Informative Multimodal Unsupervised Image-to-Image Translation

Tien Tai Doan[1,2], Guillaume Ghyselinck[1], and Blaise Hanczar[2]

[1]Dental Monitoring, Paris, France

[2]IBISC Laboratory, University of Evry Val d'Essonne, Evry-Courcouronnes, France

Abstract

We propose a new method of multimodal image translation, called InfoMUNIT, which is an extension of the state-of-the-art method MUNIT. Our method allows controlling the style of the generated images and improves their quality and diversity. It learns to maximize the mutual information between a subset of style code and the distribution of the output images. Experiments show that our model cannot only translate one image from the source domain to multiple images in the target domain but also explore and manipulate features of the outputs without annotation. Furthermore, it achieves a superior diversity and a competitive image quality to state-of-the-art methods in multiple image translation tasks.

Keywords

Multimodal Image-to-Image Translation, Mutual Information, GANs, Manipulating Features, Disentangled Representation

1. Introduction

Image-to-image translation can be described as the general problem of mapping an image from one domain to another domain. This seemingly simple approach is the foundation of many applications in the field of computer vision such as colorization [1], style transfer [2], super-resolution [3], denoising, inpainting [4]. Moreover, image-to-image translation has been also applied for data augmentation and achieved competitive results [5] [6] [7]. Based on the availability of data, the problem can be considered as supervised learning where the dataset contains paired samples; or unsupervised learning where the dataset consists of two independent sets of images. This work focuses on the unsupervised image-to-image problem which is more applicable due to its ease of obtaining data but also more challenged in terms of training.

Unsupervised image-to-image translation leads us to the idea that an image in a domain can be translated into multiple images in the second domain, which means the translation can be multimodal. For example, in image colorization, one image can be colored in multiple ways. Some methods [8] [9] have been proposed to use a noisy vector as an additional input of the decoder. The style of the generated images can then be manipulated by changing the values of the style-vector. However, the style-vectors in existing methods are entangled and the translated images are not interpretable as a result. Lacking control over features of the output can be problematic when important information are linked to these features. In the work of Cohen et al. [10], it is shown that CycleGAN was adding/removing tumors from images when transforming MRI images from Flair to T1, especially when there is an imbalance among classes in the training data. Therefore, learning to control the features of the translated images is essential. In this paper, we

David C. Wyld et al. (Eds): SPTM, IPPR, CSIT, BDAP - 2021
pp. 37-51, 2021. CS & IT - CSCP 2021 DOI: 10.5121/csit.2021.110503

propose some improvements on MUNIT [9] - a standard in the field of multimodal image translation - by applying the mutual information maximization technique. Our method, called InfoMUNIT, generates more diverse images and especially can manipulate their textural and structural features without requiring any annotation.

2. Related Works

2.1. Multimodal unsupervised image-to-image translation

The translation of images from one domain to another has been a challenging problem in computer vision. Thanks to the evolution of convolutional neural networks, especially generative adversarial networks (GANs) [11], many deep learning models have been recently proposed to address the problem of image translation and achieved impressive outcomes.

The research of Isola et al. in [12] is one of the earliest works on image-to-image translation based on GANs. In [13], the method is upgraded using multi-scale generators and discriminators to translate high-resolution images. These methods require paired data for training which is not usually available in practice.

Learning to translate images using unpaired data is more challenging than with paired data because we do not know exactly which data-point in the source domain corresponds to which one in the target domain. Thus, it is reasonable to add some constraints to the training when it is possible. One popular assumption in most image-to-image translation research is that the structure of an image must not be changed too much by the translation. This is similar to language translation, in which, a phrase must have the same meaning after being translated to another language. Shrivastava et al. [14] propose a training strategy in which, a deep network learns to transform the style of synthesized images to make them look more real. To preserve the annotation, they add a pixel-wise loss between the input and output of the style transfer network. Similar approaches are applied in later works such as specific-task loss [15], semantic features [16], or distance between pairs of input samples [17] and so on. These constraints are useful for some specific tasks and datasets but cannot be applied robustly.

Cycle consistency is another well-known loss function being used in many bi-direction image translation models such as DualGAN [18], CycleGAN [19], and DiscoGAN [20]. In these networks, an image being translated from domain A to domain B can be also translated backward to obtain the original image. As this cycle loss is not domain related, it can be applied to most of the bi-direction translation models. In [21], Almahairi et al. extend CycleGAN for learning a many-to-many mapping by combining images with noises. Despite its ease of use, cycle loss does not assure any consistency in terms of annotation which means labels of images can be flipped by the translation. Hoffman et al. [7] proposed to use both cycle consistency and semantic consistency during the training. However, this semantic constraint is not always accessible because it requires a pretrained classifier of a similar dataset.

Another way to preserve the structural information after the transformation is to define a shared latent space where domain-independent features are stored. In UNIT [6], Liu et al. propose to break the translation into two stages: encoding the source image to a latent code and then decoding this code to an image in the target domain. To gain some control over features of the translated image, Huang et al. develop MUNIT as an extension of UNIT, by splitting the latent code into two parts: content and style. With this network, multimodal translation can be done by combining a content code of an image with randomized style codes. The output images inherit content (or structure) from the input image but differ in style (Eg. textures or colors). In DRIT++ [22], a similar idea to

MUNIT is introduced but differs slightly in style transformation techniques. Both MUNIT and DRIT++ store image style in a completely entangled manner, they offer no control over the style of output images despite their diversity. In this work, we extend MUNIT by disentangling the style code without requiring any additional annotations or pretrained networks.

2.2. Unsupervised disentangled representation learning

Learning the features of images in an unsupervised fashion has received attention from the computer vision community for years.

Most methods in the early stage were based on restricted Boltzmann machines [23] and stacked auto-encoders [24]. Models in [25] and [26] were proposed for semi-supervised learning and achieved promising results on the MNIST dataset. In [27], a GANs-based method were shown to represent the dataset in a code space where basic linear structures are supported.

Another branch of research uses labeled data to learn disentangled representation. The representation is divided into two parts: one for the given labels and on for other features. Similar fashions of training can be found in different model structures such as bilinear models [28], multi-view perceptron [29], variational autoencoders (VAEs) [30] and adversarial autoencoder [31].

For minimizing the dependency on variation labels, weakly supervised methods were developed. Reed et al. [32] propose correspondence-based training strategies for a higher-order Boltzmann machine consisting of hidden units groups and each group represent a factor of variation. A similar technique is applied to VAE in [33] to manipulate brightness and pose in images of 3D objects. These two methods share one drawback that they require grouped data points which are difficult to collect in real-life applications.

There are not many works on completely unsupervised disentangled representation learning. In [34], hossRBM is introduced as a generalized version of spike-and-slab restricted Boltzmann machine, which entangles variation factors using its higher-order interactions on latent variables. However, the method is not effective in terms of computation cost.

In InfoGAN [35], Chen et al. develop an extension of GAN which maximizes the mutual information between certain variables in the latent code and samples from an unlabeled dataset. This technique enables the model to learn the disentangled representation of images without asking for labels. In this work, we upgrade MUNIT with the mutual information learning objective from InfoGAN to enable it to manipulate features of the translated images.

3. Method

Our objective is to translate images from a source domain A to a target domain B, and at the same time to learn the representation of the target domain. Following the idea called *partially shared latent space* in [9], we assume that each image can be encoded as a content code which contains general structural information and a style code which defines how the image will look like. In the state-of-the-art methods, this style latent code is entangled. In this work, we disentangle this style code by maximizing the mutual information between this code and the generated image.

3.1. Network architecture

Let x_A and x_B be two images from domain A and B respectively. Our objective is to learn a function $F_{A\to B}$ that projects images from domain A to domain B, $\hat{x}_{A\to B} =$

$F_{A\to B}(x_A)$. This function can be decomposed into parts: the encoder and the generator. The encoder E_A^c extracts the content code c_A from the image. The content code is a matrix representing the content of an image independently of its style. The generator G_B generates images in domain B from an content code and a style code s : $\hat{x}_{A\to B} = G_B(c_A, s) = G_B(E_A^c(x_A), [s', i])$. Since we want a one-to-many projection, a style code s_B is inputted in the generator to introduce variability in the generated images. The style code s is a vector created by concatenating two parts s' and i where s' stores entangled style of the generated images, i contains disentangled features of the generated images. s' and i are drawn from a normal distribution $N(0, I)$. The generator learns a function that links the points from a Gaussian distribution to the different ways to apply the style of domain B to a content code. In the same way, we define the function that projects images from domain B to domain A with generator G_A and encoder E_A^c. Notice that the content space and style space are common to both domains. This is the generators that project a couple of points from these common spaces to the image sub-spaces corresponding to their domain.

For the learning of these functions, we need to complete our architecture with autoencoders and discriminators. Autoencoders are used to reconstruct the original images from their decomposition into a content code and a style code. Let E_A^s (resp. E_B^s) denote the encoder that extracts from an image of domain A (resp. B) its style code $s_A = [s'_A, i_A]$ (resp. $s_B = [s'_B, i_B]$). The autoencoder of domain A is therefore defined by $\hat{x}_A = G_A(E_A^c(x_A), E_A^s(x_A))$. Autoencoders are also used to reconstruct the content $\hat{c}_A = E_A^c(G_B(c_A, s))$ and style codes $\hat{s} = E_B^s(G_B(c_A, s))$. The discriminator D_B is used to align the distribution of images produced by the generator G_B with the distribution of original images from domain A. It is also used to disentangle the style variables contained in the vector i. In the same way, we define the autoencoders $\hat{x}_B = G_B(E_B^c(x_B), E_B^s(x_B))$, $\hat{c}_B = E_B^c(G_A(c_B, s))$, $\hat{s} = E_A^s(G_A(c_B, s))$ and discriminator D_A. Figure 1 shows the complete architecture of InfoMUNIT.

3.2. Model learning

The training of our model consists to minimize a combination of reconstruction losses and adversarial losses while maximizing the variational mutual information.

Similar to most auto-encoder based architecture, the encoders E_A^C and E_A^S compress input images to content code and style code while the generator G_A takes them to reconstruct the original image from domain A. The image reconstruction loss $\mathcal{L}_{rec}^{x_A}$ makes sure the encoder and decoder inverse each other. L_1 loss is chosen for the image reconstruction as it usually obtains well the sharpness of the reconstructed image. For the same reason, we have similar reconstruction losses for content code $\mathcal{L}_{rec}^{c_A}$ and style code $\mathcal{L}_{rec}^{s_A}$.

$$\mathcal{L}_{rec}^{x_A} = E_{x_A\sim p(x_A)}[\| G_A(E_A^c(x_A), E_A^s(x_A)) - x_A \|_1] \tag{1}$$

$$\mathcal{L}_{rec}^{x_B} = E_{x_B\sim p(x_B)}[\| G_B(E_B^c(x_B), E_B^s(x_B)) - x_B \|_1] \tag{2}$$

$$\mathcal{L}_{rec}^{c_A} = E_{c_A\sim p(c_A), s\sim\ p(s)}[\| E_B^c(G_B(c_A), s) - c_A \|_1] \tag{3}$$

$$\mathcal{L}_{rec}^{c_B} = E_{c_A\sim p(c_B), s\sim p(s)}[\| E_A^c(G_A(c_B), s) - c_B \|_1] \tag{4}$$

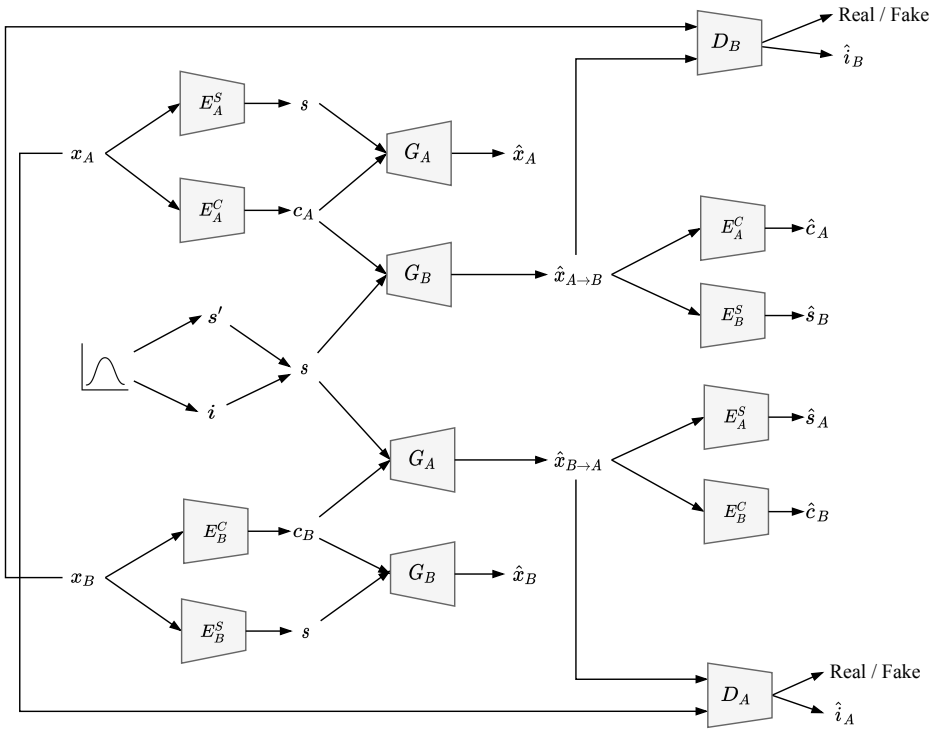


Figure 1: Overview of InfoMUNIT. In this structure, each image is encoded by two encoders into a style code and a content code, and reconstructed by a decoder (also called generator). For translating an image from a domain to another domain, we firstly extract its content code, then combine it with a random style code, and send them both to the generator of the target domain. A part of the style code is used to store disentangled features of output images. We also train a pair of discriminators to distinguish between generated images and real images for each domain. The generators are also trained to maximize the mutual information between features being extracted by those discriminators and the disentangled part in the style code.

$$\begin{aligned}\mathcal{L}^s_{rec} = & E_{c_A\sim p(c_A),s\sim p(s)}[\| E^s_B(G_B(c_A),s) - s \|_1] \\ & + E_{c_B\sim p(c_B),s\sim p(s)}[\| E^s_A(G_A(c_B),s) - s \|_1] \end{aligned} \quad (5)$$

where $p(x_A)$ (resp. $p(x_B)$) is the distribution of images from domain A (resp. B), $p(c_A)$ (resp. $p(c_B)$) is the distribution of content code extracted from images from domain A (resp. B), and $p(s)$ is the distribution of style code that is the unit Gaussian distribution $N(0, I)$. Note that the distributions $p(c_A)$ and $p(c_B)$ are unknown and the learning set do not contains examples of c_A and c_B., we need there fore to generate c_A and c_B samples from the encoders and training images $c_A = (E^c_A(x_A))$ and $c_B = (E^c_B(x_B))$.

The objective of the adversarial losses associated to the discriminators is to align the distributions of the real images with the distribution of the generated images. Like in the GAN, the discriminators try to predict if an image is a real one or an artificial image produced the generator. When the generators are frozen, the generators try to fool the discriminators in generating images close to the real ones. The adversarial losses are

defined by:

$$\mathcal{L}_{adv}^{A} = E_{x_B \sim p(x_B), s \sim p(s)}[log(1 - D_A(G_A(E_B^C(x_B); s)))] + E_{x_A \sim p(x_A)}[logD_A(x_A)] \quad (6)$$

$$\mathcal{L}_{adv}^{B} = E_{x_A \sim p(x_A), s \sim p(s)}[log(1 - D_B(G_B(E_A^C(x_A); s)))] + E_{x_B \sim p(x_B)}[logD_B(x_B)] \quad (7)$$

where the output of the discriminator $D_A(x)$ (resp. $D_B(x)$) is the probability that the image x is a real image from the domain A (resp. B).

Inspired by the idea of InfoGAN [35], we want a part of the style code to be disentangled features of the output in order to control and improve the diversity of the translated images. The style code is split into two parts $s = [s', i]$. To encourage the subvector i to represent disentangled features of the output, we maximize the mutual information between i and the generated images.

$$\mathcal{I}(i, G_B(c_A, [s, i])) \quad \text{and} \quad \mathcal{I}(i, G_A(c_B, [s, i])) \quad (8)$$

In practice, maximizing this mutual information is not achievable without access to the distribution $P(i|x)$ which is not available in our case. However, according to [36], we can define an additional distribution $Q(i|x)$ as an approximation of $P(i|x)$, and get a lower bound of the mutual information term. Thus we have:

$$\mathcal{I}(i, G_B(c_A, [s, i])) \geq L_{mi}(G_B, Q_B) = E_{i \sim p(i), x_{A \to B} \sim P(G_B(c_A, [s', i]))}[logQ_B(i|x_{A \to B})] \quad (9)$$

Where $p(i)$ is a normal distribution and $P(G_B(c_A, [s', i]))$ is the distribution of the images generated by G_B with the style vector $[s', i]$. In practice, Q_B shares the same layers of the discriminator D_B as they both extract features from $G_B(c_A, [s', i])$. Q_B is implemented as a secondary output of the discriminator D_B that is notes $\hat{i}$. This means the closer the vector i and predicted vector $\hat{i}$ are, the more mutual information between i and the generated image is achieved. In the same way, we define $L_{mi}(G_A, Q_A)$.

The learning of our model consists both to minimize the total loss w.r.t the encoders and generators and to maximize it w.r.t the discriminators :

$$\min_{E_A, E_B, G_A, G_B} \max_{D_A, D_B} \mathcal{L}(E_A, E_B, G_A, G_B, D_A, D_B) = \mathcal{L}_{dis}^{x_A} + \mathcal{L}_{dis}^{x_B} + \lambda_x(\mathcal{L}_{rec}^{x_A} + \mathcal{L}_{rec}^{x_B}) + \lambda_c(\mathcal{L}_{rec}^{c_A} + \mathcal{L}_{rec}^{c_B}) + \lambda_s(\mathcal{L}_{rec}^{s}) - \lambda_{mi}(L_{mi}(G_A, Q_A) + L_{mi}(G_B, Q_B)) \quad (10)$$

where λ_x, λ_c, λ_s and λ_{mi} represent the importance of each loss. In our trainings, we set $\lambda_x = 10$, $\lambda_c = \lambda_s = \lambda_{mi} = 1$ as the image reconstruction is the most important loss in our structure.

4. Experiments

4.1. Implementation Details

Our network consists of a content encoder, a style encoder, a generator, and a discriminator for each domain. We give the implementation details of each of these network.

4.1.1. Content Encoder

Input images are firstly led to the content encoder where they are down-sampled by strided convolutional layers and further processed by residual blocks. We apply Instance Normalization for all convolutional layers in the content encoder. The output of the content encoder is the content code in a form of a tensor.

4.1.2. Style Encoder

Similarly, the style encoder also down-samples input images using strided convolutional layers and a global pooling layer. A fully connected (FC) layer is applied to produce a style code as a vector consisting of 8 digits, in which, 2 final digits represent the information code (disentangled style) I_i of the image.

4.1.3. Generator

The generator takes content code and style code as inputs to reconstruct the initial input image. The content code goes through residual blocks and upsampling layers. These residual blocks are upgraded with Adaptive Instance Normalization (AdaIn) layers [37] which receive style parameters from a multilayer perception (MLP) which has the style code as its input.

4.1.4. Multi-purpose Discriminator

Our discriminator consists of two branches. The first branch is a traditional discriminator which can be found in most of the GAN-based models. The second branch consists of convolutional blocks to learn the Q distribution. These two branches share the first convolutional blocks.

4.1.5. Hyperparameters

In all our experiments in the paper, we apply Adam optimizer with β_1 and β_2 as 0.5 and 0.999 respectively. The learning rate is initially set to 0.0001, with a weight decay of 0.0001 applied every 100 thousand iterations. Our weight losses are $\lambda_x = 10$, $\lambda_c = \lambda_s = \lambda_{mi} = 1$.

4.1.6. Baselines

We compare our proposed method InfoMUNIT with following unpaired image-to-image translation techniques: CycleGAN[19], MUNIT[9] and DRIT++[22]. The training procedures of those methods are done using official source code and configurations provided by their authors on GitHub.com.

4.2. Evaluation

We use three performance measures that estimate the quality and the diversity of the generated images, to compare InfoMUNIT with the baselines.

4.2.1. Conditional Inception Score

Based on Inception Score (IS) [38], Huang et al. [9] introduced Conditional Inception Score (CIS) specified for evaluation of multimodal image-to-image tasks. While IS measures the quality and diversity of all generated images at once, CIS focuses on the diversity of images that are translated from the same input image. Having multiple input images in the test set, we compute CIS for each group of images generated from the same input, and finally, take the mean CIS for the whole test set.

4.2.2. Frechet Inception Distance

Frechet Inception Distance (FID) [39] computes the distance between the set of generated images and the set images in the target domain. It is computed by calculated the distance between the Inception feature vectors for the two sets of images. Thus, FID can be used for evaluating networks that are trained on specific datasets without requiring a classifier pretrained on an alike dataset. The lower FID we have, the more realistic the generated images are. Normally, those feature vectors are taken from the third pooling layer of the Inception model which contains 2048 features. Due to the small size of our datasets, we compute the distance using features of the second pooling layer containing 192 features.

4.2.3. LPIPS Distance

The translation diversity is also measured by LPIPS distance which is shown in [40] to be highly correlated with human judgment. We compute LPIPS distance on generated samples of each input image, then take the average value. The larger distance among them, the more diverse they are.

4.3. Datasets

We use multiple datasets for evaluating InfoMUNIT and compare its performance with state-of-the-art techniques on the task of image-to-image translation. Each dataset contains two sets of images and our network is trained to transform images between the two domains. We crop and down-sample all images to the size of 64×64, in RGB-color mode.

4.3.1. Edges↔Shoes and Edges↔Bags

These two datasets contain images of shoes and handbags along with their edges, introduced in the work of Isola et al. [12]. The edges↔shoes dataset contains 138667 pairs of samples while the edges↔bags dataset contains 49925 pairs. From each dataset, we keep 200 pairs of samples for testing and the rest for training. Note that we do not use the paired information of these two datasets.

4.3.2. Cats↔Dogs

The dataset is comprised of 1364 photos of dogs and 871 photos of cats, cropped to their heads [22]. We keep 100 images from each set for testing while the rest is used for training.

4.3.3. Portraits (Painted↔Real)

This dataset consists of 1814 painted portraits and real 6452 portraits captured by cameras [22]. We keep 100 images from each set for testing while using the rest for training.

5. Results

5.1. Image Quality

The qualitative comparison of InfoMUNIT and other methods is shown in Figure 2. The objective of InfoMUNIT is to increase the diversity and ability to control features of generated images compared to MUNIT and the state-of-the-art, while not hurting their quality. As being shown in Figure 2, the quality of images generated by InfoMUNIT are at least as good as the images from other methods. The result is confirmed in Table 1 where we apply FID to quantitatively evaluate the realism of the generated images. Even

Figure 2: Random samples generated by our method and baselines, trained on two datasets: edges→bags (left) and cats→dogs (right). The input images (and ground-truths) are displayed in the first column. Other columns show random outputs of baseline methods and InfoMUNIT.

Table 1: Frechet Inception Distance (FID). Lower value means better performance.

	InfoMUNIT	**MUNIT**	**CycleGAN**	**DRIT++**
edge2bag	2.81	2.56	4.23	**1.69**
bag2edge	7.68	8.52	58.53	**5.64**
edge2shoe	1.28	1.44	4.86	**1.13**
shoe2edge	4.69	8.83	88.03	**4.24**
dog2cat	9.24	13.48	**2.56**	21.59
cat2dog	6.31	6.31	**2.02**	18.14
paint2real	2.96	3.02	**2.56**	7.29
real2paint	8.60	8.51	**3.97**	18.85
Average	**5.45**	6.58	20.84	9.82

though InfoMUNIT does not out outperform other methods in terms of image quality in any task, its performance is stable across all tasks. The performance of InfoMUNIT is close to the best method for each datatset. InfoMUNIT gives performance equivalent or better than MUNIT. This shows that the disentangled features have also an impact on the quality of the images. Notice that DRIT++ is the best for the first four tasks but totally fails in the last four tasks. This is illustrated by the strange dog images generated by DRIT++ in the Figure 2. On the opposite, CycleGAN gives the best performance for the last four tasks but is bad in the first four tasks and especially in the bag2edge and shoe2edge tasks. On average, InfoMUNIT achieves the best FID value among the four image-to-image translation methods. The good quality of images generated by InfoMUNIT is stable on multiple datasets.

5.2. Image Diversity

Table 2 and Table 3 respectively shows the CIS and LPIPS scores that evaluate the diversity of generated images. CycleGAN is not a multimodal method, it can generate only one output from one input so it does therefore not appear in this table. The LPIPS and CIS scores of InfoMUNIT are clearly superior to the scores of DRIT++ and MUNIT. The only exceptions are for the shoe2edge task where the LPIPS of DRIT++ is higher and for the real2paint task where the LPIPS of MUNIT is higher. In both cases the LPIPS of InfoMUNIT is very close to the best score and still higher than the LPIPS of the third

Table 2: LPIPS distance. Higher value means better performance.

	InfoMUNIT	**MUNIT**	**DRIT++**
edge2bag	**3.00**	2.07	2.13
bag2edge	**2.01**	1.07	1.60
edge2shoe	**2.35**	2.23	1.76
shoe2edge	1.44	1.00	**1.51**
dog2cat	**2.24**	1.97	1.11
cat2dog	**2.65**	2.24	1.09
paint2real	**1.96**	1.91	1.74
real2paint	2.14	**2.26**	1.14
Average	**2.40**	1.88	1.51

Table 3: Conditional Inception Score (CIS). Higher value means better performance.

	InfoMUNIT	**MUNIT**	**DRIT++**
edge2bag	**0.42**	0.29	0.30
bag2edge	**0.35**	0.04	0.22
edge2shoe	**0.26**	0.24	0.22
shoe2edge	**0.24**	0.00	0.12
dog2cat	**0.32**	**0.32**	0.04
cat2dog	**0.30**	0.28	0.03
paint2real	**0.25**	**0.25**	0.11
real2paint	**0.33**	0.30	0.06
Average	**0.31**	0.21	0.14

method. Over all datasets, the scores of InfoMUNIT are significantly better than the other methods. Figure 3 and Figure 4 illustrate the higher diversity of InfoMUNIT compared to MUNIT. This results show that InfoMUNIT generates significantly more diverse outputs than MUNIT and DRIT++.

5.3. Controlling Features

In this subsection, we show the advantage of InfoMUNIT over its predecessor MUNIT in manipulating features. From Figure 3, we can observe that varying values of style code in MUNIT can lead to slight changes like color of the object. With InfoMUNIT, we can significantly manipulate the features of the object. The first disentangled feature controls the size of the bag and the second one control the color from white to black. We also notice that InfoMUNIT is able to propose different textures of the bag.

The performance of InfoMUNIT on the edges→shoes task is illustrated in Figure 4 and Figure 5. While MUNIT can only change some small details of the shoes, we can significantly manipulate the color of the shoes with InfoMUNIT. Varying the first info style code makes the color changed from bright to dark, while varying the second one changes the color from cold to warm. In Figure 4, we can see that the first info style code is also responsible for the style of the shoes. From the left to the right, it turns a sneaker to a pump and makes it darker at the same time. This effect makes sense as pumps are more likely to have dark colors than sneakers.

Please note that the value of each disentangled feature in this test is plotted from -2 to 2 instead of -1 to 1 in the training phase, which means the generator is receiving style code values that it has never seen before. This explains why the images on the border looks a

Figure 3: Manipulating two last digits in the style code of MUNIT and InfoMUNIT on edges→bags task.

Figure 4: Manipulating two last digits in the style code of MUNIT and InfoMUNIT on edges→shoes task.

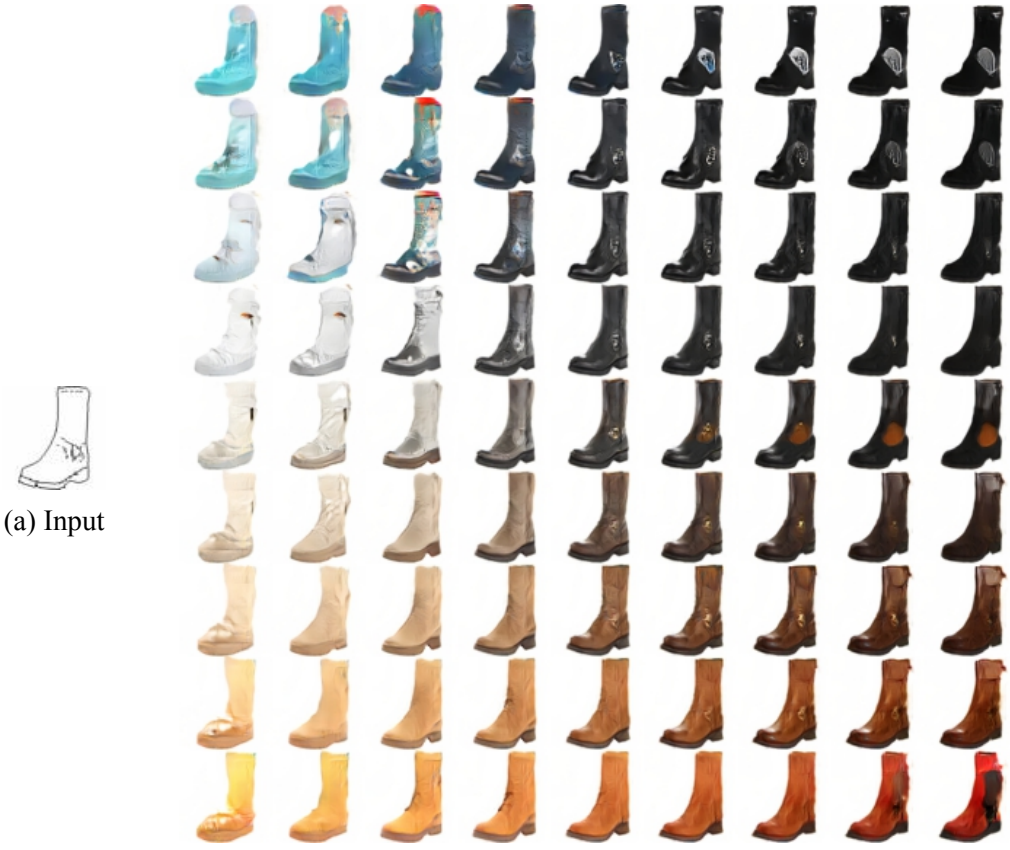
bit extreme.

5.4. The length of information latent code

We perform some experiments to investigate the impact of the length of information latent code i on the generated images in varying this value from 1 to 8. Table 4 shows some of these results on the *edge2shoe* datasets. We see that the FID, CIS and LPIPS weakly vary with the length of i. We conclude from these results that the quality and diversity of the generated images by InfoMUNIT are robust to the length of the information latent code.

6. Conclusion

We proposed an extension of MUNIT called InfoMUNIT which can manipulate features of the translated images. Our method is demonstrated in multiple image-to-image translation tasks. It achieves comparable translated image quality to state-of-the-art approaches and outperforms them in terms of outputs diversity. Moreover, our method improves the control of the user on the generated images, this kind of tool can make the image manipulation method more usable for real life applications.



(a) Input

(b) Outputs generated by **InfoMUNIT**

Figure 5: Combination of the two last digits in the style code of InfoMUNIT on edges→shoes task. From left to right (b), we vary the value of the first information latent code. From top to bottom, we vary the second one.

Table 4: Performance of InfoMUNIT with different lengths of information latent code.

Length of i	**1**	**2**	**4**	**6**	**8**
FID	2.99	**2.81**	3.19	3.11	3.37
CIS	2.59	3.00	3.64	3.61	**3.65**
LPIPS	0.42	0.42	**0.47**	**0.47**	0.46

References

[1] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification," *ACM Transactions on Graphics (ToG)*, vol. 35, no. 4, pp. 1–11, 2016.

[2] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2414–2423, 2016.

[3] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *European conference on computer vision*, pp. 184–199, Springer, 2014.

[4] J. Xie, L. Xu, and E. Chen, "Image denoising and inpainting with deep neural networks," in *Advances in neural information processing systems*, pp. 341–349, 2012.

[5] Z. Murez, S. Kolouri, D. Kriegman, R. Ramamoorthi, and K. Kim, "Image to image translation for domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4500–4509, 2018.

[6] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in neural information processing systems*, pp. 700–708, 2017.

[7] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell, "Cycada: Cycle-consistent adversarial domain adaptation," *arXiv preprint arXiv:1711.03213*, 2017.

[8] H.-Y. Lee, H.-Y. Tseng, J.-B. Huang, M. Singh, and M.-H. Yang, "Diverse image-to-image translation via disentangled representations," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 35–51, 2018.

[9] X. Huang, M.-Y. Liu, S. Belongie, and J. Kautz, "Multimodal unsupervised image-to-image translation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 172–189, 2018.

[10] J. P. Cohen, M. Luck, and S. Honari, "Distribution matching losses can hallucinate features in medical image translation," in *International conference on medical image computing and computer-assisted intervention*, pp. 529–536, Springer, 2018.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[12] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

[13] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8798–8807, 2018.

[14] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2107–2116, 2017.

[15] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3722–3731, 2017.

[16] Y. Taigman, A. Polyak, and L. Wolf, "Unsupervised cross-domain image generation," *arXiv preprint arXiv:1611.02200*, 2016.

[17] S. Benaim and L. Wolf, "One-sided unsupervised domain mapping," in *Advances in neural information processing systems*, pp. 752–762, 2017.

[18] Z. Yi, H. Zhang, P. Tan, and M. Gong, "Dualgan: Unsupervised dual learning for image-to-image translation," in *Proceedings of the IEEE international conference on computer vision*, pp. 2849–2857, 2017.

[19] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2223–2232, 2017.

[20] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1857–1865, JMLR. org, 2017.

[21] A. Almahairi, S. Rajeswar, A. Sordoni, P. Bachman, and A. Courville, "Augmented cyclegan: Learning many-to-many mappings from unpaired data," *arXiv preprint arXiv:1802.10151*, 2018.

[22] H.-Y. Lee, H.-Y. Tseng, Q. Mao, J.-B. Huang, Y.-D. Lu, M. Singh, and M.-H. Yang, "Drit++: Diverse image-to-image translation via disentangled representations," *International Journal of Computer Vision*, pp. 1–16, 2020.

[23] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[24] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103, 2008.

[25] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko, "Semi-supervised learning with ladder networks," in *Advances in neural information processing systems*, pp. 3546–3554, 2015.

[26] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther, "Improving semi-supervised learning with auxiliary deep generative models," in *NIPS Workshop on Advances in Approximate Bayesian Inference*, 2015.

[27] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[28] J. B. Tenenbaum and W. T. Freeman, "Separating style and content with bilinear models," *Neural computation*, vol. 12, no. 6, pp. 1247–1283, 2000.

[29] Z. Zhu, P. Luo, X. Wang, and X. Tang, "Multi-view perceptron: a deep model for learning face identity and view representations," in *Advances in Neural Information Processing Systems*, pp. 217–225, 2014.

[30] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Advances in neural information processing systems*, pp. 3581–3589, 2014.

[31] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[32] D. Barber and F. V. Agakov, "Kernelized infomax clustering," in *Advances in neural information processing systems*, pp. 17–24, 2006.

[33] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum, "Deep convolutional inverse graphics network," in *Advances in neural information processing systems*, pp. 2539–2547, 2015.

[34] G. Desjardins, A. Courville, and Y. Bengio, "Disentangling factors of variation via generative entangling," *arXiv preprint arXiv:1210.5474*, 2012.

[35] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in neural information processing systems*, pp. 2172–2180, 2016.

[36] D. Barber and F. V. Agakov, "The im algorithm: a variational approach to information maximization," in *Advances in neural information processing systems*, p. None, 2003.

[37] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1501–1510, 2017.

[38] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in neural information processing systems*, pp. 2234–2242, 2016.

[39] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in neural information processing systems*, pp. 6626–6637, 2017.

[40] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.