GENERALIZING FACTORIZATION OF GANS BY CHARACTERIZING CONVOLUTIONAL LAYERS

Yuehui Wang *Qing Wang* *Dongyu Zhang* *

Sun Yat-Sen University

2022 IEEE International Conference on Multimedia and Expo (ICME) | 978-1-6654-8563-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICME52920.2022.9859692

ABSTRACT

Existing unsupervised disentanglement methods in latent space of the Generative Adversarial Networks (GANs) rely on the analysis and decomposition of pre-trained weight matrix. However, they only consider the weight matrix of the fully connected layers, ignoring the convolutional layers which are indispensable for image processing in modern generative models. This results in the learned latent semantics lack interpretability, which is unacceptable for image editing tasks. In this paper, we propose a more *generalized* closed-form factorization of latent semantics in GANs, which takes the convolutional layers into consideration when searching for the underlying variation factors. Our method can be applied to a wide range of deep generators with just a few lines of code. Extensive experiments on multiple GAN models trained on various datasets show that our approach is capable of not only finding semantically meaningful dimensions, but also maintaining the consistency and interpretability of image content.

Index Terms— Latent Semantic Interpretation, Generative Adversarial Network, Image Synthesis, Deep Learning

1. INTRODUCTION

Nowadays, Generative Adversarial Networks (GANs) [2] have become a leading paradigm of generative modeling in the computer vision domain. The stateof-the-art GANs like BigGAN [3] and StyleGAN [1, 4], are powerful image synthesis models that can generate a wide variety of high-quality images. The exceptional generation quality paves the road to ubiquitous usage of GANs in applications, *e.g.*, image editing [5, 6], super-resolution [7] and many others.

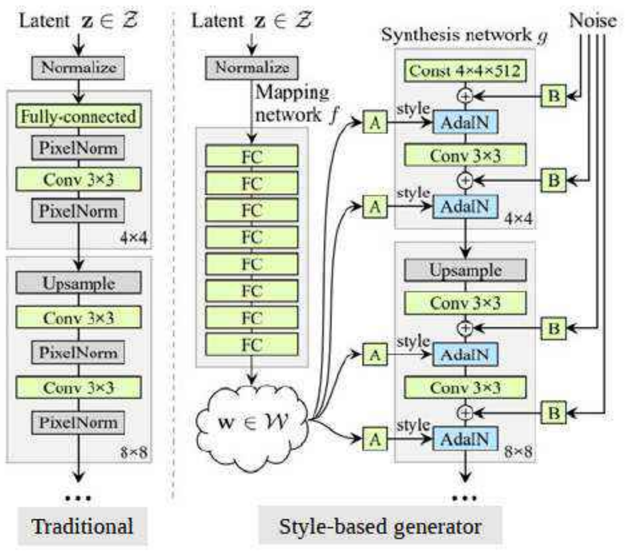
Given their powerful capabilities, current research interest is shifted to generating controllable images, some of previous works attempt to add user control over the output focus on supervised learning of latent directions with labeled images. To be specific, [8, 9] propose to add regularizers into the training process to explicitly learn an interpretable factorized representation. However, they heavily rely on the attribute predictors to get the label and require expensive manual supervision for each new control to be learned. Another line of researchers study unsupervised semantic discovery in GANs [10–12], Specifically, [10] jointly learn a candidate matrix and a classifier such that the semantic directions in the matrix can be properly recognized by the classifier. [11] perform PCA on the sampled data to find primary directions in the latent space. [12] studies the generation mechanism of GANs and propose a closed-form factorization method by analyzing the eigenvalue decomposition of the pre-trained weight matrix, which is independent of training or sampling.

The above papers need to analyze and decompose the pre-trained weight matrix, but they only consider the fully connected layers. Convolutional layesr are indispensable in the state-of-the-art GAN models, thus the above methods can only be applied to models dominated by fully connected layers (*e.g.* StyleGAN), not all GAN models. Figure 1(a) illustrates the difference between traditional generator architecture and style-based generator architecture. To overcome this limitation, we propose to decompose the matrix of *all* layers in the model (including the convolutional layers and the fully connected layers) to obtain latent dimensions with better consistency and interpretability.

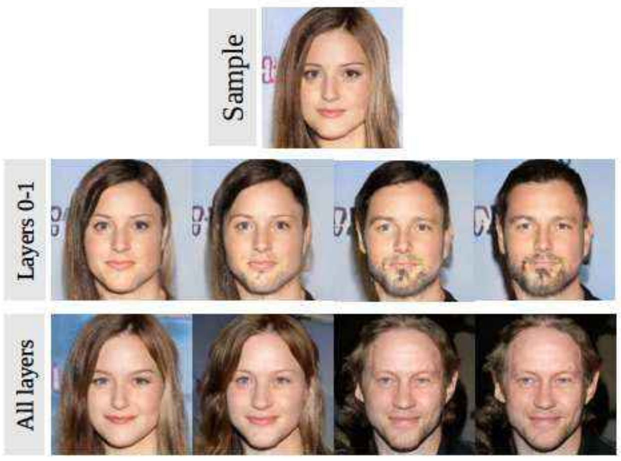
To be concrete, [12] only investigate the first projection step because the matrix decomposition of the fully connected layers does not require extra operations, while we investigate the whole network by taking convolutional layesrs into consideration. [13] computes the largest singular value of a convolutional layesr using a power iteration method, which proved to be a useful heuristic for regularization. [14] provides a more efficient way to characterize the singular values of the linear transformation associated with a standard 2D multi-channel convolutional layer.

With matrix decomposition methods for fully connected layers and convolutional layers respectively, we propose a generalized closed-form method that can identify versatile semantics from the latent space by merely using the pre-trained weights of the generator. More importantly, these variation factors are accurate and in a wider range compared to the

*Corresponding author (e-mail: zhangdy27@mail.sysu.edu.cn). This work was partly supported by the National Natural Science Foundation of China under Grant No.61876224, the Natural Science Foundation of Guangdong Province under Grant No.2019A1515010939, the Science and Technology Projects in Guangzhou under Grant No. 201904010493, the Featured Innovative Projects of Department of Education of Guangdong Province under Grant No. 2019KTSCX175, the Major Projects of Guangdong Education Department for Foundation Research and Applied Research under Grant No. 2018KZDXM066.



(a) Traditional generator architecture and style-based generator architecture. A mapping network (fully connected layers) is used in Style-based generator to encode the input vector into an intermediate vector for controlling different visual features. Figure from [1].



(b) The first row is a sample from a pre-trained generator, the second row (only weight matrices of layers 0-1 are considered) and third row (weight matrices of all layers are considered) show the output images by moving the latent code on the interpretable direction (gender here).

Fig. 1. (a) shows the traditional generator architecture and style-based generator architecture. Note that the traditional generator is mainly composed of convolutional layers, while the style-based generator uses fully connected layers to find latent vectors in the initial stage. (b) shows some results of considering only the fully connected layers and all layers (including the fully connected layer and the convolutional layer) on BigGAN, which adopts a traditional GAN architecture as backbone.

state-of-the-art supervised and unsupervised approaches. Figure 1(b) shows some results produced by the proposed generalized closed-form factorization of latent semantics in GANs. Note that we consider *all* layers, including convolutional layers and fully connected layers, while [12] only consider the fully connected layers in the initial stage. We provide qualitative and quantitative results through extensive experiments to show that our approach is efficient and applicable to most popular GAN models that are trained on different datasets.

2. METHOD

2.1. Unsupervised Semantic Factorization With Fully Connected layers

We first introduce how GANs generate images. The goal of a generator $G(\cdot)$ in GANs is to learn a mapping from a latent space $\mathcal{Z} \subset \mathbb{R}^d$ in a lower dimension to a higher dimensional image space $\mathcal{I} \subset \mathbb{R}^{H \times W \times 3}$, *i.e.*, $I = G(z)$ where $I \in \mathcal{I}$ and $z \in \mathcal{Z}$ denote the output image and the input latent code respectively. In the training process of GANs, $G(\cdot)$ projects a given latent code randomly sampled from the latent space to the final image space step by step. Each step learns a transformation from one space to another. In particular, it can be formulated as an affine transformation as

$$G_i(z) \triangleq y = Az + b, \tag{1}$$

where y is the projected code. Suppose $y \in \mathbb{R}^m$, then $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$ denote the weight and bias used in the ith transformation step $G_i(\cdot)$ respectively.

Then we introduce how to manipulate the generation process in GAN latent space. The latent space of GANs has recently been shown to encode rich semantic knowledge. These semantics can be further applied to image editing with the vector arithmetic property [15]. More concretely, prior work proposed to use a certain direction $n \in \mathbb{R}^d$ in the latent space to represent a semantic concept. After identifying a semantically meaningful direction, the manipulation can be achieved via the following model

$$edit(G(z)) \triangleq G(\hat{z}) = G(z + \alpha n), \tag{2}$$

which is commonly used in the existing approaches. Here, $edit(\cdot)$ denotes the editing operation. In other words, we can alter the target semantic by linearly moving the latent code z along the identified direction n, and α indicates the manipulation intensity.

As discussed above, the generator in GANs can be viewed as a multi-step function that gradually projects the latent space to the image space. To reveal the explanatory factors (*i.e.*, the direction n in Eq. 2) from the latent space of GANs, we simplify the manipulation model in Eq. 2 as

$$\begin{aligned} \hat{y} \triangleq G_i(\hat{z}) &= G_i(z + \alpha n), \\ &= Az + b + \alpha An = y + \alpha An. \end{aligned} \tag{3}$$

Note that the manipulation process is instance independent according to Eq. 3. That's to say, given any latent code z

[redacted]

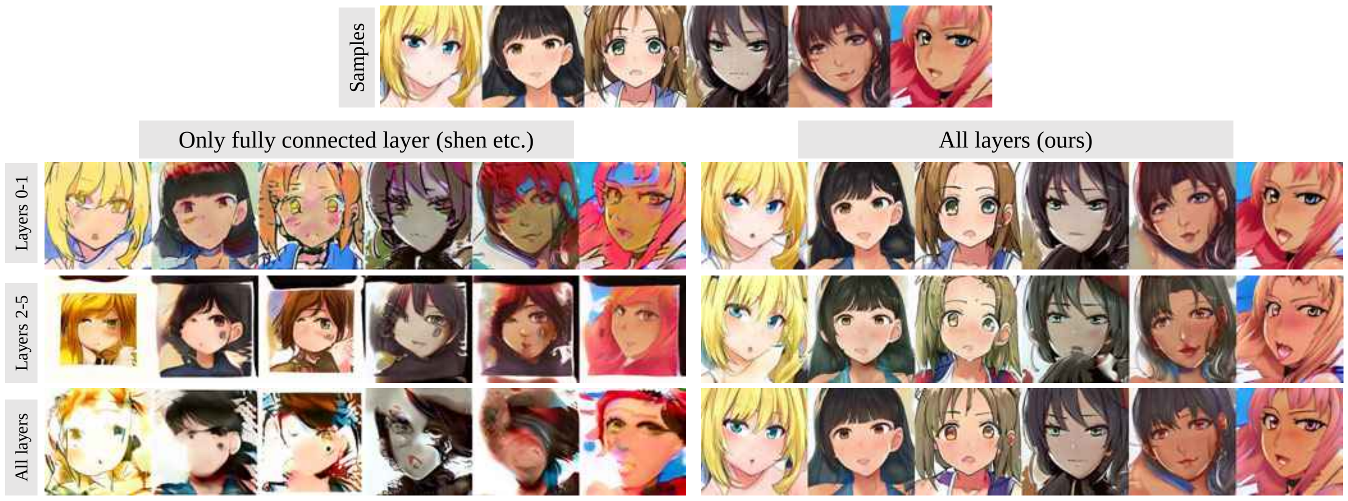


Fig. 2. Qualitative Results. The images in the first row are samples from a pre-trained BigGAN generator on anime faces, the following three rows show the output images when moving the the latent vector on the interpretable direction as the number of layers increases. Note that our method can better maintain the consistency of the image content.

together with a certain latent direction n, the editing can be always achieved by adding the term αAn onto the projected code. From this perspective, the weight parameter A should contain the essential knowledge of the image variation. Thus the important latent directions could be discovered by decomposing A. For semantic factorization by solving the following optimization problem

$$n^\star = \underset{n^Tn=1}{\arg\max} \|An\|^2, \tag{4}$$

where $\|\cdot\|$ denotes the l_2 norm.

This problem aims at finding the directions that can cause large variations after the projection of A. Intuitively, if some direction $\hat{n}$ is projected to a zero-norm vector, *i.e.*, $A\hat{n} = 0$, the editing operation in Eq. 3 turns into $\hat{y} = y$, which will keep the output synthesis unchanged, let clone alter the semantics occurring in it. When the case comes to finding k most important directions $n_1, n_2, n_3, \cdots, n_k$, we expand Eq. 4 into

$$N^\star = \underset{n_i^Tn=1}{\arg\max} \sum_{i=1}^{k} \|An_i\|^2, \tag{5}$$

where $N = \{n_1, n_2, n_3, \cdots, n_k\}$ correspond to the top-k semantics. To solve this problem, we introduce the Lagrange multipliers $\{\lambda_i\}_{i=1}^k$ into Eq. 5 as

$$\begin{aligned} N^\star &= \arg\max \sum_{i=1}^{k} \|An\|^2 - \sum_{i=1}^{k} \lambda_i \left(n_i^Tn_i - 1\right), \\ &= \arg\max \sum_{i=1}^{k} \left(n_i^TA^TAn_i - \lambda_i n_i^Tn_i + \lambda_i\right). \end{aligned} \tag{6}$$

By taking the partial derivative on each n_i, we have

$$2A^TAn_i - 2\lambda_i n_i = 0. \tag{7}$$

All possible solutions to Eq. 7 should be the eigenvectors of the matrix A^TA. To get the maximum objective value and make $\{n_i\}_{i=1}^k$ distinguishable from each other, we choose columns of N as the eigenvectors of A^TA associated with the k largest eigenvalues.

2.2. The Eigenvalues of Convolutional Layers

Almost all state-of-the-art GAN models [1, 3, 4] typically adopt convolutional neural networks as the generator architecture. The remaining problem is to find a characterization of the eigenvalues of a convolutional layer.

Some regularization methods have been proposed to improve the generalizability of deep learning models by imposing constraints on the weight matrices to reduce the sensitivity to input perturbation. To be more specific, consider a convolutional layer with m input channels, m output channels, and a $k_w \times k_h$-sized kernel, and the sizes of input and output feature map are both $n \times n$. To keep things simple, here we consider the stride is 1 and input channel equals output channels. [13] proposed to reshape the given $m \times m \times k_w \times k_h$ kernel into a $mk^2 \times m$ matrix, and compute its largest singular value using a power iteration method. [14] push this further, they calculated *all* singular values of a convolutional layer by first apply Fast Fourier Fransform (FFT) on the kernel and then perform Singular Value Decomposition (SVD), the whole process takes $\mathcal{O}(n^2m^2(m + \log n))$ time.

However, as discussed in the last section, we only need top k largest eigenvalues in semantic factorization, that is why we propose our method: we first apply FFT on the convolutional kernel as suggested in [14], then we reshape the result transform matrix A into $mn^2 \times m$, and finally we only need to solve the m eigenvalues in $A^T A$. The whole calculation process can be implemented by the following lines of code.

```
def EigenValuesOfCNN(kernel, input_shape):
    # kernel.shape = C_out, C_in, k_h, k_w
    # input_shape = [H_in, W_in]
    A = np.fft.fft2(kernel, input_shape, axes
    =[-2, -1])
    C_out, C_in, H_in, W_in = A.shape
    A = A.reshape(C_out*H_in*W_in, C_in)
    return np.linalg.eig(A.T.dot(A))
```

Here $kernel$ denotes the learnable convolution kernel and $input\_shape$ is the shape of the feature map to be convolved. The algorithm first performs m^2 2D FFT on $n \times n$ input, which takes $\mathcal{O}(m^2 n^2 \log n)$), and then performs eigenvalue decomposition algorithm in $\mathcal{O}(m^3)$, the whole process takes $\mathcal{O}(m^2(m + n^2 \log n)))$ time. Practically, this process may be executed in parallel, resulting in $\mathcal{O}(\max(m^3, n^2 \log n))$ time complexity.

Algorithm 1 Characterizing eigenvalues of a convolutional layer.

```
$weights = []$
**for** layer in layers **do**
    $weight = layer.weight$
    $shape = layer.weight.shape$
    **if** layer is a convolutional layer **then**
        $weight = np.fft.fft2(weight, shape)$
        $C_{out}, C_{in}, H_{in}, W_{in} = weight.shape$
        $weight = weight.reshape(C_{out} * H_{in} * W_{in}, C_{in})$
    **else if** layer is a fully connected layer **then**
        $weight = weight$
    **end if**
    $weights.append(weight)$
**end for**
$A = np.concatenate(weights, axis = 1)$
$transform = A.T.dot(A)$
return $np.linalg.eig(transform)$
```

Specifically, for a generator $G(\cdot)$ containing both convolutional layers and fully connected layers, we pass the weight matrix of the convolutional layers through an FFT, leaving the weight matrix of the fully connected layers unchanged. Finally, we concatenate all the weight matrices to get A, and perform eigen-decomposition on $A^T A$. The specific implementation is shown in Algorithm 1.

Exposing the eigenvalues of a convolutional layer opens the door to find a more accurate semantic dimension in the latent space in GANs, the results in the following section show the effectiveness of our method.

3. EXPERIMENTS

3.1. Models, Datasets and Metrics

We compare our method with [12] on a wide range of the state-of-the-art GAN models, including StyleGAN [1], StyleGAN2 [16] and BigGAN [3]. Note thet StyleGAN like architectures use fully connected layers in the initial stage to learn the latent code that controls the generated image style, while BigGAN like models are entirely composed of convolutional layers without any fully connected layer. Details are as follows:

StyleGAN. StyleGAN [1] firs proposed the style-based generator to improve traditional generators. Instead of producing the first spatial feature map from the latent code, StyleGAN employs a constant feature map as the input. Meanwhile, the latent code, which is learned from a series of fully connected layers, is fed into each convolution layer to modulate the feature map. In particular, for each layer, the latent code is transformed to a style code, which is used to alter the channel-wise mean and variance of the feature map through the Adaptive Instance Normalization (AdaIN) [17] operation. For this GAN type, the content of the generated images is mainly controlled by the style code, so reasonable results can be obtained by considering only the fully connected layers, but taking the convolutional layers into account allows us to find dimensions with semantic information in a larger latent space (bigger step). Experimental results are shown in Figure 2 and Figure 3.

BigGAN. BigGAN [3] is a large-scale GAN model primarily designed for conditional generation. The latent code is both mapped to the initial feature map and fed into each convolution layer to control the conditional batch normalization (BN) operation. BigGAN uses the traditional generator architecture, which is mainly composed of convolutional layers. In the following experiments, we can observe that as the number of layers considered increases, if only the fully connected layer is considered, then the generated image will not guarantee the semantic validity, and our method can guarantee that the generated image is consistent with the original one semantically.

Datasets and Metrics. For qualitative results, we use models trained on human faces (FF-HQ [18]) and anime faces [19]. For quantitative comparison, we use Fr´echet Inception Distance (FID) [20], which indicates the distribution matching between the real and generated images, to evaluate the quality of samples generated by GANs, the lower the value of FID, the higher the quality of the generated images.

3.2. Qualitative Results.

Figure 2 and Figure 3 show the qualitative results. To be specific, Figure 2 compares the results of our method with [12].

Notice that as the number of layers increases, our method can still guarantee the interpretability of the latent vector,

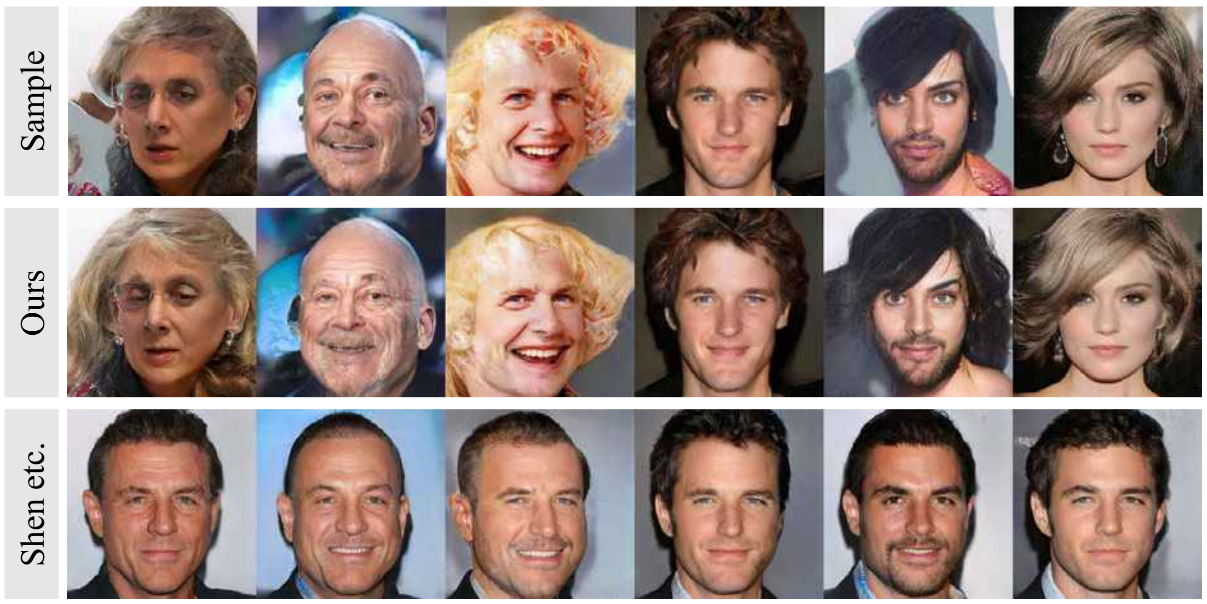


Fig. 3. Qualitative results: (a) Source samples from pre-trained BigGAN on FF-HQ. (b) and (c) are obtained by moving the latent code on the interpretable direction (hair style here) with the same step $\alpha = 30$. Note that our method can keep the consistency of the content with larger step size, while [12] fail to maintain the semantic meaning.

while they only consider fully connected layers in the initial stage, result in uninterpretable results. In addition, we are able to find latent code in a larger space with the proposed method, *i.e.*, with a larger step size α. As shown in Figure 2, with the same step size, the results produced by [12] fail to maintain the content, yet the results produced by our method keep the main content of the image remains unchanged with only hairstyle changed. This feature allows us to fine-tune the details of the image, *e.g.*, the length of the hair.

3.3. Quantitative Results.

We conduct experiments on the state-of-the-art models trained on various datasets, where we randomly generated the step size $\alpha \in [-100, 100]$ and generate 10,000 images to calculate FID for quantitative comparisons. It can be seen that in SytleGAN like architectures, which uses a fully connected layer to learn hidden code in the initial stage, our method produce comparable results to the SOTA method. However, in BigGAN like architectures, which entirely consist of convolutional layers, the proposed method improve the results greatly. The results are shown in Table 1.

In addition, we further compare the method proposed in this paper with the method proposed in [14] for solving the eigenvalues of a convolutional layer, and the experimental results are illustrated in Figure 4, where Full represents the brute force solution which would take $\mathcal{O}(n^6 m^3)$ time. We can see that our method will be much faster when we only need to solve the top k largest eigenvalues for semantic factorization.

	shen etc.	Ours
StyleGAN	8.22	8.31
StyleGAN2	7.78	7.65
BigGAN	10.42	**6.52**

Table 1. Quantitative comparison with StyleGANs and BigGAN on FF-HQ.

4. CONCLUSION

In this work, we propose a **generalized** closed-form factorization of latent semantics to improve the factorized latent semantics learned by GANs. Our method takes into account both the fully connected layers and the convolutional layers, which allows us to decompose semantics in a wider latent space, and maintain that the main content of the generated images when applied to a wide range of deep generators.

5. REFERENCES

[1] Tero Karras, Samuli Laine, and Timo Aila, "A style-based generator architecture for generative adversarial networks," in *CVPR*, 2019, pp. 4401–4410.

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, "Generative adversarial nets," in *NeurIPS*, 2014, pp. 2672–2680.

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan,

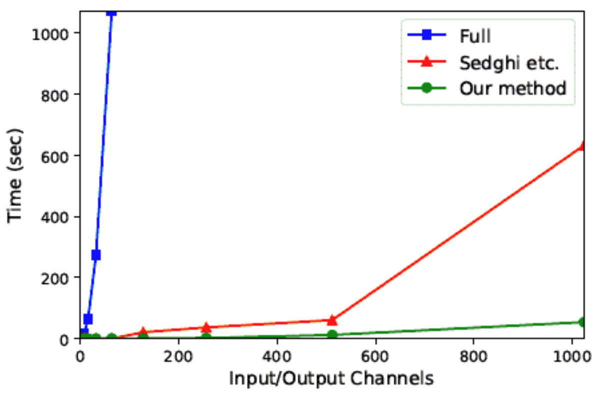


Fig. 4. Time used to compute top k largest eigenvalues, a numpy implementation. We perform a 3×3 convolution on a 16×16 image with the number of input/output channels on the x-axis.

"Large scale gan training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[4] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8110–8119.

[5] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros, "Image-to-image translation with conditional adversarial networks," in *CVPR*, 2017, pp. 1125–1134.

[6] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[7] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al., "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4681–4690.

[8] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 2180–2188.

[9] Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *ICLR*, 2017.

[10] Andrey Voynov and Artem Babenko, "Unsupervised discovery of interpretable directions in the gan latent space," in *International Conference on Machine Learning*. PMLR, 2020, pp. 9786–9796.

[11] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris, "Ganspace: Discovering interpretable gan controls," *arXiv preprint arXiv:2004.02546*, 2020.

[12] Yujun Shen and Bolei Zhou, "Closed-form factorization of latent semantics in gans," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 1532–1540.

[13] Yuichi Yoshida and Takeru Miyato, "Spectral norm regularization for improving the generalizability of deep learning," *arXiv preprint arXiv:1705.10941*, 2017.

[14] Hanie Sedghi, Vineet Gupta, and Philip M Long, "The singular values of convolutional layers," *arXiv preprint arXiv:1805.10408*, 2018.

[15] Alec Radford, Luke Metz, and Soumith Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[16] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville, "Improved training of wasserstein gans," in *NeurIPS*, 2017, pp. 5767–5777.

[17] Xun Huang and Serge Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1501–1510.

[18] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[19] the Danbooru community, "Danbooru2018: A large-scale crowdsourced and tagged anime illustration dataset," January 2019, Accessed: DATE.

[20] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *NeurIPS*, 2017, pp. 6626–6637.