Dynamics of Diffusion

A Thesis Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Landon K. Harris

May 2024

© by Landon K. Harris, 2024
All Rights Reserved.

Acknowledgments

My journey through this thesis would not have been possible without the encouragement and support of several individuals. First and foremost, I would like to express my deepest appreciation to my advisor and committee chair, Dr. Hairong Qi, for believing in me back in the EcoCAR days - and continuing to do so today. Thank you for your patience, invaluable advice, and unwavering support. I would also like to thank the members of my thesis committee, Dr. Dan Wilson and Dr. Weizi Li for their time and input.

I would like to thank my friends in the Advanced Imaging and Collaborative Information Processing (AICIP) group: Andrei Cozma, Coby White, Fanqi Wang, Weisheng Tang, Konstantinos Georgio, Maofeng Tang, Connor Parham, and Quan Zhou. The insightful discussions and collaborative atmosphere fostered by this group have not only provided support along the way but a welcome change of perspective every now and then.

Lastly, I'm grateful to my family for their support, love, and enthusiasm. Without a powerful role model, I never would have climbed to where I am today.

Aut viam inveniam aut faciam

Abstract

This thesis presents a novel approach to understanding Denoising Diffusion Probabilistic Models (DDPMs) by viewing the nonlinear diffusion process as a dynamical system through the lens of Koopman Operator Theory. By representing the nonlinear nature of the reverse diffusion process in DDPMs as a linear operator using Koopman Operator Theory, this work provides unique insights into how diffusion transforms probability distributions between a simple input distribution and a complex target distribution. This perspective allows for the identification of a subspace where the diffusion system is linear, enabling a linear operator to be learned using Predictive Autoencoders, a task previously accomplished using nonlinear U-Net or transformer models with large capacity.

This work contributes to the field by offering a different perspective on the image diffusion process, connecting it to a broader range of dynamical systems and providing a new avenue for understanding and improving the performance of diffusion models. Additionally, this work presents a comprehensive analysis of the properties of the linear subspace and its implications for the stability and robustness of the diffusion process. By thoroughly investigating the characteristics of this subspace, this thesis provides insights into the fundamental mechanisms that govern the behavior of diffusion models and offers guidelines for designing more effective and reliable systems.

Table of Contents

1 Introduction **1**

1.1 Motivation 1

1.2 Thesis Contributions 2

1.3 Thesis Outline 3

2 Koopman Operator Theory **5**

2.1 Dynamical Systems 5

2.2 The Koopman Operator 9

2.3 Koopman Invariant Subspaces 13

2.4 Approximation Methods 13

2.4.1 Dynamic Mode Decomposition 14

2.4.2 Predictive Autoencoders 17

3 Diffusion Modeling **20**

3.1 Linear Inverse Problems 20

3.1.1 Definition 21

3.1.2 Bayesian Approach to Inverse Problems 22

3.2 Diffusion Probabilistic Models 24

3.2.1 Denoising Diffusion Probabilistic Models 24

3.2.2 Latent Diffusion Models 30

4 Methodology **32**

4.1 Approximation Methods . . . 32
4.1.1 Dynamic Mode Decomposition . . . 32
4.1.2 Predictive Autoencoders . . . 35

5 Results and Discussion **40**
5.1 Evaluation . . . 40
5.1.1 Datasets . . . 40
5.1.2 Characterization of Dynamics . . . 42
5.2 Dynamic Mode Decomposition . . . 45
5.2.1 Exact Dynamic Mode Decomposition . . . 45
5.2.2 Extended Dynamic Mode Decomposition . . . 47
5.2.3 Hankel Dynamic Mode Decomposition . . . 49
5.3 Predictive Autoencoder . . . 50
5.3.1 Compressive-Predictive Autoencoder . . . 50
5.3.2 Expansive-Predictive Autoencoder . . . 52
5.3.3 Extension to Other Datasets . . . 55
5.3.4 Hyperparameter Search . . . 56
5.4 Comparison With DDPMs . . . 58

6 Conclusion and Future Work **60**

Bibliography **62**

Appendix **67**
A Dynamics Classification . . . 67
B Experiment Results . . . 69
C Supplementary Code . . . 94

Vita **100**

List of Tables

5.1 Parameters used for the two selected Compressive-Predictive Autoencoder models. 51
5.2 Parameters used for the two selected Expansive-Predictive Autoencoder models. 54
5.3 Parameters used for the three selected models trained on the CIFAR-10 dataset. 57

List of Figures

2.1 Phase portrait of a simple, nonlinear ODE. 11
2.2 Phase portrait of a simple, nonlinear ODE with trajectories. 12
2.3 Phase portrait of a simple, nonlinear ODE with DMD trajectories. . . 18

3.1 Effect of different beta-schedules on the forward diffusion process for the MNIST dataset. 26
3.2 A directed graph representing the diffusion process 29

4.1 Structure of a Compressive-Predictive Autoencoder. 36
4.2 Structure of an Expansive-Predictive Autoencoder. 36
4.3 Comparison of loss functions on the MNIST dataset, averaged across all 10 digit classes. 39

5.1 An example of the Swiss Roll dataset 41
5.2 15 samples of each digit from the MNIST dataset. 43
5.3 15 samples of each class from the CIFAR-10 dataset. 43
5.4 Noising process of the MNIST dataset before and after lifting to the frequency domain for Extended DMD 48

A.1 Example images from each class of the dynamics 67
A.2 Example images from each class of the correct dynamics 68
B.1 Denoising samples for the Swiss Roll Dataset. 69
B.2 Summary of the Exact DMD operator and the captured dynamics of the Swiss Roll dataset. 70

B.3 Exact DMD reconstruction of the MNIST dataset. 71
B.4 Summary of the Exact DMD operator and the captured dynamics of the MNIST dataset. 72
B.5 Extended DMD reconstruction of the MNIST dataset. 73
B.6 Summary of the Extended DMD operator and the captured dynamics of the MNIST dataset. 74
B.7 Hankel DMD reconstruction of the MNIST dataset. 75
B.8 Summary of the Hankel DMD operator and the captured dynamics of the MNIST dataset. 76
B.9 Denoising process for 10 digits, generated from Model A. 77
B.10 Denoising process for 10 digits, generated from Model B. 78
B.11 Modes of C-PAE Model A. 79
B.12 Modes of C-PAE Model B. 80
B.13 Discrete eigenvalues of the linear operator in a C-PAE. 81
B.14 Denoising process for 10 digits, generated from Model C. 82
B.15 Modes of C-PAE Model C with coefficeitns to solve for denoised sample. 83
B.16 Reconstruction of a denoised 7 using the calculated coefficients of the modes of Model C. 84
B.17 Denoising process for 10 digits, generated from Model D. 85
B.18 Modes of E-PAE Model C. 86
B.19 Modes of E-PAE Model D. 87
B.20 Discrete eigenvalues of the linear operator in an E-PAE. 88
B.21 Denoising process for 10 classes of CIFAR-10, generated from Model E. 89
B.22 Denoising process for 10 classes of CIFAR-10, generated from Model F. 90
B.23 Denoising process for 10 classes of CIFAR-10, generated from Model G. 91
B.24 Four samples generated using the DDPM U-Net model. 92
B.25 Diffusion process prediction from a DDPM U-Net model. 93

List of Attachments

- **1_Step_Regression.mp4**: Effect of parameters on single step dynamics
- **50_Step_Regression.mp4**: Effect of parameters on 50 step dynamics
- **100_Step_Regression.mp4**: Effect of parameters on 100 step dynamics

List of Symbols

Linear Algebra

$\mathbf{U}$	Left singular vectors
$\mathbf{\Sigma}$	Diagonal matrix of singular values
$\mathbf{V}$	Right singular vectors

Dynamical Systems

$\mathbf{A}$	Matrix representation of linear dynamics
$\tilde{\mathbf{A}}$	Reduced-order dynamics on POD space
$F_t : \mathcal{M} \to \mathcal{M}$	Nonlinear map depicting the dynamics of a system
$g : \mathcal{M} \to \mathbb{C}^n$	A measurement or *observable* function of a system's state
$\mathcal{K}$	Koopman Operator
$S^t(\mathbf{x}_0)$	Continuous-time state of a system after evolving t steps from $\mathbf{x}_0$
$\mathbf{X}$	Data matrix, $X \in \mathbb{R}^{n \times (t-1)}$
$\mathbf{X}'$	Shifted data matrix, $X' \in \mathbb{R}^{n \times (t-1)}$
$\mathbf{x}(t)$	Continuous-time state of a system at time t
$\mathbf{x}_t$	Discrete-time state of a system at time t
$\mathbf{z}$	Encoded state vector. $\mathbf{z} = \varphi(\mathbf{x})$
θ	Koopman Eigenfunctions
ψ	Koopman Modes

Diffusion

β	Variance of the noise added at each step of the diffusion process.
$G : \mathbb{R}^n \to \mathbb{R}^m$	Linear map representing a process, system, or degradation of a signal

$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x})$	Reverse denoising process
$q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$	Forward noising process

Autoencoders

φ	Predictive Autoencoder Encoder
φ^{-1}	Predictive Autoencoder Decoder

List of Acronyms

AIS	Annealed Importance Sampling
C-PAE	Compressive-Predictive Autoencoder
DDPM	Denoising Diffusion Probabilistic Model
DMD	Dynamic Mode Decomposition
DPM	Diffusion Probabilistic Model
E-PAE	Expansive-Predictive Autoencoder
FFT	Fast Fourier Transform
IFFT	Inverse Fast Fourier Transform
PAE	Predictive Autoencoder
LDM	Latent Diffusion Model
MCMC	Markov Chain Monte Carlo
ODE	Ordinary Differential Equation
SDE	Stochastic Differential Equation
SSIM	Structural Similarity Index Measure
SVD	Singular Value Decomposition

Chapter 1

Introduction

1.1 Motivation

Diffusion is a natural solution to linear inverse problems as it transforms one probability distribution to another. Dynamic Mode Decomposition (DMD), on the other hand, is a less obvious yet still feasible solution to many linear inverse problems that follow certain conditions (Tu et al., 2014). By looking at the linear inverse problems from two different perspectives (probabilities and dynamical systems), two distinc methods arise - with U-Net and transformer based diffusion models being purpose-built for image diffusion. This thesis explores the idea of a hybrid approach that maintains the probability transition formulation as well as a dynamical systems perspective in the form of predictive autoencoders. In this way, the nonlinear nature of the reverse diffusion process can be represented as a linear operator using Koopman Operator Theory. A linear representation of a naturally nonlinear process provides unique insight into how diffusion transforms probability distributions between a simple input distribution and a complex target distribution.

1.2 Thesis Contributions

Founded on Koopman Operator Theory, this work explores the notion that the nonlinear diffusion process can be viewed through the lens of dynamical systems theory. This work also shows that a *Koopman-invaraint subspace* can be found where the diffusion system is linear. Within this subspace, a Koopman operator can be learned using Predictive Autoencoders (PAEs) that capture the system's dynamics, a task previously done using U-Net or transformer models with large capacity.

This interpretation allows for a different perspective on the image diffusion process, connecting the diffusion process to a wider range of dynamical systems and providing a new avenue for understanding and improving the performance of diffusion models.

Additionally, this work presents a comprehensive analysis of the properties of the Koopman-invariant subspace and its implications for the stability and robustness of the diffusion process. By investigating the characteristics of this subspace, the thesis provides insights into the fundamental mechanisms that govern the behavior of diffusion models and how they change based on model design choices.

In summary, the contributions of this thesis are twofold:

1. This work modifies the existing dynamical systems perspective of diffusion models, particularly Denoising Diffusion Probabilistic Models (DDPMs), by integrating Koopman Operator Theory. While traditional approaches often view these models through the lens of Stochastic Differential Equations (SDEs) and probability flow Ordinary Differential Equations (ODEs), this thesis leverages Koopman Operator Theory to provide a linearized perspective on the inherently nonlinear reverse diffusion process. This integration offers a refined analytical lens to examine how diffusion transforms data distributions.

2. By introducing a method for analyzing the dynamics of diffusion models through the construction of a Koopman-invariant subspace, this approach provides a new framework that facilitates a detailed examination of the linear dynamics within

the complex behavior of diffusion models. It enhances our ability to understand and dissect the internal workings of these models, leading to potential insights into their stability and efficiency.

1.3 Thesis Outline

The remainder of this thesis is organized as follows:

Chapter 2 gives an overview of Koopman Operator theory. Koopman Operator Theory is described starting from the foundations of dynamical systems and the properties that make this theory useful. This chapter finishes with two methods used to approximate the Koopman operator, both of which are used as an approach to capture the dynamics of the diffusion process.

Chapter 3 discusses the diffusion process, including linear inverse problems. Among linear inverse problems, the chapter introduces various methods to analyze such problems, most notably with a Bayesian approach. A more specific solution involves using Diffusion Probabilistic Models and Denoising Diffusion Probabilistic Models.

Chapter 4 presents the notion that the nonlinear diffusion process described in Ch. 3 can be captured using the nonlinear systems theory from in Ch. 2. This chapter lays the theoretical foundation for how such a process can be modeled using Koopman Operator Theory. It introduces the datasets used for experimentation and outlines the methodology for applying Dynamic Mode Decomposition and Predictive Autoencoders to the diffusion process.

Chapter 5 provides the results and a discussion of the experiments conducted. It presents the findings from applying Dynamic Mode Decomposition and Predictive Autoencoders to the diffusion process and analyzes the implications of these results for understanding and improving the performance of diffusion models.

Chapter 6 concludes the thesis by summarizing the key findings and contributions of this work. It also suggests directions for future research in the area of dynamical systems theory and its application to diffusion models in machine learning.

Chapter 2

Koopman Operator Theory

2.1 Dynamical Systems

A conventional, continuous-time dynamical system is described by

$$\frac{d}{dt}\mathbf{x}(t) = f\left(\mathbf{x}(t)\right), \tag{2.1}$$

where $\mathbf{x}(t) \in \mathcal{M}$ represents the state of the system and $f(\cdot)$ is a (possibly) nonlinear vector-valued smooth function. The nonlinear dynamics may also be parameterized by the current time t, but this work considers the more simple autonomous (time-invariant) class of dynamical systems. The state space $\mathcal{M}$ includes all potential states of the system, with each point representing a unique state. The system's trajectory in this space is a continuous curve over time.

In contrast, a discrete-time dynamical system is given by

$$\mathbf{x}_{k+1} = F_t(\mathbf{x}_k), \tag{2.2}$$

where $\mathbf{x} \in \mathcal{M}$ denotes the state of the system, and $F_t : \mathcal{M} \to \mathcal{M}$ is a map depicting the dynamics over the time interval t. The flow map from Eq. 2.2 is derived by

integrating the continuous-time system from Eq. 2.1 over a given time interval t:

$$F_t\left(\mathbf{x}_{t_0}\right) = \mathbf{x}_{t_0+t} = \mathbf{x}(t_0) + \int_{t_0}^{t_0+t} f\left(\mathbf{x}(\tau)\right) d\tau. \tag{2.3}$$

Dynamical systems can be linear or nonlinear. Nonlinear systems, where $f(\cdot)$ is a nonlinear function, exhibit complex behaviors such as chaos, which are not present in linear systems characterized by linear functions of the state. The stability of these systems, or how they respond to minor disturbances from equilibrium states, is a fundamental area of study. Stability analysis often employs Lyapunov's methods to determine the system's response to perturbations (Strogatz, 2015, Ch. 1).

Given the complexity of nonlinear dynamical systems, analytical solutions are rare. Therefore, numerical methods, particularly Runge-Kutta methods, are commonly used in engineering and scientific applications for solving these systems. These methods approximate the system's behavior over time, allowing for analysis and control of systems where exact solutions are not feasible (Strogatz, 2015, Ch. 2).

Furthermore, dynamical systems theory reveals that small parameter changes can lead to significant changes in system behavior, known as bifurcation. Chaos theory studies unpredictable behaviors in deterministic systems, highlighting the complexity inherent in nonlinear dynamics (Strogatz, 2015, Ch. 3).

System Identification

Following the understanding of a system's dynamics, system identification becomes crucial. System identification is the process of building mathematical models of dynamical systems from observed data. This is especially important in real-world applications where the exact dynamics are unknown or too complex to model from first principles.

The following are some common techniques:

- **Least Squares:** Estimates parameters of a model by minimizing the sum of the square of the residuals (the differences between observed values and model

predictions). It works well for linear models, producing a best-fit line in a least-squares sense. However, its performance degrades with non-Gaussian noise or outliers, as it emphasizes more significant errors due to squaring the residuals (Björck, 1996).

- **Maximum Likelihood Estimation:** Finds the parameter values that maximize the likelihood of making the observations given the model. MLE is powerful for many models, including non-Gaussian noise distributions. Its limitation is computational complexity and sensitivity to the initial guess, especially in nonlinear models (Myung, 2003).

- **ARMA:** Models a time series using a combination of Autoregressive (AR) and Moving Average (MA) components. ARMA efficiently captures a wide range of time series patterns but requires the time series to be stationary. Determining the correct order of the AR and MA components can be challenging, and the method struggles with non-stationary data (Box et al., 2015, Ch. 4).

- **SINDy:** Discovers the governing equations of dynamical systems from data. SINDy effectively identifies sparse, interpretable models for nonlinear dynamics but relies heavily on the choice of candidate functions in the library. It requires enough clean data to capture the system's dynamics and may fail if the correct dynamics are not represented in the function library (Brunton et al., 2016b).

Linearization

In control theory, linearization and system identification play a vital role. It allows for designing controllers for systems whose exact dynamics are either unknown or too complex to use directly. Controllers are then designed using the linearized model to achieve the desired performance and stability characteristics.

Some of the common linearization techniques and their drawbacks are listed below:

- **Taylor Series Expansion:** Performs linearization around a point, often a fixed or an equilibrium point. This involves expanding the function in terms of its derivatives at that point and keeping only the first-order (linear) terms. Higher-order terms are discarded as they contribute to the nonlinearity. Because of this loss of information, accuracy drops significantly away from the expansion point (Khalil, 2013).

- **Jacobian Linearization:** Simplifies a nonlinear system to a linear one around a specific point by using the first-order partial derivatives of the system's dynamics. This, however, is limited to a system's points of equilibrium (Astrom and Murray, 2008).

- **Perturbation Methods:** Treats the nonlinear terms as a small perturbation to a linear system and analyzes the effect of these perturbations on the system's behavior. Perturbation methods are limited to systems where the nonlinearity is small (Hinch, 1991).

- **Piecewise Linearization:** In some cases, the system can be linearized in different regions separately, and the overall behavior can be approximated as a piecewise linear system. This method is useful when the system exhibits various types of behavior in different regions of the state space. On the other hand, this method may lead to discontinuities between regions and doesn't capture global dynamics (Astolfi and Marconi, 2010).

Real-world applications of these theories are vast, including physics, engineering, biology, economics, and social sciences.

2.2 The Koopman Operator

In contrast, Koopman theory aims to express a nonlinear dynamical system as a linear rule by employing an observable function. This allows one to achieve more accurate predictions of how a system advances in time and gain insight into the system itself.

Let $S^t(\mathbf{x}_0)$ denote the position at time t along the trajectory described by Eq. 2.1 when starting at state $\mathbf{x}_0$; this is sometimes called the *flow* in classical dynamical systems theory. Note that the set $\mathcal{M}$ can be an arbitrary set with no structure and thus requires a measurement (often called *observable*) function $g : \mathcal{M} \to \mathbb{C}^n$ to study the evolution of $\mathbf{x}_t$ (Budišić et al., 2012). However, in the context of pure Koopman theory, n is conceptually infinite as the function space of all possible observable functions is not explicitly limited (Koopman, 1931).

For the continuous time system, the Koopman operator $\mathcal{K}$ is defined to act on $\mathbb{C}^n$ from $g(\mathbf{x})$ by

$$\mathcal{K}^t g(\mathbf{x}_0) = g(S^t(\mathbf{x}_0)). \tag{2.4}$$

For a discrete-time system, this can be simplified to

$$\mathcal{K} g(\mathbf{x}) = g(F(\mathbf{x})). \tag{2.5}$$

Notably, $\mathcal{K}^t$ and $\mathcal{K}$ are linear operators despite acting on nonlinear dynamics. The Koopman operator evolves observables from g rather than states and acts on the space of observables directly (Arbabi and Mezic, 2017).

In general, this implies that every nonlinear function has a corresponding set of observable functions that can be used to generate a linear representation. A pedagogical example to demonstrate this can be seen by observing the dynamical system defined by the following ODE:

$$\begin{aligned} \tfrac{d}{dt} x_1 &= \mu x_1 \\ \tfrac{d}{dt} x_2 &= \lambda \left(x_2 - x_1^2 \right) \end{aligned} \tag{2.6}$$

The phase portrait of the system defined in Eq. 2.6 is shown in Figure 2.1. The state can be augmented with a single observable function $y_3 = x_1^2$ to obtain a linear representation of this dynamical system. With this simple ODE, this linearization technique is closely related to Carleman linearization (Carleman, 1932), which allows for an analytical solution of the form

$$\left.\begin{aligned} \frac{d}{dt}x_1 &= \mu x_1 \\ \frac{d}{dt}x_2 &= \lambda(x_2 - x_1^2) \end{aligned}\right\} \quad \Rightarrow \quad \frac{d}{dt}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \underbrace{\begin{bmatrix} \mu & 0 & 0 \\ 0 & \lambda & -\lambda \\ 0 & 0 & 2\mu \end{bmatrix}}_{\mathcal{K}} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \tag{2.7}$$

when using the lifted coordinate system

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \end{bmatrix}. \tag{2.8}$$

In this lifted subspace, the system can be evolved using the linear dynamics defined in Eq. 2.7. Trajectories from the original ODE and the linearized system dynamics are shown in Figure 2.2, where the trajectories match exactly. This is a classic example of a *Koopman-invariant subspace*, discussed later in section 2.3.

This analytical method has its own flaws and, in some cases, may not be possible. For example, this linearization method fails if there is more than one fixed point or general attractor because the observables include the original state. Intuitively, the original state cannot be included as observables in systems with more than one fixed point because linear systems (with one attractor at the origin) are not topologically conjugate to nonlinear systems with more than one attractor (Brunton et al., 2016a).

Finally, analytical linearization methods require knowledge of the system's guiding dynamics, which may not always be available. Instead, Koopman Operator

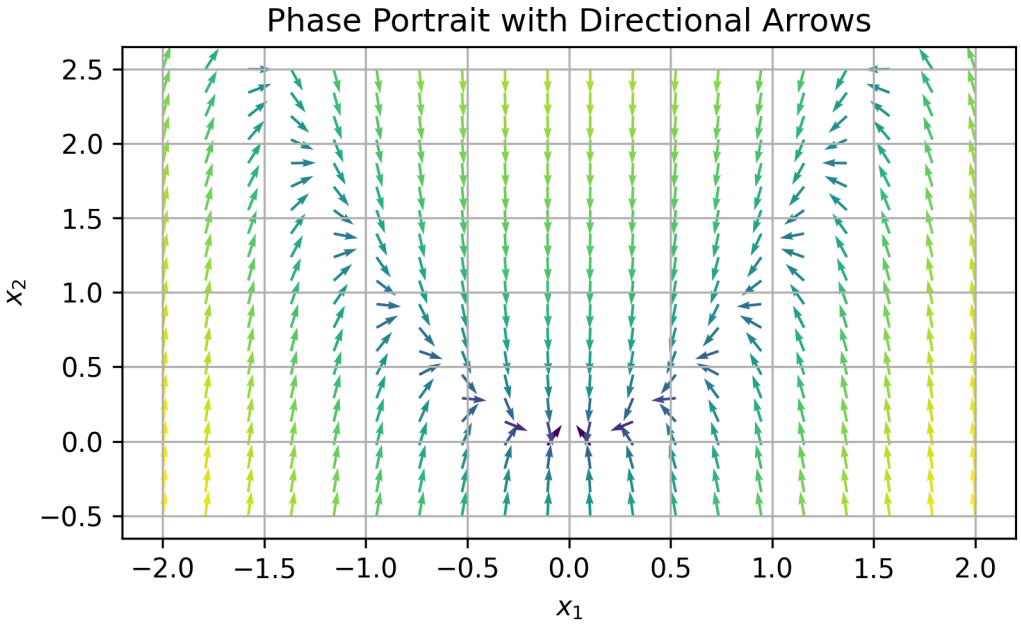


Figure 2.1: The behavior of the system described in Eq. 2.6 with parameters $\mu = -0.25$ and $\lambda = -1$. The arrows' direction and color indicate the direction and magnitude of the system's state changes at various points in the x_1-x_2 plane, respectively. Figure can be regenerated with Code C.1. [back]

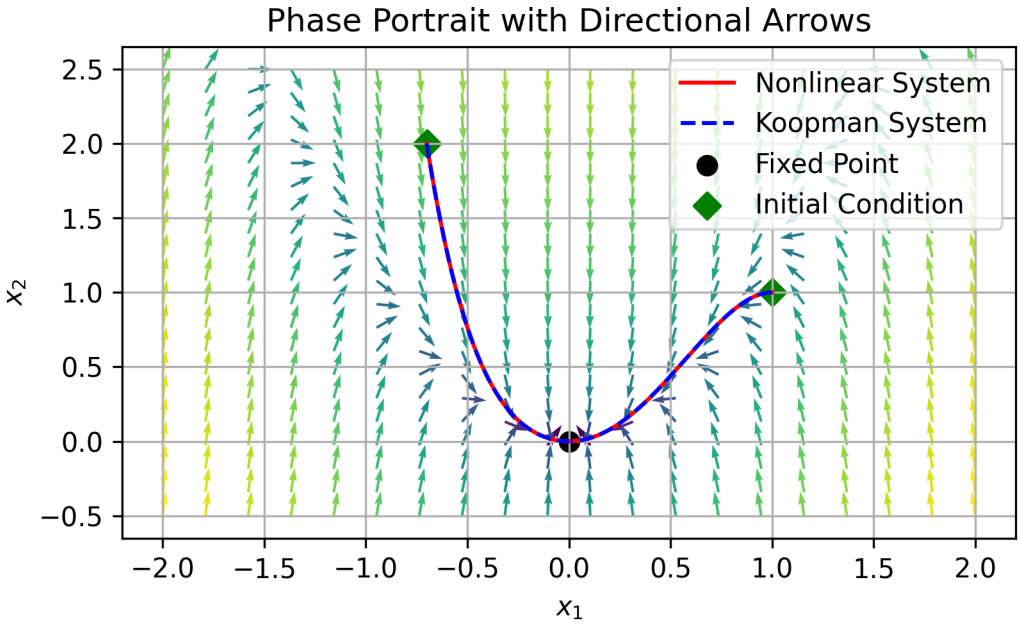


Figure 2.2: The phase portrait of the system defined by the ODE in Eq. 2.6 with parameters $\mu = -0.25$ and $\lambda = -1$. The red trajectories result from the ODE's dynamics; the blue trajectories show the dynamics of the linearized system from Eq. 2.7. Trajectories can be generated with Code C.2. [back]

Theory provides a data-driven approach to discovering a linear representation of fundamentally nonlinear systems.

2.3 Koopman Invariant Subspaces

A Koopman-Invariant Subspace is particularly defined as a subspace that is invariant under the action of the Koopman operator. This means that if a function belongs to this subspace, its image under the Koopman operator will also belong to the same subspace.

For the system defined in Eq. 2.7, consider a subspace spanned by a set of observable functions, for instance, span $\{x_1, x_2, x_1^2\}$. This subspace is deemed Koopman-Invariant if, for every function in the subspace, the Koopman operator maps it to another function within the same subspace. This implies that the system's dynamics can be fully captured and described within this subspace. In the context of this specific system, this is exemplified by the relationship $y_3 = x_1^2$, which indicates that the observable y_3 is a function of the state variable x_1 and is closed under the action of the Koopman operator.

The concept of Koopman-Invariant Subspaces is significant in analyzing dynamical systems as it allows for a linear representation of nonlinear dynamics in a finite-dimensional subspace. By identifying such subspaces, one can apply tools from linear system theory to analyze and control nonlinear systems. This approach is particularly useful for complex systems where traditional methods might be infeasible or inefficient (Brunton et al., 2016a).

2.4 Approximation Methods

Since there is no principled way to calculate the Koopman operator and identify a Koopman-invariant subspace for any given nonlinear system, approximations are typically used. It should be noted that these approximation methods are different

from typical linearization methods in that they find a globally linear representation of the nonlinear system instead of linearizing around some fixed point.

2.4.1 Dynamic Mode Decomposition

DMD is a data-driven method for approximating and analyzing the Koopman operator and is particularly useful in complex dynamical systems. It provides a practical approach to decompose the system into a set of modes that evolve linearly over time (Schmid, 2010).

The essence of DMD lies in its ability to dissect a high-dimensional system into simpler, interpretable components. Given a sequence of data snapshots $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$ from a dynamical system, DMD aims to find the best linear operator $\mathbf{A}$, such that

$$\mathbf{x}_{t+1} \approx \mathbf{A}\mathbf{x}_t \quad \text{for } t = 1, \ldots, m-1. \tag{2.9}$$

The operator $\mathbf{A}$ captures the system's dynamics as closely as possible in a linear framework.

Algorithm Overview

The standard DMD algorithm can be outlined in several steps:

1. Arrange the data snapshots into two matrices

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{m-1}] \text{ and } \mathbf{X}' = [\mathbf{x}_2, \mathbf{x}_3, \ldots, \mathbf{x}_m]. \tag{2.10}$$

2. Compute the rank-truncated Singular Value Decomposition (SVD) of $\mathbf{X}$, such that

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*. \tag{2.11}$$

3. Define a reduced-order representation of $\mathbf{A}$, denoted as

$$\tilde{\mathbf{A}} = \mathbf{U}^* \mathbf{X}' \mathbf{V} \boldsymbol{\Sigma}^{-1}. \tag{2.12}$$

4. Compute the eigenvalues and eigenvectors of $\tilde{\mathbf{A}}$ to obtain the DMD modes and eigenvalues.

5. Solve for the amplitudes that result in a best fit to the data snapshot at any given time step.

These steps yield a set of dynamic modes, each associated with a specific eigenvalue, providing insight into the dominant dynamics of the system (Tu et al., 2014).

Connections to Koopman Operator Theory

The connections to Koopman Operator Theory were originally discussed by Rowley et al. (2009). These connections can be summarized as follows: Consider a discrete-time dynamical system

$$\mathbf{x}_{k+1} = F_t(\mathbf{x}_k) \tag{2.13}$$

where $\mathbf{x} \in \mathcal{M}$. As previously discussed, the Koopman Operator $\mathcal{K}$ acts on the space of observable functions $g : \mathcal{M} \to \mathbb{C}^n$ where

$$\mathcal{K} g(\mathbf{x}_k) = g(F(\mathbf{x}_k)) = g(\mathbf{x}_{k+1}). \tag{2.14}$$

Since $\mathcal{K}$ is linear, despite the possible nonlinear nature of F, $\mathcal{K}$ can be written as its eigendecomposition

$$\mathcal{K}\theta_j(x) = \lambda_j \theta_j(x), \qquad j = 1, 2, \ldots, \tag{2.15}$$

where $\theta_j(\cdot)$ are eigenfunctions and λ_j are the respective eigenvalues. The vector-valued observable function g can then be expressed in terms of these eigenfunctions

of $\mathcal{K}$

$$g(\mathbf{x}) = \sum_{j=1}^{\infty} \theta_j(\mathbf{x})\psi_j \tag{2.16}$$

where $\psi_j \in \mathbb{C}^n$. The vectors ψ_j are typically referred to as the *Koopman Modes* of the map F and θ_j as the *Koopman Eigenfunctions*.

By applying Eq. 2.16 to Eq. 2.14, $\mathbf{x}_k$ can be written in terms of these Koopman eigenfunctions

$$g(\mathbf{x}_k) = \sum_{j=1}^{\infty} \lambda_j^k \theta_j(\mathbf{x}_0)\psi_j. \tag{2.17}$$

Eq. 2.17 shows that each mode ψ_j has a frequency governed by λ_j, which entirely governs the evolution of the system from the initial state $\mathbf{x}_0$ (Rowley et al., 2009).

Now, suppose the standard SVD-DMD technique where a sequence of data snapshots are captured and stacked to compute a matrix $\mathbf{A}$ is used. If the DMD eigenvalues are distinct (meaning none share the same growth/decay rates) and the snapshots are linearly independent, then the snapshots can be reconstructed with

$$\begin{aligned} \mathbf{x}_k &= \sum_{i=0}^{m-1} \tilde{\lambda}_i^k \tilde{\psi}_i, && k = 0, \ldots, m-1 \\ \mathbf{x}_m &= \sum_{i=0}^{m-1} \tilde{\lambda}_i^m \tilde{\psi}_i + \mathbf{r}, && \mathbf{r} \perp \text{span}\{\mathbf{x}_0, \ldots, \mathbf{x}_{m-1}\} \end{aligned} \tag{2.18}$$

where $\tilde{\lambda}$ and $\tilde{\psi}$ are the empirical Ritz values and vectors respectively. It is important to note that the residual $\mathbf{r}$ is only present at the last snapshot since DMD minimizes the reconstruction of $\mathbf{x}_m$ from $\{\mathbf{x}_k\}_{k=0}^{m-1}$ (Chen et al., 2012).

Eq. 2.18 is noticeably similar to Eq. 2.17, replacing an infinite sum with a finite one. Further, when there is no residual ($\mathbf{r} = 0$), the DMD modes cannot be distinguished from true Koopman modes. When $\mathbf{r} \neq 0$, then there is some error, but this is the best that can be done to approximate $m + 1$ data vectors with m modes. This is why DMD is accepted as a method for computing approximate Koopman eigenvalues and eigenmodes (Rowley et al., 2009; Tu et al., 2014).

Applications

DMD has found extensive applications in fluid dynamics, data-driven control, and even financial modeling. It excels in scenarios where the system dynamics are complex and high-dimensional yet exhibit low-rank behavior.

DMD can be applied to the system in Eq. 2.6 by simply analyzing trajectories induced by the system dynamics, shown in Figure 2.3. The system's dynamics can be accurately reconstructed for the simple trajectory that does not begin on the attractor, but generalizes to an average for trajectory that does.

Limitations

However, there are limitations to DMD. Its linear approximation may not adequately capture highly nonlinear dynamics, and its performance relies heavily on the choice and quality of the data snapshots. Recent advancements in DMD have aimed to address these limitations, incorporating robustness to noise and extending the methodology to more general classes of dynamical systems (Kutz et al., 2016).

Several variants have been proposed to address the limitations of DMD. These include Exact DMD, Extended DMD, and Hankel DMD. Extended DMD and Exact DMD are extensions of DMD that incorporate nonlinear observables. In contrast, Hankel DMD is a variant that utilizes Hankel matrices to capture the complex time-dynamics of a system (Williams et al., 2015; Tu et al., 2014; Arbabi and Mezic, 2017; Chen et al., 2012).

2.4.2 Predictive Autoencoders

Another approach to approximating the Koopman operator comes from learning both the observable function and linear operator with gradient descent. Inspired by the traditional Deep Autoencoder, a PAE consists of three main components: an encoder φ, a decoder φ^{-1}, and the linear operator $\mathbf{A}$ (Lusch et al., 2018). The PAE is trained with three loss functions: reconstruction loss, prediction loss, and linearity loss.

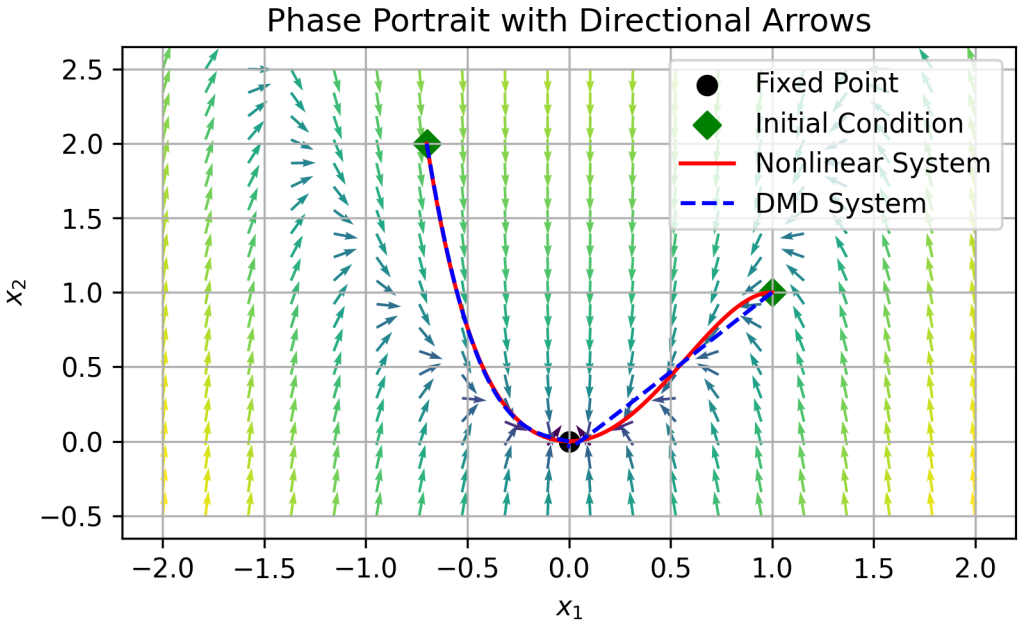


Figure 2.3: The phase portrait of the system defined by the ODE in Eq. 2.6 with parameters $\mu = -0.25$ and $\lambda = -1$. The red trajectories result from the ODE's dynamics; the blue trajectories show the system reconstruction from DMD. Trajectories can be generated with Code C.3. [back]

The reconstruction loss, defined by

$$\mathcal{L}_{\text{recon}} = \|\mathbf{x}_k - \varphi^{-1}\left(\varphi(\mathbf{x}_k)\right)\|_2, \tag{2.19}$$

identifies the intrinsic coordinates where the dynamics evolve linearly, along with an inverse $\mathbf{x} = \varphi^{-1}(\mathbf{z})$ that allows for recovery of the original state. Prediction loss learns the system's linear dynamics in the encoder's latent space (while also shaping the latent space itself). This loss function manifests in two main forms: a single-step equation that minimizes the one-step error

$$\mathcal{L}_{\text{pred}} = \|\mathbf{x}_{k+1} - \varphi^{-1}\left(\mathbf{A}\varphi(\mathbf{x}_k)\right)\|_2, \tag{2.20}$$

or a more generalized version that minimizes the error over m steps of the system

$$\mathcal{L}_{\text{pred}} = \|\mathbf{x}_{k+m} - \varphi^{-1}\left(\mathbf{A}^m\varphi(\mathbf{x}_k)\right)\|_2. \tag{2.21}$$

Finally, a linearity loss is often used to enforce linearity in the latent space (and to disentangle the linearity constraint from the decoder training objective). The linearity loss function, from Azencot et al. (2020), is defined as

$$\mathcal{L}_{\text{lin}} = \|\varphi(\mathbf{x}_{k+m}) - \mathbf{A}^m\varphi(\mathbf{x}_k)\|_2 \tag{2.22}$$

where m is commonly 1.

For systems with more complex, nonlinear dynamics, the latent dimension d may be larger than the state space n to capture such a system's dynamics adequately. This transformation - called an *immersion* - is prone to overfitting and is commonly trained with L1 regularization to elicit both a sparse operator and encoder (Alford-Lago et al., 2022).

Chapter 3

Diffusion Modeling

3.1 Linear Inverse Problems

Inverse problems constitute a pivotal research area across various scientific disciplines. These problems are characterized by the endeavor to infer causative factors from observed outcomes. This methodology inversely aligns with forward problem-solving, where predictions are made based on known processes. The crux of inverse problems lies in the deconvolution of observed data to reveal underlying parameters or causative elements, thereby inverting the conventional predictive model.

A simple example could involve the inspection of how a system or process operates. The **forward problem** may ask, "If I know how the system is behaving and what its starting conditions are, what will the outcome be?". Conversely, an **inverse problem** would be "I observed an outcome from this system; what can I figure out about the underlying system and initial conditions that lead to this result?".

Resolving inverse problems presents many challenges, primarily due to their inherent ill-posed nature. This ill-posedness implies that such problems fail to meet one or more criteria of well-posedness:

- The existence of a solution
- The uniqueness of a solution

- The stability of the solution with respect to the input data

To address these challenges, the field has developed various mathematical and computational strategies, notably regularization methods. These methods introduce additional constraints or incorporate prior information into the problem formulation, thereby enhancing the stability and reliability of the derived solutions.

The exploration of inverse problems transcends the mere development of solution methodologies for specific applications; it also entails a deep investigation into the fundamental constraints that govern the inference of causality from data. This chapter delineates the theoretical foundations, practical implications, and methodological advancements associated with linear inverse problems. It underscores the intricate relationship between observational data, theoretical models, and the assumptions that underpin this relationship.

3.1.1 Definition

At the core, linear inverse problems can be formulated using linear algebra. Let $\mathbf{y} \in \mathbb{R}^m$ represent a vector of observed measurements, and $\mathbf{u} \in \mathbb{R}^n$ represent an unknown vector of the true signal or image we aim to recover. A linear inverse problem seeks to reverse the following relationship:

$$\mathbf{y} = G(\mathbf{u}). \tag{3.1}$$

This may be complicated when the equation is perturbed by noise ($\eta \in \mathbb{R}^n$ a vector representing noise or errors corrupting the observations), so the relationship that should really be considered becomes

$$\mathbf{y} = G(\mathbf{u}) + \eta. \tag{3.2}$$

Here, $G : \mathbb{R}^n \to \mathbb{R}^m$ maps $\mathbb{R}^n$ into a proper subset of itself ($n \geq m$) and models the physical process, system, or degradation that transforms the true signal $\mathbf{u}$ into the

observations $\mathbf{y}$. This map might represent blurring in imaging, convolution in signal processing, or the forward modeling processes in geophysics or medical imaging. The goal is to reconstruct or estimate $\mathbf{u}$ given $\mathbf{y}$ and knowledge of G (Miller, 2003).

Another complication arises when $n > m$, leading the system to be *underdetermined*: the number of equations is less than the number of unknowns. In both of these complications, a solution lies in viewing the problem through the lens of probabilities.

3.1.2 Bayesian Approach to Inverse Problems

The Bayesian approach provides a probabilistic framework for solving inverse problems, incorporating prior knowledge and uncertainty quantification into the solution. This approach treats the unknown parameters, the observed data, and the model as random variables, allowing for a comprehensive analysis of uncertainty in the solution.

In the Bayesian framework, the solution to an inverse problem is expressed in terms of probability distributions. Continuing from the definitions in Eq. 3.2, $\mathbf{u}$ represents the unknown parameters we aim to estimate, and $\mathbf{y}$ represents the observed data. The goal is to compute the posterior distribution $p(\mathbf{u} \mid \mathbf{y})$, the probability of $\mathbf{u}$ given $\mathbf{y}$. From Bayes' theorem, the posterior distribution is given by:

$$p(\mathbf{u} \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid \mathbf{u}) \cdot p(\mathbf{u})}{p(\mathbf{y})}, \tag{3.3}$$

where $p(\mathbf{y} \mid \mathbf{u})$ is the likelihood function, $p(\mathbf{u})$ is the prior distribution, and $p(\mathbf{y})$ is the marginal likelihood or evidence.

The *prior distribution* $p(\mathbf{u})$ encodes any prior knowledge or assumptions about the unknown parameters. For example, if the solution is known to be smooth, a prior can be chosen that penalizes roughness (like sparsity). The choice of prior can significantly impact the solution, especially when the data is sparse or noisy.

The *likelihood function* $p(\mathbf{y} \mid \mathbf{u})$ describes the probability of observing the data $\mathbf{y}$ given a set of parameters $\mathbf{u}$. It is typically derived from the forward model and

the noise characteristics of the measurement process. For example, in the presence of additive Gaussian noise, the likelihood function is often a Gaussian distribution centered around the forward model prediction.

The *posterior distribution* $p(\mathbf{u} \mid \mathbf{y})$ combines the information from the prior distribution and the likelihood function, providing a complete probabilistic description of the unknown parameters. The posterior distribution can be used to make predictions, estimate uncertainties, and guide decision-making (Dashti and Stuart, 2017).

Computing the posterior distribution directly is often infeasible, especially for high-dimensional problems. Various computational methods, such as Markov Chain Monte Carlo (MCMC) and Variational Inference, are used to approximate the posterior distribution (Hastings, 1970).

The Bayesian approach offers several advantages, including the ability to incorporate prior knowledge, quantify uncertainty, and update solutions as new data becomes available. It provides a flexible and robust framework for solving inverse problems, particularly when the data is incomplete or noisy.

In tackling inverse problems, the Bayesian approach offers a robust method, particularly when dealing with uncertainties and noisy data. This approach provides a probabilistic framework that integrates prior knowledge with observed data, enhancing the ability to make informed inferences even in the presence of uncertainty. However, traditional Bayesian techniques can encounter significant computational hurdles, especially in high-dimensional spaces or when confronted with intricate data distributions. These challenges arise primarily from the complexity of calculating posterior distributions, which often require intensive computational resources and can be prohibitive in terms of time and feasibility. This is where Diffusion Probabilistic Models (DPMs) present an intriguing alternative, harnessing stochastic processes to model the data distribution and facilitate inference.

3.2 Diffusion Probabilistic Models

DPMs are a class of models that have gained popularity for their ability to produce high-quality samples of complex data distributions. These models are based on ideas from physics, specifically the quasi-static process, and draw inspiration from Annealed Importance Sampling (AIS), which is a traditional solution in the context of inverse problems. However, unlike AIS, DPMs do not rely on variational Bayesian methods. Instead, they simulate a diffusion process that gradually converts a simple, known distribution (a Gaussian, for example) into the target data distribution and then learn to reverse this process (Sohl-Dickstein et al., 2015).

In the context of inverse problems, DPMs can be viewed as a way to learn the mapping from simple distributions to the underlying clean signals. While AIS uses a Markov chain to create a path of intermediate distributions that connect a simple, easy-to-sample distribution (in the denoising case, this is the noised samples) to the often complex target distribution, DPMs achieve this by training a neural network to denoise samples at each step of the reverse diffusion process (Neal, 1998). This effectively approximates the posterior distribution $p(\mathbf{u} \mid \mathbf{y})$.

3.2.1 Denoising Diffusion Probabilistic Models

DDPMs, a subset of DPMs, have gained specific popularity because of their generative ability, specifically in the image domain. These models are the backbone for many popular image generation models, including Imagegen and Stable Diffusion (Saharia et al., 2022; Rombach et al., 2021).

The training process for DDPMs involves two main components:

Forward Process: A Markov chain that gradually adds noise to the data, transforming the original distribution into a tractable distribution (e.g., Gaussian noise). This is sometimes written as $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$ where $\mathbf{x}_t$ is the state of the system at time t, and $q(\cdot \mid \cdot)$ is the noising function. The noising process is traditionally

represented by the following transition probability

$$q(\mathbf{x}_t \mid \mathbf{x}_{t-1}) := \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}\right) \tag{3.4}$$

where $\mathcal{N}(\cdot; \mu, \Sigma)$ is a normal distribution with mean μ and covariance matrix Σ. The β_t parameter is determined by a noise scheduler that controls the noise variance added at each step of the forward process. Four noise schedules are commonly used for DDPMs: linear, geometric, squared cosine-capped, and quadratic.

The linear noise schedule, shown in Figure 3.1a, is the most simple to implement. The variance at each time β_t follows a simple linear formula

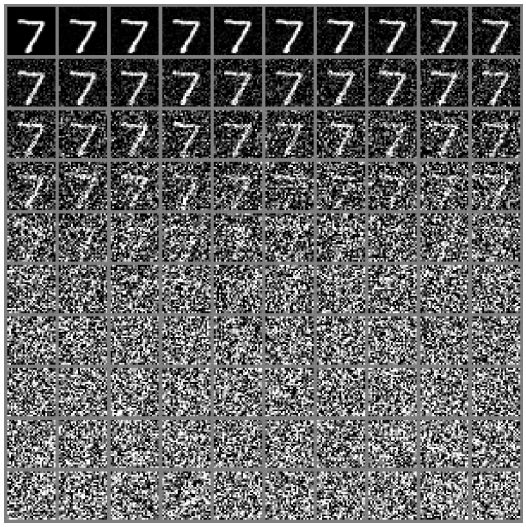
$$\beta_t = \beta_{start} + (\beta_{end} - \beta_{start}) \times \frac{t}{T-1} \tag{3.5}$$

where t is the current time step, and T is the total number of time steps. β_{start} and β_{end} are common hyperparameters that can vary by application but are typically 0.02 and 0.0001, respectively (Nichol and Dhariwal, 2021).

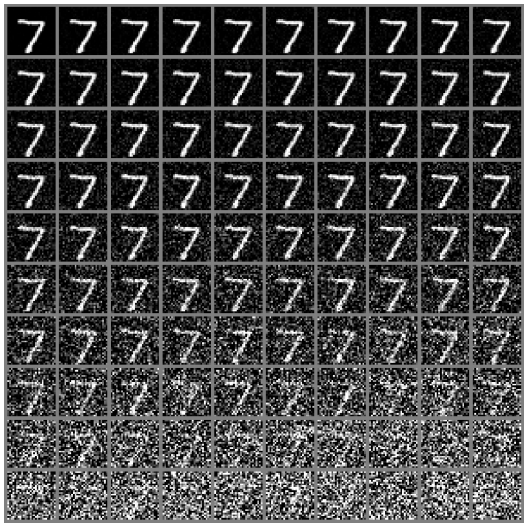
Additional noise schedules have been added to address the rate at which the original signal is destroyed in the linear noise schedule, each with its own advantages. The geometric schedule, shown in Figure 3.1b increases the noise levels geometrically, providing a balanced progression where the noise increases gradually at first and then more rapidly towards the end. The schedule increases the noise geometrically or exponentially over time, following

$$\beta_t = \beta_{start} \times \left(\frac{\beta_{end}}{\beta_{start}}\right)^{\frac{t}{T-1}}. \tag{3.6}$$

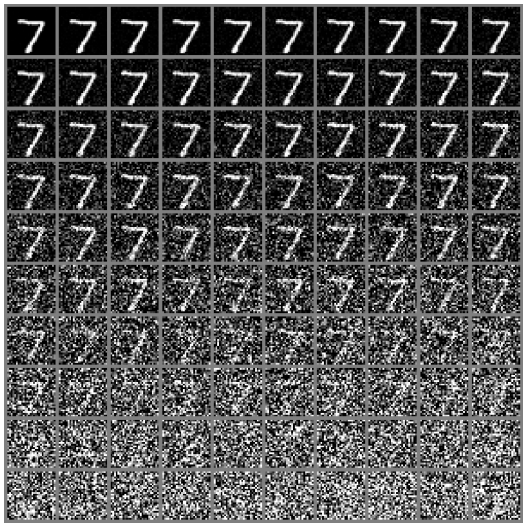
This can be beneficial when a gentle start is desired, allowing the model to initially focus on learning finer details before tackling larger noise levels.



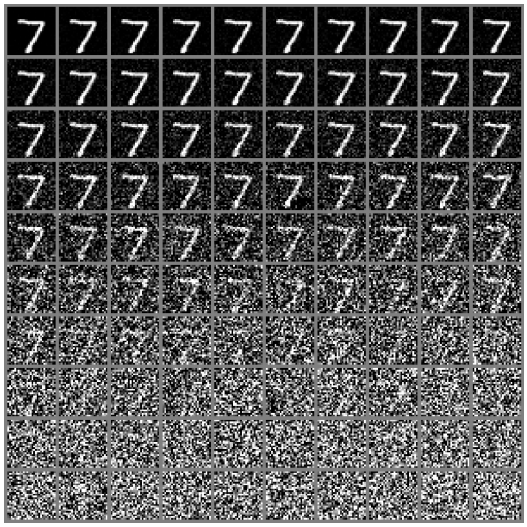
(a) Linear



(b) Geometric



(c) Squared Cosine-Capped



(d) Quadratic

Figure 3.1: Effect of different beta-schedules on the forward diffusion process for the MNIST dataset. All figures show 100 denoising steps, from step 0 in the top left to step 99 in the bottom right. [back]

The quadratic schedule (shown in Figure 3.1d, on the other hand, increases the noise levels quadratically, following

$$\beta_t = \beta_{start} + (\beta_{end} - \beta_{start}) \times \left(\frac{t}{T}\right)^2. \tag{3.7}$$

This offers a more aggressive approach where the noise ramps up more quickly early on and can be useful when the model needs to adapt to higher noise levels faster, potentially speeding up the learning process.

Lastly, the squared cosine-capped schedule, shown in Figure 3.1c, combines elements of both cosine and capping strategies. It uses a squared cosine function to ensure a smooth transition between noise levels, similar to the cosine schedule, but also caps the maximum noise level to prevent it from becoming too high (von Platen et al., 2023). The schedule is generated by following

$$\beta_t = \min\left(\beta_{\max}, \beta_{start} + (\beta_{end} - \beta_{start}) \times \left(\frac{sin(\frac{\pi t}{2T})}{sin(\frac{\pi}{2})}\right)^2\right). \tag{3.8}$$

This capping can enhance the stability of the training process and improve the quality of the generated samples by ensuring that the noise does not overwhelm the signal in the later stages of the diffusion process. Each of these schedules offers a different way to navigate the trade-off between learning efficiency, sample quality, and model stability, making them valuable tools in the design of diffusion models.

Reverse Process: The second component in the training process of DDPMs, the reverse process is a learned Markov chain that aims to reverse the forward process, denoising the data at each step to recover the original distribution. The reverse process is notated as $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ where $p(\cdot \mid \cdot)$, parameterized by θ, is the learned denoising function. The denoising - or reverse - process is represented by the following *learned* transition probability

$$p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) := \mathcal{N}\left(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \Sigma_\theta(\mathbf{x}_t, t)\right) \tag{3.9}$$

where $\mu_\theta(\mathbf{x}_t, t)$ and $\Sigma_\theta(\mathbf{x}_t, t)$ are both learned functions that predict the mean and covariance, respectively.

The reverse process is parameterized by a neural network, which is trained to minimize the difference between the true data distribution and the distribution generated by the reverse process. This process is depicted in Figure 3.2. This training objective can be formulated as a variational bound, similar to Variational Inference in the Bayesian framework.

However, it's important to note that while the reverse process of DDPMs can be seen as a form of Bayesian inference, where the model infers the clean signal from noisy observations by traversing the reverse diffusion path, the overall framework of DPMs and DDPMs is not purely based on Bayesian principles. Instead, it combines ideas from non-equilibrium thermodynamics and score-based generative modeling (Sohl-Dickstein et al., 2015) - opening the door for analysis from a dynamical systems perspective.

Once trained, a DDPM can generate clean samples by starting from noise and iteratively applying the learned denoising function (Ho et al., 2020). However, these models are often expensive if run in state space. The standard architectures for DDPM models range between U-Net (Ronneberger et al., 2015) and transformer (Vaswani et al., 2017) architectures. These models are chosen because of their large capacity and ability to generalize easily among various tasks. The downside, however, is that because of their complexity, they become expensive to run inference, especially when the inference process for a diffusion model requires the U-Net/transformer model to be evaluated multiple times. Many methods have been introduced that aim to reduce both training and inference time (by either reducing the number of diffusion steps required or reducing the number of training samples needed to represent the target distribution) (Chang et al., 2023).

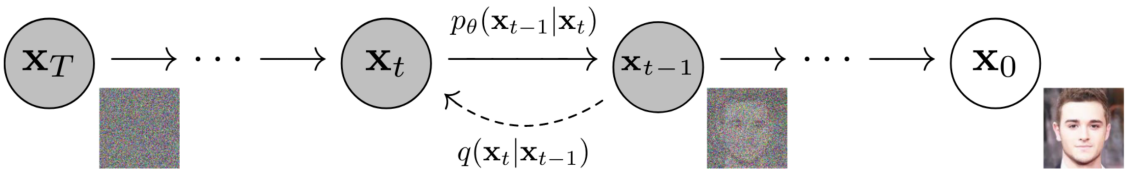


Figure 3.2: A directed graph representing the diffusion process where $p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t)$ represents the reverse process and $q(\mathbf{x}_t \mid \mathbf{x}_{t-1})$, the forward process (Ho et al., 2020). [back]

3.2.2 Latent Diffusion Models

One of the more significant improvements comes from augmenting the state space itself. Latent Diffusion Models (LDMs) are built on top of an encoder-decoder framework where the diffusion model functions in the latent space of the encoder. This alleviates several issues commonly faced by diffusion models that work in the raw image space:

1. Spending too much capacity on imperceptible details: Several methods have been proposed to prevent this (like using structural similarity instead of mean squared error and undersampling the initial denoising steps).

2. Requiring large amounts of computational resources: Because of the first issue, the training time grows beyond what is feasible for most researchers in the field.

3. Slow to inference: Evaluating large models is expensive in both time and memory, putting a lower bound on resource requirements for the end user. This comes from the inherent nature of diffusion models having to run sequentially for many steps.

By operating in the latent space, LDMs can significantly reduce the dimensionality of the data, leading to faster training and inference times. The encoder compresses the high-dimensional data (such as images) into a lower-dimensional latent representation, and the diffusion model then learns to denoise this latent representation. The decoder reconstructs the denoised data back into the original high-dimensional space.

The use of LDMs also allows for more efficient use of model capacity, as the encoder can learn to ignore irrelevant details and focus on the essential features of the data. This can lead to better generalization and more robust models. Additionally, the reduced dimensionality of the latent space means that fewer diffusion steps are needed, further speeding up the inference process.

There are also a variety of works that show both DPMs and LDMs to be solution techniques for linear inverse problems (Rout et al., 2023; Dou and Song, 2023; Chung

et al., 2023). All of which leverage the stochastic nature of the diffusion process to reverse the distribution transformation.

Overall, LDMs represent a significant advancement in the field of diffusion models, addressing some of the key limitations of traditional DDPMs and opening up new possibilities for efficient and effective generative modeling (Rombach et al., 2021).

Chapter 4

Methodology

This chapter centers on the idea that the dynamics of the nonlinear diffusion process can be captured using Koopman Operator Theory and used to predict the denoising process without complex network structures like U-Nets or transformers.

4.1 Approximation Methods

In this section, two methods are explored for approximating a Koopman operator and defining a subspace where the diffusion process is linear: Dynamic Mode Decomposition and Predictive Autoencoders.

4.1.1 Dynamic Mode Decomposition

DMD has been shown to solve linear inverse problems under certain conditions (Tu et al., 2014). The most notable condition is that a low-rank model governs the forward process. If this is not the case and the nonlinear effects are significant, then the model will result in an average of different models fit at various subsets of the original set of snapshots (Goswami and Shukla, 1991).

When using DMD, the initial test is to determine if the nonlinear effects of the diffusion process overpower the linear effects or if there is a low-rank model that can

be used to capture the underlying dynamics. Several variants of DMD are used to this end.

For each of the experiments - including those with the PAE - the squared cosine-capped noise schedule is used so that more samples with useful information are present instead of being limited to the first few samples of the noising process.

Exact Dynamic Mode Decomposition

Exact DMD is only a subtle modification of the standard DMD algorithm where, instead of generating a $\mathbf{X}$ and $\mathbf{X}'$ (where $\mathbf{X}'$ is $\mathbf{X}$ shifted by one timestep), Exact DMD uses snapshot pairs $\mathbf{X}$ and $\mathbf{Y}$. This can be viewed as a collection of snapshots from the system

$$\mathbf{y}_t = A\mathbf{x}_t \tag{4.1}$$

instead of

$$\mathbf{x}_{t+1} = A\mathbf{x}_t. \tag{4.2}$$

This format is particularly useful for diffusion as the operator must capture the dynamics of a variety of different samples. In this way, the operator will fit to the entire underlying probability distribution instead of that of a single image class. Algorithmically, Exact DMD modifies DMD by allowing for a more general relationship between the snapshot pairs, where $\mathbf{Y}$ can represent the system at a different time or under different conditions than $\mathbf{X}$. This flexibility is crucial for modeling the diffusion process, as it allows the operator to capture the transition from a noised state to a denoised state across a range of noise levels and image characteristics.

Extended Dynamic Mode Decomposition

For this experiment, the Fast Fourier Transform (FFT) is applied to each image, and Fourier basis functions are used. The Extended DMD algorithm can then be applied in this new basis.

DMD itself is a linear technique in the sense that it approximates the dynamics of a system by decomposing the data into a set of modes, each associated with a specific frequency and growth/decay rate, which are governed by a linear operator.

While DMD can be applied to data from nonlinear systems, its ability to capture the full range of nonlinear behaviors is limited. This is because DMD seeks to identify a linear relationship between successive snapshots of the system's state, which may not fully represent the underlying nonlinear dynamics.

Extended DMD aims to address this limitation by transforming the data into a higher-dimensional space where the dynamics can be more accurately approximated by linear operators, thus capturing a broader range of nonlinear behaviors. Extended DMD does this by using a dictionary of basis functions with which to augment the state and capture more important features of the system.

For this experiment, the FFT is applied to each image, and Fourier basis functions are used.

Hankel Dynamic Mode Decomposition

Hankel DMD provides another way of lifting the state vector to a higher dimension, in this case using Taken's delay embeddings. This method involves constructing a Hankel matrix, which is a structured matrix where each row is a time-delayed version of the state vector. By embedding the state vector in this way, Hankel DMD can capture the temporal dynamics of the system over multiple time steps, providing a richer representation of the system's behavior.

The Hankel matrix is then used as the input to the DMD algorithm, allowing for the extraction of dynamic modes that represent the underlying patterns in the data. This approach is particularly useful for systems with complex temporal dynamics, as it leverages the information contained in the time-delayed observations to uncover the underlying structure of the system (Arbabi and Mezic, 2017).

While this makes the inference step difficult - namely because it would require multiple consecutive time steps along the denoising process to generate a single state

vector - it serves as an indication for whether DMD fails because of complex spatial dynamics or temporal dynamics.

4.1.2 Predictive Autoencoders

In the case that a Koopman-invariant subspace cannot be discovered using SVD, even when the state vector is augmented with trivial functions of the original state (like the FFT or delay embeddings), the observable function $g(\mathbf{x})$ can be learned using gradient descent.

Two different variations of PAEs were applied to this problem: a Compressive-Predictive Autoencoder (C-PAE) structure where $dim(\mathbf{z}) < dim(\mathbf{x})$ (where $\mathbf{z}$ is the encoded state vector and $\mathbf{x}$ is the state vector itself), and an Expansive-Predictive Autoencoder (E-PAE) structure where $dim(\mathbf{z}) \geq dim(\mathbf{x})$ based on the work of Alford-Lago et al. (2022).

Compressive-Predictive Autoencoder

The PAE architecture, shown in Figure 4.1, incorporates components of a standard autoencoder that learns with a reconstruction loss and a linear operator that adds both the task of learning the dynamics within the operator and a map from state space to latent space that linearizes the system.

For the diffusion system, where many samples can contain a large amount of information (typically manifested through noise or disorder in an image), reconstructing from a lower dimensional state can lead to a lot of loss in information.

Expansive-Predictive Autoencoder

By adding a linearity constraint to the autoencoder, the model has the added requirement of learning a latent space that linearizes the system, so the latent space must contain more information than that of the original state itself (like the distance along a particular trajectory). As a result, $dim(\mathbf{z}) \geq dim(\mathbf{x})$, as shown in Figure 4.2.

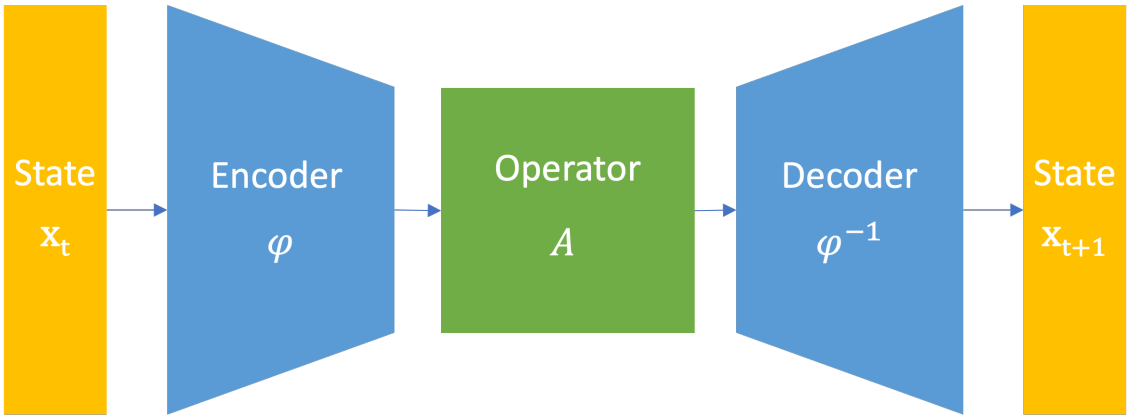


Figure 4.1: Structure of a Compressive-Predictive Autoencoder. [back]

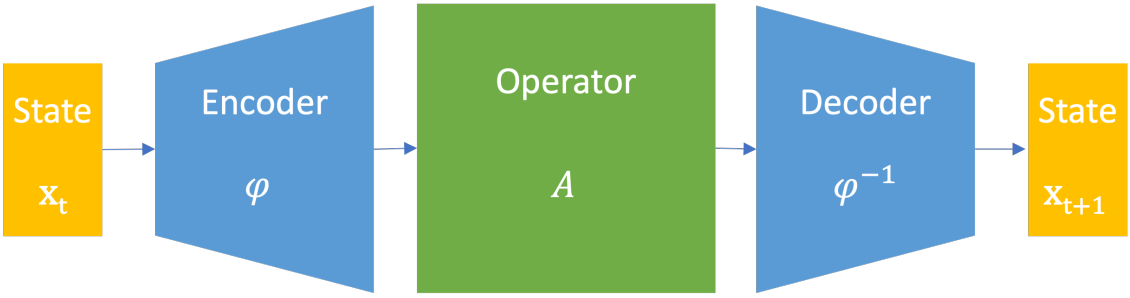


Figure 4.2: Structure of an Expansive-Predictive Autoencoder. [back]

In typical autoencoders, the encoding function aspires to be an embedding, which is globally injective, ensuring that each point in the input space maps to a unique point in the latent space. An encoder that achieves this creates a distinct and recoverable representation for each input state, which is essential for accurate reconstruction by the decoder. However, an encoder that acts as an *immersion* is characterized by its local injectivity, where each point has a neighborhood that is mapped injectively into the latent space. Despite this, it may not establish a one-to-one mapping globally across the entire input space. Lemma 1 from Alford-Lago et al. (2022) extends this concept by stating that if an immersion is paired with a corresponding *submersion*, that acts as its left inverse (a decoder that undoes the encoder's action), the composite function is the identity map on the input space. This implies that the encoder is not just locally injective but globally injective over its entire domain, qualifying it as an embedding. Under this condition, an encoder that is initially an immersion can indeed behave like a bijective embedding with respect to the subset of the latent space it maps onto, enabling a bi-directional, unambiguous mapping between the input space and its image in the latent space.

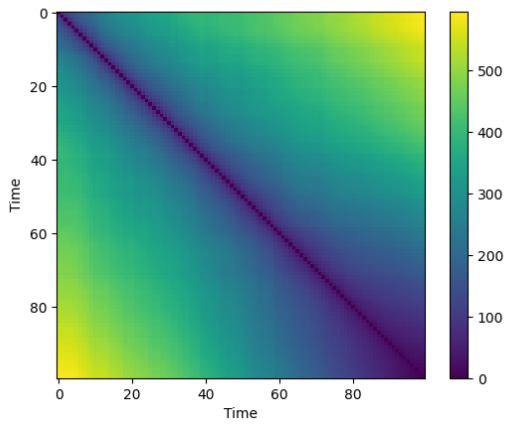
By observing pure Koopman operator theory, this may not be a true approximation of the Koopman operator as the encoder is a nonlinear function itself. However, the goal of the E-PAE is to learn a latent space that captures the dynamics of the system, which can be used to predict the future state of the system.

Model Design

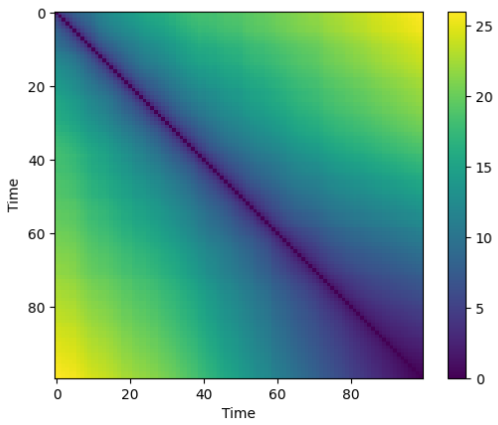
Different from typical DDPMs and DPMs, the PAE methods use the L_2 norm as a distance metric instead of the Structural Similarity Index Measure (SSIM) introduced by Wang and Bovik (2009), which serves to better distinguish between human perception rather than true error. While SSIM is certainly preferable for images with larger size (and perhaps even full RGB color), the images in the MNIST dataset are too small to characterize into the variety of distortion classes SSIM is designed to detect. The small size of the MNIST images results in the small amount of noise

that causes large L_2 values to also cause large SSIM values. Various loss functions are explored to measure the error between different states along the denoising process of various images, shown in Figure 4.3. The L_2 distance is chosen as the main distance metric for this work because 1) it approximates SSIM well, 2) it is easy to compute, and 3) it comes with a simple derivative.

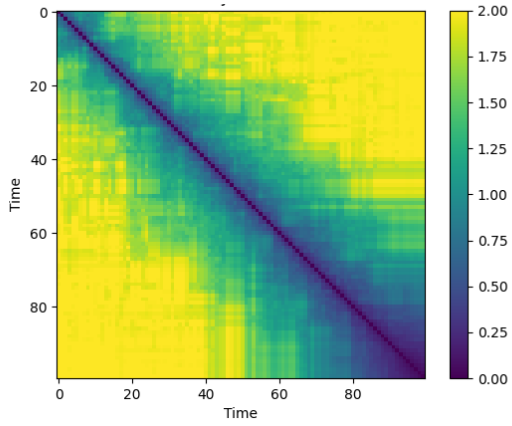
In this study, the decision to utilize 100 steps for the denoising process, as opposed to the more commonly used 1000 (Ho et al., 2020; Rombach et al., 2021), is grounded in several considerations. Firstly, the simplicity of the data being operated on, specifically MNIST, suggests that fewer denoising steps are required to effectively capture the dynamics within the operator. The intrinsic characteristics of MNIST, being a relatively straightforward dataset, do not necessitate the extensive number of steps often employed for more complex data like the ImageNet dataset used in the DDPM and LDM foundational works. Secondly, by reducing the number of steps, there is a deliberate shift in the model's focus towards short-term dynamics, which are inherently easier to learn and model than long-term dynamics in time-series models (Bengio et al., 1994; Hochreiter and Schmidhuber, 1997; Khalil, 2013, Ch. 3). Finally, fewer denoising steps allow for larger batch sizes, speeding up the training process significantly. This modification in the number of steps is strategic, aiming to optimize the learning process by concentrating on more immediate changes and interactions within the data, which are crucial for accurate denoising and reconstruction in this context. This decision is only made for the PAE models; the DMD evaluation utilizes 1,000 denoising time steps.



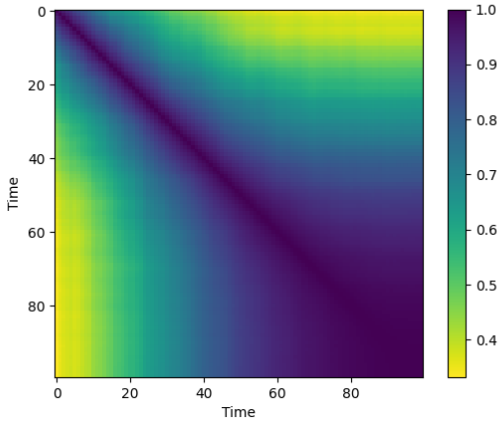
(a) L_1 Distance



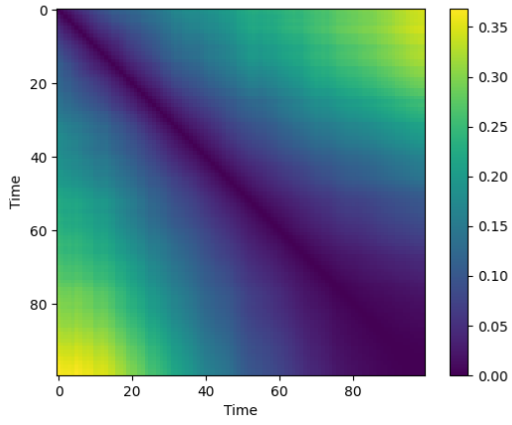
(b) L_2 Distance



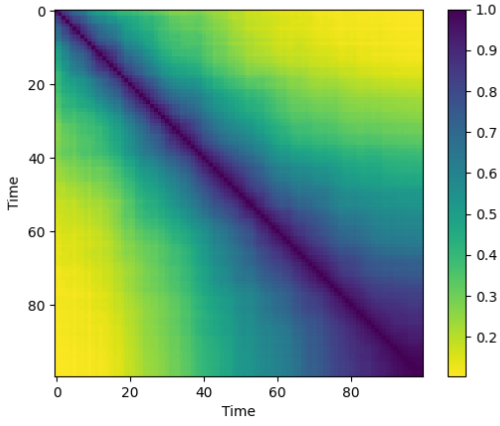
(c) L_∞ Distance



(d) Cosine Similarity*



(e) KL-Divergence



(f) SSIM*

Figure 4.3: Comparison of loss functions on the MNIST dataset, averaged across all 10 digit classes. The "time" axes are the number of diffusion steps out of a 100-step process. * indicates the colormap is reversed; for these metrics, smaller values indicate samples are less similar. [back]

Chapter 5

Results and Discussion

Chapter 5 presents an analysis and discussion of the results obtained by applying the methods described in Chapter 4. This chapter evaluates the effectiveness of the proposed approach in capturing and explaining the linear dynamics within the complex nonlinear behaviors of diffusion processes. Emphasis is placed on assessing how well the Koopman-invariant subspace facilitates a deeper understanding of the diffusion models, particularly in terms of the modes of each of the underlying operators. Finally, the results are compared to traditional methods to highlight the differences between these two approaches.

5.1 Evaluation

5.1.1 Datasets

Three datasets are used to demonstrate results from this method: the Swill Roll dataset (Marsland, 2014), the MNIST dataset (LeCun et al., 2010), and the CIFAR-10 dataset (Krizhevsky, 2009).

The Swiss Roll dataset, shown in Figure 5.1, is typically used for manifold learning and dimensionality reduction algorithms but is used here because of its structure and easily recognized pattern. The original dataset is three-dimensional, resembling a

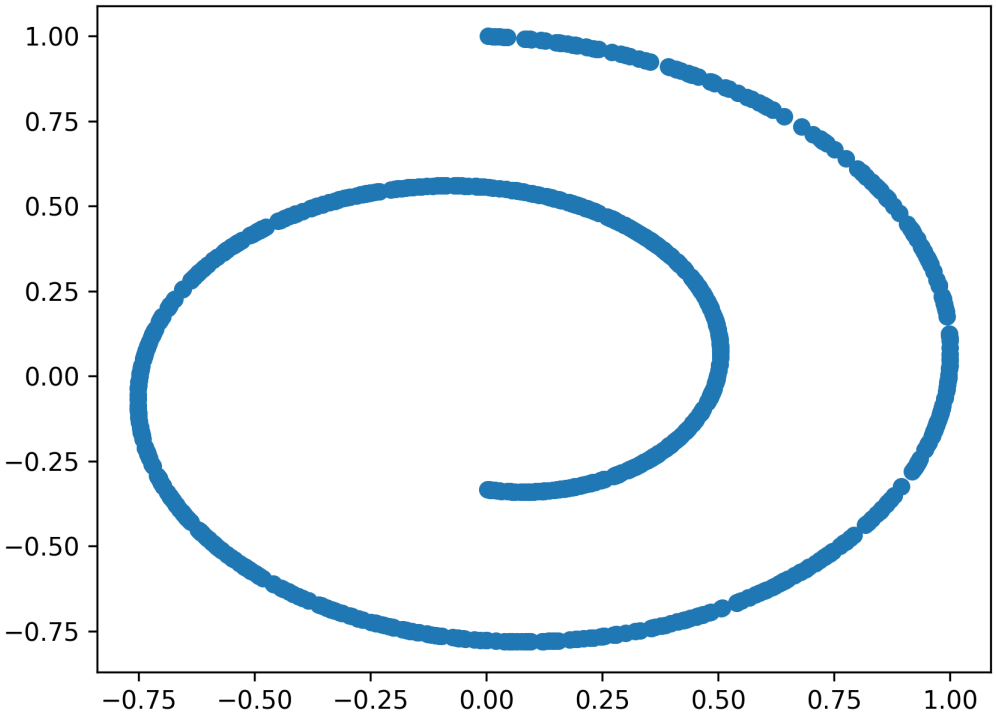


Figure 5.1: An example of the Swiss Roll dataset, generated with 1000 samples and Gaussian noise with $\sigma = 0.05$. The figure can be regenerated with Code C.4. [back]

rolled-up sheet (providing its namesake) with points distributed along a spiral curve in three-dimensional space. For this application, the points are projected onto a 2D plane orthogonal to the axis of the spiral, and the third dimension is omitted.

Experiments with the Swiss Roll dataset are conducted slightly differently. Instead of the image pixels representing the system, the state vector contains the location of each point in space as a flattened 2D vector. For the experiments in this work, 1000 points are used, so the state vector contains 2000 entries.

The MNIST dataset is a large collection of handwritten digits commonly used for training and testing in the field of machine learning. The dataset consists of 70,000 images of handwritten digits from 0 to 9. Each image is a grayscale image of size 28×28 pixels. An example of 15 images from each digit class is shown in Figure 5.2. This figure captures the intraclass variance present in the dataset, for example, the optional stroke on the "7" digit or the *serif vs. non-serif* "1".

The CIFAR-10 dataset is a well-known collection of color images used for training and testing in the field of machine learning. The dataset consists of 60,000 images divided into 10 classes, each representing a different object category. Each image is a 3-channel color image of size 32×32 pixels. An example of 15 images from each class is shown in Figure 5.3. This figure highlights the diversity within the dataset, showcasing the variations in appearance, pose, and background for each object category.

5.1.2 Characterization of Dynamics

For the PAEs, various parameter combinations are explored to determine their effect on the underlying dynamics of the reconstructed system. Because of the larger number of hyperparameters and a wider variety of discernible dynamics that result from the PAEs, it is possible to characterize the effect of each of these parameters on the dynamics and construct guidelines for generating stable PAEs for diffusive systems. The effect of these hyperparameters is studied on three different tasks:

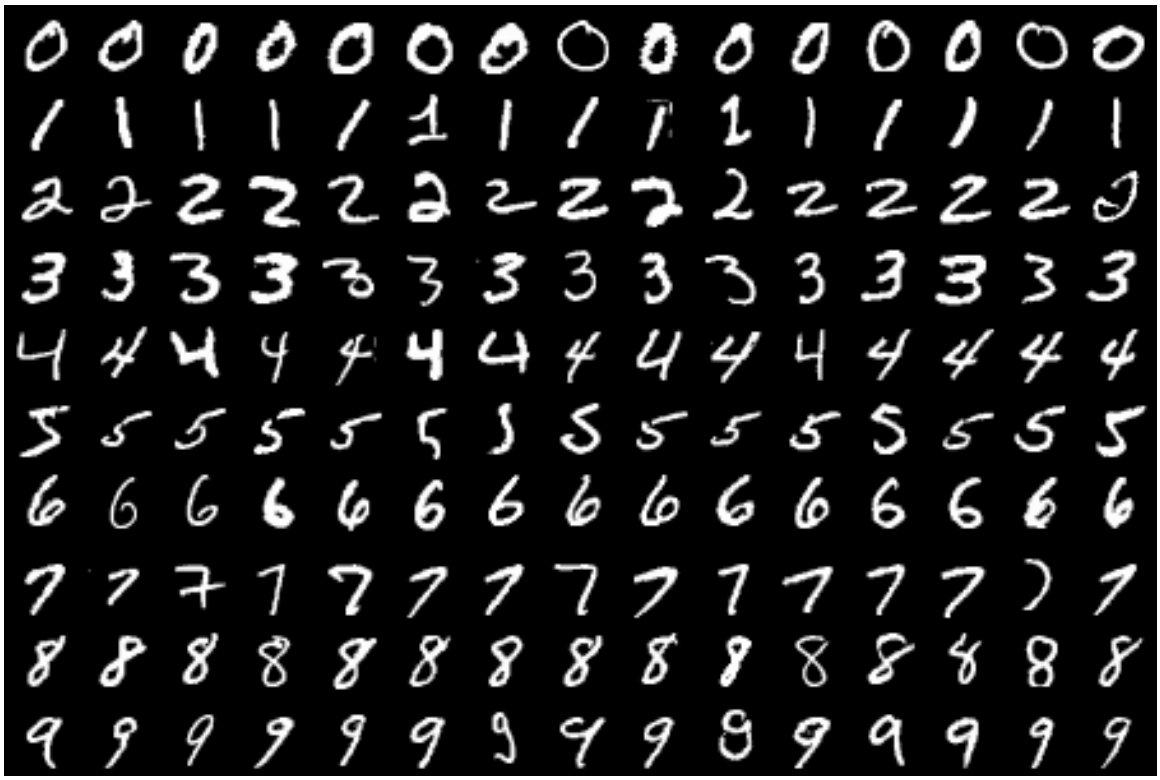


Figure 5.2: 15 samples of each digit from the MNIST dataset. [back]

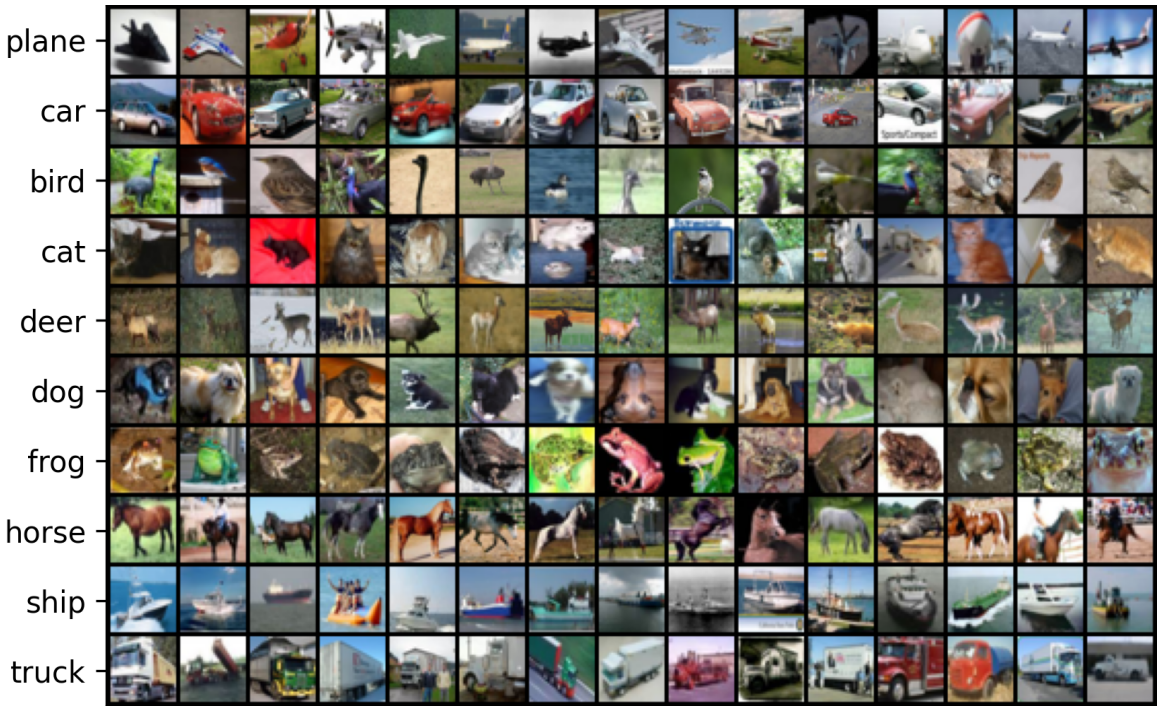


Figure 5.3: 15 samples of each class from the CIFAR-10 dataset. [back]

- **Single Step Prediction**: Given x_t, predict x_{t+1} for all $t \in [0, 99]$
- **50 Step Prediction**: Given x_{50}, predict x_{50+i} for all $i \in [0, 49]$
- **100 step Prediction**: Given x_0, predict x_{0+i} for all $i \in [0, 99]$

These tasks demonstrate both the reconstruction ability and the denoising or generative abilities for long and short-term dynamics. The dynamics that appear within these tasks are classified into the following categories:

- **Constant Pattern: No Number**: A pattern is visible and does not change throughout the denoising process but does not resemble any digit. (Figure A.1a)
- **Constant Pattern: Number**: A pattern is visible and does not change throughout the denoising process and resembles a digit. (Figure A.1b)
- **Evolving Pattern: Some Numbers**: Patterns are visible and change throughout the denoising process; some of them resemble digits, but some do not. (Figure A.1c)
- **Evolving Pattern: All Numbers**: Patterns are visible and change throughout the denoising process; all of the steps in the denoising process resemble digits (but may not be the same digit). (Figure A.1d).
- **Evolving Pattern: No Numbers**: Patterns are visible and change throughout the denoising process; none of the steps resemble numbers. (Figure A.1e)
- **Mostly Solid**: In most of the denoising steps, the samples are black, grey, or white. (Figure A.1f)
- **Noise**: The samples resemble no pattern and are mostly noise. (Figure A.1g)

For the "Single Step Prediction" and "50 Step Prediction" tasks, several additional categories are added to determine how well the system captures the original dynamics of the denoising process. The additional categories are as follows:

- **Correct: All Faint**: The input digit is the same as the reconstructed digit, and all digits are faint. (Figure A.2a)
- **Correct: All Obvious**: The input digit is the same as the reconstructed digit, and all digits are obvious and easily distinguishable from the background. (Figure A.2b)
- **Correct: Obvious to Faint**: The input digit is the same as the reconstructed digit, and digits from early in the denoising process are faint, while later steps are more obvious. (Figure A.2c)
- **Correct: Faint to Obvious**: The input digit is the same as the reconstructed digit, and digits from early in the denoising process are obvious, while later steps are more faint. (Figure A.2d)
- **Correct: Noise to Obvious**: The initial steps of the denoising process are mostly noise, while the later steps are an easily distinguishable digit. (Figure A.2e)

5.2 Dynamic Mode Decomposition

5.2.1 Exact Dynamic Mode Decomposition

The nature of the Exact DMD algorithm is to find low-rank structure within the data. This may not be the case in complex, highly nonlinear systems. When this happens, reconstruction using Exact DMD yields a result similar to fitting a linear regressor to random subsets of the original snapshots of the system (Goswami and Shukla, 1991).

Swiss Roll Dataset

The original data used to fit all DMD models on the Swiss Roll dataset is shown in Figure B.1a. For this dataset, noise is sampled from the normal distribution

$\mathcal{N}(0, 0.025)$ and added to each point in the dataset. This system is then reversed to generate the "denoising" process. Unlike standard DDPM models that use a nonlinear noise schedule (which increases the variance at a *nonlinear* rate to preserve the signal for a larger portion of the denoising process), the method used for the Swiss Roll dataset uses a linear noise scheduler where the variance increases linearly with time. This gives an advantage to the DMD methods, as using a linear noise schedule yields simpler dynamics. The system is initialized with 1000 random (x, y) points sampled from a uniform distribution on the interval $[-1, 1]$ for all reconstruction methods using the Swiss Roll dataset.

This behavior is displayed in the diffusion system of the Swiss Roll dataset, shown in Figure B.1b. By looking at the dynamics discovered by Exact DMD in Figure B.2, it becomes clear why this method produces the averaging result in Figure B.1b. Figure B.2a shows the percent variance captured by each of the modes, indicating that there is one dominant mode that captures a majority of the variance for this system. In Figure B.2b, the largest mode (in red) is shown to be unstable, close to the unit circle, and has a very small imaginary component. Together, this indicates that the dominant mode does not exhibit oscillatory behavior, but its contribution is slowly diverging and will not settle to a constant value. This is further demonstrated in Figure B.2f, which shows the time evolution of the dominant mode.

The spatial components of all modes (shown in Figures B.2c, B.2d, and B.2e), however, reveal that the model has only uncovered noise within the signal and failed to discover any discernible pattern, especially anything that resembles the original structure of the dataset.

MNIST Dataset

This experiment was also conducted on the MNIST dataset, allowing for a more interpretable analysis of the emerging spatial patterns. However, instead of using a constant noise schedule, where the variance of the normal distribution that noise is sampled from does not change with time, this experiment uses a squared-cosine noise

schedule. This allows for features of the images to be discernible for more samples of the denoising process before they are overcome with noise.

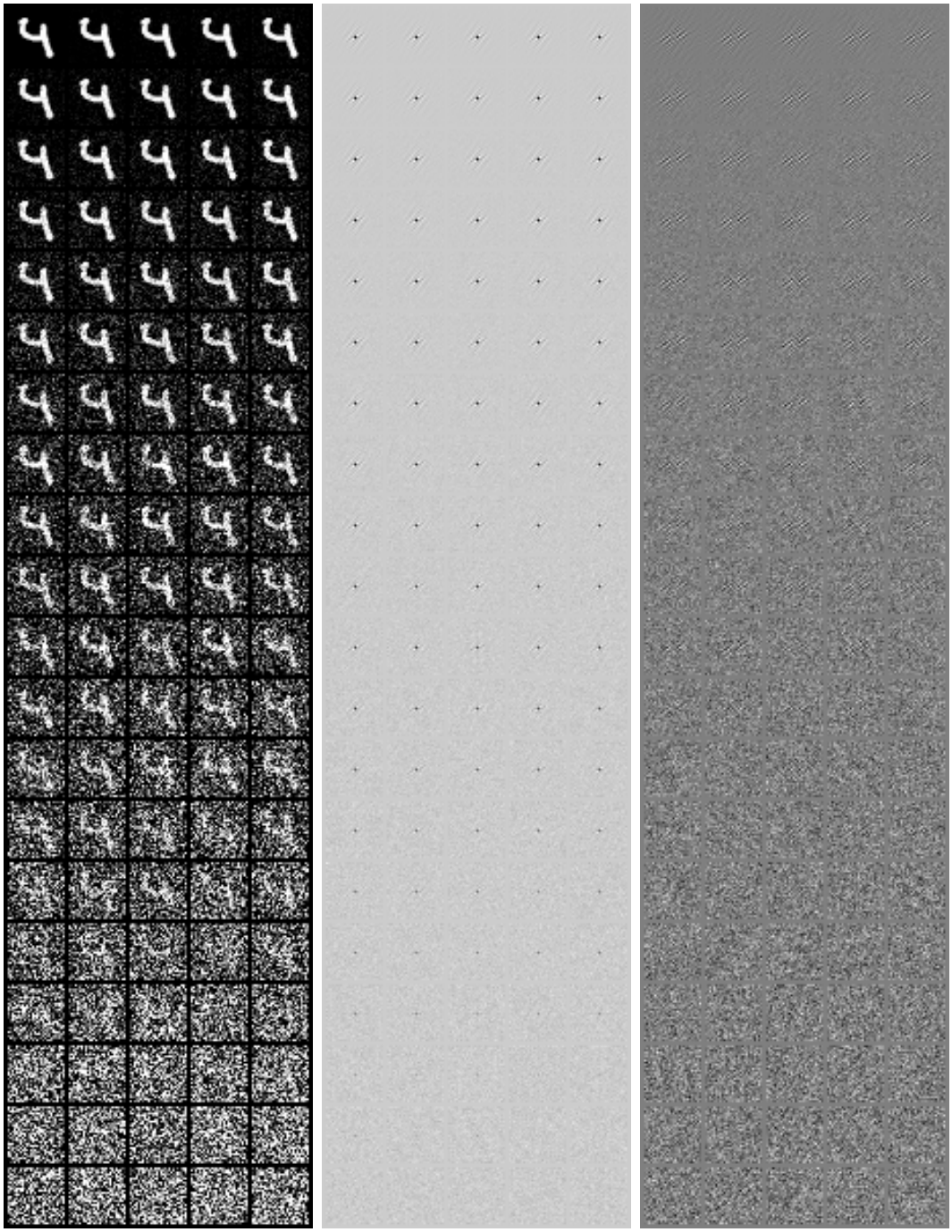
From this experiment, it is clear that the dynamics are not governed by a trivial, low-rank operator for two main reasons: 1) The calculated dynamics do not evolve over time but remain constant, and 2) The eigenvalues are close to the unit circle, resulting in false attractors.

The mode with the eigenvalue closest to the unit circle, shown in Figure B.4c, is the primary attractor visible in the reconstructed images in Figure B.3. Another observation is that these modes represent patterns common across all images in the MNIST dataset rather than patterns specific to individual image categories. Consequently, it is impossible for Exact DMD to disentangle the features contributing to a single image class. Furthermore, this means Exact DMD cannot produce a sample image of a single class without features from other classes bleeding in, more specifically the features in the modes of the operator itself (the first three are shown in Figures B.4c, B.4d, and B.4e). This limitation arises from the inherent nature of the DMD algorithm, which uses SVD to find a set of vectors that globally capture the most variance in the dataset. These vectors are not guaranteed to contain information about only a single image category.

5.2.2 Extended Dynamic Mode Decomposition

By using Extended DMD, the nonlinear dynamics that occur among different bases can be discovered. In this case, the FFT is used to lift the original state to the frequency domain - which is then used as input to the Exact DMD algorithm. Each prediction step is then transformed back to state space using the Inverse Fast Fourier Transform (IFFT).

The noising process before and after lifting that is used to fit the Exact DMD operator is shown in Figure 5.4. Note that the *noising* process is shown in Figure 5.4 while the *denoising* process is used to fit all DMD operators in this work. To achieve



(a) State space **(b)** Real component **(c)** Imaginary component

Figure 5.4: Noising process of the MNIST dataset before and after lifting to the frequency domain for Extended DMD. This system is reversed before being used as snapshots for Extended DMD. In effect, the top left image is state 1,000; the bottom right image is state 1. [back]

this, the noising process is temporally reversed. Even with the lifted system, the reconstruction (shown in Figure B.5) does not represent the true denoising process of the input image - a "4" in this case.

The modes reveal insight into why this occurs. The leading mode, shown in Figure B.6c in the spatial domain, again resembles an average of the images in the MNIST dataset. The following modes, Figures B.6d and B.6e, capture a very small amount of the system's variance, shown in Figure B.6a, and maintain no discernible pattern in their spatial structure or their dynamics, indicating that these are simply capturing random patterns present in the noise.

In comparison with Exact DMD, Extended DMD calculates modes that are closer to the unit circle, indicating slower-moving dynamics. This is further indicated by the mostly stationary dynamics of the reconstruction in Figure B.5 where the patterns in the Exact DMD reconstruction become faint by the $1,000^{\text{th}}$ step.

5.2.3 Hankel Dynamic Mode Decomposition

Contrary to expectations, the results from Hankel DMD do not significantly improve over Extended DMD. As depicted in Figure B.7, the reconstructed states using Hankel DMD are of a similar or even lower fidelity when compared to Extended DMD.

The dynamics, shown in Figures B.8f, B.8g, and B.8h, convey a similar story to each of the other DMD methods where the system changes very little over time - contrary to the true dynamics of the diffusion process.

The modes obtained from Hankel DMD, shown in Figures B.8c, B.8d, and B.8e, display similar structure to those from Exact and Extended DMD. This alone reveals insight into what DMD is fitting to. By observing the leading modes from these three DMD methods, all of which employ different methods of augmenting the state, have similar leading modes (Figures B.4c, B.6c, and B.8c). This leading mode represents the most common features among all digits in the MNIST dataset: a rounded top that is present in the 9, 8, 3, 2, 0, and some variants of the 6; and a diagonal slash

that is present in the 9, 7, 4, and 1. It is because of this - that features of many digits comprise a single mode - that DMD is not a likely candidate for this method of analysis.

Because all of the DMD variants failed to reconstruct the denoising process, it is difficult to determine if this is because of the complex spatial or temporal dynamics. This is, however, one of the limitations of DMD as this system is governed entirely by a stochastic process, and DMD is not well suited for systems with large amounts of noise.

5.3 Predictive Autoencoder

DMD's inability to adequately capture the dynamics of the denoising process indicates that a more complex set of observable functions may be required to find a Koopman-invariant subspace. Various parameters for the PAEs are explored, including the dimension of the latent space, weight of L_1 regularization, probability of dropout, activation function, and depth of the encoder/decoder.

5.3.1 Compressive-Predictive Autoencoder

The first case of PAEs is the variant where the dimension of the latent space is less than that of the state space. For these experiments, two models are used for demonstration based on their performance. A summary of the parameters of these models is provided in Table 5.1.

- **Model A** does not change digit classes as the system advances in time. Given a noised input image, the model will maintain that digit class throughout the denoising process. Figure B.9 shows the reconstruction for all digits.

- **Model B** shows evolution between digit classes as the system advances in time. This behavior does not resemble the dynamics of the diffusion system but hints at generative capabilities. Figure B.10 shows the reconstruction for all digits.

Table 5.1: Parameters used for the two selected Compressive-Predictive Autoencoder models.

Parameter	Model A	Model B
Dropout	0	0.25
Layers	8	4
L_1 weight	1×10^{-5}	1×10^{-4}
Activation	Leaky ReLU	Sigmoid
Latent Dim.	392	392
Parameters	5.9M	3.0M

The modes of Model A, shown in Figure B.11, reveal features that rarely cross the digit/class boundary - indicating that it can recreate samples of individual digit categories without features from other digits bleeding in. Model B's modes, on the other hand - shown in Figure B.12 - are less structured and share features from several digits. This explains the observed behavior characterized by the appearance of several digits over time - the modes do not independently contain features from distinct digits.

These modes, ψ, are calculated by computing the eigendecomposition of the linear operator $\mathbf{A}$ and then decoding each of the eigenvectors $\mathbf{v}$

$$\lambda \mathbf{v} = \mathbf{A}\mathbf{v} \tag{5.1}$$

$$\psi = \varphi^{-1}(\mathbf{v}). \tag{5.2}$$

The spectrum of $\mathbf{A}$ (the linear operator between the encoder and decoder), shown in Figure B.13a for Model A and Figure B.13b for Model B, indicates that the operator's eigenvalues are composed entirely of real numbers (with the imaginary component being 0 for all eigenvalues).

Model A's stability in preserving digit classes makes it suitable for tasks such as denoising and reconstruction, where retaining the original class of the input is crucial. On the other hand, Model B's ability to transition between digit classes demonstrates its potential in generative tasks, where the aim is to uncover underlying patterns or generate new samples.

5.3.2 Expansive-Predictive Autoencoder

The C-PAE can reconstruct complete digits, but this behavior does not reproduce noise as would be expected if the model were to represent the true denoising process. This section will explore the hypothesis that by adding to the latent dimension

of the encoder/decoder - not necessarily the capacity of the network itself - the reconstruction will contain the noise present in the input.

The E-PAE is not as capable of reconstructing the true digits in the MNIST dataset but captures the more complex patterns present in the denoising process, like the evolution of noise. Two models are selected to demonstrate the behavior of this variant. A summary of the parameters of these models is provided in Table 5.2. The selected models are:

- **Model C** reconstructs the true input sample with noise in the initial steps, but samples converge to a common shape at the end of the denoising process. Figure B.14 shows the reconstruction for all digits.

- **Model D** reconstructs some digits, but either results in an incorrect digit at the end of the denoising process, or the final digit is more faint than the initial. Figure B.17 shows the reconstruction for all digits.

The modes of Model C, shown in Figure B.18, are mostly comprised of noise. These patterns of noise are not random but specific combinations that are capable of reconstructing the true input sample. By calculating the coefficient for each mode that results in the true input sample, the capability of the model to reconstruct the actual sample can be determined. The coefficients are shown in Figure B.15, and the digit that results from using these coefficients of the modes is shown in Figure B.16. From this reconstruction, it is clear that the model can reconstruct plausible digits *without* noise. From the noised input experiment with all digits (Figure B.14), it is clear that the model is capable of reconstructing digits *with* noise. These observations point to an important capability of the E-PAE variant of PAEs: they learn the underlying dynamics as functions of noise instead of image features. More specifically, the patterns of a single digit class are split across many different modes, but in a way that additional modes can be added to negate or "cancel out" the features from undesirable digits.

Table 5.2: Parameters used for the two selected Expansive-Predictive Autoencoder models.

Parameter	Model C	Model D
Dropout	0	0.25
Layers	2	8
L_1 weight	1×10^{-5}	1×10^{-5}
Activation	Leaky ReLU	ReLU
Latent Dim.	1568	1568
Parameters	8.0M	25.4M

The time spectrum of the E-PAE operators are shown in Figure B.20a and B.20b. These operators' eigenvalues are also real, indicating that there is no oscillatory behavior in any of these modes - a recurrence across all PAE variants.

The modes of Model D (shown in Figure B.19), on the other hand, indicate that the model is not learning to integrate the noise into any of the modes but learning smooth modes that are more representative of actual digits (or steps later in the denoising process). Additionally, the presence of so many modes - especially in comparison to the C-PAE - leads to the integration of features from other digits. While this also happens in models with a smaller latent dimension, it is more obvious in the C-PAE and leads to transitions between digit classes.

5.3.3 Extension to Other Datasets

The MNIST dataset is relatively simple, with large-scale features shared not only within the same digit class but also across them. A better-fit image processing dataset, such as the CIFAR-10 dataset, can be used to more comprehensively test this class of models.

For this experiment, an additional hyperparameter is introduced to vary the model structure. If this hyperparameter is "true", the encoder and decoder will be CNN-based. If it is "false", it will be comprised of only linear layers and activation functions.

While a comprehensive analysis is not performed on this dataset, several models are selected for comparison, with their parameters shown in Table 5.3:

- **Model E** reconstructs the original image with apparent quantization noise to an image characterized by a large amount of Gaussian blur. Figure B.21 shows the reconstruction for all 10 classes.

- **Model F** reconstructs the original image with large contrast shifts and transitions to a high-contrast pattern that reappears across multiple initial conditions. Reconstruction for the 10 image classes is shown in Figure B.22.

- **Model G**, the only E-PAE shown here, fails to reconstruct any initial or final images. Instead, it reconstructs a common pattern with varying contrast levels. Figure B.23 shows Model G's reconstruction for all 10 classes.

These models, similar to the MNIST dataset, do not exactly reconstruct the denoising process but show potential for capturing the underlying dynamics of the denoising system. Of particular interest is the behavior of models E and F, which show the ability to transition between different patterns that are not obviously present in the data itself (like contrast shifts or quantization noise). These reconstruction dynamics alone yeild insight into the effect of noise on the underlying data and provide a glimpse into the interactions between noise and image features. However, it is important to note that Model G fails to reconstruct any meaningful images. Instead, it reconstructs a common pattern with varying contrast levels, indicating that the model may struggle to capture the intricate dynamics of the denoising process in the CIFAR-10 dataset. Further investigation and experimentation are required to fully understand the capabilities and limitations of E-PAEs in the context of more complex image datasets.

5.3.4 Hyperparameter Search

The hyperparameter grid search generated 443 different network architectures. Each network is evaluated, and the reconstruction dynamics are classified. The hyperparameter combination and dynamics classification are then fit with a nominal logistic regression to generate a probability distribution of each classification for a given combination of hyperparameters. The interactive profile of the three nominal logistic regressions is shown in the attached mp4 files titled "0_Step_Regression.mp4", "50_Step_Regression.mp4", and "100_Step_Regression.mp4". These videos show how each parameter affects the probability of certain dynamics occurring for the single-step prediction task, 50-step prediction task, and 100-step prediction task, respectively.

Table 5.3: Parameters used for the three selected models trained on the CIFAR-10 dataset.

Parameter	Model E	Model F	Model G
Dropout	0	0	0
Layers	5	5	2
L_1 weight	1×10^{-5}	1×10^{-5}	1×10^{-4}
Activation	Leaky ReLU	Leaky ReLU	Sigmoid
Latent Dim.	1536	3072	6144
Network Structure	CNN	CNN	FCN
Parameters	3.9M	11.7M	122.7M

Dropout has a negligible effect on the resulting dynamics while the number of layers and weight of L_1 loss component are directly correlated with the resulting distribution for most activation functions. The tanh activation function has the least diverse set of dynamics, including the "noise", "mostly solid", and "constant pattern: no number" classes. Across all combinations of parameters, as the number of layers and latent dimension increases (not necessarily together), the reconstruction less closely resembles the true samples. This is evidenced by the growing probability of the "constant pattern: no number", "evolving pattern: no numbers", and "noise" classes. This is contrary to the original hypothesis that a model with a larger latent space would better model the dynamics of diffusion. It also fails to meet a secondary hypothesis that a model with a larger capacity would be better equipped to reproduce complex, stochastic dynamics.

5.4 Comparison With DDPMs

Using a U-Net with 61M parameters trained on the MNIST dataset, the model can reconstruct shapes resembling digits. Naturally, the model can reconstruct the true denoising process of the input image. This model, however, also suffers from an inherent inability to separate features from different classes of digits, leading to some images (like those shown in Figures B.24a and B.24c) that are hybrids of two or more digits. Other digits, like those shown in Figures B.24b and B.24d, are not clearly recognizable but contain defects around the borders of the digits. While the background is solid black for all digits recreated by the diffusion model, the digits themselves are easily distinguished from true MNIST samples. The samples do not become visible until the final few steps of the denoising process. The entire predicted denoising process from the U-Net is shown in Figure B.25. In this figure, the 6 does not become visible until the 90th step, with the remaining steps simply containing noise.

While many have successfully used DDPMs for image denoising, the results presented here are not as successful. There are two likely reasons for this: 1) The model is trained in state space, not in latent space. While an MNIST image is relatively simple, performing state-space diffusion opens the door for introducing artifacts that may not affect the SSIM but are perceptible to the human eye. 2) The training process and model structure have not been fine-tuned for this specific task, as this was not the focus of this work. With a larger model, more augmentations, different learning rate scheduler etc., the model would likely be able to produce images that are indistinguishable from the true MNIST dataset while still being able to reconstruct the denoising process.

In summary, employing a U-Net-style architecture obfuscates many of the underlying dynamics that contribute to this branch of AI's generative capability. It's possible to explore U-Nets using conventional attention-map or heatmap techniques, but these methods have their own shortfalls, such as sensitivity to input or confusion between correlation and causation. The PAE suite of models provides - not only smaller models - but models that are more easily interpretable. Additionally, the use of a linear operator introduces the opportunity to use classical control methods to guide generative processes that were otherwise complex in DDPM methods.

Chapter 6

Conclusion and Future Work

This work has explored the application of Koopman Operator Theory to the study of DDPMs, offering a new perspective on the dynamics of diffusion processes. By representing the nonlinear reverse diffusion process in DDPMs as a linear operator within a Koopman-invariant subspace, this work has provided valuable insights into the transformation of probability distributions during the diffusion process. The utilization of PAEs for learning the Koopman operator has demonstrated the potential for capturing the system's dynamics in a more interpretable and analyzable manner.

The investigation of various approximation methods, including DMD and PAEs, has shed light on the potential for understanding the underlying dynamics of diffusion processes. The application of these methods to datasets such as the Swiss Roll, MNIST, and CIFAR-10 has illustrated the versatility and effectiveness of the Koopman-based approach in different contexts and some of the limitations of this approach.

Furthermore, this thesis has contributed to the theoretical understanding of diffusion models by providing an analysis of the Koopman-invariant subspace's properties. This analysis has implications for the stability and robustness of diffusion processes, offering a foundation for future research in this area.

There are several areas to explore and expand upon in this work. Of most interest is the training paradigm. It is well-known that diffusion models are data-hungry and require specific fine-tuning for feasible results. In the same light, this branch of models may require investigating different optimizers, learning rate schedulers, augmentations, and loss functions. Specifically, a scheduler to increment from a single-step prediction loss function to a multi-step prediction could prove beneficial in capturing the long-term dynamics of these complex systems.

Another area of interest is further exploring the properties of the learned operators to determine if there is a pattern that could be leveraged to improve accuracy and efficiency. A simple example is the symmetric nature of every operator trained - while there are no oscillatory patterns in the diffusion process, it is interesting that the noise does not manifest as oscillations of modes.

In closing, this thesis has opened new avenues for applying Koopman Operator Theory in the realm of diffusion models, particularly in the context of DDPMs. The findings and methodologies presented herein lay a solid groundwork for further exploration and refinement of these models, with an emphasis on enhancing their interpretability, stability, and robustness. As the field continues to evolve, it is anticipated that integrating Koopman Operator Theory with diffusion processes will lead to more sophisticated and efficient approaches for modeling complex dynamical systems across various domains.

Bibliography

Alford-Lago, D. J., Curtis, C. W., Ihler, A. T., and Issan, O. (2022). Deep learning enhanced dynamic mode decomposition. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(3):033116. 19, 35, 37

Arbabi, H. and Mezic, I. (2017). Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the koopman operator. *SIAM Journal on Applied Dynamical Systems*, 16(4):2096–2126. 9, 17, 34

Astolfi, A. and Marconi, L., editors (2010). *Analysis and design of nonlinear control systems*. Springer, Berlin, Germany. 8

Astrom, K. J. and Murray, R. M. (2008). *Feedback systems*. Princeton University Press, Princeton, NJ. 8

Azencot, O., Erichson, N. B., Lin, V., and Mahoney, M. W. (2020). Forecasting sequential data using consistent koopman autoencoders. 19

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166. 38

Björck, Å. (1996). *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics. 7

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis*. Wiley Series in Probability and Statistics. John Wiley & Sons, Nashville, TN, 5 edition. 7

Brunton, S. L., Brunton, B. W., Proctor, J. L., and Kutz, J. N. (2016a). Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control. *PLOS ONE*, 11(2):e0150171. 10, 13

Brunton, S. L., Proctor, J. L., and Kutz, J. N. (2016b). Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113(15):3932–3937. 7

Budišić, M., Mohr, R., and Mezić, I. (2012). Applied koopmanism. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4). 9

Carleman, T. (1932). Application de la théorie des équations intégrales linéaires aux systèmes d'équations différentielles non linéaires. 10

Chang, Z., Koulieris, G. A., and Shum, H. P. H. (2023). On the design fundamentals of diffusion models: A survey. 28

Chen, K. K., Tu, J. H., and Rowley, C. W. (2012). Variants of dynamic mode decomposition: Boundary conditions, Koopman, and Fourier analyses. *Journal of Nonlinear Science*, 22(6):887–915. 16, 17

Chung, H., Kim, J., Mccann, M. T., Klasky, M. L., and Ye, J. C. (2023). Diffusion posterior sampling for general noisy inverse problems. 30

Dashti, M. and Stuart, A. M. (2017). The bayesian approach to inverse problems. In Ghanem, R., Higdon, D., and Owhadi, H., editors, *Handbook of Uncertainty Quantification*, pages 311–428. Springer International Publishing, Cham. 23

Dou, Z. and Song, Y. (2023). Diffusion posterior sampling for linear inverse problem solving: A filtering perspective. In *The Twelfth International Conference on Learning Representations*. 30

Goswami, B. N. and Shukla, J. (1991). Predictability of a coupled ocean-atmosphere model. *Journal of Climate*, 4. 32, 45

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109. 23

Hinch, E. J. (1991). *Cambridge texts in applied mathematics: Perturbation methods series number 6*. Cambridge University Press, Cambridge, England. 8

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. 28, 29, 38

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780. 38

Khalil, H. K. (2013). *Nonlinear Systems: Pearson New International Edition*. Pearson Education, London, England, 3 edition. 8, 38

Koopman, B. O. (1931). Hamiltonian systems and transformation in hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318. 9

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report. 40

Kutz, J. N., Brunton, S. L., Brunton, B. W., and Proctor, J. L. (2016). *Dynamic Mode Decomposition*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics. 17

LeCun, Y., Cortes, C., and Burges, C. (2010). Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2. 40

Lusch, B., Kutz, J. N., and Brunton, S. L. (2018). Deep learning for universal linear embeddings of nonlinear dynamics. *Nature Communications*, 9(1):4950. 17

Marsland, S. (2014). *Machine Learning: An Algorithmic Perspective, Second Edition*. Chapman & Hall/Crc Machine Learning & Pattern Recognition. Chapman and Hall/CRC, 2 edition. 40

Miller, E. L. (2003). *Fundamentals of Inverse Problems*. 22

Myung, I. J. (2003). Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, 47(1):90–100. 7

Neal, R. M. (1998). Annealed importance sampling. 24

Nichol, A. and Dhariwal, P. (2021). Improved denoising diffusion probabilistic models. 25

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2021). High-resolution image synthesis with latent diffusion models. https://arxiv.org/abs/2112.10752v2. 24, 31, 38

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. 28

Rout, L., Raoof, N., Daras, G., Caramanis, C., Dimakis, A. G., and Shakkottai, S. (2023). Solving linear inverse problems provably via posterior sampling with latent diffusion models. 30

Rowley, C. W., Mezić, I., Bagheri, S., Schlatter, P., and Henningson, D. S. (2009). Spectral analysis of nonlinear flows. *Journal of Fluid Mechanics*, 641:115–127. 15, 16

Saharia, C., Chan, W., Saxena, S., Li, L., Whang, J., Denton, E., Ghasemipour, S. K. S., Ayan, B. K., Mahdavi, S. S., Lopes, R. G., Salimans, T., Ho, J., Fleet, D. J.,

and Norouzi, M. (2022). Photorealistic text-to-image diffusion models with deep language understanding. https://arxiv.org/abs/2205.11487v1. 24

Schmid, P. J. (2010). Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28. 14

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. 24, 28

Strogatz, S. H. (2015). *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Westview Press, a member of the Perseus Books Group, Boulder, CO, second edition edition. 6

Tu, J. H., Rowley, C. W., Luchtenburg, D. M., Brunton, S. L., and Kutz, J. N. (2014). On Dynamic Mode Decomposition: Theory and Applications. *Journal of Computational Dynamics*, 1(2):391–421. 1, 15, 16, 17, 32

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. 28

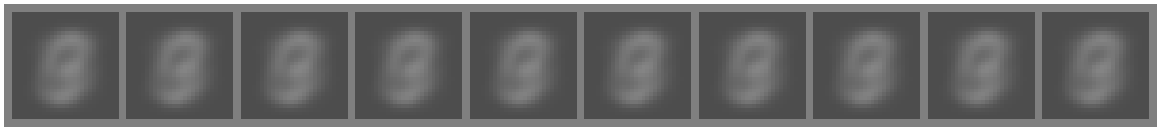
von Platen, P., Patil, S., Lozhkov, A., Cuenca, P., Lambert, N., Rasul, K., Davaadorj, M., Nair, D., Paul, S., Liu, S., Berman, W., Xu, Y., and Wolf, T. (2023). Diffusers: State-of-the-art diffusion models. `https://github.com/huggingface/diffusers`. Accessed: 2024-04-07. 27

Wang, Z. and Bovik, A. C. (2009). Mean squared error: Love it or leave it? a new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1):98–117. 37

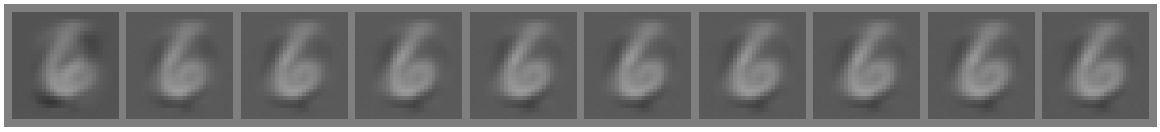
Williams, M. O., Kevrekidis, I. G., and Rowley, C. W. (2015). A data-driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science*, 25(6):1307–1346. 17

Appendix

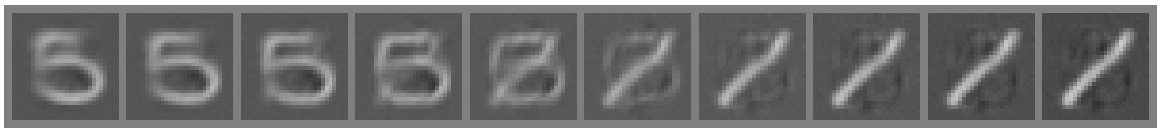
A Dynamics Classification



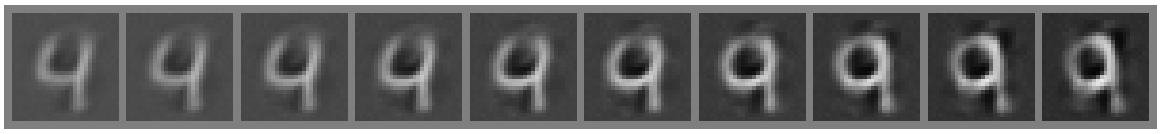
(a) Constant Pattern: No Number



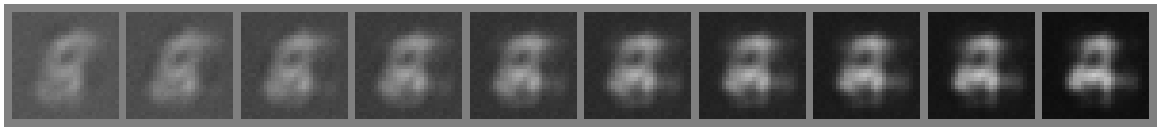
(b) Constant Pattern: Numbers



(c) Evolving Pattern: Some Numbers



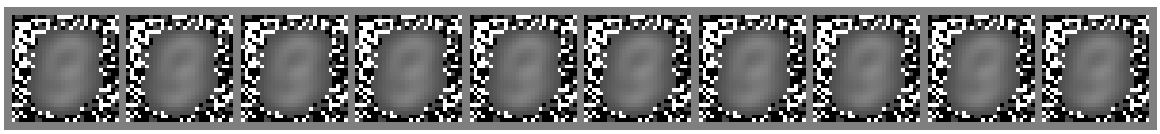
(d) Evolving Pattern: All Numbers



(e) Evolving Pattern: No Numbers

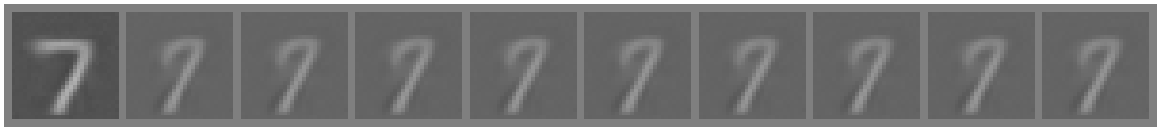


(f) Solid Pattern

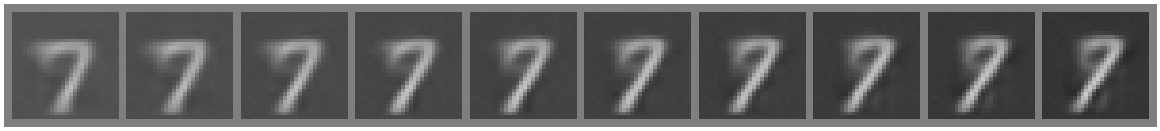


(g) Noise

Figure A.1: Example images from each class of the dynamics [back]



(a) Correct: All Faint



(b) Correct: All Obvious



(c) Correct: Obvious to Faint



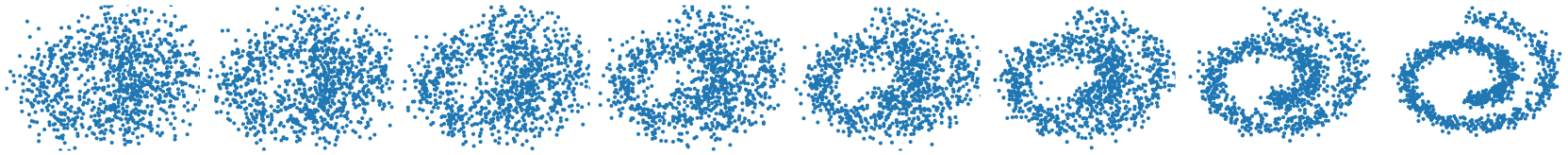
(d) Correct: Faint to Obvious



(e) Correct: Noise to Obviouse

Figure A.2: Example images from each class of the correct dynamics [back]

B Experiment Results

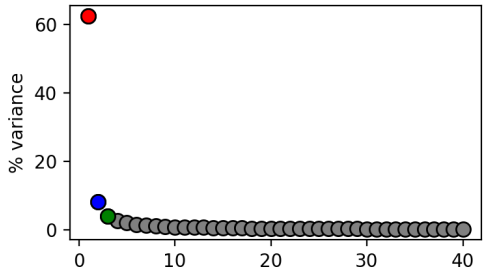


(a) Ground truth denoising samples used to fit DMD models.

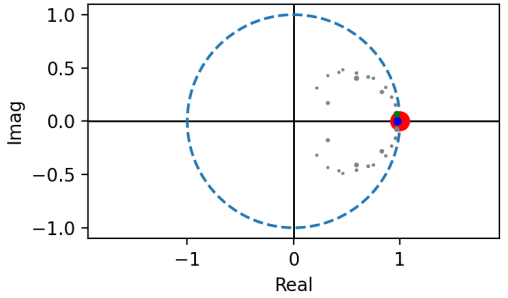


(b) Exact Dynamic Mode Decomposition reconstruction.

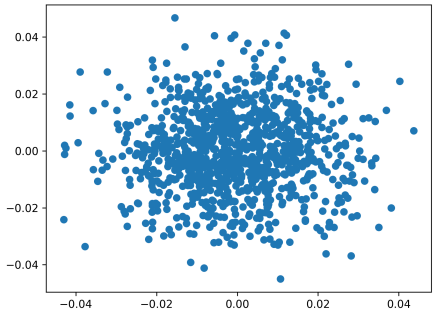
Figure B.1: Denoising samples for the Swiss Roll Dataset.
Left: State 0, **Right**: State 70. Every 10^{th} state is shown. [back]



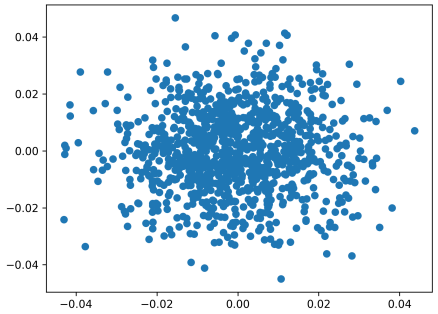
(a) Variance explained by singular values



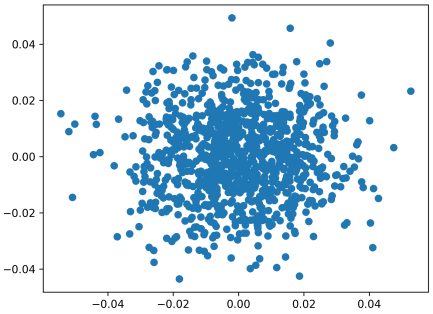
(b) Discrete eigenvalues



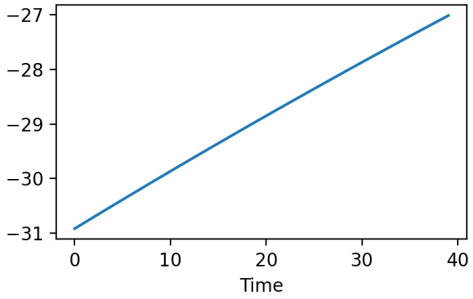
(c) Mode 1



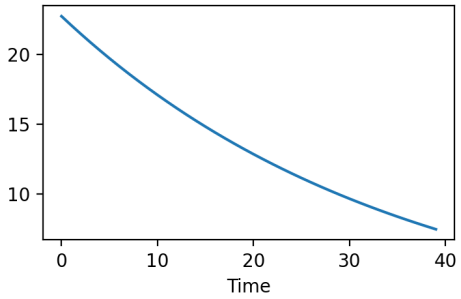
(d) Mode 2



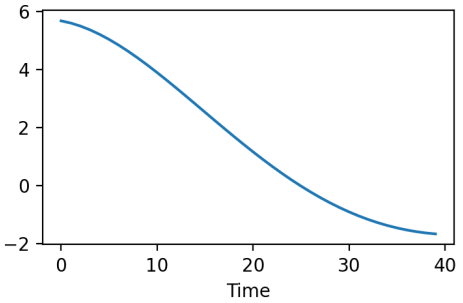
(e) Mode 3



(f) Mode 1 Dynamics



(g) Mode 2 Dynamics



(h) Mode 3 Dynamics

Figure B.2: Summary of the Exact DMD operator and the captured dynamics of the Swiss Roll dataset. [back]

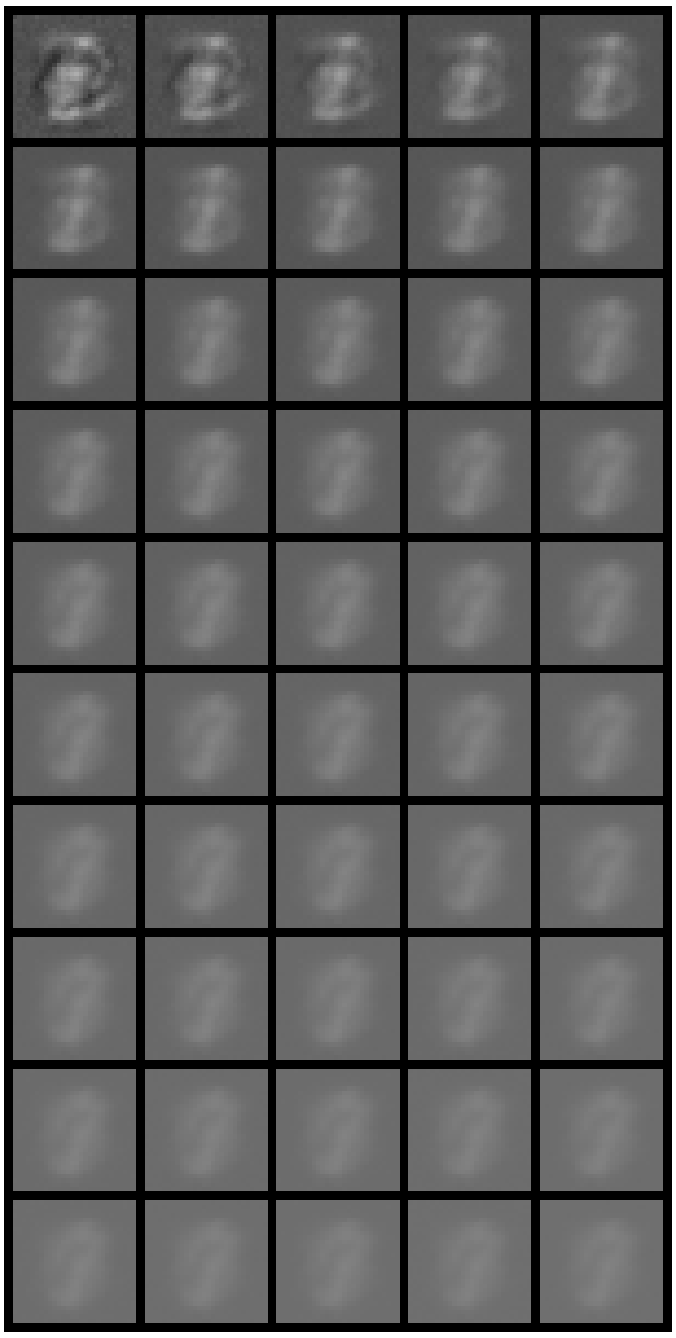
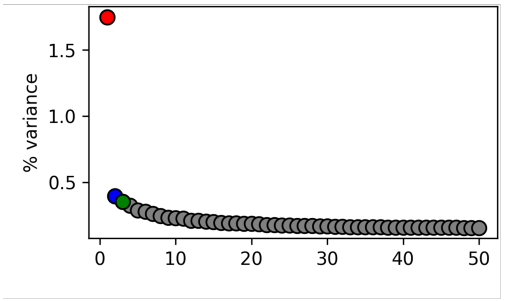
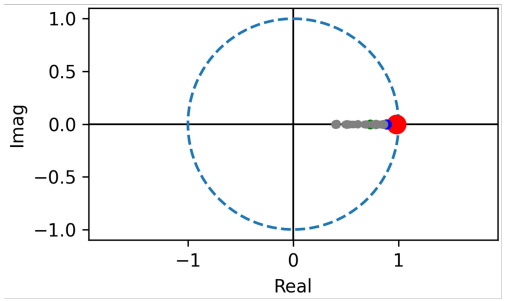


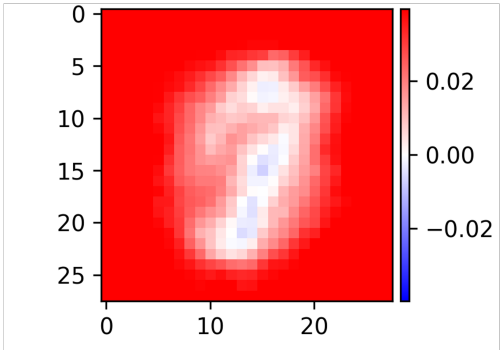
Figure B.3: Exact DMD reconstruction of the MNIST dataset. **Top Left:** State 500. **Bottom Right:** State 1,000 [back]



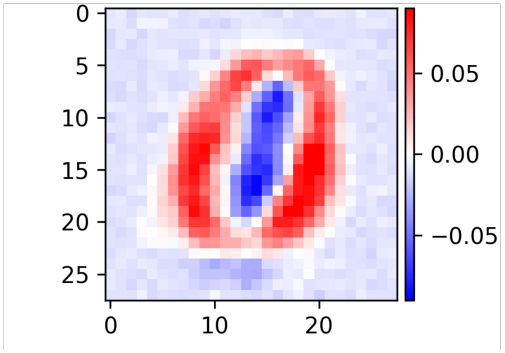
(a) Variance explained by singular values



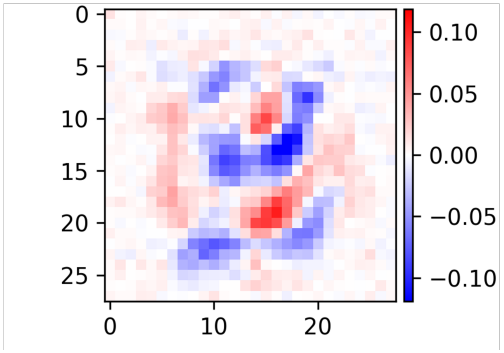
(b) Discrete eigenvalues



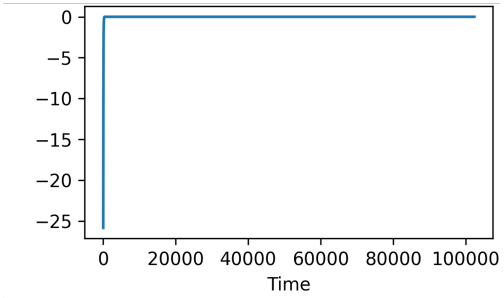
(c) Mode 1



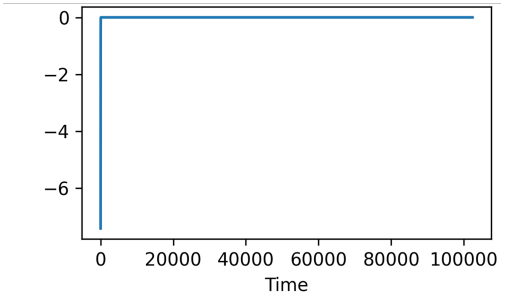
(d) Mode 2



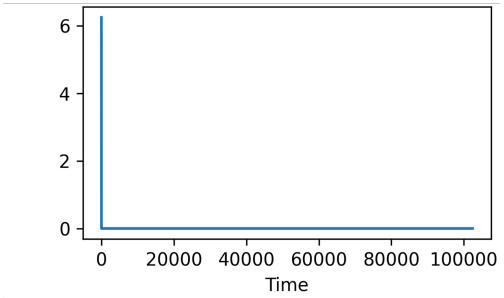
(e) Mode 3



(f) Mode 1 Dynamics



(g) Mode 2 Dynamics

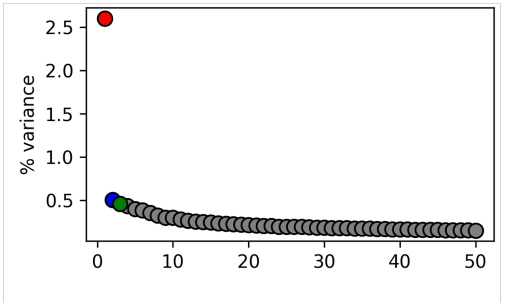


(h) Mode 3 Dynamics

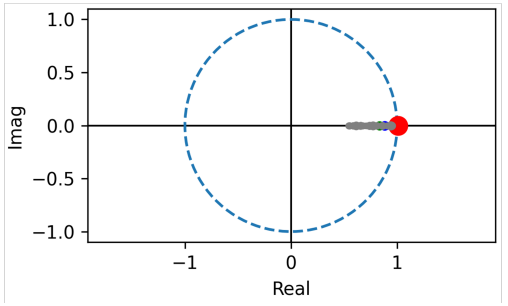
Figure B.4: Summary of the Exact DMD operator and the captured dynamics of the MNIST dataset. [back]



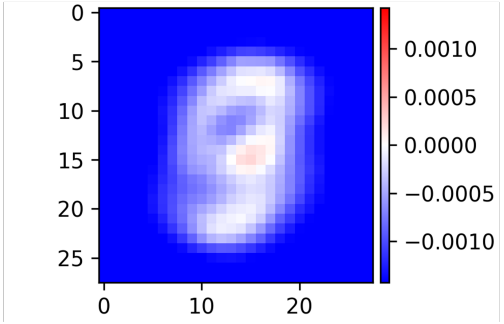
Figure B.5: Extended DMD reconstruction of the MNIST dataset. **Top Left:** State 500. **Bottom Right:** State 1,000 [back]



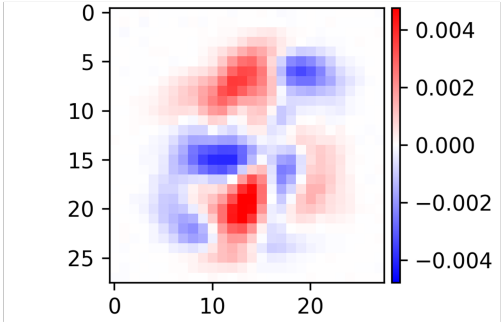
(a) Variance explained by singular values



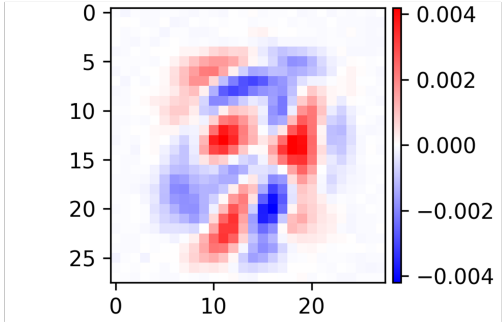
(b) Discrete eigenvalues



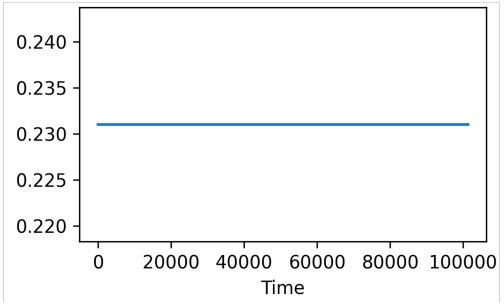
(c) Mode 1 in the spatial domain



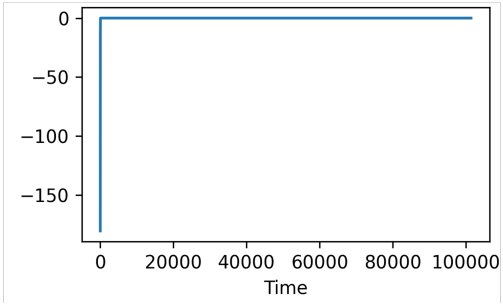
(d) Mode 2 in the spatial domain



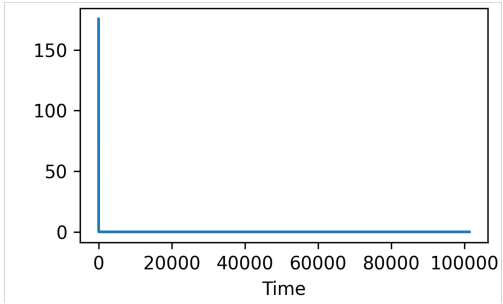
(e) Mode 3 in the spatial domain



(f) Mode 1 Dynamics



(g) Mode 2 Dynamics

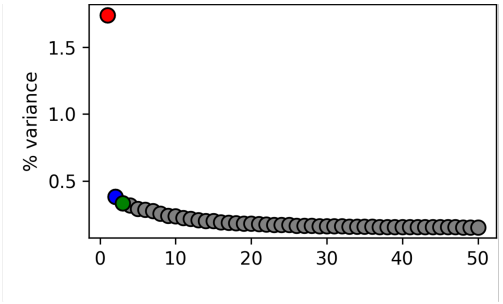


(h) Mode 3 Dynamics

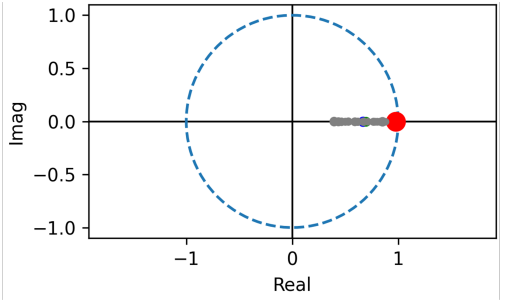
Figure B.6: Summary of the Extended DMD operator and the captured dynamics of the MNIST dataset. [back]



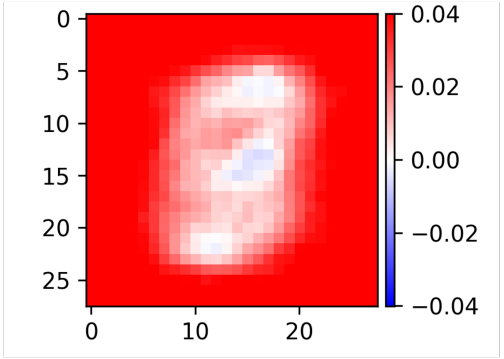
Figure B.7: Hankel DMD reconstruction of the MNIST dataset. **Top Left:** State 500. **Bottom Right:** State 1,000 [back]



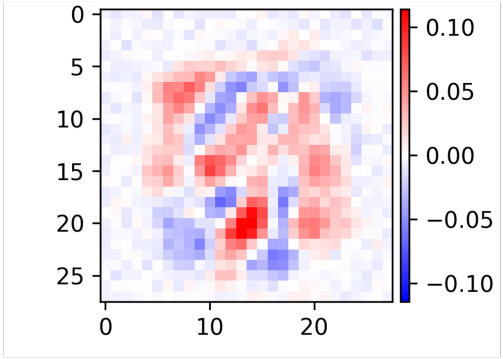
(a) Variance explained by singular values



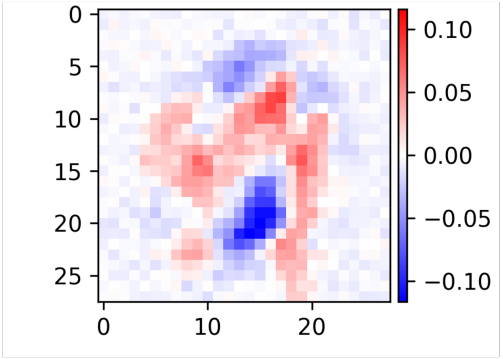
(b) Discrete eigenvalues



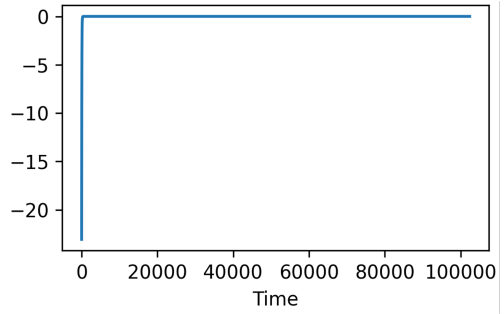
(c) Mode 1 in the spatial domain



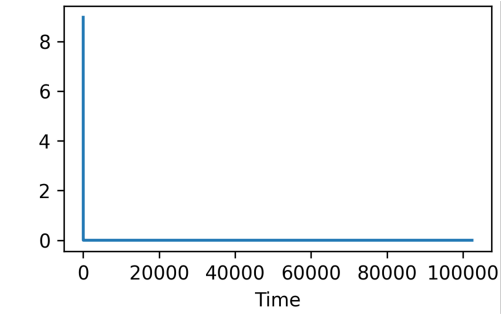
(d) Mode 2 in the spatial domain



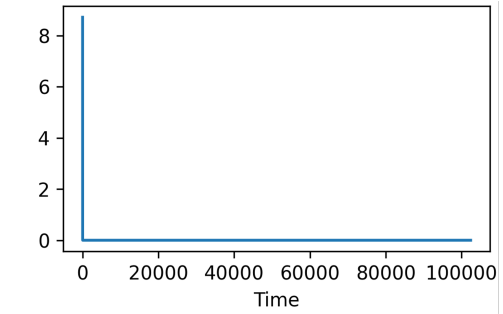
(e) Mode 3 in the spatial domain



(f) Mode 1 Dynamics



(g) Mode 2 Dynamics



(h) Mode 3 Dynamics

Figure B.8: Summary of the Hankel DMD operator and the captured dynamics of the MNIST dataset. [back]

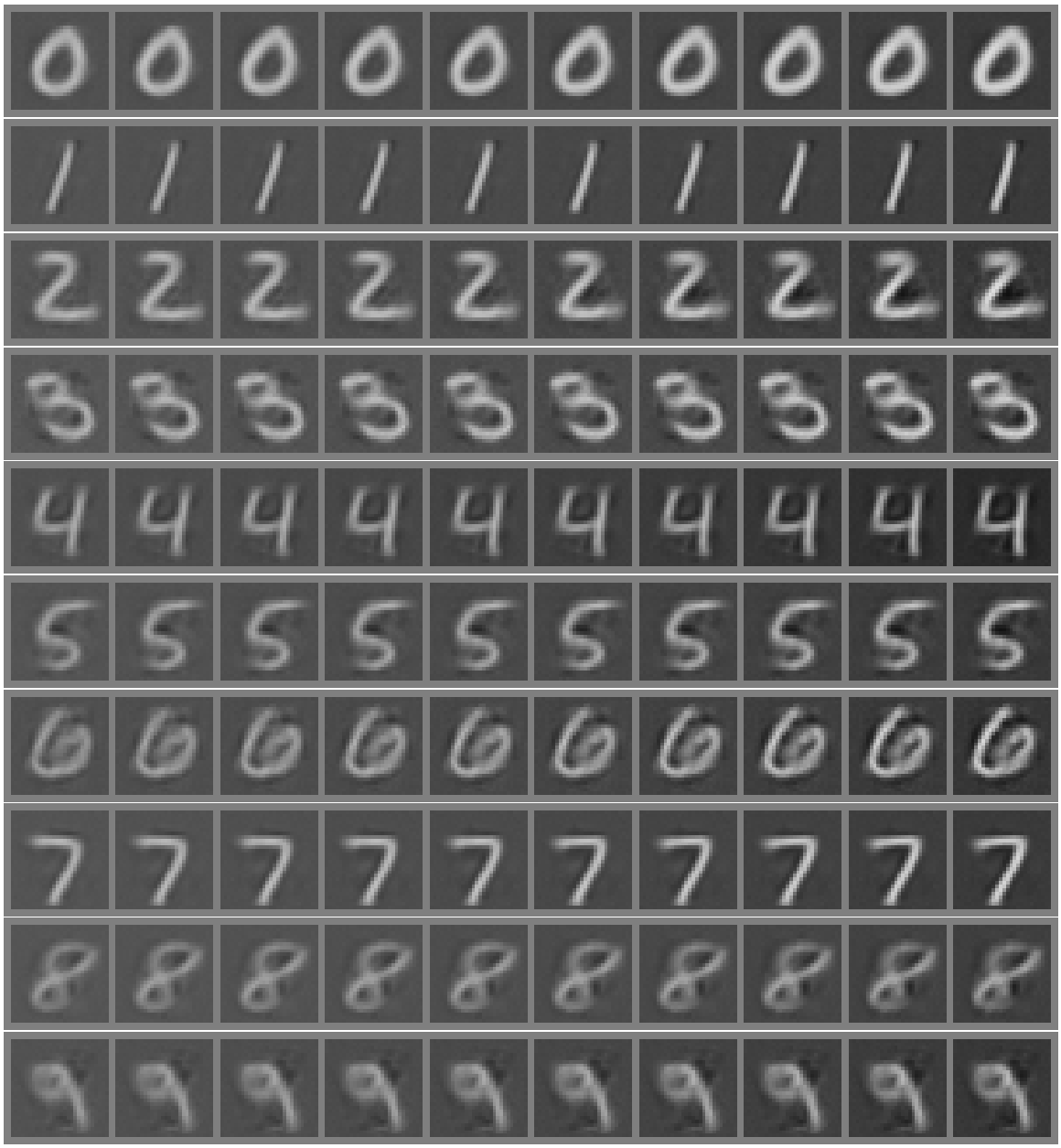


Figure B.9: Denoising process for 10 digits, generated from Model A. **Left**: Step 50, **Right**: Step 100. [back]



Figure B.10: Denoising process for 10 digits, generated from Model B. **Left**: Step 50, **Right**: Step 100. [back]

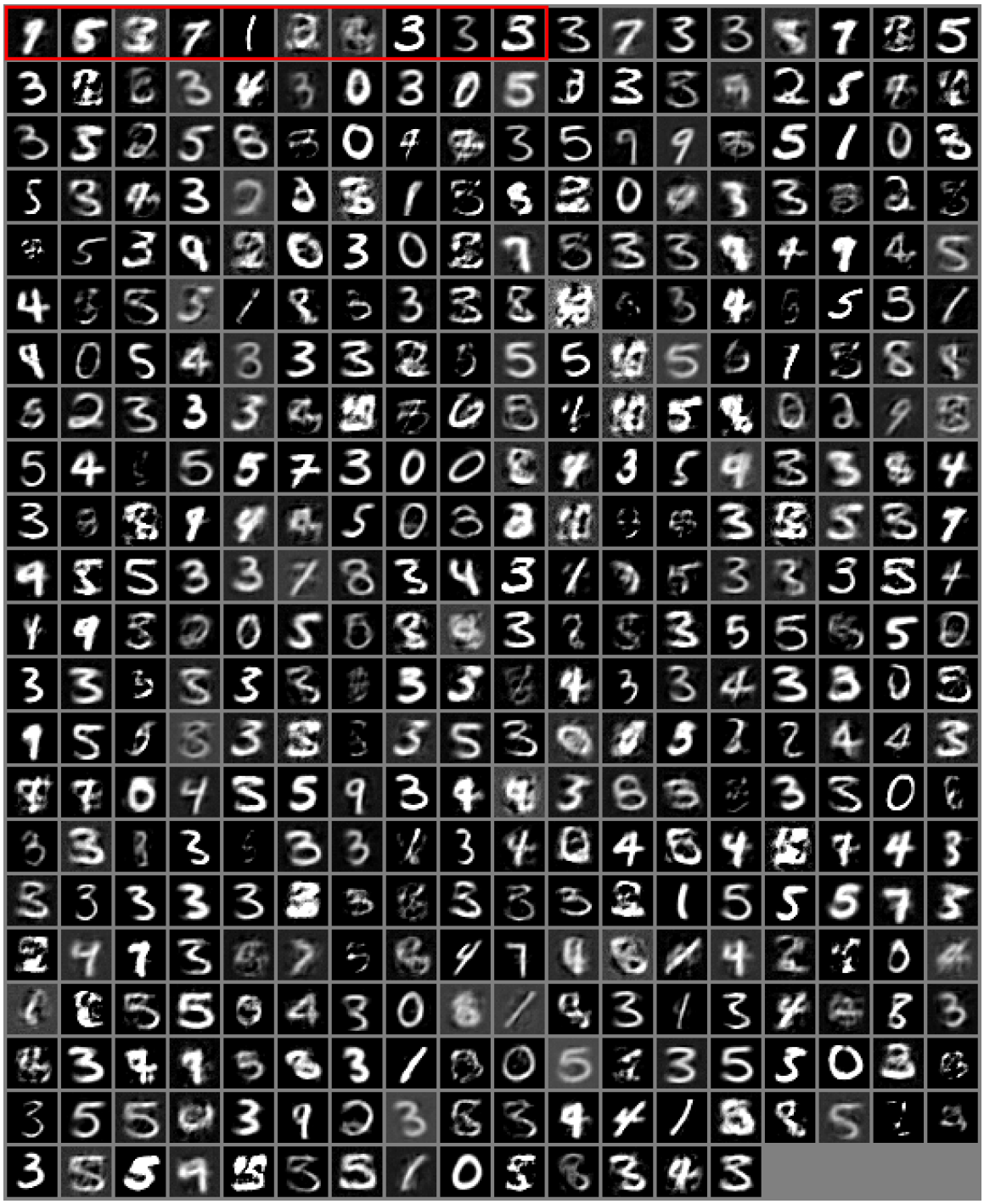


Figure B.11: Modes of Model A, a Compressive-Predictive Autoencoder, ordered by magnitude of the eigenvalue. The red rectangle indicates unstable modes. [back]

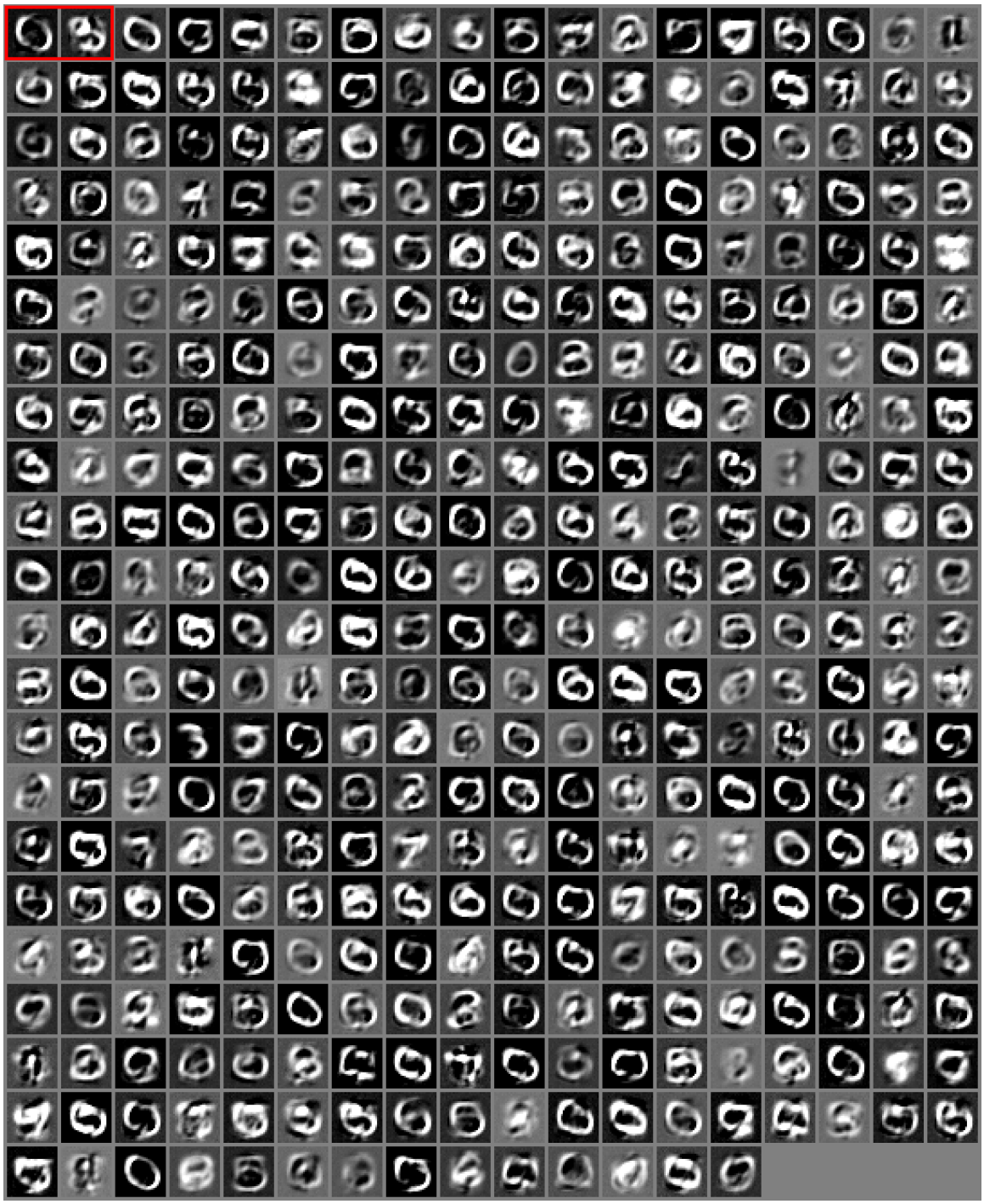
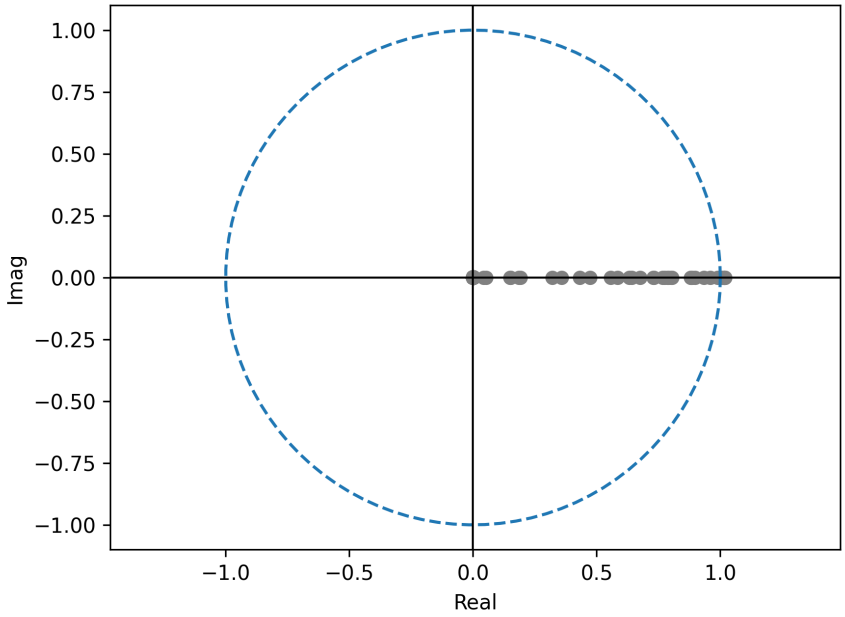
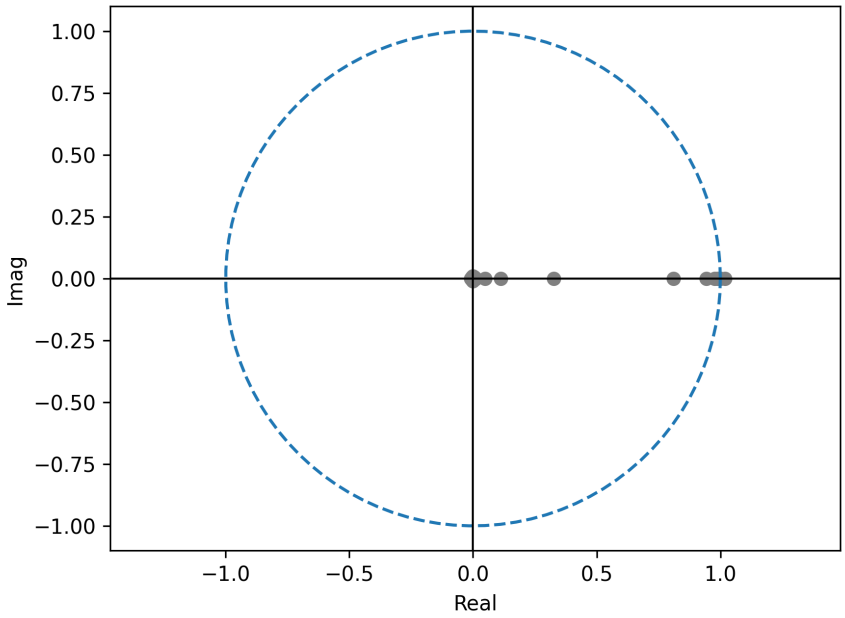


Figure B.12: Modes of Model B, a Compressive-Predictive Autoencoder, ordered by magnitude of the eigenvalue. The red rectangle indicates unstable modes. [back]



(a) Model A



(b) Model B

Figure B.13: Discrete eigenvalues of the linear operator in a Compressive-Predictive Autoencoder. [back]

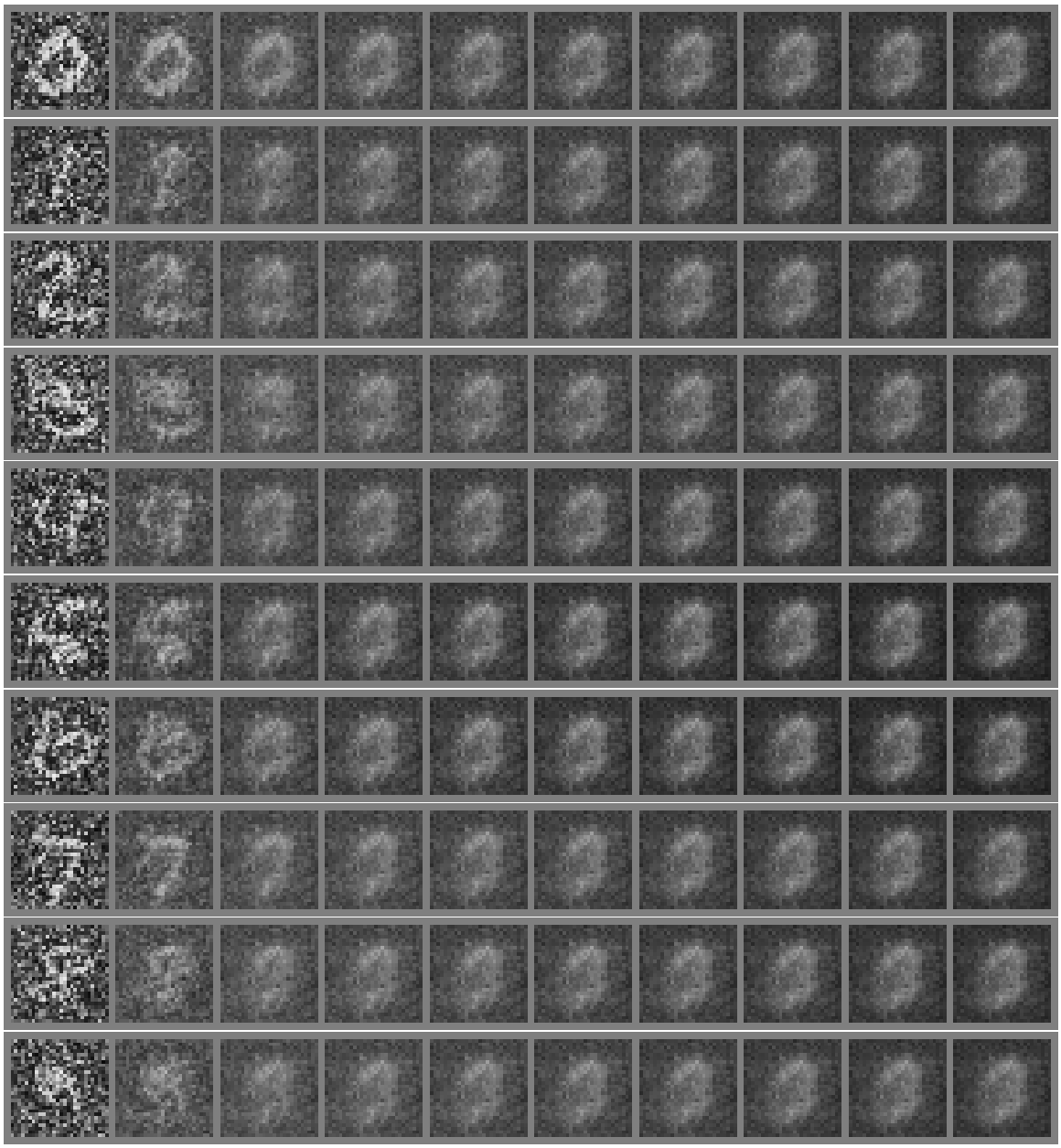


Figure B.14: Denoising process for 10 digits, generated from Model C.
Left: Step 50, **Right**: Step 100. [back]

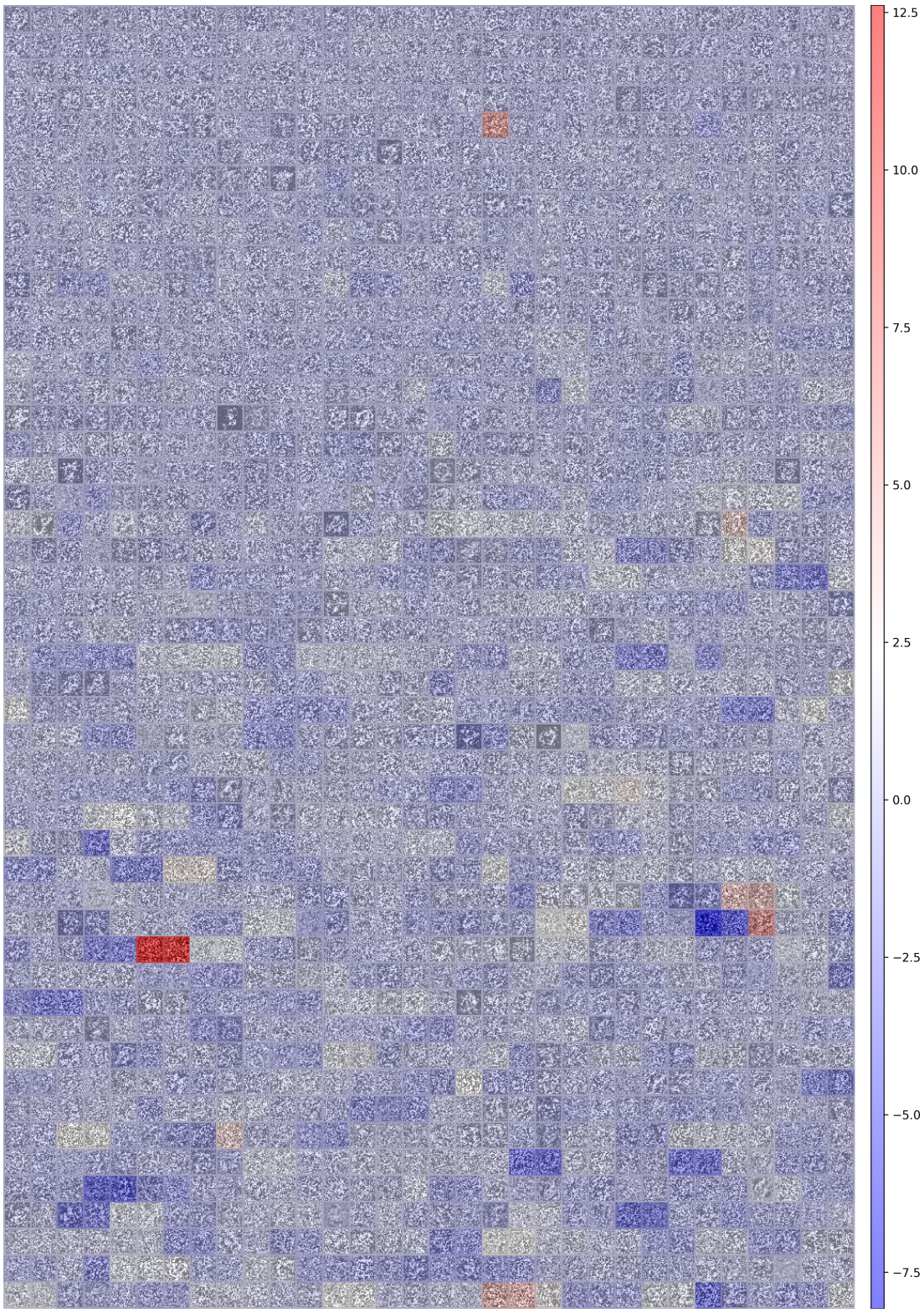


Figure B.15: Modes of Model C highlighted by the value of the coefficient required to reconstruct a denoised 7. [back]

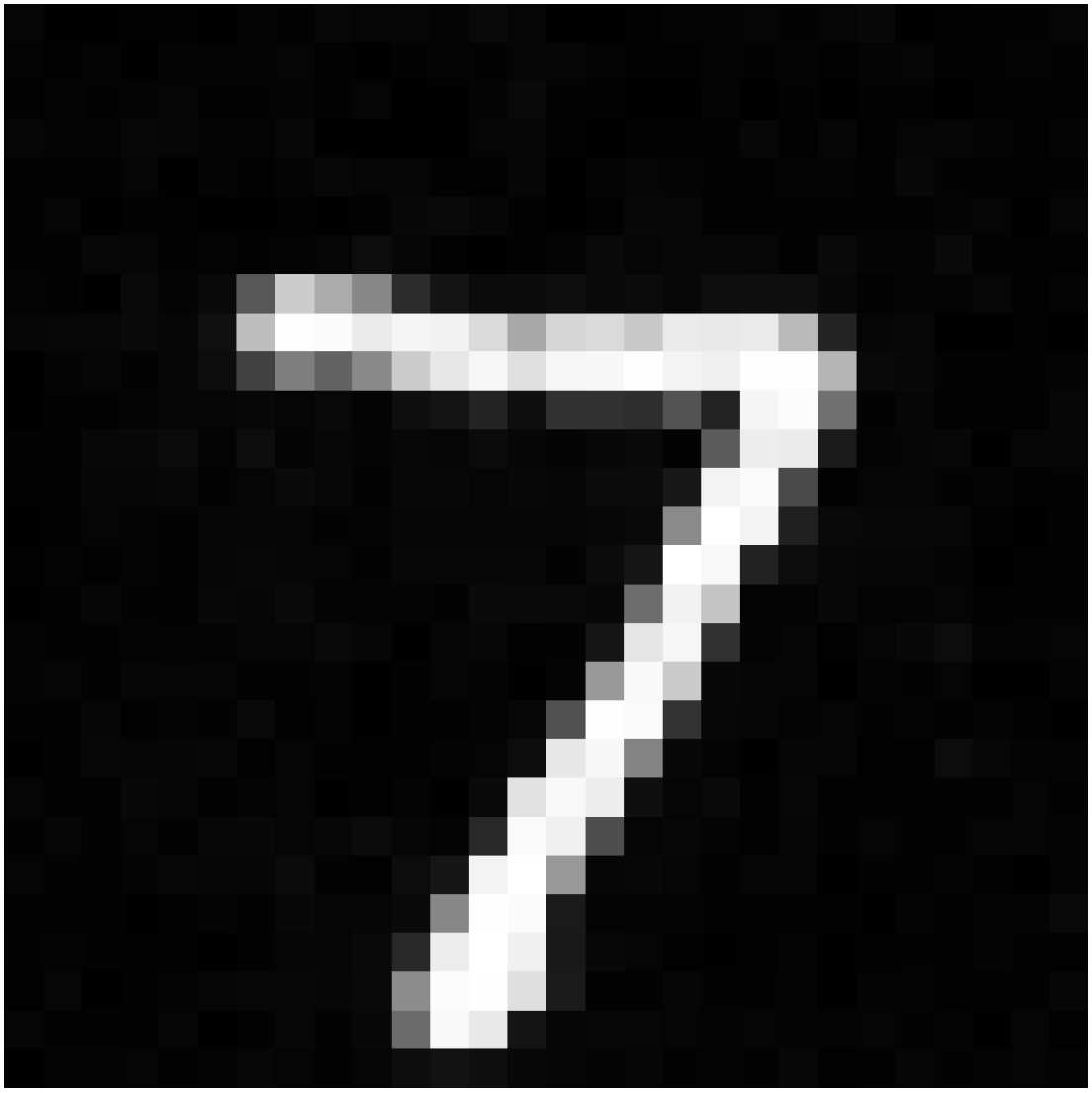


Figure B.16: Reconstruction of a denoised 7 using the calculated coefficients of the modes of Model C. [back]

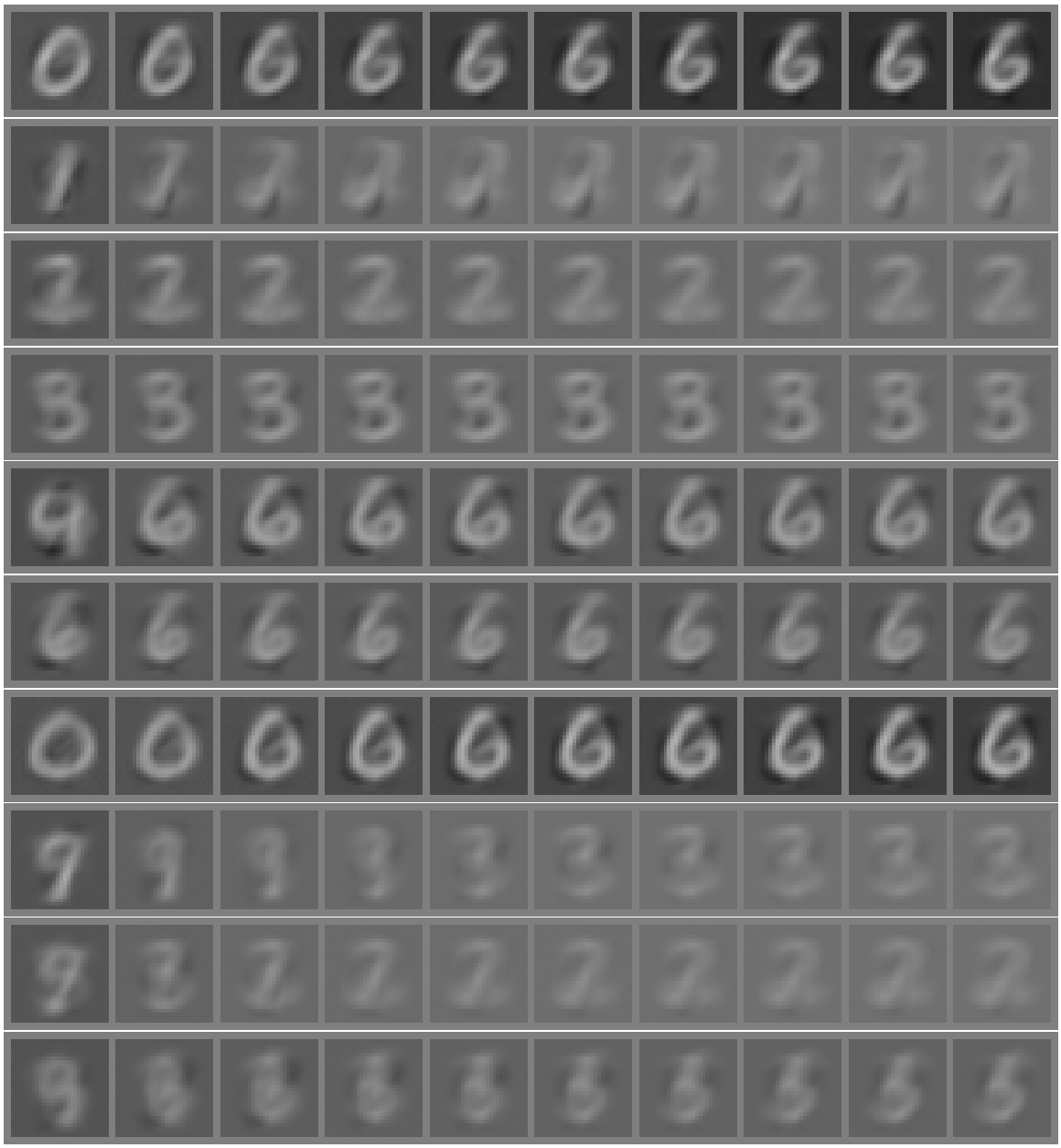


Figure B.17: Denoising process for 10 digits, generated from Model D.
Left: Step 50, **Right**: Step 100. [back]

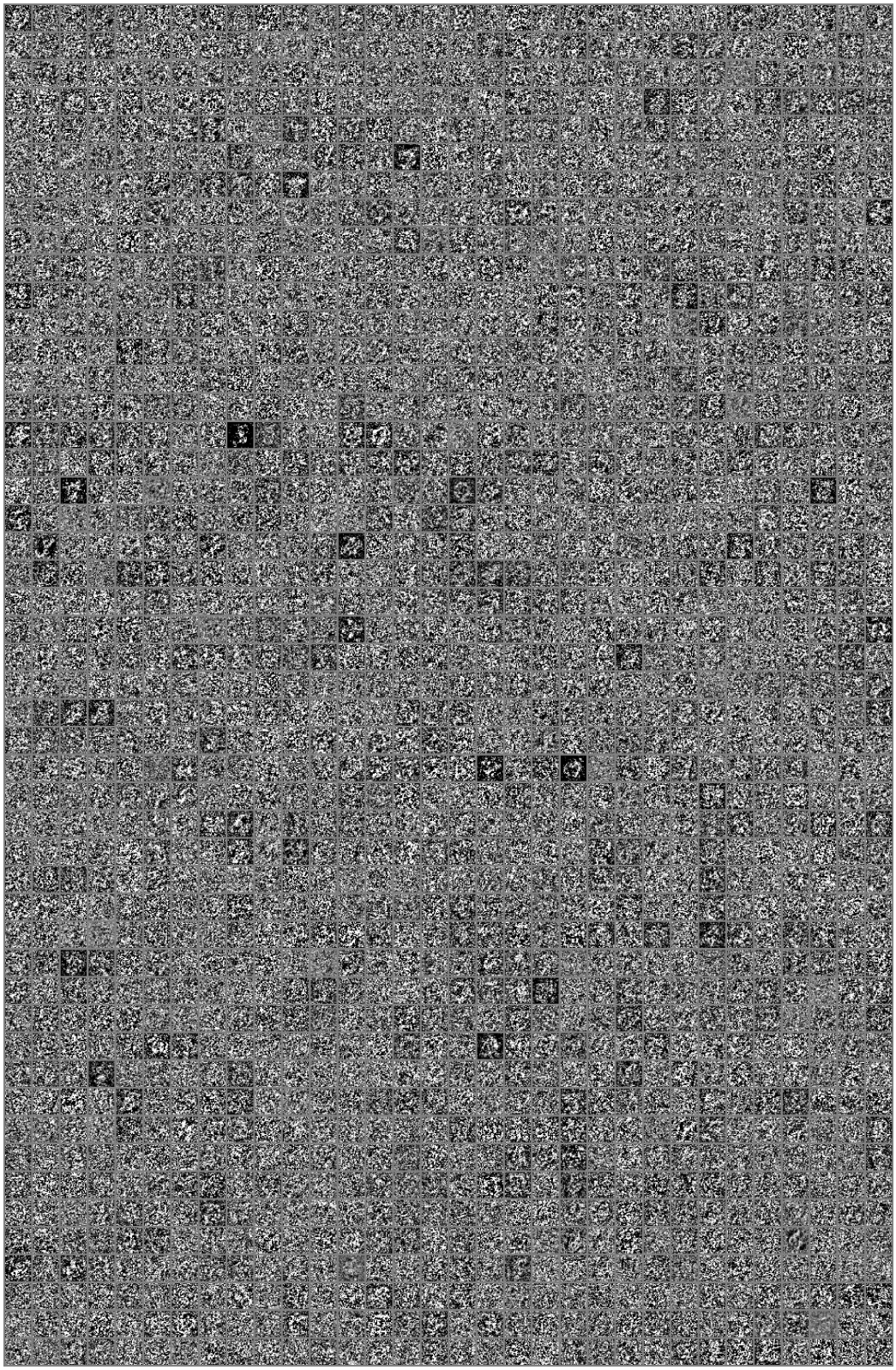


Figure B.18: Modes of Model C, an Expansive-Predictive Autoencoder, ordered by magnitude of the eigenvalue. All modes are stable. [back]

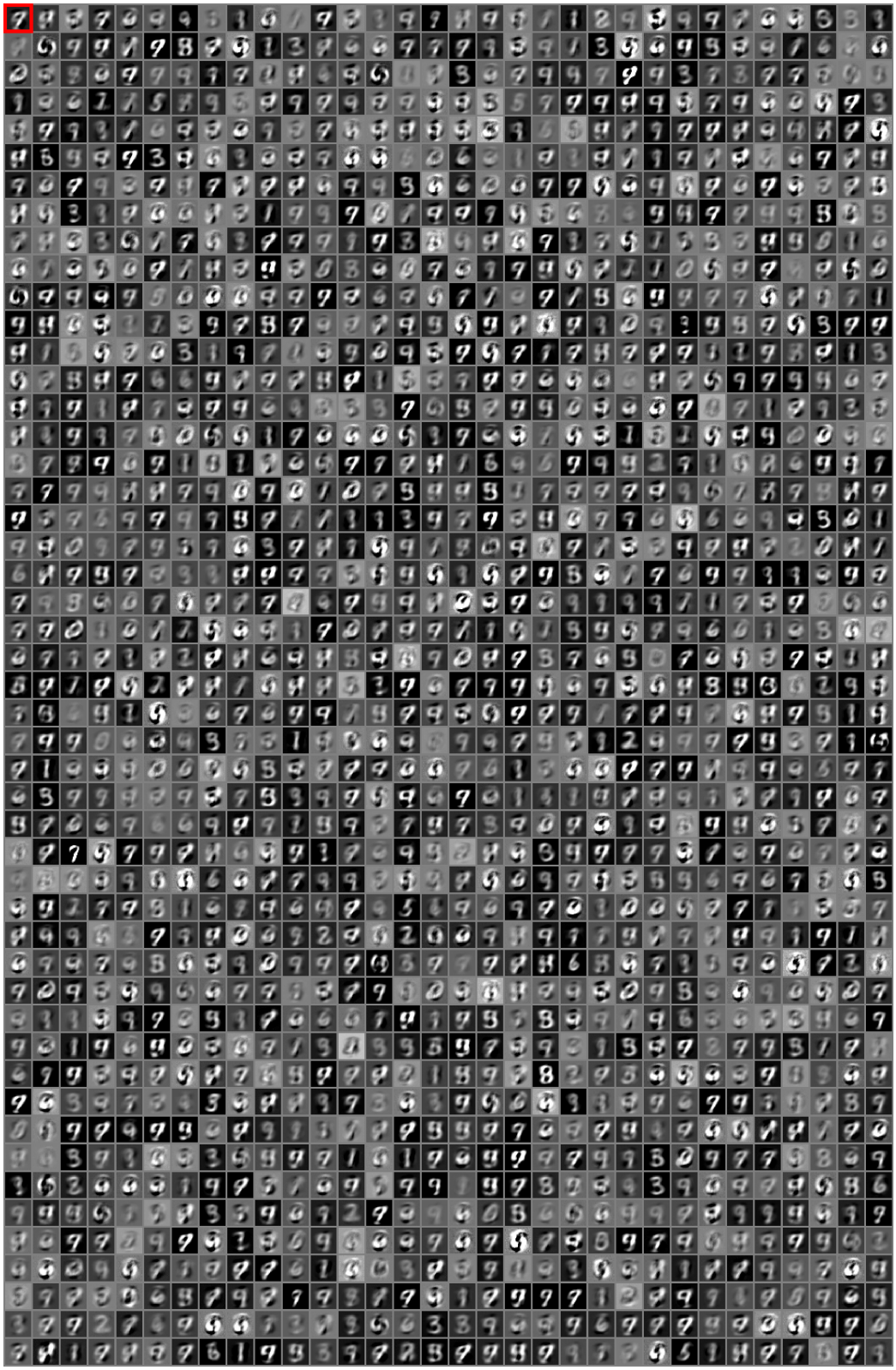
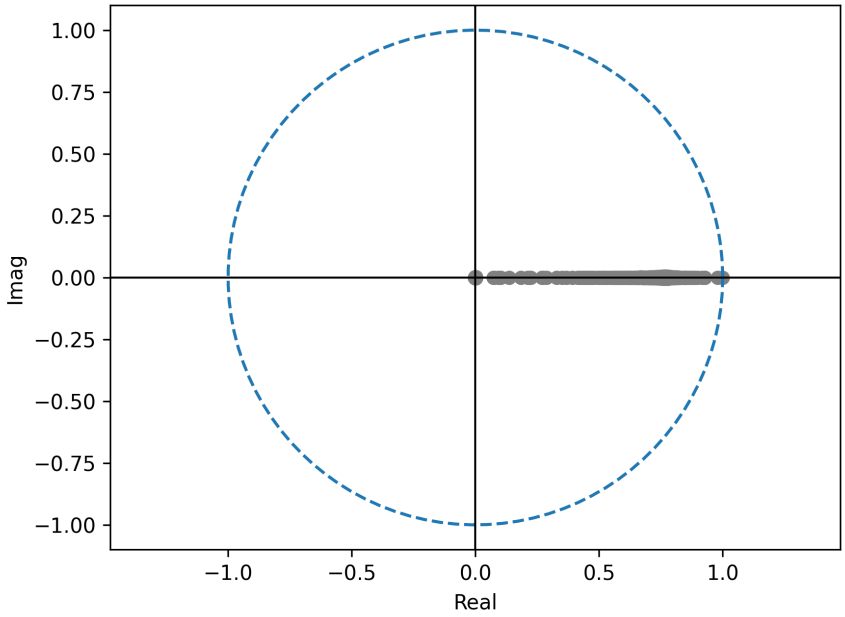
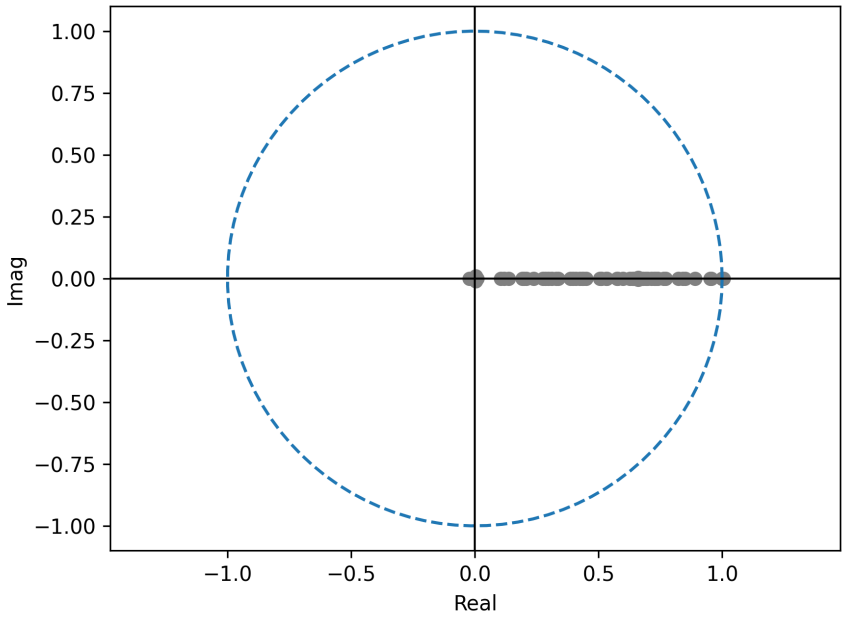


Figure B.19: Modes of Model D, an Expansive-Predictive Autoencoder, ordered by magnitude of the eigenvalue. The red rectangle indicates unstable modes. [back]



(a) Model C



(b) Model D

Figure B.20: Discrete eigenvalues of the linear operator in an Expansive-Predictive Autoencoder. [back]

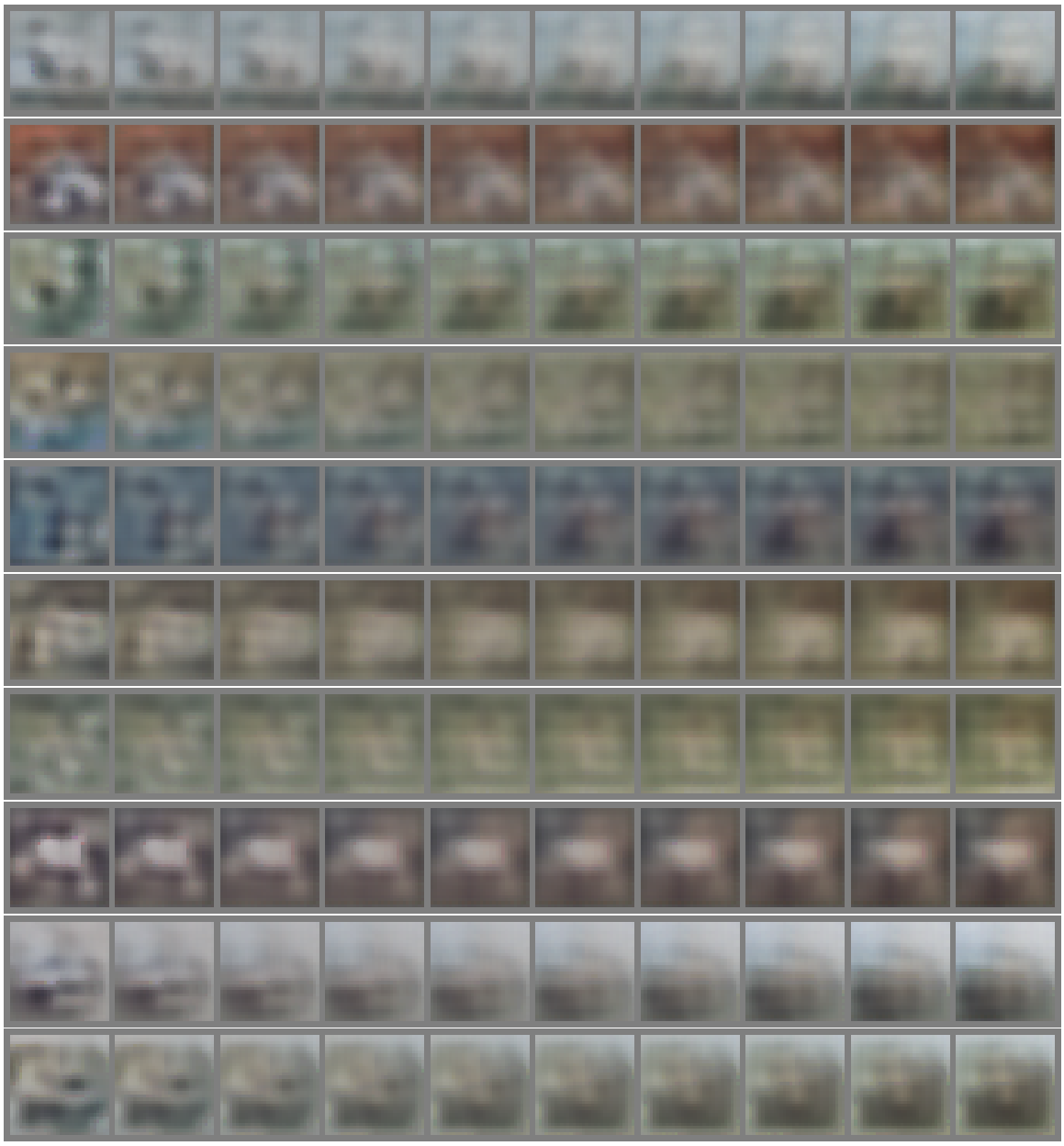


Figure B.21: Denoising process for 10 classes of CIFAR-10, generated from Model E.
Left: Step 50, **Right**: Step 100. [back]

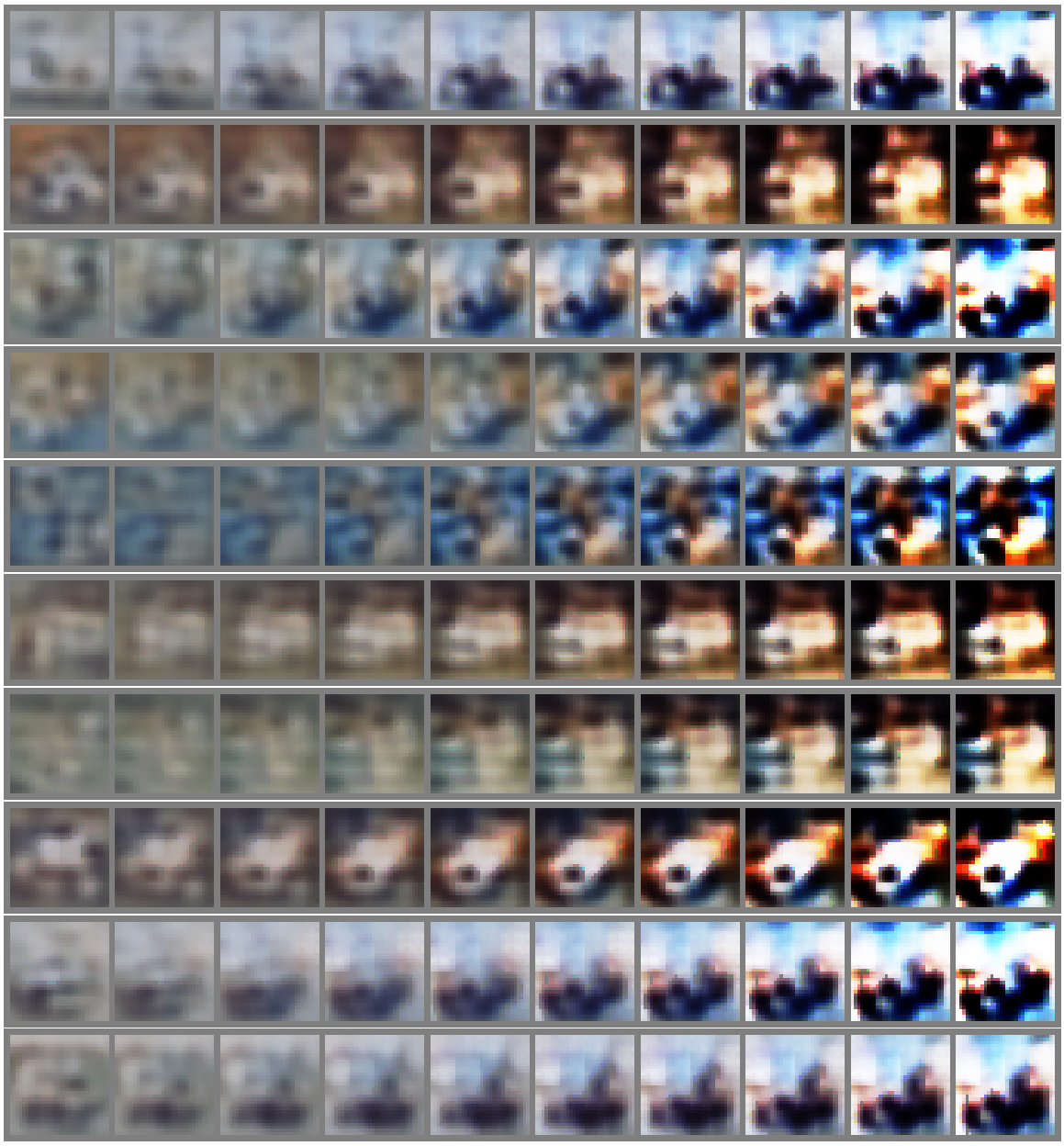


Figure B.22: Denoising process for 10 classes of CIFAR-10, generated from Model F.
Left: Step 50, **Right**: Step 100. [back]

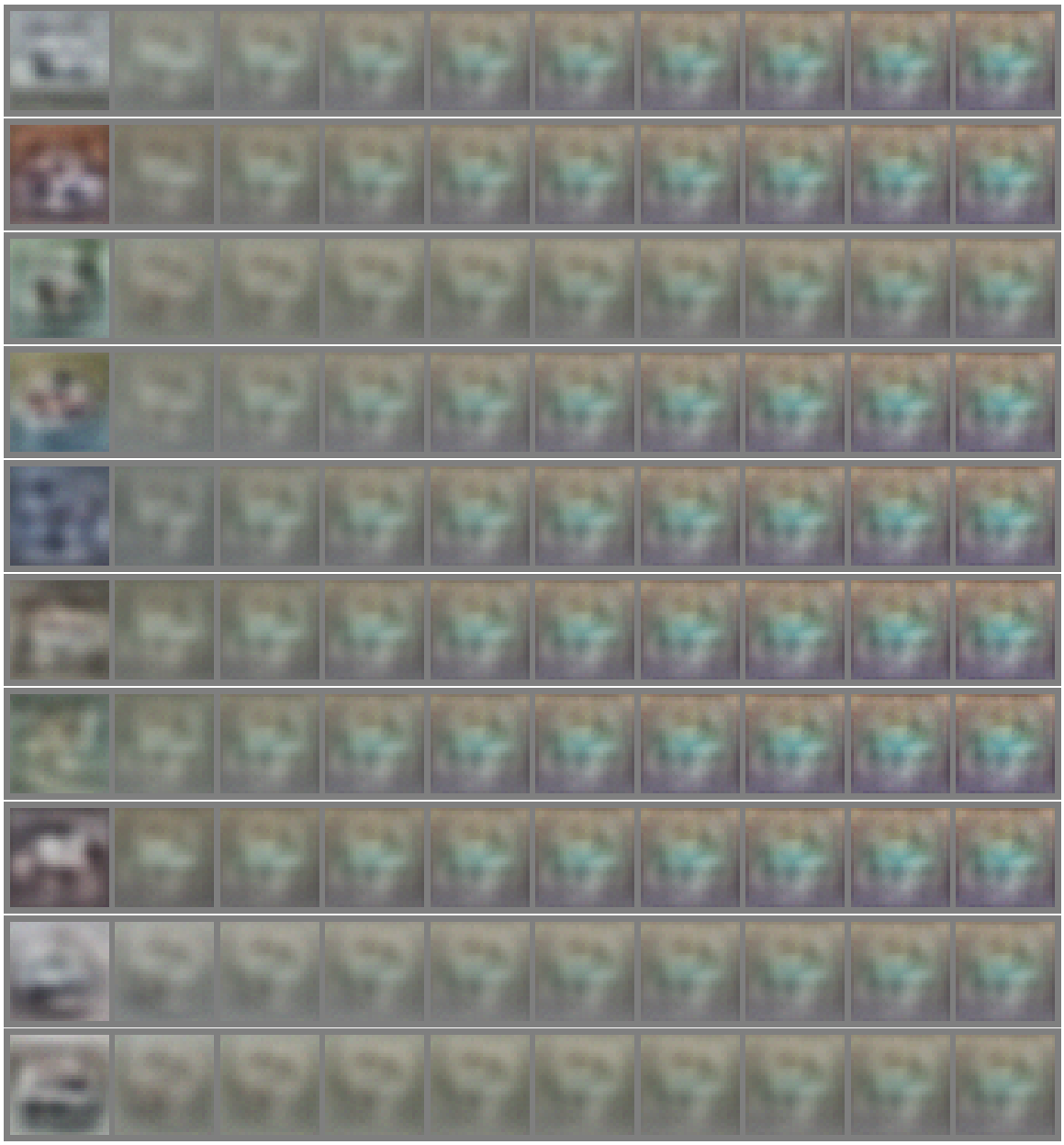


Figure B.23: Denoising process for 10 classes of CIFAR-10, generated from Model G.
Left: Step 50, **Right**: Step 100. [back]



(a) Sample A



(b) Sample B



(c) Sample C



(d) Sample D

Figure B.24: Four samples generated using the DDPM U-Net model. [back]

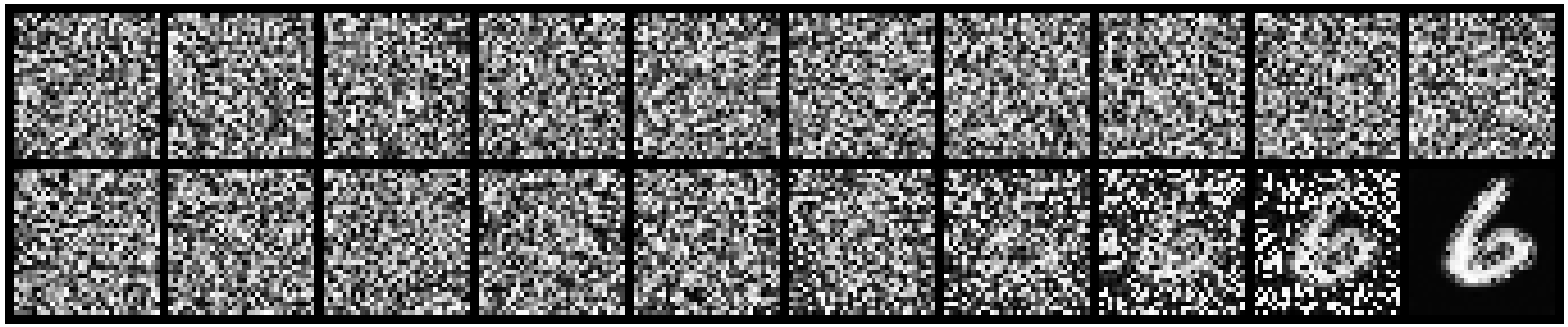


Figure B.25: Diffusion process prediction from a DDPM U-Net model. **Top Left**: State 0, **Bottom Right**: State 100. [back]

C Supplementary Code

Code C.1: Example code to generate the phase portrait from Figure 2.1

```
import numpy as np
from matplotlib import pyplot as plt

def system(x1, x2, mu, lambda_val):
    return mu * x1, lambda_val * (x2 - x1**2)

# Constants
mu = -0.5
lambda_val = -1

# Creating a grid of x1 and x2 values
x1_values = np.linspace(-2, 2, 20)
x2_values = np.linspace(-0.5, 2.5, 20)
X1, X2 = np.meshgrid(x1_values, x2_values)

# Computing the derivatives at each grid point
DX1, DX2 = system(X1, X2, mu, lambda_val)

# Normalizing the arrows for better visualization
N = np.sqrt(DX1**2 + DX2**2)
N[N == 0] = 1  # Avoid division by 0
DX1 /= N
DX2 /= N

# Plotting
plt.figure(figsize=(6, 3.5))
N = np.log(N)
cm = plt.quiver(X1, X2, DX1, DX2, N, cmap='viridis')
plt.xlabel(r'$x_1$')
plt.ylabel(r'$x_2$')
plt.title('Phase Portrait with Directional Arrows')
plt.grid(True)
plt.subplots_adjust(bottom=0.15)
plt.show()
```

Code C.2: Example code to generate trajectories in Figure 2.2

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt

def system(x1, x2, mu, lambda_val):
    """Returns the derivatives for the system."""
    return mu * x1, lambda_val * (x2 - x1**2)

mu = -0.25
lambda_val = -1
A = np.array([[mu,          0,           0],
              [0,  lambda_val, -lambda_val],
              [0,           0,      2 * mu]])

x1_values = np.linspace(-2, 2, 20)
x2_values = np.linspace(-0.5, 2.5, 20)
X1, X2 = np.meshgrid(x1_values, x2_values)
# Computing the derivatives at each grid point
DX1, DX2 = system(X1, X2, mu, lambda_val)

# Normalizing the arrows for better visualization
N = np.sqrt(DX1**2 + DX2**2)
N[N == 0] = 1  # Avoid division by 0
DX1 /= N
DX2 /= N

def differential_system(x, t, mu, lambda_val):
    x1, x2 = x
    dx1dt = mu * x1
    dx2dt = lambda_val * (x2 - x1**2)
    return [dx1dt, dx2dt]

def koopman_system(x, t, A):
    return np.dot(A, x)

t = np.linspace(0, 50, 1000)    # Time range for the simulation

initial_conditions = [[1, 1], [-0.7,2]]  # Starting values for x1 and x2

plt.figure(figsize=(6, 3.5))
cm = plt.quiver(X1, X2, DX1, DX2, N, cmap='viridis')
```

Code C.2: continued

```
for i, ic in enumerate(initial_conditions):
    solution = odeint(differential_system, ic, t, args=(mu, lambda_val))
    x1_solution, x2_solution = solution.T
    label = 'Nonlinear System' if i == 0 else '_nolegend_'
    plt.plot(x1_solution, x2_solution, 'r-', label=label)

    ick = [ic[0], ic[1], ic[0]**2]
    lin_solution = odeint(koopman_system, ick, t, args=(A,))
    label = 'Koopman System' if i == 0 else '_nolegend_'
    x1_solution, x2_solution, _ = lin_solution.T
    plt.plot(x1_solution, x2_solution, 'b--', label=label)

plt.grid(True)
plt.legend()
plt.show()
```

Code C.3: Example code to generate DMD trajectories in Figure 2.3

```
import numpy as np
from scipy.integrate import odeint
import matplotlib.pyplot as plt
from pydmd import DMD

def system(x1, x2, mu, lambda_val):
    """Returns the derivatives for the system."""
    return mu * x1, lambda_val * (x2 - x1**2)

mu = -0.25
lambda_val = -1
A = np.array([[mu,          0,           0],
              [0,  lambda_val, -lambda_val],
              [0,           0,      2 * mu]])

x1_values = np.linspace(-2, 2, 20)
x2_values = np.linspace(-0.5, 2.5, 20)
X1, X2 = np.meshgrid(x1_values, x2_values)
# Computing the derivatives at each grid point
DX1, DX2 = system(X1, X2, mu, lambda_val)

# Normalizing the arrows for better visualization
N = np.sqrt(DX1**2 + DX2**2)
N[N == 0] = 1  # Avoid division by 0
DX1 /= N
DX2 /= N

def differential_system(x, t, mu, lambda_val):
    x1, x2 = x
    dx1dt = mu * x1
    dx2dt = lambda_val * (x2 - x1**2)
    return [dx1dt, dx2dt]

t = np.linspace(0, 50, 1000)    # Time range for the simulation

initial_conditions = [[1, 1], [-0.7,2]]  # Starting values for x1 and x2

plt.figure(figsize=(6, 3.5))
cm = plt.quiver(X1, X2, DX1, DX2, N, cmap='viridis')

for i, ic in enumerate(initial_conditions):
    # plot the original system
```

Code C.3: continued

```
    solution = odeint(differential_system, ic, t, args=(mu, lambda_val))
    x1_solution, x2_solution = solution.T
    label = 'Nonlinear System' if i == 0 else '_nolegend_'
    plt.plot(x1_solution, x2_solution, 'r-', label=label)

    # plot the DMD reconstruction
    dmd = DMD(svd_rank=2)
    system = np.stack([x1_solution, x2_solution], axis=0)
    dmd.fit(system)
    label = 'DMD System' if i == 0 else '_nolegend_'
    plt.plot(dmd.reconstructed_data[0], dmd.reconstructed_data[1], 'b--', label=
    label)


plt.grid(True)
plt.legend()
plt.show()
```

Code C.4: Example code to generate the Swiss Roll dataset in Figure 5.1

```
from sklearn.datasets import make_swiss_roll
import matplotlib.pyplot as plt

dset, _ = make_swiss_roll(n_samples=1000, noise=0.05, random_state=112)
dset[:,0] /= dset[:,0].max()
dset[:,1] /= dset[:,1].max()
dset[:,2] /= dset[:,2].max()

plt.plot(dset[:, 0], dset[:, 2], 'o')
plt.savefig('swiss_roll.png', dpi=300)
```

Vita

Landon completed his Bachelor's in Computer Science at the University of Tennessee, Knoxville, in 2022. Throughout his undergraduate career, he was involved with the University's Connected and Automated Vehicles (CAVs) team through the EcoCAR program. He studied sensor fusion and object tracking in this program and its supporting research activities and met Dr. Hairong Qi. He officially started his graduate study in the Advanced Imaging and Collaborative Information Processing (AICIP) group in 2022. His topics of interest include dynamical systems, optimization, multi-object tracking, and sensor fusion.