Real or Fake?
Style and Morphological Transformations Using a CycleGAN

Johan Brandby Tomas Lundberg
Group 42

Abstract—A CycleGAN, first proposed by Zhu et al. [1], can be used to transform images from one domain to another without the need of paired datasets. Zhu et al. [1] achieved impressive results where the domains where morphologically similar, such as between horses and zebras, but not when they were not, such as between dogs and cats. In this report, a modified CycleGAN is implemented with the goal of producing satisfactory results for the transformation between human and anime faces, two domains that are similar but where morphological differences exist. Most notably, a U-Net architecture was used for the generators. Unfortunately, in the time given, no successful translation between human and anime faces was achieved. In fact, only an identity mapping was learned modulo some slight color change. The leading explanation is that perhaps the U-Net is poorly suited for the problem.

1 Introduction

With the advent of generative adversarial networks (GANs), presented by Goodfellow et al. [2], it became possible to mimic a target distribution and generate samples that are perceptually identical—an example of which is on images. The GAN architecture optimizes two separate neural networks. The first is called a discriminator and attempts to classify if a given image is real or fake; while the second, usually referred to as the generator, decodes a latent, random vector into the target domain and tries to learn how to fool the discriminator. That is, the two models are pitted against each other in a zero-sum game, where their optimization is either done jointly or alternatingly. Customarily, the goal of a GAN is to have the generator achieve a satisfactory level of realism from its generated examples.

A slew of extensions has been proposed; ranging from mitigating known issues such as mode-collapses [3], [4]—i.e. where the generator converged to a local minima that only predicts the same subset of outputs—to attempts for conditioning its networks for greater control [5]. One notable version is the cycle-consistent GAN (CycleGAN), presented by Zhu et al. [1], which allows for style and attribute translation between two image domains. In fact, it has the additional property of supporting unpaired image-to-image translations—that is, map examples from one domain to that of an other without any explicitly curated datasets. From the original paper of Zhu et al. [1], they reported that the CycleGAN framework achieved stunning results when it came to style-transfer, but failed when it comes to morphological changes—in their case it related to mapping images of dogs to cats and vice versa.

The goal of this report is to re-implement a modified version of the CycleGAN and investigate if the structural changes can be achieved. The first change is to modify the generator network, which is an auto-encoder in the original paper. The difference is to replace it with a `U-Net`, first proposed by Ronneberger et al. [6] and originally intended for medical image segmentation. It is chosen with the reason that its skipping connections between the down- and up-sampling modules could potentially allow for more localized information to pass through the network—see Fig. 1 for a visualization. The second change has to do with the learning algorithm, where the original paper utilizes a buffer to store the last 50 images outputted by the two generators and use an optimizer based on stochastic gradient descent, for more stable learning—this strategy's inception is attributed to Shrivastava et al. [7]. Instead of the buffer, the batch size is increased to 32 and omits the entire stabilizing concept, for the interest of time.

Moving on, this report takes inspiration from Zhu et al. [1] and, to facilitate application from the course's topics, tries to implement the modified version from scratch, using `PyTorch`. For transparency's sake, it is important to note that the aforementioned publication does provide their own code base [8], in the chosen `Python` package. However, notice that these has not been used in any way, shape or form during the iterative development process.

To the end of training the new configuration, the two domains chosen are those of realistic human, H, and drawn anime, A, faces—Japanese cartoon caricatures of the human body. Specifically, the former is approximated by the `Flickr-Faces-HQ` (FFHQ) from Nvidia labs [9] and the latter by `Danbooru2019 Portraits` from Gwern Branwen [10]. These represent the same semantic concept: the front of the human head; albeit with different proportions and style. Thus, the chosen datasets fulfill the goal of having two domains with distinct morphological differences.

1.1 Background Theory

As mentioned before, the networks in a standard GAN is comprised of a discriminator D and a generator G [2]. The goal of the discriminator D is to predict whether a given example is real or fake through a scalar signal; that is, if it belongs to either the target domain $\{\mathbf{x}\}$ or is a generated example of $\{G(\mathbf{z})\}$. In this formulation, $\mathbf{z} \sim p_{\text{latent}}(\mathbf{z})$ is a random variable, whose samples act as the latent vectors that the generator G decodes. Borrowing the notation of Goodfellow et al. and with slight modifications, the loss function can be formulated as

$$\mathscr{L}_{\text{GAN}}(D,G) = \mathbb{E}_{\mathbf{x}}\big[\log D(\mathbf{x})\big] + \mathbb{E}_{\mathbf{z}}\big[\log\big(1 - D(G(\mathbf{z}))\big)\big], \qquad (1)$$

- *Johan Brandby:* johbran@student.chalmers.se
- *Tomas Lundberg:* lutomas@student.chalmers.se

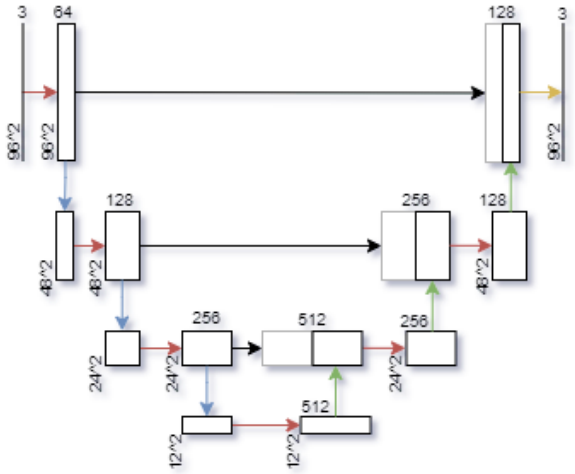

Fig. 1: `U-Net` architecture. Each rectangle represent the current dimensions, where the horizontal text is the image size and the text above the number of channels. In the upper left is the input image which is 96x96 with 3 color channels. In the upper right is the output image with the same size. The red rightward pointing arrows represent three 3x3 convolutional filters with 1x1 padding and relu activations (`conv-relu-conv-relu-conv`). The downfaced green arrows are 2x2 maxpool downsampling layers. The green, upwards pointing arrows, are a 2x2 transpose convolutional layers with 2x2 stride. The black, rightward pointing arrows are identity mappings. Finally the yellow arrow is four 3x3 convolutional filters with 1x1 padding and relu activations.

where $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ denotes the data distribution. Making the optimization objective take the form

$$\min_G \max_D \mathscr{L}_{\text{GAN}}(D,G). \tag{2}$$

Moving on, CycleGAN can be interpreted as having two separate GAN systems operating jointly, with extra criterion and reformulations on their parameters and network designs, respectively. For instance, the generators no longer acts like a pure decoder, but as mapper that transforms a sample from one domain A to an other B and vise versa [1]—usually defined as a variant of the auto-encoder, to ensure that the generators learn to prioritize when pushing information through the architecture's low-dimensional, latent space. Furthermore, the original version also imposes a cycle-consistency constraint on its generators; namely, if G_{AB} maps from A to B and G_{BA} follows by analogy, then

$$\begin{aligned}\mathscr{L}_{\text{Cycle}}(G_{AB},G_{BA}) = \\ \mathbb{E}_{\mathbf{a}}\big[|G_{BA}(G_{AB}(\mathbf{a}))-\mathbf{a}|\big]+\mathbb{E}_{\mathbf{b}}\big[|G_{AB}(G_{BA}(\mathbf{b}))-\mathbf{b}|\big].\end{aligned} \tag{3}$$

That is, that the cycled image should be as close to the original as possible. This results in the complete objective

$$\begin{aligned}\min_{G_{AB},G_{BA}} \max_{D_A,D_B} \mathscr{L}_{\text{GAN}}(D_A,G_{BA})+\mathscr{L}_{\text{GAN}}(D_B,G_{AB}) \\ +\lambda\mathscr{L}_{\text{Cycle}}(G_{AB},G_{BA}),\end{aligned} \tag{4}$$

assuming that the corresponding discriminators are denoted D_A and D_B. Furthermore, the parameter $\lambda > 0$ is a scalar that determines the balance between the losses.

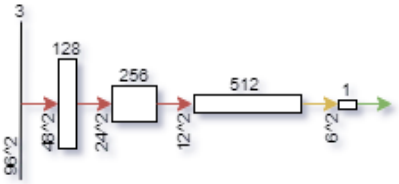

Fig. 2: `Patch-GAN` architecture. Each rectangle represent the current dimensions, where the horizontal text is the image size and the text above the number of channels. Red arrows correspond to 4x4 convolutional layer with stride 2, padding and relu activation. The yellow arrow is the same convolutional layer but instead followed by a sigmoid activation. Finally, the green arrow is an adaptive pooling layer with output size of 1.

2 Methods

A `U-Net` architecture was used for the generators and a `Patch-GAN` architecture for the discriminators. In this section, these architectures are explained in a bit more detail. Thereafter, the adapted loss function from Eq. (4) is constructed and finally a description of the training process is presented.

2.1 Generators

The name `U-Net` comes from the appearance of the architecture, see Fig. 1. The input image is first down-sampled through a sequence of pooling layers. When the bottom level of the `U` is reached, the image is instead up-sampled with transpose convolutions until the same size as the input image is reached. In between the up/down-sampling, three 3x3 convolutional layers with padding and Rectified Linear Units (ReLU) as activations are applied. Moreover, there are identity mappings from each level of the down-sampling that is concatenated with the input to the corresponding up-sampling level. The final layer uses an additional convolutional layer. See Fig. 1 for details.

2.2 Discriminators

The `Patch-GAN` variant can be seen in Fig. 2. It is a network with four convolutional layers with 4x4 filters, padding and 2x2 stride, thus the image is downsampled by a factor of 2 at every convolution. The convolutional layers are followed by a final adaptive pooling layer, producing a scalar output. The first three convolutional layers use relu activation functions while the last instead use a sigmoid activation. Note also that the last layer has only one output channel. Thus, each of the 6x6 numbers after the final convolutional layer corresponds to probabilities about if its 70x70[1] patch is fake or not. The final layer weighs these 6x6 numbers into an output that corresponds to the probability that the input image belongs to the target domain.

The `Patch-Gan` was chosen mainly for its simplicity and the fact that it has few parameters relative to other networks that would do the same task. Therefore, the training process could be made a lot faster.

2.3 Loss Function

For the implementation of the loss function, Eq. (4) is divided into three distinct losses that are used to train the networks in different ways. First, there is the cycle loss, $\hat{\mathscr{L}}_{\text{Cyc}}$, which is the same as Eq. (3), that is the absolute error of the cycled images minus

1. The receptive field of four 4x4 convolutions with stride 2

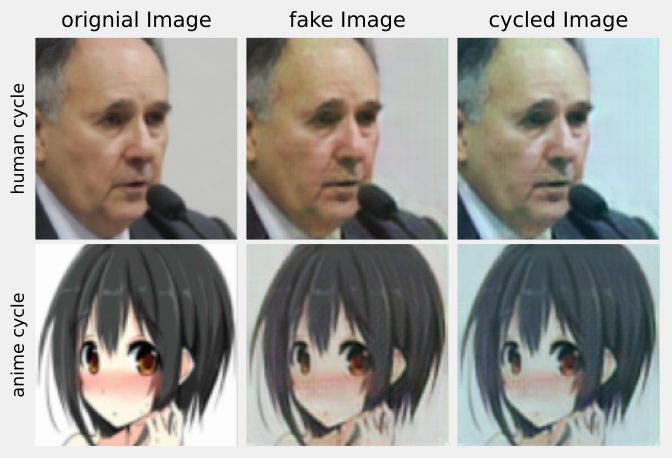

Fig. 3: Arbitrary examples of original, fake and cycled images from the test set. The cycled images are indeed similar to the originals but there seems to be no style or morphological changes to the fake images.

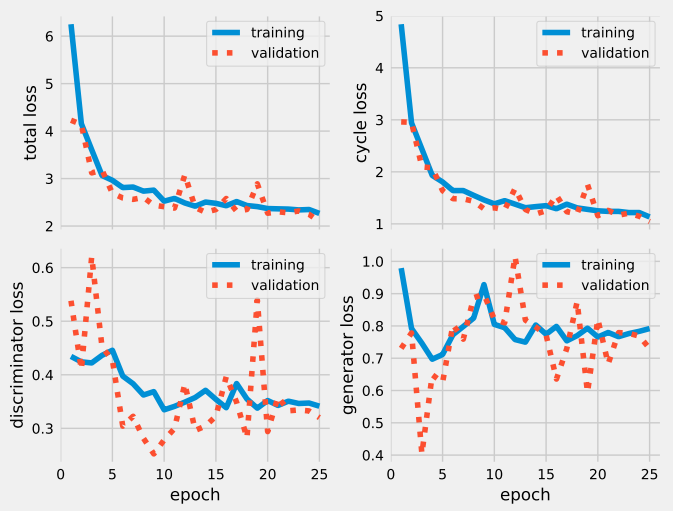

Fig. 4: The total (*upper left*), cycle (*upper right*), discriminator (*bottom left*) and generator (*bottom right*) losses per epoch for the train and validation set. See Section 2.3 for explanation.

the original. $\hat{\mathscr{L}}_{\text{Cyc}}$ is used to optimize the generators for cycle consistency. This loss is weighted by λ. Second, the generator loss, $\hat{\mathscr{L}}_{\text{Gen}}$, is the mean squared error (MSE) of the prediction that the discriminator makes on the fake image against the target of 1. That is, its goal is that the fakes produced by the generator should be classified to belong to the target domain. This loss i weighted by μ. Note that other implementations lets this error be defined by the binary cross-entropy loss, but with the work of Zhu et al. as inspiration, the MSE is employed.

Third, the discriminator loss, $\hat{\mathscr{L}}_{\text{Dis}}$, is also a MSE loss on the predictions of the discriminator. In fact, it is the sum of the MSEs on the predictions of the real images, with target 1, and the fake images, with target 0. $\hat{\mathscr{L}}_{\text{Dis}}$ is used to train the discriminator to be a good discriminator.

In summary, the objective is to find the generator G_{HA} and D_H such that

$$\arg\min_{G_{HA},D_A} \hat{\mathscr{L}}_{\text{Dis}}(D_A) + \mu\hat{\mathscr{L}}_{\text{Gen}}(D_A) + \lambda\hat{\mathscr{L}}_{\text{Cyc}}(G_{HA}), \qquad (5)$$

and analogously for G_{AH} and D_A. Following Zhu et al. [1], $\lambda = 10$ and $\mu = 2$ are used.

As an evaluation metric, the average accuracies for the discriminator networks are tracked. If the generator produces satisfactory fakes, this number should be close to 0.5.

2.4 Training Setup

The dataset consisted of approximately 60,000 images from the domains H and A, respectively. These were resized to 96x96 and normalized with the training set's mean and standard deviation. In addition, to improve robustness of the network, data augmentation was utilised: rotations (max $\pm 15°$), horizontal flips, resize (max $\pm 10\%$) and crop (max -25% of width). A train, validation, test split of 90%, 5% and 5%, respectively, was employed . To clarify, a batch of size n contained n images from H and n images from A. The batch size used was 32.

Furthermore, four different ADAM optimizers, one for each network, with standard `pytorch` settings and learning rate 10^{-5}, was implemented.

For each batch, first the $\hat{\mathscr{L}}_{\text{Dis}}$ was calculated for D_H and D_A and one optimizer step for these was taken. Then, the gradients for the discriminators was reset upon which $\hat{\mathscr{L}}_{\text{Gen}}$ for D_H and D_A was computed. Note that the discriminators was not optimized on this loss but instead the interest lie in the gradients that were accumulated for the *generators* with respect to this loss. Finally, the evaluation of $\hat{\mathscr{L}}_{\text{Cyc}}$ for G_{HA} and G_{AH} was done and an optimizer step was taken for these. The CycleGAN was trained for 25 epochs.

3 Results

The original, fake and cycled image for one arbitrary test set sample are shown in Fig. 3. It can be noted that the cycle consistency is very satisfactory but some slight color change. However, the fake images are close to identical to the original ones and not at all similar to the target domain. It is obvious that the CycleGAN has failed.

Moreover, in Fig. 4, the losses explained in Section 2.3 are displayed. The total loss (*upper left*) is the sum of the cycle (*upper right*), discriminator (*lower left*) and generator (*lower right*) loss. The cycle loss $\hat{\mathscr{L}}_{\text{Cyc}}$, which dominates the total loss due to the relative large weight λ goes down over the 25 epochs as a typical learning curve. This can be confirmed by cycle consistent images in Fig. 3.

The discriminator and generator losses, however, does not decrease as typical learning curves. This is nonetheless expected since there should in fact be a tug-of-war between the generator and discriminator. If the generator produces better fakes (generator loss goes down) it is more difficult for the discriminator to predict which domain the fake image belongs to (discriminator loss goes up) and vice versa. Nevertheless, there is no obvious pattern of tug and pull between the discriminator and the generator that shows up in the loss in Fig. 4. This is confirmed by the complete lack of target domain features in the fake images displayed in Fig. 3.

Similarily, the average accuracies of the discriminators shown in Fig. 5 indicates that the discrimination between the true and fake images was rather easy. The 0.5 target, which would indicate good fakes, was not reached. However, since the fakes are almost

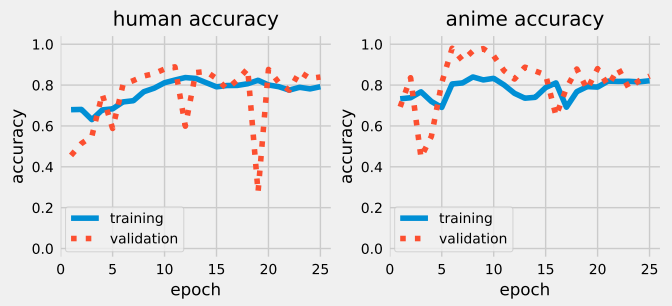

Fig. 5: The average accuracies for the discriminator networks on training and validation data. See Section 2.3 for explanation.

identical to the original images[2] it is perhaps a bit strange that the average accuracies of the discriminators is not higher.

Finally, in Table 1, the ultimate values of the losses and the accuracies are displayed for the training, validation and test sets. It is noted that the discrepancies are minimal for the three datasets.

4 Discussion and Conclusion

The generator architecture change, from the auto-encoder to the `U-Net`, did not achieve the sought result of allowing for morphological changes. In fact, it did the opposite and learned to map the images almost without any loss of information—it even lost the ability of style-change. This can perhaps be attributed to the `U-Net`'s skipping connections, the black arrows in Fig. 1. The skipping connections might let too much information from each individual example through the network. The most it accomplished is some slight discoloration of the input images.

Furthermore, notice in Fig. 4 how the discriminator and generator losses barely changed throughout the training process. Combine this with the fact that the discriminator accuracies never approached the 50% target, indicating that the generator failed to learn the target distribution. This is potentially due to that the cycle-consistency constraint is relatively too strong and or that the `U-net` is the wrong type of architecture for this task, as it need not generalize due to the skipping connections. It might also be the case that the discriminator is not learning properly (perhaps because λ dominates). Given the produced fakes, the task of discriminating these from real should be an easy one, however the average accuracies are just over 80%. Nevertheless, when testing the `Patch-GAN` on H and A, it reached an accuracy of 99% so the architecture should not be to blame.

Moving on from the depressing results, a CycleGAN model is extremely time-consuming to train. Depending on the training algorithm and the dataset size, it can take enormous time for one epoch to be completed. In this report, the final model took roughly one hour for every five epochs of training, which put a damper on the iterative development process of the system. Even using a subsample for early prototyping only mitigated this problem slightly; due to lowering the sample size meant more epochs are needed to get the same effect as when using the training set; that is, when having more batches, more update steps will be done and the model convergences faster per epoch. Add the need to tune the hyper-parameters λ and μ, and it can quickly become a project that takes months, if not the right hardware is available. If given more time, a structured grid or random search over the hyper-parameter space would have been conducted and more alternative architectures tested.

2. At least to the naked eye. Information might be hidden in the fake that is used for reconstructing the cycled image [11].

TABLE 1: The final losses and discriminator accuracies for the training, validation and testing sets (multiplied by their weights).

Dataset	Losses			Accuracy (%)	
	Disc.	Gen.	Cycle	Human	Anime
Train	0.322	0.726	1.042	84.94	83.60
Valid.	0.320	0.728	1.042	84.83	83.65
Test	0.322	0.728	1.039	84.85	84.29

If one tries to combat the aforementioned problem with a higher learning rate, the model's convergence can become highly unstable; in worst case scenarios, experienced from this project, it will spiral out of control and land in a sub-optimal solution. This instability can (somewhat) be seen on the two bottom panels of Figure 4, where especially the validation profiles often make large jumps—as the y-axis scales are relatively small. This can probably be associated to the many optimization objectives interacting with each other; that is, their co-dependence on each others' parameters, makes those with less dominant gradients (e.g. the discriminators) see their loss functions' values change drastically from update to update (their input distribution changes with the other networks' updates).

In conclusion, even though this work did not yield any spectacular results, the knowledge gained from working with developing multi-task self-learning systems has been invaluable. For example, understanding how the different objectives interact or how the change in architectures might not always lead to the expected results, both gives important experiences for when one is in a professional setting. Although, the chosen topic might have been too much work for a final course project, especially since the creators are new to deep learning.

References

[1] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2242–2251, 2017.

[2] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[3] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," *ArXiv*, vol. abs/1611.02163, 2017.

[4] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *ICML*, 2017.

[5] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *ArXiv*, vol. abs/1411.1784, 2014.

[6] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[7] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," 2017.

[8] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "CycleGAN," 2017. Accessed: 2021-09-28, https://github.com/junyanz/CycleGAN.

[9] NVIDIA Corporation, "Flickr-faces-hq." Accessed: 2021-09-28, https://github.com/NVlabs/ffhq-dataset.

[10] G. Branwen, Anonymous, and D. Community, "Danbooru2019 portraits: A large-scale anime head illustration dataset." https://www.gwern.net/Crops#danbooru2019-portraits, March 2019. Accessed: 2021-09-28, https://www.gwern.net/Crops#danbooru2019-portraits.

[11] C. Chu, A. Zhmoginov, and M. Sandler, "Cyclegan, a master of steganography," *CoRR*, vol. abs/1712.02950, 2017.