Sym-Parameterized Dynamic Inference for Mixed-Domain Image Translation

Simyung Chang[1,2], SeongUk Park[1], John Yang[1], Nojun Kwak[1]
[1]Seoul National University, Seoul, Korea
[2]Samsung Electronics, Suwon, Korea
{timelighter, swpark0703, yjohn, nojunk}@snu.ac.kr

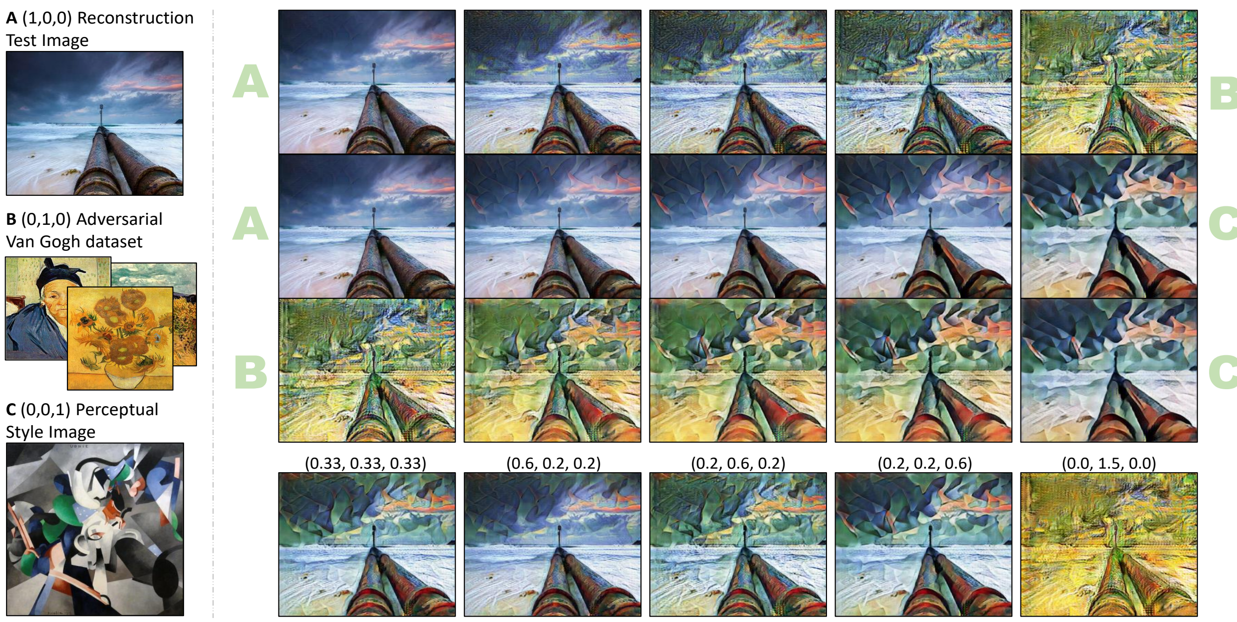

Figure 1. Results of image translation to mixed-domains. These images are obtained by learning the SGN for three losses (reconstruction, adversarial, perceptual) and then by inferring the input of a test image in a single generator with only changing the sym-parameter. The numbers in the parentheses are sym-parameters for each A, B, and C domain.

Abstract

Recent advances in image-to-image translation have led to some ways to generate multiple domain images through a single network. However, there is still a limit in creating an image of a target domain without a dataset on it. We propose a method that expands the concept of 'multi-domain' from data to the loss area and learns the combined characteristics of each domain to dynamically infer translations of images in mixed domains. First, we introduce Sym-parameter and its learning method for variously mixed losses while synchronizing them with input conditions. Then, we propose Sym-parameterized Generative Network (SGN) which is empirically confirmed of learning mixed characteristics of various data and losses, and translating images to any mixed-domain without ground truths, such as 30% Van Gogh and 20% Monet and 40% snowy.

1. Introduction

Recently, literature on multi-domain deep image translation has introduced many methods that learn the joint distribution of two or more domains and find transformations among them. Particularly, a single generator is able to translate images to multiple domains based on training data distributions [3, 19, 24]. However, translating style features across domains seen by the model is different from 'creativeness'. Consider a case of generating a translated image with a style that is 20% of Van Gogh, 50% of Picasso and 30% of the original image. Since the ground truths for learning such a translation do not exist, the target distribution to approximate can not explicitly be provided for conventional deep generative networks.

If we construe that the optimum of the target style is a weighted sum of optima of the candidate styles, then the objective function can be defined by a weighted sum of objective functions of those. In this end, if the weights are set as hyper-parameters, they can be preselected and learned to generate images outside of a domain even without ground truths [5]. Even in such a case, not only a criterion on selections of the parameters is vague, but every training for cross-domain translations must also be impractically done with a unique set of weights. Therefore, it would be more efficient to dynamically control them during inferences for desired translations. We, in this paper, present a concept of *sym-parameters* that enable human users to control them so that influences of candidate domains on final translations can be heuristically adjusted during inferences.

In our method, along with inputs, sym-parameters are inputted as a condition into our proposed generator network, Sym-parameterized Generative Network (SGN). And also, these sym-parameters are synchronously set as weights for the linear combination of multiple loss functions. With the proposed setting, we have verified that a single network is able to generate corresponding images of a mixed-domain based on an arbitrarily weighted combination of loss functions without direct ground truths. While an SGN utilizes multiple loss functions that conventional image-to-image translation models use (e.g. reconstruction loss, GAN (adversarial) loss, perceptual loss), the sym-parameter conditions the weights of these losses for variously purposed translations. If an SGN, as an exemplar case depicted in Figure 1, uses GAN loss for training Van Gogh style and perceptual loss for *Udnie* of *Francis Picabia*, sym-parameters allow adjusting the ratio of the styles to create correspondingly styled images. Through experiments, we found that sym-parameters conditioned within a model in the ways performed in typical conditional methods [3, 15, 28], fail to yield our intended generations. To overcome this, we additionally propose Conditional Channel Attention Module (CCAM).

To summarize, our contributions are as follows:

(1) We propose the concept of sym-parameter and its learning method that can control the weight between losses during inferences.

(2) We introduce SGN, a novel generative network that expands the concept of 'multi-domain' from data to the loss area using sym-parameters.

(3) Experimental results show that SGN could translate images to mixed-domain without ground truths.

2. Related work

Generative Adversarial Networks Recently, Generative Adversarial Networks (GANs) [6] have been actively adapted to many image generation tasks [1, 11, 16, 20]. GANs are typically composed of two networks: a generator and a discriminator. The discriminator is trained to distinguish the generated samples (fake samples) from the ground-truth images (real samples), while the generator learns to generate samples so that the discriminator misjudges. This training method is called adversarial training which our method uses for both the generator and the discriminator to learn the distribution of a real dataset.

Conditional Image Synthesis By conditioning concurrently with inputs, image generation methods learn conditional distribution of a domain. CVAE [22] uses conditions to assign intentions to VAEs [13]. Conditional image generation methods that are based on GANs also have been developed [2, 4, 17, 18, 19, 26], using class labels or other characteristics. Conditional GANs are also used in domain transfers [12, 23] and super-resolution [16]. While the methods from [3, 19] use discrete conditioning (*either 0 or 1*), our method uses continuous values for input conditioning.

Image to Image Translation While there exist generative models that generate images based on sampling (*i.e. GANs, VAEs*), models that generate images for given base input images are also studied. They mostly use autoencoders [14] and among them, one of the most representative and recent works is [9], which uses adversarial training with conditions. CycleGAN [27] and DiscoGAN [12] translate either style or domain of input images. Johnson *et al.* [10] have proposed perceptual loss in order to train feed-forward network for image style transformation. Since most of them use convolutional autoencoders with ResBlocks [7] or U-Net [21] structure, we also utilize the structure of CycleGAN [27], but additionally apply sym-parameters to image-to-image translation tasks in the form of continuous valued conditions. We also adopt the perceptual loss harmonized with the GAN loss and the reconstruction loss terms.

One generator to multiple domains Many have extended the study of image-to-image translations to multiple domains with a single generator network. IcGAN [19], StarGAN [3] and SingleGAN [24] address the problem of previously reported generative models that they are stuck with two domains, and achieve meaningful results on their extended works by using hard labels of each domain. Methods regarding image generation problems also have been proposed. ACGAN [18] uses auxiliary classifier to generate images by providing class information as a condition in the input. In different perspective, CAN [5] tries generating artworks by blending multiple domains. The method trains the generator of GAN to confuse the auxiliary classifier to judge fake samples in forms of uniform-distribution. In this study, we make a good use of conditions synchronized with loss functions not only to transfer to multiple domains, but also to mix each styles simultaneously by using expandable loss terms.

3. Proposed Method

Our goal is to learn distributions from multiple domains by varying weighted loss functions in order to dynamically translate images into a mixed-domain. In order to control mixing ratio during inferences, the corresponding conditions must be inputted and trained with the model. For such purpose, we present *sym-parameters* which are symmetrically set inside (as condition inputs) as well as outside (as weights of multiple loss functions) of a generator. The symparaemters allow transitive learning without explicit ground truth images among diverse mixtures of multiple domains and loss functions. The generator of sym-parameterized generative networks (SGN) can thus be controlled during inferences unlike conventional generators that infer strictly as optimized for a specific dataset or a particular loss function.

3.1. Sym-parameter

By trying to find not only the optimum of each candidate objective function but also the optima for various combinations of them, we desire to control the mixing weights during inferences. We propose human controllable parameters, ***Sym-parameters***, that can replace typical hyper-parameters for weighing multiple loss functions. As the prefix "sym-" is defined in dictionaries as "with; along with; together; at the same time", sym-parameters are fed into a model, *symmetrically* set as weights of the candidate loss functions and *synchronized* after training. If k number of different loss functions, $\mathcal{L}_1, \cdots, \mathcal{L}_k$, are engaged, then the sym-parameter S is defined as a k-dimensional vector $(s_1, \cdots, s_k)$. The total loss of a model $f(x, S)$ that takes inputs x and sym-parameters S is:

$$\mathcal{L}(f, S) = s_1\mathcal{L}_1(f(x, S)) + ... + s_k\mathcal{L}_k(f(x, S))$$
$$\text{where} \sum_{i=1}^{k} s_i = 1 \text{ and } s_i \geq 0 \text{ for all } i \in [1, k]. \tag{1}$$

Having the total sum of sym-parameters to be 1, the total loss $\mathcal{L}$ defined by the function f and sym-parameters S is a weighted sum of sub-loss functions, each of which is weighed by the corresponding element of S. The conventional hyper-parameter model predicts output y using the $\hat{y} = f_{h_t}(x)$ model with hyper-parameter h_t. In this case, it is difficult to predict y for $h_{t'}$, that is not used during training, because the network f is not a conditional function for the hyper-parameter $h_{t'}$ that was not used at training time. However, our model uses the weight of losses as input S in the form $\hat{y} = f(x, S)$, and f has conditional output for S as well as x. Thus, it can predict $\hat{y}$ for various combinations of losses that were used in training. Figure 2 depicts the concept showing the difference between a sym-parametrized model and conventional models using multiple loss functions weighted by hyper-parameters. While a new model is required for learning each combination of weights if conventional methods are used, our method allows a single model to manage various combination of weights through one learning. We, in the experiment section, verify that not

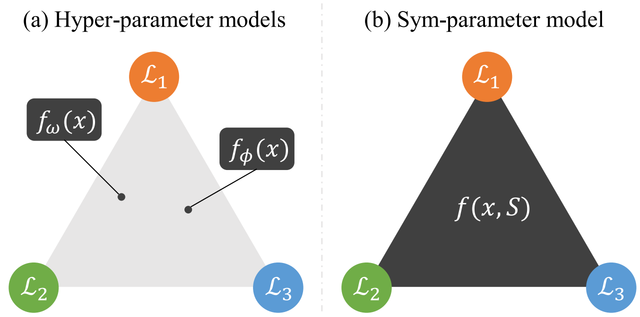

Figure 2. **The concept of sym-parameter** (a) This model should be learned by changing the hyper-parameters when it is required to use different weights among multiple losses. (b) Our proposed method has the effect of modifying the weight among losses by changing the sym-parameter S for inference in a single model. $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ represent different types of losses. x and f are the input and the output function. ω and ϕ are the hyper-parameters

only each sub-objective function is optimized but various linear sums of them also are. For example, if a neural model wants to perform both the regression and the classification tasks, the optimization is processed towards minimizing the regression loss when $S = (1, 0)$, the classification loss when $S = (0, 1)$ and the weighted sum of the losses when $S = (i, j)_{i,j \in \mathbb{R}, i+j=1}$.

Training with Dirichlet Distribution As mentioned above, a sym-parameter is represented as a vector which has the same number of dimensions as the number of loss functions. The vector's values are randomly selected during training in order to synchronize accordingly with sym-parameterized combinations of loss functions. To do so, the sym-parameter values are sampled based on Dirichlet distribution. The probability distribution of a k-dimensional vector with the sum of positively valued elements being 1, can be written as:

$$p(S) = \frac{1}{B(\alpha)} \prod_{i=1}^{k} s_i^{\alpha_i - 1}, \text{ where } B(\alpha) = \frac{\prod_{i=1}^{k} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{k} \alpha_i)}. \tag{2}$$

Here, $B(\alpha)$ is a normalization constant and $\Gamma(\cdot)$ represents Gamma function. When $k = 2$, the distribution boils down to Beta distribution. Using Dirichlet distribution allows the sum of sym-parameter values to be 1 and enables adjusting the distribution with by changing the concentration vector $\alpha = (\alpha_1, \cdots, \alpha_k)$.

3.2. Sym-parameterized Generative Networks

Using sym-parameters that allow inferences of various mixtures of losses, we propose sym-parametrized generative networks (SGN) that translate images to a mixed-domain. Our method is able to either generate images from latent inputs or translate styles with image inputs as long as sym-parameters are inputted along with the inputs and they define a linear combination of loss functions. Figure 3

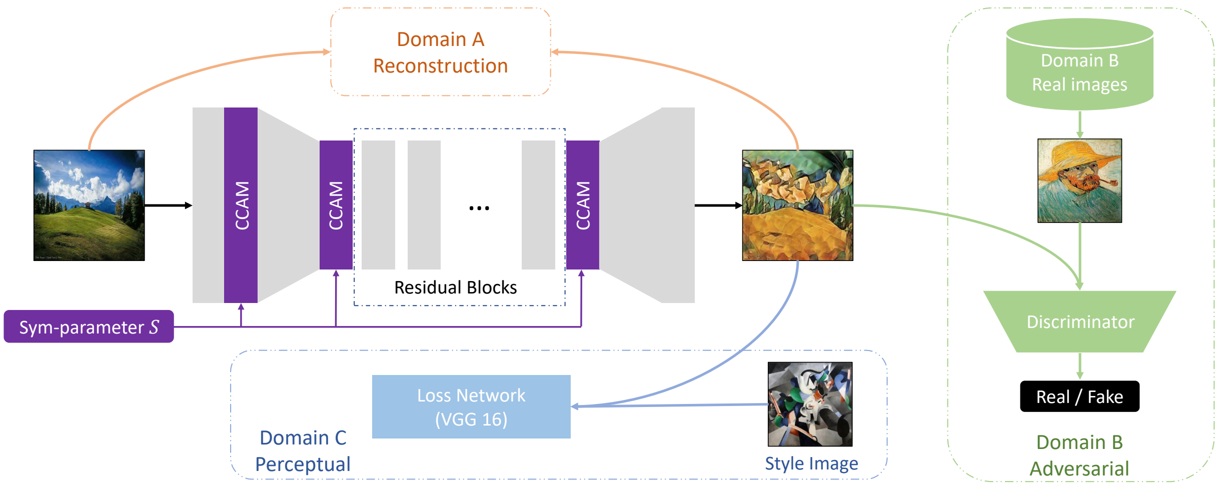

Figure 3. **Overall Structure of SGN for Three Different Losses** This diagram illustrates the case where SGN uses *reconstruction*, *adversarial* and *perceptual* loss as A, B and C domain. For the A, B, and C domains, the SGN uses the weighted sum of the losses with the sym-parameter $S = (s_1, s_2, s_3)$. Therefore, the full objective of the generator is $\mathcal{L}_G = s_1\mathcal{L}_A + s_2\mathcal{L}_B + s_3\mathcal{L}_C$.

illustrates the structure of SGN for image-to-image translation. Selection of loss functions are not necessarily limited for image generation tasks, and the following is a representative loss for a generator G with reconstruction loss $\mathcal{L}_{rec}$, adversarial loss $\mathcal{L}_{adv}$ and perceptual loss $\mathcal{L}_{per}$ weighted by sym-parameters (s_1, s_2, s_3):

$$\mathcal{L}_G = s_1\mathcal{L}_{rec} + s_2\mathcal{L}_{adv} + s_3\mathcal{L}_{per}. \tag{3}$$

Here, each loss function may cope with different objective and dataset for more diverse image generations such as using two of adversarial losses one of which for Van Gogh and another for Monet style domain.

While either reconstruction loss or perceptual loss does not require to train an additional network, a discriminator must be trained along with an SGN model for the adversarial loss, and a distinct training criterion of the discriminator is needed for SGN. Because our method generates images based on linearly combined losses with sym-parametrized weights, the weight on the loss of a discriminator must be also set accordingly. Therefore, discriminator must be trained with the weight assigned on the generator loss in an adversarial manner:

$$\mathcal{L}_D = -s_2\mathcal{L}_{adv}. \tag{4}$$

A trained SGN can translate images with specific sym-parameters, or generate random images from random variables sampled from Dirichlet distribution defined by an input image.

3.2.1 Conditional Channel Attention Module

SGN takes a continuous valued sym-parameter vector along with inputs and generates images reflecting characteristics

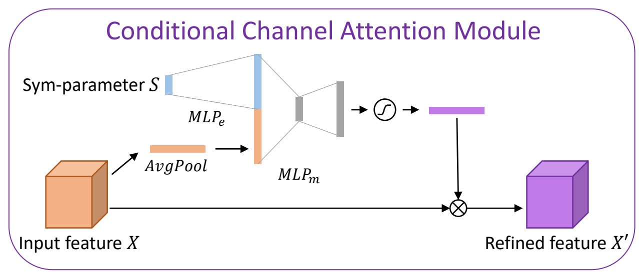

Figure 4. **Structure of CCAM** CCAM is a lightweight module that takes a feature of previous layer X and a sym-parameter S as an input, generates a map for channel attention through MLP layers, and refines the input feature with this attention map. $\otimes$ denotes a channel-wise multiplication.

of various parts of a mixed-domain, which covers wider range of target distribution than multi-domain models do with discrete conditions. Since domain injection used in conventional conditional generators is empirically shown to be inadequate for our purpose, we propose another injection method for sym-parameterized conditionings, named Conditional Channel Attetion Module (CCAM). Inspired by SENet [8], CCAM is a channel attention model that selectively gates feature channels based on sigmoid attentiveness. CCAM also allows SGN to have a fully convolutional structure and manage various spatial sizes. CCAM's structural details are depicted in Figure 4, and the module can be written as:

$$CCAM(X, S) = X \cdot \sigma(MLP_m([MLP_e(S), AvgPool(X)])), \tag{5}$$

where $X \in \mathbb{R}^{H\times W\times C}$ represents feature maps outputted from a preceding layer and $[\cdot,\cdot]$ denotes the concatenation operation. The features are shrunk to $1 \times 1 \times C$ through an average pooling, and the sym-parameters are represented

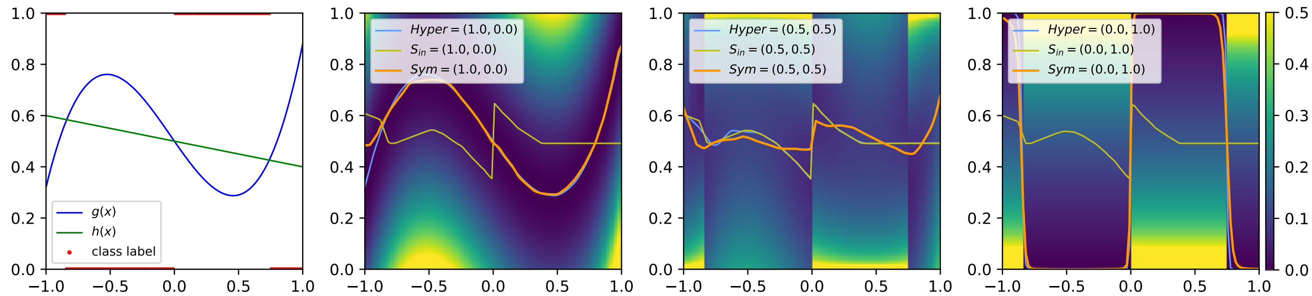

Figure 5. **Result of 1-D toy problem with sym-parameter.** The left image shows the function $g(x)$, which defines the regression label, and class labels are defined as 1 if $g(x) < h(x)$ or 0 otherwise. The color maps on the right depict the calculated loss values for $L(f, S)$ and the plot *Sym* is the actual output of the trained $f(x, S)$ on the given S. *Hyper* denotes the output of multiple hyper-parameter models for each weights. S_{in} uses the sym-parameter as an input, but uses fixed weight (0.5, 0.5) for each loss. This result shows that our method follows the weighted loss according to the sym-parameter S.

in the same size as the pooled features through MLP_e. After MLP_m creates attention maps based on the concatenation of input features and sym-parameters, a sigmoid function σ activates to produce the channel attention map, $M \in \mathbb{R}^{1\times1\times C}$. The output features of CCAM is then created by channel-wise multiplication of the channel attention map M and the original feature map X. For such a continuous conditioning case, CCAM allows superior efficiency of image generations over channel-wise inclusion of domain information along with RGB channels [3] or domain code injection through central biasing normalization [24].

4. Implementation of SGN

Network Architecture The base architecture of SGN has adopted the autoencoder layers from [3, 10, 27], which consists of two downsampling convolution layers, two upsampling transposed convolution layers with strides of two and nine residual blocks in between them. In this architecture, CCAM modules are inserted for conditioning at three positions, before the downsampling convolutions, after the downsampling convolutions and lastly, before the upsampling convolutions. For $\mathcal{L}_{adv}$, we use PatchGAN discriminator, introduced by Isole *et al.* [9] where discriminator is applied at each image patch separately. The choice of feedforward CNN for $\mathcal{L}_{per}$ is VGG16 with consensus on the work of Johnson *et al.* [10]. More details of the architecture and training will be handled on supplementary materials.

5. Experiments

Since, to the best of our knowledge, our method is the first approach that allows neural networks to dynamically adjust the balance among losses at inference time, there are no baselines for direct comparisons. We perform experiments for qualitative and quantitative evaluations on how well the sym-parameter works. And to focus on fair investigations based on the results, we have preserved the models of the existing methods and their individual losses as reported except for few minor changes. In this section, we first investigate the behavior of sym-parameters based on different loss functions through a 1-D toy problem. We then experiment an SGN on image translations to mixed-domains. And lastly, CCAM's role within the SGN is reviewed.

Weights	**Sym-parameter model**			**Hyper-parameter models**		
$(\mathcal{L}_r, \mathcal{L}_c)$	$\mathcal{L}$	$\mathcal{L}_r$	$\mathcal{L}_c$	$\mathcal{L}$	$\mathcal{L}_r$	$\mathcal{L}_c$
(1.00, 0.00)	0.0001	**0.0001**	0.2148	0.0000	**0.0000**	0.2167
(0.75, 0.25)	0.0483	0.0063	0.1743	0.0481	0.0062	0.1737
(0.50, 0.50)	0.0824	0.0347	0.1302	0.0815	0.0353	0.1277
(0.25, 0.75)	0.0878	0.1674	0.0613	0.0846	0.1597	0.0592
(0.00, 1.00)	0.0062	0.4113	**0.0062**	0.0013	0.4254	**0.0013**

Table 1. **Comparison against hyper-parameter models on 1D toy problem.** The single model with sym-parameter has a similar loss value to those from the hyper-parameter models learned separately. $\mathcal{L}_r$ and $\mathcal{L}_c$ denote the regression loss and the classification loss, respectively. $\mathcal{L}$ is weighted loss of $\mathcal{L}_r$ and $\mathcal{L}_c$ with the given weight parameters.

5.1. Toy Example: Regression and Classification with Single Network

In order to understand sym-parameters, we have designed an 1-D toy problem that we can calculate exact loss and visualize it. This allows us to confirm that our method can minimize multiple set of weights of multiple losses. And we can compare with hyper-parameter models whether a single sym-parameter model can replace various hyper-parameter models. We have defined a polynomial function $g(x)$ that takes one dimensional vector $x \in [-1, 1]$ and outputs y_r. Also, with $g(x)$ and a linear function $h(x)$, y_c is represented as a binary class label of 1 if $g(x) < h(x)$ or 0 otherwise. We have created a dataset consisting of (x, y_r, y_c) tuples. A sym-parametrized MLP network $f(x, S)$ concisely structured with three hidden layers

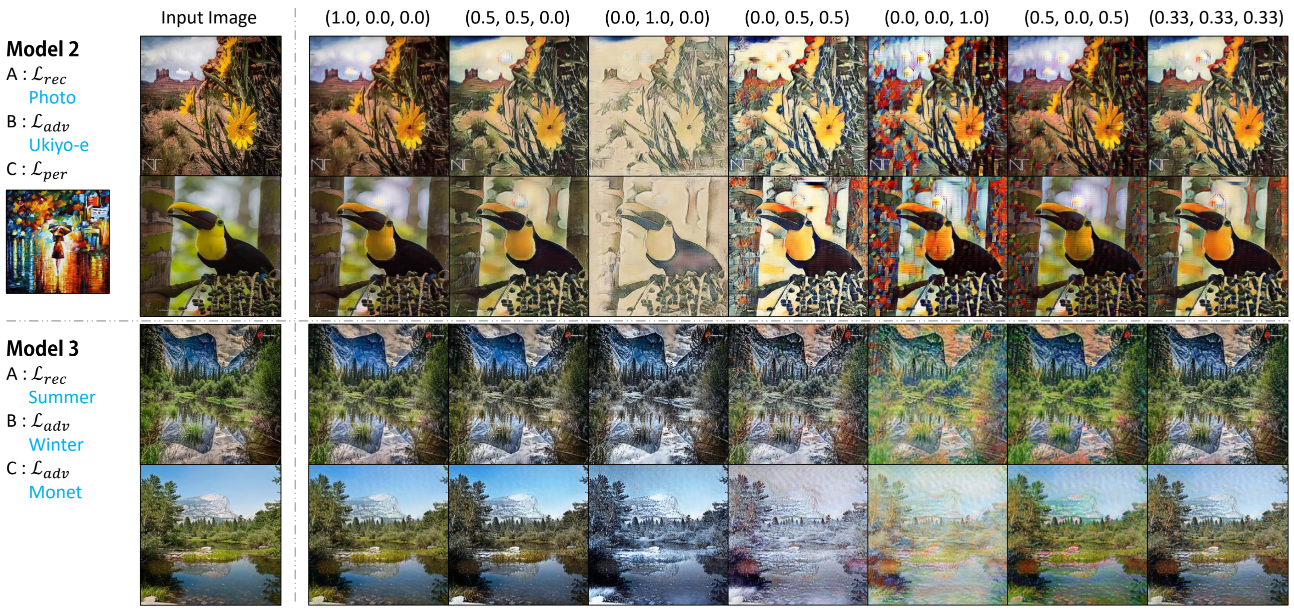

Figure 6. **Image Translation Results.** These images are translated from the test dataset with SGN model 2 and 3 according to the given sym-parameters S. The numbers in parentheses are the sym-parameters for the A, B, and C domains, respectively.

	A	B	C
Model 1	$\mathcal{L}_{rec}$, Photo	$\mathcal{L}_{adv}$, Van Gogh	$\mathcal{L}_{per}$, Udnie
Model 2	$\mathcal{L}_{rec}$, Photo	$\mathcal{L}_{adv}$, Ukiyo-e	$\mathcal{L}_{per}$, Rain
Model 3	$\mathcal{L}_{rec}$, Summer	$\mathcal{L}_{adv}$, Winter	$\mathcal{L}_{adv}$, Monet

Table 2. **The configuration of models.** We configure A, B and C domains for the three models used in the SGN experiments. Each domain is defined as a combination of loss and data.

Weights	**Single sym-parameter model**				**Hyper-parameter models**			
$(\mathcal{L}_{rec}, \mathcal{L}_{adv}, \mathcal{L}_{per})$	$\mathcal{L}_G$	$\mathcal{L}_{rec}$	$\mathcal{L}_{adv}$	$\mathcal{L}_{per}$	$\mathcal{L}_G$	$\mathcal{L}_{rec}$	$\mathcal{L}_{adv}$	$\mathcal{L}_{per}$
(1.0, 0.0, 0.0)	0.164	0.164	0.829	0.209	0.159	0.159	2.540	0.252
(0.5, 0.5, 0.0)	0.305	0.285	0.323	0.253	0.407	0.423	0.391	0.270
(0.0, 1.0, 0.0)	0.211	0.756	0.211	0.311	0.308	1.008	0.308	0.276
(0.0, 0.5, 0.5)	0.171	0.718	0.256	0.085	0.205	0.924	0.247	0.163
(0.0, 0.0, 1.0)	0.040	0.443	0.525	0.040	0.015	1.039	1.375	0.015
(0.5, 0.0, 0.5)	0.139	0.201	0.889	0.077	0.117	0.184	3.054	0.050

Table 3. **Comparisons with six hyper-parameter models in image-to-image translation domain.** The weighted loss, $\mathcal{L}_G$, of the single SGN has a similar test error to the hyper-parameter models trained separately for each set of weights. This experiment uses the setting of *Model 1* in the Table 2.

is trained with the dataset to perform regression and classification in terms of y_r and y_c, respectively. The total loss is defined as $\mathcal{L}(f, S) = s_1\mathcal{L}_r + s_2\mathcal{L}_c$ with a sym-parameter $S = (s_1, s_2)$.

Figure 5 shows the results of the experiment. Colormaps on the right sub-figures show computed loss values when weight for each losses are $(1, 0)$, $(0.5, 0.5)$ and $(0, 1)$, respectively. And the plots in each sub-figure denote the results of sym-parameter model (Sym), hyper-parameter models ($Hyper$) and an additional ablation (S_{in}). The figures clearly show the sym-parametrized model $f(x, S)$ properly minimizes weighted losses according to the sym-parameter value. And the results are similar to the multiple hyper-parameter models, which are trained individually. Model S_{in} uses the sym-parameter as an input, but the weights of each losses are fixed to $(0.5, 0.5)$ at training. As the illustrated results show, outputs follow the case $H = (0.5, 0.5)$ regardless of the input S because the model S_{in} did not learn to change the loss function. Also, the same sym-parametrized model is quantitatively compared against various hyper-parameter models that are separately trained.

As can be seen in Table 1, the loss values of the models separately trained with hyper-parameterized weights do not differ much from the values of the single model trained with sym-parametrized weights. This implies that a single training of a model using sym-parameters may replace the models trained in multiple cases of hyper-parameters.

5.2. Image Translation to Mixed-Domain

We have set up three SGN models for experiments on image translations to mixed-domain. With the use of sym-parameters, SGN extends the concept of multi-domain to loss functions, and each domain is thus defined with combinations of loss functions and data. For our experiments, the datasets used in [10, 27] are combined with loss criteria of $\mathcal{L}_{rec}$, $\mathcal{L}_{adv}$ and $\mathcal{L}_{per}$ from Section 4 to train each model as presented in Table 2.

Qualitative Result Figure 1 illustrates translation results

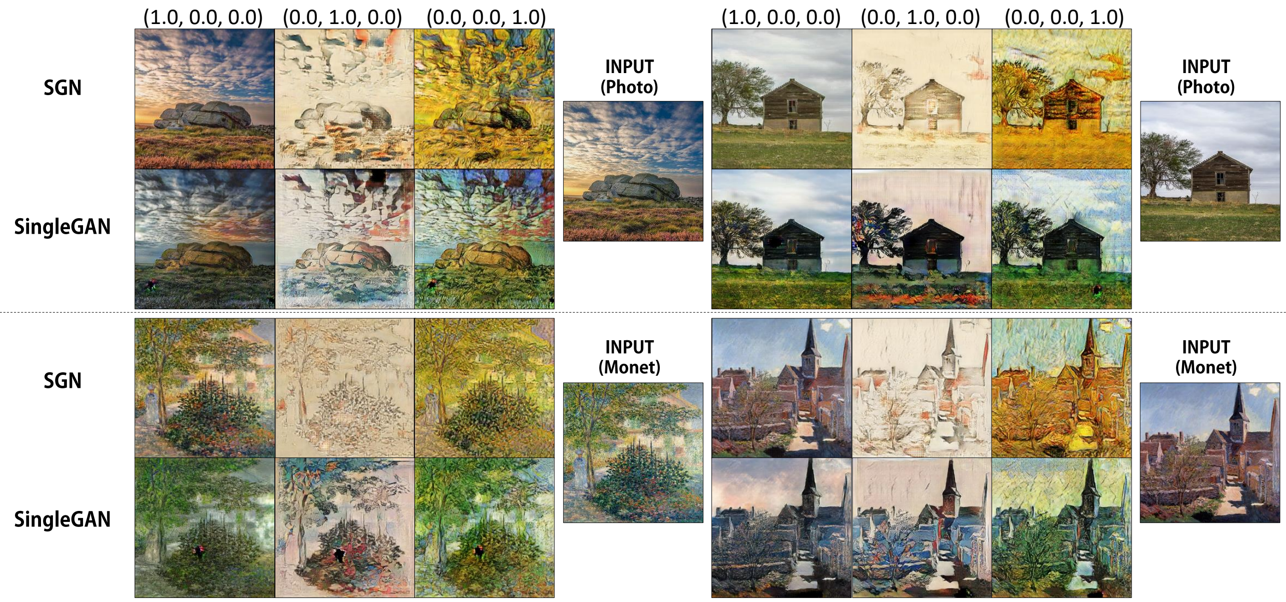

Figure 7. **Comparison with Style Transfer Model using Multiple Datasets** SingleGAN can generate images with multiple domains, but the output relies on trained datasets. Therefore it cannot reconstruct image even if it transfers the pictures of the same domain. But SGN can reconstruct new domain images like Monet because it is trained with reconstruction losses for sym-parameter S(1,0,0).

of Model 1 with various sym-parameters. Not only inter-translations among A, B and C domains are well generated, but the model also produces mixed generations with weighted characteristics of candidate domains. Especially, the successful image translations between domain B with adversarial loss and domain C with perceptual loss should be noticed; the generated images are influenced more by colors and styles of Van Gogh than by the test image input. As the numbers in the parentheses represent sym-parameter values, correspondingly styled images are well produced when mixed with 3 types of domains. Lastly, generated image with $S = (0.0, 1.5, 0.0)$ is an extrapolated result with a sym-parameter that is set outside of the trained range, and Van Gogh's style is still bolstered accordingly.

Figure 6 shows image translations yielded by two other models. While an SGN can be trained with different loss functions and datasets as done for Model 1 and 2, it can also be trained with domains defined with two GAN losses for different datasets. As it can be seen in the figure, Model 3 is able to translate a summer Yosemite image not only to a winter Yosemite image but also to a winter image with Monet style with $S = (0.0, 0.5, 0.5)$. This result well depicts how weighted optima of multiple objective functions can be expressed. More test results of each model are provided in the supplementary material.

Quantitative Result It is generally known to be difficult to define quantitatively evaluating metrics for generative models. Nonetheless, it is logical to quantitatively examine by checking loss values for different S values since our aim is to find optimal of variously weighted sum of losses. If trained sufficiently in terms of (1) and (3), the corresponding domain's loss should be minimized when a domain's weight is 1, and when the weight is mid-valued as 0.5, resultant loss of the domain should be valued in between the values when the weight is 1 and 0. Furthermore, we can compare these values with multiple hyper-parameter models that are individually trained.

Loss values of trained Model 1 are measured with a test dataset and enumerated in Table 5.2. The numbers in the table represent averaged loss values of each loss function the generator uses. In the table, each domain yields its minimum loss value when weighted with 1. And these are similar to the multiple models trained using a corresponding hyper-parameter. The largest difference is that the hyper-parameter models have a high loss value when the weight of a specific loss is zero at the time of testing, such as $S = (0, 0.5, 0.5)$, while the SGN has a relatively low value. This is because it is advantageous for the SGN to learn that the loss is minimized for similar S values such as $S = (0.1, 0.45, 0.45)$.

Multiple Datasets vs Multiple Losses We have performed a comparison with SingleGAN [24], which can handle multiple datasets, to clearly show the difference of our multi-loss approach from the conventional methods. For fair comparison, SingleGAN has been trained using Photo as input with three output domains: Photo, Ukiyo-e and Van Gogh. SGN used the reconstruction loss for Photo, GAN losses for Ukiyo-e and Van Gogh. As shown in Figure 7, SingleGan performs style transfer, not reconstruction, for the condition (1,0,0). These results mean that the input images are trans-

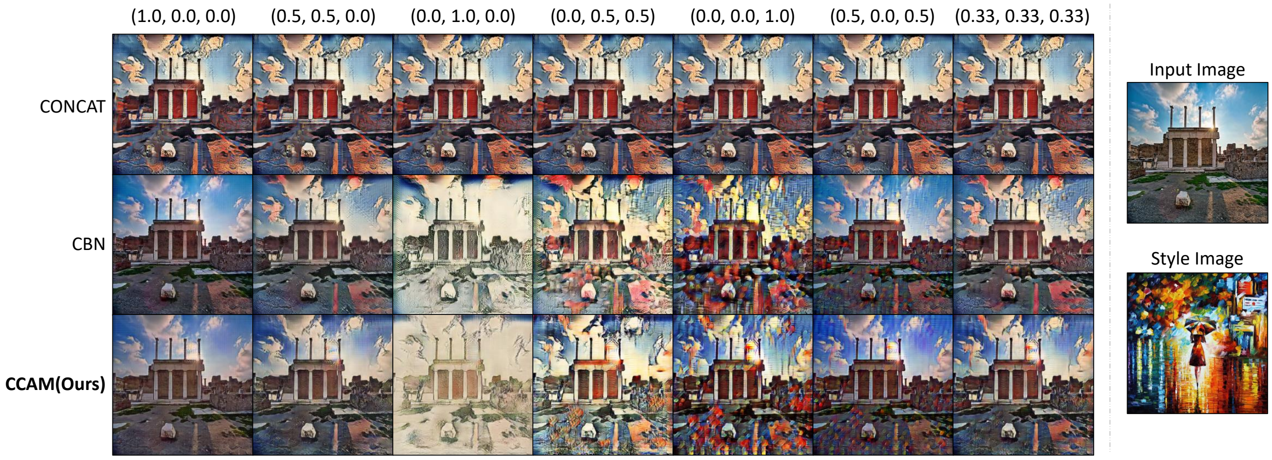

Figure 8. **Results based on different injection methods of sym-parameters.** All settings except for the sym-parameter injection method are equivalently set up as Model 2 (A: $\mathcal{L}_{rec}$, B: $\mathcal{L}_{adv}$, Ukiyo-e, C: $\mathcal{L}_{per}$).

ferred to Photo style. This can be seen more clearly by using Monet images as input, which are not used for training. In contrast, SGN performs reconstruction for S = (1, 0, 0) according to the definition of the trained loss, and also operates to reconstruct Monet images that are not used in the training phase at all. This result indicates that SGN behaves according to the loss characteristics learned as we intended.

Continuous Translation Since the sym-parameter is represented as a k-dimensional continuous vector, continuous inter-domain translations are achieved during mixed-domain image translations. Such characteristics of SGN allows seamless mixed-domain transitions among candidate domains through video which is included in the supplementary.

5.3. Conditional Channel Attention Module

We perform sym-parameter injections into SGN through CCAM since conventional condition injecting methods are empirically shown inefficient for our method. We have thus experimentally compared the performance results among different cases of condition injections, which is depicted in Figure 8. The injection method labeled CONCAT represents the method of depth-wise concatenations with a latent coded vector repeated to be in the same spatial size as features. Central biasing normalization (CBN) [25] is also experimented for comparisons. We have adjusted the instance normalization of the down-sampling layers and the residual blocks of the generator network as similarly used in [24].

CONCAT method produces outputs that are alike despite of various combinations of sym-parameters and fails to generate images in mixed-domain correspondingly. This method works promisingly for discrete condition injections as reported in [3, 15, 28] but struggles in the SGN using continuous conditions. This phenomenon is reasonable considering that the normalization within the generator perturbs the values of the sym-parameter and differences in the values are susceptible. CBN method performs better than CONCAT, generating comparably more various images with given sym-parameters. Yet, its generations are rather biased to one domain than a mixed-domain targeted by sym-parameters, and also follow with some artifacts. Since CBN uses bias and thus is hard to exclude the possibility on influence of a particular channel, additional interferences among candidate domains may occur. Among the experimented condition injecting methods, our proposed CCAM is more suitable for the sym-parameter and SGN.

6. Conclusion

In this paper, we propose a sym-parameter that can extend the concept of domain from data to loss and adjust the weight among multiple domains at inference. We then introduce SGN, which is a novel network that can translate image into mixed-domain as well as each domain using this sym-parameter. It is hard to say which method is proper and optimal when it comes to making a valid result for a mixed-domain without ground truth. However, if optimizing to the weighted objective of each domain is one of the effective methods for this purpose, SGN performs well to translate image to a target mixed-domain as shown in the experiments. We expect that the research will be extended to apply sym-parameter to more diverse domains, and to find more effective models for sym-parameter.

Acknowledgement
This work was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF-2017M3C4A7077582).

References

[1] David Berthelot, Thomas Schumm, and Luke Metz. Began: boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[2] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

[3] 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. *arXiv preprint*, 1711, 2017.

[4] Alexey Dosovitskiy, Jost Tobias Springenberg, Maxim Tatarchenko, and Thomas Brox. Learning to generate chairs, tables and cars with convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(4):692–705, 2017.

[5] Ahmed Elgammal, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. Can: Creative adversarial networks, generating" art" by learning about styles and deviating from style norms. *arXiv preprint arXiv:1706.07068*, 2017.

[6] 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.

[7] 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.

[8] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks.

[9] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[10] 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.

[11] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[12] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. *arXiv preprint arXiv:1703.05192*, 2017.

[13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[14] Mark A Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

[15] Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, et al. Fader networks: Manipulating images by sliding attributes. In *Advances in Neural Information Processing Systems*, pages 5967–5976, 2017.

[16] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, volume 2, page 4, 2017.

[17] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[18] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. *arXiv preprint arXiv:1610.09585*, 2016.

[19] Guim Perarnau, Joost van de Weijer, Bogdan Raducanu, and Jose M Álvarez. Invertible conditional gans for image editing. *arXiv preprint arXiv:1611.06355*, 2016.

[20] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[21] 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.

[22] 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.

[23] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[24] Xiaoming Yu, Xing Cai, Zhenqiang Ying, Thomas Li, and Ge Li. Singlegan: Image-to-image translation by a single-generator network using multiple generative adversarial learning. *arXiv preprint arXiv:1810.04991*, 2018.

[25] Xiaoming Yu, Zhenqiang Ying, Ge Li, and Wen Gao. Multi-mapping image-to-image translation with central biasing normalization. *arXiv preprint arXiv:1806.10050*, 2018.

[26] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaolei Huang, Xiaogang Wang, and Dimitris Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. *arXiv preprint*, 2017.

[27] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint*, 2017.

[28] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, pages 465–476, 2017.