Towards Kinetic Manipulation of the Latent Space

Diego Porres
Computer Vision Center (CVC) & Universitat Autònoma de Barcelona
Barcelona, Spain, 08193
dporres@cvc.uab.es

Abstract

The latent space of many generative models are rich in unexplored valleys and mountains. The majority of tools used for exploring them are so far limited to Graphical User Interfaces (GUIs). While specialized hardware can be used for this task, we show that a simple feature extraction of pre-trained Convolutional Neural Networks (CNNs) from a live RGB camera feed does a very good job at manipulating the latent space with simple changes in the scene, with vast room for improvement. We name this new paradigm *Visual-reactive Interpolation*, and the full code can be found at https://github.com/PDillis/stylegan3-fun.

1 Motivation and Background

Interaction with generative models such as Generative Adversarial Networks (GANs) [GPAM+14] and diffusion-based text-to-image models (T2I) [SWMG15] has been relinquished to the land of software, specifically Graphical User Interfaces (GUIs). The main advantage of using these is the speed at which they can be modified and shared between users, in addition to simplifying interaction with the model, including it's high-dimensional latent space. However, we believe this also introduces a rift between the generative models and their users, if not also with their creators, as we forego the most basic interface: the human body.

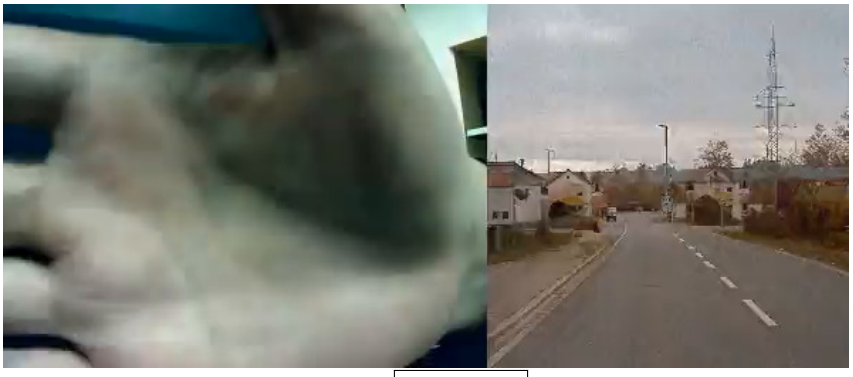
Indeed, [Fre08] notes, among other things: firstly, human beings possess a 'centric' quality (they interact with their environment from the vantage point of a *centre*); secondly, we shouldn't separate human bodies from the minds, but as acting mind-body unities; and thirdly, the body is instrumental in communication both within the self as with the environment (be it motor or hormonal activity). Our goal is to create a system by which we can have a live performer be the one controlling the latent space of a generative model with their body or facial movements and scenery changes via moving objects in the scene.

We argue then, that there are two *centers* on an image, whether real or synthesized: the subject or focus of the image, and the camera (real or virtual). As such, we should then aim at letting the camera controller to be another actor in the image synthesizing pipeline by changing the focused area of the performer, lightning, or even camera lenses. For our purposes, we wish the two actors to interact through a manipulation of the latent space, made visible via the synthesized images that a pre-trained Generator of a GAN will provide. This will allow the two to forge a common story. This work aims to document the creation of this tool.

2 Latent Space Interaction

Since our aim is to control the image synthesis generation during a live performance show, we will quickly explore other already existing alternatives. Concretely, we will focus our attention to manipulate the latent space of StyleGAN1/2/2-ADA [KLA18, KLA+19, KAH+20], some of the most popular and widely trained networks, but we will look at inspiration for what has been done

38th Conference on Neural Information Processing Systems (NeurIPS 2024).



Play/Pause

Figure 1: **Test 1:** Visual-reactive live demo using a StyleGAN2 model (512×512 resolution) trained on urban scenes from the A2D2 dataset [GKM+20]. We use style mixing with a static latent, and the encoded camera image (resolution 426×320) will procure the `coarse` and `middle` noise scales. Note the video control above does not work in browsers, but works fine with Adobe Acrobat. Click here for an online version of the video.

in others as well. Our focus is on GANs as as their sampling cost is low compared to other models, which translates to a smoother live performance, but hopefully this work (or an iteration of it) can be also applied to other generative model such as T2I latent diffusion models [RBL+22].

There have been many recent efforts towards enhancing the interaction with generative models, both offline as well as during live shows. Most notably, [Bro20] uses and exploits the architecture of StyleGAN1/2 (noise injection, network bending [BLG20], style mixing, among others), as well as the features of music itself for creating enthralling audio-reactive latent interpolations. This allows for expressive coverage of the latent space and sets the stage for further expanding the our pipeline's capacities.

[Smi20] questions the interactive capabilities of a mouse and keyboard when trying to explore interactions with trained GANs. A such, they program a MIDI controller to guide StyleGAN1/2/2-ADA, selecting different controllers to change local or global variables, such as the truncation factor ψ or for quick sampling of the latent space. While this requires adding hardware and its respective programming, it highlights the general need for better tools and methods of the to explore the latent space of generative models. Likewise, text (or speech) can be used to interact with generative models on how to edit the synthesized 2D image [SOB+23] or 3D animation [HTF+24].

On the other hand, [Bur21] seeks to project a live video feed into the disentangled latent space $\mathcal{W}$ of StyleGAN. A camera feed is iteratively projected to $\mathcal{W}$ and then used to synthesize an image, showcasing the expressivity of the network. However, we must caution that this process has a model selection bias, as FFHQ's latent space is perhaps the only model where we can effectively project any image [AQW19].

Independently of live shows, tools such as GANSpace [HHLP20] allow for automatically creating interpretable controls in the latent space of StyleGAN or BigGAN [BDS18] by identifying important latent directions using PCA. This method is limited to static sliders that are not always the directions the user wishes to move towards, but we note that these could be mapped to different actions in the scene. DragGAN [PTL+23] and DragDiffusion [SXP+23] seek to give more precise editing control to the user by adding points to "drag". However, the movement of these points is an optimization, increasing the compute cost of this approach.

For real-time generation, Xoromancy [Cra19] has explored hand-based latent space manipulation. However, it relies on specialized hardware (Leap Motion) and fixed camera placement, limiting its accessibility and flexibility. In the same vein, the art installation *Fencing Hallucination* [QL23] uses a Kinect to accurately extract the pose of the user as part of its pipeline. We argue that it is possible to eliminate the need for specialized equipment by using readily available pre-trained models and affordable cameras, in turn broadening the scope of interaction with generative models and democratizing the access to latent space manipulation.

Furthermore, by removing the constraint of fixed camera positioning, we introduce the camera itself as a "second actor" in the image generation pipeline, opening up new possibilities for performative art where both the subject and the camera operator can collaboratively influence the generated output, effectively turning the computer as a *peer* and not just as a *sub-contractor* [LR23]. The flexibility of our approach extends to its compatibility with a wide range of pre-trained GANs and in the future to Text-to-Image (T2I) Diffusion Models [SWMG15, RBL+22] or Consistency Models [SDCS23, LTH+23], enhancing its applicability and potential for creative expression.

3 Visual-reactive Interpolation

Given that StyleGAN lacks an encoder back into its latent space $\mathcal{Z}$, unlike other architectures such as BiGAN [DKD16], we must look at alternatives. Although using the last Fully Connected layer of the GAN's Discriminator has been proposed for this purpose as it has the desired dimensionality and is essentially the same network [Por21], our tests have shown that the resulting encoding fails to cover the majority of the latent space, resulting in nigh-static images.

Training an encoder for each available pre-trained GAN is out of the question, so we opt to use pre-trained feature extractors F. As such, we will use the intermediate representations of these, effectively acting as encoders into the latent space $\mathcal{Z}$. We discard using SOTA models such as CLIP [RKH+21], as we wish to minimize the footprint of this part of the model as much as possible. Likewise, Vision Transformers (ViT) [DBK+20] have shown great promise in image classification tasks, but our experiments show that almost no benefit is seen when used in our pipeline. For now, we gear towards VGG16 [SZ15], as its intermediate representations have some desired characteristics, but note that future work can make use of more recent architectures that are lighter to run on the edge.

We start our testing by exploiting these feature extractors as well as the style-mixing characteristic of StyleGAN. Based on user feedback, we then move to more fine-grained control of specific parts of the Generator. Lastly, we list some possible venues to take, but note that the possibilities are endless, especially if we include multimodal inputs to our setting.

3.1 Test 1: Visual Encoding and Style Mixing

In the following, we will use the notation of [GEB15]. We denote by x the frame captured by a camera C that we will feed to the selected feature extractor F. When we pass each frame through this network, it will be encoded at layer l as $F^l \in \mathbb{R}^{N^l \times M^l}$,where N^l denotes the number of feature maps of size M^l. The size refers to the product of the height h^l and width w^l of the feature maps, so $M^l = h^l \cdot w^l$, with $h^l, w^l \in \mathbb{Z}^+$. We must convert these feature maps into a latent, for which we make use of a function g. For more details, see Appendix A.

The dimensions of x must also be carefully selected, as this will also affect the real-time inference that can be achieved by the available compute. Note that the camera C is not limited to being an RGB camera. Indeed, this technique works with any other type of sensor, so long as a pre-trained network F is available (or equivalent data pipeline). The full pipeline for this process is shown in Figure 2 and a demo is shown in Figure 1. The input image has been encoded to follow a strict $4:3$ aspect ratio, but this can be modified.

User Feedback Initial user testing at the *ExperimentAI 2023* and the *Festa de la Ciència 2023* revealed that users were intrigued by this new interaction method, but they quickly became disengaged due to the lack of fine-grained control. Users found that changing objects in the scene sometimes resulted in camera autofocusing, reverting the generated image to its original state. This feedback prompted us to explore more precise manipulation techniques in our subsequent tests.

3.2 Test 2: Manipulation of Learned Constants

Based on user feedback from Test 1 (Section 3.1) indicating a need for more fine-grained manipulation, we turned our attention to manipulating specific parts of the Generator, particularly some of the learned constants. During training, StyleGAN2 learns a constant parameter (`G.synthesis.b4.const`) that will aid it in positioning certain aspects of the generated image (eyes, nose, ears, hairs when tasked with mimicking faces, for example). If we corrupt this, then the network will lose track of where to

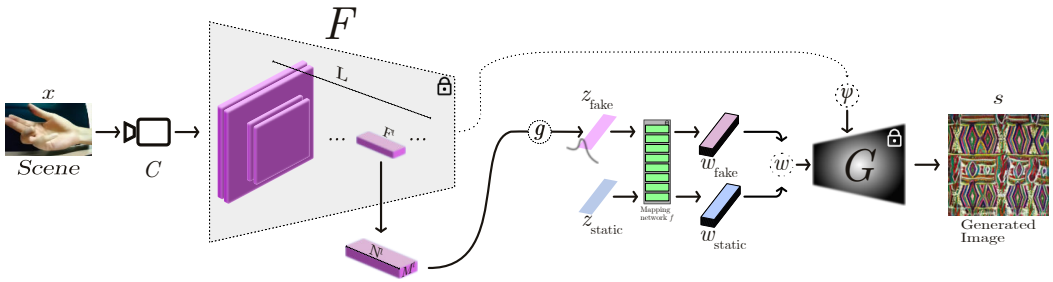


Figure 2: **Test 1 pipeline.** A frame x of a scene is captured with a camera C, which is then fed to the feature extractor F. We select F to contain L convolutional layers. At layer l, the representation of x will be F^l. We pass this representation through our selected function g, turning it into a latent vector z_{fake}, which is then passed to the (frozen) mapping network f of the Generator. To perform style-mixing, we can use a static latent z_{static}, to produce a final disentangled latent vector w. It, along the truncation parameter ψ, will be used by G to generate the final synthesized image s. Note that ψ could also be influenced or controlled via the encoded scene.

place the eyes, for example, in turn generating multiple pairs of them. This is essentially what Flesh Digressions does, which we adapt to our setting.

A simple way to corrupt this constant is by extracting keypoints from one or more human body parts using MediaPipe or MMpose. We can define the center of the image as the learned constant and the average distnace of the selected body part will define the amount of corruption we introduce to this constant. We show a demo in Figure 3.

On the other hand, StyleGAN3 [KAL+21] learns a fixed affine transform to apply to the latent space in order to correctly translate (and optionally rotate) the generated objects. This affine transform can also be corrupted in a more intuitive way: for example, by calculating the angle between the vertical axis and the middle finger, we can set how much to rotate the image. The distance form the hand to the center of the image can set how much to scale the image. We showcase a demo in Figure 4.

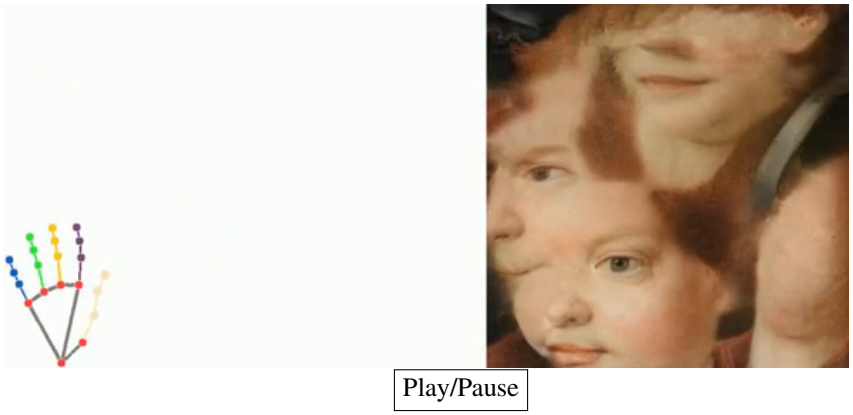
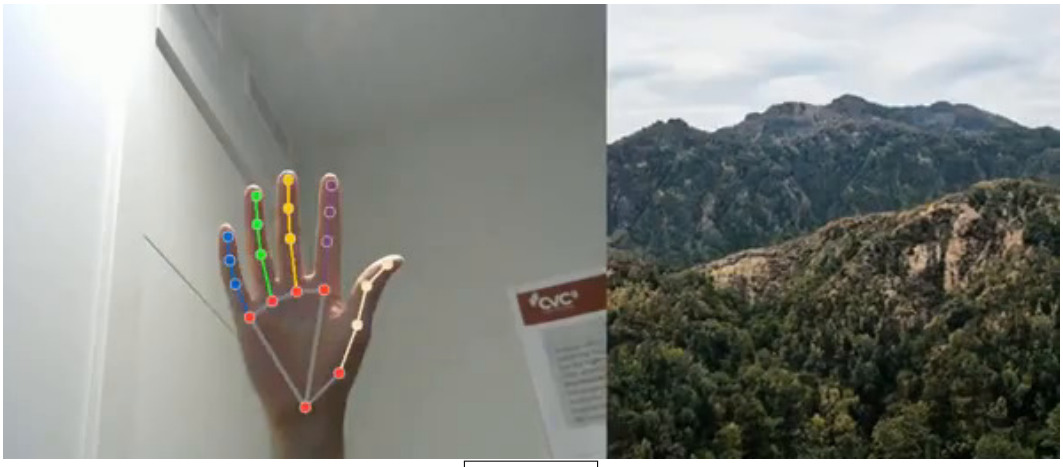


Figure 3: **Test 2:** Live manipulation of the learned constant in StyleGAN2. Note the video control above does not work in browsers, but works fine with Adobe Acrobat. Click here for an online version of the video.

User Feedback Originally, we had intended to use two hands to perform these corruptions. However, user testing at the *Festa de la Ciència 2023* revealed significant challenges with this approach. While the system ran smoothly with one hand, the introduction of a second hand caused confusion. Despite MediaPipe's ability to distinguish between left and right hands, the active hand controlling the movement could unpredictably switch between left and right when both were detected. This led to a counterintuitive experience, disrupting users' spatial understanding and control. To address these issues, we ultimately opted for a single-hand approach. This decision resulted in much cleaner and better results, as well as maintaining users' interest in exploring how to manipulate these systems.



Play/Pause

Figure 4: **Test 2:** Manipulating the learned affine transformation matrix in StyleGAN3. Note the video control above does not work in browsers, but works fine with Adobe Acrobat. Click here for an online version of the video.

4 Future Work

While the code is readily available at `https://github.com/PDillis/stylegan3-fun`, we note that it will be changing constantly as we introduce more options to the user, as well as perform more tests with it. The main bulk of work will be dedicated to port all of the previous tests to the GUI provided in the StyleGAN3 repository.

Beyond run-time optimization, the following ideas are planned to be explored to some extent in the coming months, or at least variations of them. We note, however, that the proposed parameters to manipulate via visual-reactive interpolation are interchangeable, and even more can be exploited from each model:

- Replace VGG16 with smaller footprint networks such as MobileNetV3 [HSC+19] or EfficientNet-B0 [TL19].
- Conversely, use *self-supervised* visual features such as DINOv2 [ODM+23].
- Use monocular depth estimation or optical flow estimation for manipulating the truncation trick parameter ψ via e.g. the normalized average depth of the scene using pre-trained models such as Depth-Anything-V2-Small [YKH+24].
- ψ can be thought of as controlling the expressiveness of the generator G. We can place virtual objects (such as disks) in a scene and allow users to manipulate them with their hands. By calculating the total kinetic energy $K = \sum_i m_i \|\mathbf{v}_i\|^2$, we can dynamically adjust ψ and thus the expressivity of G.
- Use semantic segmentation or classification models for exploiting class-conditional StyleGAN models, whether for manipulating the latent vectors or the class vectors themselves.
- Using PCA to extract notable directions in the latent space, as done in GANSpace [HHLP20], and move towards/away from them using specific facial gestures.
- Add audiorreactive capabilities [Bro20], effectively generating ***audiovisual***-reactive interpolations, allowing for a live band to join the performance.
- Add network bending [BLG20] capabilities to the extracted features, or for specific regions in the image.
- Extend this work to T2I Diffusion Models [SWMG15, RBL+22], with a focus on Consistency Models [SDCS23, LTH+23] for real-time manipulation.

References

[AQW19] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? *CoRR*, abs/1904.03189, 2019.

[BDS18] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. *CoRR*, abs/1809.11096, 2018.

[BLG20] Terence Broad, Frederic Fol Leymarie, and Mick Grierson. Network bending: Manipulating the inner representations of deep generative models. *CoRR*, abs/2005.12420, 2020.

[Bro20] Hans Brouwer. Audio-reactive latent interpolations with stylegan. In *Proceedings of the 4th Workshop on Machine Learning for Creativity and Design at NeurIPS 2020*, 12 2020.

[Bur21] Grigore Burloiu. GANs: interactive inference, 2021. [Online; accessed September 15th, 2024].

[Cra19] Gray Crawford. Developing Embodied Familiarity with Hyperphysical Phenomena, 7 2019. [Online; accessed September 15th, 2024].

[DBK+20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.

[DKD16] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *CoRR*, abs/1605.09782, 2016.

[Fre08] Peter Freund. The expressive body: A common ground for the sociology of emotions and health and illness. *Sociology of Health and Illness*, 12:452 – 477, 06 2008.

[GEB15] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, 2015.

[GKM+20] Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S. Chung, Lorenz Hauswald, Viet Hoang Pham, Maximilian Mühlegg, Sebastian Dorn, Tiffany Fernandez, Martin Jänicke, Sudesh Mirashi, Chiragkumar Savani, Martin Sturm, Oleksandr Vorobiov, Martin Oelker, Sebastian Garreis, and Peter Schuberth. A2D2: Audi Autonomous Driving Dataset, 2020.

[GPAM+14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[HHLP20] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable GAN controls. *CoRR*, abs/2004.02546, 2020.

[HSC+19] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. *CoRR*, abs/1905.02244, 2019.

[HTF+24] Han Huang, Fernanda De La Torre, Cathy Mengying Fang, Andrzej Banburski-Fahey, Judith Amores, and Jaron Lanier. Real-time animation generation and control on rigged models via large language models, 2024.

[KAH+20] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *CoRR*, abs/2006.06676, 2020.

[KAL+21] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *CoRR*, abs/2106.12423, 2021.

[KLA18] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018.

[KLA+19] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. *CoRR*, abs/1912.04958, 2019.

[LR23] Zhiyu Lin and Mark Riedl. An ontology of co-creative ai systems, 2023.

[LTH+23] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *ArXiv*, abs/2310.04378, 2023.

[ODM+23] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.

[Por21] Diego Porres. Discriminator synthesis: On reusing the other half of generative adversarial networks. *CoRR*, abs/2111.02175, 2021.

[PTL+23] Xingang Pan, Ayush Tewari, Thomas Leimkühler, Lingjie Liu, Abhimitra Meka, and Christian Theobalt. Drag your gan: Interactive point-based manipulation on the generative image manifold. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Proceedings*, SIGGRAPH '23, page 1–11. ACM, July 2023.

[QL23] Weihao Qiu and George Legrady. Combating the "sameness" in ai art: Reflections on the interactive ai installation fencing hallucination. *ArXiv*, abs/2311.17080, 2023.

[RBL+22] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, June 2022.

[RKH+21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021.

[SDCS23] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, 2023.

[Smi20] Cameron Smith. Livegan: Programming a hardware midi controller for interacting with generative models. In *Proceedings of the 4th Workshop on Machine Learning for Creativity and Design at NeurIPS 2020*, 12 2020.

[SOB+23] Junichi Shimizu, Ireti Olowe, Terence Broad, Gabriel Vigliensoni, Prashanth Thattai Ravikumar, and Rebecca Fiebrink. Interactive machine learning for generative models. In *NeurIPS 2023 Workshop on Machine Learning for Creativity and Design*, December 2023.

[SWMG15] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *CoRR*, abs/1503.03585, 2015.

[SXP+23] Yujun Shi, Chuhui Xue, Jiachun Pan, Wenqing Zhang, Vincent YF Tan, and Song Bai. Dragdiffusion: Harnessing diffusion models for interactive point-based image editing. *arXiv preprint arXiv:2306.14435*, 2023.

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[TL19] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019.

[YKH+24] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv:2406.09414*, 2024.

Appendix

A Feature extraction

Transformation into $\mathcal{Z}$ After capturing the internal representations, we wish to convert these into useful objects that the Generator may then use: latent vectors. We are then interested in a family of functions $\mathcal{G}$ that will transform our intermediate representation F^l into a vector z_{fake}, i.e., turn the features into *fake* latent vectors to feed our Generator G. We carefully select this family of functions such that,

$$\mathcal{G} \ni g : F^l \to \mathcal{Z} \sim \mathcal{N}(0, \mathbb{I}) \tag{1}$$

Afterwards, the mapping function f of StyleGAN (`G.mapping`) will be in charge of mapping these "fake latents" into the disentangled latent space $\mathcal{W}$. For example, at its most basic, g could simply be the channel-wise average of the feature representation F^l of VGG16. More generally, we can also do a weighted average of different layers l (as is done in the content loss in neural style transfer [GEB15]), e.g.:

$$g(F^l) = \alpha_l \sum_l \left(\frac{1}{M^l} \sum_{i,j} F^l_{ij} \right) \tag{2}$$

where F^l_{ij} denotes the i-th filter of layer l at position j, and α_l is the weight given to the representation at layer l. Our experiments so far have shown that using a single layer is sufficient, but leave the possibility to use more to each individual case.

We note this could limit the expressiveness of our generator G, so we could ensure the condition in Equation 1 is met. However, we have found that Equation 2 is sufficient, mostly because of the selection of VGG16 and its separation of features in the last layers is channel-wise, which matches the dimensionality of the default latent space $\mathcal{Z}$ of StyleGAN.

Layer selection The default dimensionality of StyleGAN's latent space is $|\mathcal{Z}| = 512$. Then, for simplicity, we are interested in the layers that have $N^l = |\mathcal{Z}| = 512$. Table 1 shows the convolutional layers in VGG16 that share this property, but note that our code is general enough that *any* layer (convolutional or not) in VGG16 can be used to obtain z_{fake}. The user should experiment and select the best option for their particular setup.

Table 1: Feature representations names and shapes for an 256×256 RGB image passed through a VGG16 network. We showcase here only those for which $N^l = |\mathcal{Z}| = 512$.

Layer Name	$\mathbf{N^l}$	$\mathbf{M^l}$
`conv4_1`	512	32×32
`conv4_2`	512	32×32
`conv4_3`	512	32×32
`conv5_1`	512	16×16
`conv5_2`	512	16×16
`conv5_3`	512	16×16
`adavgpool`	512	7×7

Image Synthesis To make better use of the expressivity of the trained StyleGAN, we can partition F^l into different sections before passing it into g, each producing a different disentangled latent vector w. In Figure 1, we used w_{coarse}, w_{middle}, and w_{fine} to refer to the disentangled latent vectors obtained from the top, bottom left, and bottom right parts of the input image x. Furthermore, we can set a static latent vector w_{static} and use the style-mixing properties of StyleGAN to more effectively control different aspects of the synthesized image. For example, one region of the input image will control the fine details (colors) of the synthesized image (w_{fine}), while in others control larger structures (w_{coarse} and w_{middle}).