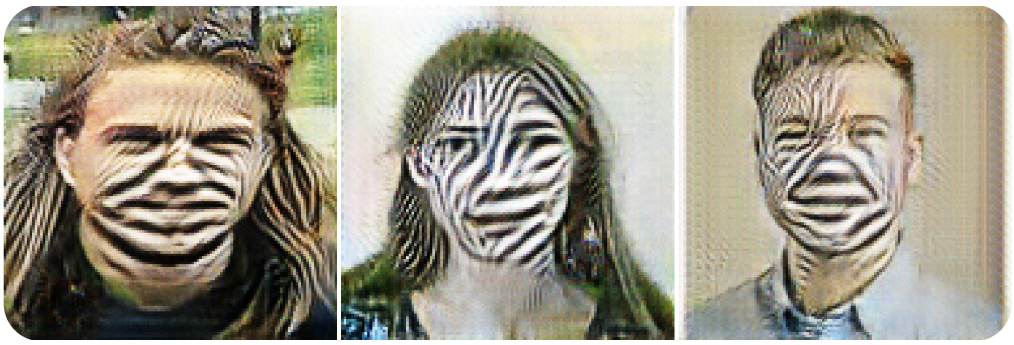
The Technical University of Denmark

Deep Learning in Computer Vision

Project 3 - Generative Adversarial Networks

Authors:
Tobias Konradsen (s144077)
Helena Hansen (s153178)
Christian Dandanell Glissov (s146996)

Supervisors:
Aasa Feragen
Morten Hannemose



June 25, 2020, Kongens Lyngby

DTU Compute
Department of Applied Mathematics and Computer Science

DTU

1 GAN generation - MNIST

A Generative Adversarial Network (GAN) is a generative model, that are capable of creating new data that resemble the training data. The network consists of two parts: a generator, that learns to generate plausible data from an input of random noise, and a discriminator, a classifier that learns to separate the generator's fake data from the real data. The generator learns to create fake data by incorporating feedback from the discriminator by calculating loss from the discriminators classification, which in turn will make it harder for the discriminator to distinguish between the real and fake data, which forces both models to grow. As a result, both models should learn at the same pace in order to get better results. If a GAN continues training after the discriminators feedback becomes more random, as it cannot distinguish between fake and real data, then the quality of the generator suffers.

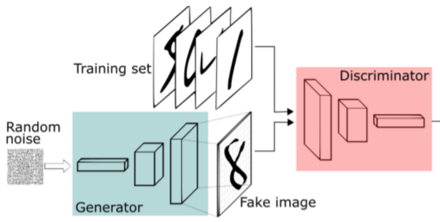


Figure 1: Overview of GAN for MNIST (Slides)

GAN can be utilized on the MNIST dataset, which is a set consisting of 70,000 small square 28×28 images featuring numbers ranging from 0 to 9, to generate new plausible handwritten digits, as illustrated in Figure 1.

1.1 Architecture

Two different implementations of GANs were tested, the first implements both the generator and discriminator network as fully connected neural networks and will be referred to as Vanilla GAN (VGAN), while the second follows the guidelines for Deep Convolutional GANs (DC-GAN), Radford et al., 2015. These difference can among other be found in the removal of the fully connected hidden layers and new utilization of strided convolutions. Convolutional nets looks for spatial correlations within an images and the local spatial coherence by looking at patches of the images from DC-GAN allows for a much more efficient training.

The architecture of the DC-GAN generator consist of four stacks of 2D transposed convolution layers with batch-normalization and ReLu functions, taking 100 input channels, going to 1024 in the first stack, then halving the number of channels in each stack until the last where it outputs 1 channel. The DC-GAN discriminator consist of four stacks of convolutional layers with batch-normalization and Leaky ReLu functions, taking 1 input channel, going to 128 in the first stack, then doubling the number of channels in each stack until the last where it outputs 1 channel. Both generator and discriminator had the same learning-rate of 0.0001 for `Adam`.

1.2 Loss function

For the VGAN we utilize the minimax loss, Goodfellow et al., 2014, the loss featured in the paper that introduced GANs, where the generator seeks to minimize the Shannon-Jensen divergence, e.g. the divergence between log-probability of labeling real and fake images correctly, while the discriminator seeks to maximize it:

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

Another loss function we tested was the Wasserstein-1 loss function

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim \mathbb{P}_r}\left[f_w(x)\right] - \mathbb{E}_{z \sim p(z)}\left[f_w\left(g_\theta(z)\right]\right.$$

Here the discriminator tries to push the means of the two distributions as far apart as possible. To limit the ability of the discriminant to push the distributions arbitrary apart, the Lipschitz continuity must be satisfied. In our case it is being satisfied by using spectral normalization of the convolutions. For the MNIST data the results between the two losses did not seem to significantly differ, so we only show results using the minimax loss.

1.3 Results

The results of some randomly generated digits can be seen in Figure 2 for VGAN and DC-GAN. When comparing the two it can be seen, that those produced by DC-GAN have much less noise and better shapes, and it is thus, a visible improvement for the digit generation. It was tested, that if the

VGAN was allowed to run for 15 more epochs most of the noise vanishes, but DC-GAN seems to still produce consistently better results.

The simplicity of the generated objects helps to make most of the generated numbers seem plausible to the human eye. As numbers are quite similar and few changes of details are needed in order to transition one number to another, e.g. 7, 9, 2, the visually worst results came from numbers with more distinct shapes. 8 being one of the harder, as the generator needs to make two distinct closed circles.

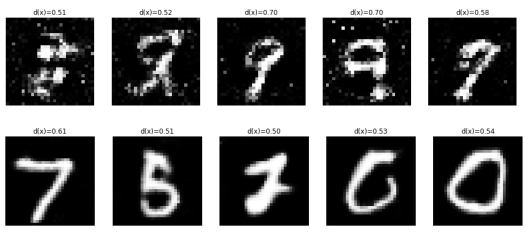


Figure 2: Results of VGAN (first row) and DC-GAN (second row) after 5 epochs

2 CycleGAN - Horse/Zebra

CycleGAN is a way of performing image-to-image translation from a source domain to a target domain without paired examples. It learns the mapping $G : X \to Y$ and its inverse $F : Y \to X$ as two different sets of weights for a generator.

The data set consist of 1067 training and 120 test images for horse, and 1334 training and 140 test images for zebras. For the CycleGAN to be successfully, it has to be capable of giving plausible transformations of horses into zebras, and vice-versa.

2.1 Architecture

In order for the CycleGAN to function we optimize over two discriminators, one capable of classifying for horses (A) and one for zebras (B), and two generator, one transforming horses to zebras (A2B), and one transforming zebras to horses (B2A). An overview for part of the training can be seen on figure Figure 3. For an input of type A (Input_A) , generator A2B will generate an image of type B (Generated_B). This image is then recovered into type A again by using the generator B2A (Cyclic_A). The discriminator A is applied on Input_A image, where a decision is made. Likewise for the generated image, but with discriminator B. Both the input, generated and cyclic image is used to train the generator.

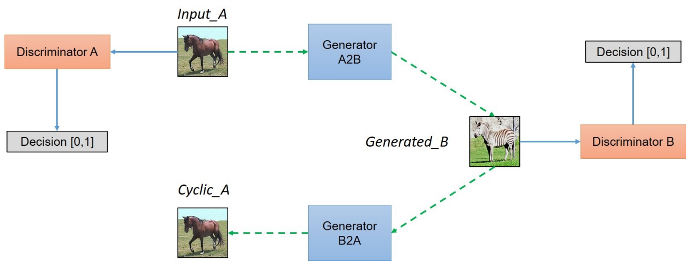


Figure 3: Network cycle for input of type A (horse) https://hardikbansal.github.io/CycleGANBlog/

The same sequence of transformations are used on an input of type B, but where the generators and discriminators are switched around.

2.1.1 Discriminator

The discriminator architecture consists of stacks of convolutional layers with instance normalization, and leaky ReLU functions, it follows mainly the structure from, Zhu et al., 2017. The instance normalization reduces the chance of vanishing gradients, by reducing covariate shift from the change of distributions between the non-linear activations. It normalizes over each batch independently and the spatial locations, which also avoids noise from other instances, in contrast to batch normalization that normalizes across the entire batch. The leaky ReLU helps in preventing mode collapse by avoiding sparse gradients. Mode collapse refers to the generator learning to produce one especially plausible output, and thus repeatedly produces only that output.

For the CycleGAN case, the LSGAN loss was used. For both the discriminator A and B the loss function we utilize is:

$$\min_D V_{\text{LSGAN}}(D) = \frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}\left[(D(\boldsymbol{x}) - b)^2\right] + \frac{1}{2}\mathbb{E}_{\boldsymbol{y}\sim p_{\boldsymbol{y}}(\boldsymbol{y})}\left[(D(F(\boldsymbol{y})) - a)^2\right]$$

Where we choose $a = 0$, $b = 1$ as this can be understood as the fake image has label 0 and the real image, label 1. The discriminator should minimize the squared distance to 0, e.g. classify the fake images as fake. The same reasoning applies for the real images. This minimizes the Pearson χ^2 distance between $p_{data} + p_g$ and $2p_g$, thus a way for p_g to converge to p_{data}, Mao et al., 2016.

2.1.2 Generator

The generator utilized in the project was handed out as part of the assignment. We tested for different number of residual blocks and use of dropout, as a way to provide noise and avoid overfitting.

In order to train the generator we distinguish between three different losses; fooling the discriminator, maintaining cycle consistency and minimizing identity loss, where the loss associated with fooling the discriminator utilize the same loss function, LSGAN, as the discriminator. The total loss function for the generator is then:

$$\min_G V_{\text{LSGAN}}(G) = \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_y(\boldsymbol{z})}\left[(D(F(\boldsymbol{y}))-c)^2\right] + \lambda_{cyc}\left\|F(G(x))-x\right\|_1 + \lambda_{id}\left\|F(x)-x\right\|_1$$

Where $c = 1$, indicating the "real image", so to say that G wants the discriminator to believe the fake image is real. The loss for cycle consistency and identity ensures, that the generator mappings remains consistent with the original input, as $F(G(x)) \sim x$ and $F(x) \sim x$ for generator A2B, as well as $G(F(y)) \sim y$ and $G(y) \sim y$ for generator B2A. The direct pixel comparison between two images means, that the L1 or L2 norm can be utilized for this, where L1 tends to produces sharper results, Isola et al., 2016. The values utilized are $\lambda_{cyc} = 10$ and $\lambda_{id} = 5$ as they were found to be optimal in the paper Zhu et al., 2017.

2.2 Methods, training and modifications

2.2.1 Data Augmentation

Two types of data augmentation was applied to the images, a horizontal flip, and a random cropping to a 128 by 128 image. To make the random cropping work, we resized the image to slightly bigger than the 128 by 128 image in order to get a slight random cropping of the image. These augmentations were chosen to ensure that the motive of the image is still preserved.

2.2.2 Augmentations and consistency loss

Consistency loss is a loss (weighted by a hyper-parameter λ_c) added to the discriminator loss to make sure the discriminator is invariant to small transformations:

$$L_{cr} = \|D(x) - D(T(x))\|_2^2$$

Where x is the input into the discriminator, so in our case the generated images. $D(x)$ is the output of the discriminator with input x. $T(x)$ is then the transformed input. From Zhang et al., 2019 a small translation of a few pixels and then a horizontal flip, was the best method for the CIFAR-10 dataset to reduce the FID-measure of the generated images with a λ_c of 100. In our case a λ_c value of 100, is resulting in a totally smooth generated image. A λ_c-value of 1 is used in this case. With this value the image was found to have better continuity. As the augmentation is on the input of the discriminator, it was necessary to make a separate augmenter using the package `imgaug`, Jung, 2018.

2.2.3 Adding a buffer

One problem with only generating new images, is the possibility that some errors are learned or forgotten by the generator repeatedly, thus never getting closer to the wanted target. In order to alleviate this, a pool of old refined images is also added to the training. This enables old generated images to be stored in the the buffer, which we can sample from. To introduce new samples in the buffer we replace uniformly on average half of the images with newly generated images. The method improves stability of the training Shrivastava et al., 2016. The amount of stored generated images is set to 50, but more experimentation is needed in order to know whether this is optimal.

2.2.4 Decreasing learning rate

The learning rate will decrease after a certain amount of epochs, following the same training schedule as in Zhu et al., 2017. The learning rate is set to 0.0002, for both discriminators and generators, and will then linearly decay after 100 epochs. This prevents overshooting of the convergence algorithm, which in this case is `Adam`, Kingma and Ba, 2014.

2.2.5 Fretchet Inception Distance

To evaluate if a transformation possesses the invariance property or the quality of images generated by GANs, the FID is used. It uses the Fréchet distance, Dowson and Landau, 1982, to compare two

Gaussian distributions, $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\mathcal{N}(\boldsymbol{\mu}_\mathcal{T}, \boldsymbol{\Sigma}_\mathcal{T})$ and is given by:

$$d^2((\boldsymbol{\mu}, \boldsymbol{\Sigma}), (\boldsymbol{\mu}_\mathcal{T}, \boldsymbol{\Sigma}_\mathcal{T})) = \|\boldsymbol{\mu} - \boldsymbol{\mu}_\mathcal{T}\|_2^2 + \mathrm{Tr}\left(\boldsymbol{\Sigma} + \boldsymbol{\Sigma}_\mathcal{T} - 2\left(\boldsymbol{\Sigma}\boldsymbol{\Sigma}_\mathcal{T}\right)^{1/2}\right).$$

The parameters $(\boldsymbol{\mu}, \boldsymbol{\Sigma}), (\boldsymbol{\mu}_\mathcal{T}, \boldsymbol{\Sigma}_\mathcal{T})$ are obtained from the distribution of the activation of a deep layer in the classification network when original images and augmented images are fed to the network, respectively. The Gaussian distribution is used because it has highest entropy given specific first and second moments, Heusel et al., 2018. In this case it is used to measure how well our GAN performs using a pre-trained `inceptionv3` network to get the features. A lower FID indicates better-quality images, conversely, a higher score indicates a lower-quality image.

2.3 Results

We trained both a CycleGAN and a CycleGAN with consistency regularization. The CycleGAN was trained for 180 epochs and the CycleGAN with consistency regularization for 220. As can be seen in Figure 4 the discriminator for the model with consistency regularization, does not converge as fast and fluctuates more but has a slightly lower generator loss, it perhaps indicates a better regularization. We also see that the adversarial loss increases, indicating that it becomes more difficult for the generator to fool the discriminator.

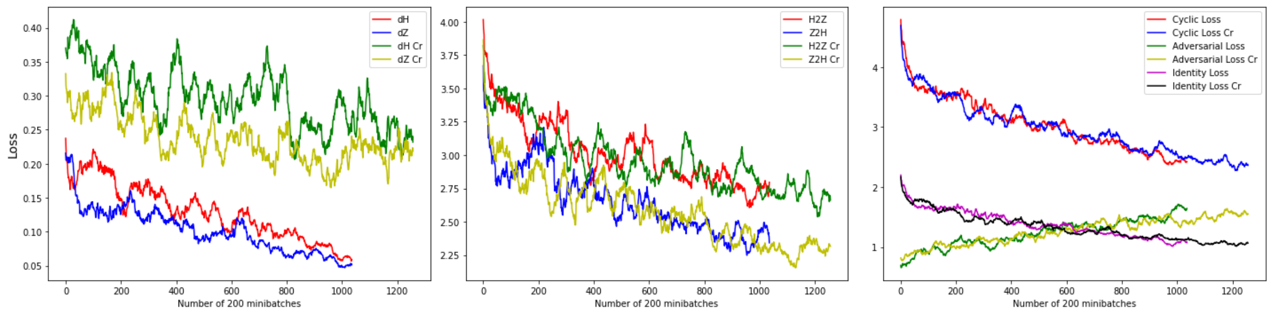


Figure 4: The different GAN losses during training. A moving average has been used, as using only 1 batch creates large fluctuations in the losses, hiding the trend.

Examples of the CycleGAN's transformations can be seen in Figure 5. Here the generated images seem plausible. The generated images are more blurry, than the real images, and is not completely convincing of the opposite class, but the generated images definitely closer reassembles the opposite class, than their true class, so the model do, to some extend actually produce proper images.

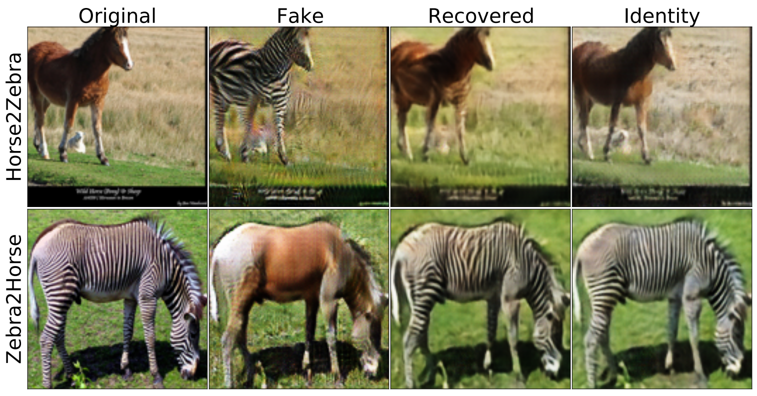


Figure 5: Example of results for the CycleGAN

The CycleGAN struggled in some cases. It seems to struggle making black horses to zebras at times, this is maybe because of the limited training set consisting of black horses. The CycleGAN also struggled when the background was in the same color range as the horse, then the model had a difficult time segmenting background and motive. Finally, when the horses and the zebras are very zoomed in, e.g. you can not see the whole horse, then the CycleGAN might not be able to produce good looking images, this might be because the CycleGAN is very reliant on the shape of the horse when transforming. The CycleGAN also struggled with the the opposite cases with very far away horses or outlier images, as the horses might look like small dots in the horizon and the outlier images might not lie on the models training manifold, generating strange examples. (See appendix subsection 4.1)

In Figure 6 a comparison of the FID-score between two different models are made and these are compared against three baselines. In the figure the FID-score is calculated for generated images and real images, with and without a consistency loss. The models are compared to two baselines (H/H baseline, Z/Z baseline), which is the FID-score between two real images. This is done as to have an approximate lower limit on best achievable FID-score for the model. Baseline (H/Z) is the difference between the real images of zebras and horses. This is to have an approximate upper limit of the FID score for furthest distance between a real horse image and zebra image. If the model achieve an FID-score lower than baseline (H/Z) then we achieve a better similarity between our generated and real images.

For the generated images of horses (Z2H) the FID-score did not seem to get closer to the lower limit when compared to the upper limit baseline (H/Z), this could be because the stripes of the zebra is difficult to transform, and the finer detail of the zebras is difficult to reconstruct, while still maintaining the cyclic and identity loss. It may also be partly due to the horses having a larger span of possible colours, while the model seems to favour a transformation only into brown hues, while the zebras have much less variety.

The generated images of the zebras got closer to the lower limit baseline. This is possibly due to a horse having less detail, and therefore is easier for the model to find plausible details of the generated zebra, while still maintaining the cyclic and identity loss.

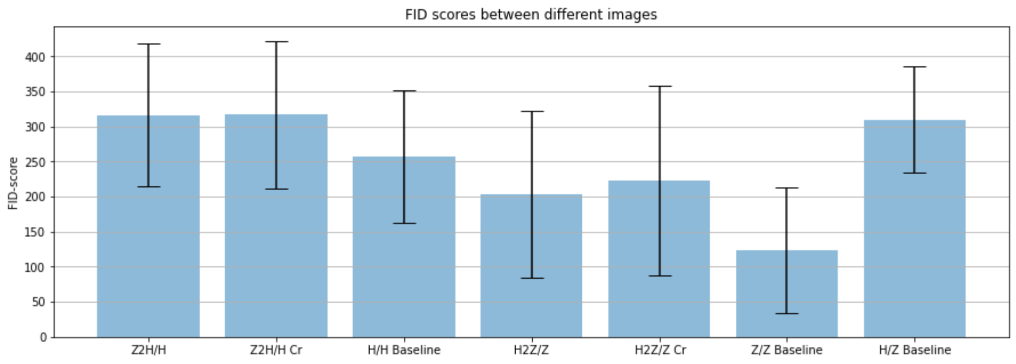


Figure 6: H indicates the horse images and Z the zebra images. H2Z is then the generator transforming a horse image to a zebra image. The slash indicate what we compare with in the FID, so H2Z/Z means we compare the fake images to the real images of zebras. Cr is the model with consistency regularization. The baselines are used to compare the score of the generated images.

2.4 Reflection

The cycle consistency loss enables the CycleGAN to train without paired data, as it forces the real and reconstructed image to be similar, but details are lost in the transformations, A2B and B2A. If the stripes from a zebra is totally removed, then the reconstructing will not be capable of recreating these to match completely and the cyclic loss will keep detecting this.

GANs can produce sharper pictures than a Variational Autoencoder due to multiple possible reasons. A main reason is that the VAE tries to generalize data by forwarding it through a bottleneck. When generalizing data, information is lost and this will cause a blur in the reconstructions. The discriminator for GAN may also learn to detect blurry edges, which allows the generator to learn to produce sharper results in order to improve its loss function, this is called super-resolution.

Both L1 and L2 are sensitive to pixel changes, but L2 has a disadvantage in GANS. The L2 penalizes errors a lot more, so if the ground truth of an image is two different colours, then the L2 will choose a "safe" approach and take the average between them, which visually can be a colour quite different from the two options. An example being red and blue as ground truth and yellow as output. As we have multiple colour options for the horses, the model would quite possibly suffer from a L2 loss.

In conclusion, we managed to transform a zebra to a horse and vice versa. It seems that the CycleGAN models are better at transforming horses to zebras. Adding a consistency loss did not seem to improve the overall performance of the model, and actually seems to make it worse, and there is still room for improvements.

3 References

Dowson, D., & Landau, B. (1982). The fréchet distance between multivariate normal distributions. *Journal of multivariate analysis*, *12*(3), 450–455.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial networks.

Heusel, M., Ramsauer, H., Unterthiner, T., & Nessler, B. (2018). Gans trained by a two time-scale update rule converge to a local nash equilibrium, In *Arxiv:1706.08500v6*.

Isola, P., Zhu, J., Zhou, T., & Efros, A. A. (2016). Image-to-image translation with conditional adversarial networks. *CoRR*, *abs/1611.07004*arXiv 1611.07004. http://arxiv.org/abs/1611.07004

Jung, A. B. (2018). imgaug [[Online; accessed 30-Oct-2018]].

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization.

Mao, X., Li, Q., Xie, H., Lau, R. Y. K., Wang, Z., & Smolley, S. P. (2016). Least squares generative adversarial networks.

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks [cite arxiv:1511.06434Comment: Under review as a conference paper at ICLR 2016]. http://arxiv.org/abs/1511.06434

Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., & Webb, R. (2016). Learning from simulated and unsupervised images through adversarial training.

Zhang, H., Zhang, Z., Odena, A., & Lee, H. (2019). Consistency regularization for generative adversarial networks.

Zhu, J.-Y., Park, T., Isola, P., & Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks.

4 Appendix

4.1 Failure cases

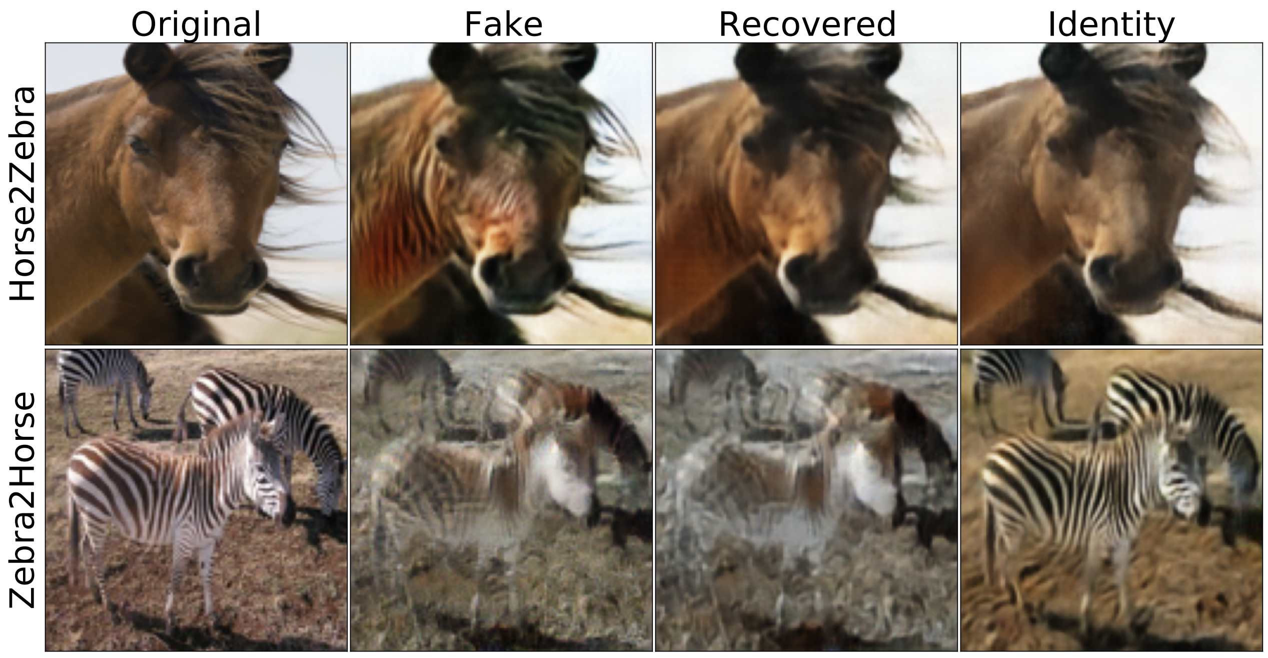


Figure 7: Failure cases of the CycleGAN, here the zebra is make color as the background and the horse is not trasformed.

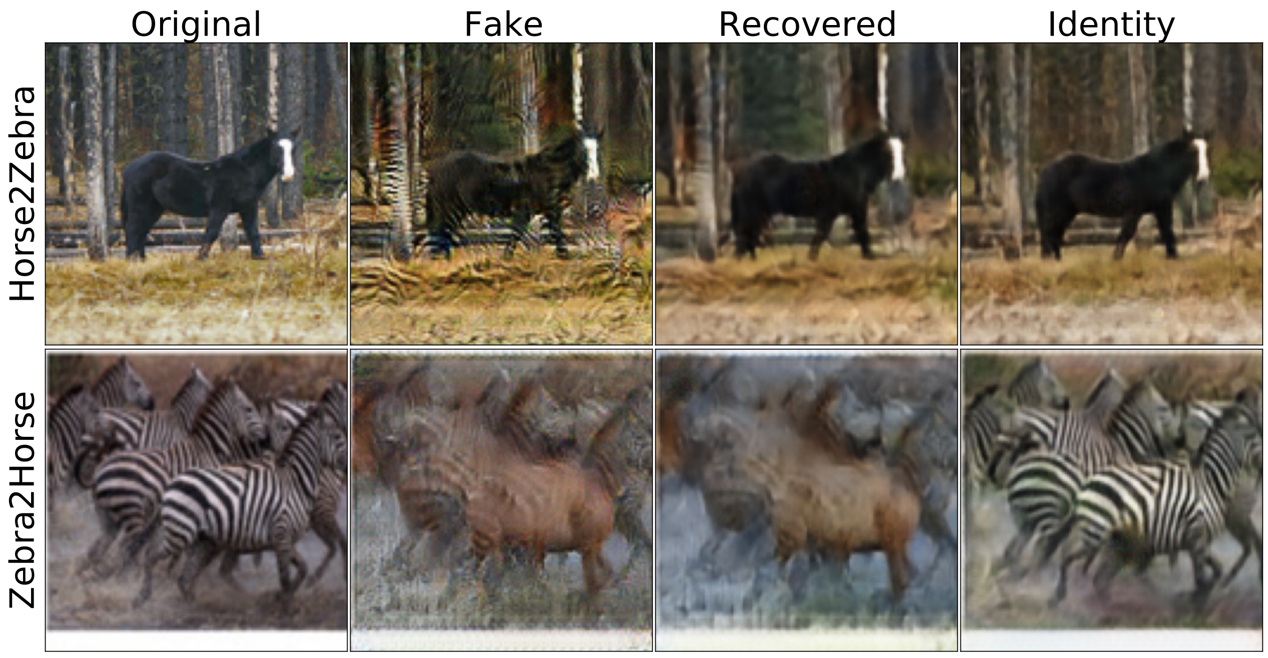


Figure 8: Failure cases of the CycleGAN, here the horse is not transformed as it is black, and the zebra is not segmented properly.

Diary

19-06-2020

This day all of the team member worked together to implement the training exercises of the MNIST-dataset.

21-06-2020

In collected effort an working implementation of the cycleGan network was implemented. Helena made the dataloader, and discriminator architecture. Tobias made the first prototype of the cyclegan architecture. Christian made a more workable version of the code.

22-06-2020

All group members worked on improvements of the architecture. Started writing on the project(Helena). Implemented consistency loss (Tobias). Collected, debugged and adjusted the code, there was a bug in the loss, making the code give bad results (Christian).

23-06-2020

Writing on the report (Tobias, Christian and Helena). Debugged the code. (Christian).

24-06-2020

Evaluating the results with the FID and training (Tobias and Christian). Writing the result and reflection sections (Helena, Christian and Tobias).