Crossing-Domain Generative Adversarial Networks for Unsupervised Multi-Domain Image-to-Image Translation

Xuewen Yang
Stony Brook University
xuewen.yang@stonybrook.edu

Dongliang Xie*
Beijing University of Posts and Telecommunications
xiedl@bupt.edu.cn

Xin Wang
Stony Brook University
x.wang@stonybrook.edu

ABSTRACT

State-of-the-art techniques in Generative Adversarial Networks (GANs) have shown remarkable success in image-to-image translation from peer domain X to domain Y using paired image data. However, obtaining abundant paired data is a non-trivial and expensive process in the majority of applications. When there is a need to translate images across n domains, if the training is performed between every two domains, the complexity of the training will increase quadratically. Moreover, training with data from two domains only at a time cannot benefit from data of other domains, which prevents the extraction of more useful features and hinders the progress of this research area. In this work, we propose a general framework for unsupervised image-to-image translation across multiple domains, which can translate images from domain X to any a domain without requiring direct training between the two domains involved in image translation. A byproduct of the framework is the reduction of computing time and computing resources, since it needs less time than training the domains in pairs as is done in state-of-the-art works. Our proposed framework consists of a pair of encoders along with a pair of GANs which learns high-level features across different domains to generate diverse and realistic samples from. Our framework shows competing results on many image-to-image tasks compared with state-of-the-art techniques.

CCS CONCEPTS

• **Computing methodologies** → **Image representations**; *Neural networks*; *Unsupervised learning*;

KEYWORDS

GAN, Image-to-image translation, Unsupervised learning, Neural networks

ACM Reference Format:
Xuewen Yang, Dongliang Xie, and Xin Wang. 2018. Crossing-Domain Generative Adversarial Networks for Unsupervised Multi-Domain Image-to-Image Translation. In *2018 ACM Multimedia Conference (MM '18), October 22–26, 2018, Seoul, Republic of Korea.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3240508.3240716

*Dr. Xie is the contact author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MM '18, October 22–26, 2018, Seoul, Republic of Korea
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-5665-7/18/10…$15.00
https://doi.org/10.1145/3240508.3240716

1 INTRODUCTION

In this work, we define multi-domain as multiple datasets or several subsets of one dataset that are applied to complete the same task, but these datasets (or subsets) have different statistical biases. As some examples, images taken at Alps in the summer and in the winter are considered as two different domains, while faces with hair and faces with eyeglasses form another two different domains. Under this domain definition, for faces with black hair and faces with yellow hair, the black hair and yellow hair are two different *attributes* of the same domain. In *multi-domain learning*, each sample $\boldsymbol{x}$ is drawn from a domain d specific distribution $\boldsymbol{x} \sim p_d(\boldsymbol{x})$ and has a label $y \in \{0, 1\}$, with $y = 1$ signifying $\boldsymbol{x}$ from domain d, $y = 0$ signifying $\boldsymbol{x}$ not from domain d.

Image-to-image translation is the task of learning to map images from one domain to another, e.g., mapping grayscale images to color images [2], mapping images of low resolution to images of high resolution [12], changing the seasons of scenery images [23], and reconstructing photos from edge maps [8]. The most significant improvement in this research field came with the application of Generative Adversarial Networks (GANs) [3, 14].

The image-to-image translation can be performed in supervised [8] or unsupervised way [23], with the unsupervised one becoming more popular since it does not need to collect ground-truth pairs of samples. Despite the quick progress of research on image-to-image translation, state-of-the-art results for unsupervised translation are still not satisfactory. In addition, existing research generally focuses on image-to-image translation between two domains, which is limited by two drawbacks. First, the translation task is specific to two domains, and the model has to be retrained when there is a need to perform image translation between another pair of similar domains. Second, it can not benefit from the features of multiple domains to improve the training quality. We take the most representative work in this research field CycleGAN [23] as an example to illustrate the first limitation. The translation between two image domains X and Y is achieved with two generators, $G_{X\to Y}$ and $G_{Y\to X}$. However, this model is inefficient in completing the task of multi-domain image translation. To derive mappings across all n domains, it has to train $n(n-1)$ generators, as shown in Fig. 1a.

To enable more efficient multi-domain image translation with unsupervised learning where image pairing across domains is not predefined, we propose Crossing-Domain GAN (CD-GAN), which is a multi-domain encoding generative adversarial network that consists of a pair of encoders and a pair of generative adversarial networks (GANs). We would like the encoders to efficiently encode

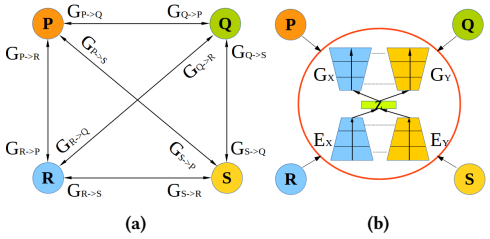

Figure 1: Image-to-image translation of 4 domains. (a) CycleGAN needs 4×3 generators. (b) Our model only needs 2 encoder-generator pairs. In every iteration, we randomly pick two domains, and sample two batches of training data from the domains to train the model. The two encoders first encode domain information into a latent code z using two encoders E_X and E_Y and then generate two samples of the two domains using the generators G_X and G_Y.

the information of all domains to form a high-level feature space with an encoding process, then images of different domains will be translated by decoding the high-level features with a decoding process. CD-GAN achieves this goal with the integrated use of three techniques. First, the two encoders are constrained by a *weight sharing* scheme, where the two encoders (or the two generators) share the same weights at both the highest-level layers and the lowest-level layers. This ensures that the two encoders can encode common high-level semantics as well as low-level details to obtain the feature space, based on which generators can decode the high-level semantics and low-level details correctly to generate images of different domains. Second, we use a selected or existing label to guide the generator to generate images of a corresponding domain from the high-level features learnt. Third, we propose an efficient training algorithm that jointly train the model across domains by randomly selecting two domains to train at each iteration.

Different from [14] where only weights at high-level layers of generators are shared, in CD-GAN, we propose the concurrent sharing of the lowest-level and the highest-level layers at both the encoders and the generators to improve the quality of image translation between *any two* domains. The sharing of highest layers between two encoders helps to enable more flexible cross-domain image translation, while the sharing of the lowest layers across domains helps improve the training quality by taking advantage of the transferring learning across domains.

The contributions of our work are as follows:

- We propose CD-GAN that learns mappings across multiple domains using only two encoder-generator pairs.
- We propose the concurrent use of weight-sharing at highest-level and lowest-level layers of both encoders and generators to ensure that CD-GAN generates images with sufficient useful high-level semantics and low-level details across all domains.
- We leverage domain labels to make a conditional GAN training that greatly improves the performance of the model.
- We introduce a cross-domain training algorithm that efficiently and sufficiently trains the model by randomly taking samples from two of domains at a time. CD-GAN can fully exploit data from all domains to improve the training quality for each individual domain.

Our experiment results demonstrate that when trained on more than two domains, our method achieves the same quality of image translation between any two domains as compared to directly training for translation between the pair. However, our model is established with much less training time and can generate better quality images for a given amount of time. We also show how CD-GAN can be successfully applied to a variety of unsupervised multi-domain image-to-image translation problems.

The remainder of this paper is organized as follows. Section 2 reviews the relevant research for image-to-image translation problems. Section 3 describes our model and training method in details. Section 4 presents our evaluation metrics, experimental methodology, and the evaluation results of the model's accuracy and efficiency on different datasets. Finally, we discuss some limitations of our work and conclude our work in Section 5.

2 RELATED WORK

2.1 Generative Adversarial Networks (GANs)

GANs [3] were introduced to model a data distribution using independent latent variables. Let $\boldsymbol{x} \sim p(\boldsymbol{x})$ be a random variable representing the observed data and $\boldsymbol{z} \sim p(z)$ be a latent variable. The observed variable is assumed to be generated by the latent variable, i.e., $\boldsymbol{x} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}|z)$, where $p_{\boldsymbol{\theta}}(\boldsymbol{x}|z)$ can be explicitly represented by a generator in GANs. GANs are built on top of neural networks, and can be trained with gradient descent based algorithms.

The GAN model is composed of a discriminator $D_{\boldsymbol{\phi}}$, along with the generator $G_{\boldsymbol{\theta}}$. The training involves a min-max game between the two networks. The discriminator $D_{\boldsymbol{\phi}}$ is trained to differentiate 'fake' samples generated from the generator $G_{\boldsymbol{\theta}}$ from the 'real' samples from the true data distribution $p(\boldsymbol{x})$. The generator is trained to synthesize samples that can fool the discriminator by mistaking the generated samples for genuine ones. They both can be implemented using neural networks.

At the training phase, the discriminator parameters $\boldsymbol{\phi}$ are firstly updated, followed by the update of the generator parameters $\boldsymbol{\theta}$. The objective function is given by:

$$\min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D_{\boldsymbol{\phi}}(x)] + \mathbb{E}_{\boldsymbol{z}\sim p(z)}[\log(1 - D_{\boldsymbol{\phi}}(G_{\boldsymbol{\theta}}(z)))] \quad (1)$$

The samples can be generated by sampling $\boldsymbol{z} \sim p(z)$, then $\hat{\boldsymbol{x}} = G_{\boldsymbol{\theta}}(z)$, where $p(z)$ is a prior distribution, for example, a multivariate Gaussian.

2.2 Image-To-Image Translation

Image-to-image translation problem is a kind of image generation task that given an input image $\boldsymbol{x}$ of domain X, the model maps it into a corresponding output image $\boldsymbol{y}$ of another domain Y. It learns a mapping between two domains given sufficient training data [8]. Early works on image-to-image translation mainly focused on tasks where the training data of domain X are similar to the data of domain Y [5, 15], and the results were often unrealistic and not diverse.

In recent years, deep generative models have shown increasing capability of synthesizing diverse, realistic images that capture both fine-grained details and global coherence of natural images [4, 11, 17]. With Generative Adversarial Networks (GANs) [8, 9, 23], recent studies have already taken significant steps in image-to-image translation. In [8], the authors use a conditional GAN on different image-to-image translation tasks, such as synthesizing photos from label maps and reconstructing objects from edge maps. However, this method requires input-output image pairs for training, which is in general not available in image-to-image translation problems. For situations where such training pairs are not given, in [23], the authors proposed CycleGAN to tackle unsupervised image-to-image translation. With a pair of Generators G and F, the model not only learns a mapping $G : X \rightarrow Y$ using an adversarial loss, but constrains this mapping with an inverse mapping $F : Y \rightarrow X$. It also introduces a cycle consistency loss to enforce $F(G(X)) \approx X$, and vice versa. In settings where paired training data are not available, the authors showed promising qualitative results. The authors in [9] and [21] use similar idea to solve the unsupervised image-to-image translation tasks.

These approaches only tackle the problems of translating images between two domains, and have two major drawbacks. First, when applied to n domains, these approaches need $n(n-1)$ generators to complete the task, which is computationally inefficient. To train all models, it would either require a significant amount of time to complete if the training is performed on one GPU, or it will require a lot of hardware and computing resources if training is run over multiple GPUs. Second, as each model is trained with only two datasets, the training cannot benefit from the data of other domains.

Our work is inspired by *multimodal learning* [16], which shows that data features can be better extracted using one modality if multiple modalities are present at feature learning time. The intuition of our method is that if we can encode the information of different domains together and generate a high-level feature space, it would be possible to decode the high-level features to build images of different domains. In this work, rather than generating images from random noise, we incorporate an encoding process into a GAN model. The image-to-image translation can be achieved by first encoding real images into high-level features, and then generating images of different domains using the high-level features through a decoding process. The encoding process and the decoding process are constrained by a weight-sharing technique that both the highest layer and the lowest layer are shared across the two encoders as well as the two generators. Sharing the high-level layers makes sure that the generated images are semantically correct, while sharing the low-level layers ensures that important low-level features be captured and transferred between domains. Our model is trained end-to-end using data from all n domains.

3 CROSS-DOMAIN GENERATIVE ADVERSARIAL NETWORK

To conduct unsupervised multi-domain image-to-image translation, a direct approach is to train a CycleGAN for every two domains. While this approach is straightforward, it is inefficient as the number of training models increases quadratically with the number of domains. If we have n domains, we have to train $n(n-1)$ generators, as shown in Fig. 1a. In addition, since each model only utilizes data from two domains to train, the training cannot benefit from the useful features of other domains.

To tackle these two problems, a possible way is to encode useful information of all domains into common high level features, and then to decode the high-level features into images of different domains. Inspired by work [16] from *mutimodal learning*, where training data are from multiple modalities, we propose to build a multi-domain image translation model that can encode information of multiple domains into a set Z of high-level features, and then use features in Z to reconstruct data of different domains or to do image-to-image translation. The overview of the model applied to 4 domains is shown in Fig. 1b, where only one model is used.

In this section, we first present our proposed CD-GAN model, then describe how image translation can be performed across domains, and finally introduce our cross-domain training method.

3.1 CD-GAN with Double Layer Sharing

We first describe how to apply our model to multi-domain image-to-image translation in general then illustrate it using two domains as an example. As shown in Fig. 2a, our proposed CD-GAN model consists of a pair of encoders followed by a pair of GANs. Taking domain X and Y as an example, the two encoders E_X and E_Y encode domain information from X and Y into a set of high-level features contained in a set Z. Then from a high-level feature z in space Z, we can generate images that fall into domain X or Y. The generated images are then evaluated by the corresponding discriminators D_X and D_Y to see whether they look real and cannot be identified as generated ones. For example, following the red arrows, the input image $\boldsymbol{x}$ is first encoded into a high-level feature $\boldsymbol{z}_x$, then $\boldsymbol{z}_x$ is decoded to generate the image $\hat{\boldsymbol{y}}$. The image $\hat{\boldsymbol{y}}$ is the translated image in domain Y. Similar processes exist for image $\boldsymbol{y}$.

Our model is also constrained by a reconstruction process shown in Fig. 2b. For example, following the red arrows, the input image $\boldsymbol{x}$ is first encoded into a high-level feature $\boldsymbol{z}_x$, then $\boldsymbol{z}_x$ is decoded to generate the image $\boldsymbol{x}\prime$, which is a reconstruction of the input image. Similar processes exist for image $\boldsymbol{y}$.

Learning with deep neural networks involves hierarchical feature representation. In order to support flexible cross-domain image translation and also to improve the training quality, we propose the use of *double-layer sharing* where the highest-level and the lowest-level layers of the two encoders share the same weights and so does the two generators. By enforcing the layers that decode high-level features in GANs to share weights, the images generated by different generators can have some common high-level semantics. The layers that decode low-level details then map the high-level features to images in individual domains.

Sharing weights of low-level layers has the benefit of transferring low-level features of one domain to the other, thus making the image-to-image translation more close to real images in the respective domains. Besides, sharing layers reduces the complexity of the model, making it more resistant to the over-fitting problem.

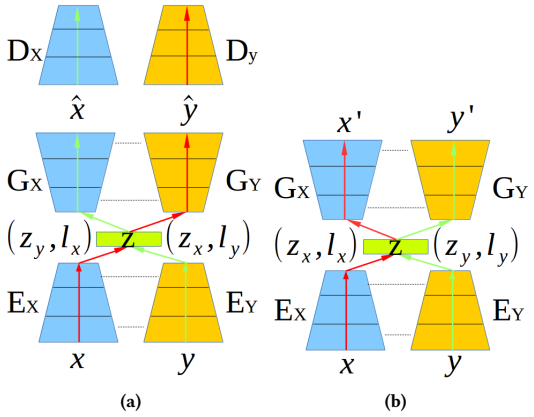

Figure 2: The proposed CD-GAN model. (a) The translation mappings: the input image $\boldsymbol{x}$ is first encoded as a latent code z_x through $E_X(\boldsymbol{x})$, which is then decoded into a translated image $\hat{\boldsymbol{y}}$ through $G_Y(z_x, \boldsymbol{l}_y)$. The process is identified with red arrows. There is a similar process for the image $\boldsymbol{y}$. D_X and D_Y are adversarial discriminators for the respective domains to evaluate whether the translated images are realistic. (b) The reconstruction mappings: the input image $\boldsymbol{x}$ is first encoded as a latent code z_x through $E_X(\boldsymbol{x})$, which is then decoded into a reconstructed image $\boldsymbol{x}\prime$ through the generator $G_X(z_x, \boldsymbol{l}_x)$. The process is signified in red arrows. A similar process exists for image $\boldsymbol{y}$. Note: the dashed lines indicate that the two layers share the same parameters.

3.2 Conditional Image Generation

In state-of-the-art techniques, like CycleGAN, each domain is described by a specific generator, thus there is no need to inform the generator which domain the input image is generated to. However, in our model, multiple domains share two generators. For an input image, we have to include an auxiliary variable to guide the generation of image for a specific domain. The only information we have is the domain labels. To make use of this information, the inputs of the model are not images $\boldsymbol{x}$, $\boldsymbol{y}$, but image pairs $(\boldsymbol{x}, \boldsymbol{l}_y)$ and $(\boldsymbol{y}, \boldsymbol{l}_x)$ where the labels $\boldsymbol{l}_y$ and $\boldsymbol{l}_x$ inform the generators which domains to generate an image for. These image pairs are not the same as the image pairs of supervised image-to-image generation tasks, which are $(\boldsymbol{x}, \boldsymbol{y})$. Thus no matter which domain images are the input, the model can always generate images of a domain of interest.

We denote the data distributions as $\boldsymbol{x} \sim p(\boldsymbol{x})$ and $\boldsymbol{y} \sim p(\boldsymbol{y})$. As illustrated in Fig. 2, our model includes four mappings, two translation mappings $X \rightarrow Z \rightarrow Y$, $Y \rightarrow Z \rightarrow X$ and two reconstruction mappings $X \rightarrow Z \rightarrow X$, $Y \rightarrow Z \rightarrow Y$. The translation mappings constrain the model by a GAN loss, while the reconstruction mappings constrain the model by a reconstruction loss. To further constrain the auxiliary variable, we introduce a classification loss by applying a classifier to classify the real or generated images into different domains. The intuition is that if images are generated with the guidance of the auxiliary variable, then it can be correctly classified into the domain specified by the auxiliary variable. Next, we introduce these model losses in more details as follows.

GAN Losses Following the translation mapping $X \rightarrow Z \rightarrow Y$, we can translate image $\boldsymbol{x}$ from domain X to $\hat{\boldsymbol{y}}$ of domain Y using $z_x = E_X(\boldsymbol{x})$, $\hat{\boldsymbol{y}} = G_Y(z_x, \boldsymbol{l}_y)$. With the purpose of improving the quality of the generated samples, we apply adversarial loss. We express the objective as:

$$\begin{aligned}\mathcal{L}_{GAN_Y} &= \mathbb{E}_{\boldsymbol{y}\sim p(\boldsymbol{y})} \log(D_Y(\boldsymbol{y})) \\ &+ \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})} \log(1 - D_Y(G_Y(E_X(\boldsymbol{x}), \boldsymbol{l}_y)))\end{aligned} \tag{2}$$

where G_Y tries to generate images $\hat{\boldsymbol{y}} = G_Y(z_x, \boldsymbol{l}_y)$ that look similar to images from domain Y, while D_Y aims to distinguish between translated samples $\hat{\boldsymbol{y}}$ and real samples $\boldsymbol{y}$. The similar adversarial loss for $Y \rightarrow Z \rightarrow X$ is

$$\begin{aligned}\mathcal{L}_{GAN_X} &= \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})} \log(D_X(\boldsymbol{x})) \\ &+ \mathbb{E}_{\boldsymbol{y}\sim p(\boldsymbol{y})} \log(1 - D_X(G_X(E_Y(\boldsymbol{y}), \boldsymbol{l}_x)))\end{aligned} \tag{3}$$

The total GAN loss is:

$$\mathcal{L}_{GAN} = \mathcal{L}_{GAN_X} + \mathcal{L}_{GAN_Y} \tag{4}$$

Reconstruction Loss The reconstruction mappings $X \rightarrow Z \rightarrow X$, $Y \rightarrow Z \rightarrow Y$ encourage the model to encode enough information to the high-level feature space Z from each domain. The input can then be reconstructed by the generators. The reconstruction process of domain X is $z_x = E_X(\boldsymbol{x})$, $\boldsymbol{x}\prime = G_X(z_x, \boldsymbol{l}_x)$. Similar reconstruction process exists for domain Y. With l_2 distance as the loss function, the reconstruction loss is:

$$\begin{aligned}\mathcal{L}_{rec} &= \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}(||\boldsymbol{x} - G_X(E_X(\boldsymbol{x}), \boldsymbol{l}_x)||_2) \\ &+ \mathbb{E}_{\boldsymbol{y}\sim p(\boldsymbol{y})}(||\boldsymbol{y} - G_Y(E_Y(\boldsymbol{y}), \boldsymbol{l}_y)||_2)\end{aligned} \tag{5}$$

Latent Consistency Loss With only the above losses, the encoding part is not well constrained. We constrain the encoding part using a latent consistency loss. Although $\boldsymbol{x}$ is translated to $\hat{\boldsymbol{y}}$, which is in domain Y, $\hat{\boldsymbol{y}}$ is still semantically similar to $\boldsymbol{x}$. Thus, in the latent space Z, the high-level feature of $\boldsymbol{x}$ should be close to that of $\hat{\boldsymbol{y}}$. Similarly, the high-level feature of $\boldsymbol{y}$ in domain Y should be close to the high-level feature of $\hat{\boldsymbol{x}}$ in domain X. The latent consistency loss is the following:

$$\begin{aligned}\mathcal{L}_{lcl} &= \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}(||E_X(\boldsymbol{x}) - E_Y(G_Y(E_X(\boldsymbol{x}), \boldsymbol{l}_y))||) \\ &+ \mathbb{E}_{\boldsymbol{y}\sim p(\boldsymbol{y})}(||E_Y(\boldsymbol{y}) - E_X(G_X(E_Y(\boldsymbol{y}), \boldsymbol{l}_x))||)\end{aligned} \tag{6}$$

Classification Loss We consider n domains as n categories in the classification problems. We use a network C, which is an auxiliary classifier, on top of the general discriminator D to measure whether a sample (real or generated) belongs to a specific fine-grained category. The output of the classifier C represents the posterior probability $P(c|\boldsymbol{x})$. Specifically, there are four classification losses, i.e., for real data $\boldsymbol{x}$, $\boldsymbol{y}$, and generated data $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$. For image-label pairs $(\boldsymbol{x}, \boldsymbol{l}_x)$ and $(\boldsymbol{y}, \boldsymbol{l}_y)$ with $\boldsymbol{l}_x \sim p(\boldsymbol{l}_x)$ and $\boldsymbol{l}_y \sim p(\boldsymbol{l}_y)$ our goal is to translate $\boldsymbol{x}$ to $\hat{\boldsymbol{y}}$ with label $\boldsymbol{l}_y$, and to translate $\boldsymbol{y}$ to $\hat{\boldsymbol{x}}$ with label $\boldsymbol{l}_x$. The four classification losses are:

$$\begin{aligned}\mathcal{L}_c &= -\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}), \boldsymbol{l}_x\sim p(\boldsymbol{l}_x)}[\log P(\boldsymbol{l}_x|\boldsymbol{x})] \\ &= -\mathbb{E}_{\boldsymbol{y}\sim p(\boldsymbol{y}), \boldsymbol{l}_y\sim p(\boldsymbol{l}_y)}[\log P(\boldsymbol{l}_y|\boldsymbol{y})] \\ &= -\mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x}), \boldsymbol{l}_y\sim p(\boldsymbol{l}_y)}[\log P(\boldsymbol{l}_y|G_Y(E_X(\boldsymbol{x}), \boldsymbol{l}_y))] \\ &= -\mathbb{E}_{\boldsymbol{y}\sim p(\boldsymbol{y}), \boldsymbol{l}_x\sim p(\boldsymbol{l}_x)}[\log P(\boldsymbol{l}_x|G_X(E_Y(\boldsymbol{y}), \boldsymbol{l}_x))]\end{aligned} \tag{7}$$

This loss can be used to optimize discriminators D_X, D_Y, generators G_X, G_Y, and encoders E_X, E_Y.

Cycle Consistency Loss Although the minimization of GAN losses ensures that $G_Y(E_X(\boldsymbol{x}), \boldsymbol{l}_y)$ produce a sample $\hat{\boldsymbol{y}}$ similar to samples drawn from Y, the model still can be unstable and prone to failure. To tackle this problem, we further constrain our model with a cycle-consistency loss [23]. To achieve this goal, we want mapping from domain X to domain Y and then back to domain X to reproduce the original sample, i.e., $G_X(E_Y(G_Y(E_X(\boldsymbol{x}), \boldsymbol{l}_y)), \boldsymbol{l}_x) \approx \boldsymbol{x}$ and $G_Y(E_X(G_X(E_Y(\boldsymbol{y}), \boldsymbol{l}_x)), \boldsymbol{l}_y) \approx \boldsymbol{y}$. Thus, the cycle-consistency loss is:

$$\begin{aligned}\mathcal{L}_{cyc} &= \mathbb{E}_{\boldsymbol{x}\sim p(\boldsymbol{x})}[||G_X(E_Y(G_Y(E_X(\boldsymbol{x}), \boldsymbol{l}_y)), \boldsymbol{l}_x) - \boldsymbol{x}||] \\ &+ \mathbb{E}_{\boldsymbol{y}\sim p(\boldsymbol{y})}[||G_Y(E_X(G_X(E_Y(\boldsymbol{y}), \boldsymbol{l}_x)), \boldsymbol{l}_y) - \boldsymbol{y}||]\end{aligned} \tag{8}$$

Final Objective of CD-GAN To sum up, the goal of our approach is to minimize the following objective:

$$\mathcal{L}(E, G, D) = \mathcal{L}_{GAN} + \alpha_0\mathcal{L}_{rec} + \alpha_1\mathcal{L}_{lcl} + \alpha_2\mathcal{L}_c + \alpha_3\mathcal{L}_{cyc} \tag{9}$$

where E, G, and D signify encoders E_X, E_Y, generators G_X, G_Y, and discriminators D_X, D_Y, and α_0, α_1, α_2, α_3 control the relative importance of the losses. Same as solving a regular GAN problem, training the model involves the solving of a min-max problem, where E_X,E_Y, G_X, and G_Y aim to minimize the objective, while D_X and D_Y aim to maximize it.

$$E^*, G^* = arg\min_{E,G}\max_{D} \mathcal{L}(E, G, D) \tag{10}$$

3.3 Cross-Domain Training

Our proposed model has two encoder-generator pairs, but we have data from *n* domains. To train the model using samples of all domains equally, we introduce a cross-domain training algorithm. As shown in Fig. 1b, there are 4 domains. At each iteration, we randomly select two domains R and S, and feed training data of these two domains into the model. At the next iteration, we might take another two domains P and Q, and perform the same training. We train the model using all data samples of 4 domains at every epoch for several iterations. The training algorithm is shown in Algorithm 1. *Cross-domain training* ensures the model to learn a generic feature representation of all domains by training the model equally on independent domains.

Algorithm 1 Joint domain training on CD-GAN using mini-batch stochastic gradient descent

Require: Training samples from n domains
Initialize $\boldsymbol{\theta}_E^X$, $\boldsymbol{\theta}_E^Y$,$\boldsymbol{\theta}_G^X$, $\boldsymbol{\theta}_G^Y$,$\boldsymbol{\theta}_D^X$, and $\boldsymbol{\theta}_D^Y$ with the shared network connection weights set to the same values.
while Training loss has not converged **do**
Randomly draw two domains X and Y from n domains
Randomly draw N samples from the two domains, $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots \boldsymbol{x}_N; \boldsymbol{y}_1, \boldsymbol{y}_2, \ldots \boldsymbol{y}_N\}$
Get the domain labels of the samples from the two domains, $\{\boldsymbol{l}_X^i, \boldsymbol{l}_Y^i\}_{i=1}^N$
(1) Update D_X, D_Y with fixed G_X, G_Y, E_X, E_Y
Generate fake samples using the real ones

$$\hat{\boldsymbol{x}}_i = G_X(E_Y(\boldsymbol{y}_i), \boldsymbol{l}_x^i), \hat{\boldsymbol{y}}_i = G_Y(E_X(\boldsymbol{x}_i), \boldsymbol{l}_y^i), i = 1 \ldots N$$

Update $\boldsymbol{\theta}_D = (\boldsymbol{\theta}_D^X, \boldsymbol{\theta}_D^Y)$ according to the following gradients

$$\nabla_{\theta_D}\Big[\frac{1}{N}\sum_{i=1}^{N}\Big[-\log D_X(\boldsymbol{x}_i) - \log(1 - D_X(\hat{\boldsymbol{x}}_i)) - \log D_Y(\boldsymbol{y}_i) - \log(1 - D_Y(\hat{\boldsymbol{y}}_i)) + \alpha_2\big[\log P(\boldsymbol{l}_x|\boldsymbol{x}_i) + \log P(\boldsymbol{l}_y|\boldsymbol{y}_i)\big]\Big]\Big]$$

(2) Update E_X, E_Y, G_X, G_Y with fixed D_X, D_Y
Update $\boldsymbol{\theta}_{E,G} = (\boldsymbol{\theta}_E^X, \boldsymbol{\theta}_E^Y, \boldsymbol{\theta}_G^X, \boldsymbol{\theta}_G^Y)$ according to the following gradients

$$\begin{aligned}&\nabla_{\theta_{E,G}}\Big[\frac{1}{N}\sum_{i=1}^{N}\Big[\log(1 - D_X(\hat{\boldsymbol{x}}_i)) + \log(1 - D_Y(\hat{\boldsymbol{y}}_i)) \\ &+ ||\boldsymbol{x}_i - G_X(E_X(\boldsymbol{x}_i), \boldsymbol{l}_x^i)||_2 + ||\boldsymbol{y}_i - G_Y(E_Y(\boldsymbol{y}_i), \boldsymbol{l}_y^i)||_2 \\ &+ ||E_X(\boldsymbol{x}_i) - E_Y(\hat{\boldsymbol{y}}_i)|| + ||E_Y(\boldsymbol{y}_i) - E_X(\hat{\boldsymbol{x}}_i)|| \\ &+ \log P(\boldsymbol{l}_x|\hat{\boldsymbol{x}}_i) + \log P(\boldsymbol{l}_y|\hat{\boldsymbol{y}}_i) \\ &+ \alpha\big[||\boldsymbol{x}_i - G_X(E_Y(\hat{\boldsymbol{y}}_i), \boldsymbol{l}_x^i)|| + ||\boldsymbol{y}_i - G_Y(E_X(\hat{\boldsymbol{x}}_i), \boldsymbol{l}_y^i)||\big]\Big]\Big]\end{aligned}$$

end while

4 EXPERIMENT

In this section, we conduct experiments over three datasets to compare our proposed model with reference models in terms of image translation quality and efficiency.

4.1 Datasets

To evaluate the scalability and effectiveness of our model, we test it on a variety of multi-domain image-to-image translation tasks using the following datasets:

Alps Seasons dataset [1] is collected from images on Flickr. The images are categorized into four seasons based on the provided timestamp of when it was taken. It consists of four categories: *Spring*, *Summer*, *Fall*, and *Winter*. The training data consists of 6053 images of four seasons, while the test data consists of 400 images.

Painters dataset [23] includes painting images of four artists *Monet*, *Van Gogh*, *Cezanne*, and *Ukiyo-e*. We use 2851 images as the training set, and 200 images as the test set.

We run all the experiments on a Ubuntu system using an Intel i7-6850K, along with a single NVIDIA GTX 1080Ti GPU.

4.2 Reference Models

We compare the performance of our proposed CD-GAN with that of two reference models:

CycleGAN [23] This method trains two generators $G : X \rightarrow Y$ and $F : Y \rightarrow X$ in parallel. It not only applies a standard GAN loss respectively for X and Y, but applies forward and backward cycle consistency losses which ensure that an image $\boldsymbol{x}$ from domain X be translated to an image of domain Y, which can then be translated back to the domain X, and vice versa.

DualGAN [21] This method uses a dual-GAN mechanism, which consists of a primal GAN and a dual GAN. The primal GAN learns

to translate images from domain X to domain Y, while the dual-GAN learns to invert the task. Images from either domain can be translated and then reconstructed. Thus a reconstruction loss can be used to train the model.

UNIT [13] This method consists of two VAE-GANs with a fully shared latent space. To complete the task of image-to-image translation between n domains, it needs to be trained $\frac{n\times(n-1)}{2}$ times.

DB [7] This method addresses the multi-domain image-to-image translation problem by introducing n domain-specific encoders/decoders to learn an universal shared-latent space.

4.3 Evaluation Metrics

There is a challenge to evaluate the quality of synthesized images [18]. Recent works have tried using pre-trained semantic classifiers to measure the realism and discriminability of the generated images. The idea is that if the generated images look to be more close to real ones, classifiers trained on the real images will be able to classify the synthesized images correctly as well. Following [8, 20, 22], to evaluate the performance of the models in classifying generated images quantitatively, we apply the metric *classification accuracy*. For each experiment, we generate enough number of images of different domains, then we use a pre-trained classifier which is trained on the training dataset to classify them to different domains and calculate the classification accuracy.

4.4 Network Architecture and Implementation

The design of the architecture is always a difficult task [17]. To get a proper model architecture, we adopt the architecture of the discriminator from [8] which has been proven to be proficient in most image-to-image generation tasks. It has 6 convolutional layers. We keep the discriminator architecture fixed and vary the architectures of the encoders and generators. Following the design of the architectures of the generators in [8], we use two types of layers, the regular convolutional layers and the basic residual blocks [6]. Since the encoding process is the inverse of the decoding process, we use the same layers for them but put the layers in the inverse orders. The only difference is the first layer of the encoder and the last layer of the generator. We apply 64 channels (corresponding to different filters) for the first layer of the encoders, but 3 channels for the last layer of the generators since the output images have only 3 RGB channels. We gradually change the number of convolutional layers and the number of residual blocks until we get a satisfying architecture. We don't apply *weight sharing* initially. The performance of different architectures is evaluated on the *Painters* dataset and shown in Fig. 3. We can see that when the model has 3 regular convolutional layers and 4 basic residual blocks, the model has the best performance. We keep this architecture fixed for other datasets.

We then vary the number of weight-sharing layers in the encoders and the generators. We change the number of weight-sharing layers from 1 to 4. Sharing 1 layer means sharing the highest layer and the lowest layers in the encoder pair. Sharing 2 layers means sharing the highest and lowest two layers. The same sharing method applies for the generator pair (not including the output layer). The results are shown in table 1. We found that sharing 1 layer is enough to have a good performance.

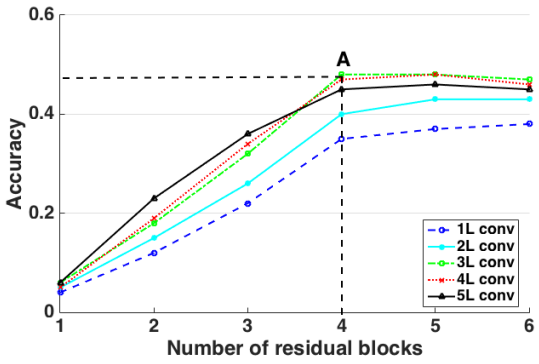

Figure 3: The accuracy on varying number of residual blocks and number of convolutional layers.

Table 1: Classification accuracy on number of shared layers in encoders and generators.

# of shared layers	acc. % (Painters)	acc. % (Alps Seasons)
0	49.75	29.95
1	52.54	33.78
2	52.81	33.54
3	51.13	33.06

Table 2: Network architecture for the multi-modal unsupervised image-to-image translation experiments. $cxkysz$ **denote a Convolution-InstanceNorm-ReLU layer with** x **filters, kernel size** y**, and stride** z**.** Rm **denotes a residual block that contains two** 3×3 **convolutional layers with the same number of filters on both layers.** un **denotes a** 3×3 **fractional-strided-Convolution-InstanceNorm-ReLU layer with** n **filters, and stride** $\frac{1}{2}$**.** n_d **denotes number of domains.** Y **and** N **denote whether the layer is shared or not.**

Layer	Encoders	Generators	Discriminators
1	$c64k7s1(Y)$	$R256(Y)$	$c64k3s2(N)$
2	$c128k3s2(N)$	$R256(N)$	$c128k3s2(N)$
3	$c256k3s2(N)$	$R256(N)$	$c256k3s2(N)$
4	$R256(N)$	$R256(N)$	$c512k3s2(N)$
5	$R256(N)$	$u256(N)$	$c1024k3s2(N)$
6	$R256(N)$	$u128(Y)$	$c(1+n_d)k2s1(N)$
7	$R256(Y)$	$u3(N)$	

In summary, for the testbed evaluation, we use two encoders each consisting of 3 convolutional layers and 4 basic residual blocks. The generators are composed with 4 basic residual blocks and 3 fractional-strided convolutional layers. The discriminators consist of a stack of 6 convolutional layers. We use LeakyReLU for nonlinearity. The two encoders share the same parameters on their layers 1 and 7, while the two generators share the same parameters on layers 1 and 6, which is the lowest-level layer before the output layer. The details of the networks are given in table 2. We evaluate various network architectures in the evaluation parts. We fix the network architecture as in Table 2.

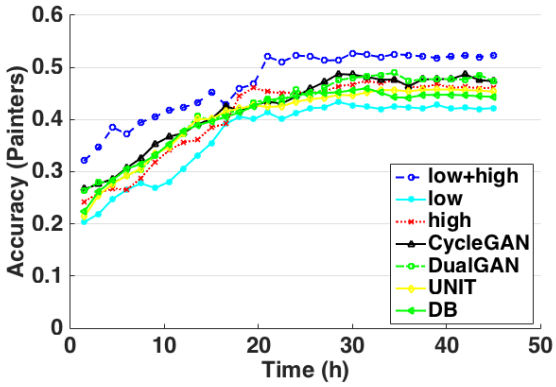

Figure 4: The classification accuracy on *Painters* dataset. The 7 models are the proposed model with *the lowest and the highest layer sharing, the lowest layer sharing only, the highest layer sharing only,* CycleGAN, DualGAN, UNIT, and DB.

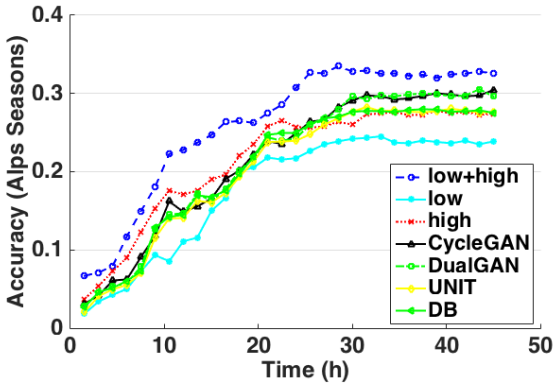

Figure 5: The classification accuracy on *Alps Seasons* dataset. The 7 models are the prosed model with *lowest and highest level layers sharing, lowest level layers sharing, highest level layers sharing,* CycleGAN, DualGAN, UNIT, and DB.

We use ADAM [10] for training, where the training rate is set to 0.0001 and momentums are set to 0.5 and 0.999. Each mini-batch consists of one image from domain X and one image from domain Y. Our model has several hyper-parameters. The default values are $\alpha_0 = 10$, $\alpha_1 = 0.1$, $\alpha_2 = 0.1$, and $\alpha_3 = 10$. The hyper-parameters of the baselines are set to the suggested values by the authors.

4.5 Quantitative Results

We evaluate our model on different datasets and compare it with baseline models.

4.5.1 Comparison on Painters Dataset. To compare the proposed model with baseline models *Painters* dataset, we first train the state-of-the-art VGG-11 model [19] on training data and get a classifier of accuracy 94.5%. We then score synthesized images by the classification accuracy against the domain labels these photos were synthesized from. We generate around 4000 images for every 5 hours and the classification accuracies are shown in Fig. 4.

We can see that our model achieves the highest classification accuracy of 52.5% when using both the highest layer and lowest layer sharing, with the training time less than the other reference models in reaching the peak.

4.5.2 Comparison on Alps Seasons Dataset. We train VGG-11 model on training data of *Alps Seasons* dataset and get a classifier of accuracy 85.5% trained on the training data. We then classify the generated images by our model and the classification accuracies are shown in Fig. 5.

Similar to Fig. 4, our model achieves the highest classification accuracy of 33.8% with the training time less than the baseline models in reaching the peak.

4.6 Analysis of the loss function

We compare the ablations of our full loss. As GAN loss and cycle consistency loss are critical for the training of unsupervised image-to-image translation, we keep these two losses as the baseline model and do the ablation experiments to see the importance of other losses.

Table 3: Ablation study: classification accuracy of *Painters* and *Alps Seasons* datasets for different losses. The following abbreviations are used: R:reconstruction loss, LCL: latent consistency loss, C: classification loss.

Loss	acc.% (Painters)	acc. % (Alps Seasons)
Baseline	35.23	20.81
Baseline + R	36.86	21.59
Baseline + LCL	44.42	25.05
Baseline + C	43.63	24.01
Baseline + R + LCL	45.79	27.19
Baseline + R + C	44.82	26.63
Baseline + LCL + C	50.74	32.51
Baseline + R + LCL + C	52.54	33.78

As shown in Tabel 3, the reconstruction loss R is least important with accuracy improvement of about 4.6% on *Painters* dataset and 3.7% on *Alps Seasons* dataset. The latent consistency loss LCL brings the model an accuracy improvement of 26.1% on *Painters* dataset and 20.4% on *Alps Seasons* dataset. The accuracy is improved by 23.8% on *Painters* dataset and 15.4% on *Alps Seasons* dataset by the classification loss C.

4.7 Qualitative Results

We demonstrate our model on three unsupervised multi-domain image-to-image translation tasks.

Painting style transfer (Fig. 6, Fig. 7) We train our model on *Painters* dataset and use it to generate images of size 256×256. The model can transfer the painting style of a specific painter to the other painters, e.g., transferring the images of *Cezanne* to images of other three painters *Monet, Ukiyoe* and *Vangogh*. In Fig. 7, we also compare the performance of our model with those of reference models when given the same test image.

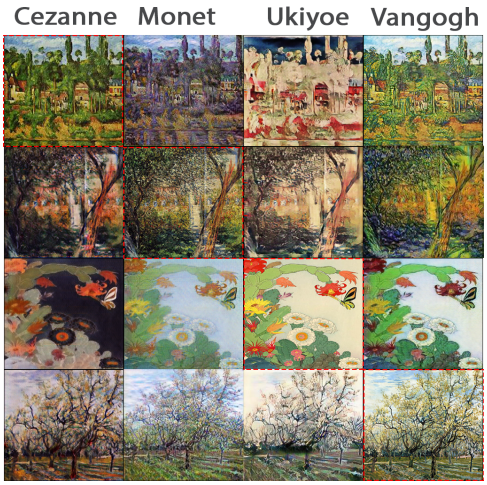

Figure 6: *Painters* translation results. The original images are displayed with a dashed square around. The other images are generated according to different painters.

Figure 8: *Alps Seasons* translation results. The original images are displayed with a dashed square around. The other images are generated according to different seasons.

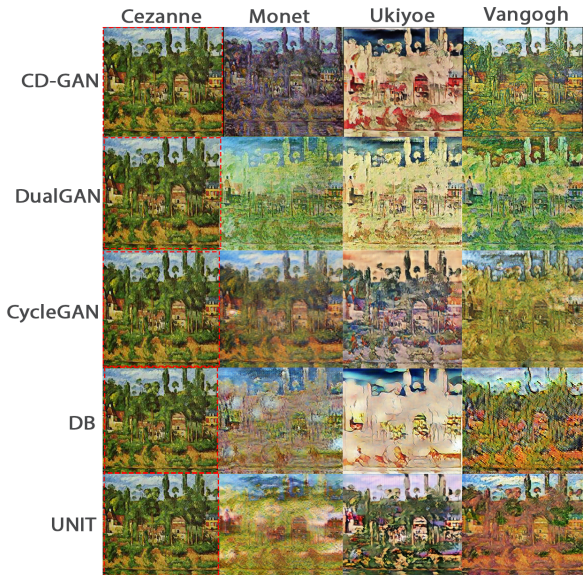

Figure 7: *Painters* translation results. The original images are displayed with a dashed square around. The other images are generated according to different painters.

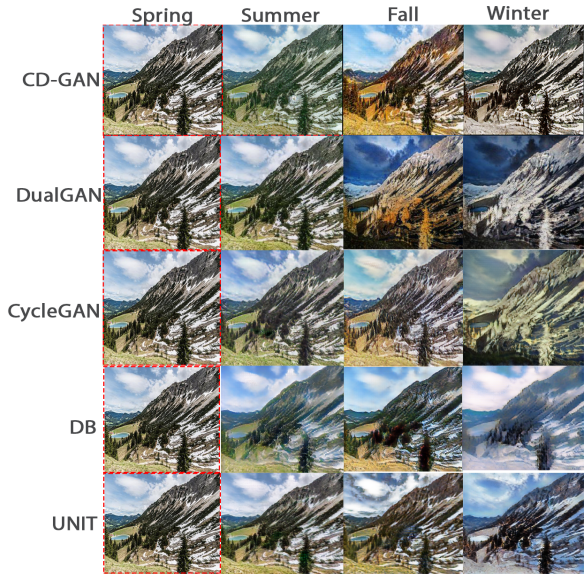

Figure 9: *Alps Seasons* translation results. The original images are displayed with a dashed square around. The other images are generated according to different seasons.

Season transfer (Fig. 8, Fig. 9) The model is trained on the *Alps Seasons* dataset. We use the trained model to generate images of different seasons. For example, we generate an image of summer from an image of spring and vice versa. In Fig. 9, we also compare the performance of our model with those of reference models when given the same test image.

5 CONCLUSION

In this paper, we propose a Cross-Domain Generative Adversarial Networks (CD-GAN), a novel and scalable model to conduct unsupervised multi-domain image-to-image translation. We show its capability of translating images from one domain to many other domain using several datasets. However, the diversity of the generated images are constrained by the cycle consistency loss. We plan to address this problem in the future work.

REFERENCES

[1] A. Anoosheh, E. Agustsson, R. Timofte, and L. Van Gool. 2017. ComboGAN: Unrestrained Scalability for Image Domain Translation. *ArXiv e-prints* (Dec. 2017). arXiv:cs.CV/1712.06909

[2] Yun Cao, Zhiming Zhou, Weinan Zhang, and Yong Yu. 2017. Unsupervised Diverse Colorization via Generative Adversarial Networks. In *ECML/PKDD (1) (Lecture Notes in Computer Science)*, Vol. 10534. Springer, 151–166.

[3] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2672–2680. http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[4] K. Gregor, I. Danihelka, A. Graves, D. Jimenez Rezende, and D. Wierstra. 2015. DRAW: A Recurrent Neural Network For Image Generation. *ArXiv e-prints* (Feb. 2015). arXiv:cs.CV/1502.04623

[5] Raj Kumar Gupta, Alex Yong-Sang Chia, Deepu Rajan, Ee Sin Ng, and Huang Zhiyong. 2012. Image Colorization Using Similar Images. In *Proceedings of the 20th ACM International Conference on Multimedia (MM '12)*. ACM, New York, NY, USA, 369–378. https://doi.org/10.1145/2393347.2393402

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *CVPR*. IEEE Computer Society, 770–778.

[7] L. Hui, X. Li, J. Chen, H. He, C. gong, and J. Yang. [n. d.]. Unsupervised Multi-Domain Image Translation with Domain-Specific Encoders/Decoders. *ArXiv e-prints* ([n. d.]). arXiv:1712.02050

[8] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), 5967–5976.

[9] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. 2017. Learning to Discover Cross-Domain Relations with Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Doina Precup and Yee Whye Teh (Eds.), Vol. 70. PMLR, International Convention Centre, Sydney, Australia, 1857–1865. http://proceedings.mlr.press/v70/kim17a.html

[10] D. P. Kingma and J. Ba. 2014. Adam: A Method for Stochastic Optimization. *ArXiv e-prints* (Dec. 2014). arXiv:cs.LG/1412.6980

[11] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *Proceedings of the Second International Conference on Learning Representations (ICLR 2014)*.

[12] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew P. Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. 2017. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. 105–114. https://doi.org/10.1109/CVPR.2017.19

[13] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. 2017. Unsupervised Image-to-Image Translation Networks. In *Advances in Neural Information Processing Systems 30*.

[14] Ming-Yu Liu and Oncel Tuzel. 2016. Coupled Generative Adversarial Networks. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 469–477. http://papers.nips.cc/paper/6544-coupled-generative-adversarial-networks.pdf

[15] Xiaopei Liu, Liang Wan, Yingge Qu, Tien-Tsin Wong, Stephen Lin, Chi-Sing Leung, and Pheng-Ann Heng. 2008. Intrinsic Colorization. *ACM Trans. Graph.* 27, 5, Article 152 (Dec. 2008), 9 pages. https://doi.org/10.1145/1409060.1409105

[16] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. 2011. Multimodal Deep Learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML'11)*. Omnipress, USA, 689–696. http://dl.acm.org/citation.cfm?id=3104482.3104569

[17] A. Radford, L. Metz, and S. Chintala. 2015. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *ArXiv e-prints* (Nov. 2015). arXiv:cs.LG/1511.06434

[18] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. 2016. Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 2234–2242. http://papers.nips.cc/paper/6125-improved-techniques-for-training-gans.pdf

[19] K. Simonyan and A. Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv e-prints* (Sept. 2014). arXiv:cs.CV/1409.1556

[20] X. Wang and A. Gupta. 2016. Generative Image Modeling using Style and Structure Adversarial Networks. *ArXiv e-prints* (March 2016). arXiv:cs.CV/1603.05631

[21] Z. Yi, H. Zhang, P. Tan, and M. Gong. 2017. DualGAN: Unsupervised Dual Learning for Image-to-Image Translation. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 2868–2876. https://doi.org/10.1109/ICCV.2017.310

[22] R. Zhang, P. Isola, and A. A. Efros. 2016. Colorful Image Colorization. *ArXiv e-prints* (March 2016). arXiv:cs.CV/1603.08511

[23] J. Y. Zhu, T. Park, P. Isola, and A. A. Efros. 2017. Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 2242–2251. https://doi.org/10.1109/ICCV.2017.244