Diffusion Model for Generating new Pokémon

Paper

Michael Hüppe

08.12.2023

Abstract

Making its gaming debut in 1996 with "Pokémon Red and Green" and earning approximately 368 million from game sales alone, the Pokémon series stands as one of Nintendo's oldest and most well-known franchises. With a unique blend of character designs and turn-based role-playing, the game introduced an entire generation to a distinctive gaming style, earning its place as the seventh most successful game of all time. Consequently, the franchise's current cultural significance comes as no surprise.

The community's embrace of "Fakémons," or self-made Pokémon, has reached a point where these creations feature prominently in entire fan-made games. This widespread enthusiasm paved the way for the exploration of generating new Pokémon through the application of deep learning. In recent years the generation of images has become more popular with the easy access to generative networks such as DALL-E, StyleGAN2 or GPT-3. Diffusion models describe one of the most popular architectures. This paper outlines the methodology employed for scraping and standardizing the data, along with the construction of the network architecture.

1 Introduction

The following presents the possibility and methodology of using Denoising Diffusion Probabilistic Models (DDPM) to generate novel designs for Pokémons. Firstly, previous approaches and their used models are presented in Section 2. Following, the methodology is described namely the model architecture in Section 3 and the data acquisition and processing Section 4. Finally, the results are presented and discussed in Section 5 and 6. Section 7 presents the shortcomings of this study and what can be improved upon.

2 Related Work

The following section is divided into two. Firstly, related works regarding Generative Adversarial Network (GAN)s are discussed with a closer look into the current uses of DDPMs. Secondly, previous approaches using GANs to create new images of Pokémon are analyzed. There are multitudes of variation of GANs, with notable architectures like Deep Convolutional Generative Adversarial Network (DCGAN) (Radford *et al.*, 2016), StyleGAN (Karras *et al.*, 2019), and BigGAN (Brock *et al.*, 2019). While all of these models have the same underlying purpose, i.e. generating novel images based on an existing data pool, their architecture has been optimized for a specific task. For example the StyleGan architecture is used for combining images, often applying certain features of one image onto another (Karras *et al.*, 2019). In contrast, the BigGan architecture is used to enhance the quality of images often up-scaling low resolution images to a higher resolution (Brock *et al.*, 2019).
Given their ability to model intricate patterns and dependencies within data DDPMs have been employed in various tasks. For example, image translation (Sasaki *et al.*, 2021), inpainting (Lugmayr *et al.*, 2022) or image restoration (Nair *et al.*, 2023). DDPMs have also been used beyond image generation, examples are feature extraction (Bandara *et al.*, 2024), anomaly detection (Wyatt *et al.*, 2022) and time-series forecasting (Rasul *et al.*, 2021). Given their variability in use-cases DDPMs are a good fit for generating new Pokémon designs.

Generating Pokémon using deep learning is not a novel idea. Kleiber, 2020 used a GAN to create new Pokémon based on a dataset of 800 data samples. The presented results seen in Figure 1 can definitely be approved upon.

Using the same dataset Chambel, 2022 trained a DCGAN and achieved better results. The colour of the generated images is much closer to the often vibrant colour presented in the dataset. The results can be seen in Figure 2.

Figure 1: Results of Kleiber, 2020. The object somewhat resemble the colour scheme and form of a Pokémons. However, no image is recognizable as a complete design.

Figure 2: Results presented by Chambel, 2022.

3 Model

The model employed was based on the Denoising Diffusion Probabilistic Models (DDPM) presented by Ho *et al.*, 2020 who presents both an unconditional Model and a conditional one. For generating the images I only employed the conditional architecture. However, the architecture was slightly modified. The most significant change was the implementation of a scale factor which gives the ability to increase/decrease the amount of model parameters.

DDPM are a class of generative models that aim to model the data distribution by defining a diffusion process. The primary idea behind DDPMs is to transform a simple distribution (e.g., a Gaussian distribution) into the target data distribution through a series of carefully designed transformations. This diffusion process is done step-by-step slowly transforming the input noise to an image resembling the training data.
At each step of the diffusion process, a noise is added to the data, and the data is transformed to a new state. This process is repeated multiple times until the input noise has been transformed to an image resembling the training data.
The model is trained by optimizing its parameters to maximize the likelihood of the observed data. During training, the model learns the transformations that map the simple initial distribution to the target data distribution. This involves adjusting the parameters in a way that the generated samples match the characteristics of the training data.
Once trained, the DDPM can generate new samples by starting with samples from the simple initial distribution and applying the learned reverse transformations in the reverse order.
Overall, DDPMs provide a probabilistic framework for generative modeling, leveraging a diffusion process to model the transformation from a simple distribution to a complex data distribution. This allows them to capture intricate patterns and dependencies in the data.

The diffusion model approach can be used in combination with a multitude of

model architectures. The most common and the model used by us was the U-Net as presented by Ronneberger *et al.*, 2015.
The U-Net is named after the shape of the models' architecture. As a DDPM the U-Net aims to predict noise, to achieve this it creates a feature map for the input image. Using convolutions and max-pooling the resolution of the feature map is decreased at each step. This resembles the idea of a decoder to break down each image into it's most important components. Similarly, the U-Net implements an encoding process where resolution of the previously reduced feature map are then increased to replicate an image from decoding noise. This is done by using up-convolutions. Additionally, the U-Net implements skip connections at each resolution change. An original illustration of the U-Net is depicted in Figure 3.

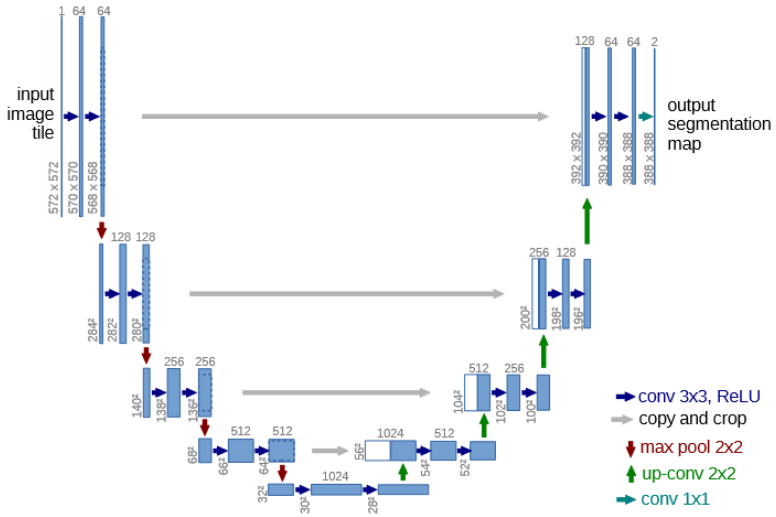

Figure 3: U-Net architecture as presented by Ronneberger *et al.*, 2015.

4 Data Acquisition and Preprocessing

4.1 Acquisition

There is no official Pokémon dataset that provides the extensive sample size typically required for training generative deep learning models (Yang *et al.*, 2023). Most of the available datasets offer only one image per Pokémon, resulting in approximately 1000 images in total. In comparison, the Canadian Institute for Advanced Research - 10 (CIFAR-10) dataset consists of 60,000 samples. Moreover, each Pokémon has a unique and easily distinguishable design, making it challenging to identify similarities between them. To address this limitation, I opted to create my own Pokémon dataset using web scraping.

I used web scraping to extract data from the Pokémon database, which lists all Pokémon along with various design variations for each. The Python packages BeautifulSoup and requests packages provide functions to parse the website into

Hypertext Markup Language (HTML) and identify relevant tags. The national Pokédex which provides a list of all current Pokémon organized by Generation [1] represents the start point of the acquisition pipeline.

I collected links for each entry, simplifying the retrieval of all accessible images through a systematic naming convention. The link structure is as follows: "pokemondb.net/identifier/Pokémon." Changing the identifier granted access to two distinct datasets. The "artwork" identifier encompassed both official and alternative artwork, while the "sprites" section stored all in-game renditions of the Pokémon. In total, I obtained 26,896 unique images (22,825 sprites and 4,071 artworks). The term "unique" requires caution, as I intentionally included both male and female versions (even if differing by minor details) and normal and shiny versions (same Pokémon but with different color coding). Additionally, the database provides GIFs depicting the idle animation of the Pokémon. The start and middle frame of these animations were added to the dataset. The back view of Pokémon were ignored since the goal was to generate images from the front.

The code for web scraping the images can be found here. Figure 4 shows an example of artworks and sprites for the Pokémon entry for Charizard.

Figure 4: Different renditions of the same Pokémon. Both samples from the artwork and sprite dataset are presented

Note that the trained model is a conditional diffusion model, thus the samples should be categorized in a logical scheme. There are multiple approaches on how the samples can be grouped. Typically, Pokémon are grouped based on their typing [2]. However, optimally similar images are grouped together. Pokémon belonging to the same type can differ drastically in regards to their appearance. Since the goal was to maximize the inner-class visual similarity and the between-class discrepancy, I opted to group samples based on the body from the Pokémon. This

[1] In the context of Pokémon, a "generation" refers to a specific group or series of Pokémon games released by Nintendo and Game Freak. Each generation introduces a new set of Pokémon species, game mechanics, and often a new region to explore.

[2] Typing refers to the elemental or thematic category (fire, water, ghost, fairy etc.) that a Pokémon belongs to. Each Pokémon has one or two types, which determine its strengths and weaknesses in battles. There are 18 different types in total, and each type has its own set of interactions with other types.

resulted in following classes.

Pokémon with:

1. only a head
2. a head and legs
3. fins
4. insectoid body
5. quadruped body
6. single pair of wings
7. multiple bodies
8. tentacles
9. base and legs
10. bipedal with tail
11. bipedal without tail
12. two or more pairs of wings
13. serpentine body
14. a head and arms

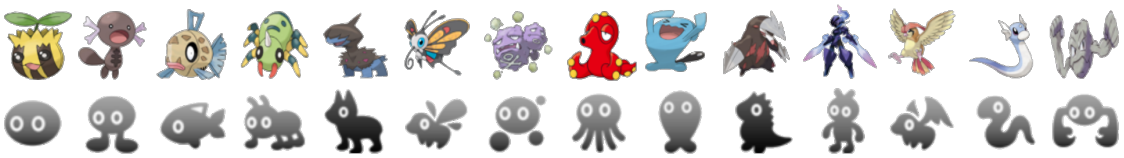

Figure 5: Different body types used to categorise the Pokémon.

4.2 Preprocessing

The next problem I encountered was the difference in their background, their image mode and type. At first I resize every image to 64x64. The original artwork and sprites were all transparent ".png", meaning they have no background and therefore a fourth channel encoding the transparency of the given pixel. This fourth channel resulted in partially transparent Pokémon and added unnecessary parameters to the network. Therefore I converted each of the "RGBA" images to uniform "RGB" images with a white background to keep their characteristic black outlining.
As the alternative artworks were collected from a variety of different artists there was no convention resulting in different file types (jpg, jpeg and vector png), backgrounds and image modes. To keep the format I established in the previous step I again removed the additional transparency channel and converted each image to png. Additionally, I detected each image which did not have a white background and adjusted it accordingly.
Moreover, was there a difference in coverage of the image. While the artworks used all available space of the image the sprites only used around 10% therefore exposing a lot of white background. This was apparent after training a few test epochs which resulted in solely white images with a small focus somewhere in the center (which was no surprise as the sprites represented the majority of the dataset). To counteract this, I located the focus point of the image (location of Pokémon) by removing each row/column only containing white pixels. Then to avoid morphing the shape of the Pokémon I padded it to a square. At last I again resized the cropped image to get a uniform size of 64x64 pixels.
The code for making the images uniform can be found here.

An example for a sample before and after applying the uniform process can be seen in Figure 6.

Figure 6: A sample before and after applying the uniform process.

The previous preprocessing steps were done before training the model to ensure that the focus for each samples is on the Pokémon. In addition to this, more preprocessing was done while training to each sample. This included applying a random crop to the image, randomly flipping it on the vertical axis and normalize each colour channel to have a mean of 1 and a standard deviation of 1.

5 Results

The following describes the results reported on both the CIFAR-10 dataset and our own Pokémon dataset. Since the quality of the generated image is somewhat subjective the Mean-square Error (MSE) during training is presented as well.

5.1 CIFAR-10

To validate the model pipeline I first trained the model on the CIFAR-10 dataset since the model has already reported good results on this data. The CIFAR-10 dataset is a collection of 60,000 32x32 color images in 10 different classes, with 6,000 images per class. Each image belongs to one of the following classes:

1. airplane
2. car
3. bird
4. cat
5. deer
6. dog
7. frog
8. horse
9. ship
10. truck

Similarly, our model reported good results as Ill.

Figure 7 shows the change of MSE during training. Additionally, epoch 30 and 100 are highlighted with sample images from these epochs. The MSE decreases rapidly at the start of the training. For the following epochs the MSE hovers around 0.01 and 0.04. While stable within this range there is no steady (in-)decrease. However, the images differ heavily when it comes to resemblance to the training images. Figure (TBA) shows the predicted images over time. As one can see the images predicted in the early stages of training were not reminiscent of the input data and mostly show either random noise or images containing only one colour. Similar to how the MSE decreased over time the quality of the images increased. The final images clearly depict objects of the respected class.

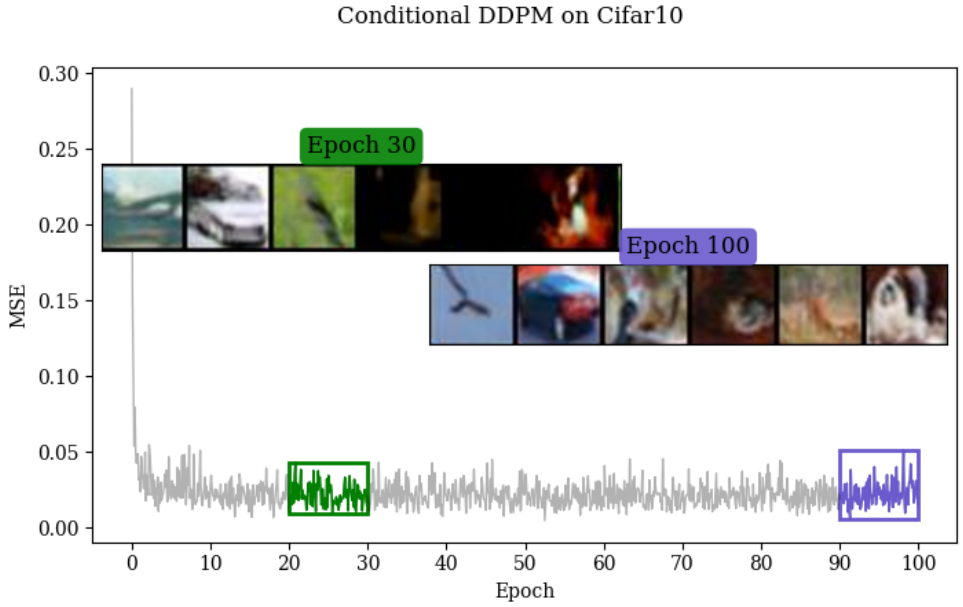

Figure 7: CIFAR-10 MSE over time.

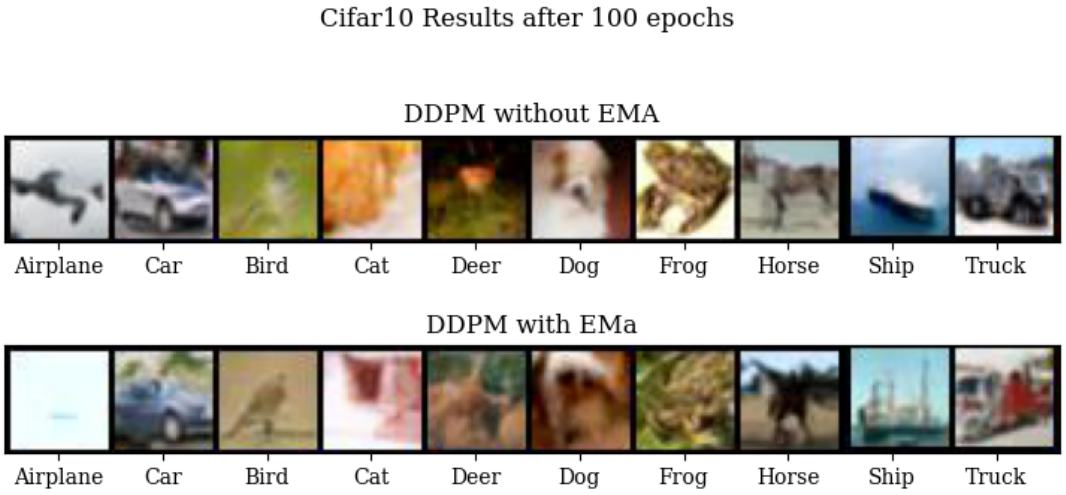

Figure 8: Generated images of DDPM

5.2 Pokémon

The results presented in Section 5.1 validated that the DDPM was able to replicate images from the training dataset specific for each class. The following describes model performance on the self-crafted Pokémon dataset.

Figure 9 displays generated images after training for each respective class showing the conditionality of the model. Most generated images resemble the body type they are attributed to.

Figure 9: Pokémon per body type.

Figure 10 shows the change of MSE during training. Similarly to Figure 7 there is a rapid decrease of MSE at the start of the training with a MSE staying relatively constant. Consistent with the results for the CIFAR-10 dataset the images keep improving even though the loss seems to be stagnant. However, there is a variety in quality of generated pictures with respect to class. Namely, the body types with wings seem to be replicated the best.

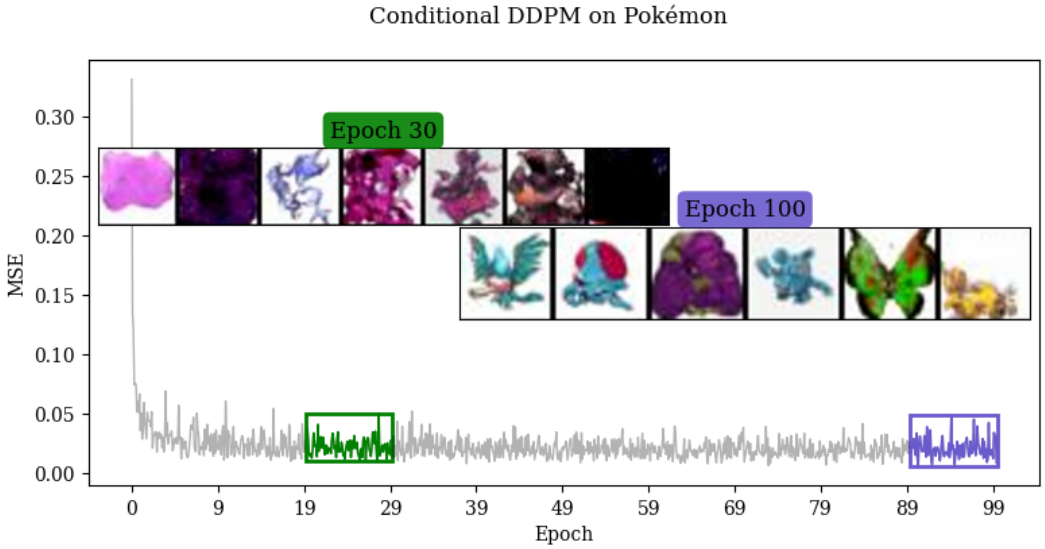

Figure 10: Change of MSE over time. The MSE values leading up to the 30 and 100 epoch are highlighted. Additionally, generated samples for both highlighted epochs are shown.

A known problem with GANs is mode collapse (Zhang *et al.*, 2018). This phenomenon describes models which focus on the most common feature in the dataset to generate new images. Instead of generating diverse samples that cover the entire data distribution, the generator might focus on generating samples with certain features. This lack of diversity in the generated data can be seen in the class "only head".
This category includes the Pokémon "Unown", which mimics the Latin alphabet and thus has 28 variations. Though different, all variations follow the same design rules, black lines and one eye. Accounting for all different image types 53% of the images in this category were just depicting some variation of this Pokémon. This can be seen in the results generated for this class. Mainly, most of th generated images follow the "Unown" design. Examples can be seen in Figure 11.

Figure 11: Different generated images for the head category reminiscent of the Pokémon "Unown".

Moreover, there is a noticeable difference in the coherence of images among classes. Images generated for a specific class exhibit greater similarities, whereas other classes display a higher variance in the generated images. This is illustrated in Figure 12. The samples generated for the class "quadrupled" differ more strongly in comparison to images generated for the class "winged".

Figure 12: Comparison between classes "with wings" (left) and "bipedal without tail" (right).

6 Discussion

Our model demonstrated comparable performance on both the benchmark dataset and our custom created dataset. The incorporation of the diffusion model and the expansion of the dataset proved to exert a positive influence on the quality of the predicted images. Notably, our images exhibited a more refined design compared to previous studies (refer to Figure 1 & 2). This outcome was anticipated, as generative models tend to excel when operating with larger datasets (Yang *et al.*, 2023). Furthermore, it is noteworthy that DDPMs exhibited the capability to generate images of higher quality compared to simpler GANs (Guarnera *et al.*, 2023).

The displayed examples in Figure 9 depict that the model is generally able to generate Pokémon for a given body type. The samples were grouped to maximize the similarity within a class and the difference between two classes. However, Pokémon expressed their associated body type to various degrees blurring the lines between classes and decreasing the similarity within a class.
Additionally, given the unbalanced nature of the dataset the model learned to focus on the Pokémon that were represented more often for the given class resulting in a mode collapse for the class "only head" as presented in Figure 11. The mode collapse can be avoided by filtering the images within the class or employing sub sampling i.e. train more on the Pokémons in this category that are represented less. Moreover, as shown in Figure 12 the difference in inner-class variety was fairly

high. If one wants to increase the similarity within a class other grouping methods can be of use. An additional approach would be to have vectorized class instead of a one-class system. Meaning, one image can have various attributes linked to it and fed into the model. Classes based on the image might be size (in percentage of image coverage) or colour. Classes based on the presented Pokémon could be type, generation, evolution state or region.

The following phenomena were found regardless of dataset: Even though the MSE was not steadily decreasing after a certain amount of time the generated images were still improving. The quality of images was dependent on the class were images for some classes were more indicative of the training images than other classes. The images that showed the highest image quality for the CIFAR-10 dataset were images belonging to the class car, truck and ship. For the Pokémon dataset the body types showing the highest quality in generated images were single/multiple pair of wings and only head. Additionally, no significant difference in quality between images generated using Exponential Moving Average (EMA) compared to images where EMA was not used can be seen. This is depicted in Figure 8.

7 Conclusion

Increasing the dataset and using class dependent DDPM has shown positive effects on the image quality in comparison to previous approaches. While the model was overall successful in generating Pokémon based on body type, the training was mainly impeded by the lack of hardware. The model was trained using an NVIDIA GeForce GTX 1650 with 4GB of dedicated GPU memory. This allowed only images of the size 34x34 to be trained on and generated. Additionally, the biggest batch size without exceeding the available space was 8 samples, which increased the training time and hindered the model from training on multiple samples at once. Future studies should aim to increase the size of the images to be trained on and implement other grouping methods or class encodings.

References

Bandara, W. G. C., Nair, N. G., & Patel, V. M. (2024). Ddpm-cd: Denoising diffusion probabilistic models as feature extractors for change detection.

Brock, A., Donahue, J., & Simonyan, K. (2019). Large scale gan training for high fidelity natural image synthesis.

Chambel, G. (2022). Generating realistic pokemons using a dcgan. https://medium.com/@goncalorrc/generating-realistic-pokemons-using-a-dcgan-331c7f75e211

Guarnera, L., Giudice, O., & Battiato, S. (2023). Level up the deepfake detection: A method to effectively discriminate images generated by gan architectures and diffusion models.

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (pp. 6840–6851, Vol. 33). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf

Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks.

Kleiber, J. (2020). Pokegan: Generating fake pokemon with a generative adversarial network. https://medium.com/@jkleiber8/pokegan-generating-fake-pokemon-with-a-generative-adversarial-network-f540db81548d

Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., & Van Gool, L. (2022). Repaint: Inpainting using denoising diffusion probabilistic models. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11461–11471.

Nair, N. G., Mei, K., & Patel, V. M. (2023). At-ddpm: Restoring faces degraded by atmospheric turbulence using denoising diffusion probabilistic models. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 3434–3443.

Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks.

Rasul, K., Seward, C., Schuster, I., & Vollgraf, R. (2021, 18–24 Jul). Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th international conference on machine learning* (pp. 8857–8868, Vol. 139). PMLR. https://proceedings.mlr.press/v139/rasul21a.html

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation.

Sasaki, H., Willcocks, C. G., & Breckon, T. P. (2021). Unit-ddpm: Unpaired image translation with denoising diffusion probabilistic models.

Wyatt, J., Leach, A., Schmon, S. M., & Willcocks, C. G. (2022). Anoddpm: Anomaly detection with denoising diffusion probabilistic models using simplex noise. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 650–656.

Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Zhang, W., Cui, B., & Yang, M.-H. (2023). Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys*, *56*(4), 1–39.

Zhang, Z., Li, M., & Yu, J. (2018). On the convergence and mode collapse of GAN. *SIGGRAPH Asia 2018 Technical Briefs*, 1–4. https://doi.org/10.1145/3283254.3283282