Robust High Dimensional Image Data Translation via Disentangled Feature Representations

Wendi Li
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
wendili@berkeley.edu

Abstract

The image-to-image (I2I) translation's goal is to learn the mapping from the source visual domain to the target visual domain. Many models focus on building a latent space (encode process) and use the latent space to generate the translated images. However, it is unguaranteed that the proposed methods can separate the features successfully. It is hard to determine whether they are able to find the latent codes that correspond to specific features. So in this project, we build a robust I2I model under multiple features. We use a data-driven way to disentangle the features. By applying two encoders with the feature switched and the generator produces different generated images, the neural network is forced to learn the disentangled features. Experimental results show that the fused translated images have more intra-class variation than typical I2I models.

1 Introduction

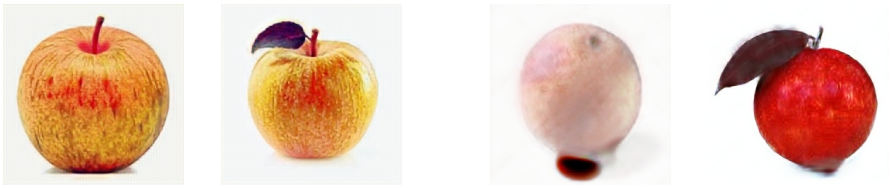
Image translation can be regarded as a process that generates high-dimensional images from the source visual domain to the target visual domain. The generated images should look authentic to human observers. The goal of image-to-image (I2I) translation is to learn the mapping from the source visual domain to the target visual domain. Many models [Isola et al., 2017, Liu et al., 2017, Murez et al., 2018] focus on building a latent space, which can be regarded as an encoding process. The latent space is then used to generate the translated images.

However, such generated images may be vulnerable, since the encoding process is uncontrollable and causes the tangled encoding, the noise perturbation may cause unpredictable damage [Luo et al., 2017]. It is even worse when the model includes multiple features. This is because the mapping may become multimodal — a single latent code may correspond to multiple features [Zhu et al., 2017b, Lee et al., 2018]. These disadvantages can be shown in methods without the disentangling process. Typical I2I models try to make the network do the compressing and the generating by totally themselves, which may lead the network to be "smartly lazy" — it only learns the most significant feature, and do nothing on the minor significant features. Such a strategy can be clearly observed when we take a closer look at what the translated images are under the entangled I2I model.

In Figure 1, it is the example of the entangled I2I model's performance. It can be seen that the model learns a lazy way, it only translates the texture, which is the most significant feature of the model. By doing that, it can cheat the mechanism that evaluates the generated result. This shows the disadvantage of the entangled model, the translation features may be limited.

Consider how to improve the I2I model, a perfect I2I requires the successful translation of each feature. In the I2I model, the semantic contents that should be preserved or can be changed should be determined. Consider an example — the I2I model to translate a "horse" into a "zebra" category, it can be determined that the things to be preserved include:

Preprint. Under review.



(a) Translated “orange” images (b) Translated “apple” images

Figure 1: Examples of entangled I2I model on apple & orange. The I2I model only learns to transit the most significant feature — the texture in this example.

1. The postural form of the object. That is when we translate a horse to a zebra, what its current posture should remain unchanged. If the source horse raises its front feet and keeps its head looking up, the translated zebra should also be in the posture that raises its front feet and keep head up. Otherwise, there is no difference that we only replace the result with an arbitrary zebra image.

2. The topological relations of objects should be the same. That is the translated image should keep the same positional relation as the original image. If the original image has 3 horses in a line, the translated zebra image should have the same number of zebra and the same position of each zebra. Otherwise, it can be seen that we only match the image with the random zebra distributing on the translated image.

Next, we need to determine what factors the I2I model should change or drop:

1. The category the object belongs to. This is the basic one, it means that we input a horse image, the translated image must be a zebra image, not still the horse image. This requires the translated image to include as many features translated as possible. Figure 1 shows only the texture changed, which makes us feel the translation process unreal.

2. The distribution domain of intra-class diversity. It means when we translate the object from one to another, we should keep the intra-class diversity as much as possible. In the horse & zebra example, the horse may include draft horses, Ferghana horse, light horses, gaited horses, and pony. The variation among these different kinds of horses is intra-class diversity. For the zebra, they may include plains zebra, mountain zebra, and Grevy’s zebra. What a perfect I2I model needs are the translation process should perfectly convert the horse’s distribution to the zebra’s distribution, and the distribution should match the whole zebra’s category as much as possible. If all the horses will be translated into one kind of zebra, mountain zebra, for example, it means the I2I model is not good, because it does not make a good distribution domain mapping.

Therefore, in this project, to make the I2I model robust, we focus on disentangling the features. We aim to make the model automatically disentangle the features so that the translation process can convert the category as much as possible. By doing that, the distribution domain can also be well cleared and the distribution convert can yield a better intra-class diversity in the translation process. In the network architecture, we want to use a data-driven way to force the network to disentangle the representations of the object’s features. We will illustrate why the proposed model can work and how the result should be.

2 Related Works

Our proposed model involves Generative Adversarial Net (GAN) [Goodfellow et al., 2014] combined with Variational Autoencoder (VAE) [Kingma and Welling, 2013], so we will have a brief introduction about these models. Then we will introduce some previous disentangled I2I models and verify their disability.

2.1 Variational Autoencoder

As it is a type of autoencoder, the VAE will encode the input image into a latent code, and use its decoder to extract the output. However, VAE does not focus on the distribution of the latent vector, it focuses on estimating the distribution of each sample. The VAE presents the distribution by two parameters, the mean value $\boldsymbol{\mu}$ and the standard deviation $\boldsymbol{\sigma}$ in the normal distribution assumption. Its main idea is to use a simplified model to replace the original model, so the training process is to make these parameters fit the real distribution.

As the decoder, it receives the latent code $\boldsymbol{z}$ and gets the output image $\boldsymbol{x}$, so according to the Bayes formula, the posterior probability should be:

$$p(\boldsymbol{x}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{\int_{\boldsymbol{z}} p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})\mathrm{d}\boldsymbol{z}} \tag{1}$$

However, the number of samples is very large, so it is unpractical to get $p(\boldsymbol{z}|\boldsymbol{x}_i)$ for each $\boldsymbol{x}_i \in \boldsymbol{X}$, where the $\boldsymbol{X}$ is the whole image domain.

Therefore, VAE tries to find a surrogate $q(\boldsymbol{z}|\boldsymbol{x})$ instead of the $p(\boldsymbol{z}|\boldsymbol{x})$. The regularization is implemented by adding the prior on the $p(\boldsymbol{z})$. Normally, it is the normal distribution ($p(\boldsymbol{z}) \in \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$). So the training process of the VAE is to minimize the expected likelihood under the prior regularization:

$$\begin{aligned} \mathcal{L}_V =& -\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}|\boldsymbol{z})p(\boldsymbol{z})}{q(\boldsymbol{z}|\boldsymbol{x})}\right] \\ =& -\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log p(\boldsymbol{x}|\boldsymbol{z})\right] \\ &+ \mathcal{D}_{KL}\left(q(\boldsymbol{z}|\boldsymbol{x})\|p(\boldsymbol{z})\right) \end{aligned} \tag{2}$$

where the $\mathcal{D}_{KL}$ means the KL divergence:

$$\mathcal{D}_{KL}(Q\|P) = \sum_{\boldsymbol{x}_i \in \boldsymbol{X}} q(\boldsymbol{x}_i) \log \frac{q(\boldsymbol{x}_i)}{p(\boldsymbol{x}_i)} \tag{3}$$

2.2 Generative Adversarial Net

The GAN includes two parts, the generator network part G and the discriminator network part D. The typical GAN's generator receives random noise as random seed and outputs the generated image, the discriminator tries to discriminate whether the input is fake or truth. So the training process can be described as each part fighting with each other. The optimizing process of the value function of G and D can be represented as:

$$\begin{aligned} \min_G \max_D V(D,G) =& \mathbb{E}_{\boldsymbol{x}\in\boldsymbol{X}}\left[\log D(\boldsymbol{x})\right] \\ &+ \mathbb{E}_{\boldsymbol{z}\in p(\boldsymbol{z})}\left[\log(1 - D(G(\boldsymbol{z})))\right] \end{aligned} \tag{4}$$

In our method, the GAN input will be the latent code from two autoencoders and the $\boldsymbol{z}$ can be regarded as the additional information from the encoded code. Some methods combines VAE and GAN [Larsen et al., 2015, Genevay et al., 2017, Liang et al., 2017], but as our best known, none of them use dual encoder parts and discriminaor parts to force the generator concurrently output dual results with the code reversed.

2.3 Disentangled Image-to-image Translation

Disentanglement, also known as decoupling, is the transformation of entangled changes in the original data space into a representation space, where the changes in different elements can be separated from each other. Disentangled variables usually contain interpretable semantic information and can reflect the separated factors in data.

For the disentangling works, many methods have been proposed. Some methods aim to separate the shared features and the latent space between two domains by constructing temp domains [Lee et al., 2018], separating network structures [Huang et al., 2018] and using pre-trained VGG net [Gatys et al., 2016].

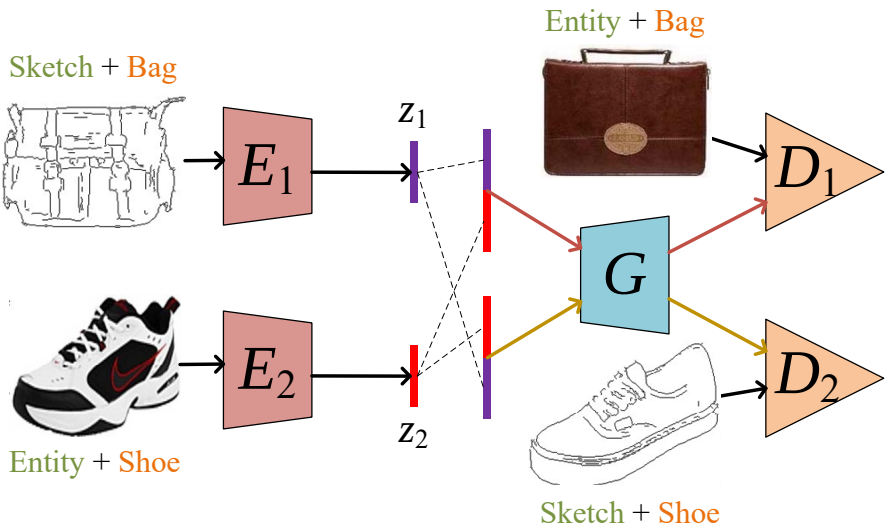


Figure 2: The proposed model. E_1 and E_2 are encoders, the generator G receives the concentrated code and outputs the generated images. D_1 and D_2 are corresponding discriminators. The generator outputs different types of images with the code reversed.

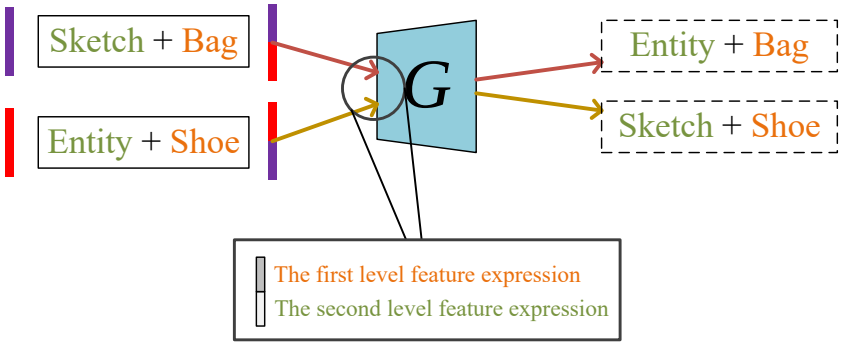


Figure 3: The training process forces the generator to learn the disentangled representation in a data-driven way.

However, it is unguaranteed that the proposed methods can separate the features successfully. It is hard to determine whether they can find the latent codes that correspond to specific features. Therefore, what we want to propose is the model that can be clearly illustrated what it is doing, without any taken for granted assumptions. We want to use a data-driven way to make the disentangled feature representation.

3 Dual Disentangled Image Translation

3.1 Network Architecture

The network architecture is shown in Figure 2. Two encoders will encode two category data with totally different features, and there are two outputs in the pipeline, too. The generator receives two encoded code concurrently and outputs the translated images. The generator will output a different result when the code gets reversed. This is the key disentangle process in our model. We do not require the data to be in pairs.

In the training process, we use multi-task learning, for both two discriminators, two encoders and the generator can be trained together. The encoders can alternatively be pre-trained using autoencoder reconstruction loss. The training process can be described as Algorithm 1.

Algorithm 1 Training Dual Disentangled GAN

1: Training E_1 and E_2, use the autoencoder reconstruction loss ▷ This is a pre-training process, it is optional
2: **for** each training iteration **do**
3: Concentrate the two encoded codes $\{z_1, z_2\}$, feed it to the generator G, its output is fed to D_1, using GAN loss
4: Concentrate the codes in the reversed direction, $\{z_2, z_1\}$, feed it to the generator G, its output is fed to D_2, using GAN loss
5: Do the optimization process, adjusting both two discriminators and the generator, the encoders may also be adjusted if needed
6: **end for**

3.2 Model Analysis

We can describe why such network architecture works for disentangling the features. As Figure 3 shows, for each encoded code, it includes two-level features. In our experiment, they are:

1. The expression form. This is shown in green text. In the example, there are two kinds of this feature, a sketch or an entity image.
2. The category in the human's view. This is shown in orange text. In the example, there are also two kinds of this feature, a bag or a shoe.

As two codes are concentrated, it can be regarded that the concentrated code has all four kinds of information (sketch/entity, bag/shoe). However, when the code is concentrated as $\{z_1, z_2\}$, the generator will output the results with the attributes **entity** + **bag**. When the code is reversed, that is $\{z_2, z_1\}$, the generator will output results with the attributes **sketch** + **shoe**. Such a process forces the generator to discard different information in the latent code's different dimensions. In Figure 3, the higher part will discard the second level feature and uses the first level feature, which is the first part determines whether the output will be a **bag** or **shoe**. On the other hand, the lower part will discard the first level feature and determines whether the output will be a **sketch** or **entity**. This is the data-driven disentangling process. And in this way, we finish the feature disentanglement.

Similar concepts can be expanded to other datasets. For example, for human's face images, two high-level features can be:

1. The person's identity of the face image.
2. The pose of this face image.

Thus, our method can be used on the data that has multiple-level features, and the disentanglement process is totally data-driven. We may expect a robust I2I translation once the representation space has a clear separated feature expression.

4 Experiment

In the experiment, we use two datasets mentioned in [Isola et al., 2017]. We try two I2I directions:

1. From **sketch** + **bag** and **entity** + **shoe** to **entity** + **bag** and **sketch** + **shoe**.
2. From **sketch** + **shoe** and **entity** + **bag** to **entity** + **shoe** and **sketch** + **bag**.

The results are shown in Figure 4. In the experiment, we have not done any hyperparameter tuning yet (each training process requires 20+ hours). The performance can be expected to improve. The advantages of the model:

1. The model does not require paired data — the data can be shuffled during the training process.
2. The information from the input that generated image does not need can be regarded as the random seed of this input. (For example, for each **entity** + **bag**, the **sketch** and **shoe**

(a) From **sketch** + **bag** and **entity** + **shoe** to **entity** + **bag** and **sketch** + **shoe**

(b) From **sketch** + **shoe** and **entity** + **bag** to **entity** + **shoe** and **sketch** + **bag**

Figure 4: The experiment results on shoes/bag & sketch/entity dataset.

information can be regard as the random seed for generating the image, since we shuffle the pairs during training process. For **sketch** + **shoe**, vice versa.)

3. From the result, we can see that the fused model can have more intra-class variation than typical I2I models such as cycleGAN [Zhu et al., 2017a]. It is not just "change the textures of the original ones", the types of the intra-class object also have changes in topological shapes and colors.

5 Future Work

For future work, we may search for strategies for a more balanced training process, since the current multi-task learning may easily fall on a certain one and dropped another one. We may need to figure out a stable strategy to make a balance between two training tasks. For example, a self-adaption adjusting strategy that can choose the correct relative learning rates of both tasks. This can be a research direction for future investigation.

References

Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016.

Aude Genevay, Gabriel Peyré, and Marco Cuturi. Gan and vae from an optimal transport point of view. *arXiv preprint arXiv:1706.01807*, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 172–189, 2018.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

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

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*, 2015.

Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 35–51, 2018.

Xiaodan Liang, Lisa Lee, Wei Dai, and Eric P Xing. Dual motion gan for future-flow embedded video prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1744–1752, 2017.

Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in neural information processing systems*, pages 700–708, 2017.

Junyu Luo, Yong Xu, Chenwei Tang, and Jiancheng Lv. Learning inverse mapping by autoencoder based generative adversarial nets. In *International Conference on Neural Information Processing*, pages 207–216. Springer, 2017.

Zak Murez, Soheil Kolouri, David Kriegman, Ravi Ramamoorthi, and Kyungnam Kim. Image to image translation for domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4500–4509, 2018.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017a.

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

Appendix

The detailed network architectures are shown below:

Table 1: Encoder E

layer	activation	output
input	-	$64 \times 64 \times 3$
conv 1×1	lReLU	$64 \times 64 \times 16$
conv 3×3	lReLU	$32 \times 32 \times 16$
conv 5×5	lReLU	$32 \times 32 \times 32$
conv 3×3	lReLU	$32 \times 32 \times 32$
conv 5×5	lReLU	$16 \times 16 \times 64$
conv 3×3	lReLU	$16 \times 16 \times 64$
conv 5×5	lReLU	$8 \times 8 \times 128$
conv 3×3	lReLU	$8 \times 8 \times 128$
conv 5×5	lReLU	$4 \times 4 \times 256$
conv 3×3	lReLU	$4 \times 4 \times 256$
conv 5×5	lReLU	$2 \times 2 \times 512$
conv 3×3	lReLU	$2 \times 2 \times 1024$
flatten	-	$1 \times 1 \times 4096$

Table 2: Discriminator D

layer	activation	output
input	-	$64 \times 64 \times 3$
conv 1×1	lReLU	$64 \times 64 \times 16$
pool 2×2	-	$32 \times 32 \times 16$
conv 5×5	lReLU	$32 \times 32 \times 32$
conv 3×3	lReLU	$32 \times 32 \times 64$
pool 2×2	-	$16 \times 16 \times 64$
conv 3×3	lReLU	$16 \times 16 \times 128$
pool 2×2	-	$8 \times 8 \times 128$
conv 3×3	lReLU	$8 \times 8 \times 256$
pool 2×2	-	$4 \times 4 \times 256$
conv 3×3	lReLU	$4 \times 4 \times 256$
pool 2×2	-	$2 \times 2 \times 256$
conv 3×3	lReLU	$2 \times 2 \times 128$
flatten	-	$1 \times 1 \times 512$
dense	sigmoid	$1 \times 1 \times 1$

Table 3: Generator G

layer	activation	output
input	-	$1 \times 1 \times 8192$
reshape	-	$4 \times 4 \times 512$
conv 3×3	lReLU	$4 \times 4 \times 512$
deconv 3×3	lReLU	$8 \times 8 \times 512$
conv 3×3	lReLU	$8 \times 8 \times 256$
deconv 3×3	lReLU	$16 \times 16 \times 256$
conv 3×3	lReLU	$16 \times 16 \times 128$
deconv 3×3	lReLU	$32 \times 32 \times 128$
conv 3×3	lReLU	$32 \times 32 \times 64$
deconv 3×3	lReLU	$64 \times 64 \times 64$
conv 3×3	lReLU	$64 \times 64 \times 32$
conv 3×3	lReLU	$64 \times 64 \times 32$
conv 1×1	tanh	$64 \times 64 \times 3$