CycleGAN reimplementation for fantasy style transfer

Kim Targan, Kristof Engelhardt

April 2021

Contents

1 Introduction

In the last few years, painting, literature, music as well as other forms of art have been added to the vast fields of applications of artificial neural networks. Neural "art" includes replicas of famous artist's styles [e.g. 1, 9, 10] as well as new artistic creations on the basis of more varied sources (e.g. [3], art by Klingemann [7]). While in the beginning Convolutional Neural Networks (CNNs) have been used to create paintings from photos using a specific artist's style [1], an increasing number of network

architectures are developed for neural style transfer. A main obstacle is the limited availability of paired training images in the context of artistic image-to-image transformation: it may be possible to find and photograph some of the scenes long gone artists have painted but one can not ask the artists to just paint some more. Thus, for several applications paired training data may not be available. As a result, increasingly, approaches focus on using variants of generative adversarial networks to approach neural image translation as they work well without paired training images [e.g. 4, 10, 19]. In the first neural style transfer by Gatys *et al.* [1] the style of one specific piece of art was applied to their training dataset, the focus of this paper is the approach by Zhu et al. [19], who strive for a more varied and realistic transfer of style, using a collection of "style images" instead of just one single reference picture. On the basis of two datasets, their cycle-based network is designed to translate images between two domains in an unsupervised manner. This opens up a field of possibilities for image generation through combinations of content and style that are not limited to generating "real-world" scenes.

In this paper, we tackle the task of neural style transformation with the objective of producing fantasy world scenes. For this purpose we re-implement the cycle-consistent generative adversarial network proposed in 2017 by Zhu, Park, Isola and Efros [19] and consider different variations on their architecture. In the following we will discuss our motivation for the task at hand and shortly outline the general evolution of the field of "neural style transfer" as relevant to our project. In Section 3 we will outline the details of our Cycle-consistent Generative Adversarial Network (CycleGAN) architecture. This is followed by a discussion of the loss objective and training choices. Finally, Section 5 addresses the results we obtained and how these compare to the outcome of ablation studies on both architectural and training details.

2 Motivation

The creation of art can be both a political medium for activist aims, as well as a means of self-expression. However, arts and especially imagery is not all about the artist or creator behind it, in the mind of the observer art takes on a life of its own. In one's imagination non-pictorial forms of art become figurative and detailed scenes may appear, for this, the underlying idea of the creation process itself may become completely irrelevant. This is true for paintings and sculptures, but seems to have just as much relevance for novels, especially those from the realm of fantasy. Growing up with (and arguably in) the worlds built from fantasy author's words might have created a natural attachment to surreal images. For us, the discussion of neural art thus soon trailed back to the idea of (re-)creating images of the fantasy places that happy memories are connected to. Transforming landscape pictures into fantasy landscapes has thus come as an exciting objective to us. This objective may not be easily achieved with an architecture and dataset that fits the scope of this project and the outcomes may not really resemble the "Fangorn forest", "the shire" or "Hogsmeade", but nevertheless, it still seemed worth a try. Section 5.1 will document our attempts. However, due to the limited computational resources available to us and limited availability of reasonable datasets for the task, we will firstly discuss the performance of our network on one of the classic datasets in the context of CycleGANs: the dataset horse2zebra [19]. After studying the architectural details by evaluating our outcomes on

this dataset, we have then trained our network on scenic landscape images as well as oranges (see apple2orange [19]) and pattern datasets to see if some surreal fantasy images could be created.

As mentioned above, we have reimplemented the architecture proposed by Zhu *et al.* [19], which has been influenced by many previous approaches to the field of image transformation. In the following we will provide an introduction to their paper [19]. Furthermore, this section shortly touches upon some of the work that has shaped this field of research.

2.1 Image-to-image translation with Zhu *et al.*

In their paper [19], Zhu *et al.* propose a network that learns image-to-image translation without paired training examples. Therefore, their approach is valuable for settings where aligned input and output images are not readily available. To allow for an un- or at least semi-supervised training, Zhu *et al.* rely on two Generative Adversarial Networks (GANs). Thus, the loss is not based on a target image but rather implicitly imposed through the adversarial loss and additional objectives. In the absence of supervision through mapped ground truth images, the model is trained to learn a distribution close to the original distribution underlying the output domain. Thus, any single image of the input domain has a large corresponding space of mapping possibilities and the model does not just learn to create a single target image. To ensure that the model performs well nonetheless, Zhu *et al.* enforce the cycle-consistency of translation through an additional loss objective. As a result, the model is more resistant to mode collapse and is able to translate between both domains. Their approach differs from most previous implementations for image-to-image translation as they do not rely on predefined similarity functions or pretrained task-specific models and thus enables a more general application [19]. Especially in the context of neural style transfer this facilitates training, since no hand-designed feature statistics need to be employed. Finally, the network outperforms their chosen baselines and often offers similar performance as the "fully supervised" [19] conditional GAN "pix2pix" [4] previously proposed by some of the same authors [19]. In section 3, the approach by Zhu *et al.* and their architecture as adopted by us, is outlined in more detail.

2.2 Related work

Neural Style Transfer The famous approach by Gatys *et al.* [1] has been the start of various endeavours to use CNN variants for neural style transfer. Gatys *et al.* showed that the use of CNNs allowed for efficiently recombining content and style of images, as style is learnt in higher network layers while more semantically rich content information is captured in lower layers. The task of neural style transfer was consecutively adopted frequently and different architectures and methods have been proposed to either allow for better results or lower computational costs.

Perceptual losses and real time style transfer

On the basis of the earlier achievements in the context of image translation, Johnson *et al.* [6] aim to provide a faster approach to image transformation and style transfer as tackled by Gatys *et al.* [1]. By using a perceptual loss objective they were able to move away from the limited expressiveness

of pixel-based loss terms. A pixel-based loss is mainly useful for supervised training on paired input images where the generated image can be compared pixel-wise to the ground truth and it captures only low-level (pixel-based) features [6]. In their approach, Johnson *et al.* propose to incentivize higher level perceptual features by focusing on the reconstruction of (semantic) features not pixel-reconstruction. To accomplish neural style transfer, Johnson *et al.* combine an image transformation network and a loss network. For image transformation, Johnson *et al.* use a deep residual convolutional network that combines in-network downsampling, residual-blocks and consecutively upsampling to transform the input image to an output image. The approach of Johnson *et al.* [6] is particularly important for us, as Zhu *et al.* have adopted their image transformation network architecture for their generators.

pix2pix

With their Conditional GAN (cGAN) Isola *et al.* have proposed a method for image-to-image translation that can be easily applied to different datasets and tasks [4]. The paper goes hand in hand with the release of their pix2pix software which is available and can be used for transforming pictures online. Isola *et al.* propose a framework that allows to move away from hand-designed loss terms to impose certain features and towards a more generally applicable approach. This is enabled by the use of cGANs, as the adversarial loss term depends on the input images at hand and will ensure that the generator implicitly learns the features that the generated images need to satisfy to fool the discriminator [4]. The training is done in a supervised manner on paired input images, while the CycleGAN by Zhu *et al.* can be trained in an unsupervised setting. Nonetheless, the CycleGAN architecture is reminiscent of the pix-to-pix approach. Besides, as mentioned above, the architecture proposed by Isola *et al.* has been one of the baselines for Zhu *et al.*. One main difference can be found in the generator architecture where pix2pix is based on a "U-Net" structure [4], whereas Zhu *et al.* use a ResNet-based generator [19].

3 Architecture

Our network is oriented towards the implementation by Zhu *et al.* and thus, the referenced work that has been adopted by them in their approach. To allow for training the network in an unpaired unsupervised setting it is based on both adversarial training and a cycle-consistency loss objective. The resulting cycle-based generative adversarial network is built from two generators and two corresponding discriminators. The generators learn to translate images between domains: one generator is trained to translate the images from domain A (e.g. horses) to domain B (e.g. zebras), the other one works the other way round. This combination of generators allows for a cyclic translation: one can feed an image of domain A to the first generator and then the generated image to the second generator. The resulting image of the original domain A should in principle be close to the original image. This allows for introducing an efficient loss term based on cycle-consistency as further explained in Section 4.1. The introduction of the cycle consistency loss adds an element of implicit supervision to the network. For a precise visualization see Figure 1, which was used by Zhu et al. in [19]. Coherently, there

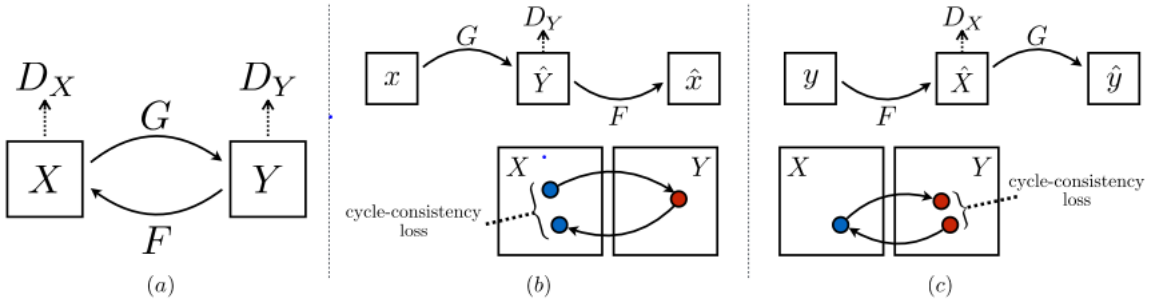

Figure 1: Illustration of the CycleGAN by Zhu *et al.* [19]. Visualizes both the structure of the GAN architecture as well as the cycle-consistency component. The cycle consistency loss is split into two components as illustrated in b) and c). An image of domain Y is translated to domain Y by generator G. As a second step the generated image is passed to generator F to recreate an image of the original domain X. On this basis one side of the cycle consistency loss can be computed.

are also two discriminators: one for each domain. The discriminators learn to distinguish generated images and images from the underlying true multiplex of non-linear distributions (instead of assuming a latent space described by a single gaussian normal distribution) [10]. The true images are example images from the dataset of the respective domain. In the following the implementation of both network components is further illustrated.

3.1 Generator

For their generator architecture, Zhu*et al.* refer to Johnson's implementation of an image transformation network [6]. This refers to a convolutional network which includes both downsampling and upsampling. Additionally, the architecture includes five residual blocks which, as introduced by He *et al.*, enables the network to learn the identity function more easily [2]. This is especially important for neural style transfer, where large parts of the input image should remain the same while only the style such as texture and colour should be changed. Downsampling is done prior to the residual layers and is achieved by using strided convolutions rather than pooling layers. The "convolutional layers are followed by batch normalization [...] and ReLU nonlinearities" [6]. Interestingly, Zhu *et al.* claim that "[s]imilar to Johnson *et al.* [...] [, they] use instance normalization". This inconsistency is likely due to the fact that the implementations of Johnson's network which are published online include instance normalization [5]. Besides, the training choices by Zhu *et al.* with image datasets in batches of size 1, do not allow for batch normalization (batch norm). Thus, the closest design choice is relying on instance normalization as implemented by Ulyanov et al. [15, 16] which has often been deemed promising for neural style transfer. Instance normalization works in a similar manner as batch norm, however instead of normalizing a whole batch of images to one gaussian distribution, instance norm normalizes over the feature maps (channels) of a single image with image specific gaussians. As a result, the contrast within single images is normalized and has a lower impact on the learning process, which according to Ulyanov *et al.*. is desirable for image stylization [16]. In addition to this, instance

normalization is also used during testing. However, even though instance normalization has a positive impact on performance, different explanations have been proposed. While Ulyanov *et al.* attribute the improvements to the resulting contrast normalization [16], Huang *et al.* claim that it causes a style normalization [3]. Nonetheless, the implementation of the earlier released pix2pix paper [4] employs batch normalisation instead and due to the fact that Zhu *et al.* reference the cGAN from pix2pix as a basis of their work, some re-implementations of CycleGAN that can be found online reuse their architecture [e.g. 14].

Upsampling is achieved by using fractionally (0.5) strided convolutions, thus it is performed in-place. Computationally, employing down- and upsampling even though input and output are supposed to have the same size is beneficial, as the costs for a deeper network can be reduced. Additionally, downsampling is advantageous as it allows to efficiently change larger image parts by increasing the receptive field (i.e. the area affected) of the convolution on each pixel. Thus, with the same number of layers more convolutional transformation of the image sections is enabled [6]. For our generator we focussed on the architectural details explicitly mentioned by Zhu *et al.*, however, some of the differences that exist between Johnson *et al.* and other publications referenced by Zhu *et al.* are addressed in further ablation studies (see Section 6). Finally, the generator we used for neural style transfer follows Zhu et al. in their implementation for 128x128 images and thus comprises 6 instead of 5 residual blocks as employed by Johnson *et al.* [6], 7x7 sized kernels for the first and final layer instead of 9x9 and instance normalization [19]. Further, reflection padding is used in order to reduce artifacts that occur when using zero padding. With reflection padding the underlying data distribution is better represented as it simply "reflects" the values of the row or column that it extends [17]. The specific implementation we used is taken from keras and published in this article [12]. However, the guidelines by Zhu *et al.* about reflection padding are not completely clear and the implementations published on their website [18] differ. Thus, we chose to compare the simple use of reflection padding instead of zero padding which seems to be intended by Zhu *et al.* to the way reflection padding is implemented by Johnson *et al.*. For further discussions of these differences see Section 6.2.

3.2 Discriminator

The CycleGAN relies on two Discriminators for training in an adversarial fashion on both image classes. Zhu et al. use 70x70 PatchGAN discriminators which evaluate 70x70 image patches instead of the whole image at once. As described by Isola et al. [4], the discriminator consists of several convolutional layers with Instance Normalization and LeakyReLU activation. The convolutions are performed using 4x4 kernels with strides 2. For each convolutional layer we use zero padding. Thus, the resulting feature-map size is halved every convolutional layer. For our 128 x 128 images the output of the discriminator is an array of dimensions 5x5 with a classification score for each image patch. The use of PatchGAN is beneficial as the smaller patches reduce the number of parameters, allow for faster computation and work on arbitrarily sized input images. Plus, by penalizing image patches instead of the whole image, one can introduce a lower-feature texture (style) loss [4]. In comparison to point-wise feature analysis, patches are said to produce more photo-realistic results [10]. Patch based approaches

such as Li and Wand's markovian GAN [10] allow for capturing statistical distributions on local patches but might not always be able to reproduce the overall image structure. Finally, PatchGAN works well for neural style transfer, however, it might also be a reason why there are problems with adapting underlying global shapes. Zhu *et al.* describe problems with transforming dog to cat pictures [19] due to this as well. In Section 6, further insights into the effect of patch size variations are discussed.

4 Training Details

4.1 Losses

As mentioned above the whole network relies on the introduction of a cycle-consistency objective. However, this is not the only loss term employed. Mainly, the network is based on the defining characteristic of GANs - adversarial losses for generator and discriminator and in some cases an additional identity loss term is used.

Adversarial losses

The adversarial character of GANs allows for learning mappings between input and target domain. GANs rely on two networks (or in this case four) that work against each other. A generator that produces images and a discriminator that tries to predict whether an image is generated or real. The losses of both network components are based on the discriminators classification of the generated images. Additionally, the discriminator loss is dependent on the discriminators performance on real images from the domain. Finally in the case of a CycleGAN, the generator that translates images of domain A to domain B depends on the discriminator of domain B and a second pair works the other way round. Zhu *et al.* use a least-squares loss for the adversarial objective [19]:

> In particular, for a GAN loss $\mathcal{L}_{GAN}(G, D, X, Y)$, we train the [Generator] G to minimize $\mathbb{E}_{x\sim p_{data}(x)}[(D(G(x))-1)^2]$ and train the [Discriminator] D to minimize $\mathbb{E}_{y\sim p_{data}(y)}[(D(y)-1)^2] + \mathbb{E}_{x\sim p_{data}(x)}[D(G(x))^2]$.

However, the actual implementations referenced on their website [18] show-case a range of losses. While the least-squares loss is employed most often, some also include implementations of the common negative log likelihood objective using binary cross-entropy for classification [11]. Further studies on the loss ablations are included in Section 6.

Cycle-consistency loss

To avoid mode collapse, a common problem with generative adversarial networks, Zhu et al. have introduced a cycle consistency loss term. The objective ensures translation consistency in both directions. The generated images of the generator translating domain A to domain B (generator B) are passed to the other generator (A) in order to recreate an image from the original domain. This resulting double translated image of generator A is compared to the original input image and the difference between both images is measured. Importantly, cycle consistency is computed in both directions, thus resulting

in a forward and backward loss term. For computing the loss of one generator, both cycle consistency terms are taken into consideration. Zhu et al. stress that using only forward cycle consistency resulted in significantly worse performance. Finally, by ensuring cycle-consistency the generated images are tied closely to the original image and thus, mode collapse becomes less likely. The cycle consistency loss is weighted by the factor lambda which is generally set to 10.

Identity loss

In the original paper [19], an identity loss term is used for certain applications in addition to the other loss objectives to allow for further preservation of input image properties. Zhu *et al.* use the identity loss for artistic style generation on a Monet dataset, arguing that it helps to preserve the colours of the original input. If it is used, the identity loss is generally weighted with 0.5*lambda where lambda is the weighting factor used on the cycle consistency loss (often lambda is set to 10). Several re-implementations such as the simplified CyleGAN tutorial by tensorflow [14] make use of this identity loss. The underlying idea is to test the performance of each generator on the images of its own output domain. Thus, for the horse2zebra dataset the generator that transforms horses to zebras will be fed with images of zebras and will output zebra images on this basis. These generated images are consecutively compared to the original images and the loss is the mean of the absolute difference between the images. [19]:

$$\mathcal{L}_{\text{identity}}(G, F) = \mathbb{E}_{y \sim p_{data}(y)}[\|G(y) - y\|_1] + \mathbb{E}_{x \sim p_{data}(x)}[\|F(x) - x\|_1]$$

For our purposes, the use of the identity loss seemed to significantly improve stability and general performance, thus we decided to include it in our main implementation. However, further discussion of its influence on generated images are discussed in Section 6.

Further training choices

As a start we adopted the hyperparameters proposed by Zhu. *et al.*: a batch size of 1, Adam optimizer with a learning rate of 0.0002 and lambda l=10 for weighting the loss objective. Zhu *et al.* train their network for 200 epochs and use a gradually decreasing learning rate. Besides, the discriminator loss is calculated on the basis of the last 50 generated pictures for which an image buffer is used. For our final implementation, we trained the network for 100 epochs with Adam optimizer and the fixed 0.0002 learning rate. The discriminator is not pre-trained as this did not seem to improve results.

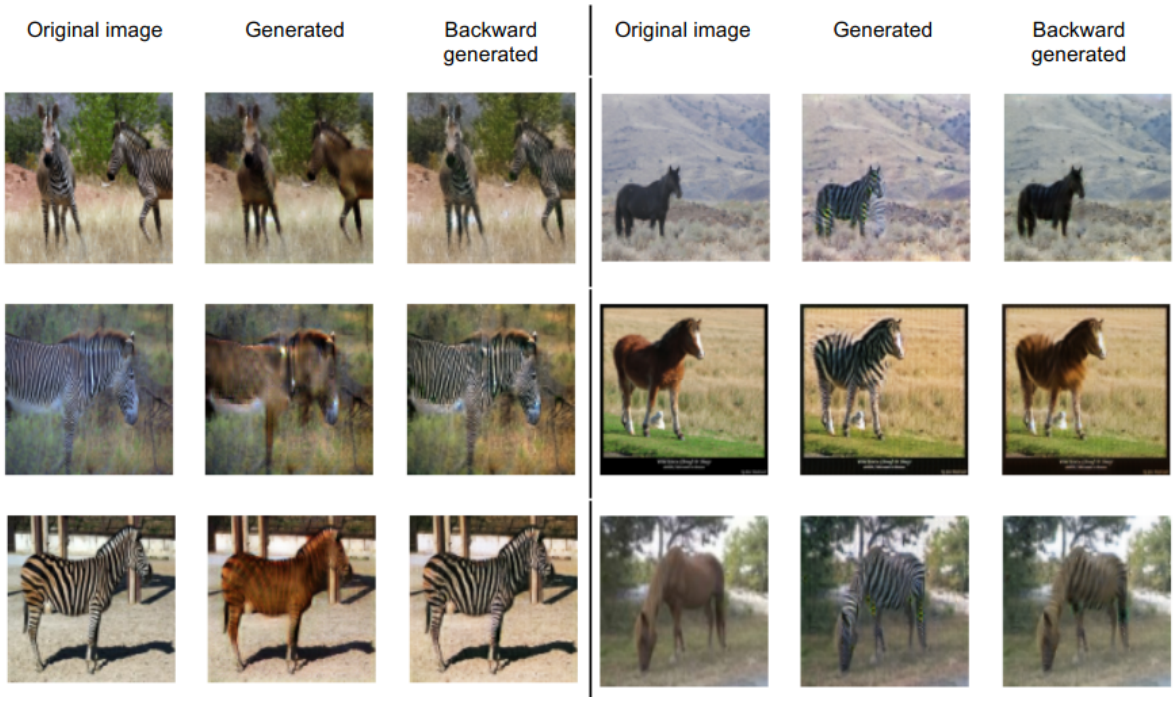

Figure 2: Generated images after training for 100 Epochs

5 Results and applications

As mentioned before, the CycleGAN was optimized on the basis of the horse2zebra dataset [19] associated with [19]. For this task of translating images of horses to zebras and vice versa, the network trained for 30 epochs using the GPU offered on Google Colab already provided very good results for some training images. Further training for 100 epochs improved and stabilized the model performance. Images generated after 100 epochs can be seen in Figure 2 and showcase that the model already performs well. Nonetheless, there are still several images that the network seems to struggle with. Overall, images where the whole animal is visible but not too small and portrayed on a lighter background work well. As Zhu *et al.* train for 200 epochs we assume that additional training will further improve performance, however we are satisfied with the results already obtained and aimed to limit the use of GPUs. Therefore, we opted for a shorter training.

5.1 Fantasy images

We approached our objective to obtain images that resembled fantasy scenes and worlds by combining different datasets. Several experiments were conducted with a landscape dataset published on kaggle [13] and additional smaller self-designed datasets from images that were released with freedom of use on pixabay. The self-created datasets comprised between one and two dozen style images, which we further copied and randomly cropped to obtain a larger style dataset. As a style reference we used

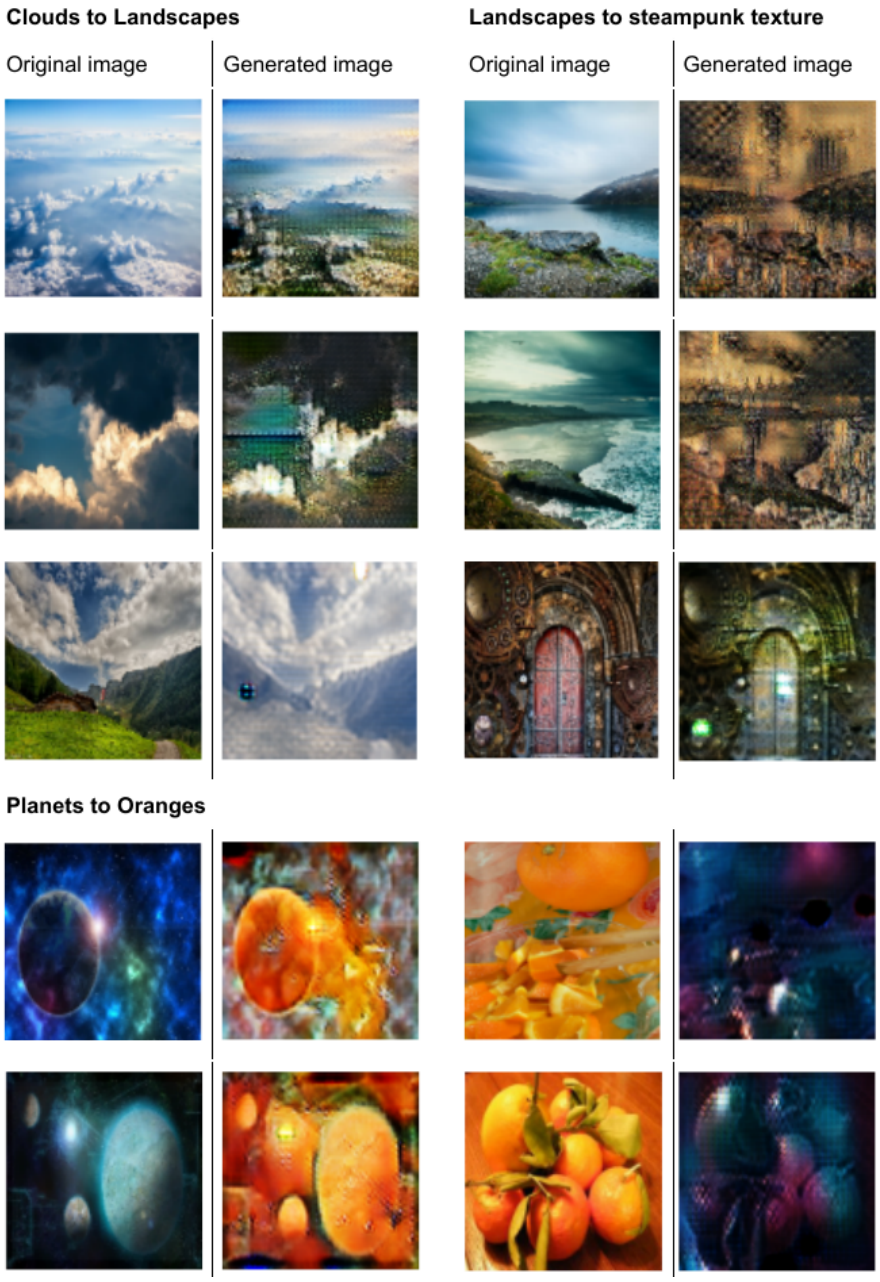

Figure 3: Results after training for 30 epochs on several "fantasy" datasets.

either simple texture images (e.g. repeated metallic 'steampunk' textures) or fantasy scenes. Some interesting results for this setup were obtained, however several limitations can be observed. Mainly, the loss objectives and patch-GAN discriminator choice work well for translating styles and texture onto images that have underlying similar conceptual shapes. However, the introduction of new shapes into the underlying image is generally not achieved. The fact that more extreme transformations were not achievable was already noted by Zhu *et al.* in their paper [19]. Thus, it is advisable to use datasets with a similar underlying image structure. Nonetheless, the use of mere pattern images for one domain did translate to the generated images even after as little as 1 epoch. Therefore, it is possible to apply textures to images even if there is no structural similarity in the content of the images. The results of pairing landscape scenes with different textures, are relatively blurry versions of the original scenes with different colour schemes. The pixelation is likely due to the shorter training (30 epochs) and might be further improved by adapting the loss ratio. Figure 3 depicts some of the generated images for a 'steampunk' style set as well as a 'cloud' style set. Additionally, we combined the orange pictures from the dataset "apple2orange" [19] with pictures of planets and universe scenes, to create some surreal new planets. The resulting floating orange worlds can be observed in Figure 3 as well. Finally, longer training and a larger style dataset may result in even clearer transformations. Nonetheless, the universe of oranges might also offer an interesting setting and inspiration for the next science fiction series?

6 Ablation studies

As alluded to in Section 3, there were several differences between referenced works and the architecture by Zhu *et al.*. Besides, overall, the architecture descriptions offered by Zhu *et al.* are limited. This allowed us to examine different architectural choices and their influence on the results. However, in consequence of the scope of the project, it was not possible to accurately measure model performance with suitable methods such as large scale evaluation procedures as performed by Zhu *et al.* with their perceptual studies on a group of participants [19]. Thus, we had to rely on our own observations on the pictures generated by models trained for only 30 epochs instead. To obtain comparable images, we took the results of our final network (trained for 30 epochs) as a baseline (denoted as CycleGAN in all figures). All other results were obtained with slight variations on the network and training process but also stem from training for 30 epochs. We predefined the first 30 images of the test datasets as our sample space and let each network generate images on this basis. Thus, we were able to compare the generated images of the networks on the same subset of test images. On this basis, each of us compared the generated datasets to the baseline output and took note of observations to assess the network's performance. This "evaluation" method is not designed to meet scientific standards, but rather to allow for drawing conclusions about general trends. It has to be noted that the performance of a specific model on different images varies which can further distort evaluation. It would have been more ideal to include a larger sample but this would exceed the scope of the project assignment. Nonetheless, to allow for some comparisons, we still think that the two independent evaluations of each of us will enable some cautious conclusions and provide some insights into the efficiency of certain

design choices. The Figures 4 and 6 depict the results of several ablations on our final CycleGAN architecture.

6.1 Training Procedure and baseline

It should be noted that we slightly changed the training procedure in comparison to the implementations of CycleGAN published online [18]. Zhu et al. state in their paper, that they "update the discriminators using a history of generated images rather than the ones produced by the latest generators. [...] [They] keep an image buffer that stores the 50 previously created images" [19], which we interpreted as training the discriminator with 50 of the generated images every training step. As the data is batched to samples of 1 the discriminator is trained with only one "true" sample and in our case 50 generated samples, thus there is a difference in proportion. Several implementations online only train the discriminator with one image from the training buffer [e.g. 8, 11]. However, we noticed that for us the results were improved when training on the last 50 images in every training step (exemplary images can be found in Figure 5 and 4). Thus, our final training procedure includes this slight variation which is already part of our baseline CycleGAN. Besides, we decided against pretraining the discriminator as the results we obtained after 15 epochs were already clearly worse. Overall the generated pictures were more colourful, however few stripes were generated or removed and the network was still stuck with recreating the input image. In the following, the results of several variations will be discussed.

Batch Normalization

One main advantage of batching the images (batchsize 16) and applying batch normalization instead of instance normalization is faster training epochs, since less normalization operations are needed. However, the generated images are relatively noisy especially for white horses and white patches in the sky. Most importantly, the model is not able to generate any zebra pattern in the first 30 epochs. When transforming zebras to horses, some brownish patches appear. Nonetheless, the network performs significantly worse than the baseline model with instance normalisation, since hardly any stripes are obscured. One possible explanation for the inferior image generation might be that normalizing images in relation to the other images in the batch, shifts the focus away from the specific images and thus does not address specific variations that could obscure the process. According to Huang *et al.* [3] batch normalization causes a whole batch to be normalized to one style but does not prevent intra-batch style differences (i.e. differences between single images). In contrast to this, instance normalization "can normalize the style of each individual sample to the target style" [3]. Finally, instance normalization is generally found to be superior to batch normalization [e.g. 3, 16].

6.2 Architectural variations

Comparison to the architecture by Johnson *et al.* [6]

Even though their generator architecture is based on Johnson *et al.* [6], Zhu *et al.* have significantly adapted the network. Thus, we were curious to see if the original implementation by Johnson *et al.*

would offer a worse performance. The main architectural differences can be found in the way that padding is used. Johnson *et al.* rely on a 40x40 reflection padding on the input image to obtain in- and output images of the same size. Throughout the residual layers, no padding is used, instead the input image is cropped. This is necessary to allow for the subsequent in- and output addition between each residual block. Besides, Johnson *et al.* apply a 9x9 instead of a 7x7 filter in the first and last layer and use less residual blocks than Zhu *et al.*. Overall the Johnson architecture seems to perform nearly as good as our baseline model. One downside we observe is that the zebra stripes are sometimes more pixelated and noisier. However, more zebra patterns are introduced. For the zebra to horse transformation, the performance is similar as well. At first, it may seem like the baseline model is superior because it adds reddish brown color patches to the zebras. However, the "Johnson model" is equally good in obscuring zebra stripes - the color is just not as striking, so the horses have a more greyish tone. Finally, the models would need to be trained for more epochs to better evaluate and compare the performance.

PatchGAN discriminator

Further variations can be introduced by changing the discriminator implementation. While the use of PatchGAN which uses smaller image patches seems to work well, we did assess the models performance on the full "128x128 patch" image. In comparison to our baseline model with patch size 70x70 the performance on the horse to zebras transformation is clearly worse. The baseline model is able to apply at least a weak zebra pattern to the majority of horses. In contrast, the full image PatchGAN has difficulties with applying any pattern. When there is a style transformation, it is very noisy and hardly identifiable as black and white stripes. Surprisingly the quality of generated horse images is nearly equal to the performance of our baseline model. It seems to be only slightly worse at obscuring zebra stripes. The success seems to be due to a general blurring of the relevant areas, thus the results are lacking in clarity. However, the dark brown color applied to the zebra pictures (when using a full GAN discriminator) seems to be a little more natural than the reddish brown tone of our classic model. The discriminator with patch size 128x128 is clearly better equipped for detecting global patterns, which might cause the success in pattern transfer. However, it has difficulties with local textures (as described in Section 3.2), which would explain that the applied patterns are more pixelated. Finally, it may be easier for the model to generate large brown patches for transforming zebras to horses, while it has difficulties generating the more detailed zebra patterns.

6.3 Ablations of Loss Objective

Binary Cross-Entropy

Swapping the least-squares loss proposed by Zhu *et al.* for a negative log likelihood objective, results in worse model performance. Thus, even though some implementations online use [14] or include binary cross-entropy [11], it is advisable to stick with the replacement. Generated images for binary cross-entropy can be observed in Figure 4 and 5. The generated "zebras" are very pixelated - there are patches of black and white roughly covering the animals which slightly resemble stripes. However, the

blurring exceeds the edges and the overall results are worse than the baseline model with least-squares loss. In the zebra to horse transformation a change in colour is introduced which we did not observe in any other implementation.

Identity loss

As the identity loss is an optional but frequently used component of the loss objective, we also experimented with training without it. Leaving out the identity loss resulted in the generation of more zebra patterns on the horse images and a compared to the baseline model similar removal of patterns on the zebra images. However, the performance is inferior to the baseline in two ways: firstly, the shapes of the animals are not segregated well when generating, thus large parts of the surrounding area are transformed to the target pattern as well. Secondly, the generated images are overall more blurry. This makes sense, since the identity loss punishes the two generators if they are bad at mapping images of their own target domain to the domain itself, as described in Section 6.3. Therefore the loss incentivises the network to recreate the original image in more detail. Finally, similarly to Zhu *et al.* [19], we also observed a change in colour when leaving out the identity loss, which also hints at the inferior performance at preserving original image features.

Weighting factor

Another ablation on the loss objective can be performed by changing the weighting factor that controls the influence of the different loss terms. The weighting factor lambda influences how much of a focus is placed on the cycle-consistency and identity loss. For the horse2zebra dataset we initially adopted a lambda of 10 as proposed by Zhu *et al.* [19]. The results of this choice can be observed in the column for our baseline model in Figure 6 and 7. Attempts of lowering lambda to 5 and increasing it to 20 are depicted in the respective columns of the same Figure. Overall, lowering the lambda factor seems to result in a significant worsening of the generated results. The pattern translation is not as accurately bounded to the animals but rather covers large areas, producing significant artifacts. Especially, for the transformations of zebras to horses the main achievement seems to be a blurring process on the stripes. To sum up, it seems like the identity loss and cycle-consistency loss are crucial for model performance and weighting them with 10 instead of 5 significantly improves performance. Coherently, we also trained the network with a weighting factor of 20. The generated zebra images did not differ drastically from our baseline and establishing a definite trend is thus hard. If anything, the performance seems to improve for an increased lambda. However, the generated horse images showcased a green strain and it is hard to establish what exactly went wrong. Finally, the weighting term seems to be an important influence on model performance and there might be a sweet spot somewhere in the range of 10 to 20 for our model and dataset.

Overall, it seems like the architectural choices by Zhu *et al.* already maximize performance at least for short training periods. Neither explicit changes in the loss function, nor batching resulted in compelling images. However, especially increasing the weighting factor lambda offer the potential of improving the performance on the horse2zebra dataset. Besides, a generator implementation more closely oriented

towards the architecture by Johnson et al. seems promising (if not even superior) as well.

7 Conclusion

We re-implemented the CycleGAN proposed by Zhu *et al.* to transform the style of a landscape dataset with the help of different pattern image sets. We have discussed the basis of their approach and implemented different variations based on the rough guidelines of their implementation. The results on both the dataset horses2zebras [19] and paired custom datasets were delineated above. Finally, the goal of transforming scenic images to fantasy landscapes was approached with a small corpus of datasets and we obtained a range of different results. Specific textures and colours can be easily transferred to the landscape images, which allows for creating more surreal images. Nonetheless, as mentioned by Zhu *et al.* the network is not designed for achieving more radical content transformations [19]. Finally, we did create some interesting space images from combining an orange and universe dataset, which had a global shape commonality - round shapes on a darker background. Thus, we conclude that it would be worthwhile to put in the effort and design appropriately matched datasets which, with a longer training, are likely to result in more detailed fantasy images.

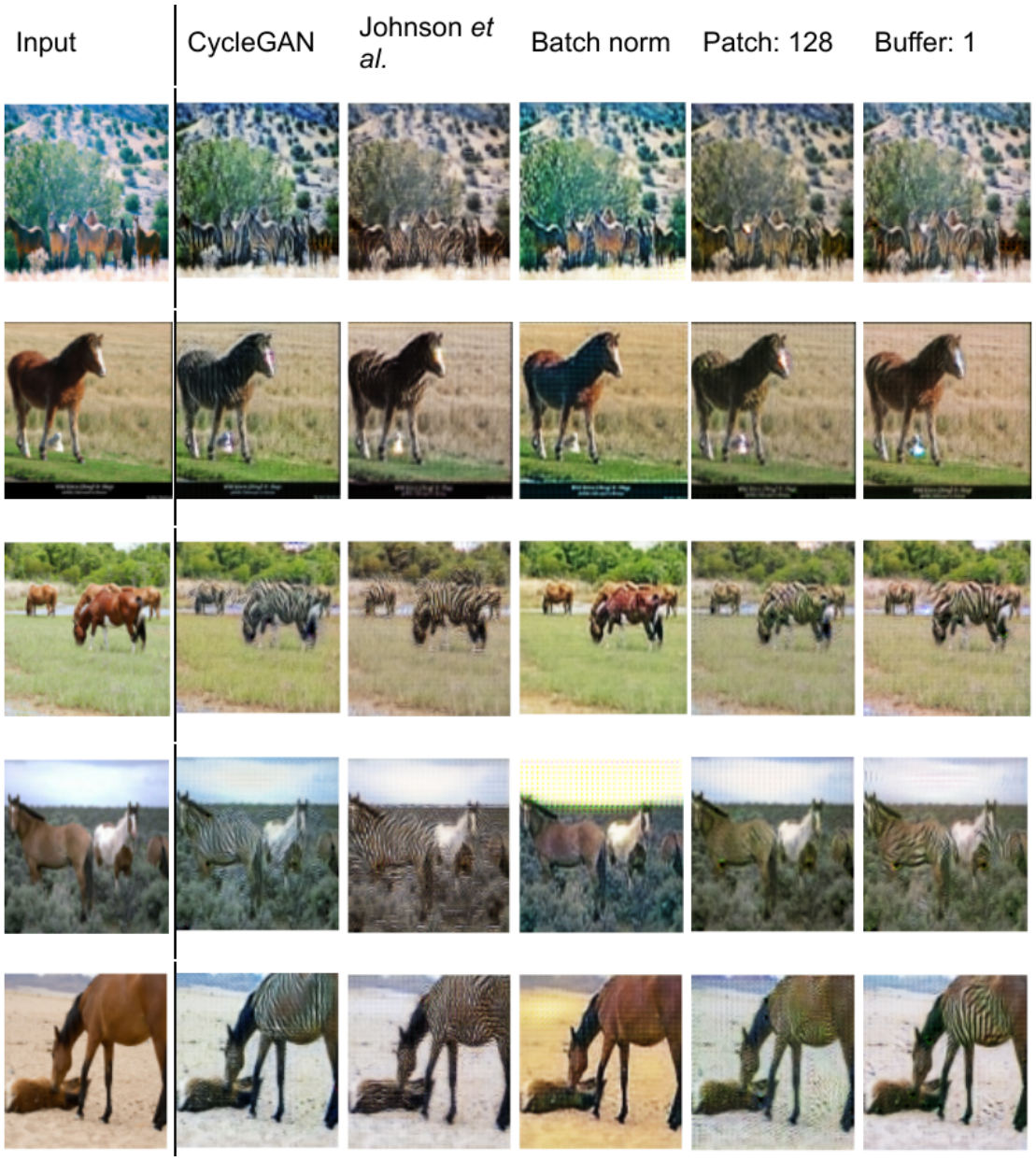

Figure 4: Generated zebra images after training variations of the network for 30 epochs as discussed in Section 6

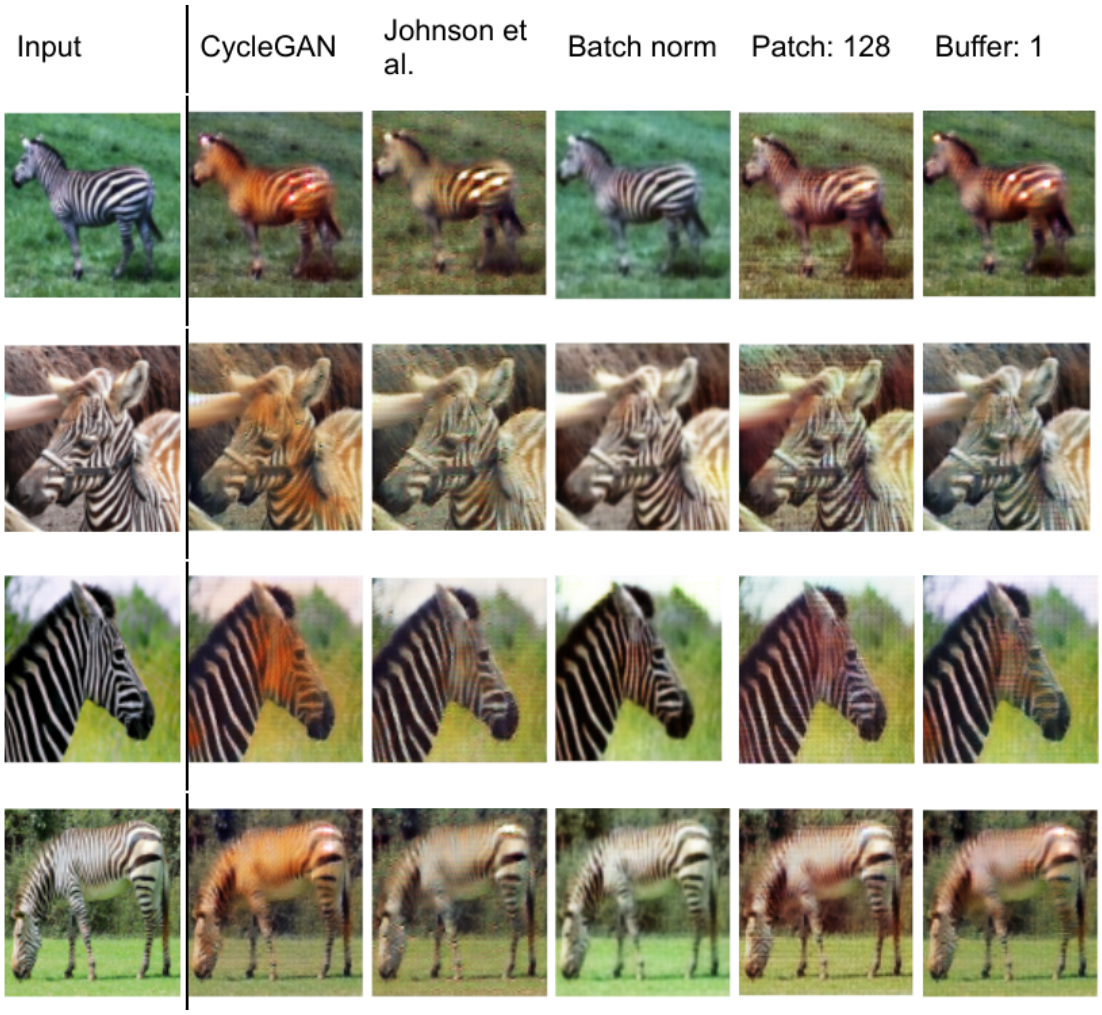

Figure 5: Generated horse images after training variations of the network for 30 epochs as discussed in Section 6

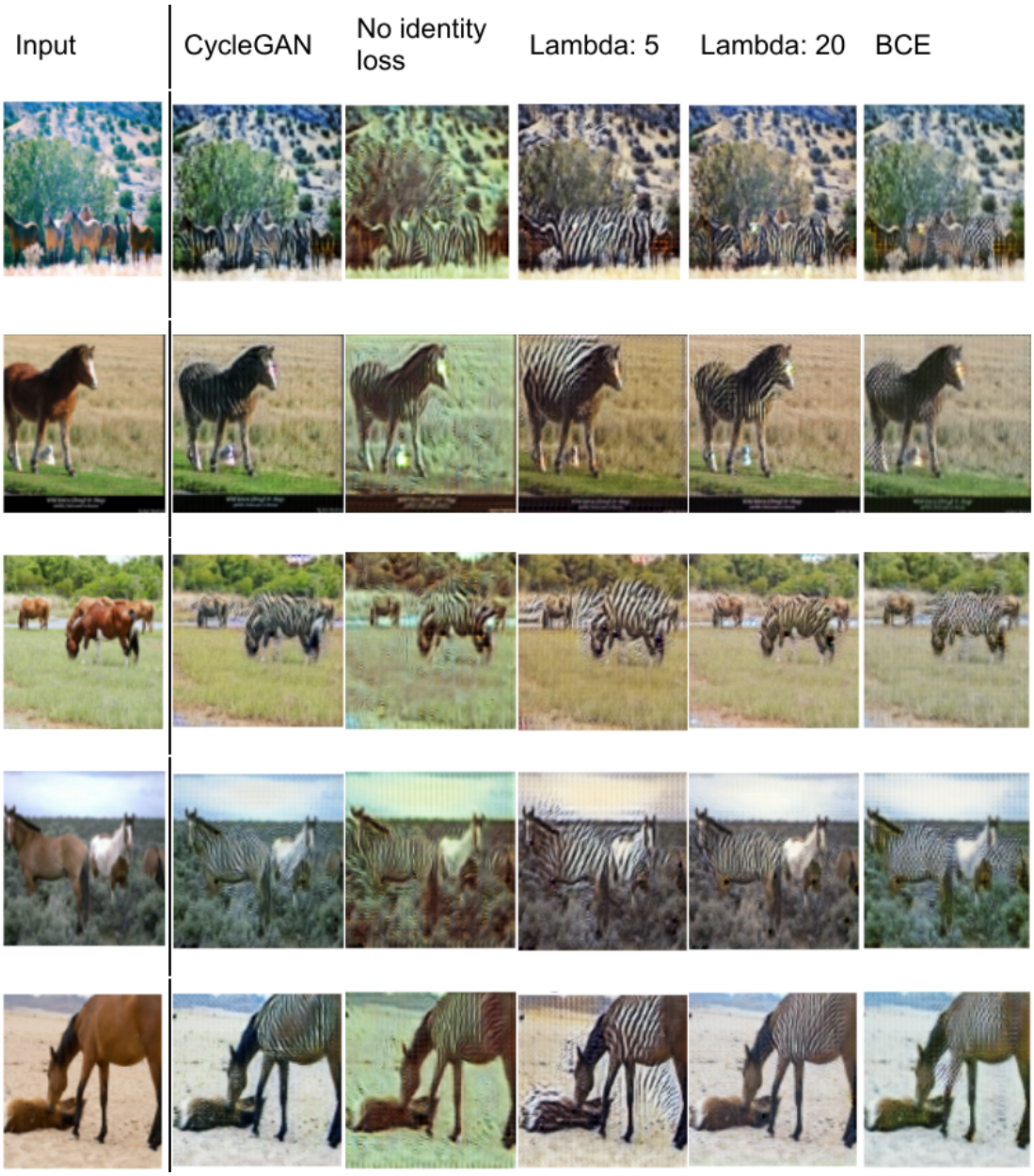

Figure 6: Generated zebra images after training variations of the network for 30 epochs as discussed in Section 6. This specific table displays loss ablations. BCE indicates the use of binary cross-entropy as adversarial loss function.

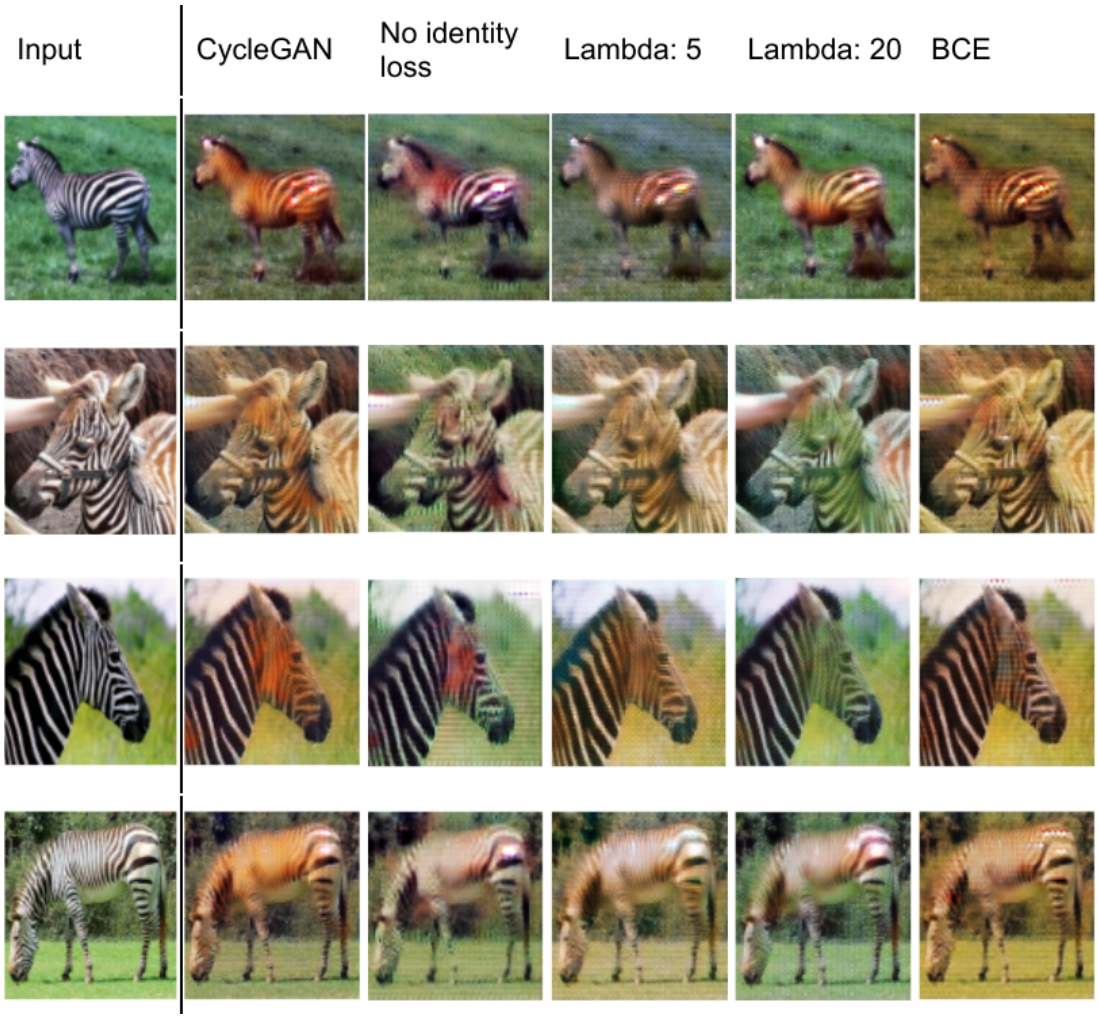

Figure 7: Generated horse images after training variations of the network for 30 epochs as discussed in Section 6. This specific table displays loss ablations. BCE indicates the use of binary cross-entropy as adversarial loss function.

References

1. Gatys, L. A., Ecker, A. S. & Bethge, M. *A Neural Algorithm of Artistic Style* in. **abs/1508.06576** (2015). `https://dblp.org/rec/journals/corr/GatysEB15a.bib`.

2. He, K., Zhang, X., Ren, S. & Sun, J. *Deep Residual Learning for Image Recognition* in (June 2016), 770–778.

3. Huang, X. & Belongie, S. J. Arbitrary Style Transfer in Real-Time with Adaptive Instance Normalization. *2017 IEEE International Conference on Computer Vision (ICCV),* 1510–1519 (2017).

4. Isola, P., Zhu, J.-Y., Zhou, T. & Efros, A. *Image-to-Image Translation with Conditional Adversarial Networks* in (July 2017), 5967–5976.

5. Johnson, J. *jcjohnson/fast-neural-style* https://github.com/jcjohnson/fast-neural-style.
6. Johnson, J., Alahi, A. & Fei-Fei, L. *Perceptual Losses for Real-Time Style Transfer and Super-Resolution* in *Computer Vision – ECCV 2016* (eds Leibe, B., Matas, J., Sebe, N. & Welling, M.) (Springer International Publishing, Cham, 2016), 694–711. ISBN: 978-3-319-46475-6.
7. Klingemann, M. *Quasimondo* https://underdestruction.com/category/work/.
8. Leehomyc. *leehomyc/cyclegan-1* https://github.com/leehomyc/cyclegan-1.
9. Li, C. & Wand, M. Combining Markov Random Fields and Convolutional Neural Networks for Image Synthesis. *CoRR* **abs/1601.04589.** arXiv: 1601.04589. http://arxiv.org/abs/1601.04589 (2016).
10. Li, C. & Wand, M. *Precomputed Real-Time Texture Synthesis with Markovian Generative Adversarial Networks* in *Computer Vision – ECCV 2016* (eds Leibe, B., Matas, J., Sebe, N. & Welling, M.) (Springer International Publishing, Cham, 2016), 702–716. ISBN: 978-3-319-46487-9.
11. LynnHo. *LynnHo/CycleGAN-Tensorflow-2* https://github.com/LynnHo/CycleGAN-Tensorflow-2.
12. Nain, A. K. *Keras documentation: CycleGAN* Aug. 2020. https://keras.io/examples/generative/cyclegan/.
13. Rougetet, A. *Landscape Pictures: Datasets of pictures of natural landscapes* https://www.kaggle.com/arnaud58/landscape-pictures?select=00000000_%5C%282%5C%29.jpg.
14. Tensorflow. *tensorflow/docs* Jan. 2021. https://github.com/tensorflow/docs/blob/master/site/en/tutorials/generative/cyclegan.ipynb.
15. Ulyanov, D., Vedaldi, A. & Lempitsky, V. *Improved Texture Networks: Maximizing Quality and Diversity in Feed-forward Stylization and Texture Synthesis* 2017. arXiv: 1701.02096.
16. Ulyanov, D., Vedaldi, A. & Lempitsky, V. *Instance Normalization: The Missing Ingredient for Fast Stylization* 2017. arXiv: 1607.08022.
17. Versloot, C. *Using Constant Padding, Reflection Padding and Replication Padding with TensorFlow and Keras* Feb. 2020. https://www.machinecurve.com/index.php/2020/02/10/using-constant-padding-reflection-padding-and-replication-padding-with-keras/#reflection-padding.
18. Zhu, J., Park, T., Isola, P. & Efros, A. A. *CycleGAN* 2021. https://junyanz.github.io/CycleGAN/.
19. Zhu, J., Park, T., Isola, P. & Efros, A. A. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *CoRR* **abs/1703.10593.** eprint: 1703.10593. http://arxiv.org/abs/1703.10593 (2017).