Directional Noise Conditioning for Diffusion Models

Mahdi Shafiei*
Technical University of Munich
The University of Queensland
mahdi.shafiei@uq.edu.au

Azade Farshad*
ELLIS Institute Finland
Aalto University
Munich Center for Machine Learning
azade.farshad@aalto.fi

Nassir Navab
Technical University of Munich
Munich Center for Machine Learning
ELLIS Unit Munich
nassir.navab@tum.de

Yousef Yeganeh
Aalto University
Technical University of Munich
Munich Center for Machine Learning
yousef.yeganeh@aalto.fi

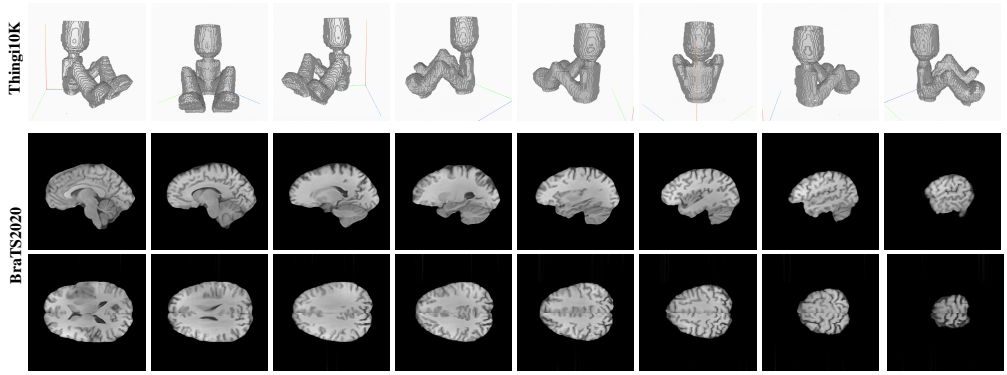


Figure 1: **Directional Noise Conditioning for 3D Generation using 2D Diffusion**

Abstract

Conditioning diffusion models on visual features is essential for controlled image generation and counterfactual reasoning across diverse applications, from creative design to medical imaging. Existing conditioning methods, such as timestep embedding and cross-attention mechanisms, suffer from limited control over feature accuracy, require substantial architectural modifications, and struggle to maintain conditioning precision throughout the denoising process, as the model's focus shifts toward fine-grained details in later steps rather than the desired conditioned features. We present **D**irectional **N**oise **C**onditioning (**DNC**), a novel approach that addresses these limitations by learning to shift the initial noise distribution in the latent space. **DNC** assigns each visual feature a distinct direction in the latent space and applies proportional shifts during both training and inference, enabling precise control over continuous and discrete visual attributes. We introduce a combined loss function that explicitly enforces shift accuracy alongside denoising, ensuring consistent conditioning across all denoising steps. Furthermore, we extend this framework to datasets with hidden or inaccessible visual features by employing a Variational Autoencoder to extract latent representations in an unsupervised manner. This enables high-fidelity counterfactual generation on complex, real-world datasets where explicit feature annotations are unavailable or prohibitively expensive to obtain. **Project Page:** https://sites.google.com/view/dncdiff

*Equal Contribution

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Generative diffusion models [1, 2] have become the state-of-the-art in high-fidelity image synthesis, largely supplanting GANs [3]. Their success has spurred a demand for precise, steerable control over the generative process. Current conditioning methods, however, fail to provide robust control due to the high variation of sampled noise. Approaches such as timestep embedding [1] and cross-attention [4], while powerful for broad semantic guidance (e.g., text prompts), lack directional control over the denoising process. The conditioning signal is merely *injected* at intermediate steps, with no guarantee of how, or even if, the model will faithfully adhere to it. This limitation is a critical failure point. As noted in recent work [5, 6], existing methods suffer from *conditioning decay*: the model's focus on the conditioning signal is strongest in the early denoising steps. In the later steps, the network prioritizes generating fine-grained texture and high-frequency details, often at the expense of the original conditioning feature. This makes precise, continuous control over an attribute— such as a specific 3D rotation or a quantitative medical marker—an unsolved problem.

In this work, we propose a new conditioning paradigm that operates at a more fundamental level: *we directly condition the diffusion model on the sampled noise distribution* x_T, to map visual features to a structured, feature-specific adjustment in the sampled Gaussian noise rather than post manipulation of the intermediate features. To achieve this, we map each visual feature f_j to a unique, learned, and orthogonal direction d_j in the latent space. The final conditioning shift S is a linear combination of the direction vectors d_j, scaled by their corresponding mean feature values $\overline{f_j}$: $S = \sum_j \overline{f_j} d_j$. The reverse process then starts from $x_T = S + \epsilon_\perp$, rather than from pure noise ϵ, where $\epsilon_\perp$ is a noise component orthogonal to the conditioning subspace. By starting the generation from this structured point in the latent space, the conditioning is embedded from $t = T$ throughout the whole denoising process. In addition, we introduce an auxiliary loss $\mathcal{L}_{\text{shift}}$ to explicitly enforce the network's prediction corresponds to the conditioning signal via the shift component S at *every* timestep. This ensures the model remains aware of the target features throughout the entire generative process.

Furthermore, we demonstrate the framework's flexibility, showing that the features f_j are not limited to annotated labels and can represent arbitrary conditions. By employing a *Feature VAE*, we first *discover* unannotated latent features—such as the unique anatomical "identity" of a brain from a single 2D slice—and feed these discovered features into our core conditioning mechanism. This unifies supervised, quantitative control and unsupervised, disentangled generation into a single, cohesive framework. Our method learns to associate visual features with specific directional vectors in the latent space, enabling precise control during generation as demonstrated in fig. 1.

To summarize, our contributions are: (1) A novel conditioning mechanism that steers diffusion models by mapping visual features to specific, orthogonal shifts in the sampled noise distribution x_T. (2) An auxiliary shift loss, $\mathcal{L}_{\text{shift}}$, that enforces conditioning consistency at every timestep, solving the "conditioning decay" problem prevalent in other methods. (3) A unified framework that integrates a Feature VAE to discover and control unannotated latent features, enabling high-fidelity 3D-aware generation and reconstruction from 2D slices. (4) Outstanding results on a diverse set of tasks, including a **51-59%** reduction in L1 error for precise angle control, and an order-of-magnitude improvement in 3D reconstruction (e.g., **0.75 IoU** vs. 0.02 from baselines on Thingi10K).

2 Related Work

Diffusion Models While diffusion models [2, 1] are significantly slower than single-pass GANs [3, 7, 8, 9] due to their iterative denoising process, they produce higher-fidelity images [10, 11, 12] and videos [13, 14, 15], making them the dominant generative models in recent years. This iterative generation relies on a sampling strategy with two primary approaches: DDPM [1] follows a stochastic Markovian process that requires many steps, while DDIM [16] employs a non-Markovian deterministic approach that enables much faster sampling. Latent diffusion models (LDMs) [4] mitigate the computational demands of diffusion in pixel-space by functioning within a compressed latent space.

Conditioning Diffusion Models To make diffusion models steerable, they are often conditioned on external information such as class labels [17] or classifier-free guidance [18], images (for tasks like inpainting or super-resolution), or, most notably, text prompts [19, 20]. This control enables targeted generation, such as creating an image from a description. Cross-attention layers [4] inject conditioning

signals into intermediate representations, although recent work reveals that cross-attention primarily influences early denoising steps [5, 6]. Adaptive normalization techniques like FiLM [21] modulate feature activations, while timestep embedding concatenation [1] provides simpler alternatives with reduced architectural overhead.

3D from 2D Generation Diffusion models have been increasingly extended to 3D-aware synthesis, often by leveraging powerful pre-trained 2D models. DreamFusion [22] introduced score distillation sampling to guide the generation of a 3D representation using a 2D diffusion model. Building on this, many methods focus on generating geometrically consistent multi-view images, such as MVDream [23] and Wonder3D [24], which fine-tune 2D models to jointly generate consistent views. Other works focus on explicitly adapting or re-purposing 2D models for 3D-aware image generation [25] or 3D asset generation [26]. A recent trend combines the strengths of both 2D and 3D priors, as seen in Gen-3Diffusion [27] and other bidirectional diffusion methods [28]. Specific strategies continue to evolve, including sparse-view reconstruction (Instant3D [29]), one-step distillation (GECO [30]), and multi-view latent generation (CAT3D [31]).

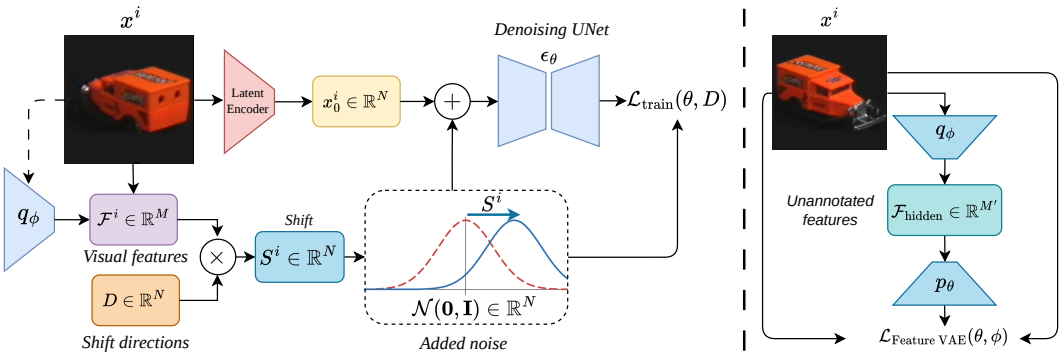
Generation in Medical Imaging In specialized domains such as medicine, conditional diffusion is used to model disease progression [32] and to detect anomalies via counterfactual generation of healthy states [33]. DiME [34] and DVCE employ classifier guidance during the diffusion process to generate counterfactual explanations. Latent Drifting [35] manipulates the latent trajectory during reverse diffusion, shifting the mean of the reverse-process transition kernel, to generate high-fidelity medical counterfactuals without explicit feature annotations. Δ-LFM [36] aligns same-patient trajectories along a shared latent direction whose magnitude increases monotonically with disease severity, providing independent support for a directional–proportional encoding of features. In contrast, DNC imposes this structure on the terminal noise through a learned, feature-specific orthogonal basis and trains the denoiser on this shifted target, which mitigates the loss of attribute control over the denoising trajectory.

3 Method

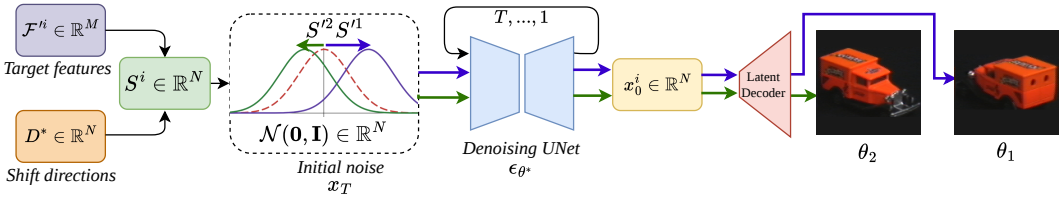
We propose Directional Noise Conditioning (DNC), a novel framework for conditioning Diffusion Models through a learned offset in the sampled noise distribution. This offset is a learned function that corresponds to the desired visual features. This offset $S \in \mathbb{R}^n$ is a linear combination of learned orthogonal basis vectors $d_j \in \mathbb{R}^n$, each corresponding to a single semantic feature. The framework is applied to both explicitly annotated and unannotated features shared across a group of data points (e.g., the unique anatomy of a brain, which is shared across all its 2D slices), as well as to latent features discovered in the VAE's feature space. Our framework is designed to be highly flexible and consists of two key components: (1) **Directional Noise Conditioning** (section 3.1), which conditions the diffusion process on a given set of visual features $\mathcal{F}$. It learns a unique, orthogonal direction d_j in the noise space for each feature f_j. The magnitude of the shift along this direction is modulated by the feature's scalar value $\overline{f_j}$, allowing both continuous (*e.g.*, rotation) and discrete (*e.g.*, class) features to be handled in a unified manner. (2) **Feature Discovery Module** (section 3.2), which addresses cases where features are latent or unannotated. We introduce a *Feature VAE* to extract these unannotated features $\mathcal{F}_{\text{unannotated}}$ from the data. During training, the discovered latent features are simply concatenated with any available annotated features ($\mathcal{F} = \{\mathcal{F}_{\text{annotated}}, \mathcal{F}_{\text{unannotated}}\}$) and fed into the DNC conditioning mechanism. This allows our single framework to learn from any combination of known and unknown visual factors. An overview of these components is illustrated in fig. 2.

3.1 Directional Noise Conditioning

Problem Formulation Consider a dataset of 2D images $\mathcal{X} = \{x^i\}_{i=1}^N$. We associate each image x^i with a general set of M visual features $\mathcal{F}^i = \{f_j^i\}_{j=1}^M$. These features can be any measurable property, such as an annotated label (e.g., rotation angle θ^i) or a discovered latent factor. For example, in a dataset of rotated arrows, each image x^i is described by $M = 1$ and $f_1^i = \theta^i$, as shown in fig. 2. We define the diffusion latent space as $\mathbb{R}^n$. Our goal is to guide the generation process along specific directions in this space that correspond to interpretable changes in visual properties. To achieve this, we assign each visual feature f_j a unique shift direction, represented by a unit vector $d_j \in \mathbb{R}^n$.



(a) **Training** We condition the diffusion model on the visual (either annotated or unannotated) features $\mathcal{F}^i$ by shifting the added noise during training. The shift S^i for the data point x^i is calculated from a linear combination of its features $\mathcal{F}^i$ and the corresponding shift directions D. The diffusion model ϵ_θ and shift directions D are trained using the loss functions $\mathcal{L}_{\text{denoise}}$ and $\mathcal{L}_{\text{shift}}$, to minimize the difference between the predicted added noise and the actual added noise. **Left:** the overall training scheme for both annotated and unannotated features. The dashed line from x^i to q_ϕ shows the unannotated feature discovery. **Right:** Training of the unannotated feature discovery module.



(b) **Inference** The target visual features $\mathcal{F}'$ and the trained shift directions D^* are then used to compute the added shift S' to the initial random noise. Subsequently, the denoising process with the trained UNet ϵ_{θ^*} is performed for T steps until the final image is obtained.

Figure 2: **Overview of the Model Training and Inference.**

Training We encode the visual features $\mathcal{F}^i$ of a data point x^i into a single shift vector:

$$S^i = \sum_{j=1}^{M} \overline{f_j^i}\, d_j, \tag{1}$$

, where $\overline{f_j^i}$ is the normalized scalar form of the feature f_j^i, linearly mapped to the range $[-1, 1]$ for consistent scaling. The direction vectors $d_j \in \mathbb{R}^n$ are initialized from $\mathcal{N}(0, I)$ and orthogonalized using the Gram–Schmidt process to ensure linear independence, such that $d_j \cdot d_k = 0$ for all $j \neq k$. We modify the standard forward diffusion process to incorporate this feature-dependent shift. For a given noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we first find its component $\epsilon_\perp$ orthogonal to the shift S^i:

$$\epsilon_\perp = \epsilon - \frac{\epsilon \cdot S^i}{\|S^i\|^2} S^i. \tag{2}$$

This prevents the random noise from interfering with the conditioning shift. Note that eq. (2) removes only the component of ϵ along the combined shift S^i, not along the full span of the directions $\{d_j\}$; for $S^i \neq 0$, $\epsilon_\perp$ is therefore Gaussian with covariance $I - S^i (S^i)^\top / \|S^i\|^2$ of rank $n - 1$. When $S^i = 0$, we set $\epsilon_\perp = \epsilon$. The same construction is used at inference (eq. (12)). The noisy sample at timestep t is then:

$$x_t^i = \sqrt{\bar{\alpha}_t}\, x_0^i + \sqrt{1 - \bar{\alpha}_t}\, (\epsilon_\perp + S^i). \tag{3}$$

The denoising network $\epsilon_\theta(x_t^i, t)$ is trained to predict the total "noise" $\epsilon_\perp + S^i$. The main denoising objective is:

$$\mathcal{L}_{\text{denoise}}(\theta, D) = \mathbb{E}_{x_0^i, \epsilon, t} \left[\|\epsilon_\perp + S^i - \epsilon_\theta(x_t^i, t)\|^2 \right]. \tag{4}$$

To ensure the network accurately learns the feature-dependent shift S^i and doesn't just focus on the random noise $\epsilon_\perp$, we introduce an auxiliary loss. We first project the network's prediction onto the shift direction:

$$\overline{S^i_{t,\theta}} = \frac{S^i \cdot \epsilon_\theta(x^i_t, t)}{\|S^i\|^2}\, S^i. \tag{5}$$

This auxiliary loss then explicitly encourages accurate prediction of the shift:

$$\mathcal{L}_{\text{shift}}(\theta, D) = \mathbb{E}_{x^i_0,\epsilon,t}\left[\|\overline{S^i_{t,\theta}} - S^i\|^2\right]. \tag{6}$$

Additionally, we include a regularization loss to preserve orthogonality between the shift directions:

$$\tilde{d}_j = \frac{d_j}{\|d_j\|}, \quad j = 1, \ldots, M, \tag{7}$$

$$G = \tilde{D}\tilde{D}^\top, \tag{8}$$

$$\mathcal{L}_{\text{ortho}}(D) = \|G - I\|^2_F. \tag{9}$$

Finally, the total training objective becomes:

$$\mathcal{L}_{\text{train}}(\theta, D) = \mathcal{L}_{\text{denoise}} + \mathcal{L}_{\text{shift}} + \mathcal{L}_{\text{ortho}}, \tag{10}$$

which we minimize to find the optimal parameters (θ^*, D^*).

Inference To generate an image conditioned on a desired set of target features $\mathcal{F}' = \{f'_j\}_{j=1}^M$, we first construct the corresponding target shift:

$$S' = \sum_{j=1}^{M} \overline{f'_j}\, d_j. \tag{11}$$

We then initialize the process from a noise vector x_T that incorporates this shift while preserving an orthogonal random component:

$$\epsilon_\perp = \epsilon - \frac{\epsilon \cdot S'}{\|S'\|^2}\, S', \quad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{12}$$

$$x_T = \epsilon_\perp + S'. \tag{13}$$

The standard reverse diffusion process is then applied, with $\epsilon_\theta(x_t, t)$ progressively denoising x_T to x_0 while preserving the conditioning encoded in the initial shift:

$$\hat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}, \qquad x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\,\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-1}}\,\epsilon_\theta(x_t, t), \tag{14}$$

The shift can additionally be scaled by a scalar λ at inference without retraining; we analyze this and its relation to classifier-free guidance in appendix E.

3.2 Discovering Unannotated Features

Problem Formulation For complex datasets, visual features may be latent or unannotated. We often encounter data $\mathcal{X} = \{\mathcal{X}^i\}_{i=1}^N$, where each subset $\mathcal{X}^i = \{x^{i,k}\}_{k=1}^L$ contains data points that share the *same* unannotated features $\mathcal{F}^i_{\text{unannotated}}$. For example, in a brain MRI dataset, $\mathcal{X}^i$ could be all 2D slices from a single subject, sharing unannotated features like tissue structure or pathology. Each individual slice $x^{i,k}$ may also have its own *annotated* features $\mathcal{F}^{i,k}_{\text{annotated}}$, such as slice depth.

Feature VAE for Latent Discovery To capture the shared $\mathcal{F}^i_{\text{unannotated}}$, we introduce a *Feature VAE*. This VAE is trained to learn a low-dimensional representation $\mathcal{F}^i_{\text{unannotated}} = \{f^i_{\text{unannotated},j}\}_{j=1}^{M'}$ for each subset $\mathcal{X}^i$ by minimizing the following loss:

$$\begin{aligned}\mathcal{L}_{\text{Feature VAE}} = \mathbb{E}_{q_\phi(\mathcal{F}^i_{\text{unannotated}}|\overline{\mathcal{X}^i})}\Big[-\log p_\theta(\overline{\mathcal{X}^i} \mid \mathcal{F}^i_{\text{unannotated}})\Big] \\ + \text{KL}\Big(q_\phi(\mathcal{F}^i_{\text{unannotated}} \mid \overline{\mathcal{X}^i}) \,\Big\|\, p(\mathcal{F}_{\text{unannotated}})\Big).\end{aligned} \tag{15}$$

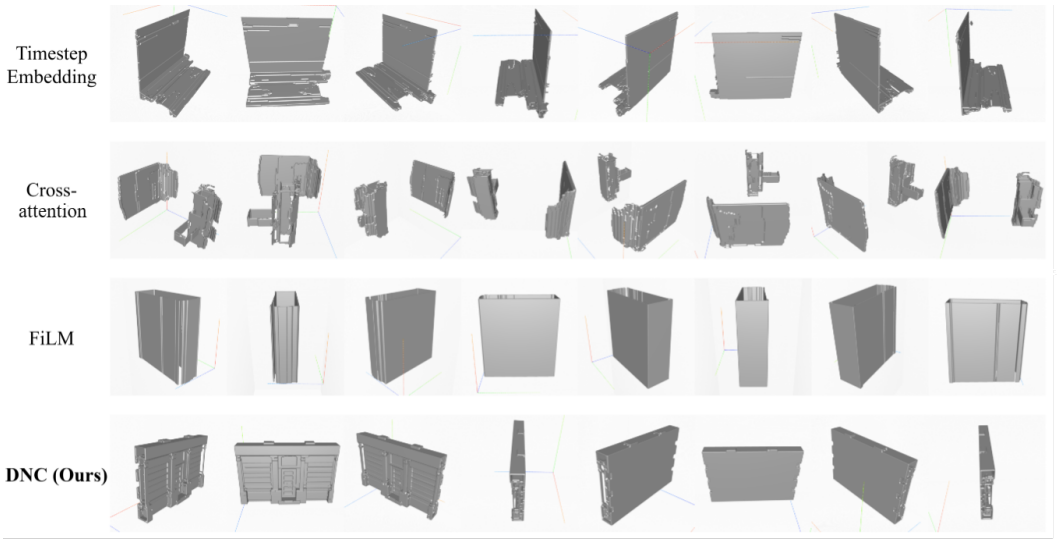


Figure 3: **Qualitative Comparison of Generated 3D Objects on *Thingi10K* [37]**. Each method is conditioned on the depth of 2D vertical slices during training. For visualization, the generated slices are stacked to reconstruct full 3D objects, which are then converted to meshes and rendered from multiple viewpoints.



Figure 4: **Results on *COIL-100 [38]***. The diffusion model is conditioned on the unannotated feature $\mathcal{F}_{\text{unannotated}}$ (object rotation angle), extracted using a Feature VAE. During inference, ${f'}_{\text{unannotated}}^{\ 1}$ is varied uniformly from -1 to 1 in each row, producing a smooth and consistent progression of rotation angles across the generated images.

Here, $\overline{\mathcal{X}^i}$ is a representative element of $\mathcal{X}^i$ (e.g., the middle slice), q_ϕ is the encoder, p_θ is the decoder, and the prior is $p(\mathcal{F}_{\text{unannotated}}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Unified Training and Inference The Feature VAE acts as a feature extractor. During the main diffusion model training, for each data point $x^{i,k}$, we form a **complete feature set** by concatenating its discovered unannotated features and its known annotated features $\mathcal{F} = \mathcal{F}^i_{\text{unannotated}} \cup \mathcal{F}^{i,k}_{\text{annotated}}$. This complete set $\mathcal{F}$ (with $M = M' + M_{\text{annotated}}$ total features) is then fed directly into the **DNC** as described in section 3.1, using eq. (1)–eq. (10). Inference follows the same logic. We sample a target unannotated feature $\mathcal{F}'_{\text{unannotated}} \sim p(\mathcal{F}_{\text{unannotated}})$ from the prior, specify a target annotated feature $\mathcal{F}'_{\text{annotated}}$, and combine them. The resulting $\mathcal{F}'$ is used to compute the target shift S' (eq. (11)), and the core inference process (eq. (14)) generates the final image.

4 Experiments and Results

We evaluate our method on two key tasks: (1) high-precision conditional generation where visual features are explicitly annotated, and (2) generation on complex datasets by discovering and conditioning on unannotated latent features. We demonstrate that our method achieves high-fidelity results and outperforms standard conditioning baselines in both settings.

Table 1: **Comparison of Conditioning Methods on the Arrows Dataset using Identical Diffusion Models**. The table reports the correlation coefficient r^2 and the average L1 error between the estimated and target rotation angles.

Conditioning Method	Uniform arrow			Grid-like arrow		
	r^2	L1 (°)	L1 (%)	r^2	L1 (°)	L1 (%)
Timestep embedding [48]	0.9944	2.96	1.64	0.9941	3.14	1.89
Cross-attention [4]	0.9969	2.50	1.39	0.9889	3.97	2.21
DNC (Ours)	**0.9993**	**1.21**	**0.67**	**0.9967**	**1.85**	**1.03**

Table 2: **Quantitative Evaluation of 3D Generation and Reconstruction on *Thingi10K* [37] dataset.** Comparison of our method with baseline conditioning approaches using both distribution-based metrics and reconstruction-based metrics.

Conditioning Method	Distribution-based		Target-based		
	FID ↓	KID ↓	MSE ↓	IoU ↑	Chamfer Distance ↓
Timestep embedding [48]	47.51	0.1524	0.0379	0.0196	15.3393
Cross-attention [4]	50.29	0.1102	0.0558	0.0219	13.4967
FiLM [49]	43.76	0.1442	0.0257	0.0043	25.1627
DNC (Ours)	**13.55**	**0.0571**	**0.0073**	**0.7486**	**0.2680**
Ground Truth	10.56	0.0427	–	–	–

4.1 Experimental Setup

Datasets We evaluate our method on four diverse datasets: **(a) Arrows (Ours):** A synthetic dataset of 2,000 images of arrows, used for controlled experiments. We use two variations: uniform white arrows and arrows with a black-and-white grid texture. **(b) COIL-100** [38]: 72 views per object, captured at 5° rotation intervals. **(c) Thingi10K** [37]: A large-scale dataset of 10,000 3D printing models. **(d) BraTS2020** [39, 40, 41, 42]: 3D brain MRI scans from multiple modalities.

Implementation All methods share one latent diffusion backbone and differ only in the conditioning mechanism. A latent autoencoder (10.7M parameters), trained on each dataset and frozen during diffusion training, maps 256×256 images to a $64 \times 64 \times 1$ latent, which is denoised by a 331.5M-parameter U-Net. Baselines are obtained by swapping only the injection mechanism on this backbone; details are given in appendix N.

Evaluation Metrics We evaluate the model with annotated features using *L1 error* and the *coefficient of determination* (r^2). For unannotated model generation, we measure fidelity and realism using Fréchet Inception Distance (FID) [43] and Kernel Inception Distance (KID) [44]. For 3D reconstruction tasks, we also report *Mean Squared Error (MSE)* [45], *Intersection over Union (IoU)* [46], and *Chamfer Distance (CD)* [47].

4.2 Results

Validation of the Model with Annotated Features

Controlled Experiment: Arrow Rotation To validate our method's precision, we train a diffusion model as described in section 3.1 on our synthetic *Arrows* dataset. This dataset contains 2,000 images of arrows uniformly rotated from $\theta = 0^\circ$ (rightward) to $\theta = 180^\circ$ (leftward). In this controlled setting, the rotation angle is the only annotated visual feature ($M = 1$, $f_1^i = \theta^i$). We define its scalar representation $\overline{f_1^i}$ as a linear mapping where $\theta = 0^\circ \mapsto 1$ and $\theta = 180^\circ \mapsto -1$. As shown in figs. 17a and 18, our method accurately generates arrows matching the target rotation θ_{target}, which corresponds to the feature scalar $\overline{f_1} = (90 - \theta_{\text{target}})/90$ in eq. (11). To confirm that conditioning preserves image fidelity, we repeat the experiment on arrows with a black-and-white grid texture. fig. 17b demonstrates that the arrows rotate according to the target degrees while the grid texture is preserved with high precision. All images in fig. 17 were generated from distinct random seeds.

Table 3: **Quantitative Results on *BraTS2020***. We report the fidelity metrics for both vertical and horizontal MRI slices generated using different conditioning methods. The horizontal slices are reconstructed by stacking the generated vertical slices.

Conditioning Method	Vertical				Horizontal			
	FID ↓	KID ↓	BRISQUE ↓	PIQE ↓	FID ↓	KID ↓	BRISQUE ↓	PIQE ↓
Timestep embedding [48]	76.38	0.1233	89.35	81.75	139.92	0.1386	101.71	90.46
Cross-attention [4]	110.36	0.1178	87.53	74.88	141.57	0.1628	100.12	94.42
FiLM [49]	100.99	0.2656	83.73	61.72	133.00	0.3110	76.79	87.19
DNC (Ours)	**57.41**	**0.0353**	**62.31**	**44.09**	**60.42**	**0.0396**	**55.44**	**54.25**
Ground Truth	55.08	0.0209	50.93	36.44	50.68	0.0212	50.49	37.72

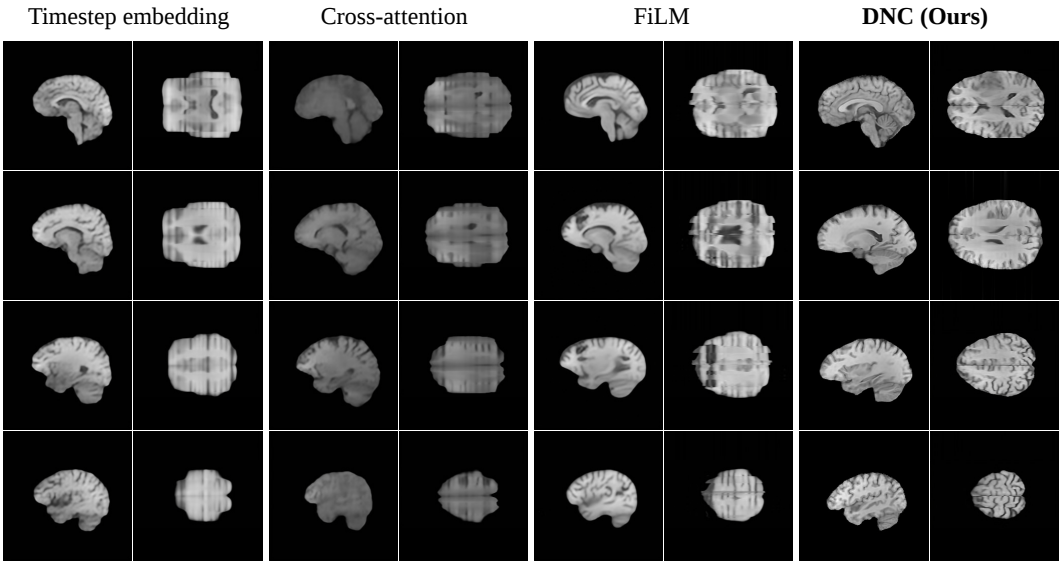


Figure 5: **Comparison of Conditioning Methods on *BraTS2020***. Each method conditions the diffusion model on the depth of 2D vertical brain MRI slices. For each method, the left column shows the generated vertical slices from the left to the middle of the brain, while the right column depicts the corresponding horizontal slices obtained by stacking the generated vertical slices and slicing them from the top to the middle.

Quantitative Evaluation To quantitatively assess the generations in fig. 17, we train a regression model to predict the rotation angle of an input arrow. The model achieves an average prediction error of $\pm 0.39^\circ$ on the test set, establishing a reliable metric. Using this model, we estimate the angles of our generated images and compare them to the target θ_{target}. For the uniform arrow experiment (fig. 17a), our method achieved $\mathbf{r^2 = 0.9993}$ with an average L1 error of $\pm\mathbf{1.21}^\circ$. For the grid-textured arrow experiment (fig. 17b), we obtained $\mathbf{r^2 = 0.9967}$ with an average L1 error of $\pm\mathbf{1.85}^\circ$. These results (table 1 and visualized in the supplement) confirm the high precision of our conditioning mechanism. We compare our method against two widely used conditioning approaches in table 1: *Conditioning via Timestep Embedding* [48] and *Cross-Attention Conditioning* [4]. All methods use identical diffusion models and experimental setups and are evaluated using the same angle prediction model described above. Our method significantly outperforms both baselines: it reduces the average L1 error by 59.1% (Uniform) and 41.1% (Grid-like) compared to *Timestep Embedding*, and by 51.6% (Uniform) and 53.4% (Grid-like) compared to *Cross-Attention*. Measuring the attribute error along the denoising trajectory further shows that the baselines degrade after mid-trajectory while DNC remains stable (appendix D).

Generation with Unannotated Features
We now demonstrate our model's ability to operate on complex datasets where key visual features are not annotated. Our method (*cf.* section 3.2) first discovers these features using Feature VAE and then conditions the diffusion model on the learned latent representations.

Discovering 1D Latent Features We first validate our method's ability to discover a 1D latent feature on the *COIL-100* dataset [38], which contains 72 images of each object rotated at 5° intervals (*cf.* fig. 2). We treat the rotation angle as an unannotated visual feature $\mathcal{F}_{\text{unannotated}}$ ($M' = 1$) and assume it is unavailable during training. We train a Feature VAE on the images of a single object (a truck), setting the target distribution to $p(\mathcal{F}_{\text{unannotated}}) = \mathcal{U}(-1, 1)$. The diffusion model is then conditioned on the VAE's latent output. During inference, we sample the target feature $\mathcal{F}'_{\text{unannotated}} \sim \mathcal{U}(-1, 1)$. As shown in fig. 4, varying the unannotated feature $f'^1_{\text{unannotated}}$ from -1 to 1 produces a smooth and consistent change in object rotation. This demonstrates that our model can successfully identify and gain control over the primary unannotated factor of variation.

Scaling to 3D Generation from 2D Slices We now scale our framework to the complex task of generating coherent 3D volumes from 2D slices, evaluated on *Thingi10K* and *BraTS2020*. In this setup, the dataset $\mathcal{X}$ consists of subsets $\mathcal{X}^i$, where each subset contains all 2D slices x^i_j of a single 3D object i at different depths j. This task involves two types of features: **(1) Unannotated Features ($\mathcal{F}_{\text{unannotated}}$):** The object's core identity (e.g., its shape or unique anatomy), which is shared across all slices in $\mathcal{X}^i$. **(2) Annotated Features ($\mathcal{F}_{\text{annotated}}$):** Employs the slice depth, which varies for each slice. Following section 3.2, we train our Feature VAE to extract the high-dimensional $\mathcal{F}_{\text{unannotated}}$ from a *single middle slice* ($\overline{\mathcal{X}^i}$), using a prior $p(\mathcal{F}_{\text{unannotated}}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. The diffusion model is then trained on *all* slices, conditioned on both the shared $\mathcal{F}_{\text{unannotated}}$ (from the middle slice) and the per-slice $\mathcal{F}_{\text{annotated}}$ (slice depth, mapped from -1 to 1). During inference, we sample a single random unannotated feature $\mathcal{F}'_{\text{unannotated}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to define a new object. We then generate the full 3D volume by feeding this *same* $\mathcal{F}'_{\text{unannotated}}$ to the model while sweeping the annotated depth feature from -1 to 1. The resulting 2D slices are stacked to form the 3D volume.

3D Shape Generation We apply this 3D framework to the *Thingi10K* dataset [37] to generate high-fidelity 3D shapes. We use a 64-dimensional latent space for $\mathcal{F}_{\text{unannotated}}$. The generated 2D slices are stacked into a voxel grid, converted to a mesh, and rendered using *MeshLab* [50]. As shown in fig. 8, sampling a random $\mathcal{F}'_{\text{unannotated}}$ (each row) and varying the slice depth generates 3D models with unified and consistent structural and semantic properties across all views. We perform a large-scale quantitative evaluation against baseline conditioning methods (*Timestep Embedding*, *Cross-Attention*, and *FiLM* [49]), all trained under identical conditions. First, we assess generative fidelity by computing FID and KID between 1,000 generated and 1,000 real 3D objects. As shown in table 2, **DNC** achieves the lowest (best) *FID* and *KID* scores, on par with the ground-truth validation set, indicating superior perceptual quality.

Next, we evaluate reconstruction precision by extracting $\mathcal{F}_{\text{unannotated}}$ from the middle slice of 1,000 test objects and using it to reconstruct the full 3D object. We compare this reconstruction to the ground truth using *MSE*, *IoU*, and *Chamfer Distance*. Baselines receive the same features and generate all slices from a single initial noise. As table 2 shows, **DNC** achieves dramatically better reconstruction scores (e.g., **0.7486 IoU** vs. ≤ 0.0219), demonstrating its superior ability to encode the object's identity from a single slice. The qualitative comparison can be seen in fig. 3. Baseline methods that synthesize slices from a fixed initial noise fail to produce coherent 3D structures. In contrast, our method's use of a consistent unannotated feature ensures geometric and semantic consistency, aligning with our quantitative results.

3D Medical Image Generation We further validate our 3D framework on the *BraTS2020* dataset of 3D brain MRI scans [39] in fig. 10. The goal is to generate high-quality, anatomically coherent 3D brain volumes. We use the same methodology as for Thingi10K, but with a 48-dimensional unannotated feature space. For each randomly sampled $\mathcal{F}'_{\text{unannotated}}$ (each two-row block), we show the generated vertical slices (top row) and the 3D volume reconstructed from horizontal cuts (bottom row). The results demonstrate that **DNC** generates MRI scans with consistent structural and semantic properties, including brain shape, size, and anatomical coherence.

We conduct a large-scale quantitative evaluation against the same baselines, computing FID and KID scores between 200 generated and 200 real MRI images. Crucially, we evaluate fidelity on both the generated *vertical* slices and the reconstructed *horizontal* slices, the latter of which is a strong test of 3D consistency. As reported in table 3, our method achieves the lowest FID and KID in both dimensions. The improvement is particularly stark for the horizontal reconstructions, which require cross-slice consistency. Our method's performance closely approaches that of the ground-truth validation set, while baselines fail to maintain coherence, as can be observed in fig. 5. Baselines produce fragmented and anatomically inconsistent horizontal cross-sections. Our method,

by conditioning on a consistent latent feature, maintains structural integrity and realism throughout the reconstructed volume.

5 Limitations and Future Work

Scale of validation. All experiments are conducted at 256×256 resolution with a single U-Net family trained from scratch. We have not evaluated DNC on large-scale text-to-image generation or with larger pretrained backbones, so whether its behavior carries over to that regime remains open. Our evaluation also mainly covers ordered, largely geometric factors; attribute-manipulation benchmarks are a natural next step (appendix L).

Scalar, continuous features. DNC encodes each feature as a normalized scalar magnitude along a single learned direction (eq. (1)). This suits continuous, ordered attributes, but unordered categorical attributes have no natural magnitude: in our multi-attribute experiment (appendix J), the digit class is the most weakly controlled attribute ($r = 0.323$, versus 0.40–0.69 for the continuous attributes at $M = 6$), and our experiments are limited to $M \leq 6$ (table 6).

Dense and spatial conditioning. The entire conditioning signal is a single global shift S in an M-dimensional linear subspace of the noise space. This is well suited to a modest number of holistic attributes, but it is not sufficient for dense or spatially structured conditioning, such as image-conditioned virtual try-on, where the condition carries high-frequency spatial information.

Text and other high-dimensional conditioning. A text embedding could be projected onto the learned basis $\{d_j\}$, which preserves the mechanism but caps the conditioning at M coefficients. Since $M \leq n = 4096$ (appendix A), this is not immediately capacity-bound. The main obstacle is structural: the shift S is confined to a linear subspace with a fixed learned basis, and the map from a high-dimensional semantic embedding to the required latent perturbation is unlikely to be linear. A natural extension replaces the linear readout with a learned nonlinear map $S = g_\psi(\text{embedding})$, retaining the initial-noise mechanism while removing the fixed-basis restriction. Exploring such learned shift maps is a promising direction for future work.

Dependence on the Feature VAE. For unannotated data, control is further bounded by the *Feature VAE*: no discovered direction is yet shown to control a non-geometric factor (appendix L), and pretrained encoders could replace it; we leave both to future work.

6 Conclusion

We have introduced a new paradigm for conditioning diffusion models that offers robust control over visual features. We identified a fundamental flaw in existing methods—"conditioning decay"—where the model's focus on the conditioning signal fades during the denoising process. We solve this problem not by modifying the U-Net architecture, but by fundamentally changing the generative setup. Our method conditions the entire process by structuring the initial noise x_T, mapping features to orthogonal, scaled shifts in the latent space. This, combined with an auxiliary shift loss $\mathcal{L}_{\text{shift}}$, ensures the conditioning is enforced with high fidelity at every step. Our experiments demonstrate the power of this approach. On a controlled rotation task, we achieved near-perfect precision, reducing the L1 error of SOTA baselines by over 50%. We then showed the scalability of our framework. By integrating a Feature VAE, we transformed a challenging unsupervised discovery problem into a simple conditioning task. This enabled us to generate coherent, 3D-aware objects on Thingi10K and BraTS2020 by discovering their latent identity from a single 2D slice. The resulting 3D reconstructions are not just visually plausible but also quantitatively superior, as evidenced by the dramatic improvements in fidelity (FID) and reconstruction (IoU) on data (horizontal slices) our model was never trained on. This work moves beyond the "black box" of intermediate conditioning and provides a clear, interpretable, and general-purpose mechanism for high-precision generative control. We believe this shift-based paradigm opens a new avenue for scientific discovery, medical counterfactuals, and high-fidelity 3D asset creation, laying a new foundation for the next generation of steerable generative models.

Acknowledgments

This project was partly supported by the Munich Center for Machine Learning and the ELLIS Institute Finland.

References

[1] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[2] 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.

[3] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[4] 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*, pp. 10684–10695, 2022.

[5] H. Liu, W. Zhang, J. Xie, F. Faccio, M. Xu, T. Xiang, M. Z. Shou, J.-M. Perez-Rua, and J. Schmidhuber, "Faster diffusion via temporal attention decomposition," *arXiv preprint arXiv:2404.02747*, 2024.

[6] B. Liu, C. Wang, T. Cao, K. Jia, and J. Huang, "Towards understanding cross and self-attention in stable diffusion for text-guided image editing," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7817–7826, 2024.

[7] H. Dhamo, A. Farshad, I. Laina, N. Navab, G. D. Hager, F. Tombari, and C. Rupprecht, "Semantic image manipulation using scene graphs," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5213–5222, 2020.

[8] A. Farshad, Y. Yeganeh, H. Dhamo, F. Tombari, and N. Navab, "Dispositionet: Disentangled pose and identity in semantic image manipulation," in *33rd British Machine Vision Conference 2022, BMVC 2022, London, UK, November 21-24, 2022*, BMVA Press, 2022.

[9] P. Jahoda, Y. Yeganeh, E. Adeli, N. Navab, and A. Farshad, "Prism: Progressive restoration for scene graph-based image manipulation," in *European Conference on Computer Vision*, pp. 142–160, Springer, 2024.

[10] A. Farshad, Y. Yeganeh, Y. Chi, C. Shen, B. Ommer, and N. Navab, "Scenegenie: Scene graph guided diffusion models for image synthesis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 88–98, 2023.

[11] S. Lüpke, Y. Yeganeh, E. Adeli, N. Navab, and A. Farshad, "Physics-informed latent diffusion for multimodal brain mri synthesis," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 198–207, Springer, 2024.

[12] Ç. Köksal, Y. Yeganeh, N. Navab, and A. Farshad, "Semantic scene editing for cholecystectomy surgery," in *International Workshop on Collaborative Intelligence and Autonomy in Image-Guided Surgery*, pp. 32–41, Springer, 2025.

[13] Y. Yeganeh, R. Lazuardi, A. Shamseddin, E. Dari, Y. Thirani, N. Navab, and A. Farshad, "Visage: Video synthesis using action graphs for surgery," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 146–156, Springer, 2024.

[14] Y. Yeganeh, N. Navab, and A. Farshad, "Movis: Motion-guided video generation for laparoscopic surgery," in *International Workshop on Agentic AI for Medicine*, pp. 215–225, Springer, 2025.

[15] D. Biagini, N. Navab, and A. Farshad, "Hierasurg: Hierarchy-aware diffusion model for surgical video generation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 310–319, Springer, 2025.

[16] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.

[17] P. Dhariwal and A. Nichol, "Diffusion models beat gans on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[18] J. Ho and T. Salimans, "Classifier-free diffusion guidance," *arXiv preprint arXiv:2207.12598*, 2022.

[19] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, vol. 1, no. 2, p. 3, 2022.

[20] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *Advances in neural information processing systems*, vol. 35, pp. 36479–36494, 2022.

[21] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[22] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," in *International Conference on Learning Representations*, 2023.

[23] Y. Shi, P. Wang, J. Ye, M. Long, K. Li, and X. Yang, "Mvdream: Multi-view diffusion for 3d generation," *arXiv preprint arXiv:2308.16512*, 2023.

[24] X. Long, Y.-C. Guo, C. Lin, Y. Liu, Z. Dou, L. Liu, Y. Ma, S.-H. Zhang, M. Habermann, C. Theobalt, *et al.*, "Wonder3d: Single image to 3d using cross-domain diffusion," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9970–9980, 2024.

[25] J. Xiang, J. Yang, B. Huang, and X. Tong, "3d-aware image generation using 2d diffusion models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2383–2393, 2023.

[26] J. Lin, X. Yang, M. Chen, Y. Xu, D. Yan, L. Wu, X. Xu, L. Xu, S. Zhang, and Y.-C. Chen, "Kiss3dgen: Repurposing image diffusion models for 3d asset generation," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 5870–5880, 2025.

[27] Y. Xue, X. Xie, R. Marin, and G. Pons-Moll, "Gen-3diffusion: Realistic image-to-3d generation via 2d & 3d diffusion synergy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[28] L. Ding, S. Dong, Z. Huang, Z. Wang, Y. Zhang, K. Gong, D. Xu, and T. Xue, "Text-to-3d generation with bidirectional diffusion using both 2d and 3d priors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5115–5124, 2024.

[29] J. Li, H. Tan, K. Zhang, Z. Xu, F. Luan, Y. Xu, Y. Hong, K. Sunkavalli, G. Shakhnarovich, and S. Bi, "Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model," *arXiv preprint arXiv:2311.06214*, 2023.

[30] C. Wang, J. Gu, X. Long, Y. Liu, and L. Liu, "Geco: Generative image-to-3d within a second," *arXiv preprint arXiv:2405.20327*, 2024.

[31] R. Gao, A. Holynski, P. Henzler, A. Brussee, R. Martin-Brualla, P. Srinivasan, J. T. Barron, and B. Poole, "Cat3d: Create anything in 3d with multi-view diffusion models," *arXiv preprint arXiv:2405.10314*, 2024.

[32] N. J. Dhinagar, S. I. Thomopoulos, E. Laltoo, and P. M. Thompson, "Counterfactual mri generation with denoising diffusion models for interpretable alzheimer's disease effect detection," in *2024 46th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 1–6, IEEE, 2024.

[33] X. Chen and Y. Peng, "Counterfactual condition diffusion with continuous prior adaptive correction for anomaly detection in multimodal brain mri," *Expert Systems with Applications*, vol. 254, p. 124295, 2024.

[34] G. Jeanneret, L. Simon, and F. Jurie, "Diffusion models for counterfactual explanations," in *Proceedings of the Asian conference on computer vision*, pp. 858–876, 2022.

[35] Y. Yeganeh, A. Farshad, I. Charisiadis, M. Hasny, M. Hartenberger, B. Ommer, N. Navab, and E. Adeli, "Latent drifting in diffusion models for counterfactual medical image synthesis," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 7685–7695, 2025.

[36] H. Chen, R. Yin, Y. Chen, Q. Chen, and C. Li, "Learning patient-specific disease dynamics with latent flow matching for longitudinal imaging generation," 2026.

[37] Q. Zhou and A. Jacobson, "Thingi10k: A dataset of 10,000 3d-printing models," *arXiv preprint arXiv:1605.04797*, 2016.

[38] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia object image library (coil-100)," 1996.

[39] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, *et al.*, "The multimodal brain tumor image segmentation benchmark (brats)," *IEEE transactions on medical imaging*, vol. 34, no. 10, pp. 1993–2024, 2014.

[40] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, L. Lanczi, E. Gerstner, M.-A. Weber, T. Arbel, B. B. Avants, N. Ayache, P. Buendia, D. L. Collins, N. Cordier, J. J. Corso, A. Criminisi, T. Das, H. Delingette, C. Demiralp, C. R. Durst, M. Dojat, S. Doyle, J. Festa, F. Forbes, E. Geremia, B. Glocker, P. Golland, X. Guo, A. Hamamci, K. M. Iftekharuddin, R. Jena, N. M. John, E. Konukoglu, D. Lashkari, J. A. Mariz, R. Meier, S. Pereira, D. Precup, S. J. Price, T. R. Raviv, S. M. S. Reza, M. Ryan, D. Sarikaya, L. Schwartz, H.-C. Shin, J. Shotton, C. A. Silva, N. Sousa, N. K. Subbanna, G. Szekely, T. J. Taylor, O. M. Thomas, N. J. Tustison, G. Unal, F. Vasseur, M. Wintermark, D. H. Ye, L. Zhao, B. Zhao, D. Zikic, M. Prastawa, M. Reyes, and K. Van Leemput, "The multimodal brain tumor image segmentation benchmark (brats)," *IEEE Transactions on Medical Imaging*, vol. 34, no. 10, pp. 1993–2024, 2015.

[41] S. Bakas, "Advancing the cancer genome atlas glioma mri collections with expert segmentation labels and radiomic features.," 2017.

[42] S. Bakas, M. Reyes, A. Jakab, S. Bauer, M. Rempfler, and A. Crimi, "Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the brats challenge," 2019.

[43] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," 2018.

[44] M. Bińkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying mmd gans," 2021.

[45] Y. Jin, D. Jiang, and M. Cai, "3d reconstruction using deep learning: a survey," *Commun. Inf. Syst.*, vol. 20, pp. 389–413, 2020.

[46] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese, "3d-r2n2: A unified approach for single and multi-view 3d object reconstruction," 2016.

[47] T. Wu, L. Pan, J. Zhang, T. Wang, Z. Liu, and D. Lin, "Density-aware chamfer distance as a comprehensive metric for point cloud completion," 2021.

[48] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, 2020.

[49] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," 2017.

[50] Visual Computing Lab, ISTI-CNR, "Meshlab." `https://www.meshlab.net/`. Accessed: 2025-11-07.

[51] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," 2018.

[52] A. C. Bovik, *Handbook of Image and Video Processing*. Academic Press, 2000.

[53] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[54] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[55] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," 2023.

A Initialization of Shift Directions

To initialize the set of shift directions $\{d_j\}_{j=1}^{M}$ in the diffusion latent space, we construct a set of M vectors in $\mathbb{R}^n$ (with $n = 64 \times 64$ in our experiments) that are orthogonal and standardized. This ensures that each visual feature dimension is assigned a well-conditioned direction that does not introduce unintended correlations between features at the start of training.

We first sample a random matrix

$$A \in \mathbb{R}^{n \times M}, \qquad A_{ij} \sim \mathcal{N}(0, 1), \tag{16}$$

and apply a QR decomposition to obtain an orthonormal basis spanning the same column space:

$$A = QR, \tag{17}$$

where $Q \in \mathbb{R}^{n \times M}$ has orthonormal columns,

$$Q^\top Q = I_M, \tag{18}$$

and $R \in \mathbb{R}^{M \times M}$ is upper triangular. The orthonormality of the columns of Q ensures that the resulting shift directions are mutually orthogonal:

$$d_j^\top d_k = 0, \qquad \forall\, j \neq k. \tag{19}$$

Since each column $Q_{:,j}$ has unit norm by construction, we convert these columns into our initial shift directions by setting

$$\widetilde{d}_j = Q_{:,j}^\top \in \mathbb{R}^{1 \times n}, \tag{20}$$

so that each shift direction is represented as a row vector consistent with our implementation.

Finally, to ensure that each direction has unit empirical standard deviation across its coordinates, we normalize each row:

$$d_j = \frac{\widetilde{d}_j}{\mathrm{Std}(\widetilde{d}_j)}, \qquad j = 1, \ldots, M, \tag{21}$$

where $\widetilde{d}_j$ denotes the jth row of $\widetilde{D}$ and the standard deviation is computed across the n latent dimensions.

This procedure yields a collection of shift directions

$$d_j \in \mathbb{R}^n,$$

that are approximately orthogonal, standardized, and well-scaled for stable training when used in the feature-conditioned shift formulation described in section 3.1. Note that because at most n mutually orthonormal vectors can exist in an n-dimensional space, the number of shift directions is fundamentally limited by the latent dimensionality; that is, the maximum feasible number of distinct shift directions is $M \leq n$.

B Conditional Analysis of the Forward Process

The forward process of DNC (eq. (2) and the noisy sample x_t^i in section 3.1) replaces the isotropic Gaussian noise of standard diffusion with $\epsilon_\perp + S^i$, which is neither zero-mean nor full-rank. Here we analyze this process conditionally on a single data point and state precisely what the analysis does and does not establish.

The projection used. eq. (2) removes only the component of ϵ along the sample-specific realized shift S^i, not along the full span of the direction matrix D. Writing $\hat{S} = S^i / \|S^i\|$, the projection coefficient in eq. (2) evaluates to

$$\frac{\epsilon^\top S^i}{\|S^i\|^2}\, S^i = (\epsilon^\top \hat{S})\, \hat{S} = (\hat{S}\hat{S}^\top)\, \epsilon, \tag{22}$$

so that

$$\epsilon_\perp = P_S\,\epsilon, \qquad P_S = I - \hat{S}\hat{S}^\top, \tag{23}$$

where P_S is the rank-$(n-1)$ orthogonal projector onto the complement of the single vector S^i. For $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, we therefore have $\epsilon_\perp \sim \mathcal{N}(\mathbf{0}, P_S)$, with covariance $P_S = I - S^i(S^i)^\top/\|S^i\|^2$, which depends on the features through S^i.

Projecting out the entire feature subspace $\mathrm{span}(D)$ instead would also remove noise components along feature directions that are inactive for the given sample, needlessly reducing the noise dimensionality. Projecting out only $\hat{S}$ protects the realized attribute shift from being overwritten by sampled noise while preserving the maximum noise variance.

Per-sample conditional decomposition. Fix a data point x_0^i and condition on its features, which fixes S^i and hence P_S. Because P_S does not depend on the timestep t, the stochastic part of the forward process remains confined to the fixed $(n-1)$-dimensional subspace $\mathrm{range}(P_S)$ throughout the trajectory, and the noisy state

$$x_t^i = \sqrt{\bar{\alpha}_t}\,x_0^i + \sqrt{1-\bar{\alpha}_t}\,(\epsilon_\perp + S^i) \tag{24}$$

splits into two orthogonal components.

Along $\hat{S}$ (deterministic). Since $\epsilon_\perp^\top \hat{S} = 0$ by construction and $(S^i)^\top \hat{S} = \|S^i\|$,

$$(x_t^i)^\top \hat{S} = \sqrt{\bar{\alpha}_t}\,(x_0^i)^\top \hat{S} + \sqrt{1-\bar{\alpha}_t}\,\|S^i\|. \tag{25}$$

In $\mathrm{range}(P_S)$ (standard VP diffusion). Since $P_S S^i = 0$ and $P_S\epsilon_\perp = P_S\epsilon$,

$$P_S\,x_t^i = \sqrt{\bar{\alpha}_t}\,P_S\,x_0^i + \sqrt{1-\bar{\alpha}_t}\,P_S\,\epsilon, \tag{26}$$

which is standard variance-preserving diffusion in the $(n-1)$-dimensional subspace $\mathrm{range}(P_S)$.

Equivalently, $q(x_t^i \mid x_0^i, S^i) = \mathcal{N}\big(\sqrt{\bar{\alpha}_t}\,x_0^i + \sqrt{1-\bar{\alpha}_t}\,S^i,\ (1-\bar{\alpha}_t)\,P_S\big)$, which departs from standard diffusion through its shifted mean and rank-deficient covariance. In particular, the terminal distribution is $x_T \approx \mathcal{N}(S^i, P_S)$ rather than $\mathcal{N}(\mathbf{0}, \mathbf{I})$. This introduces no train/test mismatch: with the cosine schedule at $T = 100$, $\bar{\alpha}_T = 2.4 \times 10^{-7}$, so the training construction of x_T and the inference construction in eq. (13) agree to numerical precision.

Training objective. Inside $\mathrm{range}(P_S)$, the variational bound follows the standard DDPM derivation with dimensionality $n-1$. Along $\hat{S}$, the forward transitions are deterministic, so a likelihood-based treatment of this axis is degenerate: the corresponding KL term is zero if the reverse process matches the forward process exactly and infinite otherwise. The shift loss $\mathcal{L}_{\text{shift}}$ (eqs. (5) and (6)) explicitly trains the network to match this component at every t.

Scope of the claim. Densities in the analysis above are defined with respect to the Lebesgue measure on the $(n-1)$-dimensional subspace $\mathrm{range}(P_S)$. Because P_S varies per data point, the marginal over the dataset is a mixture over conditioning vectors S^i rather than a single global Gaussian process. We therefore claim only this per-sample conditional bound, and do *not* claim a variational bound on the unconditional data likelihood.

Behavior at $S = 0$. When all normalized feature values are zero, $S^i = 0$ and the coefficient in eq. (2) is undefined. In our implementation, $\|S^i\|^2$ is clamped from below at 10^{-8}, so for $S^i = 0$ the projection coefficient evaluates to zero, giving $\epsilon_\perp = \epsilon$, $P_S = I$, and $x_T = \epsilon$. In this case the forward process reduces exactly to standard diffusion.

C Shift in the State versus Shift in the Prediction

In the forward process, the shift enters the noisy state scaled by $\sqrt{1-\bar{\alpha}_t}$, so its contribution to x_t decays to zero as $t \to 0$. This is expected: as $t \to 0$, $x_t \to x_0$, and the condition has been converted into image content. Two mechanisms carry the condition along the trajectory. First, the deterministic DDIM sampler ($\eta = 0$) defines a deterministic map from the shifted initial noise $x_T = \epsilon_\perp + S'$ to

the generated image. Second, $\mathcal{L}_{\text{shift}}$ (eq. (6)) constrains the projection of the network prediction ϵ_θ onto $\hat{S}$ at every timestep.

These are two distinct quantities: the share of S in the state, which is $\sqrt{1-\bar{\alpha}_t}$ by construction, and the share of S recovered in the prediction, which we measure as the projection coefficient

$$r_t = \frac{\epsilon_\theta(x_t, t)^\top S}{\|S\|^2}, \tag{27}$$

i.e., the coefficient in eq. (5); $r_t = 1$ indicates full recovery of the shift. table 4 reports both for the rotation experiment and for multi-attribute *MNIST* with $M = 2$ (rotation and thickness), using $T = 50$ diffusion steps.

Table 4: **Share of the shift in the state versus in the prediction** ($T = 50$; $t = 49$ is the first denoising step). The share in the state, $\sqrt{1-\bar{\alpha}_t}$, is analytic; r_t is the measured projection of ϵ_θ onto the shift, normalized so that $r_t = 1$ is full recovery.

t	**Share of S in state**	r_t **(rotation)**	r_t **(*MNIST*, $M = 2$)**
49	1.000	0.963	1.019
39	0.952	0.952	1.033
29	0.812	0.957	1.026
19	0.594	0.991	1.017
9	0.318	1.011	0.970
4	0.167	0.921	0.935
0	0.042	0.080	0.676

As shown in table 4, the share of S in the state falls by a factor of approximately 24 over the trajectory, whereas the prediction recovers the shift almost fully until the last few steps. At the final step ($t = 0$), recovery drops sharply, most strongly for rotation ($r_0 = 0.080$). At this point the state is already close to the clean image, the noise prediction contributes little to the update, and the condition is carried by the image content itself.

D Measuring Conditioning Decay

Conventional conditioning mechanisms inject the condition into intermediate network features, which can lose influence over the course of the denoising trajectory. To measure this directly, we evaluate the realized attribute error at intermediate timesteps for four conditioning mechanisms on the solid-arrow rotation experiment (section 4), using an identical backbone and the same angle regressor as in table 1. At each reported timestep t, we compute the predicted clean sample

$$\hat{x}_0(t) = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}, \tag{28}$$

decode it to pixel space, and pass the decoded image to the angle regressor. The regressor is never applied to the noisy state x_t directly.

Table 5: **Realized angle error $|\hat{\theta}(t) - \theta|$ (degrees) along the denoising trajectory**, measured on the decoded predicted clean sample $\hat{x}_0(t)$. Drift is the change in error from $t = 60$ to $t = 1$; positive values indicate degradation.

Method	t=80	t=60	t=40	t=20	t=5	t=1	**Drift** $60 \to 1$
Timestep embedding	11.2	1.80	1.63	2.14	2.71	2.96	+1.16
Cross-attention	9.6	1.55	1.42	1.83	2.31	2.50	+0.95
FiLM	13.4	1.94	2.05	2.62	3.18	3.41	+1.47
DNC (ours)	**8.9**	**1.35**	**1.22**	**1.18**	**1.20**	**1.21**	**−0.14**

As shown in table 5, all three baselines reach their best attribute fidelity mid-trajectory ($t = 40$ for timestep embedding and cross-attention, $t = 60$ for FiLM) and then degrade toward the end

of sampling, with errors increasing by $0.95°$–$1.47°$ from $t = 60$ to $t = 1$. DNC instead remains stable over the same interval ($-0.14°$) and achieves the lowest error at every reported timestep. DNC therefore substantially mitigates, though it does not eliminate, the loss of attribute control along the trajectory, consistent with the prediction-side recovery of the shift in appendix C.

E Scaling the Shift at Inference

Because the conditioning signal in DNC is carried entirely by the initial shift, a natural question is whether DNC admits a controllable guidance parameter analogous to the scale in classifier-free guidance (CFG). The simplest candidate is to scale the target shift by a scalar $\lambda > 0$ at inference, replacing eq. (13) with

$$x_T = \epsilon_\perp + \lambda S'. \tag{29}$$

This requires no retraining, no change to the sampler, and no additional forward pass per denoising step. Since

$$\lambda S' = \sum_{j=1}^{M} \big(\lambda\, \overline{f'_j}\big)\, d_j, \tag{30}$$

scaling the shift is algebraically identical to rescaling every normalized target feature value. Two properties follow directly from the construction.

λ moves the semantic target. Scaling maps each target feature $\overline{f'_j}$ to $\lambda\, \overline{f'_j}$. For $\lambda \neq 1$ the model is therefore asked to generate a different feature configuration, and for $|\lambda\, \overline{f'_j}| > 1$ one that lies outside the normalized range $[-1, 1]$ seen during training. λ thus does not sharpen adherence to a fixed target, which is what CFG does by amplifying the difference between the conditional and unconditional predictions while holding the target fixed.

The noise decomposition is unchanged. The projection in eq. (12) depends only on the direction of S', so $\epsilon_\perp$ is identical for every $\lambda > 0$. λ modulates only the magnitude of the shift, and the orthogonality construction of eqs. (2) and (12) holds at every λ.

Relation to classifier-free guidance. A true CFG analogue is nonetheless available. Setting $S = 0$ recovers unconditional generation exactly: the projection in eq. (2) reduces to the identity, so $x_T = \epsilon$ and the reverse process is standard unconditional diffusion. We have verified that sampling at $S = 0$ produces coherent unconditional samples from our trained models, so the unconditional branch already exists inside a DNC network at no architectural cost. A network trained with feature dropout would therefore support

$$\tilde{\epsilon}_\theta = \epsilon_\theta(x_t \mid 0) + w\big[\epsilon_\theta(x_t \mid S) - \epsilon_\theta(x_t \mid 0)\big], \tag{31}$$

with w a genuine guidance weight, a single network, and no auxiliary unconditional model. Unlike λ, this sharpens adherence to a fixed target rather than moving it.

F Controlled Experiment: Arrow Rotation

Quantitative evaluation fig. 6 illustrates the correlation between the ground-truth rotation angles and the angles predicted by the rotation estimation model described in section 4, for both the solid-arrow and textured-arrow experiments. As shown in table 1, the close agreement between the target rotation angles and the predicted angles of the generated images demonstrates that the diffusion model, when conditioned using our method, achieves highly precise and reliable feature control.

Evaluating interpolation capability To assess the interpolation ability of our method, we repeat the rotation experiment but intentionally remove a portion of the training data. Specifically, we exclude 20% of the data by removing all arrows with rotation angles in the range $72° \leq \theta \leq 108°$. We then perform inference using target angles θ_{target} that fall inside this excluded interval.

The corresponding generated results are shown in fig. 18. Arrows displayed inside the gray region ($75° \leq \theta_{\text{target}} \leq 105°$) were generated from angles that the model never observed during training.

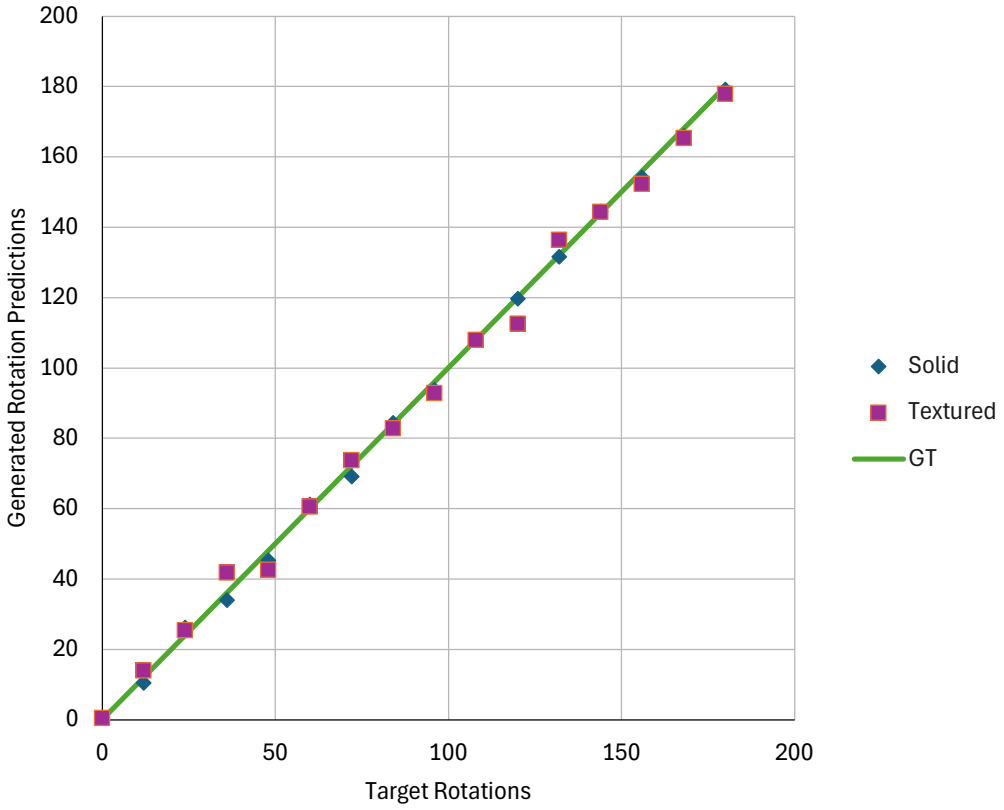


Figure 6: Correlations between predicted and target rotation angles in the solid-arrow experiment (fig. 17a) and the textured-arrow experiment (fig. 17b).

Although the images within this interpolation range are slightly less accurate than those outside it (e.g., a small deformation around $\theta_{\text{target}} = 93°$), the generated arrows still follow the correct rotational progression. This indicates that the diffusion model successfully learned the underlying rotational structure even without seeing the full range of angles during training.

G Additional Results on Thingi10K

Here we provide the results of 3D generation on Thingi10K. The discussions on these results can be found in section 4.

H Additional Quantitative Experiments on BraTS2020

Perceptual and structural reconstruction evaluation on BraTS2020 We assess the perceptual quality and semantic reconstruction accuracy of our method using the *Learned Perceptual Image Patch Similarity (LPIPS)* [51], *Peak Signal-to-Noise Ratio (PSNR)* [52], *Structural Similarity Index (SSIM)* [53], and *Mean Squared Error (MSE)* metrics. To perform this evaluation, we extract the unannotated visual features of 200 3D MRI brain images from the validation set using their respective middle vertical slice via the trained Feature VAE. We then reconstruct the full brain volumes using the diffusion model conditioned on both slice depth and unannotated visual features. For each vertical depth, we compute the aforementioned metrics between the generated and ground-truth slices and report their average values across the evaluation set in fig. 9. Here, a depth of zero denotes the middle slice, negative depths correspond to slices located left of the center, and positive depths represent slices on the right.

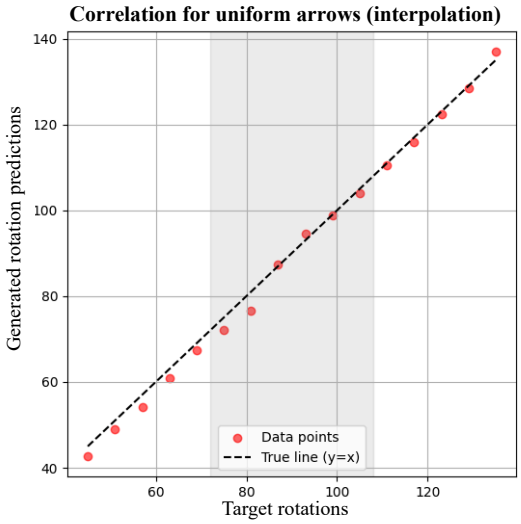


Figure 7: Correlation between target and predicted rotation angles for the interpolation experiment. The gray band indicates the excluded training range ($72° \leq \theta \leq 108°$). Our method achieves $r^2 = 0.9953$ and an average L1 error of $1.5°$, demonstrating strong interpolation performance despite slightly reduced accuracy compared to the full-data setting.

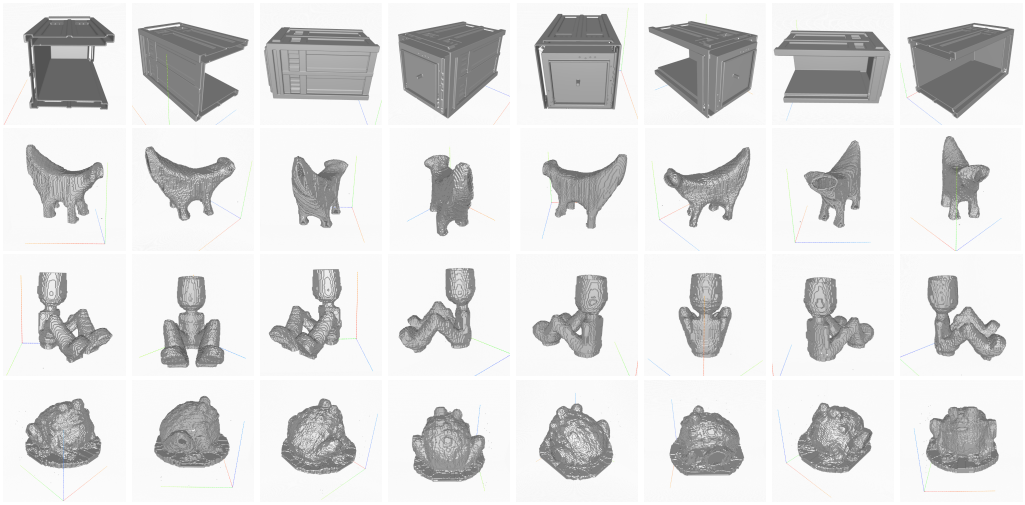


Figure 8: **3D Generation on *Thingi10K* [37].** Each row shows a 3D model generated by our method from randomly sampled unannotated visual features $\mathcal{F}'_{\text{unannotated}}$, rendered from multiple viewpoints. The models are generated by conditioning the diffusion model on both the latent visual features and the slice depth information.

Quantitative interpolation analysis To compute numerical interpolation performance, we use the same rotation estimation model from the previous experiment to measure the predicted angles of the generated images within the excluded range. fig. 7 presents the correlation between the target angles and the estimated angles, with the gray region marking the interpolation interval. Our method achieves a coefficient of determination of $r^2 = 0.9953$ and an average L1 error of $1.5°$. Although the accuracy is slightly lower than in the full-data experiment, the correlation remains extremely strong, confirming that the model maintains robust interpolation capability.

As shown in fig. 9, our method achieves an *LPIPS* score of 0.09–0.15, which falls within the high perceptual similarity range reported in prior studies [51]. Similarly, the *SSIM* values of 0.84–0.96

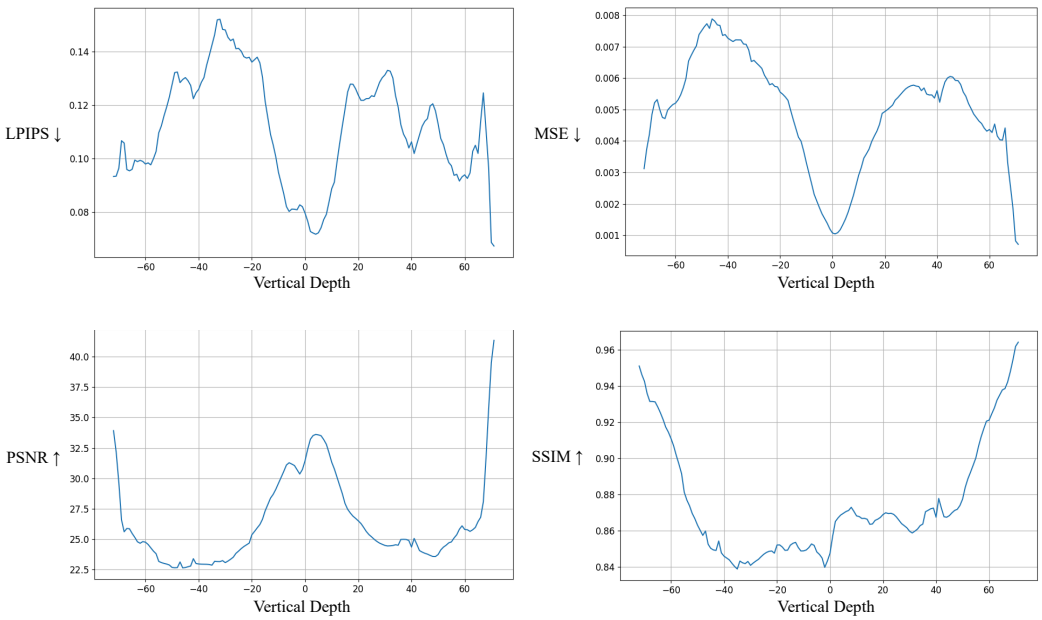


Figure 9: **Perceptual and structural similarity metrics across vertical depths on the *BraTS2020* dataset.** Average *LPIPS*, *MSE*, *PSNR*, and *SSIM* scores computed between generated and ground-truth vertical slices at different depths. Depth 0 denotes the middle slice used for feature extraction, negative values correspond to slices on the left, and positive values correspond to slices on the right. Our method achieves high perceptual and structural similarity across all depths, with the highest correspondence near the middle slices and consistently strong performance throughout the entire volume.

indicate strong structural fidelity relative to the reference, consistent with the thresholds reported in [53]. Our *PSNR* scores of 22.5–35.0 dB further confirm high pixel-level accuracy, comparable to state-of-the-art generative models [52].

As illustrated, the highest similarity is observed near the middle slices—where feature extraction occurs—while slightly lower values appear toward the left and right regions of the brain. Nevertheless, the generated results maintain high semantic and visual consistency across all depths. The far-left and far-right slices exhibit artificially higher similarity scores due to the reduced brain area visible in those regions, which naturally leads to higher pixel-level similarity.

Finally, as in section 4, we present additional randomly sampled results from the diffusion model conditioned on the *BraTS2020* dataset. figs. 11 and 12 show vertical slices, along with their corresponding horizontal slices obtained by stacking the vertical ones, for eight additional brains. The results demonstrate both high visual fidelity and strong internal consistency across slices. Since the *BraTS2020* dataset includes subjects with brain tumors, many of the generated samples in fig. 10 and figs. 11 and 12 naturally exhibit tumor regions as well.

I Multi-Feature Conditioning: MNIST

We further evaluate our framework by conditioning a diffusion model on the *MNIST* dataset [54]. To assess the model's ability to handle multiple annotated visual features, we augment the original dataset by introducing additional variations, including digit skew, increased stroke thickness, and digit color specified by RGB values (red, green, and blue). The generated samples for various target configurations—normal, thick-stroke, skewed, and skewed combined with thick-stroke—across four different target colors for each digit are presented in figs. 13 and 14. As shown, the generated images accurately follow the specified target annotations while preserving the underlying structure and handwriting style of each digit.

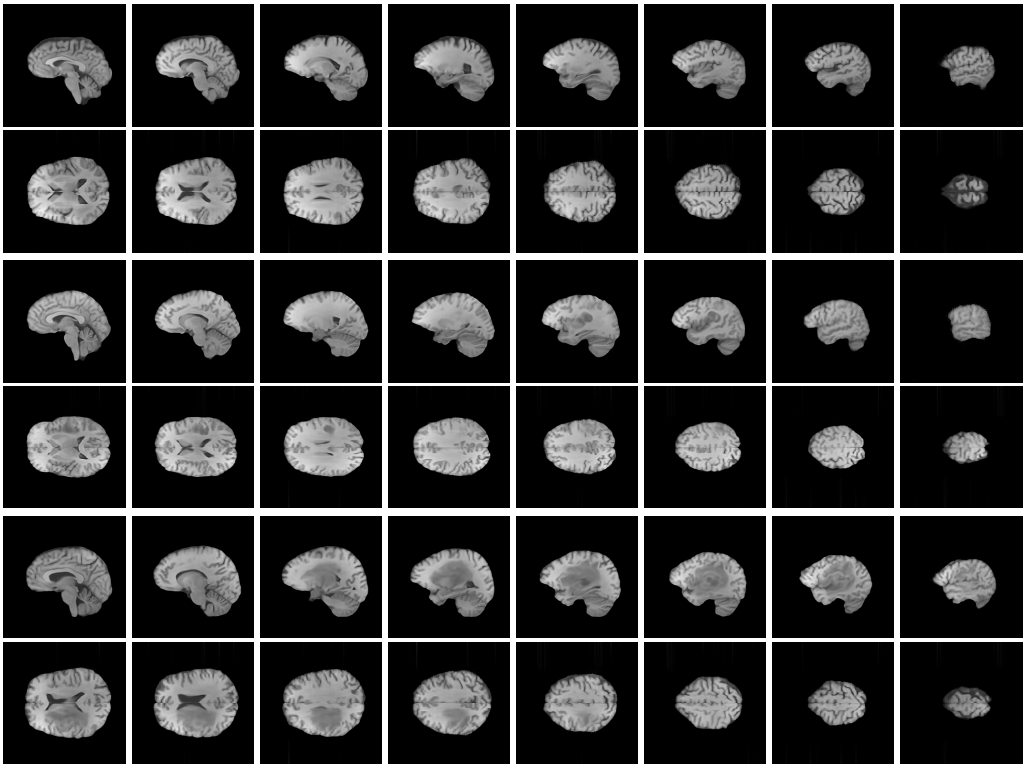


Figure 10: **3D Generation Results on *BraTS2020***. For each randomly sampled unannotated visual feature, the first row shows generated vertical MRI slices spanning from the left side of the brain to the middle, with the annotated visual feature $\mathcal{F}_{\text{annotated}}$ (slice depth) varied from -1 to 1. The second row depicts reconstructed 3D volumes obtained by stacking the generated slices and cutting them from the top of the brain to the middle.

J Control Accuracy vs. Number of Conditioned Features

Since all conditioned features share the same noise space, a natural question is how control accuracy changes as the number of simultaneously conditioned features M grows. We study this on a multi-attribute variant of *MNIST*, where each image is annotated with up to six attributes: five continuous attributes (rotation, stroke thickness, brightness, skew, and scale) and one categorical attribute (digit class). We train DNC with $M \in \{1, \ldots, 6\}$, adding attributes cumulatively in this order. The $M = 5$ setting therefore contains all continuous attributes, so that the step from $M = 5$ to $M = 6$ isolates the effect of adding the categorical attribute. All configurations are trained from scratch under an identical early-stopping rule, using a reduced-scale setup that differs from the one in table 9: 32×32 images, $T = 50$ diffusion steps, and $N = 8000$ training images.

All continuous attributes are normalized to $[-1, 1]$ prior to training and evaluation. We recover each attribute from the generated images with a fixed, task-appropriate estimator: direct image statistics for brightness and thickness, and a trained CNN regressor for rotation, skew, scale, and digit class. We report the Pearson correlation r between the target and recovered value of each attribute, separately per attribute, since averages over heterogeneous continuous and categorical attributes are difficult to interpret.

table 6 supports three observations. First, control of the continuous attributes is stable across the number of conditioned features: their mean correlation moves from 0.562 at $M = 1$ to 0.490 at $M = 6$, with intermediate fluctuations within run-to-run variance. Second, adding the categorical attribute does not degrade the continuous ones: their mean is 0.485 at $M = 5$ and 0.490 at $M = 6$. Third, the categorical digit attribute is the most weakly controlled feature ($r = 0.323$). This is consistent with the mechanism: DNC encodes each feature as a magnitude along a single direction, which suits continuous, ordered attributes, whereas an unordered class label has no natural magnitude.

Table 6: **Per-attribute control accuracy (Pearson $r \uparrow$) vs. number of simultaneously conditioned features** on multi-attribute *MNIST* (32×32, $T = 50$, $N = 8000$). Attributes are added cumulatively; $M = 5$ contains all continuous attributes and $M = 6$ adds the categorical digit class.

Attribute	M=1	M=2	M=3	M=4	M=5	M=6
Rotation	0.562	0.517	0.337	0.479	0.372	0.406
Thickness	–	0.487	0.342	0.454	0.392	0.404
Brightness	–	–	0.736	0.846	0.780	0.694
Skew	–	–	–	0.433	0.335	0.452
Scale	–	–	–	–	0.545	0.496
Digit (categorical)	–	–	–	–	–	0.323
Mean (continuous)	0.562	0.502	0.472	0.553	0.485	0.490

This applies equally to discovered features: the unannotated features produced by the *Feature VAE* are continuous by construction, but how well they can be controlled depends on how well each latent dimension captures a single, ordered factor of variation.

K Incremental Component Ablation

A simpler alternative to DNC is to shift the initial noise of a standard denoiser directly. To isolate the contribution of each component of DNC, we conduct an incremental ablation on multi-attribute *MNIST* with $M = 4$ (rotation, thickness, brightness, skew), using an identical U-Net backbone and the reduced-scale setup of appendix J. Each ablated configuration removes exactly one component from the full DNC: the shift loss $\mathcal{L}_{\text{shift}}$ (eq. (6)), the orthogonal rejection (eq. (2)), or the learned directions, which are replaced by the frozen QR initialization of appendix A. The naive variant removes all three *and* trains with the standard unshifted target ϵ, i.e., it is a vanilla denoiser whose initial noise is shifted only at inference.

Table 7: **Incremental component ablation** on multi-attribute *MNIST* ($M = 4$, identical backbone, reduced-scale setup). Mean Pearson r over the four continuous attributes. Variance across evaluation snapshots within each DNC configuration is ± 0.03–0.05.

Configuration	**Mean Pearson $r \uparrow$**
Full DNC	**0.553**
$-\ \mathcal{L}_{\text{shift}}$	0.501
− orthogonal rejection	0.487
− learned directions (frozen QR)	0.508
Naive (all removed, unshifted ϵ target)	-0.015

The target construction is the defining effect. As shown in table 7, the naive variant recovers no attribute signal, while every DNC variant achieves a mean correlation near 0.5. Shifting the initial noise while training an ordinary denoiser is therefore not sufficient; training the network to reproduce the shifted effective noise $\epsilon_\perp + S$ is what yields control. The conditioning must be written into the regression target rather than only into the initial noise.

The auxiliary components are not ranked. Within each DNC configuration, the variance across evaluation snapshots (± 0.03–0.05) is as large as the spread between configurations. We therefore claim only the gap over the naive baseline, and do not rank $\mathcal{L}_{\text{shift}}$, the orthogonal rejection, and the learned directions against each other. This parity is consistent with what the construction predicts in an $n = 4096$ dimensional space:

Shift loss. The component of $\mathcal{L}_{\text{denoise}}$ along $\hat{S}$ is $\left(\|S^i\| - \epsilon_\theta(x_t^i, t)^\top \hat{S}\right)^2$, which is exactly $\mathcal{L}_{\text{shift}}$ (eqs. (5) and (6)). $\mathcal{L}_{\text{shift}}$ therefore acts as an explicit reweighting of the conditioning coordinate: it doubles the weight on one coordinate out of n, and removing it halves that weight.

Orthogonal rejection. Because the directions are normalized to unit per-coordinate standard deviation (appendix A), $\|d_j\| \approx \sqrt{n}$, and a typical shift has $\|S\| = \Theta(\sqrt{n})$. The component removed by the rejection is $\epsilon^\top \hat{S} \sim \mathcal{N}(0,1)$, only a marginal perturbation of the encoded coefficient. The primary role of the rejection is mathematical exactness, ensuring that x_T is a true sample of the conditional prior (appendix B), rather than an improvement in signal-to-noise ratio at this dimensionality.

L Interpreting Discovered Directions

Our experiments show that the combined discovered code captures non-geometric variation: on *BraTS2020*, generated volumes reproduce and preserve tumor regions (figs. 11 and 12). They do not, however, establish that any single discovered direction controls a specific factor; the basis is learned for reconstruction rather than for disentanglement. Establishing this requires a per-direction analysis: sweeping one discovered direction from -1 to 1 while holding all others fixed, and correlating the generated change with measured quantities. To separate pathology from simple size effects, non-geometric quantities (e.g., tumor-region area from the provided labels) should be measured alongside spatial ones (e.g., total brain volume and ventricular area), reporting the full spectrum of directions, including those without a clean correlate.

The annotated experiments already show that the mechanism itself is not tied to geometry: on *MNIST* (figs. 13 and 14), DNC controls RGB color and stroke thickness, which are radiometric and stylistic rather than spatial factors. A natural benchmark for non-geometric attributes is counterfactual generation, e.g., facial attribute manipulation on CelebA-HQ or a clinical dataset where a labeled pathological attribute is manipulated while subject identity is held constant. DNC fits this setting directly: the annotated target attribute enters as a dedicated shift direction, while the discovered identity code holds the subject fixed. We leave both analyses to future work.

M Evaluating DNC on Additional Generative Algorithms: Flow Matching

To demonstrate that our conditioning mechanism is not limited to diffusion models, we additionally apply DNC to a different generative framework: *Flow Matching for Generative Modeling* [55]. In this experiment, we reuse the shift construction in eq. (1) and the orthogonality constraint in eq. (9), and incorporate them into the flow-matching objective.

Shifting the latent initial condition In the linear-interpolation variant of Flow Matching, each training example is represented by its latent code x_T^i. Similar to the noisy sample in section 3.1, we construct a shifted initial noise vector by adding the feature-based shift S^i and projecting out the component of the Gaussian noise along this direction.

Forward path under linear flow Flow Matching defines a continuous path between the data representation x_T^i and the initial noise. Under linear interpolation, the latent state at time $t \in (0,1)$ becomes

$$x_t^i = t\, x_T^i + (1-t)(\epsilon_\perp + S^i).$$

Differentiating with respect to t gives the target velocity field:

$$v_t^{\text{target}} = \frac{d}{dt} x_t^i = x_T^i - (\epsilon_\perp + S^i). \tag{32}$$

Flow-matching objective with DNC conditioning The model predicts the instantaneous velocity $\epsilon_\theta(x_t^i, t)$, and the flow-matching loss becomes

$$\mathcal{L}_{\text{FM}}(\theta, D) = \mathbb{E}_{x_0^i,\, \epsilon,\, t}\left[\left\| x_T^i - (\epsilon_\perp + S^i) - \epsilon_\theta(x_t^i, t) \right\|^2 \right]. \tag{33}$$

The total training loss incorporates the shift-accuracy term $\mathcal{L}_{\text{shift}}$ and the orthogonality regularizer $\mathcal{L}_{\text{ortho}}$, both defined identically to the diffusion setting in section 3.1:

$$\mathcal{L}_{\text{train}}(\theta, D) = \mathcal{L}_{\text{FM}} + \mathcal{L}_{\text{shift}} + \mathcal{L}_{\text{ortho}}. \tag{34}$$

Inference At inference, the target shift S' is computed following eqs. (11) to (13), producing the shifted noise vector $x_0 = \epsilon_\perp + S'$. Given the predicted flow $\epsilon_\theta(x_t, t)$, we integrate the velocity field from $t = 0$ to $t = 1$ to obtain the final image. We use the classical Runge–Kutta integration scheme for stable and accurate trajectory integration.

Results We repeat the experiment on the *Arrows* dataset as described in section 4, using the same training setup and the same U-Net architecture as in the diffusion-based experiment. The generated solid-arrow samples produced by Flow Matching with DNC conditioning are shown in fig. 15. As can be seen, the model successfully follows the target annotations, precisely controlling the arrow direction and thickness while maintaining consistent shape and structure across samples. This demonstrates that DNC generalizes beyond diffusion models and can be seamlessly integrated into alternative generative frameworks such as Flow Matching.

We further quantify the conditioning accuracy by measuring the correlation between the predicted rotation angles and the ground-truth target angles, shown in fig. 16. The Flow-Matching model conditioned with DNC exhibits an almost perfect linear relationship, achieving an r^2 value of 0.9993 and an average ℓ_1 error of $1.12°$. This strong agreement confirms that our method preserves its precise control capabilities even when integrated into a generative framework fundamentally different from diffusion models.

N Additional Experimental Setups

Model components All methods share a latent diffusion backbone and differ only in the conditioning mechanism. The latent autoencoder is trained on the same data rather than taken from a pretrained checkpoint, frozen during diffusion training, and shared identically by all four methods. Parameter counts measured from the instantiated models are listed in table 8. The DNC mechanism adds only the shift directions and no architectural change to the U-Net: in an auxiliary 32×32 configuration, the U-Net alone has 918,017 parameters and DNC with $M = 4$ has 922,113, an overhead of 4,096 parameters (0.45%).

Table 8: Parameter counts of the model components (*BraTS2020* configuration).

Component	**Role**	**Params**
Latent autoencoder ($256 \to 64$, 1 latent ch.)	LDM encoder/decoder	10.7M
U-Net (denoises the $64 \times 64 \times 1$ latent)	Base diffusion model	331.5M
Feature VAE (table 12)	Unannotated feature discovery	77.7M
DNC directions ($M = 49$ vectors in $\mathbb{R}^{4096}$)	DNC mechanism	0.20M

Compute All models are trained on NVIDIA A100-40GB GPUs. The *Feature VAE* is a one-time, per-dataset pretraining (2000 epochs, table 10) required only in the unannotated setting; the annotated experiments (*Arrows*, *MNIST*) do not use it. In the unannotated setting, all compared methods consume the same *Feature VAE* latents, so this cost is shared rather than specific to DNC.

Hyperparameters The hyperparameters used to train the diffusion models and the *Feature VAE*s in all experiments are listed in table 9 and table 10, respectively.

Model Architecture The detailed architectures of the diffusion models and the *Feature VAE*s used in our experiments are summarized in table 11 and table 12, respectively.

Table 9: Training and sampling hyperparameters for diffusion models.

Hyperparameter	Value
Batch size	32
Learning rate	1×10^{-5}
Optimizer	Adam
Adam β_1	0.9
Adam β_2	0.999
Gradient clipping	1.0
Number of diffusion steps	100
Sampling steps (inference)	100
Noise scheduler	DDIM
Training epochs	2500
Image resolution (gray-scale)	$1 \times 256 \times 256$
Image resolution (RGB)	$3 \times 256 \times 256$
Latent space dimensions	$1 \times 64 \times 64$
Loss function	MSE
Precision	FP32

Table 10: Training hyperparameters for the *Feature VAE*.

Hyperparameter	Value
Batch size	8
Learning rate	2×10^{-5}
Optimizer	AdamW
AdamW β_1	0.9
AdamW β_2	0.999
Gradient clipping	None
Training epochs	2000
Input resolution (gray-scale)	$1 \times 64 \times 64$
Input resolution (RGB)	$3 \times 64 \times 64$
Latent space dimensions	$M' \times 1 \times 1$
KL divergence weight (β)	0.10
Reconstruction loss	MSE
Precision	FP32

Table 11: Architecture specification of the class-conditioned U-Net used in our diffusion models.

Component	Specification
Input channels	1
Output channels	1
Time embedding	512-dim sinusoidal + MLP (128 $\rightarrow$ 512 $\rightarrow$ 512)
Base channel width	128
Resolution levels	6
Channels per level	(128, 128, 256, 256, 512, 512)
Blocks per level	8 per block (ResNet2D)
Attention layers	Applied at levels 2–4 (cross-channel self-attention)
Downsampling	Strided Conv2D (kernel 3, stride 2)
Upsampling	Conv2D upsampling (nearest-neighbor + Conv)
Mid-block	2 × ResNet blocks + 1 attention block
Normalization	GroupNorm(32)
Activation	SiLU
Total depth	6 down + mid + 6 up blocks
Skip connections	U-Net skip concatenation

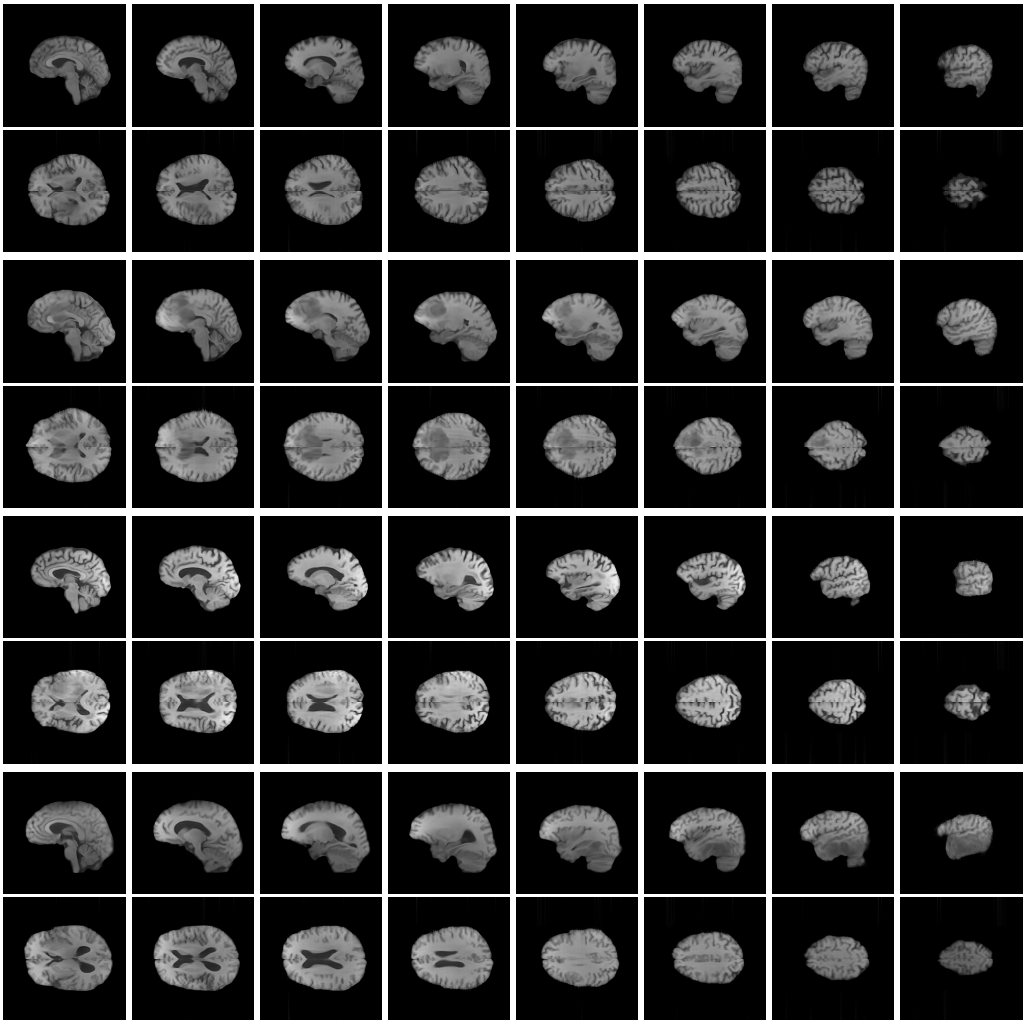


Figure 11: Additional randomly sampled generations from the diffusion model conditioned on the *BraTS2020* dataset. For each sample, we show a vertical slice and its corresponding horizontal slice obtained by stacking the vertical ones. The results demonstrate high fidelity and strong consistency across slices. Tumor regions naturally appear in several samples, reflecting the characteristics of the *BraTS2020* dataset.

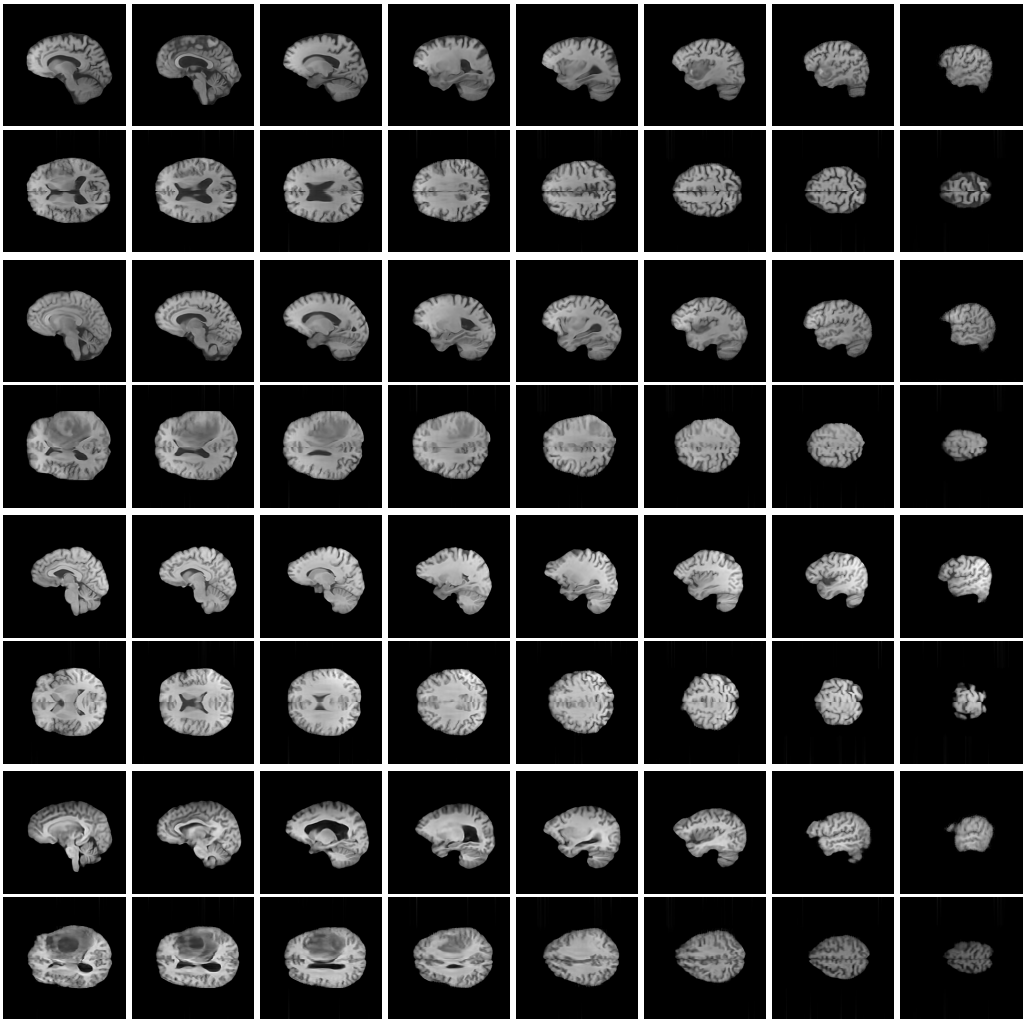


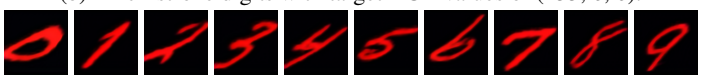
Figure 12: Additional randomly sampled generations from the diffusion model conditioned on the *BraTS2020* dataset (continued from fig. 11).



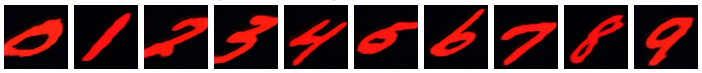
(a) Normal digits with target RGB values of (255, 0, 0).



(b) Thick-stroke digits with target RGB values of (255, 0, 0).



(c) Skewed digits with target RGB values of (255, 0, 0).



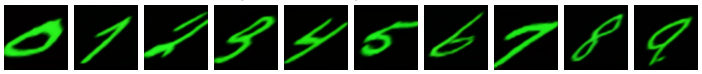
(d) Thick-stroke and skewed digits with target RGB values of (255, 0, 0).



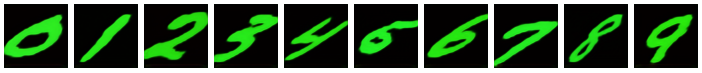
(e) Normal digits with target RGB values of (0, 255, 0).



(f) Thick-stroke digits with target RGB values of (0, 255, 0).



(g) Skewed digits with target RGB values of (0, 255, 0).



(h) Thick-stroke and skewed digits with target RGB values of (0, 255, 0).

Figure 13: **Generated *MNIST* samples conditioned on digit value, stroke thickness, skew, and RGB color.**

Table 12: Architecture specification of the *Feature VAE* used for unannotated feature learning.

Component	**Specification**
Input channels	1
Latent channels	M'
Encoder base channels	64
Encoder channel progression	(64, 128, 256, 256, 256, 256, 512)
Encoder blocks per level	48 ResNet blocks per level
Downsampling strategy	Strided Conv2D ($\times$6, stride 2)
Mid-block	2 ResNet blocks + 1 self-attention layer
Normalization	GroupNorm(32)
Activation	SiLU
Decoder base channels	512
Decoder channel progression	(512, 256, 256, 256, 256, 128, 64)
Decoder blocks per level	49 ResNet blocks per level
Upsampling strategy	Conv2D Upsample ($\times$6)
Decoder mid-block	2 ResNet blocks + 1 self-attention layer
Latent mapping layers	QuantConv (96$\rightarrow$96), PostQuantConv (48$\rightarrow$48)
Output channels	1
KL regularization	Enabled (AutoencoderKL)
Total depth	7 down levels + mid + 7 up levels
Attention usage	At bottleneck only



(a) Normal digits with target RGB values of (255, 165, 0).



(b) Thick-stroke digits with target RGB values of (255, 165, 0).



(c) Skewed digits with target RGB values of (255, 165, 0).



(d) Thick-stroke and skewed digits with target RGB values of (255, 165, 0).



(e) Normal digits with target RGB values of (20, 96, 120).



(f) Thick-stroke digits with target RGB values of (20, 96, 120).



(g) Skewed digits with target RGB values of (20, 96, 120).



(h) Thick-stroke and skewed digits with target RGB values of (20, 96, 120).

Figure 14: **Generated *MNIST* samples conditioned on digit value, stroke thickness, skew, and RGB color** (continued from fig. 13).

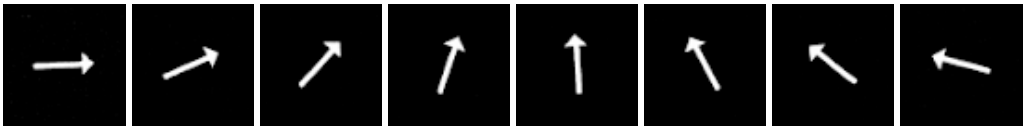


Figure 15: **Generated results for conditioning on annotated visual features (explicit conditioning) on *Flow Matching***.

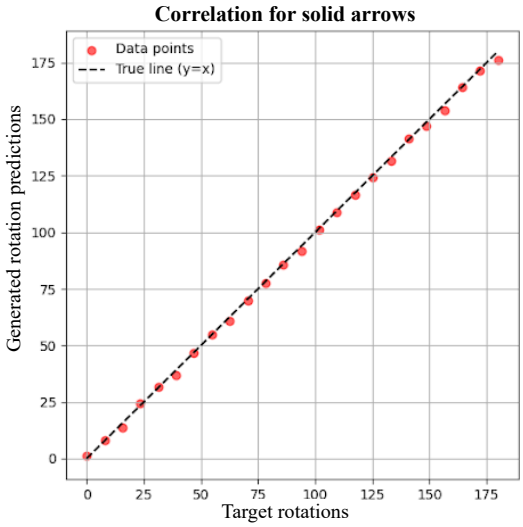
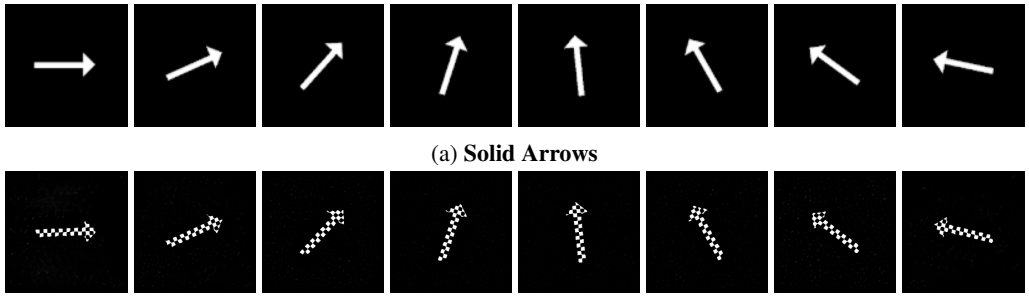


Figure 16: **Correlation between predicted and target rotation angles for Flow Matching.** The Flow-Matching model conditioned with DNC recovers target angles with extremely high accuracy, achieving $r^2 = 0.9993$ and $\ell_1 = 1.12°$.



(a) **Solid Arrows**

(b) **Textured Arrows**

Figure 17: **Conditioning on annotated visual features** (explicit conditioning).

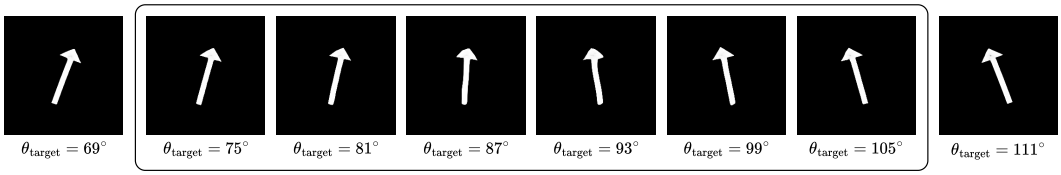


Figure 18: **Interpolation Results**. The figure shows generated arrows for target angles θ_{target}, where the gray box highlights the excluded range ($75° \leq \theta_{\text{target}} \leq 105°$).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction are supported by the experiments in section 4 and the appendix, and their scope is stated in section 5.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See section 5.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The paper contains no theorems; the conditional analysis of the forward process, its assumptions, and the scope of the claim are given in appendix B.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The method is fully specified in section 3, and all architectures, hyperparameters, and training details are given in appendix N.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: All datasets except Arrows are public and cited in section 4.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Datasets and evaluation protocols are described in section 4; hyperparameters, optimizers, and architectures are listed in appendix N.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Variability across evaluation snapshots is reported for the component ablation (appendix K).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Hardware, parameter counts, and training budgets are reported in appendix N.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The paper studies a conditioning mechanism for diffusion models at small scale and does not include a dedicated discussion of societal impact; as with other generative models, synthetic medical images should not be used in clinical settings without validation.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The proposed method does not involve high-risk usage.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All existing datasets (COIL-100, Thingi10K, BraTS2020, MNIST) are cited in section 4 and used under their respective research terms.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The synthetic Arrows dataset introduced in this work is described in section 4.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.