MIXGAN: Learning Concepts from Different Domains for Mixture Generation

Guang-Yuan Hao[1], **Hong-Xing Yu**[1,4], **Wei-Shi Zheng**[1,2,3] *
[1] School of Data and Computer Science, Sun Yat-sen University, China
[2] Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education, China
[3] Collaborative Innovation Center of High Performance Computing, NUDT, China
[4] Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, China
guangyuanhao@outlook.com, xKoven@gmail.com, wszheng@ieee.org

Abstract

In this work, we present an interesting attempt on mixture generation: absorbing different image concepts (e.g., content and style) from different domains and thus generating a new domain with learned concepts. In particular, we propose a mixture generative adversarial network (MIXGAN). MIXGAN learns concepts of content and style from two domains respectively, and thus can join them for mixture generation in a new domain, i.e., generating images with content from one domain and style from another. MIXGAN overcomes the limitation of current GAN-based models which either generate new images in the same domain as they observed in training stage, or require off-the-shelf content templates for transferring or translation. Extensive experimental results demonstrate the effectiveness of MIXGAN as compared to related state-of-the-art GAN-based models.

1 Introduction

When you are looking at a red T-shirt in eBay, you could easily imagine how you would look like when you wear it: you know well the shape of your body, you have an image of this red T-shirt in your mind, and thus you can wear it in your imaginary world. However, can a learning machine do a job like this? This means that the machine should have the ability to learn from different domains (people and T-shirts, in the running example) and extract some specific concepts from them, respectively (people's body shapes and T-shirts' color style). Then, it is expected to join the specific kinds of concepts and thereby generate a new domain (imagination on wearing the T-shirt). In realistic applications, this generation strategy might be more desirable in some scenarios other than the conventional generation strategy where only one training domain exists and the generated domain is expected to be the same as the existing one. We show another example in Figure 1. Here we aim to generate images in a new domain (i.e. colorful bags), given the content of one domain (the shape of the bags) and style of another domain (the color style of the shoes). By this way, it helps bring new ideas and provides visualizations for the bag designers who only have some raw ideas about designing bags of different color styles from the existing ones.

Figure 1: Mixture generation and conventional generation. Mixture generation requires absorbing concepts from different domains. In particular, here we learn content concept from the bags and style concept from the shoes for mixture generation. In contrast, conventional generation does not go beyond the available training domain.

We recognize such problems as the *mixture generation* problems, where we need to *jointly* absorb different kinds of concepts for generating a new domain beyond the available ones. [1] Unfortunately, as illustrated in the right part in Figure 1, existing GAN-based generative methods are restricted to generate new samples similar to the ones from training domains [Goodfellow *et al.*, 2014; Makhzani *et al.*, 2016; Odena *et al.*, 2017; Berthelot *et al.*, 2017]. On the other hand, although the style transfer [Gatys *et al.*, 2016; Johnson *et al.*, 2016; Ulyanov *et al.*, 2016] and image-to-image translation [Hertzmann *et al.*, 2001; Sangkloy *et al.*, 2017; Karacan *et al.*, 2016] models can translate an existing image to another style, they are also restricted in that they require off-the-shelf content templates, and thus cannot deal with the problems requiring going beyond the available content templates (e.g., designing new shapes as well as styles of bags). Therefore, the mixture generation problem still remains an open issue.

To explicitly learn different types of concepts from different domains, in this work we focus on learning content (e.g.,

*Corresponding Author

[1] Here we interpret the mixture of concepts in a simplified way. For better understanding we refer the readers to the materials about *conceptual blending* [Fauconnier and Turner, 2008; Goguen and Harrell, 2010].

shape of bags) from one domain, and style (e.g., color style of shoes) from another. The main idea is that, in the generation process, the style concept "joins" the content concept, so that a generator can not only keep the content concept in mind, but also absorb the style concept to generate images of a new domain. To this end, we develop a framework in which the content concept is represented as hierarchical features, while the style concept is learned and embedded in a hierarchical decoder. During generation, the hierarchical features (with content concept) progressively passes through the decoder (with style concept), and meanwhile the decoder "releases" the style concept. Thereby, the style concept joins the content concept and finally the decoder produces an image based on them. As an analogy which might not be very precise but helps to understand, one can imagine barbecuing. We have pork and fish at hand (content), put them on the grill (decoder), spread oil on them at an early stage and sprinkle spice right before enjoying them (progressively absorbing the style concept). Also note that in the very beginning, we pay certain amount of money (noise for generation) to buy the pork and spice.

More specifically, we propose a *mixture generator* model in our framework for mixture generation. The mixture generator consists of a content decoder and a mixture decoder. The content decoder learns from images of the content domain, and thus provides the mixture decoder with rich hierarchical features. These features, from mid-level to low-level, contain corresponding content concept and they are then progressively fed to the mixture decoder. The mixture decoder is designed to not only learn the style concept from the style domain, but also learn how to join both kinds of concepts for mixture generator.

With the mixture generator, we form a mixture generative adversarial network (MIXGAN). We evaluate our MIXGAN on several tasks. The experimental results show that our model can learn to generate images in a new domain, e.g., generating hand-written colorful digits provided that our model only observes black-and-white hand-written digits [LeCun *et al.*, 2010] and colorful type-script ones [Netzer *et al.*, 2011]. The main contributions are as follows:

1. We recognize the mixture generation problem that requires jointly absorbing concepts from different domains for generation, and propose to address it by joining style concept and content concept during generation.

2. We propose an unsupervised framework as well as a novel model, i.e., mixture generator, to join the style and content concept for mixture generation. Specifically, the learned content concept is represented as hierarchical features and the style concept is embedded in a mixture decoder. During generation, the decoder "releases" the style concept which can thus be absorbed by the content concept.

3. We show that our model can learn to generate images with content of one domain and style of another in several mixture generation tasks.

2 Related Work

GAN-based generative models. Generative Adversarial Nets (GAN) [Goodfellow *et al.*, 2014] is a popular framework for generation. It is like a two-player game, where a discriminator learns to distinguish real images from fake ones, while a generator tries its best to fool the discriminator. It is trained in an adversarial learning pattern, where the discriminator and generator iteratively improves themselves to beat the other one. Recently, lots of GAN-based models have largely promoted many generation fields, e.g., image generation [Goodfellow *et al.*, 2014; Radford *et al.*, 2016], image editing [Zhu *et al.*, 2016], and variational inference [Makhzani *et al.*, 2016], interpretable representation learning [Salimans *et al.*, 2016; Liu and Tuzel, 2016],etc.

Our work is closely related to GANs, as our framework adopts the adversarial learning pattern for training. On the other hand, our model is different from these models in that our model is designed for mixture generation, and therefore can generate a new domain. In contrast, these models are designed for conventional generation problems that do not require generating samples beyond training domains.

Style Transfer and Image-to-Image Translation. Our work is also closely related to the style transfer models [Gatys *et al.*, 2016; Johnson *et al.*, 2016; Ulyanov *et al.*, 2016; Yoo *et al.*, 2016] and image-to-image translation models [Hertzmann *et al.*, 2001; Sangkloy *et al.*, 2017; Karacan *et al.*, 2016]. Style transfer models aim to learn how to transfer the style of a source image to a target image [Gatys *et al.*, 2016], while the image-to-image translation models learn a translation mapping from images of one domain to those of another domain [Isola *et al.*, 2017]. Typically the translation models learn the mapping from one style to another [Isola *et al.*, 2017], e.g., translating a photo to a painting. Some image-to-image translation models adopt conditional generative adversarial network to learn a mapping from paired images, e.g. translating sketches to photos [Isola *et al.*, 2017; Eigen and Fergus, 2015; Hertzmann *et al.*, 2001; Johnson *et al.*, 2016; Laffont *et al.*, 2014; Long *et al.*, 2015; Shih *et al.*, 2013; Wang and Gupta, 2016; Xie and Tu, 2015; Zhang *et al.*, 2016]. Recently, unpaired image translation also receives much research efforts [Zhu *et al.*, 2017; Kim *et al.*, 2017; Yi *et al.*, 2017; Taigman *et al.*, 2017; Liu *et al.*, 2017; Dong *et al.*, 2017; Choi *et al.*, 2017].

Our framework is related to them since our model and these models both need to learn style from a specific domain. However, the translation models translate images in one domain to another domain and the style transfer models learn how to transfer style in a pair of images, whereas our framework aims to learn concepts from different domains for mixture generation.

3 Framework

In this section we present our framework. In the following, we will first give an overview of our framework, and then demonstrate how to learn content concept from a specific domain. Based on the content concept we develop the mixture generator which jointly learns the style concept and learns to join both kinds of concepts for mixture generation.

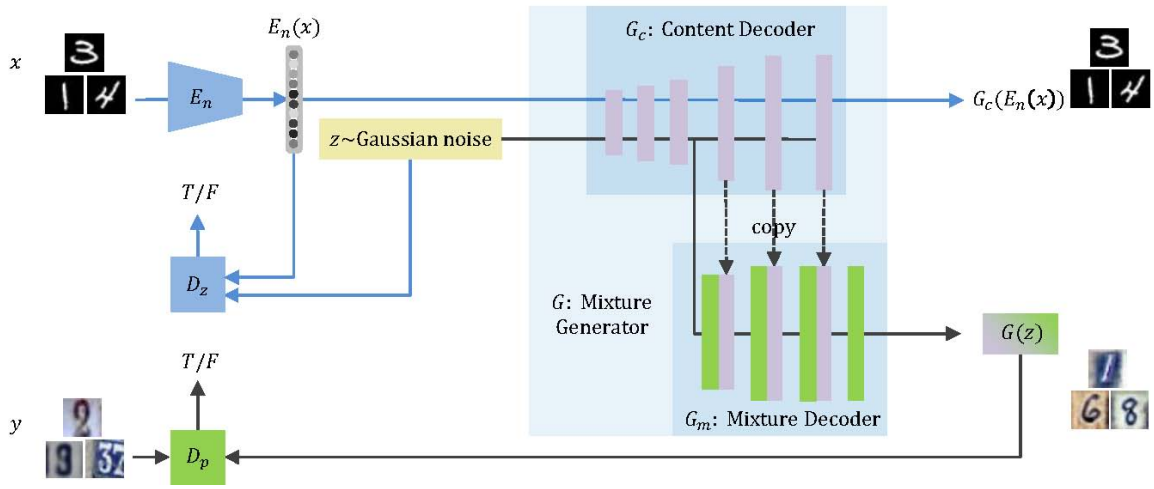

Figure 2: The structure of our mixture generative adversarial network (MIXGAN). The core of our framework is the mixture generator G, which consists of a content decoder G_c and a mixture decoder G_m. The content decoder connects the content generation network (upper part in the figure) and the mixture generation network (mixture generator with a patch discriminator D_p) for joint learning, while the mixture decoder absorbs content concept of x and style concept of y for mixture generation. ($Best\ viewed\ in\ color.$)

3.1 Overview

We propose an adversarial framework for jointly learning two kinds of concepts from two domains, respectively. The general idea is to introduce a jointly learned mixture generator G to absorb the two concepts. The mixture generator consists of a content decoder G_c which learns the content concept, and a mixture decoder G_m which jointly learns the style concept as well as how to join them for mixture generation. The content decoder G_c is learned within an adversarial autoencoder together with an encoder E_n and a discriminator D_z. G_c provides the mixture decoder G_m with rich hierarchical information on the content concept, so as to connect the content generation and the style generation. G_m is jointly learned with G_c (for learning content concept) and a patch discriminator D_p (for learning style concept). Thus, G_m learns to join these two kinds of concepts. By this way, the two concepts are mutually learned and connected within the mixture generator G for mixture generation.

In summary, our full objective for our framework is formulated as:

$$\min_{E_n,G_c,G_m} \max_{D_z,D_p} \mathcal{L}(E_n, G_c, G_m, D_z, D_p), \tag{1}$$

where

$$\begin{aligned}&\mathcal{L}(E_n, G_c, G_m, D_z, D_p)\\ =&\mathcal{L}_{content}(E_n, G_c, D_z) + \mathcal{L}_{mixture}(G_c, G_m, D_p).\end{aligned} \tag{2}$$

In the above formulation, $\mathcal{L}_{content}$ corresponds to learning the content concept and $\mathcal{L}_{mixture}$ corresponds to the joint learning for mixture generator.

We refer to our framework as MIXture GAN (MIXGAN). In the following, we elaborate each of them.

3.2 Learning Content

We consider content as containing general spacial relationship of pixels, but without suffcient details in an image. Hence, we assume that content can be encoded by low-dimensional and abstract latent variables, and thus we model learning the content by an autoencoder structure with a bottleneck. On the other hand, as our objective is to learn the content concept for generation purpose, we would further like to force the latent variables to follow a prior, so that we can easily sample from it. Such a consideration leads us to an adversarial autoencoder (AAE) structure [Makhzani *et al.*, 2016], as shown in the upper left part in Figure 2. A discriminator D_z aims to distinguish the noise z (sampled from a gaussian distribution $\mathcal{N}$) from the latent variables, and the encoder E_n has an extra task: to foolish the discriminator by forcing the latent variables follow the same gaussian distribution $\mathcal{N}$.

Formally, let $\mathcal{L}_{content}$ be the loss function for learning content concept, and let G_c be the content decoder in the AAE structure. We have our objective in learning content concept:

$$\begin{aligned}&\min_{E_n,G_c} \max_{D_z} \mathcal{L}_{content}(E_n, G_c, D_z)\\ =&\mathcal{L}_{adversarial}(E_n, D_z) + \lambda\mathcal{L}_{reconstruction}(E_n, G_c)\\ =&\mathbb{E}_{z\sim\mathcal{N}}[(D_z(z)-1)^2] + \mathbb{E}_{x\sim p_{data}(x)}[(D_z(E_n(x)))^2]\\ &+\lambda\mathbb{E}_{x\sim p_{data}(x)}[\| x - G_c(E_n(x)) \|_1],\end{aligned} \tag{3}$$

where x is an image from the content domain and λ controls the relative importance of the reconstruction loss over the adversarial loss. The L_1 reconstruction loss encourages the encoder to capture the low frequencies accurately but less high-frequency details than original images [Zhu *et al.*, 2017]. For acquiring more stable learning, we use a least square form of the adversarial loss [Mao *et al.*, 2017] instead of the negative log likelihood in the original GAN [Goodfellow *et al.*, 2014].

3.3 Mixture Generator

Now that we have learned some content concept in the AAE structure, we can propose our mixture generator G. We show

the architecture in the right of Figure 2. The mixture generator contains a content decoder G_c and a mixture decoder G_m. The content decoder G_c is also a part of AAE and thus has the ability to reproduce content concept by simply receiving noise sampled from $\mathcal{N}$. The content decoder G_c provides the mixture decoder G_m with rich hierarchical features which carry content concept of different levels. The mixture decoder G_m progressively takes these features as inputs (so it obtains content concept) and learns the style concept from outside (detailed in the next paragraph). The style concept is kept in the mixture decoder G_m's architecture, and therefore the style concept can be "released" and join the content concept while G_m is processing the hierarchical features. As a result, the mixture generator G jointly learns both kinds of concepts and can generate images from a new domain.

Regarding learning the style concept, we consider style as information related to high-frequency details and local structures in local areas, e.g., color and sharpness. Its structural information can thus be independent across different patches of the whole image. Hence, we model learning the style by independently observing and learning from small patches of images. To this end, we employ a patch discriminator D_p which looks at small patches of some candidate images and aims to distinguish the ones of the style domain from those generated by our mixture generator [Isola *et al.*, 2017]. The patch discriminator D_p is illustrated in the lower left in Figure 2. Here, D_p is like a teacher passing style concept to a student, i.e., the mixture generator G, who absorbs the style concept in the mixture decoder G_m.

Formally, our objective of the mixture generator is:

$$\min_{G_c,G_m}\max_{D_p}\mathcal{L}_{mixture}(G,D_p)=\mathcal{L}_{mixture}(G_c,G_m,D_p)$$
$$=\mathbb{E}_{y\sim p_{data}(y)}[((D_p(y)-1)^2]+\mathbb{E}_{z\sim\mathcal{N}}[(D_p(G(z)))^2]. \quad (4)$$

where y is an image from the style domain and z is the noise sampled from the prior, i.e., gaussian distribution $\mathcal{N}$.

3.4 Network Structure of the Mixture Generator

As shown in Figure 2, the content decoder consists of a block of three fully connected layers which process the high-level features, and a block of three convolutional layers which process mid-level and low-level features. Similar to [Isola *et al.*, 2017], each of these layers is followed by a Batchnorm layer and a ReLU activation. The structure of mixture decoder G_m is almost identical to the convolutional block of the content decoder G_c, as G_m only captures mid-level and low-level local style concept, only except that it has one more convolutional layer to fully learn from the low-level features.

Implementation details. As the joint learning of the mixture decoder G_m is based on the hierarchical features provided by the content decoder G_c. This is a two-step training process. We need to first train G_c with images of the content domain. Therefore, we first optimize Eq. (3) in an adversarial learning pattern [Makhzani *et al.*, 2016; Goodfellow *et al.*, 2014]. During our training, we use Adam [Kingma and Ba, 2014] solver with learning rate 0.0002 and $\beta_1 = 0.5, \beta_2 = 0.999$. It reaches convergence typically within 100 epoches. Then, as G_c can already "reproduce" the learned concept, we optimize Eq. (4) to train the mixture generator G, also in an iterative adversarial learning pattern. We use the same Adam solver and training typically converges within 300 epoches. During inference stage, as the latent variables space (ideally) follows the prior, we can simply sample from the prior $\mathcal{N}$ for mixture generator.

Figure 3: Samples in four training datasets.

Figure 4: The first four rows: samples in MNIST (content domain) and SVHN (style domain). The last three rows: our results in mixture generation.

4 Experiments

We show experimental results on three aspects. Firstly we show that MIXGAN can achieve our goal, i.e., mixture generation. Then we show that in mixture generator, the content decoder and the mixture decoder learn content concept and style concept, respectively. Finally, we present comparative results with other related generative models to show that our framework performs better in mixture generation.

4.1 Settings

For evaluation of our framework as well as for comparison, we design two groups of experiments, with different content domains and style domains.

The first group focuses on digits. We take hand-written digits from the MNIST dataset [LeCun *et al.*, 2010] and type-script digits from the SVHN dataset [Netzer *et al.*, 2011]. For hand-written digits, the content refers to the relatively wild shapes, in contrast to the relatively uniform shapes of type-script digits. The style refers to their color schemes, where hand-written digits are black-and-white (BW) and type-script digits are colorful. We show some examples in the first two rows in Figure 3.

The other group focuses on a more difficult task. We aim to learn content and style from bags and shoes, respectively,

Figure 5: The first four rows: samples in SVHN (content domain) and MNIST (style domain). The last three rows: our results in mixture generation.

Figure 6: The first four rows: samples in grayscale bags (content domain) and shoes (style domain). The last three rows: our results in mixture generation.

and in reverse. These images are available from [Isola *et al.*, 2017]. The content refers to the distinct shapes of both of them, and the style refers to the color style, as shown in the last two rows in Figure 3.

Since we are targeting the mixture generation which typically involves generating a new domain, there is no groundtruth available for quantitative evaluation. Thus, except visual results, we also design various proper methods including human evaluations and quantitative evaluations for evaluating our framework as well as comparing with other methods.

4.2 MIXGAN for Mixture Generation

In this section we show the results of MIXGAN for mixture generation. Successful mixture generation results account for above 70% in every task, so mixture generation is a big probability event for MIXGAN (please refer to Sec. 4.4 in details). Thus, we select successful samples to show the effectiveness of MIXGAN.

Experiments on digits. Our first task of this experiment group is to learn to generate images with content of MNIST, i.e., handwritten digits, and style of SVHN, i.e., colorful style like SVHN. We show the results in Figure 4. We can see from Figure 4 that MIXGAN can generate images containing hand-written digits while with a color style of the type-script ones. The generated domain is new as MIXGAN does not observe any images that look like the generated ones. Similar observations can be found in the mirror task, where the content is type-script digits and with BW color style. We show the results in Figure 5.

Figure 7: The first four rows: samples in grayscale shoes (content domain) and bags (style domain). The last three rows: our results in mixture generation.

Experiments on bags and shoes. Bags and shoes are a more difficult domain than digits since the objects are different and they are of higher resolution (64 by 64) compared to digits (32 by 32). We show in Figure 6 the experimental results where the content is from the bags and the style from shoes. Note that for better learning the content concept, we first transform the images of bags to grayscale images. This can be easily done and does not require considerable efforts. We can observe from Figure 6 that, although the content and style are from different objects with higher resolution, MIXGAN can learn to absorb different kinds of concepts from them, and generate a new domain.

We also show the results in the mirror task in Figure 7. We can see that MIXGAN can also learn content concept from the shoes and style concept from the bags.

4.3 Component Evaluation for Mixture Generator

In this subsection, we evaluate how each component contributes to the MIXGAN. Specifically, we evaluate the components of the proposed mixture generator. Recall that in the mixture generator (Figure 2), the content decoder serves to learn the content concept, and the mixture decoder learns the style concept as well as joins them for mixture generation. Here we take two tasks from the two experiment groups (one for each group) for example to illustrate the effects.

We visualize the intermediate results produced by the content decoder, i.e., $G_c(z)$ where G_c is the content decoder and z is noise. We show these results in the upper part of Figure 8. We can see that the intermediate digits already have the handwritten shapes, although without the desired color style. In

Figure 8: Left: results with MNIST (content domain) and SVHN (style domain) as training datasets. Right: results with grayscale bags (content domain) and shoes (style domain) as training datasets. The intermediate results: the content decoder outputs in the first three rows. The final results: the mixture generator outputs in the last three rows. Corresponding pairs of samples share the same content in the lower and upper parts.

contrast, the mixture generator produces images with desired color style, as shown in the lower part of Figure 8. Similar observation can be found on the bags. This observation verifies that the content decoder can learn the content concept, while the mixture decoder can learn the style concept and join them for mixture generation.

Models	AAE	LSGAN	CycleGAN	MIXGAN
TASK1	$3.47\% \pm 3.90\%$	$2.82\% \pm 1.56\%$	$2.54\% \pm 2.63\%$	$85.63\% \pm 6.37\%$
TASK2	$1.17\% \pm 1.64\%$	$7.32\% \pm 5.33\%$	$0.39\% \pm 0.72\%$	$71.13\% \pm 7.53\%$
TASK3	$1.95\% \pm 2.33\%$	$0.25\% \pm 0.71\%$	$0.00\% \pm 0.00\%$	$86.17\% \pm 6.62\%$
TASK4	$1.95\% \pm 1.62\%$	$0.50\% \pm 0.92\%$	$0.00\% \pm 0.00\%$	$90.70\% \pm 4.67\%$

Table 1: Success rate evaluated by human annotators. Measured by average rate $\pm$ std. Task 1: generating colorful handwritten digits (training datasets: MNIST, SVHN). Task2: generating black-and-white type-script digits (training datasets: MNIST, SVHN). Task3: generating colorful bags (training datasets: grayscale bags and colorful shoes). Task4: generating colorful shoes (training datasets: grayscale shoes and colorful bags).

Models	AAE	LSGAN	CycleGAN	MIXGAN
MNIST	22.47 ± 1.06	21.46 ± 0.30	48.98 ± 2.10	248.11 ± 1.88
SVHN	103.48 ± 1.34	102.17 ± 4.84	17.47 ± 0.78	251.96 ± 3.44

Table 2: MMD distances of the generated samples and the two training sets in Task 1: generating colorful handwritten digits (training datasets: MNIST, SVHN). The MMD distances are computed using the feature from the last hidden layer of the pretrained deep binary classifier network.

4.4 Comparison to Related Models

Now we compare MIXGAN with several related generative models by comprehensive visual and quantitative evaluation, including GAN-based generation models LSGAN [Mao *et al.*, 2017], adversarial autoencoder (AAE) [Makhzani *et al.*, 2016] and a state-of-the-art image translation model cycleGAN [Zhu *et al.*, 2017].

Visual results. From Figure 9 we can see that the conventional generation models, i.e., AAE and LSGAN, can learn to generate images within a specific domain, either BW handwritten digits or colorful type-script ones. However, they could not generate a new domain for mixture generation. We

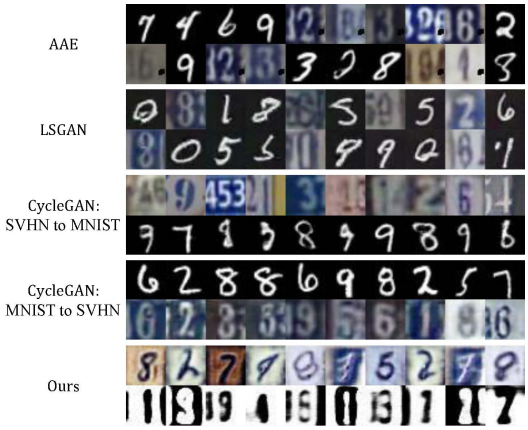

Figure 9: Comparative experiment results. Training datasets: the MNIST dataset and the SVHN dataset.

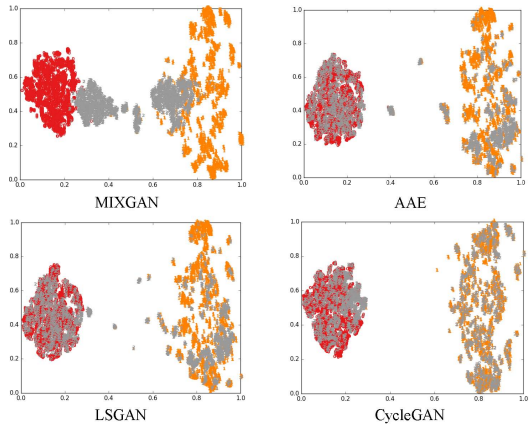

Figure 10: Visualization of the relationships between the generated samples and the training samples in generating colorful hand-written digits (Task 1 in Table 1). The red, yellow and grey points correspond to samples from MNIST, SVHN and the generated ones, respectively

can also observe that although cycleGAN can translate an image of hand-written digit to another of type-script digit, the content and style remain the same as in their original domains, while MIXGAN can achieve the mixture generation. In contrast, MIXGAN has specifically designed strategies to learn content concept, style concept and how to join them, respectively. Therefore, MIXGAN can learn different concepts from the two domains for mixture generation (i.e., colorful type-script digits).

To further explore the differences, we evaluate the mixture generation by visualizing the distributions of both the training samples and the generated samples using t-SNE [Maaten and Hinton, 2008]. We show the comparative results in the task of generating colorful hand-written digits in Figure 10 (corresponding to the Task 1 results in Table 1). We can see that MIXGAN is able to generate samples belonging to a distribution which is "between" the two training distributions. In contrast, the compared models fail in this mixture generation

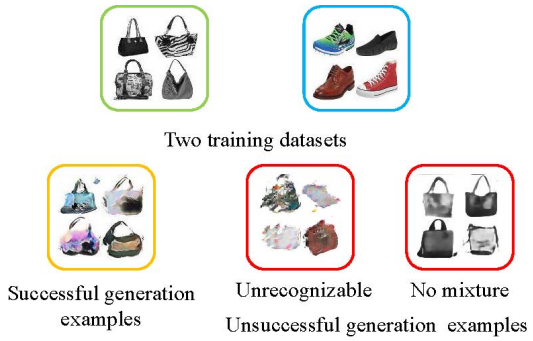

Figure 11: Illustration of the criteria of judging successful mixture generation by human annotators.

task. They only generate samples belonging to the original training distributions.

Human evaluations. We organize 100 human annotators to evaluate success rate of MIXGAN and the compared models in the mixture generation task. As our objective is mixture generation, we define that a generated image is "successful" if it can be recognized as in a new domain combining the content and style from the two training domains, respectively. Here the key criteria are recognizability and combination/mixture of content and style. For example, we show in Figure 11 some typical successful and failed cases in the task of combining the content of shoes and style of handbags. For each of our evaluation tasks (4 in total), each human annotator judges 200 samples generated by MIXGAN whether successful or not. The same evaluation is performed on the compared models, namely AAE, LSGAN and CycleGAN. We show the comparative results in Table 1. The average success rate of MIXGAN is above 70%, while the second best is always below 10%.

Quantitative evaluations. Although we can intuitively recognize how well the models can achieve mixture generation in a 2-D embedding in Figure 10, we further evaluate it quantitatively. To this end, we adopt the maximum mean discrepancy (MMD) between the generated samples and the training samples. Smaller MMD distance suggests that the generated sample still belong to one of the original training distributions. But we note that, larger MMD distance between the generated samples and training samples does not directly suggest a good mixture generation performance (because it can just fail to generate meaningful samples). Therefore, this measure should be considered complementary to the human annotator success rate: higher success rate with a higher MMD distance suggests better mixture generation.

To compute the MMD distance, we need to first of all determine which training distribution a given sample is compared with, because there are two distinct training distributions. To achieve this, we first assign each generated sample to a "nearer" training set, and then compute the MMD distance between the samples and their corresponding training set. We define "near" by training a deep binary classifier network which is trained using the two training sets and produces a probability that a sample is belonging to a specific training set. Then we use this classifier to judge to which training set each generated sample should be assign.

We show the comparative results in Table 2. Together with Table 1, we can see that MIXGAN has higher success rate and MMD distance, indicating the better performance in terms of mixture generation.

In summary, as compared to AAE, LSGAN and cycleGAN, our MIXGAN model can learn to generate samples belonging to a new domain with concepts absorbed from different domains.

5 Conclusion

We present our work on mixture generation, where we aim to generate a new domain beyond the training ones. In particular, we develop MIXGAN to generate a new domain that contains the content and style concepts extracted from two different domains. The core of MIXGAN is the mixture generator, which jointly learns two kinds of concepts as well as how to join them for mixture generation. Our experiments have shown its successful applications as compared to the sate-of-the-art GAN-based methods.

Acknowledgements

This work was supported partially by the National Key Research and Development Program of China (2018YFB1004903), NSFC (61522115, 61661130157, 61472456, U1611461), Guangdong Province Science and Technology Innovation Leading Talents (2016TX03X157), and the Royal Society Newton Advanced Fellowship (NA150459).

Finally we thank the anonymous reviewers for their invaluable comments.

References

[Berthelot *et al.*, 2017] David Berthelot, Tom Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[Choi *et al.*, 2017] Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. *arXiv preprint arXiv:1711.09020*, 2017.

[Dong *et al.*, 2017] Hao Dong, Paarth Neekhara, Chao Wu, and Yike Guo. Unsupervised image-to-image translation with generative adversarial networks. *arXiv preprint arXiv:1701.02676*, 2017.

[Eigen and Fergus, 2015] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *ICCV*, 2015.

[Fauconnier and Turner, 2008] Gilles Fauconnier and Mark Turner. *The way we think: Conceptual blending and the mind's hidden complexities*. Basic Books, 2008.

[Gatys *et al.*, 2016] Leon A Gatys, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. Preserving color in neural artistic style transfer. *arXiv preprint arXiv:1606.05897*, 2016.

[Goguen and Harrell, 2010] Joseph A Goguen and D Fox Harrell. Style: A computational and conceptual blending-based approach. In *The structure of style*, pages 291–316. Springer, 2010.

[Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *NIPS*, 2014.

[Hertzmann *et al.*, 2001] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. *SIGGRAPH*, 2001.

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

[Johnson *et al.*, 2016] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *ECCV*, 2016.

[Karacan *et al.*, 2016] Levent Karacan, Zeynep Akata, Aykut Erdem, and Erkut Erdem. Learning to generate images of outdoor scenes from attributes and semantic layouts. *arXiv preprint arXiv:1612.00215*, 2016.

[Kim *et al.*, 2017] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. *ICML*, 2017.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Laffont *et al.*, 2014] Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, Chao Qian, and James Hays. Transient attributes for high-level understanding and editing of outdoor scenes. *TOG*, 2014.

[LeCun *et al.*, 2010] Yann LeCun, Corinna Cortes, and Christopher JC Burges. Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2010.

[Liu and Tuzel, 2016] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. *NIPS*, 2016.

[Liu *et al.*, 2017] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *NIPS*, 2017.

[Long *et al.*, 2015] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CVPR*, 2015.

[Maaten and Hinton, 2008] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9:2579–2605, 2008.

[Makhzani *et al.*, 2016] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *ICLR*, 2016.

[Mao *et al.*, 2017] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. *ICCV*, 2017.

[Netzer *et al.*, 2011] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. *NIPS workshop on deep learning and unsupervised feature learning*, 2011.

[Odena *et al.*, 2017] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. *ICML*, 2017.

[Radford *et al.*, 2016] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *ICLR*, 2016.

[Salimans *et al.*, 2016] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *NIPS*, 2016.

[Sangkloy *et al.*, 2017] Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. *CVPR*, 2017.

[Shih *et al.*, 2013] Yichang Shih, Sylvain Paris, Frédo Durand, and William T Freeman. Data-driven hallucination of different times of day from a single outdoor photo. *TOG*, 2013.

[Taigman *et al.*, 2017] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised cross-domain image generation. *ICLR*, 2017.

[Ulyanov *et al.*, 2016] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. Texture networks: Feed-forward synthesis of textures and stylized images. *ICML*, 2016.

[Wang and Gupta, 2016] Xiaolong Wang and Abhinav Gupta. Generative image modeling using style and structure adversarial networks. *ECCV*, 2016.

[Xie and Tu, 2015] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. *ICCV*, 2015.

[Yi *et al.*, 2017] Zili Yi, Hao Zhang, Ping Tan Gong, et al. Dualgan: Unsupervised dual learning for image-to-image translation. *ICCV*, 2017.

[Yoo *et al.*, 2016] Donggeun Yoo, Namil Kim, Sunggyun Park, Anthony S Paek, and In So Kweon. Pixel-level domain transfer. In *ECCV*, 2016.

[Zhang *et al.*, 2016] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. *ECCV*, 2016.

[Zhu *et al.*, 2016] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. *ECCV*, 2016.

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