An Introduction to Image Synthesis with Generative Adversarial Nets

He Huang, Philip S. Yu and Changhu Wang

Abstract—There has been a drastic growth of research in Generative Adversarial Nets (GANs) in the past few years. Proposed in 2014, GAN has been applied to various applications such as computer vision and natural language processing, and achieves impressive performance. Among the many applications of GAN, image synthesis is the most well-studied one, and research in this area has already demonstrated the great potential of using GAN in image synthesis. In this paper, we provide a taxonomy of methods used in image synthesis, review different models for text-to-image synthesis and image-to-image translation, and discuss some evaluation metrics as well as possible future research directions in image synthesis with GAN.

Index Terms—Deep Learning, Generative Adversarial Nets, Image Synthesis, Computer Vision.

1 Introduction

With recent advances in deep learning, machine learning algorithms have evolved to such an extent that they can compete and even defeat humans in some tasks, such as image classification on ImageNet [1], playing Go [2] and Texas Hold'em poker [3]. However, we still cannot conclude that those algorithms have true "intelligence", since knowing how to do something does not necessarily mean understanding something, and it is critical for a truly intelligent agent to understand its tasks. "*What I cannot create, I do not understand*", said the famous physicist Richard Feynman. To put this quote in the case of machine learning, we can say that, for machines to understand their input data, they need to learn to create the data. The most promising approach is to use generative models that learn to discover the essence of data and find a best distribution to represent it. Also, with a learned generative model, we can even draw samples which are not in the training set but follow the same distribution.

As a new framework of generative model, Generative Adversarial Net (GAN) [4], proposed in 2014, is able to generate better synthetic images than previous generative models, and since then it has become one of the most popular research areas. A Generative Adversarial Net consists of two neural networks, a generator and a discriminator, where the generator tries to produce realistic samples that fool the discriminator, while the discriminator tries to distinguish real samples from generated ones. There are two main threads of research on GAN. One is the theoretical thread that tries to alleviate the instability and mode collapse problems of GAN [5] [6] [7] [8] [9] [10], or reformulate it from different angles like information theory [11] and energy-based models [12]. The other thread focuses on the applications of GAN in computer vision (CV) [5], natural language processing (NLP) [13] and other areas.

There is a great tutorial given by Goodfellow in NIPS 2016 on GAN [14] where he describes the importance of generative models, explains how GAN works, compares GAN with other generative model and discusses frontier research topics in GAN. Also, there is a recent review paper on GAN [15] which reviews several GAN architectures and training techniques, and introduces some applications of GAN. However, both of these papers are in a general scope, without going into details of specific applications. In this paper, we specifically focus on image synthesis, whose goal is to generate images, since it is by far the most studied area where GAN has been applied. Besides image synthesis, there are many other applications of GAN in computer vision, such as image in-painting [16], image captioning [17] [18] [19], object detection [20] and semantic segmentation [21].

Research of applying GAN in natural language processing is also a growing trend, such as text modeling [13] [22] [23], dialogue generation [24], question answering [25] and neural machine translation [26]. However, training GAN in NLP tasks is more difficult and requires more techniques [13], which also makes it a challenging but intriguing research area.

The main goal of this paper is to provide an overview of the methods used in image synthesis with GAN and point out strengths and weaknesses of current methods. We classify the main approaches in image synthesis into three methods, i.e. direct methods, hierarchical methods and iterative methods. Besides these most commonly used methods, there are also other methods which we will briefly mention. We then give a detailed discussion in two of the most important tasks in image synthesis, i.e. text-to-image synthesis and image-to-image translation. We also discuss the possible reasons why GAN performs so well in certain tasks and its role in our goal to artificial intelligence. The goal of this paper is to provide a simple guideline for those who want to apply GAN to their problems and help further research in GAN.

The rest of this paper is organized as follows. In Section 2 we first review some core concepts of GAN, as well as some

- *He Huang and Philip S. Yu are with the Department of Computer Science, University of Illinois at Chicago, USA.*
  Emails: {hehuang, psyu}@uic.edu
- *Changhu Wang is with ByteDance AI Lab, China.*
  Email: wangchanghu@bytedance.com

variants and training issues. Then in Section 3 we introduce three main approaches and some other approaches used in image synthesis. In Section 4, we discuss several methods in text-to-image synthesis and some possible research directions for improvements. In Section 5, we first introduce supervised and unsupervised methods for image-to-image translation, and then turn to more specific applications like face editing, video prediction and image super-resolution. In Section 6 we review some evaluation metrics for synthetic images, while in Section 7 we discuss the discriminator's role as a learned loss function. Conclusions are given in Section 8.

2 GAN Preliminaries

In this section, we review some core concepts of Generative Adversarial Nets (GANs) and some improvements.

A generative model G parameterized by θ takes as input a random noise $\mathbf{z}$ and output a sample $G(\mathbf{z};\theta)$, so the output can be regarded as a sample drawn from a distribution: $G(\mathbf{z};\theta) \sim p_g$. Meanwhile, we have a lot of training data $\mathbf{x}$ drawn from p_{data}, and the training objective for the generative model G is to approximate p_{data} using p_g.

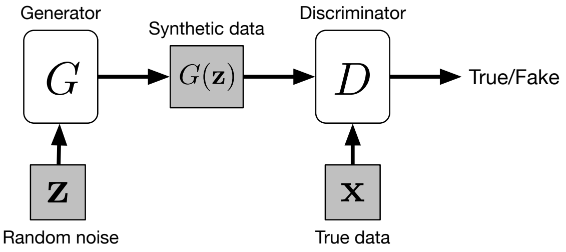

Fig. 1. General structure of a Generative Adversarial Network, where the generator G takes a noise vector $\mathbf{z}$ as input and output a synthetic sample $G(\mathbf{z})$, and the discriminator takes both the synthetic input $G(\mathbf{z})$ and true sample $\mathbf{x}$ as inputs and predict whether they are real or fake.

Generative Adversarial Net (GAN) [4] consists of two separate neural networks: a generator G that takes a random noise vector $\mathbf{z}$, and outputs synthetic data $G(\mathbf{z})$; a discriminator D that takes an input $\mathbf{x}$ or $G(\mathbf{z})$ and output a probability $D(\mathbf{x})$ or $D(G(\mathbf{z}))$ to indicate whether it is synthetic or from the true data distribution, as shown in Figure 1. Both of the generator and discriminator can be arbitrary neural networks. The first GAN [4] uses fully connected layer as its building block. Later, DCGAN [5] proposes to use fully convolutional neural networks which achieves better performance, and since then *convolution* and *transposed convolution* layers have become the core components in many GAN models. For more details on (transposed) convolution arithmetic, please refer to this report [27].

The original way to train the generator and discriminator is to form a two-player min-max game where the generator G tries to generate realistic data to fool the discriminator while discriminator D tries to distinguish between real and synthetic data [4]. The value function to be optimized is shown in Equation 1, where $p_{data}(\mathbf{x})$ denotes the true data distribution and $p_z(\mathbf{z})$ denote the noise distribution.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1-D(G(\mathbf{z})))] \quad (1)$$

However, when the discriminator is trained much better than the generator, D can reject the samples from G with confidence close to 1, and thus the loss $\log(1-D(G(\mathbf{z})))$ saturates and G can not learn anything from zero gradient. To prevent this, instead of training G to minimize $\log(1-D(G(\mathbf{z})))$, we can train it to maximize $\log D(G(\mathbf{z}))$ [4]. Although the new loss function for G gives a different scale of gradient than the original one, it still provides the same direction of gradient and does not saturate.

2.1 Conditional GAN

In the original GAN, we have no control of what to be generated, since the output is only dependent on random noise. However, we can add a conditional input $\mathbf{c}$ to the random noise $\mathbf{z}$ so that the generated image is defined by $G(\mathbf{c},\mathbf{z})$ [28]. Typically, the conditional input vector $\mathbf{c}$ is concatenated with the noise vector $\mathbf{z}$, and the resulting vector is put into the generator as it is in the original GAN. Besides, we can perform other data augmentation on $\mathbf{c}$ and $\mathbf{z}$, as in [29]. The meaning of conditional input $\mathbf{c}$ is arbitrary, for example, it can be the class of image, attributes of object [28] or an embedding of text descriptions of the image we want to generate [30] [31].

2.2 GAN with Auxiliary Classifier

In order to feed more side-information and to allow for semi-supervised learning, one can add an additional task-specific auxiliary classifier to the discriminator, so that the model is optimized on the original tasks as well as the additional task [32] [6]. The architecture of such method is illustrated in Figure 2, where C is the auxiliary classifier. Adding auxiliary classifiers allows us to use pre-trained models (e.g. image classifiers trained on ImageNet), and experiments in AC-GAN [33] demonstrate that such method can help generating sharper images as well as alleviate the *mode collapse* problem. Using auxiliary classifiers can also help in applications such as text-to-image synthesis [34] and image-to-image translation [35].

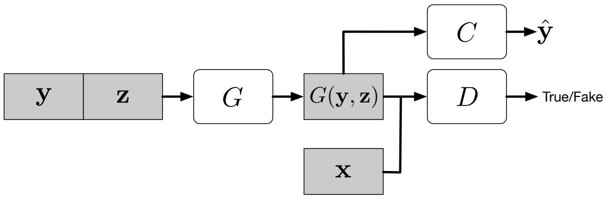

Fig. 2. Architecture of GAN with auxiliary classifier, where $\mathbf{y}$ is the conditional input label and C is the classifier that takes the synthetic image $G(\mathbf{y},\mathbf{z})$ as input and predict its label $\hat{\mathbf{y}}$

2.3 GAN with Encoder

Although GAN can transform a noise vector $\mathbf{z}$ into a synthetic data sample $G(\mathbf{z})$, it does not allow inverse transformation. If we treat the noise distribution as a latent feature space for data samples, GAN lacks the ability to map data sample $\mathbf{x}$ into latent feature $\mathbf{z}$. In order to allow such mapping, two concurrent works BiGAN [36] and ALI [37] propose to add an encoder E in the original GAN framework, as shown in Figure 3. Let $\Omega_{\mathbf{x}}$ be the data space and $\Omega_{\mathbf{z}}$ be the latent feature space, the encoder E takes $\mathbf{x} \in \Omega_{\mathbf{x}}$ as input and produce a feature vector $E(\mathbf{x}) \in \Omega_{\mathbf{z}}$ as output. The discriminator D is modified to take both a data sample and a feature vector as input to calculate $P(Y|\mathbf{x}, \mathbf{z})$, where $Y = 1$ indicates the sample is real and $Y = 0$ means the data is generated by G.

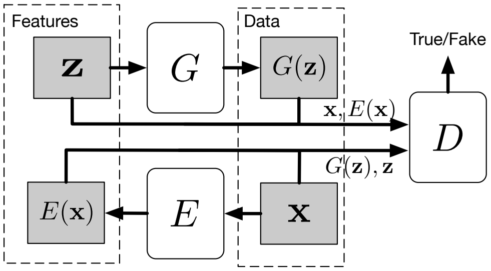

Fig. 3. Architecture of BiGAN/ALI

The objective is thus defined as:

$$\min_{G,E} \max_{D} V(G, E, D) = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} \log D(\mathbf{x}, E(\mathbf{x})) + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} \log(1 - D(G(\mathbf{z}), \mathbf{z})) \quad (2)$$

2.4 GAN with Variational Auto-Encoder

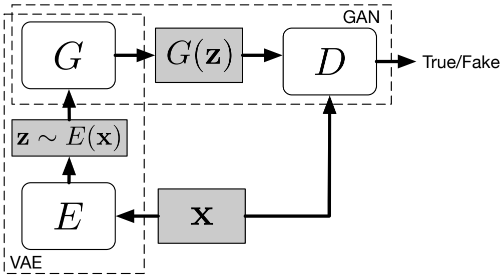

Fig. 4. Architecture of VAE-GAN

VAE-GAN [38] proposes to combine Variational Auto-Encoder (VAE) [39] with GAN [4] to exploit both of their benefits, as GAN can generate sharp images but often miss some modes while images produced by VAE [39] are blurry but have large variety. The architecture of VAE-GAN is shown in Figure 4. The VAE part regularize the encoder E by imposing a prior of normal distribution (e.g. $\mathbf{z} \sim \mathcal{N}(0, 1)$), and the VAE loss term is defined as:

$$\mathcal{L}_{VAE} = -\mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \log[p(\mathbf{x}|\mathbf{z})] + D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{x})), \quad (3)$$

where $\mathbf{z} \sim E(\mathbf{x}) = q(\mathbf{z}|\mathbf{x})$, $\mathbf{x} \sim G(\mathbf{z}) = p(\mathbf{x}|\mathbf{z})$ and D_{KL} is the Kullback-Leibler divergence.

Also, VAE-GAN [38] proposes to represent the reconstruction loss of VAE in terms of the discriminator D. Let $D_l(\mathbf{x})$ denotes the representation of the l-th layer of the discriminator, and a Gaussian observation model can be defined as:

$$p(D(\mathbf{x})|\mathbf{z}) = \mathcal{N}(D(\mathbf{x})|D(\tilde{\mathbf{x}}), \mathbf{I}), \quad (4)$$

where $\tilde{\mathbf{x}} \sim G(\mathbf{z})$ is a sample from the generator, and $\mathbf{I}$ is the identity matrix. So the new VAE loss is:

$$\mathcal{L}_{VAE} = -\mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \log[p(D(\mathbf{x})|\mathbf{z})] + D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{x})), \quad (5)$$

which is then combined with the GAN loss defined in Equation 1. Experiments demonstrate that VAE-GAN can generate better images than VAE or GAN alone.

2.5 Handling Mode Collapse

Although GAN is very effective in image synthesis, its training process is very unstable and requires a lot of tricks to get a good result, as pointed out in [4] [5]. Despite its instability in training, GAN also suffers from the *mode collapse* problem, as discussed in [4] [5] [40] . In the original GAN formulation [4], the discriminator does not need to consider the variety of synthetic samples, but only focuses on telling whether each sample is realistic or not, which makes it possible for the generator to spend efforts in generating a few samples that are good enough to fool the discriminator. For example, although the MNIST [41] dataset contains images of digits from 0 to 9, in an extreme case, a generator only needs to learn to generate one of the ten digits perfectly to completely fool the discriminator, and then the generator stops trying to generate the other nine digits. The absence of the other nine digits is an example of *inter-class mode collapse*. An example of *intra-class mode collapse* is, there are many writing styles for each of the digits, but the generator only learns to generate one perfect sample for each digit to successfully fool the discriminator.

Many methods have been proposed to address the *model collapse* problem. One technique is called *minibatch features* [6], whose idea is to make the discriminator compare an example to a minibatch of true samples as well as a minibatch of generated samples. In this way, the discriminator can learn to tell if a generated sample is too similar to some other generated samples by measuring samples' distances in latent space. Although this method works well, as discussed in [14], the performance largely depends on what features are used in distance calculation. MRGAN [7] proposes to add an encoder which transforms a sample in data space back to latent space, as in BiGAN [36]. The combination of encoder and generator acts as an auto-encoder, whose reconstruction loss is added to the adversarial loss to act as a mode regularizer. Meanwhile, the discriminator is also trained to discriminate reconstructed samples, which acts as another mode regularizer. WGAN [8] proposes to use Wasserstein distance to measure the similarity between true data distribution and the learned distribution, instead of using Jensen-Shannon divergence as in the original GAN [4]. Although it theoretically avoids mode collapse, it takes a longer time for the model to converge than previous GANs. To alleviate this problem, WGAN-GP [9] proposes to use gradient penalty, instead of weight clipping in WGAN. WGAN-GP generally produces good images and greatly avoid mode collapse, and it is easy to apply this training framework to other GAN models. More tips for training GANs can be found in Soumith's NIPS 2016 tutorial "How to train a GAN"[1].

[1] https://github.com/soumith/ganhacks

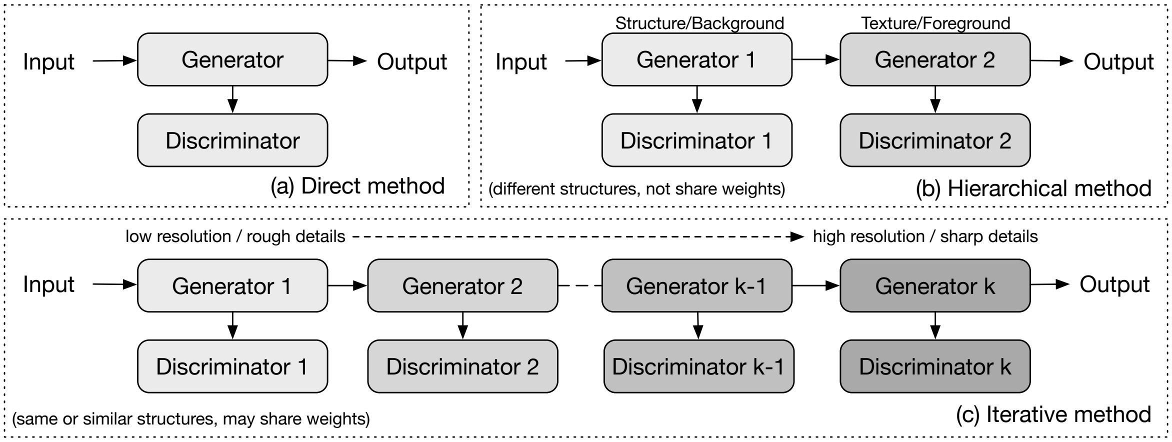

Fig. 5. Three approaches of image synthesis using Generative Adversarial Networks. The direct method does everything with only one generator and one discriminator, while the other two methods have multiple generators and discriminators. Hierarchical method usually use two layers of GANs, where each GAN plays a fundamentally different role than the other one. Iterative method, however, contains multiple GANs that perform the same task but operate at different resolution.

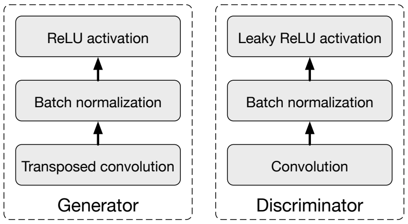

Fig. 6. Building blocks of DCGAN, where the generator uses transposed convolution, batch-normalization and ReLU activation, while the discriminator uses convolution, batch-normalization and LeakyReLU activation

3 General Approaches of Image Synthesis with GAN

In this section, we summarize the three main approaches used in generating images, i.e. *direct methods*, *iterative methods* and *hierarchical methods* respectively, which form the basis of all applications mentioned in this paper. The overall structures of these three methods are shown in Figure 5.

3.1 Direct Methods

All methods under this category follows the philosophy of using one generator and one discriminator in their models, and the structures of the generator and the discriminator are straight-forward without branches. Many of the earliest GAN models fall into this category, like GAN [4], DCGAN [5], ImprovedGAN [6], InfoGAN [11], f-GAN [42] and GAN-INT-CLS [30]. Among them, DCGAN is one of the most classic ones whose structure is used by many later models such as [11] [30] [43] [19]. The general building blocks used in DCGAN are shown in Figure 6, where the generator uses transposed convolution, batch-normalization and ReLU activation, while the discriminator uses convolution, batch-normalization and LeakyReLU activation.

This kind of method is relatively more straight-forward to design and implement when compared with *hierarchical* and *iterative* methods, and it usually achieves good results.

3.2 Hierarchical Methods

Contrary to the *Direct Method*, algorithms under the *Hierarchical Method* use two generators and two discriminators in their models, where different generators have different purposes. The idea behind those methods is to separate an image into two parts, like “styles & structure” and “foreground & background”. The relation between the two generators may be either parallel or sequential.

SS-GAN [44] proposes to use two GANs, a Structure-GAN for generating a surface normal map from random noise $\hat{\mathbf{z}}$, and another Style-GAN that takes both the generated surface normal map as well as a noise $\tilde{\mathbf{z}}$ as input and outputs an image. The Structure-GAN uses the same building blocks as DCGAN [5], while the Style-GAN is slightly different. For Style-Generator, the generated surface normal map and the noise vector go through several convolutional and transposed convolutional layers respectively, and then the results are concatenated into a single tensor which will go through the remaining layers in Style-Generator. As for the Style-Discriminator, each surface normal map and its corresponding image are concatenated at the channel dimension to form a single input to the discriminator. Besides, SS-GAN assumes that, a good synthetic image should also be used to reconstruct a good surface normal map. Under this assumption, SS-GAN designs a fully-connected network that transforms an image back to its surface normal map, and uses a pixel-wise loss that enforces the reconstructed surface normal to approximate the true one. A main limitation of SS-GAN is that it requires to use Kinect to obtain groundtruth for surface normal maps.

As a special example, LR-GAN [45] chooses to generate the foreground and background content using different generator, but only one discriminator is used to judge the images while the recurrent image generation process is related to the iterative method. Nonetheless, experiments of LR-GAN demonstrate that it is possible to separate the generation of foreground and background content and produce sharper images.

3.3 Iterative Methods

This method differentiates itself from *Hierarchical Methods* in two ways. First, instead of using two different generators that perform different roles, the models in this category use multiple generators that have similar or even the same structures, and they generate images from coarse to fine, with each generator refining the details of the results from the previous generator. Second, when using the same structures in the generators, *Iterative Methods* can use weight-sharing among the generators [45], while *Hierarchical Methods* usually can not.

LAPGAN [40] is the first GAN that uses an iterative method to generate images from coarse to fine using Laplacian pyramid [46]. The multiple generators in LAPGAN perform the same task: takes an image from previous generator and a noise vector as input, and then outputs the details (a residual image) that can make the image sharper when added to the input image. The only difference in the structures of those generators is the size of input/output dimension, while an exception is that the generator at the lowest level only takes a noise vector as input and outputs an image. LAPGAN outperforms the original GAN [4] and shows that iterative method can generate sharper images than direct method.

StackGAN [29], as an iterative method, has only two layers of generators. First generator takes an input $(\mathbf{z}, \mathbf{c})$ and then outputs a blurry image that can show a rough shape and blurry details of the objects, while the second generator takes $(\mathbf{z}, \mathbf{c})$ and the image generated by the previous generator and then output a larger image with more photo-realistic details.

Another example of *Iterative Methods* is SGAN [47] which stacks generators that takes lower level feature as input and outputs higher level features, while the bottom generator takes a noise vector as input and the top generator outputs an image. The necessity of using separate generators for different levels of features is that SGAN associates an encoder, a discriminator and a Q-network [47] (which is used to predict the posterior probability $P(\mathbf{z}_i|\mathbf{h}_i)$ for entropy maximization, where $\mathbf{h}_i$ is the output feature of the i-th layer) for each generator, so as to constrain and improve the quality of those features.

An example of using weight-sharing is the GRAN [48] model, which is an extension to the DRAW [49] model which is based on variational autoencoder [39]. As in DRAW, GRAN generates an image in a recurrent way that feeds the output of the previous step into the model and the output of the current step will be fed back as the input in the next step. All steps use the same generator, so the weights are shared among them, just like classic Recurrent Neural Network (RNN).

3.4 Other Methods

PPGN [50] produces impressive images in several tasks, such as class-conditioned image synthesis [28], text-to-image synthesis [30] and image inpainting [16]. Different from other methods mentioned earlier, PPGN uses *activation maximization* [51] to generate images, and it is based on sampling with a prior learned with denoising autoencoder (DAE) [52]. To generate an image conditioned on a certain class label y, instead of using a feed-forward way (e.g. recurrent methods can be seen as feed-forward if unfolded through time), PPGN runs an optimization process that finds an input $\mathbf{z}$ to the generator that makes the output image highly activate a certain neuron in another pretrained classifier (in this case, the neuron in the output layer that corresponds to its class label y).

In order to generate better higher resolution images, ProgressiveGAN [53] proposes to start with training a generator and discriminator of 4×4 pixels, after which it incrementally adds extra layers that doubles the output resolution up to 1024×1024. This approach allows the model to learn coarse structure first and then focus on refining details later, instead of having to deal with all details at different scale simultaneously.

4 Text-to-Image Synthesis

When we apply GAN to image synthesis, it is desired to control the content of generated images. Although there are label-conditioned GAN models like cGAN [28] which can generate images belong to a specific class, it remains a great challenge to generate images based on text descriptions. *Text-to-image synthesis* is kind of the holy grail of computer vision, since if an algorithm can generate truly realistic images from mere text descriptions, we can have a high confidence that the algorithm actually understands what is in the images, where computer vision is about teaching computers to see and understand visual contents in the real world.

GAN-INT-CLS [30] provides the first attempt of using GAN to generate images from text descriptions. The idea is similar to conditional GAN that concatenates the condition vector with the noise vector, but with the difference of using the embedding of text sentences instead of class labels or attributes. The embedding method used in GAN-INT-CLS [30] is from another paper [54] that tries to learn robust embeddings of images and sentences given the image-sentence pairs. Except for the sentence embedding method, the generator of GAN-INT-CLS follows the same architecture of DCGAN [5]. As for the discriminator, in order to take into account the text description, the text embedding vector of length K is *spatially replicated* to become a text embedding tensor of shape $[W \times H \times K]$, where W and H are the width and height of the generated image's feature tensor after going through several convolutional layers in the discriminator. Then the text embedding tensor is combined with the image feature tensor of shape $[W \times H \times C]$ at the channel dimension C and forms a new tensor of shape $[W\times H\times(C+K)]$, which then goes through the rest layers of the discriminator. The intuition behind this approach is that, by spatial replication and depth concatenation, each "*pixel*" (of shape $[1 \times 1 \times (C + K)]$) of the image's feature tensor contains all the information of text description, and then the convolutional layers can learn to align the image's content with certain parts of the text feature by using multiple convolution kernels.

GAN-INT-CLS [30] also proposes to distinguish between two sources of errors: *unrealistic image with any text*, and *realistic image with mismatched text*. To train the discriminator to distinguish these two kinds of errors, three types of input are fed into the discriminator at every training step: {*real image, right text*}, {*real image, wrong text*} and {*fake image, right text*}. The experimental results in [30] show that such training technique is important in generating high quality images, since it tells the model not only how to generate realistic images, but also the correspondence between text and images.

In addition, GAN-INT-CLS [30] proposes to use manifold interpolation to obtain more text embeddings, by simply interpolating between captions in the training set. Since the number of captions for each image is usually no more than five and an image can be described in many ways, doing such interpolation allows the model to learn from possible text descriptions that are not in the training set. According to the authors, the interpolated text embeddings need not correspond to actual human-written text, so there is no extra labeling required.

TAC-GAN [34] is a combination of GAN-INT-CLS [30] and AC-GAN [33]. With the auxiliary classifier, TAC-GAN is able to achieve higher *Inception Score* [6] than GAN-INT-CLS [30] and StackGAN [29] on the Oxford-102 [55] daataset.

4.1 Text-to-Image with Location Constraints

Although GAN-INT-CLS [30] and StackGAN [29] can generate images based on text description, they fail to capture the localization constraints of the objects in images. To allow encoding spatial constraints, Reed et al. propose GAWWN [31] that presents two possible solutions.

The first approach proposed in GAWWN [31] is to learn a bounding box for an object by putting a spatially replicated text embedding tensor through a *Spatial Transformer Network* [56]. Here the *spatial replication* is the same process mentioned in GAN-INT-CLS [30]. The output of spatial transformer network is a tensor with the same dimension as input, but values outside of the bounding are all zeros. The output tensor of the spatial transformer goes through several convolutional layers to reduce its size back to a 1-dimensional vector, which not only preserves the information of text but also provides a constraint on object's location by the bounding box. A benefit of this approach is that it is end-to-end, without requiring additional input.

The second approach proposed in GAWWN [31] is to use user-specified keypoints to constrain the different parts (e.g. head, leg, arm, tail, etc.) of the object in the image. For each keypoint, a mask matrix is generated where the keypoint position is 1 and others 0, and all the matrices are combined through depth concatenation to form a mask tensor of shape $[M \times M \times K]$, where M is the size of the mask and K is the number of keypoints. The tensor is then flattened into a binary matrix with 1 indicating the presence of a keypoint and 0 otherwise, and then replicated depth-wise to become a tensor to be fed into remaining layers. Although this approach allows more detailed constraints on the object, it requires extra user input to specify the keypoints. Even if GAWWN can infer unobserved keypoints from a few user-specified keypoints so that the user does not need to specify all keypoints, the cost of extra user input is still non-trivial.

The remaining part of GAWWN [31] is similar to GAN-INT-CLS [30], with the difference of using a separate pathway to process bounding box tensor or keypoint tensor. Although GAWWN provides two approaches that can enforce location constraints on the generated images, it only works on images with single objects, since neither of the proposed methods is able to handle several different objects in an image. From the result shown in [31], GAWWN works well on the CUB dataset [57], while the synthetic images generated from the model trained on MPII Human Pose (MHP) dataset [58] are very blurry and it is hard to tell what the content is. This may be due to the fact that the poses of a standing bird are very similar to each other (note that birds in the CUB dataset are at standing poses), while a human's poses can be of uncountable types.

The main benefit of specifying the locations of each part of the objects is that it yields more interpretable results, and that it is desirable that the model can understand the concepts of different parts of objects, which is one of the ultimate goals of computer vision.

4.2 Text-to-Image with Stacked GANs

Instead of using only one generator, StackGAN [29] proposes to use two different generators for text-to-image synthesis. The first generator is responsible for generating low-resolution images that contain rough shapes and colors of objects, while the second generator takes the output of the first generator and produces images with higher resolution and sharper details. Each generator is associated with its own discriminator. Besides, StackGAN also proposes a conditional data augmentation technique to produce more text embeddings for the generator. StackGAN randomly samples from a Gaussian distribution $\mathcal{N}(\mu(\phi_t), \Sigma(\phi_t))$, where the mean vector $\mu(\phi_t)$ and diagonal variance matrix $\Sigma(\phi_t)$ are functions of text embedding ϕ_t. To further enforce the smoothness over conditional manifold, $\mathcal{N}(\mu(\phi_t), \Sigma(\phi_t))$ is constrained to approximate a standard Gaussian distribution $\mathcal{N}(0, 1)$ by adding a Kullback-Leibler divergence regularization term.

As an improved version of StackGAN, StackGAN++ [59] proposes to have more pairs of generators and discriminators instead of just two, adds an unconditional image synthesis loss to the discriminators, and uses a color-consistency regularization term calculated by mean-square loss of the means and variances between real and fake images.

AttnGAN [60] further extends the architecture of StackGAN++ [59] by using attention mechanism over image and text features. In AttnGAM, each sentence is embedded into a global sentence vector and each word of the sentence is also embedded into a word vector. The global sentence vector is used to generate a low resolution image at the first stage, and then the following stages use the input image features from the previous stage and the word vectors as input to the attention layer and calculate a word-context vector which will be combined with the image features and form the input to the generator that will generate new image features. Besides, AttnGAN also proposes a Deep Attentional Multimodal Similarity Model (DAMSM) that uses attention layers to compute the similarity between images and text using both global sentence vectors as well as fine-grained word vectors, which provides an additional fine-grained image-text matching loss for training the generator. Experiments of

AttnGAN not only show the effectiveness of using attention layers in image synthesis, but also make the model more interpretable.

With stacked generators, StackGAN [29], StackGAN++ [59] and AttnGAN [60] produce sharper images than GAN-INT-CLS [30] and GAWWN [31] on the CUB [57] and Oxford-102 [55] datasets. Although AttnGAN [60] claims to have significantly higher *Inception Score* [6] than PPGN [50] on the COCO [61] dataset, the examples it provides do not visually look apparently better.

4.3 Text-to-Image by Iterative Sampling

Different from previous approaches that directly incorporate the text information in the generation process, PPGN [50] proposes to use *activation maximization* [51] method to generate images in an iterative sampling way. The model contains two separate parts: a pretrained image captioning model, and an image generator. The image generator is a combination of denoising auto-encoder and GAN, trained independent of the image captioning model. Let $p(\mathbf{x})$ be the distribution of images, and $p(\mathbf{y})$ be the distribution of text descriptions, we want to sample image from the joint distribution $p(\mathbf{x}, \mathbf{y})$. PPGN factorizes the joint distribution into two factors: $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y}|\mathbf{x})$, where $p(\mathbf{x})$ is modeled by the generator, and $p(\mathbf{y}|\mathbf{x})$ is modeled by the image captioning model. According to the paper [50], given a trained image generator and image captioning model, the following iterative sampling process is performed to obtain an image $\mathbf{x}$ based on the description $\mathbf{y}^*$:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \epsilon_1 \frac{\partial \log p(\mathbf{x}_t)}{\partial \mathbf{x}_t} + \epsilon_2 \frac{\partial \log p(\mathbf{y} = \mathbf{y}^* | \mathbf{x}_t)}{\partial \mathbf{x}_t} + N(0, \epsilon_3^2), \tag{6}$$

where the ϵ_1 term takes a step from current image $\mathbf{x}_t$ to a more realistic image (regardless of the text description), the ϵ_2 term takes a step from current image to an image that better matches the description $\mathbf{y}^*$ (here the LRCN model [62] is used for the $p(\mathbf{y} = \mathbf{y}^*|\mathbf{x})$ term), and the ϵ_3 term adds a small noise to allow for a broader search in the latent space.

Although such iterative sampling method takes more time to generate an image in test phrase, it can generate higher resolution images with better quality than previous methods like [30] [31] [29], and its performance is among the best in both class-conditioned and text-conditioned image synthesis.

4.4 Limitations of Current Text-to-Image Models

Present text-to-image models perform well on datasets with single object per image, such as human faces in CelebA [63], birds in CUB [57], flowers in Oxford-102 [55], and some objects in ImageNet [1]. Also, they can synthesize reasonable images for scenes like bedrooms and living rooms in the LSUN [64], even though the objects in the scenes lack sharp details. However, all present models work badly in situation where multiple complicated objects are involved in one image, as it is in the MS-COCO dataset [61].

A plausible reason why current models fail to work well on complicated images is that the models only learn the overall features of an image, instead of learning the concept of each kind of objects in it. This gives an explanation why synthetic scenes of bedrooms and living rooms lack sharp details, since the model do not distinguish between a bed and a desk, all it sees is that some patterns of shapes and colors should be put somewhere in the synthetic image. In other words, the model does not really understand the image, but just remembers where to put some shapes and colors.

Generative Adversarial Network certainly provides us a promising way to do text-to-image synthesis, since it produces sharper images than any other generative methods so far. To take a further step in text-to-image synthesis, we need to figure out novel ways to teach the algorithms the concepts of things. One possible way is to train separate models that can generate different kinds of objects, and then train another model that learns how to combine different objects (i.e. the reasonable relations between objects) into one image based on the text descriptions. However, such method requires large training sets for different objects, and another large dataset that contains images of those different objects, which is hard to acquire. Another possible direction may be to make use of the *Capsule* idea proposed by Hinton et al. [65] [66] since *capsules* are designed to capture the concepts of objects, but how to efficiently train such *capsule*-based network is still a problem to be solved.

5 Image-to-Image Translation

In this section, we discuss another type of image synthesis, *image-to-image translation*, which takes images as conditional input. *Image-to-image translation* is defined as the problem of translating a possible representation of one scene into another, such as mapping BW images into RGB images, or the other way around [67]. This problem is related to *style transfer* [68] [69], which takes a content image and a style image and output an image with the content of the content image and the style of the style image. *Image-to-image translation* can be viewed as a generalization of *style transfer*, since it is not limited to transferring the styles of images, but can also manipulate attributes of objects (as in applications of *face editing*). In this section, we introduce several models that work for general *image-to-image* translation, from supervised methods to unsupervised ones. The "supervision" here means that for each image in the source domain, there is a corresponding ground-truth image in the target domain. Later we will turn into different specific applications such as *face editing*, *image super resolution*, *image in-painting* and *video prediction*.

5.1 Supervised Image-to-Image Translation

Pix2Pix [67] proposes to combine the loss of a conditional Generative Adversarial Network (cGAN) with L1 regularization loss, so that the generator is not only trained to fool the discriminator but also to generate images as close to ground-truth as possible. The reason for using L1 instead of L2 is that L1 produces less blurry images [70].

The conditional GAN loss is defined as:

$$\begin{aligned}\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{data}(\mathbf{x},\mathbf{y})}[\log D(\mathbf{x}, \mathbf{y})] + \\ \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x}),\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(\mathbf{x}, G(\mathbf{x}, \mathbf{z}))],\end{aligned} \tag{7}$$

where $x, y \sim p(x, y)$ are images of the same scene with different styles, $z \sim p(z)$ is a random noise as in the regular GAN [4].

The L1 loss for constraining self-similarity is defined as:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{data}(\mathbf{x},\mathbf{y}),\mathbf{z}\sim p_z(\mathbf{z})}[||\mathbf{y} - G(\mathbf{x},\mathbf{z})||_1], \quad (8)$$

The overall objective is thus given by:

$$G^*, D^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda\mathcal{L}_{L1}(G), \quad (9)$$

where λ is a hyper-parameter to balance the two loss terms.

The authors of Pix2Pix [67] find that the noise z does not have obvious effect on the output, so they provide the noise in the form of dropout at training and test time instead of drawing samples from a random distribution.

The generator structure for Pix2Pix [67] is based on U-Net [71], which belongs to the encoder-decoder framework but adds skip connections from encoder to decoder so as to allow circumventing the bottleneck for sharing low-level information like edges of objects.

Pix2Pix [67] proposes PatchGAN (the patch-based idea was previously explored in MGAN [72]) as the discriminator, which, instead of classifying the whole image, tries to classify each $N \times N$ path of the image and average all the scores of patches to get the final score for the image. The motivation of this method is that, although L1 and L2 losses produce blurry images and fail to capture high frequency details, in many cases they can capture the low frequencies quite well. In order to capture the high frequency details, Pix2Pix [67] argues that it is sufficient to restrict the discriminator to focus only on local patches. From the experiments, it is found that, for an 256×256 image, a patch-size of 70×70 works best.

Although Pix2Pix produces very impressive synthetic images, the major limitation is that it must use paired images as supervision, as is shown in Equation 8 that data pair $(\mathbf{x}, \mathbf{y})$ is drawn from the joint distribution $p(\mathbf{x}, \mathbf{y})$.

5.2 Supervised Image-to-Image Translation with Pairwise Discrimination

PLDT [35] proposes another method to do supervised *image-to-image translation*, by adding another discriminator D_{pair} that learns to tell whether a pair of images from different domains is associated with each other. The architecture of PLDT is shown in Figure 7. Given an input image $\mathbf{x}_s$ from source domain, its ground-truth image $\mathbf{x}_t$ in the target domain, an irrelevant image $\mathbf{x}_t^-$ in the target domain, and the generator G transfers $\mathbf{x}_s$ into an image $\hat{\mathbf{x}}_t$ in the target domain, the loss for D_{pair} can be defined as:

$$\begin{aligned}\mathcal{L}_{pair} = &-t\cdot\log[D_{pair}(\mathbf{x}_s,\mathbf{x})]\\ &+(t-1)\cdot\log[1-D_{pair}(\mathbf{x}_s,\mathbf{x})],\\ \text{s.t. } t = &\begin{cases}0 & \text{if } \mathbf{x}=\mathbf{x}_t\\ 0 & \text{if } \mathbf{x}=\hat{\mathbf{x}}_t\\ 1 & \text{if } \mathbf{x}=\mathbf{x}_t^-.\end{cases}\end{aligned} \quad (10)$$

The generator of PLDT [35] is implemented in an encoder-decoder fashion using (transposed) convolutions, while the two discriminators are implemented as fully convolutional networks. As shown in the experimental results, PLDT [35] performs domain transfer that modifies the geometric shapes of objects while trying to keep the texture consistent among all associated images.

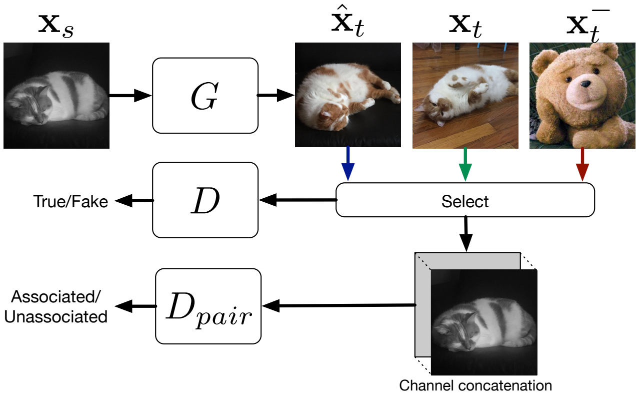

Fig. 7. Architecture of Pixel-Level Domain Transfer (PLDT) [35]. In this example, the source domain is BW color space, while the target domain is RGB space. As explained in [35], this model can performs geometric modifications on the input image, while keeping the object in the output image the same as input.

5.3 Unsupervised Image-to-Image Translation with Cyclic Loss

Two concurrent works CycleGAN [73] and DualGAN [74] propose to add a self-consistency (reconstruction) loss that tries to preserve the input image after a cycle of transformation. CycleGAN and DualGAN share the same framework, which is shown in Figure 8. As we can see, the two generators G_{AB} and G_{BA} are doing opposite transformations, which can be seen as a kind of *dual learning* [75]. Besides, DiscoGAN [76] is another model that utilizes the same cyclic framework as Figure 8.

Here we use CycleGAN as an example. In CycleGAN, there are two generators, G_{AB} that transfer an image from domain A to B and G_{BA} that performs the opposite transformation. Also, there are also two discriminators D_A and D_B that predicts whether an image belongs to that domain. For a pair of G_{AB} and D_B, the adversarial loss function is defined as:

$$\begin{aligned}\mathcal{L}_{GAN}(G_{AB}, D_B) = &\ \mathbb{E}_{\mathbf{b}\sim p_B(\mathbf{b})}[\log D_B(\mathbf{b})]\\ &+\mathbb{E}_{\mathbf{a}\sim p_A(\mathbf{a})}[1-\log(D_B(G_{AB}(\mathbf{a}))], \end{aligned} \quad (11)$$

and similarly for the pair G_{BA} and D_A, we can define the adversarial loss as $\mathcal{L}_{GAN}(G_{BA}, D_A)$.

Besides the adversarial loss, a cycle-consistency loss is designed to minimize the reconstruction error after we translate an image of one domain to another and then translate it back to the original domain, i.e. $\mathbf{a} \rightarrow G_{AB}(\mathbf{a}) \rightarrow G_{BA}(G_{AB}(\mathbf{a})) \approx \mathbf{a}$. Since this cycle can be defined from two directions, the cycle-consistency loss is defined as:

$$\begin{aligned}\mathcal{L}_{cyc}(G_{AB}, G_{BA}) = &\ \mathbb{E}_{\mathbf{a}\sim p_A(\mathbf{a})}[||\mathbf{a} - G_{BA}(G_{AB}(\mathbf{a}))||_1]\\ &+\mathbb{E}_{\mathbf{b}\sim p_B(\mathbf{b})}[||\mathbf{b} - G_{AB}(G_{BA}(\mathbf{b}))||_1].\end{aligned} \quad (12)$$

Then the overall loss function is:

$$\begin{aligned}\mathcal{L}(G_{AB}, G_{BA}, D_A, D_B) = &\ \mathcal{L}_{GAN}(G_{AB}, D_B)\\ &+\mathcal{L}_{GAN}(G_{BA}, D_A)\\ &+\lambda\mathcal{L}_{cyc}(G_{AB}, G_{BA}),\end{aligned} \quad (13)$$

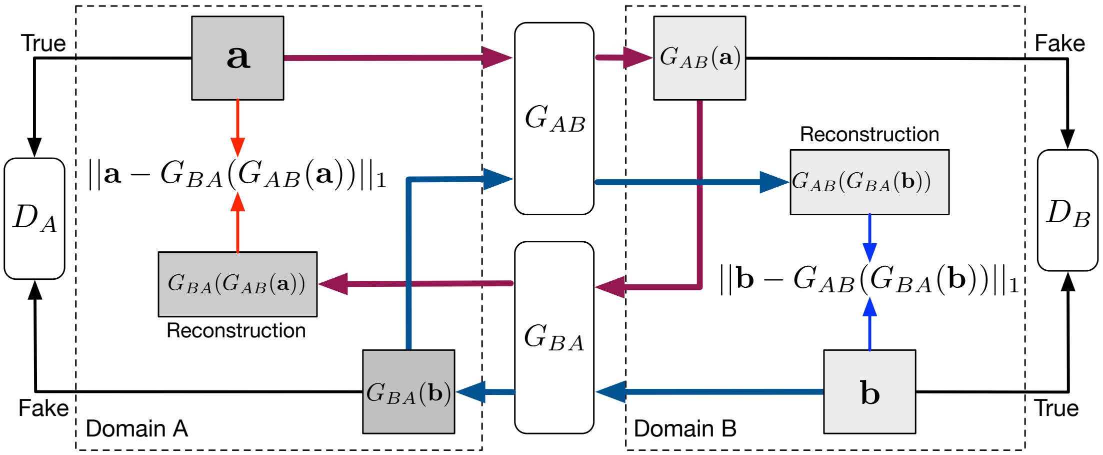
True
a
$G_{AB}(\mathbf{a})$
Fake
G_{AB}
Reconstruction
$G_{AB}(G_{BA}(\mathbf{b}))$
D_A
$||\mathbf{a} - G_{BA}(G_{AB}(\mathbf{a}))||_1$
D_B
$G_{BA}(G_{AB}(\mathbf{a}))$
$||\mathbf{b} - G_{AB}(G_{BA}(\mathbf{b}))||_1$
Reconstruction
G_{BA}
$G_{BA}(\mathbf{b})$
b
Fake
Domain A
Domain B
True

Fig. 8. Framework of CycleGAN and DualGAN. A and B are two different domains. There is a discriminator for each domain that judges if an image belong to that domain. Two generators are designed to translate an image from one domain to another. There are two cycles of data flow, the red one performs a sequence of domain transfer $A \to B \to A$, while the blue one is $B \to A \to B$. L1 loss is applied on the input $\mathbf{a}$ (or $\mathbf{b}$) and the reconstructed input $G_{BA}(G_{AB}(\mathbf{a}))$ (or $G_{AB}(G_{BA}(\mathbf{b}))$) to enforce self-consistency.

where λ is a hyper-parameter to balance the losses. Then the objective is to solve for:

$$G_{AB}^*, G_{BA}^* = \arg \min_{G_{AB}, G_{BA}} \max_{D_B, D_A} \mathcal{L}(G_{AB}, G_{BA}, D_A, D_B). \tag{14}$$

Although CycleGAN [73] and DualGAN [74] have the same objective, they use different implementations for generators. CycleGAN uses the generator structure as proposed in [68], while DualGAN follows the U-Net [71] structure as in [67] [71]. Both CycleGAN and DualGAN use the PatchGAN with size 70 × 70 as in [67].

Besides different generator architectures, CycleGAN [73] and DualGAN [74] also use different techniques to stabilize the training process. DualGAN follows the training procedure proposed in WGAN [8]. CycleGAN applies two techniques. First, instead of using the log loss [77] for $\mathcal{L}_{GAN}$ in Equation 11 with a least square loss that in practice performs more stably and produces higher quality images:

$$\begin{aligned}\mathcal{L}_{LSGAN}(G_{AB}, D_B) = \mathbb{E}_{\mathbf{b}\sim p_B(\mathbf{b})}[(D_B(\mathbf{b}) - 1)^2] \\ + \mathbb{E}_{\mathbf{a}\sim p_A(\mathbf{a})}[(D_B(G_{AB}(\mathbf{a}))^2].\end{aligned} \tag{15}$$

The second technique used in CycleGAN [73] is that, in order to reduce model oscillation [4], CycleGAN follows SimGAN's [78] strategy and updates discriminators D_A and D_B using a history of 50 previously generated images instead of the ones produced by latest generators.

Experiments of CycleGAN demonstrate the potential of performing high-quality image-to-image translation using unpaired data only, even though supervised method like Pix2Pix [67] still outperforms CycleGAN by a noticeable margin. CycleGAN also conducts experiments that show the importance of using both circles in the circle-consistency loss defined in Equation 12. However, failure cases provided in [73] show that, just as Pix2Pix [67], CycleGAN does not work in cases that necessitate geometric transformations, such as *apple* ↔ *orange* and *cat* ↔ *dog*. More examples are available on CycleGAN's project website[2].

5.4 Unsupervised Image-to-Image Translation with Distance Constraint

DistanceGAN [79] discovers that, the distance $||\mathbf{x}_i - \mathbf{x}_j||$ between two images in the source domain A is highly positively correlated to the distance of their counterparts $||G_{AB}(\mathbf{x}_i) - G_{AB}(\mathbf{x}_j)||$ in the target domain B, which can be seen from Figure 1 of the paper [79]. According to [79], let d_k be the distance $||\mathbf{x}_i - \mathbf{x}_j||$, and d'_k be $||G_{AB}(\mathbf{x}_i) - G_{AB}(\mathbf{x}_j)||$, a high correlation indicates that $\sum d_k d'_k$ should also be high. The pair-wise distances d_k in source domain are fixed, and maximizing $\sum d_k d'_k$ causes d_k with large value to dominate the loss, which is undesirable. So the authors propose to minimize $\sum |d_k - d'_k|$ instead.

DistanceGAN [79] proposes to use a pair-wise distance loss defined as:

$$\begin{aligned}\mathcal{L}_{dist}(G_{AB}, p_A) = \mathbb{E}_{\mathbf{x}_i,\mathbf{x}_j\sim p_A}|\frac{1}{\sigma_A}(||\mathbf{x}_i - \mathbf{x}_j||_1 - \mu_A) \\ - \frac{1}{\sigma_B}(||G_{AB}(\mathbf{x}_i) - G_{AB}(\mathbf{x}_j)||_1 - \mu_B)|,\end{aligned} \tag{16}$$

where μ_A, μ_B (σ_A, σ_B) are the pre-computed means (standard deviations) of pair-wise distances in training sets of domain A, B respectively.

In order to support stochastic gradient descent where only one data sample is fed into the model at a time, DistanceGAN proposes another self-distance constraint:

$$\begin{aligned}\mathcal{L}_{self\text{-}dist}(G_{AB}, p_A) &= \mathbb{E}_{\mathbf{x}\sim p_A}|\frac{1}{\sigma_A}(||L(\mathbf{x}) - R(\mathbf{x})||_1 - \mu_A) \\ &- \frac{1}{\sigma_B}(||G_{AB}(L(\mathbf{x})) - G_{AB}(R(\mathbf{x}))||_1 - \mu_B)|,\end{aligned} \tag{17}$$

where $L(\mathbf{x})$ and $R(\mathbf{x})$ indicate the left and right half of the image $\mathbf{x}$, and only the left (right) parts of images are taken into account when calculating μ_A, σ_A (μ_B, σ_B).

[2] https://junyanz.github.io/CycleGAN/

Thus the overall loss of DistanceGAN is given by:

$$\begin{aligned}\mathcal{L} = \alpha_{1A}\mathcal{L}_{GAN}(G_{AB}, D_B) + \alpha_{1B}\mathcal{L}_{GAN}(G_{BA}, D_A)\\ + \alpha_{2A}\mathcal{L}_{dist}(G_{AB}, p_A) + \alpha_{2B}\mathcal{L}_{dist}(G_{BA}, p_B)\\ + \alpha_{3A}\mathcal{L}_{self\text{-}dist}(G_{AB}, p_A) + \alpha_{3B}\mathcal{L}_{self\text{-}dist}(G_{BA}, p_B)\\ + \alpha_4\mathcal{L}_{cyc}(G_{AB}, G_{BA}),\end{aligned} \tag{18}$$

where $\mathcal{L}_{cyc}(G_{AB}, G_{BA})$ is defined in Equation 12 and $\mathcal{L}_{GAN}(\cdot)$ is defined in Equation 11.

DistanceGAN conducts extensive experiments using different losses: $\mathcal{L}_{cyc}$ alone (DiscoGAN [76] and CycleGAN [73]), one-sided distance loss $\mathcal{L}_{dist}$ ($A \to B \to A$ or $B \to A \to B$), the combination of $\mathcal{L}_{cyc}$ and one-sided $\mathcal{L}_{dist}$, and one-sided self-distance loss $\mathcal{L}_{self\text{-}dist}$ alone. The implementation of DistanceGAN [79] is the same as baseline (DiscoGAN [76] or CycleGAN [73]), dependent on which one is to be compared with. Results show that the one-sided distance loss or self-distance loss outperforms DiscoGAN and CycleGAN in several tasks, and that the combination of cyclic loss and distance loss achieves the best results in some cases. However, the paper does not explore other possible combinations, such as using both distance losses or even the full loss function defined in Equation 18.

One interesting thing is, as stated in [79], that DistanceGAN computes the distances in raw RGB space and still achieves better performance than baselines, but it may help if the distances are calculated in images' latent feature space where the features can be extracted using pre-trained image classifiers.

In DistanceGAN [79], the authors argue that the high positive correlation between d_k and d'_k implies that $\sum d_k d'_k$ should be high. However, it is unclear how high d'_k should be. For example, if $d_k = 1$ and we know that d'_k should be high, we still cannot definitely say that $d'_k = 3$ is more desirable than $d'_k = 2$. The concept of "high" is blurry, so it may be better to use the concept "higher". An alternative statement could be, let $\mathbf{x}, \mathbf{y}, \mathbf{z}$ be images from domain A, if $d(\mathbf{x}, \mathbf{y}) > d(\mathbf{x}, \mathbf{z})$, then $d(G_{AB}(\mathbf{x}), G_{AB}(\mathbf{y})) > d(G_{AB}(\mathbf{x}), G_{AB}(\mathbf{z}))$. And thus we can design a loss like $\max(\delta, d(G_{AB}(\mathbf{x}), G_{AB}(\mathbf{y})) - d(G_{AB}(\mathbf{x}), G_{AB}(\mathbf{z})))$ for all triplets $\mathbf{x}, \mathbf{y}, \mathbf{z} \in A$ such that $d(\mathbf{x}, \mathbf{y}) > d(\mathbf{x}, \mathbf{z})$.

Regardless of the objective of maximizing $\sum d_k d'_k$, the actual loss used in DistanceGAN, i.e. $\sum |d_k - d'_k|$, is forcing d'_k to be as close to d_k as possible. In other words, how two images of a domain differ from each other should be reflected in the same way when they are translated into another domain, which can be called "equivariance". The distance loss and self-distance loss proposed in DistanceGAN [79] is essentially capturing this "equivariance" property.

5.5 Unsupervised Image-to-Image Translation with Feature Constancy

As we mention previously, besides minimizing the reconstruction error at raw pixel level, we can also do this at higher feature level, which is explored in DTN [80]. The architecture of DTN is shown in Figure 9, where the generator G is composed of two neural networks, a convolutional network f and an transposed convolutional network g such that $G = g \circ f$.

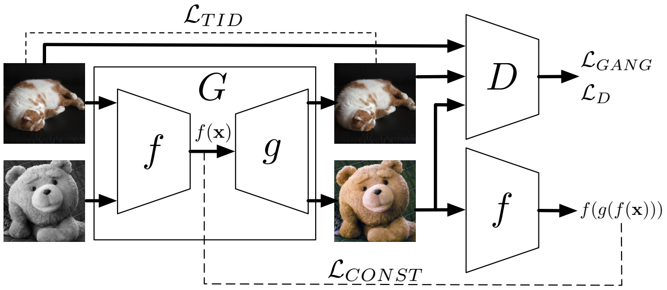

Fig. 9. Architecture of domain transfer network (DTN). As an example here, the model transfers images from BW color space to RGB color space. The generator is expected to be an identity matrix to images in the target domain, so that in this example, an RGB image remains unchanged when put into the generator, while a BW image is transformed into RGB by the generator.

Here f acts as a feature extractor, and DTN [80] tries to preserve high level features of an input image from source domain after it is transferred into target domain. Let X_s denote source domain and X_t be target domain. Given an input image $\mathbf{x} \in X_s$, the output of generator is $G(\mathbf{x}) = g(f(\mathbf{x}))$, then the feature reconstruction error can be defined with a distance measure d (DTN uses mean squared error (MSE)):

$$\mathcal{L}_{CONST} = \sum_{\mathbf{x} \in X_s} d(f(\mathbf{x}, f(g(f(\mathbf{x}))))). \tag{19}$$

Besides, DTN also expects the generator G to act as an identity matrix to images from the target domain. Given a distance measure d_2 (DTN uses mean squared error (MSE)), an identity mapping loss for target domain is defined as:

$$\mathcal{L}_{TID} = \sum_{\mathbf{x} \in X_t} d_2(\mathbf{x}, g(f(\mathbf{x}))). \tag{20}$$

In addition, DTN use a multi-class discriminator instead of a binary one in regular GAN. Here D is a ternary classification function that maps an image to one of the three classes {1,2,3}, where class 1 means that the image is from source domain but transformed by G, class 2 means that the input is an image from target domain but transformed by G, and class 3 means that the input image is from target domain without any transformation. Let $D_i(\mathbf{x})$ denotes the probability of $\mathbf{x}$ belongs to class i, the discriminator loss $\mathcal{L}_D$ is defined as:

$$\begin{aligned}\mathcal{L}_D = & -\mathbb{E}_{\mathbf{x} \in X_s} \log D_1(g(f(\mathbf{x})))\\ & -\mathbb{E}_{\mathbf{x} \in X_t} \log D_2(g(f(\mathbf{x}))) - \mathbb{E}_{\mathbf{x} \in X_t} \log D_3(\mathbf{x}).\end{aligned} \tag{21}$$

Similarly, the generator's adversarial loss $\mathcal{L}_{GANG}$ is defined as:

$$\begin{aligned}\mathcal{L}_{GANG} = & -\mathbb{E}_{\mathbf{x} \in X_s} \log D_3(g(f(\mathbf{x})))\\ & -\mathbb{E}_{\mathbf{x} \in X_t} \log D_3(g(f(\mathbf{x}))).\end{aligned} \tag{22}$$

In order to slightly smoothen the generated images, DTN adds an anisotropic total variation loss [81] $\mathcal{L}_{TV}$ defined on generated images $\mathbf{z} = [z_{ij}] = G(\mathbf{x})$:

$$\mathcal{L}_{TV}(\mathbf{z}) = \sum_{i,j}((\mathbf{z}_{i,j+1} - \mathbf{z}_{ij}^2 + (\mathbf{z}_{i+1,j} - \mathbf{z}_{ij})^2)^{\frac{1}{2}}. \tag{23}$$

Then the overall generator loss $\mathcal{L}_G$ is defined as:

$$\mathcal{L}_G = \mathcal{L}_{GANG} + \alpha\mathcal{L}_{CONST} + \beta\mathcal{L}_{TID} + \gamma\mathcal{L}_{TV}. \tag{24}$$

Experiments show that DTN [80] produces impressive images on face-to-emoji task, which is competitive with some existing emoji generating programs.

5.6 Unsupervised Image-to-Image Translation with Auxiliary Classifier

Bousmalis et. al [82] propose to use a task-specific auxiliary classifier to help unsupervised *image-to-image translation*. We denote their model as DAAC (short for "Domain Adaption with Auxiliary Classifier") here for convenience. DAAC [82] contains a task-specific classifier $C(\mathbf{x}) \rightarrow \mathbf{y}$ that assigns a label vector $\mathbf{y}$ to an image in either source domain or target domain. This classifier C, for example, can be an image classification model. The architecture of DAAC is similar to Figure 2.

Given a training set $\mathbf{X}_s$ in which each image $\mathbf{x} \in \mathbf{X}_s$ has a class label $\mathbf{y}_\mathbf{x}$, the objective of C is to minimize the cross-entropy loss $\mathcal{L}_C$:

$$\mathcal{L}_c(G,C) = \mathbb{E}_{\mathbf{x}\in\mathbf{X}_s}[-\mathbf{y}_\mathbf{x}^\intercal \log[C(G(\mathbf{x}))] - \mathbf{y}_\mathbf{x}^\intercal \log[C(\mathbf{x})]]. \tag{25}$$

Besides, DAAC [82] also proposes a content-similarity loss in cases with prior knowledge on what information should be preserved after the domain adaption process. For example, we may expect the hues of the source image and the adapted image to be the same. In [82], the authors consider the case where they render objects with black background and expect the adapted images to have the same objects but different backgrouds. In this case, each image $\mathbf{x}$ is associated with a binary mask $\mathbf{m}_\mathbf{x} \in \mathbb{R}^k$ (k is the number of pixels in $\mathbf{x}$) to separate foreground and background, and the content-similarity loss $\mathcal{L}_s$, which is a variation of the pairwise mean squared error (PMSE) [83], can be defined as:

$$\begin{aligned}\mathcal{L}_s(G) = \mathbb{E}_{\mathbf{x}\in\mathbf{X}_s}[&\frac{1}{k}||(\mathbf{x}-G(\mathbf{x}))\circ\mathbf{m}_\mathbf{x}||_2^2 \\ &-\frac{1}{k^2}((\mathbf{x}-G(\mathbf{x}))^\intercal\mathbf{m}_\mathbf{x})^2],\end{aligned} \tag{26}$$

where $||\cdot||_2^2$ is the squared L2 norm, and $\circ$ is the Hardamard product. Note that here the image x is flattened into a vector form.

The total objective is thus given by:

$$\arg\min_{G,C}\max_{D} \alpha\mathcal{L}_{GAN}(G,D) + \beta\mathcal{L}_c(G,C) + \gamma\mathcal{L}_s(G), \tag{27}$$

where $\mathcal{L}_{GAN}(G,D)$ is the regular GAN objective defined in Equation 1, and α,β,γ are hyper-parameters to balance different terms of the objective.

Although $\mathcal{L}_c(G,C)$ and $\mathcal{L}_s(G)$ help in generating better adapted images, the amount of labeled images is limited in real world, and fine-grained pixel-level masks are even harder to obtain. Nonetheless, using auxiliary classifier can also help to avoid *mode collapse*, as in AC-GAN [33].

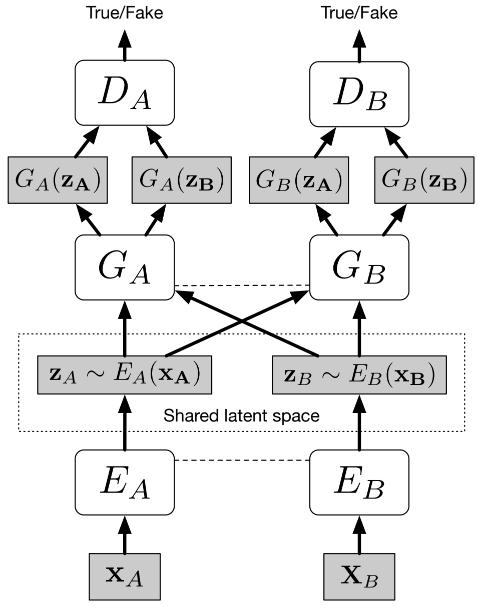

Fig. 10. Architecture of UNIT. The two encoders share weights at the last few layers, while the two generators share weights at the first few layers, as indicated by the dashed lines.

5.7 Unsupervised Image-to-Image Translation with VAE and Weight Sharing

UNIT [84] proposes to add VAE to CoGAN [85] for unsupervised image-to-image translation, as illustrated in Figure 10. In addition, UNIT assumes that both encoders share the same latent space, which means, let $\mathbf{x}_A, \mathbf{x}_B$ be the same image in different domains, and then the shared latent space implies that $E_A(\mathbf{x}_A) = E_B(\mathbf{B}_B)$. Based on the shared-latent space assumption, UNIT enforces weight sharing between the last few layers of the encoders and between the first few layers of the generators. The objective function for UNIT is a combination of the objectives of GAN and VAE, with the difference of using two sets of GANs/VAEs and adding hyper-parameters λs to balance different loss terms. Also, UNIT states that the shared latent space assumption implies the cycle-consistency [73] [76] [74], so it adds another constraint to its objective which is VAE-like:

$$\begin{aligned}\mathcal{L}_{cc1} =& \lambda_3 D_{KL}(q_A(\mathbf{z}_A|\mathbf{x}_A)||p_\eta(\mathbf{z})) \\ &+ \lambda_3 D_{KL}(q_B(\mathbf{z}_B|F_{AB}(\mathbf{x}_A))||p_\eta(\mathbf{z})) \\ &- \lambda_4 \mathbb{E}_{\mathbf{z}_B \sim q_B(\mathbf{z}_B|F_{AB}(\mathbf{x}_A))}[\log p_{G_A}(\mathbf{x}_A|\mathbf{z}_B)],\end{aligned} \tag{28}$$

where $q_A(\mathbf{z}_A|\mathbf{x}_A) = \mathcal{N}(\mathbf{z}_A|E_A(\mathbf{x}_A),\mathbf{I})$, $q_B(\mathbf{z}_A|\mathbf{x}_B) = \mathcal{N}(\mathbf{z}_B|E_A(\mathbf{x}_B),\mathbf{I})$, $p_\eta(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0,\mathbf{I})$, and $F_{AB}(\mathbf{x}) = G_B(E_A(\mathbf{x}))$. And $\mathcal{L}_{cc2}$ can be defined similarly by reversing subscripts of A and B.

Although UNIT performs better than models like DTN [80] and CoGAN [85] on MNIST [41], Street View House Number (SVHN) [86] datasets in terms of cross-domain classification accuracy, it does not compare with other unsupervised methods such as CycleGAN [73] and DiscoGAN [76], nor does it use other widely adopted evaluation metrics like *Inception Score*.

5.8 Unsupervised Multi-domain Image-to-Image Translation

Previous models can only transform images between two domains, but if we want to transform an image among several domains, we need to train a separate generator for each pair of domains, which is costly. To deal with this problem, StarGAN [87] proposes to use one generator that can generate images of all domains. Instead of taking only an image as conditional input, StarGAN also takes the label of target domain as input, and the generator is designed to transform the input image to the target domain indicated by the input label. Similar to DAAC [82] and AC-GAN [33], StarGAN uses an auxiliary domain classifier which classifies an image into its belonged domain. In addition, a cycle-consistency loss [73] is used to preserve the content similarity between input and output images. In order to allow StarGAN to train on multiple datasets that may have different sets of labels, StarGAN uses an additional one-hot vector to indicate the datset and concatenates all label vectors into one vector, setting the unspecified labels as zero for each dataset.

5.9 Summary on General Image-to-Image Translation

So far we have discussed some general *image-to-image translation* methods, the different losses they use are summarized in Table 1.

The simplest loss is the *pixel-wise L1 reconstruction loss* operates in the target domain, which requires paired training samples. Both *one-sided* and *bi-directional* reconstruction loss can treated as the unsupervised version of *pixel-wise L1 reconstruction loss*, since they enforce cycle-consistency and do not require paired training samples. The additional *VAE loss* is based on the assumption of shared latent space of both source and target domain, and it also implies the bi-directional cycle-consistency loss. The *equivariance loss*, however, does not try to reconstruct images, but to preserve the difference between images across source and target domain.

Different from previously mentioned losses that work on the generators directly, *pair-wise discriminator loss*, *ternary discriminator loss* and *auxiliary classifier loss* work on the discriminator side and make the discriminator better at distinguishing real and fake samples, and then the generator can also learn better with the enhanced discriminator.

Among all mentioned models, Pix2Pix [67] produces sharpest images, even though the L1 loss is just a simple add-on component to the original GAN model. It may be interesting to combine L1 loss with the pair-wise discriminator in PLDT [35] which may improve the model's performance on image-to-image translations that involve geometric changes on images. Also, Pix2Pix may benefit from preserving similarity information between images in source and target domains, as done in some unsupervised methods like CycleGAN [73] and DistanceGAN [79]. As for unsupervised methods, although their results are not as sharp as supervised methods like Pix2Pix [67], they are a promising research direction, since they do not require paired data and collecting labeled data is very costly in real world.

TABLE 1
Summary of losses used in image *image-to-image translation*, “sup” is short for “supervision”. “$\surd$” in both “source domain” and “target domain” columns indicates that the loss requires images from both domains (but not necessarily paired).

Loss	**Source domain**	**Target domain**	**Sup**	**Models**
Pixel-level L1		$\surd$	$\surd$	Pix2Pix [67]
Pair-wise discriminator		$\surd$	$\surd$	PLDT [35]
Bi-directional reconstruction	$\surd$	$\surd$		CycleGAN [73], DualGAN [74], DiscoGAN [76], StarGAN [87]
One-sided reconstruction	$\surd$			DTN [80]
Ternary discriminator		$\surd$		DTN [80]
Equivariance	$\surd$	$\surd$		DistanceGAN [79]
Auxiliary classifier	$\surd$	$\surd$		DAAC [82], StarGAN [87]
VAE	$\surd$	$\surd$		UNIT [84]

5.10 Task-Specific Image-to-Image Translation

In this section, we are going to introduce some models that work on more specific *image-to-image translation* applications.

5.10.1 Face Editing

Face editing is closely related to *image-to-image translation*, since it also takes images as input and produces images. However, *Face editing* focuses more on manipulating the attributes of humans' faces while *image-to-image translation* is a more general scope.

IcGAN [43] proposes to learn two separate encoders, $E_{\mathbf{z}}$ that maps an image to its latent vector $\mathbf{z}$ and $E_{\mathbf{y}}$ that learns the attribute information vector $\mathbf{y}$. Attribute manipulation is performed by tuning the attribute vector $\mathbf{y}$, concatenating it with $\mathbf{z}$ and then putting the combined vector as input to the generator.

Instead of generating images directly, Shen et. al [88] propose to learn *residual* images of attributes. To add an attribute to an image $\mathbf{x}$, a residual image $G(\mathbf{x})$ is obtained by putting it through a generator G, and then the manipulated image is obtained by $\mathbf{x}+G(\mathbf{x})$. The framework of this model follows the *dual learning* approach we discuss earlier, and the discriminator is similar to the ternary classifier in DTN [80], which distinguishes whether the image is from ground-truth or is manipulated by either of the two generators (G_{AB}, G_{BA}).

Recently, Brock et. al propose Introspective Adversarial Network (IAN) [89] for neural photo editing, which, similar to VAE-GAN [38], is also a combination of GAN [4] and VAE [39]. However, unlike VAE-GAN that uses different networks for discriminator and encoder, IAN combines the discriminator and encoder into a single network. The intuition behind it is that features learned by a well trained discriminator tend to be more expressive than those learned by maximum likelihood, which is the reason why they are more suitable for inference. Specifically, the encoder is implemented as a fully-connected layer on top of the last convolution layer of the discriminator.

Besides manipulating face attributes, there are also other forms of "face editing" such as generating the frontal view of a person's face based on pictures of the face's side view. Huang et. al propose Two-Pathway Generative Adversarial Network (TP-GAN) [90] that performs such task. TP-GAN consists of two pathways, a global structure pathway and a local texture pathway, where each pathway contains a pair of encoder and decoder. The global pathway constructs a blurry frontal view that captures the global structure of the face, while the local pathway with four sub-networks attends to local texture details around four facial landmarks, which are *left eye center*, *right eye center*, *nose tip* and *mouth center*. Outputs of four sub-networks in the local pathway are concatenated with the output of global pathway, and then the combined tensor is fed into successive convolution layers to produce the final synthetic image. In addition to L1 pixel loss, TP-GAN proposes to use a symmetry loss to constrain the symmetry of human faces, and an identity preserving loss to preserve the identity of input face. The identity loss is implemented as a perceptual loss [68].

5.10.2 Image Super-Resolution

Another application of GAN that is related to *image-to-image translation* is image super-resolution, which is the task of taking a low resolution image as input and outputs a high resolution one with sharp details. SRGAN [91] proposes to use a residual block [92] based generator and a fully convolutional discriminator [5] to do single image super-resolution. Besides adversarial loss, SRGAN also combines pixel-wise MSE loss, perceptual loss [68] and regularization loss [68]. SRGAN outperforms several baselines on some metrics by a small margin, but the difference in synthetic images is not easy to tell without zooming in.

5.10.3 Video Prediction

A special kind of related application is *video prediction*, which aims to predict the next frame of a video given the current frame (or a history of frames).

VGAN [93] proposes to use a hierarchical model for video prediction. The generator has two data streams, a foreground stream f consisting of 3D transposed convolutions, and a background stream b consisting of 2D transposed convolutions. The foreground stream is also responsible for generating a mask m that is used to merged the results of two streams: $m \odot f + (1-m) \odot b$. The discriminator is a set of spatial-temporal convolution layers.

Mathieu et. al propose Adv-GDL [94] which generates future frames in an iterative way, from low to high resolution. Let $s_1, s_2, ..., s_k$ be a set of sizes and u_k be the upsampling operator from size s_{k-1} to s_k. Let $\mathbf{X}_k, \mathbf{Y}_k$ be ground-truth current and future frames of size s_k and G_k be the generator that produces images of size s_k,then the predicted future frame $\hat{\mathbf{Y}}_k$ is calculated by $\hat{\mathbf{Y}}_k = u_k(\hat{\mathbf{Y}}_{k-1}) + G_k(\mathbf{X}_k, u_k(\hat{\mathbf{Y}}_{k-1}))$. The discriminator is a series of multi-scale convolutional networks with scalar output.

Although both methods produce reasonable output videos when the time interval is short, video quality becomes worse as the time increases. Objects in synthetic videos lose their original shapes and get morphed to indistinguishable objects in some cases, which may be due to the models' inability to learn legal movements of objects.

6 Evaluation Metrics on Synthetic Images

It is very hard to quantify the quality of synthetic images, and metrics like RMSE are not suitable since there is no absolute one-to-one correspondence between synthetic and real images. A commonly used subjective metric is to use the Amazon Mechanical Turk (AMT)[3] that hires humans to score synthetic and real images according to how realistic they think the images are. However, people often have different opinions of what is good or bad, so we also need objective metrics to evaluate the quality of images.

Inception score (IS) [6] evaluates an image based on the entropy in class probability distribution when it is put into a pre-trained image classifier. One intuition behind *Inception score* is that the better an image $\mathbf{x}$ is, the lower the entropy of conditional distribution $p(y|\mathbf{x})$ should be, which means the classifier have high confidence of what the image is about. Also, to encourage the model to generate various classes of images, the marginal distribution $p(y) = \int p(y|\mathbf{x} = G(\mathbf{z}))d\mathbf{z}$ should have high entropy. Combining these two intuition, the *Inception score* is calculated by $\exp(\mathbb{E}_{\mathbf{x}\sim G(\mathbf{z})} D_{KL}(p(y|\mathbf{x})||p(y))$. As discussed in [95], *Inception score* is neither sensitive to prior distribution of labels, nor a proper distance measure. Also, *Inception score* suffers from *intra-class mode collapse*, since a model only needs to generate one perfect sample for each class to get a perfect *Inception score*.

Similar to *Inception score*, *FCN-score* [67] adopts the idea that if the synthetic images are realistic, classifiers trained on real images will be able to classify the synthetic images correctly. However, an image classifier does not require the input image to be very sharp as to give a correct classification, which means that metrics based on image classifier may not be able to tell between two images with only small difference in details. Worse still, research in adversarial examples [96] shows that the decision of a classifier does not necessarily depend on visual content of images but can be highly influenced by noise invisible to humans, which raises more questions on this metric.

Fréchet Inception Distance (FID) [97] provides a different approach. First, generated images are embedded into a latent feature space of a chosen layer of the Inception Net. Second, embeddings of generated and real images are treated as samples from two continuous multivariate Gaussians so that their means and covariances can be calculated. Then the quality of generated images can be determined by the Fréchet Distance between the two Gaussians:

$$\text{FID}(x, g) = ||\mu_x - \mu_g||_2^2 + \text{Tr}(\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g)^{\frac{1}{2}}), \quad (29)$$

where (μ_x, μ_g) and (Σ_x, Σ_g) are the means and covariances of the samples from the true data distribution and generator's learned distribution respectively. The authors of [97] show that FID is consistent with human judgment and that there is a strong negative correlation between FID and the quality of generated images. Furthermore, FID is less sensitive to noise than IS and can detect intra-class mode collapse.

Besides *Inception score* (IS), *FCN-score* and *Fréchet Inception Distance* (FID), there are also other metrics like *Gaussian Parzen Window* [4], *Generative Adversarial Metric* (GAM) [48]

[3] https://www.mturk.com/

and *MODE Score* [7]. Among all these metrics, *Inception score* is the most widely adopted one for quantitatively evaluating synthetic images, and there is a recent study [98] that uses IS to compare several GAN models. Although FID is relatively new, it has been shown to be better than IS [97] [95].

7 Discriminators as Learned Loss Functions

Generative adversarial network (GAN) is powerful and effective in that the discriminator acts as a learned loss function instead of a fixed one designed carefully for each specific task. This is particularly important for image synthesis tasks whose loss functions are hard to be explicitly defined in math. For example, in style transfer task, it is hard to write down a math equation that evaluates how well an image matches a certain painting style. For image synthesis tasks, each input may have many legal outputs, but samples in training set cannot cover all situations. In this case, it is inappropriate to only minimize the distance between synthetic and ground-truth images, since we want the generator to learn the data distribution instead of remembering training samples. Although we can design feature-based losses that try to preserve feature consistency instead of at raw pixel level, as done in the *perceptual loss* [68] for image style, such losses are constrained by pre-trained image classification models they use, and it remains a question of which layers to pick for calculating feature loss when we switch to another pre-trained model. A discriminator, on the other hand, does not require explicit definition of the loss, since it learns how to evaluate a data sample as it trains against the generator. Thus the discriminator is able to learn a better loss function given enough training data.

The fact that the discriminator acts as a learned loss function has significant meaning for general artificial intelligence. Traditional pattern recognition and machine learning require us to define what features to be used (e.g. SIFT [99] and HOG [100] descriptors), and we design specific loss functions and decide what optimization methods to be applied. Deep learning free us from carefully designing features, by learning low-level and high-level feature representations by itself during training (e.g. CNN kernels), but we still need to work hard at designing loss functions that work well. GAN takes us one step forward on our path towards artificial intelligence, in that it learns how to evaluate data samples instead of being told how to do so, although we still need to design the adversarial loss and combine it with other auxiliary losses. In other words, previously we design how to calculate how close an output is to the corresponding ground-truth ($\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}})$), but the discriminator learns how to calculate how well an output matches the true data distribution ($\mathcal{L}(\mathbf{x})$). Such property allows models to be more flexible and more likely to generalize well. Furthermore, with learn2learn [101] which allows neural networks to learn to optimize themselves, there is a possibility that we may no longer need to choose what optimizers (such as RMSprop [102], Adam [103] etc.) to use and let models handle everything themselves.

8 Discussion and Conclusion

In this paper, we review some basics of Generative Adversarial Nets (GAN) [4], and classify image synthesis methods into three main approaches, i.e. *direct method*, *hierarchical method* and *iterative method*, and mention some other generation methods such as iterative sampling [50]. We also discuss in details two main forms of image synthesis, i.e. *text-to-image synthesis* and *image-to-image translation*.

For *text-to-image synthesis*, current methods work well on datasets where each image contains single object such as CUB [57] and Oxford-102 [55], but the performance on complex datasets such as MSCOCO [61] is much worse. Although some models can produce realistic images of rooms in LSUN [64], it should be noted that rooms do not contain living things, and a living thing is certainly much more complicated than static objects. This limitation probably stems from the models' inability to learn different concepts of objects. We also propose that one possible way to improve GAN's performance in this task is to train different models that generate single object well and train another model that learns to combine different objects according to text descriptions, and that CapsNet [66] may be useful in such tasks.

For *image-to-image translation*, we review some general methods from supervised to unsupervised settings, such as pixel-wise loss [67], cyclic loss [73] and self-distance loss [79]. Besides, we also introduce some task-specific image-to-image translation models for face editing, video prediction and image super-resolution. Image-to-image translation is certainly an interesting application of GAN, which has great potential to be incorporated into other software products, especially mobile apps. Although research in unsupervised methods seems more popular, supervised methods may be more practical since they still produce better synthetic images than unsupervised methods.

Finally, we review some evaluation metrics for synthetic images, and discuss GAN's role on our path towards artificial intelligence. The power of GAN largely lies in its discriminator's acting as a learned loss function, which makes the model perform better on tasks whose output is hard to evaluate by designing an explicit math equation.

Acknowledgments

He Huang would like to thank Chenwei Zhang and Bokai Cao for the valuable discussions and feedback.

References

[1] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] N. Brown and T. Sandholm, "Safe and nested subgame solving for imperfect-information games," *arXiv preprint arXiv:1705.02955*, 2017.

[4] 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*, 2014, pp. 2672–2680.

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

[6] 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*, 2016, pp. 2226–2234.

[7] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li, "Mode regularized generative adversarial networks," *arXiv preprint arXiv:1612.02136*, 2016.

[8] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[9] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gan," *arXiv preprint arXiv:1704.00028*, 2017.

[10] D. Berthelot, T. Schumm, and L. Metz, "Began: Boundary equilibrium generative adversarial networks," *arXiv preprint arXiv:1703.10717*, 2017.

[11] 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*, 2016, pp. 2172–2180.

[12] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," *arXiv preprint arXiv:1609.03126*, 2016.

[13] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient." in *AAAI*, 2017, pp. 2852–2858.

[14] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[15] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *arXiv preprint arXiv:1710.07035*, 2017.

[16] R. A. Yeh, C. Chen, T. Lim, M. Hasegawa-Johnson, and M. N. Do, "Semantic image inpainting with perceptual and contextual losses," *CoRR*, vol. abs/1607.07539, 2016. [Online]. Available: http://arxiv.org/abs/1607.07539

[17] T.-H. Chen, Y.-H. Liao, C.-Y. Chuang, W.-T. Hsu, J. Fu, and M. Sun, "Show, adapt and tell: Adversarial training of cross-domain image captioner," *arXiv preprint arXiv:1705.00930*, 2017.

[18] X. Liang, Z. Hu, H. Zhang, C. Gan, and E. P. Xing, "Recurrent topic-transition gan for visual paragraph generation," *arXiv preprint arXiv:1703.07022*, 2017.

[19] W. Zhao, W. Xu, M. Yang, J. Ye, Z. Zhao, Y. Feng, and Y. Qiao, "Dual learning for cross-domain image captioning," in *CIKM*, 11 2017, pp. 29–38.

[20] X. Wang, A. Shrivastava, and A. Gupta, "A-fast-rcnn: Hard positive generation via adversary for object detection," *arXiv preprint arXiv:1704.03414*, 2017.

[21] P. Luc, C. Couprie, S. Chintala, and J. Verbeek, "Semantic segmentation using adversarial networks," *arXiv preprint arXiv:1611.08408*, 2016.

[22] S. Rajeswar, S. Subramanian, F. Dutil, C. J. Pal, and A. C. Courville, "Adversarial generation of natural language," *CoRR*, vol. abs/1705.10929, 2017. [Online]. Available: http://arxiv.org/abs/1705.10929

[23] J. Guo, S. Lu, H. Cai, W. Zhang, Y. Yu, and J. Wang, "Long text generation via adversarial training with leaked information," *CoRR*, vol. abs/1709.08624, 2017. [Online]. Available: http://arxiv.org/abs/1709.08624

[24] J. Li, W. Monroe, T. Shi, A. Ritter, and D. Jurafsky, "Adversarial learning for neural dialogue generation," *arXiv preprint arXiv:1701.06547*, 2017.

[25] Z. Yang, J. Hu, R. Salakhutdinov, and W. W. Cohen, "Semi-supervised qa with generative domain-adaptive nets," *arXiv preprint arXiv:1702.02206*, 2017.

[26] Z. Yang, W. Chen, F. Wang, and B. Xu, "Improving neural machine translation with conditional sequence generative adversarial nets," *arXiv preprint arXiv:1703.04887*, 2017.

[27] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[28] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[29] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," *arXiv preprint arXiv:1612.03242*, 2016.

[30] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, "Generative adversarial text to image synthesis," *arXiv preprint arXiv:1605.05396*, 2016.

[31] S. E. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee, "Learning what and where to draw," in *Advances in Neural Information Processing Systems*, 2016, pp. 217–225.

[32] A. Odena, "Semi-supervised learning with generative adversarial networks," *arXiv preprint arXiv:1606.01583*, 2016.

[33] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier gans," *arXiv preprint arXiv:1610.09585*, 2016.

[34] A. Dash, J. C. B. Gamboa, S. Ahmed, M. Z. Afzal, and M. Liwicki, "Tac-gan-text conditioned auxiliary classifier generative adversarial network," *arXiv preprint arXiv:1703.06412*, 2017.

[35] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. S. Kweon, "Pixel-level domain transfer," in *European Conference on Computer Vision*. Springer, 2016, pp. 517–532.

[36] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," *arXiv preprint arXiv:1605.09782*, 2016.

[37] V. Dumoulin, I. Belghazi, B. Poole, A. Lamb, M. Arjovsky, O. Mastropietro, and A. Courville, "Adversarially learned inference," *arXiv preprint arXiv:1606.00704*, 2016.

[38] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," *arXiv preprint arXiv:1512.09300*, 2015.

[39] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[40] E. L. Denton, S. Chintala, a. szlam, and R. Fergus, "Deep generative image models using a laplacian pyramid of adversarial networks," in *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015, pp. 1486–1494.

[41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[42] S. Nowozin, B. Cseke, and R. Tomioka, "f-gan: Training generative neural samplers using variational divergence minimization," *arXiv preprint arXiv:1606.00709*, 2016.

[43] G. Perarnau, J. van de Weijer, B. Raducanu, and J. M. Álvarez, "Invertible conditional gans for image editing," *arXiv preprint arXiv:1611.06355*, 2016.

[44] X. Wang and A. Gupta, "Generative image modeling using style and structure adversarial networks," *arXiv preprint arXiv:1603.05631*, 2016.

[45] J. Yang, A. Kannan, D. Batra, and D. Parikh, "Lr-gan: Layered recursive generative adversarial networks for image generation," *arXiv preprint arXiv:1703.01560*, 2017.

[46] P. J. Burt and E. H. Adelson, "The laplacian pyramid as a compact image code," in *Readings in Computer Vision*. Elsevier, 1987, pp. 671–679.

[47] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie, "Stacked generative adversarial networks," *arXiv preprint arXiv:1612.04357*, 2016.

[48] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic, "Generating images with recurrent adversarial networks," *CoRR*, vol. abs/1602.05110, 2016. [Online]. Available: http://arxiv.org/abs/1602.05110

[49] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," *arXiv preprint arXiv:1502.04623*, 2015.

[50] A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, and J. Clune, "Plug & play generative networks: Conditional iterative generation of images in latent space," *arXiv preprint arXiv:1612.00005*, 2016.

[51] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 3387–3395.

[52] 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*. ACM, 2008, pp. 1096–1103.

[53] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[54] S. Reed, Z. Akata, B. Schiele, and H. Lee, "Learning deep representations of fine-grained visual descriptions," *arXiv preprint arXiv:1605.05395*, 2016.

[55] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[56] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2017–2025.

[57] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," 2011.

[58] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis," in *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition*, 2014, pp. 3686–3693.

[59] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas, "Stackgan++: Realistic image synthesis with stacked generative adversarial networks," *CoRR*, vol. abs/1710.10916, 2017. [Online]. Available: http://arxiv.org/abs/1710.10916

[60] T. Xu, P. Zhang, Q. Huang, H. Zhang, Z. Gan, X. Huang, and X. He, "Attngan: Fine-grained text to image generation with attentional generative adversarial networks," *arXiv preprint arXiv:1711.10485*, 2017.

[61] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[62] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625–2634.

[63] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[64] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.

[65] G. E. Hinton, A. Krizhevsky, and S. D. Wang, "Transforming auto-encoders," in *International Conference on Artificial Neural Networks*. Springer, 2011, pp. 44–51.

[66] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," *arXiv preprint arXiv:1710.09829v2*, 2017.

[67] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *arXiv preprint arXiv:1611.07004*, 2016.

[68] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*. Springer, 2016, pp. 694–711.

[69] Y. Jing, Y. Yang, Z. Feng, J. Ye, and M. Song, "Neural style transfer: A review," *CoRR*, vol. abs/1705.04058, 2017. [Online]. Available: http://arxiv.org/abs/1705.04058

[70] H. Zhao, O. Gallo, I. Frosio, and J. Kautz, "Loss functions for neural networks for image processing," *CoRR*, vol. abs/1511.08861, 2015. [Online]. Available: http://arxiv.org/abs/1511.08861

[71] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[72] C. Li and M. Wand, "Precomputed real-time texture synthesis with markovian generative adversarial networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 702–716.

[73] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *arXiv preprint arXiv:1703.10593*, 2017.

[74] Z. Yi, H. Zhang, P. T. Gong *et al.*, "Dualgan: Unsupervised dual learning for image-to-image translation," *arXiv preprint arXiv:1704.02510*, 2017.

[75] D. He, Y. Xia, T. Qin, L. Wang, N. Yu, T. Liu, and W.-Y. Ma, "Dual learning for machine translation," in *Advances in Neural Information Processing Systems*, 2016, pp. 820–828.

[76] T. Kim, M. Cha, H. Kim, J. Lee, and J. Kim, "Learning to discover cross-domain relations with generative adversarial networks," *arXiv preprint arXiv:1703.05192*, 2017.

[77] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2017, pp. 2813–2821.

[78] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," *arXiv preprint arXiv:1612.07828*, 2016.

[79] S. Benaim and L. Wolf, "One-sided unsupervised domain mapping," *arXiv preprint arXiv:1706.00826*, 2017.

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

[81] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5188–5196.

[82] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, "Unsupervised pixel-level domain adaptation with generative adversarial networks," *arXiv preprint arXiv:1612.05424*, 2016.

[83] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in neural information processing systems*, 2014, pp. 2366–2374.

[84] M.-Y. Liu, T. Breuel, and J. Kautz, "Unsupervised image-to-image translation networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 700–708.

[85] M.-Y. Liu and O. Tuzel, "Coupled generative adversarial networks," in *Advances in neural information processing systems*, 2016, pp. 469–477.

[86] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS workshop on deep learning and unsupervised feature learning*, vol. 2011, no. 2, 2011, p. 5.

[87] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "Stargan: Unified generative adversarial networks for multi-domain image-to-image translation," *arXiv preprint arXiv:1711.09020*, 2017.

[88] W. Shen and R. Liu, "Learning residual images for face attribute manipulation," *CoRR*, vol. abs/1612.05363, 2016. [Online]. Available: http://arxiv.org/abs/1612.05363

[89] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Neural photo editing with introspective adversarial networks," *arXiv preprint arXiv:1609.07093*, 2016.

[90] R. Huang, S. Zhang, T. Li, and R. He, "Beyond face rotation: Global and local perception GAN for photorealistic and identity preserving frontal view synthesis," *CoRR*, vol. abs/1704.04086, 2017. [Online]. Available: http://arxiv.org/abs/1704.04086

[91] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," *CoRR*, vol. abs/1609.04802, 2016. [Online]. Available: http://arxiv.org/abs/1609.04802

[92] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[93] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," in *Advances In Neural Information Processing Systems*, 2016, pp. 613–621.

[94] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," *arXiv preprint arXiv:1511.05440*, 2015.

[95] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, "Are GANs Created Equal? A Large-Scale Study," *ArXiv e-prints*, Nov. 2017.

[96] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[97] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a nash equilibrium," *CoRR*, vol. abs/1706.08500, 2017. [Online]. Available: http://arxiv.org/abs/1706.08500

[98] W. Fedus, M. Rosca, B. Lakshminarayanan, A. M. Dai, S. Mohamed, and I. Goodfellow, "Many Paths to Equilibrium: GANs

Do Not Need to Decrease a Divergence At Every Step," *ArXiv e-prints*, Oct. 2017.

[99] D. G. Lowe, "Object recognition from local scale-invariant features," in *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2. Ieee, 1999, pp. 1150–1157.

[100] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[101] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," in *Advances in Neural Information Processing Systems*, 2016, pp. 3981–3989.

[102] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[103] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

He Huang is a second-year PhD student from the department of Computer Science, University of Illinois at Chicago, USA. He received his bachelor's degree in Software Engineering from Sun Yat-sen University, China. His research interests are GAN, computer vision and data mining, especially cross-model tasks (such as image captioning, cross-model retrieval and text-to-image). He is a Bayesian person, and loves probabilistic graphical models for their elegance.

Phillip S. Yu is a Disthinguished Professor in the Department of Computer Science at UIC and also holds the Wexler Chair in Information and Technology. Before joining UIC, he was with IBM Thomas J. Watson Research Center, where he was manager of the Software Tools and Techniques department. His main research interests include big data, data mining (especially on graph/network mining), social network, privacy preserving data publishing, data stream, database systems, and Internet applications and technologies.

Changhu Wang is currently a technical director of Toutiao AI Lab, Beijing, China. He is leading the visual computing team in Toutiao AI Lab, working on computer vision, multimedia analysis, and machine learning. Before joining Toutiao AI Lab, he worked as a lead researcher in Microsoft Research from 2009 to 2017. He was a research engineer at the department of Electrical and Computer Engineering in National University of Singapore in 2008.