Image inpainting: generative adversarial networks vs diffusion models

Ana Maria Jaramillo
anamaria.jaramillorojas@edu.fh-kaernten.ac.at
Saghar Ghaffari
saghar.ghaffari@edu.fh-kaernten.ac.at
Kärnten University of Applied Sciences

January 19, 2024

Abstract

This research is based on an image-processing task known as image inpainting, which involves the reconstruction of missing portions of an image based on contextual background information. The study compares two prominent techniques, Generative Adversarial Networks (GAN) and Diffusion Models, focusing on their efficacy in restoring and enhancing images. The experimental methodology involves the creation of a binary mask to simulate damaged image regions for inpainting, followed by the implementation of GAN and Diffusion Models on the "Places" dataset. The GAN architecture employs a generator-discriminator framework, while Diffusion Models introduce Gaussian noise to simulate the diffusion process. The research evaluates both approaches based on Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity Index (SSIM) metrics. Results indicate that the Diffusion Model outperforms GAN in terms of image quality despite GAN exhibiting computational efficiency. The findings align with existing literature, emphasizing the superior quantitative and qualitative performance of Diffusion Models in image inpainting despite the cost of increased computational demands.
Keywords*: image inpainting, generative adversarial networks, diffusion models, artificial neural networks, deep learning, generative models.*

I. Introduction

According to [1] and [2], inpainting can be defined as the process of filling in or reconstructing missing parts of an image based on the surrounding context. The main aims of doing it are based on creating a complete, realistic, and visually coherent representation of the image. This technique is usually applied to restore or enhance images [3] and recover missing or corrupted parts of them [4], given that specific pixels do not exist or are not wanted.

Looking forward to enriching the dataset to perform in-painting, a binary mask was implemented. In a specified region of the original image, some pixels were equaled to 0, creating black squares used as the damaged image that works as the input of the developed algorithms.

In order to create our own implementation for image inpainting, a brief description of two techniques that have been widely used for this task in the machine-learning world will be presented below. We have decided to present both implementations as an academic exercise of comparison

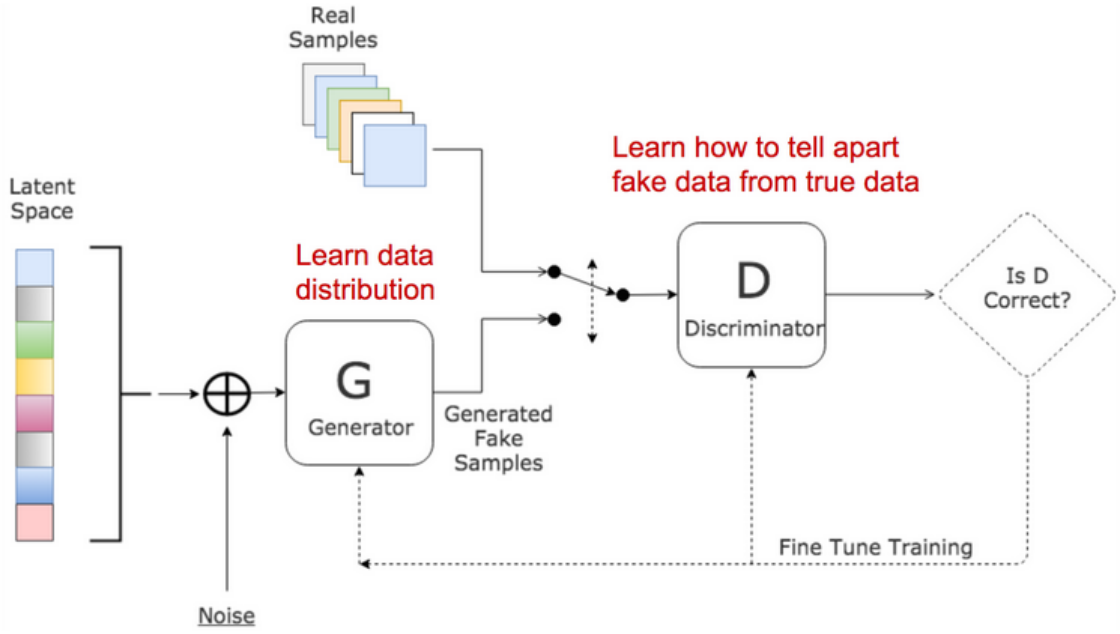

Figure 1: *Architecture of a generative adversarial network. (Image source: [7]).*

between both. The first implementation corresponds to Generative Adversarial Networks (GAN), and the second one is based on diffusion models. The methodology used for each implementation will be described in the fourth chapter, as well as the pre and post-processing of the images.

II. GAN

GAN is a type of machine learning model used for generating data. It consists of an architecture integrated by two neural networks competing against each other [5] [6]; this is a generator and a discriminator. The first one aims to create new data samples by a training process empowered by Stochastic Gradient Descent (SGD) or Adam optimizer (which was used in our implementation and is better explained below). The second one judges if the sample is an actual data point or generated data [1] [2]. This technique enables models to learn from mistakes, given that in the process of creation of new samples and auto judgment, the model improves its power to create every time images that are more and more difficult to distinguish from the original ones in order to make accurate predictions in the generation process.

In Figure 1 a schematical arquitecture of the GAN model is presented.

III. Diffusion model

Diffusion is a robust machine learning framework based on adding Gaussian noise to an image to learn the revert process and obtain the original image out of the noise [6]. It works based on variational algorithms or patch similarity processes, with the goal of propagating the information

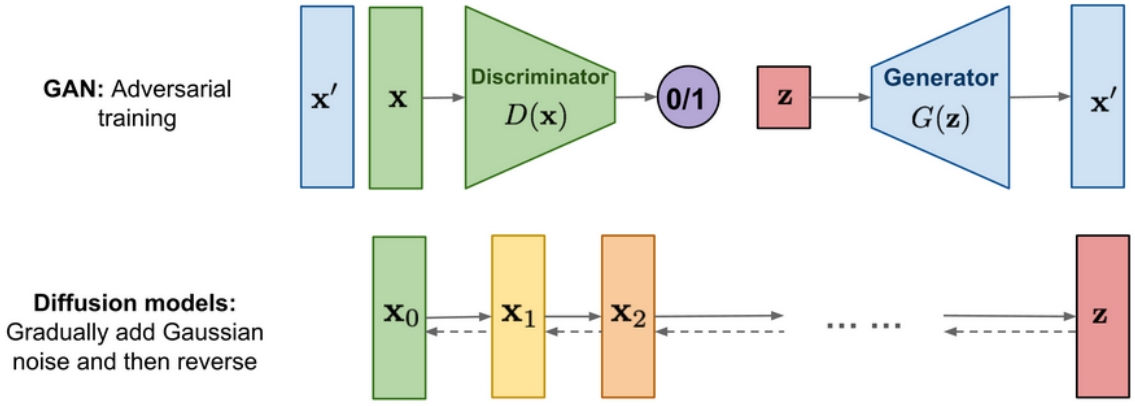

Figure 2: *Overview of generative models: GAN and Diffusion model (Image source: [9]).*

from background regions to the missing parts (or areas of interest) in the analyzed images [1]. This technique has been proven to work appropriately with stationary data [1]. The concept behind it is based on modeling the image pixel values as a random walk [8]. Diffusion models are used for image generation, and they have been the traditional method used in image inpainting; that is why this approach was incorporated into this project to present a side-by-side comparison of the techniques. In Figure 2 an overview, comparing the two generatives approaches is presented.

IV. Metodology

In the following link, you will find the developed notebook for this project: www.colab.com/ANN-IIProject

This project aims to self-implement a model for inpainting images with missing sections. We propose two approaches to compare the performance of the techniques used: GAN and Diffusion models. To do so, we will present below two paths conducted to generate the missing part of the images and repair them. The dataset "Places," available in [10], was used to train the models. Given the available computational resources, only the validation package was used. The dataset consists of 36.000 images of places, each 256 by 256 pixels.

i. Image pre-processing:

In order to guarantee the reproducibility of this implementation and to establish the same conditions for all images used for the training and validation process of this project, we established a common size for images equal to 128 by 128 pixels in three channels. For the inpainting implementation, it was necessary to have a dataset composed of two groups of images—first, the so-called "damaged" image, which was produced out of the original image. A code was developed to create the input ("damaged") dataset to make a black squares section inside the image. It works by selecting a random square rectangle of pixels and equaling it to zero; this means the creation of a black spot in the image. In the second group, we have the original images called "undamaged"

images. Both groups of images were adapted to the same parameters conditions as described in the previous section. Our task is to generate an image that fills the missing part based on the surroundings, producing a repaired image.

ii. GAN:

The architecture for the GAN implementation has two main parts: the generator and the discriminator. As described before, these two structures work together to create the best result.

For the generator, a simple, fully connected neural network was developed. It has three layers of increasing size and a final layer of the same size as the input image. The input to the generator is a random noise vector with a defined size. The activations between the layers use the LeakyReLU activation function, which is known to be beneficial for GANs. Batch normalization is also used between the layers to normalize the activations and improve training stability. Finally, the output layer uses the hyperbolic tangent activation function (tanh) to ensure that the output is in the range of [-1, 1], which is typical for image data.

The proposed discriminator uses a similar architecture as the generator. Still, it takes an image of a defined shape as input and produces a single output value between 0 and 1, representing the probability of the input being real or fake.

A manual trial-and-error process was done to tune the parameters involved in the generator and discriminator neural network, given that the implementation of an automated grid search for the best-performing parameters was beyond our computing resources' capability.

As the last but not most minor step for the GAN architecture, we have the optimizers. They play an essential role in the training process, given that they are responsible for tuning the attributes of the network to reduce the loss and improve the model performance. The Adam optimizer is a popular adaptive combination of momentum-based SGD and RMSprop, and it scales the learning rate using squared gradients while leveraging momentum using the gradient's moving average. An existing function for the Adam optimizer is available in TensorFlow, and it was used in our implementation.

The discriminator and generator work separately during the training process to optimize different objectives. The generator is trained to generate realistic images from a random noise vector, while the discriminator is trained to distinguish between authentic and generated images. The repair generator is trained to restore a damaged image by taking the original image and the corresponding damaged image as input and producing a repaired image as output.

The loss is measured in GAN for the three main processes: discriminator loss, generator loss, and repair generator loss. The discriminator loss is calculated using the binary cross-entropy loss function. This results in two separate loss values, one for the real images and one for the fake images, which are then averaged to obtain the final loss value for the discriminator. The generator

loss is calculated using the binary cross-entropy loss function. The generator loss is minimized to improve the ability of the generator to generate realistic images. The repair generator loss is calculated using the MSE loss function. The repair generator is trained to reconstruct undamaged images from their damaged counterparts by minimizing the difference between the repaired and undamaged images.

iii. Diffusion model:

A diffusion model for inpainting images is another generative model that simulates the process of diffusing noise into a damaged image to generate a repaired version [6]. For the self-made implementation in this project, we built a diffusion model using Tensorflow probability to model the diffusion process with a Gaussian distribution. This provides a probabilistic approach to diffusion modeling.

The image restoration model uses a partial differential equation to update the image's pixel values iteratively. That way, it restores the original image by iteratively diffusing the pixel values until the image converges to its original state. This implementation uses the mean squared error (MSE) loss function and the Adam optimizer.

Once built, the model should be compiled and trained. To do so, the model takes the undamaged and damaged couple and passes it through a fully connected layer with a tangent hyperbolic activation function to produce the diffusion output. This is then added to the undamaged input to produce the repaired output.

The next step is a training loop, where the model is trained on the damaged and corresponding undamaged images to create the restoration. During each training epoch, the model predicts the repaired images and then trains on a batch of images to evaluate the metrics.

iv. Metrics:

Two proposed parameters measure the performance of the model: Peak Signal-to-Noise Ratio (PSNR) is a metric used to measure the quality of reconstructed images, which focuses on pixel-wise reconstruction accuracy [2] [4]. PSNR range of values can vary depending on the specific application and the quality of the images being compared. Higher PSNR values generally indicate better image quality and lower differences between the original and restored images.

Structural Similarity Index (SSIM) is a pixel-level objective metric for measuring image distortion. It models image distortion by structure, luminance, and contrast and is commonly used for quantitative comparison of image inpainting results [4]. The scores range from -1 to 1, with 1 indicating perfect similarity between the two images. A negative score suggests that the generated image is structurally different from the original image. It is worth mentioning that SSIM is just a metric, and it might not capture all aspects of image quality.

Table 1: *Comparison of Repair Epoch, Repair Loss, PSNR, and SSIM*

Method	Repair Epoch	Repair loss	PSNR	SSIM
GAN	999/1000	0.349	10.78	0.46
Diffusion	999/1000	0.098	17.39	0.82

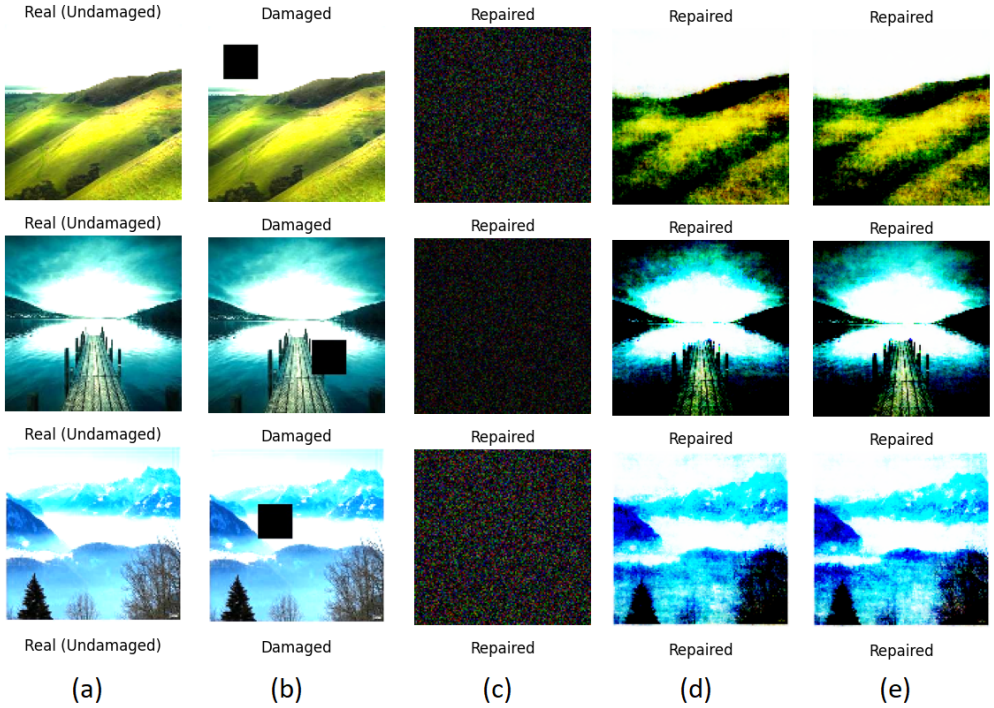

Figure 3: *Percentage of data per each emotion.*

V. Results

Table 1 presents the metrics for the validation set on the implemented models.

Figures 3 and 4 are presented in order to show a graphical example of the training results. Figures 3a and 4a present the real images (original or undamaged). Figures 3b and 4b present the damaged images, which correspond to the original image with the subtracted square. Figure 3c presents the generated image in the first epoch for the GAN implementation. Figure 3d presents the generated image in the epoch 4300 for the GAN implementation, and Figure 3e presents the generated image in the epoch 8000 for the GAN implementation.

Figure 4c presents the case scenario for the diffusion model implementation for the first epoch. Figure 4d presents the generated image in epoch 600 for the diffusion model implementation. Figure 4d presents the generated image in epoch 999 for the diffusion model implementation. The number of epochs presented in Diffusion models is reduced, given the consumption of

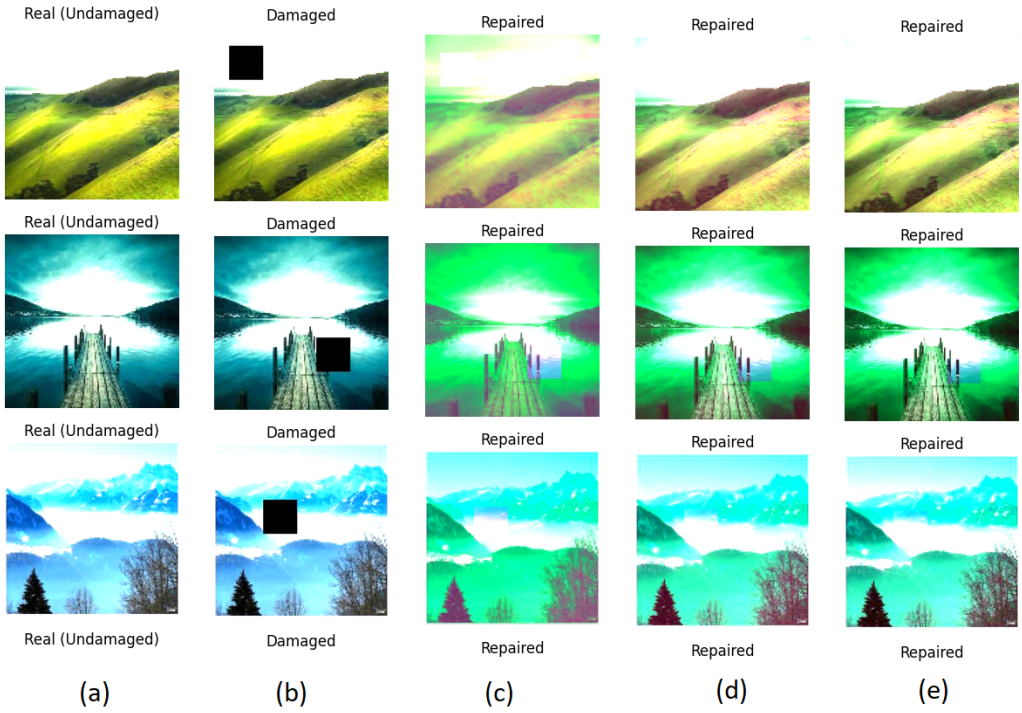

Figure 4: *Percentage of data per each emotion.*

computational resources demanded by this approach.

VI. Conclusions and discussion

During this project's implementation, we had a profound look at the parameters and computational resources that affect generation models. After many different tries of tuning parameters given the conditions of memory that we had available and a closer view of the obtained results (see images 3 and 4), we were able to conclude that the Diffusion model is giving a better-repaired image, even though the presented GAN model was more efficient in time and computational resources.

After an analysis of the results, we have found that diffusion models show significant quantitative and qualitative improvements in image inpainting (compared to GAN-based models). This approach has outstanding generative capabilities and can effectively capture the compositional structure of images. In contrast, GAN-based models are efficient, leading to a faster result but trading it with the resulting quality of the generated image, which is supported by [5]. Also, [8] mentions that Diffusion models produce high-quality images for the inpainting processes. Still, the cost of this is high computational power and longer training times—making it less suitable for complex projects.

VII. Future work

In order to increase the generation power of the diffusion model, we developed an implementation that incorporates a residual block structure within the diffusion process. This structure is based on convolutional neural networks (CNNs), allowing for more complex feature learning within the diffusion process. Therefore, this approach was not concluded and not included in the final code. It was proven to be highly memory-expensive, and given the complexity of our approach, we were out of processing memory to continue it. Trying this implementation, we concluded that diffusion models are not optimal regarding computational resources. Still, having them available, we could have gotten much better metrics for the generations. For this reason, we would like to point out that there is still a long way to go in the scope of diffusion models.

References

[1] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, and T. S. Huang, "Generative image inpainting with contextual attention," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5505–5514.

[2] C.-T. Li, "Introduction to generative models for image inpainting and review: Context encoders," 2020, published by Medium. [Online]. Available: https://medium.com/analytics-vidhya/introduction-to-generative-models-for-image-inpainting-and-review-context-encoders-13e48df30244

[3] Q. Fu, Y. Guan, and Y. Yang, "Image inpainting and object removal with deep convolutional gan," Tech. rep. URl: http://stanford. edu/class/ee367/Winter2018 . . . , Tech. Rep.

[4] Y. Yu, L. Zhang, H. Fan, and T. Luo, "High-fidelity image inpainting with gan inversion," in *European Conference on Computer Vision*. Springer, 2022, pp. 242–258.

[5] A. Burak Yildirim, V. Baday, E. Erdem, A. Erdem, and A. Dundar, "Inst-inpaint: Instructing to remove objects with diffusion models," *arXiv e-prints*, pp. arXiv–2304, 2023.

[6] S. K. Mandala, "A contextualized survey on the state and future directions of diffusion models," 2023.

[7] L. Weng, "From gan to wgan," *lilianweng.github.io*, 2017. [Online]. Available: https://lilianweng.github.io/posts/2017-08-20-gan/

[8] A. Muhammad, K. Lee, C. Lim, J. Hyeon, Z. Salman, and D. Han, "Gan vs diffusion: Instance-aware inpainting on small datasets," in *2023 International Technical Conference on Circuits/Systems, Computers, and Communications (ITC-CSCC)*. IEEE, 2023, pp. 1–5.

[9] L. Weng, "What are diffusion models?" *lilianweng.github.io*, Jul 2021. [Online]. Available: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

[10] M. I. of Technology, "Dataset: Places2," accessed: 2024-01-19. [Online]. Available: http://places2.csail.mit.edu/download-private.html