Denoising for Manifold Extrapolation

Zeyu Yun
UC Berkeley
chobitstian@berkeley.edu

Galen Chuang
UC Berkeley
galenc@berkeley.edu

Derek Dong
University of Cambridge & UC Davis
jd976@cam.ac.uk

Yubei Chen
UC Davis
ybchen@ucdavis.edu

Abstract

Deep neural network-based image denoisers are the key component in high quality diffusion models. Unlike latent variable models, denoisers are not explicitly trained to extract latent manifolds. However, recent work suggests that denoisers implicitly capture the geometry of these manifolds. Given that images lie on a low-dimensional latent manifold embedded in a high dimensional space, this raises the question: can we recover a latent manifold from a diffusion model? Here, we demonstrate that a manifold embedded in a trained denoiser can be extracted and visualized through manifold extrapolation. We provide a theoretical framework and an algorithm for performing manifold extrapolation using a denoiser, and we show that our approach outperforms traditional manifold extrapolation methods. Finally, we demonstrate that when the latent manifold is simple and low-dimensional, it can be extracted using manifold extrapolation with only two nearby points.

1 Introduction

The manifold hypothesis suggests that many high-dimensional datasets lie along low-dimensional latent manifolds [1, 2, 3]. Various methods have been developed to model such latent manifolds, including manifold learning approaches [4, 5, 6] and latent variable models [7, 8]. The power of deep neural networks has significantly expanded the capacity of latent variable models, enabling them to represent complicated data distributions such as natural images [9, 10, 11]. Previous work has focused on imposing additional constraints on the latent space to extract low-dimensional latent manifolds from data [12, 13, 14, 15]. However, these models are empirically challenging to train and scale for large datasets [16, 17, 18, 10, 19, 20]. One approach is to use a deep learning model trained on a simpler learning objective: denoising. Rather than explicitly modeling the latent manifold, diffusion models are a different class of generative models that leverage denoising [21, 22, 23, 19].

Why do diffusion models perform so well? Recent studies show that the essential component of diffusion models—a deep neural network-based denoiser—implicitly captures the geometric structure of the data manifold [24, 25, 26, 27]. However, these studies do not demonstrate how to visualize, interpret, or evaluate the implicit manifold learned by a denoiser. Unlike latent variable models, where the latent space explicitly represents the manifold or its flattened version, we show that the implicit manifold learned by a denoiser can be visualized through **manifold extrapolation**. Similar techniques have been employed to study the geometry of latent variable models [12, 28]. In this work, we provide a theoretical framework and an algorithm for manifold extrapolation using a denoiser. We demonstrate that when the latent manifold is simple and low-dimensional, it can be effectively extrapolated given the denoiser and two points on the manifold.

Workshop on Scientific Methods for Understanding Deep Learning, NeurIPS 2024.

2 Method

2.1 Extrapolation on the image manifold

As illustrated Fig. 1.A, an image transformation can be modeled as a manifold extrapolation process. Let $p(x)$ denote the likelihood function of the image space. The image manifold is defined as the region with equally high probability, represented as a dark red circle in Fig. 1B.

The extrapolation algorithm can be summarized in two steps. First, move in a direction within the *tangent space* $T_x\mathcal{M}$ of the manifold. However, since the tangent space is only a linear approximation of the manifold, this movement does not perfectly align with the manifold, leading to a small error that causes the trajectory to drift off the manifold. Next, to correct for this deviation, a step is taken in the direction of the gradient of the log-likelihood function $\nabla \log p(x)$, which is also known as the *score function*. This step pulls the point back toward the manifold, compensating for the error introduced by the initial movement within the tangent space. This procedure is shown in Alg. 1.

Algorithm 1 Tangent Space Extrapolation

```
Require: initial real image vectors $u_0, u_1 \in \mathbb{R}^N$; step size $\eta$; noise level $\sigma$
  $x_0 := u_0 + \epsilon_0$                                      ▷ $\epsilon_i$ is Gaussian noise $\sim \mathcal{N}(0, \sigma^2 I)$
  $x_1 := u_1 + \epsilon_1$                                      ▷ $x_i$ denotes image vector in latent (noisy) space
  frames := $[x_0, x_1]$
  for $i = 2 \dots$ steps do
      $V_i :=$ GetTopKEigenvectors$(\phi_\theta, x_{i-1})$           ▷ $V_i$ is $K \times N$ matrix, $\phi_\theta$ is denoiser (Alg. 4)
      $v_i := x_{i-1} - x_{i-2} \in \mathbb{R}^N$                 ▷ new velocity vector between two previous noisy image
      $p_i := V_i^\top V_i v_i$                                  ▷ projection of velocity vector onto eigenvectors (tangent space)
      $p_i = \frac{\|v_i\|_2}{\|p_i\|_2} p_i$                         ▷ normalize the velocity vector
      $x_{pi} := x_{i-1} + \eta p_i$                             ▷ step in the direction of the projected velocity vector
      $u_i := x_{pi} - \phi_\theta(x_{pi})$                         ▷ denoise to go back to the manifold
      $x_i := u_i + \epsilon_i$                                  ▷ add noise for next step
      Append(frames, $u_i$)                                   ▷ save to visualize
  end for
  return frames
```

In the following section, we explain how to compute the *tangent space* and the *score function*, detailing the methods used to derive and apply them in the extrapolation process.

2.2 Parameterization of the score function

The score function can be parameterized as a denoiser. The connection between the score function and denoiser was first studied as Empirical Bayes by statisticians [29, 30]. This idea was later popularized and rediscovered in numerous works, and is commonly referred to as Tweedie's formula [31, 27, 32, 33, 34, 24, 22]. Consider $x = u + \epsilon$, where u is the clean image vector and $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ is the additive Gaussian noise. The Minimum Mean Square Estimator (MMSE) denoiser is given by $\hat{x} = \mathbb{E}(u|x) = \int uP(u|x)du$. Tweedie's formula states:

$$\hat{x} = \mathbb{E}(u|x) = x - \sigma^2 \nabla \log p_\sigma(x) \quad (1)$$

where $p_\sigma(x)$ denotes the blurred prior, or the likelihood function of the image with additive noise of standard deviation σ. This formula suggests that the score function can be parameterized using an MMSE denoiser, and previous work has shown that convolutional neural networks can effectively estimate MMSE denoisers [35, 36, 37]. Thus, we can parameterize the score function by training a neural network ϕ_θ for the denoising task:

$$\nabla \log p_\sigma(x) = \frac{1}{\sigma^2}\phi_\theta(x) \quad \text{s.t.} \quad \theta = \arg\min_\theta \mathbb{E}_{x\sim p_{data}(x)} \mathbb{E}_{\epsilon \sim \mathcal{N}(0,\sigma^2 I)} ||\epsilon - \phi_\theta(x+\epsilon)||^2 \quad (2)$$

We use a convolutional neural network with UNet structure [38] and residual connections to model the denoiser ϕ_θ, and optimize with the Adam optimizer [39].

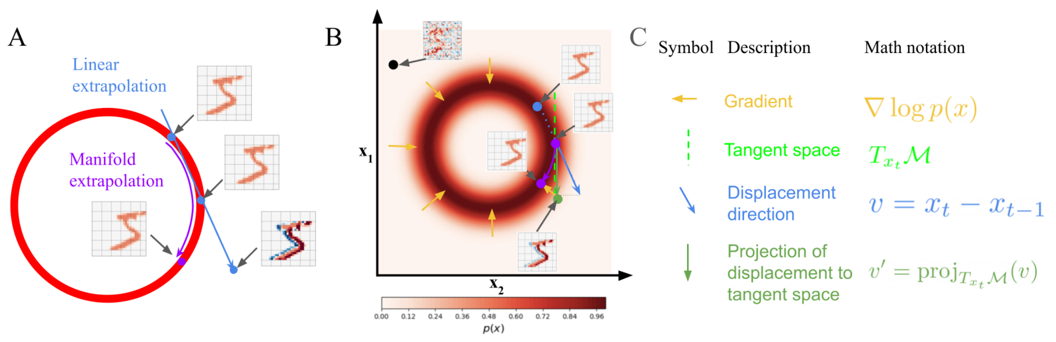

Figure 1: (A) An illustration of the difference between linear extrapolation and manifold extrapolation. Red represents positive value, and blue represent negative value pixels. Manifold extrapolation results in a non-linear transform of an image, while a linear transform superimposes two images. (B) Illustration for the manifold extrapolation algorithm. First, we use the heat map to show the likelihood function $p(x)$. All real digits have high density and lie on a circles. Noisy images (black dot at top right corner) has low density. To extrapolate on the manifold, starting at the purple dot, we first calculate the displacement vector as the difference between current state (purple dot) and previous state (blue dot). We project this displacement vector onto the tangent space. Next, we move in the tangent direction (green arrow), but because it is a linear approximation of the image manifold, we briefly fall off the manifold (green dot). To get back onto the manifold (bottom purple dot), we take a step in the direction of the gradient. (C) Notation used in (B).

2.3 Extrapolation and tangent space of the manifold

To sample an image, we follow the gradient of the log-likelihood function. This implies that the gradient will be zero at real images i.e. $\nabla \log p_\sigma(x) = 0$. Consequently, we can define the image manifold as the set of critical points of $p_\sigma(x)$, that is, $\mathcal{M} = \{x \mid \nabla \log_\sigma p(x) = 0\}$. Extrapolation is equivalent to moving to another point on the manifold. Formally, we want to find a displacement vector v such that $\nabla \log p_\sigma(x+v) - \nabla \log p_\sigma(x) \approx 0$. By Taylor expansion, we have

$$\nabla \log p_\sigma(x+v) - \nabla \log p_\sigma(x) \approx \nabla^2 \log p_\sigma(x) \cdot v \tag{3}$$

To enforce the above quantity to be zero, we want v in nullspace of $\nabla^2 \log p_\sigma(x)$. By corollary 0.1, the nullspace of Hessian is the tangent space of the image manifold $\mathcal{M}$. Therefore, the extrapolation direction v should be in the tangent space of the manifold.

Corollary 0.1 Let $\mathcal{M} = \{x : \nabla \log p(x) = 0\}$. Then, $T_x\mathcal{M} = Null(\nabla^2 \log p(x))$.

Computationally, the tangent space can be determined as the nullspace of the Hessian, where the Hessian is equivalent to the Jacobian of the score function. Since the score function is parameterized as a denoising neural network, the Jacobian of the denoiser can be efficiently computed using back-propagation. To calculate the nullspace of the Jacobian, we compute the eigenvectors corresponding to zero eigenvalues.

For practical purposes, we focus on the k-smallest eigenvectors, which represent the low-curvature linear subspace of the log-likelihood function. We hypothesize that the manifold of real images is rough and bumpy. To avoid getting stuck in local minima, extrapolation is conducted within this low-curvature subspace rather than strictly within the tangent space. Additionally, since computing the full Jacobian is computationally intensive, we accelerate the process by calculating only the k lowest eigenvectors, leveraging the Jacobian Vector Product (JVP) combined with power iteration. This method, detailed in Algorithm 4, allows efficient approximation while capturing essential low-curvature directions for robust extrapolation.

3 Results

We first evaluate our algorithm on two synthetic datasets: **translated MNIST** and **rotated MNIST**. The latent manifold for an image in each dataset is known: a line segment $[0, 1]$ and a unit circle S^1,

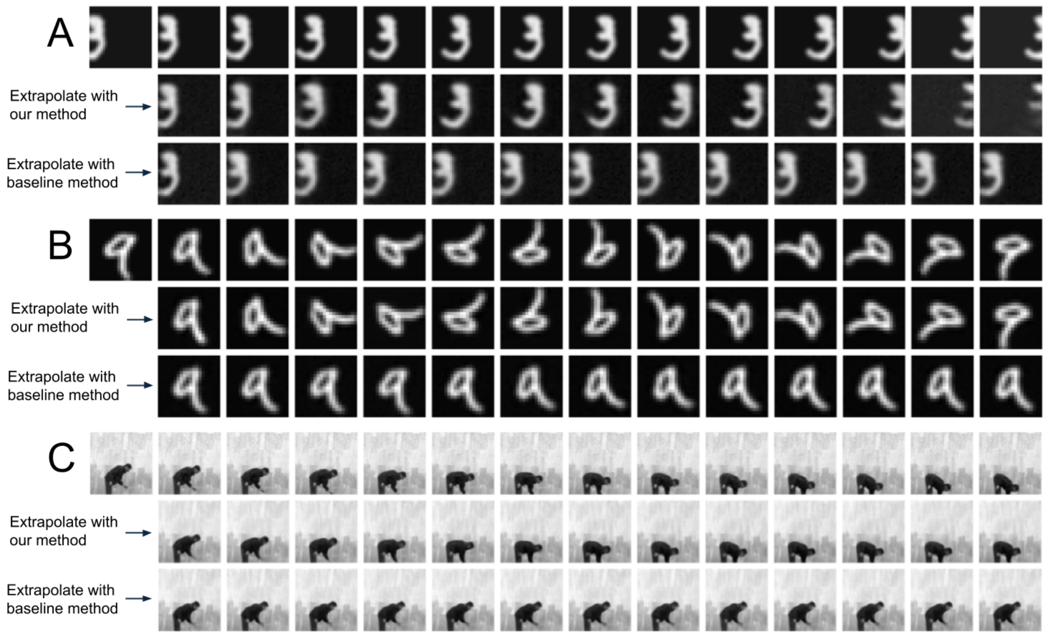
A
Extrapolate with our method
Extrapolate with baseline method
B
Extrapolate with our method
Extrapolate with baseline method
C
Extrapolate with our method
Extrapolate with baseline method

Figure 2: Visualization of the extrapolated video sequence compared to the ground truth video sequence. Each sub-figure shows the extrapolation result from different datasets: (A) translated MNIST, (B) rotated MNIST, and (C) the Weizmann Action dataset. The first row in each sub-figure contains the ground truth video sequence. The second and third rows show the extrapolation results given the denoiser and the first two frames from the ground truth video. The second row presents the extrapolation results using our algorithm, while the third row shows the extrapolation results using a baseline method.

respectively. Finally, we evaluate our algorithm on a dataset of real images: **the Weizmann Action dataset** [40], where the underlying manifold is unknown.

3.1 Qualitative results

For the transformed MNIST datasets, the denoiser is trained on a random ordering of transformed images. For the Weizmann dataset, the the frames are randomly shuffled. Once the denoiser trains to convergence, we use the first two frames from each video to extrapolate future frames. The extrapolation results are visualized in Fig. 2. A common approach to interpolate between two images using a denoiser involves linearly interpolating in the noisy image space and then denoising the interpolated result [19]. Similarly, we perform linear extrapolation in the noisy space and use this as a baseline for our manifold extrapolation algorithm (see Algorithm 2. As shown in Fig. 2, the extrapolated video closely resembles the ground truth video and is qualitatively better than the extrapolated video produced by the baseline method. Furthermore, We demonstrate that our manifold extrapolation algorithm can recover the entire latent manifold from the denoiser in Fig. 3.

3.2 Quantitative results

To evaluate how well the model extrapolates, we require a measure of how well the extrapolated video aligns with the ground truth video. First, we define the distance between two individual video frame as $d(u_1, u_2) = 1 - SSIM(u_1, u_2)$, where $SSIM(u_1, u_2)$ refers to the Structural Similarity Index [41] between frame u_1 and u_2. Because the extrapolated video sequence does not perfectly align with ground truth video sequence, we use *Dynamic Time Warping (DTW)* to align the two sequences and evaluate the distance between them. We use *DTW-SSID* to denote this distance. More detailed alignment algorithm and the visualization of the alignment can be found in Appendix A.2.

Table 1 demonstrates that our method outperforms both baseline approaches. The experiment setups that produced these results are detailed in Appendix A.4, A.5, and A.6.

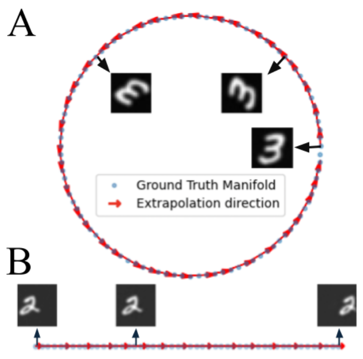

Table 1: Comparison of methods across different datasets. For each dataset, the results are averaged over 5 trials. Both rotated MNIST and translated MNIST are averaged over 10 video sequences. Weizmann Action is averaged over 9 video sequences. Check Appendix A.4, A.5, and A.6 for more details.

Method	**Translated MNIST**	**Rotated MNIST**	**Weizmann Action**
Denoiser Baseline	0.727	0.700	0.158
VAE	0.834	0.840	0.515
Our Method	0.119	0.273	0.142

Figure 3: We represent each ground truth video frame as a blue dot on its latent manifold. The red arrow represents the direction of movement on the manifold from each extrapolation step. (A) Extrapolating a rotated MNIST digit, where the latent manifold is a unit circle. (B) Extrapolating a translated MNIST digit, where the latent manifold is a line segment.

4 Discussion and future work

We demonstrate that our proposed manifold extrapolation method, using deep neural network-based denoisers, outperforms baseline methods. By capturing the geometric structure of simple latent manifolds, our method enables smoother and more accurate extrapolation of image sequences with very few assumptions. This is particularly evident in low-dimensional settings, where our approach recovers and extends latent manifolds with minimal computational complexity. This method only requires a pre-trained, differentiable denoiser, and there are no assumptions on the structure of the neural network, the curvature of the latent space, nor any supervision. The combination of manifold learning and diffusion models offers a promising direction for future research in image and video generation, particularly for tasks requiring long-term prediction and temporal consistency.

There are many promising directions to further enhance this work. We have provided a theoretical framework for how a "good" denoiser can be used for image extrapolation. However, we have not yet analyzed what defines a "good" denoiser or how the features learned by the denoiser enable effective extrapolation. A key may lie in further analysis of the generalization abilities of the Jacobians of deep denoisers as in [25]. Additionally, the standard deviation σ of the added noise is an important part of the algorithm, and if chosen incorrectly, can affect the extrapolation quality. More analysis is needed on the effect of σ on denoiser generalization properties.

Our method has so far only been applied to simple datasets and transformations. As the space of transformations present in the data becomes more complex, so does the latent image manifold. Although diffusion models can be empirically scaled to much larger datasets, the corresponding manifolds can become entangled, causing extrapolation to be “trapped" in local minima. To improve the model’s performance on more realistic datasets, different methods should be explored, such as interpolation strategies, conditional denoisers, and latent diffusion models.

Acknowledgments and Disclosure of Funding

We thank the anonymous reviewers for the Workshop on Scientific Methods for Understanding Deep Learning for their thoughtful questions and comments, and have incorporated their feedback into the future work section.

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE2146752.

References

[1] Alexander N Gorban and Ivan Yu Tyukin. Blessing of dimensionality: mathematical foundations of the statistical physics of data. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2118):20170237, 2018.

[2] Lawrence Cayton et al. *Algorithms for manifold learning*. eScholarship, University of California, 2008.

[3] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.

[4] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[5] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[6] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[7] Anthony J Bell and Terrence J Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159, 1995.

[8] Aapo Hyvärinen and Erkki Oja. A fast fixed-point algorithm for independent component analysis. *Neural computation*, 9(7):1483–1492, 1997.

[9] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

[10] 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.

[11] Diederik Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[12] Hang Shao, Abhishek Kumar, and P Thomas Fletcher. The riemannian geometry of deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 315–323, 2018.

[13] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

[14] Irina Higgins, Loic Matthey, Arka Pal, Christopher P Burgess, Xavier Glorot, Matthew M Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR (Poster)*, 3, 2017.

[15] Yijia Zheng, Tong He, Yixuan Qiu, and David P Wipf. Learning manifold dimensions with conditional variational autoencoders. *Advances in Neural Information Processing Systems*, 35:34709–34721, 2022.

[16] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge? In *International conference on machine learning*, pages 3481–3490. PMLR, 2018.

[17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[18] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[20] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[21] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015.

[22] Saeed Saremi and Aapo Hyvärinen. Neural empirical bayes. *Journal of Machine Learning Research*, 20(181):1–23, 2019.

[23] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[24] Zahra Kadkhodaie and Eero P. Simoncelli. Solving linear inverse problems using the prior implicit in a denoiser, 2021.

[25] Zahra Kadkhodaie, Florentin Guth, Eero P. Simoncelli, and Stéphane Mallat. Generalization in diffusion models arises from geometry-adaptive harmonic representations, 2024.

[26] Peyman Milanfar and Mauricio Delbracio. Denoising: A powerful building-block for imaging, inverse problems, and machine learning, 2024.

[27] Mauricio Delbracio and Peyman Milanfar. Inversion by direct iteration: An alternative to denoising diffusion for image restoration. *arXiv preprint arXiv:2303.11435*, 2023.

[28] Binxu Wang and Carlos R Ponce. The geometry of deep generative image models and its applications. *arXiv preprint arXiv:2101.06006*, 2021.

[29] Koichi Miyasawa et al. An empirical bayes estimator of the mean of a normal population. *Bull. Inst. Internat. Statist*, 38(181-188):1–2, 1961.

[30] Herbert E Robbins. An empirical bayes approach to statistics. In *Breakthroughs in Statistics: Foundations and basic theory*, pages 388–394. Springer, 1992.

[31] Martin Raphan and Eero P Simoncelli. Least squares estimation without priors or supervision. *Neural computation*, 23(2):374–420, 2011.

[32] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[33] Durk Kingma and Yann LeCun. Regularized estimation of image statistics by score matching. *Advances in neural information processing systems*, 23, 2010.

[34] Guillaume Alain and Yoshua Bengio. What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593, 2014.

[35] Sreyas Mohan, Zahra Kadkhodaie, Eero P Simoncelli, and Carlos Fernandez-Granda. Robust and interpretable blind image denoising via bias-free convolutional neural networks. *arXiv preprint arXiv:1906.05478*, 2019.

[36] Kai Zhang, Wangmeng Zuo, Yunjin Chen, Deyu Meng, and Lei Zhang. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, 26(7):3142–3155, 2017.

[37] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018.

[38] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[39] P Kingma Diederik. Adam: A method for stochastic optimization. *(No Title)*, 2014.

[40] Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. Actions as space-time shapes. *Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, December 2007.

[41] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[42] John Hubbard and Barbara Hubbard. *Vector calculus, linear algebra, and differential forms: a unified approach*. Matrix Editions, 2015.

A Appendix

A.1 Proof of the corollary

Theorem 1 *Suppose that $U \subset \mathbb{R}^n$ is an open subset, $F : U \to \mathbb{R}^{n-k}$ is a C^1 mapping, and manifold $\mathcal{M}$ can be described as the set of points that satisfy $F(z) = 0$. If $\nabla F(c)$ is onto for $c \in \mathcal{M}$, then the tangent space $T_c\mathcal{M}$ is the kernel of $\nabla F(c)$.*

Theorem 1 is proven using implicit function theorem and can be found as Theorem 3.2.4 in [42]. The corollary 0.1 simply follows from Theorem 1. It can be shown by substituting function F in Theorem 1 as $\nabla \log p(x)$.

A.2 Calculating distance between two video sequences

We define the SSIM distance (SSID) between two individual video frames as

$$\mathrm{ssid}(u_1, u_2) = 1 - \mathrm{ssim}(u_1, u_2),$$

where $\mathrm{ssim}(u_1, u_2)$ represents the Structural Similarity Index Measure (SSIM) [41] between frames u_1 and u_2. A distance of $\mathrm{ssid} = 0$ indicates identical frames, $\mathrm{ssid} = 1$ implies no correlation, and $\mathrm{ssid} > 1$ denotes a significant mismatch.

However, because the extrapolated video may have a different temporal progression compared to the ground truth video, they are often misaligned in time. To facilitate meaningful comparisons between the two sequences, even when the generated video is temporally stretched or compressed, we apply Dynamic Time Warping (DTW) to optimally align the two sequences by warping them non-linearly in the temporal dimension.

The process proceeds as follows:

1. **Frame distance matrix construction**: We compute a $N_1 \times N_2$ distance matrix D, where entry d_{n_1,n_2} corresponds to the SSIM distance between the n_1-th frame of the first video and the n_2-th frame of the second video. Here, N_1 and N_2 represent the number of frames in the two video sequences.

2. **Optimal path finding**: Using the distance matrix D, we identify the optimal alignment path between the two sequences, which corresponds to a trajectory from d_{N_1,N_2} to $d_{0,0}$ that minimizes the cumulative SSIM distance. When the algorithm moves to a point (n_1, n_2), it decides the next step by evaluating whether to move to $(n_1 - 1, n_2)$, $(n_1, n_2 - 1)$, or $(n_1 - 1, n_2 - 1)$, based on which of these points has the minimal SSIM distance. SSIM distance matrix and optimal alignment path are visualised in Fig. 4 (1).

3. **Sequence alignment**: After determining the optimal path, we adjust the extrapolated video (with N_2 frames) to match the length of the ground truth video (with N_1 frames). This aligning process is visualised in Fig. 4 (2). This DTW-aligned extrapolated video is then directly comparable to the ground truth video. An example of SSIM distance values compared to the ground truth video are visualised in Fig. 4 (3).

4. **Video distance calculation**: Finally, the overall distance between the two videos is calculated as the average frame-wise distance. We denote this measure as *DTW-SSID* and use it to quantify the quality of the extrapolated video sequence.

A.3 Algorithms

We show our algorithms for efficiently calculating the top eigenvectors and extrapolation on the image manifold.

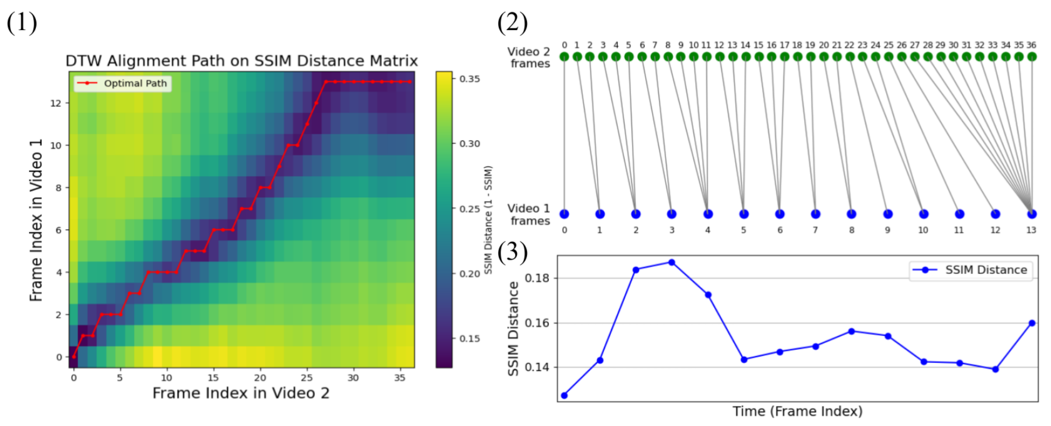

Figure 4: (1) Visualisation of DTW Alignment Path on SSIM Distance Matrix. The heatmap shows the SSIM distances between corresponding frames of the extrapolated video (on the horizontal axis) and the ground truth video (on the vertical axis). The red line represents the optimal path found using Dynamic Time Warping (DTW), which minimizes the overall SSIM distance. (2) Frame-to-frame alignment between video 1 and video 2, according to the DTW alignment path. Blue points represent frames from video 1, while green represents video 2. The connecting lines illustrate how individual frames from video 2 (extrapolated) are aligned with video 1 (ground truth) frames. (3) SSIM distance after alignment. The horizontal axis represents the frame index of the ground truth video and it matches video 1 frames in (2). In this example, the consistently low SSIM distance values along the trajectory suggest a strong correlation between the DTW-aligned extrapolated video and ground truth videos.

Algorithm 2 Baseline Extrapolation

```
Require: initial real image vectors $u_0, u_1 \in \mathbb{R}^N$; step size $\eta$; noise level $\sigma$
  $x_0 := u_0 + \epsilon_0$                          ▷ $\epsilon_i$ is Gaussian noise $\sim \mathcal{N}(0, \sigma^2 I)$
  $x_1 := u_1 + \epsilon_1$                          ▷ $x_i$ denotes image vector in latent (noisy) space
  frames := $[x_0, x_1]$
  for $i = 2 \dots$ steps do
      $v_i := x_{i-1} - x_{i-2} \in \mathbb{R}^N$        ▷ new velocity vector between two previous noisy image
      $x_{vi} := x_{i-1} + \eta v_i$                 ▷ step in the direction of the velocity vector
      $u_i := x_{vi} - \phi_\theta(x_{vi})$           ▷ denoise to visualize
      Append(frames, $u_i$)
      $x_i := u_i + \epsilon_i$                      ▷ add noise for next step
  end for
  return frames
```

Algorithm 3 VAE Extrapolation

```
Require: initial real image vectors $u_0, u_1 \in \mathbb{R}^N$; step size $\eta$
  $x_0 :=$ encode$(u_0)$                            ▷ $x_i$ denotes image vector in latent space
  $x_1 :=$ encode$(u_1)$
  frames := $[x_0, x_1]$
  for $i = 2 \dots$ steps do
      $v_i := x_{i-1} - x_{i-2} \in \mathbb{R}^N$        ▷ new velocity vector between two latent vectors
      $x_{vi} := x_{i-1} + \eta v_i$                 ▷ step in the direction of the velocity vector
      $u_i :=$ decode$(x_{vi})$                      ▷ decode to visualize
      Append(frames, $u_i$)
  end for
  return frames
```

Algorithm 4 GetTopKEigenvectors

```
Require: denoiser $\phi_\theta$; noisy image $x \in \mathbb{R}^d$; number of eigenvectors to return $K$; placeholder array
    for eigenvector, initialized as $V = [x, x, ...x]$
  for $i$ in $1 \ldots$ steps do
      $v_0 = x$
      while $||v_t - v_{t-1}|| > \epsilon$ do
          $v_t = \frac{\text{JVP}(\phi_\theta, v_{t-1}) - VV^t v_{t-1}}{||v_{t-1}||}$    ▷ Recursively applying Jacobian Vector Product
      end while
      Append $v_t$ to V
  end for
  return $V$    ▷ $K \times N$ matrix
```

A.4 Dataset preparation

MNIST Rotation The first ten digit images from the MNIST dataset were cropped from their original size of 28x28 pixels to 20x20 pixels. Each image was subsequently rotated with a step size of 3.6 degrees, resulting in 100 discrete frames per image. This process yielded ten videos, each containing 100 frames of a rotating digit. These 10x100 frames were used to train the denoiser models, with 200 frames randomly selected as the test set. The ten videos also served as ground truth and were compared to the extrapolation results.

MNIST Translation The first ten digit images from the MNIST dataset were cropped from their original size of 28x28 pixels to 20x20 pixels. Each image was then translated horizontally in increments of 0.12 pixels, producing 100 discrete frames per image, covering a translation range from -6 to +6 pixels. This process generated ten videos, each consisting of 100 frames of a translating digit. These 1,000 frames were used to train the denoiser models, with 200 frames randomly selected as the test set. The ten videos also served as ground truth and were compared to the extrapolation results.

Weizmann Action All video frames from the Weizmann Action dataset were used as the training set, with 200 frames randomly selected for the test set.

A.5 Model details

UNet

We use a UNet of the same architecture and training procedure as in [25], with 512 latent dimensions.

VAE

We use a basic convolutional VAE with the following architecture:

VAE encoder
Conv $3 \times 3 \times 32$ (stride 2), batch norm, leaky ReLU
Conv $3 \times 3 \times 64$ (stride 2), batch norm, leaky ReLU
Conv $3 \times 3 \times 128$ (stride 2), batch norm, leaky ReLU
Conv $3 \times 3 \times 256$ (stride 2), batch norm, leaky ReLU
Conv $3 \times 3 \times 512$ (stride 2), batch norm, leaky ReLU
FC 512

The decoder is transpose convolutions in the reverse order, and a final convolution layer that differs in dimension for each dataset. As with the UNet used in the main experiments, the latent dimension is 512. We optimize using Adam [39] with a learning rate of 0.005.

A.6 Extrapolation details

This section outlines the extrapolation setups used to generate Table 1. For each dataset, the final value was obtained by averaging the results of 5 trials. The following settings are for the denoiser method.

MNIST Rotation Each of the 10 ground truth videos contained 100 frames. The first and second frames of each video were provided to the extrapolators, which generated 53 subsequent frames. The step size was set to $\eta = 1$, and the noise level was $\sigma = 25/255$.

MNIST Translation Each of the 10 ground truth videos contained 100 frames. The first and second frames of each video were provided to the extrapolators, which generated 40 subsequent frames. The step size was set to $\eta = 0.3$, and the noise level was $\sigma = 15/255$.

Weizmann Action Each of the 9 ground truth videos contained 5 frames. The first and second frames of each video were provided to extrapolators, which then generated 6 frames. The step size was $\eta = 0.15$, and the noise level was $\sigma = 15/255$.

For the VAE, the same settings as above were used, but with step sizes of $\eta = 0.8$ for rotated MNIST, $\eta = 0.9$ for translated MNIST, and $\eta = 0.3$ for Weizmann.