GENERATIVE MODEL VIA QUANTILE ASSIGNMENT

Georgi Hrusanov[1,2]*, **Oliver Y. Chén**[1,2], **and Julien S. Bodelet**[1,2,3]

[1]University of Lausanne, Lausanne, Switzerland

[2]Lausanne University Hospital (CHUV), Lausanne, Switzerland

[3]Department of Statistics, Stanford University, Stanford, USA

georgi.hrusanov@chuv.ch
olivery.chen@chuv.ch
julien.bodelet@chuv.ch

ABSTRACT

Deep Generative models (DGMs) play two key roles in modern machine learning: (i) producing new information (e.g., image synthesis) and (ii) reducing dimensionality. However, traditional architectures often rely on auxiliary networks such as encoders in Variational Autoencoders (VAEs) or discriminators in Generative Adversarial Networks (GANs), which introduce training instability, computational overhead, and risks like mode collapse. In this work, we present NeuroSQL, a new generative paradigm that eliminates the need for auxiliary networks by learning low-dimensional latent representations implicitly. NeuroSQL leverages an asymptotic approximation that expresses the latent variables as the solution to an optimal transportation problem. Specifically, NeuroSQL learns the latent variables by solving a linear assignment problem and then passes the latent information to a standalone generator. To demonstrate NeuroSQL's efficacy, we benchmark its performance against GANs, VAEs, and a budget-matched diffusion baseline on four independent datasets on handwritten digits (MNIST), faces from the CelebFaces Attributes Dataset (CelebA), animal faces from Animal Faces HQ (AFHQ), and brain images from the Open Access Series of Imaging Studies (OASIS). Compared to VAEs, GANs, and diffusion models: (1) in terms of image quality, NeuroSQL achieves overall lower mean pixel distance between synthetic and authentic images and stronger perceptual/structural fidelity, under the same computational setting; (2) computationally, NeuroSQL requires the least amount of training time; and (3) practically, NeuroSQL provides an effective solution for generating synthetic data when there are limited training data (e.g., data with a higher-dimensional feature space than the sample size). Taken together, by embracing quantile assignment rather than an encoder, NeuroSQL provides a fast, stable, and robust way to generate synthetic data with minimal information loss.

1 INTRODUCTION

Deep generative models (DGMs) have become a cornerstone of machine learning and have made meaningful contributions to image synthesis, data augmentation, and creative content generation. Over the past decade, they have also become well established in diverse scientific fields, such as genomics and neuroimaging, for complex data analysis tasks, including data interpretation, image/sequence decoding, and the generation of realistic datasets. A large share of these advances has been powered by variational autoencoders (VAEs; Kingma & Welling, 2014) and generative adversarial networks (GANs; Goodfellow et al., 2014), which remain the two dominant paradigms for generative modeling from lower-dimensional latent spaces. Both frameworks adopt a common strategy: pairing a generator with a complementary deep neural network (DNN). In VAEs, an encoder maps observations to latent variables, whereas in GANs, a discriminator provides adversarial feedback to train the generator.

*Corresponding author.

Despite their promise, training DGMs remains challenging, with practical and conceptual limitations. Practically, optimizing multiple networks simultaneously can be unstable and may lead to failures such as mode collapse. Training auxiliary networks that operate directly on high-dimensional observations is often more difficult and data-intensive than training the generator itself, which typically takes low-dimensional latent inputs. GANs, in particular, are prone to convergence failures such as mode collapse (Mescheder et al., 2018). VAEs often produce blurred reconstructions due to variational approximations and pixel-wise reconstruction losses such as mean squared error (MSE). More broadly, joint training of multiple deep networks increases sample complexity, computational cost, and instability, and these issues are exacerbated when data are high-dimensional relative to sample size.

Theoretically, there is no guarantee that the required encoder or discriminator, which works as a continuous mapping that a DNN can approximate, exists in the first place. While the generator depends on a latent variable that can have a much lower dimension than the observations, both the encoder and discriminator are functions of the data itself and may therefore encounter a curse of dimensionality (Stone, 1985). Although DNNs are believed to achieve fast convergence rates (Schmidt-Hieber, 2020) or, under favorable conditions, mitigate the curse of dimensionality (see e.g. Suzuki, 2019), these properties rely on strict assumptions (Golestaneh et al., 2025) that are not typically satisfied by DGMs. These limitations are reflected in theoretical work on VAEs and GANs, which often sidestep the resulting difficulties (Bodelet et al., 2025; Chae et al., 2023; Biau et al., 2020).

More recently, Denoising Diffusion Probabilistic Models (DDPMs) have set new benchmarks for image quality through iterative sampling. Yet diffusion models remain computationally intensive and require significant training time. In resource-constrained or low-data regimes, they often suffer from high-variance score estimates and diluted gradients.

To address the limitations in existing methods, we propose NeuroSQL, a latent-variable DGM that does not rely on amortized inference (encoders), adversarial feedback (discriminators), or iterative sampling (diffusion). Instead, NEUROSQL uses a discrete assignment mapping grounded in an asymptotic approximation: the unknown latent realizations can be represented, as $n \to \infty$, by a permutation of fixed (multi)variate quantiles of the assumed prior. This reduces latent estimation to identifying the permutation, which we show is identifiable and can be recovered by solving a linear assignment problem. In the multivariate setting, quantiles are constructed via an optimal-transport formulation, yielding a principled partition of the latent space into n regions. NEUROSQL generalizes Statistical Quantile Learning (SQL; Bodelet et al., 2025), which focused on additive models and univariate quantiles.

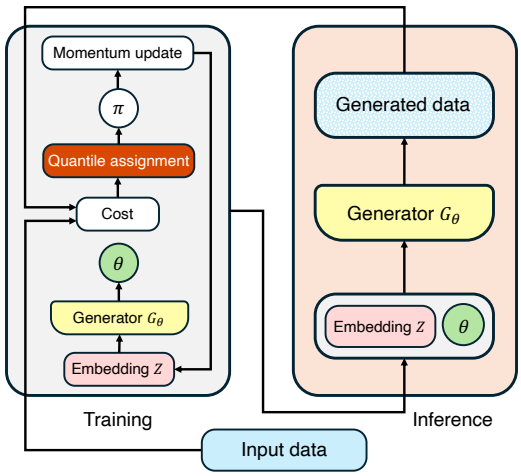


Figure 1: **A schematic representation of the NEUROSQL architecture.** Left: Algorithm for optimizing latent embeddings and generator parameters θ. Right: Data flow in NEUROSQL for data synthesis. Here, "cost" refers to cost matrix entries: $C_{i,k} = \ell\big(X_i, G_\theta(Q_k)\big)$ solved by the linear assignment problem. "Momentum update" indicates: $\hat{Z}_i^{(t)} \leftarrow \rho\, Q_{\pi^{(t)}(i)} + (1-\rho)\, \hat{Z}_i^{(t-1)}$.

In its essence, NEUROSQL trains a unique DNN for the generator, alongside an assignment procedure. Compared to existing DGMs, NEUROSQL offers several key benefits: It reduces the total parameter count, simplifies the training procedure to focus exclusively on the generator, avoids the curse of dimensionality typically encountered by auxiliary networks, and eliminates the instability challenges inherent in simultaneous multi-network optimization. Thus, NEUROSQL is fast, stable, and resource-friendly, consistently delivering superior image quality under matched computational constraints, proving the conceptual benefits of this new paradigm.

2 METHODOLOGY

2.1 THE MODEL

In what follows, we first state the model we aim to learn. We consider a dataset $\mathcal{D}$ composed of n independent samples of p-dimensional random vectors, $\mathcal{D} = \{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n\}$. We assume that the data are driven by unobserved continuous latent variables $\boldsymbol{Z}_i \in \mathcal{Z} \subseteq \mathbb{R}^d$. Specifically, we consider a probabilistic generative model:

$$\boldsymbol{X}_i = \boldsymbol{G}(\boldsymbol{Z}_i) + \boldsymbol{\epsilon}_i, \tag{1}$$

where $\boldsymbol{G} : \mathcal{Z} \to \mathbb{R}^p$ is an unknown generator, and $\boldsymbol{\epsilon}_i$ are independent random errors. We assume that the latent variables $\boldsymbol{Z}_i$ follow a known continuous distribution P_Z. The latent dimension is typically (much) smaller than the ambient dimension ($d << p$) to enable dimensionality reduction. This, therefore, contrasts with flow-based generative models, where the generator must be invertible. As DGMs are not identifiable, one can select any distribution as long as it has a continuous cumulative distribution function. Regarding the prior distribution, it is common to use either the standard normal distribution, $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, or a Uniform distribution. We approximate the generator using deep neural networks (DNN), that is $\boldsymbol{G} \approx \boldsymbol{G}_{\boldsymbol{\theta}}$, where:

$$\boldsymbol{G}_{\boldsymbol{\theta}} \in \{W_L \circ \sigma \circ W_{L-1} \circ \sigma \cdots \circ \sigma \circ W_1\},$$

and W_l denote affine transformations and σ is an activation function. The vector $\boldsymbol{\theta} \in \Theta$ contains the DNN parameters. We aim to learn both the generator parameter $\boldsymbol{\theta}$ and the latent variables $\boldsymbol{Z}_i$.

2.2 LATENT SPACE APPROXIMATION AND LOSS FUNCTION

For deep generative models, likelihood-based estimation is typically tractable only in simple settings (e.g., linear factor analysis). Jointly learning the generator and latent variables is, therefore, computationally demanding. In NEUROSQL we adopt an approach inspired by the sieve method, which replaces an intractable optimization over a complex parameter space by tractable problems on a growing sequence of simpler subspaces that are dense in the original space (see, e.g., Chen (2007)).

NEUROSQL aims to approximate the latent space to obtain a tractable problem. It partitions the latent space into n regions via quantiles $\boldsymbol{Q}_1^n, \ldots, \boldsymbol{Q}_n^n \in \mathcal{Z} \subseteq \mathbb{R}^d$. Let $\boldsymbol{Z} := (\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_n)'$ and $\boldsymbol{Q}^n := (\boldsymbol{Q}_1^n, \ldots, \boldsymbol{Q}_n^n)'$ denote the $n \times d$ matrices of latent variables and quantiles, respectively. The key idea of NEUROSQL is that there exists a permutation π such that

$$\boldsymbol{Z} \approx \boldsymbol{Q}_{\pi}^n, \tag{2}$$

where, for a matrix (or column vector) $\boldsymbol{A}$ and a permutation π, $\boldsymbol{A}_\pi$ denotes the matrix obtained by permuting the rows of $\boldsymbol{A}$ according to π.

Since $\mathbb{R}^d$ has no canonical ordering, there is no universal definition of multivariate quantiles. For clarity, we formalize the approximation in equation 2 for the univariate case ($d = 1$); the d-dimensional construction is given in Section 2.3.

For $d = 1$, define the n quantiles $\boldsymbol{Q}_i^n = F^{-1}\left(\frac{i}{n+1}\right)$, $i = 1, \ldots, n$, where F is the CDF of P_Z. Using the delta method (see, e.g., van der Vaart 2000), one has $\left|\boldsymbol{Z}_{(i)} - \boldsymbol{Q}_i^n\right| = \mathcal{O}_p\left(\frac{1}{\sqrt{n}}\right)$, where $\boldsymbol{Z}_{(i)}$ denotes the order statistics of the latent variables. Since the $\boldsymbol{Z}_{(i)}$'s are distinct almost surely, they are almost surely a permutation of the $\boldsymbol{Z}_i$'s, which implies the approximation error bound

$$\min_{\pi \in S_n} \frac{1}{n} \sum_{i=1}^{n} \left|\boldsymbol{Z}_i - \boldsymbol{Q}_{\pi(i)}^n\right|^2 = \mathcal{O}_p\left(\frac{1}{n}\right), \tag{3}$$

with S_n the symmetric group of order n. Hence, the latent variables can be approximated by learning an appropriate permutation of the quantiles; the approximation error in equation 3 vanishes as $n \to \infty$.

Leveraging this approximation, we consider the criterion $\mathcal{L}(\boldsymbol{\theta}, \pi) = \frac{1}{n} \sum_{i=1}^{n} \ell\left(\boldsymbol{X}_i, \boldsymbol{G}_{\boldsymbol{\theta}}\left(\boldsymbol{Q}_{\pi(i)}^n\right)\right)$ for a loss $\ell : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}_+$. We define the NEUROSQL solutions by

$$(\hat{\boldsymbol{\theta}}, \hat{\pi}) = \underset{\boldsymbol{\theta} \in \Theta,\ \pi \in S_n}{\arg\min}\ \mathcal{L}(\boldsymbol{\theta}, \pi)\ +\ \lambda \mathcal{R}(\boldsymbol{\theta}), \tag{4}$$

where $\mathcal{R}(\boldsymbol{\theta})$ controls the complexity of $\boldsymbol{G}_{\boldsymbol{\theta}}$ and $\lambda > 0$ is a tuning parameter. Given $\hat{\pi}$, the latent-variable estimator is $\hat{\boldsymbol{Z}} = \boldsymbol{Q}^n_{\hat{\pi}}$.

2.3 MULTIVARIATE QUANTILES ($d > 1$)

Building on the univariate case, here we extend NEUROSQL to $d > 1$. We discuss how NEUROSQL approximates the latent space in higher dimensions and show that the approximation error vanishes as $n \to \infty$. As there is no canonical ordering when $d > 1$, several methods have been developed to construct "multivariate quantiles". We concentrate on recent developments that leverage the optimal transport approach (Hallin, 2022; Chernozhukov et al., 2017; Hallin et al., 2021; Ghosal & Sen, 2022), which offer a conceptually clean and practical way to define quantiles in multivariate dimensions.

To build those multivariate quantiles, we use a regular grid, $\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_n \in \mathcal{U}_d$, where $\mathcal{U}_d := \{\boldsymbol{z} \in \mathbb{R}^d : \|\boldsymbol{z}\| < 1\}$ is the unit ball. This grid does not have to be perfectly regular, as it is in general not possible for $d \geq 3$. We require only that the discrete distribution with probability $1/n$ at each grid point converges weakly to the uniform distribution over $\mathcal{U}_d$. In practice, it is suitable to select a grid with a low discrepancy in order to obtain fast convergence rates.

In the case where the latent variables are uniformly distributed over $\mathcal{U}_d$, we define the multivariate quantiles as $\boldsymbol{Q}^n_i := \boldsymbol{U}_i$. For more general distributions F, we define the multivariate quantiles as $\boldsymbol{Q}^n_i := F^{-1}_{\pm}(U_i)$, where $F_{\pm}$ denotes the center-outward distribution function. Specifically, $F_{\pm}$ is the unique gradient of a convex function pushing P_Z forward to the uniform distribution over the unit ball. We refer to Hallin et al. (2021) and Hallin (2022) for a detailed explanation. Without loss of generality, we will assume here that the distribution P_Z is uniform over $\mathcal{U}_d$, yielding $\boldsymbol{Q}^n_i = \boldsymbol{U}_i$. This is reasonable because the latent distribution of (deep) generative models is not identifiable and should be selected (we refer to Bodelet et al., 2025 for a discussion).

The following proposition shows that the approximation error also vanishes asymptotically in the multivariate case.

Proposition 1. *Assume that the discrete distribution with probability $1/n$ at each grid point $\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_n \in \mathcal{U}_d$ converges weakly to the uniform distribution over $\mathcal{U}_d$. Then, as $n \to \infty$, the following holds:*

$$\min_{\pi \in S_n} \|\boldsymbol{Z} - \boldsymbol{Q}^n_{\pi}\|^2 \to 0, \text{ a.s.}$$

The proof of Proposition 1 is in the Appendix K, following Theorem 2.4 in Hallin et al. (2021).

2.4 COMPUTATIONAL ALGORITHM

We note that, for fixed $\boldsymbol{\theta}$, the inner problem in Eq 4 can be formulated as a *linear assignment problem*:

$$\min_{\pi \in S_n} \frac{1}{n} \sum_{i=1}^{n} C_{i,\pi(i)}, \qquad C_{i,k} := \ell\big(\boldsymbol{X}_i, \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{Q}_k)\big),$$

where $\boldsymbol{C} := (C_{i,k})_{1 \leq i,k \leq n}$ is the cost matrix. This is solvable *exactly* by the Hungarian method in $O(n^3)$ time (Kuhn, 1955), but see Sec 2.5 where one can reduce complexity to $O(n^2)$ and also do assignment in mini-batches. Furthermore, for fixed π, Eq 4 reduces to standard supervised regressions of $\boldsymbol{X}$ on assigned codes $\{z_{\pi(i)}\}$. We therefore solve Eq 4 as follows: (i) Given a permutation π, minimize the loss function with respect to $\boldsymbol{\theta}$ (Generator step); (ii) Given $\boldsymbol{\theta}$, we solve the linear assignment matching problem (via Hungarian $O(n^3)$ or Greedy method) which reduces the complexity to $O(n^2)$. We iterate (i) and (ii) until convergence. Furthermore, we introduce a momentum update after each assignment, that is $\widehat{z}^{(t)} \leftarrow \rho\, z_{\pi^{(t)}(i)} + (1-\rho)\, \widehat{z}^{(t-1)}$ for some $0 \leq \rho < 1$, in order to stabilize training. The exact steps are detailed in Algorithm 1.

2.5 COMPLEXITY AND SCALABILITY

Each outer iteration forms the cost matrix $C^{(t)}$ using n decoder forward passes over z_k and then solves a single assignment problem, for a total complexity of $O(n \cdot c_G + n^3)$, where c_G is the

Algorithm 1 NEUROSQL (full-batch assignment)

Input: data $\{\boldsymbol{X}_i\}_{i=1}^n$, prior P_Z, lattice $\boldsymbol{Q}^n$, outer iters T, momentum $\rho \in [0, 1)$, $\lambda > 0$
Initialize $\pi^{(0)}$ (e.g., random or PCA-sorted);
  set $\widehat{\boldsymbol{Z}}^{(0)} = (\boldsymbol{Q}^n)_{\pi^{(0)}}$
for $t = 1, \ldots, T$ **do**
  Decoder step:
  $$\boldsymbol{\theta}^{(t)} \in \arg\min_{\boldsymbol{\theta}} \left[\frac{1}{n} \sum_{i=1}^{n} \ell\big(\boldsymbol{X}_i, \boldsymbol{G}_{\boldsymbol{\theta}}(\widehat{\boldsymbol{Z}}_i^{(t-1)})\big) + \lambda \mathcal{R}(\boldsymbol{\theta}) \right]$$
  Cost matrix: $C_{i,k}^{(t)} \leftarrow \ell(\boldsymbol{X}_i, \boldsymbol{G}_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{Q}_k))$
  Assignment: $\pi^{(t)} \leftarrow \arg\min_{\pi \in S_n} \mathrm{Tr}(\boldsymbol{C}_{\pi}^{(t)})$
  Momentum: $\widehat{\boldsymbol{Z}}_i^{(t)} \leftarrow \rho\, (\boldsymbol{Q}^n)_{\pi^{(t)}(i)} + (1 - \rho)\, \widehat{\boldsymbol{Z}}_i^{(t-1)}$
end for
Output: $\widehat{\boldsymbol{\theta}} = \boldsymbol{\theta}^{(T)}$, $\widehat{\boldsymbol{Z}} = \widehat{\boldsymbol{Z}}^{(T)}$.

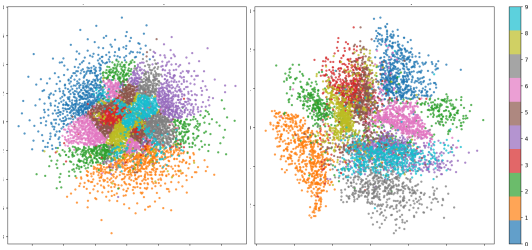
decoder's forward cost. Fundamentally, the assignment step is independent of the data dimension p, which is why NEUROSQL scales favorably to very high-dimensional observations.

The $O(n^3)$ Hungarian algorithm can become a bottleneck with large n. To reduce this overhead (especially for $n \gtrsim 10^4$), we also propose a simple greedy assignment with $O(n^2)$ complexity (see Section G for details). In our benchmarks, the greedy strategy provides a $6.54\times$ speedup, reducing the mean time per call from $\approx$ 89ms to $\approx$ 13ms when loading the full batch, without a noticeable change in performance.

Across datasets, this substitution does not materially affect the qualitative quality of generated samples, making greedy assignment a practical choice in higher-throughput settings. For ultra-large datasets, one can further run NEUROSQL in a mini-batch regime with batch size $m \ll n$, reducing assignment cost to $O(m^2)$ and making the computation of NEUROSQL independent of the total dataset size.

2.6 INTERPRETABILITY OF THE LATTICE CODES

NEUROSQL replaces the stochastic VAE encoder with a deterministic and explainable assignment procedure, hypothesizing that removing variational noise yields a more structured latent space. To probe this, we visualize 2D MNIST embeddings for NEUROSQL and a VAE baseline with the same generator. As shown in Fig. 2b, the VAE latent space forms a characteristic "fuzzy cloud" under the Gaussian prior $\mathcal{N}(0, I)$: the KL regularization term crowds samples toward the origin, and class regions overlap substantially. This phenomenon is most prominent in the dense center, where digits such as 3, 5, and 8 become difficult to disentangle. Thus, a more precise latent analysis becomes harder.



(a) NEUROSQL (b) VAE

Figure 2: **A Comparison of Latent Space of NEUROSQL and VAE.** Visualization of the two-dimensional latent space obtained from MNIST using NEUROSQL (left) and VAE (right). NeuroSQL's latent space forms more distinct, better-separated clusters than those of the VAE, whose clusters show more overlaps.

In contrast, NeuroSQL (Fig. 2a) produces a more separated geometry. By approximating latents via a linear assignment to a fixed, high-variance lattice, it uses the space more evenly (roughly $[-8, 8]$ versus $[-3, 3]$ for the VAE) and organizes digit classes into compact, well-isolated "islands" (e.g., '1' and '0' form tight clusters with minimal spillover). This suggests that the Optimal-Transport-based assignment preserves semantic discreteness without supervision and reduces the overlap seen with the VAE, supporting a more transparent mapping between latent codes and generated samples.

3 EXPERIMENTS

We evaluate NEUROSQL under a *sparse-resource* regime across four data domains of increasing structural complexity: digits (MNIST), human faces (CelebA), animal faces (AFHQ), and brain imaging (OASIS). More details on the data are in Section C.

For comparison, we consider ConvNet, ResNet, and U-Net generators. Our emphasis lies on *paradigm evaluation*: models share (as closely as possible) the same generator backbone, data budgets, and optimization schedules, so that if there are any differences, they are likely to arise from the learning principle (quantile–assignment vs. probabilistic/adversarial/denoising) rather than model capacity or compute power. As part of the evaluation pipeline, we train a downstream classifier using synthetic images.

3.1 MODELS AND TRAINING

To ensure that performance differences are not attributable to architectural or training advantages, we compare NeuroSQL, VAE, and GAN under the same three models: ConvNet(baseline), ResNet, and U-Net. Since Diffusion Probabilistic Models require a specific denoising architecture (usually MLP or a U-Net) and cannot use the controlled generator backbone(ConvNet/ResNet) shared across NEUROSQL and the other two baselines, we matched the DDPM by scaling its width. This was achieved by implementing the DDPM with a custom denoising U-Net whose channel width was scaled to match the trainable parameter count of the competing generators within a $\pm 10\%$ margin, ensuring comparable model capacity.

For all methods, we matched computational budgets by fixing the total number of training iterations and using identical optimizer schedules (AdamW) and early stopping, thereby strictly controlling for both model size and compute time. Implementation details are in Appendix D.

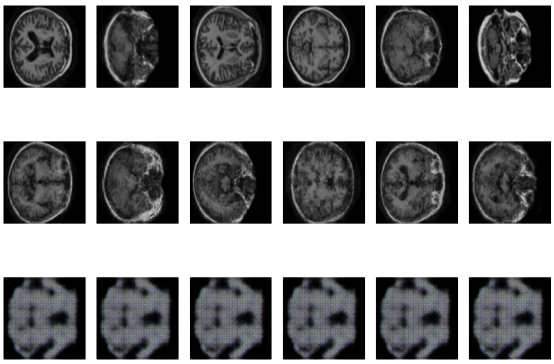


Figure 3: 2D brain images generated from NEUROSQL (top row), VAE (middle row) and GAN (bottom row) with U-Net generator.

3.2 EVALUATION

To quantify the model performance, we report, for each scenario, a proxy of FID (Fréchet Inception Distance; lower is better), LPIPS (Learned Perceptual Image Patch Similarity; lower is better), and SSIM (Structural Similarity Index; higher is better). Discussions on the metrics and datasets are in Sections C and E of the Appendix. We present the results across a wide range of experimental specifications in Appendix M.

3.3 RESULTS

For demonstration, we compare and discuss NEUROSQL's performance against that of VAE and GAN on generating brain imaging data and MNIST digits. We include a comparison between NEUROSQL and diffusion models as well as model performance of these models on human and animal faces in the Appendix.

For the brain imaging data (OASIS), we observe that, overall, NEUROSQL outperforms both VAE and GAN across various combinations of latent dimension and generator. Particularly, NEUROSQL outperforms VAE and GAN in terms of LPIPS (measuring perceptual similarity between images (i.e., how similar they look to humans) and SSIM (measuring pixel-level structural similarity) scores in all scenarios. All models with a ConvNet generator yield much better results compared to those with a ResNet. The choice of generator, however, has a marginal effect on FID scores. For FID, which measures distribution similarity between generated and real images in feature space and pixel distance, NEUROSQL outperforms VAE and GAN in all cases with a ResNet generator. It outperforms VAEs and GANs when the latent dimension is moderate (between 16 and 64) with a ConvNet

generator. When the latent dimension is very small or very large, VAE with ConvNet shows only modest improvement. With a U-Net generator, NEUROSQL attains the best LPIPS and SSIM, while the FID (proxy) varies and is lowest for VAE on average.

Taken together, our results demonstrate the overall effectiveness of NEUROSQL compared to existing methods in generating synthetic data across different domains that are unrelated to each other. In particular, NEUROSQL with a ConvNet generator achieves superior performance across different scenarios, particularly in metrics evaluating perceptual similarity between images (how similar they look to humans) and pixel-level structural similarity, and its performance improves as the dimension of its latent space increases.

A more comprehensive quantitative comparison is detailed in Table 1, which reports the mean performance metrics across all tested architectures and latent dimensions. These results confirm that NEUROSQL delivers a robust balance of image quality and structural similarity across digits, faces, and brain imaging data with the smallest amount of trainable parameters.

Table 1: Performance comparison across datasets (mean $\pm$ std). Lower is better for proxy FID; higher is better for SSIM.

Dataset (Res.)	Model	pFID↓	SSIM↑	P
MNIST (28×28)	NEUROSQL	**0.61±0.12**	**0.616±0.074**	**2.79**
	VAE	1.26±0.26	0.179±0.030	4.17
	GAN	2.00±0.22	0.233±0.018	5.56
	Diffusion	0.65±0.34	0.492±0.055	147.91
CelebA (64×64)	NEUROSQL	**5.81±4.05**	**0.262±0.036**	**7.54**
	VAE	10.75±7.92	0.196±0.047	14.03
	GAN	18.02±6.78	0.137±0.034	10.33
	Diffusion	23.79±8.11	0.013±0.066	147.91
AFHQ (128×128)	NEUROSQL	**19.03±11.31**	**0.290±0.056**	**119.58**
	VAE	39.15±30.64	0.190±0.051	144.98
	GAN	46.00±14.83	0.082±0.029	122.37
	Diffusion	22.99±14.83	0.0388±0.032	147.91
OASIS (128×128)	NEUROSQL	**16.36±12.25**	**0.252±0.010**	**243.74**
	VAE	24.24±24.46	0.196±0.058	269.09
	GAN	68.23±15.52	0.145±0.063	246.50
	Diffusion	21.22±23.71	0.04759±0.058	252.66

3.4 MNIST DOWNSTREAM CLASSIFIER

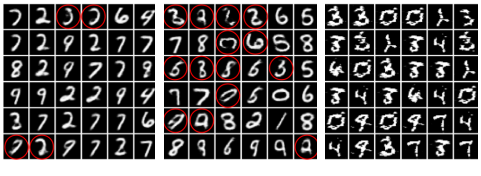
To further evaluate the quality of the generated samples from NeuroSQL, we train a downstream classifier using sampled images. We trained a standard CNN classifier solely on 10k synthetic images generated by NEUROSQL and evaluated it on real MNIST data. The model achieved **87.71% accuracy, 0.88 precision, and 0.88 recall**. Notably, our method significantly outperforms the VAE baseline, which achieved 67.49% accuracy under the same conditions. Therefore, the generated samples likely effectively capture the class-conditional distributions of the underlying manifold.

Table 2: Downstream classification performance on MNIST. A classifier was trained solely on synthetic data (10k samples) generated by NEUROSQL and VAE and evaluated on real test data.

Method	Prec.	Rec.	Acc.
NEUROSQL	**0.8788**	**0.8763**	**0.8771**
VAE	0.7236	0.6705	0.6749

3.5 COMPUTE BUDGET, DATA SCALE, AND EVALUATION SCOPE

All experiments are designed to run end-to-end on a single Google Colab under a fixed allowance of *200 compute units*. Within this budget, we cap the training set at 2000 images and focus on resolutions in the range 64×64–128×128 (apart from MNIST), which in turn controls model capacity and the number of optimizer updates. Under these conditions, diffusion models underperform: with $T=1000$ diffusion steps and $N\approx1000$–2000 training images, supervision per noise level scales as $O(N/T)$, producing high-variance score estimates; moreover, train-



(a) NEUROSQL (b) VAE (c) GAN

Figure 4: Qualitative comparison of 36 randomly generated images for models trained on MNIST.

ing and sampling cost scale with T, which further reduces effective learning progress at a fixed wall-clock budget.

Despite the advantages of NeuroSQL, we highlight that our main goal with this work is a **paradigm-level proof of concept**. We aim not to compete with large-scale, high-resolution diffusion pipelines, but to demonstrate that NEUROSQL provides a viable training principle for generative modeling in compute- and data-frugal regimes, such as in small research labs and non-profit organizations where pretraining, heavy augmentation, or distillation entail substantial challenges.

4 RELATED WORK

Discrete latent representations and vector quantization. Discrete latent space models, including VQ-VAE (van den Oord et al., 2017), VQ-VAE-2 (Razavi et al., 2019), and VQ-GAN (Esser et al., 2021), utilize learned codebooks to map inputs to discrete vectors, a paradigm further advanced by masked (Chang et al., 2022) and autoregressive (Tian et al., 2024) modeling. While NEUROSQL also employs discrete representations, it fundamentally departs from this framework by replacing the learned encoder and its associated sample complexity with a fixed *a priori* quantile lattice, determining embeddings via combinatorial assignment rather than gradient-based inference.

Optimal Transport and Rectified Flows. A related and rapidly evolving line of work addresses this bottleneck by combining generative modeling with optimal transport (OT). Rectified Flows (Liu et al., 2023) formulate generative modeling as learning straight ODE trajectories between noise and data distributions, with subsequent work improving training efficiency (Lee et al., 2024), achieving one-step generation (Geng et al., 2025), and enabling forward-only regression training (Rehman et al., 2025). These methods share a common paradigm: parameterizing continuous neural transport maps optimized to minimize Wasserstein distance, enabling few-step sampling. NEUROSQL uses OT fundamentally differently. Rather than learning continuous trajectories, we use OT-based multivariate quantiles (Hallin et al., 2021; Chernozhukov et al., 2017; Hallin, 2022; Ghosal & Sen, 2022) to define a fixed lattice that partitions the latent space, then solve discrete assignment problems to match data to quantiles.

DGM for low sample regimes. Existing approaches for low-sample generation, ranging from data augmentation to specialized architectures like FastGAN (Liu et al., 2021), generally remain within paradigms that require auxiliary networks to process high-dimensional data. Consequently, these methods face severe sample complexity challenges when the sample size is small relative to the data dimension ($n \ll p$). NEUROSQL addresses this bottleneck by eliminating encoders and discriminators, thereby sidestepping the difficulty of learning functions from high-dimensional observation spaces.

5 CONCLUSIONS

In this paper, we introduced NeuroSQL, a deep generative model that replaces stochastic encoders with rank-based quantile assignment. By learning embeddings through assignment rather than amortised inference, NEUROSQL eliminates the posterior collapse typical of VAEs and avoids the adversarial dynamics of GANs. Furthermore, the deterministic assignment mechanism yields distributions substantially more interpretable than those of GANs, diffusion models, or VAEs. Unlike diffusion models, which require extensive denoising and large datasets, NEUROSQL demonstrates superior performance in constrained settings (under 10^5 samples) by avoiding high-variance score estimation. Evaluated on MNIST, CelebA, AFHQ, and OASIS, the model proves consistently competitive and often superior under matched experimental settings.

The primary contribution of NEUROSQL is the introduction of a new principle for developing DGMs, prioritizing methodological innovation over high-capacity architecture. While our experiments validated the paradigm under controlled conditions, future work can evaluate its scalability to higher resolutions and comparisons against baselines like StyleGAN3. In terms of applications, particularly in medical data science, a natural next step is to experiment with the MedMNIST (Yang et al., 2023) benchmark suite. Moving forward, we foresee two key directions for NeuroSQL: (i) scaling the assignment mechanism to new modalities (e.g., audio, 3D), and (ii) expanding the theory of quantile-assignment training beyond its current empirical scope.

REFERENCES

Ninad Aithal. Oasis alzheimer's detection (images). Kaggle dataset, 2023. URL https://www.kaggle.com/datasets/ninadaithal/imagesoasis. Accessed 2025-09-23.

Gérard Biau, Benoît Cadre, Maxime Sangnier, and Ugo Tanielian. Some theoretical properties of gans. *The Annals of Statistics*, 48(3):1539–1566, 2020.

Julien Bodelet, Guillaume Blanc, Jiajun Shan, Graciela Muniz Terrera, and Oliver Y Chén. Statistical quantile learning for large additive latent variable models. *Journal of the American Statistical Association*, pp. 1–22, 2025.

Minwoo Chae, Dongha Kim, Yongdai Kim, and Lizhen Lin. A likelihood approach to nonparametric estimation of a singular distribution using deep generative models. *Journal of Machine Learning Research*, 24(77):1–42, 2023.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. MaskGIT: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11315–11325, 2022.

Xiaohong Chen. Large sample sieve estimation of semi-nonparametric models. In *Handbook of Econometrics*, volume 6, pp. 5549–5632. Elsevier, 2007.

Victor Chernozhukov, Alfred Galichon, Marc Hallin, and Marc Henry. Monge–kantorovich depth, quantiles, ranks and signs. *The Annals of Statistics*, 45:223–256, 2017.

Yunjey Choi, Minje Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8188–8197, 2020.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12873–12883, 2021.

Zhiwei Geng, Mingyuan Deng, Xiaopeng Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

Promit Ghosal and Bodhisattva Sen. Multivariate ranks and quantiles using optimal transport: Consistency, rates and nonparametric testing. *The Annals of Statistics*, 50(2):1012–1037, 2022.

Pegah Golestaneh, Mahsa Taheri, and Johannes Lederer. How many samples are needed to train a deep neural network? In *The Thirteenth International Conference on Learning Representations*, 2025.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27, 2014.

Marc Hallin. Measure transportation and statistical decision theory. *Annual Review of Statistics and Its Application*, 9(1):401–424, 2022.

Marc Hallin, Eustasio Del Barrio, Juan Cuesta-Albertos, and Carlos Matrán. Distribution and quantile functions, ranks and signs in dimension d: A measure transportation approach. *The Annals of Statistics*, 49:1139–1165, 2021.

Diederik P Kingma and Max Welling. Stochastic gradient vb and the variational auto-encoder. In *International Conference on Learning Representations*, 2014.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2002.

Sangyun Lee, Zinan Lin, and Giulia Fanti. Improving the training of rectified flows. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Bingchen Liu, Yizhe Zhu, Kunpeng Song, and Ahmed Elgammal. Towards faster and stabilized GAN training for high-fidelity few-shot image synthesis. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=1Fqg133qRaI.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 3730–3738, 2015.

Daniel S. Marcus, Tracy H. Wang, Jamie Parker, John G. Csernansky, John C. Morris, and Randy L. Buckner. Open access series of imaging studies (oasis): Cross-sectional mri data in young, middle aged, nondemented, and demented older adults. *Journal of Cognitive Neuroscience*, 19(9):1498–1507, 2007. doi: 10.1162/jocn.2007.19.9.1498.

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 3481–3490, 2018.

Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. *Advances in Neural Information Processing Systems*, 32, 2019.

Danyal Rehman, Oscar Davis, Jiarui Lu, Jian Tang, Michael M Bronstein, Yoshua Bengio, Alexander Tong, and Joey Bose. FORT: Forward-only regression training of normalizing flows. *arXiv preprint arXiv:2502.10818*, 2025. Also in ICML 2025 Workshop on Generative Biology.

Johannes Schmidt-Hieber. Nonparametric regression using deep neural networks with relu activation function. *The Annals of Statistics*, 48(4):1875–1897, 2020.

Charles J Stone. Additive regression and other nonparametric models. *The Annals of Statistics*, 13 (2):689–705, 1985.

Taiji Suzuki. Adaptivity of deep relu network for learning in besov and mixed smooth besov spaces: optimal rate and curse of dimensionality. In *The 7th International Conference on Learning Representations (ICLR2019)*, volume 7, 2019.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Yang, and Wanli Peng. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Aad W van der Vaart. *Asymptotic statistics*, volume 3. Cambridge University Press, 2000.

Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2 - a large-scale lightweight benchmark for 2d and 3d biomedical image classification. *Scientific Data*, 10(1), January 2023. ISSN 2052-4463. doi: 10.1038/s41597-022-01721-8. URL http://dx.doi.org/10.1038/s41597-022-01721-8.

A ACKNOWLEDGMENTS

This project is partly funded by the Swiss National Science Foundation (SNSF) grants 3200-0-239967 and CR00I5-235987. We acknowledge the use of large language models for grammar checking, punctuation correction, spelling verification, and synonym suggestions to enhance writing clarity.

B ETHICAL AND DOMAIN-SPECIFIC NOTES (OASIS)

In order to keep the experimental setup free of data leakage within the train/val/test splits, we performed subject-stratified cross-validation. We highlight that one must not equate synthetic imaging data with clinical data. Synthetic imaging data, however, may have practical utilities, such as for treating missing data, but this is beyond the scope of this paper. Here, we generate synthetic data to demonstrate the efficacy of NEUROSQL as a generative model; we use the generated imaging data to evaluate the model's performance; we do not claim its clinical utility. Future work should verify this independently, and we will release seeds, splits, and scripts to facilitate further validations.

C DATASETS

MNIST (LeCun et al., 2002). Handwritten digits (60k train, 10k test, 28×28 grayscale). We replicate to RGB for evaluation only.

CelebA (Liu et al., 2015). Face dataset ($202{,}599$ images in full) of which we use around 2500 by preprocessing them. We use them center-cropped and resized to 128×128. Training is unconditional despite available attributes.

AFHQ (Choi et al., 2020). Animal Faces HQ contains $15{,}000$ high-quality animal face images across cats, dogs, and wildlife at 512×512 resolution originally. Within experiments, image size was reduced to 128×128 and total number of images used was about 2000.

OASIS (Marcus et al., 2007; Aithal, 2023). Neuroimaging dataset with $\sim$ 80k MRI slices from 416 subjects, downsampled to 128×128. Medical images test resistance to overfitting in constrained domains. We used the version of OASIS that is publicly available on Kaggle containing a chunk of the whole dataset. It is pre-structured and is organized in four subfolders, namely: Mild Dementia, Moderate Dementia, Non Demented, and Very Mild Demented. We use it for the fact that it is publicly available.

D IMPLEMENTATION DETAILS

Generator backbones. Within each experimental domain, NeuroSQL, VAE, and GAN share the ***identical*** generator architecture and initialization. The following generators were used.

ConvNet (Baseline) is a parameter-efficient standard deconvolutional network (similar to DCGAN). It consists of a linear projection followed by a stack of transposed convolution layers with halving channel widths ($512\to256\to128\to\dots$) and batch normalization. **ResNet:** A high-capacity generator comprising a linear projection followed by four residual upsampling blocks. Unlike the baseline, this backbone maintains a *constant channel width* of 512 throughout all blocks. **U-Net:** A conditional architecture where the latent vector z modulates a learnable constant input tensor via Feature-wise Linear Modulation (FiLM) layers injected at every resolution level. To keep the evaluation pipeline of the work fair, for diffusion we employ a standard U-Net, scaling the channel width so that the total trainable parameter count remains within $\pm10\%$ of the competing generator.

Budget parity. All methods are trained under the same compute budget per dataset: identical number of epochs, batch size, optimizer (AdamW), learning-rate schedule (cosine decay), gradient clipping, and early stopping criteria.

Latent dimensionality sweep. We vary the latent dimension $q \in \{2, \dots, 128\}$ to assess sensitivity to representation size and to verify that trends are not driven by a particular latent choice. Although we conducted experiments with a latent dimension of 128, our experiments suggest that such high dimensionality is unnecessary to capture the intrinsic geometry of the datasets studied.

Resolution ablation. On OASIS, we evaluate two spatial resolutions, 64×64 and 128×128, to probe robustness to increased output dimensionality.

Loss sensitivity for assignment. We ablate the cost used to form the assignment matrix, comparing a pixelwise ℓ_2 loss against our default perceptual cost $\ell = \frac{1}{2}(1 - \text{SSIM}) + \frac{1}{2}\|\cdot\|_1$. Unless stated otherwise, the main results use the SSIM+ℓ_1 cost (additional details in the Appendix).

Training of NeuroSQL. For NEUROSQL we used the Sobol or Uniform lattice in $[0, 1]^d$ to construct the quantiles. We performed the *Hungarian* or *Greedy* algorithm at every K epochs (with $K \in \{2, 3, 5\}$ treated as a hyperparameter), and selected the momentum parameter as ρ=0.7.

Additional details. To ensure computational parity, we standardize the optimization protocol across all methods using AdamW, cosine decay, gradient clipping, and early stopping. Full architectural details and hyperparameters are provided in Appendix I

E EVALUATION METRICS

FID (proxy; ↓): Because our image domains differ from ImageNet/COCO, we report a *proxy-FID* as a coarse indicator only. Specifically, we extract features using a *fixed* lightweight convnet (random initialization with a fixed seed; 128-d features) and compute the Fréchet distance between the resulting feature distributions of real and generated samples. We match the sample counts ($n_{\text{real}} = n_{\text{gen}} = 2048$ by default) and fix the metric sampling seed for reproducibility. **LPIPS (↓)**: VGG backbone via LPIPS (fallback: cosine similarity on VGG features); mean over 50 paired real–fake images. **SSIM (↑)**: mean over the same 50 pairs. Unless noted, we evaluate NEUROSQL in *sampled-z* mode ($z \sim \mathcal{N}(0, I)$); *paired-z* (reconstruction) results appear in the supplement.

F MINI-BATCH TRAINING NEUROSQL

While Algorithm 1 describes the exact full-batch procedure, NEUROSQL scales to large datasets ($n \gg 10^4$) via a stochastic approximation outlined in Algorithm 2. In this regime, we maintain a persistent "memory bank" of latent codes $\widehat{\boldsymbol{Z}}$. In each iteration, we sample a mini-batch of data $\boldsymbol{X}_{\mathcal{B}}$ and a corresponding random subset of the lattice $\boldsymbol{Q}_{\mathcal{K}}$. The assignment problem is solved locally on the $m \times m$ cost matrix. This reduces the computational complexity of the assignment step from $O(n^3)$ to $O((n/m) \cdot m^3) = O(n \cdot m^2)$ per epoch using the Hungarian method, or $O(n \cdot m^2)$ using the Greedy alternative. The momentum parameter ρ is critical here, acting as a temporal smoother that stabilizes the stochastic assignment trajectory towards the optimal transport map.

This computational benefit is reflected in Fig. 5, which reports mean epoch time as a function of sample size (N): NEUROSQL scales comparably to a VAE baseline and substantially more favorably than a GAN as (N) grows. Moreover, increasing the local assignment size (m) increases runtime in line with the $O(n \cdot m^2)$ per-epoch dependence predicted by the mini-batch scheme.

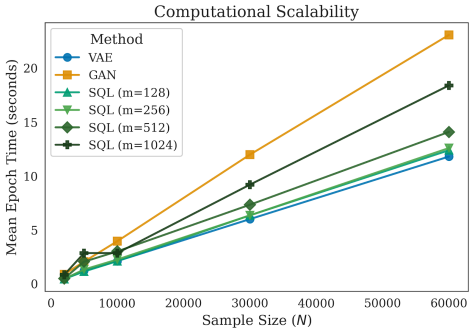


Figure 5: Computational scalability: mean epoch time versus sample size N for VAE, GAN, and NEUROSQL with different local assignment sizes m.

Algorithm 2 Quantile Assignment in Mini-batch Setting (Large Scale)

Input: data $\{\boldsymbol{X}_i\}_{i=1}^n$, prior P_Z, lattice $\boldsymbol{Q}^n$, batch size m, epochs E, momentum $\rho \in [0, 1)$, step size η
Initialize: global latent codes $\widehat{\boldsymbol{Z}} = \{\widehat{\boldsymbol{Z}}_1, \ldots, \widehat{\boldsymbol{Z}}_n\}$ by random assignment from $\boldsymbol{Q}^n$
for $e = 1, \ldots, E$ **do**
  Shuffle indices $\{1, \ldots, n\}$
  for each mini-batch $\mathcal{B} \subset \{1, \ldots, n\}$ with $|\mathcal{B}| = m$ **do**
    1) Generator update:
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \left[\frac{1}{m} \sum_{i \in \mathcal{B}} \ell\big(\boldsymbol{X}_i, \boldsymbol{G}_{\boldsymbol{\theta}}(\widehat{\boldsymbol{Z}}_i)\big) \right]$
    2) Stochastic quantile assignment:
    Sample lattice subset indices $\mathcal{K} \subset \{1, \ldots, n\}$ with $|\mathcal{K}| = m$
    Form batch cost matrix $\boldsymbol{C} \in \mathbb{R}^{m \times m}$ with entries
      $C_{j,k} \leftarrow \ell\big(\boldsymbol{X}_{\mathcal{B}[j]}, \boldsymbol{G}_{\boldsymbol{\theta}}(\boldsymbol{Q}_{\mathcal{K}[k]})\big)$, for $j, k \in \{1, \ldots, m\}$
    Solve batch assignment π_{batch} on $\boldsymbol{C}$ (Hungarian or greedy)
    3) Momentum update (sparse):
    for $j = 1, \ldots, m$ **do**
      $\widehat{\boldsymbol{Z}}_{\mathcal{B}[j]} \leftarrow \rho\, \boldsymbol{Q}_{\mathcal{K}[\pi_{\text{batch}}(j)]} + (1 - \rho)\, \widehat{\boldsymbol{Z}}_{\mathcal{B}[j]}$
    end for
  end for
end for
Output: generator $\boldsymbol{G}_{\boldsymbol{\theta}}$, aligned latents $\widehat{\boldsymbol{Z}}$

Algorithm 3 Greedy Assignment ($\mathcal{O}(n^2)$)

Input: cost matrix $\boldsymbol{C} \in \mathbb{R}^{n \times n}$
Initialize: unassigned columns $\mathcal{S} \leftarrow \emptyset$, permutation $\pi \in \{1, \ldots, n\}^n$
for $i = 1, \ldots, n$ **do**
  $c_{\min} \leftarrow +\infty$, $j^\star \leftarrow 0$
  for $j = 1, \ldots, n$ **do**
    if $j \notin \mathcal{S} \ \wedge \ C_{i,j} < c_{\min}$ **then**
      $c_{\min} \leftarrow C_{i,j}$
      $j^\star \leftarrow j$
    end if
  end for
  $\pi(i) \leftarrow j^\star$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{j^\star\}$
end for
Output: assignment π

G Greedy Assignment Algorithm

To address the scalability limitations of the Hungarian algorithm (which scales as $\mathcal{O}(n^3)$) for larger batch sizes or datasets, we implement a Greedy Assignment strategy. While the Hungarian algorithm guarantees the global minimum cost for the linear assignment problem, the Greedy approach provides an approximation that is computationally efficient ($\mathcal{O}(n^2)$) and sufficient for maintaining training stability in the paradigm we introduce with our model.

Algorithm Description. The greedy strategy iterates over each row of the cost matrix. For each row i, it selects the column j that minimizes the cost $C_{i,j}$, provided that column j has not already been assigned to a previous row. Once a column is selected, it is removed from the pool of available columns.

Computational Complexity. The outer loop runs exactly n times (once for each data sample). The inner loop scans n columns to find the minimum unassigned cost. Although the number of available columns decreases by 1 in each iteration, the upper bound of the search remains n. Consequently, the total complexity is proportional to $\sum_{i=1}^{n} n = n^2$, yielding a time complexity of $\mathcal{O}(n^2)$.

Table 3: Ablation Study: U-Net Latent Dimensions

Latent Dim	**Method**	**FID (proxy)** $\downarrow$	**LPIPS** $\downarrow$	**SSIM** $\uparrow$
8	Greedy	10.764394	0.376178	0.296806
	Hungarian	11.121147	0.378670	0.274640
16	Greedy	11.632382	0.398841	0.241317
	Hungarian	11.798410	0.390295	0.240504
24	Greedy	13.222514	0.384961	0.272349
	Hungarian	11.338643	0.379329	0.271508

With Table 3 we try to show the quantitative difference that occurs when the assignment algorithm changes.

H Quantitative evaluation of Runtime and Memory

Here, we would like to highlight and provide a quantitative evaluation of the memory and runtime of our proposed model. The following Table 4 is based on the metrics gathered from an MNIST run.

Table 4: Ablation: Assignment Methods and Resource Usage on MNIST (seed=11). Metrics averaged over epochs where applicable.

Latent Dim	Assign. Method	Method	Peak VRAM (MB)	RAM Used (MB)	Mean Epoch Time (s)	Mean Epoch GPU (MB)	Mean Epoch ΔRAM (MB)	FID (proxy)↓	LPIPS↓	SSIM↑
2	greedy	NeuroSQL	81.5	2764	1.40	116	11.2	0.628	0.045	0.580
	hungarian	NeuroSQL	284.6	3078	1.37	293	0.39	0.576	0.041	0.616
		VAE	319.8	3091	1.33	320	0.10	1.259	0.050	0.249
		GAN	405.6	3115	1.56	406	0.17	2.060	0.064	0.223
3	greedy	NeuroSQL	284.8	3206	1.39	293	0.14	0.567	0.040	0.574
	hungarian	NeuroSQL	284.9	3298	1.38	293	0.00	0.556	0.037	0.607
		VAE	320.7	3284	1.34	321	0.10	1.544	0.053	0.185
		GAN	404.6	3293	1.56	404	0.01	5.443	0.327	0.070

I Models and training protocol

Common setup. Images are scaled to $[0, 1]$ (diffusion uses $[-1, 1]$ internally). We use AdamW, cosine annealing, gradient clipping, and early stopping on validation loss. Generator backbones are matched across methods for capacity parity.

NeuroSQL (ours). We construct a size-n deterministic latent lattice via scrambled Sobol points mapped coordinatewise through $F_{Z_\ell}^{-1}$ (Sobol→Gaussian). Every K epochs we solve an exact global assignment (Hungarian) between data and lattice codes, where $K \in \{2, 3, 5\}$ is selected as a hyperparameter. After each assignment, we apply latent momentum $\widehat{z}^{(t)} \leftarrow \rho\, z_{\pi^{(t)}(i)} + (1-\rho)\, \widehat{z}^{(t-1)}$ with $\rho = 0.7$. The decoder is trained by regression on assigned codes using $\ell = \frac{1}{2}(1 - \mathrm{SSIM}) + \frac{1}{2}\|\cdot\|_1$.

VAE. We reuse the same generator backbone as the decoder; the encoder is an MLP on flattened pixels (to keep capacity modest). Training uses SSIM+L1 reconstruction plus a β-scaled KL term with $\beta = 0.005$.

GAN. The generator backbone is identical to NeuroSQL's. The discriminator is a lightweight four-layer CNN. We use the non-saturating objective with BCE logits, sharing optimizer, scheduler, and gradient clipping with NeuroSQL.

Diffusion (DDPM). A compact U-Net (base width 32) is trained with a linear β schedule for $T = 1000$ steps; default sampling uses 100 steps to match compute. Inputs are normalized to $[-1, 1]$ following common practice.

Reproducibility knobs. We fix random seeds, match the number of training epochs and batch sizes across methods, and report all per-method hyperparameters (including learning rates, $K \in \{2, 3, 5\}$, and augmentations).

J Preprocessing and splits

For each dataset, we standardize a lightweight pipeline to minimize confounds while allowing small variations across runs.

- **Resize/crop.** `MNIST`: native 28×28. `CelebA`: center-crop then resize, typically to 32×32 (primary), with occasional 64–128 experiments. `OASIS`: center-crop then resize, primarily 128×128 with 64×64 ablations.
- **Scaling.** Inputs mapped to $[0, 1]$ (no per-image standardization during training).
- **Channel handling.** `MNIST`/`OASIS` trained as single-channel; for backends expecting 3 channels (e.g., LPIPS/VGG), we replicate channels *at metric time only*.
- **Splits.** `MNIST`: standard train/test. `CelebA`: official train/val/test. `OASIS`: subject-wise 80/10/10 to prevent slice leakage. Seeds and split indices are provided in the supplementary.

Optimization and budgets (typicals/ranges). AdamW (or Adam), cosine warm restarts; weight decay $\sim 10^{-4}$; gradient clip ≈ 1.0; early stopping on validation loss with patience $\sim$ 25–30 epochs. Learning rates are usually in $[1\times10^{-4},\ 3\times10^{-4}]$ for convolutional decoders; diffusion runs use comparable schedules at matched compute. Batch sizes depend on resolution: `MNIST` 32–64, `CelebA` 64–128, `OASIS` 32–64. Epoch caps are typically 120–250 across datasets, with early stopping often terminating earlier. We sweep latent dimension d over $\{2, 4, 8, 16, 32, 64, 128\}$ and report results (Sec. M).

K Proof of Proposition 1

Proof. Following Hallin et al. (2021), we define the center-outward empirical distribution function as the solution of the optimal transportation problem:

$$F^n_{\pm} = \underset{T\in\mathcal{T}}{\arg\min} \sum_{i=1}^{n} \|\boldsymbol{Z}_i - T(\boldsymbol{Z}_i)\|^2,$$

where the minimum is taken over $\mathcal{T}$, the set of all bijective mappings between $\boldsymbol{Z}_1, ..., \boldsymbol{Z}_n$ and the grid $\boldsymbol{U}_1, \ldots, \boldsymbol{U}_n$. In fact, this is equivalent to solve a linear assignment problem:

$$\pi^* = \underset{\pi\in S_n}{\arg\min} \|\boldsymbol{Z} - \boldsymbol{U}^n_\pi\|^2,$$

and set $F^n_{\pm}(\boldsymbol{Z}^n) := \boldsymbol{U}^n_{\pi^*}$. Then we apply Theorem 2.4 in Hallin et al. (2021) to obtain:

$$\max_{1\le i\le n} \|F^n_{\pm}(\boldsymbol{Z}_i) - F_{\pm}(\boldsymbol{Z}_i)\|^2 \to 0 \text{ a.s. as } n \to \infty.$$

In our case, as $F_{\pm}$ is the uniform distribution over $\mathcal{U}_d$, it is relatively straightforward to see that:

$$\min_{\pi\in S_n} \|\boldsymbol{Q}^n_\pi - \boldsymbol{Z}\|^2 \to 0, \text{ a.s. as } n \to \infty.$$

□

L Practical training details

Loss choices. For images, we use a perceptual, scale-stable loss:

$$\ell(\hat{x}, x) = \frac{1}{2}\big(1 - \mathrm{SSIM}(\hat{x}, x)\big) + \frac{1}{2}\|\hat{x} - x\|_1,$$

which is the exact loss used in our codebase.

We instantiate gen with lightweight decoders so that comparisons against VAEs/GANs/Diffusion control for capacity and compute:

- **ConvNet.** Transposed-convolution stack mapping $z \in \mathbb{R}^q$ to $X \in \mathbb{R}^{3\times H\times H}$ (stride-2 upsampling).
- **ResNet** Four residual upsampling blocks (512→256→128→64), followed by a 3×3 head with sigmoid output in $[0, 1]$. We optionally initialize residual weights from ResNet-18 where shapes match.

- **U-Net decoder.** A small transformer decoder on patchified embeddings of z followed by an MLP head back to pixels.

Our experiments keep these decoders small and matched across methods to stress that gains come from the *quantile–assignment loop*, not decoder sophistication.

- **Loss and normalization.** Images are scaled to $[0, 1]$. We use $\ell = \frac{1}{2}(1 - \mathrm{SSIM}) + \frac{1}{2}\ell_1$ in both decoder and cost matrix.
- **Optimization.** AdamW with cosine annealing and gradient clipping; early stopping on validation ℓ.
- **Latent momentum.** After each assignment, a momentum update $\widehat{z}^{(t)} \leftarrow \rho\, z_{\pi^{(t)}(i)} + (1 - \rho)\, \widehat{z}^{(t-1)}$ stabilizes training (we use $\rho = 0.7$).
- **Resource parity.** For fair comparisons to VAEs, GANs, and Diffusion, we fix the *same* generator backbone and training budget; only the learning paradigm changes.

M Additional Results on OASIS, CelebA, AFHQ and MNIST

In this section, we present a comprehensive evaluation of NeuroSQL on the OASIS brain imaging dataset, the CelebA dataset, and the AFHQ dataset, using the different distinct generator backbones: ConvNet, ResNet, and U-Net. We benchmark performance against VAE and GAN baselines using the same generators across a wide range of latent dimensions to assess model stability. In tables reporting average results across all experiments, the best achieved metrics are highlighted in bold.

Table 5: Results on OASIS, a brain imaging dataset, using ConvNet across latent dimensions. Lower is better for FID (proxy) and LPIPS; higher is better for SSIM.

Latent dimension	Method	FID (Proxy) ↓	LPIPS ↓	SSIM ↑
2	NeuroSQL	7.914	0.390	0.259
	VAE	8.198	0.410	0.246
	GAN	9.936	0.450	0.218
4	NeuroSQL	9.241	0.403	0.257
	VAE	8.720	0.463	0.212
	GAN	19.420	0.509	0.206
8	NeuroSQL	8.286	0.411	0.256
	VAE	8.029	0.485	0.215
	GAN	12.766	0.558	0.193
16	NeuroSQL	8.602	0.455	0.267
	VAE	12.768	0.539	0.171
	GAN	12.485	0.584	0.200
32	NeuroSQL	8.856	0.401	0.257
	VAE	12.490	0.525	0.178
	GAN	14.852	0.593	0.174
64	NeuroSQL	7.385	0.388	0.265
	VAE	16.003	0.567	0.151
	GAN	14.453	0.602	0.125
128	NeuroSQL	18.743	0.453	0.248
	VAE	16.329	0.571	0.158
	GAN	29.792	0.620	0.084

Table 6: Results on OASIS, a brain imaging dataset, using ResNet across latent dimensions. Lower is better for FID (proxy) and LPIPS; higher is better for SSIM.

Latent dimension	Method	FID (proxy) ↓	LPIPS ↓	SSIM
16	NeuroSQL	25.410	0.249	0.301
	VAE	51.139	0.309	0.171
	GAN	168.993	0.727	0.008
32	NeuroSQL	31.957	0.257	0.242
	VAE	66.045	0.362	0.128
	GAN	158.103	0.687	0.008
64	NeuroSQL	30.892	0.220	0.221
	VAE	47.146	0.358	0.135
	GAN	156.088	0.741	0.135
128	NeuroSQL	34.346	0.224	0.196
	VAE	52.317	0.397	0.135
	GAN	156.288	0.723	0.124

Table 7: Results on OASIS, a brain imaging dataset, using a U-Net across latent dimensions. Lower is better for FID (proxy) and LPIPS; higher is better for SSIM.

Latent dimension	Method	FID (proxy) ↓	LPIPS ↓	SSIM ↑
4	NeuroSQL	8.519902	0.386861	0.254752
	VAE	7.858056	0.377805	0.265082
	GAN	26.234322	0.597633	0.206676
8	NeuroSQL	8.579974	0.401890	0.238080
	VAE	5.976874	0.384821	0.264277
	GAN	35.444614	0.609803	0.172225
16	NeuroSQL	10.037155	0.391049	0.260999
	VAE	7.178138	0.402099	0.231028
	GAN	20.907921	0.616312	0.225444
32	NeuroSQL	8.405630	0.388190	0.253740
	VAE	4.424680	0.402910	0.238710
	GAN	28.069950	0.613550	0.200620
64	NeuroSQL	8.259449	0.385646	0.262159
	VAE	6.098939	0.395427	0.272117
	GAN	30.385052	0.644507	0.153054
128	NeuroSQL	7.553965	0.371082	0.282864
	VAE	9.101107	0.415775	0.257312
	GAN	30.505713	0.553681	0.202557

Table 8: Results on OASIS, a brain imaging dataset, using a U-Net — with results averaged across latent dimensions $\{4, 8, 16, 32, 64, 128\}$.

Method	FID (proxy) ↓	LPIPS ↓	SSIM ↑
NeuroSQL	8.559346	**0.387453**	**0.258766**
VAE	**6.772966**	0.396473	0.254754
GAN	28.591262	0.605914	0.193429

On average, across latent dimensions, NeuroSQL attains the best LPIPS and SSIM, while VAE has the lowest FID (proxy).

Table 9: Results on CelebA, a face attributes dataset, using ConvNet across latent dimensions. Lower is better for FID (proxy) and LPIPS; higher is better for SSIM.

Latent dimension	Method	FID (proxy) ↓	LPIPS ↓	SSIM ↑
2	NeuroSQL	9.64453	0.22094	0.31376
	VAE	11.49087	0.24068	0.27396
	GAN	26.56548	0.47277	0.10216
4	NeuroSQL	6.99789	0.22845	0.31645
	VAE	8.57467	0.23091	0.26471
	GAN	29.68144	0.49874	0.06955
8	NeuroSQL	6.686607	0.244427	0.297947
	VAE	32.183292	0.281024	0.226046
	GAN	28.772638	0.461147	0.058802
16	NeuroSQL	17.252127	0.304758	0.296877
	VAE	11.955338	0.282781	0.160219
	GAN	20.922581	0.419844	0.117695
32	NeuroSQL	4.04047	0.19269	0.25707
	VAE	6.57785	0.21120	0.13975
	GAN	13.92531	0.31649	0.12201
64	NeuroSQL	3.94787	0.20649	0.25782
	VAE	3.93301	0.21478	0.15485
	GAN	16.27481	0.35050	0.10456
128	NeuroSQL	4.91280	0.20557	0.26119
	VAE	5.57425	0.20361	0.16837
	GAN	17.16048	0.34640	0.15798

Table 10: Results on CelebA, a face attributes dataset, using ResNet across latent dimensions. Lower is better for FID (proxy) and LPIPS; higher is better for SSIM.

Latent dimension	Method	FID (proxy) ↓	LPIPS ↓	SSIM ↑
2	NeuroSQL	4.40599	0.18367	0.27506
	VAE	8.01552	0.19720	0.24972
	GAN	19.01699	0.20240	0.13535
4	NeuroSQL	4.47511	0.16968	0.26754
	VAE	6.03891	0.20652	0.20760
	GAN	14.21587	0.26898	0.14515
8	NeuroSQL	4.60623	0.18201	0.25536
	VAE	4.86620	0.25218	0.17823
	GAN	13.94172	0.18593	0.17628
32	NeuroSQL	3.99240	0.18115	0.21823
	VAE	6.00124	0.25017	0.12915
	GAN	11.08812	0.23644	0.16986

Table 11: Results on CelebA, a face attributes dataset, using U-Net (unconditional) across latent dimensions. Lower is better for FID (proxy) and LPIPS; higher is better for SSIM.

Latent dimension	Method	FID (proxy) ↓	LPIPS ↓	SSIM ↑
2	NeuroSQL	4.96169	0.18253	0.27537
	VAE	7.33227	0.19269	0.25696
	GAN	14.09393	0.22041	0.12822
4	NeuroSQL	3.54758	0.18932	0.27993
	VAE	4.99991	0.19061	0.22841
	GAN	7.66398	0.19251	0.14899
8	NeuroSQL	3.96161	0.19244	0.24416
	VAE	5.48428	0.20413	0.20573
	GAN	20.46054	0.16577	0.18463
16	NeuroSQL	2.72089	0.17970	0.24505
	VAE	10.00917	0.19001	0.18914
	GAN	14.48838	0.18500	0.15935
32	NeuroSQL	14.65015	0.21850	0.27135
	VAE	31.10957	0.24081	0.21405
	GAN	30.37928	0.31335	0.13607

Table 12: Results on AFHQ, an animal faces dataset, using ConvNet across latent dimensions. Lower is better for FID (proxy) and LPIPS; higher is better for SSIM.

Latent dimension	Method	FID (proxy) ↓	LPIPS ↓	SSIM ↑
2	NeuroSQL	22.023664	0.563209	0.339032
	VAE	44.276279	0.531442	0.284219
	GAN	74.178604	0.693773	0.080698
4	NeuroSQL	35.913769	0.538813	0.357857
	VAE	35.430927	0.527985	0.285893
	GAN	48.706165	0.669665	0.129869
8	NeuroSQL	37.569298	0.505324	0.353983
	VAE	83.373650	0.516064	0.211640
	GAN	30.621849	0.797056	0.191322
16	NeuroSQL	52.609642	0.564259	0.370944
	VAE	88.436768	0.527512	0.161437
	GAN	98.826027	0.620977	0.066925
32	NeuroSQL	31.695450	0.512806	0.339188
	VAE	95.740288	0.499635	0.172815
	GAN	46.238804	0.751076	0.072116
64	NeuroSQL	28.105133	0.532103	0.347139
	VAE	127.357986	0.525766	0.140224
	GAN	50.671597	0.753972	0.067132
128	NeuroSQL	17.399908	0.591030	0.368674
	VAE	23.283272	0.789889	0.374535
	GAN	59.287437	0.754374	0.042504

Table 13: Results on AFHQ, an animal faces dataset, using ConvNet — with results averaged across latent dimensions.

Method	FID (proxy) ↓	LPIPS ↓	SSIM ↑
NeuroSQL	**32.19**	**0.544**	**0.354**
VAE	71.13	0.560	0.233
GAN	58.36	0.720	0.093

Table 14: Results on AFHQ, an animal faces dataset, using ResNet across latent dimensions. Lower is better for FID (proxy) and LPIPS; higher is better for SSIM.

Latent dimension	Method	FID (proxy) ↓	LPIPS ↓	SSIM ↑
2	NeuroSQL	7.267559	0.459967	0.267403
	VAE	15.259568	0.448991	0.235907
	GAN	39.100639	0.550003	0.091928
4	NeuroSQL	11.560558	0.460007	0.246023
	VAE	13.387914	0.432443	0.192707
	GAN	37.873501	0.356168	0.120893
8	NeuroSQL	13.611592	0.471643	0.241053
	VAE	11.736956	0.431859	0.204782
	GAN	29.584837	0.486513	0.081718
16	NeuroSQL	17.987249	0.341935	0.208179
	VAE	9.721145	0.347468	0.178343
	GAN	17.546219	0.483133	0.107947
32	NeuroSQL	14.799847	0.438891	0.255200
	VAE	8.249439	0.341844	0.188594
	GAN	22.425539	0.363643	0.113281
64	NeuroSQL	10.561233	0.558646	0.252005
	VAE	14.458963	0.404593	0.202796
	GAN	40.478588	0.423956	0.107915
128	NeuroSQL	11.019302	0.489376	0.261104
	VAE	13.060718	0.353215	0.208177
	GAN	18.462259	0.381437	0.104066

Table 15: Results on AFHQ, an animal faces dataset, using ResNet — with results averaged across latent dimensions.

Method	FID (proxy) ↓	LPIPS ↓	SSIM ↑
NeuroSQL	12.40	0.460	**0.247**
VAE	**12.27**	**0.394**	0.202
GAN	29.35	0.435	0.104

Table 16: Results on AFHQ, an animal faces dataset, using U-Net across latent dimensions. Lower is better for FID (proxy) and LPIPS; higher is better for SSIM.

Latent dimension	Method	FID (proxy) ↓	LPIPS ↓	SSIM ↑
2	NeuroSQL	17.626825	0.595854	0.272737
	VAE	10.595321	0.620105	0.279837
	GAN	23.709433	0.405830	0.116950
3	NeuroSQL	10.193194	0.617973	0.278050
	VAE	29.156761	0.578171	0.173115
	GAN	17.098032	0.607903	0.074634
4	NeuroSQL	10.773172	0.561935	0.258762
	VAE	14.463408	0.576194	0.117610
	GAN	13.078835	0.626831	0.057038
8	NeuroSQL	11.994763	0.638746	0.275091
	VAE	42.187984	0.571152	0.062857
	GAN	75.077110	0.618699	0.039482
16	NeuroSQL	10.660514	0.659894	0.266318
	VAE	51.288338	0.550339	0.135791
	GAN	97.593185	0.703616	0.010010
32	NeuroSQL	16.301731	0.595429	0.252468
	VAE	28.342377	0.566271	0.076232
	GAN	25.781809	0.604119	0.025050
64	NeuroSQL	9.855629	0.660927	0.276445
	VAE	62.289070	0.585000	0.093420
	GAN	99.730621	0.702493	0.010348

Table 17: Results on AFHQ, an animal faces dataset, using U-Net — with results averaged across latent dimensions $\{2, 3, 4, 8, 16, 32, 64\}$.

Method	FID (proxy) ↓	LPIPS ↓	SSIM ↑
NeuroSQL	**12.486547**	0.618680	**0.268553**
VAE	34.046180	**0.578176**	0.134123
GAN	50.295575	0.609927	0.047645

On average, NeuroSQL achieves the best FID (proxy) and SSIM; VAE attains the lowest LPIPS.

Table 18: Ablation results on MNIST, a database of handwritten digits (all runs). Lower is better for FID (proxy) and LPIPS; higher is better for SSIM.

Latent dimension	Seed	Method	FID (proxy) ↓	LPIPS ↓	SSIM ↑
2	11	NeuroSQL	0.696835	0.039503	0.564344
		VAE	1.070473	0.058065	0.200167
		GAN	2.157639	0.054278	0.245780
3	11	NeuroSQL	0.527451	0.030257	0.668574
		VAE	1.443453	0.060009	0.157831
		GAN	1.849820	0.057282	0.220785

Table 19: Results on MNIST, a database of handwritten digits — averaged over latent dimensions $\{2, 3\}$.

Method	**FID (proxy)** $\downarrow$	**LPIPS** $\downarrow$	**SSIM** $\uparrow$
NeuroSQL	**0.612143**	**0.034880**	**0.616459**
VAE	1.256963	0.059037	0.178999
GAN	2.003730	0.055780	0.233283

NeuroSQL is best on all three metrics (FID (proxy), LPIPS, and SSIM).