the official codebases[4], closely follow their proposed hyper-parameters and train all models for 200K iterations. We use the holdout test images as the inputs for evaluation. For each image in the source domain, we randomly sample 20 images in the target domain to extract the attributes and generate 20 output images. We consider four evaluation metrics: 1) FID [23]: Fréchet Inception Distance between the output images and all test images in the target domain, 2) $\text{LPIPS}_{\text{attr}}$ [78]: average LPIPS distance between each output image and its corresponding input image in the target domain, 3) $\text{LPIPS}_{\text{cont}}$: average LPIPS distance between each output image and its input in the source domain, and 4) perceptual loss (VGG) [32, 62]: L1 distance between the VGG-19 Conv4_4 features [8] of each output image and its corresponding input in the source domain. The FID and $\text{LPIPS}_{\text{attr}}$ are used to estimate how likely the outputs are to belong to the target domain, while $\text{LPIPS}_{\text{cont}}$ and VGG loss are adopted to estimate how much the outputs preserve the structural information in the inputs. All of them are distance metrics where lower is better. The original implementations of DRIT and MUNIT assume differently sized input images (216x216 and 256x256, respectively), which precludes a direct comparison across approaches.

5.2 Results of Attribute Controlled Image Translation

Figure 5 shows the qualitative results on the Cat ↔ Dog dataset. (Here we show the results of MUNIT' + PONO-MS which will be explained later.) We observe a clear trend that PONO-MS helps these two models obtain more plausible results. We observe the models with PONO-MS is able to capture the content features and attributes distributions, which motivates baseline models to digest different information from both domains. For example, in the first row, when translating cat to dog,

[4]https://github.com/NVlabs/MUNIT/ and https://github.com/HsinYingLee/DRIT

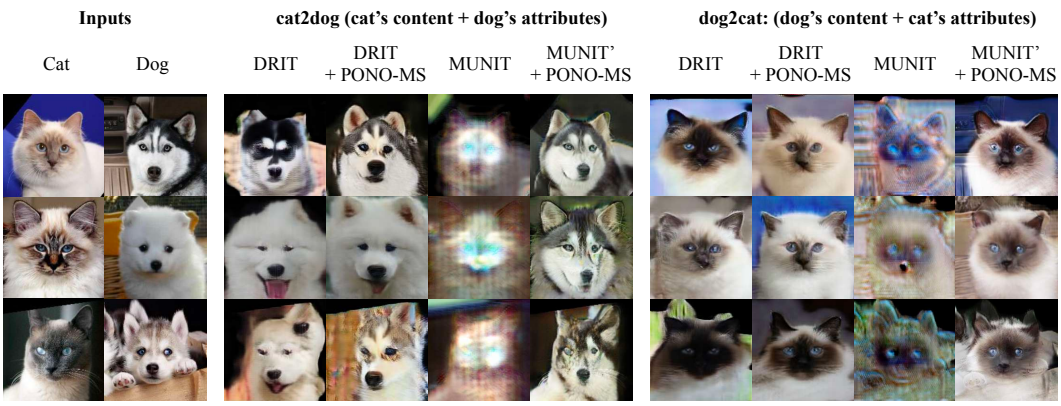

Figure 5: PONO-MS improves the quality of both DRIT [40] and MUNIT [28] on Cat ↔ Dog.

	Portrait → Photo				**Portrait ← Photo**			
	FID	$\text{LPIPS}_{\text{attr}}$	$\text{LPIPS}_{\text{cont}}$	VGG	FID	$\text{LPIPS}_{\text{attr}}$	$\text{LPIPS}_{\text{cont}}$	VGG
DRIT	131.2	0.545	0.470	1.796	104.5	0.585	0.476	2.033
DRIT + PONO-MS	**127.9**	**0.534**	**0.457**	**1.744**	**99.5**	**0.575**	**0.463**	**2.022**
MUNIT	220.1	0.605	0.578	1.888	149.6	0.619	0.670	2.599
MUNIT + PONO-MS	270.5	0.541	0.423	1.559	127.5	0.586	0.477	2.202
MUNIT'	245.0	0.538	0.455	1.662	158.1	0.601	0.620	2.434
MUNIT' + PONO-MS	**159.4**	**0.424**	**0.319**	**1.324**	**125.1**	**0.566**	**0.312**	**1.824**
	Cat → Dog				**Cat ← Dog**			
	FID	$\text{LPIPS}_{\text{attr}}$	$\text{LPIPS}_{\text{cont}}$	VGG	FID	$\text{LPIPS}_{\text{attr}}$	$\text{LPIPS}_{\text{cont}}$	VGG
DRIT	**45.8**	0.542	0.581	**2.147**	42.0	0.524	**0.576**	2.026
DRIT + PONO-MS	47.5	**0.524**	**0.576**	**2.147**	**41.0**	**0.514**	0.604	**2.003**
MUNIT	315.6	0.686	0.674	1.952	290.3	0.629	0.591	2.110
MUNIT + PONO-MS	254.8	0.632	0.501	1.614	276.2	0.624	0.585	2.119
MUNIT'	361.5	0.699	0.607	1.867	289.0	0.767	0.789	2.228
MUNIT' + PONO-MS	**80.4**	**0.615**	**0.406**	**1.610**	**90.8**	**0.477**	**0.428**	**1.689**

Table 5: PONO-MS can improve the performance of MUNIT [28], while for DRIT [40] the improvement is marginal. MUNIT' is MUNIT with one more Conv3x3-LN-ReLU layer before the output layer in the decoder, which introduces 0.2% parameters into the generator. Note: for all scores, the lower the better.

DRIT with PONO-MS is able to capture the cat's facial expression, and MUNIT with PONO-MS could successfully generate dog images with plausible content, which largely boosts the performance of the baseline models. More qualitative results of randomly selected inputs are provided in the Appendix.

Table 5 show the quantitative results on both Cat ↔ Dog and Portrait ↔ Photo datasets. PONO-MS improves the performance of both models on all instance-level metrics ($\text{LPIPS}_{\text{attr}}$, $\text{LPIPS}_{\text{cont}}$, and VGG loss). However, the dataset-level metric, FID, doesn't improve too much. We believe the reason is that FID is calculated based on the first two order statistic of Inception features and may discard some subtle differences between each output pair.

Interestingly MUNIT, while being larger than DRIT (30M parameters vs. 10M parameters), doesn't perform better on these two datasets. One reason for its relatively poor performance could be that the model was not designed for these datasets (MUNIT uses a much larger unpublished *dogs to big cats* dataset), the dataset are very small, and the default image resolution is slightly different. To further improve MUNIT + PONO-MS, we add one more Conv3x3-LN-ReLU layer before the output layer. Without this, there is only one layer between the outputs and the last re-introduced μ and σ. Therefore, adding one additional layer allows the model to learn a nonlinear function of these μ and σ. We call this model MUNIT' + PONO-MS. Adding this additional layer significantly enhances the performance of MUNIT while introducing only 75K parameters (about 0.2%). We also provide the numbers of MUNIT' (MUNIT with one additional layer) as a baseline for a fair comparison.

Admittedly, the state-of-the-art generative models employ complex architecture and a variety of loss functions; therefore, unveiling the full potential of PONO-MS on these models can be nontrivial and required further explorations. It is fair to admit that the results of all model variations are still largely unsatisfactory and the image translation task remains an open research problem.

However, we hope that our experiments on DRIT and MUNIT may shed some light on the potential value of PONO-MS, which could open new interesting directions of research for neural architecture design.

6 Conclusion and Future Work

In this paper, we propose a novel normalization technique, Positional Normalization (PONO), in combination with a purposely limited variant of shortcut connections, Moment Shortcut (MS). When applied to various generative models, we observe that the resulting model is able to preserve structural aspects of the input, improving the plausibility performance according to established metrics. PONO and MS can be implemented in a few lines of code (see Appendix). Similar to Instance Normalization, which has been observed to capture the style of image [27, 34, 68], Positional Normalization captures structural information. As future work we plan to further explore such disentangling of structural and style information in the design of modern neural architectures.

It is possible that PONO and MS can be applied to a variety of tasks such as image segmentation [46, 57], denoising [42, 75], inpainting [76], super-resolution [14], and structured output prediction [63]. Further, beyond single image data, PONO and MS may also be applied to video data [43, 71], 3D voxel grids [6, 67], or tasks in natural language processing [13].

Acknowledgments

This research is supported in part by the grants from Facebook, the National Science Foundation (III-1618134, III-1526012, IIS1149882, IIS-1724282, and TRIPODS-1740822), the Office of Naval Research DOD (N00014-17-1-2175), Bill and Melinda Gates Foundation. We are thankful for generous support by Zillow and SAP America Inc.

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] Sanjeev Arora, Zhiyuan Li, and Kaifeng Lyu. Theoretical analysis of auto rate-tuning by batch normalization. In *International Conference on Learning Representations*, 2019.

[3] David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 342–350. JMLR. org, 2017.

[4] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[5] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. In *Advances in Neural Information Processing Systems*, pages 7694–7705, 2018.

[6] João Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 4724–4733. IEEE Computer Society, 2017.

[7] Ting Chen, Mario Lucic, Neil Houlsby, and Sylvain Gelly. On self modulation for generative adversarial networks. *arXiv preprint arXiv:1810.01365*, 2018.

[8] Yang Chen, Yu-Kun Lai, and Yong-Jin Liu. Cartoongan: Generative adversarial networks for photo cartoonization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9465–9474, 2018.

[9] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8789–8797, 2018.

[10] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems*, pages 6594–6604, 2017.

[12] 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.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[14] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Learning a deep convolutional network for image super-resolution. In *European conference on computer vision*, pages 184–199. Springer, 2014.

[15] Ian L Dryden. Shape analysis. *Wiley StatsRef: Statistics Reference Online*, 2014.

[16] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *Proc. of ICLR*, 2, 2017.

[17] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346. ACM, 2001.

[18] Alexei A Efros and Thomas K Leung. Texture synthesis by non-parametric sampling. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1033–1038. IEEE, 1999.

[19] William T. Freeman and Edward H Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, pages 891–906, 1991.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[21] 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.

[22] David J Heeger and James R Bergen. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 229–238. Citeseer, 1995.

[23] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[24] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[25] Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. In *Advances in Neural Information Processing Systems*, pages 2160–2170, 2018.

[26] 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.

[27] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1501–1510, 2017.

[28] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 172–189, 2018.

[29] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[30] 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.

[31] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th international conference on computer vision*, pages 2146–2153. IEEE, 2009.

[32] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[33] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[34] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *arXiv preprint arXiv:1812.04948*, 2018.

[35] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1646–1654, 2016.

[36] Taesup Kim, Inchul Song, and Yoshua Bengio. Dynamic layer normalization for adaptive neural acoustic modeling in speech recognition. *arXiv preprint arXiv:1707.06065*, 2017.

[37] 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.

[38] Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, Chao Qian, and James Hays. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM Transactions on Graphics (TOG)*, 33(4):149, 2014.

[39] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[40] Hsin-Ying Lee, , Hung-Yu Tseng, Jia-Bin Huang, Maneesh Kumar Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *European Conference on Computer Vision*, 2018.

[41] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[42] Boyi Li, Xiulian Peng, Zhangyang Wang, Jizheng Xu, and Dan Feng. Aod-net: All-in-one dehazing network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4770–4778, 2017.

[43] Boyi Li, Xiulian Peng, Zhangyang Wang, Jizheng Xu, and Dan Feng. End-to-end united video dehazing and detection. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[44] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6389–6399. Curran Associates, Inc., 2018.

[45] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[46] 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.

[47] Ping Luo, Jiamin Ren, and Zhanglin Peng. Differentiable learning-to-normalize via switchable normalization. *arXiv preprint arXiv:1806.10779*, 2018.

[48] Ping Luo, Xinjiang Wang, Wenqi Shao, and Zhanglin Peng. Towards understanding regularization in batch normalization. In *International Conference on Learning Representations*, 2019.

[49] Siwei Lyu and Eero P Simoncelli. Nonlinear image representation using divisive normalization. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[50] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *Proc. of ICLR*, 2018.

[51] Genevieve B Orr and Klaus-Robert Müller. *Neural networks: tricks of the trade*. Springer, 2003.

[52] Robert Osada, Thomas Funkhouser, Bernard Chazelle, and David Dobkin. Shape distributions. *ACM Transactions on Graphics (TOG)*, 21(4):807–832, 2002.

[53] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[54] 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.

[55] Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Weight standardization. *arXiv preprint arXiv:1903.10520*, 2019.

[56] Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007.

[57] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[58] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–, October 1986.

[59] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 901–909. Curran Associates, Inc., 2016.

[60] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pages 2483–2493, 2018.

[61] Wenqi Shao, Tianjian Meng, Jingyu Li, Ruimao Zhang, Yudian Li, Xiaogang Wang, and Ping Luo. Ssn: Learning sparse switchable normalization via sparsestmax. *arXiv preprint arXiv:1903.03793*, 2019.

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

[63] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015.

[64] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[65] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[66] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[67] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 4489–4497. IEEE Computer Society, 2015.

[68] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[69] Twan van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017.

[70] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[71] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *CVPR*, 2018.

[72] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488. ACM Press/Addison-Wesley Publishing Co., 2000.

[73] Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. In *International Conference on Learning Representations*, 2019.

[74] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

[75] Junyuan Xie, Linli Xu, and Enhong Chen. Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems*, pages 341–349, 2012.

[76] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5505–5514, 2018.

[77] Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Residual learning without normalization via better initialization. In *International Conference on Learning Representations*, 2019.

[78] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[79] 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.

Appendices

A Algorithm of PONO-MS

The implementation of PONO-MS in TensorFlow [1] an PyTorch[54] 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

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.

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

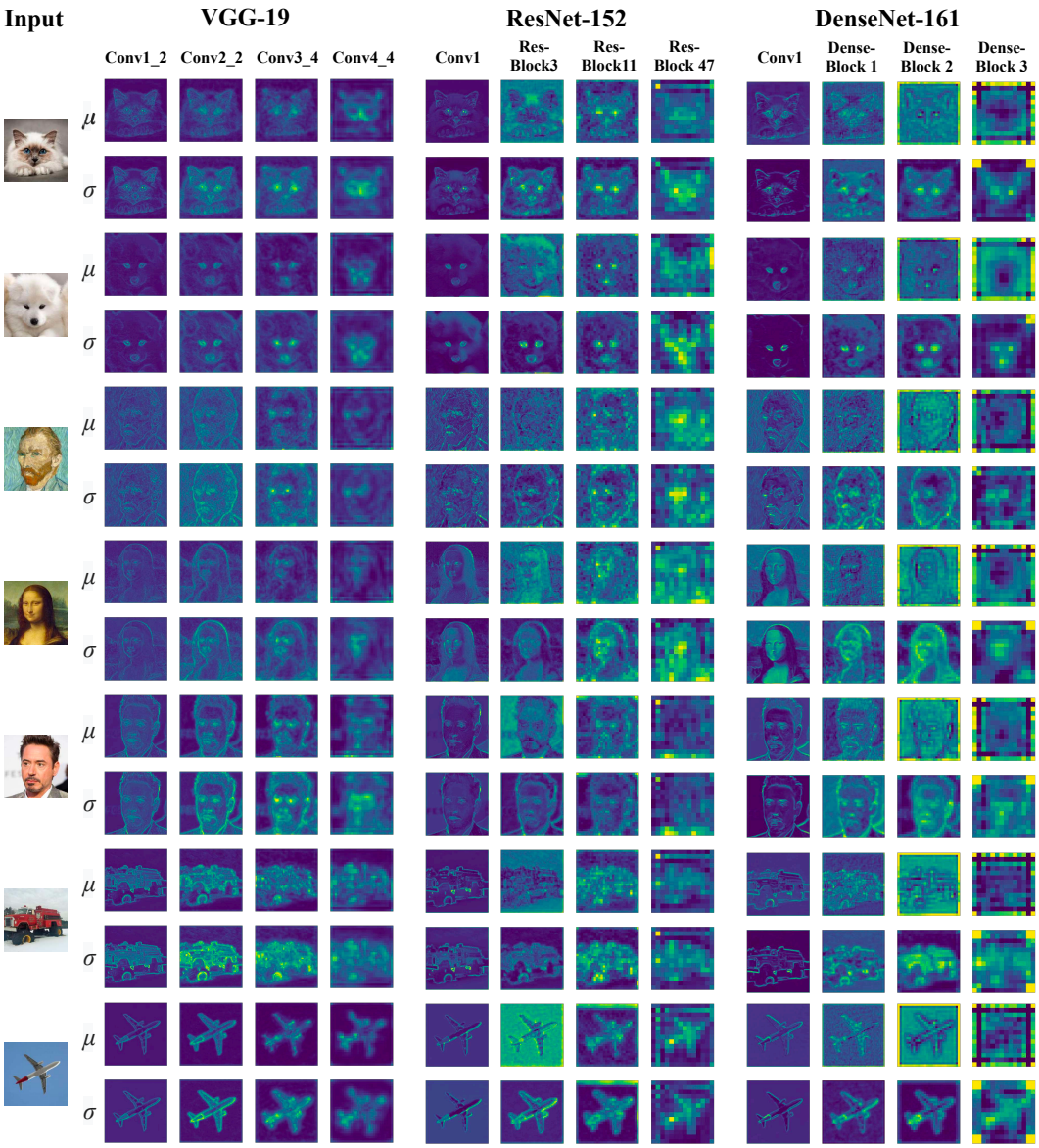

Figure 6: We extract the PONO statistics from VGG-19, ResNet-152, and Dense-161 at layers right before downsampling (max-pooling or strided convolution).

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 6 shows the means and the standard deviations extracted by PONO based on the features generated by VGG-19 [62], ResNet-152 [21], and DenseNet-161 [26] pretrained on ImageNet [12].

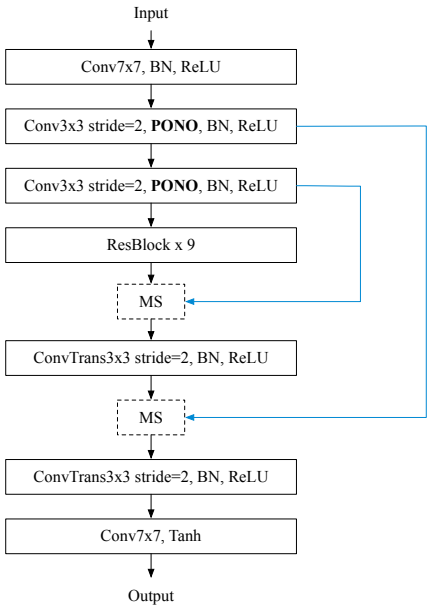

Figure 7: The generator of CycleGAN + PONO-MS. Pix2pix uses the same architecture. The operations in a block is applied from left to right sequentially. The **blue** lines show how the first two moments are passed. ConvTrans stands for transposed convolution. Each ResBlock has Conv3x3, BN, ReLU, Conv3x3, and BN.

D Implementation details

We add PONO to the encoder right after a convolution operation and before other normalization or nonlinear activation function. Figure 7 shows the model architecture of CycleGAN [79] with Positional Normalization. Pix2pix [30] uses the same architecture.

E Qualitative Results Based on CycleGAN and Pix2pix

We show some outputs of CycleGAN in Figure 8. The Pix2pix outputs are shown in Figure 9.

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 10.

G PONO in Image Classification

To evaluate PONO on image classification task, we add PONO to the begining of each ResBlock of ResNet-18 [21] (also affects the shortcut). We followed the common training procedure base on Wei Yang's open sourced code [5] on ImageNet [37]. Figure 11 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.

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

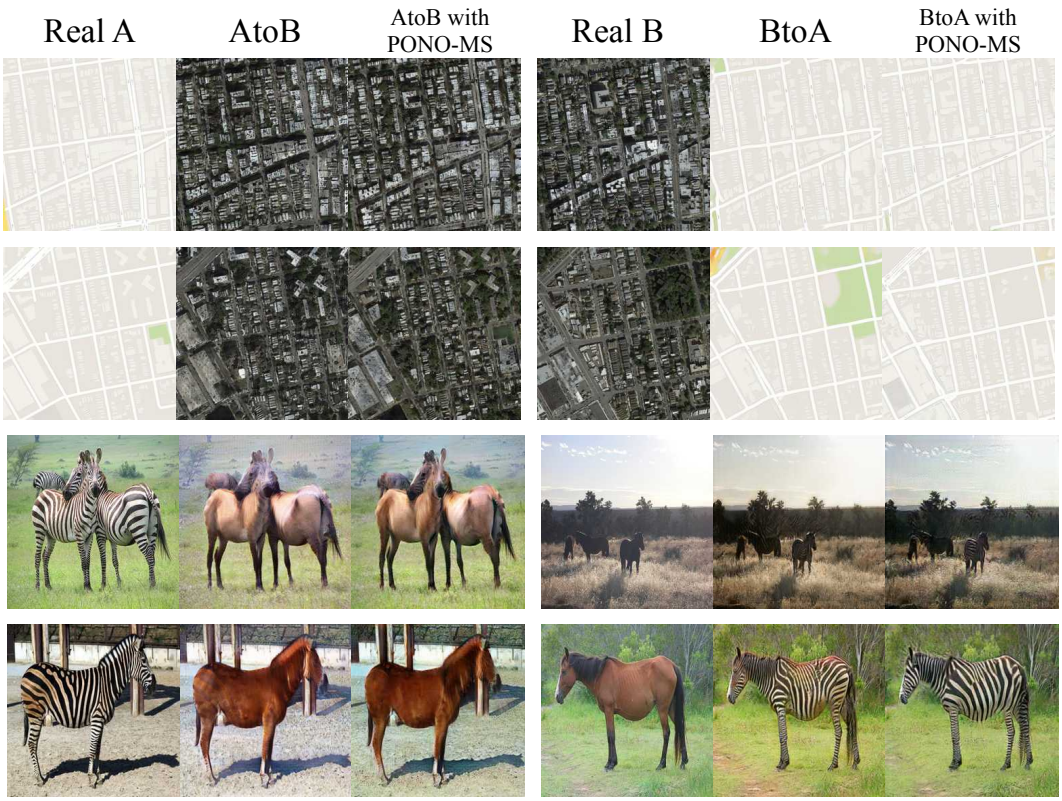

Figure 8: Qualitative results of CycleGAN (with/without PONO-MS) with randomly sampled inputs.

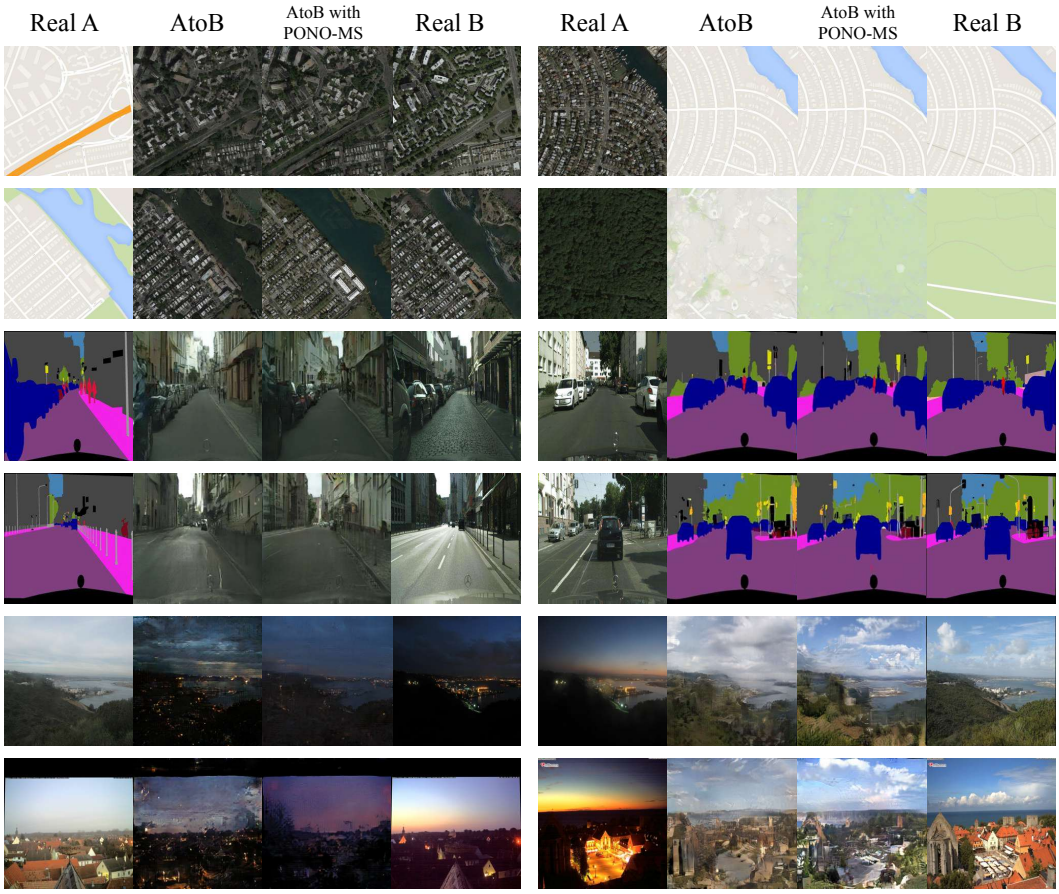

Figure 9: Qualitative results of Pix2pix (with/without PONO-MS) with randomly sampled inputs.

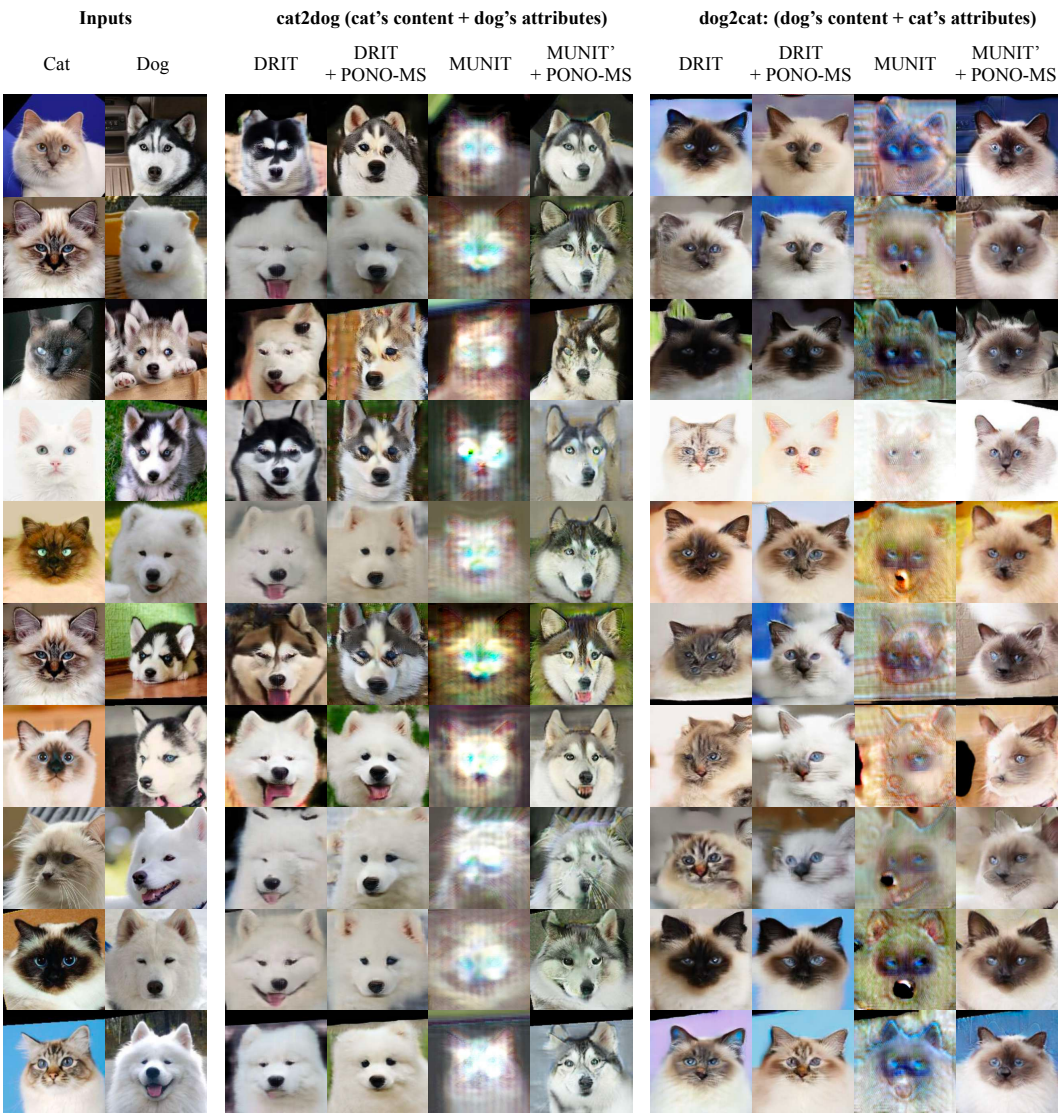

Figure 10: Qualitative results of DRIT and MUNIT (with/without PONO-MS) with randomly sampled inputs.

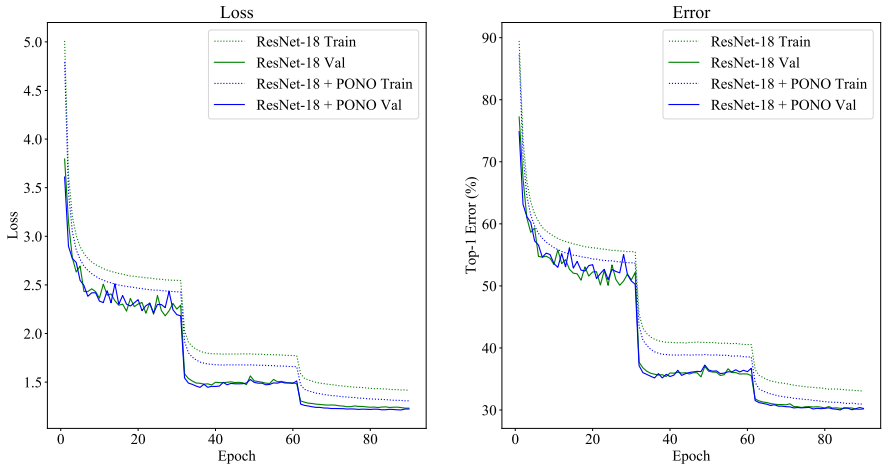

Figure 11: Training and validation curves of ResNet-18 and ResNet-18 + PONO on ImageNet.