Agnostic latent diversity enhancement in generative modeling

Mariia Zameshina, Mathurin Videau, Alessandro Leite, Marc Schoenauer, Laurent Najman, Olivier Teytaud

Agnostic latent diversity enhancement in generative modeling

Mariia Zameshina
Universite Paris-Est, Equipe A3SI, ESIEE Paris
mariia.zameshina@esiee.fr

Mathurin Videau
TAU, Inria Saclay, LISN

Alessandro Leite
TAU, Inria Saclay, LISN

Marc Schoenauer
TAU, Inria Saclay, LISN

Laurent Najman
Universite Paris-Est, Equipe A3SI, ESIEE Paris

Olivier Teytaud
TAU, Inria Saclay, LISN

Abstract

Generative modeling methods can generate images from textual or visual inputs. However, diversity in the generated images persists as a major challenge of the existing approaches. In this work, we address this issue head-on by demonstrating that: (a) the diversity of a generated batch of images is intrinsically linked to the diversity within the latent variables; (b) leveraging the geometry of the latent space, we can establish an effective metric for quantifying diversity; and (c) employing this insight allows one to achieve a significantly enhanced diversity in image generation beyond the capabilities of traditional random independent sampling. This advancement is consistent across a variety of generative models, including Generative Adversarial Networks (GANs) and latent diffusion models. To facilitate further research and application in this field, we are also releasing a comprehensive package that enables easy reproduction of our experiments. We integrate our contributions into a widely recognized tool for generative image modeling, ensuring that our improvements are accessible to the broader community.

1 Introduction

Latent generative modeling involves learning a latent representation of the data that captures its underlying structure and using this representation to generate new data points. Specifically, the term "latent" refers to hidden variables or features that are inferred from the observed data without being directly observed. In this context, latent generative modeling aims to learn a model that map observed data into a latent space (LS), where each point represents a set of underlying features or factors that explain the variation in the data. This latent space has typically a lower dimension than the original one, which helps one capturing essential features while reducing redundancy. Examples of latent generative models include Variational Autoencoders (VAEs) [1] and Generative Adversarial Networks (GANs) [2]. A major application of latent generative modeling is text-to-image synthesis [3, 4], frequently referred to as "text2image". This is a type of generative modeling where the goal is to generate realistic images from textual descriptions. In this context, a model takes a textual description as input and generates an image that corresponds to that description. While

Preprint. Under review.

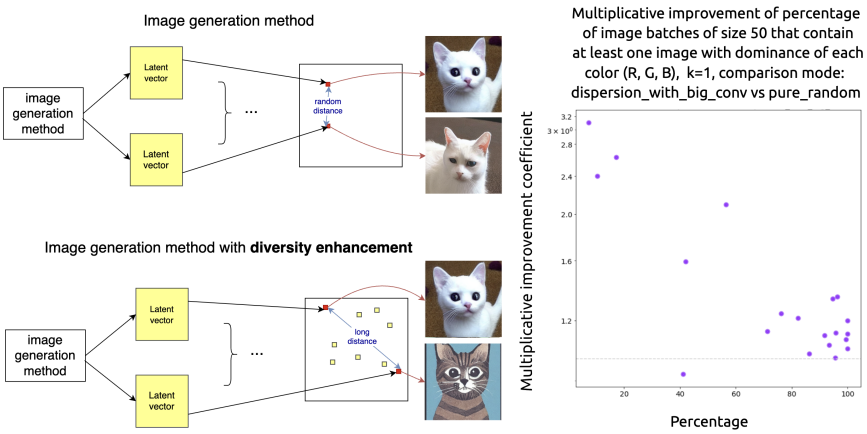


Figure 1: Left: General schema for diversity enhancement. A vanilla image generation method is compared to a method with enhanced diversity. Instead of being randomly generated, latent vectors are chosen on the basis of their pairwise distance, which results in better image diversity. Right: Comparison of dispersion-with-big-conv and standard SD (pure-random) for various prompts w.r.t the multiplicative percentage of batches containing images with at least 2 different dominant colors for the following parameters: $K = 1$, batch size $= 50$. In general we get better improvement for least represented groups.

latent generative modeling works well for text2image [3], it suffers from diversity loss and mode collapse [5–8]. To alleviate these issues, we propose to replace the regular, pure random generator of point configurations by another generator, that produces a well-distributed point configuration. In particular, well-distributed point configurations [9] refer to the arrangements of points in a space where the points are spread out evenly and uniformly, often with respect to certain criteria or constraints.

The concept of well-distributed point configurations depends on the context and the specific requirements of the application. Some common criteria for defining well-distributed point configurations include:

1. **Uniformity**: Points should be distributed uniformly across the space, meaning that there are no regions with significantly higher or lower point density compared to others.
2. **Packing density**: Points should be packed densely enough to cover the space adequately but not so densely that they become clustered or overlapping. Achieving an optimal packing density depends on the dimensions of the space and the desired properties of the point configuration.
3. **Symmetry**: In some cases, symmetry or regularity in the arrangement of points is desired, especially in geometric modeling or tessellation applications.
4. **Smoothness**: Points should be distributed smoothly across the space, without abrupt changes in density or clustering.

Well distributed point configurations have been applied to many settings [9]. The literature on well distributed point configurations contains only few tools for the case of small point sets in high-dimensional spaces, and our goal is to fill this gap. Figure 1 provides a general scheme for the application of well distributed point configurations to latent generative modeling to improve the diversity of generated images, instead of 50 random independent images usually used by existing approaches. More precisely, a well distributed point configurations is generated in the LS, so that the batch of generated images is more diverse than in the vanilla case. In the present paper, we analyze the max-pooling method [10], compared to a vast sampling of tools for high-dimensional well distributed point configurations. After careful examination, we show that dispersion combined with convolution (a) approximates the performance of max-pooling in the low-dimensional case and (b) performs better in large latent spaces used in modern latent diffusion models. We note that the method described in [11] includes another strategy to ensure diversity; it considers a different context, and the

proposed strategy is entangled in the diffusion process [12]. Due to this entanglement, [11] cannot be applied to other methods, such as SDXL-Turbo [13].

The rest of this paper is organized as follows. Section 2.1 introduces the metrics for assessing the quality of point configurations. We explore the metrics designed to evaluate the distribution of images in Section 2.2. Section 3 describes the tools for generating well-distributed point configurations, while Section 4 details the experimental results derived from these tools. Appendix C discusses how our methodologies have been integrated into existing codebases. Finally, a mathematical analysis of our findings is provided in Appendix L.

2 Measuring diversity

In this section, we present tools for measuring the point configurations diversity in the LS. These tools are instrumental in the development of our sampling methods. Likewise, we use them as a proxy to measure image diversity.

2.1 Figures of merit for point configurations: artificial metrics in the LS

Let μ be the Lebesgue measure on $\mathbb{S}^d$, the unit sphere in $\mathbb{R}^d$. Given a set S of distinct points in $\mathbb{S}^d$, $S = \{s_1, \dots, s_n\}$, defines the empirical measure associated to $S \subset \mathbb{S}^d$ as $\hat{\mu}_S(C) = \frac{\#C \cap S}{\#S}$. The set of spherical caps for a threshold r is $C_r = \{\{x \in \mathbb{S}^d; x.u \geq r\}; u \in \mathbb{S}^d\}$. All the spherical caps for a same threshold r have the same measure. C_r is nontrivial if $r \in [-1, 1]$, and usual values lie in $r \in [0, 1)$. This provides **spherical cap diversity** measures for a point set S, namely mappings of the form $u \mapsto \{x \in \mathbb{S}^d; x.u \geq r\}$ which carries the Lebesgue measure to C_r. **Covering: average and worst case.** We use an average covering by default, i.e., $\mathit{covering}(S) = \mathbb{E}_{x \in \mathbb{S}^d} \inf_{s \in S} ||x - s||$. We also consider a worst case covering, namely $\sup_{x \in \mathbb{S}^d} \inf_{s \in S} ||x - s||$. **Packing/Dispersion.** The packing, $\mathit{packing}(S) = -\inf_{1 \leq i < j \leq \#S} ||s_i - s_j||$, is also based on a distance. The minus is used so that all our metrics are to be minimized. We also consider an average packing, namely $\mathit{avg\text{-}packing}(S) = \mathbb{E}_{x,y \in S, x \neq y} x.y$. **Riesz.** The Riesz potential is also based on a distance, with $\mathit{Riesz}(S) = -\sum_{1 \leq i < j \leq \#S} ||s_i - s_j||^{-s}$. We consider $s = \frac{1}{2}, s = 1, s = 2$. **Extending metrics using convolutions.** Typically, the LS is a space of tensors of a given shape. For example, some latent diffusion models use $64 \times 64 \times 4$, where 64×64 corresponds to spatial coordinates and 4 to the number of unordered channels. Different works [14, 10] have shown the importance of the spatial proximity of the coordinates. Likewise, the authors in [10] propose to group pixels in the LS by groups of 8×8. In this work, we propose to use convolutions. Thus, instead of measuring a distance between x and y, we consider the distance between $\mathit{conv}(x, k)$ and $\mathit{conv}(y, k)$, where k corresponds to a kernel for a Gaussian blurring of some radius (8 by default) on the spatial coordinates. In our setting, BigConv and MiniConv correspond to a radius of 24 and 2. **Combining metrics.** We also consider the average between all the above metrics without considering any weight. More metrics are presented in Appendix A.

2.2 Figures of merit for the diversity of images

We now consider diversity measures which are convenient for batches of images. Given a batch B of images and a set S of classes, the diversity $\mathit{Div}_{S,B}$ (to be maximized) typically comprises the number of elements in S which contain at least one element of B. This diversity strategy can be used for different criteria, discussed below. For a given generator of batches, B is a random variable, and by abuse of notation, $\mathit{Div}_{S,B}$ is used for the expectation $\mathbb{E}\mathit{Div}_{S,B}$. We can consider a binary criterion with value 1 if the batch reaches some predefined threshold and 0 otherwise. For example, we can consider the probability that we have at least 2 or 3 classes (up to all classes) present in the batch, i.e., $\mathit{Div}^c_{S,B} = P(\mathit{Div}_{S,B} > c)$. **Color diversity.** To assess the color diversity in an image batch B, we employ a method from [10] that involves extracting color information from each image in the batch using the RGB color model, which represents colors as a combination of red, green, and blue channels. We compute the mean value for each channel $(\mathrm{red}(I), \mathrm{blue}(I), \mathrm{green}(I))$ in a given image I, and identify whether one of these colors is predominantly present in the image: red

(resp. green, blue) is said to be *dominant for a parameter* K if $\text{red}(I) \geq K \cdot \text{green}(I)$ and $\text{red}(I) \geq K \cdot \text{blue}(I)$. $K > 1$ corresponds to rarer classes, which can be relevant. We can consider color diversity as follows:

- $Div_{\{C_r, C_g, C_b\}, B}$, where $C_r = \{I; \text{red}(I) > \text{green}(I) \wedge \text{red}(I) > \text{blue}(I)\}$, and C_g and C_b the same with *green* or *blue* in the place of *red*.
- The case above leads to $Div_{S,B} \leq 3$: we can have max 6 classes (instead of 3) by considering both which color is maximum and which color is minimum. In these examples, except for equality cases, a class represents a partition.
- We can increase further the number of classes by considering classes of the form $C_{red,green,K}$, denoting cases in which $\text{red}(I) \geq \max(\text{green}(I), \text{blue}(I))$ and $\text{red}(I) > K \times \min(\text{green}(I), \text{blue}(I))$, and $\text{green}(I) \leq \text{blue}(I)$: there are 18 such classes when considering all colors and $K \in \{1., 1.1, 1.2\}$. For short, we denote this by $Div_{18}(B)$.
- We also consider a discretization of the vectors $(\text{red}(I), \text{green}(I), \text{blue}(I))$: each discretized value leads to a class/partition. For example, we can consider the color diversity $Div_{depth=d}$ for various depth values d. Then, the classes are chosen as follows. The image is reduced to a single pixel, encoded in three channels with 2^d possible values. Thus, the maximum number of classes is $2^{3\times d}$. $Div_{depth=d}(B)$ is the number of classes observed at least once in B.

Ethnic diversity. In a real-world setting, we consider ethnicity for human faces, leading to $Div_{ethnicity,B}$ corresponding to 4 ethnicities (identified by DeepFace [15]). While the first metrics have the advantage of being easy and fast to compute, the last one has the advantage of being an example of a real-world metric. Many other diversity measures could be considered, which we leave as future work.

3 Generators of well distributed point configurations

The state-of-the-art has predominantly focused on datasets with low-dimensional characteristics, often yielding bounds that are non trivial only for large sample sizes. In the context of this paper, methodologies tools such as Halton [16] and Sobol [17] are not applicable due to the high-dimensional nature of the LS under study. We highlight four main methods in this paper: **covering**, **dispersion-with-big-conv**, **max** and **max-pooling**. Table 12 in Appendix I describes the other strategies.

- The **covering** generator maximizes the covering criterion defined above.
- The **dispersion-with-big-conv** method minimizes the worst case dispersion, modified by convolution with a kernel size (24, 24).
- In the **max** and **max-pooling method** [10], the **"max"** setting comprises a maximum number of iterations in randomly searching for a new vector that would have a maximal minimal distance to all the already selected vectors in the batch. The parameters of this method are the target size of the batch B and a maximum number of iterations $N_{\max}$. In the **"max-pooling"** version, the distance is computed after processing the vectors by the average pooling on blocks of size 8×8 on the spatial coordinates, which down-samples the vector size to $8 \times 8 \times 4$ in the case of a LS shape of $64 \times 64 \times 4$.

4 Experimental results

We emphasize in red our recommended methods: the **covering** method in the non-spatial case (i.e., only channels as non-trivial dimensions) and the **dispersion-with-big-conv** one in the spatial case, in green the **max-pooling** method, and in gray a **pure random strategy**. Appendix D presents additional results in terms of artificial metrics, as presented in Section 2.1. We focus here on diversity as measured on the images. In real-life, the batch size varies from a few units to thousands (when people have converged to a prompt they like, and create a big batch overnight). Consequently, we perform experiments with various batch sizes. We consider generative models (Table 1), which convert a latent tensor and possibly an input into an image: $image = model(input, latent)$. Given a point set and a classifier m with values in a finite set, we consider various diversity metrics $Div_{m,S}$ of the set I_S of images $I_S = \{model(input, latent); latent \in S\}$, with $S = \{s_1, \ldots, s_n\}$. $Div_{m,S}$ is

Table 1: We consider spherical domains in the LS.

Model name	Model type	Input type	Output type	Latent Space
SDXL-Turbo	text2image	Text (prompt)	Image	$512 \times 512 \times 4$
SD	text2image	Text (prompt)	Image	$64 \times 64 \times 4$
BigGan	class to image	None	Image	$1 \times 1 \times 128$

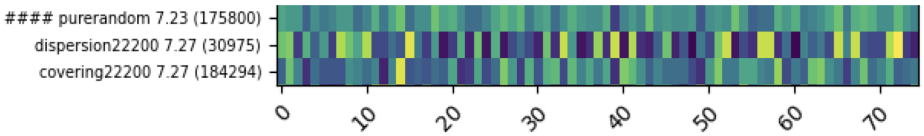


Figure 2: BigGan has 75 classes, studied separately in 75 columns. For each method (row) and each BigGan class (col), we consider the average color diversity Div_{18}. The score for that method (e.g. 7.23 for pure-random) is then the average Div_{18} over the 75 classes. Methods are ordered by average score (the greater, the better; best at the bottom). Batch size 48: additional results and details in Fig. 9 in the appendix. Reading guide: pure-random has 7.23 classes on average, evaluated on 175800 batches. Overall, as shown by the results in Appendix, our methods frequently outperform pure-random for BIGGAN but the gap with pure-random is much lower than for latent diffusion models.

typically equal to the cardinal of $\{m(i); i \in I_S\}$ (on average, for a stochastic S). We refer to Section 2.2 for all criteria.

We distinguish, in the following subsections two main categories of LSs: with and without spatial coordinates.

4.1 BigGan: no spatial coordinate, 128 channels

We consider a LS with 128 channels and no spatial coordinate, with BigGan [18]. We present the color diversity results in Fig. 2 (more results are illustrated in Fig. 9 in the appendix). Overall, the covering method appears to be a reasonable criterion for this context, but the success is moderate, and a greater gap is observed for latent diffusion models, as in the next section.

4.2 Stable Diffusion: spatial coordinates 64x64, and 4 channels

Stable Diffusion (SD) is a text-to-image generation model that begins with a noisy LS representation and progressively refines it into a coherent image. The LS is $64 \times 64 \times 4$ in dimensions, which is derived from compressing larger, high-resolution images (e.g., $512 \times 512 \times 3$). This model employs a reverse diffusion process. It starts with a compressed noisy latent representation and then, gradually denoises it through a series of steps to form a detailed image. If a batch of images needs to be generated, the same process applies to each of them. We call this generation process pure-random. **Color diversity.** We aim to determine the dominant colors in each image in each batch of generated images. We define a dominant color based on the coefficient K, and calculate the diversity metrics. The results for $K = 1$ (easier context, less differences between methods) are presented in Fig. 1. Additional results are presented in Table 18 and Fig. 10 in the appendix. The results for K=1.1 are presented in Table 2 and Fig. 3 (left), and in Fig. 11 in the appendix. Finally, the results for K=1.2 are presented in Table 3 and Fig. 3 (right), and Fig. 12 in the appendix. **Ethnicity classification for images portraying humans.** SD may lack diversity in ethnicity representation as highlighted in [19]. That is why, for the prompts that we use for the human face generation, we compare ethnic diversity when generating images through our methods and through the basic version of SD (a.k.a. pure-random strategy). We use DeepFace ethnicity recognition model [15] to identify the ethnicity of a person present in an image. In particular, we consider the following groups of ethnicities: (i) Black, (ii) Asian, (iii) Hispanic, (iv) White or Middle Eastern. We compute the percentage of batches in which all the different ethnicities are present, or when at least 3 out of 4 ethnicities,

Table 2: SD (LS 64x64). Comparison of different techniques for various prompts w.r.t the percentage of batches containing images with at least 2 different dominant colors for the following parameters: $K = 1.1$, batch size $= 50$; the methods are sorted by the average percentage over the different prompts. 3s computational cost. More methods are compared in Table 14 in the Appendix. Both max-pooling and dispersion-with-big-conv perform significantly better than pure-random.

Mode	bird	butterfly	cat	horse	rose
max-pooling	100.00	100.00	94.83±1.23	98.74±0.71	95.10±1.17
max	100.00	100.00	91.91±1.64	97.44±1.10	94.88±1.31
covering	99.26±0.33	100.00	77.26±1.20	96.06±0.70	85.73±1.09
dispersion-with-big-conv	100.00	100.00	93.02±1.37	96.65±1.06	87.09±1.68
dispersion-with_conv	100.00	100.00	91.09±1.49	97.18±0.90	84.52±1.74
dispersion-with_mini_conv	99.70±0.30	100.00	86.71±1.70	97.40±0.94	86.23±1.56
dispersion	99.49±0.30	100.00	77.83±1.28	92.83±0.94	83.56±1.21
pure-random	99.67±0.33	100.00	74.32±1.89	90.49±1.52	84.62±1.73
Riesz_	99.70±0.21	100.00	77.32±1.16	95.21±0.78	84.02±1.09

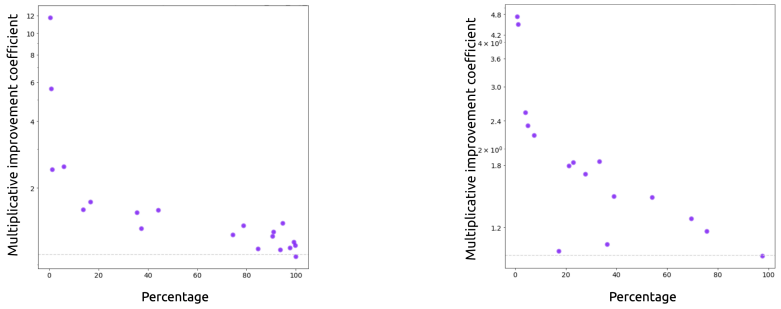


Figure 3: Multiplicative improvement of percentage of image batches of size 50 that contain images with dominance of different colors (at least 2 from 3): we compare dispersion-with-big-conv vs pure-random. X-axis = proportion of satisfactory batches (defined as: at least two of the three dominant colors for $K = 1$) in the pure random case. Y-axis: multiplication factor, i.e., proportion for dispersion-with-big-conv divided by the proportion for pure-random (i.e. > 1 means an improvement compared to pure-random). Left: $K = 1.1$. Right: $K = 1.2$. In general we get better improvement for least represented groups (low percentage, i.e., left of each subplot). Each dot represents a prompt from the list in Section N.

are present (similarly to colors). Tables 4 and 5 show that, in addition to our artificial criteria (above), our approach can successfully improve the diversity of generated images for real-world criteria.

Table 3: SD with LS 64x64. Comparison of different techniques for various prompts w.r.t the percentage of batches containing images with 2 different dominant colors for the following parameters: $K = 1.2$, batch size $= 50$; the methods are sorted by the maximum average percentage across the different prompts. 3s computational cost. Both recommended methods perform well for this 64x64 LS (512x512 images), with better results for max-pooling. More methods are compared in Table 15 in the Appendix.

Mode	bird	butterfly	cat	horse	rose
max-pooling	95.08±1.27	99.67±0.33	45.86±1.59	73.11±2.10	53.59±1.53
max	88.89±1.90	99.19±0.57	37.87±1.94	63.08±2.18	41.73±1.80
covering	76.93±1.25	76.30±1.22	9.86±0.97	21.66±1.21	36.80±1.11
dispersion-with_conv	77.53±1.82	75.43±1.91	15.51±1.74	31.03±1.79	40.65±1.66
dispersion-with-big-conv	77.34±1.94	77.41±1.87	16.61±1.79	22.30±1.97	41.72±1.66
dispersion-with_mini_conv	71.30±1.77	73.75±1.87	10.30±1.57	30.48±1.95	39.94±1.54
Riesz_	78.36±1.25	75.48±1.21	9.04±0.93	25.40±1.24	36.08±1.09
pure-random	71.80±1.85	75.25±1.89	6.42±1.33	24.59±1.86	37.82±1.71
dispersion	75.81±1.35	72.53±1.23	9.12±0.96	23.99±1.28	34.56±1.22

Table 4: SD with LS 64x64. Comparison of different modes for various prompts w.r.t. the percentage of batches featuring people of all 4 ethnicity groups, batch size = 50, the methods are sorted by the average percentage across the different prompts (the greater the better). More methods in Table 16 (Appendix).

Mode	A close-up photograph of an elderly person's face	A passport-style photograph of a person's face	A professional photograph of an adult person face	face
dispersion_with_big_conv	59.09±6.19	52.63±5.33	23.44±2.24	49.54±1.72
dispersion_with_mini_conv	63.64±6.50	40.00±5.87	22.69±2.20	46.81±1.94
dispersion_with_conv	50.00±4.73	46.15±5.26	28.02±2.13	39.57±1.93
pure_random	54.84±4.90	36.00±6.14	21.40±2.13	39.64±1.98
covering_	42.47±3.33	33.68±3.22	20.10±1.32	45.50±1.06
dispersion	33.82±3.80	25.00±4.69	19.26±2.74	43.52±1.31

Table 5: SD (LS 64x64, i.e., images 512x512). Comparison of different modes for various prompts w.r.t the percentage of batches featuring people of at least 3 out of 4 ethnicity groups, batch size = 50. The methods are sorted by average percentage over the different prompts. More methods are compared in Table 17 in the Appendix.

Mode	A close-up photograph of an elderly person's face	A passport-style photograph of a person's face	A professional photograph of an adult person face	face
dispersion_with_big_conv	90.91±5.58	94.74±4.86	75.12±2.25	89.81±1.85
dispersion_with_mini_conv	90.91±5.58	92.00±5.00	76.39±2.22	86.70±2.15
covering_	93.15±2.76	88.42±2.91	71.21±1.33	89.62±1.07
dispersion_with_conv	82.14±5.95	96.15±3.63	75.86±2.14	86.38±1.94
pure_random	93.55±4.13	88.00±5.73	69.43±2.12	87.84±1.93
dispersion	82.35±3.81	93.75±3.27	69.63±2.76	89.45±1.29

4.3 Statistics in Automatic1111/ SDXL-Turbo: LS 512x512x4, 3 seconds

Before integrating our work in main public codebases, we check that it also performs well in a different contexts, including low batch size (8) and a different image generator (SDXL Turbo). We limit the time to three seconds, which is negligible compared to the image generation process in a MacBook Pro M1. We use the color diversity measure $Div_{depth=d}$ for various depth values d and compare in Table 6 (dispersion-with-big-conv) and Table 7 (max-pooling) the diversity of batches created by well distributed point configurations in the case of this big latent spaces. As a baseline, we also use the same images randomly grouped in batches of the same size. As demonstrated by the results, dispersion-with-big-conv is always beneficial. Sometimes, in Section 4.2 with smaller latent spaces, max-pooling was better: however, Table 7 (also Table 11 in the appendix) shows that for the same parametrization (i.e. same pooling size 8×8) as in [10], or for a parametrization scaling linearly with the spatial coordinates (i.e. 64×64-pooling as we switch from a 64×64 LS to a 512×512 LS), max-pooling fails in this high-dimensional context. We therefore recommend dispersion-with-big-conv, which works over all tested settings, including cases with no spatial coordinates (for which the convolution does not do anything); but then covering is usually better (Section 4.1).

4.4 Image quality

We start the discussion with gradient-based methods (described in Appendix I). We observe a big quality loss for images generated from latent variables created by those approaches. This can be explained as follows: As noted in [10], points in the LS with large norm, though they are theoretically possible in the Gaussian generator of latent variables, are rare, hence the diffusion does not work when the norm is large. We need points which have (for example) norm roughly $\sqrt{d}$ in dimension d. This is aligned with the results observed in [10], as we operate within a normalized space. However, a second property of almost all generated points is that local averages (in the LS with spatial coordinates) are always close to 0. We should not have more local uniformity than the traditional random points. We observe that the gradient naturally "pushes" neighboring points in the same direction, leading to

Table 6: Experiments on SDXL Turbo (LS 512x512). Diversity Div_{depth} for various depth levels, for dispersion-with-big-conv and pure-random. All experiments with 800 images. Even on this completely different setting, results are positive in all cases. We removed only the cases in which all images were in the same class. Our approach was never detrimental.

Prompt	Depth	dispersion-with-big-conv	Baseline
a beautiful woman	1	**2.16 (±0.05)**	2.08 (±0.02)
	2	**2.16 (±0.05)**	2.10 (±0.02)
	3	**2.75 (±0.08)**	2.63 (±0.02)
	4	**4.9 (±0.1)**	4.74 (±0.03)
a handsome man	1	**3.23 (±0.06)**	3.09 (0.02)
	2	**3.23 (±0.06)**	3.08 (±0.02)
	3	**3.83 (±0.08)**	3.61 (±0.03)
	4	**5.42 (±0.1)**	5.22 (±0.03)
a man and a woman dancing together	1	**3.31 (±0.07)**	3.22 (±0.02)
	2	**3.31 (±0.07)**	3.22 (±0.02)
	3	**4.95 (±0.09)**	4.93 (±0.03)
	4	**6.2 (±0.1)**	6.17 (±0.03)
a mma fight between two pharaohs	1	**2.24 (±0.04)**	2.20 (±0.01)
	2	**2.24 (±0.04)**	2.22 (±0.01)
	3	**3.55 (±0.07)**	3.45 (±0.02)
	4	**5.46 (±0.1)**	5.33 (±0.03)
a unicolor image	1	**2.80 (±0.07)**	2.70 (0.02)
	2	**4.79 (±0.09)**	4.46 (±0.03)
	3	**7.05 (±0.09)**	6.57 (±0.03)
	4	**7.84 (±0.04)**	7.71 (±0.01)
unicorn + dragon dancing in a magical garden under a rainbow	1	**3.35 (±0.08)**	3.23 (±0.02)
	2	**3.35 (±0.08)**	3.22 (±0.02)
	3	**3.35 (±0.08)**	3.23 (±0.02)
	4	**3.8 (±0.1)**	3.64 (±0.03)

uniform local areas that diffusion approaches cannot handle. This leads to weird images, as illustrated in Fig. 4. Figure 5 (appendix) shows that one cannot see any difference when using gradient-free methods. Additionally, human-raters statistics, depicted in Table 8, pinpoint that our method does not deteriorate the quality of the images.

4.5 Discussion

The case without spatial coordinates is dominated by covering, which is intuitively simple and satisfactory. With spatial coordinates (as in most latent diffusion models), overall, max-pooling from [10] performs very well in the setting of [10], i.e., with a relatively small LS 64x64x4. In the case of 512x512x4, it becomes comparable, or even weaker, than pure-random (Table 7 for the default parametrization; another scaling of parametrization is presented in Table 11 in the appendix). Cap_pooling (proposed in [10]) sometimes has a huge computational time, making it irrelevant in our high-dimensional context.

Table 7: Max-Pooling with 8x8 blocks vs random for 512x512 LS. We present the diversity for a discretization of image colors as detailed in Section 4.3. 400 images per prompt. The success of max-pooling (compared to pure-random) is questionable in this high-dimensional LS context. Hence, our preference for the more robust dispersion-with-big-conv (see Table 6).

Prompt	Depth	Max-Pooling	Baseline
a beautiful manga character killing animals	1	**2.46** (±0.08)	2.45 (±0.02)
	2	2.46 (±0.08)	**2.47** (±0.03)
	3	2.46 (±0.08)	**2.46** (±0.02)
	4	**3.7** (±0.1)	3.65 (±0.03)
a gothic witch laughing and playing with a bazooka in the middle of hell	2	**1.08** (±0.04)	1.08 (±0.01)
	3	3.44 (± 0.09)	**3.46** (±0.02)
	4	3.1 (± 0.1)	**3.09** (± 0.04)
a man and a woman dancing together	1	2.86 (± 0.09)	**2.96** (± 0.03)
	2	2.86 (± 0.09)	**2.97** (± 0.03)
	3	4.3 (±0.2)	**4.36** (± 0.04)
	4	5.3 (± 0.2)	**5.47** (± 0.05)
an incredible image	1	1.48 (± 0.07)	**1.48** (± 0.02)
	2	**1.48** (± 0.07)	1.46 (± 0.02)
	3	**2.9** (± 0.1)	2.88 (± 0.03)
	4	4.0 (± 0.2)	**4.13** (±0.05)

Figure 4: Two first images: Examples of images obtained by SD on points obtained by gradient-based methods, even after standardizing the norm: images do not make any sense. Other images: examples of images generated by dispersion-with-big-conv with batch size 50 for prompts in Appendix N.2.

Table 8: Human ratings: statistics of preferences between SDXL Turbo, SDXL Turbo with dispersion-with-big-conv. Counting no preference as 50%, this is $53.3 \pm 2.1\%$ preference for the quality of our images. While this is not a statistically significant improvement, the quality is at least preserved; furthermore, our other results show a significant improvement in diversity, in particular in difficult cases. The exact question is "Do you prefer the image on the right or on the left for prompt XXX ? (Left, Right, No opinion) "

SDXL Turbo	SDXL Turbo + dispersion-with-big-conv	No preference
27.2% %	33.8%	39.0%

5 Conclusions

We applied well distributed point configurations for sampling LSs, for enhancing the diversity of generated images. Compared to [10], we consider a broader range of cases (cases without spatial coordinates and bigger LSs), and propose new methods. For the applications without spatial coordinates, the covering method performs best overall; and, in contexts with spatial coordinates, two strong methods are max-pooling and dispersion-with-big-conv. We note that taking into account the topology of variables (by convolution or pooling) is essential when working on LSs with spatial coordinates, and carefully choosing the size of convolution matters. max-pooling is doing something similar to convolution, though by blocks, and performs best for small LSs with spatial coordinates. For bigger LSs, dispersion-with-big-conv performs best; and we did not find a case in which dispersion-with-big-conv is detrimental, so we recommend it until better metrics are found. The case of gradient-based methods shows that the selected metrics are not perfect. We need the LS to be roughly sampled as in the original methods, and black-box optimization methods do that, whereas gradient-based methods do not. A nice achievement would be to design a metric that would simultaneously ensure diversity and quality: for the moment, only black-box optimization of our metrics was good at preserving quality (as shown by the human ratings) while improving diversity. We note that strong computational budgets do not have a clear positive impact, and results with gradient-based methods (numerically excellent but leading to poor quality) even suggest that over-optimizing can be detrimental: we recommend a time budget always tiny compared to the image generation part, and even when using large batch sizes (> 500) never more than a few seconds. A classical method for increasing the diversity is to add text suffixes: e.g., different skin colors or genders. However, as shown by many recent counter-examples recently [20], this has several drawbacks, such as unrealistic outputs. **Future work.** A recent paper [14] proposed to use user preferences for guiding local perturbations in the LS: combining such approaches with our tools are a natural further work. The present work

is a step in very high dimensional well distributed point configurations: maybe the same methods could be applied to other unexplored areas of applied mathematics. **Limitations.** Our work can not create images that can not be created by the original model, or concepts unavailable in the original training data. Our method has a big impact for rare classes in latent diffusion models, but a moderate (though beneficial) impact on GANs. For small latent spaces, Max-Pooling is frequently better than our proposal.

References

[1] D. P. Kingma, M. Welling *et al.*, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, pp. 307–392, 2019.

[2] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath, "Generative adversarial networks: An overview," *IEEE signal processing magazine*, vol. 35, no. 1, pp. 53–65, 2018.

[3] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 10 684–10 695.

[4] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *International conference on machine learning*. Pmlr, 2021, pp. 8821–8831.

[5] D. Marwood, S. Baluja, and Y. Alon, "Diversity and diffusion: Observations on synthetic image distributions with stable diffusion," *arXiv:2311.00056*, 2023.

[6] C. T. Ho, "Stable diffusion: Why are diverse results so hard to come by?" in *anaconda.com*, 2023.

[7] F. Bianchi, P. Kalluri, E. Durmus, F. Ladhak, M. Cheng, D. Nozza, T. Hashimoto, D. Jurafsky, J. Zou, and A. Caliskan, "Easily accessible text-to-image generation amplifies demographic stereotypes at large scale," in *ACM Conference on Fairness, Accountability, and Transparency*, 2023, pp. 1493–1504.

[8] K. C. Fraser, S. Kiritchenko, and I. Nejadgholi, "Diversity is not a one-way street: Pilot study on ethical interventions for racial bias in text-to-image systems," *ICCV*, 2023.

[9] H. Niederreiter, *Random Number Generation and quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, 1992.

[10] M. Zameshina, O. Teytaud, and L. Najman, "Diverse diffusion: Enhancing image diversity in text-to-image generation," *arXiv:2310.12583*, 2023.

[11] G. Corso, Y. Xu, V. D. Bortoli, R. Barzilay, and T. S. Jaakkola, "Particle guidance: non-I.I.D. diverse sampling with diffusion models," in *The Twelfth International Conference on Learning Representations*, 2024.

[12] L. Yang, Z. Zhang, Y. Song, S. Hong, R. Xu, Y. Zhao, W. Zhang, B. Cui, and M.-H. Yang, "Diffusion models: A comprehensive survey of methods and applications," *ACM Computing Survey*, vol. 56, no. 4, 2023.

[13] D. Podell, Z. English, K. Lacey, A. Blattmann, T. Dockhorn, J. Müller, J. Penna, and R. Rombach, "SDXL: Improving latent diffusion models for high-resolution image synthesis," 2023.

[14] M. Videau, N. Knizev, A. Leite, M. Schoenauer, and O. Teytaud, "Interactive latent diffusion model," in *Genetic and Evolutionary Computation Conference*, 2023, pp. 586–596.

[15] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 1701–1708.

[16] J. Halton, "On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals." *Numerische Mathematik*, vol. 2, pp. 84–90, 1960.

[17] I. M. Sobol, "On the systematic search in a hypercube," *SIAM Journal on Numerical Analysis*, vol. 16, no. 5, pp. 790–793, 1979.

[18] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," in *International Conference on Learning Representations*, 2019.

[19] D. Theron, "Evidence of unfair bias across gender, skin tones & intersectional groups in generated images from stable diffusion," in *towardsdatascience.com*, 2023.

[20] N. Bonyhady, "Black George Washington," 2024, accessed on March 1st, 2024. [Online]. Available: tinyurl.com/4km6yfpu

[21] J. Rapin and O. Teytaud, "Nevergrad - A gradient-free optimization platform," github.com/facebookresearch/nevergrad, 2018.

A Figures of merit for point configurations: additional artificial metrics

Cap discrepancy. The spherical cap discrepancy or cap discrepancy, for a threshold r, is defined in the present paper as the variance $Disc_r = Var_{c \in C_r} \hat{\mu}_S(c)$. By default, we use $r = r_d = \frac{1}{\sqrt{d}}$, so that the cap discrepancy is for us $Disc_{r_d}$.

This cap discrepancy has the advantage that it is easy to compute. Other cap discrepancies might consider $E(\hat{\mu}_S(C) - \mu(C))^2$, which is equivalent.

Half-sphere discrepancy. The half sphere discrepancy is the special case $Disc_0$ of cap discrepancy.

B Examples of generated images

In Figure 5 we provide some examples of images generated by SDXL Turbo.

Figure 5: Images generated by SDXL Turbo (left of each pair) and SDXL-Turbo with dispersion-with-big-conv (right of each pair). Prompts: “a landscape from a planet in outer space”, “a unicorn and a dragon dancing in a magical garden under a rainbow”. No clear difference spotted. Human rates (Table 8) do not show a clear preference at the level of individual images either.

C Integration in public codes and reproducibility

We provide our tools as a module in Nevergrad (starting from version 0.15.0). Given a batch of latent variables, with a shape $(n, s_1, s_2, \ldots, s_k, c)$ the code will assume that we have n latent variables with shape $(s_1, s_2, \ldots, s_k, c)$ with s_i spatial coordinates and c the number of channels. A convolution will be applied on $s_1, \ldots, s_k$ (at least if $k \geq 1$) where the last coordinate will be considered as channels. The code is as follows:

```
import nevergrad as ng
my_latent = ng.common.quasi_randomize(my_latent)
```

The code is publicly available in [21] and as plugins for well known codes such as Automatic1111 , link `https://github.com/mathuvu/sd-webui-diversity`.

D Additional experimental results: artificial metrics in the LS

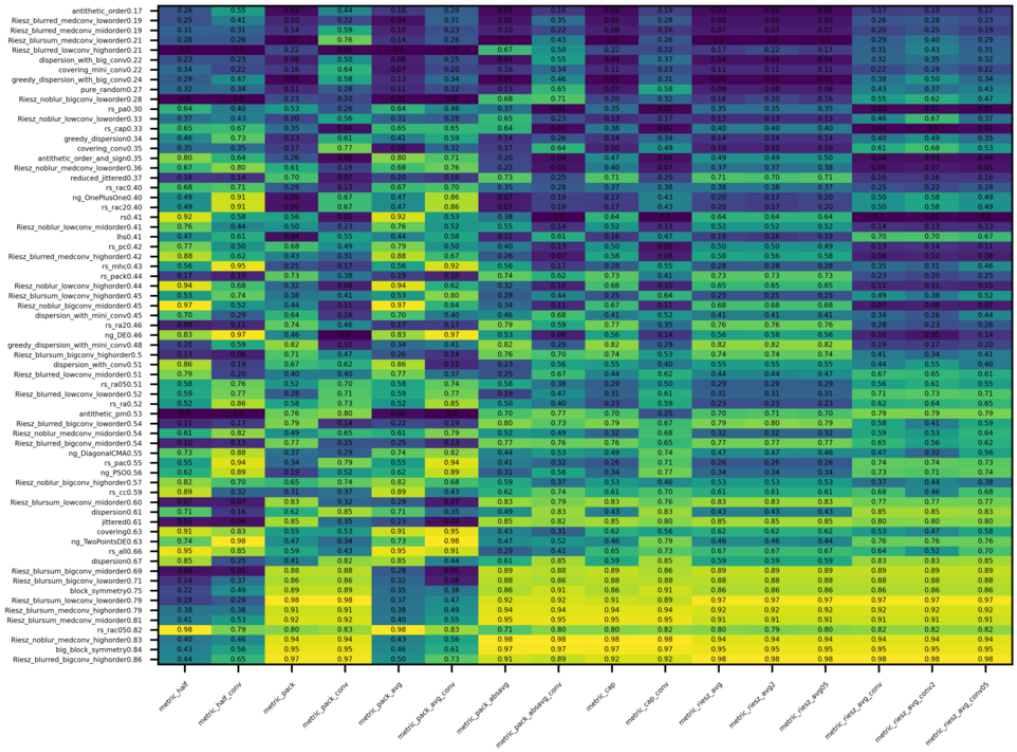
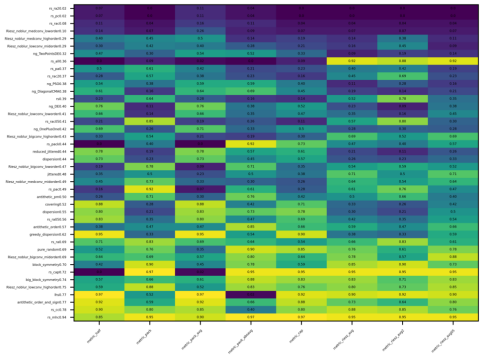


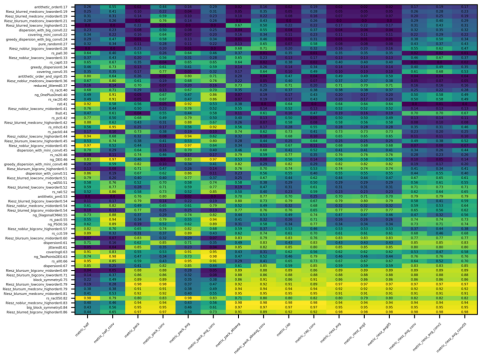
Figure 6: Comparison between generators of well distributed point configurations (rows) for artificial metrics (columns), in the case 64x64x4 corresponding to StableDiffusion. Numbers = proportion of methods which did better than this method for this criterion, i.e. 50% of methods did better than dispersion-with-big-conv for the metric packing-with-conv. Batch size 48. Scores = average per row, used for ranking, so that low values (at the top) are better.

Figures 6 to 8 present artificial metrics. Each column corresponds to a different figure of merit for well distributed point sets. Each row corresponds to a method. Each number $q_{m,c}$ in the heatmap corresponds to the proportion of methods doing better than method m for that metric c: the lower the better. Methods (rows) are ranked by averages of $q_{m,c}$ and this average is mentioned next to their name. Figure 6 presents results for an image LS corresponding to StableDiffusion, namely 64x64x4. Our observations are straightforward. Good designs in terms of spherical cap discrepancy with convolution, or worst-case packing with convolution, are difficult to obtain without optimizing that specific criterion. Spherical cap discrepancy is very low (i.e. good) when we optimize the covering, and more unexpectedly, designs created by LHS also perform well for this criterion. 2-antithetic (simple centered symmetry) is excellent for half-sphere discrepancy (including with convolution), average packing (also including with convolution). Covering or dispersion are good for worst case packing, and more unexpectedly latin hypercube sampling is also good for this criterion. Fig. 7 and 8 present results for channels only, i.e., shapes of the form $1 \times 1 \times c$. Also, we note that no artificial metric is really good at predicting if a point configuration is good for the image generation that follows. Packing with average looks not too bad, but gradient method are a counterexample: it turns out that the way we optimize a criterion has an impact on the usability of generated points by latent diffusion models. This shows that criteria are not perfect (see Section 4.4).

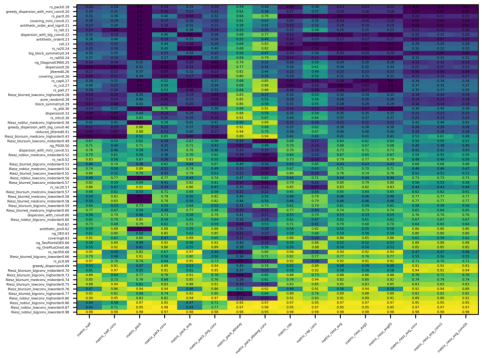
As expected, antithetic sampling performs well for half-spheres discrepancy and for average packing, and optimizing the sum of all metrics is effective for being reasonably good on all metrics. Most methods are worse than random for at least one metric: this shows the complexity of high-dimensional point configurations.



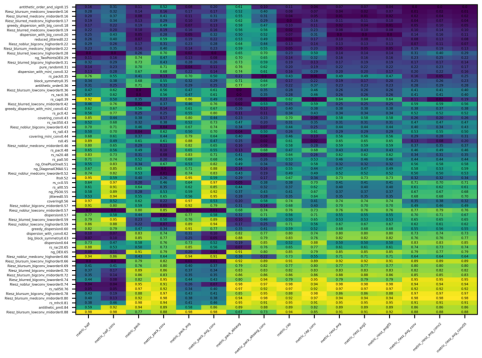
384 points, shape (1,1,3)



48 points, shape (16,16,2)

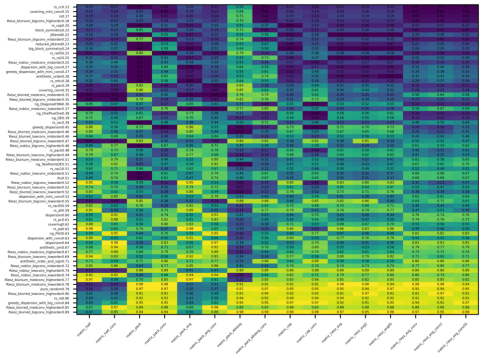


6 points, shape (32, 32, 4)

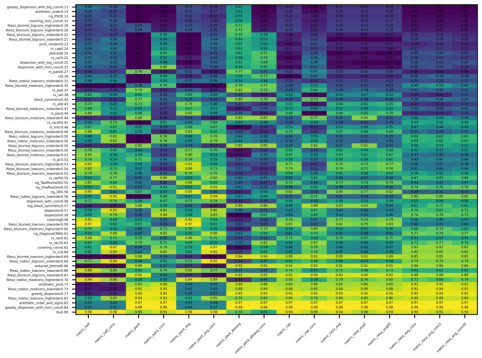


96 points, shape (32,32,4)

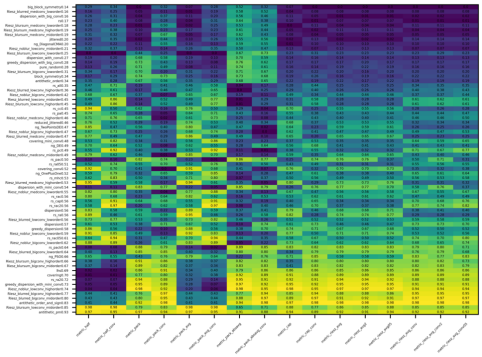
Figure 7: Additional comparisons between generators of well distributed point configurations (rows) for artificial metrics (columns): similar to Fig. 6 but with different batch sizes and shapes. Low values (at the top) are better.



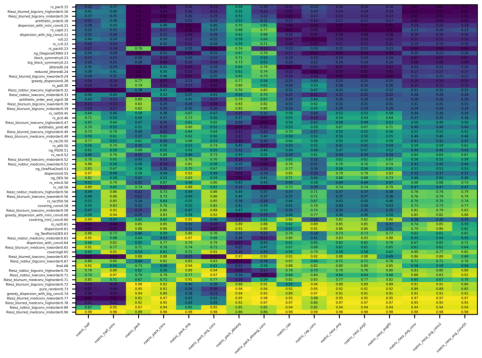
12 points, shape (32,32,2)



12 points, shape (32,32,3)



50 points, shape (64,64,3)



12 points, shape (64,64,4)

Figure 8: Additional comparison between generators of well distributed point configurations (rows) for artificial metrics (columns). Similar to Fig. 6 and Fig. 7 but with different batch sizes and shapes. Low values (at the top) are better.

E Additional results on BigGan

Figure 9 extends Fig. 2 (diversity Div_{18} for BigGan) by considering several batch sizes.

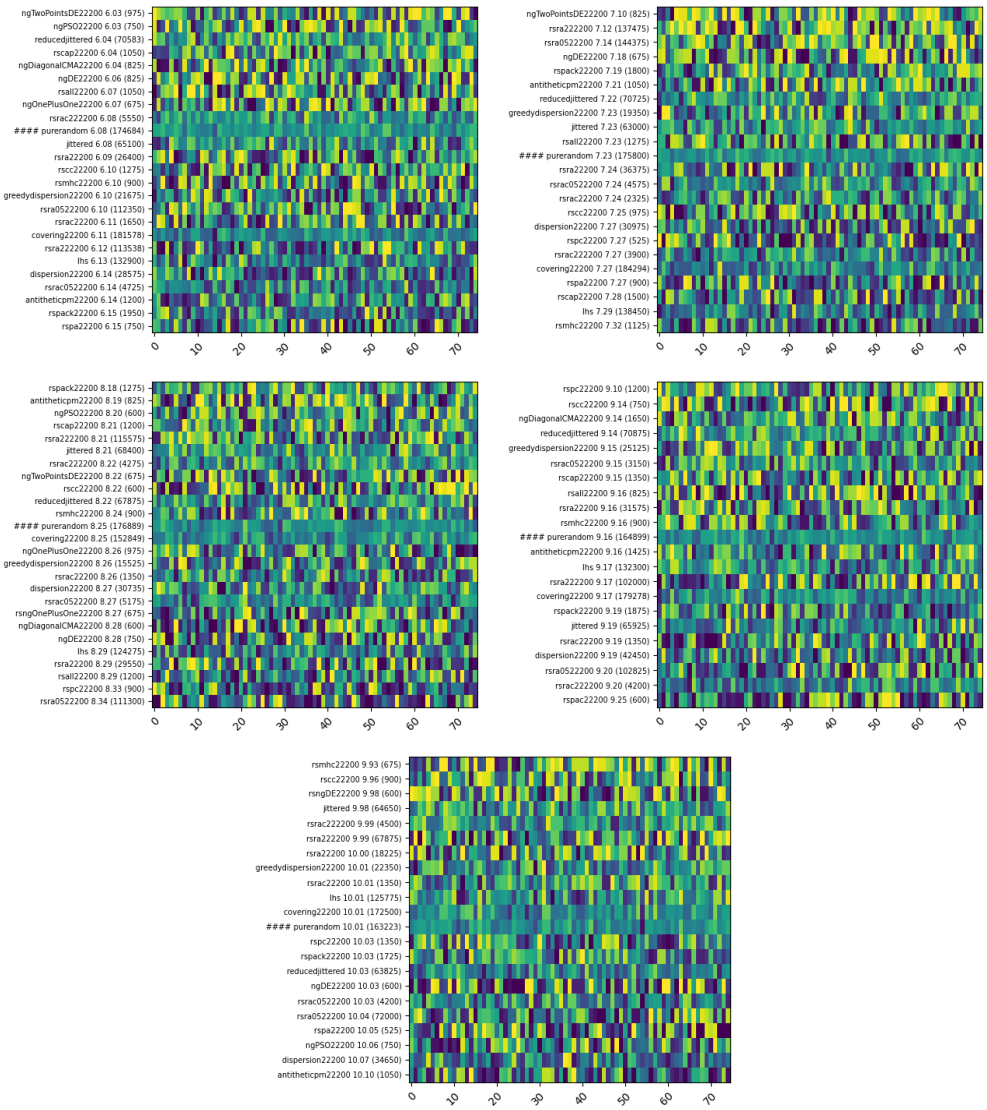


Figure 9: Extension of results with BigGan in Fig. 2, for batch size 24, 48, 96, 192, and 384. The 75 BigGan classes are studied separately in 75 columns. For each method (row) and each BigGan class (col), we consider the average color diversity Div_{18}. The score for that method is then the average Div_{18} over the 75 classes. Methods are ordered by average score (the greater, the better; best at the bottom). Results are good for the covering method on average (or almost equal performance), but the difference is small, compared to cases with spatial latent classes for which there is a big difference. 22200 refers to the time budget in centiseconds.

F Results with 3s, 6 cores, Stable Diffusion with LS 64x64x4

Results are also positive with a limited computational power (Tables 9 and 10).

Table 9: SD with LS 64x64x4. Percentage of image batches of size 50 that contain at least one image with dominance of each color (Red, Green, Blue), k = 1.1. Number of cores: 6. Average latent vector generation time for different methods: 0s for pure-random, 2s for dispersion-with-big-conv, 3s for max-pooling, 3s for max. Even with very limited computational power, dispersion-with-big-conv improves the color diversity w.r.t pure-random.

Mode	butterfly	cat
ng_	75.69±2.41	12.70±2.11
dispersion-with-big-conv	80.28±3.79	8.05±2.68
dispersion	77.56±2.26	10.43±1.89
covering	73.45±1.96	12.50±1.73
rs_	75.34±0.92	9.36±0.76
greedy_dispersion-with-big-conv	67.37±3.24	16.98±3.03
dispersion-with_conv	70.75±3.13	13.19±3.08
greedy_dispersion-with_mini_conv	73.00±3.24	10.23±2.91
antithetic_pm	74.19±3.37	6.38±2.36
greedy_dispersion	73.96±3.31	6.25±2.54
dispersion-with_mini_conv	69.15±3.30	11.00±2.78
pure-random	74.12±3.53	6.00±2.23
Riesz_	6.60±0.42	2.19±0.25

Table 10: SD with LS 64x64x4. Percentage of image batches of size 50 that contain at least one image with dominance of each color (Red, Green, Blue), k = 1.2. Number of cores: 6. Average latent vector generation time for different methods: 0s for pure-random, 2s for dispersion-with-big-conv, 3s for max-pooling, 3s for max.Dispersion-with-big-conv outperforms pure-_random for the color diversity.

Mode	butterfly	cat
dispersion-with-big-conv	44.78±11.47	0.00±1.00
ng_	39.18±7.76	1.59±3.06
antithetic_pm	36.96±10.85	1.11±3.74
greedy_dispersion-with-big-conv	37.89±10.57	0.00±1.00
rs_	36.99±2.99	0.84±0.03
covering	35.63±6.53	1.43±2.39
greedy_dispersion-with_mini_conv	36.84±10.69	0.00±1.00
dispersion-with_mini_conv	32.58±11.45	1.00±1.00
dispersion	32.83±7.67	0.49±0.06
greedy_dispersion	31.18±11.31	1.25±4.20
pure-random	29.63±12.21	0.00±1.00
dispersion-with_conv	27.45±10.96	0.00±1.00
Riesz_	3.04±1.04	0.03±0.01

G Additional experiments with dispersion-with_conv

Our main focus is on the more successful dispersion that one obtain with Big Conv. Thus, we check if variants (such as dispersion-with_conv) are also robust enough for being beneficial or at least non-detrimental. We observe in Figs. 10 to 12 that dispersion-with_conv becomes better and better for more difficult cases, compared to pure random. Additionally, when the classes are rare, the frequency has a multiplicative effect.

Diversity improvement for different time settings. Here we detail an experiment that aims at exploring the impact of time on color diversity in the generated images. The results are presented in Figs. 13 and 15 for the "dispersion-with_conv" and "dispersion-with-big-conv" methods.

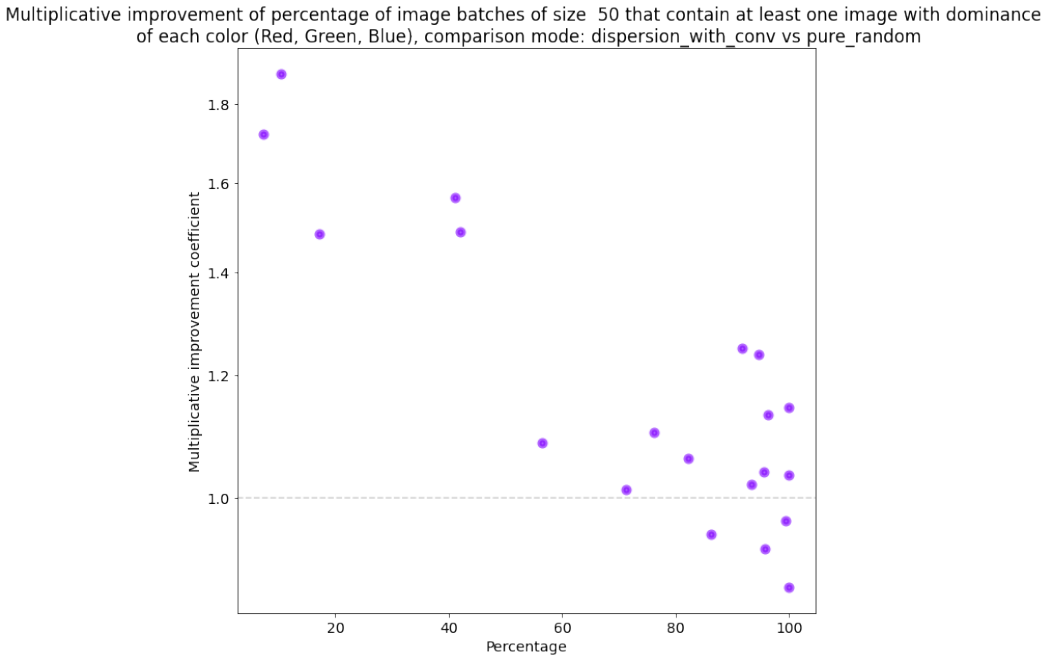


Figure 10: SD with LS 64x64x4. Comparison of dispersion-with-conv and standard SD (pure-random) for various prompts w.r.t. the multiplicative percentage of batches containing images with all 3 dominant colors for the following parameters: $K = 1$, batch size $= 50$. In general we get better improvements (greater y-axis) for the most difficult cases (low x-axis value).

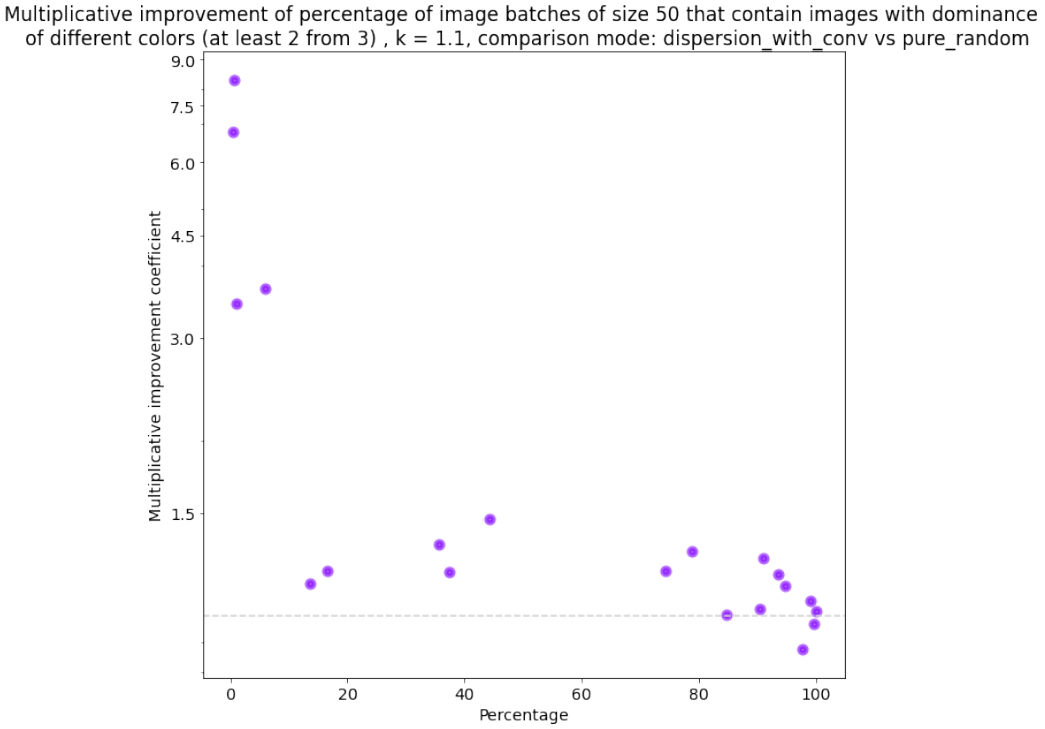


Figure 11: SD with LS 64x64x4. Comparison of dispersion-with_conv and standard SD (pure-random) for various prompts w.r.t the multiplicative percentage of batches containing images with at least 2 of the 3 different dominant colors for the following parameters: $K = 1.1$, batch size = 50. In general we get better improvements (greater y-axis values) for least represented groups (lower x-axis values).

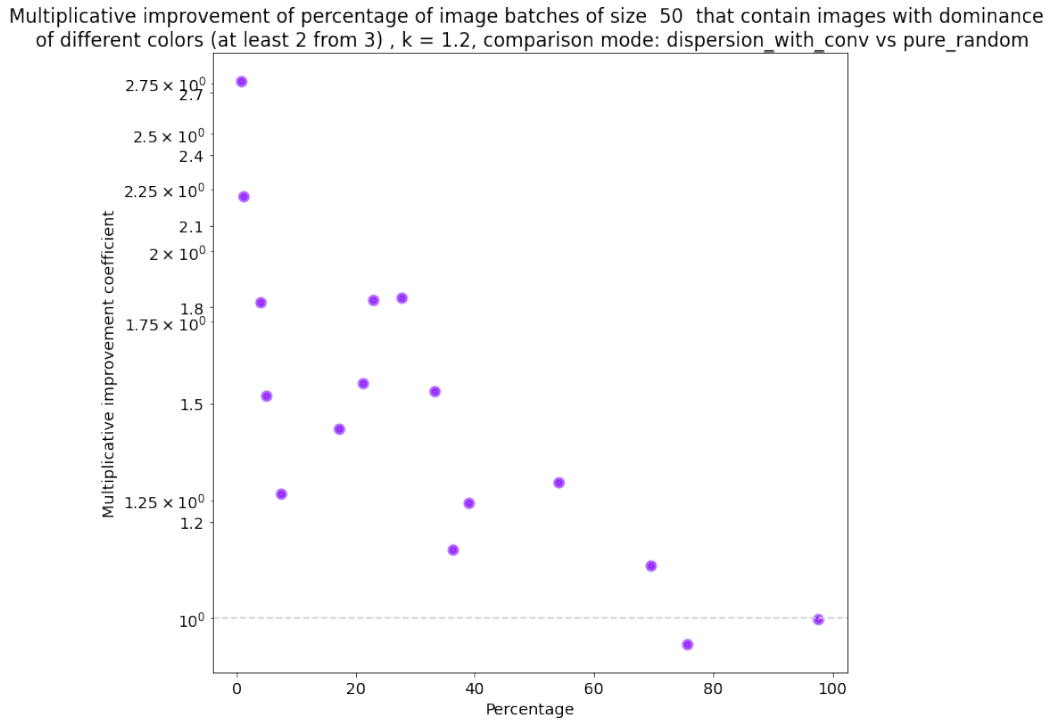


Figure 12: SD with LS 64x64x4. Comparison of dispersion-with_conv and standard SD (pure-random) for various prompts w.r.t regard to the multiplicative percentage of batches containing images with at least 2 different dominant colors for the following parameters: $K = 1.2$, batch size = 50. In general we get better improvements (greater y-axis values) for least represented groups (lower x-axis values).

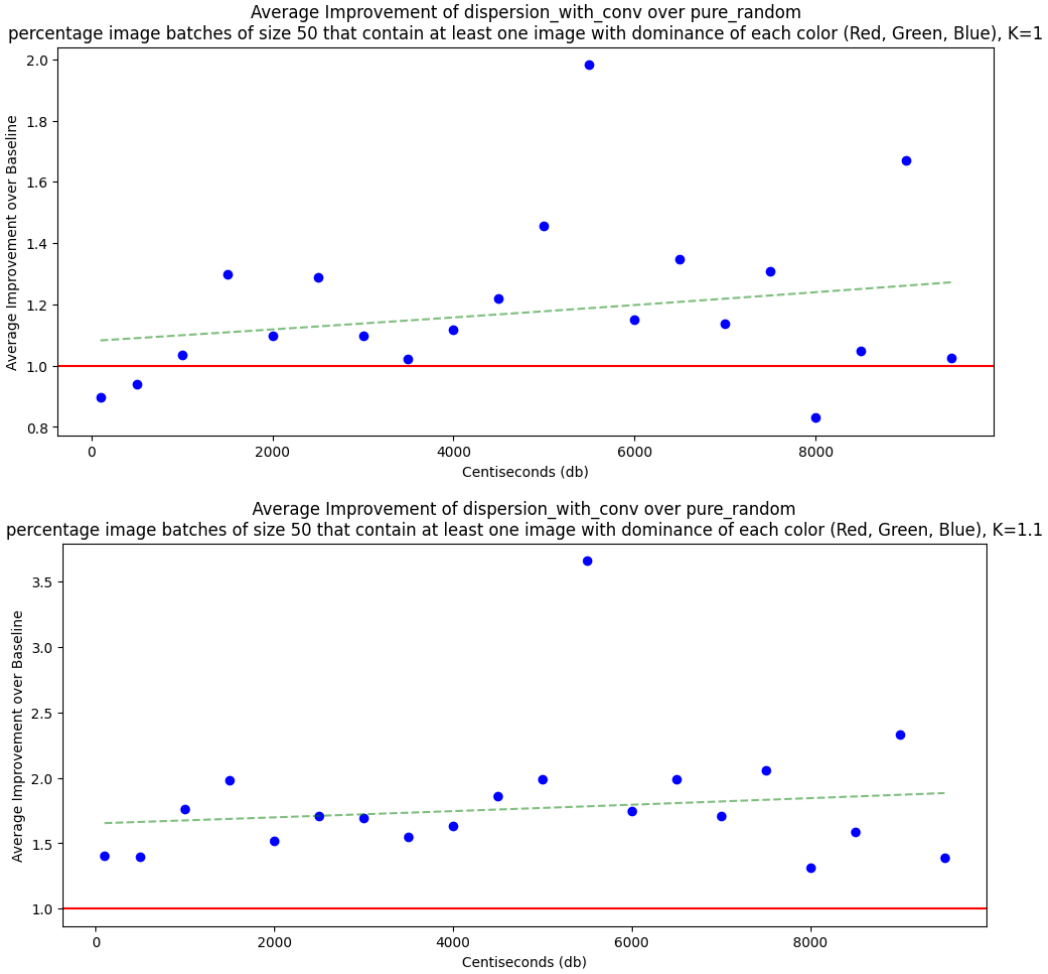


Figure 13: SD with LS 64x64x4. This figure shows the variation in color diversity depending on the computational cost for the 'dispersion-with_conv' method. It compares the improvement in color diversity against the 'pure-random' baseline, demonstrating a slight increase in the percentage of image batches containing all dominant colors as the computational budgets progresses: we do not need a big computational power. In Section 4.3, we also show that 3s is enough for an impact on a larger LS 512x512x4 as the one of SDXL Turbo.

H Additional experiments with max-pooling

The max-pooling method, successful in the 64x64x4 LS, is tested in the present 512x512x4 case. We test both 8x8 pooling (as in the 64x64 LS), and 64x64 pooling (i.e., scaling the size of blocks proportionally to the LS size).

We observe in Table 7 (with 8x8 blocks, so without scaling the block size when moving from 64x64 to 512x512) that results are not good for max-pooling when the LS is large.

We can also note in Table 11 that scaling the kernel size from 8x8 to 64x64 (when switching from a LS with spatial coordinates 64x64 to a LS with spatial coordinates 512x512) does not make max-pooling much better than random either (we tested a 8x8 kernel, i.e., no scaling proportionally to the LS spatial coordinates.)

Table 11: max-pooling with 64x64 blocks vs random for 512x512 LS. We present the diversity for a discretization of image colors by depth as detailed in Section 2.2 and num classes refers to the number of classes that are represented at least once in our tests (num classes at most $2^{3\times d}$ for a depth d, by definition of discretization at d bits with 3 colors).

Depth	max-pooling	pure-random
Prompt = a gothic witch laughing and playing with a bazooka in the middle of hell		
(num classes: 5, depth 3)		
3	3.300 (0.086897)	**3.311667** (0.026412)
(num classes: 9, depth 4)		
4	**3.020** (0.122857)	3.001667 (0.033236)
a unicolor image		
(num classes: 5, depth 1)		
1	**2.700** (0.095831)	2.690 (0.029855)
(num classes: 11, depth 2)		
2	**3.960** (0.133890)	3.933333 (0.038159)
(num classes: 26, depth 3)		
3	5.600 (0.145686)	**5.648333** (0.041561)
(num classes: 83, depth 4)		
4	**7.540** (0.081866)	7.311667 (0.030518)
Prompt=an incredible image		
(num classes: 3, depth 1)		
1	1.420 (0.070508)	**1.480** (0.021215)
(num classes: 3, depth 2)		
2	1.420 (0.070508)	**1.476667** (0.021470)
(num classes: 6, depth 3)		
3	2.720 (0.127903)	**2.846667** (0.034424)
(num classes: 15, depth 4)		
4	**4.120** (0.172946)	4.108333 (0.045497)

I Additional information on methods

Table 12 presents the methods used in the present paper, in particular those not developed in the main text.

Most of our results are based on ad hoc methods for specific criteria or black-box optimization based on [21]. However, we also include methods using gradient-based optimization, for the Riesz potential methods. They have "Riesz_" as a prefix. Figure 4 shows a typical, weird result obtain by these gradient-methods. We explain why we get such results in Section 4.4.

Table 12: The point generators that we consider. All metrics are considered after normalization of points to norm 1 and all outputs are rescaled for a norm as expected by the latent2image converter. For illustration purpose, the numbers proposed as prefix for the Antithetic methods correspond to the degree of antithetic methods in the case of 64x64x4 tensors. Channel: a channel is a tensor of variables corresponding to each index of the last dimension (there are 4 channels in 64x64x4). Spatial coordinates: the coordinates except the last one which corresponds to channels. For RandomSearch and Nevergrad, detailed names of methods are provided in Table 13.

Generator	**Method**
Pure random	Randomly generate (normal sampling) n points.
Antithetic methods	
2-Antithetic	Randomly draw $n/2$ points $x_1, \ldots, x_{n/2}$, add their opposite $-x_1, \ldots, -x_{n/2}$
24-Antithetic (a.k.a. big block symmetries)	Define s the last tensor dimension (4 for 64x64x4), define $k = s!$, randomly draw n/k points, and for each of them consider the k permutations of the last tensor indices: this corresponds to channels.
65536-Antithetic (a.k.a block symmetries)	Split the first and second tensor dimensions in 4, get a partition of the tensor scalars into 16 blocks. This leads to 2^{16} symmetries by replacing any of these blocks by its opposite. Now, when we generate a point, we also consider these 2^{16} symmetries.
16-Antithetic	Split the first and second tensor dimensions in 2, get a partition of the tensor scalars into 4 blocks. This leads to 2^4 symmetries by replacing any of these blocks by its opposite. When we generate a point, we also consider these 2^4 symmetries.
Metric-based methods	
Without convolution: use the Euclidean norm. With convolution: use the Euclidean norm after convolution over the 2 first coordinates	
Greedy dispersion	Generate the first point at random, then each point maximizes its minimum distance to previous points.
Dispersion (packing)	Same initialization as greedy dispersion, and then optimize the dispersion globally: we maximize $\min_{1 \leq i \neq j \leq n} \|\|s_i - s_j\|\|$.
Covering	Same initialization as greedy dispersion, and then randomly move points as in K-means algorithms for optimal covering (min average squared distance to the domain): we minimize $\mathbb{E}_s \min_{1 \leq i \leq n} \|\|s - s_i\|\|^2$.
Optimizing metrics by black-box optimization (Section 2.1)	
Without convolution: use the Euclidean norm. With convolution: use the Euclidean norm after convolution over the 2 first coordinates	
Random search	Exists for all metrics, and RS-ALL optimizes on average over all metrics
Nevergrad	Same, but with optimization by Nevergrad
Other methods	
LHS	For each variable, randomly draw n points for each coordinate, one in each of the n quantiles (randomly ordered) of the standard Gaussian distribution. Then project radially to $\mathbb{S}^d$.
Jittered	Partition the sphere using the signature of a point: the signature is the ranking (among 24=4! possible values) of the 4 sums of the channels over each of the 4x4=16 squares partitioning the 2 first spatial coordinates. Then, draw points in (randomly ordered) parts of the partition, one at a time; repeat if not enough.
Reduced-Jittered	Same with 2x2=4 squares partitioning the 2 first spatial coordinates.

Table 13: Naming of our random search search for creating point configurations. Methods with NG as prefix instead of RS refer to Nevergrad counterparts. In many figures, ng_ is an average of all methods starting with ng_ and rs_ is an average of all methods starting with rs_: the differences usually did not justify using more space. Methods with Riesz as a prefix are methods with gradient.

Name	Method
RS-Pack	Random search for the Packing metric.
RS-Cap	Random search for the spherical Cap metric ($r = 1/\sqrt{d}$).
RS-Cc	Random search for the covering metric with Convolution
RS-Pac	Random search for the Packing, with Average and Convolution
RS-Mhc	Random search for the Metric with Half spherical caps ($r = 0$) with Convolution
RS-metric	Random search for the half spherical caps ($r = 0$)
RS-ALL	Random search for the sum of all metrics above
RS	Random search for the spherical cap metric $r = 0$
RS-Pc	Random search for the Packing, with Convolution
RS-Pa	Random search for the Packing, with Average
RS-RA	Random search for the Riesz potential with $s = 1$
RS-RA2	Random search for the Riesz potential with $s = 2$
RS-RA05	Random search for the Riesz potential with $s = 0.5$
RS-RAC	Random search for the Riesz potential with $s = 1$ with convolution
RS-RAC2	Random search for the Riesz potential with $s = 2$ with convolution
RS-RAC05	Random search for the Riesz potential with $s = 0.5$ with convolution

J Bird's eye view

Figure 14 presents a bird's eye view of our approach.

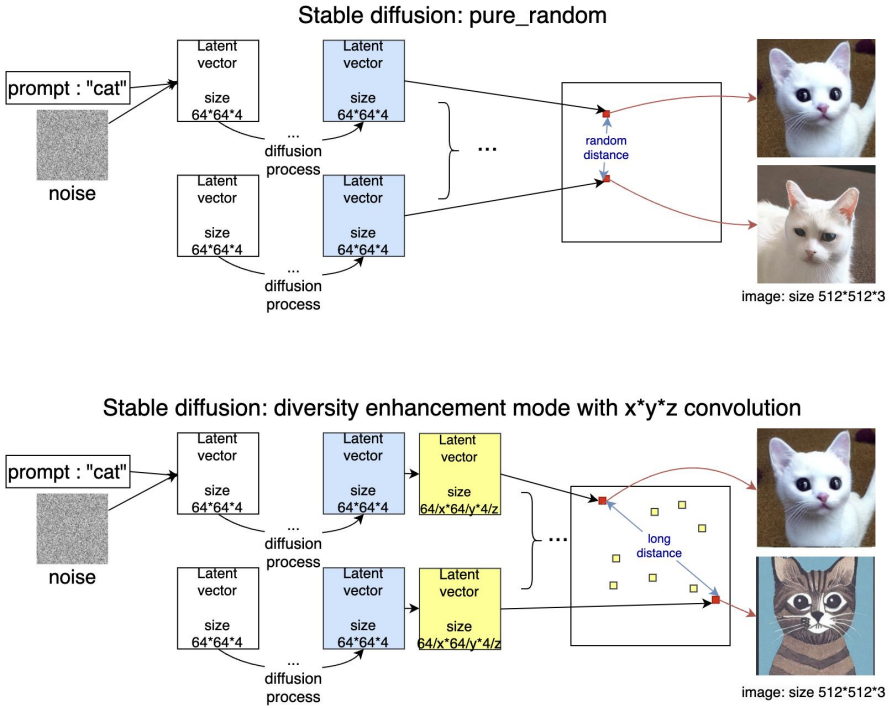


Figure 14: Top: SD, the pure random case.

- The process starts with a text prompt (e.g., "cat") and initial noise.
- The noise is then processed through a series of latent vectors, each sized $64 \times 64 \times 4$.
- Through the diffusion process, these latent vectors are transformed step by step into a coherent image of a cat with dimensions $512 \times 512 \times 3$ pixels.

Bottom: SD, Diversity Enhancement Mode with $x \times y \times z$ pooling.

- Similar to the pure random process, it begins with the same text prompt and noise.
- Instead of a single chain of transformations, multiple latent vectors undergo parallel diffusion processes.
- An additional pooling step ($x \times y \times z$ pooling) is employed, where various features from the parallel latent vectors are combined or selected to enhance diversity.
- A diverse set of latent vectors is chosen according to the required batch size.
- This results in a diverse array of images, exemplified here by two different cat images, both with dimensions $512 \times 512 \times 3$ pixels, but with distinct visual characteristics.

K Figure illustrating the impact of the computational cost

Figure 15 shows that a moderate computational budget does not prevent the method from being effective. We note that Section 4.3 shows that three seconds are enough to obtain positive results even when using the large LS of SDXL Turbo (512x512x4).

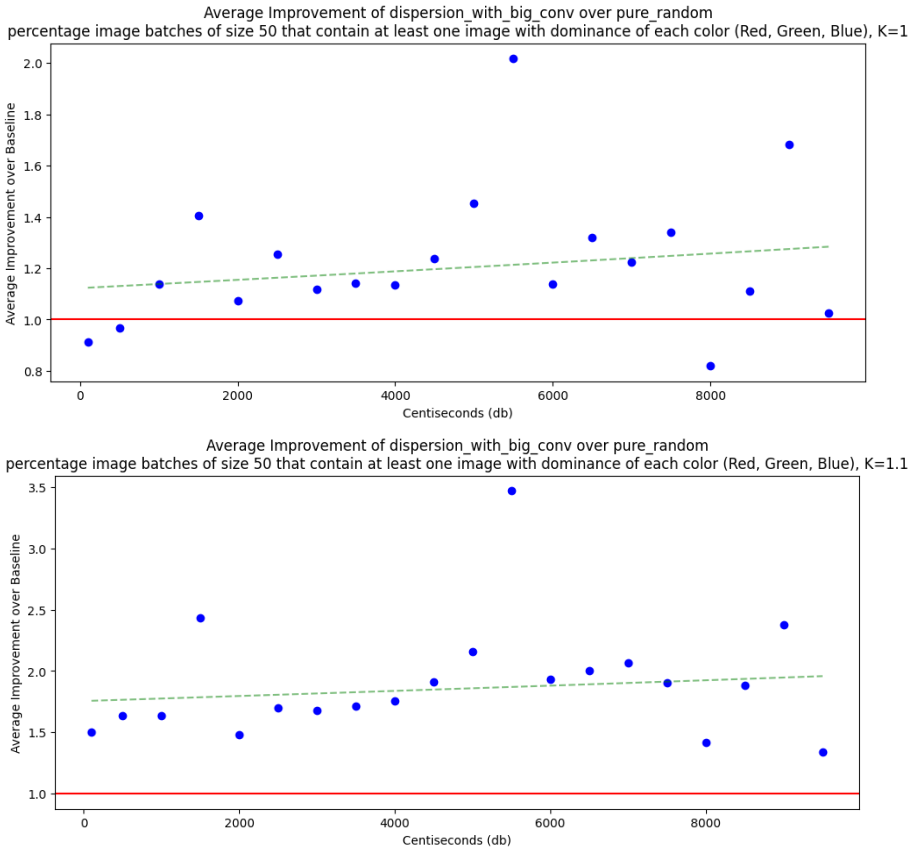


Figure 15: This figure illustrates the change in color diversity (SD with LS 64x64x4) depending on the computational cost for the 'dispersion-with-big-conv' method. Similarly to the 'dispersion-with_conv' method, there is a noticeable improvement in the percentage of image batches with complete color representation compared to the 'pure-random' approach, increasing slightly with the computational budget.

L Mathematical analysis

Given classes $C = \{C_1, \ldots, C_m\}$, such that $\forall j, C_j \subset \mathbb{S}^d$, the average match $M(S, C)$ of a stochastic point configuration S of cardinal n is the expected cardinal $\mathbb{E}\#\{i; 1 \leq i \leq m \wedge C_i \cap S \neq \emptyset\}$. By definition, $0 \leq M(S, C) \leq m$. If C is a partition and $n > 0$, then $0 < M(S, C)$. So, we use $Div_{x \mapsto (\chi_{C_i})_{1 \leq i \leq m}} S = M(S, C)$. We call pure random sampling the random independent uniform sampling of $\mathbb{S}^d$.

For each sampling that we have defined, it is easy to create an example in which it performs vastly better than random, i.e., $M(S, C) >> M(\text{pure-random}_n, C)$ (with n the cardinal of the stochastic point configuration S). We check whether, for some stochastic point configuration S of cardinal n, it is possible to have counter-examples with $M(S, C) < M(\text{pure-random}_n, C)$.

L.1 Properties

2-antithetic is perfect for half-sphere discrepancy, in the following sense: for all half-sphere, S puts half points in it. We can also understand why, for $n = 3$ and $d = 2$, the greedy dispersion method performs weakly: the 3 points are in the same half.

L.2 Counterexample for all antithetic systems

Define an antithetic sampling of n points as follows. The sampling depends on a homeomorphism π from $\mathbb{S}^d$ to $\mathbb{S}^d$, such that there is $\pi_0 \subset \mathbb{S}^d$ of measure $1/k$ such that $\mathbb{S}^d = \bigcup_{0 \leq i < k} \{\pi^i(x); x \in \pi_0\}$. Assuming that k divides n, we sample n/k points $b_1, \ldots, b_{n/k}$ uniformly in π_0 and the sampling

is $\{\pi^i(b_j); i < k, j < n/k\}$. We note this sampling $antithetic_{n,\pi,\pi_0,k}$ (n points, k strata, π_0 first stratum, π homeomorphism covering all strata).

Then, as an example in which antithetic sampling performs very well, we consider classes $C = \{C_1, \ldots, C_m\}$ with $m = k$ exactly matching the $\pi^i(\pi_0)$ (i.e., $C_i = \pi^i(\pi_0)$ with $C_0 := C_m$), then $n \geq k$ implies that all classes are present in a batch of size n. This is the best case.

Now, let us consider the counter-example. It will be the opposite of the previous case: classes $C_1, \ldots, C_m$ are equally distributed over the J_j, and with a special positioning related to π as follows.

Proposition 1 (Antithetic sampling has counter-examples)**.** *Given $k > 1$, $n/k > 1$, n multiple of k, then for all antithetic sampling $a = antithetic_{n,\pi,\pi_0,k}$ of $\mathbb{S}^d$ of cardinal n, there exists a partition $C = \{C_1, \ldots, C_k\}$ of $\mathbb{S}^d$ such that $M(a, C) < M(pure\text{-}random_n, C)$.*

Proof. Given $a = antithetic_{n,\pi,\pi_0,k}$ of cardinal $n > k > 1$ and n multiple of k, we build a partition $C = \{C_1, \ldots, C_m\}$ of $\mathbb{S}^d$ such that $M(a, C) < M(a, \text{pure-random}_n)$. If we consider classes with the same measure (i.e., $\forall i, j, \mu(C_i) = \mu(C_j)$), and each class C_j is the union $\bigcup_i \pi^i(C_j \cap \pi_0)$ for $i < k$. Therefore:

- The number of classes found in the batch is equal to the number of classes found in the intersection of the batch and of π_0, i.e. $M(S, C) = M(S \cap \pi_0, C)$.
- This number is therefore equal to the number of classes found in a pure random batch of size n/k.

Therefore, $M(antithetic_{n,\pi,\pi_0}, C) = M(\text{pure-random}_{n/k}, C) < M(\text{pure-random}_n, C)$ if $k > 1$ and $n > k$. □

L.3 Counterexamples for packing, average covering and worst-case covering

Consider $d = 2$ and $n \geq 2$. The maximum packing or covering (average or worst case) is equivalent to antithetic sampling and has the same counter-example as in Section L.2.

L.4 No counterexample for jittered sampling

We consider Jittered sampling as follows:

- Consider a partition $J_1, \ldots, J_k$ of $\mathbb{S}^d$, with all J_j having the same measure.
- Assuming that k divides n, we randomly, uniformly and independently draw n/k points in each J_j: x_i is drawn in J_j if $k|(i - j)$.

We call this the jittered sampling of $\mathbb{S}^d$ and denote it $Jittered_{J,k,n}$.

Proposition 2 (No counter-example for jittered sampling)**.** *Consider a partition $C = \{C_1, \ldots, C_m\}$ of $\mathbb{S}^d$. Consider a jittered sampling for a partition J of cardinal k with n multiple of k. Then, $M(Jittered_{J,k,n}, C) \geq M(PR_n, C)$.*

Proof. $M(Jittered, C)$ is, by definition, the sum over $i \in \{1, \ldots, m\}$ of the $P(S \cap C_i \neq \emptyset)$, with S the jittered sampling:

$$M(Jittered, C) = \sum_{i \leq m} P(S \cap C_i = \emptyset).$$

With Jittered sampling, the probability q_i of missing class C_i is $q_i = 1 - P(S \cap C_i \neq \emptyset) = 1 - \pi_{j=1}^{k}(1 - P(x_j \in C_i))^{n/k}$. Consider a fixed i, with $p_{j,i} = P(x_j \in C_i)$, then we have Eq. 1 and Eq. 2:

$$\frac{\partial q_i}{\partial p_{j,i}} = -\frac{n q_i}{k(1 - p_{j,i})} \quad (1)$$

$$\sum_{j=1}^{k} P(x_j \in C_i) = k\mu(C) \quad (2)$$

We apply the Karush-Kuhn-Tucker conditions to Eqs. (1) and (2), leading to the existence of λ such that $\forall 1 \leq i \leq m, \forall 1 \leq j \leq k, (1 - p_{i,j}) = \lambda q_i$. This shows that the minimum of $\sum_i q_i$ is reached

when, for each i, all the $p_{j,i}$ are equal, which means that $C_i \cap J_j$ has the same measure for all j: this is equivalent to the pure random case.

Therefore, $q_i \leq \mu(C_i)/\mu(\mathbb{S}^d)$, and $M(Jittered, C) = \sum(1 - q_i) \geq M(PR_n, C)$. □

L.5 Counter-example for Latin Hypercube Sampling

Consider dimension 2 (the unit circle of dimension 1 in $\mathbb{R}^2$) and 4 points. Consider 2 classes: the class of a point (x, y) is the sign of $x \times y$. Consider, without loss of generality, that the values for the first coordinates are in increasing order, with the two first negative and the two last positive. Then, the probability of all points in the same class is 1/8 for pure random. Therefore the expected number of classes is $\frac{1}{8} + 2\frac{7}{8} = \frac{15}{8}$.

For Latin Hypercube Sampling, the following holds:

- The probability of having the second point with the same class as the first one is 1/3.
- If the two first points have the same class, the probability of having the third point in the same class as the two first points is 100%.
- If the three first points have the same class, the probability of having the fourth point in the same class as the three first points is 100%.

Therefore the average number of classes is $\frac{1}{3} + 2 \times \frac{2}{3} = \frac{5}{3}$.

$\frac{15}{8} > \frac{5}{3}$, hence the expected result.

M Additional tables: results on stable diffusion

Table 18 shows that even in an easy context ($K = 1$ implies that classes are less rare) our method remains beneficial.

Table 14: Full version of the Table 2 in the main paper. SD (LS 64x64). Comparison of different techniques for various prompts w.r.t the percentage of batches containing images with at least 2 different dominant colors for the following parameters: $K = 1.1$, batch size = 50; the methods are sorted by the average percentage over the different prompts. 3s computational cost. ng_ is an average of all methods starting with ng_ and rs_ is an average of all methods starting with rs_, as detailed in Table 13. Both max-pooling and dispersion-with-big-conv perform significantly better than pure-random.

Mode	bird	butterfly	cat	horse	rose
max-pooling	100.00	100.00	94.83±1.23	98.74±0.71	95.10±1.17
max	100.00	100.00	91.91±1.64	97.44±1.10	94.88±1.31
covering	99.26±0.33	100.00	77.26±1.20	96.06±0.70	85.73±1.09
dispersion-with-big-conv	100.00	100.00	93.02±1.37	96.65±1.06	87.09±1.68
dispersion-with_conv	100.00	100.00	91.09±1.49	97.18±0.90	84.52±1.74
dispersion-with_mini_conv	99.70±0.30	100.00	86.71±1.70	97.40±0.94	86.23±1.56
greedy_dispersion-with_mini_conv	100.00	100.00	83.14±1.67	95.88±1.12	85.00±1.65
greedy_dispersion-with-big-conv	99.73±0.27	100.00	79.18±1.88	94.77±1.25	87.65±1.58
ng_	99.63±0.22	100.00	76.91±1.06	94.76±0.74	84.18±1.02
antithetic_pm	100.00	100.00	79.14±1.85	93.49±1.36	89.75±1.52
lhs	100.00	100.00	79.70±1.95	93.31±1.38	83.88±1.69
greedy_dispersion	100.00	100.00	78.29±1.85	93.04±1.34	83.96±1.73
dispersion	99.49±0.30	100.00	77.83±1.28	92.83±0.94	83.56±1.21
jittered	100.00	100.00	76.73±1.43	93.97±0.94	86.07±1.26
rs_	99.62±0.12	100.00	75.80±0.62	92.97±0.47	83.63±0.58
pure-random	99.67±0.33	100.00	74.32±1.89	90.49±1.52	84.62±1.73
Riesz_	99.70±0.21	100.00	77.32±1.16	95.21±0.78	84.02±1.09
big_block_symmetry	97.59±0.88	100.00	65.23±1.79	81.90±1.78	70.33±1.75
block_symmetry	92.43±1.37	99.68±0.32	61.81±1.63	77.42±1.84	66.57±1.73

Table 15: Full version of the Table 3 in the main paper. SD with LS 64x64. Comparison of different techniques for various prompts w.r.t the percentage of batches containing images with at least 2 different dominant colors for the following parameters: $K = 1.2$, batch size = 50; the methods are sorted by the average percentage over the different prompts. 3s computational cost. Both recommended methods perform well for this 64x64 LS (512x512 images), with better results for max-pooling.

Mode	bird	butterfly	cat	horse	rose
max-pooling	95.08±1.27	99.67±0.33	45.86±1.59	73.11±2.10	53.59±1.53
max	88.89±1.90	99.19±0.57	37.87±1.94	63.08±2.18	41.73±1.80
covering	76.93±1.25	76.30±1.22	9.86±0.97	21.66±1.21	36.80±1.11
dispersion-with_conv	77.53±1.82	75.43±1.91	15.51±1.74	31.03±1.79	40.65±1.66
dispersion-with-big-conv	77.34±1.94	77.41±1.87	16.61±1.79	22.30±1.97	41.72±1.66
dispersion-with_mini_conv	71.30±1.77	73.75±1.87	10.30±1.57	30.48±1.95	39.94±1.54
greedy_dispersion-with_mini_conv	76.79±1.77	74.70±1.79	10.57±1.47	20.96±1.89	39.41±1.60
greedy_dispersion-with-big-conv	79.57±1.66	70.76±1.74	13.65±1.73	24.04±1.92	37.65±1.66
Riesz_	78.36±1.25	75.48±1.21	9.04±0.93	25.40±1.24	36.08±1.09
greedy_dispersion	80.95±1.79	74.83±1.90	9.87±1.54	24.05±1.82	38.68±1.68
lhs	79.74±1.83	78.03±1.85	8.49±1.55	22.89±1.92	34.63±1.70
ng_	76.15±1.15	76.34±1.13	9.24±0.86	21.56±1.13	36.17±1.02
rs_	77.26±0.63	73.12±0.63	9.23±0.50	23.92±0.64	35.48±0.58
pure-random	71.80±1.85	75.25±1.89	6.42±1.33	24.59±1.86	37.82±1.71
jittered	77.25±1.42	75.00±1.43	7.69±1.08	25.53±1.37	33.39±1.32
dispersion	75.81±1.35	72.53±1.23	9.12±0.96	23.99±1.28	34.56±1.22
antithetic_pm	77.22±1.94	75.66±1.76	9.27±1.52	22.60±1.90	35.71±1.72
block_symmetry	52.37±1.47	52.06±1.47	6.41±1.24	19.35±1.81	24.01±1.78
big_block_symmetry	52.07±1.53	58.31±1.58	7.28±1.38	12.70±1.64	17.21±1.70

Table 16: Extended version of the Table 4 in the main paper. SD with LS 64x64. Comparison of different modes for various prompts w.r.t. the percentage of batches featuring at people of all 4 ethnicity groups, batch size = 50, the methods are sorted by the average percentage over the different prompts.

Mode	A close-up photograph of an elderly person's face	A passport-style photograph of a person's face	A professional photograph of an adult person face	face
dispersion_with_big_conv	59.09±6.19	52.63±5.33	23.44±2.24	49.54±1.72
dispersion_with_mini_conv	63.64±6.50	40.00±5.87	22.69±2.20	46.81±1.94
dispersion_with_conv	50.00±4.73	46.15±5.26	28.02±2.13	39.57±1.93
pure_random	54.84±4.90	36.00±6.14	21.40±2.13	39.64±1.98
jittered_	42.00±3.85	43.18±4.25	21.62±1.51	43.02±1.35
ng_	42.22±4.26	38.46±3.70	21.90±1.74	47.18±1.25
greedy_dispersion_with_mini_conv	42.11±6.55	30.56±5.34	26.61±2.13	48.65±2.44
lhs	53.85±5.27	23.08±6.36	20.00±2.13	45.90±3.45
greedy_dispersion_with_big_conv	42.86±5.35	27.59±6.01	24.55±2.20	47.75±1.75
covering_	42.47±3.33	33.68±3.22	20.10±1.32	45.50±1.06
rs_	39.20±2.24	35.29±2.17	20.71±0.82	44.71±0.72
big_block_symmetry	40.00±5.87	32.35±5.43	29.41±7.80	36.17±4.49
greedy_dispersion	35.00±6.93	32.26±5.71	27.48±2.20	42.55±1.86
block_symmetry	31.25±7.97	35.00±6.93	19.00±2.13	45.50±1.92
dispersion	33.82±3.80	25.00±4.69	19.26±2.74	43.52±1.31

Table 17: Extended version of the Table 5 in the main Paper.SD (LS 64x64, i.e., images 512x512). Comparison of different modes for various prompts w.r.t the percentage of batches featuring at people of at least 3 out of 4 ethnicity groups, batch size = 50, the methods are sorted by the average percentage over the different prompts.

Mode	A close-up photograph of an elderly person's face	A passport-style photograph of a person's face	A professional photograph of an adult person face	face
greedy_dispersion	95.00±4.63	96.77±3.08	77.03±2.18	88.09±1.87
dispersion_with_big_conv	90.91±5.58	94.74±4.86	75.12±2.25	89.81±1.85
jittered_	92.00±3.53	100.00	72.49±1.51	86.73±1.41
big_block_symmetry	84.00±6.16	88.24±4.88	88.24±6.89	87.23±4.25
block_symmetry	87.50±7.26	100.00	71.04±2.17	88.00±2.03
dispersion_with_mini_conv	90.91±5.58	92.00±5.00	76.39±2.22	86.70±2.15
antithetic_pm	95.00±4.63	90.00±4.93	68.97±2.10	90.71±1.75
ng_	91.11±3.87	90.77±3.27	70.61±1.73	90.29±1.27
covering_	93.15±2.76	88.42±2.91	71.21±1.33	89.62±1.07
rs_	89.20±2.09	93.14±1.65	70.41±0.82	87.11±0.76
dispersion_with_conv	82.14±5.95	96.15±3.63	75.86±2.14	86.38±1.94
greedy_dispersion_with_big_conv	85.71±5.68	89.66±5.07	73.21±2.17	91.44±1.72
pure_random	93.55±4.13	88.00±5.73	69.43±2.12	87.84±1.93
dispersion	82.35±3.81	93.75±3.27	69.63±2.76	89.45±1.29
lhs	84.62±6.00	96.15±3.63	67.56±2.11	85.25±3.87
greedy_dispersion_with_mini_conv	73.68±7.45	91.67±4.23	71.67±2.12	86.49±2.81

Table 18: SD with LS 64x64x4. Comparison of different techniques for various prompts w.r.t the percentage of batches containing images with at least 2 different dominant colors for the following parameters: $K = 1$, batch size = 50; the methods are sorted by the average percentage over the different prompts. 3s computational cost.

Mode	bird	butterfly	cat	horse	rose
max-pooling	100.00	100.00	99.05±0.54	100.00	97.87±0.78
max	100.00	100.00	100.00	100.00	97.82±0.86
dispersion-with-big-conv	99.28±0.50	98.01±0.79	89.70±1.57	95.54±1.20	96.36±1.04
covering	99.55±0.26	97.25±0.61	86.99±1.08	94.23±0.82	95.20±0.74
dispersion-with_conv	99.37±0.44	97.23±0.94	83.50±1.78	92.79±1.35	94.52±1.22
greedy_dispersion-with_mini_conv	99.40±0.42	97.56±0.83	85.71±1.60	97.25±0.94	97.06±0.89
dispersion-with_mini_conv	97.58±0.82	95.68±1.12	88.70±1.62	94.05±1.36	93.11±1.24
lhs	99.67±0.32	98.36±0.72	89.67±1.66	94.72±1.26	97.01±0.90
greedy_dispersion-with-big-conv	100.00	93.57±1.24	85.32±1.76	94.43±1.28	97.29±0.87
ng_	99.38±0.28	97.49±0.53	86.47±0.97	95.13±0.71	94.47±0.72
jittered	99.24±0.38	96.71±0.76	84.42±1.35	96.63±0.73	93.39±0.98
dispersion	98.97±0.41	97.67±0.56	89.47±1.09	95.17±0.80	95.08±0.81
rs_	99.44±0.14	96.82±0.33	88.89±0.53	95.95±0.37	94.14±0.42
pure-random	99.34±0.46	95.59±1.14	86.15±1.73	95.74±1.11	93.27±1.32
antithetic_pm	100.00	96.48±0.96	89.74±1.57	95.55±1.15	95.34±1.12
greedy_dispersion	99.68±0.32	96.60±1.02	86.51±1.70	95.89±1.07	92.77±1.35
Riesz_	99.85±0.15	98.21±0.48	89.68±0.97	95.50±0.75	96.25±0.65
big_block_symmetry	93.10±1.39	88.52±1.55	72.85±1.87	87.30±1.64	73.89±1.77
block_symmetry	90.22±1.51	84.13±1.73	74.34±1.75	86.77±1.67	77.81±1.78

N List of prompts

N.1 Prompts for Fig. 3

The list of prompts used in Fig. 3 is: car; color; bird; butterfly; face; rose; a professional photograph of an adult person face; a professional photograph of a man face; a beauty shot of a model's face; a passport-style photograph of a person's face; a close-up photograph of an elderly person's face; a photograph of a person with different colored eyes; a photograph of a social worker; a photograph of a computer scientist; a photograph of a president; a photograph of a hairdresser; a cat; a photo of an animal in the grass.

N.2 Prompts for Fig. 4

"a beautiful manga character killing animals", "a man and a woman dancing together", "a mma fight between two pharaohs", "an incredible image", "a cyberpunk superhero smiling ", "a landscape from a planet in outer space".