Interactive Latent Diffusion Model

Mathurin Videau, Nickolai Knizev, Alessandro Leite, Marc Schoenauer, Olivier Teytaud

Interactive Latent Diffusion Model

Mathurin Videau
Meta & Inria
Paris, France

Nickolai Knizev
Meta
London, UK

Alessandro Leite
TAU, Inria Saclay
Orsay, France

Marc Schoenauer
TAU, Inria Saclay
Orsay, France

Olivier Teytaud
Meta & Inria
Paris, France

ABSTRACT

This paper introduces Interactive Latent Diffusion Model (IELDM), an encapsulation of a popular text-to-image diffusion model into an Evolutionary framework, allowing the users to steer the design of images toward their goals, alleviating the tedious trial-and-error process that such tools frequently require. The users can not only designate their favourite images, allowing the system to build a surrogate model based on their goals and move in the same directions, but also click on some specific parts of the images to either locally refine the image through dedicated mutation, or recombine images by choosing on each one some regions they like. Experiments validate the benefits of IELDM, especially in a situation where Latent Diffusion Model is challenged by complex input prompts.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Classification and regression trees**; **Learning latent representations**.

ACM Reference Format:
Mathurin Videau, Nickolai Knizev, Alessandro Leite, Marc Schoenauer, and Olivier Teytaud. 2023. Interactive Latent Diffusion Model. In *Proceedings of The Genetic and Evolutionary Computation Conference 2023 (GECCO'23).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3583131.3590471

1 INTRODUCTION

Latent Diffusion Models (LDMs) become very popular. LDMs have allowed a real popularization of the synthesis of high-resolution images, compared to previous approaches that required orders of magnitude more compute time and energy to train a model, and even to use an already trained one. Although several applications can be developed using LDMs, like inpainting, colourization, stroke-based synthesis etc., this work focuses on image synthesis from text, a field that has gained a high momentum in the recent months, at the point that an open platform exists, and that allows users to generate images from free text within a few minutes.

However, though much more stable than Generative Adversarial Networks (GAN), and much more tractable than bare Diffusion Models, LDM still exhibits some limitations, that are reflected in the repeated posts on open discussion groups such as:

(1) **Issue 1**: missing features in complicated compositions (e.g., character X on the left, using tool Y, with character Z on the right staring at X), especially if these features never appeared in the same image in the training set. Partial solutions have been proposed, like using an image to condition the new image generation. However, LDM cannot easily "understand" an image enough to use it as a starting point. For example, some users mention "I've noticed with portraits that a lot of times people show up that don't look anything like the picture".
(2) **Issue 2**: biases due to unusual statistics in the training set: when a person is known publicly only from images in front of an audience (e.g., people who are, in the training dataset, frequently interviewed by journalists), it is difficult to obtain an image in which they do something else . Or, for a famous singer frequently depicted next to biblical creatures, a user reported obtaining a mix between a cow and a devil. Other users also report failing to reproduce some memes like "why can't I hold all these lemons". But the most famous example of such biases concerns requests of salmons swimming in the river that, because of too many images of salmon food in the training set, led to salmon fillets "swimming" in the river, including a geyser as if they were whales, or a bear trying to catch them.
(3) **Issue 3**: need for many re-runs, either because of local issues (e.g., four-armed people, three-eyed persons, or people with too many fingers) or because of large-scale errors (e.g., bad positioning of different characters).

These unrealistic results are routinely tackled by running multiple times similar requests, until the output fits the users' initial idea. Hundreds of rolls are usual for people working professionally with LDM: such trial-and-error practice can be viewed as a random search in the space of images. *In this context, the main goal of the present work is to add to LDM the possibility to perform more efficient search, allowing the user to steer the search, hence getting better results more quickly.*

Several components are needed to this aim: a search space, an objective function defined on this search space, and a search procedure to tackle this objective function by efficiently sampling the search space. Searching the space of (possibly high resolution) images is almost out-of-reach due to the curse of dimensionality. However, one of LDM features is to learn a latent space, that reflects the perceptual properties of the images, and is of much lower dimension than the space of pixels, while retaining the useful information for

GECCO'23, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0119-1/23/07…$15.00
https://doi.org/10.1145/3583131.3590471

the task at hand. Furthermore, LDM uses a latent representation that has the same spatial structure as that of the images, making it easy to move between both representations. From an evolutionary computation perspective, this latent space can be seen as a genotype, and the genotype is reconstructed by the decoder part of LDM. It seems hence possible to search the latent space at a tractable cost. It is even possible to allow the users to act on the image, porting their actions to the latent space where the search takes place, an original function that is not possible in PicBreeder or ArtiE-Fract, for instance, (to be discussed next).

Indeed, in general, even users are unable to define their goals in analytical terms, as they are summarized in a simple sentence, that cannot capture all the details of user's idea. The definition of objective function is a clear case for Interactive Evolution, a process in which the user repeatedly rates the images that are proposed by the system, driving indirectly the search. However, to avoid the biases due to the user's fatigue [20], one can build a surrogate model of the user's ratings, gradually learning to quantify the user's hidden goal, thus decreasing the number of interactions before a satisfactory result is obtained. Interactive Evolutionary Computation (IEC) has been around since the early ages of Evolutionary Computation, when Richard Dawkins evolved his insect-like animals [2]. Excellent surveys of the field of IEC have been written by Hiroyuki Takagi [9, 17, 18], that showed how the field has been constantly active, and was applied in many domains, quite often to explore design spaces of engineering applications [16, 20]. When it comes to evolving images, for instance, EIC has been applied to evolve GP functions that define a fractal image [6], where, interestingly, saved populations of past evolutions were used by the artist like a paint palette: the only way to guide the design process was to choose carefully the initial population from past results... and hope for the best. A similar experience was demonstrated by PicBreeder, Ken Stanley's collaborative platform where users could choose pictures evolved through applications of CPPN [13]. However, in those examples, the users can only interact with the genotype (if knowledgeable enough, as this is not available in the proposed public interfaces), and hope that the resulting phenotype (the plain image) approaches their goals.

This paper introduces Interactive Latent Diffusion Model (IELDM), an interactive evolutionary system encapsulating LDM. At each generation, LDM produces λ images, and the user can choose to rate them, and to interact with none, one, or several of them. In particular, it comprises an interface allowing users to (a) indicate their preferences, and these preferences will be used to build a surrogate model of their goals; (b) select different images for crossover or mutations, possibly even selecting several regions of the images where they would like the variation operators to operate. Of course, the result of these operators is still stochastic, but the guidance of the user is much more directed than in all previous works discussed above.

The paper is organized as follows. Section 2 describes the existing basic bricks used by IELDM. Section 3 details the different components of the evolution involving interactions with the user. Finally, Section 4 presents the experimental results that validate the proposed approach, detailing in particular how to increase certain characteristics of an image, how to handle out of statistics prompt, and how to make local editions.

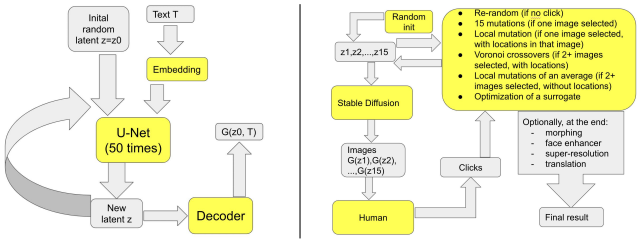

Figure 1: Left: Schematic view of LDM (see Section 2.1) with $K = 50$. Right: Bird's eye view of IELDM (see Section 3) with $\lambda = 15$: the Latent Diffusion Modelblock is the block on the left.

2 TOOLS

This section describes the basic existing bricks that have been used to design IELDM: the Latent Diffusion Model algorithm that is encapsulated inside IELDM, and the different components that are used to build the offspring images from the selected ones.

2.1 Latent Diffusion Model

The text-to-image model we are using is a Latent Diffusion Model. It is used here as an image synthesizer from a text description. For lack of space, and also because LDM is used as a black box here, we will not detail here the complete architecture of LDM, that involves several components like denoising variational auto-encoders and attention mechanisms. Figure 1-Left displays a schematic view of the generation process of LDM (i.e., after training): an initial random z in the latent space undergoes K iterations of a trained U-Net, conditioned by the description text, encoded with a trained embedding. The iterated z slowly moves closer and closer to the description embedding. The resulting point in the latent space is then decoded into an image $G(z)$, or, more precisely, $G(z, embedding(text))$, as it strongly depends on the description text T.

We use a latent diffusion model, so that z is a normal distribution, with shape $4 \times 64 \times 64$ and variance one in each coordinate. Furthermore, G is deterministic, which allows us to define specific crossover and mutation operators.

Latent Diffusion Model is at the heart of IELDM, and its latent space is the search space of the whole algorithm. Note however that the only requirement needed to search a latent space is a decoder that transforms points of the latent space into images: in evolutionary terms, the latent space can be viewed as the genotypic space, the image space as the phenotypic space, and the decoding function G is similar to the morphogenesis. Hence, from an evolutionary perspective, searching such a latent space "only" requires the definition of variation operators (i.e., crossover and mutation) on the latent space. From the perspective of image synthesis, any generative method for images that defines a latent space with a generative model G could have been used, like GANs [1, 4, 5, 7]. However, beside their lack of robustness (like e.g., mode collapse), one of the ambitions here is to be able to define operators that use information from the image space, as entered by the user: the latent space z needs to be spatially distributed, which is the case for LDM, but not for GANs.

Figure 2: *Prompt: Three cute monsters walking in the grass.* Illustration of the Voronoi crossover for combining the left part of an image and the right part of another: we combine the left monster of the first image of the top row (violet) and the right monster (half visible) of the first image of the second row (orange). We get the $\lambda = 13$ other images: the first new image (bottom left) is by construction (see the contraction factor in Section 3.3) the one with the lowest diversity: it matches exactly the requirement of the left violet monster and the orange half monster on the right. Other images match more loosely.

2.2 Fitness function

The fitness function is defined from users' inputs regarding some high-level features that meet their goal, taste, or expectation. From thereon, a surrogate model is built on the fly (after every user input), and is here possibly used, depending on the user, until the next iteration. We will describe in turn the user inputs, the learning tools that are used to build the surrogate fitness, and the black-box optimizer used to maximize this proxy-fitness.

2.2.1 Image Quality. To train the surrogate model, we keep each generated image along with their *Quality* in an archive. In this archive, each image is labelled either *Good* or *Bad* by using users' clicks. Therefore, we consider an image as *Good* when the user has clicked on it and *Bad* in the opposite case. When the amount of data is sufficient (at least ten *Bad* images and one *Good* one), this small dataset is then used to train the surrogate model all along the evolutionary process.

2.2.2 Learning Toolbox. In the context of IELDM, the idea is to build a surrogate model of the user preferences in the latent space, from past pairs $(z, Quality(z))$. However, only a rather limited number of examples is available. Hence, we preferred to focus on Logistic Regression and a small Multi-layer Perceptron.

The *Scikit-learn* library is used in all experiments with default configurations, except for the MLP for which we used (solver='lbfgs', alpha=1e-5, hidden-layer-sizes=(5, 2), random-state=1).

To make optimization easier, the latent space undergoes a simple dimension reduction via a down-sampling mechanism: the latent space is of the form $(1, 4, 64, 64)$, and is down sampled to $(1, 4, 8, 8)$ by summing over groups of 8×8 pixels. The surrogate model is learned in this lower-dimensional space, without degrading the results. We discard the cases with zero *Good* or *Bad* samples

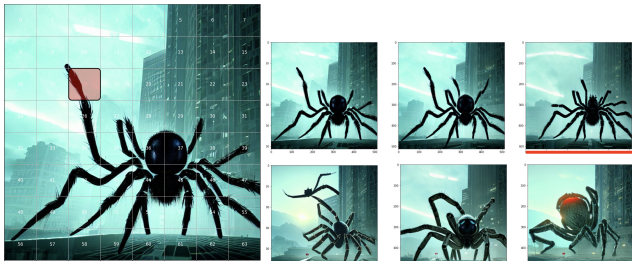

Figure 3: *A giant cyberpunk spider attacks a building.* Illustration of a local modification. Left: we do not like the leg of this spider at location 18 (the red square), so we request a modification at 18. Right: we get 6 new variants, including one without that (badly positioned) leg (underlined in red). Consistently with the specification (radius in Section 3.1), the first images are closer to the original and the last ones are more different.

2.2.3 Black-box Optimization. Searching for z maximizing some function directly involving the LDM generative model G is difficult: G involves a loop, which will make the gradient computation quite expensive. On the other hand, IELDM is based on human feedback (see Figure 1, right), and even when replaced by a surrogate model learned on the fly, it might require some black-box optimization for an efficient exploitation during the search (see Algorithm 1). In the experiments of Section 4, the powerful and versatile Nevergrad toolbox [11] has been used whenever possible (see Section 4.1).

3 INTERACTIVE LATENT DIFFUSION MODEL

A bird's-eye view of the IELDM algorithm is depicted on Fig. 1-Right, while Algorithm 1 details it more formally.

A population of size λ (15 in Fig. 1) evolved in the latent space (the genotypic space), is transformed by LDM into images (the phenotypes). The user interacts with those λ images by selecting $\mu \in [0, \lambda]$ of them and eventually then clicking more precisely on specific regions on the selected images. Depending on μ and the possible location clicks, the algorithm applies crossover, mutations, and/or uses a surrogate model (learnt on the fly) of the qualities of the most recent images to generate new images around the clicked ones (details in Algorithm 1). In all cases, λ new images are created from the interaction with the user. The rest of this section describes the different components of the algorithm.

3.1 Local mutations

Given a chosen image in an offspring, the user can choose to generate multiple variations of it by specifying points to be modified in the image (Line 9 of Algorithm 1). This will create λ images from the original one, following Algorithm 2: all "pixels" in the latent space that are sufficiently close from one of the user's click are replaced by a random value. Following the same normalized Gaussian distribution as all "pixels" of the latent "image" (Section 2.1). The λ images differ by the radius of the ball in which the perturbations

Algorithm 1 IELDM algorithm

Require: A Latent Diffusion Model $G : z \mapsto G(z, p)$, a text prompt p, a population size λ (15 by default)

1: Initialize $\sigma \leftarrow 1$, $\epsilon \leftarrow 0.01$, $archive \leftarrow [\,]$
2: Generate λ images by entering random Gaussian priors in LDM
3: **while** The user is not satisfied **do**
4: User clicks on μ images ▷ μ = user's choice
5: Update archive according to μ (clicked vs not clicked)
6: **if** $\mu = 0$ **then**
7: Randomly re-initialize the population
8: **else if** $\mu = 1$ **and** user clicks on several locations **then** ▷ Local mutation
9: Apply local mutations and get λ new images (Section 3.1)
10: **else if** $\mu = 1$ **then** ▷ Random mutation
11: Generate λ new latent values $z_1, \ldots, z_\lambda$, with noise $(\frac{i}{\lambda} \times \sigma)_{1 \leq i \leq \lambda}$ (Section 3.2)
12: **if** *archive* of at least 10 rejected and 1 clicked images **then**
13: Learn a surrogate model s of $z \mapsto Quality(G(z))$, using all non-clicked images and only the most recently clicked one.
14: For each i, use a black box algorithm to optimize $x \mapsto s(z_i + \epsilon x)$
15: with budget 30 iterations: best found x_i is the recommended perturbation:
16: $z_i \leftarrow z_i + \epsilon x_i$
17: $\sigma \leftarrow 0.7 \times \sigma$ (Section 3.2)
18: Generate the λ images $G(z_1), \ldots, G(z_\lambda)$.
19: **else**
20: Combine the μ selected images using the Voronoi operator (Section 3.3).
21: Post-treatment (Section 3.4)

will be made, that varies from $\frac{r}{20}\lambda$ to $\frac{r}{20}$ (r is the total width of the latent "image", and factor 20 has been empirically chosen after some trial-and-error tests.

3.2 Random mutations

When the user selects one and only one image in the current offspring, we can not apply the Voronoi crossover. Similarly, if the user selects multiple images, but without specifying some parts, we can not apply the Voronoi crossover: we might prefer to mutate an average of the selected latent variables. Therefore, we generate new elements, using an increasing level of noise. The user selects an image with latent variable z_0. We propose $(z_i)_{1 \leq i \leq 15}$ created as follows. We add a random noise proportional to $\frac{i-1}{\lambda}$, with a factor σ. For adapting the step-size across iterations, σ is decreased when the user selects a single image (Line 17 of Algorithm 1). We re-normalize generated images to a norm $\sqrt{dimension} = \sqrt{4 \times 64 \times 64}$ for matching the original prior of our baseline LDM code (normal with variance 1 per coordinate). The random mutations are not applied when we have multiple images selected by the user: then the Voronoi mutation ensures enough diversity (Section 3.3). We also ignore this random mutation mechanism when the user has specified local areas to be mutated (Section 3.1).

Algorithm 2 Local mutations

Require: A number λ of children to generate
Require: A Latent Diffusion Model pipeline G
Require: An input image $I = G(z)$ and its latent counterpart z of width r
Require: Coordinates $x_1, \ldots, x_k$ in the image. This is user's clicks where perturbations are asked.

1: Convert $x_1, \ldots, x_k$ to coordinates $x'_1, \ldots, x'_k$ in the latent space. ▷ Simple linear down-sampling here
2: **for** $i \in \{1, \ldots, \lambda\}$ **do**
3: Define $radius := (i/\lambda) \times r/20$ ▷ the first images are closer to $I = G(z)$
4: Create z' a copy of z ▷ (to be modified below)
5: **for** $j \in \{1, \ldots, k\}$ **do** ▷ Let us perturb z' around location x'_j
6: **for** each location p in the L^∞ ball of radius $radius$ centered on x'_j **do**
7: $z'_p \leftarrow \mathcal{N}^4(0, 1)$ (4-dimensional Gaussian) ▷ In the neighborhood of x'_j, the 4 channels are randomly drawn
8: yield $G(z')$ ▷ λ images generated

3.3 Voronoi crossover operator

When the users find interesting features that are present in different images, they have the possibility to click on several points, say of coordinates $p_1, \ldots, p_k$, in the different images $I_1, \ldots, I_n$, with $I_j = G(z_j)$ for some latent tensors $z_1, \ldots, z_k$. This will trigger (Line 20 of Algorithm 1) the application of a recombination of those images that preserve the areas around the clicked locations, using a crossover operator based on Voronoi diagrams, inspired from [14, 15]. An offspring image $I_{Voronoi} = G(z_{Voronoi})$ is built as follows, for a given contraction factor r:

$$z_{Voronoi}(x) = z_j(x) \text{ if } \forall u \neq j, ||x - p_j|| < \frac{1}{r}||x - p_u|| \quad (1)$$

$$z_{Voronoi}(x) \sim \mathcal{N}(0,\ 1) \text{for each channel otherwise} \quad (2)$$

where, $z(x)$ denotes the vector of channels for z at coordinates x. Note that in latent diffusion models, there is a straightforward linear correspondence between the coordinates in the latent space and those in the image ($I = G(z)$ has shape $(3, 512, 512)$ and z has shape $(4, 64, 64)$), so that any $x \in [0, 1] \times [0, 1]$ has a natural counterpart both in I_j (on which the user is clicking) and in z_j (the latent space).

λ images are generated for different values of the contraction factor r, evenly distributed in $(1, 2)$, and proposed to the user at the next iteration.

3.4 Post-treatments

Optionally, after the search has completed (Line 21 of Algorithm 1), the user can choose to apply a morphing with a second text prompt (Section 3.4.1) and/or a translation of the image with a chosen offset (Section 3.4.2).

Figure 4: Translation in the latent space: translating z in the latent space moves the subject in image $G(z)$ accordingly. Moving the tree from right to left.

Figure 5: For each image, the same latent variable z is used with a different prompt. All subjects are misplaced on the right of the image .

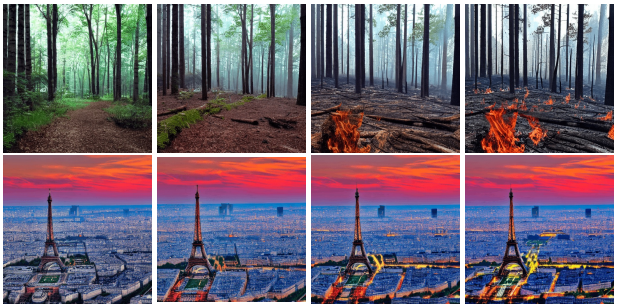

Figure 6: Image generated using linearly interpolated points between two prompts in the text embedding space. Top: *A peaceful forest* to *A forest burned to the ashes.* Bottom: *Paris at sunrise* to *Paris at sunset.*

3.4.1 Translating the latent space. It is a known effect that LDM sometimes poorly crops images. We observe that translating z is a simple solution (Figure 4). During translation, points translated outside the latent space come back from the other side (e.g., points exiting the right side comeback on the left side and vice-versa). This seems intuitive because the model relies on CNNs, which are equivariant. However, moving the latent space too much, typically putting the subject outside of the image, can produce drastic changes.

Also, less intuitively, a latent variable z that produces a poorly cropped image for a given prompt tends to produce a not well centred image for multiple other prompts, even if the prompt mentions a location on the image (Fig. 5). Translation is optionally performed at the end when the loop is over.

3.4.2 Interpolation in the text embedding space. Manipulating the latent space of the diffusion can be tedious. A small perturbation in this latent space can lead to big changes in the final image. This is why we try to manipulate the text embedding space to get smoother variations. By switching to this latent space, we lose the spatial information but gain in semantic abstraction and smoothness. Figure 6 shows this gain clearly by interpolating between two prompts: creating a path from $G(z, \text{text}_1)$ to $G(z, \text{text}_2)$ by considering the $G(z, \alpha embedding(\text{text}_1) + (1-\alpha) embedding(\text{text}_2))$ for $\alpha \in [0, 1]$ produces smoother animations than a path from $G(z_1, \text{text}_1)$ to $G(z_2, \text{text}_2)$ with $z_1 \neq z_2$: the dependency in z is too abrupt so that it is better to keep the same z and interpolate on the text side only. An example is presented in Fig. 6.

4 EXPERIMENTAL RESULTS

This section presents experimental results obtained with IELDM. We are aware of other interactive systems using diffusion like Midjourney (*midjourney.com*) or DALL-E [10] but none of them are open sourced nor provide any details of how they operate internally. Moreover, they both use their own proprietary generative model different from the one we are using, making the comparison even more complicated. Searching the latent space of generative models is not new, but [8] or [12, 19] use this for completely different tasks. Therefore, we try to show the benefits of our approach against Latent Diffusion Model alone. In Section 4.1, we validate the surrogate model. In Sections 4.2 to 4.4, we illustrate and validate the global mutation and its noise adaptation mechanism, the Voronoi crossover, and the local mutations; and lastly in Section 4.5, the overall tool.

4.1 Validate the surrogate approach: Simulated interactive experiments

Two different approaches are proposed to simulate interaction without any human being.

The first series of experiments relies on some random search using a fully hand-labelled dataset (Section 4.1.1): the algorithm uses these labels for generating new images, which are then labelled by humans for the sake of evaluation.

The second one (Section 4.1.2) is based on a measurable "user's goals" that can be easily computed from any image: these are artificial users goals, such as the dominant colour in the image: they can be computed automatically. So, we can entirely replace the human by the labelling and validate the global approach.

In both cases, the surrogate model is built using the *Scikit-learn* models *LogisticRegression* and *MLPClassifier*[1].

4.1.1 Surrogate model and random search in an apriori labelled dataset. This experiment assumes that the initial image dataset is fully labelled. Images are labelled *Good* or *Bad*. The corresponding points in latent space are labelled accordingly. These labels encode users' preferences in different use cases. The search then proceeds as follows:

- Split the labelled dataset in latent space in two parts: training and test.
- Learn the surrogate model on the training set.
- Randomly draw n images in the test set (we use different values of n, see results below), and retain the one with maximum probability of being Good according to the surrogate model.

[1] default arguments for *LogisticRegression* and *MLPClassifier* with arguments $solver = 'lbfgs'$, $alpha = 1e-5$, $hidden\_layer\_sizes = (5, 2)$, $random\_state = 1$

- Compare the surrogate model frequency of success (finding an image labelled as Good in the test set) against random selection in test set n.

Datasets include: whether the crop is good or bad, whether there is nature in the background, the colour of hair. This pseudo random search is the only way to proceed here because the interaction is predefined within a fixed dataset. If a real optimization algorithm was being used, it would, for sure, suggest points outside the fixed labelled dataset, that would be impossible to qualify. This procedure, even if brutal, nevertheless gives some indication of the goodness of the selected images (a different protocol is provided in the next section).

Supp. Tab. 3 and 4 present frequencies of satisfactory generations with moderate budgets, showing that some methods work even in the case of both a small sample for learning the surrogate model, and a random search. But these results only concern easy cases, when the proportion of success is away from zero. The main advantage of IELDM, however, is when successes are very rare, as this is when users have to make hundreds of trials with bare LDM before obtaining something acceptable. And this is where using feedback can really help to speed up such tedious trial-and-error naive approach.

4.1.2 Surrogate model and evolution with a computable interaction. This experiment aims at validating the surrogate approach using a real optimization procedure, in cases where the success with the bare LDM method is sparse and difficult to obtain. The interaction is simulated as follows:

- The label represents "$r > max(c \times g, c \times b)$". r, g and b represent the average red, green and blue channels. Additionally, some noise is added to these labels: g or b are used instead of r (all equally likely). We randomly draw the constant $c > 1$ so that we have different contexts of rarity: c greater means a rarer success.
- The prompt is randomly drawn in "Cyberpunk man", "Cyberpunk woman", "Giant spider and a building", "Cleopatra and a cucumber".
- At each iteration,
  - we build $\lambda = 15$ images (randomly for the first iteration, using random search on the surrogate model afterwards).
  - Positive images are detected.
  - All images are used for retraining the model.

Furthermore, there is no initialization of the optimization by crossover or selection, so that *only* the surrogate model is at work here because the goal here is to validate the surrogate model itself, not the crossover or mutation operators.

The optimization algorithm used here is Lengler as implemented in Nevergrad [3], with a budget of 30.

This experiment corresponds to Algorithm 1 in which everything else except the surrogate model has been disabled, which means that at each iteration:

- $\lambda = 15$ images are generated (randomly for the first iteration, Nevergrad using the surrogate model afterwards).
- The user clicks on image(s) (s)he likes (no click on locations inside the images is possible)
- The model is retrained from all images.

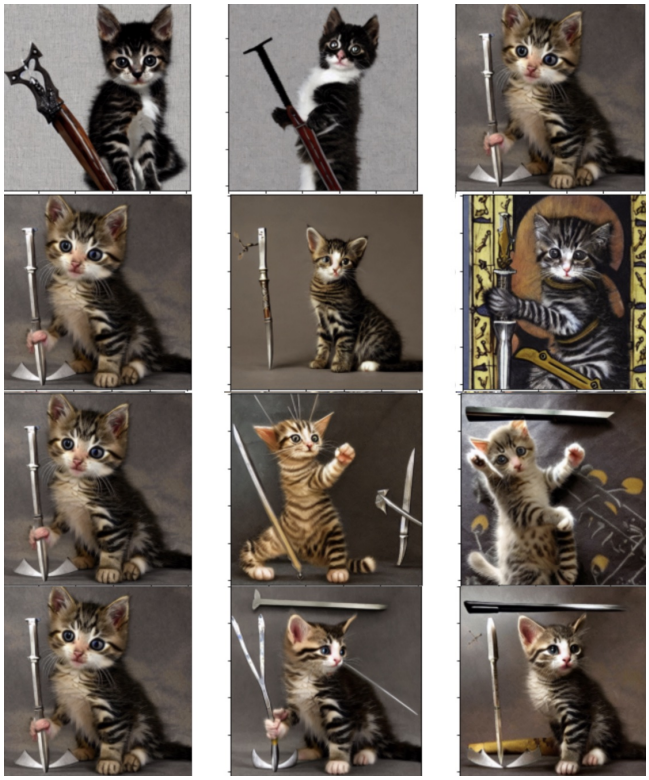

Figure 7: *A kitten with a medieval weapon.* Illustration of the mutation strength adaptation as in Section 4.2 for global mutations: effect of the 0.7 parameter in Algorithm 1. Each batch is a row of $\lambda = 3$ images. In the first batch (i.e.,first row), we click on the top right image, which is then the left one for batches (rows) 2, 3 and 4 because we always click on that same image. We observe as expected that (i) the first image of each batch is the selected one (ii) the rightmost image is more mutated than the middle one (iii) as we always click on the same single image, the mutation strength visibly decreases from a row to the next (multiplication of σ by 0.7): the first row has a lot of diversity whereas differences are more subtle in the last row. This is by design (see Algorithm 1): when the user accepts only one image, we want new images to be closer to that only selected image.

We then compare the frequency of Good images during the first iteration and after some given number of iterations. Results are presented in Supp. Tab. 2: when success is rare (low success rate at iteration 0), it increases a lot for later iterations after using the surrogate model, demonstrating its usefulness.

4.2 Decreasing mutation strength: fine-tune a good-enough image by small global mutations

When a single image is selected by the user, the mutation strength is decreased by an empirical factor (0.7, see Line 17 in Algorithm 1). Figure 7 shows the advantage of this strategy to fine-tune an image.

Problem	Tool	Success rate before user feedback	Success rate after user feedback	Figure or table
Fighting a dinosaur with cucumbers	Voronoi crossover	3/15 (2nd batch)	10/13	-
Character eating ice cream + candy	Voronoi crossover	0/15 (2nd batch)	7/13	-
Specified Dominant color (Green or red or blue)	Surrogate model, 2nd batch, log. reg.	v	v-33% to v+199% (median: +44%)	Supp. Tab. 2
	3rd batch, log. reg.	v	v -13% to v+179% (median: +20%)	
	4th batch, log. reg.	v	v -6% to v+259% (median: +36%)	
	5th batch, log. reg.	v	v +3% to v+212% (median: +80%)	
Specified Dominant color (Green or red or blue)	Surrogate model, 2nd batch, MLP	v	v -60% to v + 180% (median +13%)	Supp. Tab. 2
	3rd batch, MLP	v	v-70% to v+170% (median +25%)	
	4th batch, MLP	v	v-67% to v+167% (median +15%)	
	5th batch, MLP	v	v+5% to v+170% (median +22%)	
Medusa	Global mutation	1/14	6/14 for 2nd batch, 10/14 for 3rd batch	-

Table 1: Numerical validation of our methods. Results in Supp. Tab. 2 are based on simple criteria (dominant color) so that we managed to automatize the evaluation, others are based on humans clicks for building the set and on human evaluation for validating the results. These human evaluations can be checked by the reader in corresponding figures. Local mutations are not discussed here: the reader is referred to Fig. 3. In the case of a specified dominant color, the success rate v is variable by design (see Section 4.1.2 for the specifications and Supp. Tab. 2 for detailed results) and we present the improvement in the success rate.

The impact of the decrease in the mutation strength is clearly visible on the successive output images, that are less and less changed. The rationale behind this strategy (and the empirical choice of factor 0.7) is that when the user starts to like an image, the strength of the mutations should decrease fast enough to allow fine-tuning of that image, but not too fast, to avoid premature convergence.

4.3 Voronoi crossover: take the best of several images

Generating a ninja fighting a dinosaur with a cucumber is far from being easy with standard LDM. Typically, the dinosaurian nature of the dinosaur will override the ninja and make it a dinosaur, or the dinosaur will be a ninja, or the cucumber will be a dinosaur: Both are green, which adds to the confusion: the reader is encouraged to experiment "a panda behind a cow" with a bare LDM image generation to see how the mixing of the colours of the two animals can mess up the result. Figure 2 present other examples of application of the Voronoi crossover, by clicking on 5 points on a monster in one image and on 5 points on another monster in another image.

4.4 Local mutations: choose where to make changes

Results demonstrating the usefulness of local mutations (Section 3.1) are presented in Fig. 3. These results are slightly unstable: sometimes modifying a small part of the latent variable can lead to a larger than expected change in the output images. However, because of the variable noise used for the different offspring (see i/λ in Algorithm 2), at least a few mutations give satisfactory results.

4.5 Evolutionary Latent Diffusion Model in practice

We noted empirically that the crossover is mainly effective a bit close to the optimum, not during the few first iterations. Clicking on only one image (or even zero, leading to a restart from the latest clicked image, in some hard cases) is frequently a good idea during the first iterations. We also observed that it works well with as many chosen images as we want: the operator scales the randomness effectively with $r \in (1, 2)$ as in Section 3.3, so that we get diversity, no matter how many images we have. We can compare initial batch against last batch and count how many images fit the user goals (Table 1).

5 CONCLUSION

LDM is based on a loop (Figure 1). Evolutionary LDM adds one more loop, with the user providing feedback by clicking. Many image generation codes contain a multiple re-rolls mechanism. This basically means applying random search: we replace the randomly generated z by z iteratively improved by human clicks. We note that a choice of z by human selection exists in MidJourney and DALL-E (not open-sourced): to the best of our knowledge, our code is the only open-sourced one providing the possibility of local mutations and cross-overs and surrogate model. Thanks to these tools, IELDM is convenient for exploiting a partial success or for combining several partial successes: this reduces the need for hundreds of re-rolls for satisfying expectations. We improve LDM by various tools from evolutionary computation:

- **Machine learning in the latent space:** we applied machine learning regression, on the fly during the user session. Typically, illustrators use dozens or hundreds of generations before being satisfied: this means dozens or hundreds of examples, specific to the current use case. Sufficiently many models work in spite of that moderate number of examples (in our experiments, 15 examples at the first iteration for creating the second population, 30 examples for creating the third population, etc) and a rather large dimension ($4 \times 64 \times 64$). Regarding LDM difficulties, as discussed in the introduction, this reduces the need for many reruns (issue 3), as shown by results in Supp. Tab. 2.
- **Voronoi for spatially combining user decisions.** A newly proposed Voronoi crossover operator is applied at the user

request (when she specifies several images with possibly several locations on these images). Combining variables in the latent space works and leads to results as in Fig. 2. This mitigates the problem mentioned in the introduction as issue 1.

- **Generating global mutations of a given image.** We used an adaptive mutation rate, for reducing the mutation rate when the user wants to fine-tune a single image (see Fig. 7). Furthermore, our method generates several images with different mutation strengths, as it is difficult for the users to specify their expectations in terms of noise level. We include an option for doing global mutation of the average between several chosen images.
- **Local selection of mutations.** Users can specify where they wants the image to be modified (Figure 3). This mitigates the problem mentioned in the introduction as issue 2: we got compositions which are difficult to get by classical LDM.

Exploiting the locality of the latent variable z (changes in z impact the corresponding part of the image $G(z)$), we also proposed that the user uses the translation (Figure 4) or exports an animation (Figure 6).

All our variation operators (crossovers and mutations) generate several outputs, with a varying mutation rates: we typically display, first, images close to the user selection, and then images farther, until we reach the offspring population size of λ. Our results are validated as in Table 1.

Due to length constraints, some figures were moved to the supplementary material.

REFERENCES

[1] Martín Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. In *ICML*. 214–223.

[2] Richard Dawkins. 1986. *The blind watchmaker: Why the evidence of evolution reveals a universe without design.* W.W. Norton and Company.

[3] Benjamin Doerr, Carola Doerr, and Johannes Lengler. 2019. Self-Adjusting Mutation Rates with Provably Optimal Success Rules.

[4] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. 2017. Improved Training of Wasserstein GANs. *CoRR* abs/1704.00028 (2017).

[5] Naveen Kodali, Jacob D. Abernethy, James Hays, and Zsolt Kira. 2017. How to Train Your DRAGAN. *CoRR* abs/1705.07215 (2017).

[6] Evelyne Lutton, Emmanuel Cayla, and Jonathan Chapuis. 2003. ArtiE-Fract: The Artist's Viewpoint. In *EvoApps*. Springer-Verlag, 510–521.

[7] Xudong Mao, Qing Li, Haoran Xie, Raymond Y. K. Lau, Zhen Wang, and Stephen Paul Smolley. 2017. Least Squares Generative Adversarial Networks. *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), 2813–2821.

[8] Sachit Menon, Alexandru Damian, Shijia Hu, Nikhil Ravi, and Cynthia Rudin. 2020. PULSE: Self-Supervised Photo Upsampling via Latent Space Exploration of Generative Models. arXiv:2003.03808 [cs.CV]

[9] Yan Pei and Hideyuki Takagi. 2018. Research progress survey on interactive evolutionary computation. *Journal of Ambient Intelligence and Humanized Computing* (2018), 1–14.

[10] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with CLIP latents. *arXiv:2204.06125* (2022).

[11] Jeremy Rapin and Olivier Teytaud. 2018. Nevergrad - A gradient-free optimization platform. github.com/FacebookResearch/Nevergrad.

[12] Jacob Schrum, Jake Gutierrez, Vanessa Volz, Jialin Liu, Simon Lucas, and Sebastian Risi. 2020. Interactive evolution and exploration within latent level-design space of generative adversarial networks. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 148–156.

[13] Jimmy Secretan, Nicholas Beato, David B. D'Ambrosio, Adelein Rodriguez, Adam Campbell, Jeremiah T. Folsom-Kovarik, and Kenneth O. Stanley. 2011. Picbreeder: A Case Study in Collaborative Evolutionary Exploration of Design Space. *Evol. Comput.* 19, 3 (2011), 373–403.

[14] Dong-il Seo and Byung Ro Moon. 2002. Voronoi Quantizied Crossover For Traveling Salesman Problem. In *GECCO*, William B. Langdon et al. (Ed.). Morgan Kaufmann, 544–552.

[15] Jeong-Yeon Seo, Sang-Min Park, Seoung Soo Lee, and Deok-Soo Kim. 2005. Regrouping Service Sites: A Genetic Approach Using a Voronoi Diagram. In *ICCSA*, Gervasi, Osvaldo et al. (Ed.). 652–661.

[16] Christopher L. Simons, Ian C. Parmee, and Rhys Gwynllyw. 2010. Interactive, Evolutionary Search in Upstream Object-Oriented Class Design. *IEEE Trans. Software Eng.* 36, 6 (2010), 798–816.

[17] Hideyuki Takagi. 2001. Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. *Proc. IEEE* 89, 9 (2001), 1275–1296.

[18] Hideyuki Takagi. 2015. Interactive Evolutionary Computation for Analyzing Human Characteristics. In *Emergent Trends in Robotics and Intelligent Systems*, P. Sinčák, P. Hartono, M. Virčíková, J. Vaščák, and R. Jakša (Eds.). Springer, 189–195.

[19] Vanessa Volz, Jacob Schrum, Jialin Liu, Simon M. Lucas, Adam Smith, and Sebastian Risi. 2018. Evolving Mario Levels in the Latent Space of a Deep Convolutional Generative Adversarial Network. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Kyoto, Japan) *(GECCO '18)*. Association for Computing Machinery, New York, NY, USA, 221–228. https://doi.org/10.1145/3205455.3205517

[20] Jun Rong Yan and Yong Min. 2011. User Fatigue in Interactive Evolutionary Computation. In *Measuring Technology and Mechatronics Automation*, Vol. 48. 1333–1336.

6 SUPPLEMENTARY MATERIAL

6.1 Surrogate model validation tables

Batch 1 vs batch 2, MLP		
Batch 1	Batch 2	
6.66	9.33	+40.00%
6.66	2.66	−59.98%
6.66	13.33	+99.99%
6.66	7.99	+20.00%
6.66	18.66	+179.98%
6.66	6.66	+0.00%
6.66	11.99	+79.99%
13.33	11.99	−9.99%
19.99	14.66	−26.66%
19.99	18.66	−6.66%
19.99	18.66	−6.66%
19.99	27.99	+40.00%
19.99	31.99	+60.00%
26.66	21.33	−19.99%
26.66	18.66	−29.99%
26.66	31.99	+20.00%
46.66	58.66	+25.72%
53.32	61.33	+15.00%
59.99	46.66	−22.21%
59.99	62.66	+4.44%
59.99	67.99	+13.33%
66.66	79.99	+20.00%
73.32	71.99	−1.81%

Batch 1 vs batch 3, MLP		
Batch 1	Batch 3	
6.66	17.99	+169.99%
6.66	7.99	+20.00%
6.66	17.33	+159.99%
6.66	8.66	+30.00%
6.66	13.99	+109.99%
6.66	1.99	−69.98%
13.33	13.99	+5.00%
19.99	27.99	+40.00%
19.99	20.66	+3.33%
19.99	15.99	−19.99%
19.99	28.66	+43.33%
19.99	36.66	+83.33%
26.66	28.66	+7.50%
26.66	18.66	−29.99%
26.66	18.66	−29.99%
46.66	61.99	+32.86%
59.99	47.99	−19.99%
59.99	66.66	+11.11%
66.66	77.99	+17.00%
73.32	69.99	−4.53%

Batch 1 vs batch 4, MLP		
Batch 1	Batch 4	
6.66	2.22	−66.65%
6.66	6.22	−6.65%
6.66	17.77	+166.65%
6.66	10.22	+53.33%
19.99	31.55	+57.78%
19.99	23.99	+20.00%
19.99	30.66	+53.33%
19.99	37.33	+86.67%
19.99	15.55	−22.21%
26.66	29.33	+10.00%
26.66	21.77	−18.32%
46.66	67.99	+45.72%
59.99	68.44	+14.08%
59.99	50.22	−16.29%
66.66	76.44	+14.67%

Batch 1 vs batch 5, MLP		
Batch 1	Batch 5	
6.66	6.99	+5.00%
6.66	17.99	+169.99%
6.66	12.33	+84.99%
19.99	24.33	+21.67%
19.99	32.33	+61.67%
26.66	31.66	+18.75%
59.99	71.33	+18.89%

Batch 1 vs batch 2, Log Reg		
Batch 1	Batch 2	
6.66	11.99	+79.99%
6.66	14.66	+119.99%
6.66	5.33	−19.99%
13.33	31.99	+140.00%
19.99	13.33	−33.32%
19.99	41.33	+106.67%
19.99	29.33	+46.67%
26.66	37.33	+40.00%
33.33	54.66	+64.00%
73.32	87.99	+20.00%
73.32	69.33	−5.44%
79.99	82.66	+3.33%

Batch 1 vs batch 3, Log Reg		
Batch 1	Batch 3	
6.66	7.33	+10.00%
6.66	18.66	+179.98%
13.33	39.33	+195.00%
19.99	17.33	−13.32%
19.99	42.66	+113.34%
19.99	27.99	+40.00%
26.66	37.33	+40.00%
33.33	60.66	+82.00%
73.32	87.99	+20.00%
73.32	71.99	−1.81%
79.99	82.66	+3.33%

Batch 1 vs batch 4, Log Reg		
Batch 1	Batch 4	
6.66	23.99	+259.98%
6.66	7.11	+6.67%
13.33	40.44	+203.33%
19.99	18.66	−6.66%
19.99	42.22	+111.11%
19.99	27.11	+35.56%
26.66	38.22	+43.34%
33.33	61.33	+84.00%
73.32	88.44	+20.61%
73.32	74.66	+1.82%
79.99	81.33	+1.67%

Batch 1 vs batch 5, Log Reg		
Batch 1	Batch 5	
13.33	41.66	+212.50%
19.99	20.33	+1.67%
19.99	43.33	+116.67%
33.33	59.99	+80.00%
79.99	82.66	+3.33%

Table 2: Using the MLP as a surrogate model (2 left tables), and Logistic Regression (right table), and evolution for increasing the frequency of correct generations as explained in Sec. 4.1.1. Experiments built as explained in Section 4.1. We consider various numbers of batches. Col. 1 (resp. col 2) in each table refer to frequencies $\times 100$ at iteration 0 (resp. over all other iterations up to the considered number of iterations). The third column shows how much the success rate increases between iteration 0 (vanilla LDM) and other iterations (due to the influence of the surrogate model): if col. 3 is positive, then we do better than vanilla LDM. Most of the time, col. 3 is > 0, validating the method. For example, the last sub-table, first row, indicates a run with Logistic Regression: we do 20 iterations, iterations 1 to 19 have on average +80% more good examples than iteration 0. Red: cases in which the approach was detrimental.We note that results are more impressive when success is rare in the original data (i.e. the third col is greater when the first col is lower).

Problem	Train set	Tool	Perf
b25885619	**150**	**LogisticRe-2**	**0.63**
b25885619	**150**	**LogisticRe-3**	**0.57**
b25885619	**150**	**MLPClassif-2**	**0.46**
b25885619	**150**	**MLPClassif-3**	**0.51**
b25885619	1	Vanilla LDM	0.41
b25885619	**20**	**LogisticRe-2**	**0.44**
b25885619	**20**	**LogisticRe-3**	**0.50**
b25885619	**20**	**MLPClassif-2**	**0.47**
b25885619	**20**	**MLPClassif-3**	**0.42**
b25885619		Vanilla LDM	0.41
b25885619	**30**	**LogisticRe-2**	**0.48**
b25885619	**30**	**LogisticRe-3**	**0.47**
b25885619	**30**	**MLPClassif-2**	**0.46**
b25885619	**30**	**MLPClassif-3**	**0.48**
b25885619		Vanilla LDM	0.41
b25885619	**40**	**LogisticRe-2**	**0.59**
b25885619	**40**	**LogisticRe-3**	**0.48**
b25885619	**40**	**MLPClassif-2**	**0.46**
b25885619	40	MLPClassif-3	0.40
b25885619		Vanilla LDM	0.41

b620683104	**150**	**LogisticRe-2**	**0.53**
b620683104	**150**	**LogisticRe-3**	**0.53**
b620683104	**150**	**MLPClassif-2**	**0.47**
b620683104	**150**	**MLPClassif-3**	**0.55**
b620683104	1	Vanilla LDM	0.36
b620683104	**20**	**LogisticRe-2**	**0.56**
b620683104	**20**	**LogisticRe-3**	**0.47**
b620683104	**20**	**MLPClassif-2**	**0.43**
b620683104	**20**	**MLPClassif-3**	**0.39**
b620683104		Vanilla LDM	0.36
b620683104	**30**	**LogisticRe-2**	**0.46**
b620683104	**30**	**LogisticRe-3**	**0.38**
b620683104	**30**	**MLPClassif-2**	**0.53**
b620683104	**30**	**MLPClassif-3**	**0.43**
b620683104		Vanilla LDM	0.36
b620683104	**40**	**LogisticRe-2**	**0.55**
b620683104	**40**	**LogisticRe-3**	**0.41**
b620683104	**40**	**MLPClassif-2**	**0.52**
b620683104	**40**	**MLPClassif-3**	**0.45**
b620683104		Vanilla LDM	0.36

b65681463	**150**	**LogisticRe-2**	**0.54**
b65681463	**150**	**LogisticRe-3**	**0.52**
b65681463	**150**	**MLPClassif-2**	**0.58**
b65681463	**150**	**MLPClassif-3**	**0.53**
b65681463	1	Vanilla LDM	0.46
b65681463	**20**	**LogisticRe-2**	**0.53**
b65681463	**20**	**LogisticRe-3**	**0.54**
b65681463	**20**	**MLPClassif-2**	**0.5**
b65681463	20	MLPClassif-3	0.46
b65681463		Vanilla LDM	0.46
b65681463	**30**	**LogisticRe-2**	**0.61**
b65681463	**30**	**LogisticRe-3**	**0.46**
b65681463	**30**	**MLPClassif-2**	**0.53**
b65681463	**30**	**MLPClassif-3**	**0.48**
b65681463		Vanilla LDM	0.46
b65681463	**40**	**LogisticRe-2**	**0.57**
b65681463	**40**	**LogisticRe-3**	**0.55**
b65681463	**40**	**MLPClassif-2**	**0.5**
b65681463	**40**	**MLPClassif-3**	**0.48**
b65681463		Vanilla LDM	0.46

b86767819	**20**	**LogisticRe-2**	**0.57**
b86767819	**20**	**LogisticRe-3**	**0.45**
b86767819	**20**	**MLPClassif-2**	**0.53**
b86767819	**20**	**MLPClassif-3**	**0.49**
b86767819		Vanilla LDM	0.36
b86767819	**30**	**LogisticRe-2**	**0.56**
b86767819	**30**	**LogisticRe-3**	**0.45**
b86767819	**30**	**MLPClassif-2**	**0.51**
b86767819	**30**	**MLPClassif-3**	**0.43**
b86767819		Vanilla LDM	0.36
b86767819	**40**	**LogisticRe-2**	**0.53**
b86767819	**40**	**LogisticRe-3**	**0.49**
b86767819	**40**	**MLPClassif-2**	**0.52**
b86767819	**40**	**MLPClassif-3**	**0.41**
b86767819		Vanilla LDM	0.36

bluedyed2101	**20**	**LogisticRe-2**	**0.59**
bluedyed2101	**20**	**LogisticRe-3**	**0.41**
bluedyed2101	**20**	**MLPClassif-2**	**0.39**
bluedyed2101	**20**	**MLPClassif-3**	**0.41**
bluedyed2101		Vanilla LDM	0.28
bluedyed2101	**30**	**LogisticRe-2**	**0.37**
bluedyed2101	**30**	**LogisticRe-3**	**0.39**
bluedyed2101	**30**	**MLPClassif-2**	**0.41**
bluedyed2101	30	MLPClassif-3	0.27
bluedyed2101		Vanilla LDM	0.28
bluedyed2101	**40**	**LogisticRe-2**	**0.50**
bluedyed2101	**40**	**LogisticRe-3**	**0.42**
bluedyed2101	**40**	**MLPClassif-2**	**0.36**
bluedyed2101	**40**	**MLPClassif-3**	**0.36**
bluedyed2101		Vanilla LDM	0.28

cropcp2236	**20**	**LogisticRe-2**	**0.62**
cropcp2236	**20**	**LogisticRe-3**	**0.71**
cropcp2236	**20**	**MLPClassif-2**	**0.6**
cropcp2236	20	MLPClassif-3	0.59
cropcp2236		Vanilla LDM	0.64
cropcp2236	30	LogisticRe-2	0.53
cropcp2236	30	LogisticRe-3	0.57
cropcp2236	30	MLPClassif-2	0.56
cropcp2236	30	MLPClassif-3	0.51
cropcp2236		Vanilla LDM	0.64
cropcp2236	**40**	**LogisticRe-2**	**0.65**
cropcp2236	**40**	**LogisticRe-3**	**0.67**
cropcp2236	40	MLPClassif-2	0.54
cropcp2236	40	MLPClassif-3	0.62
cropcp2236		Vanilla LDM	0.64

dark2145	20	LogisticRe-2	0.50
dark2145	**20**	**LogisticRe-3**	**0.53**
dark2145	**20**	**MLPClassif-2**	**0.54**
dark2145	**20**	**MLPClassif-3**	**0.56**
dark2145		Vanilla LDM	0.52
dark2145	30	LogisticRe-2	0.38
dark2145	30	LogisticRe-3	0.50
dark2145	30	MLPClassif-2	0.44
dark2145	**30**	**MLPClassif-3**	**0.53**
dark2145		Vanilla LDM	0.52
dark2145	40	LogisticRe-2	0.46
dark2145	40	LogisticRe-3	0.51
dark2145	40	MLPClassif-2	0.51
dark2145	**40**	**MLPClassif-3**	**0.56**
dark2145		Vanilla LDM	0.52

dyed1596	**20**	**LogisticRe-2**	**0.62**
dyed1596	**20**	**LogisticRe-3**	**0.65**
dyed1596	20	MLPClassif-2	0.46
dyed1596	**20**	**MLPClassif-3**	**0.62**
dyed1596		Vanilla LDM	0.58
dyed1596	**30**	**LogisticRe-2**	**0.60**
dyed1596	**30**	**LogisticRe-3**	**0.63**
dyed1596	30	MLPClassif-2	0.50
dyed1596	30	MLPClassif-3	0.53
dyed1596		Vanilla LDM	0.58
dyed1596	**40**	**LogisticRe-2**	**0.62**
dyed1596	**40**	**LogisticRe-3**	**0.65**
dyed1596	**40**	**MLPClassif-2**	0.51
dyed1596	40	MLPClassif-3	0.56
dyed1596		Vanilla LDM	0.58

Table 3: Validation of the surrogate model as presented in Section 2.2. Validation is based on Section 4.1: given a train-set extracted from a labelled dataset (Col. 1, label = dataset generator label + seed), of a given size (second col), we learn a model (third col) with a given number of random searches (third col. suffix: 2 or 3) and get a frequency of good generations as in Col. 4. *Vanilla LDM* is the baseline in which we just run the Vanilla LDM. Example: On the benchmark "dyed", labelled by humans, seed 1596, Vanilla LDM gets 58% of success rate whereas Applying LogisticRegression trained on 40 points for choosing among two randomly drawn z leads to 62%. In gray the worst result: frequently the *Vanilla LDM* baseline. Results are positive (i.e. most results are bold, i.e. better than Vanilla LDM). Over both Supp. Tab. 3 and 4, the percentage of improved satisfactory generations is: LogisticRe.2: 85%. LogisticRe.3: 81%. MLPClf.2: 78%. MLPClf.3: 61%. Compared to results in difficult contexts as in Table 2 (where +200% is common when Vanilla LDM is below 10%), the gap is however small: our method becomes more and more useful as the context becomes harder.

dyedred2854	20	LogisticRe-2	0.51
dyedred2854	20	LogisticRe-3	0.48
dyedred2854	20	MLPClassif-2	0.51
dyedred2854	20	MLPClassif-3	0.52
dyedred2854		Vanilla LDM	0.56
dyedred2854	30	LogisticRe-2	0.53
dyedred2854	30	LogisticRe-3	0.54
dyedred2854	30	MLPClassif-2	0.44
dyedred2854	30	MLPClassif-3	0.44
dyedred2854		Vanilla LDM	0.56

g531921178	**150**	**LogisticRe-2**	**0.55**
g531921178	**150**	**LogisticRe-3**	**0.46**
g531921178	**150**	**MLPClassif-2**	**0.49**
g531921178	150	MLPClassif-3	0.32
g531921178	1	Vanilla LDM	0.33
g531921178	**20**	**LogisticRe-2**	**0.53**
g531921178	**20**	**LogisticRe-3**	**0.40**
g531921178	**20**	**MLPClassif-2**	**0.49**
g531921178	**20**	**MLPClassif-3**	**0.46**
g531921178		Vanilla LDM	0.33
g531921178	**30**	**LogisticRe-2**	**0.58**
g531921178	**30**	**LogisticRe-3**	**0.38**
g531921178	**30**	**MLPClassif-2**	**0.51**
g531921178	**30**	**MLPClassif-3**	**0.38**
g531921178		Vanilla LDM	0.33
g531921178	**40**	**LogisticRe-2**	**0.52**
g531921178	**40**	**LogisticRe-3**	**0.33**
g531921178	**40**	**MLPClassif-2**	**0.44**
g531921178	**40**	**MLPClassif-3**	**0.40**
g531921178		Vanilla LDM	0.33

goodcrop1899	**20**	**LogisticRe-2**	**0.55**
goodcrop1899	20	LogisticRe-3	0.52
goodcrop1899	**20**	**MLPClassif-2**	**0.58**
goodcrop1899	20	MLPClassif-3	0.54
goodcrop1899		Vanilla LDM	0.54
goodcrop1899	**30**	**LogisticRe-2**	**0.58**
goodcrop1899	**30**	**LogisticRe-3**	**0.54**
goodcrop1899	30	MLPClassif-2	0.39
goodcrop1899	30	MLPClassif-3	0.54
goodcrop1899		Vanilla LDM	0.54
goodcrop1899	40	LogisticRe-2	0.54
goodcrop1899	40	LogisticRe-3	0.50
goodcrop1899	40	MLPClassif-2	0.45
goodcrop1899	40	MLPClassif-3	0.53
goodcrop1899		Vanilla LDM	0.54

nature1906	**20**	**LogisticRe-2**	**0.61**
nature1906	**20**	**LogisticRe-3**	**0.51**
nature1906	**20**	**MLPClassif-2**	**0.56**
nature1906	**20**	**MLPClassif-3**	**0.62**
nature1906		Vanilla LDM	0.50
nature1906	**30**	**LogisticRe-2**	**0.62**
nature1906	**30**	**LogisticRe-3**	**0.55**
nature1906	**30**	**MLPClassif-2**	**0.56**
nature1906	30	MLPClassif-3	0.46
nature1906		Vanilla LDM	0.50
nature1906	**40**	**LogisticRe-2**	**0.57**
nature1906	**40**	**LogisticRe-3**	**0.60**
nature1906	**40**	**MLPClassif-2**	**0.57**
nature1906	**40**	**MLPClassif-3**	**0.55**
nature1906		Vanilla LDM	0.50

nature591	**20**	**LogisticRe-2**	**0.51**
nature591	20	LogisticRe-3	0.43
nature591	20	MLPClassif-2	0.40
nature591	20	MLPClassif-3	0.41
nature591		Vanilla LDM	0.44
nature591	**30**	**LogisticRe-2**	**0.57**
nature591	**30**	**LogisticRe-3**	**0.46**
nature591	**30**	**MLPClassif-2**	**0.53**
nature591	30	MLPClassif-3	0.44
nature591		Vanilla LDM	0.44
nature591	**40**	**LogisticRe-2**	**0.47**
nature591	**40**	**LogisticRe-3**	**0.47**
nature591	**40**	**MLPClassif-2**	**0.51**
nature591	40	MLPClassif-3	0.41
nature591		Vanilla LDM	0.44

r9320995	**150**	**LogisticRe-2**	**0.64**
r9320995	**150**	**LogisticRe-3**	**0.64**
r9320995	**150**	**MLPClassif-2**	**0.55**
r9320995	150	MLPClassif-3	0.48
r9320995	1	Vanilla LDM	0.48
r9320995	**20**	**LogisticRe-2**	**0.49**
r9320995	**20**	**LogisticRe-3**	**0.54**
r9320995	**20**	**MLPClassif-2**	**0.51**
r9320995	**20**	**MLPClassif-3**	**0.53**
r9320995		Vanilla LDM	0.48
r9320995	**30**	**LogisticRe-2**	**0.57**
r9320995	**30**	**LogisticRe-3**	**0.58**
r9320995	**30**	**MLPClassif-2**	**0.59**
r9320995	**30**	**MLPClassif-3**	**0.53**
r9320995		Vanilla LDM	0.48
r9320995	**40**	**LogisticRe-2**	**0.53**
r9320995	**40**	**LogisticRe-3**	**0.60**
r9320995	**40**	**MLPClassif-2**	**0.62**
r9320995	40	MLPClassif-3	0.46
r9320995		Vanilla LDM	0.48

reddyed2171	**20**	**LogisticRe-2**	**0.53**
reddyed2171	**20**	**LogisticRe-3**	**0.64**
reddyed2171	20	MLPClassif-2	0.49
reddyed2171	20	MLPClassif-3	0.50
reddyed2171		Vanilla LDM	0.52
reddyed2171	**30**	**LogisticRe-2**	**0.62**
reddyed2171	**30**	**LogisticRe-3**	**0.62**
reddyed2171	**30**	**MLPClassif-2**	**0.60**
reddyed2171	**30**	**MLPClassif-3**	**0.60**
reddyed2171		Vanilla LDM	0.52
reddyed2171	**40**	**LogisticRe-2**	**0.72**
reddyed2171	**40**	**LogisticRe-3**	**0.69**
reddyed2171	**40**	**MLPClassif-2**	**0.61**
reddyed2171	**40**	**MLPClassif-3**	**0.59**
reddyed2171		Vanilla LDM	0.52

red374	**20**	**LogisticRe-2**	**0.45**
red374	**20**	**LogisticRe-3**	**0.43**
red374	**20**	**MLPClassif-2**	**0.56**
red374	**20**	**MLPClassif-3**	**0.37**
red374		Vanilla LDM	0.29
red374	**30**	**LogisticRe-2**	**0.4**
red374	**30**	**LogisticRe-3**	**0.41**
red374	**30**	**MLPClassif-2**	**0.47**
red374	**30**	**MLPClassif-3**	**0.38**
red374		Vanilla LDM	0.29
red374	**40**	**LogisticRe-2**	**0.6**
red374	**40**	**LogisticRe-3**	**0.35**
red374	**40**	**MLPClassif-2**	**0.45**
red374	**40**	**MLPClassif-3**	**0.29**
red374		Vanilla LDM	0.29

white1194	**20**	**LogisticRe-2**	**0.50**
white1194	20	LogisticRe-3	0.28
white1194	**20**	**MLPClassif-2**	**0.47**
white1194	**20**	**MLPClassif-3**	**0.43**
white1194		Vanilla LDM	0.28
white1194	30	LogisticRe-2	0.27
white1194	30	LogisticRe-3	0.26
white1194	**30**	**MLPClassif-2**	**0.55**
white1194	**30**	**MLPClassif-3**	**0.34**
white1194		Vanilla LDM	0.28

Table 4: Additional tables, as in Supp. Tab. 3. As previously, the *Vanilla LDM* baseline is frequently the worst: this shows that our method outperforms the vanilla LDM. See caption of Supp. Tab. 3 for statistical considerations.