Deep Learning Project
Inpainting on images of faces

Rick Clement (s3852903) Remco Pronk (s2533081)
Sudhakaran Jain (s3558487) Jan Willem de Wit (s2616602)

February 4, 2020

Abstract

In this paper, we present a comparative study of a conditional Generative Adversarial Network (cGAN) and an Autoencoder inspired by U-Net which are both implemented to reconstruct images of faces with parts of the face missing. We use a dataset of celebrity faces for training these networks. We then conduct test experiments on these trained networks and compare their results based on few criteria. We find that the autoencoder works well to restore faces, and using this autoencoder with adversarial loss does not improve the results.

1 Introduction

Image inpainting is the task of filling in missing parts of images. Some mathematical models exist to perform this task [7]. However, these kinds of methods only work when small parts are missing, and don't work when the missing part should have a texture. Nowadays, deep learning is used more often for noise reduction purposes. One example of this is the denoising autoencoder [8], which learns to restore an image that was corrupted by noise. In this paper, we will try to restore face images of which a large part is missing. We try to perform this task using two different neural networks namely, an autoencoder inspired by the U-Net [6] and a conditional Generative Adversarial Network (cGAN)[3]. Finally, we will compare the results of both these networks and discuss which network performed better and also propose some strategies for further improvement.

2 Methods

2.1 Dataset

We use the CelebA dataset [2]. This dataset contains 202,599 images of 10,177 people. There are two variations of the dataset: one has "in the wild" images which are the complete, larger photographs, whereas the other has images that have been aligned and cropped. We chose to use the aligned and cropped version of the dataset. The images are in colour, and have a resolution of 178 x 218 pixels. We have downscaled the images to either 64x64 or 128x128 resolution, depending on the experiment.
We will create holes in the centre of the image. The holes have a width and height that is equal to 30% of the image size. The hole is filled with white pixels. The holes are big enough to obstruct a large part of the face, but they still leave some clues about what the face is supposed to look like, such as the edges of the eyes. Figure 1 shows how such images look.

2.2 U-Net

Our architecture for a generator network was inspired by the U-Net [6]. The U-Net is a network that was originally designed for segmentation of medical images. It consists of several layers in which the input is downsampled (encoder), and then several layers in which the input is upsampled again to restore an image of the same resolution as the input image (decoder). The U-Net has skip connections between corresponding layers in the encoder and decoder stage. This way, information that is normally discarded

Figure 1: An image with a hole in the centre

in the encoding process is retained. We varied the amount of layers. The architecture of the network we used for the experiments is shown in figure 2. We trained and tested this network on images that have been downscaled to a 128x128 resolution.

2.2.1 Losses

We compared 2 different losses: the mean squared error (MSE), and Structural Dissimilarity (DSSIM). DSSIM is a variant of Structural Similarity (SSIM) [9], but scaled so that a lower value is better, so it can be used as a loss function. The SSIM between two images x and y is defined as follows:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

where $C_1 = (K_1 L)^2$ and $C_2 = (K_2 L)^2$, L is the range of pixel values (for example 255 in an RGB image), and K_1 and K_2 are small constants that we set to their default values of 0.01 and 0.03 respectively.
The DSSIM is then defined as follows:

$$DSSIM(x,y) = \frac{1 - SSIM(x,y)}{2}$$

In our experiments, we combined DSSIM with the standard MSE loss. The total loss is then:

$$L(x,y) = \frac{MSE(x,y) + DSSIM(x,y)}{2}$$

By using the structural similarity measure as a loss function, we hope that the network learns to create sharper images.

2.3 GAN

We also inpainted faces using a Generative Adversarial Network (GAN) [1]. A GAN is a network that has a generator and a discriminator part. The generator generates an image from a noise vector. The discriminator then has the task to distinguish real images from a dataset from the fake images generated by the generator. The performance of the discriminator is then used as the loss for the generator, so that the generator learns to create images that are hard to distinguish from real images. An extension of the GAN is the conditional GAN (cGAN) [3]. This takes an image as the input, and outputs a new images. This can be used for image to image translation. In our case, we want to go from an image with a hole in it to a restored image, so we will use a cGAN for this. As the generator, we use the same U-Net like network, but with a smaller amount of layers. The architecture of the discriminator is the same as in DCGAN [4]. We trained and tested this network on images that have been downscaled to a 64x64 resolution.

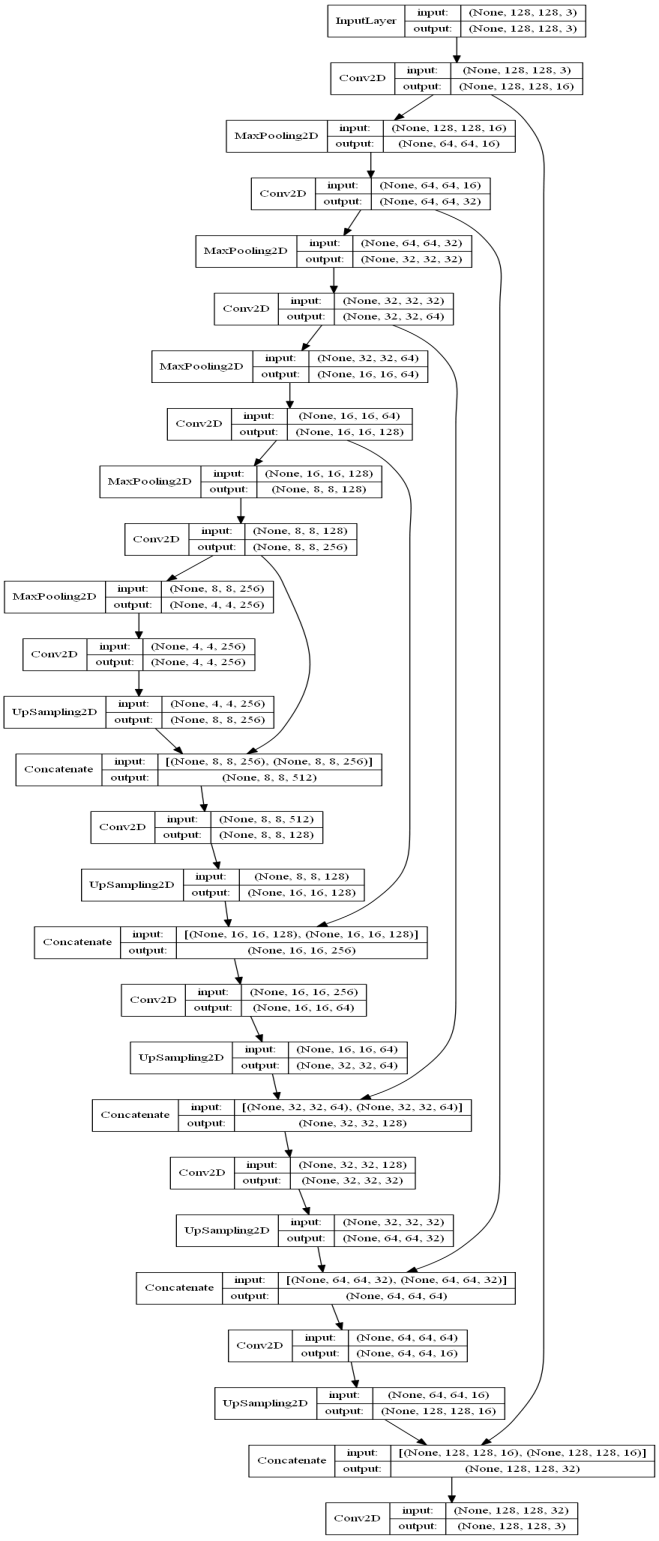

Figure 2: Architecture of our U-Net model

2.4 Activation functions

In our experiments, we included a comparison between two activation functions used in our networks. One is ReLU, the most popular activation functions, and the other is Swish [5], an activation function

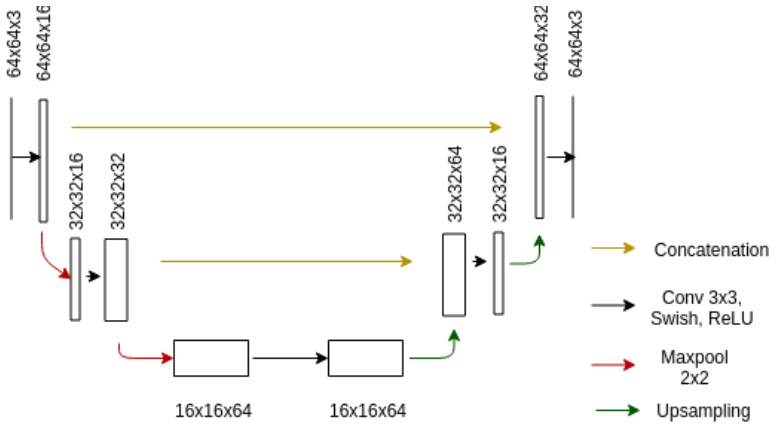

Figure 3: Our U-Net Architecture inside of the GAN

that aims to solve several problems that ReLU has. The Swish function is defined as follows:

$$f(x) = x \times sigmoid(x)$$

2.5 Training procedure

All networks were trained until convergence. We used 80% of the data for training, and the remaining 20% as validation data. For all of the networks, the Adam optimiser was used.

3 Results

Network	Loss function	Activation function	PSNR
U-Net	L1	ReLU	6.178364257571591
U-Net	L1	Swish	6.0931783550568515
U-Net	L1 + DSSIM	ReLU	6.177953085530094
U-Net	L1 + DSSIM	Swish	6.064955358215332
GAN	Adversarial	ReLU	6.152808576426424
GAN	Adversarial	Swish	6.069299951023391

Table 1: Results of different methods compared

Figure 4 shows a comparison between the results of all of the networks when tested on the validation data. Table 1 shows the average Peak Signal-to-Noise Ratio value for each of these methods on the validation data. By looking at the visual results, we think the U-Net performs a lot better than the GAN. Despite the adversarial loss, we think the GAN images look very artificial. When comparing the different methods we used for the U-Net, we think that by using the DSSIM loss, the images appear less realistic. While the method should make the image sharper, it seems like the inpainted parts become less smooth using this method. We don't see much difference between using ReLU activation and Swish activation. We think the results from the U-Net look quite good overall, and are very realistic in general. Only when the input image is unusual, for example when the person is wearing glasses, the output becomes unsharp.
Figure 5 shows some of the results found in [10], the main inspiration for our work. One important thing to note is that the images used in our method are slightly different: our autoencoder produces images with a resolution of 128x128 pixels, whereas our GAN and their work both use smaller, 64x64 images. Furthermore, our centred holes are slightly smaller, though the images used in [10] use images that are zoomed in more on the face, whereas ours show more of the body and background. While the code of [10] is public, they do not explain exactly how they process, crop and cut the holes, so we could not replicate this, and the results are not completely comparable.

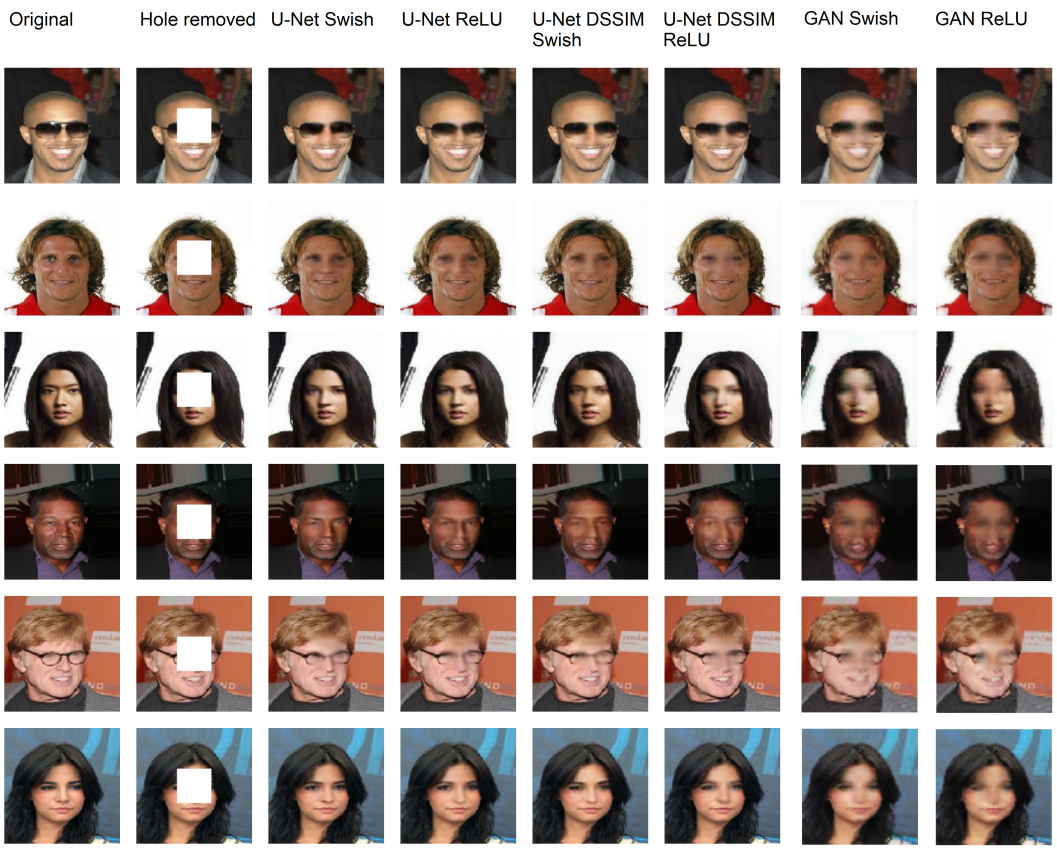

Figure 4: Results of our experiments

For a qualitative and subjective comparison, we feel our best results generally look as realistic, if not more realistic than the results of [10]. Quantitatively, however, the PSNR values they found in their own paper (average of 19.4) suggests their method still vastly outperforms ours. In fact, we found the PSNR values in our results to be surprisingly low, something we will get more into in the discussion section. Again, due to the lack of clarity about the processing in [10], we could not verify that this result is completely comparable.

Figure 5: Some results of [10]

4 Discussion

From our results, we conclude that our U-Net autoencoder outperforms our GAN network not only in results but in training time as well. We expected the results from the GAN to be better, as the GAN architecture should learn to produce results that are hard to distinguish from real. To us, it is pretty clear that the images generated are not real, unlike the images from the U-Net. It is difficult to train a GAN because it is difficult to guarantee that the learning happens in a stable way, so maybe we can improve the result by changing the parameters of the GAN. Additionally, we could vary the loss function that is used by the GAN. Right now, only adversarial loss is used. One thing that could be tried is to add some MSE loss as well in the total loss function. This could maybe help the GAN to perform more like the U-Net, with the adversarial loss improving the result when compared to the results of the U-Net on its own.
Right now, we have achieved realistic results using only the U-Net. An advantage of our network is that it also produces good results at a higher resolution of 128x128, compared to previous work that was limited to 64x64. In future work, the resolution of the images could be increased even more.
The results are still not always good though. When images have unusual attributes, the network becomes less certain and either makes mistakes, or produces blurry results.

For the U-Net, we compared two losses: the MSE, and the DSSIM combined with MSE. We were hoping that the addition of DSSIM loss would improve the results, but we think the results look worse than using only the MSE. The PSNR value is also slightly lower when using the DSSIM loss. When we look at the results, the images generated by a model trained with DSSIM loss seem a bit more pixelated in the hole that was filled in. It seems like the DSSIM loss enforces relatively high contrast between pixels, but this does not seem to prevent the uncertainty the network has about the facial attributes.

Additionally, we compared Swish and ReLU activation functions. The networks trained with ReLU give a higher PSNR value, but we think this is not reflected in the visual results. If anything, we think the results using Swish are a bit more realistic.

One thing we noticed is the surprisingly low PSNR values of our results. We suspect this is because our networks generates entire images instead of just the removed region. We believe this creates small differences in each pixel of the image that are undetectable by the human eye but add up to a large distortion in the completed image that the PSNR calculation detects, not unlike how adversarial attacks on deep networks work. However, as this affects both of our network architectures, PSNR could still be an appropriate measure of comparison between our own results. One other aspect to note about PSNR is that it is a comparative measure between our generated images and the ground truth. The goal of image inpainting, however, is not explicitly to replicate the ground truth, but to restore the image in such a way that it looks realistic, which cannot be measured using PSNR. If the goal was to purely recreate the ground truth, we would be better off by increasing the size of the network to such a size at which it would function as a memory that simply stores all the ground truth images and finds the one that matches the corrupted input image the most.

A way in which the results might be improved is to add more layers to the U-Net network. Especially in the generator we used for the GAN, the amount of layers is relatively low, so that the training did not take too long. By adding more layers, the network will likely be better at restoring fine details.

One more future addition we believe is possible is looking at even larger images than the ones we use like the ”in-the-wild” images provided in the CelebA dataset. An additional challenge using this section of the dataset is that the images are not cropped and aligned, meaning they are varied in size. The network would need to be altered to allow the input of images such as these. As this task would be significantly harder, we would likely need a different network to accomplish this. Another possible area for future research is the use of our networks on different datasets.

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[2] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[3] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[4] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[5] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Swish: a self-gated activation function. *arXiv preprint arXiv:1710.05941*, 7, 2017.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[7] Jianhong Shen and Tony F Chan. Mathematical models for local nontexture inpaintings. *SIAM Journal on Applied Mathematics*, 62(3):1019–1043, 2002.

[8] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[9] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[10] Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5485–5493, 2017.