EEE F377
DESIGN ORIENTED PROJECT

APPLICATIONS OF GAN AND IT'S VARIANTS IN IMAGE-TO-IMAGE TRANSLATION

SUBMITTED TO -

DR. SUREKHA BHANOT

SUBMITTED BY -

RIGVITA SHARMA

2016A7PS0067P

1. **Table Of Contents**

2. Problem Formulation

In the project, I have reviewed variants of Generative Adversarial Networks for the purpose of the image to image translation. Firstly, I tried Conditional GAN which doesn't only learn the mapping from the input image to the output image, but also utilizes the loss function to train the network. The other variant is learning the mapping without paired training samples. In this variant, input images - x and y are not directly related to each other.

3. Objective

i. Understanding variants of GAN and their architectures.
ii. Understanding the various evaluation metrics used for GAN.
iii. Implementing GAN, CGAN, Pix2Pix and CycleGAN architecture.
iv. Comparing the results by variant the hyperparameters.

4. Introduction

There exist many problems in image processing, computer vision which can be posed as an image translation problem. Image to image translation can be defined as the task of translating the image from one representation to another. The goal of the architectures reviewed is to construct a common framework for all such problems. GANs are highly desirable for such problems as they learn a loss function to distinguish fake and real images while simultaneously training the generator. Hence, blurry images will not be tolerated as they will definitely look fake.

In the project, I worked upon understanding GAN variants- including GAN, CGAN and then using GANs in conditional settings to solve the image-to-image translation problem. Both paired(Pix2Pix) and unpaired(CycleGAN) training examples were used as the input.

Along with this, I studied various evaluation metrics for GANs - primarily Parzen Window Estimation.

5. **Generative Adversarial Network**

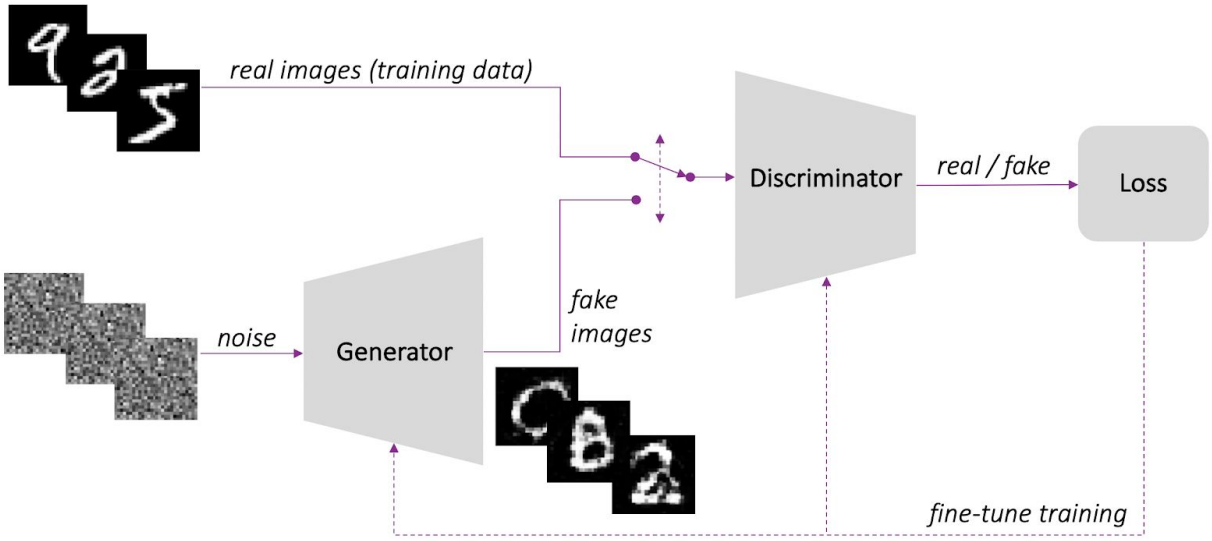


Fig 1. Architecture of Vanilla GAN

Generative Adversarial networks(GAN) were first introduced by Ian Goodfellow in 2014. Since then, GANs have been used to generate image synthesis, semantic image editing, style transfer, image super-resolution and classification. They have been powerful framework for unsupervised learning. In GANs, two models are simultaneously trained: Generator(G) and Discriminator(D). Firstly, the generator tries to generate fake data while the discriminator tries to tell the difference between the real data and the fake data generated by the generator.
To learn the generator distribution over data *x*, we feed input noise *z* from random distribution to the generator. It produces a mapping of fake generated data represented as *G(z)*. The second model D outputs a single scalar which represents whether the image is from real or fake distribution. D(x) represents the probability of *x* coming from the real data. We train D to maximize the probability of assigning correct labels to both training as well as fake samples. We simultaneously train G to minimize *log(1-D(G(x))).*
In other words, GAN is a two player min-max game between G and D with objective function:

$$min_G max_D V(D,G) = E_{x \sim p(data\ (x))}\ [log\ D(x)] + E_{z \sim p(z)}[log(1 - D(G(z)))]$$

where p(data(x)) is the distribution of real data while p(z) is the input distribution for the generator.

In the objective function, the first term is the entropy that the data from real distribution (pdata(x)) passes through the discriminator. The discriminator tries to maximize this to 1. The second term is the entropy that the data from random input (p(z)) passes through the generator, which then generates a fake sample which is then passed through the discriminator to identify the fakeness. Discriminator tries to maximize it to 0. So overall, the discriminator is trying to maximize our function V. While the generator tries to minimize the function V.
The cost function for both G and D were:

$$J^{(D)} = -(½)E_{x\sim p(data\ (x))}[\log D(x)] - (½)E_{z\sim p(z)}[\log(1 - D(G(z)))]$$
$$J^{(G)} = -J^{(D)}$$

First term in $J^{(D)}$ represents feeding the actual data to the discriminator. The discriminator would want to maximize the log probability of predicting one, indicating that the data is real. The second term represents the samples generated by G. Here, the discriminator would want to maximize the log probability of predicting zero, indicating the the data is fake. The generator, on the other hand tries to minimize the log probability of the discriminator being correct.

Pseudocode

For number of epochs do

For k steps do

- Sample minibatch of m noise samples from noise prior $p_g(z)$.
- Sample minibatch of m examples from data distribution $p_{data}(x)$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\left[\log D\left(\boldsymbol{x}^{(i)}\right)+\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right]$$

End for

- Sample minibatch of m noise samples from noise prior $p_g(z)$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m}\log\left(1-D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$$

6. Conditional Generative Adversarial Networks(CGAN)

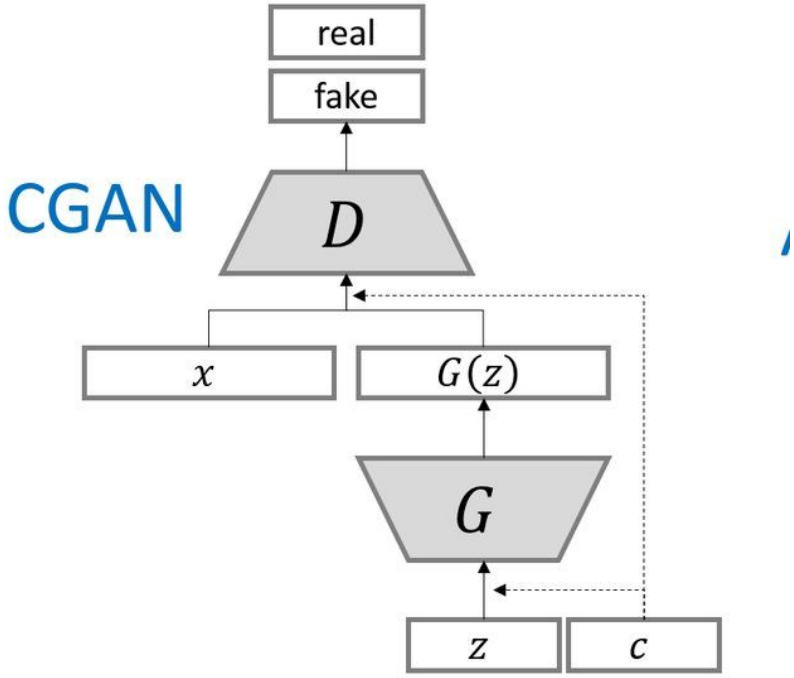


Fig 2. Architecture of CGAN

GANs can be extended to a conditional model if both the generator and discriminator are conditioned with extra information y. y could be any auxillary information such as class label or data from other modalities. y is fed to both the discriminator and generator as additional input layer. The objective function of a two player minmax game would be:

$$min_G max_D V(D,G) = E_{x \sim p(data\ (x))}\ [log\ D(x|y)] + E_{z \sim p(z)}[log(1 - D(G(z|y)))]$$

7. Pix2Pix

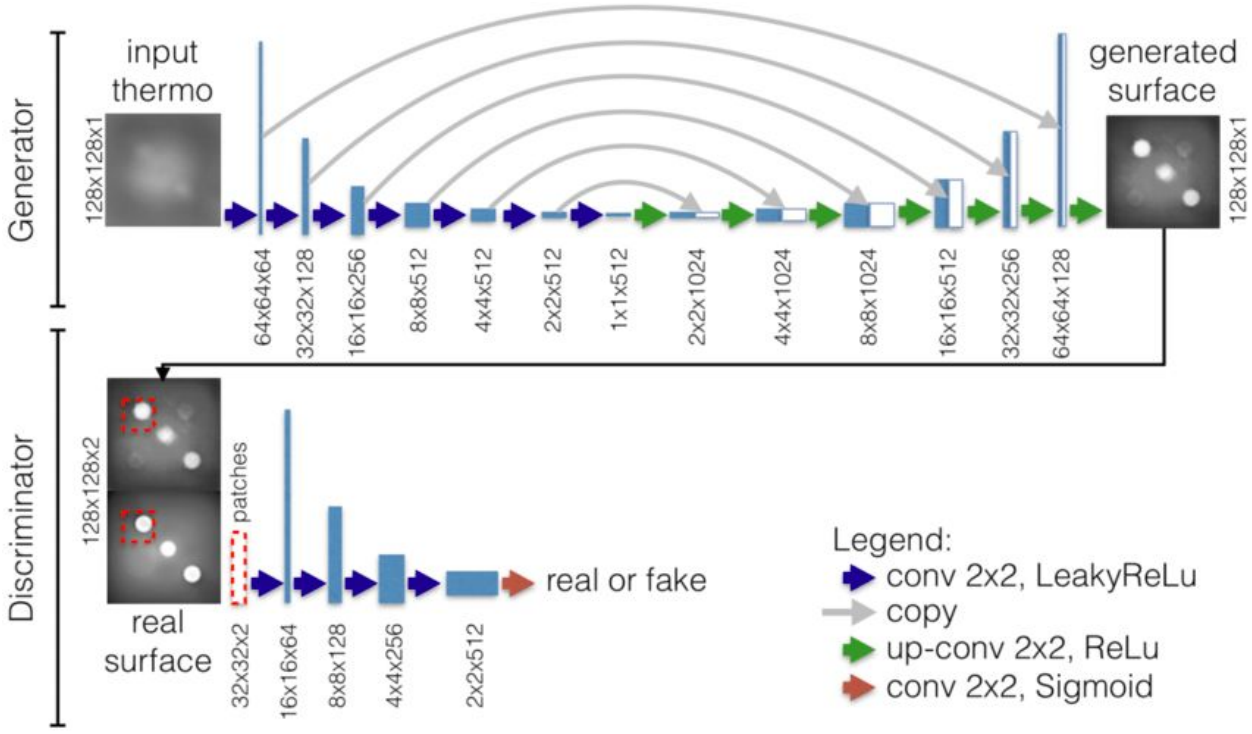


Fig 3 Architecture of Pix2PIx. Here, the Generator is a U-NET architecture and Discriminator is a PatchGAN

This explores the utilization of CGAN for the task of image-to-image translation task, where an input image is conditioned to generate a corresponding output image. It utilizes "U-Net"-based architecture for generator and "PatchGAN" classifier for discriminator.

a. <u>Objective:</u>

The discriminator's job remains unchanged, but the generator is tasked to not only fool the discriminator but also to be near the ground truth output in an (L1) sense. L1 distance is used instead of L2 as L1 encourages less blurring.

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

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

$$G^* = \arg\min_G \max_D L_{CGAN}(G,D) + \boldsymbol{\lambda}\, L_{L1}(G)$$

b. Network Architecture:

i. Generator with skips:

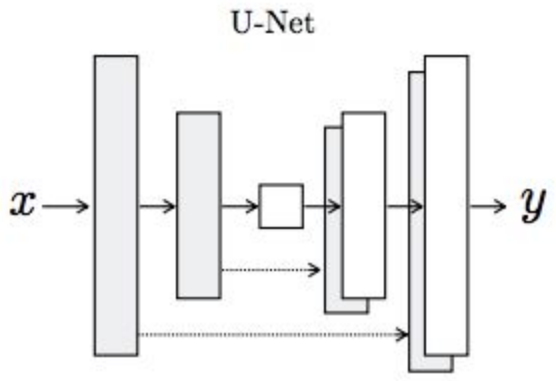


Fig 4. U-Net architecture. Encoder-decoder architecture with skip connections

The architecture used in generator is “U-NET”. The U-NET was developed by Olaf Ronneberger et al. for Bio Medical Image Segmentation. The architecture contains two paths. First path is the contraction path (also called as the encoder) which is used to capture the context in the image. The encoder is just a traditional stack of convolutional and max pooling layers. The input passes through a series of layers which progressively downsample, until a bottleneck layer. The second path is the symmetric expanding path (also called as the decoder) which is used to enable precise localization using transposed convolutions. Thus it is an end-to-end fully convolutional network (FCN), i.e. it only contains Convolutional layers and does not contain any Dense layer because of which it can accept image of any size.

To utilise the low level information shared in the layers, U-Net adds “skip connection”. Skip connections are added between matching layer pairs on either side of the bottleneck. ecifically, we 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.

ii. PatchGAN:

L1 and L2 loss function produces blurry image results, but are good to capture low frequency properties. Due to this, the GAN discriminator is restricted to just high frequency structure and utilizing L1 loss for low frequency. Hence, it is sufficient to restrict our attention to local patches. Thus, the architecture used in discriminator is “Patch-GAN” which only penalizes structure at the scale of patches. It tries to classify if each NxN patch in an image is real or fake. The discriminator is run across the image and the output D is average of patch responses.

8. CycleGAN

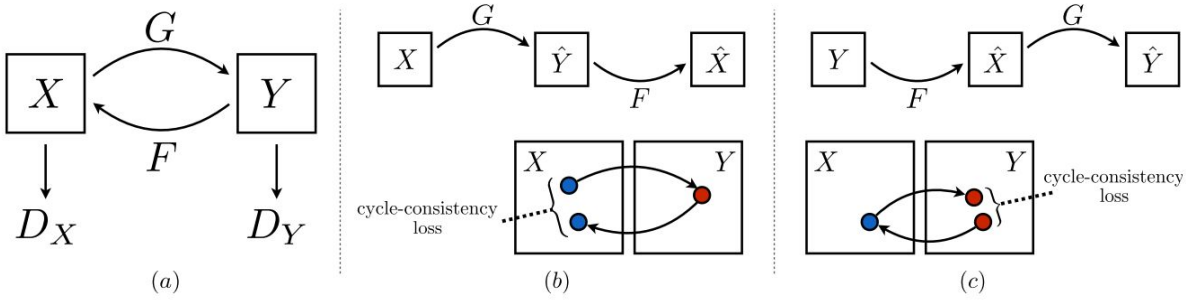


Fig 5. a)Network Architecture contains two Generators and two Discriminators. b)&c) The generators G and F are cycle consistent. L1 metric is used to calculate the cycle-consistency loss.

Image-to-image translation is a class of problems to map input image to output image using a training set of aligned image pairs. But, in most tasks, paired training datasets are not readily available. The goal is to learn the translation from source domain X to target domain Y given training samples $\{x_i\}$ such that $x_i \epsilon X$ and $\{y_j\}$ such that $y_j \epsilon Y$.
We use a deep network G: X→ Y to convert image x to y. We reverse the process with another deep network F:Y→X to reconstruct the image. Then, we use a mean square error MSE to guide the training of G and F.

In addition, we introduce two adversarial discriminators D_X and D_Y, where D_X aims to distinguish between images {x} and translated images {F(y)}; in the same way, D_Y aims to discriminate between {y} and {G(x)}.

a. Objective:

The objective function contains two types of terms: Adverserial Loss: for matching the distribution of generated image to the data distribution in the target domain and Cycle-Consistency Loss: to prevent the learnt mappings from G and F to contradict each other.

i. Adversarial Loss:

We apply adversarial loss to both the mappings. For the mapping G and its discriminator:

$$L_{GAN}(G,D_Y,X,Y) = E_{y\sim pdata(y)}[logD_Y(y)] + E_{x\sim pdata(x)}[log(1-DY(G(x))] \ ,$$

We have a similar adversarial loss for the mapping F.

ii. Cycle-Consistency Loss:

In theory, adversarial loss is enough to learn the mapping. However, with large enough capacity, a network can map the same set of input images to any random permutation of images in the target domain. Thus, adverserial loss alone can't guarntee desired output image. Also, to further reduce output space, the functions must be cycle consistent i.e $x \rightarrow G(x) \rightarrow F(G(x)) \sim x$. This is called forward cycle consistency. Similarly, $y \rightarrow F(y) \rightarrow G(F(y)) \sim y$ is called backward cycle consistency. Hence, the cycle-consistency loss:

$$L_{cyc}(G,F) = E_{x\sim pdata(x)}[\ \| F(G(x))-x \|_1] + E_{y\sim pdata(y)}[\ \| G(F(y))-y \|_1].$$

Hence, the full objective is :

$$L\ (G,F,D_X,D_Y) = L_{GAN}\ (G,D_Y,X,Y) + L_{GAN}\ (F,D_X,Y,X) + \lambda\ L_{cyc}(G,F)$$

Where λ controls the relative importance of the two objectives. We aim to solve:

$$G^*,F^* = arg\ min_{G,F} max_{Dx,DY} L(G,F,D_X,D_Y).$$

b. Network Architecture:

CycleGAN generator has three sections: an encoder, a transformer, and a decoder. The encoder shrinks the representation size while increasing the number of channels of the input. The encoder is composed of three convolution layers.The resulting activation is then passed to the transformer, a series of six residual blocks. It is then expanded again by the decoder, which uses two transpose convolutions to enlarge the representation size, and one output layer to produce the final image in RGB.

The discriminator are 70X70 PatchGANs, convolutional neural networks that look at a "patch" of the input image, and output the probability of the patch being "real" or "fake". It allows the discriminator to focus on more surface-level features, like texture. It takes fewer parameters and can also work on arbitarily sized images in fully convolutional fashion.

9. Problems in Training GANs

a. Non-Convergence

Optimization algorithms tries to approach local minima instead of global minima. GANs require finding the equilibrium to a game with two players. Even if player successfully moves downhill in player's update, same update might make the other player move uphill. Sometimes, the simultaneous gradient descent converges, but otherwise, players continue to undo each others progress and never converge. In practice, GANs often seem to oscillate.

b. Mode Collapse

Mode Collapse is the problem where generator collapses producing limited varieties of samples. It occurs when generator learns to map several input z to a small set of output points. Mode collapse may

arise because the maxmin solution to the GAN game is different from the minimax solution.

One approach to address the issue is Minibatch Features. The basic idea of minibatch features is to allow the discriminator to compare an example to a minibatch of generated samples and a minibatch of real samples.

Another approach is Unrolled GAN. In unrolled GAN, we build a computational graph describing k steps of learning in the discriminator, then backpropagate through all k of these steps of learning when computing the gradient on the generator.

c. Vanishing Gradient

This is common problem is deep learning but in GAN since the gradient of Discriminator propagates back to the generator, it gets stronger. Hence, no stability in training.

10. Parzen Window Estimation

In supervised learning, it is assumed that the underlying density function is known. It is mostly assume to be gaussian, exponential etc. But, in applications, the common parametric forms rarely fit the actual densities. Hence, we require non-parametric methods, where we don't want to assume any form of density. There are two types of Non-Parametric procedures:

i. Parzen Window (Region size is constant)
ii. Nearest Neighbours (Number of samples in Region are constant)

The most fundamental techniques rely on the fact that the probability P that a vector x will fall in a region R is given by

$$P = \int_R p(x') \, dx'$$

We can estimate value of p by estimating the probabilty P. Assuming n samples in total, $x_1, x_2, ..., x_n$ drawn independently and identically distributed according to p(x). Then by binomial equation, the probability of

k samples falling in R is

$$P_k = \binom{n}{k} P^k (1-P)^{n-k}$$

Then, the Expected value for k, E(k) = nP. Moreover, the binomial distribution for k peaks very sharply about the mean so we expect k/n to be a good estimate for P. i.e. P = k/n.
If p(x) is continuous and assuming, R to be small such that p doesn't vary significantly,

$$P = \int_{\Re} p(x')dx' \cong p(x)V$$

where V is volume of R.
Hence, p(x) = (k/n)/V. (Estimate of number of points falling in the volume V)
Theoritically, if we allow the number of samples n to go to infinity, however, we are able to establish convergence of the previous equation to equality. Let V_n be the volume of R_n, k_n be the samples falling in R_n and $p_n(x)$ be n^{th} estimate of p(x). For the convergence of

$$p_n(\vec{X}) = \frac{k_n/n}{V_n}$$

The following conditions must meet:

1. $\lim_{n\to\infty} V_n = 0$ For this condition, we assume that space average P/V will converge to p(x).
2. $\lim_{n\to\infty} k_n = \infty$ For this condition to meet, p(x) must not be zero.
3. $\lim_{n\to\infty} k_n/n = 0$ Although a huge number of samples fall in R, the fraction k/n is very small.

To satisfy the above conditions, one of the method is Parzen window.
Here the volume is fixed based on n. In this approach, the region is

considered either as a hypercube or gaussian distribution.

To obtain function for k_n, we define a window function, that tells us if the sample is in R:

$$\varphi(u) = \begin{cases} 1 & |u_j| \le \frac{1}{2} \quad j = 1,\dots,d \\ 0 & otherwise \end{cases}$$

We assign 1 to the sample if it lies within ½ of the edges of the hypercube and 0 if outside. Extending the approach to n samples we define a unit hypercube with x as a sample inside the hypercube of length h centered at x_i.

$$\varphi\left(\frac{x - x_i}{h}\right) = \begin{cases} 1 & |x - x_i| \le \frac{h}{2} \quad j = 1,\dots,d \\ 0 & otherwise \end{cases}$$

Thus,

$$k = \sum_{i=1}^{i=n} \varphi\left(\frac{x - x_i}{h}\right)$$ and,

$$p_\varphi(x) = \frac{1}{n} \sum_{i=1}^{i=n} \frac{1}{h^d} \varphi\left(\frac{x - x_i}{h}\right)$$

We can generalize this idea for other distributions too. If gaussian function is used, we have

$$p(x) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x_i - x)^2}{2\sigma^2}\right)$$

This is simply the average of n Gaussian functions with each data point as a center. Σ needs to be predetermined here.

We use non-parametric methods because of it's generality as compared to parametric methods. It has to be only used when number of samples are large as it might lead to curse of dimensionality. They should only be used we know that no simple density will work and the actual density is highly multimodal. These methods should only be used when the problem is highly complex and expensive calculations are not an issue, as they lead

to severe computation time and memory requirements.

11. Results and Discussion

GAN

In my experiment, I have used the MNIST dataset for training purpose.

a)Input Noise **b)After epoch 99000**

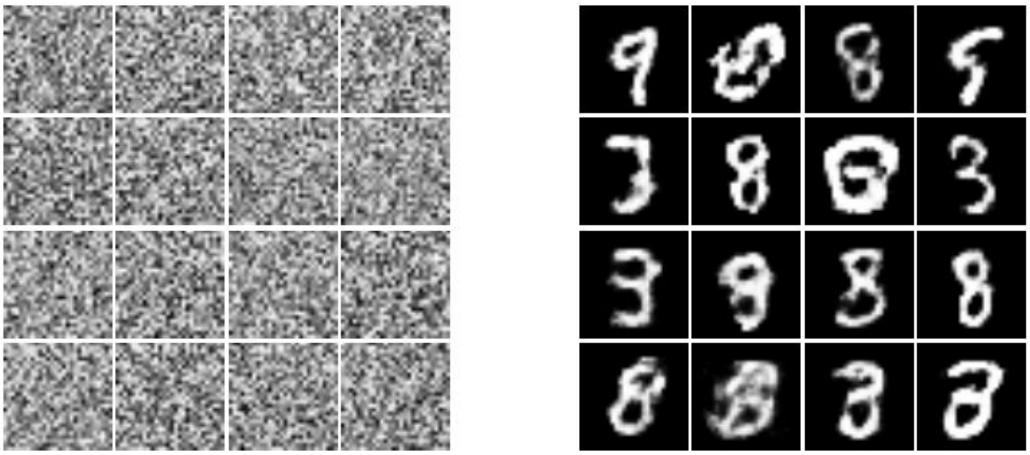


Fig 6. a)Input Noise fed to the generator b) Output after 99000 epochs

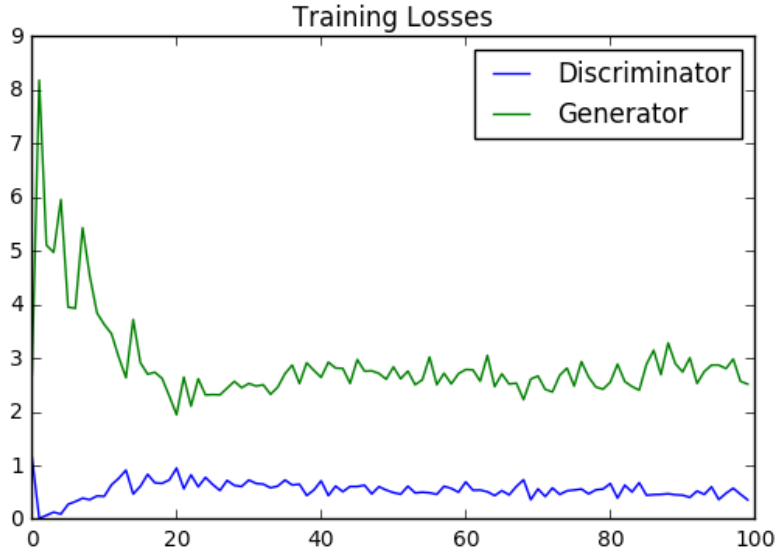


Fig 7. Graph for Training Loss (y-axis) against epochs%10000 (x-axis)

CGAN

I have trained a CGAN on MNIST images conditioned with their class labels encoded as one-hot vectors.

Input Noise **After epoch 5000**

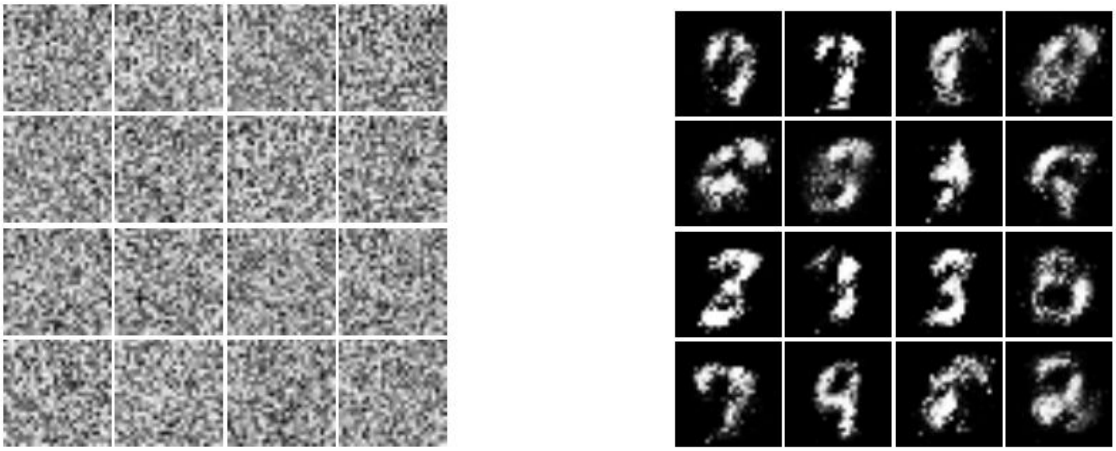


Fig 8. a)Input Noise fed to the generator b) Output after 5000 epochs

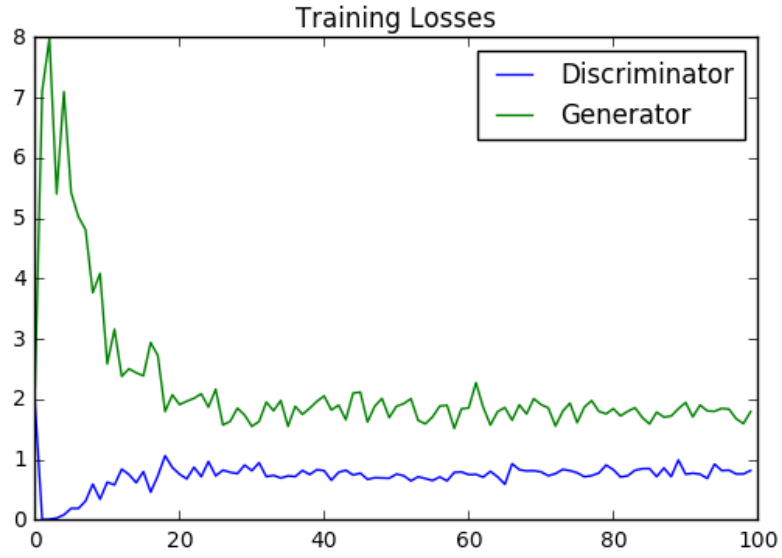


Fig 9. Graph for Training Loss (y-axis) against epochs%10000 (x-axis)

Pix2Pix

I have trained a Pix2Pix model on “facades” image dataset containing 400 training images, trained for 200 epochs. The images were first resized to 256 X 256 size. Weights were initialized from a Gaussian distribution with mean 0 and standard deviation 0.02.

Generator Architecture

Encoder-decoder architecture consists of:
Encoder: C64-C128-C256-C512-C512-C512-C512-C512
Decoder: CD512-CD512-CD512-C512-C256-C128-C64

where Ck denotes a Convolution-BatchNorm-ReLU layer with k filters. CDk denotes a Convolution-BatchNorm-Dropout-ReLU layer with a dropout rate of 50%. All convolutions are 4 × 4 spatial filters applied with stride 2. Convolutions in the encoder, and in the discriminator, downsample by a factor of 2, whereas in the decoder they upsample by a factor of 2. As an exception, BatchNorm is not applied to the first C64 layer in the encoder. All ReLU are leaky in encoder with slope 0.2 while ReLUs are not leaky in decoder.

In U-Net architecture, skip connections are also added between each i layer of encoder and n-i layer of decoder. Hence, the UNet decoder is :
CD512-CD1024-CD1024-C1024-C1024-C512-C256-C128

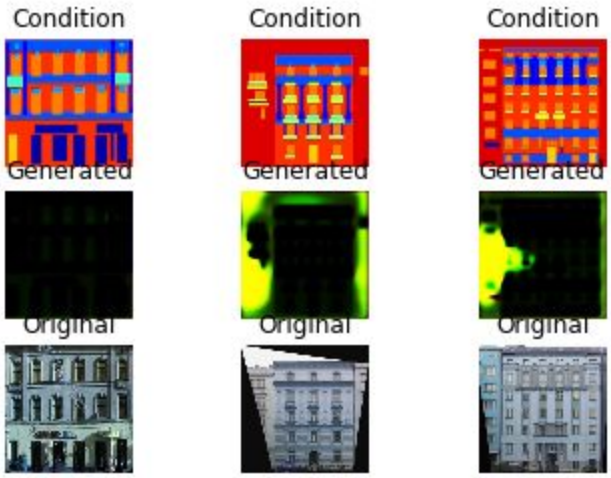
Discriminator Architecture

The 70 X 70 discriminator architecture is: C64-C128-C256-C512

After the last layer, a convolution is applied to map to a 1-dimensional output, followed by a Sigmoid function. As an exception to the above notation, BatchNorm is not applied to the first C64 layer. All ReLUs are leaky, with slope 0.2.

Loss Details

I have compared results for various combinations of losses. It can be observed from the results that Pix2Pix (adversarial loss and L1 loss) gives out the best results.



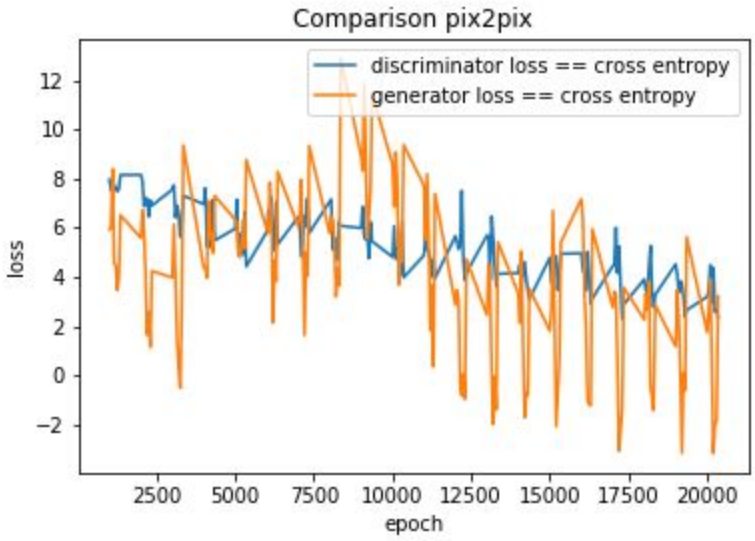
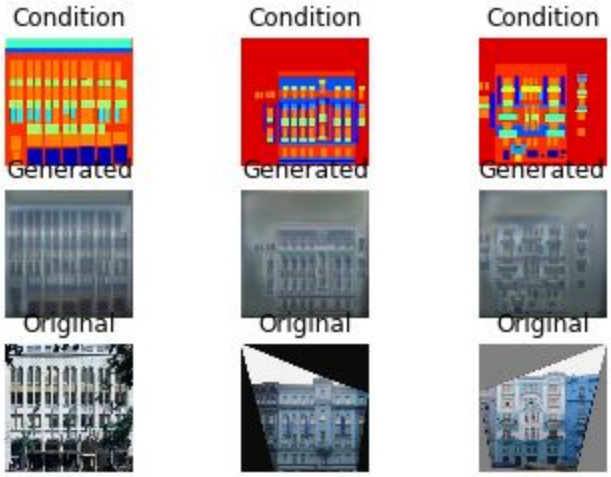


Fig.10 Pix2Pix model trained with only adversarial loss and adversarial loss calculated as cross-entropy.



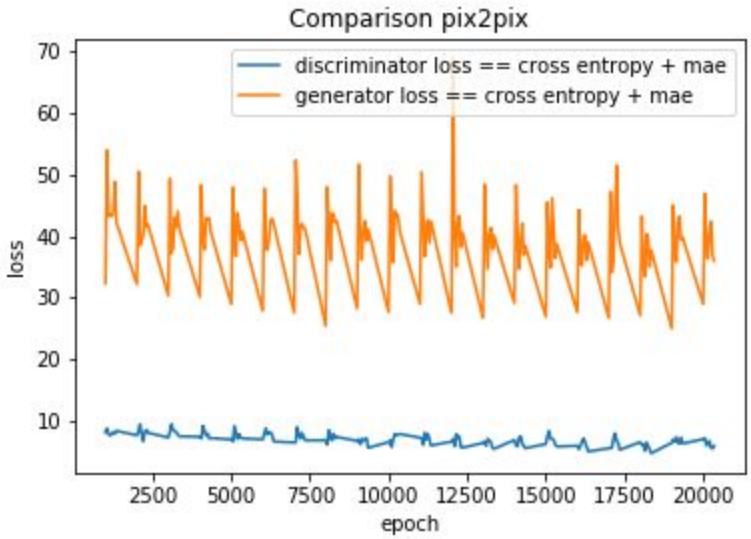
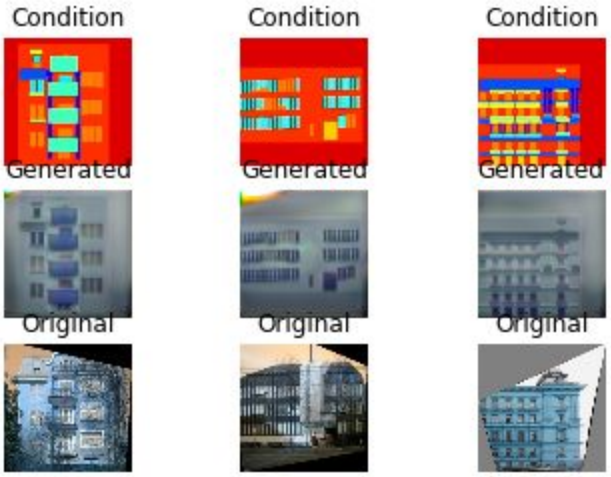


Fig.10 Pix2Pix model trained with a combination of adversarial loss and L2 loss. adversarial loss calculated as cross-entropy.



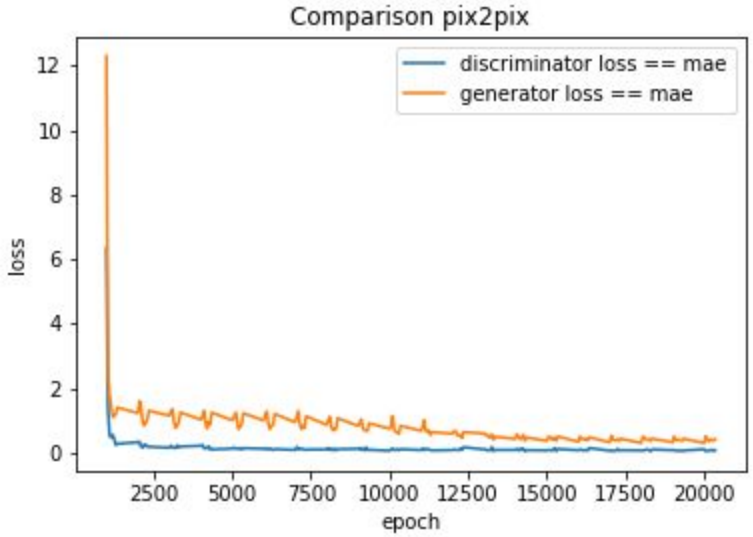
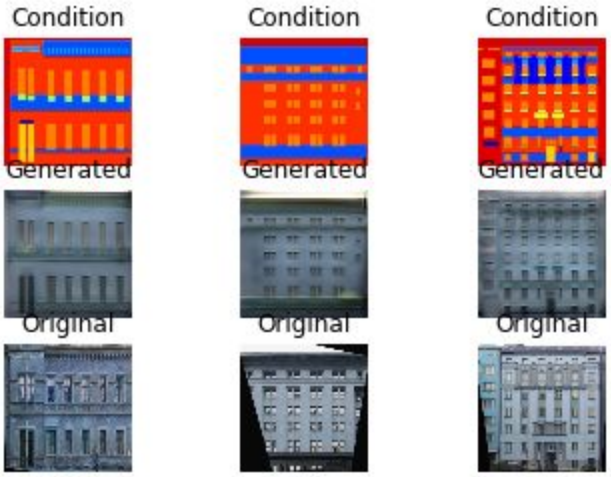


Fig.11 Pix2Pix model trained with just adversarial loss. adversarial loss calculated as mean squared error.



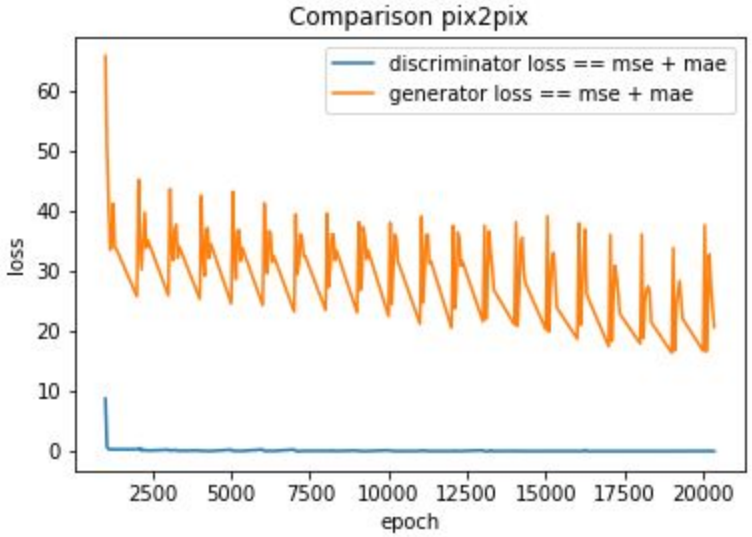


Fig.12 Pix2Pix model trained with adversarial loss and L1 loss as in the Pix2Pix paper. adversarial loss calculated as mean squared error.

CycleGAN

The network is trained with a learning rate of 0.0002. Weights are initialized from a Gaussian distribution N(0,0.02). Facades dataset is being used which contains 400 training samples.

Generator Architecture:

We use 9 residual blocks for 256 X 256 resolution images.

c7s1-64, d128, d256, R256, R256, R256, R256, R256, R256, R256, R256, R256, u128, u64, c7s1-3
Where c7s1-k denote a 7 × 7 Convolution-InstanceNorm- ReLU layer with k filters and stride 1. dk denotes a 3 X 3 Convolution-InstanceNorm-ReLU layer with k filters and stride 2. Reflection padding was used to reduce artifacts. Rk denotes a residual block that contains two 3 X 3 convolutional layers with the same number of filters on both layers. uk denotes a 3 X 3 fractional-strided-Convolution-InstanceNorm-ReLU layer with k filters and stride 1/2.

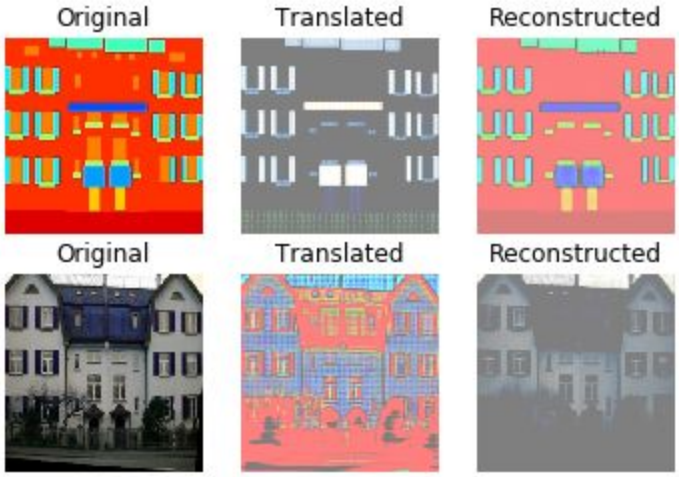
Discriminator Architecture

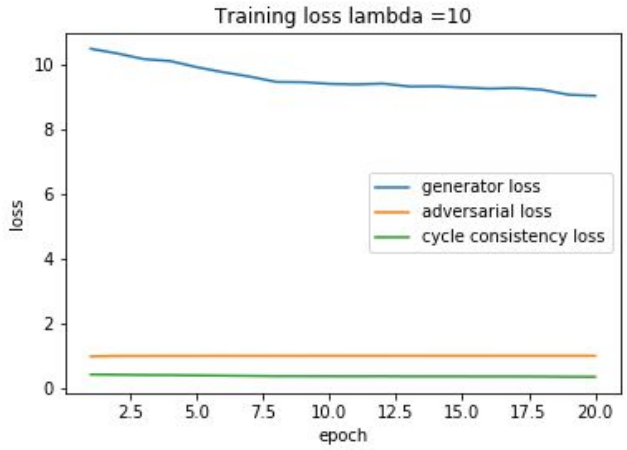
The 70 X 70 discriminator architecture is: C64-C128-C256-C512

Here Ck denotes 4 X 4 Convolution-InstanceNorm-LeakyReLU layer with k filters and stride 2. After the last layer, a convolution is applied to map to a 1-dimensional output, followed by a Sigmoid function. As an exception to the above notation, BatchNorm is not applied to the first C64 layer. All ReLUs are leaky, with slope 0.2.

Results

As can be observed from the results, we find lambda = 10 gives the best results. The texture and color are best retained as compared to other values of lambda. Other values of lambda produce hazy results. In the following page, we show the results and comparison of losses for different values of lambda, 1,5,10,15.





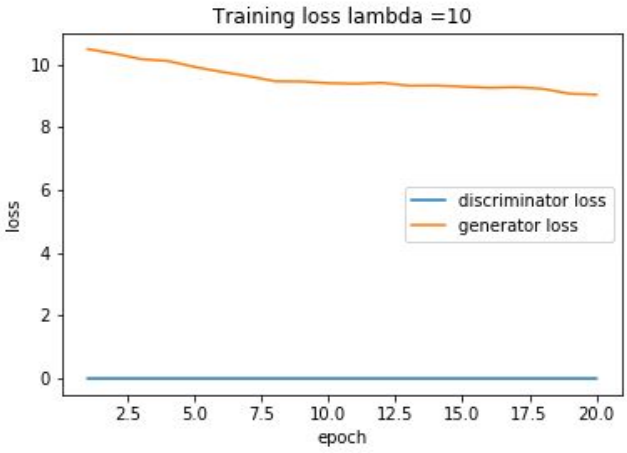
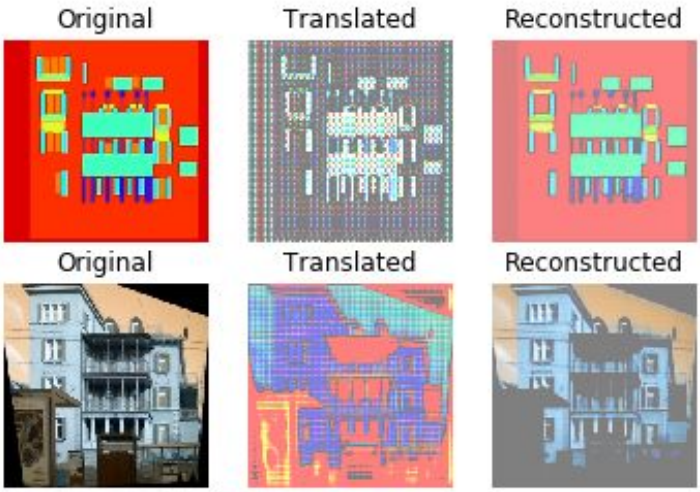
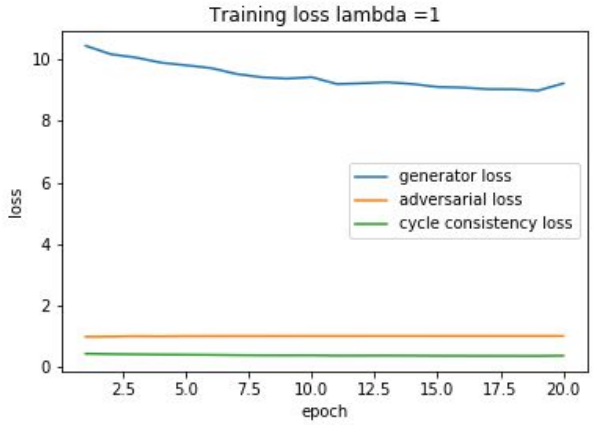


Fig 13. CycleGAN trained with lambda = 10.





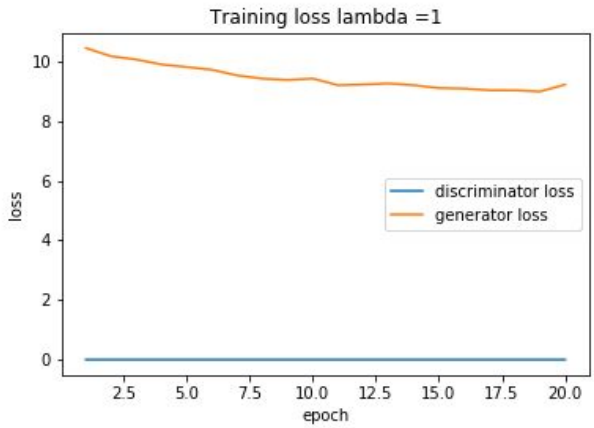
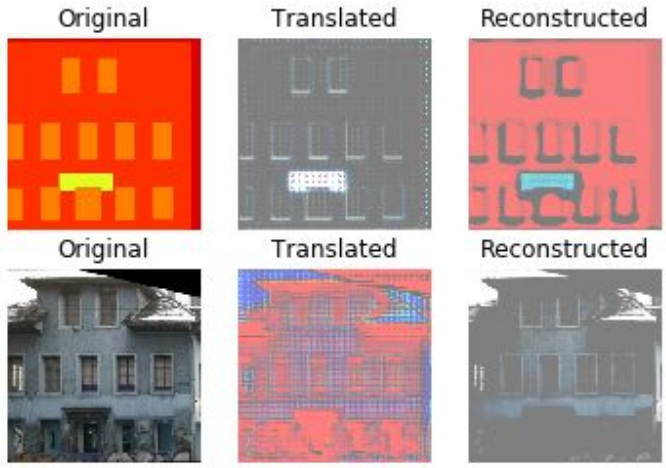
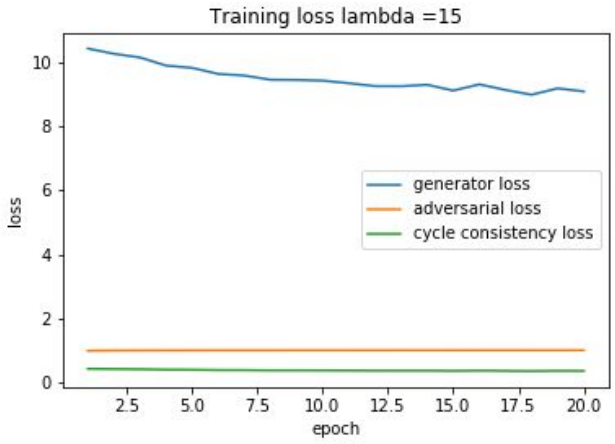


Fig 14. CycleGAN trained with lambda = 1.





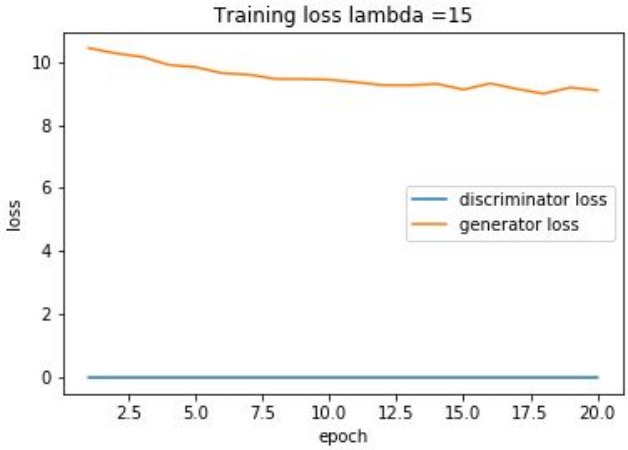
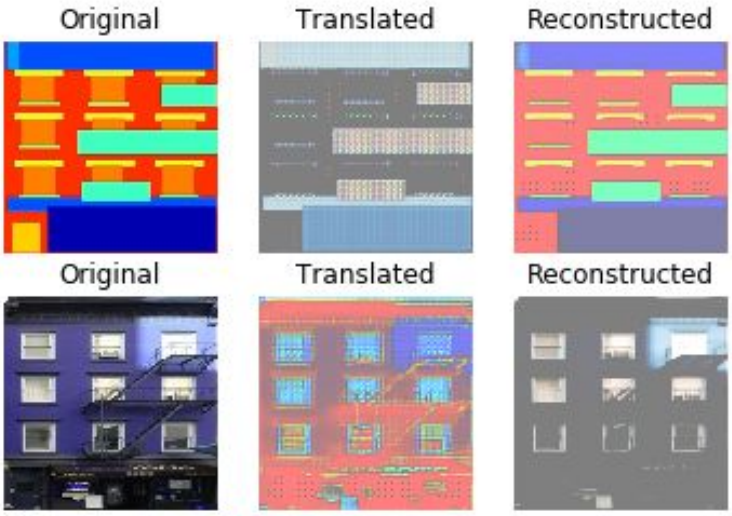
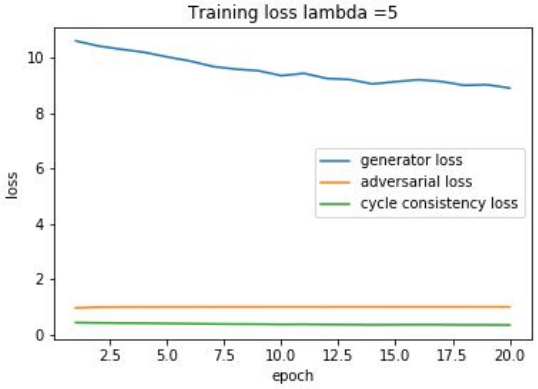


Fig 15. CycleGAN trained with lambda = 15.





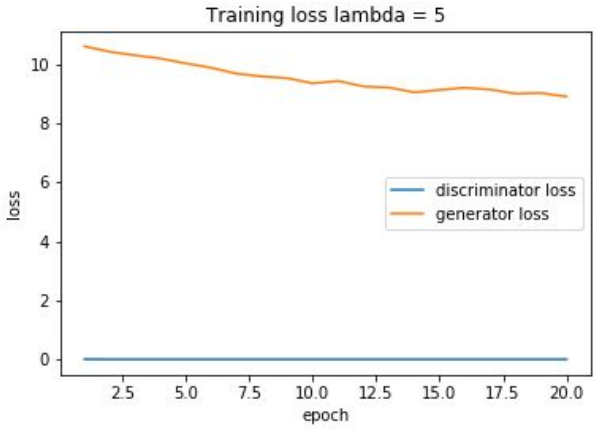


Fig 16. CycleGAN trained with lambda = 5.

12. Conclusion

- Here, we demonstrated how Pix2Pix and CycleGAN as an algorithm can be used for image translation tasks.
- We also trained GAN and CGAN to demonstrate the improvement due to conditioned label in the results on MNIST dataset.
- The results suggest that conditional adversarial networks can produce promising results in various image translation tasks.
- Results from Pix2Pix are quite closer to the input image. Though, the results still need improvement.
- Pix2Pix finds many applications in domains requiring high quality images where pre-trained network is not available with some improvements.
- Also, cycleGAN can achieve compelling results in many cases, but the results are far from uniformly positive.
- We also observe a gap between the results of the paired training dataset and unpaired training dataset. This gap might not be filled even after some improvement as cycleGAN uses unpaired training dataset and hence, has larger domain space.
- Hence, using a paired training dataset GAN approach is always better if we have sufficient training data.
- We also found that cycleGAN is more computationally expensive in terms of time and resource consumption as compared to Pix2Pix.
- CycleGAN results improves with the introduction of utilizing training results of last 50 epochs.
- CycleGAN works well for problems that involve color or texture. However, tasks that require substantial geometric changes to the image, such as cat-to-dog translations, usually fail.
- Translations on the training data often look substantially better than those done on test data.
- The probability of occurrence of Mode-Collapse or non-convergence can not be neglected as occurred during training any Generative Adversarial Network Architecture.

13. References

a. Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In NIPS'2014.
b. Ian Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks,
c. M. Mirza and S. Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784, 2014.
d. Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks.
e. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros
f. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation.
g. Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks, Chuan Li and Michael Wand
h. Deep Residual Learning for Image Recognition, Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun
i. Sequence to Sequence Learning with Neural Networks, Ilya Sutskever, Oriol Vinyals, Quoc V. Le
j. Fully Convolutional Networks for Semantic Segmentation, Jonathan Long, Evan Shelhamer, Trevor Darrell
k. Perceptual Losses for Real-Time Style Transfer and Super-Resolution, Justin Johnson, Alexandre Alahi, Li Fei-Fei
l. Improved Techniques for Training GANs. Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Alec Radford, Vicki Cheung, Xi Chen.
m. Pros and Cons of GAN Evaluation Measures, Ali Borj
n. Pattern Classification by Richard O. Duda, David G. Stork, Peter E.Hart
o. http://www.csd.uwo.ca/~olga/Courses/CS434a_541a/Lecture6.pdf
p. https://github.com/yassineELMAAZOUZ/Parzen-Window-Kernel-Density-Estimation/blob/master/projet_MAP.pdf
q. https://www.projectrhea.org/rhea/index.php/Parzen_Window_Density_Estimation
r. www.personal.reading.ac.uk/~sis01xh/teaching/CY2D2/Pattern2.pdf
s. https://milania.de/blog/Introduction_to_kernel_density_estimation_%28Parzen_window_method%29

t. http://ssg.mit.edu/cal/abs/2000_spring/np_dens/density-estimation/parzen62.pdf
u. https://sebastianraschka.com/Articles/2014_kernel_density_est.html
v. Classification with Deep Belief Networks, Hussam Hebbo, Jae Won Kim.
w. 2007 NIPS Tutorial on Deep Belief Nets, Geoffrey Hinton.
x. Hugo Larochelle, youtube videos on RBM and DBN.