How would Picasso paint that scene?

Tarun Yenamandra, Aymen Mir, Mossad Helali

Saarland University

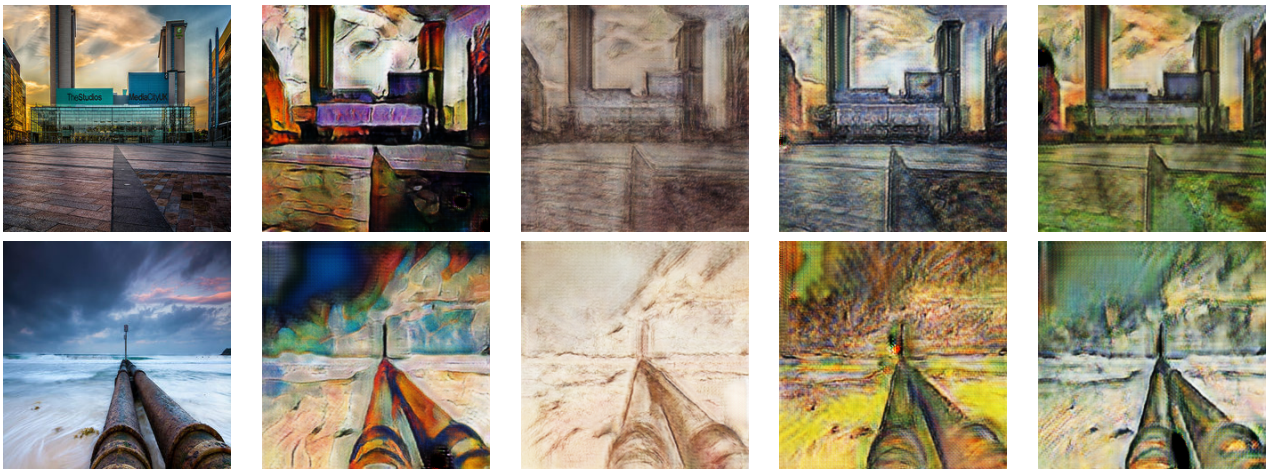

Figure: Painter style transfer using modified CGAN with modified Wasserstein loss. Left to right: Landscape image, Picasso style, Da Vinci style, Van Gogh style, Cezanne style

Abstract

We present an end to end framework for transfering the style of paintings done by medieval and modern artists to any captured image. This task can be seen as subset of the famous problem of image to image translation in computer Vision. Most image-to-image translation systems require paired data of images. Such paired data is not available for our task. We primarily base our architecture on the Cycle GAN architecture [2]. We improve the performance of the existing system by incorporating Wasserstein loss into the architecture. We trained a separate "neutral" classifier by fine-tuning a 18 layer residual network to get quantitative results for the experiment of both architecture and we found that the performance of our architecture is better than the performance of the existing cycle GAN architecture.

1. Introduction

Imagine staring at a picturesque scene while you are out for a morning walk. A question that often pops up into the mind - How would Picasso paint that scene?

Can we somehow get to know how would that scene look had it been painted by one of the famous medeiaval or modern painters.

We know how medieval and modern artists paint. All their paintings are available in museums. But we do not have access to the original scene which inspired those paintings.

Had we had access to paired image data - image of an original scene and the artists' rendering of that scene - a naive approach to solving this problem could be that we could train a conditional CNN which takes in an image and generates a new image. We would use a L2 normed difference between the generated image and the original image (ground truth) as the loss function. Such an approach leads to bad results [6] because the L2 norm is minimized by averaging over all possible outputs which causes blurring. Another possible reason for the bad performance is that it is very difficult for a CNN to generate such high dimensional data.

A possible solution is incorporate variational analysis

into the network as is suggested in [4] but this approach has not produced great results.

Another possible solution to the blurring problem is to use an adaptive loss function which forces the CNN to map the input image to "real looking" images, so blurred images will be rejected as they are not real looking images.

This adaptive loss function is exactly what the generative adversarial network can be thought to represent. 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. Conditional GANs [5] are perfect for the task at hand as they enable one to condition on a given image and generate images with the required style. [3]

A problem with this approach is that it requires paired image-image data and we do not have access to such data for most medieval and modern artists. We just have access to the collections of paintings by these artists. Any human imitator will try to infer a common thread - the style - that runs through all paintings of a particular artist and using this try to generate new paintings of a particular scene. Is there any approach that can somehow imitate this procedure?
The answer to this questions is yes: The cycle GAN architecture proposed in [8] tries to do something exactly similiar. The main idea behind cycle GANs is very simple. The architecture uses two generators - one to convert images from set A to set B and the other to convert images from set B to set A and tries to ensure that the an image from set A when converted to set B remains as similiar as possible to the original image when converted back by the other generator.

2. Related Work

2.1. Generative Adversarial Networks

Consider a training set $X \subset R^d$ where R^d represents the space of features we are interested in. Suppose that the data is distributed as p_d. We aim to train two neural networks G and D. G is sampler from a particular distribution p_g. D on the other hand is a discriminator which is tasked with trying to discriminate whether any image is sampled from p_g or p_d . The network G takes in an input $z \sim p_z$ where p_z is some random distribution and tries to generate an image which fools D.
In the original formulation of GANs proposed in [2] D is trained to maximize the probability of assigning the correct label to both training examples and samples from G. And simultaneously G is trained to minimize $log(1 - D(G(z)))$. In other words, D and G play the following two-player minimax game with value function $V(D, G)$

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_d}[logD(x)] + \mathbb{E}_{z\sim p_z}[log(1 - D(G(z)))]$$

In practice, we must implement the game using an iterative, numerical approach. Optimizing D to completion in the inner loop of training is computationally prohibitive, and on finite datasets would result in overfitting. Instead, we alternate between k steps of optimizing D and one step of optimizing G

2.2. Cycle Consistency

The idea of using transitivity as a way to regularize structured data has a long history. In the language domain, verifying and improving translations via back translation and reconciliation is a technique used by human translators as well as by machines. The general principle is best understood in terms of translation. If you are translating text into a new language, when you translate it back into the original language, the translated text should not be different from the original text otherwise there are flaws with your translation.

2.3. Wasserstein GANs

The loss function of a regular GAN can be represented as

$$L(G, D) = \int_x \Big(p_r(x) \log(D(x)) + p_g(x) \log(1 - D(x)) \Big) dx$$

The ideal discriminator can be shown to be

$$D^*(x) = \frac{p_r(x)}{p_r(x) + p_g(x)} \in [0, 1]$$

When both G and D are at their optimal values, we have $p_g = p_d$ and $D^*(x) = 1/2$ and the loss function reduces to

$$L(G, D^*) = -2 \log 2$$

It can be shown that

$$L(G, D^*) = 2D_{JS}(p_r \| p_g) - 2 \log 2$$

Essentially the loss function of GAN quantifies the similarity between the generative data distribution pg and the real sample distribution pr by JS divergence when the discriminator is optimal. The best G^* that replicates the real data distribution leads to the minimum $L(G^*, D^*) = 2log2$

The idea behind Wasserstein GAN proposed in [1] , is to replace the JS divergence with a Wasserstein distance. The loss function reduces to

$$W(p_c, p_g) = \inf_{\gamma \sim \Pi(p_d, p_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

It is intractable to exhaust all possible joint distributions in $\Pi(p_d, p_g)$ to exhaust $\inf_{\gamma\sim\Pi(p_d,p_g)}$
The formula is transformed using the Kantorovich-Rubinstein duality to

$$W(p_r, p_g) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x\sim p_r}[f(x)] - \mathbb{E}_{x\sim p_g}[f(x)]$$

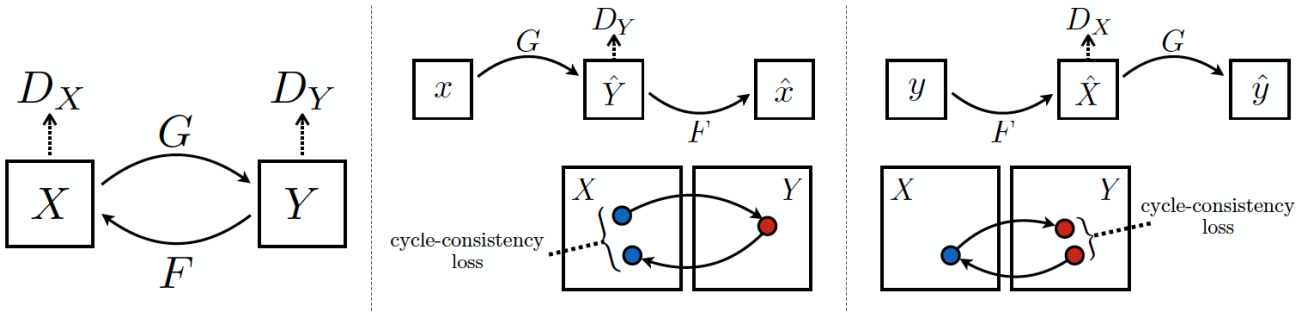

Figure 1. Visualization of Cycle consistency loss

To ensure that discriminator remains Lipschitz continuous, the weights of the discriminator are clamped between a fixed range.

3. Formulation

We have two separate data sets A and B and we want to infer the style of all images in set A (the set containing the paintings) and change the style of all images in set B using the style we have inferred. Suppose that the data in set A is distributed as p_A and the data in set B is distributed as p_B. We can formulate the problem using two conditional GANs. The generator G_A takes an image in set B (the plain images) and tries to generates images in set A by fooling the discriminator D_A. Something similiar can be said about generator G_B and the discriminator D_A. Our final objective function consists of two terms - one which compute the Wasserstein distance and the other which maintains the cycle consistency loss.

3.1. Wasserstein Loss

We use the Wassserstein loss, already described above, with both generators and discriminators G_A, G_B, D_A, D_B. For G_A and D_B We express the objective as

$$L_{gan}(G_B, D_B) = \mathbb{E}_{x \sim p_B}[D_B(x)] - \mathbb{E}_{x \sim p_A}[D_B(G_B(x))]$$

G_A tries to generate images that look similar to images from domain B , while D_B aims to distinguish between translated samples and real samples. G aims to minimize this objective against an adversary D that tries to maximize it. We introduce a similar adversarial loss for the other mapping function and its discriminator as well.

The above formulation does not contain the latent variable z that is usually used in GAN architectures. Out reasoning is that even if we used a latent variable, the output of the network would not have produced any substantial variance. This is because of the cycle consistency constraint. The cycle consistency constraint ensures that the distributions learnt are like delta distributions.

3.2. Cycle Consistency Loss

Adversarial training can, in theory, learn mappings G_A and G_B that produce outputs identically distributed as target domains A and B respectively, with large enough capacity, a network can map the same set of input images to any random permutation of images in the target domain, where any of the learned mappings can induce an output distribution that matches the target distribution. Thus, adversarial losses alone cannot guarantee that the learned function can map an individual input x to a desired output y . To further reduce the space of possible mapping a cycle consistency is maintained. Images when mapped from domain A to domain B, when mapped back to domain A must be similiar to the original image. And we also want to maintain this cycle consistency about the mappings from domain B to domain A.

This can be ensured by using a cycle consistency loss in the objective function

$$L_cyc(G_A, G_B) = \mathbb{E}_{x \sim p_A} \parallel G_A(G_B(x)) - x \parallel + \mathbb{E}_{x \sim p_B} \parallel G_B(G_A(x)) - x \parallel$$

3.3. Identity Loss

In [8], the authors introduce additional loss to encourage the mapping to preserve color composition between the input and output. We regularize the generator to be near an identity mapping when real samples of the target domain are provided as the input to the generator.

$$L_{id}(G_A, G_B) = \mathbb{E}_{x \sim p_A} \parallel G_A(x) - x \parallel + \mathbb{E}_{x \sim p_B} \parallel G_B - x \parallel$$

Without L_{id} the generators G_A and G_B are free to change the tint of input images when there is no need to. For example, when learning the mapping between an artists paintings and potographs, the generator can map paintings of rainy days to photographs taken during winter, because such a mapping may be equally valid under the adversarial

loss and cycle consistency loss. The additional constraint restricts the network to not introduce unnecessary changes.

3.4. Full Objective

Our full objective is:

$$L = L_{gan}(G_A, D_B) + L_{gan}(G_B, D_A) + \lambda L_{cyc} + \mu L_{id}$$

4. Dataset

We collected a completely new dataset for this project. To transfer the style of a painting to a landscape image, two different datasets are needed. One the landscape images themselves, and two, images of paintings from different artists. A set of 1049 landscape images were compiled from the Flickr dataset. Images of paintings have been compiled from a Kaggle challenge. Four different artists paintings were used.

Class	Number of Images
Landscape	1049
Da Vinci	181
Picasso	300
Cezanne	454
Van Gogh	449

About 50 images from each of the artist have been used as test images, as usual. These test images were not used during the training.

5. Experiments

The primary goal of this project is to transfer the painting style of an artist to a modern landscape image. To achieve this end, two different Generative Adversarial Networks the vanilla CGAN and the modified CGAN with Wasserstein distance for the discriminator loss function have been used.

5.1. Improvements to Vanilla CGAN

Three changes were made to the vanilla CGAN to use the Wasserstein distance as the loss function instead of the binary cross entropy loss at the output of the discriminator. The last layer of the vanilla CGAN, the sigmoid activation layer, has been removed. To enforce the Lipschitz continuity constraint on the discriminator, weight clipping was used. Two different limits for the weights have been tested, [-0.01,0.01] and [-0.003,0.003]. The former achieves a better result. The identity and the cycle losses of the vanilla CGAN are L1 difference between the output images.

5.2. Training

To reduce model oscillation , we follow Shrivastava et als strategy [7] and update the discriminators using a history of generated images rather than the ones produced by the latest generative networks. We keep an image buffer that stores the 50 previously generated images. For all the experiments, we set $\lambda = 10$ and $\mu = 1$. We use the RMPS prop algorithm with batch size of 6. All networks were trained from scratch with a learning rate of 0.0002. We use 200 epochs for training. We keep the same learning rate for the first 100 epochs and linearly decay the rate to zero over the next 100 epochs

Transfer of each artist style to the landscape images has been achieved by training the GANs independently for each artist. An NVIDIA Tesla P100 GPU has been used for all training purposes. Training the network to transfer the style of one artist took approximately 20 hours.

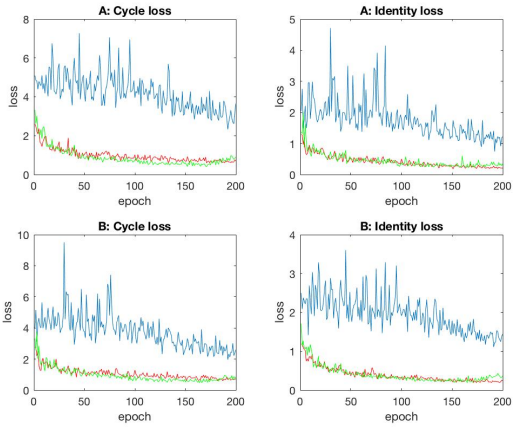

Figure 2. Quantitative comparision of Picasso style transfer to landscape images: (red) vanilla CGAN, and (blue) modified CGAN with Wasserstein loss with weight clamping to [-0.01,0.01]

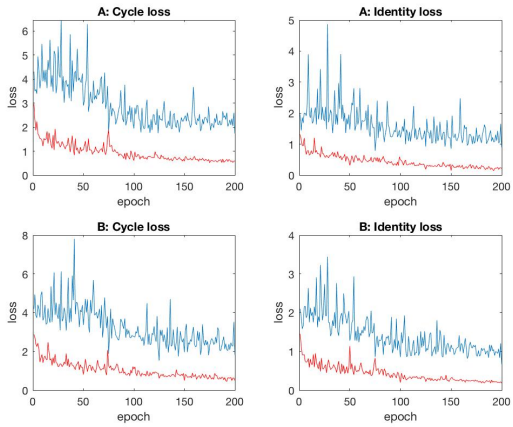

Figure 3. Quantitative comparision of Cezanne style transfer to landscape images: (red) vanilla CGAN, and (blue) modified CGAN with Wasserstein loss with weight clamping to [-0.01,0.01]

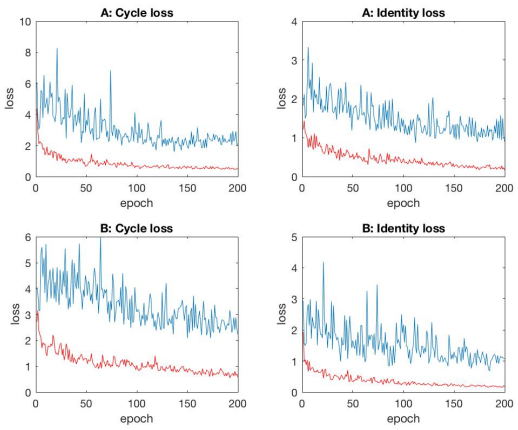

Figure 4. Quantitative comparision of Da Vinci style transfer to landscape images: (red) vanilla CGAN, and (blue) modified CGAN with Wasserstein loss with weight clamping to [-0.01,0.01]

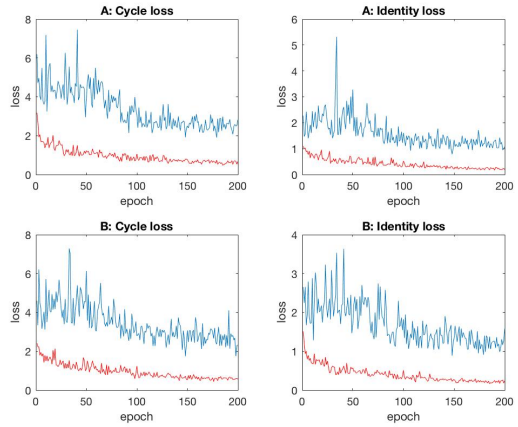

Figure 5. Quantitative comparision of Van Gogh style transfer to landscape images: (red) vanilla CGAN, and (blue) modified CGAN with Wasserstein loss with weight clamping to [-0.01,0.01]

5.3. Evaluation Metric

The output images of the modified CGAN with Wasserstein loss have been qualitatively appealing. Some selected samples are attached in the appendix. But we do not have a natural quantitative metric to quantify the performance of each network.

One possible option is to conduct user studies - asking groups of people to judge whether the images generated by the architecture is an actual painting or merely a fake one - and then to tabulate the results.

Another option is to use a new CNN classsifier to classify whether an image is actual painting or merely a fake generated one.

We chose the latter option primarily because of the time required in the former option. For this purpose, we used a new network. A pre-trained ResNet18 was fine-tuned using the paintings of the artists. The network was fine-tuned for each of the artist independently.

A natural question is why did we not use the two classifiers already trained during the training process to get a quantitative metric. Had we used one of the existing discriminators, both of the generators would have been acutely aware of the idiosyncrasies of atleast one of the discriminators as it is being trained to fool this network. In that sense the new network is "neutral"

The training error on each of the artist was about 0.01%. The fine-tuned network was then used to classify the outputs of vanilla CGAN and modified CGAN with Wasserstein loss. The following table shows the quantitative results. The numbers listed in the table represent the percentage of times the classifier was fooled into accepting the generated images as real images. The higher the score, the better is the performance of the generator.

Error	Vanilla CGAN	Wasserstein Loss
Picasso	87%	96%
Cezanne	14%	87%
Van Gogh	85%	72%
da Vinci	17%	24%

All the fake images generated by both networks - wgan and cgan - are artistic renderings of the same base image in the landscape set. So the only difference between the two images one generated by wgan and other generated by cgan is that of style. So the classifer is fooled only on the basis of style and not on the basis of some other attribute.

6. Discussion and Conclusion

The visual results for Wasserstein GAN are much better than the results for a normal cycle GAN. Artistic strokes are clearly visible in the images produced by WGan and no such features are visible in the images produced by the CGan. The quantitative results corroborate our visual findings. In three out of four cases the classification score for WGan are better than those of CGan.

The plots for cycle and identity losses tell a different story. The cycle and identity losses are lower of cycle GAN. We are not quite sure how to explain this. One possible reason could be that to generate realistic images, it is not required to lower the cycle and identity losses to absolute zero. This also suggests that a different value for the parameters λ and μ may have produced better results but we did not have sufficient time to try out such experiments. This may be an area worthy of further inverstigation

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *CoRR*, abs/1701.07875, 2017.

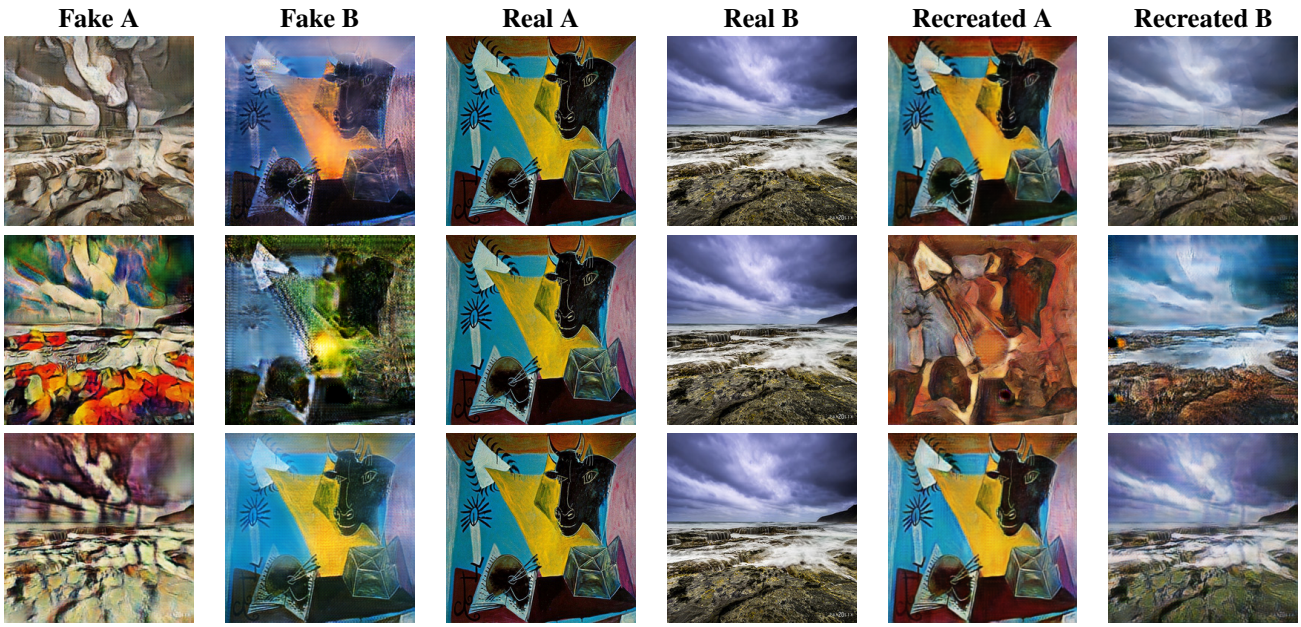

Figure 6. Weight clamping comparison on Picasso dataset. Top: Test images for a weight clamping [-0.01,0.01]. Middle: Test images for a weight clamping [-0.003,0.003] following the same order as above. Bottom: Test images for vanilla Cycle GAN. Left to right in each row: Fake Picasso painting, fake landscape image based on painting, real Picasso painting, recreated painting after cycle, recreated landscape after the cycle.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[3] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[4] E. Kobler, T. Klatzer, K. Hammernik, and T. Pock. Variational networks: connecting variational methods and deep learning. In *German Conference on Pattern Recognition*, pages 281–293. Springer, 2017.

[5] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[6] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

[7] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, volume 2, page 5, 2017.

[8] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

7. Appendix

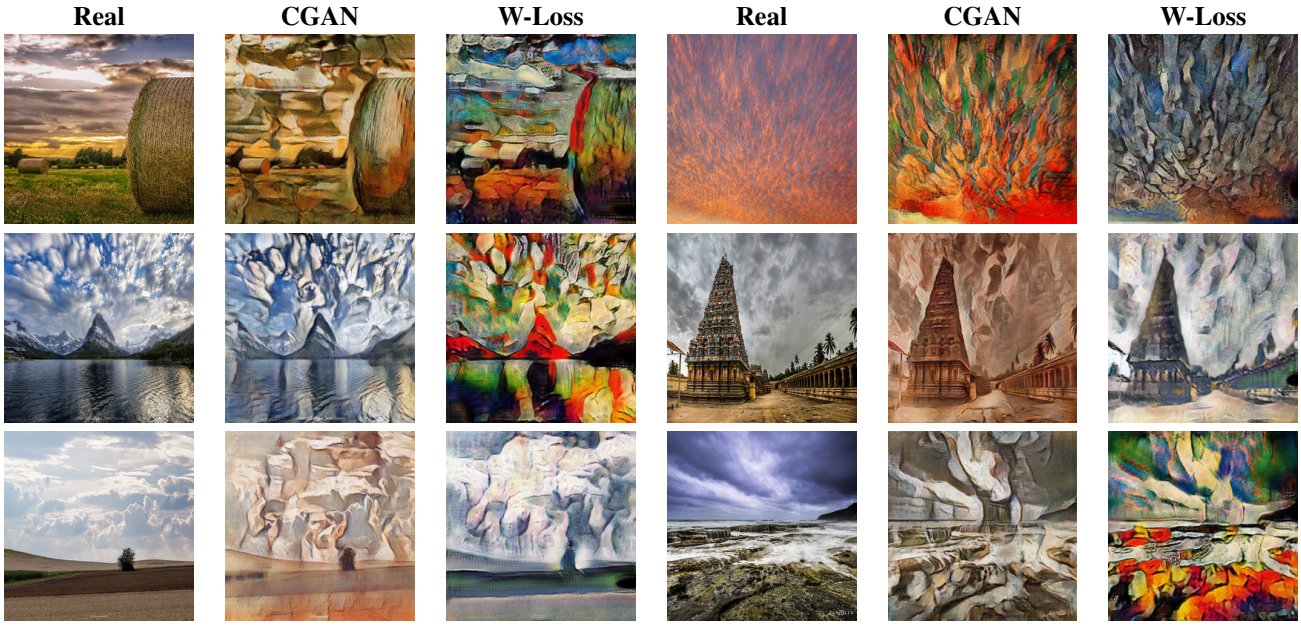

Figure 7. Results for Picasso painting style transfer to landscape images

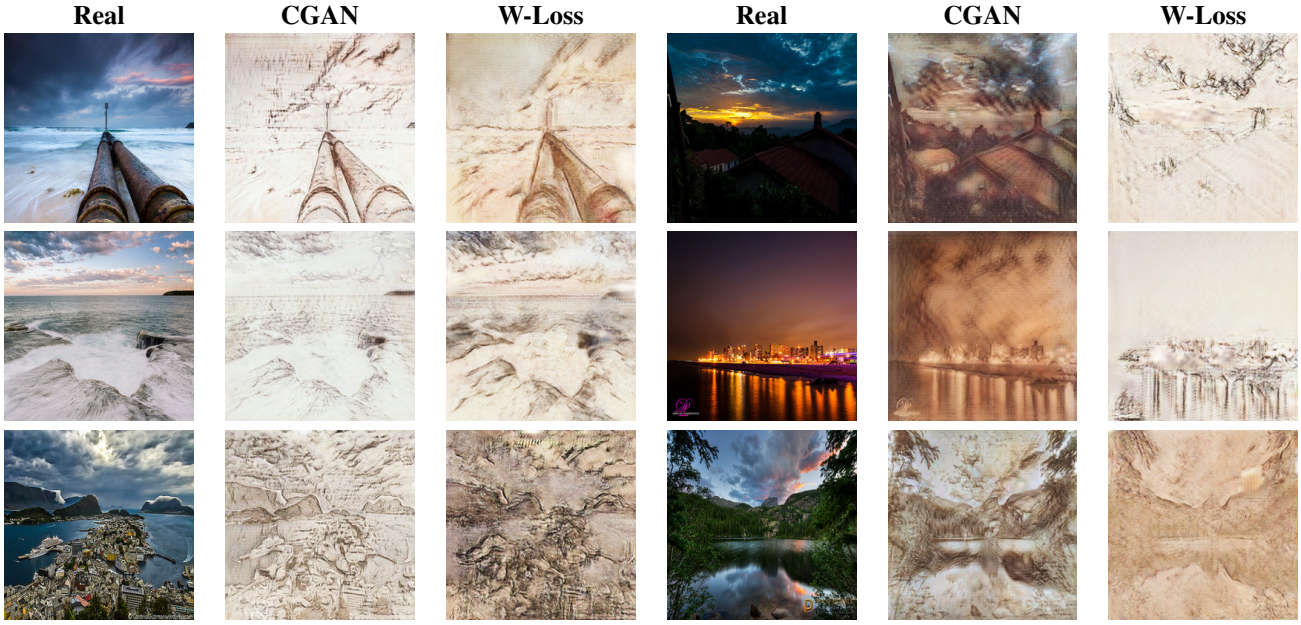

Figure 8. Results for Da Vinci painting style transfer to landscape images

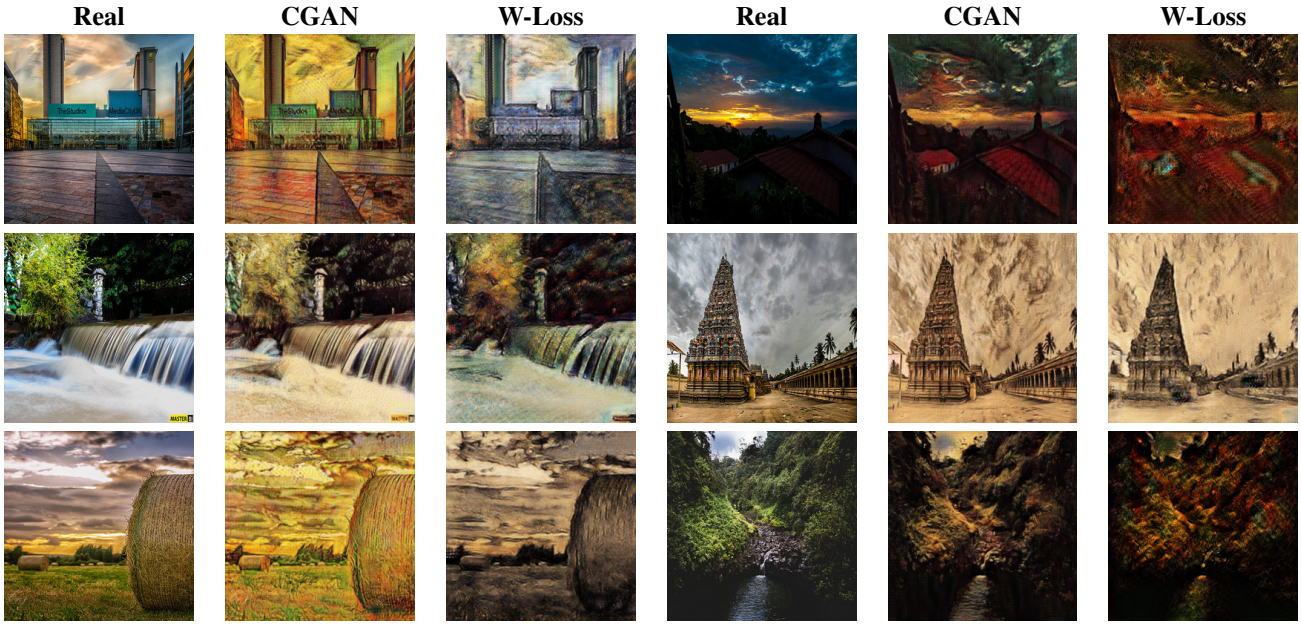

Figure 9. Results for Van Gogh painting style transfer to landscape images

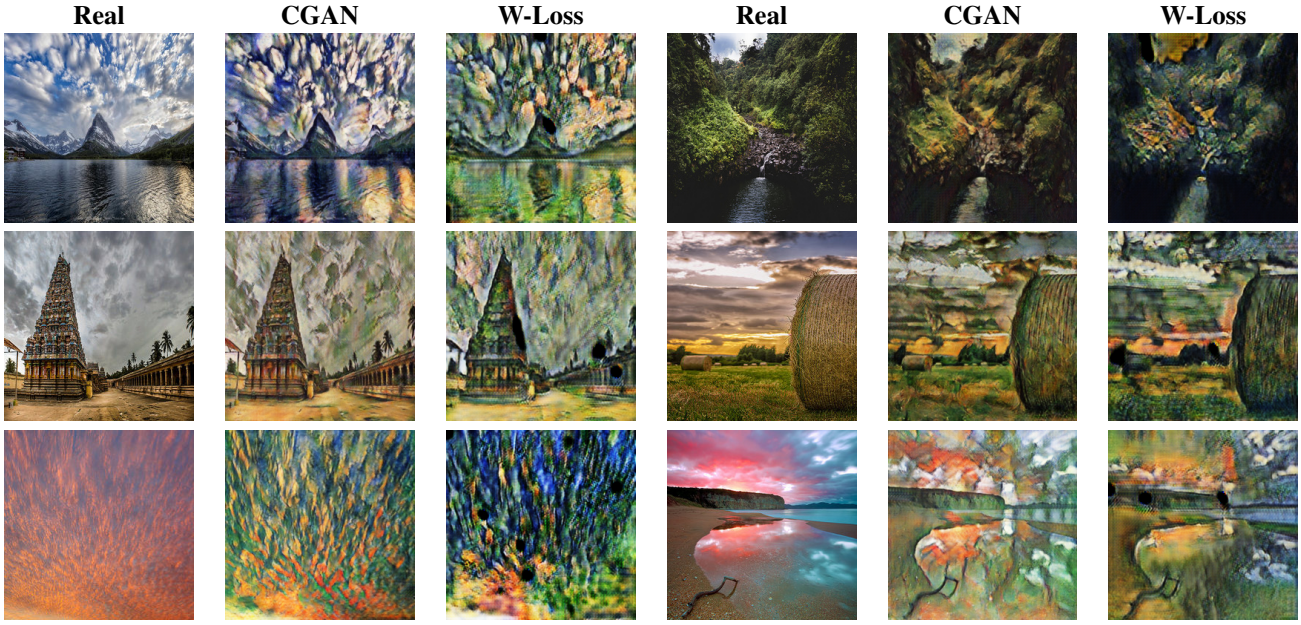

Figure 10. Results for Cezanne painting style transfer to landscape images