GANs and Image Transformation

July 21, 2019

One of the current trends in computer vision and computer graphics is the use of generative adversarial netwoks (GANs) in image transformation tasks. In this document, I discuss a series of papers [2, 7, 5, 10, 1, 8] that are useful to my personal research. The ideas of the papers paint a nice intellectual thread that is too good not to mention.

1 Introduction

- **Image transformation** is the task of transforming an input image to a new image according to some requirements.
- The definition is intentionally broad. It includes the tasks such as:
  - image segmentation,
  - super resolution,
  - denosing,
  - generating animation,
  - facial expression manipulation,
  - colorization,
  - style transfer,
  - and many more.
- Image transformation tasks are generally under constrained. Moreover, some of them (e.g., style transfer and facial expression manipulation) are hard to specify precisely.
- Machine learning, especially deep learning, is an effective solution approach to image transformation tasks.
- One can apply standard *supervised learning*. This requires:
  - *Paired* input-output data for training.
  - A suitable loss function for the task.
- However, the above two requirements are problematic:
  1. Paired training data are often hard or impossible to acquire.
     - For example, say we want to convert photographs into paintings in the style of Monet. One would have to know how to raise Monet from his grave in order to prepare the training data.
  2. It is often hard to find the right objective function.
     - L^1 and L^2 losses yield blurry images [5].

  - How does one operationalize "realistic images" or "natural facial expressions?"

- Recent research has largely overcome the above two problems. The solution is to adapt *generative adversarial networks* (GANs).
  - GANs allow one to learn a suitable loss function instead of specifying one manually.
  - Researchers have proposed tricks to enable training without paired data [10, 1].

2 GANs

- **Generative modeling** is the task of generating data items that "look like" they come from a given dataset.
- In our case, the data items are images. We use $\boldsymbol{x}$ to denote an image and say that it comes from a domain $\mathcal{X} \subseteq \mathbb{R}^{h \times w}$, where h and w are the height and the width of an image, respectively. We assume that the images are distributed according to a probability distribution $p_{\mathcal{X}}$, which we do not know.
- The goal is to find a function $f : \mathcal{Z} \to \mathcal{X}$ that transforms a random vector $\boldsymbol{z}$ from domain $\mathcal{Z} \subseteq \mathbb{R}^{\ell}$ to an image in $\mathcal{X}$. We call $\boldsymbol{z}$ the **latent vector**.
- Typically, $\boldsymbol{z}$ is sampled from a well-know distribution such as a multi-dimentional Gaussian or a uniform distribution over $[0, 1]^{\ell}$. We call this distribution the **prior distribution** and denote it with $p_{\mathcal{Z}}$.
- The goal of generative modeling can be stated more precisely as to make the distribution of $f(\boldsymbol{z})$ as close to $p_{\mathcal{X}}$ as possible.
- In the GAN approach [2], we do not model the probability distribution $p_{\mathcal{X}}$ explicitly. We instead train two networks:
  - The **generator** G acts as our function f.
  - The **discriminator** D takes an image and outputs a *reality score*, which tells how much the discriminator thinks the image comes from the real data distribution rather than from the generator.
- Training has two goals:
  - Make the discriminator good at distinguising real images from fake (i.e., generated) images.
  - Make the generator good at fooling the discriminator.

  In order to achieve the above two goals, we simultaneously train the networks to minimize two loss functions: one for the discriminator and the other for the generator.
- Goodfellow et al. [2] propose the following loss functions, which we will call the *standard GAN losses* (SGAN):

$$\mathcal{L}_D^{\text{SGAN}} := -E_{\boldsymbol{x} \sim p_{\mathcal{X}}}[\log D(\boldsymbol{x})] - E_{\boldsymbol{z} \sim p_{\mathcal{Z}}}[\log(1 - D(G(\boldsymbol{z})))]$$
$$\mathcal{L}_G^{\text{SGAN}} := -E_{\boldsymbol{z} \sim p_{\mathcal{Z}}}[\log D(G(\boldsymbol{z}))]$$

  - The above is the "practical" version of the loss functions. They provide better gradients than the ones Goodfellow et al. use to derive theoretical results.
  - Notice that the reality score $D(\cdot)$ is intepreted as the probability that the input to the discriminator comes from the input dataset.

- One might notice that we do not know $p_{\mathcal{X}}$, but the loss function for the discriminator asks us to compute an expectation with respect to it. What we do is to approximate $p_{\S}$ empirically. Let the input data set be denoted by $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, we have that:

$$E_{\boldsymbol{x}\sim p_{\mathcal{X}}}[f(\boldsymbol{x})] \approx \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{x}_i),$$

  for any function f. So,

$$E_{\boldsymbol{x}\sim p_{\mathcal{X}}}[\log D(\boldsymbol{x})] \approx \frac{1}{n}\sum_{i=1}^{n} \log D(\boldsymbol{x}_i).$$

  - Other terms in the losses can be approximated by sampling directly from $p_{\mathcal{Z}}$ and doing Monte Carlo integration.

- Over the years, many loss functions for GANs have been proposed. We shall mention two popular ones:

  - The *least square losses* (LSGAN) [6]:

$$\begin{aligned}\mathcal{L}_D^{\text{LSGAN}} &:= E_{\boldsymbol{x}\sim p_{\mathcal{X}}}[(D(\boldsymbol{x})-1)^2] + E_{\boldsymbol{z}\sim p_{\mathcal{Z}}}[D(G(\boldsymbol{z}))^2] \\ \mathcal{L}_G^{\text{LSGAN}} &:= -E_{\boldsymbol{z}\sim p_{\mathcal{Z}}}[D(G(\boldsymbol{z}))^2]\end{aligned}$$

  - The *Wasserstein GAN losses with gradient panelty* (WGAN-GP) [3]:

$$\begin{aligned}\mathcal{L}_D^{\text{WGAN-GP}} &:= -E_{\boldsymbol{x}\sim p_{\mathcal{X}}}[D(\boldsymbol{x})] + E_{\boldsymbol{z}\sim p_{\mathcal{Z}}}[D(G(\boldsymbol{z}))] + \lambda E_{\hat{\boldsymbol{x}}\sim p_{\hat{\mathcal{X}}}}\left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2\right] \\ \mathcal{L}_G^{\text{WGAN-GP}} &:= -E_{\boldsymbol{z}\sim p_{\mathcal{Z}}}[D(G(\boldsymbol{x}))]\end{aligned}$$

    Here, $\hat{\boldsymbol{x}}$ is a vector sampled uniformly along straight lines between pairs of images sampled from the input distribution $p_{\mathcal{X}}$ and the distribution of generated image $G(\boldsymbol{z})$ where $\boldsymbol{z} \sim p_{\mathcal{Z}}$.

- The training algorithm for GAN is as follows:

  - In each step, two minibatches are sampled:
    - A minibatch of $\boldsymbol{x}$'s from the training data.
    - A minibatch of $\boldsymbol{z}$'s from the distribution $p_{\mathcal{Z}}$.
  - We feed $\boldsymbol{z}$ values to G to generate a batch of fake $\boldsymbol{x}$'s.
  - Both the real batch and fake batch are used to do an optimization step on D to minimize $\mathcal{L}_D$.
  - The fake batch and its output from D are then used to perform an optimization step on G to minimize $\mathcal{L}_G$.

- Training often uses Adam as the optimization algorithm.

- The following are common modifications to the training process:

  - When the discriminator is heavily regularized, such as when using the WGAN-GP loss, one may have several discriminator optimization steps per one generator optimization step [3].
  - An alternative to the above is to use different learning rates for the generator and the discriminator [4]. This is the so called *two time-scale update rule* (TTUR). In general, the discriminator learning rate is higher than the generator learning rate.

3 Conditional GANs

- It is hard to control the output of the vanilla GAN generator because one can only control the latent vector, which is unintuitive.
- It would be nice if, in addition to the latent vector, we can provide additional information to the generator to control the output.
  - The extra info can be a class label. For example, we may want to generate an image of the digit "2" that looks like it comes from the MNIST dataset.
- Mizra and Osindero extend vanilla GANs so that one can generate images conditioned on user-provided additional info [7].
- Both the generator and the discriminator take an piece of additional info, the conditioning vector $\boldsymbol{c}$. There interfaces become:
  - $G(\boldsymbol{z}, \boldsymbol{c}) \mapsto \boldsymbol{x}$.
  - $D(\boldsymbol{x}, \boldsymbol{c}) \mapsto$ reality score.
- The conditioning information $\boldsymbol{c}$ can be:
  - Class of the image. This is typically encoded as a one-hot vector.
  - Another image, which is used in the pix2pix paper [5].
  - The target expression of a face as in [8].
  - And much more...
- It is common to concatenate $\boldsymbol{c}$ to $\boldsymbol{z}$ (and $\boldsymbol{x}$) and give the result to G (and D) as input.
  - When concatenating $\boldsymbol{c}$ to $\boldsymbol{x}$, we typically concatenate a copy of $\boldsymbol{c}$ to each pixel of $\boldsymbol{x}$.
- The input dataset now must contain pairs of an image and its associated conditioning vector; namely, $\boldsymbol{X} = \{(\boldsymbol{x}_1, \boldsymbol{c}_1), (\boldsymbol{x}_2, \boldsymbol{c}_2), \ldots, (\boldsymbol{x}_n, \boldsymbol{c}_n)\}$.
- Let $\mathcal{C}$ denote the set of all possible conditioning vectors. Let us also assume that we can randomly sample $\boldsymbol{c}$ from $\mathcal{C}$, and let the sampling probability distribution be denoted by $p_{\mathcal{C}}$.
- All GAN loss functions are changed so that take into account the conditioning vectors. For exaxmple, the standard GAN loss function may be rewritten as follows:

$$\mathcal{L}_D^{\text{cSGAN}} := -E_{(\boldsymbol{x},\boldsymbol{c})\sim p_{\mathcal{X}}}[\log D(\boldsymbol{x}, \boldsymbol{c})] - E_{\boldsymbol{z}\sim p_{\mathcal{Z}}, \boldsymbol{c}\sim p_{\mathcal{C}}}[\log(1 - D(G(\boldsymbol{z}, \boldsymbol{c})))]$$
$$\mathcal{L}_G^{\text{cSGAN}} := -E_{\boldsymbol{z}\sim p_{\mathcal{Z}}, \boldsymbol{c}\sim p_{\mathcal{C}}}[\log D(G(\boldsymbol{z}, \boldsymbol{c}), \boldsymbol{c})]$$

4 Pix2pix

- Isola et al. propose a framework for using GANs in image transformation tasks.
- The idea is that the discriminator can learn a loss function specific to the problem at hand. As a result, one does not have to design the loss function by hand.
- We revise the notations a little.
  - Since we deal with image transformation, we will have two domains of images.
  - The source (input) domain is denoted by $\mathcal{X}$, and a source image is denoted by $\boldsymbol{x}$.

  - The target (output) domain domain is denoted dby $\mathcal{Y}$, and a target image is dnoated by $\boldsymbol{y}$.

- The pix2pix system learns to transform an image in $\mathcal{X}$ to an image in $\mathcal{Y}$. It does so by learning conditinal GANs that *take $\boldsymbol{x}$ as the conditioning vector.* In other words, the signatures of the generator and the discriminator are:

$$G(\boldsymbol{z}, \boldsymbol{x}) \mapsto \boldsymbol{y}$$
$$D(\boldsymbol{y}, \boldsymbol{x}) \mapsto \text{reality score}$$

- The objective function for the discriminator is the same as that of the standard GAN paper:

$$\mathcal{L}_D := -E_{(\boldsymbol{y},\boldsymbol{x})\sim p_{\mathcal{Y}}}[\log D(\boldsymbol{y}, \boldsymbol{x})] - E_{\boldsymbol{z}\sim p_{\mathcal{Z}}, \boldsymbol{x}\sim p_{\mathcal{X}}}[\log(1 - D(G(\boldsymbol{z}, \boldsymbol{x}), \boldsymbol{x}))]$$

- Isola et al. mix the GAN generator loss with the L^1 difference between the generated image $G(\boldsymbol{z}, \boldsymbol{x})$ and the correct output $\boldsymbol{y}$:

$$\mathcal{L}_G := -E_{\boldsymbol{z}\sim p_{\mathcal{Z}}, \boldsymbol{x}\sim p_{\mathcal{X}}}[\log D(G(\boldsymbol{z}, \boldsymbol{x}), \boldsymbol{x}))] + \lambda E_{(\boldsymbol{y},\boldsymbol{x})\sim p_{\mathcal{Y}}, \boldsymbol{z}\sim p_{\mathcal{Z}}}[\|\boldsymbol{y} - G(\boldsymbol{z}, \boldsymbol{x})\|_1]$$

- The random vector $\boldsymbol{z}$ is not provided as input to G. This is because the networks often learn to ignore it. Intead, it is injected through dropout layers in the generator. The layers are used in both training and testing.
  - Nevertheless, the authors observed only minor stochasticity in the output images.

- The pix2pix paper chooses to tackle image transformation problems where the overall structure of the output image aligns with that of the input image. The main transformation is in the texture.

- Network architectures used:
  - The generator uses U-Net [9], which has connections that skip the bottleneck layers. Skip connections are useful for preserving the overal structure.
  - The discriminator is specialized to assess the high frequency details of textures. The low frequency features are left for the L^1 loss.
    * The discriminator only operates on image patches that are smaller than the generated image. So, the authors call it *PatchGAN*.
    * In the paper, the generated images are 256×256, but the discriminator accepts 70×70 images as input.
    * The reality score of the whole image is the average of PatchGAN results on the patches of the input image.
    * The discriminator is fully convolutional, so one can just feed the input image and average the output of the last layer to get the reality score.

5 CycleGAN

- The pix2pix paper solves only one problem with applying supervised learning to image transformation tasks. We do not need to carefully design the loss function any more, but we still need paired training data.

- In case where the mapping from $\mathcal{X}$ to $\mathcal{Y}$ is a bijection, the CycleGAN paper proposes a way to avoid preparing paired training data [10].

- The idea is to employ *cycle consistency.* If the mapping is a bijection, then there is an inverse mapping. If we compose the forward mapping with the inverse mapping, then we should get an identity. We can try to enforce this requirement while training.

- More specifically, the paper train two sets of GANs.
  - The forward mapping GAN:

$$\begin{aligned} G(\boldsymbol{x}) &\mapsto \boldsymbol{y} \\ D_{\mathcal{Y}}(\boldsymbol{y}) &\mapsto \text{reality score} \end{aligned}$$

  - The backward mapping GAN:

$$\begin{aligned} F(\boldsymbol{y}) &\mapsto \boldsymbol{x} \\ D_{\mathcal{X}}(\boldsymbol{x}) &\mapsto \text{reality score} \end{aligned}$$

    Notice that the generators do not accept latent vectors any more. This is because we want the mapping to be deterministic.

- The loss functions of the generators incorporate terms that enforce cycle consistency:

$$\begin{aligned} \mathcal{L}_G &:= -E_{\boldsymbol{x}\in p_{\mathcal{X}}}[\log D_{\mathcal{Y}}(G(\boldsymbol{x}))] + \lambda E_{\boldsymbol{x}\in p_{\mathcal{X}}}[\|F(G(\boldsymbol{x})) - \boldsymbol{x}\|_1] \\ \mathcal{L}_F &:= -E_{\boldsymbol{y}\in p_{\mathcal{Y}}}[\log D_{\mathcal{X}}(F(\boldsymbol{y}))] + \lambda E_{\boldsymbol{y}\in p_{\mathcal{Y}}}[\|G(F(\boldsymbol{y})) - \boldsymbol{y}\|_1] \end{aligned}$$

- The loss functions for the discriminators are the same as before:

$$\begin{aligned} \mathcal{L}_{D_{\mathcal{Y}}} &:= -E_{\boldsymbol{y}\sim p_{\mathcal{Y}}}[\log D_{\mathcal{Y}}(\boldsymbol{y})] - E_{\boldsymbol{x}\sim p_{\mathcal{X}}}[\log(1 - D_{\mathcal{Y}}(G(\boldsymbol{x})))] \\ \mathcal{L}_{D_{\mathcal{X}}} &:= -E_{\boldsymbol{x}\sim p_{\mathcal{X}}}[\log D_{\mathcal{X}}(\boldsymbol{x})] - E_{\boldsymbol{y}\sim p_{\mathcal{Y}}}[\log(1 - D_{\mathcal{X}}(F(\boldsymbol{y})))] \end{aligned}$$

  However, in the paper's implementaion, the least square loss is used instead of the standard GAN loss.

- When generating photos from paintings, the paper adds another term to make the output of the generator as close as possible to input when the input comes from the targert domain.

$$\begin{aligned} \mathcal{L}_G &:= -E_{\boldsymbol{x}\in p_{\mathcal{X}}}[\log D_{\mathcal{Y}}(G(\boldsymbol{x}))] + \lambda E_{\boldsymbol{x}\in p_{\mathcal{X}}}[\|F(G(\boldsymbol{x})) - \boldsymbol{x}\|_1] + \lambda' E_{\boldsymbol{y}\in p_{\mathcal{Y}}}[\|G(\boldsymbol{y}) - \boldsymbol{y}\|_1] \\ \mathcal{L}_F &:= -E_{\boldsymbol{y}\in p_{\mathcal{Y}}}[\log D_{\mathcal{X}}(F(\boldsymbol{y}))] + \lambda E_{\boldsymbol{y}\in p_{\mathcal{Y}}}[\|G(F(\boldsymbol{y})) - \boldsymbol{y}\|_1] + \lambda' E_{\boldsymbol{x}\in p_{\mathcal{X}}}[\|F(\boldsymbol{x}) - \boldsymbol{x}\|_1]. \end{aligned}$$

  This is done so that the networks do not change tint of the output relative to the input.

6 StarGAN

- An *attribute* is a meaningful feature inherent in an image. For example, for a facial picture, the hair color, gender, and age of the person are attributes.
- A *domain* is a set of images sharing the same attributes. For example, the set of images of woman is a domain.
- Some image datatsets are annotated with multiple attributes. For example, in the CelebA database, each image is annotated with 40 attributes, including "Eyeglasses," "Wearing Hat," and "Smiling."
- In the *multi-domain image translation* task, we transform the input image according to attributes from multiple domains. For example, when given a facial image, we may want to transform it so that the person smiles and has blonde hair.
- If all attributes are binary and there are k domains, the task above can be accomplished by training k CycleGANs that transform to the domains.
- The StarGAN paper proposes an algorithm to train a single GAN pair that can translate to all the k domains [1].

- StarGAN requires the dataset to be annotated with domain labels. In this case, each image comes with a k-dimensional binary vector $\boldsymbol{c}$ that indicates which of the k attributes are present.
- The StarGAN generator receives as input the domain label vector $\boldsymbol{c}$:

$$G(\boldsymbol{x}, \boldsymbol{c}) \mapsto \boldsymbol{x}'.$$

  Notice that we use $\boldsymbol{x}'$ as the output because it should still look like it comes from the same dataset as the input image.
- The StarGAN discriminator consists of two parts.
  - A standard discriminator that outputs a reality score. We denote this part by D_{src}. The abbreviation "src" stands for source; i.e., whether the image comes from the dataset or from the generator.

$$D_{\text{src}}(\boldsymbol{x}) \mapsto \text{reality score}.$$

  - A multi-class classifier D_{cls} that outputs the domain labels. More precisely, it outputs a vector $\boldsymbol{c}$ where c_i is the probability that the ith attribute is present in the image.

$$D_{\text{cls}}(\boldsymbol{x}) \mapsto \boldsymbol{c}.$$

  In the paper, the networks share all layers except the last.
- The StarGAN discriminator loss employs the standard GAN loss and adds a term that makes the classifier do its job on the input dataset.

$$\begin{aligned}\mathcal{L}_D = &-E_{(\boldsymbol{x},\boldsymbol{c})\sim p_{\mathcal{X}}}[\log D_{\text{src}}(\boldsymbol{x})] - E_{(\boldsymbol{x},\boldsymbol{c})\sim p_{\mathcal{X}},\boldsymbol{c}'\sim p_{\mathcal{C}}}[\log(1 - \log D_{\text{src}}(G(\boldsymbol{x}, \boldsymbol{c}')))] \\ &+ \lambda_{\text{cls}} E_{(\boldsymbol{x},\boldsymbol{c})\sim p_{\mathcal{X}}}[\text{CrossEntropy}(\boldsymbol{c}, D_{\text{cls}}(\boldsymbol{x}))].\end{aligned}$$

  where

$$\text{CrossEntropy}(\boldsymbol{c}, \boldsymbol{c}') := -\sum_{i=1}^{k} \big(c_i \log c_i' + (1 - c_i) \log(1 - c_i')\big).$$

- The StarGAN generator loss uses the standard GAN loss and two additional terms. One makes the gnerator good at following the classifier. The other, called the *reconstruction loss*, make the generator cycle consistent with respect to the attributes.

$$\begin{aligned}\mathcal{L}_G := &-E_{(\boldsymbol{x},\boldsymbol{c})\sim p_{\mathcal{X}},\boldsymbol{c}\sim p_{\mathcal{C}}}[\log D_{\text{src}}(G(\boldsymbol{x}, \boldsymbol{c}'))] \\ &+ \lambda_{\text{cls}} E_{(\boldsymbol{x},\boldsymbol{c})\sim p_{\mathcal{X}},\boldsymbol{c}'\sim p_{\mathcal{C}}}[\text{CrossEntropy}(\boldsymbol{c}', D_{\text{cls}}(G(\boldsymbol{x}, \boldsymbol{c}')))] \\ &+ \lambda_{\text{rec}} E_{(\boldsymbol{x},\boldsymbol{c})\sim p_{\mathcal{X}},\boldsymbol{c}'\sim p_{\mathcal{C}}}[\|\boldsymbol{x} - G(G(\boldsymbol{x}, \boldsymbol{c}'), \boldsymbol{c})\|_1].\end{aligned}$$

- The rationale for the reconstruction loss is that, we would like the generator to preserve as much features of the input image as possible while changing only the parts that are relevant to the attributes at hand.

7 GANimation

- StarGAN can alter input images so that it contains multiplie attributes. However, the attributes binary.

- Human facial expression, on the other hand, can be thought of as being controlled by a collection of floating point parameters. The *Factial Action Coding System* (FACS) gives a set of parameters called *Action Units* (AU). Some AUs are related to contraction of facial muscles.
- The GANimation paper extends StarGAN so that it works with floating point attributes instead of binary attributes. It also specializes StarGAN to the task of facial expression manipulation [8].
- In this seciton, $\boldsymbol{x}$ denotes a picture of a human face. The conditioning vector $\boldsymbol{c}$ is a real vector in $[0,1]^k$. Each component indicates how much the AU associated to it is activated. As such, we call $\boldsymbol{c}$ the AU activation vector.
- The input data consists of pairs $(\boldsymbol{x}_i, \boldsymbol{c}_i)$ of face image and its associated AU activation vector.
- The generator:
  - It takes the same form as the StarGAN generator: $G(\boldsymbol{x}, \boldsymbol{c}') \mapsto \boldsymbol{x}'$.
  - The image formation model for the generator is, however, different. Instead of having the generator creates the output image directly, it forces the generator to generate an alpha mask $\boldsymbol{\alpha}$ and the color change $\boldsymbol{\delta}$. The final image is given by:

    $$\boldsymbol{x}' := (1 - \boldsymbol{\alpha})\boldsymbol{\delta} + \alpha\boldsymbol{x}.$$

    This is done because the facial expression changes only involve a small part of the image. Ideally, most pixels of $\boldsymbol{\alpha}$ should close to zero.
- The discriminator is similar to that of StarGAN. It contains the D_{src} part, which outputs a reality score, and the D_{aua} part, which outputs the the AU activation vector of the input image. The two parts share all layers except their last.
- The discriminator loss is almost the same as that of the StarGAN loss. There are two main differences.
  - The GANimation paper uses the WGAN-GP loss instead of the standard GAN loss.
  - It also uses the least square loss to force D_{aua} to regress the real AU activation.

  The expression for the discriminator loss is:

  $$\begin{aligned}\mathcal{L}_D := &-E_{(\boldsymbol{x},\boldsymbol{c})\sim p_{\mathcal{X}}}[D_{\text{src}}(\boldsymbol{x})] + E_{(\boldsymbol{x},\boldsymbol{c})\sim p_{\mathcal{X}},\boldsymbol{c}'\sim p_{\mathcal{C}}}[D_{\text{src}}(G(\boldsymbol{x},\boldsymbol{c}'))] + \lambda_{\text{gp}} E_{\hat{\boldsymbol{x}}\sim p_{\hat{\mathcal{X}}}}\left[(\|\nabla_{\hat{\boldsymbol{x}}} D_{\text{src}}(\hat{\boldsymbol{x}})\|_2 - 1)^2\right] \\ &+ \lambda_{\text{aua}} E_{(\boldsymbol{x},\boldsymbol{c})\sim p_{\mathcal{X}}}[\|\boldsymbol{c} - D_{\text{aua}}(\boldsymbol{x})\|_2^2].\end{aligned}$$

- The generator loss contains the following term:
  - The WGAN-GP loss to make sure the image looks realistic.

    $$\mathcal{L}_G^{\text{gan}} := -E_{(\boldsymbol{x},\boldsymbol{c})\sim p_{\mathcal{X}},\boldsymbol{c}'\sim p_{\mathcal{C}}}[D_{\text{src}}(G(\boldsymbol{x},\boldsymbol{c})')]$$

  - The *alpha mask loss* to encourage the alpha mask to be (1) smooth and (2) closs to zero.

    $$\mathcal{L}_G^{\text{alpha}} := \lambda_{\text{tv}} E_{(\boldsymbol{x},\boldsymbol{c})\sim p_{\mathcal{X}}}\left[\sum_{i,j}\left[(\boldsymbol{\alpha}_{i+1,j} - \boldsymbol{\alpha}_{i,j})^2 + (\boldsymbol{\alpha}_{i,j+1} - \boldsymbol{\alpha}_{i,j})^2\right]\right] + E_{(\boldsymbol{x},\boldsymbol{c})\sim p_{\mathcal{X}}}[\|\alpha\|_2^2].$$

    Here, the "tv" subscript of λ_{tv} stands for "total variation."
  - The *expression loss* to make sure that the generator can generate the required expression.

    $$\mathcal{L}_G^{\text{aua}} := E_{(\boldsymbol{x},\boldsymbol{c})\sim p_{\mathcal{X}},\boldsymbol{c}'\sim p_{\mathcal{C}}}[\|\boldsymbol{c}' - D_{\text{aua}}(G(\boldsymbol{x},\boldsymbol{c}'))\|].$$

– The *reconstruciton loss* employ cycle consistency to make sure that the generator preserve as much features of the input image as possible.

$$\mathcal{L}_G^{\text{rec}} := E_{(\boldsymbol{x},\boldsymbol{c})\sim p_{\mathcal{X}},\boldsymbol{c}'\sim p_C}[\|\boldsymbol{x} - (G(\boldsymbol{x},\boldsymbol{c}'),\boldsymbol{c})\|_1].$$

The full expression of the generator loss is given by:

$$\mathcal{L}_G := \mathcal{L}_G^{\text{gan}} + \lambda_\alpha \mathcal{L}_G^{\text{alpha}} + \lambda_{\text{aua}} \mathcal{L}_G^{\text{aua}} + \lambda_{\text{rec}} \mathcal{L}_G^{\text{rec}}.$$

References

[1] Choi, Y., Choi, M., Kim, M., Ha, J., Kim, S., and Choo, J. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. *CoRR abs/1711.09020* (2017).

[2] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 2672–2680.

[3] Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved training of wasserstein gans. *CoRR abs/1704.00028* (2017).

[4] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Klambauer, G., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR abs/1706.08500* (2017).

[5] Isola, P., Zhu, J., Zhou, T., and Efros, A. A. Image-to-image translation with conditional adversarial networks. *CoRR abs/1611.07004* (2016).

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

[7] Mirza, M., and Osindero, S. Conditional generative adversarial nets. *CoRR abs/1411.1784* (2014).

[8] Pumarola, A., Agudo, A., Martinez, A. M., Sanfeliu, A., and Moreno-Noguer, F. Ganimation: Anatomically-aware facial animation from a single image. In *The European Conference on Computer Vision (ECCV)* (September 2018).

[9] Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. *CoRR abs/1505.04597* (2015).

[10] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on* (2017).