Deep Image Manipulation: Image-to-Image translation

Jagata Venkata Sriram[1] and Dr. Nagesh Bhattu Srirsty[2]

No Institute Given

Abstract. The image-to-Image translation is a subcategory of computer vision problem, where the model learns to map from one domain to another: source to target while preserving a few main features like structure. Image-to-Image translation can be done in three different settings: supervised, semi-supervised and unsupervised. The image-to-Image translation came into the limelight with pix2pix paper in 2016. It attracted many eyes across social media platforms for its incredible image transformations. Pix2Pix uses paired data to learn to map while another Image-to-Image translation model cycleGAN uses unpaired data to transform images. Both use conditional GANs to solve problems related to image-to-image translations. cycleGAN learns loss function along with mapping from source to target image domains while pix2pix has a hand-designed loss function for a different kind of mapping. In this paper, we modify the normalization layer in pix2pix to enhance the speed of the model while retaining accuracy when a lower batch size is used. We also introduce a few modifications to address problems caused by cycle consistency in cycleGAN and also run a variant of cycleGAN which uses a U-NET generator with skip connections instead of a residual generator similar to the original. Finally, we compare the results of our modified cycleGAN against the U-NET variant and vanilla cycleGAN. We observed our modified pix2pix runs faster with almost similar accuracy to that of original and our modified cycleGAN converges at early stage with lower loss values when put against vanilla cycleGAN and U-NET variant.

Keywords: Image-to-Image Translation · Pix2Pix · cycleGAN · U-NET

1 Introduction

The usage of photo editing tools/softwares are increasingly popular these days due to the rise in accessibility of the internet, social media, smartphones and cameras. One of the recent surveys showed that about 25 percent of the apps downloaded are related to photo editing stuff. However, the capability of these apps to edit photos is limited, for example these apps allow us to tweak only lower level properties like brightness and contrast, but there is no better way to tweak or change higher level features like climate or season in the photo. Why is editing higher level features challenging? Most existing photo editing methods rely on hand designed techniques to reshuffle and transform the input pixels to simulate the desired editing effect. Certainly, there are good models, such as

finding the closest match in reference images by looking at nearby patches, combining different images across frequency bands, or morphing images by warping and blending. However, to add a particular aspect like rain or fog, its visual appearance needs to be acquired apriori, independent of the test images. Now we can see the prowess of machine learning. We can use separate dataset to gather visual information and create new artifacts which are realistic and not present in the input images. However, there seems to be a subtle problem in using deep learning models for image editing. There seems to be a fundamental conflict about what proportion of information is to be extracted from the dataset and how much information must be utilized from the input image. Problem with relying too much on input image shows no use of dataset whereas if more weight-age is given to dataset then the output image won't even closely match the input image. Deep learning algorithms like GANs are used to generate images, these algorithms typically learn from a dataset but cannot separate higher level features from images like texture, so using these algorithms to edit images like we need isn't that feasible. This problem is known as the disentanglement problem. When the amount of supervision is high i.e., pixel wise supervision then the cost of collecting ground truth for each pixel is very high, by reducing the level of supervision to image wise instead of per pixel then we see more versatility in manipulating an image.

2 Related Work

2.1 GANs for image-to-image translations

Since GANs[3] were introduced, many GANs besides CycleGAN[13] and Pix2Pix[4] have been proposed for image-to-image translation. We will briefly discuss few of them which are relevant for our work.

UNIT To execute image-to-image translation, the UNIT framework employs a combination of variational autoencoders[5] and generative adversarial networks, dubbed VAE-GAN. UNIT employs two VAE-GANs[14], one for each domain. The domains share the weights of the last few layers of encoders and the first few layers of decoders. This common lower-dimensional encoding can then be decoded to one of the two domains using their respective decoders.

Domain transfer network The domain transfer network (DTN)[16] is type of GAN with a single autoencoder as the generator, which receives input from both the source and target domains and maps it to the target domain during training. Only if the input image is from the target domain is the reconstruction loss applied. If the input is from the source domain, a loss is applied to minimise the distance between the encoded input and the encoded output.

DiscoGAN DiscoGAN[15] is similar to CycleGAN: both employ a re-construction loss to ensure pixel-level cycle-consistency. DiscoGAN, on the other hand, uses cross-entropy for the adversarial loss rather than the mean squared error used by CycleGAN. DiscoGAN also does not employ a PatchGAN discriminator or residual generator.

Pix2Pix[4] Conditional adversarial networks learn input to output image mapping along with loss function unlike other previous methods. This made the hectic task of hand-engineered loss functions less as the network itself learns loss function. This helps to generalize loss functions instead of applying multiple loss functions for a problem. Minimize Euclidean distance between pixels of predicted image and ground truth image while training CNN images tend to be blurry in nature.

Objective function

A simplistic approach to Image-to-Image translation would be to completely disregard the adversarial framework. The weights of the network would be updated by passing a source image through a parametric function and comparing the resulting image to the ground truth output. However, using traditional distance measurements like L1 and L2 to create this loss function will miss many of the crucial distinguishing properties between these photos. The L1 loss function, on the other hand, has some usefulness as a weighted sidekick to the adversarial loss function, according to the authors.

$$L_{cGAN}(G,D) = \mathop{\mathbb{E}}_{x,y} \log D(x,y) + \mathop{\mathbb{E}}_{x,z} \log\left(1 - D(x, G(x,z))\right) \tag{1}$$

$$L_{L1}(G) = \mathop{\mathbb{E}}_{x,y,z} [||y - G(x,z)||_1] \tag{2}$$

using above two equations we get final object function:

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

2.2 cycleGAN

An approach for learning to translate an image from a source domain X to a target domain Y in the absence of paired examples. As mentioned in the paper[13]. the goal is to learn a mapping G : X $\rightarrow$ Y such that the distribution of images from G(X) is indistinguishable from the distribution Y using an adversarial loss. Inverse mapping F : Y $\rightarrow$ X and introduce a cycle consistency loss to enforce F(G(X)) $\approx$ X (and vice versa).

Objective function

$$L_{cyc}(G,F) = \mathop{\mathbb{E}}_{x \sim p_x(x)} [||F(G(x)) - x||_1] + \mathop{\mathbb{E}}_{y \sim p_y(y)} [||G(F(y)) - y||_1] \tag{4}$$

$$L(G, F, D_x, D_y) = L_{GAN}(G, D_y, X, Y) + L_{GAN}(F, D_x, Y, X) + \lambda L_{cyc}(G, F) \quad (5)$$

final objective function

$$G^*, F^* = \arg\min_{G,F} \max_{D_x, D_y} L(G, F, D_x, D_y) \quad (6)$$

3 Method

Image-to-Image translation is one of the image manipulation methods which produces exciting results in the world of computer vision. Pix2pix a variant of Generative Adversarial Network(GAN) gained attention of many Ml enthusiasts and researchers across social media with its translation of one image to another popular one is edges to cats. Trained using unpaired data, Cycle-Generative Adversarial Networks (CycleGAN) achieves amazing translation results in many cases where paired data is not present or tough to attain, such as Monet paintings to photos, zebras to horses, etc. Image-to-image translation plays a vital role in domain adaptation. For safety purposes, robots and artificial intelligent machines are often trained in simulated environment to avoid accidents. They also use artificially generated data or synthetic data for this simulation. Few interesting Image-to-Image translations can be images to art style of a certain artist, we can get to see how a particular photo might look when painted by a popular artist of our choice , for example photo to Van Gogh style. Another interesting take can be translating real life data into images for simulation of robots and machines. In this chapter, we identify some existing problems with the CycleGAN framework. Main focus of our work is to improve the cycle consistency loss.

3.1 Modified pix2pix

As we have seen in the previous chapter, Pix2Pix learns input-to-output mapping with a loss function. We also know that it uses the BatchNorm[2] method while training in both the generator and discriminator. We have used groupNorm[] instead of batchNorm as it isn't effective when the batch size is small. The error percentage increases as the batch size decreases, usually from batch size 8, gets worse for batch size 2. If the dataset is large, using a batch size of 32 will yield better results. groupNorm works better when the batch size is small and works similar to batchNorm when the batch size is large, so we decided to use groupNorm instead of batchNorm, keeping the rest of the architecture similar to the original one. Initially group convolutions were introduced in AlexNet[7] which helped people to see concepts of groups as a dimensional aspect in a model. It was studied that larger group numbers improve accuracy with similar computational cost.

GroupNorm layer calculates μ and σ in a set A_i as:

$$A_i = \{x | x_N = y_N, \frac{x_C}{C/Gr} = \frac{y_C}{C/Gr}\} \quad (7)$$

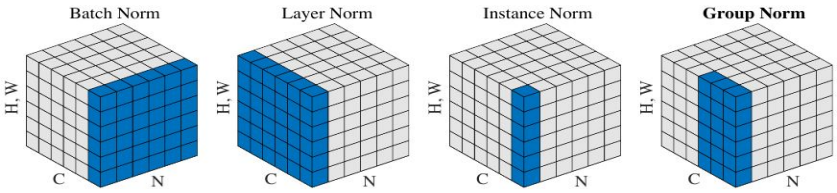


Fig. 1: Each cube is feature map, with N as the batch axis, C as the channel axis, and (H, W) as spatial axes. The blue area is pixels normalized by same stats.[12]

Here Gr represents number of groups, hyper-parameter which is pre-defined with default value 32. C/Gr denotes channels per group. x and y are indexes in the same group of channels in seq manner along C axis. GroupNorm calculates μ and σ along the (H,W) axes and along C/G channel group.

Using this knowledge we use groupNorm-RELU blocks instead of BatchNorm-RELU blocks. After experimenting with several batch sizes, we found group sizes of 6 and 8 are optimal for the dataset

3.2 Modified cycleGAN

As we have seen in the above section cycleGAN uses inverse mapping and cycle consistency loss to translate unpaired images.

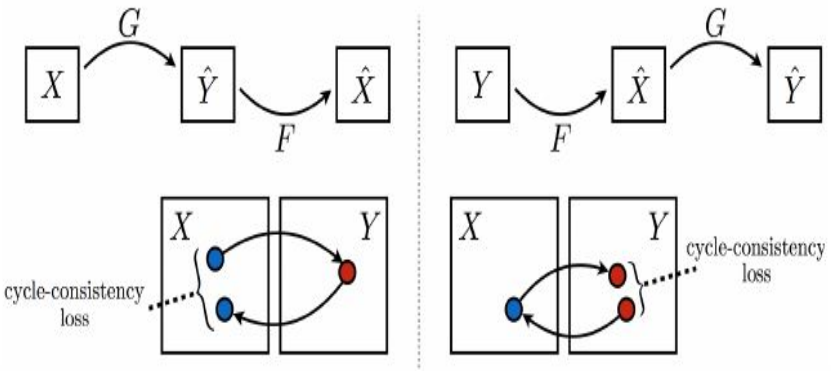


Fig. 2: cycle consistency.[13]

Fig 2 shows how cycle consistency is enforced through cycle consistency loss in order to train unpaired data. In broader perspective, cycle consistency forces generators to avoid needless changes, resulting in images with structural resem-

blance to inputs. During our experimentation, we discovered that cycle consistency guides training by rapidly driving generators to produce visuals that are similar to inputs with basic colour mappings. At earlier stage during training period, at the epoch 4 to 5, the generator learns a near-identity mapping. At training epoch 10, the generator learns to map color of the grass which can be from grey to green or yellow to green in zebra horse translation. When we examine the training dataset, we see that photos from the horse dataset have more green grass when compared to photos from zebra dataset. Because colour mappings are frequently easily reversible, cycle consistency loss and GAN loss guide or lead generators more effectively to create images with realistic colours, which is critical to whether they appear realistic.

Regularize: Cycle consistency is also a type of regularisation. By maintaining cycle consistency, the cycleGAN architecture protects generators from excessive hallucinations and mode collapse, both of which cause superfluous information loss and hence increase cycle consistency loss. Even while cycle consistency is excellent, it is not without flaws. When the cycle consistency is used at the pixel level, information loss is non-existent as it assumes one-to-one mapping between two image domains.

3.3 Modified cycle consistency

Cycle consistency is fine and dandy. As we have seen in the above section it produces undesired results due to very strong assumption. We made few changes to cycle consistency to rectify issues mentioned above.

Cycle consistency on feature level Almost always, information is lost during the translation process. We expect cycleGAN to keep overall structure of the image intact(i.e., horse outline should be intact) but not exact pixel to pixel from the image. For example, in the zebra-to-horse translation, the cycle should be regarded consistent if the rebuilt zebra image contains main features of zebra like stripes, they can be in any manner horizontal, vertical, curvy or straight. We added L1 loss on extracted CNN features from the discriminator as the weaker notion of cycle consistency. We assume that discriminator of our model has learnt good features from the image dataset. This new cycle consistency loss is linear combination of feature level(CNN) and pixel level losses:

$$L_{cyc}(G, F, D_x, D_y, \alpha) = \mathop{\mathbb{E}}_{x \sim p_x(x)} [\alpha ||z_{D_X}(F(G(x))) - z_{D_x}(x)||_1 + (1-\alpha)||F(G(x)) - x||_1] \quad (8)$$

where $z_{D(.)}$ extracts features using $D_{(.)}$ and $\alpha \in [0,1]$ represents ratio between CNN level loss at discriminator and pixel level loss. During our training we observed varying α at an instant(epoch) is better. Discriminator characteristics are poor at first so start with lower values then, linearly increase to a high number close 1 to but not equal to 1. When equals 1 there won't be any pixel

level consistency which causes excessive hallucinations and generated irrelevant images.

Weight decay for cycle consistency Cycle consistency loss helps to stabilise training in the early stages but becomes an impediment to realistic images later on. The weight of cycle consistency loss λ is gradually reduced as we train the model. λ should not be reduced to zero as it may lead generator to go berserk. Cycle consistency makes no sense when generated images are unrealistic during earlier epochs. Instead of attempting to make realistic images, both the generators can bring down the cycle consistency loss by learning color inversion mapping. The generators are unlikely to escape once stuck in local mode due to cycle consistency. As a result, imposing cycle consistency on cycles with unrealistic generated visuals actually inhibits training. By adding discriminators' output we resolve the above issue which is obtained quality of generated images. New equation is:

$$L_{cyc}(G, F, D_x, D_y, \alpha) = \\ \mathop{\mathbb{E}}_{x \sim p_x(x)} [D_X(x)(\alpha || z_{D_X}(F(G(x))) - z_{D_x}(x)||_1 + (1-\alpha)||F(G(x)) - x||_1)] \quad (9)$$

This helps us to balance both cycle consistency and GAN losses during earlier epochs of training. It basically prioritizes generators to produce realistic images over cycle consistency. We didn't back propagate gradient of this loss to discriminator as generators are constrained on cycle consistency.

Full objective function By stitching above modifications we obtain the following objective function at every epoch e:

$$L_{cyc}(G, F, D_X, D_Y, e) = L_{GAN}(G, D_Y, X, Y) + L_{GAN}(F, D_X, Y, X) \\ + \lambda_e L_{cyc}(G, F, D_X, X, \alpha_e) + \lambda_e L_{cyc}(F, G, D_Y, Y, \alpha_e) \quad (10)$$

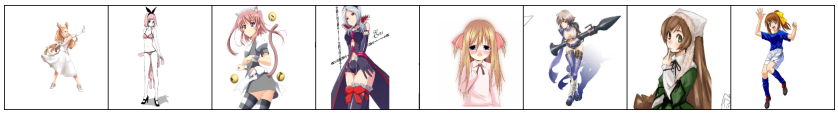
Decrease λ_e value linearly and increase α_e value linearly close to 1 to yield better results.

4 Experimental Analysis

We have used NVIDIA 1660ti mobile GPU with 6GB VRAM, 16GB RAM and Intel i5-9300H CPU to run all the experiments. We used pytorch to code and ADAM[6] solver to propagate error.

4.1 modified Pix2Pix

Dataset used is Anime colourization from kaggle. Each sample from the dataset has two images in which, the one on the left half is coloured version of the other half. Dimensions of each sample is 1024x512. Data augmentations is done using albumentations package which shuffles the data and does augmentations like adding rotation, transpose, contrast, brightness, hue shift, motion blur, gradient shift, etc. We chose albumentations package because it is optimized for better performance for rich variety of image transformations and computer vision tasks. Learning rate is 2e-4 , Group Size is 6 , Number of groups is 32, Number of worker is 1 , image size is 256, channels are 3, L1 Lambda is 100 and Number of epochs 150. We resized image to 256x512 and divided into two segments 256x256 and 256x256 for paired translation. We applied flip, contrast, color jitters. Dataloader batch size is set to 5. We translated 8 images as a batch.



(a)

(b)

(c)

(d)

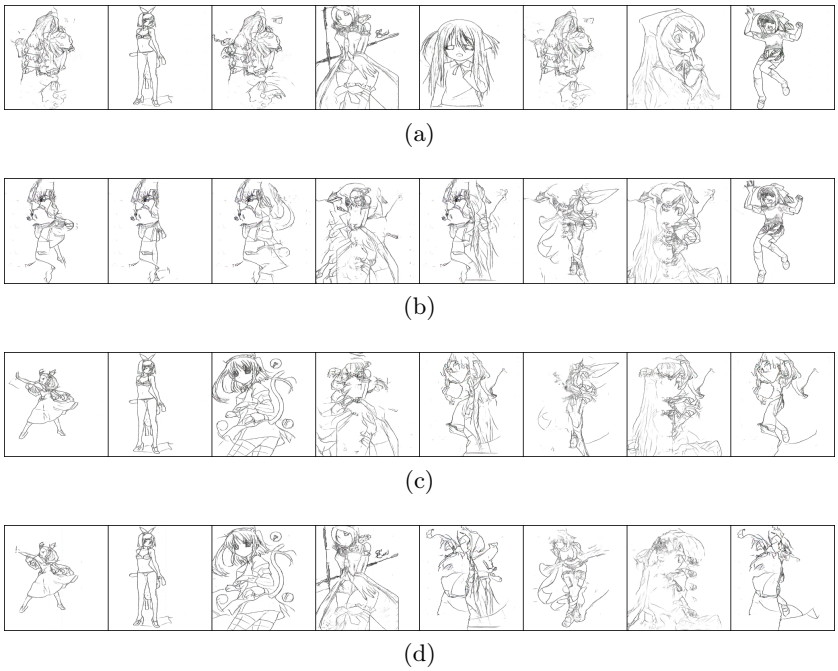
Fig. 3: Example input images at various epochs with buffer size 8. We can observe hue, rotation and contrast in each input image of different epoch. We can see difference in hair color of 7th image of (b) and 7th image of (c), orientation of the image is also different

After running both original and modified version we obtained few results. Modified version had better accuracy at earlier epochs.

Table 1: results pix2pix vs modified(ours)

	Pix2Pix		**Modified(ours)**	
Batch size	Batch size 32	Batch size 4	Group size 4	Group size 16
Iter/sec	1.17	1.4	1.15	1.175
mins/epoch	12.66	18.933	8.8	9.833
D fake	0.06	0.04	0.009	0.02
D real	0.98	0.95	0.98	0.98

D fake is probability of fake at discriminator and D real is probability of predicting image as real. These values in results table are taken after 100 epochs.



(a)

(b)

(c)

(d)

Fig. 4: Example output images during different epochs

4.2 Modified cycleGAN

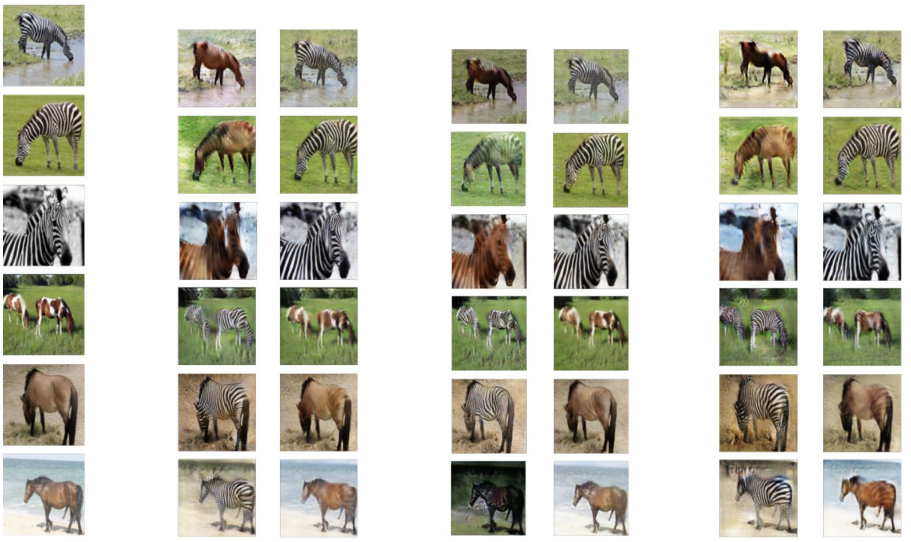
We have used horse2zebra dataset in all the experiments. Dimensions of each sample is 256x256. In all the experiments, we train using parameters from the paper[13] with constant learning rate 1e-5 for first 100 epochs and linearly decaying learning rate to 0 for remaining epochs, batch size is 1, number of epochs are 200, lambda cycle value is 10. We used pre-trained weights, one experiment

using already used cycleGAN weights and another set which were trained on AlexNet[7], we focus on the results of first experiment.

Table 2: Results Vanilla vs U-NET vs modified(ours)

	vanilla	**U-NET**	**Modified(ours)**	**with pre-trained weights**
hrs/epoch	1.75	1.78	1.72	1.29
H fake	0.324	0.401	0.295	0.241
H real	0.676	0.599	0.705	0.759

Compared results with vanilla variant, U-NET, modified(ours) and modified(our) with pre-trained weights. H fake and H real represent reconstructed loss values. We can see when pre-trained weights are used time taken to finish epoch reduced significantly while others are near to each other.



(a) input (b) vanilla (c) U-NET (d) modified(ours)

Fig. 5: comparison of outputs (a) is input image, (b) is vanilla generated image on left and reconstructed on right, (c) is U-NET generated image on left and reconstructed on right and (d) is our model generated image on left and reconstructed on right

4.3 Discussion

For modified Pix2Pix, we can see groupNorm has given better results at lower group or batch sizes. It was faster in computation and accurate. We can use different normalisation method like weight normalisation or some adaptive normalization method to see if we can achieve better results without bigger trade-off. For modified cycleGAN we can observe our method performs better than vanilla and U-NET variant with less artifacts. Although outputs aren't as good as input images but are better looking compared to other variants in realistic setting. We observed during training that even with modified discriminators' output there wasn't much improvement, always very close to 0.3, we suspected it may be due to collective training with generators. When used pre-trained weights for discriminators we got better results and lower run time.

5 Conclusion

In this paper, we identified problem with pix2pix, when batch size is small it takes time and accuracy is very bad. Improve speed and accuracy using existing normalisation method called groupNorm and compared both on anime colourization dataset. We also identified and analyzed problems in cycleGAN model due to cycle consistency. We came up with few modification to loss function to solve those problems: added L1 loss on extracted features(CNN) from corresponding discriminator, weight decay of cycle consistency loss as training goes forward, added weighted cycle consistency loss parameter from quality of generated images and used pre-trained weights for discriminators. We can see improved performance when compared to two variants U-NET and vanilla cycleGAN on horse2zebra dataset. We also pointed few interesting options to try in future work.

6 Future scope

While our modified model obtains better results and performance, there are still many interesting parts for us to explore. Fine tuning parameters is one direction where one can tweak various parameters to see changes and analyze the results. In this section, we try to show few paths in which future readers can research.

One-to-many mapping: By passing input which is stochastic in nature to the generators, making them basically one-to-many mappings. Tricky part is how to pass stochastic input while keeping cycle consistency and regularization intact into the architecture. We experimented by adding fourth channel, noise, and adjusted generators to produce image with this extra channel which has similar noise distribution used in discriminators. We failed to even produce results on par with original paper within same training parameters, this might have been due the randomness created by noise distribution.

Generators with latent variables: In image translation problem using GANs latent space is shared between two image domains. In GANs, x maps to z

and z maps to y where x and y are input and target domains respectively. cycleGAN has two generators and discriminators so one way to induce this concept is to pick a specific layer from generator which outputs latent variable z while considering both the generators as encoders/decoders which are mutual. Then, conceivably, we can impose other concepts of consistency, such as latent variable consistency and shorter cycle consistency($x \rightarrow z \rightarrow x$).

Parameter tuning: We have more parameters to tune and refine when compared to original cycleGAN. One change is refinement of λ_t and α_t values. Another fine tuning suggestion is to decide which discriminator to be used to extract features. During experiments, we observed that the quality of results is very sensitive to different kinds of parameters. We encourage is try different possibilities and variations of fine tuning different parameters. We used already trained weights of cycleGAN but we can used pre-trained weights from AlexNet or ResNet for discriminators for even better results.

Single discriminator: Since the two discriminators do classification on different image classes and domains, we can try to replace them with one discriminator network that classifies among fake images,three classes and two image domains. This way, the discriminator can see the data from both domains. Because the generators are generally always near-identity mapping at the beginning of training, such a discriminator may better drive the generators in the appropriate direction.

For pix2pix we can use different normalization methods to boost run time like weight normalization[10], layer normalization[1], spectral normalization[8] or different normalization layers at different layers.

References

1. Ba, J. L., J. R. Kiros, and G. E. Hinton (2016). Layer normalization. arXiv preprint arXiv:1607.06450.
2. Ioffe, S. and C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning. PMLR, 2015
3. Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio (2014a). Generative adversarial nets. Advances in neural information processing systems, 27.
4. Isola, P., J.-Y. Zhu, T. Zhou, and A. A. Efros, Image-to-image translation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
5. Kingma, D. and M. Welling (2014). Auto-encoding variational bayes. in international conference on learning representations (iclr).
6. Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
7. Krizhevsky, A., I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (eds.), Advances in Neural Information Processing Systems 25. Curran Associates, Inc., 2012, 1097– 1105.

8. Miyato, T., T. Kataoka, M. Koyama, and Y. Yoshida (2018). Spectral normalization for generative adversarial networks. arXiv preprint arXiv:1802.05957.
9. Montúfar, G. (2018). Restricted boltzmann machines: Introduction and review. CoRR, abs/1806.07066. URL http://arxiv.org/abs/1806.07066.
10. Salimans, T. and D. P. Kingma (2016). Weight normalization: A simple reparameterization to accelerate training of deep neural networks. Advances in neural information processing systems,29.
11. Ulyanov, D., A. Vedaldi, and V. Lempitsky (2016). Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022.
12. Wu, Y. and K. He, Group normalization. In Proceedings of the European conference on computer vision (ECCV). 2018.
13. Zhu, J., T. Park, P. Isola, and A. A. Efros (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. CoRR, abs/1703.10593. URL http://arxiv.org/ abs/1703.10593.
14. Larsen, A. B. L., S. K. Sønderby, and O. Winther (2015). Autoencoding beyond pixels using a learned similarity metric. CoRR, abs/1512.09300. URL http://arxiv.org/abs/1512. 09300.
15. Kim, T., M. Cha, H. Kim, J. K. Lee, and J. Kim (2017). Learning to discover cross-domain relations with generative adversarial networks. CoRR, abs/1703.05192. URL http: //arxiv.org/abs/1703.05192.
16. Taigman, Y., A. Polyak, and L. Wolf (2016). Unsupervised cross-domain image generation. CoRR, abs/1611.02200. URL http://arxiv.org/abs/1611.02200.