Controlling the image generation process with parametric activation functions

Ilia Pavlov
Creative Computing Institute, University of the Arts London, London UK
i.pavlov@arts.ac.uk

Abstract

As image generative models continue to increase not only in their fidelity but also in their ubiquity, the development of tools that leverage direct interaction with their internal mechanisms in an interpretable way has received little attention. In this work, we introduce a system that allows users to develop a better understanding of the model through interaction and experimentation. By giving users the ability to replace activation functions of a generative network with parametric ones and a way to set the parameters of these functions, we introduce an alternative approach to control the network's output. We demonstrate the use of our method on StyleGAN2 and BigGAN networks trained on FFHQ and ImageNet, respectively.

Introduction

The field of Explainable AI (xAI) has always lagged behind the main body of machine learning research (Malizia and Paternò, 2023). This is also true for applications of generative artificial intelligence (AI) (Schneider, 2024) in the field of computational creativity and it's use as a creativity support tool (Shneiderman, 2007).

This paper proposes an interactive tool that allows users to manipulate the behavior of a generative model by modifying its structure. Such interactions have the potential to expand the non-expert user's understanding of the underlying mechanisms behind the model's output and promote an increase in AI literacy.

The tool allows users to replace static activation functions with parametric ones (Apicella et al., 2020). This gives users the ability to manually set the parameters of these functions, and thereby affect their outputs.

Background

xAI as a field aims to develop methods to explain how specific outcomes are reached by an AI model (Weller, 2019; Samek, Wiegand, and Müller, 2017). Specific techniques for models working with visual data include Saliency Maps (Petsiuk et al., 2020), Class Activation Mapping (Zhou et al., 2015), Layer-Wise Relevance Propagation (Binder et al., 2016).

Machine learning approach to generative models works by mapping a data point drawn from the latent distribution Z to a data point taken from the data distribution X (Ruthotto and Haber, 2021). In practice, distribution Z is typically a Gaussian distribution, and distribution X is real-world or simulated data, such as images, audio, or text. There are different ways one can achieve this, some examples include the following archetypes: Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), Variational Autoencoders (Kingma and Welling, 2013), and Diffusion models (Saharia et al., 2022). It is worth to note that generative models are not exclusive to neural networks and expand into evolutionary models (Cook and Colton, 2018), agent-based models (Delarosa and Soros, 2020) and others (Briot, Hadjeres, and Pachet, 2019).

The manipulation of the generative network's feature maps is not a new area of research. The method described in the Network Bending paper (Broad, Leymarie, and Grierson, 2022) uses deterministic transformations to affect the output of the network during inference. To make this more effective they introduce a procedure for clustering activation maps based on their spatial similarity. This allows for the grouping of feature maps, which correspond to different features of the generated image.

PandA (Oldfield et al., 2022) is a method for discovering factors responsible for image appearance in feature map space. This then enables users to edit the output images with pixel accuracy by modifying these factors.

The manipulation of the behaviour of generative neural networks can be also achieved by inserting small-scale neural networks in-between layers and training them for a short time, as shown in this paper (Aldegheri et al., 2023). This introduces deviations from original data distribution the network was trained on, while also maintaining an overall structure of images.

Another prominent method used to manipulate the generative network's output is latent space exploration (Klys, Snell, and Zemel, 2018; Corneanu, Gadde, and Martínez, 2024; Shen et al., 2019). There is a large variety of work, detailing various approaches to the manipulation of network's latent space. For example, papers (Bontrager et al., 2018; Fernandes, Correia, and Machado, 2020) propose the use evolutionary algorithms for this task.

Control Method

In this section, we introduce our GUI and examine the multiple parametric activation functions we experimented with.

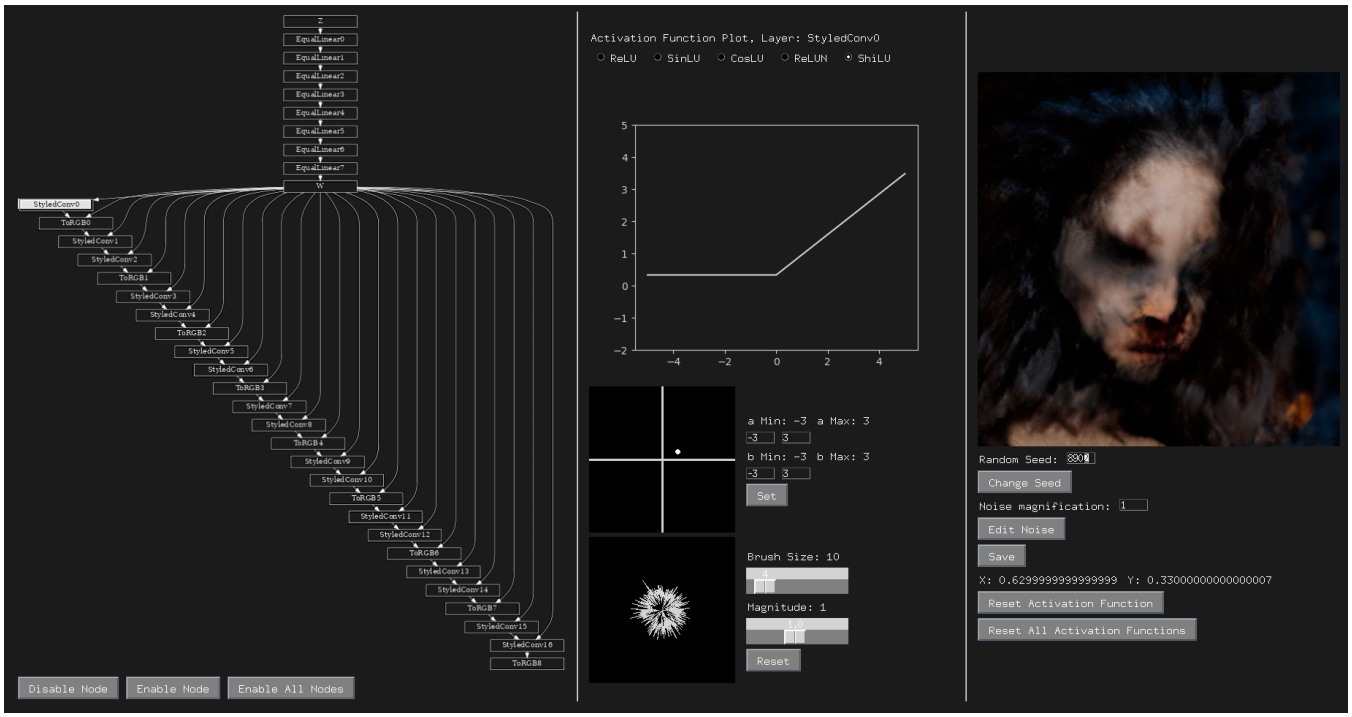

Figure 1: A Graphical User Interface was developed for our control method. The left side is responsible for picking neural layers, the center side is responsible for selecting activation functions and adjusting their parameters if applicable, and the right side allows for real-time result visualization.

The code for this paper can be found on GitHub[1] and the video demonstration of it's use is available on YouTube.[2]

Our graphical user interface, as shown in Figure 1, allows users to:

- Select the desired activation functions (left).
- Replace selected activation functions with parametric ones (center-top).
- Adjust the parameters of injected activation functions (center-bottom).
- Have a real-time feedback (right).

The GUI also has features for editing the input latent vector (center-bottom) and disabling specific layers outright (left-bottom).

All these features provide control over the structure of the network and showcase how different changes affect the outcome.

Activation functions

The Sinu-Sigmoidal Linear Unit (Paul et al., 2022) has two parameters, a and b. The parameter a is responsible for the amplitude of the sine function, while the parameter b is responsible for the frequency of the sine function. Setting any of these parameters too high can lead to unpredictable results, which could be preferable for more abstract image generation. The full formula for the activation function is shown in (1), where $\sigma(x)$ is a sigmoid function.

$$SinLU(x) = (x + a * sin(b * x)) * \sigma(x) \tag{1}$$

Rectified Linear Unit N (Pishchik, 2023) or ReLUN is a modified version of the ReLU6 activation function with one parameter, which controls the angle of the function. The full formula for the activation function is shown in (2).

$$ReLUN(x) = min(max(0, x), n) \tag{2}$$

Shifted Rectified Linear Unit (Pishchik, 2023) or ShiLU is a modified version of ReLU (Fukushima, 1969) function that has two parameters a and b. Parameter a is responsible for the slope of the function, and parameter b is responsible for shifting the function along the Y-axis. The full formula for the activation function is shown in (3).

$$ShiLU(x) = a * ReLU(x) + b \tag{3}$$

Usage of such functions causes alterations in the output by changing the intermediate feature maps (Zeiler and Fergus, 2013) of the network during the inference process. Using these functions in multiple layers will cause a cascading effect of changes. This in turn can lead to a large amount of possible divergences from initial neural network state.

We also experimented with polynomial activation functions. Using polynomial functions as activation functions in neural networks has been proposed before, but never gained

[1] https://github.com/locsor/generativeControlUI

[2] https://youtu.be/qkP9DHLicwM

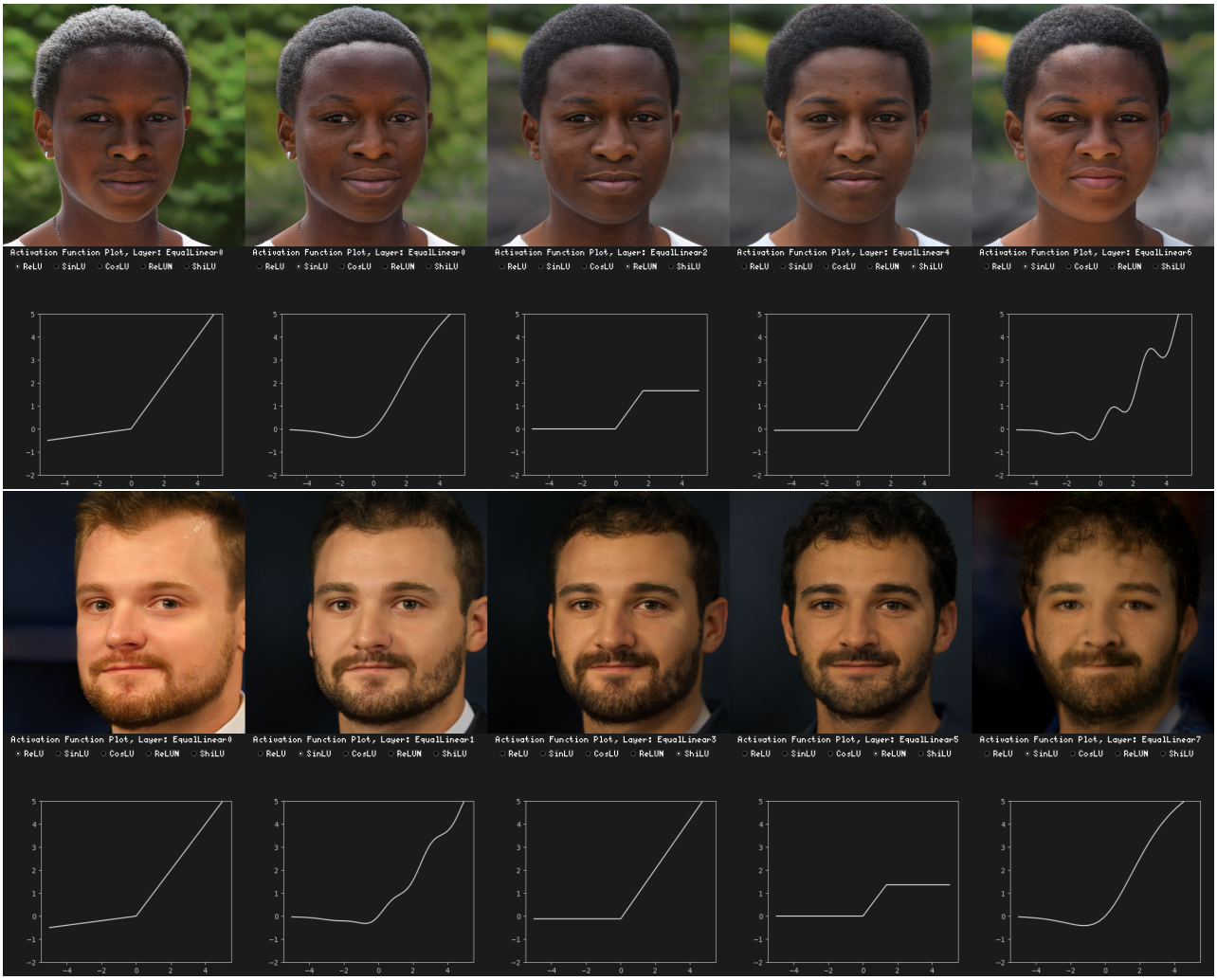

Figure 2: Examples of using parametric activation functions in the mapping network of the StyleGAN2. The left-most images in output rows (faces) are base outputs. Each image has one more base activation function replaced with a parametric one, compared with an image to its' left. Bottom rows show plots of parametric functions, which were used to produce the images above.

much popularity, due to the exploding gradient problem and increased complexity of training. The specific polynomial activation functions we used have been taken from a retracted paper (Gottemukkula, 2019). The general formula is shown in (4).

$$f_n(x) = \frac{\sum_{i=1}^{n} w_i \cdot \sigma(x)^i}{\sqrt{2}^n} = \frac{w_0 \cdot \sigma(x)^0 + w_1 \cdot \sigma(x)^1 + \ldots + w_n \cdot \sigma(x)^n}{\sqrt{2}^n} \quad (4)$$

Where each weight coefficient w_i in this function is a potential control parameter in our system and $\sigma(x)$ is a sigmoid function.

Experiments

For base networks, we used the StyleGAN2 (Karras et al., 2019) and BigGAN (Brock, Donahue, and Simonyan, 2018) models. The choice was motivated by their ubiquity in generative neural network research, stability, and speed.

The architecture of StyleGAN2 is split into two parts: a mapping network and a generator network. The purpose of a mapping network is to disentangle the input latent vector. The generator network consists of a series of convolutional blocks, each outputting an image with progressively increasing resolution. To get the final image, these intermediate images are scaled up and summed. Applying the parametric activation functions to StyleGAN2 had the following results:

- Using them in the mapping network affected the image structure. These structural changes are due to parametric activation functions affecting the resulting output of the mapping network, i.e. the disentangled latent vector.

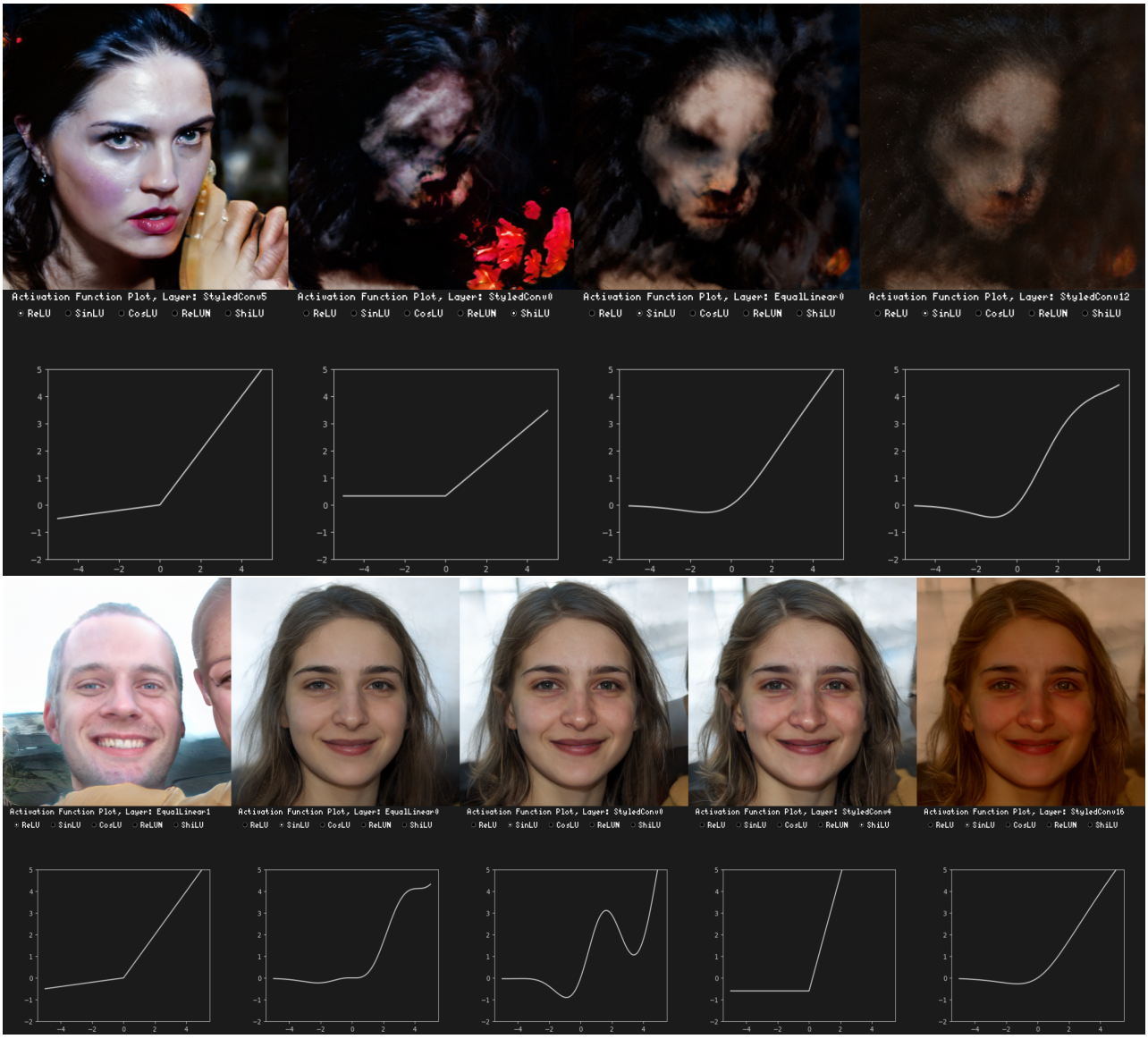

Figure 3: Examples of using parametric activation functions in both mapping and generator networks of the StyleGAN2. The left-most images in output rows (faces) are base outputs. Each image has one more base activation function replaced with a parametric one, compared with an image to its' left. Bottom rows show plots of parametric functions, which were used to produce the images above.

- Earlier layers of the mapping network offered a high level of granularity with respect to changes in the image.
- Using them in the generator network affected both style and, in cases of earlier layers, the structure of images.
- Later layers of the generator network caused changes to images in a more basic manner, such as altering the overall coloration of the image.

As shown in our examples in Figures 2 and 3, the usage of controllable activation functions can be a viable way to control the image generation process. Slight changes to the parameters of these functions will lead to slight changes in the resulting image, while larger changes lead to more significant and unpredictable alterations to both the structure and style of the image. For example, a user can infer that the latter quality is not desirable for generating realistic images, it can allow for the generation of more abstract imagery.

BigGAN is a classic example of a GAN architecture but scaled up to improve both fidelity and variety of generated images. For our experiment, we used a class-conditional implementation of this network. While the changes in images are not as varied as those observed with StyleGAN2, they still demonstrate the ability to alter the content of the image to some extent. Examples of images generated using BigGAN are shown in Figures 4 and 5.

Figure 4: Output of a BigGAN model. The left-most image is an original, the rest were generated with SinLU or ReLUN applied to the second layer of the network. The parameter values were randomly chosen.

Figure 5: Output of a BigGAN model with 3rd order polynomial activation functions. The functions' parameters were kept within $[0.8, 1.2]$ interval. The left-most images are originals, the right-most are generated with 4 modified functions, and the rest were generated with only one modified function.

Polynomial functions were not as encouraging. Using them in more than 2 neural layers yielded non-ideal results, and they were extremely sensitive to small changes in parameter values. Examples of images generated by the BigGAN model modified with a 3rd degree polynomial activation functions are shown in Figure 5. We limited the parameter values to $[0.5, 1.5]$.

Overall, in a standard use case, a user will gradually modify the network and carefully select the values of the control parameters to achieve a satisfactory result. Since our method does not offer any way to single out any specific feature of the image, the process largely consists of trial-and-error experimentation. Through continued use and real-time visual feedback, users can learn how the structural state of the network influences the output of the generation process. This process is mainly aimed at incidental exploration, instead of a guided image generation.

Conclusion

In summary, this paper introduces a novel approach to control image generative networks, with the help of parametric activation functions, and examines how this tool can be used to teach non-expert users the importance of its various parts. But to confirm these speculations, a user study will need to be conducted.

The main limitation of the current implementation is, that unguided control over image generation is imprecise and limiting. While applying this method to the text-to-image generative network can alleviate this problem, there are still no guarantees that users will be able to find a satisfactory collection of activation functions and their parameters.

References

Aldegheri, G.; Rogalska, A.; Youssef, A.; and Iofinova, E. 2023. Hacking generative models with differentiable network bending. *ArXiv* abs/2310.04816.

Apicella, A.; Donnarumma, F.; Isgrò, F.; and Prevete, R. 2020. A survey on modern trainable activation functions. *Neural networks : the official journal of the International Neural Network Society* 138:14–32.

Binder, A.; Montavon, G.; Lapuschkin, S.; Müller, K.-R.; and

Samek, W. 2016. Layer-wise relevance propagation for neural networks with local renormalization layers. *ArXiv* abs/1604.00825.

Bontrager, P.; Lin, W.-C.; Togelius, J.; and Risi, S. 2018. Deep interactive evolution. *ArXiv* abs/1801.08230.

Briot, J.-P.; Hadjeres, G.; and Pachet, F. 2019. Computational creativity: The philosophy and engineering of autonomously creative systems. *Computational Creativity*.

Broad, T.; Leymarie, F. F.; and Grierson, M. 2022. Network bending: Expressive manipulation of generative models in multiple domains. *Entropy* 24(1).

Brock, A.; Donahue, J.; and Simonyan, K. 2018. Large scale gan training for high fidelity natural image synthesis. *ArXiv* abs/1809.11096.

Cook, M., and Colton, S. 2018. Redesigning computationally creative systems for continuous creation. In *International Conference on Innovative Computing and Cloud Computing*.

Corneanu, C. A.; Gadde, R.; and Martínez, A. M. 2024. Latentpaint: Image inpainting in latent space with diffusion models. *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* 4322–4331.

Delarosa, O., and Soros, L. B. 2020. Growing midi music files using convolutional cellular automata. *2020 IEEE Symposium Series on Computational Intelligence (SSCI)* 1187–1194.

Fernandes, P.; Correia, J.; and Machado, P. 2020. Evolutionary latent space exploration of generative adversarial networks. In *EvoApplications*.

Fukushima, K. 1969. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics* 5(4):322–333.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A. C.; and Bengio, Y. 2014. Generative adversarial networks. *Communications of the ACM* 63:139 – 144.

Gottemukkula, V. 2019. Retracted: Polynomial activation functions.

Karras, T.; Laine, S.; Aittala, M.; Hellsten, J.; Lehtinen, J.; and Aila, T. 2019. Analyzing and improving the image quality of stylegan. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 8107–8116.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *CoRR* abs/1312.6114.

Klys, J.; Snell, J.; and Zemel, R. S. 2018. Learning latent subspaces in variational autoencoders. *ArXiv* abs/1812.06190.

Malizia, A., and Paternò, F. 2023. Why is the current xai not meeting the expectations? *Commun. ACM* 66(12):20–23.

Oldfield, J.; Tzelepis, C.; Panagakis, Y.; Nicolaou, M. A.; and Patras, I. 2022. Panda: Unsupervised learning of parts and appearances in the feature maps of gans. *ArXiv* abs/2206.00048.

Paul, A.; Bandyopadhyay, R.; Yoon, J. H.; Geem, Z. W.; and Sarkar, R. 2022. Sinlu: Sinu-sigmoidal linear unit. *Mathematics* 10(3).

Petsiuk, V.; Jain, R.; Manjunatha, V.; Morariu, V. I.; Mehra, A.; Ordonez, V.; and Saenko, K. 2020. Black-box explanation of object detectors via saliency maps. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 11438–11447.

Pishchik, E. 2023. Trainable activations for image classification. *Preprints*.

Ruthotto, L., and Haber, E. 2021. An introduction to deep generative modeling. *GAMM-Mitteilungen* 44.

Saharia, C.; Chan, W.; Saxena, S.; Li, L.; Whang, J.; Denton, E. L.; Ghasemipour, S. K. S.; Ayan, B. K.; Mahdavi, S. S.; Lopes, R. G.; Salimans, T.; Ho, J.; Fleet, D. J.; and Norouzi, M. 2022. Photorealistic text-to-image diffusion models with deep language understanding. *ArXiv* abs/2205.11487.

Samek, W.; Wiegand, T.; and Müller, K.-R. 2017. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *ArXiv* abs/1708.08296.

Schneider, J. 2024. Explainable generative ai (genxai): A survey, conceptualization, and research agenda. *ArXiv* abs/2404.09554.

Shen, Y.; Gu, J.; Tang, X.; and Zhou, B. 2019. Interpreting the latent space of gans for semantic face editing. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* 9240–9249.

Shneiderman, B. 2007. Creativity support tools: accelerating discovery and innovation. *Commun. ACM* 50:20–32.

Weller, A. 2019. Transparency: Motivations and challenges. In *Explainable AI*.

Zeiler, M. D., and Fergus, R. 2013. Visualizing and understanding convolutional networks. *ArXiv* abs/1311.2901.

Zhou, B.; Khosla, A.; Lapedriza, À.; Oliva, A.; and Torralba, A. 2015. Learning deep features for discriminative localization. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 2921–2929.