VQ-Flows: Vector Quantized Local Normalizing Flows Supplementary Material

Sahil Sidheekh[*1] **Chris B. Dock**[*2] **Tushar Jain**[1] **Radu Balan**[2] **Maneesh K. Singh**[†3]

[1]Verisk Analytics
[2]University of Maryland, College Park
[3]Motive Technologies, Inc.

This document presents the discussions and results left out in the main paper due to space constraints. We begin with the details regarding the considered data distributions. We then present quantitative and qualitative results left out from the main paper. We finally conclude with further information regarding the experiments and implementation.

1 DATA GENERATION

We generated and experimented with ten 3-dimensional data distributions over manifolds of varying complexity. Figure 1 provides the visualizations for each of the considered datasets. We elaborate more on the equations used to generate data from each of these distribution below.

1.1 SPHERICAL

We considered three mixture of Gaussians in the 3-dimensional space with parameters (μ_1, σ),(μ_2, σ),(μ_3, σ) respectively. Samples (x) drawn uniformly from each of the three Gaussians were then projected on to the unit sphere in 3D as $x/||x||$. The means (μ_i) were sampled from a standard normal distribution and the standard deviation was set to 0.2. The exact parameter values used are: $\mu_1 = (-0.15, -0.77, 0.94)$, $\mu_2 = (0.79, -0.75, -0.02)$, $\mu_3 = (0.04, 0.40, 1.31)$ and $\sigma = 0.2$.

1.2 HELIX

To generate the Helix data distribution, we first sample $\theta \in \mathbb{R}$ uniformly from $[0, 8\pi]$. For each θ, we then generate the datapoint $\mathbf{x} = (x, y, z) + \epsilon$, where $\epsilon \sim N(0, \sigma = 0.01)$ and $- \bullet x = \theta \bullet y = \cos\theta \bullet z = \sin\theta$

1.3 LISSAJOUS

To generate the Lissajous data distribution, we first sample $\theta \in \mathbb{R}$ uniformly from $[-\pi, \pi]$. For each θ, we then generate the datapoint $\mathbf{x} = (x, y, z) + \epsilon$, where $\epsilon \sim N(0, \sigma = 0.01)$ and $- \bullet x = \cos\theta \bullet y = 0 \bullet z = \sin(2\theta)$

1.4 TWISTED-EIGHT

To generate the Twisted-Eight data distribution, we sample $\theta \in \mathbb{R}$ uniformly from $[-\pi, \pi]$. For each θ, we then generate two datapoints $\mathbf{x_1} = (x_1, y_1, z_1) + \epsilon$ and $\mathbf{x_2} = (x_2, y_2, z_2) + \epsilon$, where $\epsilon \sim N(0, \sigma = 0.01)$ and $- \bullet x_1 = \sin\theta \bullet y_1 = \cos\theta \bullet z_1 = 0$.
$\bullet\ x_2 = 2 + \sin\theta \bullet y_2 = 0 \bullet z_2 = \cos\theta$
The final distribution is the union of the distributions over $\mathbf{x_1}$ and $\mathbf{x_2}$.

1.5 KNOTTED

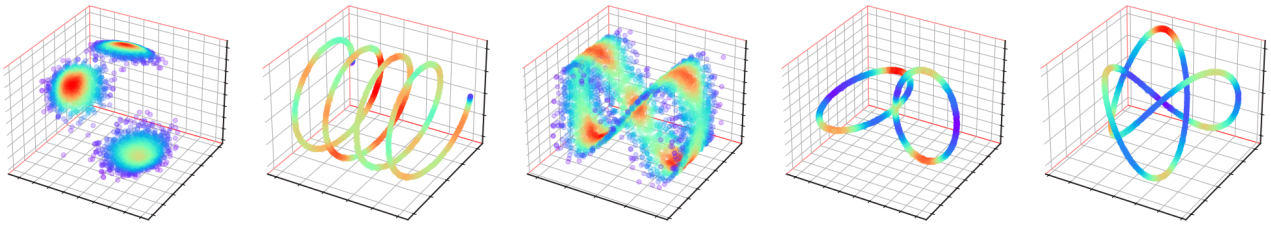
To generate the Knotted data distribution, we first sample $\theta \in \mathbb{R}$ uniformly from $[-\pi, \pi]$. For each θ, we then generate the datapoint $\mathbf{x} = (x, y, z) + \epsilon$, where $\epsilon \sim N(0, \sigma = 0.01)$ and $- \bullet x = \sin\theta + 2\sin 2\theta \bullet y = \cos\theta - 2\cos 2\theta \bullet z = \sin 3\theta$

1.6 INTERLOCKED-CIRCLES

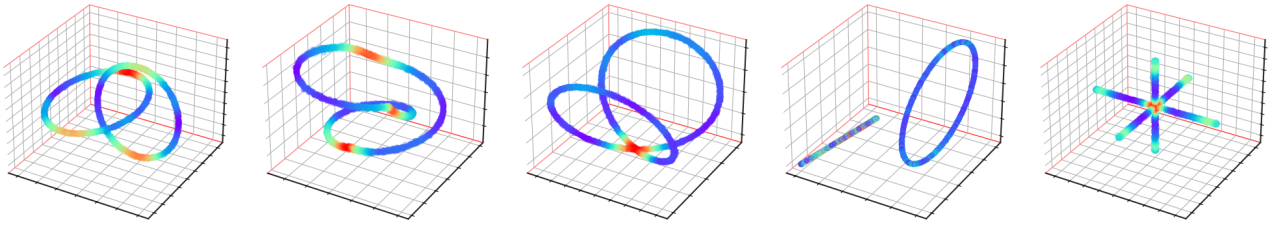
To generate the Interlocked-Circles data distribution, we sample $\theta \in \mathbb{R}$ uniformly from $[-\pi, \pi]$. For each θ, we then generate two datapoints $\mathbf{x_1} = (x_1, y_1, z_1) + \epsilon$ and $\mathbf{x_2} = (x_2, y_2, z_2) + \epsilon$, where $\epsilon \sim N(0, \sigma = 0.01)$ and $- \bullet x_1 = \sin\theta \bullet y_1 = \cos\theta \bullet z_1 = 0$.
$\bullet\ x_2 = 1 + \sin\theta \bullet y_2 = 0 \bullet z_2 = \cos\theta$
The final distribution is the union of the distributions over $\mathbf{x_1}$ and $\mathbf{x_2}$.

[*]Equal contribution
[†]Work was performed while at Verisk Analytics.

Accepted for the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022).



(a) Spherical (b) Helix (c) Lissajous (d) Twisted-Eight (e) Knotted



(f) InterlockedCircles (g) Non-Knotted (h) Bent-Lissajous (i) Disjoint-Circles (j) Star

Figure 1: Visualizations of the considered 3-dimensional data distributions

1.7 NON-KNOTTED

To generate the Non-Knotted data distribution, we first sample $\theta \in \mathbb{R}$ uniformly from $[-\pi, \pi]$. For each θ, we then generate the datapoint $\mathbf{x} = (x, y, z) + \epsilon$, where $\epsilon \sim N(0, \sigma = 0.01)$ and $-$

- $x = (1 + 0.5\cos 3\theta)\cos 2\theta$
- $y = (1 + 0.5\cos 3\theta)\sin 2\theta$
- $z = 0.5\sin\theta$

1.8 BENT-LISSAJOUS

To generate the Bent-Lissajous data distribution, we first sample $\theta \in \mathbb{R}$ uniformly from $[-\pi, \pi]$. For each θ, we then generate the datapoint $\mathbf{x} = (x, y, z) + \epsilon$, where $\epsilon \sim N(0, \sigma = 0.01)$ and $-$ • $x = \sin 2\theta$ • $y = \cos\theta$ • $z = \cos 2\theta$

1.9 DISJOINT-CIRCLES

To generate the Disjoint-Circles data distribution, we sample $\theta \in \mathbb{R}$ uniformly from $[-\pi, \pi]$. For each θ, we then generate two datapoints $\mathbf{x_1} = (x_1, y_1, z_1) + \epsilon$ and $\mathbf{x_2} = (x_2, y_2, z_2) + \epsilon$, where $\epsilon \sim N(0, \sigma = 0.01)$ and $-$ • $x_1 = -1 + \sin\theta$ • $y_1 = -1 + \sin\theta$ • $z_1 = -1 + \sin\theta$.
• $x_2 = 2 + \sin\theta$ • $y_2 = 1 + 2\cos\theta$ • $z_2 = 1 + 2\cos\theta$
The final distribution is the union of the distributions over $\mathbf{x_1}$ and $\mathbf{x_2}$.

1.10 STAR

To generate the Star data distribution, we sample $\theta \in \mathbb{R}$ uniformly from $[-\pi, \pi]$. For each θ, we then generate three datapoints $\mathbf{x_1} = (x_1, y_1, z_1) + \epsilon$, $\mathbf{x_2} = (x_2, y_2, z_2) + \epsilon$, and $\mathbf{x_3} = (x_3, y_3, z_3) + \epsilon$ where $\epsilon \sim N(0, \sigma = 0.01)$ and $-$
• $x_1 = \sin\theta$ • $y_1 = 0$ • $z_1 = 0$.
• $x_2 = 0$ • $y_2 = \sin\theta$ • $z_2 = 0$
• $x_3 = 0$ • $y_3 = 0$ • $z_3 = \sin\theta$
The final distribution is the union of the distributions over $\mathbf{x_1}$, $\mathbf{x_2}$ and $\mathbf{x_3}$.

2 ADDITIONAL RESULTS

2.1 DENSITY ESTIMATION AND SAMPLE GENERATION

We provide quantitative evaluations for density estimation and sample generation over four additional 3-dimensional data distribution discussed Section 1 in Table 1 and Table 2 respectively. We can observe that the models trained with the augmentation of our framework achieves better performance for both density estimation and sample generation than their corresponding baselines. We also validate and compare the goodness of the generated samples through qualitative visualizations in Figures 7 - 11. Note that CEFs perform poorer than the other baselines because they consist of a 2-dimensional base flow which is then raised to the 3-dimensional space using a conformal embedding. The other flows (RealNVP and MAF) are, on the other hand, trained in the 3-dimensional space. A particularly interesting observa-

tion here is that the data distributions learned by CEF without VQ-augmentation tend to be planar in the 3-dimensional space. This demonstrates the limited expressivity of global conformal dimension raising transformations. The local conformal transformations obtained with the augmentation of our framework are, on the other hand, able to better capture the global structure of the data distribution and generate better samples.

2.2 GAUSSIANIZATION

The ability of a normalizing flow to generate high fidelity samples from given data distribution is also governed by whether the latent space learned through the flow transformation matches the assumed prior. For a flow with a Normal distribution assumed in the latent space, this means that the forward flow transformations should effectively Gaussianize the given data distribution. In Figures 3 to 6 we thus visualize and compare how different data distributions are transformed gradually by each layer of a RealNVP flow trained with and without the augmentation of our proposed framework. We can observe that the models trained with VQ-augmentation learn to better transform the input space to match the assumed prior. As a result, they are also able to generate better samples.

3 IMPLEMENTATION DETAILS

To experimentally validate the efficacy of the proposed framework, we consider the ten datasets presented in Section 1. Each dataset consists of $10,000$ datapoints, $5,000$ of which we use for training and $2,500$ each for validation and testing. We train three different normalizing flows - RealNVP, Masked Autoregressive Flows (MAF) and Conformal Embedding Flows (CEF). We define each model using 5 flow transformations and train them for 100 epochs using an Adam optimizer with a learning rate of $1e-4$ and a batch size of 128. We follow the same hyperparameters for a base flow and its VQ-counterpart without any tuning and report the performance averaged over 5 independent trials. We early stop if the validation performance does not improve over 10 epochs. The architectural details pertaining to each of the models are given below:

RealNVP- We compose the RealNVP flow using 5 coupling layer transformations, each followed by a batch-normalization. We use feedforward networks with 2 hidden layers, each consisting of 128 hidden nodes as the non-linear transformation to obtain the scaling and translation parameters. We use $tanh$ as the activation function for the scale network and $relu$ as the activation function for the translation network.

MAF- We compose the MAF flow using 5 masked auto-regressive layer transformations, each followed by a batch-normalization. In each layer, we use a masked feedforward network with 1 hidden layer, consisting of 128 hidden nodes. We use $relu$ as the activation function for the feedforward network.

CEF- We compose the CEF flow using 5 coupling layer transformations in 2-dimensional space, followed by the conformal transformation that raises the dimension to 3. We use the same architecture reported above for RealNVP in the coupling transforms. The conformal embedding is parameterized as given in [1], using a composition of Scaling, Shifting, Orthogonal, Special Conformal and Padding transformations. As CEF is an injective flow, we follow [1] and train it to minimize the reconstruction loss for 20 epochs prior to employing the maximum likelihood training.

VQ-*flow*- We parameterize the encoder and decoder of the VQ-AE using feedforward neural networks. In each network, we use 4 hidden layers each consisting of 128 hidden nodes followed by batch-normalization and a leaky$-relu$ activation with negative slope of 0.2. To learn the partitioning of the data manifold, we use a latent dimension of 2 with $k = 32$ learnable quantized centers. We train the VQ-AE for 50 epochs to minimize the reconstruction loss using an Adam optimizer with a learning rate of $1e-4$ and batch size of 128.

To define the conditional normalizing flow, we use the parameterization given in [2, 3]. The key idea is to incorporate the quantized center as additional conditioning information to the unrestricted (non-invertible) neural network used in the coupling and auto-regressive transformations. Figure 2 demonstrates the construction of such a conditional coupling layer transform. To define conditional conformal transformations, we use k conformal embeddings and index into it using the quantized center. We believe that we can extend our framework to other arbitrary flows by adapting the conditional flow transformations defined in [4], which we leave to future work.

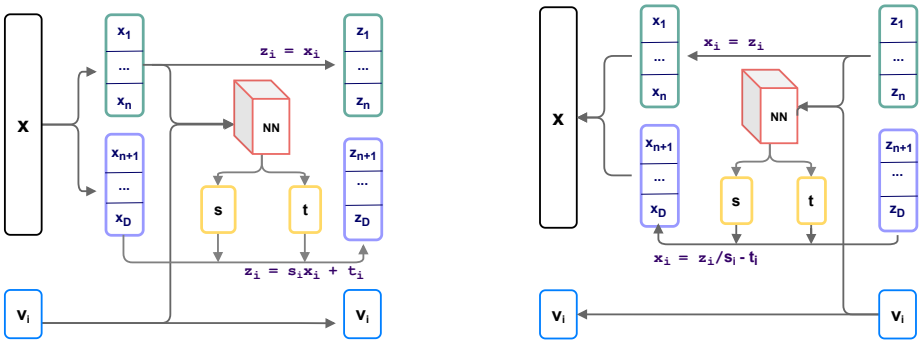
References

[1] Brendan Leigh Ross and Jesse C Cresswell. Tractable density estimation on learned manifolds with conformal embedding flows. In *Advances in Neural Information Processing Systems*, 2021.

[2] You Lu and Bert Huang. Structured output learning with conditional generative flows. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI*, pages 5005–5012, 2020.

[3] George Papamakarios, Iain Murray, and Theo Pavlakou. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, pages 2338–2347, 2017.

[4] Robert Cornish, Anthony L. Caterini, George Deligian-



(a) Forward Transformation (b) Inverse Transformation

Figure 2: Parametrizing the conditional coupling layer transformation.

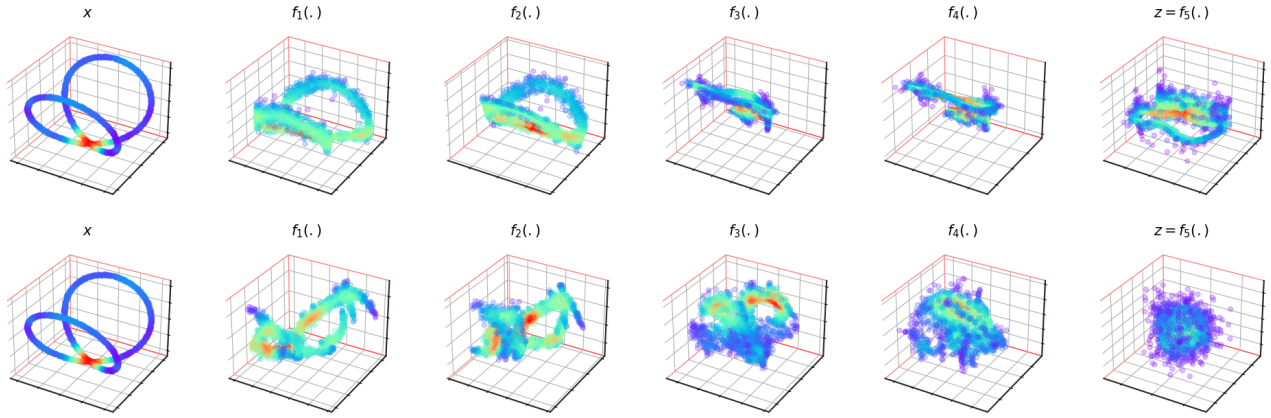


Figure 3: Visualization of the latent transformation achieved using RealNVP (Top Row) and VQ-RealNVP (Bottom Row) on the Bent-Lissajous data distribution.

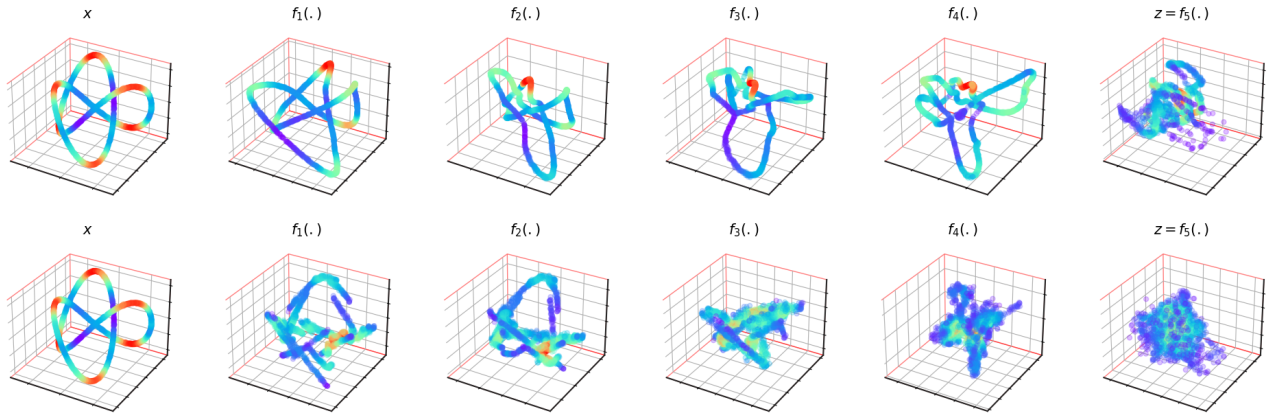


Figure 4: Visualization of the latent transformation achieved using RealNVP (Top Row) and VQ-RealNVP (Bottom Row) on the Knotted data distribution.

nidis, and Arnaud Doucet. Relaxing bijectivity constraints with continuously indexed normalising flows. In *Proceedings of the 37th International Conference on Machine Learning, ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 2133–2143, 2020.

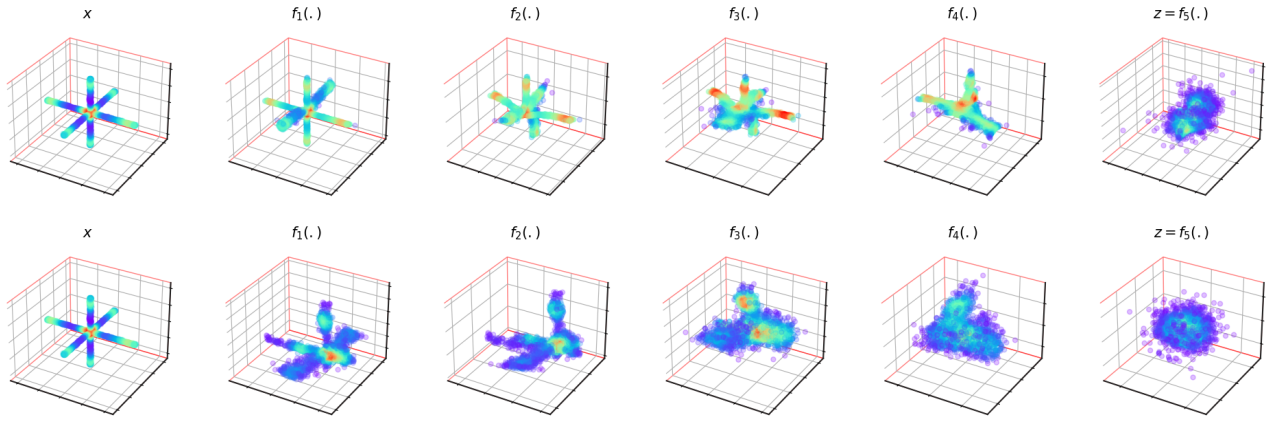


Figure 5: Visualization of the latent transformation achieved using RealNVP (Top Row) and VQ-RealNVP (Bottom Row) on the Star data distribution.

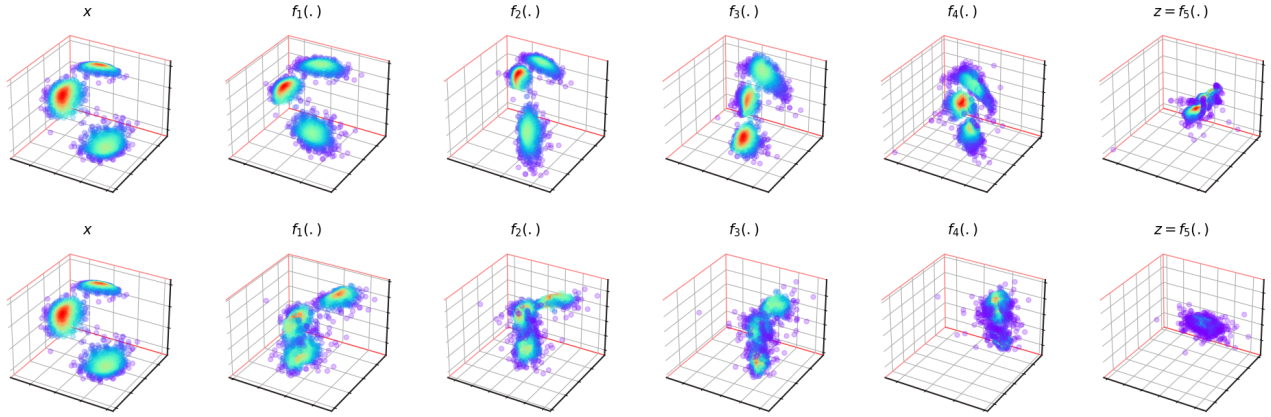


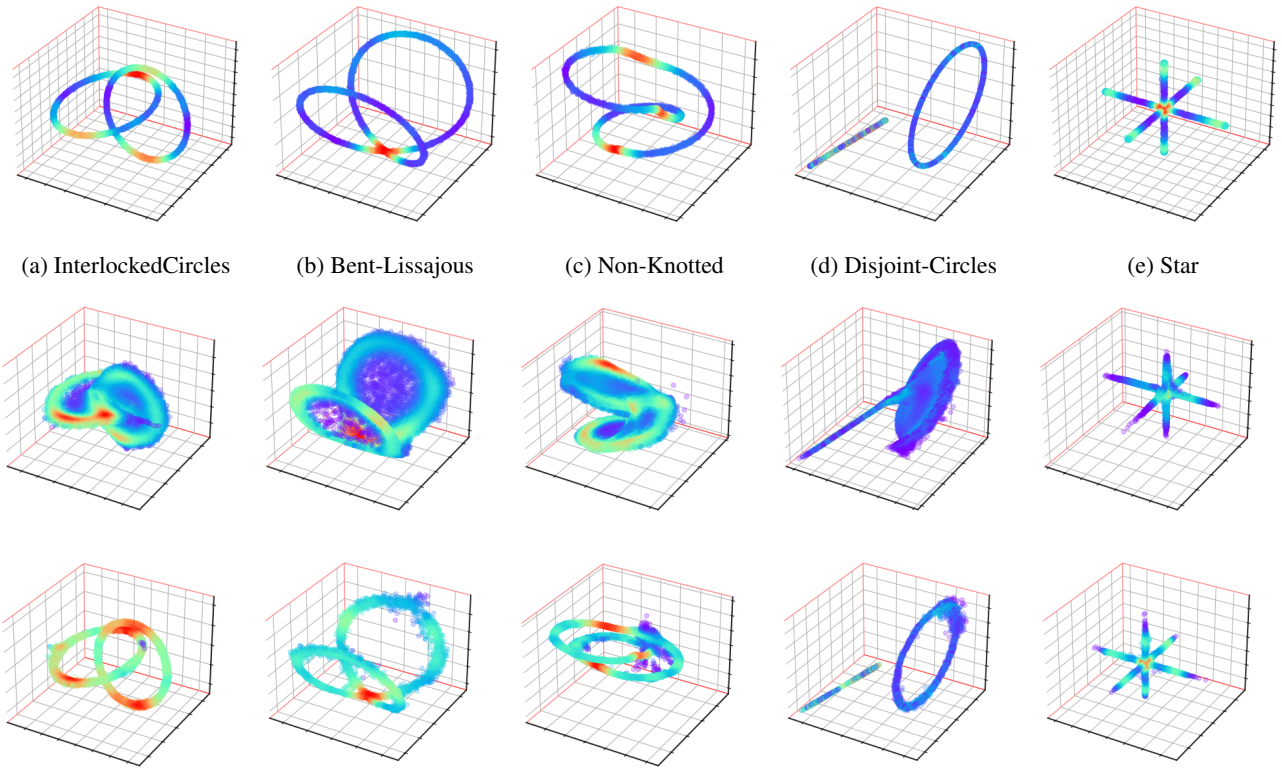
Figure 6: Visualization of the latent transformation achieved using RealNVP (Top Row) and VQ-RealNVP (Bottom Row) on the Spherical data distribution.

Model	Non-Knotted	Bent-Lissajous	Disjoint-Circles	Star
Real NVP	0.53 ± 0.18	1.04 ± 0.22	1.71 ± 0.12	3.33 ± 0.18
VQ-RealNVP	2.39 ± 0.24	2.62 ± 0.13	2.71 ± 0.19	4.23 ± 0.06
MAF	0.73 ± 0.18	1.48 ± 0.11	1.95 ± 0.12	3.53 ± 0.03
VQ-MAF	2.41 ± 0.19	2.06 ± 0.12	2.87 ± 0.07	3.59 ± 0.12
CEF	-0.46 ± 0.13	-0.51 ± 0.16	-0.71 ± 0.21	1.26 ± 0.11
VQ-CEF	-0.15 ± 0.09	-0.54 ± 0.22	0.24 ± 0.15	1.32 ± 0.02

Table 1: Quantitative performance evaluation for **Density Estimation** in terms of the test log-likelihood in nats (higher the better) on the toy 3D Datasets. The values are averaged across 5 independent trials, $\pm$ represents the 95% confidence interval.

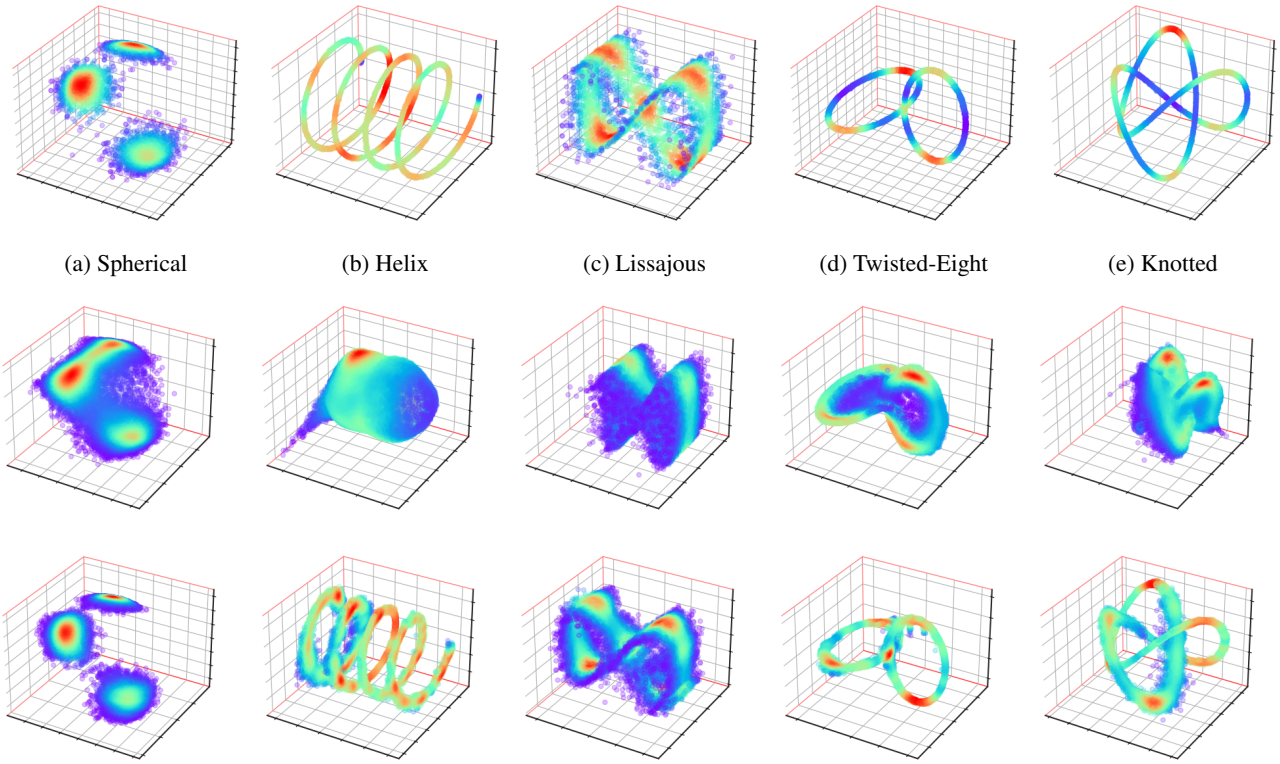
Model	Non-Knotted	Bent-Lissajous	Disjoint-Circles	Star
Real NVP	0.53 ± 0.18	1.04 ± 0.22	1.71 ± 0.12	3.33 ± 0.18
VQ-RealNVP	2.39 ± 0.24	2.62 ± 0.13	2.71 ± 0.19	4.23 ± 0.06
MAF	0.73 ± 0.18	1.48 ± 0.11	1.95 ± 0.12	3.53 ± 0.03
VQ-MAF	2.41 ± 0.19	2.06 ± 0.12	2.87 ± 0.07	3.59 ± 0.12
CEF	-0.46 ± 0.13	-0.51 ± 0.16	-0.71 ± 0.21	1.26 ± 0.11
VQ-CEF	-0.15 ± 0.09	-0.54 ± 0.22	0.24 ± 0.15	1.32 ± 0.02

Table 2: Quantitative performance evaluation for **Sample Generation** in terms of the log-likelihood in nats (higher the better) on the toy 3D Datasets. The values are averaged across 5 independent trials, $\pm$ represents the 95% confidence interval.



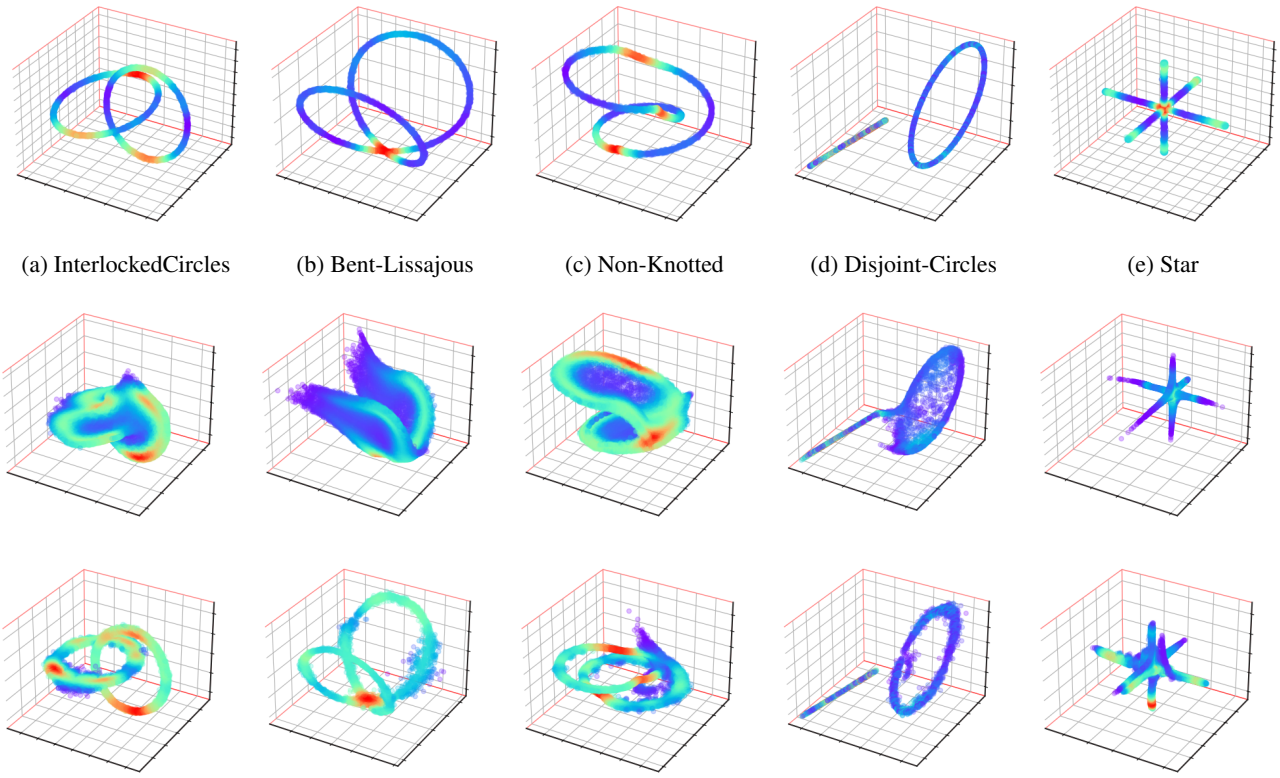
(a) InterlockedCircles (b) Bent-Lissajous (c) Non-Knotted (d) Disjoint-Circles (e) Star

Figure 7: Qualitative visualization of the samples generated by a classical flow - **RealNVP** (Middle Row) and its VQ-counterpart (Bottom Row) trained on Toy 3D data distributions (Top Row).



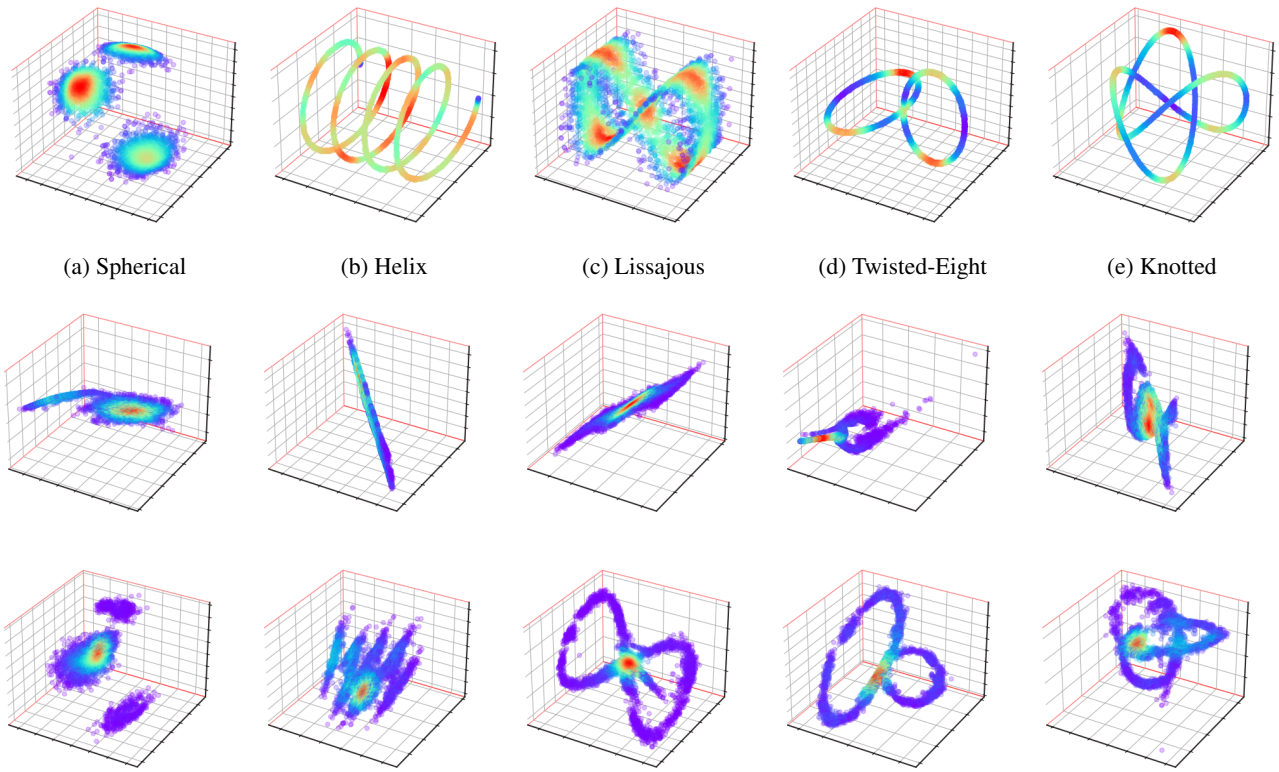
(a) Spherical (b) Helix (c) Lissajous (d) Twisted-Eight (e) Knotted

Figure 8: Qualitative visualization of the samples generated by a classical flow - **MAF** (Middle Row) and its VQ-counterpart (Bottom Row) trained on Toy 3D data distributions (Top Row).



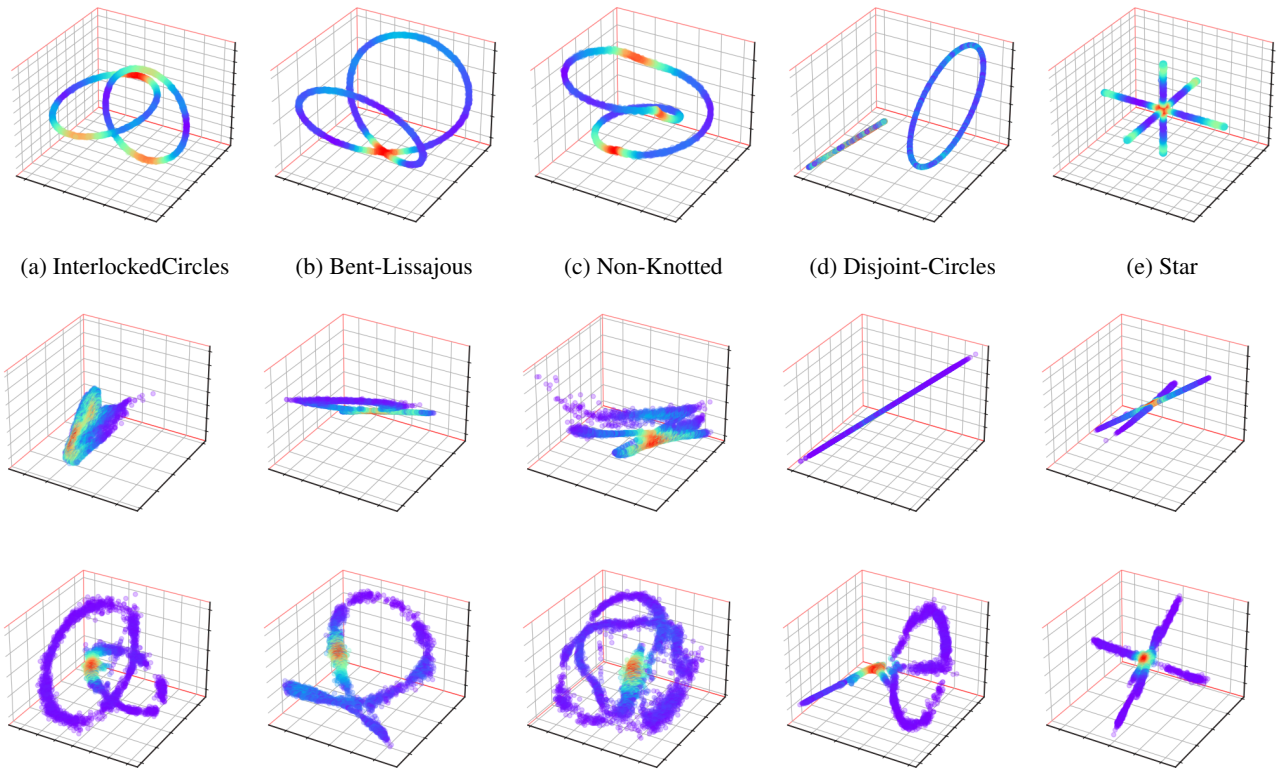
(a) InterlockedCircles (b) Bent-Lissajous (c) Non-Knotted (d) Disjoint-Circles (e) Star

Figure 9: Qualitative visualization of the samples generated by a classical flow - **MAF** (Middle Row) and its VQ-counterpart (Bottom Row) trained on Toy 3D data distributions (Top Row).



(a) Spherical (b) Helix (c) Lissajous (d) Twisted-Eight (e) Knotted

Figure 10: Qualitative visualization of the samples generated by **CEF** (Middle Row) and its VQ-counterpart (Bottom Row) trained on Toy 3D data distributions (Top Row). CEF consists of a 2-dimensional RealNVP flow post composed with a conformal embedding that raises it to 3D.



(a) InterlockedCircles (b) Bent-Lissajous (c) Non-Knotted (d) Disjoint-Circles (e) Star

Figure 11: Qualitative visualization of the samples generated by **CEF** (Middle Row) and its VQ-counterpart (Bottom Row) trained on Toy 3D data distributions (Top Row). CEF consists of a 2-dimensional RealNVP flow post composed with a conformal embedding that raises it to 3D.