Image-to-Image Translation with Conditional Adversarial Networks

Aman Krishna Gyanshu Azad Singh

***Abstract*—The paper uses Conditional Adversarial Networks to translate an input image to an output image. This way the network learns the mapping from input to output along with a loss function to train this mapping. This makes it possible to apply the same generic approach to problems that traditionally would require very different loss formulations. It demonstrates that this approach is effective at synthesizing photos from label maps, reconstructing objects from edge maps, and colorizing images, among other tasks.**

I. Introduction

Many problems in image processing, computer graphics, and computer vision can be posed as "translating" an input image into a corresponding output image. Image-to-image translation can be defined as the task of translating one possible representation of a scene into another, given sufficient training data. Earlier, for different types of image-to-image translation different solutions were proposed. Goal of this paper was to give a general purpose solution for all of these problems. Many solutions have been tried before like using CNN. But the main problem with that is a lot of manual effort is required. We have to tell CNN what we wish to minimize and that is not easy as we have to as precise as we can, otherwise it will lead to blurry pictures or many other problems. So it would be desirable if we just have to give a high-level goal, like "make the output indistinguishable from the real one". This is what has been achieved by Generative Adversarial Networks(GANs). GANs learn a loss that tries to classify if the output image is real or fake, while simultaneously training a generative model to minimize this loss. Because GANs learn a loss that adapts to the data, they can be applied to a multitude of tasks that traditionally would require very different kinds of loss functions. Further improvement is to apply GAN in a conditional setting(cGAN) which learns a conditional generative model. This makes cGANs suitable for image-to-image translation tasks, where we condition on an input image and generate a corresponding output image. The paper's primary contribution is to demonstrate that on a wide variety of problems, conditional GANs produce reasonable results. The other goal of the paper was to present a simple framework sufficient to achieve reasonable results. cGANs also have earlier been applied in many image translation problems. The paper's framework differs in that nothing is application specific. This makes the setup considerably simpler than most others. Their method also differs from the prior works in several architectural choices for the generator and discriminator. Unlike past work, for the generator they use a "U-Net"-based architecture, and for the discriminator they use a convolutional "PatchGAN" classifier, which only penalizes structure at the scale of image patches. A similar PatchGAN architecture was previously proposed in to capture local style statistics. Here we show that this approach is effective on a wider range of problems, and we investigate the effect of changing the patch size.

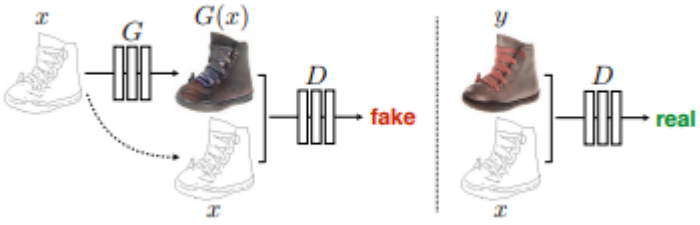

Fig. 1. Training a conditional GAN to map edges to photo. The discriminator, D, learns to classify between fake(synthesized by the generator) and real {edge, photo} tuples. The generator, G, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.

II. Method

GANs are generative models that learn a mapping from random noise vector z to output image y, $G : z \rightarrow y$. In contrast, conditional GANs learn a mapping from observed image x and random noise vector z, to y, $G : \{x, z\} \rightarrow y$. The generator G is trained to produce outputs that cannot be distinguished from "real" images by an adversarially trained discriminator, D, which is trained to do as well as possible at detecting the generator's "fakes". This training procedure is diagrammed in Figure 1.

A. Objective

The objective of a conditional GAN can be expressed as

$$\mathcal{L}_{cGAN}(G,D) = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x,z))], \quad (1)$$

where G tries to minimize this objective against an adversarial D that tries to maximize it, i.e. $G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G,D)$. We also include L1 loss as it encourages less blurring:

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1] \quad (2)$$

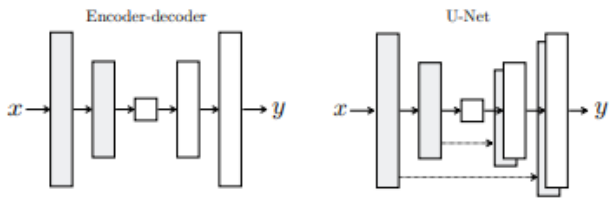

Fig. 2. Two choices for the architecture of the generator. The "U-Net" is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

Our final objective function is

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G) \qquad (3)$$

B. Network architectures

Both generator and discriminator use modules of the form convolution-BatchNorm-ReLu.

1) Generator with skips: A defining feature of image-to-image translation problems is that they map a high resolution input grid to a high resolution output grid. In addition, for the problems we consider, the input and output differ in surface appearance, but both are renderings of the same underlying structure. Therefore, structure in the input is roughly aligned with structure in the output. We design the generator architecture around these considerations. Many previous solutions to problems in this area have used an encoder-decoder network. In such a network, the input is passed through a series of layers that progressively down-sample, until a bottleneck layer, at which point the process is reversed. Such a network requires that all information flow pass through all the layers, including the bottleneck. For many image translation problems, there is a great deal of low-level information shared between the input and output, and it would be desirable to pass this information directly across the net. For example, in the case of image colorization, the input and output share the location of prominent edges. To give the generator a means to overcome the bottleneck for information like this, they add skip connections, following the general shape of a "U-Net". Specifically, they add skip connections between each layer i and layer $n - i$, where n is the total number of layers. Each skip connection simply concatenates all channels at layer i with those at layer $n - i$.

C. Optimization and inference

We alternate between one gradient descent step on D, then one step on G. Rather than training G to minimize $\log(1 - D(x, G(x, z))$, we instead train to maximize $\log D(x, G(x, z))$. In addition, we divide the objective by 2 while optimizing D, which slows down the rate at which D learns relative to G. We use minibatch SGD, with a learning rate of 0.0002, and momentum parameters $\beta_1 = 0.5, \beta_2 = 0.999$. At inference time, we run the generator net in exactly the same manner as during the training phase. This differs from the usual protocol in that we apply dropout at test time, and we apply batch normalization using the statistics of the test batch, rather than aggregated statistics of the training batch. This approach to batch normalization, when the batch size is set to 1, has been termed "instance normalization" and has been demonstrated to be effective at image generation tasks. In our experiments, we use batch sizes between 1 and 10 depending on the experiment

III. Results

The dataset we took was CMP_facade_DB_base dataset from "http://cmp.felk.cvut.cz/∼tylecr1/facade/". The dataset is of facade images assembled at the Center for Machine Perception, which includes 378 rectified images of facades from various sources, which have been manually annotated. The facades are from different cities around the world and diverse architectural styles. We resized the images according to our need and converted them into two numpy arrays, one for actual images and other for their drawings. We ran our program on it for around 10,000 epochs. Good results started coming in around 2000 epochs only. The result after 2,000 epochs and 10,000 epochs is shown in Fig. 3 and Fig. 4 respectively. Images after every 500 epochs is availabe on the given github link. Code and output is availabe on "https://github.com/misterpawan/MTP2019-ImageStyleTransfer".

IV. Conclusion

The results in the paper suggest that conditional adversarial networks are a promising approach for many image-to-image translation tasks, especially those involving highly structured graphical outputs. These networks learn a loss adapted to the task and data at hand, which makes them applicable in a wide variety of settings. Our results showed pretty satisfactory results too.

V. Acknowledgement

This project was based on the paper "Image-to-Image Translation with Conditional Adversarial Networks", Authors: Phillip Isola, Jun-Yan Zhu, Tinghui Zhou and Alexei A. Efros.

Fig. 3. Output after 2000 epochs. 1st, 4th and 7th column is the drawing, 2nd, 5th and 8th is output of our program and 3rd, 6th and 9th is the actual image respectively.

Fig. 4. Output after 10000 epochs. 1st, 4th and 7th column is the drawing, 2nd, 5th and 8th is output of our program and 3rd, 6th and 9th is the actual image respectively.