HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

PROJECT 1 REPORT

Application of Neural style transfer and GAN

Doan Ngoc Vinh - 20204935

Contents

1 Abstract

Computer vision is a branch of artificial intelligence (AI) that enables computers and systems to extract useful information from digital photos, videos, and other visual inputs and to execute actions or make suggestions based on that information. AI makes it possible for computers to think, while computer vision makes it possible for them to see, hear, and comprehend.Neural style transfer and GAN both are deep neural networks build for image generation problems. Computer vision has became more and more popular, so do Neural style transfer and GAN. In the project, i will explain and give the illustration of using both of them. Keyword: Computer vision, image generation, neural style transfer, GAN.

2 Architecture

2.1 Neural style transfer

Neural style transfer is a technique for transferring a certain look from one picture to another while preserving the original image's information. The only modification is the way the image is styled to give it a more artistic feel.

The layout or drawing is depicted in the content image, and the colors or painting used to represent the style are used. It is a computer vision application involving deep convolutional neural networks and image processing methods. Two types of pictures are involved in neural style transfer: the content image and the style image. This method aids in recreating the content image in the reference image's fashion. It transfers the aesthetic style from one image to another using neural networks. There are countless opportunities for design, content creation, and the creation of innovative technologies thanks to neural style transfer.

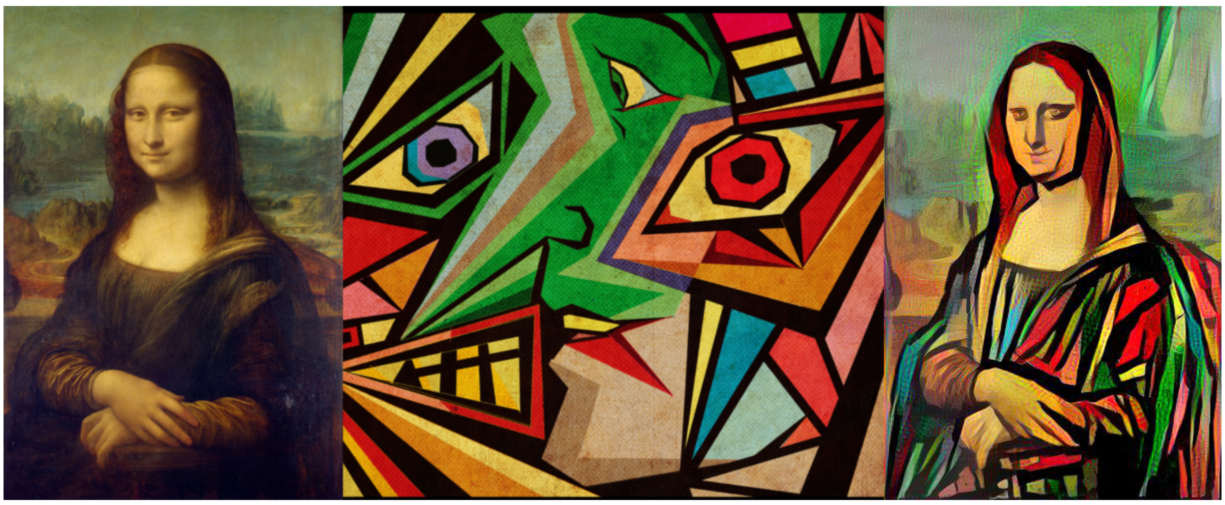

Figure 1: Neural style transfer example

As can be seen from Figure 1, the 'Mona Lisa' has been style changing without losing the content of the original image. So on, we can style changing many times which just need different style pictures.

Figure 2 is the basic demonstration of the neural style transfer. A pre-trained feature extractor and a transfer network are required for training a style transfer model. A pre-trained TensorFlow model called ImageNet-VGG is often used.

The input picture is changed into representations that focus more on the image's content than on its precise pixel values.These have to be converted into raw pixels and given to the model to

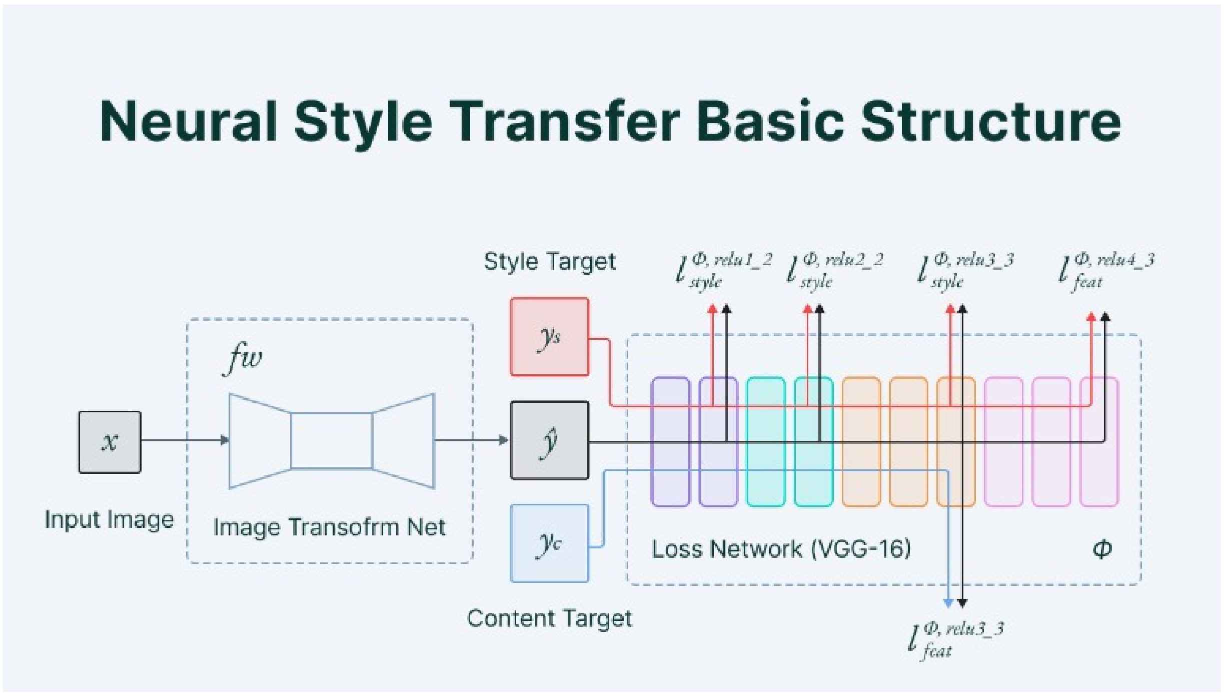
Neural Style Transfer Basic Structure
Style Target
$l_{style}^{\Phi, relu1\_2}$
$l_{style}^{\Phi, relu2\_2}$
$l_{style}^{\Phi, relu3\_3}$
$l_{feat}^{\Phi, relu4\_3}$
y_s
fw
x
$\hat{y}$
Input Image
Image Transofrm Net
y_c
Content Target
Loss Network (VGG-16)
Φ
$l_{feat}^{\Phi, relu3\_3}$

Figure 2: Basic structure

transform it into a set of features. The features we obtain from the higher layers of the model may be seen as being more closely tied to the image's content. We use the correlation between several filter responses to reflect the style of a reference image.

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^{l} - P_{ij}^{l}\right)^{2}$$

Figure 3: Content loss

Content loss: Content loss is calculated by Euclidean distance between the respective intermediate higher-level feature representation of input image (x) and content image (p) at layer l.

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - A_{ij}^l\right)^2$$

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

Figure 4: Style loss

Style loss: Style loss is calculated by the distance between the gram matrices of the generated image and the style reference image. The contribution of each layer in the style information is calculated by E(l) formula.

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

Figure 5: Style loss

Total loss: Alpha and beta are weights for content and style, respectively. They can be tweaked to alter our final result.

VGG16: It contains convolution layers and pooling layers. Conv 1-2, Conv 2-2. Conv 3-3. Conv 4-3 are extracted for style loss. Conv 4-3 is extracted for content loss.

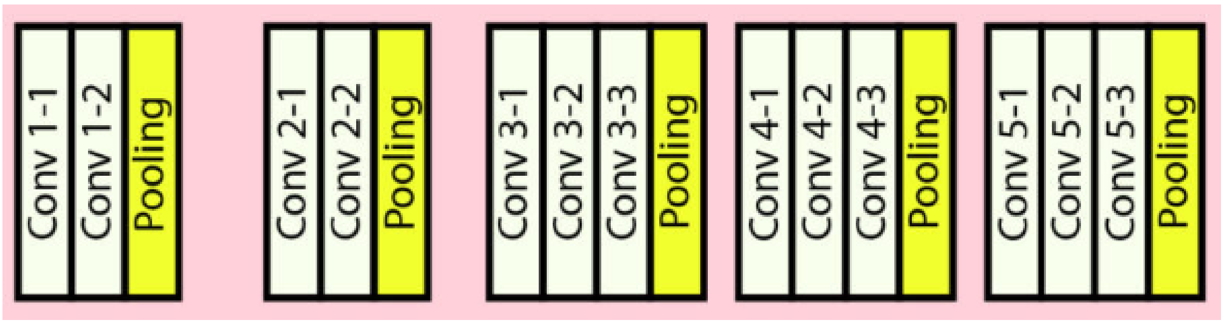

Figure 6: VGG16

2.2 Generative adversarial network

In a generative adversarial network (GAN), two neural networks fight with one another to make predictions that are as accurate as possible. The majority of the time, GANs operate unsupervised and learn through cooperative zero-sum games. In the project we will focus on animeGANv2.

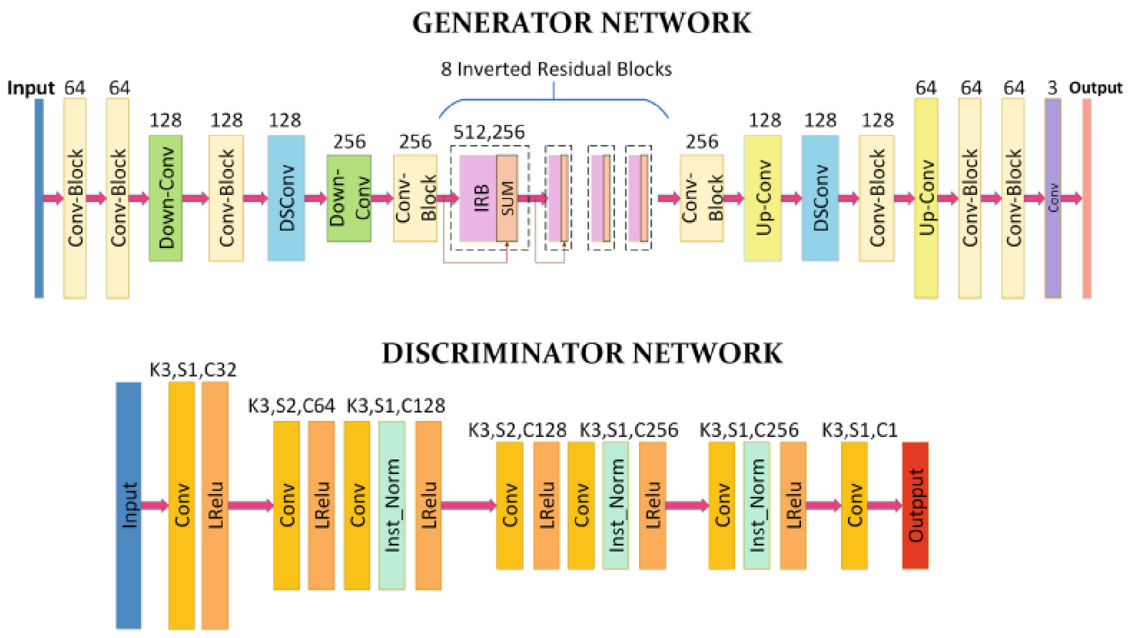

Figure 7: AnimeGANv2 Structure

As can be seen from figure 7, the structure of animeGANv2's generator is quite complicated, let's break it down into components. In figure 8, The "K" is the kernel size, "C" is the number of feature maps, "S" is the stride in each convolutional layer. The "H" is the height of the feature map, "W" is the width of the feature map. Resize means the interpolation method to set the size of the feature maps. The plus indicates the element-wise addition.

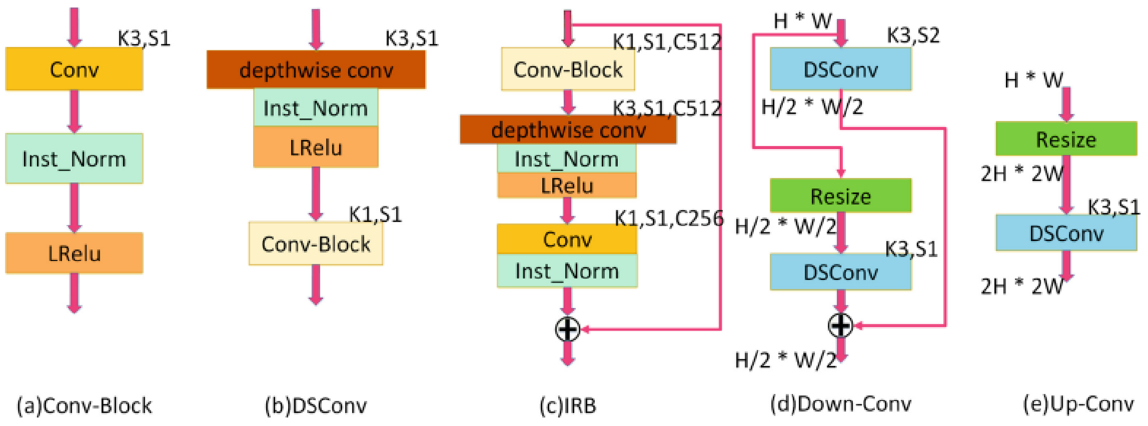

Figure 8: AnimeGANv2 Generator's components

Conv-Block: Consist of Convolutional layer, Instance normalization layer, LeakuRelu layer. This block's purpose is to obtain feature maps since Convolutional layer aids the detection of patterns, regardless the position.

DSConv: Consist of Depthwise convolutional layer, Instance normalization layer, LeakuRelu layer and a Convolutional layer. The block acts as a Conv-Block but less parameters, the Conv(K1-S1) is used to reduce the depth dimession.

IRB: Inverted residual block consist of Conv-Block, Depthwise convolutional layer, Instance normalization layer, LeakuRelu layer and a Convolutional layer. This block is the core structure of the animeGANv2. The IRB is also used to extract features but significantly reduces the parameters. The skip connection can be illustrated as: a[l+2] = g(w[l+2] x a[l+2] + b[l+2] + a[l]).

Down-Conv: Consist of DSConv block, Resize block and another DSConv block. The block is used as down-sampling module and resize it 2 times.

Up-Conv: Consist of Resize block and DSConv block. The block is used as up-sampling module and resize it 2 times.

Discriminator network: Contrary to the generator network, discriminator seems less complicated. It consist of Convolutional layer, Instance normalization layer, LeakuRelu layer.

Loss function:

- Total loss: L(G, D) = Wadv x Ladv(G, D) + Wcon x Lcon(G, D) + Wgra x Lgra(G, D) + Wcol x Lcol(G, D).
- Adversarial loss: Ladv(G, D) is the adversarial loss that affects the animation transformation process in the generator.
- Content loss: Lcon(G, D) is the content loss which helps to make the generated image retain the content of the input photo.
- Grayscale style loss: Lgra(G, D) is the grayscale style loss which makes the generated images have the clear anime style on the textures and lines.
- Color reconstruction loss: Lcol(G, D) is used as the color reconstruction loss to make the generated images have the color of the original photos.
- W parameters: We can change the to archive different final result. In Chen's paper, they set Wadv = 300,Wcon = 1.5, Wgra = 3 and Wcol = 10 which gives balance result of style and style preservation.
- Generator's loss function: L(G) = Wadv(Epi in Sdata(p))[(G(pi) - 1)2] + Wcon x Lcon(G, D) + Wgra x Lgra(G, D) + Wcol x Lcol(G, D)
- Discriminator's loss function: L(D) = Wadv[Eai in Sdata(a)[(D(ai) - 1)2] + Epi in Sdata(p) x [(D(G(pi)))2] + Exi in Sdata(x)[(D(xi))2] + 0.1Eyi in Sdata(y)[(D(yi))2]]

Loss function detail:

- The least squares loss function in LSGAN is employed as the adversarial loss Ladv(G, D).
- About content loss function: For the content loss Lcon(G, D) and the grayscale style loss Lgra(G, D), the pre-trained VGG19 is used as the perceptual network to extract the high-level semantic features of the images. Lcon(G, D) and Lgra(G, D) can be expressed as figure 9, lth layer is "conv4-4" in VGG. The l1 sparse regularization is used to calculate Lcon(G, D) and Lgra(G, D).
- Convert the RGB to YUV conversion of the picture's color to create the color reconstruction loss Lcol in order to improve the image color reconstruction (G, D). L1 loss is applied to

the Y channel in Lcol(G, D), and Huber Loss is the U and V channels are. Lcol(G, D) is formally defined as figure 10.

- About discriminator's loss function: EyiSdata(y)[(D(yi))2] represents the edge-promoting adversarial loss and ExiSdata(x)[(D(xi))2] represents the grayscale adversarial loss, 0.1 is the scaling factor.

$$L_{con}(G,D) = E_{p_i \sim S_{data}(p)}[\| VGG_l(p_i) - VGG_l(G(p_i)) \|_1]$$

$$\begin{aligned} L_{gra}(G,D) = E_{p_i \sim S_{data}(p)}, E_{x_i \sim S_{data}(x)}[\| &Gram(VGG_l(G(p_i))) \\ &-Gram(VGG_l(x_i)) \|_1] \end{aligned}$$

Figure 9: Content loss fuction and grayscale loss function

$$\begin{aligned} L_{col}(G,D) = E_{p_i \sim S_{data}(p)}[\| Y(G(p_i)) - Y(p_i) \|_1 + \| U(G(p_i)) - U(p_i) \|_H \\ + \| V(G(p_i)) - V(p_i) \|_H] \end{aligned}$$

Figure 10: Color reconstruction loss function

3 Application

I'll create a google colab notebook implementing neural style transfer for photos and videos, animeGANv2 for photos. For neural style transfer part, I am using arbitrary image stylization model - a pre-trained model from tensorflow. For animeGANv2, I am using the pre-trained model from author Xin Chen.

For a detailed demo: on this click.

4 Conclusion and future idea

In conclusion, the project is first start evolving deep learning technique: Neural style transfer and GAN. The project is a fine combination of knowing the architecture and real application.

However, the project has limitation and needs lots of improvements.The cons are lack of self-trained models, modifications, the application is still vague. For the advancement, self-creating data for training models then create unique styles. Moreover, we can develop the project to a more specific task like creating a transformer to transform a long movie to a anime-like comic. The idea is best suit for person who prefer reading comics to watch a whole movie.