Human-in-the-Loop Differential Subspace Search in High-Dimensional Latent Space

CHIA-HSING CHIU, National Taiwan University of Science and Technology
YUKI KOYAMA, National Institute of Advanced Industrial Science and Technology (AIST)
YU-CHI LAI, National Taiwan University of Science and Technology
TAKEO IGARASHI, The University of Tokyo
YONGHAO YUE, Aoyama Gakuin University (AGU)

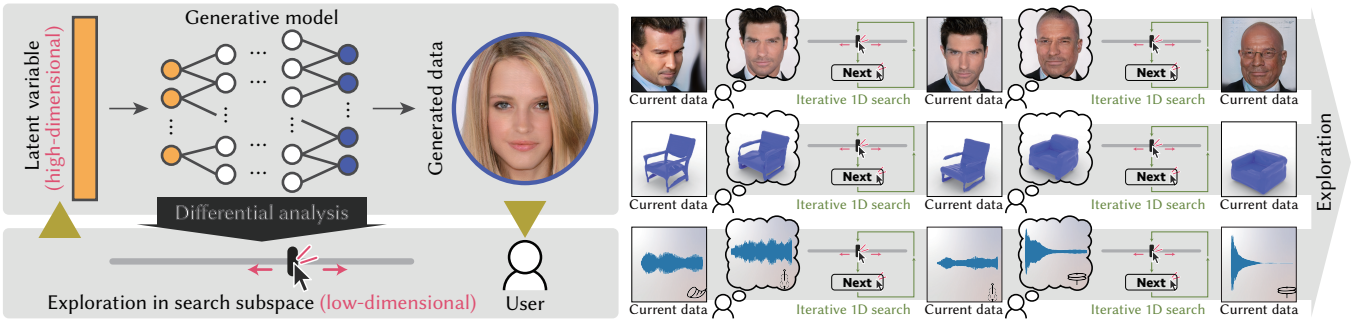


Fig. 1. We present *differential subspace search* for exploring the high-dimensional latent space of a deep generative model, by letting the user perform searches in (low-dimensional) 1D subspaces, where the search directions are provided through differential analysis of the generative model (left). The user iteratively performs searches via a slider interface and updates the subspace by pressing the 'next' button (right). Our method does not rely on domain- or data-specific assumptions and can be applied to exploratory tasks for various generative models for images, sounds, and 3D models.

Generative models based on deep neural networks often have a high-dimensional latent space, ranging sometimes to a few hundred dimensions or even higher, which typically makes them hard for a user to explore directly. We propose *differential subspace search* to allow efficient iterative user exploration in such a space, without relying on domain- or data-specific assumptions. We develop a general framework to extract low-dimensional subspaces based on a local differential analysis of the generative model, such that a small change in such a subspace would provide enough change in the resulting data. We do so by applying singular value decomposition to the Jacobian of the generative model and forming a subspace with the desired dimensionality spanned by a given number of singular vectors stochastically selected on the basis of their singular values, to maintain ergodicity. We use our framework to present 1D subspaces to the user via a 1D slider interface. Starting from an initial location, the user finds a new candidate in the presented 1D subspace, which is in turn updated at the new candidate location. This process is repeated until no further improvement can be made. Numerical simulations show that our method can better optimize synthetic black-box objective functions than the alternatives that we tested. Furthermore, we conducted a user study using complex generative models and the results show that our method enables more efficient exploration of high-dimensional latent spaces than the alternatives.

Authors' addresses: C.-H. Chiu, Y.-C. Lai, T4-305-1, No.43, Sec.4, Keelung Rd., Taipei, 106, Taiwan; Y. Koyama, Central 2, 1-1-1 Umezono, Tsukuba, Ibaraki, 305-8568, Japan; T. Igarashi, Science bldg. 7-303, 7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-0033, Japan; Y. Yue, O-525, Fuchinobe 5-10-1, Chuo-ku, Sagamihara, Kanagawa, 252-5258, Japan.

© 2020 Association for Computing Machinery.
This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *ACM Transactions on Graphics*, https://doi.org/10.1145/3386569.3392409.

CCS Concepts: • **Computing methodologies** → **Computer graphics**; *Search methodologies*.

Additional Key Words and Phrases: Human-in-the-loop optimization, dimensionality reduction, generative models

ACM Reference Format:
Chia-Hsing Chiu, Yuki Koyama, Yu-Chi Lai, Takeo Igarashi, and Yonghao Yue. 2020. Human-in-the-Loop Differential Subspace Search in High-Dimensional Latent Space. *ACM Trans. Graph.* 39, 4, Article 85 (July 2020), 15 pages. https://doi.org/10.1145/3386569.3392409

1 INTRODUCTION

With the recent developments in deep learning, we have seen a rapid advancement in using machine learning to generate data for multimedia content; there are many impressive generative models with applications ranging from images [Karras et al. 2018; Kingma et al. 2014; Miyato et al. 2018] and sounds [Donahue et al. 2018; Engel et al. 2019] to 3D models [Chen and Zhang 2019; Umetani 2017], to just list a few. A key to this success is the ability to perform efficient and effective training using complex network architectures with more layers and higher-dimensional internal representations (i.e., latent spaces). For instance, IM-GAN for 3D shapes [Chen and Zhang 2019] and SN-GANs for images [Miyato et al. 2018] have 128-dimensional latent spaces, while GANSynth for audio [Engel et al. 2019] has 256. Such richness of network architectures allows for highly nonlinear representations between the internal latent variables and the real data, which in turn enables generation of high-quality data that are almost indistinguishable from the real ones.

On the downside, excessively high dimensionality makes direct manipulation of the latent variables notoriously hard for end-users. Still, it is desired in many applications that the user can adjust the results and explore neighbor alternatives.

We propose a simple method aimed at efficient user searches in such high-dimensional spaces. Our method is *domain- and data-agnostic*, unlike previous approaches that utilize domain-specific interfaces [Bau et al. 2019; Brock et al. 2017; Umetani 2017]. In addition, our method targets *extremely high-dimensional latent spaces* with hundreds of variables, in contrast to the existing methods for exploring general multi-dimensional spaces [Brochu et al. 2007; Koyama et al. 2017]. Furthermore, our method takes only a *pre-trained* generative model as input, unlike the previous approaches to control data generation [Kingma et al. 2014; Mirza and Osindero 2014] that require the models to be re-trained with specially annotated data, which results in a still high-dimensional space hard for a user to explore directly.

Our method is built on the concept of optimization and is designed for exploratory purposes. Because the relationship between the latent variables and the generated data is in general highly nonlinear and multi-peaked, finding the exact target (or in other words, the globally optimal solution) is thus hard according to a general understanding of optimization [Nesterov 2018]. Nevertheless, the ability to find a local optimum would still be useful for the user to adjust the results and explore neighbor alternatives.

Our key idea is to let the user perform iterative searches, each of which lies in a much reduced-dimensional subspace, and to construct such subspaces on the basis of local differential analysis of the generative model. For an efficient search, a small change in such a subspace should provide a sufficient change in the resulting data. We develop a general framework that extracts low-dimensional subspaces by performing singular value decomposition on the Jacobian of the generative model and forming a subspace spanned by a set of singular vectors stochastically selected on the basis of their singular values in order to retain the chance to visit all subspaces (i.e., ergodicity). We use our framework to present 1D subspaces to the user via a 1D slider interface (Figure 1). Starting from an initial location, the user finds a new candidate in the presented 1D subspace, which is in turn updated at the new candidate location. This process is repeated until no further improvement can be made.

Numerical simulations show that our method performs much better for synthetic black-box objective functions than Bayesian optimization-based ones [Koyama et al. 2017]. We conducted a user study using pre-trained complex generative models and found that our method enables more efficient exploration in high-dimensional latent spaces. We also applied our method to various generative models for images, sounds, and 3D shapes, as listed in Table 1.

2 RELATED WORK

2.1 Domain-Specific Exploration Methods

High-dimensional latent spaces of generative models are, in general, intractable for humans to explore without computational guidance. To allow efficient user exploration, domain-specific methods have been developed. For generative image modeling, Zhu et al. [2016] and Brock et al. [2017] presented interactive methods allowing the user to directly paint colors on the generated image. Likewise, Bau et al. [2019] presented a method that allows direct annotation to be made on the generated image for object placement and removal. Hin et al. [2019] recently proposed a user-in-the-loop framework for searches in the latent space that incorporates user-provided annotations on images into the determination of queries presented to users. These methods achieve efficient latent space exploration through image-specific formulations and interfaces. For generative 3D modeling, Umetani [2017] demonstrated a 3D model-specific interface that allows direct manipulation of the shape to easily explore the latent space.

Recent research on generative adversarial networks [Goetschalckx et al. 2019; Jahanian et al. 2020; Shen et al. 2020; Yang et al. 2019] has shown that high-level semantic meanings may naturally emerge in the learned models, and directions (either straight or curved) representing such semantic meanings and useful for human manipulation can be found via optimization using domain-specific knowledge (e.g., external classification networks or image features).

Unlike these approaches, our differential subspace search method does not rely on any domain-specific formulations or specially annotated training data. Thus, it applies to various domains in the same formulation; we demonstrate its generality to generative modeling of images, sounds, and 3D models. We believe that domain-specific approaches are complementary to our work; a combined approach would allow more efficient exploration.

2.2 General Exploration Methods

Adjusting multiple parameters (typically through slider manipulations) is a ubiquitous task in visual design and graphics. A seminal work from the user interface point of view is *Design Galleries* [Marks et al. 1997], where the interface provides a two-dimensional gallery of diverse candidate designs. Another gallery-based interface, *Brainstorm*, had been in After Effects [Adobe 2017], where it repeatedly provided designs sampled stochastically with a controllable variance.

Involving humans in the analysis of design parameter spaces is another line of work. Koyama et al. [2014] proposed a crowd-powered method for building a preference model in general parameter spaces, which can efficiently guide the user's exploration through tailored interfaces for spaces with dimensions typically lower than 10. Talton et al. [2009] proposed a density-estimation-based method for guiding design exploration and successfully applied it to several applications including parametric 3D modeling with over 100 dimensions; however, this method needs many users to gather sufficiently large datasets for each application.

The parameter adjustment can also be viewed as an optimization with a perceptual objective. *Interactive evolutionary computation* [Takagi 2001], a human-in-the-loop variant of *evolutionary computation*, is one of the seminal approaches in this line. More recently, *Bayesian optimization* (BO) has become popular in the machine learning community [Shahriari et al. 2015; Snoek et al. 2012]. BO is a global optimization technique for *expensive-to-evaluate* black-box functions and is likely to find a reasonable solution with a small number of function evaluations. To handle perceptual objective functions, Brochu et al. [2007] formulated a human-in-the-loop BO,

where the user performs *pairwise comparison* tasks based on their preference iteratively. Koyama et al. [2017] extended this technique so that the user performs *line search* tasks sequentially, hence it is called *sequential line search* (SLS), to solve the original multi-dimensional problem. This method offers a dramatic improvement in performance. These human-in-the-loop BOs have been applied only to relatively lower-dimensional problems (e.g., 6–15), since BO itself is not effective in high-dimensional problems such as our target (e.g., 128–512) [Moriconi et al. 2019; Wang et al. 2016].

These previous methods are unaware of the data generation process and/or do not work well for high-dimensional spaces. In contrast, our method explicitly uses the target generative model in its formulation and is designed for high-dimensional spaces; our method differentiates the generative model to obtain effective 1D search subspaces for each iteration. The resulting user interface is the same as SLS [Koyama et al. 2017] from the user's point of view; however, our subspace selection is not BO-based, but instead is aware of the local structure of the generative models, thereby enabling efficient searches in high-dimensional spaces.

2.3 Controlled Training of Latent Spaces

Controlling the training of latent spaces is a promising direction to facilitate exploration of latent spaces. One such approach is to use conditional generative models; conditional GANs [Mirza and Osindero 2014] and conditional VAEs [Kingma et al. 2014] allow users to input conditions to better control the data generation. Yet, finding an appropriate seed from the high-dimensional latent space remains a challenging task. Also, this approach needs additional annotated datasets. Another possibility is to use unsupervised training techniques to learn interpretable latent spaces; for example, InfoGAN [Chen et al. 2016] encourages the learned latent space to be disentangled without using additional datasets. While this approach is useful for manual exploration in low-dimensional cases (e.g., small GANs for the MNIST dataset), it is less useful in high-dimensional cases since manipulating many (e.g., 256) sliders simultaneously is hard for users even if each slider has a semantic meaning. Importantly, these approaches require a new generative model to be trained for each application.

Our work is *orthogonal* to this direction: we do not modify the latent space, but rather analyze the local structure of the generative model to navigate exploration. Thus, our method allows the use of pre-trained generative models, as well as these controlled learning models.

3 EXPLORATION AS SUBSPACE GRADIENT ASCENT

3.1 Problem Formulation

Suppose that we have a generative model $\boldsymbol{f}\colon \mathcal{Z} \to \mathcal{X}$, where $\mathcal{Z} \subset \mathbb{R}^n$ is a high-dimensional latent space, say $n \geq 100$, and $\mathcal{X} \subset \mathbb{R}^m$ is a data space. Suppose that a user is trying to use this model $\boldsymbol{f}$ to generate a data $\boldsymbol{x}$, but he or she only has control over the latent space $\mathcal{Z}$. The quality of $\boldsymbol{x}$ is judged through the user's own perceptual *goodness* function $g\colon \mathcal{X} \to \mathbb{R}$, which is unknown to the system and may even be fuzzy or change during the *exploratory task*. A larger value of g indicates higher preference. Accordingly, we have the following problem:

$$\boldsymbol{z}^* = \arg\max_{\boldsymbol{z}\in\mathcal{Z}} g(\boldsymbol{f}(\boldsymbol{z})). \tag{1}$$

If g were known to the system, the machinery of optimization could be used to iteratively update the current best latent vector $\boldsymbol{z}^{(k)}$ through a gradient ascent process:

$$\boldsymbol{z}^{(k+1)} = \boldsymbol{z}^{(k)} + \Delta^{(k)} = \boldsymbol{z}^{(k)} + \alpha \left(\frac{\partial g(\boldsymbol{z}^{(k)})}{\partial \boldsymbol{z}}\right)^T, \tag{2}$$

where k in parenthesis denotes the iteration count, $\left(\frac{\partial g(\boldsymbol{z}^{(k)})}{\partial \boldsymbol{z}}\right)^T = \left[\frac{\partial g(\boldsymbol{z}^{(k)})}{\partial z_1} \cdots \frac{\partial g(\boldsymbol{z}^{(k)})}{\partial z_n}\right]^T$ is the gradient of g with respect to $\boldsymbol{z}$ at $\boldsymbol{z}^{(k)}$, and $\Delta^{(k)} = \alpha \left(\frac{\partial g(\boldsymbol{z}^{(k)})}{\partial \boldsymbol{z}}\right)^T$ is the increment. Henceforth, we will shorten $\frac{\partial g(\boldsymbol{z}^{(k)})}{\partial \boldsymbol{z}}$ to $\frac{\partial g}{\partial \boldsymbol{z}}$ for brevity. The step size $\alpha > 0$ is determined so as to maximize $g(\boldsymbol{f}(\boldsymbol{z}^{(k+1)}))$ (by using e.g., a line search):

$$\alpha = \arg\max_{\hat{\alpha}} g\left(\boldsymbol{f}\left(\boldsymbol{z}^{(k)} + \hat{\alpha}\left(\frac{\partial g}{\partial \boldsymbol{z}}\right)^T\right)\right). \tag{3}$$

If initially $\boldsymbol{z}^{(0)}$ is sufficiently close to the solution, the above iterative updates will eventually arrive at the global maximum.

In reality, the update (2) is impossible for the computer alone to perform, because g is unknown. Yet, it is hard for the user alone because of the high dimensionality of the search space $\mathcal{Z}$. In this work, we develop a method for assisting the update (2).

3.2 Human-in-the-Loop Differential Subspace Search

In viewing each update (2) as a search local to the current candidate $\boldsymbol{z}^{(k)}$, our basic idea is to limit the local search to a low-dimensional subspace that is easy for the user to manipulate. Because a generative model $\boldsymbol{f}$ obtained by an advanced machine learning technique may be highly nonlinear, such a subspace usually cannot be fixed globally and needs to be chosen locally. Likewise, limiting the local search to some particular subspace (e.g., the most significant directions) would leave some subregion in $\mathcal{Z}$ unexplored. We designed a method to avoid such drawbacks.

Formally, we let $\boldsymbol{w} \in \mathcal{W}(\boldsymbol{z}^{(k)}) \subset \mathbb{R}^l$ be a vector lying in a low-dimensional ($l \ll n$) subspace $\mathcal{W}(\boldsymbol{z}^{(k)})$ local to $\boldsymbol{z}^{(k)}$, and replace the update (2) by the following one involving the subspace:

$$\boldsymbol{z}^{(k+1)} = \boldsymbol{z}^{(k)} + \tilde{\Delta}^{(k)} = \boldsymbol{z}^{(k)} + \boldsymbol{p}_{\boldsymbol{z}^{(k)}}(\boldsymbol{w}), \tag{4}$$

$$\boldsymbol{w} = \arg\max_{\hat{\boldsymbol{w}}\in\mathcal{W}(\boldsymbol{z}^{(k)})} g(f(\boldsymbol{z}^{(k)} + \boldsymbol{p}_{\boldsymbol{z}^{(k)}}(\hat{\boldsymbol{w}}))), \tag{5}$$

where $\boldsymbol{p}_{\boldsymbol{z}^{(k)}} : \mathcal{W}(\boldsymbol{z}^{(k)}) \to \mathcal{Z}$ is an operator prolonging the low-dimensional vector $\boldsymbol{w}$ at the current position $\boldsymbol{z}^{(k)}$ to the original search space $\mathcal{Z}$. Typically, we choose $l = 1$ so that the user can easily search the subspace via a simple slider interface (following Koyama et al. [2017]). The key points of our technique are 1) how to choose a suitable subspace $\mathcal{W}(\boldsymbol{z}^{(k)})$ and 2) how to retain the possibility to explore all candidates in $\mathcal{Z}$. Our technique is summarized in Figure 2 and Algorithm 1.

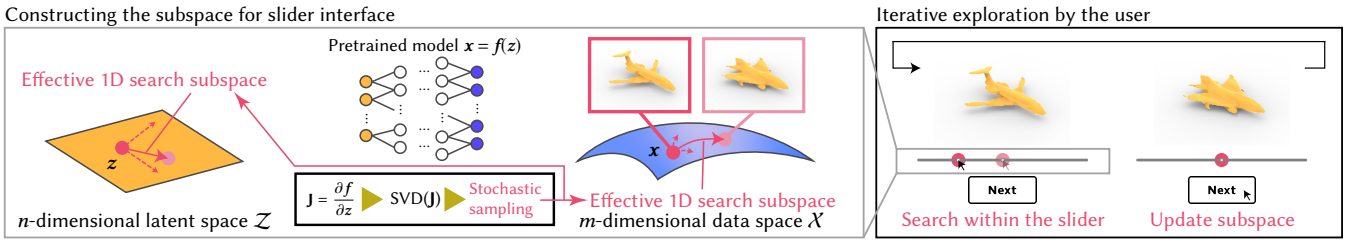


Fig. 2. We enable user exploration in the high-dimensional latent space through a slider interaction (right). This slider space corresponds to a 1D subspace of the full-dimensional latent space (left). Different search directions in the latent space (dashed and solid pink arrows) would result in different search directions in the data space, but most of them would not provide enough change in the data space (dashed pink arrows) due to the skewed distribution of the singular values of the Jacobian of the generative model. Our *differential subspace search* constructs an effective subspace (solid pink arrow) by analyzing the Jacobian and performing stochastic sampling according to the magnitude of its singular values to select the corresponding subspace (left).

Algorithm 1 `Differential_Subspace_Search`

Input: The goodness function g; The generative model $\boldsymbol{f}$; An initial guess $\boldsymbol{z}^{(0)}$ (can be a random vector);
Output: The latent vector $\boldsymbol{z}^*$ that generates the desired data;

1: $k \leftarrow 0$; ▷ k: iteration count
2: **while** User not satisfied with $\boldsymbol{f}(\boldsymbol{z}^{(k)})$ **do**
3: $\mathcal{W}(\boldsymbol{z}^{(k)}), \boldsymbol{p}_{\boldsymbol{z}^{(k)}} \leftarrow$ `SVD_Reduction`$(\frac{\partial \boldsymbol{f}(\boldsymbol{z}^{(k)})}{\partial \boldsymbol{z}})$;
4: User finds $\boldsymbol{w}^{(k)} \in \mathcal{W}(\boldsymbol{z}^{(k)})$;
5: $\boldsymbol{z}^{(k+1)} \leftarrow \boldsymbol{z}^{(k)} + \boldsymbol{p}_{\boldsymbol{z}^{(k)}}(\boldsymbol{w}^{(k)})$; $k \leftarrow k + 1$
6: **end while**
7: **return** $\boldsymbol{z}^* \leftarrow \boldsymbol{z}^{(k)}$

3.2.1 Subspace Construction. To choose an effective subspace $\mathcal{W}(\boldsymbol{z}^{(k)})$, we first write the gradient $\left(\frac{\partial g}{\partial \boldsymbol{z}}\right)^T$ by using the chain rule:

$$\left(\frac{\partial g}{\partial \boldsymbol{z}}\right)^T = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}}\right)^T \left(\frac{\partial g}{\partial \boldsymbol{f}}\right)^T, \tag{6}$$

to reveal the term,

$$\mathbf{J} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{z}} = \begin{bmatrix} \frac{\partial f_1}{\partial z_1} & \cdots & \frac{\partial f_1}{\partial z_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial z_1} & \cdots & \frac{\partial f_m}{\partial z_n} \end{bmatrix} \in \mathbb{R}^{m \times n}, \tag{7}$$

which does not require any information unknown to the system and is available through machine learning techniques that involve back propagation [Linnainmaa 1970; Schmidhuber 2015]. Implementation-wise, we switch among 1) automatic differentiation, 2) a stochastic way of automatic differentiation, and 3) finite difference depending on the required performance (see Section 8.4 for details).

According to the manifold hypothesis [Bengio et al. 2013; Rifai et al. 2011], common empirical knowledge stating that data in the real world live on low-dimensional manifolds embedded in a high-dimensional data space, we can expect that $\mathbf{J}$ has a few dominant components[1]. Formally, let $\mathbf{U} \in \mathbb{R}^{m \times m}$, $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$,

[1]The manifold hypothesis itself does not forbid the generative model to be an isometry, where the singular values are equal in magnitudes. However, such cases are unlikely to exist because of the highly nonlinear nature of the generative model.

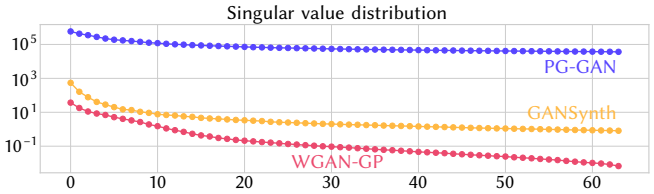


Fig. 3. Distributions of the first 64 singular values, sorted in descending order, for different generative models. The singular values of each model are computed from the Jacobian of the model at a stochastically sampled location in the latent space.

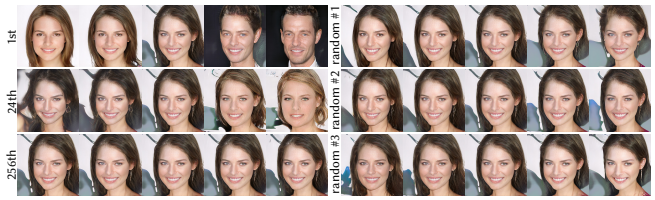


Fig. 4. We show the data variations along the directions (i.e., column vectors of $\hat{\mathbf{V}}$ in (12)) corresponding to the top, 24th, and the median (256th) singular values (left)) at a point stochastically selected from the 512-dimensional latent space of PG-GAN for images (detailed in Table 1), as well as the variations along with three stochastically chosen directions (right). The center image in each case corresponds to the data at the stochastically sampled point. The distance, in terms of the latent space, between the left- and right-most images in each case is 20% of the expected length between two random points in the latent space, the same as the slider length discussed in Section 3.2.2. We did not incorporate any approximations in the computation of the Jacobian or the singular value decomposition. While the direction corresponding to the top singular value provides enough variation in the resulting image, the direction corresponding to the median provides very little. If we choose a direction uniformly randomly, the corresponding magnitude of the variation is likely to be small.

when we consider the singular value decomposition of $\mathbf{J}$

$$\mathbf{J} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{8}$$

the diagonal singular value matrix $\mathbf{\Sigma}$ contains a few components that are much larger than the others, as in Figure 3.

We consider a low-dimensional approximation $\tilde{\mathbf{\Sigma}}$ to $\mathbf{\Sigma}$ that has only l non-zero entries. Deterministic construction, say, by always retaining the l largest entries, would result in ignorance of the remaining subspace. This is a problem because we set l to be extremely small (i.e., $l = 1$) for ease of user manipulation. Instead, we construct $\tilde{\mathbf{\Sigma}}$ in a probabilistic way such that

$$\mathbb{E}[\tilde{\mathbf{\Sigma}}] = \mathbf{\Sigma}. \tag{9}$$

Namely, we perform stochastic sampling to choose the non-zero entries, with a probability proportional to the magnitude of the singular values. This way, subspaces that influence the data more are more likely to be chosen. Now, we rewrite (8) as:

$$\mathbf{J} \approx \tilde{\mathbf{J}} = \mathbf{U}\tilde{\mathbf{\Sigma}}\mathbf{V}^T. \tag{10}$$

Because $\mathbb{E}[\tilde{\mathbf{\Sigma}}] = \mathbf{\Sigma}$, we have $\mathbb{E}[\tilde{\mathbf{J}}] = \mathbf{J}$. Further, let $\hat{\mathbf{\Sigma}} \in \mathbb{R}^{l\times l}$ be the packed version of $\tilde{\mathbf{\Sigma}}$ retaining all its non-zero singular values, and let $\hat{\mathbf{U}} \in \mathbb{R}^{m\times l}$ and $\hat{\mathbf{V}} \in \mathbb{R}^{n\times l}$ be, respectively, the l left and right singular vectors corresponding to the non-zero entries. We have

$$\mathbf{U}\tilde{\mathbf{\Sigma}}\mathbf{V}^T = \hat{\mathbf{U}}\hat{\mathbf{\Sigma}}\hat{\mathbf{V}}^T. \tag{11}$$

Hence, the increment $\Delta^{(k)}$ in (2) can be approximated as

$$\begin{aligned}\Delta^{(k)} &= \alpha\left(\frac{\partial g(\boldsymbol{z}^{(k)})}{\partial \boldsymbol{z}}\right)^T = \alpha\mathbf{J}^T\left(\frac{\partial g}{\partial \boldsymbol{f}}\right)^T\\ \approx\tilde{\Delta}^{(k)} &= \alpha\tilde{\mathbf{J}}^T\left(\frac{\partial g}{\partial \boldsymbol{f}}\right)^T = \alpha\hat{\mathbf{V}}\hat{\mathbf{\Sigma}}^T\hat{\mathbf{U}}^T\left(\frac{\partial g}{\partial \boldsymbol{f}}\right)^T\\ &= \hat{\mathbf{V}}\boldsymbol{w} = \boldsymbol{p}_{\boldsymbol{z}^{(k)}}(\boldsymbol{w}),\end{aligned} \tag{12}$$

where $\boldsymbol{w} = \alpha\hat{\mathbf{\Sigma}}^T\hat{\mathbf{U}}^T\left(\frac{\partial g}{\partial \boldsymbol{f}}\right)^T$ is an l-dimensional vector lying in $\mathcal{W}(\boldsymbol{z}^{(k)})$ formed by the column vectors of $\hat{\mathbf{V}}$. Now, we have $\mathbb{E}[\tilde{\Delta}^{(k)}] = \Delta^{(k)}$ (suppose that α is common to $\tilde{\Delta}^{(k)}$ and $\Delta^{(k)}$), and this increment $\tilde{\Delta}^{(k)}$ is an m-dimensional vector but effectively lying in a reduced l-dimensional subspace. In Section 4.2, we experimentally evaluate the effect of using this reduced subspace as opposed to the standard gradient ascent using the full space.

As in Figure 4, moving along a direction with a larger singular value would result in a larger variation in the data, while taking a random direction is expected to result in a far smaller variation, motivating our technique of stochastically selecting the subspace. In Section 4.4, we confirm this insight using a synthetic example.

3.2.2 Searching in the Subspace. Once the search subspace is constructed, we let the user find a single point $\boldsymbol{z}^{(k+1)} = \boldsymbol{z}^{(k)} + \tilde{\Delta}^{(k)}$ through an interface. By selecting $\boldsymbol{z}^{(k+1)}$, the user implicitly manipulates $\boldsymbol{w} = \alpha\hat{\mathbf{\Sigma}}^T\hat{\mathbf{U}}^T\left(\frac{\partial g}{\partial \boldsymbol{f}}\right)^T$ in the l-dimensional subspace, including its direction and magnitude, while the user does not need to be aware of any of α, $\hat{\mathbf{\Sigma}}$, $\hat{\mathbf{U}}$, and $\frac{\partial g}{\partial \boldsymbol{f}}$. This way, the effect of g is still accounted for by the user, while the l-dimensional subspace suggested by the system makes the search process much easier than one in the original full space. We take $l = 1$ to make the search process the simplest, so that the user only needs to decide the magnitude through a line search, without having to care about the direction, as in Figure 16, where $\boldsymbol{z}^{(k)}$ is displayed at the center of the slider and the user can search on both positive and negative sides.

This line search is not limited to a locally concave region; to avoid getting stuck near a local maximum, we allow the user to move across convex goodness barriers. Specifically, we take the length of the line search domain to be 20% of the expected length between two random points in the full latent space, with its center set to $\boldsymbol{z}^{(k)}$. We evaluate the effectiveness of this strategy in Section 4.3.

When the user cannot find a better choice in the subspace (i.e., optimizing (5) by the user results in the same point as the current one $\boldsymbol{z}^{(k)}$), the user can choose to stay at the current point and switch to another subspace, or to move to another point even if its goodness is worse. If the current point is still far away from the target, we encourage the user to follow the latter option, as it is easier to get away of a local maximum.

Since our method proposes the search direction solely on the basis of the generative model $\boldsymbol{f}(\boldsymbol{z})$, this direction probably does not perfectly align with the user's preference $\frac{\partial g}{\partial \boldsymbol{f}}$. However, the user can still find a point closer to the target in the search direction in principle, because of the nature of the expected alignment between two vectors in an effectively low-dimensional space: the expected absolute value of the dot product between two random vectors in a 10-dimensional space is 0.26, according to (25). Iteratively moving toward the target eventually allows convergence.

Note that it is also possible to use our technique to obtain a relatively low-dimensional (say $l = 6$) subspace first, and then use the sequential line search [Koyama et al. 2017] to enable exploration in this low-dimensional subspace by using a linear slider. We compare the performance of our differential subspace search with this hybrid option as well as other alternatives in Section 4.5.

4 EVALUATION WITH SYNTHETIC FUNCTIONS

Although a natural scenario for our method would be *exploratory tasks*, for an objective evaluation, we decided to examine its effectiveness in *goal-based tasks*: for each task, a target was presented at the beginning, and it was kept fixed throughout the evaluation. Before we examined the effectiveness of our method on real generative models, we did so for synthetic examples in a numerical simulation without users.

4.1 Designing the Synthetic Functions

We designed the functions $\boldsymbol{f}$ and g separately. We wanted to account for anisotropy in the synthetic generative function $\boldsymbol{f}^{\text{test}}\colon \mathcal{Z} \to \mathcal{X}$, where $\mathcal{Z} \subset \mathbb{R}^n$ and $\mathcal{X} \subset \mathbb{R}^m$, as well as for noise (non-concavity) in the synthetic goodness function $g^{\text{test}}\colon \mathcal{X} \to \mathbb{R}$.

We constructed the synthetic generative function as a concatenation of m anisotropic sphere functions:

$$f_i^{\text{test}}(\boldsymbol{z}) = (\boldsymbol{z} - \boldsymbol{s})^T\mathbf{A}_i(\boldsymbol{z} - \boldsymbol{s}) \quad (i = 1, \ldots, m), \tag{13}$$

where $\boldsymbol{s} \in \mathcal{Z}$ is a random shift vector and the metric tensor $\mathbf{A}_i = \mathbf{A}_i^T \in \mathbb{R}^{n\times n}$ is designed to account for the anisotropy. Considering its eigendecomposition $\mathbf{A}_i = \mathbf{Q}_i^T\tilde{\mathbf{A}}_i\mathbf{Q}_i$, we generated a rotation matrix $\mathbf{Q}_i$ and diagonal matrix $\tilde{\mathbf{A}}_i$ containing the eigenvalues as follows. First, we set $\tilde{\mathbf{A}}_i = \alpha_i\,\mathrm{diag}(a_{i,1}, \ldots, a_{i,n}) \in \mathbb{R}^{n\times n}$, where α_i controlled the magnitude of f_i^{test} and was uniformly sampled from $[0.01, 10.0]$, and $a_{i,1}, \ldots, a_{i,n}$ were uniformly sampled from $[a_{\min}, a_{\max}] = [0.01, 1.0]$ to create anisotropy; setting $a_{i,j}$ to a

constant irrespective of j would result in an isotropic setting. We chose $a_{\max} = 1.0$, because the magnitude was already accounted for by α_i, and $a_{\min} > 0.0$ to avoid degeneracy. Next, we constructed the random rotation matrix $\mathbf{Q}_i$ by generating a random matrix and then performing the Gram–Schmidt process. With our design, $10^{-4} \leq \alpha_i a_{\min} \leq \frac{f_i^{\text{test}}(\boldsymbol{z})}{\|\boldsymbol{z}-\boldsymbol{s}\|^2} \leq \alpha_i a_{\max} \leq \alpha_i \leq 10$.

Our synthetic goodness function g^{test} was based on the Rastrigin function $g^{\text{Rastrigin}}$:

$$g^{\text{Rastrigin}}(\boldsymbol{x}) = -Am - \sum_{i=1}^{m} \left(x_i^2 - A\cos(2\pi x_i) \right), \tag{14}$$

where the second term $A\cos(2\pi x_i)$ in the summation adds high-frequency noise (non-concavity) everywhere in the domain. We included α_i to relatively control the noise level of each x_i as:

$$g^{\text{test}}(\boldsymbol{x}) = -A\sum_{i=1}^{m} \alpha_i^2 - \sum_{i=1}^{m} \left(x_i^2 - A\alpha_i^2 \cos\left(2\pi x_i\right) \right), \tag{15}$$

to account for the fact that $\frac{x_i^2}{\|\boldsymbol{z}-\boldsymbol{s}\|^4} \leq \alpha_i^2$. We set A to 2.0; setting $A = 0$ would have made the model strictly concave. The overall optimization problem was written as $\max_{\boldsymbol{z}\in\mathcal{Z}} g^{\text{test}}(\boldsymbol{f}^{\text{test}}(\boldsymbol{z}))$, which has only one global maximum at $\boldsymbol{z} = \boldsymbol{s}$ but many local maxima (for $A \neq 0$), making it fairly difficult to find the optimal solution. We set $\mathcal{Z} = [-2.5, 2.5]^n$ and $m = 1024$ and evaluated the performance of each methodology with $n \in \{8, 32, 128, 512\}$.

4.2 Comparison against Gradient Ascent

From (12), our method is effectively a variant of gradient ascent that performs optimization in the subspace at each iteration, whereas the standard gradient ascent always works in the full space. We performed a comparison to see how the difference in the search direction affects the overall performance when using a concave version of the synthetic example by setting $A = 0$ in g. Since the objective in this case is strictly concave, the step sizes of both standard gradient ascent and our method were automatically determined via concave maximization of the 1D subproblem at each iteration.

Figure 5 compares the *optimality gaps*, i.e., the difference between the best found function value (g) and the optimal one. Our solution converges to the same exact solution as the standard gradient ascent. As expected, because the ascent direction lies in a subspace and thus is suboptimal compared to the full ascent direction, the convergence speed of our method is inferior to the standard gradient ascent, but the difference in performance is small.

4.3 Comparison against Local Line Search

Next, for the non-concave case ($A = 2.0$), we compare the effectiveness of setting the line search domain to be 20% of the expected length of the full latent space (**Ours**) with that of limiting it to a locally concave region (i.e., setting the next point to be the closest local maximum along the line search direction, **Ours-local**).

As shown in Figure 6, **Ours-local** easily gets stuck in a local optimum, leaving a large optimality gap, while **Ours** is able to significantly reduce it. For all our tests shown in this paper (including those with synthetic examples and generative models), we did not find any need for adaptively changing the line search domain; it is long enough for the user to make a large step for exploration, while also offering tuning, since the data within this length vary smoothly, as can be seen in our supplementary video.

4.4 Comparison against Uniformly Random Directions

Next, we compare our method with a version using uniformly randomly sampled directions (**Random-dir**) that has the same line search domain as **Ours**. As shown in Figure 7, **Random-dir** performs much worse than ours, leaving a large optimality gap, confirming the insights from Figure 4. In a skewed high-dimensional space, a randomly selected direction is likely to fail to provide enough change in the resulting function values.

4.5 Comparison against Other Alternatives

Next, we compare our method with possible alternatives using the above design of synthetic functions. Our direct counterpart is 1) the sequential-line-search (**SLS-BO**) [Koyama et al. 2017] (see Appendix A for the detailed conditions), which also involves an iterative search via a 1D slider. The difference is in how the search direction is chosen; **SLS-BO** uses Bayesian optimization. Since Bayesian optimization is known to be ineffective in high-dimensional spaces, we also tried 2) a REMBO [Wang et al. 2016] preprocessed SLS-BO (**SLS-REMBO-6D**), where the high-dimensional space is projected onto a stochastically determined low-dimensional (6D) space *a priori*, and then SLS-BO is applied to this low-dimensional space. We also compare 3) a hybrid approach combining our SVD-based dimension reduction (to 6D) with SLS-BO (**Hybrid-6D**). In **Hybrid-6D**, our SVD-based dimension reduction updates the 6D subspace after every 10 iterations of the SLS update (in the 6D subspace).

All methods work in a 1D search space in each iteration. For **SLS-BO**, **SLS-REMBO-6D** and **Hybrid-6D**, the slider length is determined automatically according to Koyama et al. [2017]. Within the 1D slider space, we uniformly sample 10,001 points to find the best one, instead of applying local optimization such as hill climbing. The uniform sampling helps the system to escape local maxima. The maximum iteration count is set to 100, a number large enough for our application; in reality, the user is likely to spend approximately 10 seconds in each iteration to explore the 1D space; thus, 100 iterations will take approximately 15 minutes, which we believe is longer than the time the user would expect to spend.

The results are shown in Figure 8. Comparing **SLS-BO** and **SLS-REMBO-6D**, **SLS-REMBO-6D** shows no performance improvement. This is perhaps due to the choice of the low-dimensional subspace in REMBO: it is globally fixed, so it cannot adapt to the highly nonlinear nature of our synthetic function. In contrast, our SVD-based dimension reduction successfully exploits this non-linear nature, and **Hybrid-6D** and **Ours** offer much smaller optimality gaps. Comparing **Hybrid-6D** and **Ours**, we see that **Ours** gives the smaller gap, despite its simple algorithmic structure.

We believe that the key difference affecting performance between the BO-based approaches and ours comes from the intrinsic nature of BO that 1) does not utilize the gradient and 2) requires learning of the objective landscape. The gradient used in our technique is higher-order information (than function values) that, as is well known in optimization, offers much better convergence. The cost

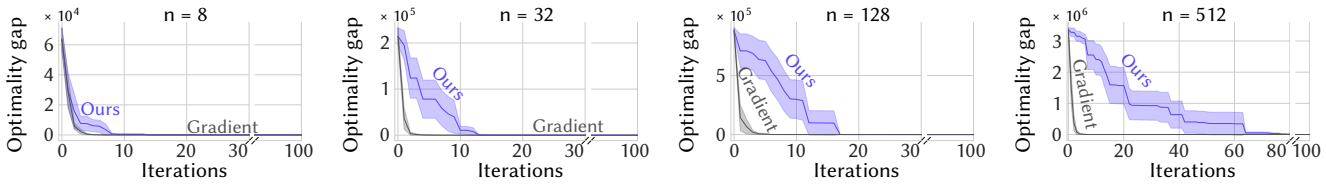


Fig. 5. We compare the convergence of our differential subspace search with that of the standard gradient ascent, for the strictly concave case as in Section 4.2. We performed 10 tests, each with a (different) stochastically sampled initial point. The solid curves show the averaged optimality gap G_{avr} as a function of the iteration count, with the filled regions corresponding to $G_{\text{avr}} \pm 0.4\sigma$, where σ is the standard deviation computed for the 10 tests.

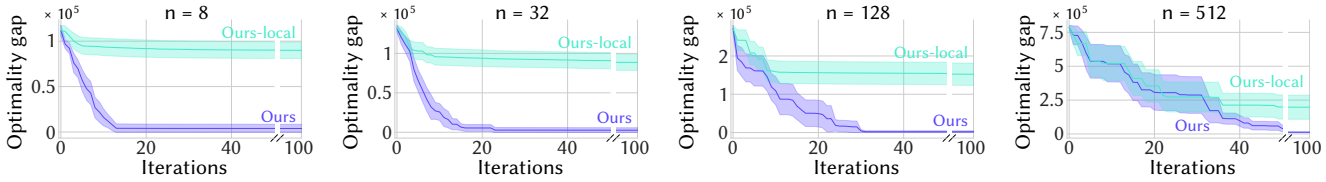


Fig. 6. We compare the performance of **Ours** and **Ours-local** for the multi-peak synthetic example in Section 4.3. We performed 10 tests, each with a (different) stochastically sampled initial point. The meanings of the solid curves and the filled regions are the same as those in Figure 5.

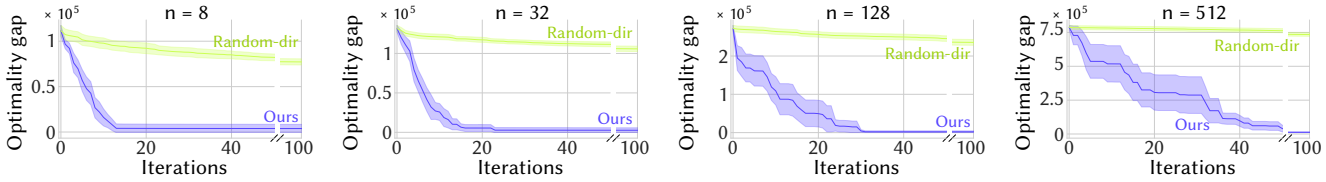


Fig. 7. We compare the performance of **Ours** and **Random-dir** for the multi-peak synthetic example in Section 4.4. We performed 10 tests, each with a (different) stochastically sampled initial point. The meanings of the solid curves and the filled regions are the same as those in Figure 5.

Table 1. Generative models considered in this paper.

App.	Dataset	Network
Image	MNIST [LeCun et al. 1998]	WGAN-GP [Gulrajani et al. 2017]
Image	CelebA-HQ [Karras et al. 2018]	PG-GAN [Karras et al. 2018]
Sound	NSynth [Engel et al. 2017]	GANSynth [Engel et al. 2019]
3D shape	ShapeNet [Chang et al. 2015]	IM-GAN [Chen and Zhang 2019]

for BO to learn the landscape of a high-dimensional problem seems intractable. BO-based methods balance local (known as "exploitation") and global (known as "exploration") samplings. **SLS-BO** in a high-dimensional space tends to perform too much exploitation at least during the first few hundred iterations, which is very local. We tried various settings (see Appendix A), but the tendencies were similar. As in Figure 9, our method outperforms BO in high-dimensional spaces even for the concave setting in Section 4.2. The cost of learning the landscape is also reflected in the performance of **Hybrid-6D**; the ten-times fewer updates of the subspaces in **Hybrid-6D** presumably lowers efficiency.

5 EVALUATION WITH DEEP GENERATIVE MODELS

First, as an extension to the evaluation using the synthetic functions, we evaluated the effectiveness of our method on a WGAN-GP [Gulrajani et al. 2017] based on MNIST [LeCun et al. 1998], a well-known dataset of hand-written digits (as in Table 1). Then, we performed user studies on more complex generative models using the GANSynth (sound) and the PG-GAN (image) generative models. Again, we focused on *goal-based tasks* for an objective evaluation.

5.1 Evaluation via Simulation

We compared **SLS-BO**, **SLS-REMBO-6D**, **Random-dir**, **Hybrid-6D**, **Ours-local**, and **Ours**, with automatic optimization without users, on a WGAN-GP network (with the architecture shown in Appendix B) that was trained with the MNIST dataset (as in Table 1). The latent variables in this case followed a uniform distribution in $[-1, 1]^n$, with $n = 64$. Here, f was the network, g the standard $L2$-norm between images, and we performed automatic optimization as in Section 4.5. We sampled 10 pairs of initial and target images, with the corresponding latent variables uniformly sampled from $[-1, 1]^n$. As shown in Figure 10, **Ours** offers a much smaller optimality gap compared with the alternatives.

5.2 User Study

Next, we recruited two disjoint sets of participants for the user study: *performers* to find data as close as possible to the given target via searches in high-dimensional latent spaces, and *evaluators* to evaluate the results of the performers. Based on the results of the experiments using synthetic examples and the above evaluation via

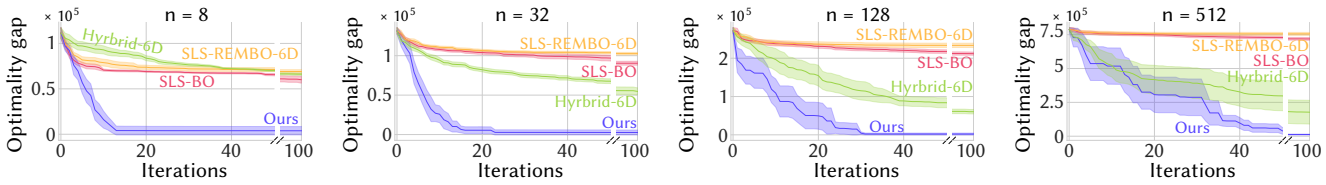


Fig. 8. We compare the performance of **Ours**, **SLS-BO**, **SLS-REMBO-6D**, and **Hybrid-6D** for the multi-peak synthetic example in Section 4.5. We performed 10 tests, each with a (different) stochastically sampled initial point. The meanings of the solid curves and the filled regions are the same as those in Figure 5.

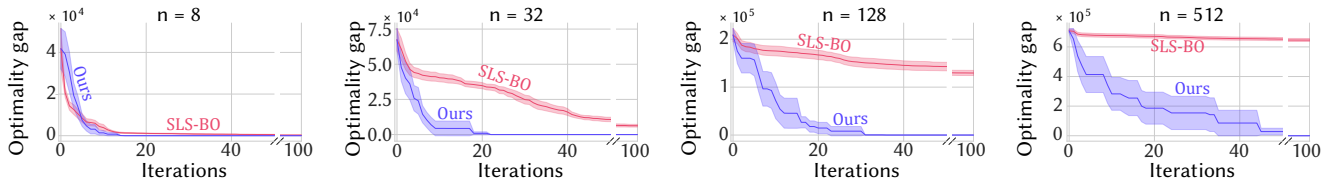


Fig. 9. We compare the performance of **Ours** and **SLS-BO** for the same concave settings as in Section 4.2. We performed 10 tests, each with a (different) stochastically sampled initial point. The meanings of the solid curves and the filled regions are the same as those in Figure 5.

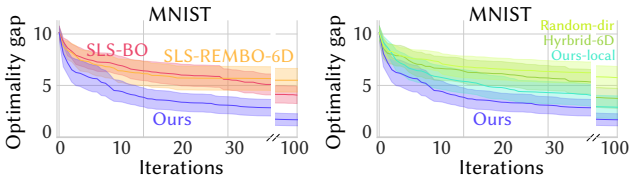


Fig. 10. We compare the performance of **Ours**, **SLS-BO**, **SLS-REMBO-6D**, **Ours-local**, **Hybrid-6D**, and **Random-dir** for the MNIST example in Section 5.1. We performed 10 tests, each with a (different) stochastically sampled initial point. The meanings of the solid curves and the filled regions are the same as those in Figure 5.

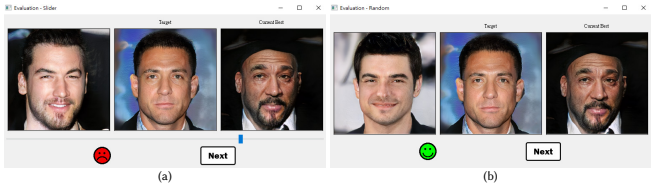


Fig. 11. The user interfaces of our user study. (a) Slider-based interface for **Ours** and **SLS-BO** and (b) the random selection user interface for **Random**. In both interfaces, by toggling the face button to a smile, the current data will be updated as the "current-best" data when the user presses the next button.

automatic optimization, we chose to compare **Ours** with **SLS-BO** and **Random** for the following reasons. Since we did not see any performance improvement for the **Hybrid-6D** option over **Ours**, we decided to discard **Hybrid-6D**. As a counterpart, we chose **SLS-BO** and discarded **SLS-REMBO-6D**, because **SLS-REMBO-6D** did not show any performance improvements over **SLS-BO**. Furthermore, we included **Random**, a pure random sampling approach for generating candidates in the latent space. We chose **Random** as a baseline because of its insensitivity to the curse of dimensionality and also because the user can perform more iterations with **Random**. Before we discuss the details of the two user tests using sound (GANSynth) and image (PG-GAN) generative models, we describe the common settings.

5.2.1 User Study Settings. We invited 12 participants (1 female and 11 males, all majors in computer science and aged between 20 and 30) to be *performers*. They were all novices to searches in high-dimensional latent spaces, and hence would have had no bias due to their skill levels. Prior to the study, the performers were given instructions on how to use the tools (**Ours**, **SLS-BO**, and **Random**), and they were asked to try them by themselves for no more than 10 minutes. **Ours** and **SLS-BO** had exactly the same user interface (as in Figure 11). They both displayed three data: the target data, the one corresponding to the slider position, and the current-best data. In addition, they had a slider and two buttons: the user could explore different data in the current subspace via the slider, update the data corresponding to the selected slider position as the new current data by pressing the *next* button, and save the current data as the "current-best" (when pressing the next button) by toggling the *face* button to a smile. For **Random**, we prepared a simple interface that displayed three data: the target data, the latest one generated by random sampling, and the current-best data. It had exactly the same buttons as the interfaces for **Ours** and **SLS-BO**. We did not disclose which interface was ours to the performers during the study.

Below, a *session* of the user test means the performance by a *single* performer in finding a candidate for a *single* given target data. Each session for the sound generative models consisted of 2 minutes and that for the image models 3 minutes. Furthermore, a *case* means the study related to a *single* target data: for a single case of the user study, each performer was asked to use all three tools in a randomized order for the same given target data. For each generative model, each performer was given 6 cases (hence in total 6×3 sessions).

To reduce the difference in task interpretation among the performers, we gave them instructions on how to judge the similarities.

For the sound generative model, we asked them to account for everything they heard. For the image generative model, we asked them to focus on personality-related characteristics, and not to account for the background colors or patterns. To stop the performers from spending too much time by taking the tasks too seriously, we instructed them to perform each slider manipulation within roughly ten seconds.

To evaluate the results, we asked 13 participants, a completely different set of people from the performers, to be *evaluators* to vote for the tool that provided the final result closest to the target. The evaluators were given the same instructions as the performers on how to account for the data similarities to reduce the difference in task interpretation. The voting was performed for each case of the same performer separately (i.e., the evaluator saw four data, the target and the three outcomes), and the votes were gathered for all performers for the same case.

5.2.2 Stochastically Chosen Initial Data with GANSynth. In the first study, we gave the performers a set (the set size was fixed to eight) of random initial data to start with. This was for a more realistic scenario of exploration. We used GANSynth [Engel et al. 2019], a pre-trained network with NSynth [Engel et al. 2017] dataset. NSynth is a large collection of annotated musical notes sampled from individual instruments with variety of pitches and velocities. GANSynth has a 256-dimensional latent space, and the latent variable follows an n-dimensional normal distribution $\mathcal{N}^n(0, \sigma^2)$, where $\sigma = 1$ is the standard deviation (each element of the variable follows an independent, identically distributed normal distribution $\mathcal{N}(0, \sigma^2)$). We stochastically sampled all the target and initials from $\mathcal{N}^n(0, \sigma^2)$ and used the same set of initials in all cases and with all performers. For each case, each performer picked one of the eight initial data to start with. Because all the data were stochastically sampled, some of the initial data were already close to a target. We did not reject such situation in order not to bias the sampling.

Table 2 left shows the voting percentage from the evaluators. For all the cases, we see that **Ours** got the most votes on average; **Ours** outperformed **Random**, although **Random** is insensitive to the curse of dimensionality and provides more iterations, as well as the generative model being highly nonlinear. In contrast, **SLS-BO** was worse than **Random**. The supplementary material gives the detailed votes and resulting sounds.

5.2.3 Controlled Distance Pairs with PG-GAN. Next, to control the difficulty of the task, we let the performers start with a single given initial data, with a constant distance between the initial and target data. For each case, the same initial data was given to all of the performers. For the test, we used PG-GAN, a pre-trained network with CelebA-HQ [Karras et al. 2018; Liu et al. 2015], a large-scale high resolution (1024×1024) face attributes dataset. In PG-GAN, the latent variable lies in a $n = 512$ dimensional space and follows $\mathcal{N}^n(0, \sigma^2)$, where $\sigma = 1$. We limited the initial and target data to the 95 percentile region centered at the origin, in order to avoid rare data which are likely to contain artifacts. This corresponds to a hypercube of $[-c, c]^n$, where $c = \sqrt{2\sigma^2}\mathrm{erf}^{-1}(0.95^{1/n})$ (see Appendix C for the derivation). We sampled the data pair within the hypercube such that the initial and target data were both distributed in proportion to $\mathcal{N}^n(0, \sigma^2)$ and simultaneously their distance was identical to the expected distance ($\sqrt{2\sigma^2 n}$) between two random points following $\mathcal{N}^n(0, \sigma^2)$. This made the initial and target images fairly distant from each other in the latent space; hence finding the exact target was a fairly hard task. To find such initial and target pairs, we performed rejection sampling as in Appendix D.

Table 2. Voting percentages of each method.

	GANSynth			PG-GAN		
	Ours	Random	SLS-BO	Ours	Random	SLS-BO
Case 1	**43.6%**	37.8%	18.6%	30.1%	**48.1%**	21.8%
Case 2	**45.5%**	30.1%	24.4%	**73.1%**	24.4%	2.6%
Case 3	**56.4%**	17.3%	26.3%	**54.5%**	28.8%	16.7%
Case 4	**50.6%**	38.5%	10.9%	**47.4%**	42.9%	9.6%
Case 5	**45.5%**	23.7%	30.8%	**62.8%**	14.7%	22.4%
Case 6	**46.8%**	29.5%	23.7%	**60.9%**	26.3%	12.8%
Total	**48.1%**	29.5%	22.4%	**54.8%**	30.9%	14.3%

Table 3. Scores (1: random sampling, 5: continuous slider) of the questionnaires for Q1 (effectiveness) and Q2 (user experience).

Performer #	1	2	3	4	5	6	7	8	9	10	11	12	Mean
Q1	5	4	4	4	4	5	4	4	4	5	5	5	4.42
Q2	5	2	3	2	5	4	5	3	5	5	5	3	3.92

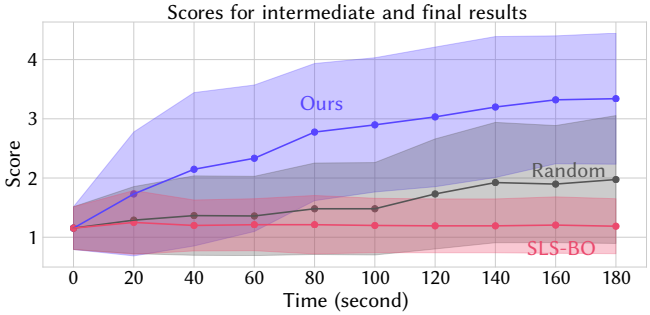


Fig. 12. Absolute scores for Case 2 of Section 5.2.3, evaluated by the 13 evaluators. The solid curves show the averaged score S_{avr} (over the performers and evaluators) as a function of time, with the filled regions corresponding to $S_{\mathrm{avr}} \pm \sigma$, where σ is the standard deviation of the scores.

The initial and target pair of each case, as well as selected final results, are shown in Figure 15. Table 2 right shows the voting results (the detailed votes and resulting images are in our supplementary material). The tendency of the results follow those in Section 5.2.2. **Ours** outperformed **Random** for most cases except Case 1.

To better see how the performance of the intermediate results changes over time, for Case 2, we quantitatively evaluated their qualities. Intermediate results were the "best" thus far during the session, with the "best" judged by the performer. Because improvements in the intermediate results occur at different timings for the three tools, we gathered them every 20 seconds.

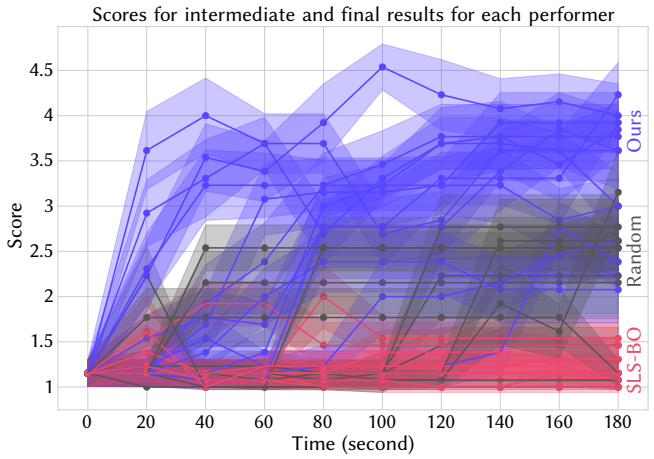


Fig. 13. Absolute scores for the results of Case 2 of Section 5.2.3 for each performer, evaluated by the 13 evaluators. The solid curves show the averaged score S_{avr} (over the evaluators) as a function of time, with the filled regions corresponding to $S_{\text{avr}} \pm 0.4\sigma$, where σ is the standard deviation of the scores.

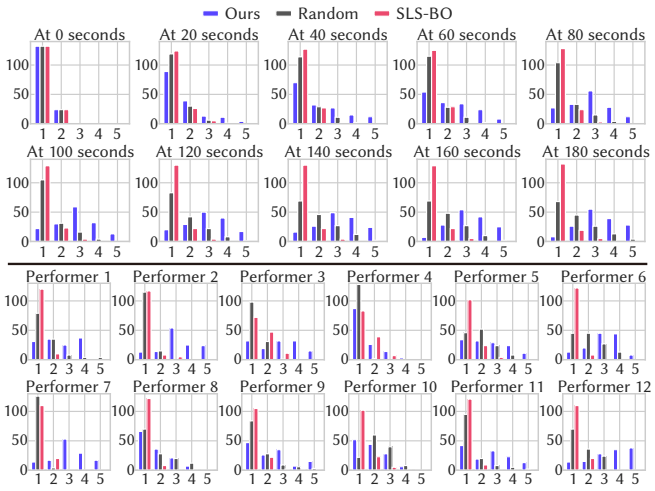
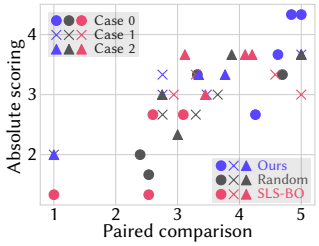


Fig. 14. Histograms of the absolute scores for the results of Case 2 of Section 5.2.3. Each of the top histograms corresponds to the scores of all performers gathered every 20 seconds, and each of the bottom histograms corresponds to the scores of each performer within the search.

Perhaps a rigorous way to quantitatively evaluate the intermediate results is to perform *paired comparisons*, by first gathering a binary answer for every possible pair of the results and then computing a score on a continuous scale for each result (by, e.g., using the choix library [Maystre 2018]). However, paired comparison may be too costly here; the total number of generated results in Case 2 is 201, leading to 20, 100 pairs of results in total, each possibly needing multiple evaluations to suppress the variance in the score.



Instead, we gathered *absolute scores* (1: very bad to 5: very good) from the same set of evaluators. Our pilot study with a small number of images (4 performers for 3 cases) showed a nice correlation between the fully paired comparison and the absolute scoring (see the right inset). Figure 12 shows the absolute scores averaged over all performers and evaluators, Figure 13 shows the separate scores of each performer averaged for all evaluators, and Figure 14 shows the detailed score distribution. Because we randomized the order of the three tools for the performer to use, we expected that on average, each tool would be used with the same amount of prior knowledge of the latent space. Hence, we expected that the experimental bias due to prior knowledge in the results of Figure 12 would be suppressed. However, each line in Figure 13 may reflect this bias.

Because the scores were given by the evaluators, it is possible that they are not monotonically increasing. We can see that the increase in the scores for **Ours** happens consistently throughout the entire session, and it occurs much faster than others, whereas with **Random** the improvement in the scores only happens infrequently during the session and remains constant for most of the time, because an improvement in the image only happens by chance.

To hear directly from the performers about the effectiveness and usability of the slider-based approach as opposed to random selection, we asked them about Q1) which way of viewing different candidates was more effective for finding the desired data, Q2) which way of viewing different candidates was more satisfactory in terms of user experience, and for Q3) free comments. For Q1 and Q2, the performers were asked to give a numerical score (1: random sampling – 5: continuous slider-based). During these three questionnaires, the identity of the tools was still hidden from the performers. Table 3 shows the results of Q1 and Q2. We can see that the slider-based approach was in general scored more highly than random selection in terms of both effectiveness and usability. For Q3, we got positive feedback for the slider-based approach as follows: the slider-based approach allows for fine scale tuning and the modification for specific features, it has the ability to control which direction to go and less chance to obtain bad results, and the performers know when they are getting toward the target. On the negative side, the feature change provided by the slider-based approach can be occasionally complicated and get stuck in some specific features.

6 APPLICATION TO VARIOUS GENERATIVE MODELS

In contrast to the user studies where the explorations were performed by novice users, here we show results performed by an experienced user using our differential subspace search for various generative deep neural networks for images (MNIST-GAN and PG-GAN as in Table 1), sounds (GANSynth), and 3D shapes (IM-GAN). IM-GAN [Chen and Zhang 2019] is a network pre-trained with the ShapeNet [Chang et al. 2015] dataset, a richly-annotated, large-scale dataset of 3D shapes. Our user interface is common to these different generative models (Figure 16).

Figure 17 shows the initial, found, and target data for these generative models (for sounds, refer to supplementary material). The user

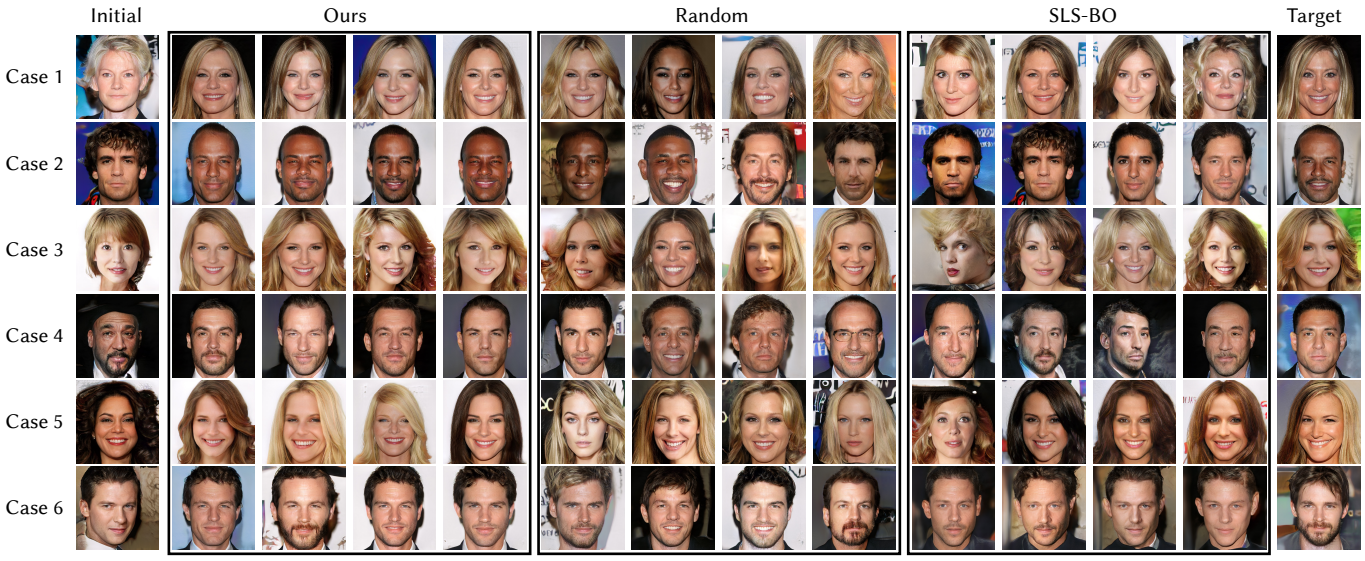


Fig. 15. Initial and target pairs, as well as selected final results for **Ours**, **Random**, and **SLS-BO**, for the six cases of the user study using PG-GAN. Refer to the supplementary material for the full set of images with a higher resolution.

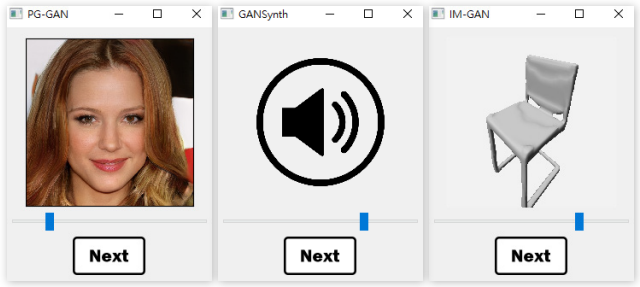


Fig. 16. Our interface using our differential subspace search is common to various generative models for images, sounds, and 3d models. The user can freely explore the subspace of the data space by tweaking the slider and click the next button to reconstruct the subspace at the selected point.

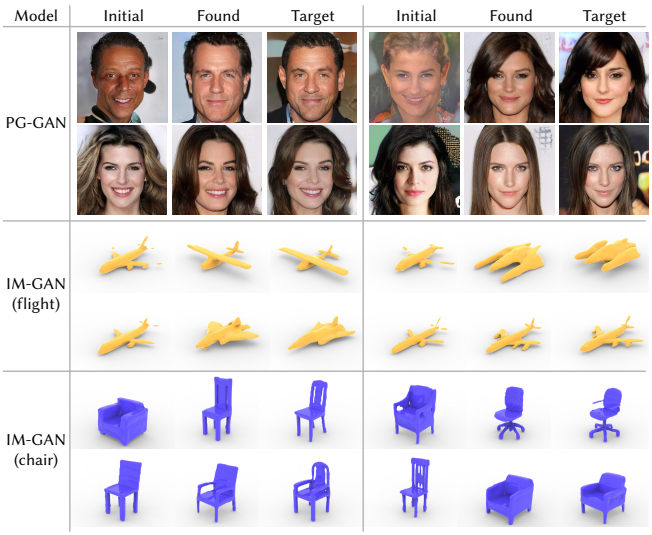


Fig. 17. Initial and target data pairs and data found by an experienced user for the generative models for images and 3D models.

was able to find results fairly close to the target within 5 minutes. Interestingly, especially for the images, such data were sometimes found against pretty different backgrounds; the found and target point in some cases were distant in terms of the L2-norm in the data space, but were perceived as being close to each other. With our differential subspace search, the user retains such a perceptual metric (i.e., the goodness function) to judge on their own.

7 ADDITIONAL PILOT STUDY OF EXPLORATORY TASK

We conducted an informal pilot study to better understand how our tool can help exploration in creative design tasks from a qualitative viewpoint. We asked a designer, who draws manga as a hobby, to use our tool with the PG-GAN generative model. Specifically, we asked her to come up with four fictional characters that she would like to put in her original story by using our tool. We show the exploration results in Figure 18. After the exploratory task, we conducted an informal interview.[2]

About the usability of our tool, she said it was *"very easy"* for her to *"learn how to use the tool and how it works."* She also commented that our tool would take her *"much less time"* to become familiar with it than typical tools that *"provide many parameters."* About the use of our tool in the design process, her impression was that *"the tool is helpful when the design concept is not that clear."* She also

[2]We conducted the interview in the interviewee's native language, and thus the following comments are a translation from the original language into English.

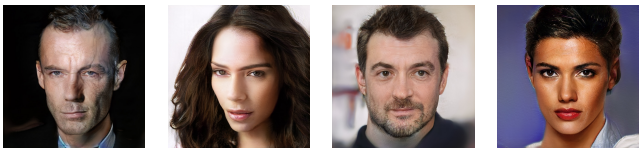


Fig. 18. Characters found by the designer when using our method in the additional pilot study. Descriptions of the characters are given below.

commented that our tool helped her *"make the concept clearer after seeing one character after another."* These comments support the finding that our tool can facilitate exploratory design scenarios. We asked her about the ability to fine-tune the data. She said she was *"able to achieve detailed modifications"* with our tool *"even though the change does not always align"* with her desires. This was thanks to *"the smooth feature change during slider tweaking."* However, she also requested an additional feature for making direct *"detailed modifications"* to important features such as *"eyes."* We believe this feature is possible by integrating domain-specific techniques (e.g., [Bau et al. 2019; Brock et al. 2017]), and the comment suggests this direction would be an important future work.

8 DISCUSSION

8.1 On Finding the Exact Target

Finding the exact target in the latent space requires us to find the global optimum for (1). Since the generative model is in general non-concave and highly nonlinear, such an optimization is NP-hard, and finding the exact target is impractical. We believe that our method offers a reasonable solution to such a hard task; with it, the user can likely find a result with a resemblance to what they had in mind in a practical amount of time (e.g., within 5 minutes).

8.2 General vs. Domain-specific Search

Our method aims for efficient exploration in high-dimensional latent spaces, which we believe is a vital step toward fully eliciting the power of generative models. We intentionally tried to construct our method to be as simple and general as possible, for it to be applicable to various generative models. However, we are not against domain-specific approaches, which, we believe, are complementary to our work.

Some GAN based generative models may admit directions (either straight or curved) that represent semantic meanings [Goetschalckx et al. 2019; Jahanian et al. 2020; Shen et al. 2020; Yang et al. 2019]. Our method does not exploit such useful properties. As a result, the proposed directions would in general change multiple attributes at the same time and result in unintuitive manipulation. It would be important to combine our method with domain-specific approaches to exploit such meaningful directions to allow for manipulations limited to a manner the user intends.

It would be also interesting to uniformize perceptual changes along search directions. For instance, we could incorporate a (differentiable) map $\boldsymbol{h} : \mathcal{X} \to \mathcal{X}'$ from the original data space $\mathcal{X}$ to a data space $\mathcal{X}'$ that is more perceptually uniform (e.g., CIELAB for colors and mel-scaled spectrogram representation for sounds)

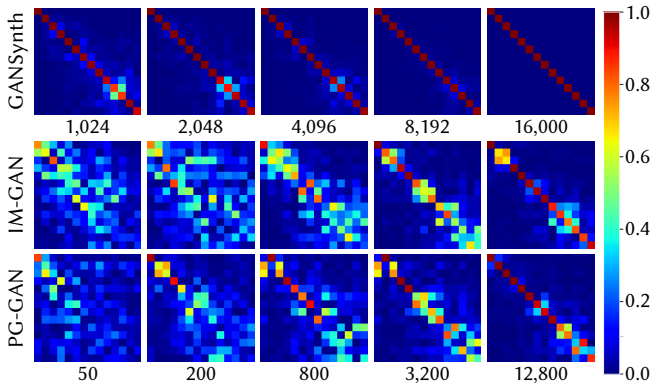


Fig. 19. Validation of our approximate Jacobian computation and singular value decomposition. For each of the three models, we stochastically sampled a point in the latent space, computed the ground truth Jacobian and its singular value decomposition without any approximation, and sorted the right singular vectors (the column vectors of $\mathbf{V}$) according to their corresponding singular values. Likewise, we computed the version with different levels of approximations, with the approximation detailed in Section 8.4. Then, we sorted the right singular vectors. For GANSynth, the numbers shown below the plots are the reduced data dimension after down-sampling, and for PG-GAN and IM-GAN, the numbers are the data dimension for the Jacobian computation. For both sets of right singular vectors, we took the top 14 vectors and computed the dot products for all their combinations, to measure the similarities of the directions. Because the user can move in positive and negative directions, we took the absolute values of the dot product. The results are presented in matrix form (the exact and approximate versions are in the rows and columns, respectively, and closer to the top and left corresponds to larger singular values). The ideal case is the identity matrix, but we believe our approximation is doing a good job, because the absolute cosine values are much larger (the green, yellow and reddish regions) than the very low expected absolute cosine value, 0.035, between two uniformly random vectors in a 512-dimensional space, according to (25). The order of the vectors may be inverted for those with close singular values, due to the randomness in the approximation. Also, we are interested in more accurately hitting the true directions with larger singular values than those with smaller ones.

and use the modified Jacobian $\mathbf{J}' = \partial \boldsymbol{h}(\boldsymbol{f}(\boldsymbol{z}^{(k)}))/\partial \boldsymbol{z}$. We could also reparameterize the latent variables according to Lindow et al. [2012].

In addition, an application-specific way of presenting initial points may be more beneficial than giving a single random point. For instance, it is possible to show the user a set of stochastically chosen points in a style like *Design Galleries* [Marks et al. 1997] or *Brainstorm* [Adobe 2017].

8.3 Different Subspace Dimensionalities

With our differential subspace search approach, we explored the case of $l = 1$, choosing a one-dimensional slider as the interface. It would be interesting to investigate interfaces in higher dimensions. We have also demonstrated the option of using our subspace construction to find a moderately low dimensional ($l = 6$) subspace and combining it with the sequential line search technique (as in **Hybrid-6D** in Section 4.5). Investigating the performance change due to a different choice of l would also be interesting.

8.4 Approximate Computation for Faster Performance

As long as the computation is fast enough, we can use automatic differentiation to compute the full Jacobian $\mathbf{J} \in \mathbb{R}^{m \times n}$ before performing singular value decomposition. We did so for the synthetic examples. If the performance with this option is not high enough, which may be caused by the availability of only modest computational resources or lack of an efficient implementation of automatic differentiation, there are several options to incorporate approximations for faster performance. For the MNIST case, we employed finite differences. For the other generative models described in this paper, we employed further approximations below for interactive updates (5 seconds at most, for a computer with an Intel® Core™ i7-8086K 4GHz CPU, 16GB main memory, and an NVIDIA® GeForce® GTX 1080 GPU with 8GB memory).

For PG-GAN and IM-GAN, we stochastically sampled 50 dimensions (i.e., $m = 50$) from the data space and then performed automatic differentiation to obtain a reduced dimensional Jacobian $\mathbf{J}' \in \mathbb{R}^{50 \times n}$. This effectively saved time in the Jacobian computation as well as in the singular value decomposition. For GANSynth, we used finite differences to compute the full Jacobian and then down-sampled the data dimension to 1024 (by removing rows). The use of finite differences here was because of the unavailability of an efficient implementation of automatic differentiation in the corresponding library, while the down-sampling was for accelerating the singular value decomposition.

It is interesting that even with such a small number of elements to approximate the Jacobian or the singular value decomposition, we could still recover the effective directions, as in Figure 19. This is perhaps because of the highly skewed distribution of the singular values of the Jacobian, which is effectively 'low' rank (if we truncate the singular values at a small threshold) and in turn allows for nice recovery from only a small fraction of its elements [Candès and Recht 2009].

9 CONCLUSIONS AND FUTURE WORK

We have proposed *differential subspace search*, a human-in-the-loop technique for exploring high-dimensional latent spaces of generative models, whose dimensionality can be as high as a few hundred. Our method does not rely on domain- or data-specific assumptions and takes a pre-trained network as input. The key is to iteratively construct a low-dimensional subspace based on local analytics of the generative model, and let the user perform search in this subspace. We have demonstrated its utility via 1D slider interfaces.

To construct the subspace, we apply singular value decomposition to the Jacobian of the generative model and form a subspace spanned by a few singular vectors stochastically selected on the basis of their corresponding singular values, to maintain ergodicity. Experimental results show that our method can better optimize synthetic black-box objective functions than the alternatives. Furthermore, we conducted a user study using complex generative models and found that our method enables the user to efficiently explore high-dimensional latent spaces.

At a high level, we have only scratched the surface of various ways to use the skewed distribution of the singular values of the Jacobian of the generative model for efficiently exploring a high-dimensional latent space with a low-dimensional interface. This idea can be extended in multiple ways. First, our technique builds on (perhaps) the simplest (yet powerful) optimization framework, gradient ascent, which uses only first-order information. It would be interesting to explore how higher-order information, such as curvature, can facilitate the search process, e.g., by allowing exploration in a curved subregion. Studying the utility of our method for choosing other ($l \neq 1$) dimensional subspaces, as well as its combination with conditional generative models or domain-specific approaches, would be also interesting. In this work, we only examined a small number of generative models; it would be interesting to see how well our method works for other models, such as body shapes and hair styles. Extending our method to high-dimensional parameter or design spaces (not necessarily for deep generative models) is another interesting direction.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful suggestions and discussions. This work was supported in part by a grant from JST CREST, JPMJCR17A1 (HCI for Machine Learning), Japan, a JSPS Grant-in-Aid for Scientific Research (A) 18H04106, Japan, and grants from the Ministry of Science and Technology of Taiwan, MOST-107-2221-E-011-114-MY2 and MOST-107-2221-E-011-112-MY2.

REFERENCES

Adobe. 2017. Using the Brainstorming tool in After Effects CS6. Retrieved April 19, 2020 from https://helpx.adobe.com/after-effects/atv/cs6-tutorials/brainstorming.html.

David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. 2019. GAN Dissection: Visualizing and Understanding Generative Adversarial Networks. In *Proc. of ICLR 2019*. https://openreview.net/forum?id=Hyg_X2C5FX

Yoshua Bengio, Aaron Courville, and Pascal Vincent. 2013. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (2013), 1798–1828. https://doi.org/10.1109/TPAMI.2013.50

Eric Brochu, Nando de Freitas, and Abhijeet Ghosh. 2007. Active Preference Learning with Discrete Choice Data. In *Advances in Neural Information Processing Systems 20 (NIPS 2007)*. 409–416. https://dl.acm.org/doi/10.5555/2981562.2981614

Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. 2017. Neural Photo Editing with Introspective Adversarial Networks. In *Proc. of ICLR 2017*. https://openreview.net/forum?id=HkNKFiGex

Emmanuel J. Candès and Benjamin Recht. 2009. Exact Matrix Completion via Convex Optimization. *Foundations of Computational Mathematics* 9, 6 (2009), 717–772. https://doi.org/10.1007/s10208-009-9045-5

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. 2015. ShapeNet: An Information-Rich 3D Model Repository. https://arxiv.org/abs/1512.03012

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 29 (NIPS 2016)*. 2172–2180. https://dl.acm.org/doi/abs/10.5555/3157096.3157340

Zhiqin Chen and Hao Zhang. 2019. Learning implicit fields for generative shape modeling. In *Proc. of CVPR 2019*. 5939–5948. https://doi.org/10.1109/CVPR.2019.00609

Chris Donahue, Julian McAuley, and Miller Puckette. 2018. Adversarial Audio Synthesis. https://arxiv.org/abs/1802.04208

Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. 2019. GANSynth: Adversarial Neural Audio Synthesis. In *Proc. of ICLR 2019*. https://openreview.net/forum?id=H1xQVn09FX

Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. 2017. Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. In *Proc. of Machine Learning Research - Volume 70 (ICML 2017)*. JMLR.org, 1068–1077. https://dl.acm.org/doi/abs/10.5555/3305381.3305492

Lore Goetschalckx, Alex Andonian, Aude Oliva, and Phillip Isola. 2019. GANalyze: Toward Visual Definitions of Cognitive Image Properties. In *Proc. of ICCV 2019*.

5744–5753.
Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. 2017. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*. 5769–5779. https://dl.acm.org/doi/abs/10.5555/3295222.3295327
Toby Chong Long Hin, I-Chao Shen, Issei Sato, and Takeo Igarashi. 2019. Interactive Subspace Exploration on Generative Image Modelling. https://arxiv.org/abs/1906.09840
Ali Jahanian, Lucy Chai, and Phillip Isola. 2020. On the "steerability" of generative adversarial networks. In *Proc. of ICLR 2020*. https://openreview.net/forum?id=HylsTT4FvB
Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *Proc. of ICLR 2018*. https://openreview.net/forum?id=Hk99zCeAb
Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, and Max Welling. 2014. Semi-Supervised Learning with Deep Generative Models. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*. 3581–3589. https://dl.acm.org/doi/10.5555/2969033.2969226
Yuki Koyama, Daisuke Sakamoto, and Takeo Igarashi. 2014. Crowd-Powered Parameter Analysis for Visual Design Exploration. In *Proc. of UIST 2014*. 65–74. https://doi.org/10.1145/2642918.2647386
Yuki Koyama, Issei Sato, Daisuke Sakamoto, and Takeo Igarashi. 2017. Sequential Line Search for Efficient Visual Design Optimization by Crowds. *ACM Transactions on Graphics* 36, 4 (Proc. of SIGGRAPH 2017) (July 2017), 48:1–48:11. https://doi.org/10.1145/3072959.3073598
Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86, 11 (1998), 2278–2324. https://doi.org/10.1109/5.726791
Norbert Lindow, Daniel Baum, and Hans-Christian Hege. 2012. Perceptually Linear Parameter Variations. *Computer Graphics Forum* 31, 2–4 (Proc. of EUROGRAPHICS 2012) (May 2012), 535–544. https://doi.org/10.1111/j.1467-8659.2012.03054.x
Seppo Linnainmaa. 1970. *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*. Master's thesis. University of Helsinki, Finland.
Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proc. of ICCV 2015*. 3730–3738. https://doi.org/10.1109/ICCV.2015.425
Joe Marks, Brad Andalman, Paul A. Beardsley, William T. Freeman, Sarah F. Friskken-Gibson, Jessica K. Hodgins, Thomas Kang, Brian V. Mirtich, Hanspeter Pfister, Wheeler Ruml, Kathy Ryall, Joshua E. Seims, and Stuart M. Shieber. 1997. Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation. In *Proc. of SIGGRAPH '97*. 389–400. https://doi.org/10.1145/258734.258887
Lucas Maystre. 2018. *Efficient Learning from Comparisons*. Ph.D. Dissertation. École Polytechnique Fédérale de Lausanne.
Mehdi Mirza and Simon Osindero. 2014. Conditional Generative Adversarial Nets. https://arxiv.org/abs/1411.1784
Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral Normalization for Generative Adversarial Networks. In *Proc. of ICLR 2018*. https://openreview.net/forum?id=B1QRgziT-
Riccardo Moriconi, Marc P. Deisenroth, and K. S. Sesh Kumar. 2019. High-dimensional Bayesian optimization using low-dimensional feature spaces. https://arxiv.org/abs/1902.10675
Yurii Nesterov. 2018. *Lectures on Convex Optimization*. Springer. https://doi.org/10.1007/978-3-319-91578-4
Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. The MIT Press.
Salah Rifai, Yann N. Dauphin, Pascal Vincent, Yoshua Bengio, and Xavier Muller. 2011. The Manifold Tangent Classifier. In *Advances in Neural Information Processing Systems 24 (NIPS 2011)*. 2294–2302. https://dl.acm.org/doi/10.5555/2986459.2986715
Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61 (2015), 85–117. https://doi.org/10.1016/j.neunet.2014.09.003
Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE* 104, 1 (2015), 148–175. https://doi.org/10.1109/JPROC.2015.2494218
Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. 2020. Interpreting the Latent Space of GANs for Semantic Face Editing. In *Proc. of CVPR 2020*. To appear.
Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical Bayesian Optimization of Machine Learning Algorithms. In *Advances in Neural Information Processing Systems 25 (NIPS 2012)*. 2951–2959. https://dl.acm.org/doi/abs/10.5555/2999325.2999464
Hideyuki Takagi. 2001. Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation. *Proceedings of the IEEE* 89, 9 (Sep. 2001), 1275–1296. https://doi.org/10.1109/5.949485
Jerry O. Talton, Daniel Gibson, Lingfeng Yang, Pat Hanrahan, and Vladlen Koltun. 2009. Exploratory Modeling with Collaborative Design Spaces. *ACM Transactions on Graphics* 28, 5 (Proc. of SIGGRAPH Asia 2009) (Dec. 2009), 167:1–167:10. https://doi.org/10.1145/1618452.1618513
Nobuyuki Umetani. 2017. Exploring Generative 3D Shapes Using Autoencoder Networks. In *SIGGRAPH Asia 2017 Technical Briefs*. 24:1–24:4. https://doi.org/10.1145/3145749.3145758
Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Freitas. 2016. Bayesian Optimization in a Billion Dimensions via Random Embeddings. *Journal of Artificial Intelligence Research* 55, 1 (February 2016), 361–387. https://doi.org/10.1613/jair.4806
Ceyuan Yang, Yujun Shen, and Bolei Zhou. 2019. Semantic Hierarchy Emerges in Deep Generative Representations for Scene Synthesis. https://arxiv.org/abs/1911.09267
Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. 2016. Generative Visual Manipulation on the Natural Image Manifold. In *Computer Vision – ECCV 2016*. 597–613. https://doi.org/10.1007/978-3-319-46454-1_36

A SEQUENTIAL-LINE-SEARCH CONDITIONS

The sequential-line-search (SLS) method [Koyama et al. 2017] needs some conditions to be selected as follows.

- **Kernel choice**: The original paper used the *automatic relevance determination (ARD) squared exponential* kernel [Rasmussen and Williams 2006] for the Gaussian process (GP) prior. Another popular choice is the *ARD Matérn 5/2* kernel, which was recommended by Snoek et al. [2012]. It is unknown which kernel is more suitable for SLS.
- **Kernel hyperparemters**: The original paper assumed a log-normal prior distribution: $\mathcal{LN}(\log 0.5, 0.1)$ (whose median is 0.5) for every kernel hyperparameter. As we handle higher-dimensional problems than theirs, it is not trivial whether this setting is still feasible or other settings are necessary.
- **Joint estimation**: The original SLS jointly estimates both the latent goodness values on the observed points and the appropriate kernel hyperparameters in the form of the *maximum a posteriori (MAP) estimate*. However, we could also estimate only the goodness values and leave the hyperparameters fixed. This alternative approach reduces the computation cost and also has a chance to improve the optimization convergence if we provide appropriate hyperparameters.

To determine which conditions should be used in our experiments, we ran optimization sequences for different combinations of the conditions using the synthetic test function defined in (13), (15) as well as an isotropic Gaussian function, both in 8-, 64-, and 512-dimensional spaces. As a result, we found that the original conditions performed reasonably well, and there were no obvious differences in optimization convergence between the different conditions (except when we set very small kernel hyperparameter values, which resulted in slower convergence). Using a fixed set of hyperparameters was helpful to reduce the computation cost, but the reduction was marginal and not significant for a reasonable iteration count (< 50) in human-in-the-loop use. Therefore, we decided to use the same conditions as in the original paper.

B ARCHITECTURE OF MNIST GENERATOR

The MNIST generator used in Section 5.1 is detailed in Table 4.

Table 4. Network architecture of the MNIST generator.

Generator	Output Size	k_{size}	k_{filters}	Activation
$z \sim U(-1, 1)$	(64)	-	-	-
dense	(128)	-	-	ReLU
dense	(512)	-	-	ReLU
reshape	(1, 1, 512)	-	-	-
deconv2d	(3, 3, 256)	3	256	ReLU
deconv2d	(7, 7, 128)	3	128	ReLU
deconv2d	(14, 14, 64)	3	64	ReLU
deconv2d	(28, 28, 1)	3	1	sigmoid

C ON THE BOX SIZE OF THE 95 PERCENTILE REGION

For the one-dimensional case, the probability p that a random variable generated according to $\mathcal{N}(0, \sigma^2)$ will lie in $[-c, c]$ is given by

$$p = \int_{-c}^{c} \frac{dx}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2}\right) = 2\int_{0}^{c} \frac{dx}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2}{2\sigma^2}\right). \quad (16)$$

Let $t = \frac{x}{\sqrt{2\sigma^2}}$, we have $dt = \frac{dx}{\sqrt{2\sigma^2}}$, $t : 0 \mapsto \frac{c}{\sqrt{2\sigma^2}}$ as $x : 0 \mapsto c$ and

$$p = \frac{2}{\sqrt{\pi}} \int_{0}^{\frac{c}{\sqrt{2\sigma^2}}} \exp(-t^2) dt = \text{erf}\left(\frac{c}{\sqrt{2\sigma^2}}\right). \quad (17)$$

Hence, we arrive at

$$c = \sqrt{2\sigma^2}\text{erf}^{-1}(p). \quad (18)$$

Next, because of the independence of the dimensions in $\mathcal{N}^n(0, \sigma^2)$, if we set $p = 0.95^{1/n}$, the probability that an n-dimensional point lies in $[-c, c]^n$ becomes $p^n = 0.95$.

D GENERATING RANDOM DATA PAIRS

First, we forget about the hypercube and generate a pair of data ($\boldsymbol{z}_A$ and $\boldsymbol{z}_B$) such that their distance is identical to a given one L: $\|\boldsymbol{z}_A - \boldsymbol{z}_B\| = L$ and that the probability to generate such a pair $p(\boldsymbol{z}_A, \boldsymbol{z}_B)$ is proportional to $q(\boldsymbol{z}_A)q(\boldsymbol{z}_B)$, where $q(\boldsymbol{z}) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp(-\sum_{i=1}^{n} \frac{z_i^2}{2\sigma^2})$ is the n-dimensional normal distribution $\mathcal{N}^n(0, \sigma^2)$. We parameterize $\boldsymbol{z}_A$ and $\boldsymbol{z}_B$ using their mid point $\boldsymbol{z}_C = \frac{1}{2}(\boldsymbol{z}_A + \boldsymbol{z}_B)$ and the unit vector $\boldsymbol{d}$ pointing from $\boldsymbol{z}_A$ toward $\boldsymbol{z}_B$, so that $\boldsymbol{z}_A = \boldsymbol{z}_C - \frac{L}{2}\boldsymbol{d}$ and $\boldsymbol{z}_B = \boldsymbol{z}_C + \frac{L}{2}\boldsymbol{d}$. Because of the isotropy of the normal distribution, $\boldsymbol{d}$ is uniformly distributed in the n-dimensional unit hypersphere. So, we first sample $\boldsymbol{d}$, by sampling each component of $\boldsymbol{d}$ according to $\mathcal{N}(0, 1)$ and normalizing $\boldsymbol{d}$. Next, with $\boldsymbol{d}$ fixed, we have

$$\begin{aligned} q(\boldsymbol{z}_A)q(\boldsymbol{z}_B) &= \left(\frac{1}{2\pi\sigma^2}\right)^n \exp\left(-\frac{1}{2\sigma^2}\sum_i \left(z_{Ci} - \frac{L}{2}d_i\right)^2 - \frac{1}{2\sigma^2}\sum_j \left(z_{Cj} + \frac{L}{2}d_j\right)^2\right) \\ &= \left(\frac{1}{2\pi\sigma^2}\right)^n \exp\left(-\frac{1}{2\sigma^2}\sum_i \left(2z_{Ci}^2 + \frac{L^2}{2}d_i^2\right)\right) \propto \exp\left(-\frac{\sum_i z_{Ci}^2}{2\cdot\left(\frac{\sigma}{\sqrt{2}}\right)^2}\right). \end{aligned} \quad (19)$$

Thus, $\boldsymbol{z}_C$ can be sampled according to $\mathcal{N}^n\left(0, \left(\frac{\sigma}{\sqrt{2}}\right)^2\right)$. After we sample $\boldsymbol{z}_C$, we check whether $\boldsymbol{z}_A$ and $\boldsymbol{z}_B$ lie inside of the hypercube. If not, we redo the sampling from the beginning.

E ALIGNMENT BETWEEN RANDOM VECTORS

Here, we consider the alignment between two unit vectors $\boldsymbol{s}, \boldsymbol{x} \in \mathbb{R}^n$, each uniformly distributed in the $(n-1)$-sphere $\mathcal{S}^{n-1}$, with their alignment measured by the length $L_{\boldsymbol{x}_p}$ of the projected vector $\boldsymbol{x}_p$ of $\boldsymbol{x}$ onto $\boldsymbol{s}$ (i.e, $L_{\boldsymbol{x}_p} = |\boldsymbol{s} \cdot \boldsymbol{x}|$). Without loss of generality, we can set $\boldsymbol{s}$ to $(1, 0, \ldots, 0)^T$.

We parametrize $\boldsymbol{x}$ by using hyperspherical coordinates, a generalization of the usual spherical coordinates to $\mathbb{R}^n$, as

$$\boldsymbol{x} = (\cos\phi_1, \cdots, \sin\phi_1 \cdots \sin\phi_{n-2}\cos\phi_{n-1}, \sin\phi_1 \cdots \sin\phi_{n-1})^T, \quad (20)$$

where $0 \le \phi_1, \cdots, \phi_{n-2} \le \pi$ and $0 \le \phi_{n-1} \le 2\pi$, and compute the expectation $E[L_{\boldsymbol{x}_p}]$.

Under the hyperspherical parametrization, $L_{\boldsymbol{x}_p} = |\cos\phi_1|$ and the distribution $\Pr(0 \le \phi_1 \le \Phi)$ of ϕ_1 becomes

$$\Pr(0 \le \phi_1 \le \Phi) = \frac{S^{n-1}(0 \le \phi_1 \le \Phi)}{S^{n-1}}, \quad (21)$$

where S^{n-1} is the surface area of the $\mathcal{S}^{n-1}$, and $S^{n-1}(0 \le \phi_1 \le \Phi)$ is the area of its subset such that $0 \le \phi_1 \le \Phi$. S^{n-1} is given by

$$\begin{aligned} S^{n-1} &= \int_{\phi_1=0}^{\phi_1=\pi} \cdots \int_{\phi_{n-2}=0}^{\phi_{n-2}=\pi} \int_{\phi_{n-1}=0}^{\phi_{n-1}=2\pi} \left|\det\left(\frac{\partial \boldsymbol{x}}{\partial(\phi_1, \cdots, \phi_{n-1})^T}\right)\right| d\phi_1 \cdots d\phi_{n-1} \\ &= \int_{\phi_1=0}^{\phi_1=\pi} \cdots \int_{\phi_{n-2}=0}^{\phi_{n-2}=\pi} \int_{\phi_{n-1}=0}^{\phi_{n-1}=2\pi} \sin^{n-2}\phi_1 \sin^{n-3}\phi_2 \cdots \sin\phi_{n-2} d\phi_1 \cdots d\phi_{n-1} \\ &= \int_{\phi_1=0}^{\phi_1=\pi} \sin^{n-2}\phi_1 d\phi_1 \cdots \int_{\phi_{n-2}=0}^{\phi_{n-2}=\pi} \sin\phi_{n-2} d\phi_{n-2} \int_{\phi_{n-1}=0}^{\phi_{n-1}=2\pi} d\phi_{n-1}. \end{aligned} \quad (22)$$

By letting $I_k = \int_{\psi=0}^{\psi=\pi} \sin^k\psi d\psi$, we have $S^{n-1} = I_{n-2}I_{n-3}\cdots I_1 2\pi$. Likewise, $S^{n-1}(0 \le \phi_1 \le \Phi) = \int_{\phi_1=0}^{\phi_1=\Phi} \sin^{n-2}\phi_1 d\phi_1 I_{n-3}\cdots I_1 2\pi$. Hence, the probability density $p(\phi_1)$ is

$$p(\phi_1)d\phi_1 = d\Pr(0 \le \phi_1 \le \Phi) = \frac{dS^{n-1}(0 \le \phi_1 \le \Phi)}{S^{n-1}} = \frac{\sin^{n-2}\phi_1 d\phi_1}{I_{n-2}}. \quad (23)$$

Hence, we have

$$\begin{aligned} E[L_{\boldsymbol{x}_p}] &= \int_0^{\pi} |\cos\phi_1| \frac{\sin^{n-2}\phi_1 d\phi_1}{I_{n-2}} \\ &= \frac{2}{I_{n-2}} \int_0^{\frac{\pi}{2}} \cos\phi_1 \sin^{n-2}\phi_1 d\phi_1 = \frac{2}{(n-1)I_{n-2}}. \end{aligned} \quad (24)$$

With the beta function $B(a, b) = 2\int_0^{\frac{\pi}{2}} \sin^{2a-1}\theta \cos^{2b-1}\theta d\theta$, we have $I_{n-2} = B\left(\frac{n-1}{2}, \frac{1}{2}\right)$. Because of the relation $B(a, b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$, where $\Gamma(x) = \int_0^{\infty} t^{x-1}e^{-t}dt$ is the gamma function, and $\Gamma(1/2) = \sqrt{\pi}$, we have $I_{n-2} = \sqrt{\pi}\Gamma\left(\frac{n-1}{2}\right)/\Gamma\left(\frac{n}{2}\right)$. Thus, we arrive at

$$E[L_{\boldsymbol{x}_p}] = \frac{2}{(n-1)\sqrt{\pi}} \frac{\Gamma\left(\frac{n}{2}\right)}{\Gamma\left(\frac{n-1}{2}\right)} = \frac{2}{n\sqrt{\pi}} \frac{\Gamma\left(\frac{n+2}{2}\right)}{\Gamma\left(\frac{n+1}{2}\right)}. \quad (25)$$

Asymptotically, $E[L_{\boldsymbol{x}_p}] \approx \frac{2}{\sqrt{2\pi n}}$ and is of order $1/\sqrt{n}$.