The Importance of Growing Up: Progressive Growing GANs for Image Inpainting

Daniel Speck[1], Theresa Pekarek Rosin[1], Matthias Kerzel[1], Stefan Wermter[1]

***Abstract*— In recent years, Generative Adversarial Networks (GANs) have proven to be a sophisticated approach for generative tasks in image processing, especially inpainting and image synthesis While most GAN approaches feature comparatively large networks, we introduce an approach to image inpainting using progressive growing GANs, which enables significantly reduced model sizes, faster convergence, and thus an overall more efficient training that is inspired by insights on the development of visual abilities in biological systems. We demonstrate the effectiveness and efficiency of our approach on Places, a comprehensive dataset encompassing a wide variety of images from diverse locations in the wild.**

***Index Terms*— progressive growing, GAN, image inpainting, model size reduction, efficient training**

I. Introduction

Image data can be damaged, incomplete, or of low quality due to noise. Image manipulation techniques, such as inpainting, super-resolution, and denoising, address these issues. This paper focuses on image inpainting, which fills blank or masked areas with context-sensitive information.

Inpainting approaches are typically categorized into sequential-based, Convolutional Neural Network (CNN)-based, and Generative Adversarial Network (GAN)-based methods [1]. Sequential-based methods include patch-based approaches [2]–[5] and diffusion-based approaches [6]–[8]. To capture global image structure, CNN-based architectures such as ShiftNet [9] and Liu et al.'s approach [10] were introduced. GAN-based techniques [11]–[13], employing coarse-to-fine networks and contextual attention modules, further enhance inpainting performance. However, they require more training time and possess a high parameter count, posing challenges for machines with limited computational resources.

Our main contribution is a progressive growing GAN [14] trained on the Places dataset [15] for image inpainting up to a resolution of 64×64 pixels. While our approach enables faster convergence it mainly focuses on a significantly smaller model size compared to other GAN- or stable diffusion-based methods by mixing the progressive growing mechanism with an efficient layer design and utilizing multiple loss functions such as perceptual loss [16]. This incremental learning approach resembles human learning in its progression from simple to complex and from lower to higher resolutions.

[1]Knowledge Technology, Department of Informatics, University of Hamburg, Germany, {daniel.speck, theresa.pekarek-rosin, matthias.kerzel, stefan.wermter}@uni-hamburg.de

II. Related Work

A. Image Inpainting with GANs

The work by Suvorov et al. [13] demonstrates a mask inpainting method called LaMa (Large Mask Inpainting) that is based on fast Fourier convolutions, originally proposed by Chi et al. [17]. Compared to standard convolution operations that compute their input based on local neighborhood kernels of fixed size, this type of convolution uses Fourier transforms. The filters compute input information on a global scale, which is particularly useful for images containing repeating structures since information can be replayed. High-resolution results of the LaMa approach with arbitrary masks look promising for images with repeating patterns and are visually nearly indistinguishable from real images. However, the performance decreases when the input does not contain many repeating patterns, and the application domain differs from the training data. Iizuka et al. [12] introduce an adversarial training approach on both the global and the local level by utilizing a second local Discriminator that only takes in a small portion of the image. This leads to high-quality results for smaller mask sizes, but their approach still struggles with large masks and heavily structured objects. Although both approaches, within their respective limitations, deliver high-quality results they require extensive training time (up to 2 months for Iizuka et al.) and a high number of iterations, 1M for LaMa and 500k for Iizuka et al., for a final resolution of 512×512 (LaMa) and 256×256 (Iizuka et al.). This demonstrates that there is a need for more dynamic approaches utilizing transfer learning to reduce training time and number of total iterations.

B. Progressive Growing

A substantial contribution towards stable GAN training and high-resolution networks is the approach by Karras et al. [14], which introduces GAN models that grow during training depending on a target resolution. Training a large neural network, with two competing networks during training, is a challenging task, and convergence for high-resolution output is hard to achieve. The work of Karras et al. addresses this problem by initially training a small network for a 4×4 resolution for a limited number of epochs. Subsequently, the resolution is doubled by incorporating additional layers. The resulting network is then trained on the new 8×8 resolution. This process is iteratively applied to the entire dataset and all epochs for each resolution until the maximum resolution is reached. Incrementally adding new layers that introduce higher-

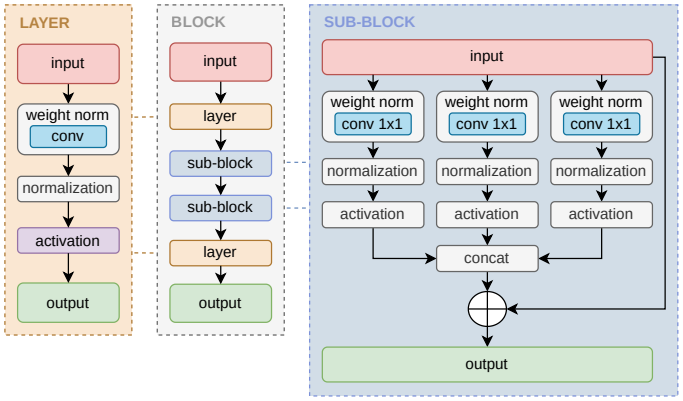

Fig. 1. **Architecture of blocks, sub-blocks, and layers.** These are the modular constructs used to build Discriminator (D) and Generator (G). Depending on the setup, weight normalization or regular normalization is used but never both at the same time.

resolution information as the training advances significantly accelerates training time and improves stability at high resolutions. In the case of Karras et al. [14], the maximum resolution is 1024×1024, and the generator is able to produce highly realistic-looking images of faces for validation on the CelebA [18] dataset.

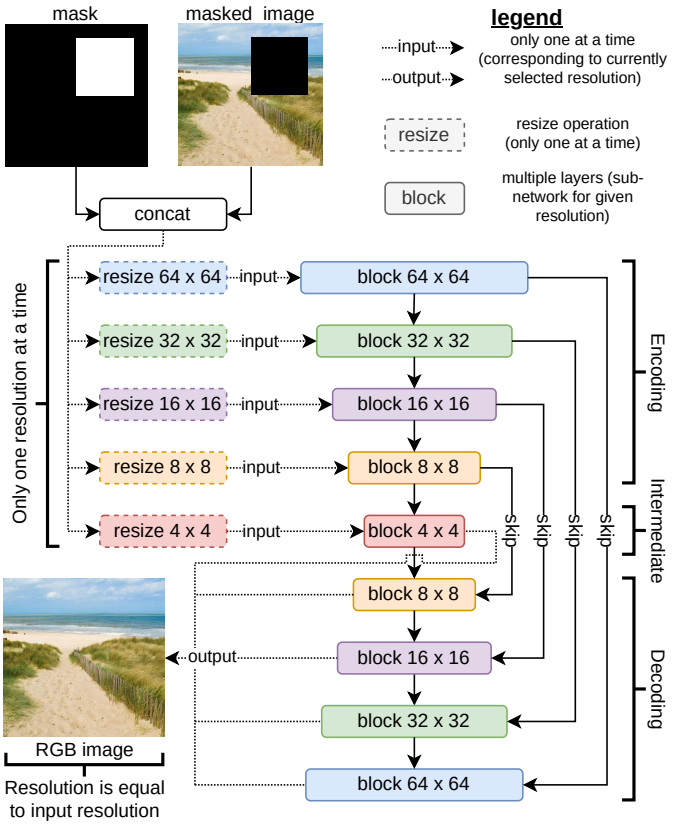

Fig. 2. **Proposed Generator (G).** The network starts to train the lowest resolution of 4×4 and grows once this resolution is fully trained. Each growing step doubles the resolution by adding blocks to the encoder and decoder part of G. This process is adapted from Karras et al. [14].

C. Advanced Loss Functions

In standard GAN scenarios, the generator G and discriminator D train adversarially using binary cross-entropy loss applied to D's output. D learns to distinguish generated samples from real ones, acquiring features and domain knowledge that G uses to generate samples resembling real images. However, the lack of connection between D and G in the loss may result in imbalanced learning rates and an absence of qualitative metrics to ensure G's output is visually close to real images might lead to less realistic images. The task becomes dependent on D's ability to learn proper image representation for discrimination. To address those issues, GAN-based approaches often employ advanced loss functions, such as BEGAN's [19] boundary equilibrium. This method balances two objectives: overall image consistency (coverage of low-frequency features) and image quality (coverage of high-frequency features). Berthelot et al. achieve this by constructing a GAN architecture resembling an autoencoder and utilizing a balanced loss term to weigh the autoencoder and GAN loss. The autoencoder loss ensures accurate target learning through low-frequency feature representation, while the GAN loss introduces variance by learning high-frequency features, adding sharpness and variation to the output.

Another approach to increase sharpness through high-frequency features in the generated images and ensure diversity in the output is the perceptual loss by Johnson et al. [16]. In our approach, we use a pre-trained network, e. g. a classification network trained on a large dataset like ImageNet [20] and feed the original as well as the generated image into only the first few layers of this pre-trained network, which consists mostly of filters, modeling rather simple, high-frequency features like edge detection. Let N denote the pre-trained network itself, x an input, and $N_{0,1}$ the first and second convolutional block of the classification network. $N_{0,1}(x)$ is the output of the network's first and second convolutional block, which functions as a feature map modeling high-frequency features that describe sharpness and a high level of detail in the image. Using these feature maps to design a loss function will result in a decent approximation of the overall image quality. The function itself compares the divergence of $N_{0,1}(x_{real})$, meaning the output of the first and second convolutional blocks of the classification network with the original image as input, and $N_{0,1}(x_{generated})$ with the generated image as input. By comparing $N_{0,1}(x_{real})$ and $N_{0,1}(x_{generated})$, we obtain the perceptual loss that tries to approximate the overall image quality in concepts like sharpness, which are otherwise very challenging to model.

III. Methodology

A. Dataset

Inpainting images recorded "in the wild" require a training dataset that covers a large variety of domains, as well as a diverse set of features within them, and consists of high-quality images that allow for learning a diverse representation of high-frequency features. Following these requirements, we chose the Places [15] dataset, which is curated for scene

understanding through a focus on typical visual tasks. This enables the model to learn a knowledge representation that includes scene understanding, context information, and object recognition. Consequently, the categories and the number of images, over 10 million in total, are chosen to reflect a diverse set of real-world areas and domains, covering many different backgrounds and objects at the same time. MNIST [21] was used for prototyping in the early development of our approach, due to its simple training requirements.

B. Architecture

Both networks, the Discriminator (D) and the Generator (G), are built with modular constructs, which we call blocks, sub-blocks, and layers. An illustration can be seen in Figure 1. A block can either be an *up* block used for upsampling or a *down* block used for downsampling. Each block represents the sub-network responsible for one resolution. G uses both *down* and *up* blocks to half or double the resolution of the image respectively. D only utilizes down blocks. A sub-block is a residual part of the architecture. It takes in feature maps, applies three different convolutions, then concatenates the results and adds the input to its output. A layer is the smallest modular construct and consists of one convolution, which can be either regular or transposed, and either strided or non-strided. To map the input to blocks, regular convolutions without strides are used, while layers connected to a block's output use strided convolutions, either regular ones for downsampling or transposed ones for upsampling. The *activation* corresponds to an activation function that is passed at the initial construction and then rolled out to the entire network. Similarly, the configuration for the normalization is passed at the initial construction and then used for the entire network.

The Generator (G) (Figure 2) is responsible for the inpainting task. G receives a 4D tensor as input, which is the concatenation of a masked RGB image (3D) and a binary mask image (1D). The network consists of modular building blocks, sub-blocks, and layers (see Figure 1). The growing process, described in Section II-B, is the main concept of how to simplify the training process for the entire GAN and keep it small. The final model consists of only 328,887 parameters since training in a low resolution first is a much easier task and faster to execute.

One benefit of working with a modular architecture is that it is possible to test many different configurations and stop training early in case of diverging gradients or bad image quality. Additionally, only a few parameters need to be changed to test regular/weight normalization, different loss configurations, different numbers of layers, or different activation functions. The final Generator architecture uses spectral normalization as weight normalization and PyTorch's default LeakyReLU class with 0.01 as the value for the negative slope as an activation function. All parameters were either taken as default from PyTorch or chosen empirically by testing various configurations. The Discriminator (D) is built from the same building blocks as G, see Figure 1. However, since D just needs to downsample the information and subsequently flatten and process it through fully-connected layers, we use only down blocks in the construction of D.

C. Perceptual Loss Network

Our advanced loss function consists of a adversarial loss, a reconstructional loss (mean-squared error), and a perceptual loss. The perceptual loss network LN is a standard VGG19 [22] classification network. We use a snapshot of this network pre-trained on ImageNet [20] via TIMM[1] and extract the first two feature blocks for the calculation of the perceptual loss in Equation (1).

$$L_{perceptual} = \frac{\sqrt{\sum (LN(G(x_{masked})) - LN(x_{real}))^2}}{c * h * w} \quad (1)$$

Here, x_{masked} is the input batch with masked images, and x_{real} is the input batch with the real (non-masked) images. $LN(G(x_{masked}))$ describes the output of the loss network LN with $G(x_{masked})$ as input, where $G(x_{masked})$ are the generated samples, meaning the output of G. $LN(x_{real})$ defines the output of LN while using x_{real} as input, thus giving an estimation of how edges and other high-frequency features should look like because these features are extracted based on real images. The term $c * h * w$ describes the number of channels, height, and width of the resulting feature maps, which normalizes the output. This step is needed to restrict the loss to a reasonable domain and give an approximation of the average quality throughout the whole image rather than a pixel-based metric or sum.

D. Training Procedure

Generator G and Discriminator D are trained in an adversarial approach, competing against each other. The progressive growing techniques extend this setup by repeating the whole procedure, i. e. training the network over all epochs for all resolutions, starting at the smallest resolution of 4×4. To prevent the continued adaptation of already learned features for a previously trained resolution, the weights of all layers corresponding to the smaller resolution are frozen after it is trained. For example, once the training for the resolution of 8×8 starts, the trained layers for the resolution of 4×4 are frozen. This process is repeated every time the resolution is doubled.

IV. Results and Discussion

A. Generative Results on Places

The final results for the Places [15] dataset were obtained by using the full architecture (see Section III-B), i.e. training until and including the resolution of 64×64. Our model generates visually plausible information up to a resolution of 16×16. Freezing the layers responsible for the resolutions of 4×4 and 8×8 lets the network train the resolution of 16×16 faster than training from scratch. Visually appealing results in a resolution of 16×16 start to appear after training approximately the first 5-10k batches (batch size of 128 images). On the

[1] https://github.com/rwightman/pytorch-image-models

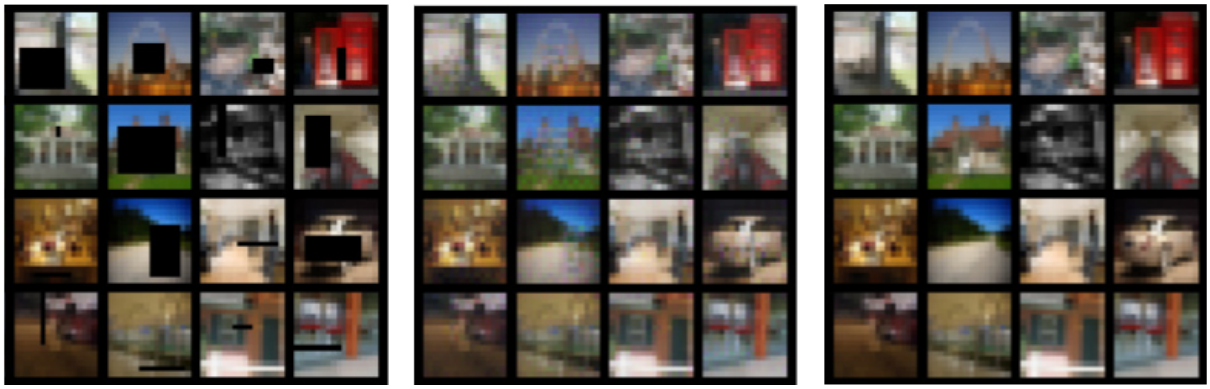

Fig. 3. **Results (final architecture) for Places dataset in 16×16 resolution** (randomly selected validation batch). From left to right: masked input, generated samples, original images. In the final architecture, the mask was passed as additional input to the network alongside the masked image, and all losses were used. As can be seen from these results, most images are reconstructed in a visually plausible way, regardless of mask size.

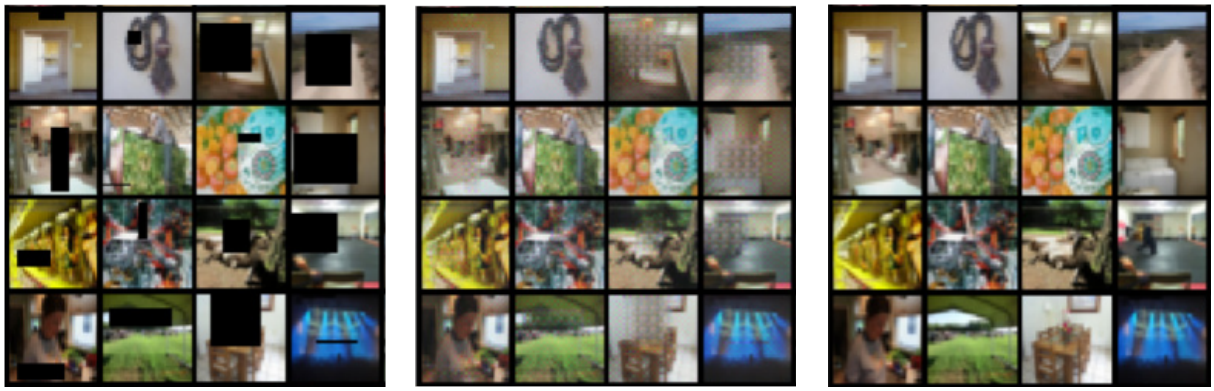

Fig. 4. **Results (final architecture) for Places dataset in 32×32 resolution** (randomly selected validation batch). From left to right: masked input, generated samples, original images. In the final architecture, the mask was passed as additional input to the network alongside the masked image, and all losses were used. For smaller masks and images with clear edges, the generator performs visually plausible inpainting. However, for larger masks, the generated information contains artifacts.

TABLE I

A DIRECT COMPARISON OF NUMBER OF PARAMETERS PER MODEL.

Ours	**DCGAN** [23]	**WGAN** small [24]	**Iizuka et al.** [12]	**WGAN** full [24]	**LaMa** [13]	**dVAE** [25]	**DDPM** [26]
0.3M	1.0M	2.5M	6.0M	10.0M	27.0M	14.0M + 84.6M	167.0M

other hand, training the architecture as a whole starting with a resolution of 16×16 from the beginning requires over 50k batches to converge far enough for generating images with the same subjective quality.

In addition to the fact that the training time can be reduced this way, the training scenario is also biologically more plausible. Biological neural networks grow and learn over time, and it is not efficient to train artificial networks with all layers from scratch for each task either. However, it is noteworthy that such architectures come with a few challenges to train. While this work implements several modular building blocks (see Section III-B and Figure 1) to allow for a more flexible and easy-to-orchestrate network-building process via loops and pre-defined data structures, it also makes the implementation more complex compared to static, non-growing networks.

We were able to achieve fast convergence with a simple binary cross-entropy loss at a resolution of 32×32, albeit with blurry output lacking high-frequency features. These findings highlight the importance of more advanced loss functions and freezing already trained layers to circumvent the convergence issues commonly faced by GANs, preventing diverging gradients. The results indicate that simple losses may be insufficient for generating visually appealing outcomes with real-world data. In a resolution of 32×32 and more often in one of 64×64, we observed smaller to medium-sized masks to be reconstructed well, but the reconstruction of others, especially larger masks, led to mosaic-like patterns that partly match in color distribution (see Figure 4). However, the potential of using a combination of losses, especially a perceptual loss (Section III-C), can be seen in Figure 3 as well as our result

Fig. 5. **Comparison of LaMa [13] and our approach on Places [15] dataset** in resolutions of 32×32 and 64×64. We used the best model "Big LaMa" for the comparison, which was also trained on the Places dataset.

comparison with LaMa [13] in Figure 5. The mix of an adversarial loss and reconstruction loss as well as the introduction of a perceptual loss in particular led to an increase in high-frequency features and sharper images, effectively reducing blurriness. In some examples, even comparatively big masks are reconstructed in a context-aware and credible fashion, which indicates that the same could be possible with further research for higher resolutions. Moreover, in some cases, our approach seems to incorporate a better scene understanding than LaMa. For example, in the third image in Figure 5 a tree is covered almost entirely with a mask on the left side of the image and while LaMa reconstructs the missing information with a high level of detail, it actually cuts the tree in half missing the context information of a tree not existing like this in reality whereas our model actually paints an entire tree into the mask, albeit with less detailed information with respect to high-frequency features.

B. Size Comparison

As stated in Section III-B, our Generator consists of approximately 330k parameters and reaches convergence for a resolution of 16×16 after 5-10k batches. Comparatively, the two GAN-based inpainting approaches introduced in Section II-A include 27M (LaMa [13]) and 6M (Iizuka et al. [12]) parameters respectively. Even the original DCGAN by Radford et al. [23], trained for an output resolution of 64×64, consists of more than 1M parameters in the re-implementation of the PyTorch-GAN[2] repository. Visually more appealing results produced by the Wasserstein GANs in the work by Shmelkov et al. [24] consist of 2.5M parameters for the smaller model and 10M parameters for the full model. Additionally, the Wasserstein GANs are trained on the CIFAR-10 [27] dataset, which has fewer classes, less variety, and is smaller in size in comparison to the Places [15] dataset.

Even beyond GAN-based approaches, diffusion models like the Denoising Diffusion Probabilistic Models (DDPM) proposed by Ho et al. [26] consist of 167M parameters. The work of Pandey et al. [25] proposes a diffusion-based variational auto-encoder (dVAE), which produces state-of-the-art, visually appealing results. Still, the VAE part of the network includes 14M parameters and the DDPM part of the network includes 84.6M parameters. Since the initial proposal of diffusion models by Sohl-Dickstein et al. [28], these types of neural architectures are often among state-of-the-art image generation models.

V. Conclusion

We present an application of progressive growing networks for image inpainting tasks with GANs, which utilizes the advantages of growing networks to tackle two challenges of GANs: training time and stable convergence. Training progressive growing networks presents challenges, such as the need to construct datasets multiple times for lower resolutions to avoid GPU under-utilization and prevent disk loading speed from becoming a bottleneck. Despite these challenges, the potential of progressive growing networks is evident. Especially for resolutions of up to 16×16, we achieved a decrease in training time until convergence and generation of visually plausible images. Additionally, since the entire network consists of just 330k parameters, it is considerably smaller than usual GANs and diffusion models, where model sizes of millions of parameters are common (see Table I and Section IV-B).

[2] https://github.com/eriklindernoren/PyTorch-GAN

The efficiency of the growing mechanism shows that a more dynamic and efficient way of building and training neural networks is possible. Furthermore it is also plausible that these underlying principles of incremental learning can, in other forms, help to explain biological learning processes.

VI. Future Work

A. Adaptive Growing

Currently, the growing process is coupled to the step-wise increase in resolution, meaning that the size of the added sub-network for the new resolution has to be estimated and empirically evaluated beforehand. Additionally, it would be more dynamic and effective to only add new layers when needed. E. g. only expanding the network when new information cannot be learned without catastrophic forgetting.

B. Anonymization

Data privacy is an essential topic in computer vision, and being able to completely remove humans from pictures is a practical application of inpainting networks. To achieve this, one could combine the training process with a detection network that detects segmentation masks for humans so that the Generator's task is not only to paste context-aware and visually plausible information into masks but also to remove humans from the picture.

ACKNOWLEDGMENT

The authors gratefully acknowledge support from the German research initiatives BMWK (SIDIMO), the DFG (CML, LeCAREbot), the European Commission (TRAIL, TERAIS) and the University of Hamburg (Excellence-Project-Funding-Initiative for student projects: "How Green Are We?"), supported by the ISA-Center of the University of Hamburg.

References

[1] O. Elharrouss, N. Almaadeed, S. Al-Maadeed, and Y. Akbari, "Image inpainting: A review," *Neural Processing Letters*, vol. 51, pp. 2007–2028, 2020.

[2] X. Jin, Y. Su, L. Zou, Y. Wang, P. Jing, and Z. J. Wang, "Sparsity-based image inpainting detection via canonical correlation analysis with low-rank constraints," *IEEE Access*, vol. 6, pp. 49 967–49 978, 2018.

[3] A. Tran and H. Tran, "Data-driven high-fidelity 2d microstructure reconstruction via non-local patch-based image inpainting," *Acta Materialia*, vol. 178, pp. 207–218, 2019.

[4] R. Xu, M. Guo, J. Wang, X. Li, B. Zhou, and C. C. Loy, "Texture memory-augmented deep patch-based image inpainting," *IEEE Transactions on Image Processing*, vol. 30, pp. 9112–9124, 2021.

[5] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," *ACM Transactions on Graphics*, vol. 28, no. 3, p. 24, 2009.

[6] G. Sridevi and S. Srinivas Kumar, "Image inpainting based on fractional-order nonlinear diffusion for image reconstruction," *Circuits Syst. Signal Process.*, vol. 38, no. 8, p. 3802–3817, August 2019.

[7] Y. Zhang, F. Ding, S. Kwong, and G. Zhu, "Feature pyramid network for diffusion-based image inpainting detection," *Information Sciences*, vol. 572, pp. 29–42, 2021.

[8] T. Alt, P. Peter, and J. Weickert, "Learning sparse masks for diffusion-based image inpainting," in *Pattern Recognition and Image Analysis*, A. J. Pinho, P. Georgieva, L. F. Teixeira, and J. A. Sánchez, Eds. Cham: Springer International Publishing, 2022, pp. 528–539.

[9] Z. Yan, X. Li, M. Li, W. Zuo, and S. Shan, "Shift-net: Image inpainting via deep feature rearrangement," in *ECCV*, 2018.

[10] H. Liu, B. Jiang, Y. Song, W. Huang, and C. Yang, "Rethinking image inpainting via a mutual encoder-decoder with feature equalizations," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 725–741.

[11] 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 (CVPR)*, June 2018.

[12] S. Iizuka, E. Simo-Serra, and H. Ishikawa, "Globally and locally consistent image completion," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–14, 2017.

[13] R. Suvorov, E. Logacheva, A. Mashikhin, A. Remizova, A. Ashukha, A. Silvestrov, N. Kong, H. Goka, K. Park, and V. S. Lempitsky, "Resolution-robust large mask inpainting with fourier convolutions," *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 3172–3182, 2022.

[14] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *6th International Conference on Learning Representations*, 2018.

[15] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 Million Image Database for Scene Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1452–1464, June 2018.

[16] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 694–711.

[17] L. Chi, B. Jiang, and Y. Mu, "Fast fourier convolution," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 4479–4488.

[18] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 3730–3738, 2015.

[19] D. Berthelot, T. Schumm, and L. Metz, "BEGAN: Boundary Equilibrium Generative Adversarial Networks," *arXiv*, 2017. [Online]. Available: http://arxiv.org/abs/1703.10717

[20] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[21] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2015.

[23] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *4th International Conference on Learning Representations (ICLR)*, Y. Bengio and Y. LeCun, Eds., May 2016.

[24] K. Shmelkov, C. Schmid, and K. Alahari, "How good is my gan?" in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[25] K. Pandey, A. Mukherjee, P. Rai, and A. Kumar, "Diffusevae: Efficient, controllable and high-fidelity generation from low-dimensional latents," *arXiv*, 2022. [Online]. Available: https://arxiv.org/abs/2201.00308

[26] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851.

[27] A. Krizhevsky, "Learning multiple layers of features from tiny images," in *Technical Report TR-2009*, 2009.

[28] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep Unsupervised Learning using Nonequilibrium Thermodynamics," in *Proceedings of the 32nd International Conference on Machine Learning*, March 2015.