Generative Image Inpainting

Darko Filipovski
Faculty of Computer Science and Engineering, University "Ss. Cyril and Methodius", Skopje
darko.filipovski@students.finki.ukim.mk

Abstract

Introduction of deep learning approaches for solving the problem of image inpainting have shown promising results. These techniques are able to produce detailed structures, but have some underlying problems causing blurriness and artifacts. This paper looks into a proposed solution for semantic image inpainting and implements an improvement proposed by the authors. The solution involves usage of a newer generative model and is pitted against the initial implementation. Both approaches have been evaluated on images from the FFHQ dataset, corrupted in a way to uncover the shortcomings of the models. Conducted experiments demonstrate that the implemented changes produce finer details, and unveil model's inherent inability to fill small scattered missing regions. Source code is available at: https://github.com/filipovskid/image-inpainting

1. Introduction

Image inpainting is a task of filling in missing regions of an image. It is an important computer vision problem often used for object removal, image reconstruction and manipulation, with applications in photo editing and computational photography. Humans have the ability to recognize even the slightest visual inconsistencies, thus, the challenge of image inpaining is to fill in the missing regions with perceptually and semantically plausible pixels that create a realistic image when combined with the remaining of the pixels.

Early works propose inpainting the missing regions based on the available information within the image background. These approaches often use differential operators for propagating background data in a diffusion-based approach to solving the problem [3, 5]. Additionally, using background patches that match the remainder of the image, either from a collection of images [7] or data within the image [2, 4] gives rather satisfying results, especially in cases where there is a pattern or texture. However, these methods assume that appropriate information can be found within the image and cannot produce complex structures that match the semantics.

Advances in deep learning approaches have expanded on the ability to capture the context of an image as well as generate samples from a given distribution, which can be seen as prerequisites for image inpainting. Convolutional neural networks (CNNs) introduced the ability to capture image features [13, 19], which often were used in classification and image recognition problems. This, combined with generative adversarial networks (GAN) [6, 18, 12] encouraged representing the image inpainting problem as a conditional image generation. Taking into account the context of the remainder of an image, when training adversarial networks, attains both the semantics and the structure of the inpainted image.

This paper focuses on the implementation of semantic image inpainting with deep generative models proposed by [21]. Their suggestion of integrating more capable GANs is also taken in consideration by using StyleGAN's generator and discriminator [12]. The goal of this approach is to find an encoding of the corrupted image that is closest to the image in the latent space. This is achieved by defining a weighed context loss, obtained from the corrupted image, which acts as a condition, and a prior loss that penalizes unrealistic results. However, even though the proposed implementation of these losses works in certain cases, there are instances which give unsatisfactory results. Consequently, I have covered the results obtained from inpainting images corrupted in different ways, pointed out the potential reasons for the bad results and proposed ways to alleviate the detected problems. Example results are shown in Figure 1.

2. Related Work

Methods used for image inpainting take advantage of both deep learning and traditional approaches. Most successful approaches, including the method explored in this paper, are based on or make use of deep generative models to achieve superior results, thus, it is important to understand the basic principles underlying GANs.

2.1. Generative Adversarial Networks

Generative Adversarial Networks introduce the adversarial approach for solving the problem of generative models.

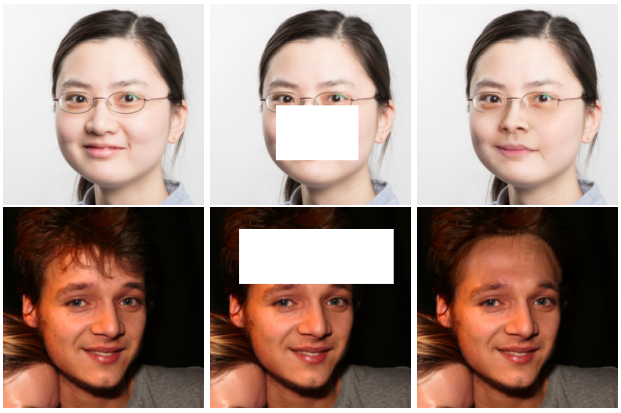

Figure 1: (Left) Target images. (Center) Corrupted images with missing regions. The missing regions are shown in white. (Right) Inpainted result using StyleGAN within the method proposed in [21].

Generative models take the training data, sampled from a distribution p_{data}, and try to learn an estimate p_{model} of that distribution. The adversarial approach sets the generative model, that learns the data distribution, against a discriminative model that determines whether a sample came from the model distribution or the data distribution. This new approach to learning is introduced in [6] and covered in this section.

The GAN framework trains two models simultaneously, a generative and a discriminative model. The generator G is a differentiable function which maps an input variable z, sampled from a prior distribution p_z, to $G(z;\theta_g)$ from p_{model}. In contrast, the discriminator D is a simple classifier which classifies samples as fake if they are coming from the generator's p_{model} distribution, and as real if they are sampled from the real training data p_{data}. More specifically, $D(x)$ is a scalar value which represents the probability that x comes from p_{data} rather than p_{model}. These models are typically represented by multilayer perceptrons with parameters θ_g, for the generator, and θ_d for the discriminator.

When it comes to the training process, we train D to maximize the probability of assigning the correct label to both the training and samples generated by G, while simultaneously training G to maximize $\log(1 - D(G(z)))$. This means that we want to maximize the probability of D making a mistake or fooling D into thinking that the generated sample comes from the training examples. The loss function which has to be optimized during this process is:

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

where x is a sample from the p_{data} distribution, whereas z is some noise input from p_z.

Figure 2: Images generated by DCGAN and StyleGAN. First row: samples from DCGAN. Second row: samples from StyleGAN

Early GAN implementations have unstable training and results which are far from the target distribution. To overcome these problems Wasserstein-GAN (WGAN) [1] minimizes Earth Mover (EM) distance, resulting in a more stable optimization problem which does not require a careful balance in training of the discriminator and the generator, and expands on the network architecture possibilities. To produce fine detailed images, novel generative models [12, 11] were introduced, of which StyleGAN was incorporated as a replacement for DCGAN. The difference in details can be noticed on Figure 2.

2.2. Image Inpainting

Approaches based on deep learning, preceding semantic inpainting, gave the inspiration for using GANs and presented the limitations of the models at the time. Context Encoders [16] take the encoder-decoder approach. The encoder captures the features of a corrupted image, while the decoder produces the missing region. This model is trained using ℓ_2 as a reconstruction loss and generative adversarial loss as the objective function. This method depends on the shape of the masks during training, so using arbitrary masks produce more artifacts.

Methods capturing fine details of an image, by focusing on the regions, within a corrupted image which carry most context, overcome the problems introduced in semantic inpainting. Contextual Attention [22] takes a two stage approach. The first stage gives a rough estimate of the missing region, using a simple dilated convolutional network. The second second stage makes use of contextual attention which tries to borrow features from background patches, using them as convolutional filters to process the generated patches. This, together with additional convolutional network is fed into a decoder. The whole network is trained using reconstruction loss and two Wasserstein GAN losses [1]. [15], with their EdgeConnect model, gives an

excellent example on how decoupling can intuitively lead to fine details. Here, image inpainting is presented as a two-stage process: edge generation and image completion. Both stages take an adversarial approach. The generators follow an architecture similar to [10], commonly used for style transfer and super-resolution. For discriminators, an architecture following PatchGAN [9] is employed. These new methods provide the means to capture complex information from images, thus produce superior results.

3. Semantic Inpainting by Constrained Image Generation

The method for image inpainting, proposed by [21], takes advantage of the generator G and the discriminator D trained on uncorrupted data. However, as we have seen previously, GANs produce entirely new and unrelated images which in our case would not be very useful. In relation to this problem, the method introduced in the paper tries to find the image closest to the corrupted one by searching trough the latent space.

The hardest problem that [21] is trying to solve is how to find an encoding $\hat{z}$ close enough to the input image. This problem is approached by taking a point z, drawn from p_z, and generating an image $G(z)$. The newly generated image needs to be "close" to the corrupted image y, so we need to "walk" over the latent manifold by updating z using some kind of loss function which captures the context of y, as shown on Figure 3.

The problem of finding $\hat{z}$ is formulated as an optimization problem where the aim is to minimize loss with respect to z. This is formulated as:

$$\hat{z} = \arg\min_{\mathbf{z}} \mathcal{L}_c(z|y, M) + \mathcal{L}_p(z) \tag{2}$$

where $\mathcal{L}_c$ is the context loss of z given the corrupted image y and a mask M, whereas, $\mathcal{L}_p$ denotes the prior loss.

3.1. Context Loss

The author takes an intuitive approach to calculating the context loss by taking advantage of "useful" pixels from the remaining available data of y. The importance of an uncorrupted pixel is taken to be positively correlated with the number of corrupted pixels surrounding it. This reasoning indicates that pixels closer to the hole play an important role in image inpainting, thus a difference between them and the original pixels in y will be punished more compared to the remaining pixels. This importance is captured by the importance weighting term **W**.

$$W_i = \begin{cases} \sum_{j \in N(i)} \frac{(1-M_j)}{|N(i)|} & if M_i \neq 0 \\ 0 & if M_i = 0 \end{cases} \tag{3}$$

where W_i denotes the importance weight of a pixel at location i and $N(i)$ refers to the set of neighbors of pixel i in a local window of some size. This importance weighting term, **W**, is illustrated on Figure 4.

Together, the context loss is defined as a weighted l_1-norm difference between the generated image and the uncorrupted portion of the input image, as shown in Eq. 4.

$$\mathcal{L}_c(z|y, M) = \|\mathbf{W} * (G(z) - y)\| \tag{4}$$

where * denotes element-wise multiplication.

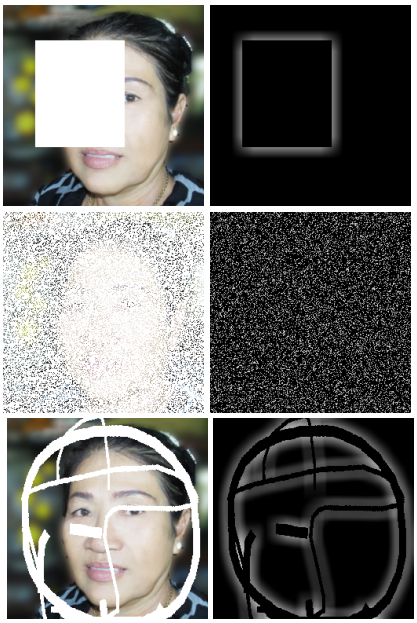

Figure 4: Importance weighting matrix **W** derived from the corrupted image. (Left) Corrupted image. (Right) Importance weights. Grayscale values closer to 1 (white) indicate higher importance, while values closer to 0 (black) indicate lower importance.

3.2. Prior Loss

Context loss takes into consideration individual pixel values with the intention to bring the resulting image pixels, which correspond to the uncorrupted pixels in y, closer to the intended value. However, the generated image may not look realistic at all, even though the pixel values are close to the original ones, and as a result an introduction of prior loss is needed. The prior loss $\mathcal{L}_p$ makes sure that the resulting image comes from the p_{data} distribution and as a result penalizes unrealistic images. Not surprisingly, $\mathcal{L}_p$ is taken to be the GAN's loss for training the discriminator D, which takes advantage of the discriminator's ability to differentiate between generated and real images, therefore the prior loss takes the form:

$$\mathcal{L}_p = \lambda \log\left(1 - D(G(z))\right) \tag{5}$$

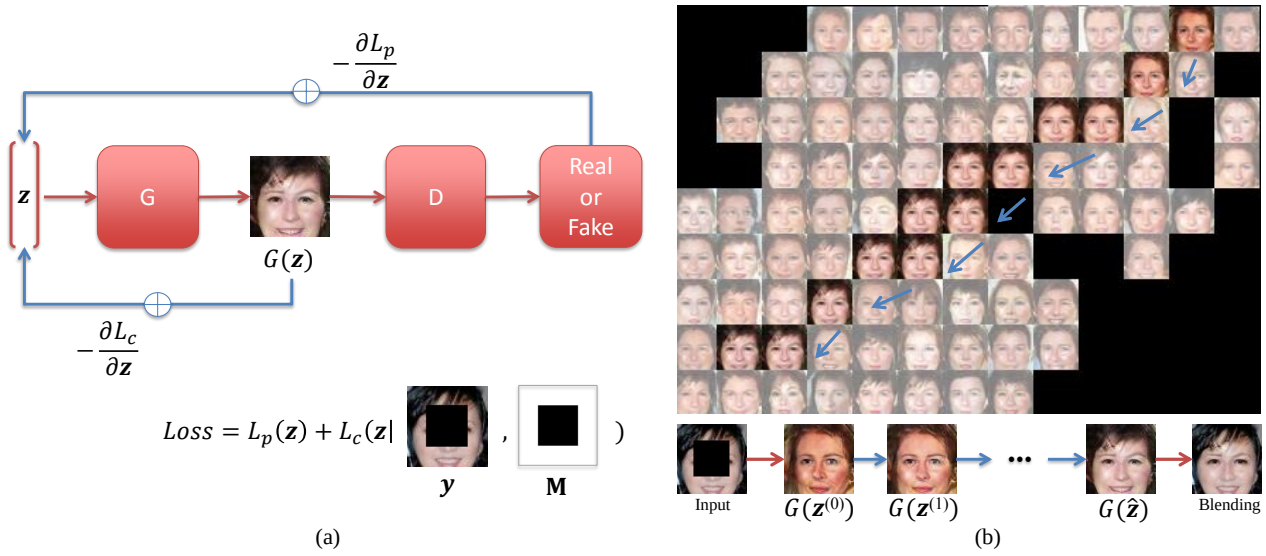

Figure 3: a) The process of updating z using the loss function and a trained GAN model. b) "Walking" over the latent manifold and updating z to a final solution $\hat{z}$. [21]

where λ balances the two losses and in this project takes a value of $\lambda = 0.003$.

3.3. Image Completion

Final steps for obtaining an inpainted image can be formulated as an optimization problem. Initially, z is drawn from a uniform distribution and is passed through the generator G. The loss is applied on the obtained result and the error is propagated backwards. These steps of generating results and propagating the error backwards are repeated for several iterations, which leads to obtaining $\hat{z}$. Generating $G(\hat{z})$ gives a result which is close to the uncorrupted portion of y, as defined by the loss, from which the previously corrupted pixels can be extracted and replaced within y. On the resulting image, Poisson blending [17] is applied to match the pixel intensities.

4. Experiments

4.1. Dataset and Image Masks

Both training of the GANs and evalution of the inpainting process are done over the Flickr-Faces-HQ (FFHQ) dataset [12]. The dataset consists of 70,000 high-quality human face images at a resolution of 1024x1024. It provides a large variation in terms of age, ethnicity and image background. Images were normalized, with no further augmentation due to compute power limitations.

For the experiments, I used three types of masks: regular, noise and irregular. Regular masks are random rectangles that are centered at a random location within the image and cover 10% to 40% of the image pixels. Noise masks are sampled from "continuous uniform" distribution over the half-open interval [0.0, 1.0) and cover 80% of the image. Irregular masks are drawn from the Quick Draw Irregular Mask dataset [8], which is a combination of 50 million strokes drawn by human hand. These masks are chosen to give a more realistic overview of different usages.

4.2. Training setup and Inpainting

Generative models are implemented in PyTorch. DCGAN model is trained on 64x64 images with a batch size of 256, for improved stability. Optimization is done using the Adam optimizer, with a learning rate of $\alpha = 0.0002$ and coefficients $\beta_1 = 0.5$, $\beta_2 = 0.999$. For the StyleGAN model, a pre-implemented and pre-trained model is used. The model was trained for 140,000 iterations over the FFHQ dataset and generates samples with a resolution of 256x256 pixels.

The inpainting model is implemented in PyTorch and has different configuration depending on which generative model is used. DCGAN based implementation samples a 100 dimensional vector from a standard normal distribution. A window size of 7 is used for generating the importance weighting term $\mathbf{W}$. Obtaining the encoding $\hat{z}$ is done by backpropagation of the loss for 1500 iterations. StyleGAN implementation uses 512 dimensional vector sampled from a standard distribution. Context loss is computed using a weighting term which captures a window of size 28. The loss is backpropagated for 3000 iterations. Differences in the window size for the weighting term are correlated with the size of the image. StyleGAN generates samples which are 256x256 pixels, while DCGAN generates samples with

a resolution of 64x64, thus, same features are captured by the importance term with StyleGAN's benefit of capturing them in a higher resolution. Additionally, StyleGAN based inpainting model needs more iterations because changes on the input vector cause only slight mutation on the output. This behaviour is expected because the model's architecture is built to support such manipulations.

Training and evaluation processes have been executed on the GPUs provided by the Google Colab platform.

5. Results

The model proposed by [21], implemented using both DCGAN and StyleGAN generative models, is evaluated on the FFHQ dataset. Results aim to give qualitative and quantitative overview of the model, including some underlying problems.

5.1. Qualitative results and problems

Main goal for inpainting models is to generate realistic results that fill missing regions without inconsistencies. Figure 5 and 6 show samples of some of the best, cherry-picked, images inpainted by DCGAN and StyleGAN based models, respectively. From the figures we can notice that both models give comparative results in terms of alignment and finding an encoding close to the original image. What differs is that the StyleGAN based model gives fine and sharp details, compared to blurrier and less convincing facial features generated by DCGAN. However, this is expected, since StyleGAN is built to produce high-quality images, and is the only area in which stronger generative models could improve the end results.

Different types of corruptions(masks) applied to the images give different inpainting quality. Large regular missing regions tend to produce best results since the generative models generate samples that are consistent and realistic as a whole. Because of the inherent behaviour of GANs and the way importance weights are constructed, masks covering small area, spread across the whole image, lead the inpainting model to generate more artifacts. Images corrupted by noise can be recovered near perfectly in terms of shape and color, however, there are lots of artefacts all over the image. Images covered with hand drawn strokes leave either blurred lines or fillings which do not match the image at all. This is the case because these strokes cover different locations and features of the image, thus, the importance term within the loss tries to motivate the generative model to produce a result satisfying all these different features. In this case, the inpainting process should do many iterations to produce satisfying results.

Besides artifacts, some results could have severely misaligned edges, or features generated where they are not supposed to be, as shown on Figure 7 and 8. These mistakes could possibly stem from the inability of context loss to capture features good enough for alignment or there is a need for more iterations for backpropagating the loss – resulting in such errors. Even though increasing the number of iterations could improve the end results, this is impractical due to the time complex architectures might take to do the backpropagation.

5.2. Quantitative results

Numerical metrics, while not the best way to evaluate image inpainting models, offer a good way to conduct performance comparison. Result quality is measured using the metrics: l_1, structural similarity index (SSIM), with a window size of 11, and peak signal-to-noise ratio (PSNR). These metrics perform pixel-wise comparison, thus, not taking into consideration the perceptive quality. As a result, Fréchet Inception Distance (FID) [14] is used to measure the Wasserstein distance between embedded image features obtained from a pre-trained Inception-V3 model [20]. The results for DCGAN and StyleGAN based inpainting models are shown in table 1.

The results indicate that the StyleGAN based model significantly outperforms the DCGAN based model on images corrupted with noise. This can even be noticed in the qualitative results. Images corrupted by regular masks and hand-drawn strokes, according to the results, are reconstructed with slightly better quality by the DCGAN based model. These results might be lower on the StyleGAN recovered images due to the higher resolution of obtained results (256x256 vs. 64x64), as well low number backpropagation iterations.

	Mask	DCGAN	StyleGAN
ℓ_1 (%)	Regular	**2.25**	2.38
	Noise	5.6	**2.79**
	Quick Draw	**0.65**	0.79
SSIM	Regular	0.875	**0.891**
	Noise	0.729	**0.812**
	Quick Draw	**0.964**	0.946
PSNR	Regular	**25.481**	24.576
	Noise	20.745	**25.654**
	Quick Draw	**34.055**	32.326
FID	Regular	**30.598**	35.455
	Noise	203.981	**148.268**
	Quick Draw	**25.351**	32.309

Table 1: Quantitative results over FFHQ obtained from both DCGAN and StyleGAN based inpainting models. The best result of each row is boldfaced.

Figure 5: (Left to Right) Original image, input image, importance term, inpainted results from DCGAN based inpainting model.

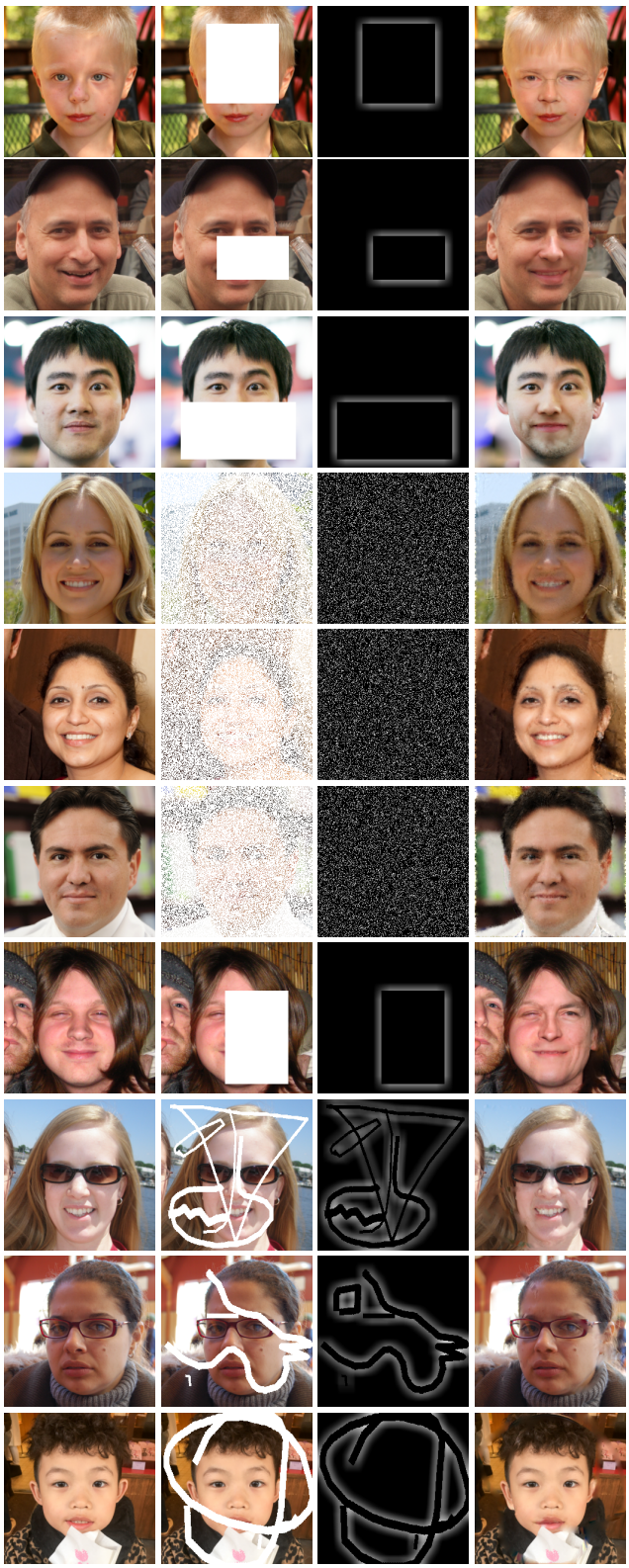

Figure 6: (Left to Right) Original image, input image, importance term, inpainted results from StyleGAN based inpainting model.

Figure 7: Bad results. (Left to Right) Original image, input image, importance term, inpainted results from DCGAN based inpainting model.

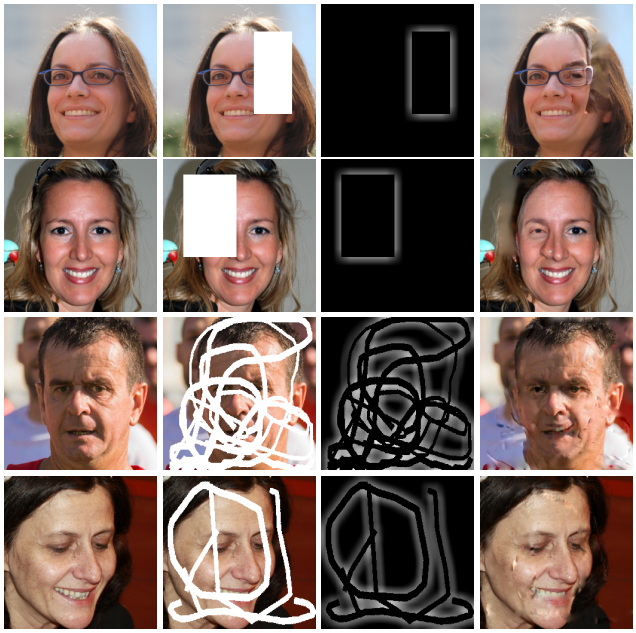

Figure 8: Bad results. (Left to Right) Original image, input image, importance term, inpainted results from StyleGAN based inpainting model.

6. Discussion and Possible Improvements

The solution for impainting images, proposed by [21], has been implemented using both DCGAN and StyleGAN generative models, as an extension proposed by the authors. It has been demonstrated that the model produces acceptable results on images missing large regions, while corruptions with small area, spanning large distances, tend to produce artifacts.

Even though the proposed model fills missing regions with good perceptual quality, there are problems which need to be solved. Problems introduced by inpainting results which do not fit the surrounding context could be mitigated by introducing a loss based on feature vectors from pretrained models, such as VGG. These features would take into consideration individual features, instead of focusing on a window around the corrupted region. This issue, along with the problem of slow inference, could be solved by hallucinating intermediate results. These intermediate results could be edges, as proposed by EdgeConnect [15], where the model combines edge generation and image completion to to fill in missing regions. Such approach would speed up the reconstruction process, by eliminating iteration, and produce results exhibiting fine details.

References

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan, 2017. 2

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 28(3), Aug. 2009. 1

[3] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. SIGGRAPH '00, page 417–424, USA, 2000. ACM Press/Addison-Wesley Publishing Co. 1

[4] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. *Proceedings of SIGGRAPH 2001*, pages 341–346, August 2001. 1

[5] S. Esedoglu. Digital inpainting based on the mumford-shah-euler image model. *European Journal of Applied Mathematics*, 13, 08 2003. 1

[6] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014. 1, 2

[7] J. Hays and A. A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics (SIGGRAPH 2007)*, 26(3), 2007. 1

[8] K. Iskakov. Qd-imd: Quick draw irregular mask dataset, 2018. 4

[9] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks, 2018. 3

[10] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016. 3

[11] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2018. 2

[12] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks, 2019. 1, 2, 4

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. 1

[14] A. Mathiasen and F. Hvilshøj. Fast fréchet inception distance, 2020. 5

[15] K. Nazeri, E. Ng, T. Joseph, F. Qureshi, and M. Ebrahimi. Edgeconnect: Generative image inpainting with adversarial edge learning. 2019. 2, 7

[16] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting, 2016. 2

[17] P. Pérez, M. Gangnet, and A. Blake. Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318, July 2003. 4

[18] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016. 1

[19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. 2015. 1

[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision, 2015. 5

[21] R. A. Yeh, C. Chen, T. Y. Lim, A. G. Schwing, M. Hasegawa-Johnson, and M. N. Do. Semantic image inpainting with deep generative models, 2017. 1, 2, 3, 4, 5, 7

[22] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang. Generative image inpainting with contextual attention, 2018. 2