Image-to-Image Translation with Conditional Adversarial Networks

Vinay Ummadi

SMST, IIT Kharagpur, India.

Abstract

This project is an implementation project of the original paper Pix2Pix. Generative adversarial networks have become very popular due to their capability of capturing training data distribution. Image to image translation is a task where an image given in one domain is translated into another domain, Ex Image to draw. This task of the image-to-image translation would require heavily engineered loss functions. Conditional adversarial networks are one kind of GANs which require input-output pairs for learning. So they fall under supervised learning problems. Given the input-output pairs, these networks learn the mapping function from input to output.Now we have a generalized GAN that can do most image-to-image translation tasks with reasonable accuracy. Labels to maps, edges to objects, segmented maps to images are now accessible. For code and results, please visit https://github.com/ummadiviany/Pix2Pix.

1 Introduction

There are many problems in the computer vision areas that require translating an image from one domain to another domain. For example, an image can be translated to edge map, semantic label map etc. In all of the tasks mentioned above, the structure of the problem is same. We have to learn a mapping function from input pixels to output pixels. Convolutional neural networks are used in most computer vision tasks. CNN's try to minimize a loss function that scores the predictions. In the case of translation, a new loss function has to be framed for every case. Although the setting of the problem is similar in most of the translation problems. The results will be blurry if we try to minimize the Euclidean distance between the generated and ground truth images. This is because of the averaging effect of minimizing Euclidean distance. Framing a loss function for sharp and realistic images will require good knowledge of the problem. We want an network which can automatically learn a loss function for the given problem. Fortunately, we have GANs [1], which do the same. GANs have a discriminator that discriminates between real and fake images and a generator that tries to generate real-looking images that deceive the discriminator. Blurry images will not be encouraged because the discriminator will catch them. As GANs learn a distribution of the data, CGANs [6] learn it in a conditional setting.

2 Related Work

Often image to image to translation tasks are formulated as per-pixel classification or regression tasks. In a sense, they are considering each of the output pixels are conditionally independent to pixels in the input image. But this is not true since a group of pixels forms structures and shapes in an image. CGANs learn a structured loss function that penalizes the joint configuration.

3 Method

GANs learn the mapping function from random noise vector z to output image y, $G : z \to y$. Conditional GANs learn the mapping function from z to y while given observed x . $G : x, z \to y$. Generator G is trained to produce real looking images which cannot be distinguished by discriminator D, which is trained to detect the generator fakes.

3.1 Objective

The objective of the conditional GAN can be expressed as (1), where G tries to minimize, and D tries to maximize the same. i.e. $G^* = \arg\ min_G\ max_D\ L_{cGAN}$

$$L_{cGAN}(G, D) = E_{x,y}[log(D(x, y)] + E_{x,z}[1 - log(D(x, G(x, z)] \quad (1)$$

Previous approaches have shown that the use of additional loss, such as $L2$ is beneficial. Now the generator's job is to generate realistic images and near ground truth in the $L2$ sense. Since $L1$ encourages less blurring it used instead of $L2$.

$$L_{L1}(G) = E_{x,y,z}[||y - G(x, z)||_1] \quad (2)$$

So the final objective is

$$G^* = arg\, min_G\, max_D\, L_{cGAN}(G, D) + \lambda L_{L1}(G) \quad (3)$$

3.2 Network architectures

The architecture skeleton for both generator G and discriminator D is adopted from DCGANs[7]. Both generator and discriminator use blocks of the form Convolution-BatchNorm-ReLU. Ck denotes Convolution-BatchNorm[2]-ReLU layer with k filters. CDk denote Convolution-BatchNorm-Dropout[10]-ReLU layer with 50% dropout rate. All convolutional filters have a 4x4 filter size and applied with a stride of 2.

3.2.1 Generator

In image-to-image translation tasks, the mapping is from high-resolution input image to a high-resolution output image. In addition to that, the appearance of the image changes, but the underlying structure is identical. The structure in the input is aligned with the structure in the output. An

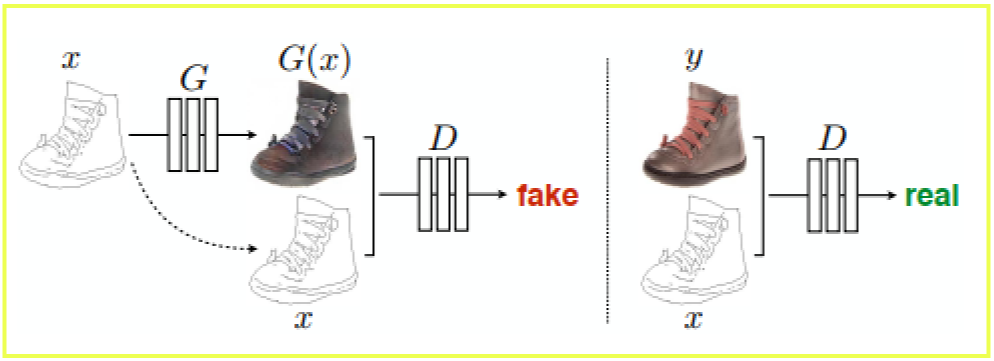


Figure 1: Conditional setting to map edge $\rightarrow$ photo. Discriminator learns to discriminate between generated and real tuples(edge, photo), while generator tries to fool discriminator by generating real-looking images. In conditional setting, both discriminator and generator observe input edge image.

Encoder-decoder network is often used in which, the input is passed through a series of layers that gradually downsample until a bottleneck layer, after which the process is reversed. For many image translation tasks, shared low-level information between input and output must be shuttled across the network. To enable this, skip connections are added, following the shape of U-Net[8] architecture. Skip connections are added between layer i and layer $n - i$, where n is the total number of layers.

Encoder **:** $C64 - C128 - C256 - C512 - C512 - C512 - C512 - C512$

Decoder **:** $CD512 - CD1024 - CD1024 - CD1024 - CD1024 - CD512 - CD256 - CD128$

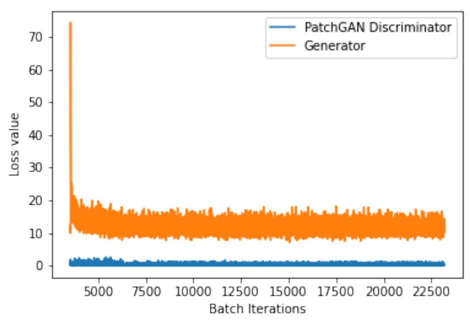
An additional convolutional layer followed by Tanh is used after the last layer of the decoder to map to the number of channels. In encoder all ReLUs are leaky with $\alpha = 0.2$, but not in decoder.

3.2.2 Discriminator(PatchGAN)

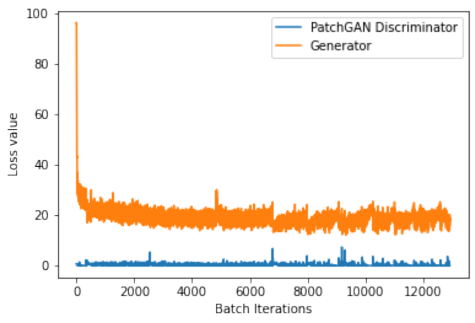
PatchGAN is a discriminator architecture that operates only on the patches of the original image. It classifies an NxN patch as real or fake. This discriminator network is applied convolutionally across the image, averaging all the responses produces a final output of D. The advantages of patchGAN are it has fewer parameters and runs faster. The 70x70 patch discriminator architecture is : C64-C128-C256-C512. An additional convolution layer is used to map to 1D output, followed by a sigmoid activation. All ReLUs are leaky with $\alpha = 0.2$.

3.2.3 Optimization and inference

Optimization is performed while alternating one gradient descent step on D, then one step on G. Adam optimizer [5] with a learning rate of 0.0002 and momentum parameters $\beta 1$=0.5 and $\beta 2$=0.999 are used. A step based learning rate scheduler is used to for learning rate decay. Learning rate is decayed by a factor of 10 for every 20 epochs.



(a) Arieal photo → Map training



(b) Anime sketch → Colourized anime sketch

Figure 2: Loss curves for datasets 1 and 2 are shown above

4 Experiments

To explore the generality of conditional GANS, the testing is performed on a variety of tasks, including graphics, like photo generation, and vision tasks, like semantic segmentation.

4.1 Datasets

- Arieal photo → Map dataset, from Kaggle [11]
  Training examples : 1096, Resolution : 600x600
- Anime sketch → Colourized anime sketch , Sketch colorization pair dataset from Kaggle [4]
  Training examples : 14224, Resolution : 512x512

4.2 Hyperparameter optimization

Although there are a large number of hyperparameters to crackdown, but I borrowed most of them from paper[3] and tuned a few. The architecture is fixed with PatchGAN discriminator and U-Net style generator right from the beginning. As we know learning rate is the crucial hyperparameter for the optimization, so it has to be chosen as per problem we are solving. Initial learning rate of 2e-4 worked well for learning high level patterns and learning rate decay helped to learn even minor details of the image. Batchsize is also one of the hyperparameter to play with, but authors[3] showed that batchsize=1 is the best. I have chosen the batchsize depending on dataset size and GPU memory to speed up the training process. There are few other hyperparameters like λ weight for L1 loss, etc are directly borrowed from the paper[3].

4.3 Training details

A cloud machine in Paperspace with 16 CPU cores, 30GB RAM and A NVIDIA Quadro P5000 16GB GPU is used for training. Different batch sizes are used depending on the dataset size to speed up the training process. Although the batch size of 1 is the best and shown by the authors [3].

Score	Mean	Standard Deviation
Translation	7.18	1.38
Difference	2.4	1.17

Table 1: Translation and Difference scores

Arieal photo → Map Maps dataset is a small dataset with only 1100 examples. Maps dataset has images of resolution 600x600, but due to computational constraints images are resized to 256x256 and then passed to the network. As a usual practice images are normalized by subtracting 0.5 mean and dividing with 0.5 standard deviation. Since we have relatively small number of examples, Random Color jitter with p=0.5 is applied to the maps datasets. Maps dataset is trained for 500 epochs with a batch size of 16. A validation set is used to check the performance of the generator.

Anime sketch → Colourized anime sketch Anime dataset has a good number of examples(14.5K) and is trained for only 100 epochs with a batch size of 32. Anime dataset has images of 512x512 resolution but due to computational constraints, we have resize images to 256x256. Images are normalized by subtracting 0.5 mean and dividing with 0.5 standard deviation. Tested on a subset of the anime test dataset. A validation set is used to check the performance of the generator.

More results are available at https://github.com/ummadiviany/Pix2Pix

4.4 Evaluation metrics - Self evaluation

Evaluating the quality of generated images is still an open-ended problem. While authors have used real vs fake perceptual studies as one metric, where the human plausibility is the final goal. Since the experimenter can rate the translated images, this still seems a better option as of now.

Evaluation of the translated images is done by myself while following the below process. Starting with a random sampling of 100 input-output pairs from the test set. Now input images are translated into the output domain using the generator. At this point, we have input-output pairs and respective translated images. The input image and translated image are visualized side by side. After looking in detail a *translation score* is marked. Once the translation score is marked, ground truth in translating domain is visualized alongside the translated image, for which a *difference score* is marked. The difference score emphasizes the differences between ground truth and translated in segmenting regions, painting colours etc.

Both of these scores are between 0 and 10. A low translation score means, translated image significantly differs from the input image and does not even contain any details of expected domain. A high translation score means the translated image perfectly has all the details of the desired domain. A high translation score and low difference score are ideally expected. Results are shown in Table 1.

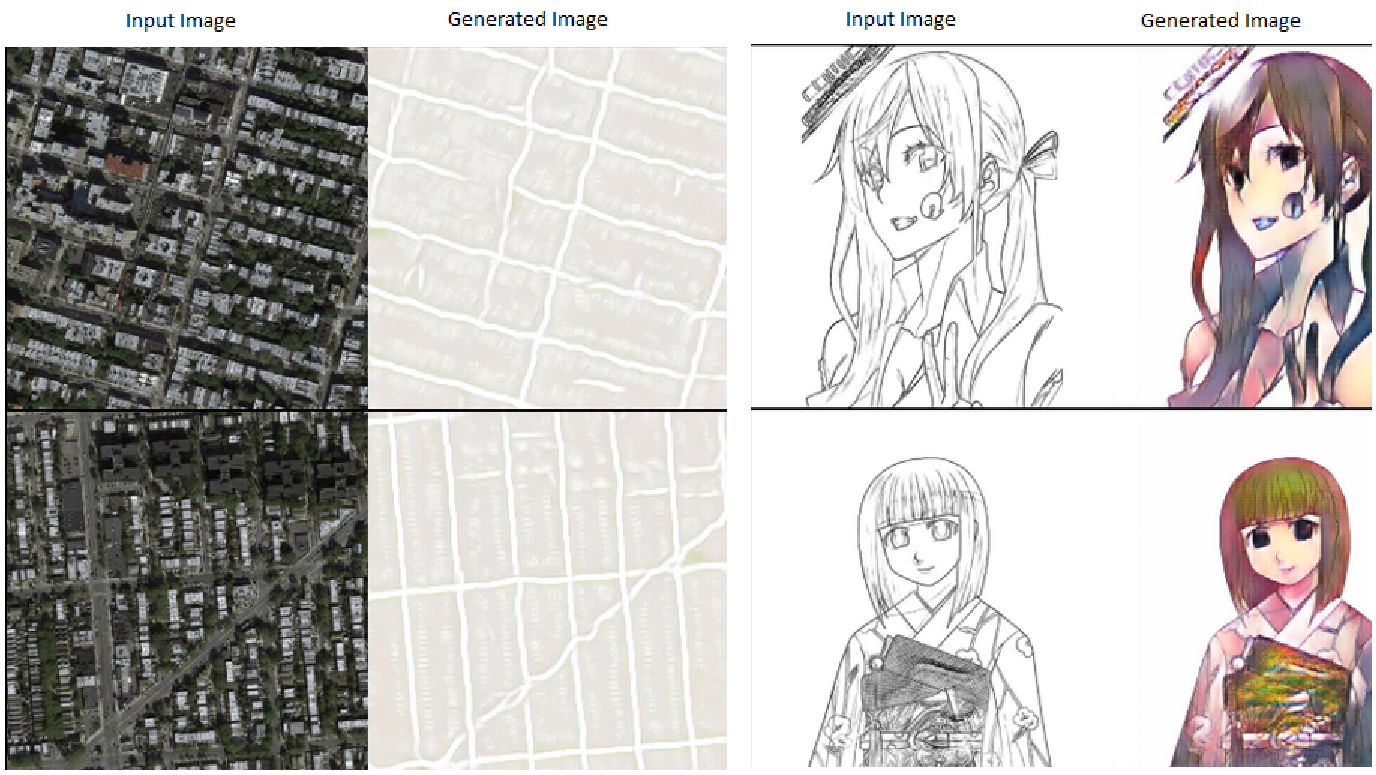


Figure 3: Example results from maps and anime dataset.

4.5 Results

Results seem to be pretty much close to actual results published by authors [3]. Authors[3] have tested this method on 9 different datasets. For this report, we are checking only on two datasets. There are few differences in results between authors[3] and this implementation. They[3] have produced much sharper results due to, inferencing on a original resolution without need for resizing. Starting with 256x256 top-left image patch is given to generator and translated image is saved, remaining parts of the input image are sent in a convolutional manner. Any how to compare, our networks have learnt a pretty good translations even not being trained for very long. In maps translation some minor details like arrow-marks on street lanes are missing. In anime translation full bright and dark are not being used by the generator.

5 Conclusion

From the results, Conditional GANs are promising for image-to-image translation tasks, which involve preserving structures. These networks learn a translation loss function, while given sufficient data without explicitly defining.

References

[1] Ian J. Goodfellow et al. *Generative Adversarial Networks.* 2014. arXiv: 1406.2661 [stat.ML].

[2] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.* 2015. arXiv: 1502.03167 [cs.LG].

[3] Phillip Isola et al. *Image-to-Image Translation with Conditional Adversarial Networks.* 2018. arXiv: 1611.07004 [cs.CV].

[4] Taebum Kim. *Anime Sketch Colorization Pair.* 2018. URL: https://www.kaggle.com/ktaebum/anime-sketch-colorization-pair.

[5] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization.* 2017. arXiv: 1412.6980 [cs.LG].

[6] Mehdi Mirza and Simon Osindero. *Conditional Generative Adversarial Nets.* 2014. arXiv: 1411.1784 [cs.LG].

[7] Alec Radford, Luke Metz, and Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.* 2016. arXiv: 1511.06434 [cs.LG].

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation.* 2015. arXiv: 1505.04597 [cs.CV].

[9] Tim Salimans et al. *Improved Techniques for Training GANs.* 2016. arXiv: 1606.03498 [cs.LG].

[10] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[11] Vikram Tiwari. *pix2pix dataset.* 2018. URL: https://www.kaggle.com/vikramtiwari/pix2pix-dataset.

[12] Jun-Yan Zhu et al. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks.* 2020. arXiv: 1703.10593 [cs.CV].