The *Tree of Diffusion Life*: Evolutionary Embeddings to Understand the Generation Process of Diffusion Models

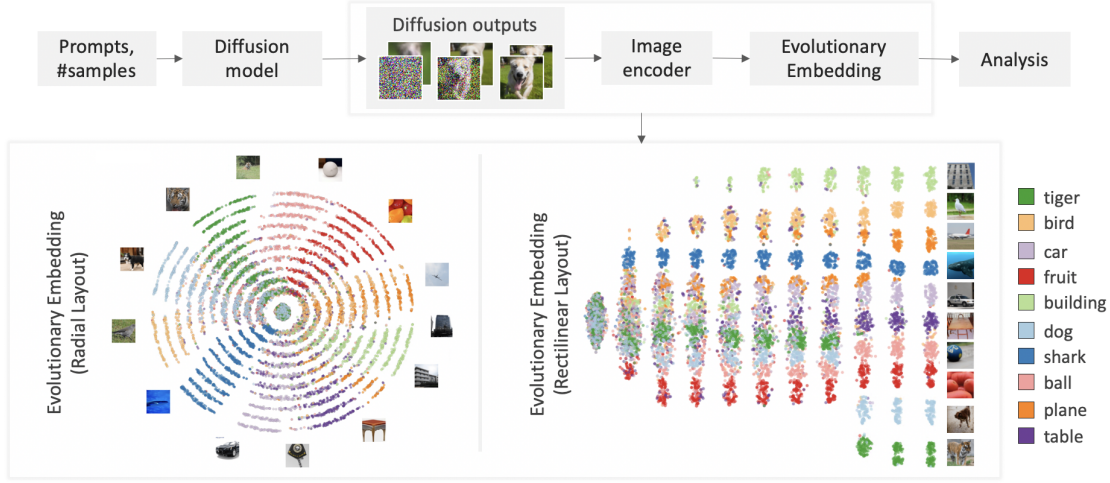Vidya Prasad, Hans van Gorp, Christina Humer, Anna Vilanova, and Nicola Pezzotti

Fig. 1: Illustration of the *Tree of Diffusion Life* (TDL) method for understanding data evolution in diffusion models. TDL samples the generative space of GLIDE [24] using $n = 1000$ instances across 10 ImageNet classes as prompts. Noisy instance images ($\mathbf{h}^i_{t-1}$) are extracted every 10th step (t in $[0, 100]$) and are projected to an intermediate space ($\widehat{\mathbf{h}}^i_{t-1}$) via an ImageNet classifier [1] for semantically meaningful distances. Noisy samples $\widehat{\mathbf{h}}^i_{t-1}$ are visualized using a novel evolutionary embedding method. The inner radial ring in the radial layout (left) and the left vertical line in the rectilinear one (right) denote the initial generation steps, progressing outward or towards the right to the final steps. Clusters are initially unclear and gradually emerge toward the end; for example, "planes" are initially confused with "birds" or "cars" based on whether they are flying or on the ground, becoming evident only towards the last few steps.

Abstract— Diffusion models generate high-quality samples by corrupting data with Gaussian noise and iteratively reconstructing it with deep learning, slowly transforming noisy images into refined outputs. Understanding this data evolution is important for interpretability but is complex due to its high-dimensional evolutionary nature. While traditional dimensionality reduction methods like t-distributed stochastic neighborhood embedding (t-SNE) aid in understanding high-dimensional spaces, they neglect evolutionary structure preservation. Hence, we propose *Tree of Diffusion Life* (TDL), a method to understand data evolution in the generative process of diffusion models. TDL samples a diffusion model's generative space via instances with varying prompts and employs image encoders to extract semantic meaning from these samples, projecting them to an intermediate space. It employs a novel evolutionary embedding algorithm that explicitly encodes the iterations while preserving the high-dimensional relations, facilitating the visualization of data evolution. This embedding leverages three metrics: a *standard t-SNE loss* to group semantically similar elements, a *displacement loss* to group elements from the same iteration step, and an *instance alignment* loss to align elements of the same instance across iterations. We present rectilinear and radial layouts to represent iterations, enabling comprehensive exploration. We assess various feature extractors and highlight TDL's potential with prominent diffusion models like GLIDE and Stable Diffusion with different prompt sets. TDL simplifies understanding data evolution within diffusion models, offering valuable insights into their functioning.

Index Terms—Explainable AI, diffusion models, dimensionality reduction, evolutionary embedding, high dimensional, visual analytics

✦

1 INTRODUCTION

Diffusion models have gained popularity across various applications [50], like generation, inverse problems, segmentation, and anomaly detection. They effectively generate diverse and high-quality samples by progressively corrupting data, usually with Gaussian noise, and subsequently, reconstruct the corrupted data via a trainable iterative

process using deep learning (DL). At each step, the same DL model, such as Unet [37], takes an intermediate noisy image, iteration number, and potentially a prompt as input and generates a less noisy image. This iterative evolution goes from an initial state of pure noise to a coherent image (see Fig. 2). Understanding this evolution of data is relevant for comprehending distribution evolution and what the model has learned [26]. It sheds light on the feature evolution and the decision-making dynamics of dataset modes. Such knowledge is relevant for refining model architectures and training strategies [6, 14, 51], controlling the generation process [26, 49], and improving overall model performance [33, 39]. This analysis could also help identify the biases and attribute entanglements relevant for robust model development [51]. However, this evolutionary process's iterative and high-dimensional nature presents challenges in comprehending the underlying dynamics.

Prior works have explored understanding diffusion models. At-

- *V. Prasad, H. van Gorp, A. Vilanova, and N. Pezzotti are with Eindhoven University of Technology. E-mail: {v.prasad, h.v.gorp, a.vilanova, n.pezzotti}@tue.nl*
- *C. Humer is with Johannes Kepler Universität Linz. E-mail: christina.humer@jku.at*
- *H. van Gorp and N. Pezzotti are with Philips. E-mail: {hans.van.gorp, nicola.pezzotti}@philips.com*

THIS IS A PREPRINT.

tribution maps explored concept prioritization of an instance over iterations [25]. Methods to understand the generation process within diffusion models, including comparisons of a few instance evolutions, have also been developed [17]. While some works delve deeper into the underlying image space in the generative process [26], the focus has been to extract a local latent structure around a sample. Insights focused on the granularity of features rather than the actual features evolving, which is our goal. Dimensionality reduction techniques like t-distributed stochastic neighborhood embedding (t-SNE) or uniform manifold approximation and projection (UMAP) [21] can visualize such an evolution over a few instances [17]. However, extending them to visualize the evolution of the entire dataset presents challenges. They fail to preserve the iterative structure crucial for understanding the diffusion process depicted in Fig. 2. Further, the stochastic positioning of clusters in these methods complicates tracing this data evolution, hindering analysis. Preserving the iterative structure is essential for studying data evolution and branching patterns across the entire dataset.

Therefore, we present the *Tree of Diffusion Life* (*TDL*), a method to support the understanding of the data evolution within the generative process of diffusion models (see Fig. 1). To address the issue of holistic understanding of the generative space, *TDL* samples this space via several instances with different prompts. Semantic meaning is extracted from this sampled evolutionary data using feature extractors or image encoders, projecting them to an intermediate space. We propose a novel evolutionary embedding algorithm to enable the understanding of all sampled data at scale while preserving the iterative context; hence facilitating the visualization of data evolution. The proposed embedding enables this via three loss metrics to - 1) cluster semantically similar elements (with the t-SNE loss), 2) place elements of the same iteration together, and 3) align elements of an instance across iterations. Rectilinear and radial layouts that explicitly represent these iterations are introduced, allowing a comprehensive exploration of data evolution. We investigate the strengths and limitations of image encoders that extract semantic meaning from the sampled evolutionary data within the proposed method. To demonstrate *TDL*'s practical relevance and versatility, we apply it to two prominent diffusion models, GLIDE [24] and StableDiffusion [36], utilizing different prompt sets for each model. By exploring how data and their specific attributes evolve, we gain valuable insights into the generation process of diffusion models. In summary, our contributions are as follows:

- A method, *TDL*, to holistically understand data evolution within diffusion models. It includes sampling the generative space, semantically encoding these samples, and a novel means to visualize the semantic evolution of these samples. The code is available at https://github.com/vidyaprsd/treeofdiffusion.

- A novel evolutionary embedding algorithm with a rectilinear or radial layout illustrating the evolution of instances.

- *TDL*'s evaluation through three cases with diverse prompt sets and diffusion models, exploring its applicability in different scenarios.

2 BACKGROUND

Diffusion models: Diffusion models [12, 23, 40] aim to approximate the true distribution $u(\mathbf{h}_0)$ of training data $\mathbf{h}_0$. For this, training data is gradually degraded, with Gaussian noise, resulting in a sequence of spaces $\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_T$ per iteration. The amount of noise added varies across iterations t and is a hyperparameter known as a noise scheduler. This noising process, i.e., moving from the original data distribution $u(\mathbf{h}_0)$ to noise $u(\mathbf{h}_T)$, is called the forward process (see Fig. 2).

For generation, a model v_θ, often a Unet [37], is trained to estimate the noise needed to go from a noisy image instance $\mathbf{h}_t^i$ to the final denoised image $\mathbf{h}_0^i$. We refer to this smooth estimate of $\mathbf{h}_0^i$ at iteration t as $\mathbf{h}_{t_0}^i$. A known level of noise based on the noise scheduler is added back to $\mathbf{h}_{t_0}^i$ to generate $\mathbf{h}_{t-1}^i$. The iterative process of estimating $\mathbf{h}_{t-1}^i$ from $\mathbf{h}_t^i$ continues until $t = 0$ and is known as the reverse generative process (see Fig. 2). Further details can be found in the original papers [12, 23]. Although there have been advancements, for example, skipping iteration steps to expedite sampling [52], the fundamental
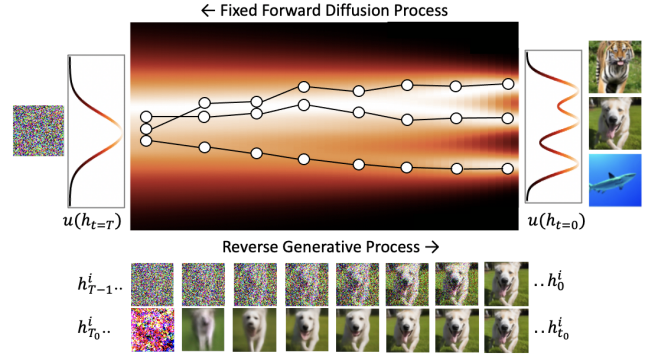

Fig. 2: Theoretical representation of the diffusion process [41, 44], where the training data distribution $u(\mathbf{h}_0^i)$ is represented as a 1D Gaussian. During generation (left to right), an instance of the "dog" evolves from noise $\mathbf{h}_T^i$ to a coherent image $\mathbf{h}_0^i$, or a fuzzy estimation $\mathbf{h}_{t_0}^i$ of the final image at t to a clear one.

methodology of iterative reconstruction, where a less noisy image is predicted at each step, remains the same. For our analysis, we extract the noisy image $\mathbf{h}_t^i$ and the estimation $\mathbf{h}_{t_0}^i$ of the final denoised image at each iteration t across instances i.

t-SNE: t-SNE [45] is a non-linear dimensionality reduction method widely used for visualizing high-dimensional data. It aims to preserve the neighborhoods between the high-dimensional data points in a low-dimensional output space. For this, t-SNE translates distances between the high-dimensional points as a symmetric joint probability distribution P. Similarly, a joint probability distribution Q describes the low-dimensional similarity. The goal is to achieve a mapping or an embedding in the low dimensional space Q that faithfully represents P. This is achieved by optimizing the positions of the low-dimensional points via a cost C, which is the KL divergence between P and Q,

$$C = KL(P||Q) = \sum_i^n \sum_{j, j \neq i}^n p^{ij} \ln \frac{p^{ij}}{q^{ij}} \quad (1)$$

p^{ij} is the similarity between two high-dimensional points $\mathbf{h}^i$ and $\mathbf{h}^j$ and is based on their conditional probabilities $p^{i|j}$ and $p^{j|i}$, shown in Eq. (2). $p^{j|i}$ is the likelihood that $\mathbf{h}^j$ is selected as the neighbor of $\mathbf{h}^i$ based on a Gaussian probability density function centered at $\mathbf{h}^i$ with variance σ_i. σ_i is defined based on the local density of $\mathbf{h}^i$ in the high-dimensional space and a defined perplexity parameter.

$$p^{ij} = \frac{p^{i|j} + p^{j|i}}{2N}, where, p^{j|i} = \frac{exp(-||\mathbf{h}^i - \mathbf{h}^j||^2 / (2\sigma_i^2))}{\sum_{k \neq i} exp(-||\mathbf{h}^i - \mathbf{h}^k||^2 / (2\sigma_i^2))} \quad (2)$$

For the low-dimensional space Q, whose point positions need to be optimized, a student t-distribution with one degree of freedom is used to compute the joint probability distribution. The similarity q^{ij} between two low-dimensional points $\mathbf{l}^i$ and $\mathbf{l}^j$ is then given by,

$$q^{ij} = \frac{(1 + ||\mathbf{l}^i - \mathbf{l}^j||^2)^{-1}}{\sum_{k \neq m} (1 + ||\mathbf{l}^k - \mathbf{l}^m||^2)^{-1}} \quad (3)$$

Our paper employs the vanilla t-SNE [45] to identify the similarity between high-dimensional images $\mathbf{h}_t^i$ from the diffusion model. We augment t-SNEs' KL divergence cost function by introducing components to group images within an iteration t and align images of an instance i across iterations. While we use the basic t-SNE for simplicity of implementation, advanced versions [31, 32] are compatible.

3 RELATED WORK

Interpreting Diffusion Models: Previous works have extensively explored the capabilities of diffusion models [50], but understanding the

inner workings of these models is important for leveraging their full potential and inspiring further research [4]. This inherent complexity of this high-dimensional evolutionary process poses challenges for exploration [46] and understanding [17], even for experts in the field.

Concerning the generation process, literature has focused on exploring the purpose of iterations, revealing that different stages in the denoising steps correspond to varying granularity of visual concepts [6, 25, 33, 39]. However, it is crucial to understand the evolution of features or dataset modes to understand the root causes of failures in these models and enable downstream development [19, 25]. On one hand, the specific feature details of each iteration are an open question. For this purpose, pixel-level attribution maps have been used to understand which parts of the generated image the prompt keywords influence the most. Tang et al. [42] do so via cross-attention keyword–pixel scores in the core denoising network in diffusion models. Such attention maps have been used for controlling output images [5, 10, 27, 43] and improving sample quality. Saliency map methods like GradCAM [38] were extended to diffusion models to study the model's focus in each iteration and how that evolves [25]. However, these existing approaches primarily concentrate on individual samples, creating a gap in understanding the global evolutionary process, which is our focus.

Prior works have also gone beyond a single sample and explored the local latent structures of the evolutionary diffusion process around an instance [16, 26]. Park et al. [26] employed Riemannian geometry, a tool for analyzing the intrinsic geometry of smooth manifolds, particularly relevant in understanding the structure of latent spaces in diffusion models. While their focus was primarily on applying this framework for image editing, they also shed light on the granularity of features evolving over iterations rather than the features themselves and the decision-making dynamics. This distinction highlights the unique focus with *TDL* on the holistic view of the evolutionary space and decision-making dynamics within diffusion models. Most of these works discussed so far are learning-based methods that focus on applying insights to create improved models rather than generic means of visualizing this evolutionary process to support understanding.

While visual analytics (VA) has provided a more generic means to analyze and understand complex DL models, most works focus, for example, on convolutional neural networks [30, 48], generative adversarial networks [13], which do not contain the iterative complexities of diffusion models. Lee et al. [17] proposed DiffusionExplainer to support a fundamental understanding of the generative process of a diffusion model at an instance level. It offered insights into how the evolution pathways of an instance deviate within an embedding space, which is obtained by projecting intermediate noisy images via UMAP onto a 2D space. However, our goal is to extend to a holistic understanding of this evolutionary process.

Visualizing high-dimensional evolutionary data: Visualizing the high-dimensional evolutionary data is crucial to understanding the complex data evolution process within diffusion models. Techniques like dimensionality reduction with tSNE or UMAP offer value in visualizing data evolution over a few instances [17]. Such methods have also been used in visualizing data evolution over several instances in strategic games [11]; however, these methods fall short when applied to diffusion model data. They struggle to preserve the iterative structure inherent in diffusion models (see Fig. 2) due to their stochastic nature, which is highlighted when visualizing large-scale samples. On the other hand, neural network features across layers [35] or optimization iterations [18] have been visualized with one projection per layer, making it challenging to study evolution. Parallel coordinates are a popular means to visualize evolutionary or temporal data in other applications [2, 8]; however, projecting high-dimensional data from diffusion models onto a single dimension presents challenges. Recent methods incorporating soft Gaussian constraints to position points in an iteration together [20] offer a potential solution but do not align images across iterations. For this purpose of alignment, other works have proposed vertical distances between pairs of points that connect together [3, 28], which is useful in our case. However, in the evolutionary data from diffusion models, early iterations hold less

relevance than later ones in the generation process. Hence, it might not be ideal to allocate the same vertical visual space to all iterations.

In our evolutionary embedding, we aim to combine soft constraints for iteration displacement and instance alignment and extend it to the context of diffusion models. We further extend these concepts to a layout that optimizes space usage across generative iterations and facilitates the comprehensive study of diffusion model evolution.

4 TREE OF DIFFUSION LIFE

We introduce a method *Tree of Diffusion Life* (*TDL*) to holistically study data evolution in the generation process of diffusion models. *TDL* involves sampling the generative space, semantically encoding these samples via image encoders, and a novel evolutionary embedding to visualize the data evolution. In this section, we detail the goals of our approach and then describe *TDL*.

The theoretical representation of data evolution, extensively illustrated in literature [41, 44], is shown in Fig. 2. During generation, the data distribution $u(\mathbf{h}_t)$, which is represented as a 1D Gaussian, evolves from a single mode, indicating pure noise, to a complex distribution containing multiple modes. This visualization not only enhances interpretability but also supports understanding the decision-making dynamics of dataset modes across iterations, i.e., branching, as well as feature evolution. However, projecting high-dimensional evolutionary data onto a low-dimensional space is challenging. Prior works either visualize localized evolution dynamics [17, 26] or explore this phenomenon on low-dimensional toy data [14]. We aim to represent the data evolution in Fig. 2 using real high-dimensional data for a practical understanding of the generation process in diffusion models. To achieve this, our method has the following goals:

- **G1**: Semantically similar elements in the high-dimensional space need to be placed together in the low-dimensional projection.

- **G2**: Elements belonging to the same iteration must be placed together, enabling the study and comparison of how different data modes evolve. For example, in Fig. 2, the topmost mode, i.e., the *shark* class, emerges sooner than the others.

- **G3**: Pathways connecting elements of a specific instance across iterations should intersect minimally. This requirement ensures we only observe actual branching patterns and avoid misalignment across iterations due to the method's stochastic nature.

We propose *TDL* to achieve these goals and holistically understand the data evolution process in diffusion models. *TDL*'s components are illustrated in Fig. 1. We first sample the diffusion models' generative space via instances with different prompts, extracting all images across instances i and iterations t. These include the evolving noisy images $\mathbf{h}_{t-1}^i$ and the smooth estimate $\mathbf{h}_{t_0}^i$ of the final image at iteration t. Since these images are high-dimensional, we project them to an intermediate space via an image encoder f for semantically meaningful distances (**G1**), i.e., $\widehat{\mathbf{h}}_{t-1}^i = f(\mathbf{h}_{t-1}^i)$ and $\widehat{\mathbf{h}}_{t_0}^i = f(\mathbf{h}_{t_0}^i)$. We explore classifiers and foundational image encoding models for this step. The intermediate space is then analyzed with a proposed evolutionary embedding algorithm that explicitly encodes iterations, supporting the exploration of data evolution and, in turn, the generative process. Our evolutionary embedding method projects the semantically encoded elements ($\widehat{\mathbf{h}}_{t-1}^i$ or $\widehat{\mathbf{h}}_{t_0}^i$) to low-dimensional ones ($\mathbf{l}_{t-1}^i$ or $\mathbf{l}_{t_0}^i$). We propose radial and rectilinear layouts where a ring or a vertical line represents an iteration, respectively, enabling analysis of the evolutionary process.

5 EVOLUTIONARY EMBEDDING

The core of *TDL* is an evolutionary embedding algorithm that aims to preserve the iterative context while enabling visualization of data evolution, as shown in Fig. 3. We propose radial and rectilinear layouts that explicitly represent iterations to facilitate this analysis. The evolutionary embedding incorporates three cost metrics, each tied to one of the goals described earlier, namely,

- **Standard t-SNE loss (G1)**: to cluster semantically similar elements. It is defined as the KL divergence between the high
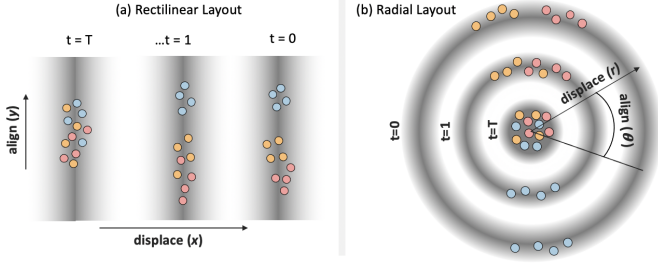
Fig. 3: Evolutionary embedding process for the (a) rectilinear and (b) radial layout. Elements of iterations t are attracted to dark, low-cost areas based on a Gaussian at a displacement value ($\bar{x}_t$ or $\bar{r}_t$) per t. Elements of an instance across iterations are aligned based on their y or θ.

and low-dimensional spaces, preserving the neighborhood of the high-dimensional space in the low-dimensional embedding [45].

- **Displacement loss (G2)**: to explicitly place elements of the same iteration together, we impose a specific position or offset per iteration, i.e., radius or horizontal position. This metric uses a Gaussian centered around the corresponding iteration offset in the low-dimensional space.

- **Alignment loss (G3)**: to align elements of the same instance across iterations, this loss is measured as the alignment offset in vertical position or radial distance among sequential pairs of elements across iterations.

The displacement and alignment losses are additional constraints on the low-dimensional embedding. We propose two layouts: a rectilinear one in Cartesian and a radial one in polar coordinates. While conceptually similar, the losses are modified to handle the different characteristics and coordinate systems of each layout.

Rectilinear Layout: The rectilinear layout models each iteration as a vertical line (see Fig. 1 and Fig. 3). In this layout, for each iteration t, the standard t-SNE loss (**G1**) is computed to preserve semantic neighborhoods from the high-dimensional space to the low-dimensional space. Each iteration t is independent from other iterations. Specifically, Eq. (1) is modified as follows to represent the semantic cost function component, C_s,

$$C_s = \sum_t \text{KL}(P_t \| Q_t) = \sum_t \sum_i \sum_{j,j \neq i} p_t^{ij} \ln \frac{p_t^{ij}}{q_t^{ij}}$$

$$\text{where,} \quad p_t^{j|i} = \frac{\exp\left(-\|\widehat{\mathbf{h}}_t^i - \widehat{\mathbf{h}}_t^j\|^2/(2\sigma_i^2)\right)}{\sum_{k \neq i} \exp\left(-\|\widehat{\mathbf{h}}_t^i - \widehat{\mathbf{h}}_t^k\|^2/(2\sigma_i^2)\right)} \quad (4)$$

$$q_t^{ij} = \frac{(1 + \|\mathbf{l}_t^i - \mathbf{l}_t^j\|^2)^{-1}}{\sum_{k \neq m}(1 + \|\mathbf{l}_t^k - \mathbf{l}_t^m\|^2)^{-1}}$$

The low-dimensional elements $\mathbf{l}_t^i$, with coordinates (x_t^i, y_t^i), are organized per iteration t with the Cartesian displacement loss C_d^c for **G2**. C_d^c is implemented as a Gaussian centered around an x-coordinate offset $\bar{x}_t$ per iteration within the low-dimensional space, as follows:

$$C_d^c = \sum_t \sum_i C_d^c(i,t), \text{ where, } C_d^c(i,t) = -\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_t^i - \bar{x}_t)^2}{2\sigma^2}} \quad (5)$$

The spread or thickness of each iteration is determined by the parameter σ, the Gaussian's standard deviation. This Gaussian attracts elements at iteration t to $\bar{x}_t$, hence explicitly encoding iterations (see Fig. 3). In practice, this σ starts with a user-defined larger value to allow the points to move freely and then is reduced across optimization iterations to move points closer to $\bar{x}_t$. We start with a $\sigma = 20$ and end with $\sigma = 10$ for all our experiments. Additionally, vertical lines are

visually positioned with a certain buffer spacing between them, determined based on user requirements. We used a spacing s of 20 in all cases[1], implying that $\bar{x}_t = s * (T - t)$. The noisy iteration with larger t is positioned on the left and has the smallest $\bar{x}_t$.

Lastly, the Cartesian alignment loss C_a^c (**G3**), aligns elements of each instance i across iterations. This loss is defined as the vertical distance between sequential pairs of elements across iterations t (see Eq. (6)). Larger t values correspond to early generation steps.

$$C_a^c = \sum_i \sum_{t=1}^T C_a^c(i,t), \text{ where, } C_a^c(i,t) = \sqrt{(y_t^i - y_{t-1}^i)^2} \quad (6)$$

The final loss C^c is the weighted sum of the semantic, displacement, and alignment losses, with respective weights α, β, and γ. For the rectilinear layout, empirical findings suggest setting $\alpha = 1, \beta = 5, \gamma = 0.2$, since these values consistently yield desired results[1].

$$C^c = \alpha * C_s + \beta * C_d^c + \gamma * C_a^c \quad (7)$$

We use a similar optimization method as proposed by van der Maaten [45]. The combined cost C^c is minimized via gradient descent. The gradients are computed as follows,

$$\frac{\partial C}{\partial x_t^i} = \alpha * \frac{\partial C_s}{\partial x_t^i} + \beta * \frac{\partial C_d^c}{\partial x_t^i} \qquad \frac{\partial C}{\partial y_t^i} = \alpha * \frac{\partial C_s}{\partial y_t^i} + \gamma * \frac{\partial C_a^c}{\partial y_t^i}$$

$$\frac{\partial C_s}{\partial x_t^i} = 4\sum_j (p_t^{ij} - q_t^{ij})(x_t^i - x_t^j)X_t^{ij} \qquad \frac{\partial C_s}{\partial y_t^i} = 4\sum_j (p_t^{ij} - q_t^{ij})(y_t^i - y_t^j)Y_t^{ij}$$

$$\frac{\partial C_d^c}{\partial x_t^i} = \frac{(x_t^i - \bar{x}_t)}{2\sigma^2} * C_d^c(i,t) \qquad \frac{\partial C_a^c}{\partial y_t^i} = \frac{(y_t^i - y_{t+1}^i)}{C_a^c(i,t)}$$

$$(8)$$

Where, $X_t^{ij} = (1 + (x_t^i - x_t^j)^2)^{-1}$, and, $Y_t^{ij} = (1 + (y_t^i - y_t^j)^2)^{-1}$. While C_s in Eq. (4) illustrates the high-dimensional elements $\mathbf{h}_t^i$, these elements can also be other high-dimensional elements such as $\widehat{\mathbf{h}}_t^i$. $\partial C/\partial x_t^i$ and $\partial C/\partial y_t^i$ are the resultant force exerted on the data point $\mathbf{l}_t^i$ by all other elements. We simultaneously optimize the three loss functions. The resultant force of the three costs leads to points moving within the vertical band corresponding to iteration t. We initialize $\mathbf{l}_t^i$ in a buffer region around the corresponding $\bar{x}_t$ for faster optimization results. The expected result is that points are semantically clustered around their corresponding x-offsets $\bar{x}_t$ per iteration while maintaining the instance alignment across iterations (see Fig. 1).

Radial layout: The rectilinear layout fulfills our goals by enhancing the visualization of branching patterns and resembling the theoretical representation of expected behavior in Fig. 2. However, it utilizes space sub-optimally since equal space is allocated to all iterations despite the early-generation steps having less relevant patterns than later ones. This results in vacant unused spaces in the initial iterations. Hence, we propose a radial layout consisting of concentric rings (see Fig. 1 and Fig. 3). Each iteration is represented as a ring, the innermost being the early-generation steps, and the outer rings are later iterations. A key difference between the rectilinear and radial layout is the transformation of the cost function to use the perspective of polar coordinates. In a radial layout, early noisy iterations occupy less space at the center compared to later iterations in the outer rings, addressing sub-optimal space utilization.

The low-dimensional elements $\mathbf{l}_t^i$, in Cartesian coordinates (x_t^i, y_t^i), can be represented in polar coordinates (r_t^i, θ_t^i). Conceptually, the same three losses C_s, C_d^c, and C_a^c are utilized, but the form is adapted to the radial layout. For the sake of simplicity and to preserve correct gradients, the semantic loss, C_s, and its gradients are still computed in Cartesian space and then converted to polar coordinates. In other words, the semantic loss is still based on the Euclidean distance.

---

[1]Exploration of these hyper parameters are in the supplementary materials.

The displacement loss (**G2**) was rewritten for the radial layout C_d^p, as it needs to organize elements of an iteration within concentric rings. This implies a Gaussian centered around a radius-offset $\bar{r}_t$ per iteration in the low-dimensional space as defined in Eq. (9). The Gaussian encodes iterations by attracting elements at iteration t toward the ring $\bar{r}_t$ (see Fig. 3). The innermost ring is represented with the offset $\bar{r}_{t=T} = 0$, forming a circle. Subsequent rings have user-defined offsets $\bar{r}_{t!=T} > 0$, determining their spacing based on user preference. In our experiments[1], this spacing was set to 20, i.e., $\bar{r}_t = 20 * (T - t)$.

$$C_d^p = \sum_t \sum_i C_d^p(i,t), \text{ where, } C_d^p(i,t) = -\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(r_t^i - \bar{r}_t)^2}{2\sigma^2}} \qquad (9)$$

Lastly, the alignment loss (**G3**) in the radial layout C_a^p aligns elements of each instance i across iterations by minimizing the cosine distance between the angles θ_t^i of the sequential pairs of corresponding elements of an instance across iterations below. By using the absolute value of the cosine similarity, we create an unstable equilibrium in the cost function when the angular difference approaches $180°$ degrees. This unstable point facilitates faster optimization by preventing the algorithm from getting stuck in the poor local minima of two related points being located at diagonally opposite ends of a ring.

$$C_a^p = \sum_i \sum_{t=1}^{T} 1 - \left|\text{sim}_t^i\right|, where, \text{sim}_t^i = \cos\left(\frac{\theta_t^i - \theta_{t-1}^i}{2}\right) \qquad (10)$$

The final loss C^p is the weighted sum of C_s, C_d^p, and C_a^p with respective weights α, β, and γ. Empirical findings suggested setting $\alpha = 1, \beta = 5, \gamma = 0.05$ for the radial layout consistently yields desired results[1]. The optimization process follows a similar approach to the one described earlier. However, in this case, the resultant force of the three cost elements leads to points moving along the respective iteration rings. The gradient updates in polar form are defined as follows:

$$\frac{\partial C_d^p}{\partial r_t^i} = \frac{(r_t^i - \bar{r}_t)}{2\sigma^2} * C_d^p(i,t)$$

$$\frac{\partial C_a^p}{\partial \theta_t^i} = \begin{cases} -\frac{1}{2}\sin\left(\frac{\theta_t^i - \theta_{t-1}^i}{2}\right) & \text{if } \text{sim}_t^i < 0 \\ \text{undefined} & \text{if } \text{sim}_t^i = 0 \\ \frac{1}{2}\sin\left(\frac{\theta_t^i - \theta_{t-1}^i}{2}\right) & \text{if } \text{sim}_t^i > 0 \end{cases} \qquad (11)$$

The gradient of C_a^p with respect to θ_t^i when $\text{sim}_t^i = 0$ is undefined. This is the unstable equilibrium point, where the point can move in either direction along the ring to reach a similar point that is diagonally opposite. While it is unlikely for $\text{sim}_t^i = 0$ in practice, one of the other two gradient options is randomly chosen during optimization.

We described the evolutionary embedding method that generates the rectilinear and radial layout, meeting our goals. Next, we present visualization aspects of evolutionary embeddings within TDL.

5.1 Visualization

The evolutionary embedding is visualized in either radial or rectilinear layouts. Iterations are represented as either concentric rings or vertical lines, highlighting progression from noisy (innermost ring or left-most vertical line) to clearer representations (outermost ring or right-most vertical line). When visualizing these embeddings to enable analysis, several considerations arise. These mainly include 1) linking the embedding and the high-dimensional diffusion-image space, 2) supporting analysis of the progression or evolution of instances through the generation process, and 3) characterizing images through prompt keyword-image links to improve comprehension of high-dimensional evolutionary data. In this section, we present a possible means to facilitate this analysis as a proof of concept[2].

---
[2]See supplementary figures for a visual representation of this method.

To comprehend the high-dimensional image space via the embedding, we display images of specific iterations ($\mathbf{h}_{t-1}^i$ or $\mathbf{h}_{t_0}^i$) at regular intervals along the circumference of the largest ring or along the length of the right-most vertical line, spaced at intervals of θ degrees or y units, respectively (refer to Fig. 1). The high-dimensional images $\mathbf{h}_{t-1}^i$ or $\mathbf{h}_{t_0}^i$ of selected embedding elements are dynamically displayed for further linked analysis. To further explore the evolution and hierarchy of data, keywords or attributes within a prompt are utilized to color the low-dimensional embedding points. For example, a prompt "a woman with black hair" can be colored by the keyword *woman* or *black hair*, analyzing the effect of each keyword on the evolution process. To enhance the visualization of instance evolution, pathways connecting images of the same instance across iterations are traced with smooth curves. Given the potential for visual clutter arising from showing all pathways, we visualize only the evolution of selected elements $\mathbf{l}_{t-1}^i$ of a specific instance i across iterations t (see Fig. 7). These pathways are drawn as smooth cardinal splines using respective elements $\mathbf{l}_t^i$ of an instance i as control points. The embedding points are clustered per iteration and keyword using DBSCAN [9] to achieve a cleaner visualization of pathways. Pathways and points within each cluster are interpolated towards the cluster centroid based on a controllable interpolation factor. While there could be better approaches to achieve this, we leave it to future work. Simple data filters on the iteration step t, path lengths, and prompt keywords are provided to facilitate analysis. Users can interactively control keyword-color associations and pathway interpolation factors. It's important to note that our contribution primarily focuses on dimensionality reduction and layout generation rather than the interactive front-end aspect. We leave it to future work to build a more comprehensive front-end.

6 EXPERIMENTS WITH TDL COMPONENTS

Within TDL, several components exist, including images to be encoded, image encoders, the two layouts, and their loss metrics. Understanding how these components interact and their effects on evolutionary embedding within TDL is crucial. In the following sections, we present our setup utilized for these experiments. We then evaluate and compare the two proposed layouts with the baseline vanilla t-SNE. Finally, we explore various image encoders and analyze the two types of diffusion images we extract: smooth and noisy representations. By doing so, we aim to gain insights into how each component influences the evolutionary embedding process within TDL and aids in tuning the embedding quality for various datasets and applications.

6.1 Experimental Setup

We use GLIDE [24], a popular text-to-image diffusion model, for our experiments. We generate a set of 1000 samples from 10 prompts, namely *tiger*, *bird*, *car*, *fruit*, *building*, *dog*, *shark*, *ball*, *plane*, *table*. Randomly selected ImageNet classes were used as prompts, with 100 samples from each prompt. GLIDE's base model was used to generate samples in 100 iterations. For the evolutionary embedding, 11 equally spaced iterations $t \in 99, 90, 80, .., 10, 0$ were used.

We examine the impact of the diffusion images ($\mathbf{h}_{t-1}$, $\mathbf{h}_{t_0}$) encoded with different image encoders f to generate intermediate spaces of $\widehat{\mathbf{h}}_{t-1}$ and $\widehat{\mathbf{h}}_{t_0}$ respectively. The evolutionary embedding is applied to this intermediate space. The encoders explored include a robust ResNet50 ImageNet classifier [1] and the image encoder of CLIP [34]. CLIP is a foundational model that learns visual concepts from natural language supervision. Additionally, we apply the evolutionary embedding directly on the raw diffusion images ($\mathbf{h}_{t-1}$ and $\mathbf{h}_{t_0}$) for comparison.

6.2 Evaluation

We explore the utility of each loss metric of Eq. (7) in Fig. 4 and compare our evolutionary embedding layouts with vanilla t-SNE [45].

For this evaluation, we project all noisy representations $\mathbf{h}_{t-1}^i$ to an intermediate semantically meaningful space of $\widehat{\mathbf{h}}_{t-1}^i$ with a robust ImageNet classifier [1]. This intermediate space is analyzed with our evolutionary embedding method and vanilla t-SNE, each optimized for 2000 iterations. The vanilla t-SNE, obtained by projecting all images
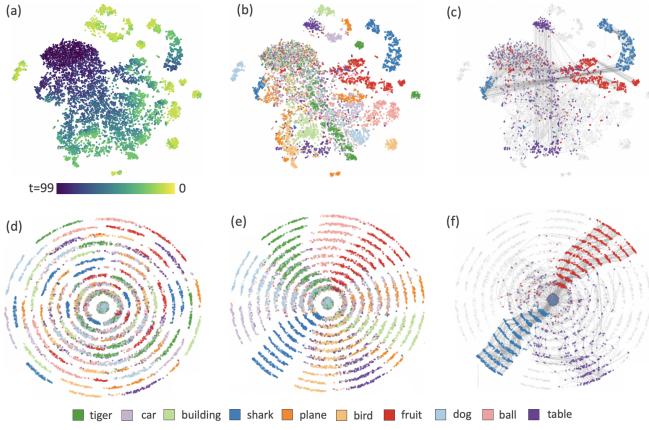
Fig. 5: Q_{trust}^t (left) and Q_{cont}^t (right) of vanilla t-SNE, and the proposed radial (*radial*) and rectilinear (*rect*) layouts with (*all*) and without (*noalign*) the instance alignment metric across iterations t. Embeddings are obtained on $\widehat{\mathbf{h}}_{t-1}^i$. $t = 100$ is noise. $\alpha = 1$.

Fig. 4: Single vanilla t-SNE [45] embedding on all $\widehat{\mathbf{h}}_{t-1}$ extracted from the GLIDE ImageNet use case with a classifier [1]. Points are colored by (a) iteration t and (b) prompt. Radial layout without (d) and with (e) the instance alignment loss. Pathways of "tables", "sharks", "fruits" are shown on the vanilla t-SNE (c) and radial layout (f).

on a single 2D embedding, shows some form of evolution seen in the continuity of points in Fig. 4a. However, as clusters become more distinct, i.e., the yellow points of Fig. 4a, they do not have continuity with previous iterations t. The non-alignment of iterations t and the stochastic placement of clusters make it challenging to trace data evolution. Our proposed radial layout in Fig. 4e, which explicitly encodes iterations, shows per cluster evolution patterns more distinctly. For example, we see that the "shark" and "fruit" instances evolve early on compared to other object types in Fig. 4f. This is not immediately apparent in the vanilla t-SNE embedding in Fig. 4c; both Fig. 4a and b need to be analyzed together to come to this conclusion. The stochastic placement of clusters in the radial layout of Fig. 4d (or C_a^c for the rectilinear layout) is minimized with the proposed instance alignment loss C_a^p, while maintaining cluster relations (see Fig. 4e).

The evolutionary embedding aims to introduce aligning constraints for visual clarity of the diffusion process while preserving the high-dimensional relations within an iteration. Hence, we compare the performance of our method against the vanilla 2D t-SNE [45] in Fig. 4 to ensure preservation of the high-dimensional neighborhoods. For this, we utilize projection quality metrics trustworthiness (Q_{trust}) and continuity (Q_{cont}) of instances [15] within an iteration t. These metrics take values in [0,1]; the higher, the better. Q_{trust}^t checks whether the k nearest neighbor of an element in the embedding is also a neighbor in the original high-dimensional space within an iteration, where $Q_{\text{trust}}^t = 1 - A_k \sum_i \sum_{x_t^j \in U_k(x_t^i)} r(x_t^i, x_t^j) - k$. Here, $r(x_t^i, x_t^j)$ signifies the rank of a sample x_t^j concerning its distance to x_t^i in the high-dimensional space. $U_k(x_t^i)$ is the set of data elements that are neighbors of x_t^i in the embedding but not in the high-dimensional space. $A(k)$ is a scaling factor. Similarly, Q_{cont}^t checks whether the k nearest neighbor of an element in the high-dimensional space is preserved in the embedding, where $Q_{\text{cont}}^t = 1 - A_k \sum_i \sum_{x_t^j \in V_k(x_t^i)} \hat{r}(x_t^i, x_t^j) - k$. Here, $V_k(x_t^i)$ are those elements that are neighbors of x_t^i in the high-dimensional but not the embedding space. We utilize $k = 7$ inline with previous works [15, 22].

Figure 5 illustrates Q_{trust}^t and Q_{cont}^t across various layouts. Notably, these metrics values for the proposed evolutionary embeddings remain comparable to vanilla t-SNE. Further, there is only a minor deterioration in Q_{trust}^t and Q_{quant}^t when the alignment losses (C_a^c and C_a^p) are introduced for the rectilinear and radial layouts in Fig. 5 compared to the metric without the alignment. This indicates that evolutionary embedding preserves the high-dimensional structure like vanilla t-SNE while providing benefits in understanding data evolution.

Additionally, the evolutionary embedding method significantly outperforms vanilla t-SNE in terms of speed for this experiment. This is primarily because we divide t-SNE into smaller segments per iteration
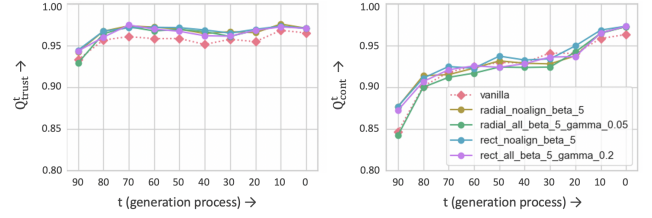
and combine them with simple alignment losses. The preprocessing step with PCA takes ≈ 1 minute for our method in comparison to ≈ 6 minutes for the vanilla t-SNE. Further, our method takes < 1 second, while vanilla t-SNE takes ≈ 9 seconds per iteration. This substantial reduction in computational time combined with the visual clarity of data evolution highlights the efficiency and effectiveness of our approach.

The quality metrics between the proposed radial layout compared to the rectilinear one are comparable, indicated by similar trends of these layouts in Fig. 5. The main decline in quality metrics of the proposed radial method comes in the largest t value early in the generation process. Since these represent almost pure noise, they are less significant than the later generation steps. We support both layouts for analysis; the rectilinear is closer to the theoretical representation, while the radial one makes better use of the visual space.

6.3 Image Encoders & Images Encoded

In this section, we show the influence that the semantic encoders have on *TDL*'s results. Figure 6 highlights the use of various image encoders f on the two types of diffusion images, the noisy $\widehat{\mathbf{h}}_{t-1}$ and smooth $\widehat{\mathbf{h}}_{t_0}$. The raw image encodings (see Fig. 6a and Fig. 6d) are ineffective in capturing semantic clusters since they capture only local-pixel-level differences rather than higher-level semantic changes. However, they capture groups, like *shark*s and *fruit*s that are significantly different from the others, evident from the red and blue clusters visible from the second or third inner-most concentric circle in Fig. 6a and Fig. 6d.

On the other hand, task-specific classifiers or feature detectors like the robust ImageNet classifier [1] lead to a more accurate representation of semantic groups, especially on the noisy elements $\widehat{\mathbf{h}}_{t-1}$ (see Fig. 6c). The robust nature of the model is also reflected in the groups in early generation iterations (inner rings) compared to CLIP encoded features in Fig. 6b. Both encoders capture the smooth representation $\widehat{\mathbf{h}}_{t_0}$ well
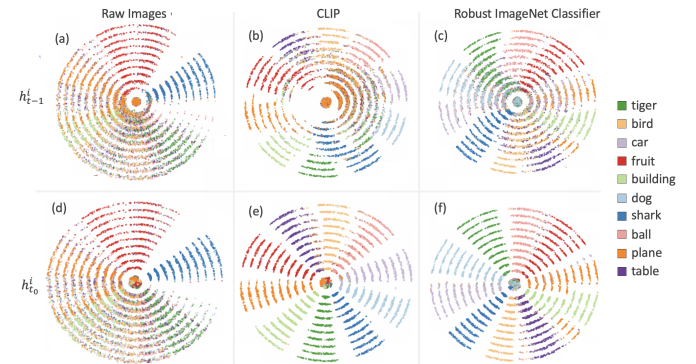

Fig. 6: Applying evolutionary embedding on GLIDE images with various encoding strategies. (a) Raw noisy images $\mathbf{h}_{t-1}$ and (d) raw smooth images $\mathbf{h}_{t_0}$ undergo evolutionary embedding. Noisy images $\mathbf{h}_{t-1}$ are encoded with (b) CLIP [34] image encoder, and (c) a robust ImageNet classifier [1] before the embedding creation. Similarly, smooth images $\mathbf{h}_{t_0}$ are encoded with (e) a CLIP and (f) a robust ImageNet classifier.
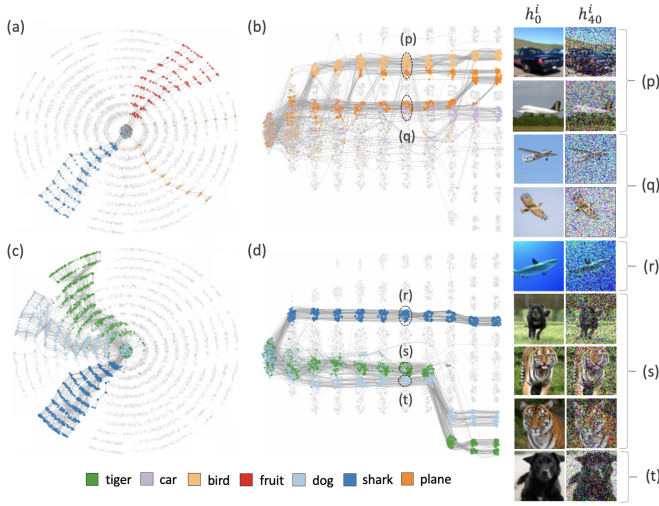
Fig. 7: Pathways of instances $\widehat{\mathbf{h}}_{t-1}^i$ from GLIDE [24] encoded with an ImageNet classifier [1] traced in the evolutionary embeddings: (a) Pathways shorter than the 5th percentile in radial layout, primarily highlighting the *shark* and *fruit* instances. (b) Evolution of *plane* class and its clusters in the rectilinear layout, with images resembling *bird*s or *car*s at noisy t=40 stages. Underwater (*shark*) and land images evolving before the subdivision within land images, i.e., *tiger*s and *dog*s in the (c) radial and (d) rectilinear layouts.

Fig. 8: Evolution of human facial attributes with Stable Diffusion. Smooth representations $\widehat{\mathbf{h}}_{t_0}$ from CLIP summarized with evolutionary embedding and color-encoded to depict *gender* (a), *hair color* (b) in the radial layout, and *age + gender* in the radial (c.1) and rectilinear layout (c.2). Sample images at iteration t are highlighted with circles. Primary separation is by *gender* followed by *age*, *grey* hair, and finally, *blonde* and *brown* hair.

(see Fig. 6e and Fig. 6f). While task-specific classifiers lead to more granular features captured in the intermediate space, foundational models like CLIP provide a generic representation across tasks. Ultimately, the choice of semantic encoder impacts the quality and granularity of semantic information captured, which is crucial for the effectiveness of *TDL* in various applications.

7 USE CASES

In the previous section, we evaluated and explored the various components of *TDL*. Here, we illustrate the potential of *TDL* for analysis of high-level object types, followed by finer attributes and prompt hierarchies to gain deeper insights into the data evolution process. Specifically, we explore three prompt sets; the first relates to high-level object types or ImageNet classes, the next is finer human facial features, and the last one relates to different styles of cat images. We use GLIDE [24] and Stable Diffusion [36] for the first and last two cases, respectively, since they are widely explored and popular text-to-image diffusion models that are available in the open-source. While these two models represent two large classes of underlying diffusion model types, we primarily focus on text-to-image models for our analysis.

7.1 Exploring ImageNet objects

In this section, we study the evolution of higher-level object types that are relatively distinct from one another. Our aim is to explore when these modes come into existence in the generative process. Specifically, the evolution of 10 ImageNet classes previously described in Section 6.1 are explored with GLIDE [24]. We start by analyzing the evolutionary embeddings of the noisy instances encoded with the robust classifier [1] in Fig. 7. The lengths of these paths traced provide insight; the shorter paths indicate distinct modes, and the longer paths indicate branching and zig-zag patterns. When exploring shortest pathways, we note that these mainly arise from the *shark* and *fruit* classes (see the red and blue highlighted points in Fig. 7a). This finding is also consistent with the distinct clusters of these two classes that emerge in the inner-most circles of in Fig. 6a and 6b, owing to their distinct colors.

Since the *plane* instances are grouped into two clusters in the last iteration of Fig. 7c, we explore them. These clusters mainly arise due to the fact that the noisy *plane* instances $\mathbf{h}_t^i$ are divided based on whether they are in the air, resembling *bird*s in Fig. 7q or are on land, resembling
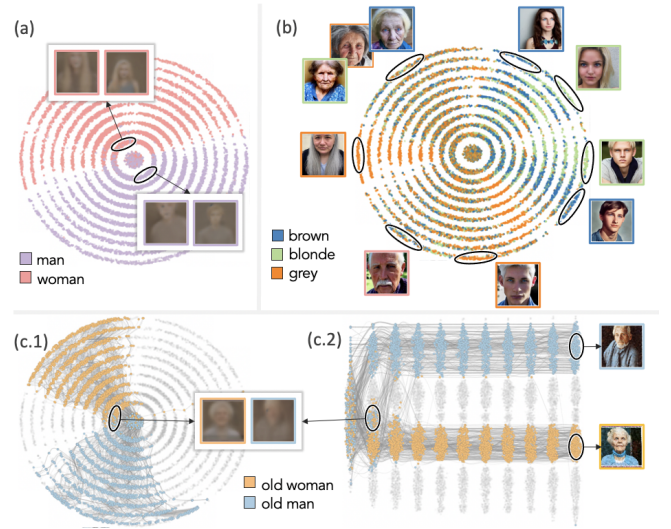
*car*s in Fig. 7p. While the class distinction becomes evident towards the end of the generation process, the noisy images $\mathbf{h}_{t=40}^i$ are visually confusing. This pattern is also seen in the CLIP-encoded features in Fig. 6b. Similarly, we observe a distinction between noisy instances $\mathbf{h}_t^i$ with grassy (green) and water (deep blue) backgrounds in Fig. 7. The grass and water instances separate out first (Fig. 7r vs. Fig. 7s) already in the second inner ring, followed by sub-clusters within the grassy images. In this case, the grassy images consisted of *dog*s and *tiger*s on green grass, and the water images were those of *shark*s.

When exploring the smooth instance representations $\widehat{\mathbf{h}}_{t_0}^i$ in Fig. 6e and 6f, the diffusion model shows an understanding of vague object structures early in the generation process. Although some classes like *plane* and *car*, or *dog*, *tiger*, and *bird* remain intertwined in Fig. 6f, most object modes emerge earlier in the generation process, primarily within the inner rings. This indicates that the model generates most global structures within the first few iterations, a finding supported by literature [6, 26, 33, 39]. Next, we dive deeper into the evolution of finer features to enhance our understanding of the generation process.

7.2 Exploring human facial features

In this section, our aim is to investigate the hierarchical evolution of features, identifying when they emerge during the evolutionary generative process with Stable Diffusion [36]. We take an example of human facial features for this purpose, as this has been explored in literature [7]. While these works also aimed to reveal such a structure [7], we aim to provide a more generic approach that is also more interpretable. We additionally aim to elucidate any entanglement between data attributes relevant for advancing diffusion models [51]. Our prompt is defined by the regular expression: "a photo of a [*young | old*] [*man | woman*]; with [*brown | grey | blonde*] hair; with [*brown | blue*] eyes; [*wearing a necklace*]?". Images are generated in 40 iterations. We further sub-sample 11 iterations within this for the evolutionary embedding. Here, we explore the attributes *age*, *gender*, *haircolor*, *eyecolor*, and whether they were *wearing a necklace*. This yielded a total of 48 prompts representing all possible keyword combinations defined above. 50 instances are sampled from each prompt for exploration. The smooth representations $\widehat{\mathbf{h}}_{t_0}$ derived from CLIP [34] are summarized through the proposed evolutionary embedding method in Fig. 8. To explore the order in which facial features emerge, we color-encode different attributes, including age, gender, and hair color, in a sequential manner.
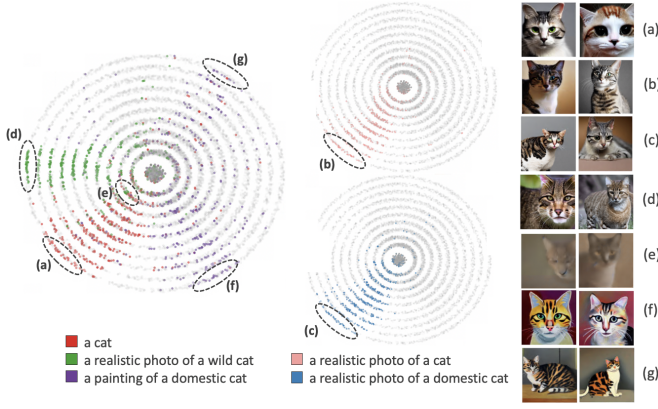
Fig. 9: Effect of prompt hierarchy on data evolution in Stable Diffusion. The radial evolutionary embedding is based on smooth representations $\widehat{\mathbf{h}}_{t_0}^i$ of instances, with points color-encoded. Images generated with different prompt hierarchies, such as "a cat" (a), "realistic photo of a cat" (b), and "realistic photo of a domestic cat" (c), are clustered in the embedding. However, images with the prompt "realistic photo of a wild cat" initially mix with others (e) before segregating into separate clusters (d). "Painting"s of cats (f, g) encompass varying painting styles.

When data points are colored by age, the layout highlights the model's capability in distinguishing between *old* and *young* individuals, evident as early as the second-most inner ring in Fig. 8a. Notably, the smooth images $\mathbf{h}_{t_0}$ depicted in the figure already exhibit discernible facial and hair outlines, with longer hair possibly indicating female subjects. Next, when points are color-encoded by both *age* and *gender*, specifically for older individuals, we observe that the model is confused at the second-most inner ring, where *age* was already apparent (see Fig. 8c.1). This mix-up is also evident in the crossing of pathways at the second generation step in Fig. 8c.1 and 8c.2. Examining the smooth images highlights at this iteration the source of this confusion: both *old men* and *old women* exhibit fuzzy, white short hair. As the generation progresses, this distinction becomes clear, as shown in images on the right side of Fig. 8c.2.

Lastly, when points are color-encoded by hair color, interesting insights emerge in Fig. 8b. We observed instances of individuals with *grey hair* in orange primarily spread across the left half of the radial layout. While this group primarily included *old*er individuals, we noted the presence of *young women* with *grey hair*. Some of these women were middle-aged rather than *young*, hinting at a potential mix-up in age attributions. This entanglement between *age* and *hair* color is further highlighted by the observation that many older individuals demonstrate whitish hair regardless of the prompt, as seen in the left half of Fig. 8b. Such entanglements are crucial to detect, as they may impact model performance. Our visualization aids in identifying and addressing such issues, facilitating the refinement of diffusion models [51]. On the other hand, while exploring hair colors of *young men*, we observed that *grey hair* separates out first, followed by *blonde* and *brown hair* much later in the generation process, particularly after the eighth ring on the right half of Fig. 8b. While foundational models like CLIP were sufficient for detecting larger features like *hair color*, they were insufficient for very fine details like the *eye color* in our case. To explore these finer features, task-specific classifiers or attribute detectors must be employed.

We showed how *TDL* helped us identify hierarchies in features or attributes within a prompt. For example, in this case, *gender* was the first feature to be distinguished based on hair outlines, followed by *age*, and then *hair colors*. *TDL* also supported the detection of entangled attributes (*age* and *hair color* in this case). It is essential to recognize that these insights can vary based on each trained diffusion model, reflecting its unique learnings and biases. Our method is meant to be a generic means to support the understanding of this evolutionary process.
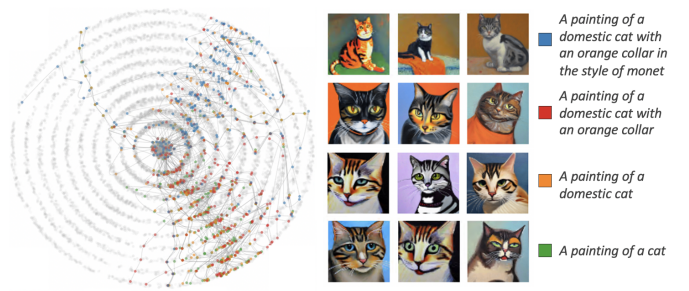

Fig. 10: Data evolution of paintings of cats with a hierarchical structure generated with Stable Diffusion. The radial evolutionary embedding is based on smooth elements $\widehat{\mathbf{h}}_{t_0}^i$, with points color-encoded by their prompt. Instances where the painting style (*monet*) is not specified, are positioned together in the bottom-right of the radial layout.

7.3 Exploring image styles

In this section, we simulate a realistic usage of prompt engineering for image generation by designing prompt hierarchies, going from generic prompts to more specific ones, and examining their impact on the evolutionary generative process in Stable Diffusion [36]. Specifically, we explore cat image styles, which are similar to cases that have also been explored in literature [17]. The prompts are defined by the expression "A [*realistic photo* of a | *painting* of a]? [*domestic* | *wild*]? cat [with a *yellow collar* | with an *orange collar*]? [in the style of *monet* | in the style of *van gogh*]?". We sampled 50 instances from each of the 39 combinations above. By organizing the prompts in a hierarchical structure, we start with general prompts and gradually introduce more specific ones. This approach allows for a more controlled exploration of the styles of cat images, mimicking the exploration of options during the image generation process. Similar to the previous case, we use CLIP [34] to encode the smooth representations $\mathbf{h}_{t_0}$ of data. This encoding generates an intermediate space of $\widehat{\mathbf{h}}_{t_0}$, summarized through the proposed evolutionary embedding method. Our initial exploration centers around analyzing the hierarchy in the specificity of prompts, aiming to understand how certain prompts encompass a larger set of instances compared to others.

Figure 9 reveals an interesting pattern. Instances prompted with "a cat" exhibit a similar evolution to those prompted with "a realistic photo of a cat" and "a realistic photo of a domestic cat" (see Fig. 9a, Fig. 9b, and Fig. 9c). The corresponding images on the right side of Fig. 9 indicate that the prompt "a cat" primarily generates real images of domestic cats. In contrast, instances prompted with "a realistic photo of a wild cat" evolve along fully different pathways in Fig. 9d. While there is some initial mixing between these two sets in cluster 9e in early iterations until the fifth ring, they subsequently separate into different directions. It is necessary to analyze this encompassing aspect of prompts to identify the distribution of images they generate. In this case of "a cat", our findings suggest a potential bias towards domestic cats, which may have implications depending on the specific use case.

On the other hand, *painting*s of cats evolve differently (see Fig. 9g and f) from the *photo*s, as expected. For example, the generic prompt "a painting of a domestic cat" in purple encompasses life-like paintings related to the style of *Monet* in cluster 9g, as well as paintings characterized by bold colors related to the style of *van Gogh* in cluster 9f. Interestingly, *painting*s of cats display a wider variety compared to *photo*s, prompting further exploration of prompt hierarchies within this category (Fig. 10). Generic prompts, such as "a painting of a cat" and "a painting of a domestic cat", evolve similarly into those with vibrant colors (see bottom-right of the radial layout in Fig. 10). This finding indicates that the model, by default, generates these more colorful images unless a specific style is specified. Conversely, when explicitly specifying a style, such as *monet*, leads to the emergence of softer-toned images in the top-right of the radial layout of Fig. 10.

In summary, our exploration into prompt hierarchies sheds light on

the data evolution within diffusion models. The findings highlight the significance of prompt specificity in influencing the generated image distribution, with more generic prompts potentially leading to (potentially) unintended biased outcomes, while specific prompts yield more specific results that are more desirable.

8 DISCUSSIONS

Evolutionary Embedding: In our implementation of the evolutionary embedding, we utilized the vanilla t-SNE [45] for a proof of concept. However, any dimensionality reduction method that groups semantically similar elements in a low-dimensional space can be utilized. For example, faster t-SNE [31, 32] or UMAP [21] could be explored.

The parameter weighting α, β, and γ of the semantic, displacement, and alignment losses, respectively, are hyper-parameters that require careful consideration since they represent different aspects of data. Through experimentation, we identified a set of weights for each layout that performed consistently well across our cases. Further analysis of these parameter weightings has been left as future work. Additionally, determining the standard deviation, σ, for the Gaussian used in the displacement losses (C_d^c, C_d^p) depends on user preferences. A larger σ is sufficient if clear distinctions between iterations are not crucial, while smaller values may be preferred otherwise.

Concerning the proposed radial layout, we utilize Euclidean distances for the C_s in the low-dimensional space. While empirically, this form was still found to work well in the radial layout, the distance does not follow the ring structure of the layout and hence can be improved. For example, if two similar points, for example, are initially positioned opposite to each other on a ring, the Euclidean distance metric tends to pull them closer across the rings and not follow the ring structure of the level. This is compensated through the displacement loss, C_d^p, which is designed to keep points close to the ring. While the average gradient results in a movement along the ring, the optimization process could be enhanced by designing a polar-space alternative for the C_s loss in the low-dimensional space. However, this poses some challenges to gradient correctness for polar coordinates.

TDL: Concerning TDL's analysis pipeline, the role of an image encoder is essential to study the evolution of specific attributes of data. Data can evolve in different dimensions of change, for example, shapes, colors, or more generic semantic changes of the images themselves. Our exploration has predominantly involved simple encoders such as classifiers and foundational image encoding models like CLIP [34]. While our focus was the evolution of higher-level semantic attributes like *hair color* or object type like *dog* in diffusion images across iterations, other evolution dimensions can also be explored. For instance, the evolution of shapes [47], internal model, i.e., Unet features [7], or the evolution of concept-keyword associations over iterations [25] across various datasets could be explored. Exploring the Unet features is not straightforward since they do not have a latent space, owing to their skip connections, and different layers are important in different iteration steps [33, 39]. Finally, developing advanced interactive interfaces could enhance the analysis of the evolutionary embedding layout generated by TDL. For example, better edge bundling methods could allow the evolution of modes to be less cluttered. Images visualized on the outer portions of the layouts could be sampled to be better representatives of the respective cluster. Additionally, techniques like word cloud embedding could support the exploration of complex prompt sets.

Scalability: Diffusion models consist of several iterations, often in the order of 1000. While popular methods [24, 36] reduce this to less than 100, the volume of data still remains substantial for visualization. This challenge is particularly observed in radial layouts, where the inner circles are allotted less space. In our exploration, we sub-sampled iterations in an evenly spaced manner to mitigate visualization difficulties. However, more effective interactive sampling methods can be used. For example, literature [6, 33] indicates a composition phase where content is created, followed by a denoising phase to refine details. Users could interactively adjust the sampling,

potentially selecting more steps initially and fewer later in the process to facilitate more efficient analysis. Another approach to address visual clutter is to leverage hierarchical embedding methods such as hierarchical stochastic neighborhood embedding or HSNE [29]. Finally, the evolution of only a few attributes can be studied at a single time due to limitations in the number of colors used to encode data attributes like *hair color* discernible by the human eye. There is, hence, a need for sequential prioritization, selection, and analysis of attributes in the current setting.

Future Applications: While the main focus of TDL is to support an understanding of data evolution in the generative process of diffusion models, its applicability extends to several practically-oriented downstream tasks. First, exploring branching patterns and when dataset modes emerge are relevant for designing architectures. For example, noise schedulers can be designed [14] to generate desired branching patterns in an informed way. Further, if all attribute modes of interest manifest in early iterations, the number of iterations could be reduced, or these steps can possibly be skipped entirely [6]. Next, the exploration of attribute entanglements is widely studied in literature [49, 51]. TDL serves as a valuable tool for visualizing these entanglements. An extension of TDL could support interactive data generation processes by providing visualizations to help users understand and influence the generation process. This enhancement would empower users to interactively explore and tune the data generation process according to their specific needs and preferences. Lastly, the proposed evolutionary embedding can be extended to diffusion models beyond text-to-image, generic DL models, such as deep convolutional neural network feature evolution across layers and potentially other (non-image) high-dimensional evolutionary data.

9 CONCLUSIONS

Diffusion models iteratively reconstruct corrupted data, evolving noisy images into refined outputs. Understanding this data evolution is relevant for interpreting model behavior and the learned distribution, but is complex due to its high-dimensional evolutionary nature. Existing methods like t-SNE to study high-dimensional data do not preserve this iterative structure, limiting its analysis. Hence, we propose the *Tree of Diffusion Life* (TDL), a method to gain a holistic understanding of the data evolution in the generative process of diffusion models. TDL samples the generative space by extracting instances of several prompts, employing image encoders to extract semantic meaning from them. We introduce an evolutionary embedding method to explicitly preserve the iterative context while maintaining the high-dimensional data structure to facilitate the analysis of data evolution. This evolutionary embedding includes three loss metrics - a standard t-SNE loss to group semantically similar elements, a displacement loss to group elements per iteration, and an instance alignment loss for an instance's elements across iterations. We propose rectilinear and radial embeddings to facilitate analysis. To achieve meaningful semantic distances, images are projected to an intermediate space using image encoders such as classifiers of foundational models like CLIP. Using different prompt sets, we show the versatility of TDL by applying it to two prominent text-to-image diffusion models, GLIDE and StableDiffusion. We demonstrate TDL's relevance in studying the evolution of data, offering valuable insights into the image generation process of diffusion models.

Looking ahead, TDL sets the basis for potential extension to various practical downstream tasks, such as specific tools to explore entanglements between data attributes or supporting interactive generation. In summary, TDL shows promise in enhancing our understanding of data evolution in diffusion models and its broader implications.

REFERENCES

[1] Y. Bai, J. Mei, A. L. Yuille, and C. Xie. Are transformers more robust than cnns? In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, eds., *Advances in Neural Information Processing Systems*, vol. 34, pp. 26831–26843. Curran Associates, Inc., 2021. doi: 10.48550/ arXiv.2111.05464 1, 5, 6, 7

[2] G. D. Cantareira, R. F. Mello, and F. V. Paulovich. Explainable adversarial attacks in deep neural networks using activation profiles. *arXiv preprint arXiv:2103.10229*, 2021. doi: 10.48550/arXiv.2103.10229 3

[3] G. D. Cantareira and F. V. Paulovich. A Generic Model for Projection Alignment Applied to Neural Network Visualization. In C. Turkay and K. Vrotsou, eds., *EuroVis Workshop on Visual Analytics (EuroVA)*. The Eurographics Association, 2020. doi: 10.2312/eurova.20201089 3

[4] Z. Chang, G. A. Koulieris, and H. P. Shum. On the design fundamentals of diffusion models: A survey. *arXiv preprint arXiv:2306.04542*, 2023. doi: 10.48550/arXiv.2306.04542 3

[5] H. Chefer, Y. Alaluf, Y. Vinker, L. Wolf, and D. Cohen-Or. Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models. *ACM Transactions on Graphics*, 42(4), article no. 148, 10 pages, jul 2023. doi: 10.1145/3592116 3

[6] K. Deja, A. Kuzina, T. Trzcinski, and J. Tomczak. On analyzing generative and denoising capabilities of diffusion-based deep generative models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds., *Advances in Neural Information Processing Systems*, vol. 35, pp. 26218–26229. Curran Associates, Inc., 2022. doi: 10.48550/arXiv.2206.00070 1, 3, 7, 9

[7] K. Deja, T. Trzciński, and J. M. Tomczak. Learning data representations with joint diffusion models. In D. Koutra, C. Plant, M. Gomez Rodriguez, E. Baralis, and F. Bonchi, eds., *Machine Learning and Knowledge Discovery in Databases: Research Track*, pp. 543–559. Springer Nature Switzerland, Cham, 2023. doi: 10.1007/978-3-031-43415-0_32 7, 9

[8] Y. Fang, H. Xu, and J. Jiang. A survey of time series data visualization research. *IOP Conference Series: Materials Science and Engineering*, 782(2):022013, mar 2020. doi: 10.1088/1757-899X/782/2/022013 3

[9] M. Hahsler, M. Piekenbrock, and D. Doran. dbscan: Fast density-based clustering with r. *Journal of Statistical Software*, 91(1):1–30, 2019. doi: 10.18637/jss.v091.i01 5

[10] A. Hertz, R. Mokady, J. Tenenbaum, K. Aberman, Y. Pritch, and D. Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022. doi: 10.48550/arXiv.2208.01626 3

[11] A. Hinterreiter, C. Steinparz, M. SchÖfl, H. Stitz, and M. Streit. Projection path explorer: Exploring visual patterns in projected decision-making paths. *ACM Trans. Interact. Intell. Syst.*, 11(3–4), article no. 22, 29 pages, sep 2021. doi: 10.1145/3387165 3

[12] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, eds., *Advances in Neural Information Processing Systems*, vol. 33, pp. 6840–6851. Curran Associates, 2020. doi: 10.48550/arXiv.2006.11239 2

[13] M. Kahng, N. Thorat, D. H. Chau, F. B. Viégas, and M. Wattenberg. Gan lab: Understanding complex deep generative models using interactive visual experimentation. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):310–320, 2019. doi: 10.1109/TVCG.2018.2864500 3

[14] T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022. doi: 10.48550/arXiv.2206.00364 1, 3, 9

[15] S. Kaski, J. Nikkilä, M. Oja, J. Venna, P. Törönen, and E. Castrén. Trustworthiness and metrics in visualizing similarity of gene expression. *BMC bioinformatics*, 4:1–13, 2003. doi: 10.1186/1471-2105-4-48 6

[16] M. Kwon, J. Jeong, and Y. Uh. Diffusion models already have a semantic latent space. In *The Eleventh International Conference on Learning Representations*, 2023. doi: 10.48550/arXiv.2210.10960 3

[17] S. Lee, B. Hoover, H. Strobelt, Z. J. Wang, S. Peng, A. Wright, K. Li, H. Park, H. Yang, and D. H. Chau. Diffusion explainer: Visual explanation for text-to-image stable diffusion. *arXiv preprint arXiv:2305.03509*, 2023. doi: 10.48550/arXiv.2305.03509 2, 3, 8

[18] M. Li, Z. Zhao, and C. Scheidegger. Visualizing neural networks with the grand tour. *Distill*, 5(3):e25, 2020. 3

[19] S. Luccioni, C. Akiki, M. Mitchell, and Y. Jernite. Stable bias: Evaluating societal representations in diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023. doi: 10.48550/arXiv.2303.11408 3

[20] Q. Luo, L. Christino, F. V. Paulovich, and E. Milios. Dimenfix: A novel meta-dimensionality reduction method for feature preservation. *arXiv preprint arXiv:2211.16752*, 2022. doi: 10.48550/arXiv.2211.16752 3

[21] L. McInnes, J. Healy, N. Saul, and L. Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018. doi: 10.21105/joss.00861 2, 9

[22] L. Meng, S. van den Elzen, N. Pezzotti, and A. Vilanova. Class-constrained

t-sne: Combining data features and class probabilities. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):164–174, 2024. doi: 10.1109/TVCG.2023.3326600 6

[23] A. Q. Nichol and P. Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pp. 8162–8171. PMLR, 2021. doi: 10.48550/arXiv.2102.09672 2

[24] A. Q. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. Mcgrew, I. Sutskever, and M. Chen. GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models. In *Proceedings of the 39th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2022. doi: 10.48550/arXiv.2112.10741 1, 2, 5, 7, 9

[25] J.-H. Park, Y.-J. Ju, and S.-W. Lee. Explaining generative diffusion models via visual analysis for interpretable decision-making process. *Expert Systems with Applications*, 248:123231, 2024. doi: 10.1016/j.eswa.2024.123231 2, 3, 9

[26] Y.-H. Park, M. Kwon, J. Choi, J. Jo, and Y. Uh. Understanding the latent space of diffusion models through the lens of riemannian geometry. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, eds., *Advances in Neural Information Processing Systems*, vol. 36, pp. 24129–24142. Curran Associates, Inc., 2023. doi: 10.48550/arXiv.2307.12868 1, 2, 3, 7

[27] G. Parmar, K. Kumar Singh, R. Zhang, Y. Li, J. Lu, and J.-Y. Zhu. Zero-shot image-to-image translation. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, article no. 11, 11 pages. Association for Computing Machinery, New York, NY, USA, 2023. doi: 10.1145/3588432.3591513 3

[28] N. Pezzotti, J.-D. Fekete, T. Höllt, B. P. Lelieveldt, E. Eisemann, and A. Vilanova. Multiscale visualization and exploration of large bipartite graphs. In *Computer Graphics Forum*, vol. 37, pp. 549–560. Wiley Online Library, 2018. doi: 10.1111/cgf.13441 3

[29] N. Pezzotti, T. Höllt, B. Lelieveldt, E. Eisemann, and A. Vilanova. Hierarchical stochastic neighbor embedding. In *Computer Graphics Forum*, vol. 35, pp. 21–30. Wiley Online Library, 2016. doi: 10.1111/cgf.12878 9

[30] N. Pezzotti, T. Höllt, J. Van Gemert, B. P. Lelieveldt, E. Eisemann, and A. Vilanova. Deepeyes: Progressive visual analytics for designing deep neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):98–108, 2018. doi: 10.1109/TVCG.2017.2744358 3

[31] N. Pezzotti, B. P. F. Lelieveldt, L. v. d. Maaten, T. Höllt, E. Eisemann, and A. Vilanova. Approximated and user steerable tsne for progressive visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 23(7):1739–1752, 2017. doi: 10.1109/TVCG.2016.2570755 2, 9

[32] N. Pezzotti, J. Thijssen, A. Mordvintsev, T. Höllt, B. Van Lew, B. P. Lelieveldt, E. Eisemann, and A. Vilanova. Gpgpu linear complexity t-sne optimization. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1172–1181, 2020. doi: 10.1109/TVCG.2019.2934307 2, 9

[33] V. Prasad, C. Zhu-Tian, A. Vilanova, H. Pfister, N. Pezzotti, and H. Strobelt. Unraveling the temporal dynamics of the unet in diffusion models, 2024. doi: 10.48550/arXiv.2312.14965 1, 3, 7, 9

[34] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021. doi: 10.48550/arXiv.2103.00020 5, 6, 7, 8, 9

[35] P. E. Rauber, S. G. Fadel, A. X. Falcão, and A. C. Telea. Visualizing the hidden activity of artificial neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):101–110, 2017. doi: 10.1109/TVCG.2016.2598838 3

[36] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10684–10695, June 2022. doi: 10.1109/CVPR52688.2022.01042 2, 7, 8, 9

[37] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds., *Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241. Springer, Cham, 2015. doi: 10.1007/978-3-319-24574-4_28 1, 2

[38] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 618–626, 2017. doi: 10.1109/ICCV.2017.74 3

[39] C. Si, Z. Huang, Y. Jiang, and Z. Liu. Freeu: Free lunch in diffusion u-net.

arXiv preprint arXiv:2309.11497, 2023. doi: 10.48550/arXiv.2309.11497 1, 3, 7, 9

[40] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. PMLR, 2015. doi: 10.48550/arXiv.1503.03585 2

[41] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. doi: 10.48550/arXiv.2011.13456 2, 3

[42] R. Tang, L. Liu, A. Pandey, Z. Jiang, G. Yang, K. Kumar, P. Stenetorp, J. Lin, and F. Ture. What the DAAM: Interpreting stable diffusion using cross attention. In A. Rogers, J. Boyd-Graber, and N. Okazaki, eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pp. 5644–5659. Association for Computational Linguistics, Toronto, Canada, July 2023. doi: 10.18653/v1/2023.acl-long.310 3

[43] N. Tumanyan, M. Geyer, S. Bagon, and T. Dekel. Plug-and-play diffusion features for text-driven image-to-image translation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1921–1930, 2023. doi: 10.1109/CVPR52729.2023.00191 3

[44] A. Vahdat, K. Kreis, and J. Kautz. Score-based generative modeling in latent space. *Advances in neural information processing systems*, 34:11287–11302, 2021. doi: 10.48550/arXiv.2106.05931 2, 3

[45] L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. 2, 4, 5, 6, 9

[46] M. Videau, N. Knizev, A. Leite, M. Schoenauer, and O. Teytaud. Interactive latent diffusion model. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '23, 11 pages, p. 586–596. Association for Computing Machinery, New York, NY, USA, 2023. doi: 10.1145/3583131.3590471 3

[47] Y. Vinker, E. Pajouheshgar, J. Y. Bo, R. C. Bachmann, A. H. Bermano, D. Cohen-Or, A. Zamir, and A. Shamir. Clipasso: semantically-aware object sketching. *ACM Transactions on Graphics*, 41(4), article no. 86, 11 pages, jul 2022. doi: 10.48550/arXiv.2202.05822 9

[48] Z. J. Wang, R. Turko, O. Shaikh, H. Park, N. Das, F. Hohman, M. Kahng, and D. H. Polo Chau. Cnn explainer: Learning convolutional neural networks with interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1396–1406, 2021. doi: 10.1109/TVCG.2020.3030418 3

[49] Q. Wu, Y. Liu, H. Zhao, A. Kale, T. Bui, T. Yu, Z. Lin, Y. Zhang, and S. Chang. Uncovering the disentanglement capability in text-to-image diffusion models. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1900–1910. IEEE Computer Society, Los Alamitos, CA, USA, jun 2023. doi: 10.1109/CVPR52729.2023.00189 1, 9

[50] 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 Surveys*, 56(4), article no. 105, 39 pages, nov 2023. doi: 10.1145/3626235 1, 2

[51] T. Yang, Y. Wang, Y. Lu, and N. Zheng. Disdiff: Unsupervised disentanglement of diffusion probabilistic models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. doi: 10.48550/arXiv.2301.13721 1, 7, 8, 9

[52] Q. Zhang, M. Tao, and Y. Chen. gDDIM: Generalized denoising diffusion implicit models. In *The Eleventh International Conference on Learning Representations*, 2023. doi: 10.48550/arXiv.2206.05564 2
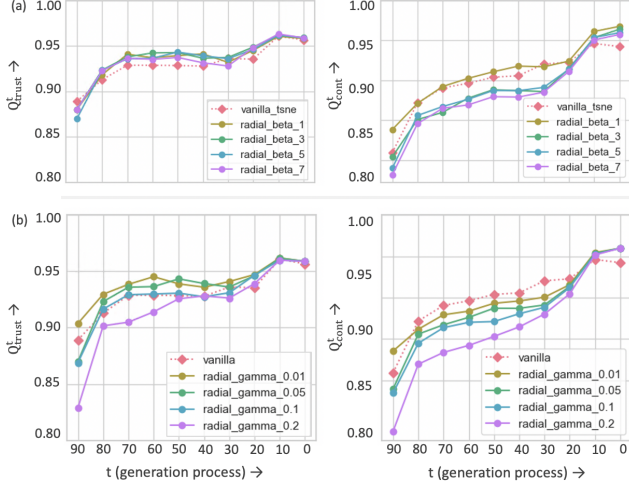
Fig. 11: Exploring neighborhood preservation with Q_{trust}^t and Q_{cont}^t for the proposed radial layout across (a) different β values with $\alpha = 1$, $\gamma = 0.05$, and (b) different γ values with $\alpha = 1$, $\beta = 5$. Embeddings are derived from $\widehat{\mathbf{h}}_{t-1}^i$. While Q_{trust}^t across various β values in (a) appear comparable, Q_{cont}^t exhibits noticeable variations. Despite $\beta = 1$ yielding the best Q_{cont}^t values, it resulted in poor visual separability between iterations shown in Fig. 15. Conversely, $\beta \in 3, 5$ are comparable in preserving high-dimensional neighborhoods. Hence, $\beta = 5$ was selected for improved iteration separation seen in Fig. 15. Further, Q_{cont}^t and Q_{trust}^t in (b) display degradation with $\gamma = 0.2$, while $\gamma \in 0.01, 0.05$ remain comparable in preserving neighborhoods. We hence choose $\gamma = 0.05$ since lower values result in insufficient alignment.
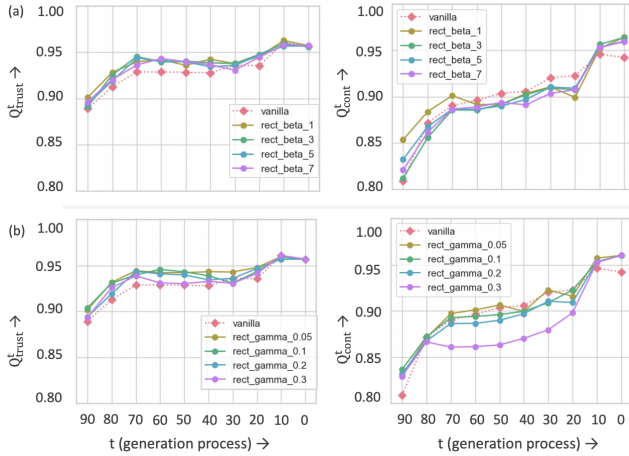

Fig. 13: Exploring the hyperparameter spacing s used to set the ring offsets ($\bar{r}_t$) per iteration, t, where $\bar{r}_t = (T - t) * s$. Here, histograms illustrate the distribution of polar radii r_t^i for radial embedding points $\mathbf{l}_t^i$, projected from $\widehat{\mathbf{h}}_t^i$ with $\alpha = 1$, $\beta = 5$, and $\gamma = 0.05$. Each histogram is color-coded by its corresponding iteration t, with T representing noise. A clear separation between iterations is observed when $s = 20$, prompting us to use this value for experiments. However, excessively large values of s run the risk of insufficient space allocation for early time steps. We observe similar insights for the rectilinear layout at $s = 20$; hence we use the same value.

Fig. 12: Exploring neighborhood preservation with Q_{trust}^t and Q_{cont}^t for the proposed rectilinear layout across (a) different β values with $\alpha = 1$, $\gamma = 0.2$, and (b) different γ values with $\alpha = 1$, $\beta = 5$. Embeddings are derived from $\widehat{\mathbf{h}}_{t-1}^i$. Q_{trust}^t and Q_{cont}^t across various β (a) are comparable. However, in (b), Q_{cont}^t and Q_{trust}^t show degradation, especially when $\gamma = 0.3$. Lower γ values preserve neighborhoods similarly. In Fig. 16, both $\gamma = 0.2$ and $\gamma = 0.3$ led to good alignments; however, the neighborhood preservation deterioration with $\gamma = 0.3$ prompts our choice of $\gamma = 0.2$.
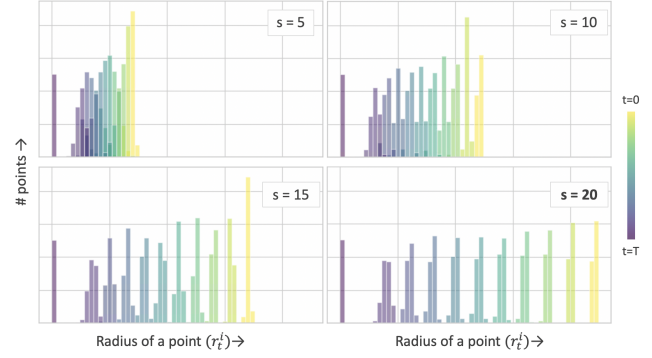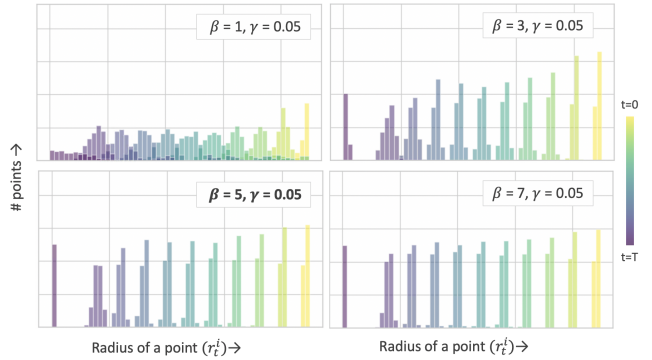

Fig. 14: Exploring the iteration separation across β values for the radial layout more systematically. Histograms are used to illustrate the distribution of polar radii r_t^i for radial embedding points $\mathbf{l}_t^i$, projected from $\widehat{\mathbf{h}}_t^i$ with $\alpha = 1$, $\gamma = 0.05$, and $s = 20$. A clear separation between iterations is observed when $\beta = 5$. The same insights are used for the rectilinear layout at $\beta = 5$. We use the same value for both the radial and rectilinear layouts.
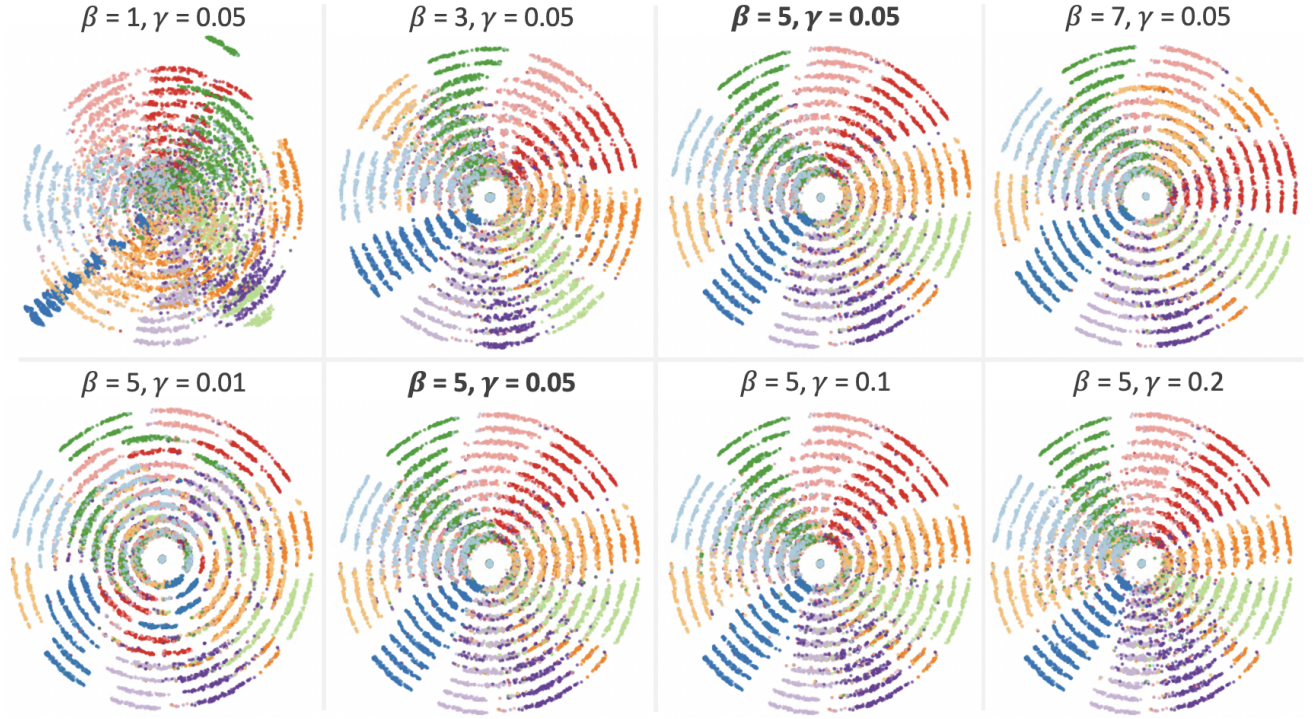
Fig. 15: Exploring different β and γ weightings for the radial layout losses C_d^p and C_a^p respectively, with $\alpha = 1$. When $\beta = 1$, the iterations are poorly separated. Increasing $\beta = 3$ enhances separation, yet inner clusters remain insufficiently distinguished. While $\beta \in \{5,7\}$, the iterations display well-defined separation. $\beta = 5$ is chosen to maintain the t-SNE cluster structure while meeting our objectives. Similarly, with $\gamma = 0.01$, points demonstrate poor alignment, hindering the exploration of data evolution. Increasing γ values of 0.05 and 0.1 leads to sufficient alignment. $\gamma = 0.05$ is selected based on the above insights and neighborhood quality metrics. All embeddings in this document are on the ImageNet objects case, where the noisy representations $\widehat{h}_t^i$ are first encoded with a classifier.
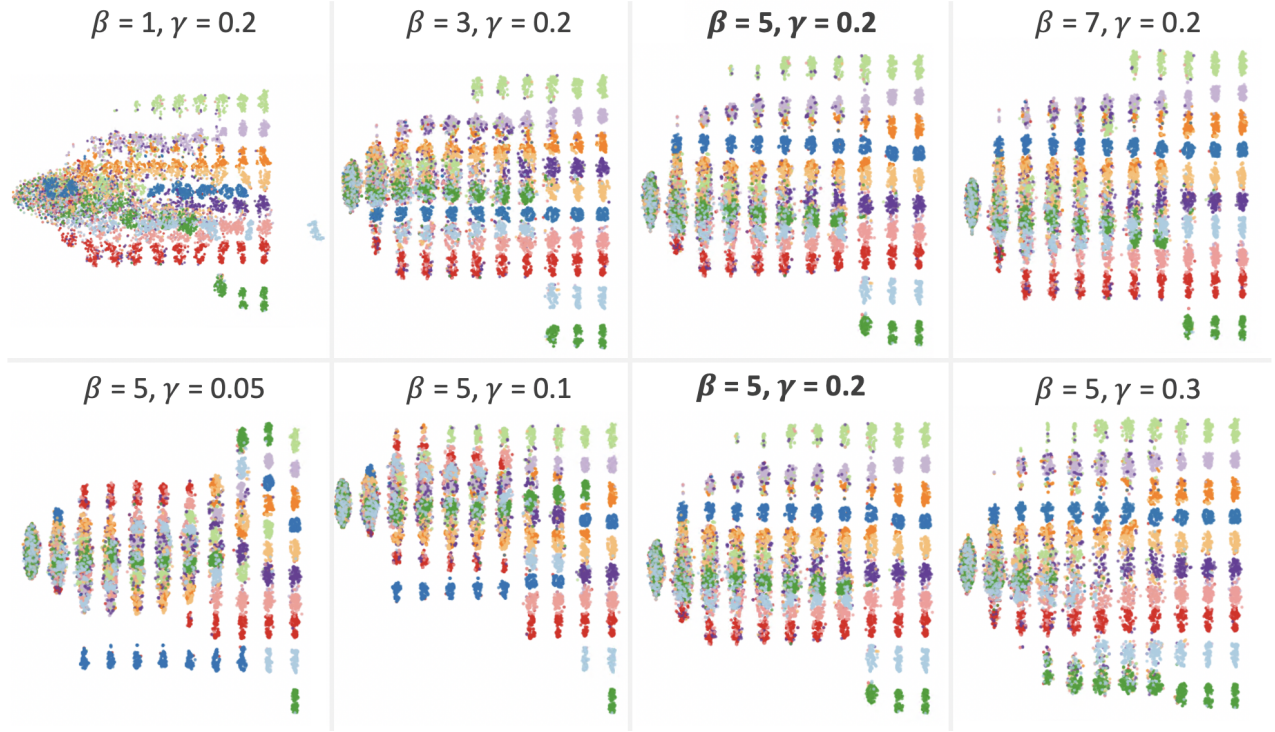

Fig. 16: Exploring different β and γ weightings for the rectilinear layout losses C_d^c and C_a^c respectively, with $\alpha = 1$. Similar to previous experiments, for $\beta \in \{1,3\}$, the iterations exhibit poor separation, with unclear links between points and iteration steps. A minimum value of $\beta = 5$ is necessary to achieve sufficient separation and is hence selected. $\gamma = 0.05$ and $\gamma = 0.1$ result in inadequate alignment of clusters across iterations. While $\gamma = 0.2$ and $\gamma = 0.3$ produce satisfactory results, we choose $\gamma = 0.2$ to minimize changes to the t-SNE layout while meeting our goals to study data evolution.

Fig. 17: Our proof of concept front-end enables analysis of the evolutionary embeddings produced by TDL. At the center (a) is our proposed evolutionary embedding, which is the outcome of TDL. The above shows the radial layout, but the rectilinear one can be retrieved with the buttons on the top-right of (a). Images corresponding to points at every n degree are displayed along the outer ring of the layout (a). A control panel on the left (b) supports data filtering, visualizing, and coloring options. Selected images are displayed on the right (c) in an image viewer to enable understanding of the embedding. Points can be filtered based on the iteration t, the path length (b.1), or prompt keywords (b.4). Some visualization options, such as deciding the number of images to be displayed on the outer ring, the smoothness of the paths, and the interpolation factor to group paths of a cluster, are provided (b.2). Points in the embedding can be encoded by adjustable colors based on the prompt or a keyword within a prompt (b.4). Lastly, images in the image viewer can be grouped by time step or color code to enable analysis (b.3).