ECE-285: Image Outpainting using Self-Attention Generative Adversarial Networks

Shreyan Sood
Halıcıoğlu Data Science Institute
s1sood@ucsd.edu
PID : A59020240

Swapnil Ghosh
Halıcıoğlu Data Science Institute
s4ghosh@ucsd.edu
PID : A59020045

Abstract

The image outpainting problem involves extending an existing image in a visually coherent and plausible manner, creating additional content that seamlessly integrates with the original image. This project aims to explore different approaches, with a particular focus on Generative Adversarial Networks (GANs). GANs have shown promising results in generating realistic and high-quality images. We also specifically investigate the use of Self-Attention GANs (SAGANs), which incorporate self-attention mechanisms to capture long-range dependencies and improve image generation quality. By examining various techniques and leveraging course insights, we aim to improve the quality and realism of the generated outpaintings.

1 Introduction

The applications of image outpainting have garnered significant attention and excitement in recent times. The objective is to generate new image content that aligns with the style and context of the original image, appearing as a natural continuation of the scene. The introduction of Adobe's Generative fill feature and the advancements in models like DALL.E-2 have made image outpainting more accessible and widely adopted. This has led to a surge in creative outputs, with numerous instances of extended versions of iconic artworks like the Mona Lisa circulating on social media. The ability to seamlessly generate visually plausible extensions to images has opened up new possibilities for artistic expression, image editing, virtual reality experiences, and even meme creation. The buzz surrounding image outpainting highlights its growing popularity and its potential to revolutionize various domains where visual content plays a crucial role.

Below is an example of the final representations that we want to achieve.

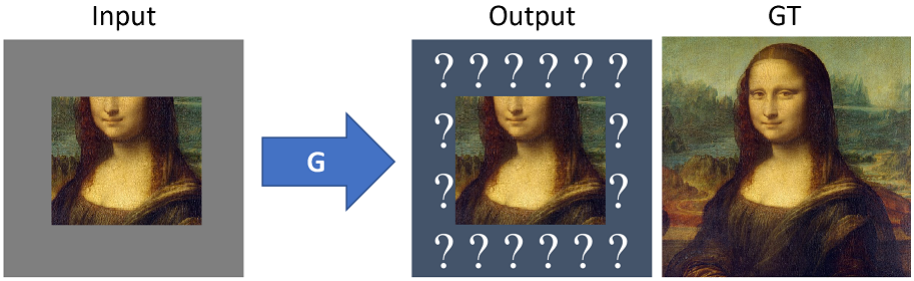


Figure 1: Expected results of Image Outpainting

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Unlike image inpainting, where the objective is to fill in missing or corrupted parts of an image, outpainting focuses on expanding the canvas beyond the existing image content. This can be useful in various applications such as image editing, virtual reality, video games, and generating high-resolution images.

The image outpainting problem poses several challenges. The generated content should be coherent and consistent with the original image, avoiding abrupt transitions or noticeable artifacts. It should also exhibit diversity, providing different plausible continuations based on the same input image. Additionally, the generated content should be semantically meaningful and aligned with the context of the original image.

We employ two different architectures: Generative Adversarial Networks (GAN) and Self-Attention Generative Adversarial Networks (SAGAN). Our objective is to explore the effectiveness of both models in generating additional content that seamlessly integrates with the original image. By training and evaluating these architectures, we conduct a comparative analysis to assess their performance and determine if SAGAN exhibits any advantages over the traditional GAN approach in the context of image outpainting.

2 Related Work

Image outpainting is a challenging task that aims to generate the surroundings of an input image, extending beyond its original boundaries. This problem encompasses various applications, including image extension, novel view synthesis, infinite landscape generation, and panorama generation, all with the objective of producing visually coherent and complete outputs. The literature in this field offers a wide range of techniques and models, each with its own strengths and limitations, providing a rich landscape for further exploration and advancements in the realm of image outpainting.

Recent developments in the field have witnessed a surge of interest in diffusion and score-based models. These models tackle the image outpainting problem by learning to reverse a sequential corruption process. They iteratively introduce small amounts of Gaussian noise to the data, gradually transforming it into random noise. By employing a series of refinement steps, these models denoise the random noise to generate samples that follow the underlying data distribution. This sequential denoising process enables diffusion and score-based models to generate high-quality samples that capture intricate details and structures present in the input image.

Deep generative models, such as Generative Adversarial Networks (GANs), provide another natural approach to image outpainting. By learning the conditional distribution of output images given the input, GANs have gained popularity for image-to-image translation tasks. Their ability to generate high-fidelity outputs, broad applicability, and efficient sampling make them a preferred choice for image outpainting tasks.

Variational Autoencoders (VAEs) have also been explored in the context of image outpainting, presenting a distinct approach from diffusion models. VAEs, as a type of deep generative model, learn the underlying distribution of the input data and generate new samples by sampling from this learned distribution. However, VAEs may face challenges in capturing fine-grained details and generating high-fidelity outpaintings compared to alternative approaches like GANs.

An emerging avenue of research in image outpainting involves utilizing transformers for image completion. Transformers offer advantages such as non-local attention, which facilitates generating global structures and maintaining contextual consistency. Moreover, sampling from the learned distribution by a transformer enables the production of more diverse image completions compared to other methods like CVAE. However, transformers can be computationally intensive, particularly when dealing with large images, which poses a challenge in practical implementation.

3 Methodology

3.1 Architecure

The generator network G incorporates the context encoder, which utilizes six convolutional layers to downsample the masked input. This downsampling enables the efficient representation of image content and object semantics within an embedding space. The convolutional layer is followed by

a self attention layer as described below. The decoder, on the other hand, employs special layers known as deconvolutional layers. These layers effectively reverse the downsampling performed by the encoder, allowing for the reconstruction of the original image.

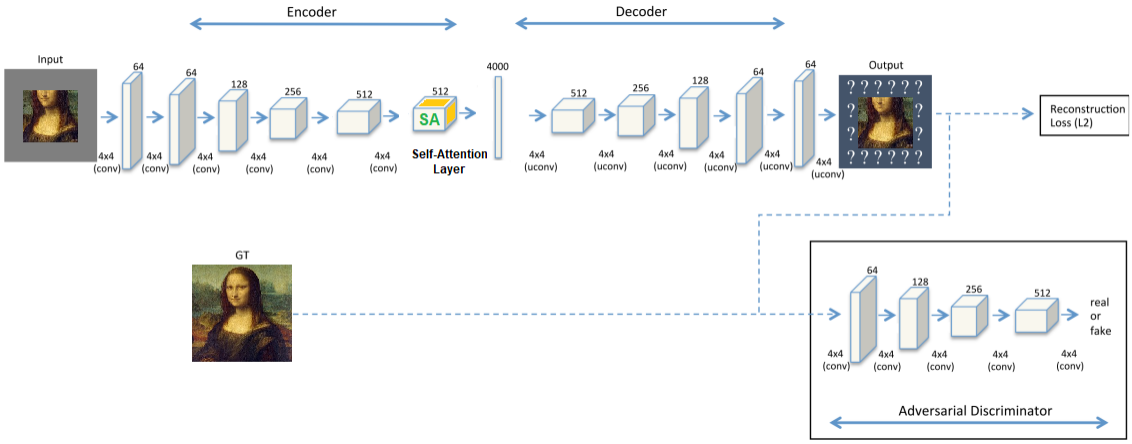


Figure 2: Self-Attention Model Architecture

The discriminator network D is a deep neural network responsible for estimating the likelihood of the ground truth or the generated image being real. In inpainting tasks, D typically only observes the generated portion of the image. However, in this project, we opt to evaluate the entire outpainted image using D to discourage the generator G from introducing noticeable perceptual inconsistencies or structural irregularities. The architecture of D produces a grid of probabilities with a size of 24×24, and the errors associated with these probabilities are averaged during the training process.

The generator network G consists of multiple layers designed to transform a random input into a realistic image. The layers in the generator network are as follows:

Table 1: Layers of the generator network G.

Type	Kernel size	(H, W, C)
Conv + Leaky ReLU	$4 * 4$	$(96, 96, 64)$
Conv + BatchNorm + Leaky ReLU	$4 * 4$	$(48, 48, 64)$
Conv + BatchNorm + Leaky ReLU	$4 * 4$	$(24, 24, 128)$
Conv + BatchNorm + Leaky ReLU	$4 * 4$	$(12, 12, 256)$
Conv + BatchNorm + Leaky ReLU	$4 * 4$	$(6, 6, 512)$
Attention	$1 * 1$	(512)
Conv	$4 * 4$	$(3, 3, 4000)$
Up-conv + BatchNorm + ReLU	$4 * 4$	$(6, 6, 512)$
Up-conv + BatchNorm + ReLU	$4 * 4$	$(12, 12, 256)$
Up-conv + BatchNorm + ReLU	$4 * 4$	$(24, 24, 128)$
Up-conv + BatchNorm + ReLU	$4 * 4$	$(48, 48, 64)$
Up-conv + BatchNorm + ReLU	$4 * 4$	$(96, 96, 64)$
Up-conv + BatchNorm + Tanh	$4 * 4$	$(192, 192, 3)$

The discriminator network D aims to distinguish between real and generated images. It consists of the following layers:

Table 2: Layers of the discriminator network D.

Type	Kernel size	(H, W, C)
Conv + Leaky ReLU	$3 * 3$	$(96, 96, 64)$
Conv + InstanceNorm + Leaky ReLU	$3 * 3$	$(48, 48, 128)$
Conv + InstanceNorm + Leaky ReLU	$3 * 3$	$(24, 24, 256)$
Conv + InstanceNorm + Leaky ReLU	$3 * 3$	$(24, 24, 512)$
Conv	$3 * 3$	$(24, 24, 1)$

3.2 Attention

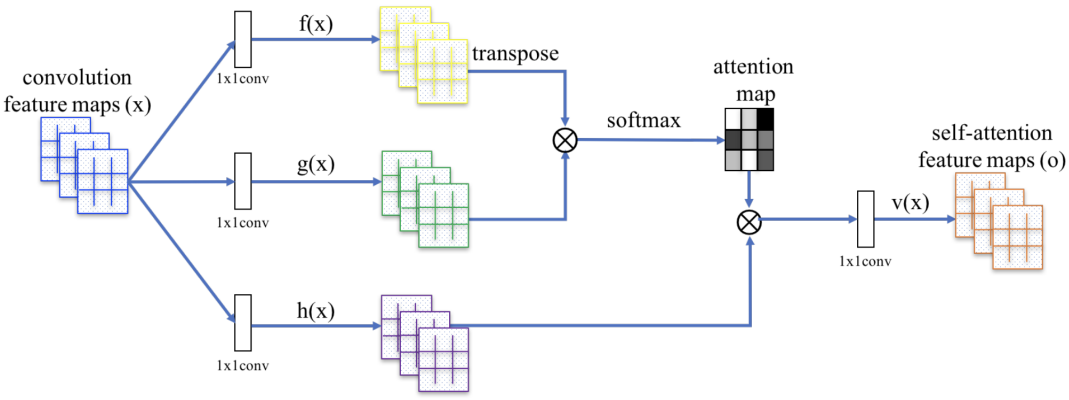


Figure 3: Expected results of Image Outpainting **?**

A self-attention layer is a fundamental component in many state-of-the-art models, such as Transformer-based architectures. It allows the model to capture long-range dependencies and attend to different parts of the input sequence.

In this formulation, the input to the self-attention layer is denoted as X with a shape of $(H \times W, C)$, where H represents the height, W represents the width, and C represents the number of channels.

The self-attention mechanism involves three linear transformations: query, key, and value. These transformations are represented as W_Q, W_K, and W_V, respectively, and each has a shape of (C, C). In practice, these linear transformations are often implemented as 1x1 convolutional layers with matching dimensions.

By passing the input X through the query, key, and value transforms, we obtain output volumes denoted as XW_Q, XW_K, and XW_V, respectively. These volumes capture important information about the relationships between different elements in the input.

The self-attention operation is then computed using the following formula:

$$\text{Attention}(X) = X + \text{Softmax}\left(\frac{XW_Q(XW_K)^\top}{\sqrt{C}}\right)XW_V$$

The attention scores are calculated by taking the dot product of the query and key transformed volumes, scaled by the square root of the channel dimension C, and passed through a softmax function. These scores determine the importance or relevance of each element in the input sequence.

Finally, the self-attention output is obtained by multiplying the attention scores with the value transformed volume and adding it back to the original input.

This mechanism allows the model to focus on different parts of the input based on their importance, facilitating the modeling of complex dependencies and enhancing performance in various natural language processing and computer vision tasks.

4 Experiments

4.1 Dataset

During the training process, we adopted a strategy of cropping images before inputting them into the generator. This approach allowed us to train the model in a self-supervised manner by enforcing the generated output to approximate the original, uncropped image. To accomplish this, we utilized a sufficiently large dataset of unlabelled, natural photos. For our experiments, we used a smaller subset of the MIT CSAIL Places365-Standard dataset, which contained millions of images depicting landscapes, buildings, and various everyday scenes for scene recognition purposes. We divided the

dataset into training, testing, and validation sets, with 10,000 images allocated for the training set, 1,000 images for the testing set, and 200 images for the validation set. This partitioning allowed us to make the most of our available computational resources while still maintaining representative subsets for training, evaluation, and model fine-tuning.

Figure 4: Selected Images from Places365 dataset

As a preprocessing step, we resized the photos to 192x192 dimensions. Subsequently, we assigned the generator the task of expanding a 128x128 crop back into a 192x192 image. While adding 32 pixels at each boundary may not have appeared significant, this configuration already posed a considerable challenge. The total output size was 2.25 times that of the input, implying that over half of all pixels needed to be generated by the network. In practice, the generator network (G) mapped a partially masked 192x192 color image to a completed version of the same dimensions, where the masked region was replaced by the model's predictions.

4.2 Training

The training process was conducted for 100 epochs, employing a fixed learning rate of $\alpha = 0.0003$ and two Adam optimizers with $\beta_1 = 0.5$ and $\beta_2 = 0.999$. We utilized a DSMLP machine with 1 GeForce GTX 1080 Ti GPU, 8 CPUs and 16 GB RAM to train the model. The batch size was set to 16, indicating that 16 images were processed in parallel during each iteration of the training process. The loss functions used in the training were defined as follows:

$$L_{rec} = ||x - G(x)||_1 \tag{1}$$

$$L_{adv} = ||D(G(x)) - 1||_2^2 \tag{2}$$

$$L_G = \lambda_{rec} L_{rec} + \lambda_{adv} L_{adv} \tag{3}$$

$$L_D = ||D(x) - 1||_2^2 + ||D(G(x)) - 0||_2^2 \tag{4}$$

To produce less blurry images, we opted for an L_1 reconstruction loss instead of L_2. The first loss function, L_{rec}, represented the L_1 reconstruction loss, measuring the pixel-wise difference between the generated image ($G(x)$) and the ground truth image (x). This loss function aimed to produce less blurry images compared to the commonly used L_2 reconstruction loss.

The second loss function, L_{adv}, represented the adversarial loss. It quantified the discrepancy between the discriminator's (D) output when applied to the generated image ($G(x)$) and the real image, targeting a value of 1. This loss function encouraged the generator to produce more realistic images that could deceive the discriminator.

To balance the contribution of the reconstruction loss and the adversarial loss, we combined them using the third loss function, L_G. The relative weight of the adversarial loss was controlled by the hyperparameter λ_{adv}.

Additionally, we employed the discriminator loss, L_D, which consisted of two terms. The first term measured the discrepancy between the discriminator's output when applied to the real image and the target value of 1.

Adjusting the weight of the adversarial loss λ_{adv} relative to the reconstruction loss λ_{rec} proved to be a challenge. Initially, we set $\lambda_{adv} = 0.001$ following the approach in various existing works. However, we encountered a problem where the GAN consistently collapsed into a failure mode,

where the adversarial loss did not deviate from 1. This indicated that the generator (G) was unable to deceive the discriminator (D), and D was always successful in distinguishing real from fake images.

To address this issue, we devised a remedy by dynamically adjusting λ_{adv} throughout training epochs. The adjustment was performed as a function of the epoch number (n) according to the following scheme:

$$\lambda_{adv}(n) = \begin{cases} 0.001, & \text{if } n \leq 10 \\ 0.005, & \text{if } 10 < n \leq 30 \\ 0.015, & \text{if } 30 < n \leq 60 \\ 0.040, & \text{otherwise} \end{cases} \tag{5}$$

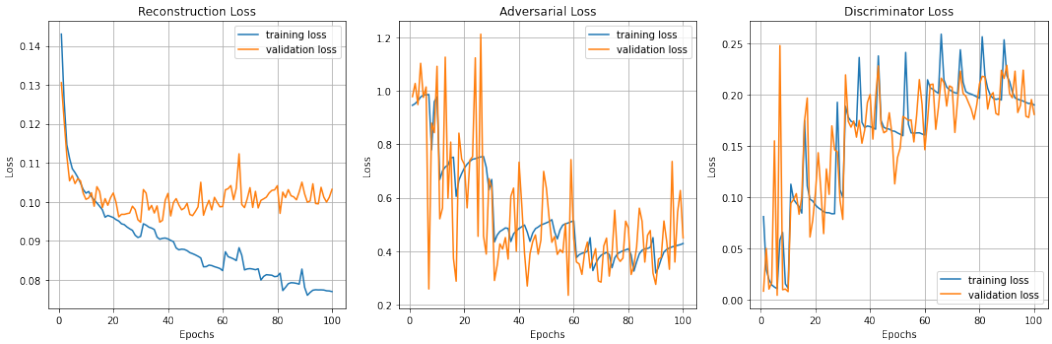
By gradually increasing the weight of the adversarial loss as training progressed, the generator was penalized more severely for generating unrealistic outputs, moving beyond a simple pixel-wise reconstruction constraint.

These techniques were applied during the training process to encourage better image quality and address the challenge of the adversarial loss.

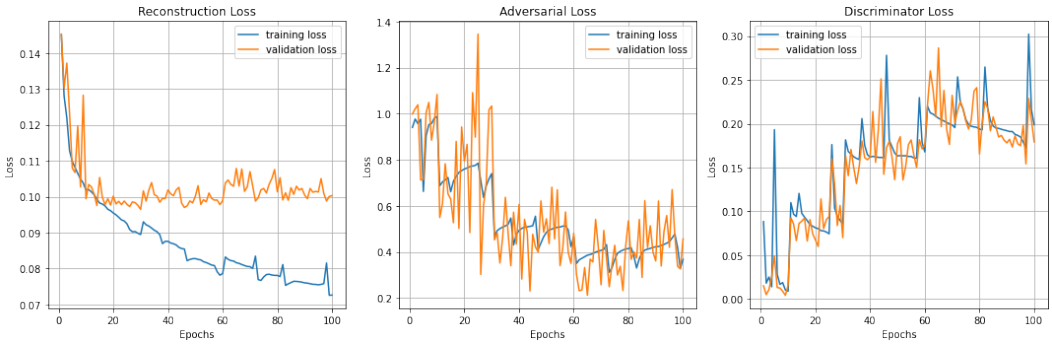
5 Results

We evaluated the performance of our image outpainting models using our GAN and and Self-Attention GAN architectures. The models were trained using the aforementioned loss functions and hyperparameters. In this section, we present the results of the training process by examining the convergence of the loss functions over the training epochs.

First, we analyze the performance of the GAN architecture. The following plot displays the variation of the pixel-wise loss (reconstruction Loss), the adversarial loss, and the discriminator loss over the 100 training epochs.



(a) GAN architecture



(b) Self Attention GAN architecture

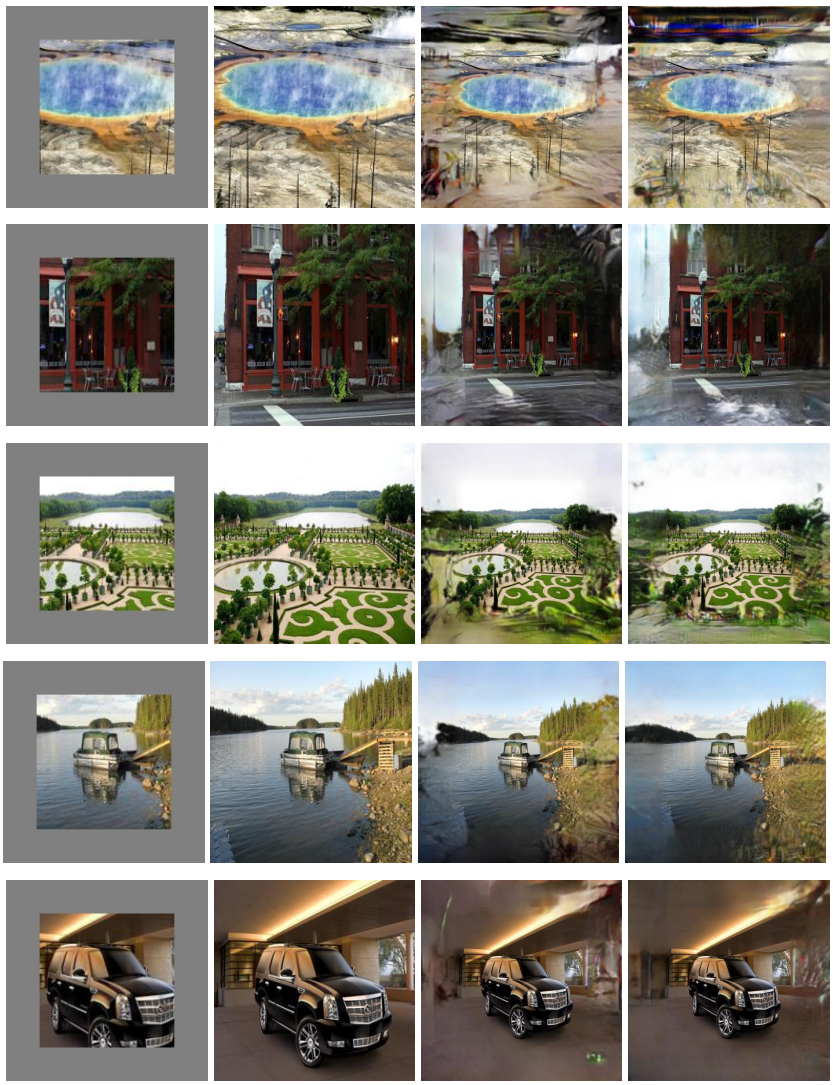
Figure 5: Learning curves for the outpainting architectures

Both the GAN and SAGAN models exhibit similar learning curves during training, but the SAGAN architecture demonstrates slightly better performance in terms of loss. This suggests that the integration of self-attention mechanisms in SAGAN contributes to improved loss optimization and better generation of realistic outpaintings.

We also evaluate and compare the final validation loss metrics of the GAN and SAGAN models after 100 epochs of training. These metrics offer valuable insights into the quality of the generated images and the level of realism achieved by the models. The results are summarized in Table 3, providing a comprehensive comparison between the two architectures.

Table 3: Validation Loss Values for both architectures after 100 training epochs

Model	L_{rec}	L_{adv}	L_D
GAN	0.1032	0.4500	0.1811
SAGAN	0.1002	0.4569	0.1792



(a) Input (b) Ground Truth (c) GAN (d) SAGAN

Figure 6: Examples of outpainting on Places365 dataset

The GAN architecture achieved a validation loss of 0.1032 for reconstruction, 0.4500 for adversarial training, and 0.1811 for discriminator learning. In comparison, the SAGAN architecture obtained slightly lower validation losses of 0.1002, 0.4569, and 0.1792 for reconstruction, adversarial training, and discriminator learning, respectively. These results suggest that both architectures performed similarly, with the SAGAN architecture exhibiting a slightly improved performance in terms of reconstruction and discriminator learning.

6 Conclusion

We can see that the SAGAN-generated outpaintings exhibit noticeably better visual quality, with sharper details, smoother transitions, and more realistic textures than the original GAN based model. For the same training method and resources, SAGAN seamlessly extends the original image, creating a coherent and visually pleasing result. The self-attention mechanism allows the model to capture global contextual information effectively. This enables SAGAN to generate outpaintings with better coherence and consistency, as it can effectively consider dependencies across distant regions of the image. It also helps the model to have a better understanding of the spatial relationships within the image. helps to mitigate the issue of generating blurry outpaintings.

7 References

1. Zhang, H., Goodfellow, I.J., Metaxas, D.N., & Odena, A. (2018). Self-Attention Generative Adversarial Networks. ArXiv, abs/1805.08318.
2. Hoorick, B.V. (2019). Image Outpainting and Harmonization using Generative Adversarial Networks. ArXiv, abs/1912.10960.
3. Esser, P., Rombach, R., & Ommer, B. (2020). Taming Transformers for High-Resolution Image Synthesis. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 12868-12878.
4. Chang, H., Zhang, H., Jiang, L., Liu, C., & Freeman, W.T. (2022). MaskGIT: Masked Generative Image Transformer. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 11305-11315.
5. Sabini, M., & Rusak, G. (2018). Painting Outside the Box: Image Outpainting with GANs. ArXiv, abs/1808.08483.
6. Wang, Y., Tao, X., Shen, X., & Jia, J. (2019). Wide-Context Semantic Image Extrapolation. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 1399-1408.
7. Yang, Z., Dong, J., Liu, P., Yang, Y., & Yan, S. (2019). Very Long Natural Scenery Image Prediction by Outpainting. 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 10560-10569.
8. Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. CoRR, abs/1511.06434.
9. Gardias, P., Arthur, E., & Sun, H. (2020). Enhanced Residual Networks for Context-based Image Outpainting. ArXiv, abs/2005.06723.
10. Jo, C., Im, W., & Yoon, S. (2021). In-N-Out: Towards Good Initialization for Inpainting and Outpainting. ArXiv, abs/2106.13953.