DEEP LEARNING FOR VISUAL INPAINTING

December 11, 2020

ABSTRACT

In this paper, we used a Dense Multi-scale Fusion Network (DMFN) to recreate images when censored by a eye-blocking censor bar. We hypothesized that adjusting the influence of feature matching could be altered to produce better results with our goal of recreating the eyes. We found that feature matching creates the most perceptually realistic images when the η-value is 5, based on the LPIPS metric. Furthermore, we found that removing feature matching generates images with a relatively low LPIPS score, however the produced features of the eyes are very similar in all images.

1 Introduction

Recreation of an image missing a region can be achieved with image inpainting techniques. Combining this with the use of generative adverserial networks (GANs), feature matching and specifically a dense multi-scale fusion network it has been proven to produce some of the perceptually best reproduced images. However, using this method, the use of feature matching and different loss calculations could potentially be adjusted to produce even better results. Especially, when focusing on a specific facial feature to recreate. Therefore we, in this paper, seek to investigate how the influence of feature matching affects facial inpainting when focusing on recreating the eyes behind censor bars popularly known to hide the identity of individuals. To obtain our code of the project follow the link at the bottom.[1]

2 Related Work

2.1 Model

Image inpainting has shown promising results with the introduction of GANs. Knowing this, multiple solutions were considered. Here Zheng et al. [2020] proposes a dense multi-scale fusion network (DMFN) with self-guided loss, geometrical alignment constraint, dense multi-scale fusion blocks and discriminator feature matching, which highly improves the quality of the produced images. Furthermore, the solution of Zheng et al. [2020] outperforms other state of the art methods. Specifically, the DMFN produces better results on all metrics (PSNR, SSIM, LPIPS) when recreating a large center regular mask on the CelebA-HQ dataset. This is also the case on the Paris Street View Dataset. Considering this, the solution of Zheng et al. [2020] perfectly suits the needs of our project showing impressive results on facial inpainting.

2.2 Feature Matching

Salimans et al. [2016] introduced "feature matching". A technique addressing the instability of GANs encouraging a convergence of the gradient descent to reach equilibrium for both the generator and discriminator when training GANs. Furthermore, feature matching prevents overtraining on the discriminator by specifically training the generator to match values of features on intermediate layers of the discriminator, instead of maximizing the output of the discriminator.

Using feature matching on our network seems sensible and adjusting its influence to find the level that provides the most stable and effective training of our network in relative to the results produced.

2.3 Metrics

Recreating a picture with facial inpainting requires a method of measuring the results. Here Zhang et al. [2018] proposes a relatively new metric named **Learned Perceptual Image Patch Similarity** (LPIPS). This metric is more similar to the judgment of human perceptional similarity than the traditionally used PSNR and SSIM metrics used to asses the perceptual similarity between two images. Striving to make our results in this project as realistic to human perception as possible. We utilize LPIPS as a comparison metric to the effect of different changes to our network and how they correspond to the visual integrity.

2.4 Relativistic Average GAN

In the paper by Jolicoeur-Martineau Jolicoeur-Martineau [2018], an alternative to standard generative adversarial networks (SGAN) is introduced and is to referred to as Relativistic GAN (RGAN) and Relativistic Average GAN (RaGAN). The motivation for the new types of GANs, lies in the fact that SGANs only utilizes the discriminator to calculate the probability of the input data being real images, and the generator attempts to increase the probability of fake images being real images.
The idea of the RGAN is to use a relativistic discriminator that not only estimates whether real data is real, but also calculates whether real data is more realistic than fake data. RaGANs are similar, but here a variant is introduced, that calculates the probability of the real data being more realistic than the fake data on average.
Jolicoeur-Martineau is able to conclude that RGAN and RaGANs are able to produce higher quality data, than their non-relativistic counterparts. It is also concluded that RaGANs can generate plausible high-resolution images from small samples, that GANs are not able to do, making RaGANs an excellent strategy for generating images.

3 Datasets

In this paper we utilize the CelebA dataset [Ziwei et al., 2015] which consists of more than 200.000 facial images of different celebrities. The dataset features both high- (1024x1024) and low-quality(178x218) images and has been widely used in the deep learning community for training on facial features. Although the original dataset features a large amount of images, we have limited the training dataset to 5000 of each quality. This becomes a necessity because our training is constrained by a maximum runtime on Google Colab of maximum 12 hours. Before applying the images in training the images are modified to fit our architecture which requires shapes of size 256x256. This includes down-scaling the HQ dataset and up-scaling the low-quality dataset before training. This re-scaling is necessary, since the network source code is not able to handle 1024x1024 images, and the low-quality image size makes testing difficult, since image sizes are hard coded in some places, and not in others.
It should be noted that the CelebA dataset is also used by Zheng et al. [2020] in their paper, when testing/experimenting with their network solution.

4 Architecture

The network by Zheng et al. [2020] that is used in this report is a GAN consisting of a generator and a discriminator. The generator's job is to produce a plausible, but incorrect, image from an input. This input is a ground truth image and a random "mask" consisting of 1s and 0s where 1s are unknown pixels and 0 are known pixels. The ground truth image is component-wise multiplied with the mask, producing the desired input. The output of the generator is an image prediction component-wise multiplied with the input mask, that is added to the original ground truth image. The discriminator is then fed with the original ground truth image along with the generated fake image output and will try to guess which image is the falsely generated one and which is the ground truth image.

4.1 Generator

Specifically the generator is a dense multi-scale fusion network (DMFN) consisting of convolutional layers and makes use of Dense multi-scale fusion blocks (DMFB). The DMFBs processes features divided in four branches with different convolution dilation factors. This solution extracts better multi scale features compared with general dilated convolution and ultimately creates more photo-realistic images.

4.2 Discriminator

The discriminator is improved using RaGAN by Jolicoeur-Martineau [2018]. RGAN differ from standard SGANs in the sense that SGANs try to determine how real an input is, as opposed to RGANs that try to calculate a difference between real and fake inputs, thus calculating a probability that the real input is more realistic than the fake input. When RGANs compute the probabilities of the data being fake or real, the values will vary a lot in different samples, RaGAN calculates the average probability that the real data is more realistic than the fake data. The discriminator also utilizes the use of both a local and a global branch. The global branch focuses on the entire "global" image, and the local branch focuses on the masked region.

4.3 Generator Loss

The generator loss is the sum of multiple individual loss calculations with different responsibilities:
Self-guided regression loss is a loss calculation created by Zheng et al. [2020] and is used to focus on semantics and uncertainties in the images. It corrects semantic errors by weighing features found in a discrepancy map from the VGG19 network [Simonyan and Zisserman, 2015].
Geometrical alignment constraint is used, to combat the misalignment issues that is present when only pixel-based evaluation is performed. This loss function helps to measure the distance between high-level features from the ground truth image and predicted image. This function aims to have little distance between both image centers, and aligns the predicted image with the ground truth image.
The **Feature matching loss** is calculated with two different functions. The first one uses the VGG19 model and compares the activation maps found in the intermediate layers of this model. The other function is a discriminator feature matching loss calculation that compares the ground truth image with the output image. The hidden layers of the discriminator are trainable, as opposed to VGG19 trained on the ImageNet dataset[Stanford-Vision-Lab, 2016], meaning that this function can extract features from the given training set that might not be found in VGG19.
Adversarial Loss is used to improve the visual quality of the results. The loss is calculated using a Relativistic Average Discriminator Jolicoeur-Martineau [2018], and takes a real or fake image pair as input.

All of the aforementioned loss calculations are added, along with a Mean Absolute Error (MAE or L1) loss, to produce the final generator loss. [Zheng et al., 2020]. The five loss calculations are each multiplied with a constant, in order to balance and control the output of the generator.

4.4 Discriminator Loss

The discriminator loss is found using the RaGAN method, and takes a real or fake image pair as input. The loss is calculated by finding the binary cross entropy loss of real image subtracted by the fake image, and 1. This result is then added to a flipped binary cross entropy loss of fake minus real, and 0. This result is divided by 2, and is the final discriminator loss.

5 Results & Qualitative evaluation

5.1 Early Testing

As it became possible to run the network source code, it was time to "look under the hood" and modify the existing network. This included modifying directory paths to fit the ones available to the Google Colab Notebook so that training could proceed. Modifications to various aspects of the network was done by providing different input arguments. These arguments made it possible to modify the batch size, learning rate, number of epochs, type of cover mask, etc. In the beginning stages of the process, most of the time was spent on testing with small datasets, to figure out how the network worked. Here, it became apparent that the highest batch size possible for this network on Google Colab, was a size of 8, since the available GPU (NVIDIA Tesla K80, 12GB) had no excess memory to run a batch size of 16.

5.1.1 Proper Mask

Testing a trained network was not difficult and also had helpful input parameters to adjust the testing. Testing is done by calling the test.py-file, and pass a list of filenames and a location of the images as input. The testing-file then adds a random mask on the input image, "removes" it again afterwards and creates a new folder with the input- and output images (input being the masked original images). The goal of this paper, is to re-paint eyes where they had been blocked out or censored and the only available masking type, is a randomly placed mask. This meant that the testing file would

have to be modified, in order to put a mask on the eyes, instead of randomly placing it on the image. This was done with OpenCV [Bradski, 2000].

5.1.2 Investigating Generator Loss

After adjusting different parameters (e.g. number of epochs, batch size, dataset size and learning rate), we instead started to focus on the generator loss. This loss is calculated with different parameters and constants, so the first test was to try and replicate the results from the paper by Zheng et al. [2020] (Table IV). These results were created by removing one of the individual loss calculations from a trained network, and compare the results with a network that was trained with all loss calculations. We tried to replicate this experiment:
The experiment was performed by training all networks equally, but setting one of the generator loss parameters to 0 (meaning that the parameter corresponding to the loss calculation would be canceled out). There would also be a "base line" that would include all of the loss calculations. The experience from training the network with varying batch sizes, learning rates, etc., showed us that the optimal way to run the network, is close to the way it is done in the paper. Here, the learning rate is set to *2e-4*, the batch size is 8 (because of Google Colab) and the generator loss parameters are $\lambda = 25$, $\eta = 5$, $\mu = 0.003$ and $\gamma = 1$ (the parameters are of course 0, when that loss calculation is omitted). 5000 images was used from the CelebA dataset with an epoch number of 15. It should also be noted that the network is trained with the Adam Optimizer having $\beta_1 = 0.5$ and $\beta_2 = 0.9$.
The results from this test closely matched the ones found in the paper by Zheng et al. [2020], and we chose to specifically focus on the discriminator feature matching loss calculation, since the output images had a clear visible difference in the heightened features (or lack thereof).

5.2 Testing Hypothesis

The testing of the discriminator feature matching loss was performed by once again using the same training parameters as mentioned earlier and only modifying the η-parameter. The modifications would be to use the suggested η-value of 5 as a baseline, since this was the chosen value in the paper, and then train the network twice with one having the η-value increased and one decreased.
The hypothesis is that an η-value that is too low would produce images that would not extract enough important features from the dataset. Conversely, an η-value that was too high would produce images that could overestimate what should and should not be an important feature.
Both cases (too low or high value) would produce poorer images, and this would be investigated using the proper metrics. To do this, LPIPS is chosen due to to its ability to compare images almost like humans would. LPIPS is also used in the paper by Zheng et al. [2020], to compare results in different experiments.

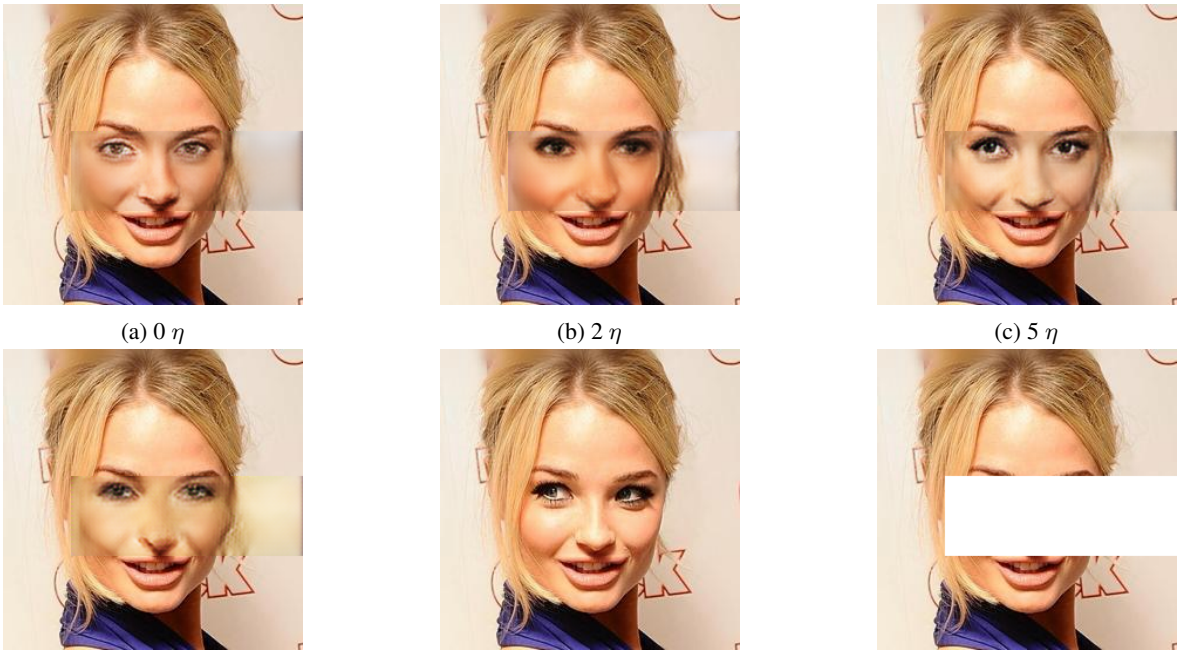
5.3 Results

The results were achieved training the DFMN network on Google Colab with 5000 pictures of the CelebA dataset. The training lasted 15 epochs with a batch size of 8 and a learning rate of $2e-4$. Testing results are the average LPIPS of 30 pictures also from the CelebA dataset.

η-value	**LPIPS $\downarrow$**
0	.0517
2	.0559
5	.0513
8	.0587

Table 1: Table of LPIPS score based on η-value

Table 1 shows that the baseline η-value of 5 produces the best results. Neither an increase or decrease of the η-value indicates better results. Figure 1 shows that the image quality is very affected by the η-value as it changes drastically on small adjustments. The η-value of 5 produces the best LPIPS score of **0.0513**.



(a) 0 η (b) 2 η (c) 5 η

(d) 8 η (e) Ground Truth (f) Input Image

Figure 1: Image samples of the results with varying η-values

6 Discussion

Interpreting the results in Table 1 it seems that the network responds very strongly to a η-value change. It is possible that small adjustments could make the network unstable as Salimans et al. [2016] claims that feature matching addresses instability in GANs. Therefore small adjustments could mean that equilibrium of the generator and discriminator is never reached. Taking a look at the results of Figure 1, image 1a and 1d could indicate that the discriminator is under trained as it seems the generator is guided in a direction of producing blurry 1b and grainy 1d images. According to our judgement image 1c with a setting of 5η results in the most similar picture to the ground truth. Furthermore, image 1c seems to reproduce the features of eyes more precisely which accommodates our goal of mainly recreating the facial structure around the eyes. Taking image 1a into consideration which was created using an η-value of 0. This setting shows promising results at first sight and also scores a good LPIPS score seen in Table 1. However, when looking at Figure 2 it becomes clear that all of the eyes look identical in all the tested pictures. Even thought the LPIPS score of 0η is promising, the features of the eyes are poorly trained. This is most likely caused by the fact that the loss calculation that is canceled out by setting η to 0, is responsible for extracting important features from the training dataset. The eyes that are created in Figure 1a and Figure 2 are similar due to the fact that the eyes are created based on the pre-trained VGG-model.

(a) (b) (c)

Figure 2: Images with η-value of 0

6.1 Limitations

6.1.1 Google Colab

The network that we have modified and worked with has been run using Google Colab. This has been a great tool, since it provides high performance hardware, and integrates easily with Google Drive for storage. In short, Google Colab has many advantages for smaller project, but the same platform also proposes limitations:
Namely Google Colab is limited in the amount of Runtime allocated to one session with a maximum duration of 12 hours. 12 hours is sufficient time to train for simpler networks with smaller datasets and as a result of this we can produce images of non-novel quality. The time barrier also limits our ability to train for extended amounts of time and because of this, we have been forced to "under-train" some networks, to ensure that the training will complete.

It should be noted that the limitations from Google Colab have been small, and using a more powerful and expensive solution could possibly have produced better results. With all in mind Google Colab still remains a great tool for experimenting with deep learning tasks.

6.1.2 Our Model

The research done for a model to complete our task, was done by searching for Facial Inpainting on Papers With Code [2]
There were multiple papers to choose, and our model was not our first choice. The first model that was investigated was called SymmFCNet Xiaoming et al. [2018] and scored better PSNR results than our DMFN model. SymmFCNet was not used, due to the fact that the source code was not written in Python, and their suggested PyTorch implementation was not available at the time. That being said, the DMFN model still produces great LPIPS results and is published more than a year later than SymmFCNet. The source code used for our model had hard coded options in some places, meaning that we had to edit the code, in order to simply run it out of the box. The path for the VGG19-network and dataset for testing, had to be written by hand. The image shape could be edited as an input parameter, but was adressed at several different points in the code, and made it very difficult to change.
In spite of the difficulties of the code, it also gave us a great understanding of the network, and made other important changes much easier.

6.1.3 Eye Blocking

The goal of this paper is to remove eye blocking, and this required the ability to block eyes in the first place. This was done quite simply using OpenCV but was difficult to integrate in the chosen network. It was achieved by locating the file where the random mask was being generated, and somewhat copying the function. The mask-points were then replaced by the points generated from the eye-location, but this had three problems:
1. The OpenCV-library is good at detecting eyes but not great. It has problems detecting closed eyes or "over"-detecting eyes. The over-detection was somewhat solved by rejecting any eye below 60% of the face height (horizontally split), and just accepting the first detected eye. The box is drawn based on the location of the first detected eye, and size of the face. The problem of not detecting eyes, was solved by running the test images through the "eye-checker" locally, and sorting out images where the eyes would cause an error.
2. Due to the image being resized in the code, the eye location would be misplaced. This was poorly solved by adding a simple constant to the eye-blocking box's dimensions.
3. The eyes are only blocked when testing the code. When training, the old mask-generator is still used. This is due to the fact that the image in question was difficult to extract under training, and an image is needed to locate the eye-position.
That being said, the eye-blocking does sometimes cover a lot more than just the eyes, and training the network to generate more than eyes is an advantage.

6.2 Conclusion

In this paper, we trained the DFMN proposed by Zheng et al. [2020] to remove censor bars from the eyes, using OpenCV to locate the eyes and modifying the existing code of the model, to thereafter recreate the image. We adjusted multiple settings to create the best results and found that feature matching particularly affected the appearance of the produced images. We hypothesized that adjusting the influence of feature matching could be altered to produce better results with our goal of recreating the eyes. Our results show that feature matching creates the most perceptually realistic images when the η-value is 5, based on the LPIPS metric. Furthermore, we found that removing feature matching generates images with a relatively low LPIPS score, however the produced features of the eyes are very similar in all images.

[2] `https://paperswithcode.com/`

References

[1] G Bradski. The opencv library. *Dr. Dobb's Journal of Software Tools*, 2000.

[2] Alexia Jolicoeur-Martineau. The relativistic discriminator: a key element missing from standard gan. *arXiv preprint arXiv:1807.00734*, 2018.

[3] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29:2234–2242, 2016.

[4] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[5] Stanford-Vision-Lab. Imagenet dataset, 2016.

[6] Li Xiaoming, Liu Ming, Zhu Jieru, Zuo Wangmeng, Wang Meng, Hu Guosheng, and Zhang Lei. Learning symmetry consistent deep cnns for face completion. *arXiv preprint arXiv:1812.07741*, 2018.

[7] Richard Zhang, Phillip Isola, Alexei Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[8] Hui Zheng, Li Jie, Wang Xiumei, and Gao Xinbo. Image fine-grained inpainting. *arXiv preprint arXiv:2002.02609*, 2020.

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