深層表現学習を活用した時系列データ解析の革新的アプローチ

Meiji University
Graduate School of
Advanced Mathematical Sciences

Academic Year 2023

Doctoral Dissertation

Innovative Approaches to Sequential Data with Deep Representation Learning

Mathematical Sciences Program

Tsuyoshi Ishizone

Contents

1 Introduction

Artificial intelligence (AI) is a multifaceted field of computer science that aims to create systems capable of performing tasks that typically require human intelligence. These tasks include learning, reasoning, problem-solving, perception, and language understanding. AI systems are powered by algorithms, trained to recognize patterns within data and make decisions with varying levels of autonomy.

AI technology has proliferated across various domains, transforming data analysis and interaction with digital systems. AI facilitates real-time translation and natural language processing, enabling computers to understand and generate human speech. In image processing, AI excels at facial recognition and image classification tasks. Audio processing has seen AI applications in voice recognition and music composition, illustrating the technology's versatility in handling diverse data types and tasks. With the advent of ChatGPT [220] in 2022, many ordinary people will be able to enjoy the benefits of AI.

Neural networks (NNs) are computational models inspired by the human brain, and they are central to many state-of-the-art AI applications. NNs are originally developed based on the threshold logic. The initial model is called perceptron and could learn to connect or associate specific inputs to specific outputs, laying the foundation for supervised learning. From the success of AlexNet [187] in the ImageNet competition in 2012, NNs became the central technique of AI. Numerous NN architectures have been proposed for image, language, time series, and other data formats. Recently, foundation models have made it easy to create task-specific NN models for images and languages. However, there is still no foundation model for time series data.

Time-series data is a sequence of data points collected or recorded at time-ordered intervals. This type of data is ubiquitously found in domains such as finance (stock prices), meteorology (weather forecasts), health (heart rate monitoring), and more. Analyzing time-series data involves understanding patterns over time, which can be complex because the temporal dimension adds to the data's inherent variability. NNs for time-series data have led to advanced applications like predictive modeling and anomaly detection. Time-series forecasting employs NNs to predict future values based on historical data, which is essential in stock market analysis and supply chain management. Anomaly detection uses AI to identify unusual patterns that do not conform to expected behavior. It is crucial for fraud detection and maintaining operational integrity in manufacturing.

However, most time series data are stored in dead storage and have not been applied by NNs. This is due to the solid data-specific characteristics of time series data. For image data, characteristic patterns such as edges and curves are highly versatile, even if the captured objects differ. Because the syntax of a single language is limited for language data, foundation models have been developed by training a large amount of high-quality data. For time series data, not only do waveform patterns differ significantly from one data set to another, but data-specific noise, missing data, unequal intervals, and non-stationarity are also prominent. What may be considered noise for some time-series data may be considered high-frequency components or deterministic chaotic trajectories for others. It is essential to deal with the non-stationarity of time series data in the same domain or task, where dynamics fluctuate as time evolves. Non-stationarity can be considered a state in which domain shifts are constantly occurring and is difficult to handle. Because of these characteristics, time series data have been a concern for many machine learning researchers and practitioners, and many time series data remain idle.

To take advantage of dormant data, creating a framework that is easy to apply is a good idea. To create such a framework, representation learning (RL) is a set of methods that allows a system to automatically discover the representations needed for multiple downstream tasks, such as feature detection or classification from raw data. The primary goal of RL is to transform raw data into a form that is easier to interpret and process by machine learning models. For example, converting image pixel values to high-level features like edges, textures, or object parts. RL provides many benefits, such as automatic feature extraction, feature generalization, and transfer learning. Representation learning reduces the need for domain expertise and labor-intensive feature engineering by automatically learning the features. Good representations often capture underlying factors of variation in the data, leading to models that generalize better to new, unseen data. Learned representations on one task can often be transferred and used to improve performance on other related tasks.

Representation learning is currently applied in a broad field, such as natural language processing, computer vision, and reinforcement learning. In natural language processing, word embeddings like Word2Vec [249, 256] or GloVe [250] are classic examples where words are represented as dense vectors in a continuous vector space, capturing semantic meaning. In computer vision, convolutional neural networks (CNNs) such as ResNet [127] and VGGNet [218] automatically learn hierarchical representations, from edges in the early layers to complex objects in deeper layers. In reinforcement learning, representation learning helps understand and interpret complex environments, leading to better decision-making policies. In this field, world models have been developed based on representation learning. World models refer to a paradigm where an agent constructs internal models or simulations of the environment it interacts with [116]. These internal models, or "world models," allow the agent to predict the future states of the environment and the results of its actions, even without directly interacting with the real environment. The concept was inspired by how humans seem to simulate potential future scenarios before acting.

Representation learning is also helpful for time-series data. The current situation of building models for each domain and task can be improved by generalizability through representation learning. This paper focuses on deep representation learning using neural networks for time series data. Chapter 5 proposes a method for anomaly detection and period estimation using weight vectors computed inside NNs as representations. Chapter 6 proposes a learning algorithm that facilitates learning time series representations. Chapter 7 proposes a representation learning method suitable for biomolecular structural data.

This paper is organized as follows. This paper describes neural networks and representation learning, which are the basis of AI technology, in Chapters 2 and 3, respectively, and outlines representation learning for time series data in Chapter 4. Chapters 5 to 7 are dedicated to research results using time-structured representation learning. Chapter 5 proposes an anomaly detection method for quasi-periodic time-series data, such as the amount of electricity used on a factory production line. Chapter 6 proposes a learning technique for time series NNs using an ensemble Kalman filter, a nonlinear data assimilation method. In Chapter 7, we propose a method for extracting slow representations from biomolecular structural dynamics data. Finally, Chapter 8 summarizes the thesis.

1.1 Notation

Set and Space For a finite set S, $|S|$ represents the cardinality of the set. For a natural number $n \in \mathbb{N}$, $\mathbb{N}_n = \{1, \cdots, n\}$ denotes the set of natural numbers less than or equal to n. For natural numbers $q, r \in \mathbb{N}$, $q\mathbb{N} + r = \{qn + r | n \in \mathbb{N}\}$ denotes the set of natural numbers whose remainder divided by q is r. Sets $\mathbb{R}_+ = \{x \in \mathbb{R} | x > 0\}$ and $\mathbb{R}_- = \{x \in \mathbb{R} | x < 0\}$ represent the set of real numbers greater than 0, less than 0, respectively.

Vector and Matrix For a vector $\boldsymbol{v} \in \mathbb{R}^d$ and a matrix $M \in \mathbb{R}^{n \times m}$, $\boldsymbol{v}^T \in \mathbb{R}^{1 \times d}$ and $M^T \in \mathbb{R}^{m \times n}$ represents the transpose of the vector and the matrix, respectively. For a vector $\boldsymbol{v} = (v_1 \cdots v_d)^T \in \mathbb{R}^d$ and a natural number $n \in \mathbb{N}$, $\boldsymbol{v}^{\circ n} = (v_1^n \cdots v_d^n)^T$ represents the element-wise power of n. For vectors $\boldsymbol{v} = (v_1 \cdots v_d)^T$ and $\boldsymbol{u} = (u_1 \cdots u_d)^T \in \mathbb{R}^d$, $\boldsymbol{v} \odot \boldsymbol{u} = (v_1 u_1 \cdots v_d u_d)^T$ represents the element-wise product. For a vector $\boldsymbol{v} = (v_1 \cdots v_d)^T$, $\|\boldsymbol{v}\| = \sqrt{\sum_{i=1}^d v_i^2}$ and $\|\boldsymbol{v}\|_1 = \sum_{i=1}^d |v_i|$ represent the L2 norm and the L1 norm, respectively. For a vector $\boldsymbol{v} = (v_1 \cdots v_d)^T \in \mathcal{V}^d$ and a scalar function $f : \mathcal{V} \to \mathbb{R}$, $f(\boldsymbol{v}) = (f(v_1), \cdots, f(v_d))^T$ represents the element-wise mapping.

I_d and O_d denote the d-dimensional identity matrix and zero matrix, respectively. I and O denote the identity matrix and zero matrix, respectively, whose dimension depends on context.

Random Variable For a random variable $X \in \mathcal{X}$, a function $f : \mathcal{X} \to \mathbb{R}$, and a distribution $p : \mathcal{X} \to [0, \infty)$, $\mathbb{E}_{p(X)}[f(X)] \in \mathbb{R}$ represents the expectation of $f(X)$ regarding to the distribution $p(X)$. For random variables $X \in \mathcal{X}$, $Y \in \mathcal{Y}$, a function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, and a conditional distribution $p(X|Y)$, $\mathbb{E}_{p(X|Y)}[f(X, Y)] : \mathcal{X} \to \mathbb{R}$ represents the conditional expectation of $f(X, Y)$ regarding to the distribution $p(X|Y)$.

Sequence Let $X = \{\boldsymbol{x}_t\}_{t=1}^T$ be the d_x-dimensional T-length observed sequence. For a vector sequence $V = \{\boldsymbol{v}_t\}_{t=1}^T$, $v_{t,i}$ represents the i-th element of the vector $\boldsymbol{v}_t$.

2 Neural Network

Neural networks (NNs) have become indispensable tools in recent machine learning tasks and mathematical modeling. NNs were originally modeled after the structure of the human brain. The human brain comprises over 100 billion neurons, which send and receive electrical signals at their connections to achieve complex processing. The perceptron is a mathematical model of the human brain and is the basis for the NNs.

This chapter is organized as follows. Section 2.1 describes the basic structure of NNs, the feedforward neural network. Sections 2.2, 2.3, and 2.4 describe the activation function, loss function, and optimization methods that are essential for NNs, respectively. Sections 2.5, 2.6, and 2.7 describe learning rate scheduling, regularization, and error backpropagation methods as learning techniques. In the latter of the paper, Sections 2.8, 2.9, and 2.10 describe convolutional NNs, NNs for sequence data, and the attention mechanism, which are the most important architectures among NNs, respectively. Section 2.11 is devoted to hyperparameter optimization.

2.1 Feedforward Neural Network

A neural network consists of real-valued elements called units, and information propagates through each element. The model represents a mapping $h_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$ from input variables $\boldsymbol{x} \in \mathcal{X}$ to output variables $\boldsymbol{y} \in \mathcal{Y}$. For example, given the task of predicting ice cream sales based on the next day's weather forecast, the parameters $\boldsymbol{\theta}$ can be adjusted to provide a mapping from temperature, humidity, and precipitation probability as input variables to the output variable of expected ice cream sales.

In a feedforward neural network (FNN, MLP: multi-layer perceptron, FCN: fully-connected network) with n_l-th layer, the mapping $h_{\boldsymbol{\theta}}$ is a composite of Affine transformations and nonlinear transformations called activation functions:

$$\boldsymbol{u}^i = f^i(W^i \boldsymbol{u}^{i-1} + \boldsymbol{b}^i),\ \boldsymbol{u}^0 = \boldsymbol{x},\ \boldsymbol{u}^{n_l+1} = \hat{\boldsymbol{y}} \tag{2.1}$$

where $\boldsymbol{u}^i \in \mathbb{R}^{d_{h_i}}$ denotes the i-th hidden unit, $W^i \in \mathbb{R}^{d_{h_i} \times d_{h_{i-1}}}$ and $\boldsymbol{b}^i \in \mathbb{R}^{d_{h_i}}$ are Affine parameters at the i-th layer, $f^i : \mathbb{R} \to \mathbb{R}$ is the i-th activation function, $\hat{\boldsymbol{y}} \in \mathcal{Y}$ is the predicted output. The network parameter $\boldsymbol{\theta} = \{(W^i, \boldsymbol{b}^i)\}_i \in \Theta$ are learned to minimize task loss, such as mean squared loss at regression task and cross-entropy loss at classification task. Figure 2.1 shows a basic FNN architecture. The circles represent units, and the lines connecting the units represent weight vectors (information propagation). As a reminder, NN with n_l layers means n_l number of intermediate layers, n_l number of linear transformations, and $n_l - 1$ number of intermediate layers, depending on three patterns literature. Here, we proceed with the discussion assuming that the number of intermediate layers is n_l.

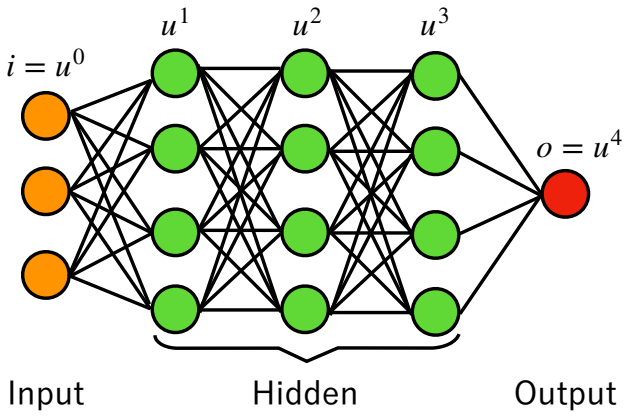


Figure 2.1: Basic architecture of neural network

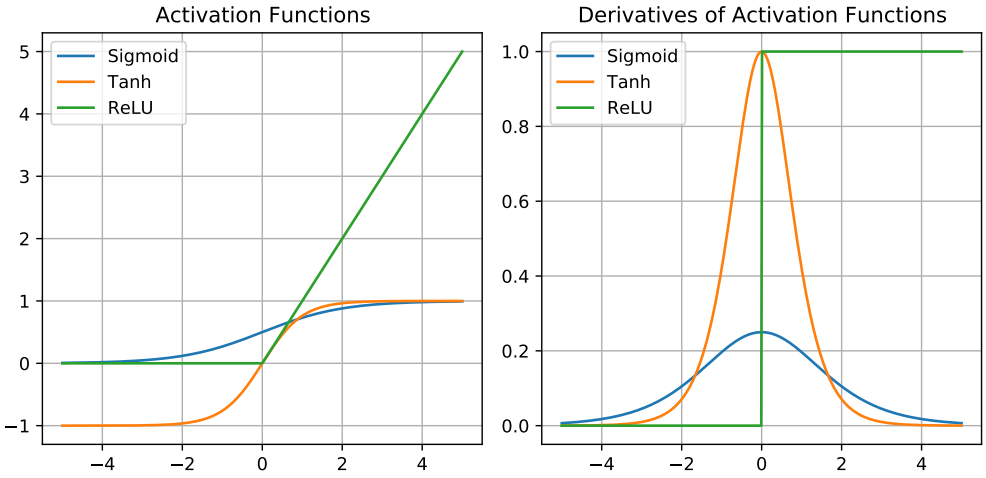


Figure 2.2: Activation functions and their derivatives

2.2 Activation Function

The following summarizes the activation functions currently commonly used in the hidden layer.

Sigmoid / Tanh unit based AFs Since human neurons are thought to take one only when they fire, the step function $\text{Step}(x) = 1\ (x \geq 0); 0\ (else)$ was used in early neural networks. To convey information other than binary values, sigmoid

$$\text{Sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

and tanh

$$\text{Tanh}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.3}$$

functions with bounded value ranges were used.

Sigmoid-weighted linear units (SiLU)

$$\text{SiLU}(x) = x \times \text{Sigmoid}(x), \tag{2.4}$$

which are bounded with respect to negative inputs (i.e., the restricted mapping to the negative domain is bounded), are widely used as an alternative to ReLU, which will be discussed later [85].

Swish

$$\text{Swish}(x) = x \times \text{Sigmoid}(\beta \times x), \tag{2.5}$$

a function that takes between linear and ReLU depending on the learning parameter β, is a frequently used activation function [297].

Rectified unit based AFs When using the sigmoid or tanh functions, there are two major problems: output saturation and gradient vanishing. Output saturation means the information that can be conveyed is limited because the value range is bounded. Gradient disappearance is the problem that causes the back-propagated gradient to become close to zero when multiple layers are stacked. To learn the parameters of a neural network, the gradients are propagated in sequence, starting with the output layer. If the gradient of the activation function is less than 1, the value of the gradient decreases exponentially with each successive layer, and the parameters are hardly updated in the layer closest to the input layer. To solve these problems, a rectified linear unit (ReLU)

$$\text{ReLU}(x) = \max\{0, x\} \tag{2.6}$$

was proposed. As shown in Figure 2.2, the ReLU's gradient is always 1 in the positive input range and is resistant to gradient vanishing [266]. However, the function does not completely eliminate gradient vanishing because the gradient is always zero for negative inputs.

To increase the variability of the derivative, leaky ReLU (LReLU) and parametric ReLU (PReLU)

$$\mathrm{LReLU}(x) = \mathrm{PReLU}(x) = p\min\{0, x\} + \max\{0, x\} \tag{2.7}$$

were proposed. These functions suppress gradient loss by providing a small slope $p \in \mathbb{R}_+$ for negative inputs [237, 128]. LReLU gives a small slope value p, such as 0.01 in advance, while PReLU treats it as a trainable parameter. The Maxout function, whose output is the maximum of several linear layers, is sometimes used as an alternative to ReLU. The function is formalized by

$$\mathrm{Maxout}(x) = \max_{i \in \mathcal{I}} x_i, \tag{2.8}$$

where $\mathcal{I}$ denote the index set and x_i is the output of i-th linear layer.

Exponential unit based AFs The exponential linear unit (ELU) mitigates ReLU's gradient vanishing problem while inheriting its saturation for negative inputs. Its function is expressed as

$$\mathrm{ELU}(x) = \begin{cases} x, & x > 0 \\ \alpha \times (e^x - 1), & x \leq 0 \end{cases}, \tag{2.9}$$

where α is the learning parameter and the range of the ELU is $[-1, \infty)$ [65]. The negative range of the function is saturated, making it more robust to noise than LReLU or PReLU.

ELU is extended with a scaling hyper-parameter λ, scaled ELU (SELU) [177]

$$\mathrm{SELU}(x) = \lambda \times \mathrm{ELU}(x). \tag{2.10}$$

The continuously differentiable ELU (CELU)

$$\mathrm{CELU}(x) = \begin{cases} x, & x > 0 \\ \alpha \times (e^{x/\alpha} - 1), & x \leq 0 \end{cases}, \tag{2.11}$$

an extension of ELU, is also used as an alternative to ELU [19].

AFs for particular tasks There are softplus and softmax functions as activation functions used for specific tasks. The softplus function

$$\mathrm{Softplus}(x) = \frac{1}{\beta} \times \log(1 + e^{\beta \times x}) \tag{2.12}$$

has a value range of $(0, \infty)$ and is used to limit the output to positive values. For example, in the variational auto-encoder (VAE) described in Section 3.3, designing a network that outputs the standard deviation and a positive value constraint is essential. The softmax function

$$\mathrm{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{2.13}$$

is used to restrict the sum to 1. For example, it is used in the output layer of a classification task or the relative weights in the attention mechanism described in Section 2.10.

2.3 Loss function

Machine learning tasks can be broadly divided into classification, regression, clustering, dimensionality reduction, etc., but they all come down to the problem of minimizing a loss function. For example, minimizing the least-squares loss in linear regression, the binary classification loss in logistic regression, the total distance from the class center in the k-means method, or the negative covariance matrix norm in principal component analysis. Since neural networks are also elements of the hypothesis set $\{h_\theta : \mathcal{X} \to \mathcal{Y} | \theta \in \Theta\}$, it is necessary to provide a loss function that depends on the parameter θ. In this section, we introduce commonly used loss functions for each task.

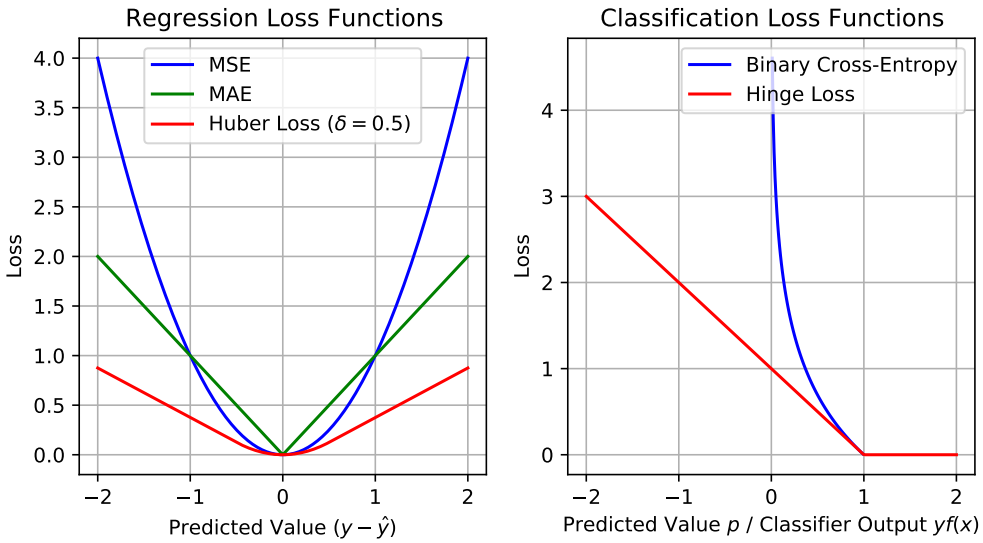


Figure 2.3: Loss functions

Regression Figure 2.3 shows representative loss functions for regression and classification tasks. Commonly used loss functions for a regression task are as follows:

- The mean squared error (MSE) is given by

$$\text{MSE}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_n (y_n - \hat{y}_n)^2, \tag{2.14}$$

  where y_n and $\hat{y}_n$ are the n-th element of ground-truth and predicted values, respectively.

- The mean absolute error (MAE) is given by

$$\text{MAE}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_n |y_n - \hat{y}_n|. \tag{2.15}$$

- The Huber loss is given by

$$\text{Huber}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_n \begin{cases} 0.5(y_n - \hat{y}_n)^2, & |y_n - \hat{y}_n| < \delta \\ \delta \times (|y_n - \hat{y}_n| - 0.5\delta), & |y_n - \hat{y}_n| \geq \delta \end{cases}, \tag{2.16}$$

  where δ controls the threshold between MSE and MAE.

Classification Classification tasks also have a set of commonly used loss functions as follows:

- The binary cross-entropy (BCE) used for binary classification tasks measures the difference between the true labels and the predicted probabilities by

$$\text{BCE}(y, p) = -y \log p - (1 - y) \log(1 - p), \tag{2.17}$$

  where $y \in \{0, 1\}$ is the true label and $p \in (0, 1)$ is the predicted probability of class 1.

- The categorical cross-entropy (CE) used for multi-class classification tasks is an extension of the binary cross-entropy for more than two classes:

$$\text{CE}(\boldsymbol{y}, \boldsymbol{p}) = -\sum_{i=1}^{C} y_i \log p_i, \tag{2.18}$$

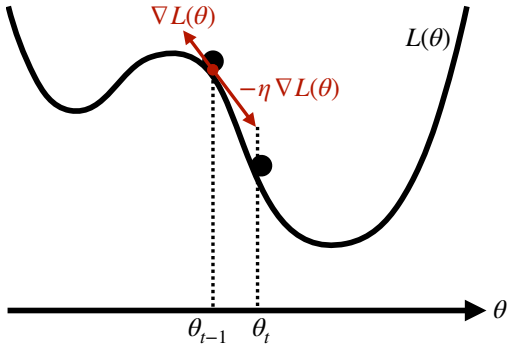


Figure 2.4: Gradient descent method

where $\boldsymbol{y} \in \{0,1\}^C$ is the one-hot encoded true label vector and $\boldsymbol{p} \in \Delta^{C-1} = \{\boldsymbol{p} \in [0,1]^C | \sum_{i=1}^C p_i = 1\}$ is the predicted probability vector.

- The hinge loss used primarily with support vector machines (SVM) for binary classification is given by

$$\text{Hinge}(y, f(x)) = \max\{0, 1 - yf(x)\}, \tag{2.19}$$

where $y \in \{-1, 1\}$ is the true label and $f(x) \in \mathbb{R}$ is the raw output of the classifier.

- The softmax loss is essentially the CE loss applied after a softmax function in multi-class classification.

2.4 Optimization

The neural network learning problem is formulated as a minimization of the expected loss problem

$$\min_{\boldsymbol{\theta} \in \Theta} \mathcal{L}(\mathcal{X}, \mathcal{Y}; \boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y}) \sim p_{\mathcal{X}\times\mathcal{Y}}}[l(\boldsymbol{y}, h_{\boldsymbol{\theta}}(\boldsymbol{x}))], \tag{2.20}$$

where $p_{\mathcal{X}\times\mathcal{Y}}$ is the true joint distribution, $h_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$ is a neural network with parameter $\boldsymbol{\theta} \in \Theta$ and $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a loss function. In statistics and machine learning, the problem of minimizing an empirical loss

$$\min_{\boldsymbol{\theta} \in \Theta} L(X, Y; \boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y}) \sim p_{(X,Y)}}[l(\boldsymbol{y}, h_{\boldsymbol{\theta}}(\boldsymbol{x}))] = \frac{1}{N} \sum_{i=1}^N l(\boldsymbol{y}_i, h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) \tag{2.21}$$

is generally solved as an alternative to the expected loss. Here, $X = \{\boldsymbol{x}_i\}_{i=1}^N$, $Y = \{\boldsymbol{y}_i\}_{i=1}^N$, $p_{(X,Y)}$ is the empirical joint distribution, $\boldsymbol{x}_i \in \mathcal{X}$ and $\boldsymbol{y}_i \in \mathcal{Y}$ are the i-th input variable and output variable, respectively.

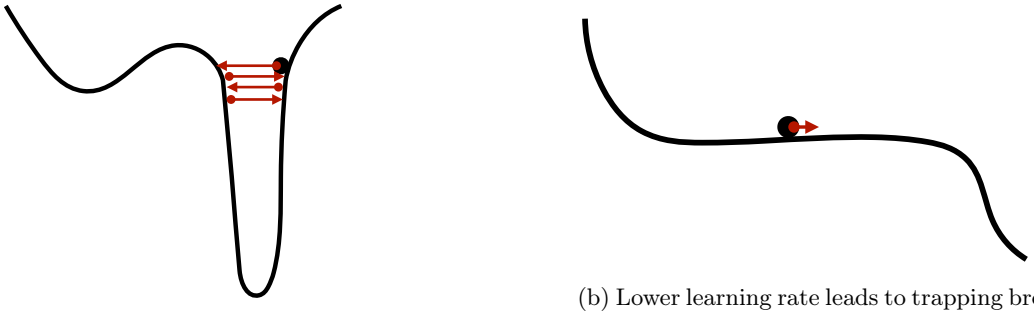
Since the optimization function $L(X, Y; \cdot) : \Theta \to \mathbb{R}$ is nonconvex with respect to the parameters $\boldsymbol{\theta}$, it is impossible to find an explicit solution. Instead, a method based on the gradient descent method is used, in which the parameters are repeatedly updated to decrease the objective value. This section outlines the gradient descent method and then describes its development.

2.4.1 Gradient Descent Method

The gradient descent (GD) method starts with an initial parameter $\boldsymbol{\theta} \in \Theta$ and repeatedly updates at a constant learning rate $\eta \in \mathbb{R}_+$ in the direction of the negative gradient:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta \nabla L(\boldsymbol{\theta}), \tag{2.22}$$

$$L(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^N l(\boldsymbol{y}_i, h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)), \tag{2.23}$$



(a) Higher learning rate lead to vibration

(b) Lower learning rate leads to trapping broader critical points

Figure 2.5: Bad cases of inappropriate learning rate

where $l : \mathcal{X} \to \mathcal{Y}$ is a loss function (Figure 2.4).

The gradient descent method requires an appropriate learning rate η. A large learning rate causes oscillations near the valleys of the loss landscape, while a small learning rate tends to trap the local optimum solution (Figure 2.5). The optimal learning rate depends on the case and should be explored in conjunction with other hyperparameters.

2.4.2 Stochastic Gradient Descent Method

If a gradient descent method falls into a large critical point, it will not update. This is also related to the fact that neural networks require a huge number of parameters. The parameter dimension of a neural network is generally several thousand or more, and for GPT-4 [277], which is used in language models, it is approximately 100 trillion. With such a huge parameter space, there are countless critical points, and it is easier to fall into a critical point than in other machine learning problems.

To ease the trap of critical points, the stochastic gradient descent (SGD) method, which limits the number of samples trained at each step, is used in neural network training. The set of samples $\mathcal{B}_t$ used for the t-th training step is called a mini-batch, and the mini-batch set is a subset of the index set of total training samples $\mathbb{N}_N$. Using mini-batches, the updated formula of the SGD method is

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta \nabla L_t(\boldsymbol{\theta}), \tag{2.24}$$

$$L_t(\boldsymbol{\theta}) = \frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} l(\boldsymbol{y}_i, h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)). \tag{2.25}$$

In general, neural network learning is hierarchically organized into epochs and iterations. The sample size N is equally divided so that the concentration of mini-batch sets is b (only the last mini-batch set can have a size less than b), and the training unit using each mini-batch set is called iteration. One equal division produces $\lfloor (N-1)/b \rfloor + 1$ iterations, and the training unit consisting of these iterations is called an epoch. Each epoch is a random, minibatch-size non-return extraction from the sample set. Once all samples have been extracted, the next epoch is started. The neural network is trained by pre-setting the mini-batch size and the number of epochs.

Since all the following methods are based on SGD, the empirical loss is described in L without dependence on t, unless otherwise noted. Parameter updates are denoted by

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{\boldsymbol{t}} + \Delta \boldsymbol{\theta}_t, \tag{2.26}$$

and only the difference $\Delta \boldsymbol{\theta}_t$ is described.

2.4.3 Momentum Method

The aim of the momentum method is to suppress oscillations around steep valleys in the loss landscape. The method suppresses oscillations in which the slope alternates between positive and negative values by taking

over the value of the slope from one period earlier:

$$\Delta\boldsymbol{\theta}_t = \mu\Delta\boldsymbol{\theta}_{t-1} - (1-\mu)\eta\nabla L(\boldsymbol{\theta}), \tag{2.27}$$

where $\mu \in [0, 1)$ is a momentum which commonly set to be $0.5 \sim 0.99$.

Nesterov's accelerated gradient method [269] is a modified version of the momentum method that changes the position at which the gradient value is evaluated:

$$\Delta\boldsymbol{\theta}_t = \mu\Delta\boldsymbol{\theta}_{t-1} - (1-\mu)\eta\nabla L(\boldsymbol{\theta} + \mu\boldsymbol{\theta}_{t-1}). \tag{2.28}$$

2.4.4 AdaGrad

The GD method uses the same learning coefficients for all coordinate directions and cannot handle the case where landscape sensitivities differ for each parameter. The adaptive subgradient method (AdaGrad) [83] addresses this problem by adaptively selecting different learning coefficients for each coordinate. The update rule of AdaGrad is

$$\Delta\theta_{t,i} = -\frac{\eta}{\sqrt{\sum_{s=1}^{t}(\nabla L(\boldsymbol{\theta}_{\boldsymbol{s}})_i)^2}}\nabla L(\boldsymbol{\theta}), \tag{2.29}$$

where $\theta_{t,i}$ is the i-th element of parameter vector at t-th training iteration. The method adaptively washes the learning rate according to the sum of past gradient changes for each coordinate, with the effect of increasing the learning rate in the direction of fewer updates.

2.4.5 RMSProp

AdaGrad is sensitive to initial values because the coordinates are rarely updated once a large gradient is recorded. To solve this problem, RMSProp [352] uses an exponential moving average of past updates:

$$\Delta\theta_{t,i} = -\frac{\eta}{\sqrt{v_{t,i}+\varepsilon}}\nabla L(\boldsymbol{\theta}_t)_i, \tag{2.30a}$$

$$v_{t,i} = \rho v_{t-1,i} + (1-\rho)(\nabla L(\boldsymbol{\theta}_t)_i)^2, \tag{2.30b}$$

where $v_{0,i} = 0$, $\rho \in (0, 1)$ is decaying rate, and small $\varepsilon > 0$ suppress divergence.

2.4.6 AdaDelta

RMSProp has the same sensitivity to learning rate as AdaGrad. A reason of the sensitivity is the physical dimensionality mismatch between $\Delta\boldsymbol{\theta}$ and ∇L. If the parameters have a *length* scale, the loss function l should be a dimensionless length-independent quantity. In AdaGrad and RMSProp, the left-hand side of the update $\Delta\boldsymbol{\theta} \sim length$, while the right-hand side of the update is a dimensionless quantity, resulting in a dimension mismatch.

AdaDelta [411] uses an inverse Hessian approximation to address this issue. In contrast to the gradient descent method, which uses first-order derivatives, the Newton method uses the Hessian, which is a second-order derivative, and the update can be written as

$$\Delta\boldsymbol{\theta} = H(\boldsymbol{\theta})^{-1}\nabla L(\boldsymbol{\theta}), \tag{2.31}$$

where the right-hand side of this equation has a *length* scale. If the Hessian is approximated by the diagonal matrix, the Hessian inverse can be approximated by

$$H_{ii}^{-1} = \frac{\Delta\theta_i}{\nabla L(\boldsymbol{\theta})_i}. \tag{2.32}$$

Using this equation, AdaDelta can be formulated as

$$\Delta\theta_{t,i} = -\frac{\sqrt{u_{t-1,i}+\varepsilon}}{\sqrt{v_{t,i}+\varepsilon}}\nabla L(\boldsymbol{\theta}_t)_i, \tag{2.33a}$$

$$u_{t,i} = \rho u_{t-1,i} + (1-\rho)(\Delta\theta_{t,i})^2, \tag{2.33b}$$
$$v_{t,i} = \rho v_{t-1,i} + (1-\rho)(\nabla L(\boldsymbol{\theta}_t)_i)^2. \tag{2.33c}$$

2.4.7 SMORMS3

SMORMS3 [99] is a hybrid method of RMSProp and vSGD [322], which is a method that calculates optimal learning coefficients based on curvature information (second-order derivatives) around the current parameters. The updated formula of SMORMS3 is

$$\Delta\theta_{t,i} = -\frac{\min\{\eta, \zeta_{t,i}\}}{\sqrt{\hat{v}_{t,i}+\varepsilon}}, \tag{2.34a}$$
$$\zeta_{t,i} = \frac{m_{t,i}^2}{v_{t,i}+\varepsilon}, \tag{2.34b}$$
$$m_{t,i} = \rho_{t,i} m_{t-1,i} + (1-\rho_{t,i})\nabla L(\boldsymbol{\theta}_t)_i, \tag{2.34c}$$
$$v_{t,i} = \rho_{t,i} v_{t-1,i} + (1-\rho_{t,i})(\nabla L(\boldsymbol{\theta}_t)_i)^2, \tag{2.34d}$$
$$\rho_{t,i} = \frac{1}{1+s_{t,i}}, \tag{2.34e}$$
$$s_{t,i} = 1 + \zeta_{t-1,i} s_{t-1,i}. \tag{2.34f}$$

2.4.8 Adam

Adam [173] is a method that takes an exponential moving average of the gradient as well as the square of the gradient:

$$m_{t,i} = \beta_1 m_{t-1,i} + (1-\beta_1)\nabla L(\boldsymbol{\theta}_t)_i, \tag{2.35a}$$
$$v_{t,i} = \beta_2 v_{t-1,i} + (1-\beta_2)(\nabla L(\boldsymbol{\theta}_t)_i)^2, \tag{2.35b}$$
$$\hat{m}_{t,i} = \frac{m_{t,i}}{1-\beta_1^t}, \tag{2.35c}$$
$$\hat{v}_{t,i} = \frac{v_{t,i}}{1-\beta_2^t}, \tag{2.35d}$$
$$\Delta\theta_{t,i} = -\eta\frac{\hat{m}_{t,i}}{\sqrt{\hat{v}_{t,i}+\varepsilon}}, \tag{2.35e}$$

where $\beta_1, \beta_2 \in (0,1)$ are decaying rate which is commonly set to $\beta_1 = 0.9$, $\beta_2 = 0.999$.

The $\hat{m}_{t,i}$ and $\hat{v}_{t,i}$ are unbiased estimators of the gradient and the square of the gradient, respectively, when the gradient follows a stationary distribution. In fact,

$$\begin{aligned}\mathbb{E}[\hat{v}_{t,i}] &= \frac{1}{1-\beta_2^t}(1-\beta_2)\sum_{s=1}^{t}(\beta_2)^{t-s}\mathbb{E}\left[(\nabla L(\boldsymbol{\theta}_s)_i)^2\right] \\ &= \mathbb{E}\left[(\nabla L(\boldsymbol{\theta}_t)_i)^2\right].\end{aligned} \tag{2.36}$$

Adam is currently one of the most popular optimization methods, with many proposed variants. AdaMax [425] extends Adam to the infinite-dimensional norm, Nadam [81] introduces Nesterov's accelerated gradient method to Adam, RAdam [217] corrects the moving average of the initial moments, AdamW [228, 229] introduces a weight decay into Adam, and Eve [125] updates based on relative changes in gradient.

2.4.9 AdaSecant

AdaSecant [40] is a method that uses curvature information, exploiting the fact that the secant approximation of the gradient is an approximation of the Hessian. It is a probabilistic rank-1 quasi-Newton method, and

the updated law is

$$g_{t,i} = \nabla L(\boldsymbol{\theta}_t)_i, \tag{2.37a}$$
$$m_{t,i} = \beta_e m_{t-1,i} + (1-\beta_e) g_{t,i}, \tag{2.37b}$$
$$v_{t,i} = \beta_e v_{t-1,i} + (1-\beta_e)(g_{t,i})^2, \tag{2.37c}$$
$$\zeta_{t,i} = \frac{1}{\tau_{t,i}}, \tag{2.37d}$$
$$n_{t,i} = (1-\zeta_{t-1,i}) n_{t-1,i} + \zeta_{t-1,i}\{(g_{t,i} - \tilde{g}_{t-1,i})(\tilde{g}_{t-1,i} - m_{t,i})\}^2, \tag{2.37e}$$
$$d_{t,i} = (1-\zeta_{t-1,i}) d_{t-1,i} + \zeta_{t-1,i}\{(g_{t,i} - m_{t,i})(\tilde{g}_{t-1,i} - m_{t,i})\}^2, \tag{2.37f}$$
$$\gamma_{t,i} = \min\left\{ \frac{\sqrt{n_{t,i}}}{\sqrt{d_{t,i}+\varepsilon}}, \gamma^M \right\}, \tag{2.37g}$$
$$\tilde{g}_{t,i} = \beta_c \frac{g_{t,i} + \gamma_{t,i} m_{t,i}}{1+\gamma_{t,i}} + (1-\beta_c) g_{t,i}, \tag{2.37h}$$
$$s_{t,i} = g_{t,i} - g_{t-1,i}, \tag{2.37i}$$
$$c_{t,i} = (1-\zeta_{t-1,i}) c_{t-1,i} + \zeta_{t-1,i} s_{t,i}, \tag{2.37j}$$
$$c^s_{t,i} = (1-\zeta_{t-1,i}) c^s_{t-1,i} + \zeta_{t-1,i} (s_{t,i})^2, \tag{2.37k}$$
$$\Delta\theta_{t,i} = -\eta \left(\frac{\sqrt{\bar{\Delta}^s_{t,i} + \varepsilon}}{\sqrt{c^s_{t,i}+\varepsilon}} - \frac{\nu_{t-1,i}}{c^s_{t,i}+\varepsilon} \right), \tag{2.37l}$$
$$\tau_{t,i} = \mathrm{clop}\left\{ \left\{ 1 - \frac{(\bar{\Delta}_{t,i})^2}{\bar{\Delta}^s_{t,i}} \right\} \tau_{t-1,i} + 1 + \varepsilon, \tau^m, \tau^M \right\}, \tag{2.37m}$$
$$\bar{\Delta}_{t,i} = (1-\zeta_{t,i}) \bar{\Delta}_{t-1,i} + \zeta_{t,i} \Delta\theta_{t,i}, \tag{2.37n}$$
$$\bar{\Delta}^s_{t,i} = (1-\zeta_{t,i}) \bar{\Delta}^s_{t-1,i} + \zeta_{t,i} (\Delta\theta_{t,i})^2, \tag{2.37o}$$
$$\bar{\nu}_{t,i} = (1-\zeta_{t,i}) \nu_{t-1,i} + \zeta_{t,i} s_{t,i}, \tag{2.37p}$$

where $\beta_e, \beta_c \in (0,1)$.

2.4.10 AMSGrad

The exponential moving average used by Adam and others has the problem that when the gradient information from a particular sample is important, the gradient in that mini-batch decays quickly and does not converge to the optimal solution. AMSGrad [303] addresses this problem by retaining the maximum value of the exponential moving average of the squared gradient:

$$m_{t,i} = \beta_{t,1} m_{t-1,i} + (1-\beta_{t,1}) \nabla L(\boldsymbol{\theta}_t)_i, \tag{2.38a}$$
$$v_{t,i} = \beta_2 v_{t-1,i} + (1-\beta_2)(\nabla L(\boldsymbol{\theta}_t)_i)^2, \tag{2.38b}$$
$$\hat{v}_{t,i} = \max\{\hat{v}_{t-1,i}, v_{t,i}\}, \tag{2.38c}$$
$$\Delta\theta_{t,i} = -\alpha_t \frac{m_{t,i}}{\sqrt{\hat{v}_{t,i}}}, \tag{2.38d}$$
$$\boldsymbol{\theta}_t = \Pi_{\mathcal{F}, \mathrm{diag}(\sqrt{\boldsymbol{v_t}})}(\boldsymbol{\theta}_{t-1} + \Delta\boldsymbol{\theta}_t), \tag{2.38e}$$

where a feasible set $\mathcal{F}$ has bounded diameter D_∞ if $\|\boldsymbol{x}-\boldsymbol{y}\|_\infty \leq D_\infty$ for all $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{F}$, the projection operator $\Pi_{\mathcal{F},A}(\boldsymbol{y})$ for a positive definite matrix A is defined as

$$\Pi_{\mathcal{F},A}(\boldsymbol{y}) = \operatorname*{arg\,min}_{x \in \mathcal{F}} \|A^{1/2}(\boldsymbol{x}-\boldsymbol{y})^2\|. \tag{2.39}$$

AMSGrad's projection is based on the concept of the proximity gradient method, which suppresses abrupt changes.

2.4.11 AdaBound, AMSBound

AMSGrad can improve the problem of unnecessarily large learning rates, but it cannot improve the problem of unnecessarily small learning rates. AdaBound and AMSBound [230] solve this problem by behaving like Adam in the early stages and SGD in the late stages. Adam's adaptive learning rate allows it to learn quickly but fluctuates wildly even after sufficient progress has been made and generalization errors are not suppressed. On the other hand, SGD can keep the final generalization error low, but it learns too slowly. AdaBound and AMSBound achieve both fast learning and low generalization error by changing from Adam to SGD and AMSGrad to SGD during the learning process.

The updated rule of AdaBound is

$$m_{t,i} = \beta_{t,1} m_{t-1,i} + (1 - \beta_{t,1}) \nabla L(\boldsymbol{\theta}_t)_i, \tag{2.40a}$$

$$v_{t,i} = \beta_2 v_{t-1,i} + (1 - \beta_2)(\nabla L(\boldsymbol{\theta}_t)_i)^2, \tag{2.40b}$$

$$\hat{\boldsymbol{\eta}}_t = \mathrm{Clip}\left\{\frac{\alpha}{\boldsymbol{v}_t}, \eta_l(t), \eta_u(t)\right\}, \tag{2.40c}$$

$$\boldsymbol{\eta}_t = \frac{\hat{\boldsymbol{\eta}}_t}{\sqrt{t}}, \tag{2.40d}$$

$$\boldsymbol{\theta}_t = \Pi_{\mathcal{F}, \mathrm{diag}(\boldsymbol{\eta}_t^{-1})}(\boldsymbol{\theta}_{t-1} - \boldsymbol{\eta}_t \odot \boldsymbol{m}_t), \tag{2.40e}$$

where $\eta_l : \mathbb{N} \to [0, \alpha^*]$ is a non-decreasing function that starts from 0 and converges to α^* asymptotically, $\eta_u : \mathbb{N} \to [\alpha^*, \infty]$ is a non-increasing function that starts from ∞ and converges to α^*. The update rule for AMSBound is simply to replace $\boldsymbol{v}_t$ with $\hat{\boldsymbol{v}}_t$. AdaBound is currently one of the most popular optimization methods after Adam.

2.5 Learning Rate Scheduling

Optimization methods such as Adam and AdaBound can adaptively determine the learning rate for each parameter, but the appropriate hyperparameters may differ in the early, middle, and late stages of learning. Scheduling the learning coefficient *eta* for each learning stage allows fast learning while avoiding traps to saddle points.

Decay Schedulers The decay schedulers learn quickly initially and gradually reduce the learning rate to suppress parameter oscillations. A step scheduler reduces the learning rate at specific intervals:

$$\eta = \eta_0 \times \alpha^{\lfloor t/I \rfloor}, \tag{2.41}$$

where $\eta_0 \in \mathbb{R}_+$ is the initial learning rate, t is the current training step, and I is the reducing interval.

Linear, exponential, and polynomial schedulers decrease the learning rate continuously by

$$\eta = \eta_0 - \alpha \times t, \tag{2.42}$$

$$\eta = \eta_0 \times e^{-k \times t}, \tag{2.43}$$

$$\eta = \eta_0 \times \left(1 - \frac{t}{T}\right)^p, \tag{2.44}$$

respectively, where T is the number of training steps. Figure 2.6(a) shows these schedulings.

Warmup Schedulers Neural network training is often strongly influenced by the initial parameters. In addition, optimization methods that use past gradient updates are strongly influenced by the parameter gradients in the initial learning phase. Warmup schedulers slowly bring the initial parameters to good initial values before learning begins. Linear warmup increases the learning rate linearly from an initial value to a target value over a specified number of training steps:

$$\eta = \eta_0 + (\eta_w - \eta_0) \times \min\left\{\frac{t}{t_w}, 1\right\}, \tag{2.45}$$

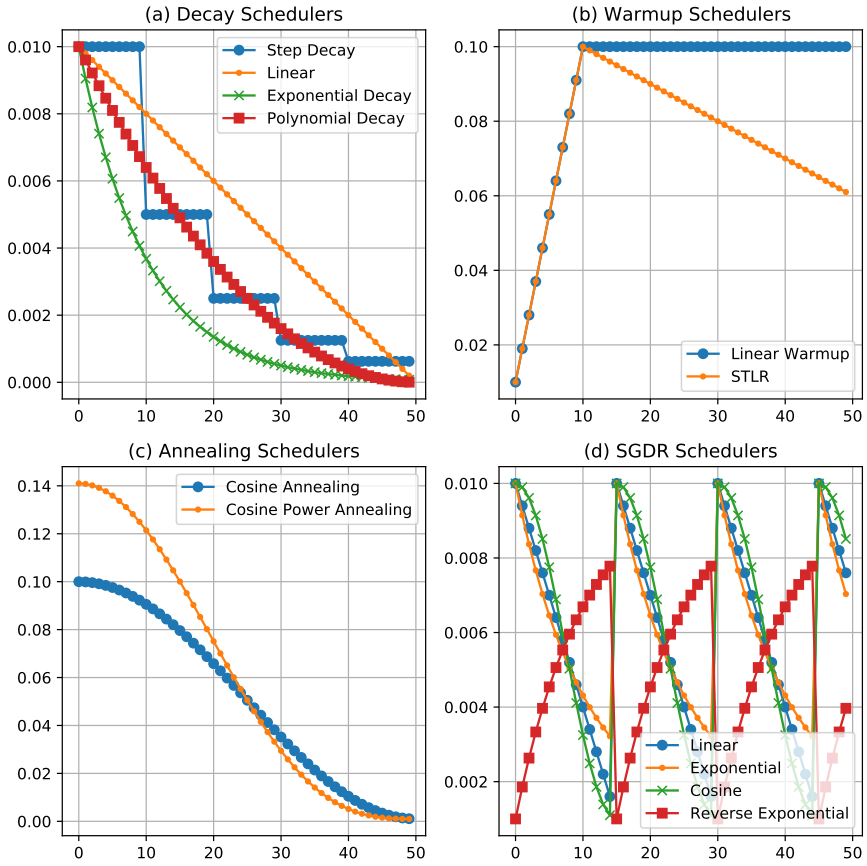


Figure 2.6: Learning rate schedulers

where η_w and t_w represent the target learning rate and the number of warmup steps, respectively.

Slanted Triangular Learning Rates (STLR) [139] first linearly increase the learning rate and then linearly decay it:

$$\eta = \eta_0 + (\eta_w - \eta_0) \times \min\left\{\frac{t}{t_w}, 1\right\} - \alpha \times \max\{t_w - t, 0\}. \tag{2.46}$$

Figure 2.6(b) shows warmup schedulers.

Annealing Schedulers Similar to decay schedulers, annealing schedulers gradually decrease the learning coefficients. Cosine annealing anneals the learning rate by

$$\eta = \eta_m + \frac{1}{2}(\eta_M - \eta_m) \times \left\{1 + \cos\left(\pi \times \frac{t}{T}\right)\right\}, \tag{2.47}$$

where η_m and η_M represent the learning rates' minimum and maximum, respectively.

Cosine power annealing [145] raise cosine function to some power by

$$\eta = \eta_m + (\eta_M - \eta_m) \times \left\{\frac{1 + \cos\left(\pi \times \frac{t}{T}\right)}{2}\right\}^p. \tag{2.48}$$

These schedulers are shown in Figure 2.6(c).

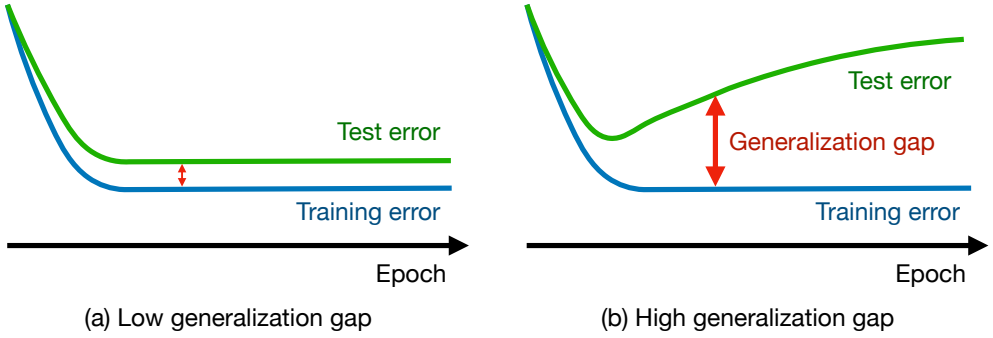


Figure 2.7: Examples of learning curves

Stochastic Gradient Descent with Warm Resterts Introducing a decay scheduler requires new hyperparameters, which need to be adjusted. If the learning rate decays quickly, the system is stuck in a bad local optimum, while if it decays slowly, learning becomes unstable. Stochastic Gradient Descent with Warm Restarts (SGDR) [227] addresses this problem by repeatedly decaying and restarting the learning rate. The scheduler was originally proposed as a form of cosine annealing

$$\eta = \eta_m^i + \frac{1}{2}(\eta_M^i - \eta_m^i) \times \left\{1 + \cos\left(\pi \times \frac{R_t}{T_i}\right)\right\}, \tag{2.49}$$

where η_m^i and η_M^i are ranges for the i-th learning rate, R_t is the number of training steps performed since the last restart, T_i is the i-th number of training steps. The scheduler decays the learning rate by T_i steps and then restarts the learning rate to the next η_M^{i+1}. This form of SGDR is also referred to simply as cosine annealing. Figure 2.6(d) shows this cosine SGDR and other decay schedulers with restart.

2.6 Regularization

SGD methods consider minimizing the empirical loss, which is a Monte Carlo approximation of the expected loss for a population. In the infinite limit of the number of training samples, where the training samples follow the population distribution i.i.d., minimizing the empirical loss asymptotically approaches minimizing the expected loss, but in general, it suffers from sampling bias with a finite training sample. The empirical loss for the training sample is called the training error, the empirical loss for the test sample is called the test error, and the expected loss for the population is called the generalization error.

Machine learning aims to reduce the generalization error, and the training error is suppressed from above by the generalization error. The difference between the generalization and training errors is called the generalization gap, and a large generalization gap is called overfitting or overlearning. Figure 2.7 shows examples of training and testing learning curves. The training error and the generalization gap must be suppressed simultaneously to reduce the generalization error.

2.6.1 Double Descent

In classical statistical machine learning, a bias-variance trade-off is said to hold between the number of model parameters and the generalization error. The generalization error is decomposed into bias and variance due to the training sample and the model's representativeness. An increase in the number of parameters of the model implies an expansion of the function space that the model can represent, which lowers the bias, which is the difference between the best error for an arbitrary measurable function and the best error for the measurable functions that the model can represent. On the other hand, increasing the representability of the model increases the dependence of the empirical loss minimization model on the training sample and raises the variance. There is an optimal number of parameters around which the generalization error is said to increase.

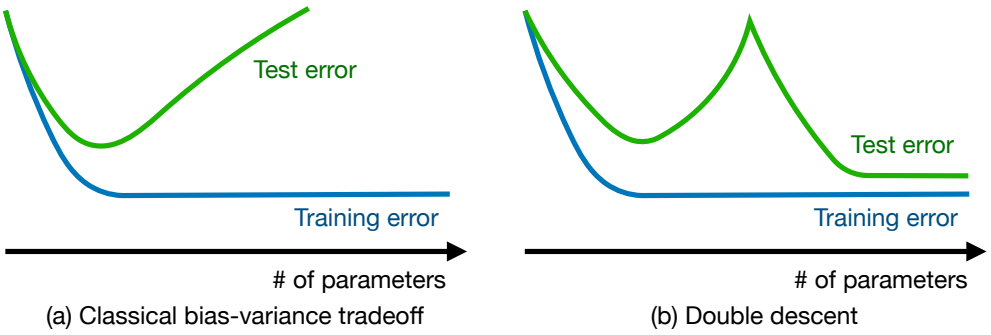


Figure 2.8: Double descent

This trade-off is often not valid in the case of neural networks. When the size of the network is small, it behaves in the same way as the bias-variance trade-off, but when it exceeds a certain threshold, the generalization error is said to start decreasing (Figure 2.8) [267, 26, 25]. This phenomenon is called double descent and has been proven in machine learning methods such as linear regression and nonnegative matrix factorization. This is supported by the development of today's large-scale infrastructure models that require trillions to 100 trillion parameters.

2.6.2 Early Stopping

Early stopping suppresses generalization error by taking advantage of the upward trend in test error that occurs when overlearning occurs [287]. Specifically, training is terminated if the test error does not improve for a set number of epochs in a row as patience. This regularization method is often used due to its simplicity of implementation cost and usefulness.

2.6.3 Weight Decay

The basis of regularization is to penalize the loss function for the values of its parameters. The method with L2 penalties commonly used in machine learning

$$L(\boldsymbol{\theta}) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} l(\boldsymbol{y}_i, h_{\boldsymbol{\theta}}(\boldsymbol{x}_i)) + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 \tag{2.50}$$

is called weight decay in the context of neural networks. In general, weight decay applies only to the weight matrix $\{W^j\}$ that governs the space expansion and contraction, not to the bias vectors $\{\boldsymbol{b}^j\}$. In neural networks, the effectiveness of the L2 penalty has been questioned due to double descent, and the debate continues [417].

2.6.4 Dropout

Dropout [338] randomly disables units in the network with probability $p \in [0, 1)$ at training time. The operation in the j-th layer of the NN is

$$\boldsymbol{u}^j = W^j(\boldsymbol{m}^j \odot \boldsymbol{u}^{j-1}) + \boldsymbol{b}^j, \tag{2.51}$$

where the i-th element of the j-th mask vector $m_i^j \sim Be(p)$ follows a Bernoulli distribution $Be(p)$ (Figure 2.9(a)). By randomly sampling the mask variable every iteration, the network's degrees of freedom are reduced during training to suppress overfitting. It also mitigates the problem of training bias toward a particular unit and improves inference accuracy by viewing the network as an ensemble of multiple networks.

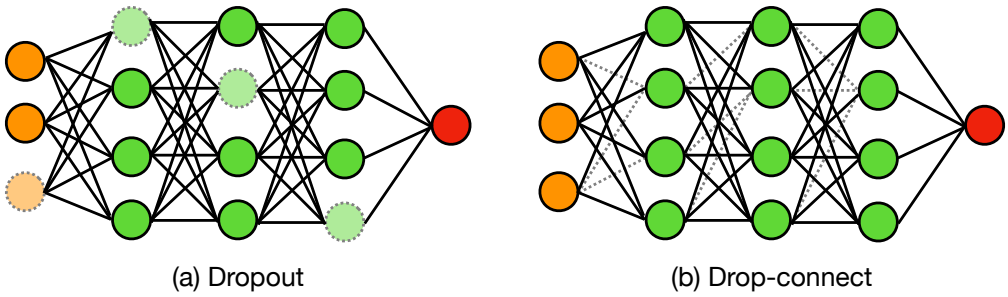


Figure 2.9: Dropout and drop-connect

Dropout uses all units with p times the joint weights during inference. MC-dropout [101] also stochastically invalidates the units during inference so that it can be viewed as a Monte Carlo average of the predictions of multiple networks, allowing inference to take into account uncertainty. Similar methods to Dropout include drop-connect (Figure 2.9(b))[369], which randomly disables edges, and stochastic maximum pooling [412], which pools stochastically.

2.6.5 Batch Normalization

Machine learning often suffers from the covariate shift problem, where data distribution diverges between training and inference. In neural networks, there is the problem of internal covariate shift, in which covariate shift occurs between layers. Since the input vectors follow a generative distribution $p(\boldsymbol{x})$, the output vectors in the intermediate layers also follow the generative distribution $p(\boldsymbol{u}^j)$. Each generative distribution in the intermediate layers is updated to fit the data, but they interfere with each other, making it difficult to learn each generative distribution.

Batch normalization (BN) [152] addresses this problem by normalizing the outputs in each layer and binding them to follow a particular generative distribution. For the middle output $\boldsymbol{u}^j$ before applying the activation function at the j-th layer, BN computes

$$\boldsymbol{\mu}^j = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \boldsymbol{u}_i^j, \tag{2.52}$$

$$(\boldsymbol{\sigma}^j)^{\circ 2} = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} (\boldsymbol{u}_i^j - \boldsymbol{\mu}^j)^{\circ 2}, \tag{2.53}$$

$$\hat{\boldsymbol{u}}_i^j = \boldsymbol{\gamma}^j \frac{\boldsymbol{u}_i^j - \boldsymbol{\mu}^j}{\sqrt{(\boldsymbol{\sigma}^j)^{\circ 2} + \varepsilon}} + \boldsymbol{\beta}^j, \tag{2.54}$$

where $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ control the distribution of the middle output at the j-th layer.

BN is frequently used in image recognition networks, but its theoretical effectiveness has not been clarified. There are several effects: smoothing the loss function [318], randomness for finite samples [231, 183], and stabilizing the residual connection in the early stages of learning by making it dominant [73].

2.6.6 Layer Normalization

BN has a problem of instability when the batch size is small. To address this problem, layer normalization (LN) [13] is a method of normalization in the unit direction:

$$\boldsymbol{\mu}_i = \frac{1}{d_{h_j}}\sum_{j=1}^{d_{h_j}} \boldsymbol{u}_i^j, \tag{2.55}$$

$$(\boldsymbol{\sigma}_i)^{\circ 2} = \frac{1}{d_{h_j}} \sum_{j=1}^{d_{h_j}} (\boldsymbol{u}^j - \boldsymbol{\mu}^j)^{\circ 2}, \tag{2.56}$$

$$\hat{\boldsymbol{u}}_i^j = \boldsymbol{\gamma}^j \frac{\boldsymbol{u}_i^j - \boldsymbol{\mu}_i}{\sqrt{(\boldsymbol{\sigma}_i)^{\circ 2} + \varepsilon}} + \boldsymbol{\beta}^j. \tag{2.57}$$

Layer normalization is often used in language models such as Transformer [365].

2.7 Backpropagation

Although the parameter optimization method was outlined in Section 2.4, the respective parameter gradients for the losses were essential. The backpropagation method is an efficient way to obtain the gradients of each parameter. The method is based on the chain rule of derivatives, and its ease of use and usefulness triggered the second AI boom in the 1980s [311].

In this section, the previous notation is slightly modified for the sake of explanation. The composition of linear transformations and activation operations at each layer is broken down into two operations and described as

$$\boldsymbol{u}^l = W^l \boldsymbol{z}^{l-1} + \boldsymbol{b}^l, \tag{2.58}$$

$$\boldsymbol{z}^l = f^l(\boldsymbol{u}^l), \tag{2.59}$$

where $\boldsymbol{u}^l, \boldsymbol{z}^l \in \mathbb{R}^{d_{h_l}}$ are hidden units before and after, respectively, operating the activation function at the l-th layer.

The loss gradient for the parameter W_{ij}^l is

$$\frac{\partial L}{\partial W_{ij}^l} = \frac{\partial L}{\partial u_i^l} \frac{\partial u_i^l}{\partial W_{ij}^l} = \delta_i^l z_j^{l-1}, \tag{2.60}$$

$$\frac{\partial L}{\partial b_i^l} = \frac{\partial L}{\partial u_i^l} \frac{\partial u_i^l}{\partial b_i^l} = \delta_i^l, \tag{2.61}$$

$$\delta_i^l := \frac{\partial L}{\partial u_i^l}. \tag{2.62}$$

The vector and matrix form of this equation is

$$\frac{\partial L}{\partial W^l} = \boldsymbol{\delta}^l \left(\boldsymbol{z}^{l-1}\right)^T, \tag{2.63}$$

$$\frac{\partial L}{\partial \boldsymbol{b}^l} = \boldsymbol{\delta}^l. \tag{2.64}$$

This means that the gradient of each parameter can be calculated from the gradient vectors $\{\boldsymbol{\delta}^l\}$. The gradient vectors can be computed for n_l-th layer neural network by

$$\boldsymbol{\delta}^l = \mathrm{diag}\{(f^l)'(\boldsymbol{u}^l)\}(W^{l+1})^T \boldsymbol{\delta}^{l+1}, \ for\ l = 1, \cdots, n_l, \tag{2.65}$$

$$\boldsymbol{\delta}^{n_l+1} = \frac{\partial L}{\partial \hat{\boldsymbol{y}}}. \tag{2.66}$$

This recurrence formula of the gradients $\{\boldsymbol{\delta}^l\}$ yields each parameter gradient by back-propagating $\boldsymbol{\delta}$ from the output layer to the input layer.

Since back-propagation of gradients is essential in NN, a differentiable mechanism is preferred. Many mechanisms have been proposed, such as the Sinkhorn algorithm [68, 319] used in optimal transport (e.g., Wasserstein GAN [12]) and differentiable data augmentation [167, 423], that allow end2end learning using only differentiable arithmetic mechanisms.

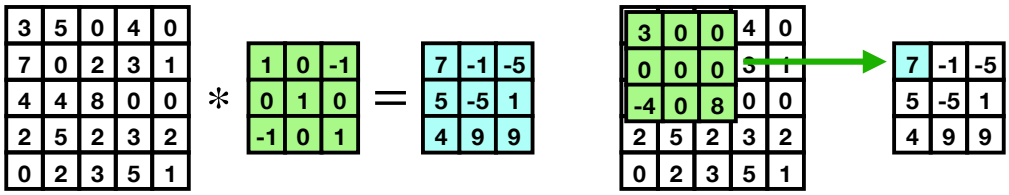


Figure 2.10: Convolutional operation

2.7.1 Gradient Vanishing Problem

The gradient backpropagation calculation is linear from the output layer to the input layer, so the gradient diverges when the gradient in each layer is large and vanishes when the gradient is small. This problem is called the gradient vanishing problem and is the main reason why deep neural networks with 20 or more layers were considered difficult. Currently, activation functions such as ReLU and ELU, batch normalization, weight initialization, and residual connections are used to alleviate this problem, making it possible to train deep neural networks such as ResNet-152 [127], which has 152 layers.

2.7.2 Automatic Differentiation

The backpropagation method is a type of automatic differentiation. Automatic differentiation is a method that automatically generates a procedure for computing the gradient of a directed acyclic graph from a procedure for computing an operation $\hat{\boldsymbol{y}} = h_{\boldsymbol{\theta}}(\boldsymbol{x})$. In packages such as PyTorch [283], and TensorFlow [1], which handle NN, the computational graph is constructed behind the scenes using automatic differentiation to compute forward propagation and gradient backpropagation.

2.8 Convolutional Neural Network

Humans have a high ability to recognize patterns from visual information. Electrical signals of visual information received by the retina are input to the primary visual system via the lateral pallidum in the thalamus while maintaining the flat structure of the image. This is where the first step of pattern recognition takes place.

Two types of cells respond to specific patterns: simple and complex. Simple cells have a narrow receptive field and fire only when a pattern is present in a specific area, while complex cells have a wide receptive field and fire when a pattern is present anywhere in the image plane. Complex cells comprise a bundle of simple cells with local receptive fields. Convolutional neural networks (CNNs) were born from this hierarchical hypothesis.

CNN is a widely used architecture in the field of image recognition and is the spark for the current third AI boom. In 2012, AlexNet [187], which uses CNNs, showed overwhelmingly superior recognition performance to conventional machine learning methods in the ILSVRC image recognition competition [313], and CNNs and deep learning have suddenly become widely studied.

2.8.1 Convolutional Layer

CNNs can be applied to a wide range of data, including time series, images, and point clouds, but for simplicity, we will focus on 2D CNNs for image data. Image data is a third-order tensor $X \in \mathbb{R}^{H \times W \times C}$ with size $H \times W$ and C channels. For example, an image using common RGB colors has three channels: R, G, and B.

The convolutional layer transforms the input tensor into an output tensor using local linear operations and activation functions. For the input tensor $Z^{l-1} \in \mathbb{R}^{H_{l-1} \times W_{l-1} \times C_{l-1}}$ of the l-layer, the output tensor of

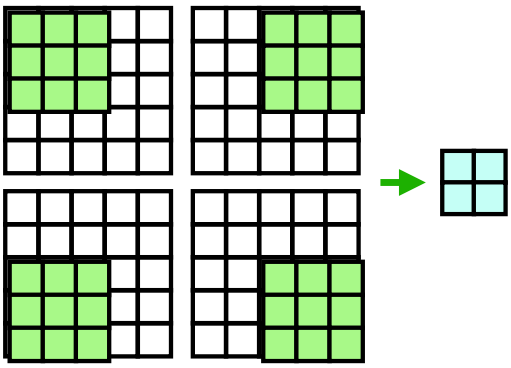


Figure 2.11: Convolution with stride $S = 2$

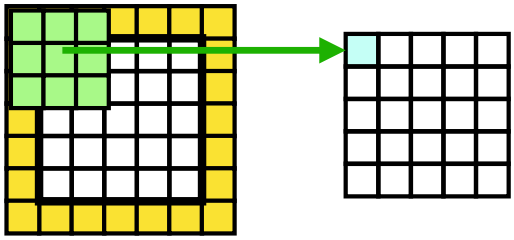


Figure 2.12: Convolution with padding $P = 1$

the layer with kernel size $K_l \in \mathbb{N}$ is

$$U^l_{ijk} = \sum_{c=0}^{C_{l-1}-1} \sum_{p,q=0}^{K_l-1} Z^{l-1}_{i+p,j+q,c} W^{lpqc}_k + B^l_{ijk}, \tag{2.67}$$

$$Z^l = f^l(U), \tag{2.68}$$

where $W^l \in \mathbb{R}^{K \times K \times C_{l-1} \times C_l}$ is the l-th weight tensor (slightly abuse of notation, weight and width are both denoted by W), $B^l \in \mathbb{R}^{H_l \times W_l \times C_l}$ is the l-th bias tensor, $f^l : \mathbb{R} \to \mathbb{R}$ is the l-th activation function. The output height H_l and output width W_l is determined by

$$H_l = H_{l-1} - K_l + 1, \; W_l = W_{l-1} - K_l + 1. \tag{2.69}$$

Figure 2.10 shows the convolution operation when the input/output channels are 1, the input height and width are 3, and the kernel size is 3. Local information is extracted at each coordinate of the output image by shifting a filter with a receptive field of only the kernel size. Adding convolution layers combines the information from each local receptive field to capture complex patterns, such as complex cells.

2.8.2 Stride

In normal filtering, the filter is moved one pixel at a time, but stride is used to capture image features more roughly. Stride $S_l \in \mathbb{N}$ means that the filter is moved by skipping an extra pixel by $S_l - 1$ (Figure 2.11), and the output tensor is

$$U^l_{ijk} = \sum_{c=0}^{C_{l-1}-1} \sum_{p,q=0}^{K_l-1} Z^{l-1}_{S_l i+p,S_l j+q,c} W^{lpqc}_k + B^l_{ijk}, \tag{2.70}$$

where the output image size is

$$H_l = \left\lfloor \frac{H_{l-1} - K_l}{S_l} \right\rfloor + 1, \; W_l = \left\lfloor \frac{W_{l-1} - K_l}{S_l} \right\rfloor + 1. \tag{2.71}$$

2.8.3 Padding

Normal convolution always results in a smaller image size, which can be a problem in image processing techniques. Padding addresses this problem by expanding the image before convolution (Figure 2.12). The image size after convolution of padding $P_l \in \mathbb{N}$ is

$$H_l = \left\lfloor \frac{H_{l-1} - K_l + 2P_l}{S_l} \right\rfloor + 1, \; W_l = \left\lfloor \frac{W_{l-1} - K_l + 2P_l}{S_l} \right\rfloor + 1. \tag{2.72}$$

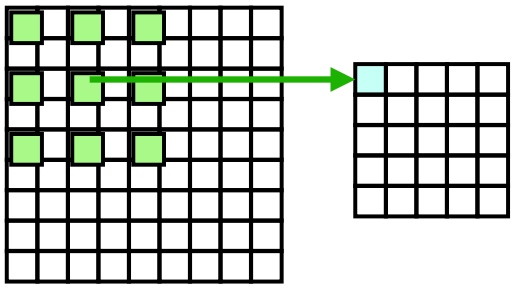


Figure 2.13: Convolution with dilation $D = 2$

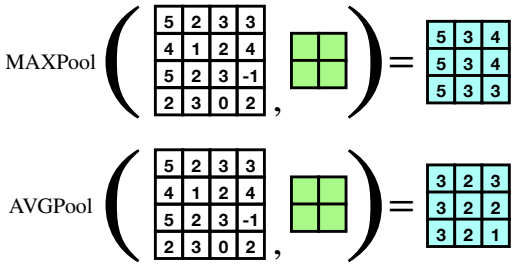


Figure 2.14: Pooling layer

There are several types of padding: zero-padding, which fills the surrounding area with zeros; replica-padding, which extrapolates the pixel value of the most surrounding area; reflect-padding, which wraps the pixel value at the most surrounding area; and circular-padding, which considers the pixel value to appear periodically.

2.8.4 Dilation

When the image size is large, it is necessary to increase the kernel size or pile up multiple convolution layers to expand the receptive field. The former makes it difficult to capture fine information, while the latter increases the learning difficulty. Dilated convolution captures fine structures while expanding the receptive field by "dilating" the filter (Figure 2.13). The output tensor of dilated convolution with dilation size D_l is

$$U^l_{ijk} = \sum_{c=0}^{C_{l-1}-1} \sum_{p,q=0}^{K_l-1} Z^{l-1}_{S_l i+D_l p, S_l j+D_l q, c} W^{lpqc}_k + B^l_{ijk}, \tag{2.73}$$

The output image size of the dilated convolution is

$$H_l = \left\lfloor \frac{H_{l-1} - D_l(K_l-1) - 1 + 2P_l}{S_l} \right\rfloor + 1,\ W_l = \left\lfloor \frac{W_{l-1} - D_l(K_l-1) - 1 + 2P_l}{S_l} \right\rfloor + 1. \tag{2.74}$$

2.8.5 Pooling Layer

The pooling layer is a layer that makes the local patterns captured by the convolution layer locally position-invariant. Since the pooling layer plays the role of a complex-type cell that bundles local patterns, it is essentially parameter-free. Commonly used pooling methods are max pooling and average pooling, but there are also extensions of these methods, such as L^P pooling and stochastic pooling, which samples representative values with relative probability when the input is positive (Figure 2.14). Max pooling, average pooling, and L^P pooling with kernel size K_l, stride S_l and dilation D_l are computed by

$$U^l_{ijk} = \max_{0 \le p,q \le K_l-1} Z^{l-1}_{S_l i+D_l p, S_l j+D_l q, k}, \tag{2.75}$$

$$U^l_{ijk} = \frac{1}{(K_l)^2} \sum_{p,q=0}^{K_l-1} Z^{l-1}_{S_l i+D_l p, S_l j+D_l q, k}, \tag{2.76}$$

$$U^l_{ijk} = \left(\frac{1}{(K_l)^2} \sum_{p,q=0}^{K_l-1} (Z^{l-1}_{S_l i+D_l p, S_l j+D_l q, k})^P \right)^{1/P}, \tag{2.77}$$

respectively.

Average pooling that matches the input image size with the kernel size is called global average pooling (GAP). The pooling is frequently used in the final output layer of a CNN to ensure that the features are location-invariant.

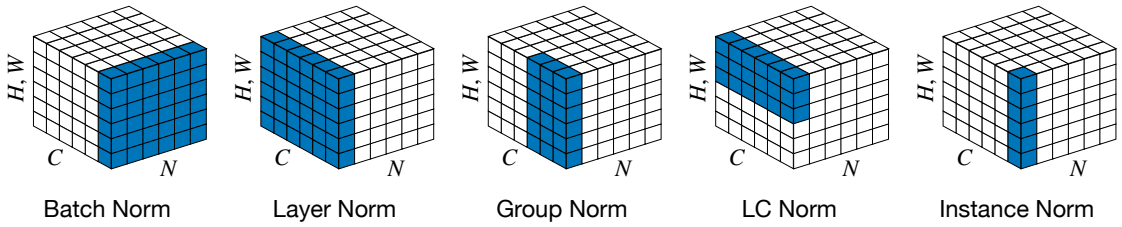


Figure 2.15: Comparison of normalization methods

2.8.6 Pointwise / Channel-wise Convolution

Convolution with a kernel size of 1×1 is called pointwise convolution, which aggregates information in the channel direction. The convolution is used to convert the number of channels or to obtain pixel-by-pixel features and is utilized in well-known CNN architectures such as ResNet [127].

Group-wise convolution divides the channels $\mathbb{N}_{C_l}$ into groups $\{C_l^g\}_g$ and performs the usual convolution operation for each group. The convolution is formulated by

$$U_{ijk}^l = \sum_{c \in C_{l-1}^g} \sum_{p,q=0}^{K_l-1} Z_{S_l i + D_l p, S_l j + D_l q, c}^{l-1} W_k^{lpqc} + B_{ijk}^l, \ k \in C_l^g. \tag{2.78}$$

The convolution is used when different groups are expected to acquire different features or when architectures separate in the middle of the process.

In group-wise convolution, when each group has only one element, it is called channel-wise convolution. The convolution is useful for acquiring spatially oriented features without interference from other channels since the convolution operation is performed for each channel. Instead of the usual convolution, channel-wise convolution and pointwise convolution can be applied in pairs to reduce the computation order from $O(K^2 C_l C_{l-1})$ to $O(K^2 C_{l-1} + C_l C_{l-1})$.

2.8.7 Upsampling / Transposed Convolution

In a normal CNN, the number of pixels monotonically decreases as the image input to the CNN propagates inside the CNN, but for some tasks, it may be desirable to increase the number of pixels. Super-resolution, which transforms a low-resolution image into a high-resolution image, is a typical example [197, 206, 415, 164, 56, 198]. In semantic segmentation, which classifies the class to which each pixel belongs, the structure of reducing the number of pixels once and then enlarging it is often used [308, 107, 374, 375, 58, 18]. The generative model described in Chapter 3 uses an architecture that generates a larger real image from a smaller image.

The simplest way to increase image size is to upsampling and interpolating in pairs. Pixel values are placed at r pixel intervals, and the pixels in between are interpolated.

The currently popular method without interpolation is transposed convolution [332]. Ignoring the bias term, the output vector $\boldsymbol{u} \in \mathbb{R}^{C_l W_l H_l}$ is always represented by

$$\boldsymbol{u} = W \boldsymbol{z}, \tag{2.79}$$

where $\boldsymbol{z} \in \mathbb{R}^{C_{l-1} W_{l-1} H_{l-1}}$ is the input vector and $W \in \mathbb{R}^{C_l W_l H_l \times C_{l-1} W_{l-1} H_{l-1}}$ is the sparse weight matrix. Using the transposed matrix W^T, we can consider a form in which the input and output are swapped, corresponding to the usual formula for the back-propagation of convolution layers. The transposed convolution is equivalent to the pair of upsampling and normal convolution.

2.8.8 Normalization of Convolutional Layer

The batch normalization introduced in subsection 2.6.5 is often used in convolutional layers. For a mini-batch size N, number of channels C, and number of pixels H, W, the normalization methods normalize within the

blue region shown in Figure 2.15. The mathematical expressions are given by

- Batch normalization [152]:

$$\mu_c = \frac{1}{NHW} \sum_{n,i,j} u^n_{ijc}, \tag{2.80}$$

$$\sigma^2_c = \frac{1}{NHW} \sum_{n,i,j} (u^n_{ijc} - \mu_c)^2, \tag{2.81}$$

$$\hat{u}^n_{ijc} = \gamma_c \frac{u^n_{ijc} - \mu_c}{\sqrt{\sigma^2_c + \varepsilon}} + \beta_c, \tag{2.82}$$

  where $\beta_c, \gamma_c \in \mathbb{R}$ are affine parameters at the c-th channel.

- Layer normalization [13]:

$$\mu^n = \frac{1}{CHW} \sum_{i,j,c} u^n_{ijc}, \tag{2.83}$$

$$(\sigma^n)^2 = \frac{1}{CHW} \sum_{i,j,c} (u^n_{ijc} - \mu^n)^2, \tag{2.84}$$

$$\hat{u}^n_{ijc} = \gamma \frac{u^n_{ijc} - \mu^n}{\sqrt{(\sigma^n)^2 + \varepsilon}} + \beta. \tag{2.85}$$

- Group normalization [391]:

$$\mu^n_g = \frac{1}{|C_g|HW} \sum_{i,j,c \in C_g} u^n_{ijc}, \tag{2.86}$$

$$(\sigma^n_g)^2 = \frac{1}{|C_g|HW} \sum_{ijc} (u^n_{i,j,c \in C_g} - \mu^n_g)^2, \tag{2.87}$$

$$\hat{u}^n_{ijc} = \gamma \frac{u^n_{ijc} - \mu^n_{G(c)}}{\sqrt{(\sigma^n_{G(c)})^2 + \varepsilon}} + \beta, \tag{2.88}$$

  where $\{C_g\}_g$ are partition of C channels i.e. $\mathbb{N}_C = \bigcup_g C_g$ and $C_g \cap C'_g = \emptyset$ $(g \neq g')$, $G : c \mapsto g$ is the assigning map.

- Local contrast normalization (divisive normalization) [233]:

$$\mu^n_{ij} = \frac{1}{C|\mathcal{N}(i,j)|} \sum_{c,(i',j') \in \mathcal{N}(i,j)} w_{i'j'c} u^n_{i'j'c}, \tag{2.89}$$

$$(\sigma^n_{ij})^2 = \frac{1}{C|\mathcal{N}(i,j)|} \sum_{c,(i',j') \in \mathcal{N}(i,j)} w_{i'j'c} (u^n_{i'j'c} - \mu^n_{ij})^2, \tag{2.90}$$

$$\hat{u}^n_{ijc} = \frac{u^n_{ijc} - \mu^n_{ij}}{\sqrt{(\sigma^n_{ij})^2 + \varepsilon}}, \tag{2.91}$$

  where $\mathcal{N}(i,j)$ are neighbors of the (i,j)-th pixel and $\{w_{i'j'c}\}_{i'j'c}$ are weights which have a maximum value at the center and decreases to the periphery.

- Instance normalization [360]:

$$\mu^n_c = \frac{1}{HW} \sum_{i,j} u^n_{ijc}, \tag{2.92}$$

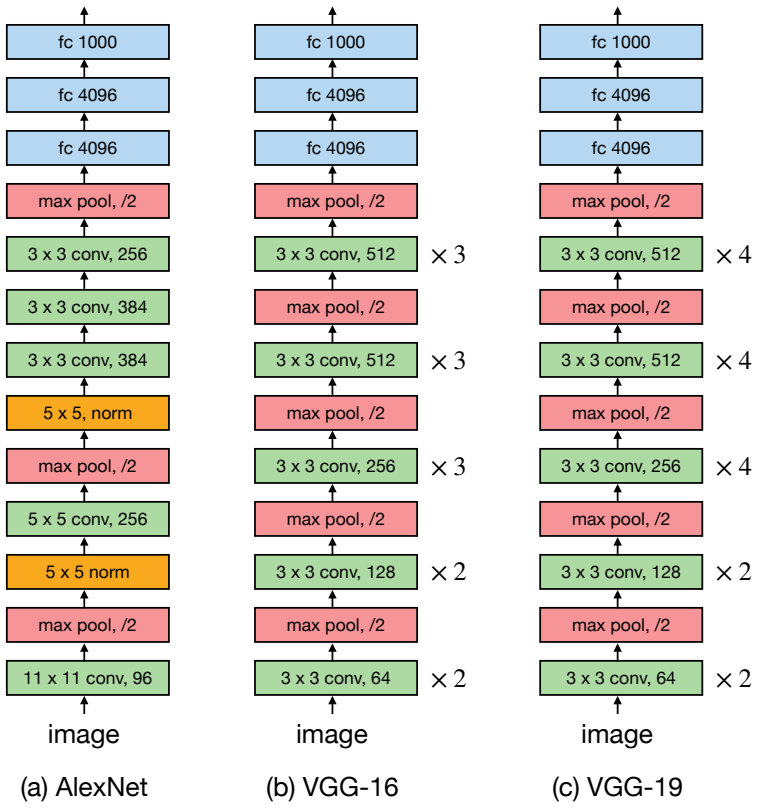


Figure 2.16: Representative CNN architectures. "$k \times k$ conv, c" means convolution layer with kernel size k and c channels, "/2" means pooling with stride 2 and kernel size 2. "5 x 5, norm" means contrast normalization layer with neighbor size 5, "fc, c" means fully-connected layer with c hidden units.

$$(\sigma_c^n)^2 = \frac{1}{HW}\sum_{i,j}(u_{ijc}^n - \mu_c^n)^2, \tag{2.93}$$

$$\hat{u}_{ijc}^n = \gamma_c \frac{u_{ijc}^n - \mu_c^n}{\sqrt{(\sigma_c^n)^2 + \varepsilon}} + \beta_c. \tag{2.94}$$

2.8.9 Representative Architecture

The structural design of CNNs has evolved from AlexNet [187], which sparked the third AI boom, to VGGNet [218], an improved version, to ResNet [127], one of the modern basic structures. AlexNet (Figure 2.16(a)) is the first CNN to successfully perform large-scale object recognition in a system comparable to humans. VGGNet (Figre 2.16(b), (c)) has evolved from AlexNet to a deeper, 16∼19-layer system.

ResNet (Figure 2.17) employs residual connections and batch normalization to significantly increase the number of layers. A residual connection is an operation that adds the current feature to the output of several layers later and is used within the res-block. Global average pooling is employed before the fully connected layer to convert to position-invariant features. ResNet is called ResNet-34 when the basic structure is used as a res-block and ResNet-50 when the bottleneck structure is used, with $(N_1, N_2, N_3, N_4) = (3, 3, 5, 2)$. Using the basic structure with $(N_1, N_2, N_3, N_4) = (2, 1, 1, 1)$, ResNet-18; using the bottleneck structure with $(N_1, N_2, N_3, N_4) = (3, 3, 22, 2)$, ResNet-101; using the bottleneck structure with $(N_1, N_2, N_3, N_4) = (3, 7, 35, 2)$, ResNet-152. The bottleneck structure can capture a wider variety of features by increasing the number of channels in a block without increasing the computational complexity through 1×1 convolution.

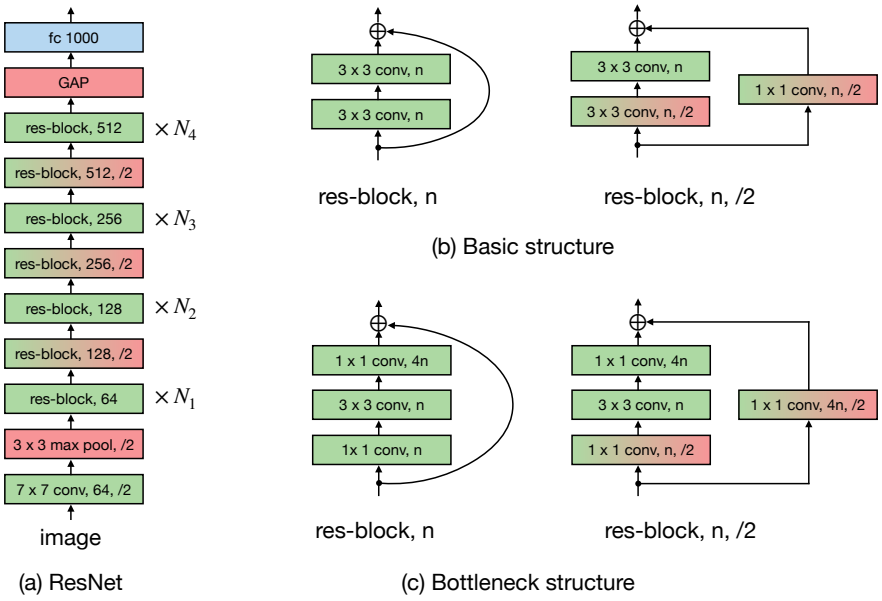


Figure 2.17: ResNet architecture. (a) Overall architecture. "/2" means convolution with 2:1 downsampling. (b) The basic architecture of res-block. (c) The bottleneck architecture of res-block

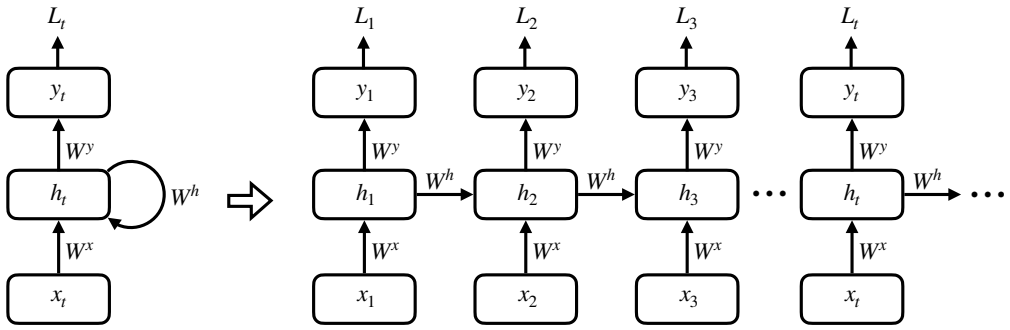


Figure 2.18: Basic architecture of recurrent neural network

2.9 Networks for Sequential Data

In this section, we deal with sequential data represented as $X = \{\boldsymbol{x}_t\}_{t=1}^T$. Sequential data includes time-series data such as hourly temperature and precipitation, string data corresponding to input text such as ChatGPT [424], and voice data for smart speakers. Common basic structures such as recurrent neural networks (RNNs) [151, 310, 61, 135], temporal convolutional networks (TCNs), and transformers [365] are used in all cases.

2.9.1 Recurrent Neural Network

Ignoring time transitions, recurrent neural networks (RNNs) are a generic term for NNs with an internally directed closed path, which, when expanded, can be regarded as NNs that share parameters at each time point (Figure 2.18). RNNs include time delay NNs (TDNN) [368], echo state networks (ESN) [158], and long short-term memory (LSTM) [135].

An RNN is formulated as

$$\boldsymbol{h}_t = f^h(W^x \boldsymbol{x}_t + W^h \boldsymbol{h}_{t-1} + \boldsymbol{b}^h), \tag{2.95}$$

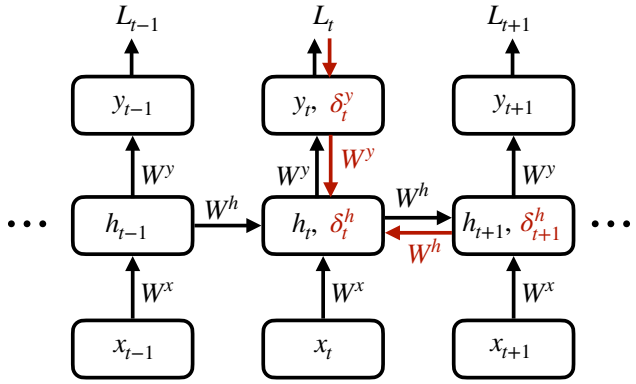


Figure 2.19: Backpropagation through time

$$\boldsymbol{y}_t = f^y(W^y \boldsymbol{h}_t + \boldsymbol{b}^y), \tag{2.96}$$

where f^h and f^y are activation functions, $\boldsymbol{h}_t \in \mathbb{R}^{d_h}$ is hidden vector at time t, $\boldsymbol{y}_t \in \mathbb{R}^{d_y}$ is output vector at time t.

The final output of RNN varies depending on the task. In the case of a time series forecast, such as a temperature time series, the goal is to output the forecast value $\hat{\boldsymbol{x}}_{t+1}$ at the next time point as $\boldsymbol{y}_t$, and for practical use, the forecast value can be used as the input for the next time point to allow forecasting to any future time point. In the case of predicting a class from T inputs, such as document classification, the loss is calculated based on the output $\boldsymbol{y}_T$ at the last time point. In tasks such as document classification, where inference is performed after a batch of sequential data is given, bi-directional RNNs (bi-RNNs) [325] are often used to integrate sequential data with the output of RNN input in the reverse direction.

2.9.2 Backpropagation Through Time

RNN training is computed by the error back-propagation method as with other networks. The backpropagation through time (BPTT) method [385, 383] is an error backpropagation method for RNNs.

For simplicity of notation, let

$$\boldsymbol{h}_t = f^h(\boldsymbol{u}_t), \qquad \boldsymbol{u}_t = W^x \boldsymbol{x}_t + W^h \boldsymbol{h}_{t-1} + \boldsymbol{b}^h, \tag{2.97}$$

$$\boldsymbol{y}_t = f^h(\boldsymbol{v}_t), \qquad \boldsymbol{v}_t = W^y \boldsymbol{h}_t + \boldsymbol{b}^y, \tag{2.98}$$

be used. The BPTT method calculates the gradient vector of the backpropagation from the chain rule of differentiation as

$$\boldsymbol{\delta}_t^v = \frac{\partial L}{\partial \boldsymbol{v}_t} = \boldsymbol{\delta}_t^y (f^y)'(\boldsymbol{v}_t), \tag{2.99}$$

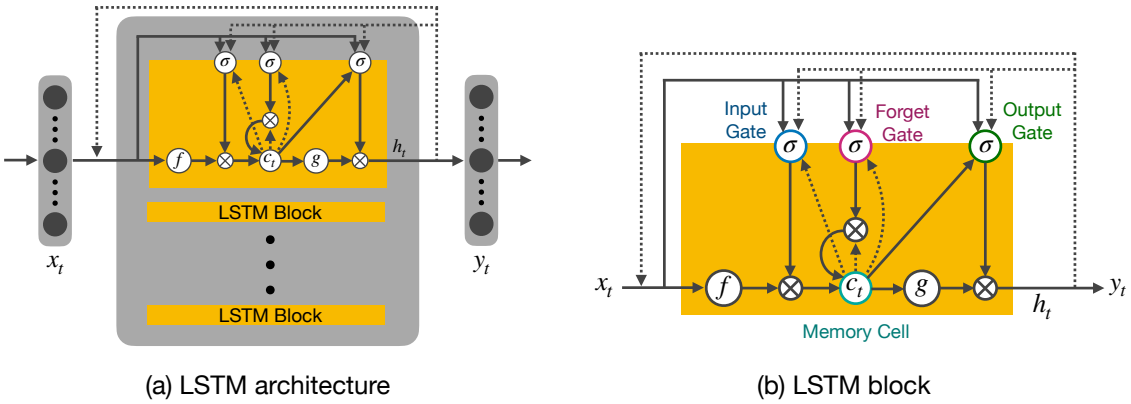
$$\boldsymbol{\delta}_t^u = \frac{\partial L}{\partial \boldsymbol{u}_t} = \boldsymbol{\delta}_t^h (f^h)'(\boldsymbol{u}_t), \tag{2.100}$$

$$\boldsymbol{\delta}_t^h = \frac{\partial L}{\partial \boldsymbol{h}_t} = (W^y)^T \boldsymbol{\delta}_t^v + (W^h)^T \boldsymbol{\delta}_{t+1}^u. \tag{2.101}$$

Figure 2.19 shows the schematic flow of the gradients in the method.

2.9.3 Long Short-Term Memory

RNN reflects the output gradient using the BPTT method, but due to the gradient vanishing problem, only about ten-time points are transmitted. Long short-term memory (LSTM) [135] is designed to retain memory over a long period of time.



(a) LSTM architecture (b) LSTM block

Figure 2.20: Architecture of long short-term memory

Figure 2.20 shows the architecture of LSTM. The network consists of the memory cell and the three gates. The cell $\boldsymbol{c}_t \in \mathbb{R}^{d_h}$ is responsible for storing the state and solving the gradient vanishing problem. The forget gate $\boldsymbol{g}_t^f \in \mathbb{R}^{d_h}$ forgets memories that are no longer needed according to the current input or past states. The input gate $\boldsymbol{g}_t^i \in \mathbb{R}^{d_h}$ adjusts the amount of memory in the cell according to the current input and the past state. The output gate $\boldsymbol{g}_t^o \in \mathbb{R}^{d_h}$ adjusts the transmission to the outside according to the current input and the past state. These are formulated by

$$\boldsymbol{g}_t^i = \sigma(W^{i,x}\boldsymbol{x}_t + W^{i,h}\boldsymbol{h}_{t-1} + W^{i,c}\boldsymbol{c}_{t-1}), \tag{2.102}$$

$$\boldsymbol{u}_t = f(W^{x}\boldsymbol{x}_t + W^{h}\boldsymbol{h}_{t-1}), \tag{2.103}$$

$$\boldsymbol{g}_t^f = \sigma(W^{f,x}\boldsymbol{x}_t + W^{f,h}\boldsymbol{h}_{t-1} + W^{f,c}\boldsymbol{c}_{t-1}), \tag{2.104}$$

$$\boldsymbol{c}_t = \boldsymbol{g}_t^i \odot \boldsymbol{u}_t + \boldsymbol{g}_t^f \odot \boldsymbol{c}_{t-1}, \tag{2.105}$$

$$\boldsymbol{g}_t^o = \sigma(W^{o,x}\boldsymbol{x}_t + W^{o,h}\boldsymbol{h}_{t-1} + W^{o,c}\boldsymbol{c}_{t}), \tag{2.106}$$

$$\boldsymbol{h}_t = \boldsymbol{g}_t^o \odot g(\boldsymbol{c}_t), \tag{2.107}$$

where $\sigma : \mathbb{R} \to [0, 1]$ is the sigmoid function, f and g are activation functions which are tanh function mainly used. Propagation from the memory cell to the gate, called peephole coupling, controls the gate by looking into its internal state and may be omitted. Each gate takes a value in [0,1] to control the amount of forgetting or input/output.

LSTM is a complex calculation and, therefore, computationally expensive. The update gate RNN (UGRNN), gated recurrent unit (GRU) [61], and intersection RNN (+RNN) [66] focus on the forget gate, which is said to be the most important part of LSTM [113, 104], to reduce computational complexity. Recurrent Kalman networks (RKNs) [23] and particle filter RNNs (PFRNNs) [236], which utilize sequential data assimilation methods such as Kalman filters [166] and particle filters [176, 111], have also been proposed.

2.10 Attention Mechanism

Attention mechanisms are a core architectural element of NN and are the foundation of large-scale language models such as GPT [277] and BERT [75]. Figure 2.21 shows the basic structure of the attention mechanism. The architecture returns an output from query, key, and value variables. When a new query comes in, it searches for keys in the dictionary and returns the value corresponding to the key with the highest match to the query. In mathematical terms, this corresponds to returning a weighted sum of value vectors $\{\boldsymbol{v}_i\}$ based

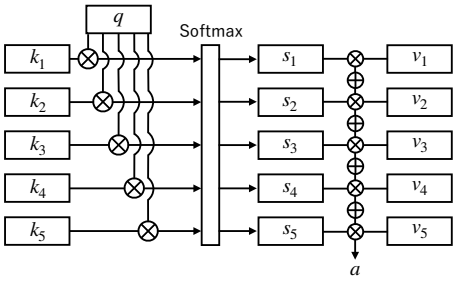


Figure 2.21: Attention mechanism

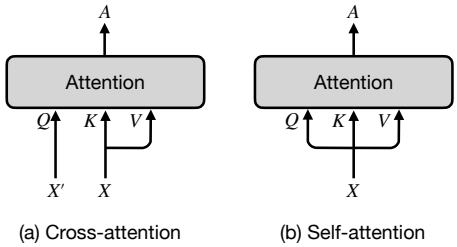


Figure 2.22: Cross-attention and self-attention

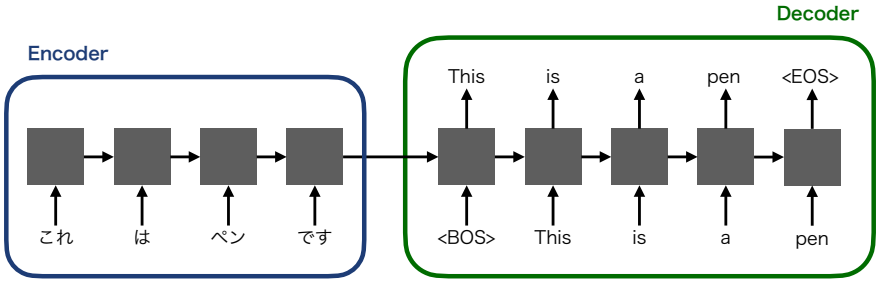


Figure 2.23: Traditional Seq2Seq architecture

on the relative similarity of the query vector $\boldsymbol{q}$ to the key vectors $\{\boldsymbol{k}_i\}$. This is formulated by

$$K = \begin{pmatrix} \boldsymbol{k}_1^T \\ \vdots \\ \boldsymbol{k}_N^T \end{pmatrix}, \; V = \begin{pmatrix} \boldsymbol{v}_1^T \\ \vdots \\ \boldsymbol{v}_N^T \end{pmatrix}, \tag{2.108}$$

$$\boldsymbol{a} = \mathrm{softmax}(\boldsymbol{q}^T K^T) V, \tag{2.109}$$

where 'softmax' is the softmax function which restricts the column sum to one.

Attention mechanisms used in NN can be broadly classified into cross-attention (source-to-target attention) and self-attention. Figure 2.22 shows the difference between the two types of attention. Cross-attention generates a key matrix K and a value matrix V from the same internal input X, and a query matrix Q from the external input X'. This attention is intuitive, given the computer-mimicking mechanism denoted above. Self-attention generates three matrices from the same internal input X. This attention implies that the input matrices are updated according to the relationships within the same input and are used in many modern architectures, including Transformer [365].

2.10.1 Seq2Seq

The Attention mechanism was originally proposed as an extension of the sequence transformation model Seq2Seq [345, 14]. The original Seq2Seq is a model that transforms an input series into an output series after it has been reduced to a single vector. Figure 2.23 shows an overview of the model. Here, we consider a Japanese-to-English machine translation task. The Encoder converts the Japanese input "これはペンです" into an embedded vector corresponding to each word, which RNN processes to obtain a hidden vector. From that vector and the first input <BOS> on the decoder side (which means the beginning of a sentence), the first output "This" is obtained. The output is treated as the next input, and the next output "is" is obtained. This is repeated until <EOS>, meaning the end of the sentence, is produced; the machine translation is complete at this point.

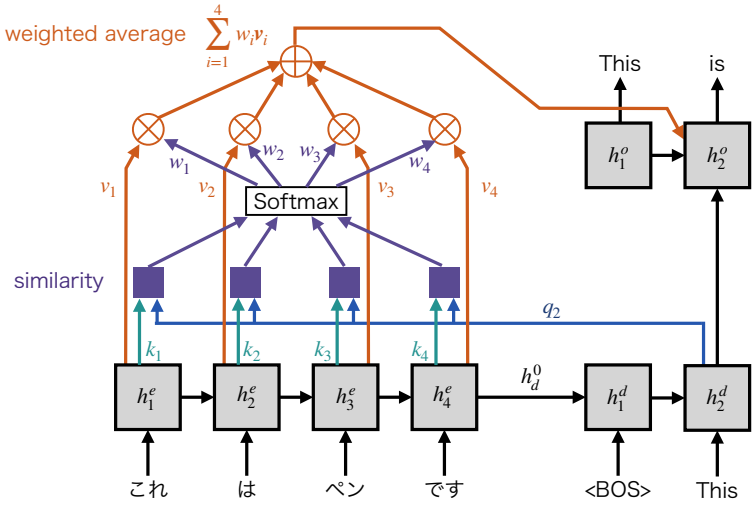


Figure 2.24: Seq2Seq with attention mechanism

Seq2Seq cannot store long-term information as the series length increases, resulting in decreased accuracy. Seq2Seq with attention generates a query vector from the hidden vectors at each time point in the decoder and crosses it with the key-value vector in the encoder (Figure 2.24). This attention allows each input of the decoder to correspond to the required input of the encoder, which improves accuracy for long-term series.

2.10.2 Transformer

Transformer [365] was originally developed for machine translation tasks in the vein of Seq2Seq, but it is now effective for almost all tasks involving language, and its application to non-language data such as images and graphs is also popular. Because of its order equivalence, its architecture can be easily applied to aggregate data other than sequential data. This means that the order of the outputs only changes when the order of the inputs is switched.

The architecture is shown in Figure 2.25. The vector corresponding to each point in the input matrix X to each attention in the architecture is called a token. The Transformer learns the relationships among the tokens by an attention mechanism and propagates them to the next layer. Assuming a machine translation task, each structure in the architecture is described step by step.

Embedding The embedding layer transforms each word into a vector of fixed dimensions, such as 784 dimensions. After one-hot encoding, a vector of the desired dimension is obtained by converting to the linear layer. This transformation of words into vectors is called distributed representation and has been studied in the fields of cognitive psychology and neuroscience. This is because the human brain can remember new events and concepts by associating them with known events and concepts based on their similarities. On the other hand, in the field of natural language processing, the distribution hypothesis [90, 122], which states that the meaning of a word depends on the surrounding words, was proposed, and these fields were combined to propose an initially distributed representation based on principal component analysis and latent Dirichlet allocation [34].

With the spread of neural networks, the acquisition of distributed representations by skip-gram and CBOW became mainstream [249, 256]. CBOW acquires the distributed representation of the features in front of the output layer by solving the task of finding the current word from surrounding words masking the current word, while skip-gram acquires the distributed representation of the features in front of the output layer by solving the task of finding the current word from words in front of the output layer. The Transformer obtains the initial embedding representation by back-propagating errors to the embedding layer.

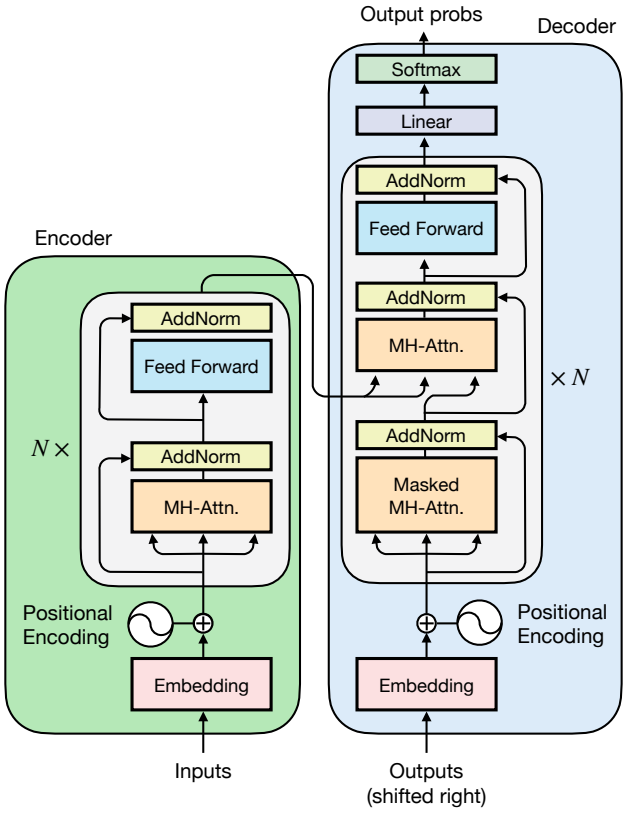


Figure 2.25: Transformer architecture. MH-Attn means multi-head attention, and AddNorm means residual connection and layer normalization. This figure is based on [365]

Positional Encoding Although the Transformer can receive set data from ordinal equivalence, it was originally mainly applied to series data, so positional encoding (PE) was used to add time information. The encoding uses a vector $\boldsymbol{p}_t \in \mathbb{R}^{d_h}$ corresponding to the time information, which is added to the input $\boldsymbol{x}_t + \boldsymbol{p}_t$ or concatenated to make $[\boldsymbol{x}_t, \boldsymbol{p}_t]^T$.

The PE vector can be obtained by fixing the operations in advance or by automatically determining them in training [103, 75, 294, 295, 38, 277]. In the former case, sinusoidal waves [365] are often used to set

$$(\boldsymbol{p}_t)_j = \begin{cases} \sin\left(\frac{t}{10000^{2j/d_h}}\right), & j \in 2\mathbb{N}, \\ \cos\left(\frac{t}{10000^{2j/d_h}}\right), & j \in 2\mathbb{N}+1. \end{cases} \tag{2.110}$$

This vector stores a sine wave with a different frequency for each element.

Multi-head Attention The attention mechanism can be expressed as

$$\mathcal{A}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_h}}\right)V, \tag{2.111}$$

for a query matrix $Q \in \mathbb{R}^{T\times d_h}$, a key matrix $K \in \mathbb{R}^{T\times d_h}$, and a value matrix $V \in \mathbb{R}^{T\times d_h}$. The weight matrices W^Q,W^K,W^V to transform from the input matrices X,X' to be multiplied later can be expressed as $Q = X'$ and $K = V = X$. Since we decided to apply the matrices later, the output of the attention mechanism can be expressed as

$$A = \mathcal{A}(QW^Q, KW^K, VW^V). \tag{2.112}$$

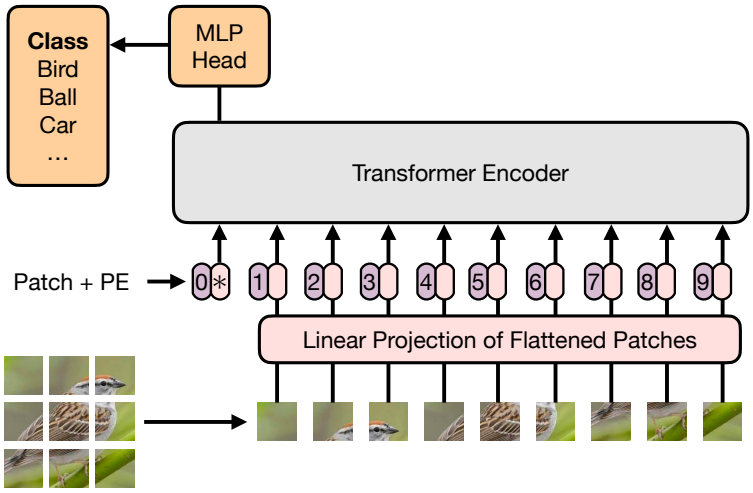


Figure 2.26: Architecture of Vision Transformer. This figure is based on [79]

In multi-head (MH) attention, H attention computations are performed in parallel, each attention h is called a head and is computed as

$$\text{head}_h = \mathcal{A}(QW_h^Q, KW_h^K, VW_h^V). \tag{2.113}$$

These head results are combined, and the output of the multi-head attention is

$$A = \mathcal{A}^{\text{MH}}(Q, K, V) = [\text{head}_1, \cdots, \text{head}_H]W^O. \tag{2.114}$$

Masked Multi-head Attention The Transformer inputs the true outputs to the decoder during training to allow fast parallel computation. Masking input parts that should not be visible enables parallel computation without cheating. For example, at "This" in the previous translation example, "is", "a", "pen", <EOS> are masked and only <BOS>, This is visible. This ensures that the output is equivalent to that obtained by executing in series without masking. The masked MH-attention is an attention mechanism that guarantees parallel computation on the decoder side by masking a part of the input series.

2.10.3 Vision Transformer

Transformers, which first appeared in the field of natural language processing, have flowed into the field of imaging and are being actively studied under the name Vision Transformer (ViT). Figure 2.26 shows the architecture of ViT. ViT divides the input image into small square regions called patches and uses a vector obtained by linearly transforming each patch as a token. The class tokens are then added to the vector and input to the Transformer encoder to obtain features that can be applied to downstream tasks such as class classification.

ViT is used for a wide range of image-based tasks such as image classification [387, 69, 409, 413], semantic segmentation [394, 374, 375, 58, 18], pose estimation [402, 41, 216, 398, 399, 400, 420], and depth estimation [299, 330, 29, 165, 410]. Many derivative methods have been proposed, such as the Swin Transformer [224, 223], which improves on ViT's windowing method, and DAT [392], which realizes deformable attention.

2.10.4 Transformers for Time-series Modeling

Transformers are also frequently used for time-series tasks such as time-series forecasting [204, 426, 175, 389, 390, 427, 386, 207, 209, 349, 219, 288, 222], time-series classification [353, 138, 333], and event detection [382]. The disadvantage of Transformers is their high computational and memory complexity, which is of

the order $O(T^2)$, i.e., the square of the time series length T. LogTrans [204] reduces the computational complexity to $O(T \log T)$ through a sparse bias using time causal convolution. Informer [426] and Reformer [175] similarly achieve $O(T \log T)$ computational complexity by limiting computation to elements with high query and key similarity. Autoformer [389] reduces the computational complexity to $O(T \log T)$ by using seasonal and trend decomposition. On the other hand, there are attempts to improve prediction accuracy through original mechanisms further. AST [390] reduces error accumulation by directly shaping the output distribution using an adversarial framework. FEDformer [427] and ETSformer [386] extract time and frequency features by applying an attention mechanism in the frequency domain. TFT [207] captures local and global dependencies to achieve multi-horizon prediction. SSDNet [209] and ProTrans [349] provide interpretability in combination with State Space Models (SSM). Pyraformer [219] acquires relationships at various time resolutions with hierarchical attention. Aliformer [288] provides knowledge-based attention and noise reduction. Non-stationary Transformers [222] incorporate non-stationarity in the attention computation.

2.11 Hyperparameter Optimization

The parameters of the NNs are optimized by gradient descent, but there are hyperparameters that are not optimized. Hyperparameters can be divided into three main categories: those that determine the learning behavior, the strength of regularization, and the architecture. Hyperparameters that determine learning behavior include the number of epochs, mini-batch size, learning rate η, and β,ρ,ε included in each optimization method described in Section 2.4. Hyperparameters determining the strength of regularization include patience for early stopping, the coefficient of weight decay, and the dropout ratio. Hyperparameters that determine the architecture include the number of layers and units, the kernel size and number of channels in the convolutional layer, and the ViT window size. The architecture design, such as the activation function and batch normalization, can also be considered a hyperparameter, and neural architecture search (NAS), which searches for architectures, has been actively studied.

Three main methods for determining these hyperparameters are grid search, randomized search, and automatic optimization. Grid search is a method that determines several candidates for each hyperparameter and tries all combinations. For example, when searching for the learning rate η $in\{10^{-n}|n \in \{1,2,3\}\}$, mini-batch size $b_s \in \{2^n|n \in \{2,3,4\}\}$ and dropout ratio $p \in \{0.1, 0.2, 0.5\}$, $3^3 = 27$. NNs are trained in $3^3 = 27$ ways. Randomized search selects each hyperparameter independently and identically, with a generator distribution for each hyperparameter. For example, the dropout ratio is assumed to follow a uniform distribution $U[0,1]$, the batch normalization is assumed to follow a Bernoulli distribution $Be(0.5)$, and the mini-batch size follows a power transform 2^n of variable n following a discrete uniform distribution $DU[2,4]$, and they are independently sampling.

2.11.1 Bayesian Optimization

Bayesian optimization (BO) is often used for automatic optimization of hyperparameters. BO is a powerful technique for optimizing expensive-to-evaluate functions. It is especially suited for scenarios where evaluating the function (often called the "objective function") is time-consuming, costly, or noisy. At the heart of Bayesian optimization is the Gaussian Process (GP) [300]. A GP is a non-parametric method used to define a distribution over functions. It provides a probabilistic way to guess the shape of the function we're trying to optimize based on the data points we have observed. A GP is fully characterized by a mean function and a covariance (or kernel) function. The GP gives a predicted value (mean) and an uncertainty (variance) for any given input point.

A crucial component of Bayesian optimization is the acquisition function. Given the current GP model of the objective function, the acquisition function helps decide where to sample next. It provides a trade-off between exploring areas of high uncertainty and exploiting areas of low estimated objective values. Common acquisition functions include Expected Improvement (EI), Probability of Improvement (PI), and Upper Confidence Bound (UCB).

The process of BO is as follows (also shown in Figure 2.27).

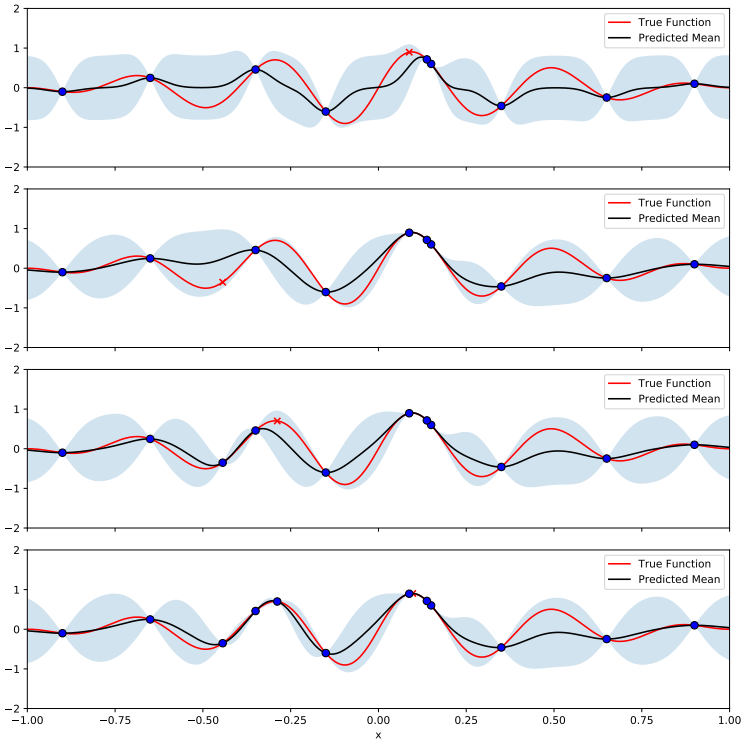


Figure 2.27: One-dimensional Bayesian optimization

1. Initialization: Start with a few initial data points, typically chosen randomly or based on prior knowledge.
2. Build GP Posterior: Using the observed data, build a GP posterior model of the objective function. This provides a mean and variance (uncertainty) estimate for each point in the input space.
3. Choose Next Point with Acquisition Function: Use the acquisition function to determine the next point to sample. This decision balances exploration (sampling where uncertainty is high) and exploitation (sampling where the function value is expected to be optimal).
4. Sample Objective Function: Evaluate the true objective function at the chosen point.
5. Update GP: Add the new data point to your dataset and update the GP.
6. Iterate: Repeat steps 2-5 for a predefined number of steps or until a convergence criterion is met.

Hyperparameter optimization using Bayesian optimization is implemented in libraries such as Optuna [4], Hyperopt [30], and Scikit-Optimize [130], and can be easily incorporated into existing code.

3 Representation Learning

Representation learning (RL) aims to acquire appropriate representations or features from data to describe that data. For example, RL acquires representations of the animal's shape, hair volume, size, color, pattern, and habitat from the animal image dataset. For linguistic data, we can obtain representations of the similarity of meaning between words, modification relations, order of occurrence, and co-occurrence, among other things. From sensor data, such as electricity consumption and seismic data, we can obtain expressions such as time-series patterns, relationships between channels, and time-delay effects. These representations can be used for downstream tasks, eliminating the need to train large-scale NNs for each task. Downstream tasks are those tasks that use the representations obtained by upstream representation learning. For example, once the representations of face images are obtained, they can be used for tasks such as personal identification, body temperature estimation, and facial skeleton estimation (Figure 3.1). Recently, there has been a significant boom in machine learning to acquire representations that can be applied to various problems.

Models that have been pre-trained with large-scale representations and have representations that can be transferred to various tasks are called foundation models. Foundation models include language models such as GPT-4 [277], Giraffe [281], and BERT [75], image models such as DALLE-2 [298] and Stable Diffusion [307], speech and music models such as MERT [205] and MusicLM [3], and multimodal models such as CLIP [293] and BLIP [200, 199]. These models have become indispensable tools in today's society, as ChatGPT and Github Copilot exemplified, and are expected to be used to realize general-purpose artificial intelligence (AGI).

Representation learning can be broadly classified into supervised and unsupervised representation learning. Supervised representation learning is a method that uses the outputs of the middle layer of NNs trained by supervised learning as representations. For example, when an image classification task is trained on ResNet [127], the output before the last fully connected layer or before the GAP layer is often used as the representation (Figure 3.2). This is because nonlinear low-dimensional features are acquired before the last linear layer when training individual supervised tasks. Supervised representation training is sometimes performed by branching into multiple tasks before the final layer. Unsupervised representational learning is a method without classification labels or explicit regression targets. It includes self-supervised learning, variational auto-encoder, and contrastive learning. This chapter focuses on these methods, described in turn in each section.

3.1 Self-supervised Learning

Self-supervised learning is a method of learning a representation by setting the teacher's data from the data itself. As shown in Figure 3.3, it is easy to grasp the image of time series data. When solving a time series prediction problem with NNs, such as predicting future values from past data, the features obtained along the way are represented. Not only prediction from the past to the future but also from the present to the past and from the past and future to the present can be regarded as self-supervised learning.

BERT [75] and GPT [277], the two major trends in language modeling, are also trained by self-supervised learning. GPT's key learning task is next-word prediction, which predicts the next word from past word

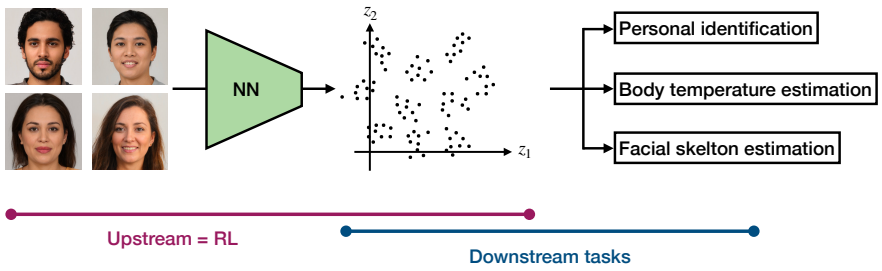


Figure 3.1: Schematic image of representation learning

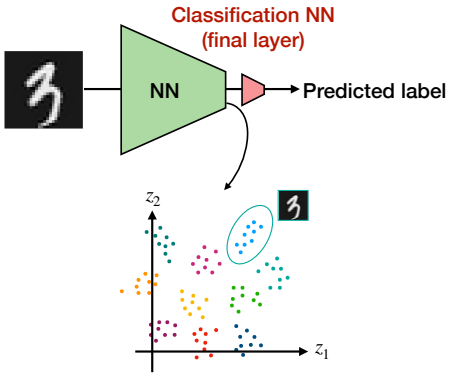


Figure 3.2: Supervised representation learning

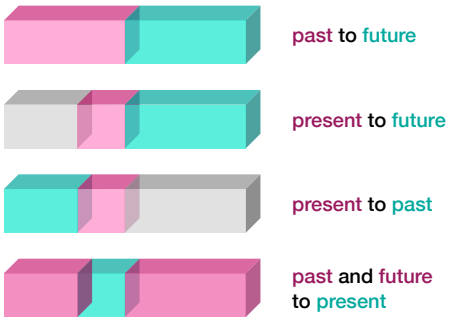


Figure 3.3: Self-supervised learning

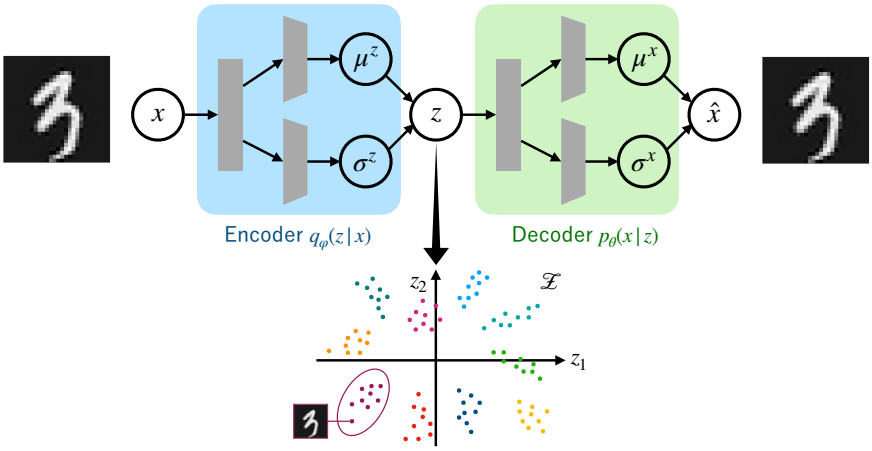


Figure 3.4: VAE architecture

sequences. BERT is pre-trained with a masked language model, which masks words and predicts them from the preceding and following sentences. Unlike individual tasks such as machine translation, sentiment estimation, or sentence summarization, these models require no teacher data. They can be trained on large corpora on the web, giving rise to today's large-scale language models.

3.2 Auto-Encoder

Auto-encoder (AE) is the predecessor to variational auto-encoder and consists of an encoder $E_{\boldsymbol{\theta}}$ and a decoder $D_{\boldsymbol{\theta}}$. The purpose of the encoder $E_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Z}$ is to extract information, often by mapping the observation space $\mathcal{X} = \mathbb{R}^{d_x}$ to a lower dimensional latent space $\mathcal{Z} = \mathbb{R}^{d_z}$. The purpose of the decoder $D_{\boldsymbol{\theta}} : \mathcal{Z} \to \mathcal{X}$ is to reconstruct the observation, mapping from the latent space to the observed space. AE is often used for dimensionality reduction and feature extraction since it can compress only the information necessary to reconstruct an observation to a lower dimension. The most basic AE loss function is the MSE:

$$l(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{1}{2}\|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2 = \frac{1}{2}\|\boldsymbol{x} - D_{\boldsymbol{\theta}} \circ E_{\boldsymbol{\theta}}(\boldsymbol{x})\|^2. \tag{3.1}$$

3.3 Variational Auto-Encoder

In recent years, generative AI has become a social phenomenon, with image generative AI such as stable diffusion [307] and DALL-E[298], and chat AI such as ChatGPT [220] and LLaMA [356, 357] becoming pop-

ular. Behind these generative AIs are generative models that describe the process of observation generation. Generative models using deep learning are called deep generative models and include variational auto-encoder (VAE) [174], generative adversarial network (GAN) [110], flow-based neural networks (Flows) [179], diffusion model (DM) [134], and consistency model (CM) [337]. VAE is a deep generative model suitable for learning representations because of its ability to acquire low-dimensional representations.

AE combines the simplicity of implementation and learning stability but overfitting and low-generation performance issues. VAE solves these problems by treating observed and latent variables as random variables. The structure of VAE is shown in Figure 3.4. Specifically, it assumes that the observed and latent variables follow a Gaussian distribution, and the output of NNs is the mean and variance, which are distribution parameters. The encoder $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ is called the inference model (recognition model, encoding model) and maps from the observations $\boldsymbol{x}$ to the latent parameters $(\boldsymbol{\mu}^z, \boldsymbol{\sigma}^z)$. Latent variables are generated by sampling from the distribution $N(\boldsymbol{\mu}^z, (\boldsymbol{\sigma}^z)^{\circ 2})$. The decoder $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ is called the generative model (decoding model) and maps latent $\boldsymbol{z}$ to observed parameters $(\boldsymbol{\mu}^x, \boldsymbol{\sigma}^x)$. Because VAE samples the latent variables independently at each iteration, overfitting is unlikely to occur because of noise effects and the diversity of inputs from which the generative model can be viewed. By assuming in advance a prior $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ for the latent variables, it is possible to generate unknown observed data.

3.3.1 Reparametrization Trick

In VAEs, during the forward pass, we want to sample from the latent space distribution $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ (usually a Gaussian) that is parameterized by the encoder's outputs (mean $\boldsymbol{\mu}^z$ and variance $\boldsymbol{\sigma}^z$). However, directly sampling from this distribution introduces stochasticity, which makes it difficult to backpropagate gradients. Instead of directly sampling $\boldsymbol{z}$ from $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$, the variable is reparametrized in such a way that the randomness is external to the model parameters. The trick involves two steps:

1. Sample $\boldsymbol{\varepsilon}$ from a standard Gaussian distribution $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, I)$.
2. Compute $\boldsymbol{z}$ using the deterministic transformation:

$$\boldsymbol{z} = \boldsymbol{\mu}^z + \boldsymbol{\sigma}^z \odot \boldsymbol{\varepsilon} \tag{3.2}$$

The randomness is decoupled from parameters $\boldsymbol{\mu}^z$ and $\boldsymbol{\sigma}^z$. The gradients can flow through $\boldsymbol{\mu}^z$ and $\boldsymbol{\sigma}^z$ without being hindered by the sampling operation, as $\boldsymbol{\varepsilon}$ is an external source of randomness.

The reparametrization trick works because it effectively separates the deterministic and stochastic parts of the model. The deterministic part, which depends on $\boldsymbol{\mu}^z$ and $\boldsymbol{\sigma}^z$, is differentiable and can be optimized using gradient-based methods. The stochastic part, which involves sampling $\boldsymbol{\varepsilon}$, doesn't depend on any model parameters, so it doesn't interfere with the gradient calculations.

3.3.2 Variational Inference

VAE optimizes parameters $(\boldsymbol{\varphi}, \boldsymbol{\theta})$ using a learning technique called variational inference (VI) [162, 22]. Variational inference is a method that approximates the maximization of the marginal log-likelihood $\log p_{\boldsymbol{\theta}}(\boldsymbol{x})$, which is computationally infeasible, by maximizing its lower bound, the evidence lower bound (ELBO)

$$\mathcal{L}^{\mathrm{ELBO}}(X, \boldsymbol{\varphi}, \boldsymbol{\theta}) := \mathbb{E}_{q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}\right] \leq \log p_{\boldsymbol{\theta}}(x). \tag{3.3}$$

This equation can be shown by Jensen's inequality from the concavity of the logarithmic function. The ELBO equation can be written as

$$\begin{aligned}
\mathcal{L}^{\mathrm{ELBO}}(X, \boldsymbol{\varphi}, \boldsymbol{\theta}) &= \mathbb{E}_{q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}\right] && (3.4)\\
&= \mathbb{E}_{q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] - \mathrm{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x}) \| p_{\boldsymbol{\theta}}(\boldsymbol{z})) && (3.5)\\
&= \log p_{\boldsymbol{\theta}}(x) - \mathrm{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x}) \| p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})). && (3.6)
\end{aligned}$$

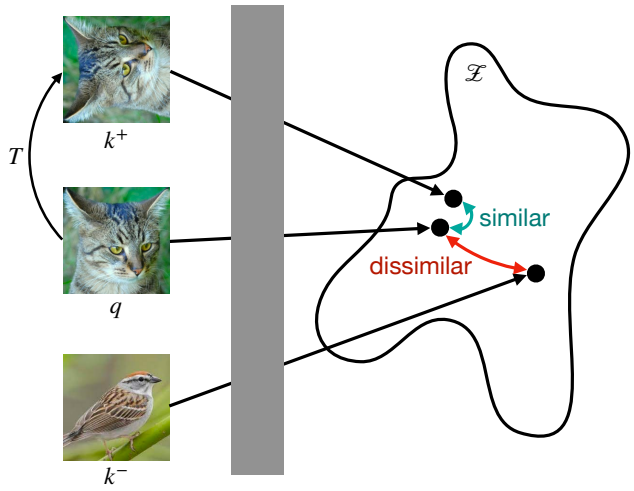


Figure 3.5: Simple image of contrastive learning

Equation (3.5) is the negative expected loss, the form used in actual training. The first term is the expected log-likelihood, the maximization of which implies learning a generative model such that plausible observations are obtained. The second term is the KL divergence between the inference distribution $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ and the prior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$, which serves as a regularization that prevents the inference distribution from deviating from the prior distribution given beforehand. Equation (3.6) implies that maximizing ELBO is equivalent to minimizing the KL divergence of the approximate posterior distribution $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ and the posterior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$. In other words, it estimates the approximate posterior distribution $q_{\boldsymbol{\varphi}^*}(\boldsymbol{z}|\boldsymbol{x})$ with the smallest deviation from the posterior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ in the variational function space $\{q_{\boldsymbol{\varphi}}|\boldsymbol{\varphi} \in \Phi\}$.

Some advanced VI methods have been proposed to achieve tighter bounds of the log marginal likelihood [414]. IWAE [39, 77] achieves tighter bounds by describing the ELBO by averages over multiple particles. TVO [244] bridges thermodynamic integration and VI. SIVI [407], DSIVI [257], and UIVI [354] expand the applicability of VI by defining expressive variational families. Some methods focus on specific model structures such as stochastic differential equations [315] and network autoregression models [192].

3.3.3 Disentanglement

Disentanglement, which separates the representations into axes, is important in representation learning. For example, in animal image data, each meaningful representation, such as animal skin color, skin pattern, hair color, hair volume, eye shape, nose shape, ear shape, and face shape, is represented on a different axis. VAE uses an uncorrelated covariance structure of $\mathcal{N}(\mathbf{0}, I)$ as a prior distribution of latent variables, which makes it easy to obtain a disentangled representation. The β-VAE [132], which strengthens the restriction to follow the prior distribution of the latent variable, and the object-centric representation [225], which gives a representation for each object, have been proposed.

3.4 Contrastive Learning

Contrastive learning is an approach in representation learning that aims to learn embeddings or representations of data by comparing similar and dissimilar pairs. The goal is to ensure that similar data points are brought closer in the embedded space while dissimilar or contrasting data points are pushed apart. Figure 3.5 shows a simple image of the method. The method has three inputs: anchor $\boldsymbol{q}$, positive example $\boldsymbol{k}^+$, and negative example $\boldsymbol{k}^-$. With the anchor as the axis, positive examples are data similar to the anchor, and negative examples are data different from the anchor. For example, if the anchor is an image, the positive example is an image obtained by rotating, coloring, or transforming the image, and the negative example is

an entirely different image. These images are passed through an encoder such as ResNet, and the resulting representations are learned to be similar or dissimilar.

A typical measure of similarity is the cosine similarity

$$\mathrm{cossim}(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{\|\boldsymbol{x}\| \cdot \|\boldsymbol{y}\|}, \tag{3.7}$$

where $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$. The similarity is determined by the angle between the vectors and corresponds to the similarity of the vector orientations. Since it is desirable to have a high similarity of positive examples to negative examples, noise contrastive estimation (NCE) [115]

$$l^{\mathrm{NCE}}(\boldsymbol{q}, \boldsymbol{k}^+, \boldsymbol{k}^-) = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{q}, \boldsymbol{k}^+)/\tau)}{\exp(\mathrm{sim}(\boldsymbol{q}, \boldsymbol{k}^+)/\tau) + \exp(\mathrm{sim}(\boldsymbol{q}, \boldsymbol{k}^-)/\tau)} \tag{3.8}$$

is used as a loss function, where $\tau \in \mathbb{R}_+$ is a temperature parameter and $\mathrm{sim} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a similarity function. Since negative examples are overwhelmingly more common than positive examples, InfoNCE [362] uses more negative examples $\{\boldsymbol{k}_i\}_{i=1}^K$ by

$$l^{\mathrm{InfoNCE}}(\boldsymbol{q}, \boldsymbol{k}^+, \{\boldsymbol{k}_i\}) = -\log \frac{\exp(\mathrm{sim}(\boldsymbol{q}, \boldsymbol{k}^+)/\tau)}{\exp(\mathrm{sim}(\boldsymbol{q}, \boldsymbol{k}^+)/\tau) + \sum_{i=1}^K \exp(\mathrm{sim}(\boldsymbol{q}, \boldsymbol{k}_i)/\tau)}. \tag{3.9}$$

3.4.1 Pretext Task

Transfer learning is the transfer of representations acquired while solving one task to solve another task. In representation learning, the pretext task to be solved in advance does not require labels. For example, the language model BERT masks a word and predicts the word. In contrastive learning, setting the optimal pretext task for learning representations is also necessary.

CL's pretext task is to increase similarity with positive examples and decrease similarity with negative examples. This comes down to data augmentation, i.e., how to generate positive example images. SimCLR [51], a typical CL method, uses color transformations such as image blurring, color distortion, and noise addition, as well as shape transformations such as image rotation and flipping. PIRL [253] learns features by shuffling images patch by patch like a jigsaw puzzle. For series data such as video, clips from the same video or observations at nearby points in time are used as positive examples [289, 159, 147].

3.4.2 Architecture

The larger the number of negative examples K in CL, the faster the learning progresses, but a large K is memory-consuming. There is also a solution that stores previous embeddings, but this time, the negative examples must be recomputed each time the parameters are updated, which is computationally expensive [253]. MoCo (Momentum Contrast) [126, 54, 57] solves these problems by storing features in a queue to update them smoothly. BYOL [114], SimSiam [55] solved these problems by using two types of encoders called projector and predictor, that only approximate positive examples and queries without using negative examples. CL has another problem: images of the same class are treated as negative examples. SwAV [42], PCL [202], and SMOG [282] solved this problem by learning while clustering in the representation space.

4 Representation Learning for Sequential Data

Time series data have different frequency characteristics, pattern characteristics, and non-stationarity for each data, making uniform modeling difficult, and no practical foundation model has been constructed. For example, temperature time series have a seasonality that rises during the day and falls at night, but many components cannot be explained by the trend or seasonal components. This component may be viewed as noise, or it may be considered to be a significant manifestation of chaos, requiring modeling that is appropriate to the situation. Stock price time series has a large volatility component, and modeling that considers extreme value components, such as the Lehman shock, is sometimes necessary. Sensor data from ECGs and factories need to be modeled with the unique property of quasi-periodicity, which lies between periodicity and non-periodicity. In object tracking of moving images, it is necessary to separate the foreground (moving objects) from the background (buildings, trees, etc.), and in a time series of athletes' actions, changes in discrete states (running, jumping, swinging, etc.) are important. It is important to separate slow time-scale phenomena in protein structure trajectories, and in communication networks, we want to capture fast changes. Representation learning for sequential data requires extracting such time-series characteristics according to the data.

The three main types of representation learning for sequential data are Transformers, sequential VAEs (SVAEs), and time contrastive learnings (TCLs). This chapter describes each of these in turn.

4.1 Transformers for Sequential RL

As mentioned in section 2.10.2, Transformer is an encoder-decoder architecture, so encoded features can be viewed as representations. BERT [75] and its variants [221, 129, 404, 64] use only the Transformer encoder to obtain the language representation. The attention mechanism used in the Transformer can capture the similarities and relationships among time series patterns. In Chapter 5, we propose a method to extract quasi-periodicity from the attention mechanism and use it for anomaly detection and period estimation.

4.2 Sequential Data Assimilation

Data assimilation (DA) is a method of obtaining new knowledge by combining deductive simulation and inductive experimental observation. The method is used to determine optimal initial and boundary conditions for simulations, estimate parameters of simulation models, interpolate non-observed spatio-temporal points, and perform sensitivity analysis. The method has been developed mainly in meteorology, and today's weather forecasting and typhoon path prediction are computed with data assimilation. Today, it is used not only in meteorology but also in a wide range of fields such as chemistry and medicine [141, 304].

4.2.1 State Space Model

State space models (SSMs) form the core of sequential data assimilation. The model is formulated as a time series of observed processes with latent state transition processes (system model) $p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})$ and observation generation processes (observation model) $p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t)$:

$$p_{\boldsymbol{\theta}}(X, Z) = p_{\boldsymbol{\theta}}(\boldsymbol{z}_1) \cdot \left(\prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) \right) \cdot \left(\prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t) \right), \tag{4.1}$$

where $\boldsymbol{z}_t \in \mathbb{R}^{d_z}$ and $\boldsymbol{x}_t \in \mathbb{R}^{d_x}$. This formulation (4.1) is called a general state-space model in the context of statistics, and when referred to as a state-space model, it is formulated as

$$\boldsymbol{z}_t = f_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}) + \boldsymbol{v}_t,\ \boldsymbol{v}_t \sim p_{\boldsymbol{\theta}}(\boldsymbol{v}_t), \tag{4.2}$$

$$\boldsymbol{x}_t = g_{\boldsymbol{\theta}}(\boldsymbol{z}_t) + \boldsymbol{w}_t,\ \boldsymbol{w}_t \sim p_{\boldsymbol{\theta}}(\boldsymbol{w}_t), \tag{4.3}$$

where $\boldsymbol{v}_t \in \mathbb{R}^{d_z}$ and $\boldsymbol{w}_t \in \mathbb{R}^{d_x}$ are additive noise, $\boldsymbol{f}_{\boldsymbol{\theta}} : \mathbb{R}^{d_z} \to \mathbb{R}^{d_z}$ is transition operator, $\boldsymbol{g}_{\boldsymbol{\theta}} : \mathbb{R}^{d_z} \to \mathbb{R}^{d_x}$ is observation operator.

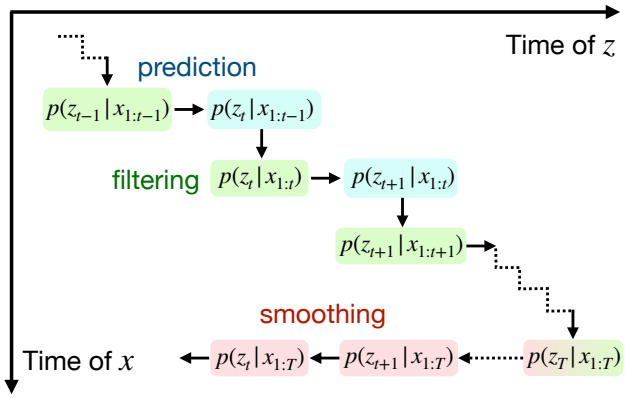


Figure 4.1: Schematic image of sequential Bayes filtering

4.2.2 Sequential Bayes Filter

SSMs have three important distributions: predicted, filtered, and smoothed distributions. These distributions can be described consistently as $p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:s})$ and are called differently depending on the amount of observed information s. When $s < t$, they are called predicted distributions; when $s = t$, filtered distributions; and when $s > t$, smoothed distributions. A predictive distribution represents future conditions based on past observations; for example, it is a distribution of future atmospheric conditions based on past meteorological observations. A filtered distribution is a distribution that represents the current state based on observations up to the present and is used to estimate the state reflecting real-time information. A smoothed distribution is a distribution that represents the past state based on observations up to the present and is used for reanalysis.

As shown in Figure 4.1, these distributions are obtained by the Markov property of SSM and Bayes' theorem in the form of recurrence formulas called sequential Bayes filter (SBF)

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-1}|\boldsymbol{x}_{1:t-1})\mathrm{d}\boldsymbol{z}_{t-1}, \tag{4.4}$$

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:t}) = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t)p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1})}{\displaystyle\int p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t)p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1})\mathrm{d}\boldsymbol{z}_t}, \tag{4.5}$$

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:T}) = p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:t}) \int \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{1:T})}{p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t+1}|\boldsymbol{x}_{1:t})}\mathrm{d}\boldsymbol{z}_{t+1}. \tag{4.6}$$

In practice, due to the complexity of the integral calculations, they are replaced by introducing hypotheses or approximations to the distribution.

4.2.3 Kalman Filter

The Kalman filter reduces the computation of the SBF to the computation of the mean vector and variance-covariance matrix by assuming a linear Gaussian state-space model (LGSSM)

$$\boldsymbol{z}_t = F_t\boldsymbol{z}_{t-1} + \boldsymbol{v}_t,\ \boldsymbol{v}_t \sim \mathcal{N}(\boldsymbol{0}, Q_t), \tag{4.7}$$

$$\boldsymbol{x}_t = G_t\boldsymbol{x}_t + \boldsymbol{w}_t,\ \boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{0}, R_t), \tag{4.8}$$

where $F_t, Q_t \in \mathbb{R}^{d_z \times d_z}$, $G_t \in \mathbb{R}^{d_x \times d_z}$, and $R_t \in \mathbb{R}^{d_x \times d_x}$. Assuming that the initial distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z}_1)$ is Gaussian, we can prove that each predictive, filter, and smoothing distribution is Gaussian. Assuming the distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:s}) = \mathcal{N}(\boldsymbol{z}_{t|s}, V_{t|s})$, the update formulas of SBF for predicting and filtering are

$$\boldsymbol{z}_{t|t-1} = F_t\boldsymbol{z}_{t-1|t-1}, \tag{4.9}$$

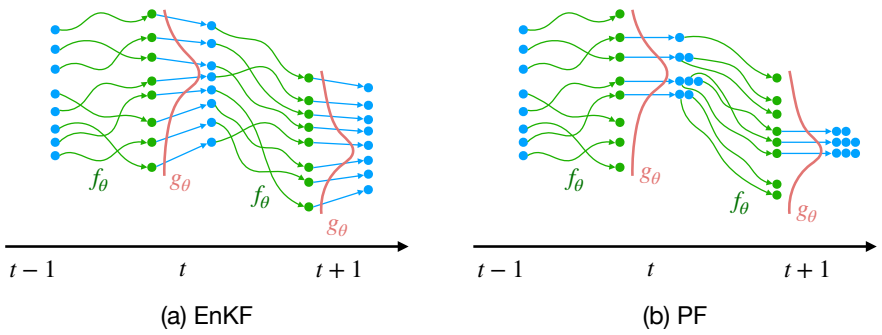


Figure 4.2: Filtering process of (a) EnKF and (b) PF

$$V_{t|t-1} = F_t V_{t-1|t-1} F_t^T + Q_t, \tag{4.10}$$
$$K_t = V_{t|t-1} G_t^T (G_t V_{t|t-1} G_t^T + R_t)^{-1}, \tag{4.11}$$
$$\boldsymbol{z}_{t|t} = \boldsymbol{z}_{t|t-1} + K_t(\boldsymbol{x}_t - G_t \boldsymbol{z}_{t|t-1}), \tag{4.12}$$
$$V_{t|t} = V_{t|t-1} - K_t G_t V_{t|t-1}, \tag{4.13}$$

where $K_t \in \mathbb{R}^{d_z \times d_x}$ is called Kalman gain, which corrects the state toward the observation in the observation space. These updated formulas are called the Kalman filter (KF).

Originally proposed in the field of control engineering, KF has been used in a wide range of fields, including robotics, civil engineering, econometrics, and cognitive science, because of its simplicity and usefulness [166, 203]. Concerning SVAE, which will be discussed in section 4.3, there have been many attempts to integrate KF with deep learning, such as Kalman VAE [92] and Deep Kalman Filter [185].

4.2.4 Probabilistic Time-Series Model

Probabilistic time-series model (PTSM) is a non-Markovian extension of SSM and is described as

$$p_{\boldsymbol{\theta}}(X, Z) = p_{\boldsymbol{\theta}}(\boldsymbol{z}_1) \prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_t | \boldsymbol{z}_{1:t-1}) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_t | \boldsymbol{z}_t). \tag{4.14}$$

Since the SVAEs discussed in section 4.3 are based on PTSM, the following two subsections 4.2.5 and 4.2.6 also refer to filtering methods based on PTSM.

4.2.5 Ensemble Kalman Filter

KF cannot be used for complex nonlinear processes because the LGSSM bounds the available models. The ensemble Kalman filter (EnKF) and particle filter (PF) can be applied to general nonlinear and non-Gaussian SSM by making a Monte Carlo approximation of each state distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z}_t | \boldsymbol{x}_{1:s})$ using particles $\{\boldsymbol{z}_{t|s}^{(i)}\}_{i=1}^N$:

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_t | \boldsymbol{x}_{1:s}) \approx \frac{1}{N} \sum_{i=1}^{N} \delta\left(\boldsymbol{z}_t - \boldsymbol{z}_{t|s}^{(i)}\right), \tag{4.15}$$

where δ denotes the Dirac's delta function. The difference between EnKF and PF is shown in Figure 4.2. EnKF updates each particle to fit the observation, whereas PF replicates particles based on likelihood. This difference will be described later in the proposed method in Chapter 6.

The original paper [88] assumes a linear observation model with additive noise

$$\boldsymbol{z}_t \sim p_{\boldsymbol{\theta}}(\boldsymbol{z}_t | \boldsymbol{z}_{t-1}), \tag{4.16a}$$

$$\boldsymbol{x}_t = G_t \boldsymbol{z}_t + \boldsymbol{w}_t,\ \boldsymbol{w}_t \sim p_{\boldsymbol{\theta}}(\boldsymbol{w}_t). \tag{4.16b}$$

The EnKF updates the particles by

$$\boldsymbol{z}_{t|t-1}^{(i)} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1|t-1}^{(i)}),\ \forall i \in \mathbb{N}_N \tag{4.17a}$$

$$K_t = \Sigma_{t|t-1}^z G_t^T (G_t \Sigma_{t|t-1}^z G_t^T + \Sigma_t^w)^{-1}, \tag{4.17b}$$

$$\boldsymbol{z}_{t|t}^{(i)} = \boldsymbol{z}_{t|t-1}^{(i)} + K_t(\boldsymbol{x}_t - G_t \boldsymbol{z}_{t|t-1}^{(i)} - \boldsymbol{w}_t^{(i)}),\ \forall i \in \mathbb{N}_N, \tag{4.17c}$$

where $\Sigma_{t|s}^z$ and Σ_t^w represent the sample covariance of $\{\boldsymbol{z}_{t|s}^{(i)}\}$ and $\{\boldsymbol{w}_t^{(i)}\}$, respectively.

The original EnKF is easily applied to the nonlinear observation model

$$\boldsymbol{x}_t \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t) \tag{4.18a}$$

$$\overset{\mathbb{E}[g_{\boldsymbol{\theta}}(\boldsymbol{z}_t)]\ exists}{\Longleftrightarrow} \boldsymbol{x}_t = \mathbb{E}[g_{\boldsymbol{\theta}}(\boldsymbol{z}_t)] + (\boldsymbol{x}_t - \mathbb{E}[g_{\boldsymbol{\theta}}(\boldsymbol{z}_t)]),\ \boldsymbol{w}_t = \boldsymbol{x}_t - \mathbb{E}[g_{\boldsymbol{\theta}}(\boldsymbol{z}_t)] \sim p_{\boldsymbol{\theta}}(\boldsymbol{w}_t) \tag{4.18b}$$

$$\Longleftrightarrow \boldsymbol{x}_t = g_{\boldsymbol{\theta}}(\boldsymbol{z}_t) + \boldsymbol{w}_t,\ \boldsymbol{w}_t \sim p_{\boldsymbol{\theta}}(\boldsymbol{w}_t) \tag{4.18c}$$

by the augmented PTSM

$$\begin{aligned}\tilde{\boldsymbol{z}}_t &= \begin{pmatrix} \boldsymbol{z}_t \\ g_{\boldsymbol{\theta}}(\boldsymbol{z}_t) \end{pmatrix} \\ &\sim \tilde{p}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{z}}_t|\tilde{\boldsymbol{z}}_{1:t-1}) = p_{\boldsymbol{\theta}}\left(\begin{pmatrix} I_{d_z} & O_{d_x} \end{pmatrix} \tilde{\boldsymbol{z}}_t | \begin{pmatrix} I_{d_z} & O_{d_x} \end{pmatrix} \tilde{\boldsymbol{z}}_{1:t-1}\right), \qquad (4.19a) \\ \boldsymbol{x}_t &= \begin{pmatrix} O_{d_z} & I_{d_x} \end{pmatrix} \tilde{\boldsymbol{z}}_t + \boldsymbol{w}_t = \tilde{G}_t \tilde{\boldsymbol{z}}_t + \boldsymbol{w}_t \\ &\sim \tilde{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\tilde{\boldsymbol{z}}_t) = p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_t | \begin{pmatrix} I_{d_z} & O_{d_x} \end{pmatrix} \tilde{\boldsymbol{z}}_t\right) \qquad (4.19b)\end{aligned}$$

for the augmented latent states $\tilde{\boldsymbol{z}}_t \in \mathbb{R}^{d_z+d_x}$. By equation (4.19b), the nonlinear emission is regarded as a linear representation; then, the augmented PTSM can be applied to equations (4.17).

It is generally known that the EnKF underestimates the state covariance matrix

$$\Sigma_{t|t}^z = \frac{1}{N-1} \sum_{i=1}^{N} (\boldsymbol{z}_{t|t}^{(i)} - \bar{\boldsymbol{z}}_{t|t})(\boldsymbol{z}_{t|t}^{(i)} - \bar{\boldsymbol{z}}_{t|t})^T \tag{4.20}$$

due to several factors such as limited particle size and model error, where $\bar{\boldsymbol{z}}_{t|s}$ is the average of $\{\boldsymbol{z}_{t|s}^{(i)}\}_{i=1}^N$. To overcome this problem, covariance inflation methods that inflate the covariance have been proposed [10, 254, 67, 416, 384]. The representative methods are multiplicative inflation [10], additive inflation [254, 67], and relaxation to prior [416, 384]. The relaxation to prior (RTP) methods relax the reduction of the spread of particles at the filtering step and are suitable for our proposed algorithm. The relaxation to prior perturbation (RTPP) [416] method blends the particles before and after filtering; the mathematical equation is

$$\tilde{\boldsymbol{z}}_{t|t}^{(i)} = \alpha \boldsymbol{z}_{t|t-1}^{(i)} + (1-\alpha)\boldsymbol{z}_{t|t}^{(i)}, \tag{4.21}$$

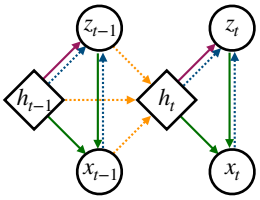
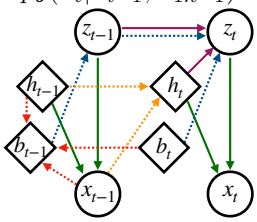
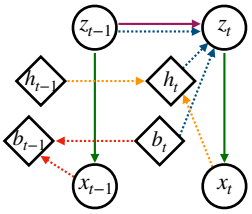
where $\alpha \in [0,1]$ is known as the inflation factor and $\tilde{\boldsymbol{z}}_{t|t}^{(i)}$ denotes the i-th latent particle after covariance inflation. The relaxation to prior spread (RTPS) [384] method multiplies the particles after filtering and inflation together; the formulation is

$$\tilde{\boldsymbol{z}}_{t|t}^{(i)} = (\alpha \boldsymbol{\sigma}_{t|t-1} + (1-\alpha)\boldsymbol{\sigma}_{t|t}) \oslash \boldsymbol{\sigma}_{t|t} \odot \boldsymbol{z}_{t|t}^{(i)}, \tag{4.22}$$

where $\alpha \in [0,1]$ is the inflation factor and $\boldsymbol{\sigma}_{t|s} \in \mathbb{R}^{d_z}$ is the sample variance of particles $\{\boldsymbol{z}_{t|s}^{(i)}\}_{i=1}^N$.

For EnKF combined with deep neural networks, Bayesian LSTM [47] represents the weights of LSTM by an ensemble of particles and propagates the particles using EnKF at each mini-batch, considered a time-step in the SSM. A similar framework [226] has been applied to time production prediction in natural gas wells, and robust estimators were obtained. Bayesian neural networks with EnKF [46] represent the distribution of the parameters by particles and update the particles using EnKF at each mini-batch.

Table 4.1: Examples of SVAEs

Network	VRNN [63]	SRNN [94]	SVO [260]
Encoder	$q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t \vert \boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:t})$	$q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t \vert \boldsymbol{z}_{t-1}, \boldsymbol{x}_{1:T})$	$q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t \vert \boldsymbol{z}_{t-1}, \boldsymbol{x}_{1:T})$
Decoder	$p_{\boldsymbol{\theta}}(\boldsymbol{x}_t \vert \boldsymbol{z}_{1:t}, \boldsymbol{x}_{1:t-1})$	$p_{\boldsymbol{\theta}}(\boldsymbol{x}_t \vert \boldsymbol{z}_t, \boldsymbol{x}_{1:t-1})$	$p_{\boldsymbol{\theta}}(\boldsymbol{x}_t \vert \boldsymbol{z}_t)$
Transition	$p_{\boldsymbol{\theta}}(\boldsymbol{z}_t \vert \boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:t-1})$	$p_{\boldsymbol{\theta}}(\boldsymbol{z}_t \vert \boldsymbol{z}_{t-1}, \boldsymbol{x}_{1:t-1})$	$p_{\boldsymbol{\theta}}(\boldsymbol{z}_t \vert \boldsymbol{z}_{t-1})$
Graph			

4.2.6 Particle Filter

EnKF was an updated formula that used up to the second-order moments of the state distribution and thus could not capture the strong nonlinearity. Particle filter (PF, SMC; sequential Monte Carlo, SIR: sequential importance resampling) [111, 176, 82] is an effective filtering method for strongly nonlinear and strongly non-Gaussian systems by using a larger number of particles and stochastically replicating them. The procedure for implementing the method is as follows

$$\boldsymbol{z}_{t|t-1}^{(i)} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z}_t | \boldsymbol{z}_{t-1|t-1}^{(i)}), \tag{4.23a}$$

$$w_t^{(i)} = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_t | \boldsymbol{z}_{t|t-1}^{(i)})}{\sum_{j=1}^{N} p_{\boldsymbol{\theta}}(\boldsymbol{x}_t | \boldsymbol{z}_{t|t-1}^{(j)})}, \tag{4.23b}$$

$$A_i \sim \text{MultiNomial}\left(\{w_t^{(i)}\}_{i=1}^{N} \right), \tag{4.23c}$$

$$\boldsymbol{z}_{t|t}^{(i)} = \boldsymbol{z}_{t|t-1}^{(A_i)}, \tag{4.23d}$$

where MultiNomial denotes the multinomial distribution.

While PF can be applied to arbitrary nonlinear, non-Gaussian problems and is easy to implement, it has the problem of particle degeneration. The problem is that only identical particles survive, and the diversity of particles decreases as time steps advance and resampling is repeated. This problem is more pronounced for medium to high-dimensional problems with ten or more dimensions due to the curse of dimensionality. Although this problem can be avoided by increasing the number of particles, the computational and spatial complexity becomes a bottleneck. Reducing the actual state dimension and adding noise regularization after resampling have been tried to alleviate the problem without much success. There are also methods to bias the particles to compare their weights. Implicit PF [62] encourages transitions that increase the likelihood of noise space. EWPF [363, 2] replaces sampling from exact posterior by sampling with equal particle weights. IEWPF [428] adjusts the target weights in EWPF to be optimal. AGMF [341] and EKPF [97] bridge EnKF and PF to achieve both strong nonlinearity and particle diversity.

4.3 Sequential Variational Auto-Encoder

Sequential VAEs (SVAEs) expand the generative model of VAE [174] to probabilistic time-series models and infer the models by variational inference. DKF [185], DMM [186], KVAE [93], E2C [380], and DynaNet [48] directly construct state space model and infer the latent probabilistic models. VRNN [63] and STORN [21] construct the transition model with RNNs such as GRU [61] and LSTM [135] to capture long-term dependencies. SRNN [94], Z-forcing [112], and SVO [260] introduce backward recursion to the forward model to capture future information. CW-VAE [321], TVA [172], RSSM [117], and SVG [74] mainly focus on image

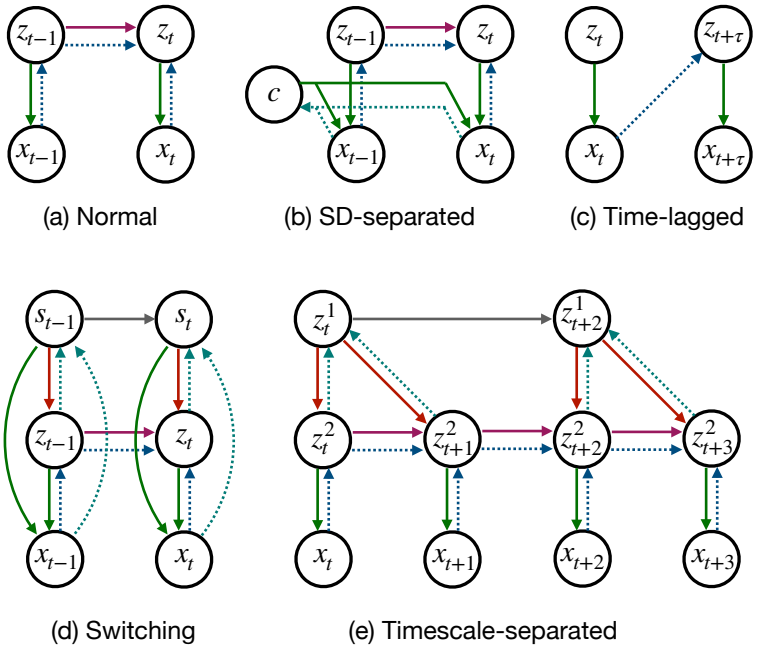


Figure 4.3: Variants of architectures of SVAEs

sequence data such as 3D maze [87] and moving handwritten data [339]. Okada et al. [275] and STAF [292] use SVAEs for pose tracking.

Table 4.1 shows probablistic models and structures of VRNN [63], SRNN [94], and SVO [260]. The dotted and solid lines represent inferential and generative processes, and the circles and squares represent stochastic and deterministic variables, respectively. While VRNN infers the latent states from past and current observations, SRNN and SVO use future observations. SVO restricts PTSM to SSM, which satisfies Markov property.

4.3.1 Static-Dynamic Separated System

There are methods to separate time-independent static representations from time-dependent dynamic representations (Figure 4.3 (b)). For example, moving images may not change the background, but only the foreground may move. The representation of the car as an object remains the same, but its position and rotation angle change. FHVAE [140] and DSVAE [408] pioneered these methods, with FAVAE [401] separating like factors and IDEL [60] minimizing the amount of mutual information between static and dynamic factors. S3VAE [429] separates only the invariant static factors by applying contrastive learning to the static factors and pseudo-supervised learning to the dynamic factors. R-WAE [118] utilizes Wasserstein AE, and C-DSVAE [15] introduces contrastive learning for both static and dynamic factors.

4.3.2 Time-lagged System

Time-lagged networks are used when you want to remove fast time-scale motion and extract slow time-scale motion (Figure 4.3 (c)). For example, a fixed-point camera on a street can be used to capture fast-moving people while ignoring slow-moving clouds. In biomolecules, slow structural dynamics are closely related to biological function. In those models, dynamics occurring at times shorter than τ steps are reduced by embedding observations $\boldsymbol{x}_t$ at time t into latent variables τ steps ahead $\boldsymbol{z}_{t+\tau}$. TAE [381] and TVAE [136] pioneered, VDE [131] introduced sample autocorrelation to loss, and VAMPNet [242] and SRV [53] focus on discrete embedding.

In Chapter 7, we propose a new SVAE for biomolecular structures. The model can capture sample-by-sample slower dynamics by introducing a simple transition prior.

4.3.3 Switching System

Switching SVAEs is a model that can capture steep changes by simultaneously acquiring continuous and discrete representations (Figure 4.3 (d)). For example, a stock price time series may have a discrete background due to rapid price fluctuations. The dynamics of a factory machine operating time series change depending on what is produced as the current lot. Such continuous-discrete systems have been considered in SSM for a long time as switching linear dynamical systems (SLDS), and it is natural to assume that they have been extended to SVAE. rSLDS [210] adds a recursive path from continuous to discrete states, and SLDS+ [24] modifies the initial distribution from SLDS. DWB [59] uses the Wasserstein distance for continuous mixture loss, and RES-SDS [11] merges rSDS and ED-SDS. DS3M [397] leverages backward RNNs, and SSNN [215] infers continuous and discrete states via intermediate representations. KVAE [92] infers discrete states with RNNs, and SRKN [270] allows switching between KVAEs in the latent space. DRBPF [191] is used as a learning technique for switching SVAEs.

4.3.4 Ensemble System

The generative model of SVAEs is often formulated as a PTSM, which can be constructed as a tighter variational lower bound using sequential data assimilation. FIVO [238], AESMC [196], and VSMC [265] proposed VI methods combined with sequential Monte Carlo (SMC). These methods are ensemble systems applied to SVAEs to infer more accurate models. Although TVSMC [194] and SMC-Twist [212] obtain tighter bounds by introducing twisted functions for capturing future information, experimental results revealed that these methods have lower log marginal likelihood compared to non-twisted methods. These SMC-based methods obtain tighter bounds for sequential data; however, they have two main drawbacks: particle degeneracy and biased gradient estimators.

PSVO [262, 261] proposed an inference framework using forward filtering backward simulation (FFBSi) [109, 82], a method to estimate the posterior conditioned on all observations in the SSM formulation, to achieve unbiased estimators and a tighter bound than SMC-based methods. The method, however, has three main disadvantages: particle degeneracy, high calculation and memory cost, and high sensitivity of the hyper-parameters. The first disadvantage also applies to SMC-based methods; the second one arises from long backward loss calculation and storage of all particles at the time forward calculation for the time backward calculation. The last one means PSVO needs careful tuning of the hyper-parameters due to training instability.

In Chapter 6, we propose EnKO, a variational inference method using EnKF. The proposed method overcomes the particle degeneracy and biased gradient estimator of SMC-based methods and the sensitivity to hyperparameters of PSVO.

4.4 Time Contrastive Learning

Time contrastive learning (TCL) originated from Hyvärinen et al. [147] but now generally refers to contrastive learning using time direction. Hyvärinen et al. [147] divide time into multiple segments and learn features by self-supervised learning that classifies which segment the data comes from. The method has been theoretically proven for identifiability [149] and extended to auxiliary variables [150]. Regarding identifiability, iVAE [169] integrates TCL and VAE as nonlinear ICA, and ICE-BeeM [170] evolves into a conditional energy model.

The representation learning method for biomolecular structural dynamics proposed in Chapter 7 can be regarded as a BYOL-type TCL without negative samples.

5 An Online System of Detecting Anomalies and Estimating Cycle Times for Production Lines

Energy consumption data of production machines often exhibit quasi-periodicity, and anomalies are observed when deviations from the quasi-periodicity are detected. It is crucial to quickly estimate the individual cycles at each time point and detect abnormalities in such data. In this study [156, 155], we propose a system that satisfies these requirements. The proposed system trains a neural network with an attention mechanism and applies the weight vectors in the mechanism to the two tasks. Experimental results demonstrate that the proposed method outperforms benchmark methods for sensor data that mimic the power consumption data of production lines.

5.1 Introduction

Industry 4.0, a new industrial stage that focuses on integrating vertical and horizontal manufacturing processes and the connection of things and the Internet, is attracting attention [70, 95, 361]. The smart factory proposed in Industry 4.0 is a system that facilitates the manufacturing process and enables efficient production by communicating among all machines and systems in the factory. Consider a production line where each machine is equipped with a smart meter that measures power consumption. If abnormal signals can be automatically detected from the time-series changes in the smart meters, the product defect rate can be reduced, and productivity can be improved.

Fig. 5.1 shows the time-series data of smart meters on a production line, where the time-series of each meter show a quasi-periodic pattern but does not include system information such as lot switching or stop status. For such data, in parallel with anomaly detection, the estimation of individual cycles, a key production performance indicator (KPPI), is strongly required. By estimating individual cycles, it is possible to estimate the degree of abnormality in the production line and detect lot switching.

There are two major approaches to anomaly detection for quasi-periodic time-series data, such as energy consumption data of production machinery. The first approach is a segment-based method that divides the time-series data into segments for one period and then performs classification using the features of the segments. In the first stage of segmentation, specific points that appear in one cycle, such as one spike in one cycle, are extracted and divided into segments using these points. In the second stage of anomaly detection, when the abnormal/normal teacher labels are given, a classification method such as naïve Bayes method [235] and Support Vector Machine [86] is used, and when not given, a clustering method such as DBSCAN [395], k-means method [422] is used. Considering that most of the data is normal, unsupervised anomaly detection methods such as the kNN method, the SVDD method [350], and deep learning methods have also been proposed [263, 264, 8]. The former classification method is difficult to apply in the energy consumption data of production machinery because of no teacher labels. The latter clustering method cannot return an immediate and accurate response because both stages require high accuracy.

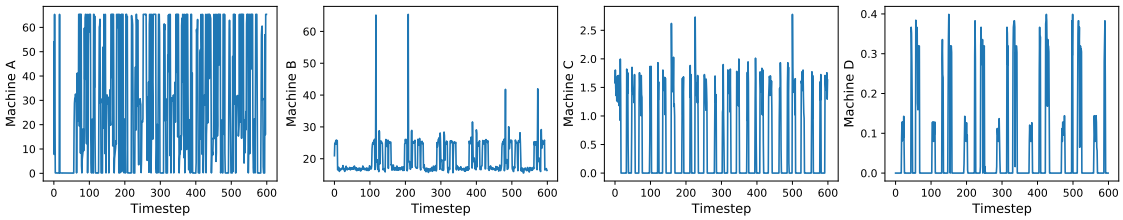


Figure 5.1: A energy consumption data in a production line

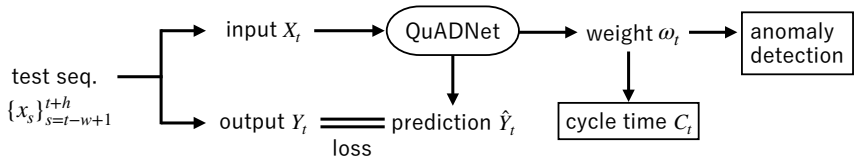


Figure 5.2: Quad system

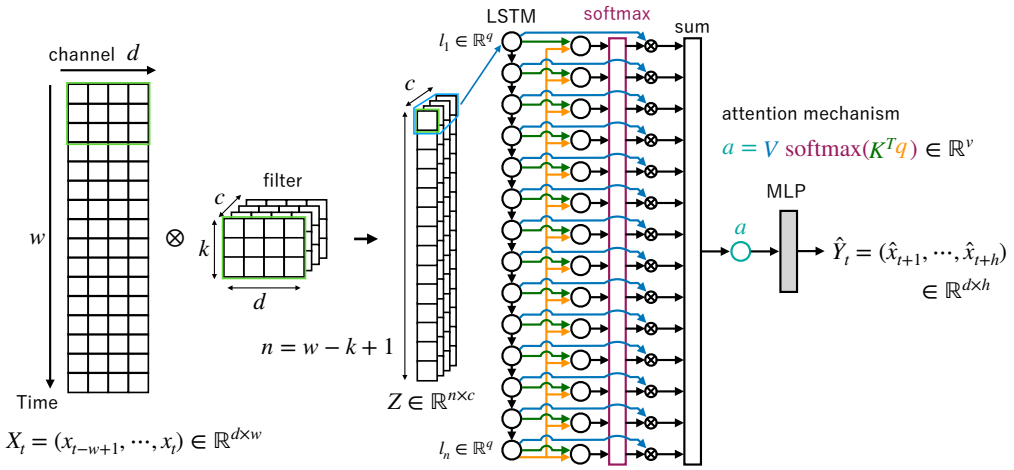


Figure 5.3: QuADNet

The second approach is a prediction-based method that enables immediate response using time-series prediction methods to evaluate anomaly levels by prediction errors. Typical prediction-based methods include studies utilizing the ARIMA model [232, 100, 248] and LSTM [240], which discriminate anomalies by the prediction confidence interval. These methods are not able to estimate individual cycles, nor are they able to detect anomalies using quasi-periodic structures.

In this study, we propose the QuAD (Quasi-periodic Anomaly Detection) system as a technology that can simultaneously detect anomalies and estimate cycle times in real-time. The QuAD system consists of the prediction network QuADNet (Network for Quasi-periodic Anomaly Detection) and components for anomaly detection and cycle time estimation (Fig. 5.2). QuADNet is a network that extracts time-local features using a CNN (convolutional neural network) and then performs multi-horizon forecasting using an attention mechanism. The QuAD system can perform real-time estimation of individual cycles using the weight vectors that appear in the attention mechanism and real-time anomaly detection using the prediction errors and weight vectors.

This chapter is structured as follows. The next section proposes a QuAD system for anomaly detection and individual cycle estimation. In section 5.3, we discuss the results of applying the proposed method and the comparison method to simulated data that mimic smart meter data from a factory. Finally, we conclude this chapter in section 5.4.

5.2 QuAD System

This section proposes a QuAD system for anomaly detection and estimation of cycle times. We first introduce the QuADNet, the core of the proposed system, and then describe the estimation method of cycle times and the anomaly detection method.

5.2.1 QuADNet

QuADNet is a deep learning model for detecting quasi-periodic anomalies and estimating cycle times. The network comprises a convolutional neural network (CNN) for extracting local features and an attention mechanism for distinguishing important features (Fig. 5.3).

The attention mechanism is represented by

$$\boldsymbol{a} = V\boldsymbol{\omega} = V\mathrm{softmax}(K^T\boldsymbol{q}), \tag{5.1}$$

where $\boldsymbol{a} \in \mathbb{R}^v$, $\boldsymbol{q} \in \mathbb{R}^q$, $K = (\boldsymbol{k}_1 \ \cdots \ \boldsymbol{k}_n) \in \mathbb{R}^{q\times n}$, $V = (\boldsymbol{v}_1 \ \cdots \ \boldsymbol{v}_n) \in \mathbb{R}^{v\times n}$, and $\boldsymbol{\omega} \in \mathbb{R}^n$ represent the output vector, a query vector, a key matrix, and a value matrix, and the weight vector, respectively. Because the output vector is represented by the weighted sum of the value vectors, the mechanism extracts useful information by the similarity between the query vector and the key vectors. The mechanism is widely used in natural language processing [208], and computer vision [119].

We assume d-dimensional multivariate time-series data $\{\boldsymbol{x}_t\}_{t=1}^T$, where T means final time index. The network is trained by h-horizon prediction error

$$\mathcal{L}(\mathcal{B}) = \frac{1}{|\mathcal{B}|}\sum_{t\in\mathcal{B}} \mathrm{Error}(\mathrm{QuADNet}(X_t), Y_t), \tag{5.2}$$

where $X_t = (\boldsymbol{x}_{t-w+1} \ \cdots \ \boldsymbol{x}_t) \in \mathbb{R}^{d\times w}$, $Y_t = (\boldsymbol{x}_{t+1} \ \cdots \ \boldsymbol{x}_{t+h}) \in \mathbb{R}^{d\times h}$, and $\mathcal{B}$ represent an input matrix, the output matrix, and a batch set, respectively. Our experiments use the L_1 error for outliers' robustness, such as spiking activity. The network first applies 1-dimensional CNN with kernel size k and number of channels c and obtains the local features $L = (\boldsymbol{l}_1, \cdots, \boldsymbol{l}_n)^T \in \mathbb{R}^{n\times q}$ by the LSTM with hidden dims q. The network then obtains the vector $\boldsymbol{a}$ via the attention mechanism with the key matrix $K = L^T$, the value matrix $V = L^T$, and the query vector $\boldsymbol{q} = \boldsymbol{l}_n$. Finally, we obtain the final output vector $\hat{Y}_t$ by applying multi-layer perceptron (MLP) to $\boldsymbol{a}$.

5.2.2 Individual Period Estimation

The proposed system estimates a cycle time at time t by

$$c_t = (w-k+1) - \underset{1\le s\le w-k+1-s_c}{\arg\max} \omega_s, \tag{5.3}$$

where s_c represents truncating interval. The weight vector $\boldsymbol{\omega}_t$ corresponds to normalized similarities between the current and past local features. The time point with the greatest weight, excluding the most recent s_c time points, is the time point that contributed the most to the forecast. The point in time is the periodic point one cycle earlier, and the interval length can be regarded as the cycle time.

The robust estimate is computed by

$$C_t = \mathrm{mode}(\{c_t + j\}_{s=t-b_p, j=-n_e}^{t+b_p, n_e}), \tag{5.4}$$

where b_p means a base period estimated by $b_p = \mathrm{mode}(\{c_t\}_{t=w}^{T-h})$ and n_e is a neighborhood number to provide robustness against quasi-periodic shaking. When a lot's change appears as a change in the basic cycle, it is possible to detect the lot change by the degree of deviation of the individual cycles from the basic cycle and relearn the network.

5.2.3 Anomaly Detection

The proposed system determines abnormality by the degree of deviation from the distribution of weight vectors estimated from normal data. Since the weight vectors have a sum of 1, the Dirichlet distribution

$$f(\boldsymbol{\omega};\boldsymbol{\alpha}) = \frac{\Gamma\left(\sum_{s=1}^{w-k+1}\alpha_s\right)}{\prod_{s=1}^{w-k+1}\Gamma(\alpha_s)} \prod_{s=1}^{w-k+1} \omega_s^{\alpha_s-1}, \tag{5.5}$$

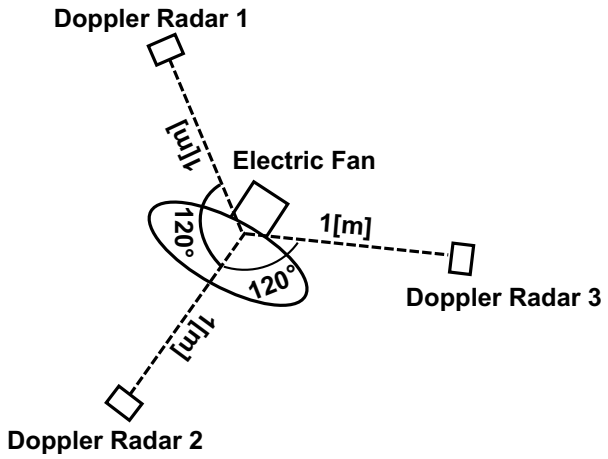


Figure 5.4: Experimental setup

where $\Gamma(\cdot)$ represent the Gamma function. The parameter $\boldsymbol{\alpha}$ is estimated by the maximum likelihood method from the set of weight vectors $\{\boldsymbol{\omega}_t\}$ obtained during training.

Since anomaly intervals are more important than anomaly points, the anomalous section is identified through post-processing based on the following five steps.

1. Moving average of the log-likelihood with interval b_p
2. Determine anomalies based on a threshold value
3. Closing the anomaly intervals with γ_1
4. Opening the anomaly intervals with γ_2
5. Closing the anomaly intervals with $\gamma_3 > \max\{\gamma_1, \gamma_2\}$

Step 1 corresponds to smoothing and is performed to correct deviations in likelihood within one cycle. Steps 3 and 5 are performed to combine anomaly intervals when the intervals are short. Step 4 is performed to remove short anomaly intervals. In our experiments, we have applied the same post-processing to other methods. The base period b_p is estimated for the proposed method, and a known value is used for the existing methods.

5.3 Experiments

Although obtaining energy consumption data from factories is possible, we cannot release them to the public for confidentiality reasons. The data is characterized by anomalies due to deviations from quasi-periodicity and is long-periodic. Due to the long periodicity, short-time Fourier transforms have low-frequency resolution and cannot capture anomalies. Since such data do not exist as a benchmark dataset for time series anomaly detection, we created an experimental dataset using a fan.

5.3.1 Fan Data

To verify the effectiveness of the proposed method, we experimented with the detection of anomalies in periodic motion using a microwave Doppler radar [359]. A transmitted radio signal v_s may be expressed by the following equation:

$$v_s(t) = A_s \cos \phi_s, \ \phi_s = 2\pi f t + o_s, \tag{5.6}$$

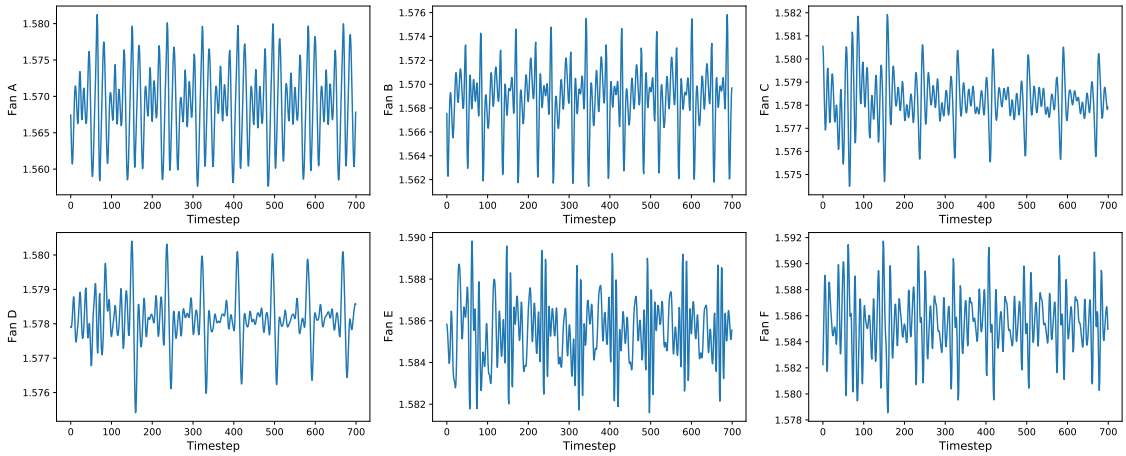


Figure 5.5: Generated fan data

Algorithm 1 Generating process of normal data

Require: original data $\{\boldsymbol{o}_t\}_{t=0}^{T-1}$, representative period of original data T_o, normal down-sampling range R_n

a current time $t_c = 0$, a residual time $t_r = 0$, a counter of new data $c = 0$, a counter of periodic point $c_p = 0$
while $t_c < T - 1$ **do**
  $i := \lfloor t_c \rfloor$, $d := t_c - i$
  $\boldsymbol{x}_c = (1 - d)\boldsymbol{o}_i + d\boldsymbol{o}_{i+1}$
  a next movement $m \sim U(R_n)$, where $U(R)$ represent uniform distribution of range R
  $t_c := t_c + m$, $t_r := t_r + m$
  if $t_r > T_o$ **then**
    $t_r := t_r - T_o$, $p_{c_p} := c$, $c_p := c_p + 1$
  end if
  $c := c + 1$
end while

Ensure: new data $X = \{\boldsymbol{x}_{t-1}\}_{t=1}^{c}$, a sequence of periodic points $\{p_{t-1}\}_{t=1}^{c_p}$

where A_s, f, and o_s respectively indicate its amplitude, frequency, and initial phase. The transmitted wave is reflected by a subject and returns as a received signal v_r after a delay of time T_r, as expressed by the following equation:

$$v_r(t) = A_r \cos \phi_r, \tag{5.7}$$

$$\phi_r = 2\pi f(t - T_r) + o_s = 2\pi f \left(t - \frac{2}{c} L(t) \right) + o_s, \tag{5.8}$$

where A_r and $L(t)$ are the amplitude of the received signal and the distance between the sensor and the subject at time t, respectively. The following signals are obtained by applying inphase/quadrature (I/Q) detection to the received signal:

$$I(t) = \frac{A_s A_r}{2} \left\{ \cos(\phi_r - \phi_s) + \cos(\phi_r + \phi_s) \right\}, \tag{5.9}$$

$$Q(t) = \frac{A_r A_s}{2} \left\{ \sin(\phi_r - \phi_s) + \sin(\phi_r + \phi_s) \right\}. \tag{5.10}$$

The direction and speed of the subject's movement can be obtained by analyzing the I/Q signal. In the experiment, we performed measurements of the periodic motion of an oscillating fan in the environment

Algorithm 2 Generating process of abnormal data

Require: original data $\{\boldsymbol{o}_t\}_{t=0}^{T-1}$, representative period of original data T_o, normal / abnormal downsampling range R_n / R_a, number of continuous normal / abnormal interval n_n / n_a, switching kernel $P = \begin{pmatrix} p_{nn} & p_{na} \\ p_{an} & p_{aa} \end{pmatrix}$

a current time $t_c = 0$, a residual time $t_r = 0$, a counter of new data $c = 0$, a counter of continuous state $c_c = 0$, a counter of periodic point $c_p = 0$, a current state $s = n$
while $t_c < T - 1$ **do**
  $i := \lfloor t_c \rfloor$, $d := t_c - i$
  $\boldsymbol{x}_c = (1 - d)\boldsymbol{o}_i + d\boldsymbol{o}_{i+1}$
  a next movement $m \sim U(R_s)$, where $U(R)$ represent uniform distribution of range R
  $t_c := t_c + m$, $t_r := t_r + m$
  if $t_r > T_o$ **then**
    if $c_c < n_s$ **then**
      $c_c := c_c + 1$
    else
      $u \sim U[0, 1]$
      if $u > p_{ss}$ **then**
        $c_c := 0$, $s := n$ $(s = a)$; a $(s = n)$
      end if
    end if
    $t_r := t_r - T_o$, $p_{c_p} := c$, $c_p := c_p + 1$
  end if
  $c := c + 1$
end while

Ensure: new data $X = \{\boldsymbol{x}_{t-1}\}_{t=1}^{c}$, a sequence of periodic points $\{p_{t-1}\}_{t=1}^{c_p}$

shown in Fig. 5.4. Three microwave Doppler radars (InnoSenT GmbH, IPS-154) were placed around a fan to measure its oscillating motion. Each I/Q signal was measured at a sampling rate of 300 Hz using an analog-to-digital converter (ADC) (Contec Co. Ltd., AIO-160802AY-USB).

The observed signals $\{\mathbf{o}_t\}$ were downsampled by Algorithm 1 and 2, and we obtained “normal" data for training and validation and “abnormal" data for testing. We set the parameters $T_o = 4301$, $R_n = [49.5, 50.5]$, $R_a = [48, 49.5]$, $n_n = 5$, $n_a = 3$, $p_{na} = 0.1$, $p_{an} = 0.6$, $p_{nn} = 1 - p_{na}$, $p_{aa} = 1 - p_{an}$. The representative cycle time of the normal data is around 85 to 86, and the abnormal data is around 86 to 88. The obtained normal data is shown in Fig. 5.5.

5.3.2 Conditions and Results

We trained QuADNet with Adam optimizer [173] with a learning rate of 0.001. Hyperparameters are tuned using the tree-structured Parzen estimator [31] in Optuna [5]. Tune hyparparameters are window size $w = 150$, horizon $h = 6$, kernel size $k = 10$, channel size $q = v = 100$. The parameter of Dirichlet distribution is estimated by the fixed iteration method and Newton method [251]. We set the truncating interval $s_c = 12$, the neighborhood number $n_e = 1$, the first closing quantity $\gamma_1 = 1$, the first opening quantity $\gamma_2 = 20$, the second closing quantity $\gamma_3 = 50$. All of the network training takes less than 10 minutes and is online.

Our comparisons are SARIMA [334], kNN [33], and SVDD [350]. Since the latter two methods are supervised learning methods, we also compare the unsupervised approach by the Douglas-Peucker-based segmentation algorithm [80, 213] called DP-kNN and DP-SVDD.

The recall was calculated as the percentage of intervals in which the half-abnormal time point within each interval was judged to be an anomaly among the true consecutive anomaly intervals. Precision was calculated as the percentage of intervals with more than half of the time points within each interval included in the true anomaly intervals out of the predicted consecutive anomaly intervals. These calculations were repeated with

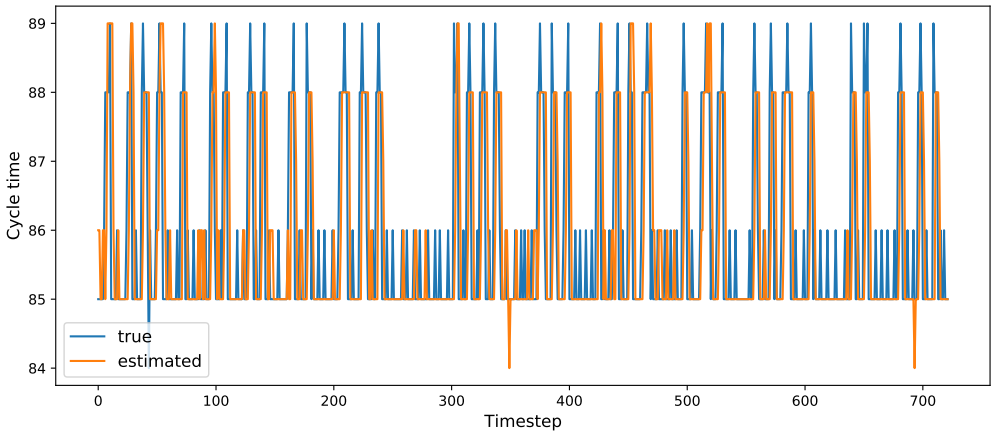


Figure 5.6: Estimated cycle times

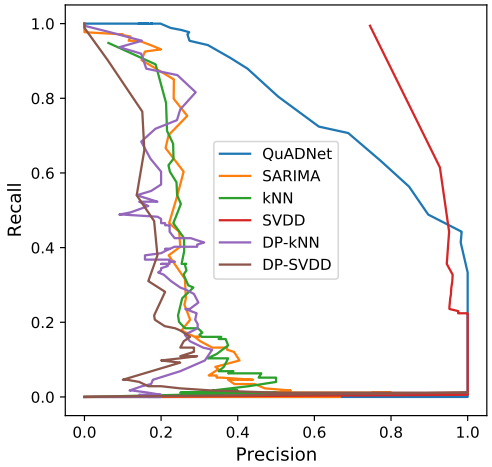


Figure 5.7: PR curve

different threshold values, and the resulting PR curves are shown in Fig. 5.7, and the AUPRC and maximum F_1 values are summarized in Table 5.1. Fig. 5.6 shows estimated cycle times taking the median within each true periodic interval.

5.3.3 Considerations

Table 5.1 shows that the proposed method outperforms other unsupervised learning methods in AUPRC and F_1. SVDD is the best when supervised learning is included, but in real-world problems, this is mainly the case when a teacher is unavailable or the high human and financial costs. Unsupervised DP-SVDD gives poor results, giving the proposed method an advantage.

The threshold determination is often based on the width of the recall that one wishes to increase. If the recall is requested to be approximately 1, QuADNet's precision is about 0.3. If the recall must be greater than 0.8, precision can also be greater than 0.4, reducing the human cost of checking for anomalies. A higher level than 0.95 is often required, and the proposed method needs further improvement.

Fig. 5.6 demonstrates that the proposed system can identify the correct individual cycle times with an error within 1. The proposed system can accurately and simultaneously detect anomalies and estimate the cycle time.

Table 5.1: F_1 score and AUPRC

	Method	F_1	AUPRC
Supervised	kNN	0.345	0.242
	SVDD	0.852	0.909
Unsupervised	DP-kNN	0.428	0.213
	DP-SVDD	0.262	0.161
	SARIMA	0.347	0.196
	QuADNet (ours)	**0.697**	**0.790**

5.4 Conclusion

This chapter proposes a system for detecting anomalies and estimating individual cycles in quasi-periodic time series data typified by energy consumption data from factory production lines. The proposed system consists of three parts: a prediction network called QuADNet, anomaly detection, and cycle time estimation. The most important part of the network is the attention mechanism, which performs anomaly detection and period estimation based on the weights calculated by the mechanism.

We generated normal and anomaly data by varying the down-sampling rate for the observed data obtained using a fan and a Doppler sensor. The proposed method outperforms the benchmark method in terms of AUPRC. The cycle time estimation by the proposed system was also highly accurate. The proposed system is also highly online, as the network training takes less than 10 minutes, and the evaluation computation takes only a few seconds. Thus, the proposed system is excellent for online anomaly detection and cycle time estimation for quasi-periodic data.

However, two issues remain with the proposed system. The first is to increase precision while keeping recall at 1. We plan to search for a more optimal structure concerning the combination of network components such as convolutional structure and attention mechanism. The second is introducing a mechanism to distinguish an anomaly from a system changeover, such as lot changes. If a signal for lot changes cannot be obtained, it is necessary to send continuously out-of-cycle information as a signal for lot changes and relearn the network.

6 Ensemble Kalman Variational Objective: A Variational Inference Framework for Sequential Variational Auto-Encoders

Time series model inference can be divided into modeling and optimization. Sequential VAEs have been studied as a modeling technique. As an optimization technique, methods combining variational inference (VI) and sequential Monte Carlo (SMC) have been proposed; however, they have two drawbacks: less particle diversity and biased gradient estimators. This chapter proposes *Ensemble Kalman Variational Objective* (EnKO), a VI framework with the ensemble Kalman filter, to infer latent time-series models [154]. Our proposed method efficiently learns the time-series models because of its particle diversity and unbiased gradient estimators. We demonstrate that our EnKO outperforms previous SMC-based VI methods in the predictive ability for several synthetic and real-world data sets.

6.1 Introduction

Inference of time-series models is essential for prediction, control, and reanalysis. For example, epidemiologists use models to predict the spread of infectious diseases, economists use models to make transactions, and meteorologists use models to reanalyze weather fields. There is no need for inference if the model is already established, but it is not, and data-driven model inference is necessary.

Model inference can be divided into two steps: modeling and optimization. The modeling phase constructs an appropriate model for representing the generating process of time series data. The optimization phase searches for the optimal model by maximizing the prediction accuracy of the parameters in the model constructed in the previous step.

Classically, ARIMA and state-space models [334] have been used for modeling; however, deep learning models have been researched [63, 94, 21, 112, 260]. We focus on a class of deep learning models called sequential VAEs (SVAEs), which extract meaningful latent variables from observed dimensions. SVAEs can be viewed as an extension of VAE [174] to sequential data, where the encoder $q_\varphi(z_t|x_{1:T})$, the decoder $p(x_t|z_t)$, and the latent transition process $p_\theta(z_t|z_{1:t-1})$ are learned to obtain the generative process

$$p_\theta(x_{1:T}, z_{1:T}) = p_\theta(z_1)\prod_{t=2}^{T} p_\theta(z_t|z_{1:t-1})\prod_{t=1}^{T} p_\theta(x_t|z_t), \tag{6.1}$$

where $\boldsymbol{z}_t$ and $\boldsymbol{x}_t$ represent the latent and the observed variables, respectively, at a given time t. Because of their high model representation capability, these techniques have been applied to a wide range of data, such as music data [238] and mouse brain voltage data [260].

VAE and SVAEs use variational inference (VI) [22, 162] to learn the parameters. VI maximizes the evidence lower bound, a lower bound of the log marginal likelihood $\log p(x_{1:T})$, instead of the intractable maximization of the log marginal likelihood. To achieve a tighter bound, objective functions are given by an ensemble of particles in IWAE [39, 77] and FIVO [238]. While IWAE gives the objective function by simple Monte Carlo integration, FIVO uses sequential Monte Carlo (SMC) to give a tighter bound than IWAE asymptotically.

However, FIVO has two drawbacks: biased gradient estimators and particle degeneracy. The resampling operations required by SMC are non-differentiable and result in a high variance of the gradient estimators. To reduce the variance, the gradient of the resampling operation can be omitted, or a continuously relaxed distribution [239] can be used, both of which lead to the biased gradient estimator. It is known that a biased gradient estimator leads to poor local optima, which destabilizes parameter learning. Particle degeneracy means only one or a few particles occupy a large weight and are copied at the resampling step of SMC, resulting in low particle diversity [82]. Particle diversity enhances the ability to represent the probability density function (PDF), and low diversity results in poor model inferences. Previous studies to obtain

This Chapter is based on "Ensemble Kalman variational objective: a variational inference framework for sequential variational auto-encoders" [154], by the same author, which appeared in the Proceedings of *Nonlinear Theory and Its Applications*,

Table 6.1: Objective functions and ensemble methods of ensemble learning frameworks

System	Method	Objective
IWAE [39]	naïve Monte Carlo	$\mathbb{E}\left[\log\left(\frac{1}{N}\sum_{i=1}^{N}\prod_{t=1}^{T}\frac{p_{\boldsymbol{\theta}}(x_t, z_t^{(i)}\|Z_{1:t-1})}{q_{\boldsymbol{\varphi}}(z_t^{(i)}\|X, Z_{1:t-1})}\right)\right]$
FIVO [238]	particle filter	$\mathbb{E}\left[\log\left(\frac{1}{N}\prod_{t=1}^{T}\sum_{i=1}^{N}\frac{p_{\boldsymbol{\theta}}(x_t, z_t^{(i)}\|Z_{1:t-1})}{q_{\boldsymbol{\varphi}}(z_t^{(i)}\|X, Z_{1:t-1})}\right)\right]$
EnKO (ours)	ensemble Kalman filter	$\mathbb{E}\left[\log\left(\frac{1}{N}\sum_{i=1}^{N}\prod_{t=1}^{T}\frac{p_{\boldsymbol{\theta}}(x_t, z_t^{(i)}\|Z_{1:t-1})}{q_{\boldsymbol{\varphi}}(z_t^{(i)}\|X, Z_{1:t-1})}\right)\right]$

particle diversity in SMC require many iterations or resampling operations [62, 363, 428]. The iterative methods spend time computing gradient propagation and learning parameters. The resampling operations lead to the biasedness of the gradient estimator same as FIVO. These problems are inherent to the SMC. In contrast, the ensemble Kalman filter (EnKF) [88, 89], a nonlinear Bayesian filtering method used mainly in geophysics [317, 91], does not suffer from these problems.

Thus, we propose a new optimization framework called *Ensemble Kalman Variational Objective* (EnKO), which uses the EnKF to learn the SVAEs and obtains rich PDF representations. The main idea of the proposed method is to take advantage of the characteristics of the EnKF.

In this chapter, our main contributions are as follows:

1. We first provide a new framework called EnKO, which uses the EnKF in the inference of latent states $\{z_t\}_{t=1}^{T}$. The proposed method has two main advantages over the FIVO framework: particle diversity and unbiased gradient estimators. The diversity provides rich information on the latent states to predict observations and quantify their uncertainty. The gradient estimators are unbiased because of no need to truncate any terms while the FIVO truncates the resampling term.

2. It is known that the EnKF often underestimates the sample covariance matrix of the latent states. We introduce a covariance inflation technique generally used in geophysics to alleviate this problem.

3. The computational complexity of EnKO is $O(d_x^2)$, which is less scalable than $O(d_x)$, such as FIVO, where d_x represents the observation dimensions. We incorporate auxiliary variables for high-dimensional data to reduce the computational complexity to $O(d_x)$.

4. EnKO has been applied to multiple synthetic and real-world data sets to achieve superior prediction accuracy and particle diversity. Furthermore, we report that the variance of the gradient estimator of EnKO is lower than that of FIVO and IWAE for toy examples in Appendix A.4.

This chapter is organized as follows. Section 6.2 describes ensemble learning frameworks for sequential VAEs. Section 6.3 introduces the proposed method, EnKO, and discusses its concrete algorithm and simple theoretical validity. Section 6.4 conducts experiments on several synthetic and real-world data sets and compares the results with existing methods. Section 6.5 discusses the experiments and the proposed method. Finally, Section 6.6 summarizes this chapter. The supplementary materials give detailed theoretical and experimental results. Our codes are available at our GitHub page `https://github.com/ZoneMS/EnKO`.

6.2 Ensemble Learning Frameworks

IWAE [39, 77] provides a tighter bound by an ensemble of particles:

$$\mathcal{L}_{\mathrm{IWAE}}^{N}(\boldsymbol{\theta}, \boldsymbol{\varphi}, X) := \mathbb{E}_{\prod_{i=1}^{N} q_{\boldsymbol{\varphi}}(Z^{(i)}|X)}\left[\log\left(\frac{1}{N}\sum_{i=1}^{N}\frac{p_{\boldsymbol{\theta}}(X, Z^{(i)})}{q_{\boldsymbol{\varphi}}(Z^{(i)}|X)}\right)\right], \tag{6.2}$$

where N denotes the number of particles, $Z^{(i)} = z_{1:T}^{(i)}$ denotes the i-th latent sequence, and $\mathcal{L}_{\mathrm{IWAE}}^{1} = \mathcal{L}_{\mathrm{ELBO}}$. When IWAE is applied to time series data, the divergence in the likelihood of particles increases as the time step advances. This means that lossy particles are left behind over time, and these particles reduce the learning efficiency.

FIVO [238] uses a particle filter to improve learning efficiency by allowing only particles with high likelihoods to survive. The ELBO of FIVO is formulated by

$$\mathcal{L}_{\mathrm{FIVO}}^{N}(\boldsymbol{\theta}, \boldsymbol{\varphi}, X) := \mathbb{E}_{Q_{\mathrm{FIVO}}(\boldsymbol{z}_{1:T}^{(1:N)}, \boldsymbol{x}_{1:T}^{(1:N)}|X)}[\log \hat{p}_N(\boldsymbol{x}_{1:T})], \tag{6.3}$$

$$\hat{p}_N(\boldsymbol{x}_{1:T}) = \prod_{t=1}^{T} \frac{1}{N} \sum_{i=1}^{N} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z}_t^{(i)}|\boldsymbol{z}_{1:t-1}^{(i)}) p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t^{(i)})}{q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t^{(i)}|\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:t-1}^{(i)})}, \tag{6.4}$$

$$Q_{\mathrm{FIVO}}(\boldsymbol{z}_{1:T}^{(1:N)}, \boldsymbol{x}_{1:T}^{(1:N)}|X) = \prod_{t=1}^{T} \prod_{i=1}^{N} q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t^{(i)}|\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:t-1}^{(i)}) p_{\boldsymbol{\theta}}(\boldsymbol{x}_t^{(i)}|\boldsymbol{z}_t^{(i)}), \tag{6.5}$$

where $p_{\boldsymbol{\theta}}(\boldsymbol{z}_1|\boldsymbol{z}_{1:0}) = p_{\boldsymbol{\theta}}(\boldsymbol{z}_1)$, $q_{\boldsymbol{\varphi}}(\boldsymbol{z}_1^{(i)}|\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:0}^{(i)}) = q_{\boldsymbol{\varphi}}(\boldsymbol{z}_1^{(i)}|\boldsymbol{x}_{1:T})$.

Objective functions and ensemble methods of these two learning frameworks and our proposed framework detailed following Section are compared in Table 6.1

6.3 Ensemble Kalman Variational Objective (EnKO)

Algorithm 3 Ensemble Kalman Variational Objectives

EnKO$(\boldsymbol{x}_{1:T}, p_{\boldsymbol{\theta}}, q_{\boldsymbol{\varphi}}, N)$:
for $t \in \{1, \cdots, T\}$ **do**
  for $i \in \{1, \cdots, N\}$ **do**
    if $t = 1$ **then**
      $\boldsymbol{z}_1^{(i)} \sim q_{\boldsymbol{\varphi}}(\boldsymbol{z}_1|\boldsymbol{x}_{1:T})$
    else
      $\boldsymbol{z}_t^{(i)} \sim q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:t-1}^{f,(i)})$
      $\boldsymbol{z}_{1:t}^{(i)} =$**CONCAT**$(\boldsymbol{z}_{1:t-1}^{(i)}, \boldsymbol{z}_t^{(i)})$
    end if
    $w_t^{(i)} = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{z}_t^{(i)}|\boldsymbol{z}_{1:t-1}^{(i)})}{q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t^{(i)}|\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:t-1}^{f,(i)})}$
  end for
  $\{\boldsymbol{z}_t^{f,(i)}\}_{i=1}^{N} =$**EnKF**$(\{\boldsymbol{z}_t^{(i)}\}_{i=1}^{N}, \boldsymbol{x}_t)$
end for
Ensure: $\hat{p}_N(\boldsymbol{x}_{1:T}) = \frac{1}{N}\sum_{i=1}^{N}\prod_{t=1}^{T} w_t^{(i)}$

EnKF$(\{\boldsymbol{z}_t^{(i)}\}_{i=1}^{N}, \boldsymbol{x}_t)$:
for $i \in \{1, \cdots, N\}$ **do**
  $\boldsymbol{x}_t^{(i)} \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t^{(i)})$
  $\boldsymbol{\mu}_t^{x,(i)} = \mathbb{E}[\boldsymbol{x}_t^{(i)}] = \boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{z}_t^{(i)})$
end for
$\bar{\boldsymbol{x}}_t = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_t^{(i)}$
$\bar{\boldsymbol{\mu}}_t^{x} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{\mu}_t^{x,(i)}$
$\bar{\boldsymbol{z}}_t = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{z}_t^{(i)}$
$\Sigma_t^{x} = \frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{x}_t^{(i)} - \bar{\boldsymbol{x}}_t)(\boldsymbol{x}_t^{(i)} - \bar{\boldsymbol{x}}_t)^T$
$\Sigma_t^{z\mu^x} = \frac{1}{N-1}\sum_{i=1}^{N}(\boldsymbol{z}_t^{(i)} - \bar{\boldsymbol{z}}_t)(\boldsymbol{\mu}_t^{x,(i)} - \bar{\boldsymbol{\mu}}_t^{x})^T$
$K_t = \Sigma_t^{z\mu^x}(\Sigma_t^{x})^{-1}$
for $i \in \{1, \cdots, N\}$ **do**
  $\boldsymbol{z}_t^{f,(i)} = \boldsymbol{z}_t^{(i)} + K_t(\boldsymbol{x}_t - \boldsymbol{x}_t^{(i)})$
end for
$\{\boldsymbol{z}_t^{c,(i)}\}_{i=1}^{N} =$**CI**$(\{\boldsymbol{z}_t^{(i)}\}_{i=1}^{N}, \{\boldsymbol{z}_t^{f,(i)}\}_{i=1}^{N})$
Ensure: $\{\boldsymbol{z}_t^{c,(i)}\}_{i=1}^{N}$

Ensemble Kalman Variational Objective (EnKO) is a framework for learning SVAEs using the EnKF in the inference phase. Thanks to the property of the EnKF in EnKO, the proposed method has three advantages over the SMC-based methods, as shown in Table 6.2:

1. EnKO provides high particle diversity because EnKF keeps particles have equal weights.
2. The gradient estimator of EnKO is a low variance because its operations are differentiable. This low variance is experimentally shown in Appendix A.4.
3. Unbiasedness of the gradient estimator is guaranteed using the full gradient estimators. A detail of the gradient estimator is shown in Appendix A.3.

Algorithm 4 Covariance Inflation

RTPP$(\{\boldsymbol{z}_t^{(i)}\}_{i=1}^N, \{\boldsymbol{z}_t^{f,(i)}\}_{i=1}^N; \alpha)$:
for $i \in \{1, \cdots, N\}$ **do**
  $\boldsymbol{z}_t^{c,(i)} = \alpha \boldsymbol{z}_t^{(i)} + (1-\alpha)\boldsymbol{z}_t^{f,(i)}$
end for
Ensure: $\{\boldsymbol{z}_t^{c,(i)}\}_{i=1}^N$

RTPS$(\{\boldsymbol{z}_t^{(i)}\}_{i=1}^N, \{\boldsymbol{z}_t^{f,(i)}\}_{i=1}^N)$:
$\boldsymbol{\sigma}_t = \frac{1}{N-1}(\boldsymbol{z}_t^{(i)} - \bar{\boldsymbol{z}}_t) \odot (\boldsymbol{z}_t^{(i)} - \bar{\boldsymbol{z}}_t)$
$\boldsymbol{\sigma}_t^f = \frac{1}{N-1}(\boldsymbol{z}_t^{f,(i)} - \bar{\boldsymbol{z}}_t^f) \odot (\boldsymbol{z}_t^{f,(i)} - \bar{\boldsymbol{z}}_t^f)$
for $i \in \{1, \cdots, N\}$ **do**
  $\boldsymbol{z}_t^{c,(i)} = (\alpha \boldsymbol{\sigma}_t + (1-\alpha)\boldsymbol{\sigma}_t^f) \oslash \boldsymbol{\sigma}_t^f \odot \boldsymbol{z}_t^{f,(i)}$
end for
Ensure: $\{\boldsymbol{z}_t^{c,(i)}\}_{i=1}^N$

Table 6.2: Comparative summary of competing ensemble systems

System	Unbiasedness		Diversity	Low variance	Robustness
	Gradient	Objective			
IWAE [39]	√	√	√	×	√
FIVO [238]	×	√	×	×	√
PSVO [262]	×	√	×	×	×
EnKO (ours)	√	×	√	√	√

The rest of this section is laid out as follows. We first describe a detailed algorithm of the proposed method in the following subsection. We then introduce our objective function and its property in subsection 6.3.2. Finally, subsection 6.3.3 is devoted to a technique to apply the proposed method to high-dimensional data.

6.3.1 Algorithm of the Proposed Method

Overall Algorithm We describe the overall algorithm of EnKO with Algorithm 3. First, the method infers the initial latent state $\boldsymbol{z}_1$ from the observations $X = \boldsymbol{x}_{1:T}$ by neural networks, e.g., $\boldsymbol{z}_1 = \text{biLSTM}_{\boldsymbol{\varphi}}(X)$ using bidirectional LSTM (biLSTM) in SRNN [94], $\boldsymbol{z}_1 = \text{MLP}_{\boldsymbol{\varphi}} \circ \text{biLSTM}_{\boldsymbol{\varphi}}(X)$ using biLSTM and dense NN in SVO [260] (*line 5* in Algorithm 3). At each time-step t, the latent state $\boldsymbol{z}_t^{(i)}$ is inferred from the observations X and the latent states until the previous time-step $\boldsymbol{z}_{1:t-1}^{(i)}$ (*line 7*). The importance weight $w_t^{(i)}$ is calculated by

$$w_t^{(i)} = \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{z}_t^{(i)} | \boldsymbol{z}_{1:t-1}^{(i)})}{q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t^{(i)} | \boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:t-1}^{(i)})} \tag{6.6}$$

$$= \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_t | \boldsymbol{z}_t^{(i)}) p_{\boldsymbol{\theta}}(\boldsymbol{z}_t^{(i)} | \boldsymbol{z}_{1:t-1}^{(i)})}{q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t^{(i)} | \boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:t-1}^{(i)})}, \tag{6.7}$$

for the objective function (*line 10*). The latent states $\{\boldsymbol{z}_t^{(i)}\}_{i=1}^N$ are corrected by the EnKF using their sample covariance and the observation at time t (*line 12*). Finally, the $\hat{p}_N(\boldsymbol{x}_{1:t})$ are computed (*output*).

Detailed Algorithm of the EnKF We describe the detailed algorithm of the EnKF in EnKO with Algorithm 3. At each time-step t and for each particle i, the mean estimate of the observation $\boldsymbol{\mu}_t^{x,(i)}$ and a sample $\boldsymbol{x}_t^{(i)}$ is generated from the emission network $p_{\boldsymbol{\theta}}(\boldsymbol{x}_t | \boldsymbol{z}_t^{(i)})$ (*lines 16-17*). For example, in the Gaussian output distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}_t | \boldsymbol{z}_t^{(i)}) = \mathcal{N}(\boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{z}_t^{(i)}), \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_t^{(i)}))$, the mean estimate $\boldsymbol{\mu}_t^{x,(i)} = \boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{z}_t^{(i)})$ is gained. In this example, $\boldsymbol{w}_t^{(i)} = \boldsymbol{x}_t^{(i)} - \boldsymbol{\mu}_t^{x,(i)}$ follows the zero-mean Gaussian distribution $\mathcal{N}(\boldsymbol{0}, \boldsymbol{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_t^{(i)}))$ and this random variable corresponds to the noise in the EnKF update formula. Covariance matrices Σ_t^x and $\begin{pmatrix} \Sigma_t^{z\mu^x} & \Sigma_t^{\mu^x} \end{pmatrix}^T$

correspond to $H_t\Sigma_t^{\tilde{z}}H_t^T + \Sigma_t^w$ and $\Sigma_t^{\tilde{z}}H_t^T$, respectively, at equation 4.17b in the augmented system, where $H_t = \begin{pmatrix} O_{d_z} & I_{d_x} \end{pmatrix}$ (*lines 22-23*). In the augmented system, because the first d_z coordinates of $\tilde{\boldsymbol{z}}_t$ constitute the only update target, the augmented gain $\tilde{K}_t \in \mathbb{R}^{(d_z+d_x)\times d_z}$ is only needed at the first d_z rows $K_t \in \mathbb{R}^{d_z \times d_x}$ (*line 24*). The latent state $\boldsymbol{z}_t^{(i)}$ is updated to $\boldsymbol{z}_t^{f,(i)}$ by the gain explaining observation $\boldsymbol{x}_t$ well (*line 26*). Finally, the updated latent states $\{\boldsymbol{z}_t^{f,(i)}\}_{i=1}^N$ are adjusted by covariance inflation methods such as RTPP and RTPS (*line 28*); these details are summarized in Algorithm 4.

6.3.2 Objective Function

The proposed method trains the EnKO to maximize the objective function

$$\mathcal{L}_{\mathrm{EnKO}}^N(\boldsymbol{\theta}, \boldsymbol{\varphi}, X) := \mathbb{E}_{Q_{\mathrm{EnKO}}(\boldsymbol{z}_{1:T}^{(1:N)}, \boldsymbol{x}_{1:T}^{(1:N)}|X)}[\log \hat{p}_N(\boldsymbol{x}_{1:T})], \tag{6.8}$$

$$\hat{p}_N(\boldsymbol{x}_{1:T}) = \frac{1}{N}\sum_{i=1}^{N}\prod_{t=1}^{T}\frac{p_{\boldsymbol{\theta}}(\boldsymbol{z}_t^{(i)}|\boldsymbol{z}_{1:t-1}^{(i)})p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t^{(i)})}{q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t^{(i)}|\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:t-1}^{f,(i)})}, \tag{6.9}$$

$$Q_{\mathrm{EnKO}}(\boldsymbol{z}_{1:T}^{(1:N)}, \boldsymbol{x}_{1:T}^{(1:N)}|X) = \prod_{t=1}^{T}\prod_{i=1}^{N} q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t^{(i)}|\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:t-1}^{f,(i)})p_{\boldsymbol{\theta}}(\boldsymbol{x}_t^{(i)}|\boldsymbol{z}_t^{(i)}), \tag{6.10}$$

where $p_{\boldsymbol{\theta}}(\boldsymbol{z}_1|\boldsymbol{z}_{1:0}) = p_{\boldsymbol{\theta}}(\boldsymbol{z}_1)$, $q_{\boldsymbol{\varphi}}(\boldsymbol{z}_1^{(i)}|\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:0}^{f,(i)}) = q_{\boldsymbol{\varphi}}(\boldsymbol{z}_1^{(i)}|\boldsymbol{x}_{1:T})$ and $\hat{p}_N(\boldsymbol{x}_{1:T})$ is the output of Algorithm 3. This formulation is the same as the objective of sequential IWAE [39] as shown in Equation 6.2 and slightly different from FIVO [238].

The following theorem and corollary are easily proved.

Definition 6.1 (linear Gaussian)**.** *An emission distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t)$ is linear Gaussian if the distribution is Gaussian and whose mean is linear transformation of $\boldsymbol{z}_t$, i.e.,*

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t) = \mathcal{N}(G_{\boldsymbol{\theta},t}\boldsymbol{z}_t, V_{\boldsymbol{\theta},t}^g), \tag{6.11}$$

where $G_{\boldsymbol{\theta},t} \in \mathbb{R}^{d_x \times d_z}$ and $V_{\boldsymbol{\theta},t}^g \in \mathbb{R}^{d_x \times d_x}$.

A variational distribution $q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:t-1}^f)$ is linear Gaussian if the distribution is Gaussian and whose mean is linear transformation of $\boldsymbol{z}_{t-1}^f$, i.e.,

$$q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:t-1}^f) = \mathcal{N}(Q_{\boldsymbol{\varphi},t}(\boldsymbol{x}_{1:T})\boldsymbol{z}_{t-1}^f, V_{\boldsymbol{\varphi},t}^q(\boldsymbol{x}_{1:T})), \tag{6.12}$$

where $Q_{\boldsymbol{\varphi},t}(\boldsymbol{x}_{1:T}) \in \mathbb{R}^{d_z \times d_z}$ and $V_{\boldsymbol{\varphi},t}^q(\boldsymbol{x}_{1:T}) \in \mathbb{R}^{d_z \times d_z}$.

Theorem 6.2. *The $\hat{p}_N(\boldsymbol{x}_{1:T})$ is an approximately unbiased estimator of the marginal likelihood $p(\boldsymbol{x}_{1:T})$. If the emission distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t)$ and the variational distribution $q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:t-1}^f)$ are linear Gaussian, the $\hat{p}_N(\boldsymbol{x}_{1:T})$ is an unbiased estimator of the marginal likelihood.*

Corollary 6.3. *The objective function $\mathcal{L}_{\mathrm{EnKO}}^N(\boldsymbol{\theta}, \boldsymbol{\varphi}, X)$ is an approximately lower bound of the log marginal likelihood $\log p(\boldsymbol{x}_{1:T})$. If the emission distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t)$ and the variational distribution $q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:t-1}^f)$ are linear Gaussian, the objective function is an unbiased estimator of the log marginal likelihood.*

There is one point to be noted here. Since the objective function is not an exact lower bound of the log marginal likelihood $\log p(\boldsymbol{x}_{1:T})$ but an approximate lower bound, it is not strictly an ELBO. However, we describe it as ELBO because it is an approximate lower bound of the log marginal likelihood and an exact lower bound under linear Gaussian constraints. This "approximation" means the filtering distribution $p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t})$ is replaced with the Gaussian distribution whose first and second moments are matched with $p(\boldsymbol{z}_t|\boldsymbol{x}_{1:t})$ at all time-steps. Our goal is to infer models efficiently, and we are not concerned with the tightness of the theoretical lower bound. In fact, in the following section, we show that the proposed method can experimentally infer more appropriate models than IWAE and FIVO.

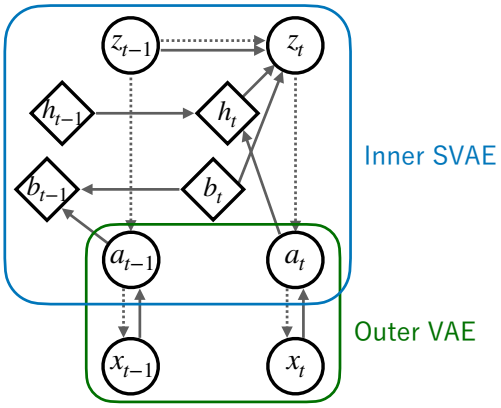


Figure 6.1: Hierarchical structure for application to high-dimensional data. We first embed the observed variables $\boldsymbol{x}_t$ to auxiliary variables $\boldsymbol{a}_t$, and then apply SVAE with EnKO to latent variables $\boldsymbol{z}_t$ and $\boldsymbol{a}_t$. In this figure, while SVAE is assumed to SVO, other SVAEs can be applied

6.3.3 High-dimensional Application of the Proposed Method

While the computational complexity of FIVO scales linearly with the observation dimension d_x, the computational complexity of EnKO scales quadratically with the dimension d_x, making it difficult to apply directly to high-dimensional data. To solve this problem, we use VAE to embed the observed variable $\boldsymbol{x}_t$ to an auxiliary variable $\boldsymbol{a}_t \in \mathbb{R}^{d_a}$, and then models SVAE for latent variables $\boldsymbol{z}_{1:t}$ and auxiliary variables $\boldsymbol{a}_{1:T}$ (Figure 6.1). In this structure, the computational complexity of EnKO scales quadratically with the dimension d_a and linearly with the observation dimension d_x, which significantly reduces the computational cost. In addition, because EnKO has particle diversity, the number of particles used in the calculation can be reduced compared to FIVO, resulting in lower computational costs than FIVO.

6.4 Experiments

We trained the SVO network [260] with EnKO, IWAE [39], and FIVO [238] for three synthetic data and one real-world benchmark data set. We used RTPP and RTPS as inflation methods in EnKO and set an inflation factor by grid search from $\{0.1, 0.2, 0.3\}$. We implemented our experiments in PyTorch, and neural network parameters were optimized using the Adam optimizer [173] with a learning rate of 0.001. The other hyper-parameters settings and the grid search details are described in Appendix A.5. All experiments were performed on three random seeds, and performance metrics were averaged over the seeds. The codes for the experiments are available at our GitHub page `https://github.com/ZoneMS/EnKO`.

6.4.1 FitzHugh-Nagumo Model

The FitzHugh-Nagumo model (FHN model) is a two-dimensional simplification of the Hodgkin-Huxley model, which models the spiking activity of neurons. The model is represented by

$$\dot{V} = V - \frac{V^3}{3} - W + I_{\text{ext}}, \tag{6.13}$$

$$\dot{W} = a(bV + d - cW), \tag{6.14}$$

where V and W represent the membrane potential and a recovery variable, respectively. In our experiments, we set $I_{\text{ext}} = 0$, $a = 0.7$, $b = 0.8$, $c = 0.08$, and $d = 0$. The initial states were uniformly sampled from $[-3, 3]^2$ to generate 400 samples using 200 for training, 40 for validation, and 160 for testing. A synthetic one-dimensional observation x_t was sampled from $\mathcal{N}(V_t, 0.1^2)$. We set the latent dimension to 2 and the number of particles to 16.

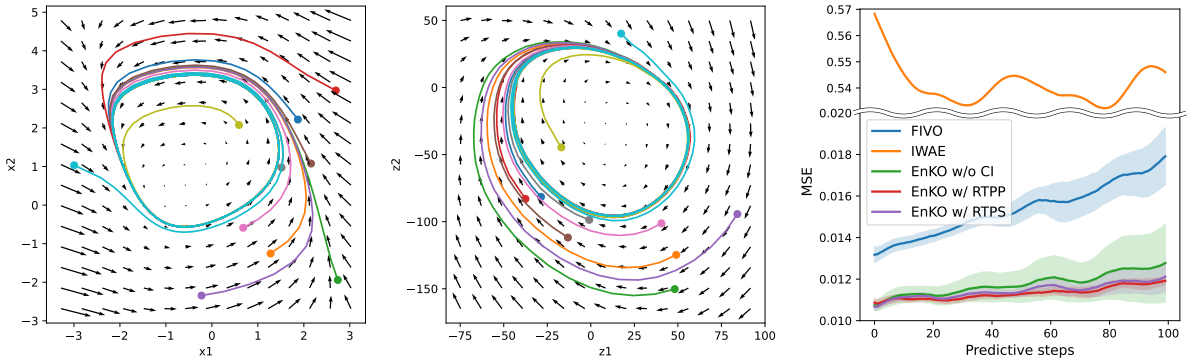


Figure 6.2: Inferred latent dynamics and prediction MSE for FitzHugh-Nagumo model. (left) True dynamics and trajectories on the first 10 test set. (center) Inferred latent dynamics and trajectories on the 10 test sets using EnKO to perform dimensionality expansion. (right) Prediction MSE for various ensemble frameworks, including FIVO (*blue*), IWAE (*orange*), and EnKO without (*green*) and with covariance inflation (RTPP: *red*, RTPS: *purple*). *Solid line* represents mean MSE over multiple seeds. *Semi-transparent filled area* represents the mean plus/minus the standard deviation of MSE over the seeds, exclusive IWAE due to the large standard deviation

The inferred latent dynamics and prediction MSE are shown in Figure 6.2. The left panel displays the original dynamics and trajectories on the first 10 test set. The center panel displays the inferred latent dynamics and trajectories on the 10 test sets using EnKO to expand dimensionality. The initial points located inside and outside the limit cycle and the order of the trajectories inside the original system are equivalent to the reconstructed system. This equivalence means that the reconstructed dynamics are topologically equivalent to the original dynamics. The right panel shows the prediction MSE comparison among ensemble frameworks. The EnKO outperforms the previous methods, especially for long-time prediction.

6.4.2 Lorenz Model

The Lorenz model is a system of three ordinary differential equations originally developed to simulate atmospheric convection. The model is described by

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \sigma(y - x), \tag{6.15}$$

$$\frac{\mathrm{d}y}{\mathrm{d}t} = x(\rho - z) - y, \tag{6.16}$$

$$\frac{\mathrm{d}z}{\mathrm{d}t} = xy - \beta z, \tag{6.17}$$

where x, y, and z are the variables of this system. In our experiments, we set $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$. These parameter settings are chaos parameters, leading to the Lorenz attractor. The initial states were randomly chosen from $[-10, 10]^3$ to generate 100 samples using 66 for training, 17 for validation, and 17 for testing. A synthetic observation $\boldsymbol{x}_t$ was generated from $N\left(\begin{pmatrix} x_t & y_t & z_t \end{pmatrix}^T, 0.1^2 \times I_3\right)$. We set the latent dimension to 3 and the number of particles to 16.

The inferred latent dynamics and prediction MSE are shown in Figure 6.3. The left panel displays the original trajectories on the first 10 test set. The center panel displays the inferred latent trajectories on the 10 test set using EnKO. The order of the trajectories inside the double-scroll attractor in the original system is equivalent to the reconstructed system. This equivalence means that the reconstructed system is topologically equivalent to the original system. The right panel shows the prediction MSE comparison among ensemble frameworks. The EnKO performs better prediction than the previous methods.

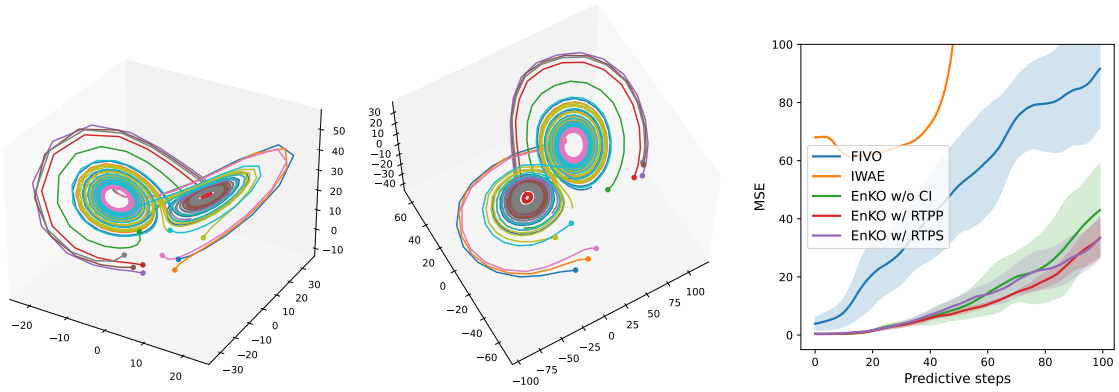


Figure 6.3: Inferred latent trajectories and prediction MSE for Lorenz model. (left) True trajectories on the first 10 test set. (center) Inferred latent trajectories on the 10 test set using EnKO. (right) Prediction MSE for various ensemble frameworks, including FIVO (*blue*), IWAE (*orange*), and EnKO without (*green*) and with covariance inflation (RTPP: *red*, RTPS: *purple*). *Solid line* represents mean MSE over multiple seeds. *Semi-transparent filled area* represents the mean plus/minus the standard deviation of MSE over the seeds, exclusive IWAE due to the large standard deviation

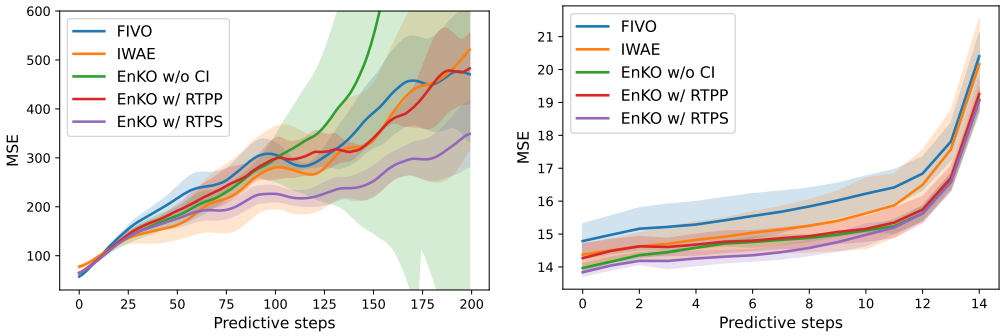


Figure 6.4: Prediction MSE for CMU walking data set (left) and rotating MNIST data set (right). *Solid line* represents mean MSE over multiple seeds. *Semi-transparent filled area* represents the mean plus/minus the standard deviation of MSE over the seeds

6.4.3 CMU Walking Data

We experiment on a data set extracted from the CMU motion capture library to demonstrate that the proposed method can capture latent structures from noisy real-world observations. We used the 23 walking sequences of subject 35 [102], which is partitioned into 16 for training, 3 for validation, and 4 for testing. We followed the preprocessing procedure of [372], after which we were left with 47-dimensional joint angle measurements and 3-dimensional global velocities. We set the latent dimension to 2 and the number of particles to 128.

The prediction MSE is shown in Figure 6.4 (left). While the error of EnKO without inflation methods diverges, EnKO with RTPS outperforms the other methods. This is probably because the EnKF tends to underestimate the state covariance when the ratio of the observation dimension to the number of particles is large. The predicted reconstructions from the proposed method are shown in Appendix A.6.

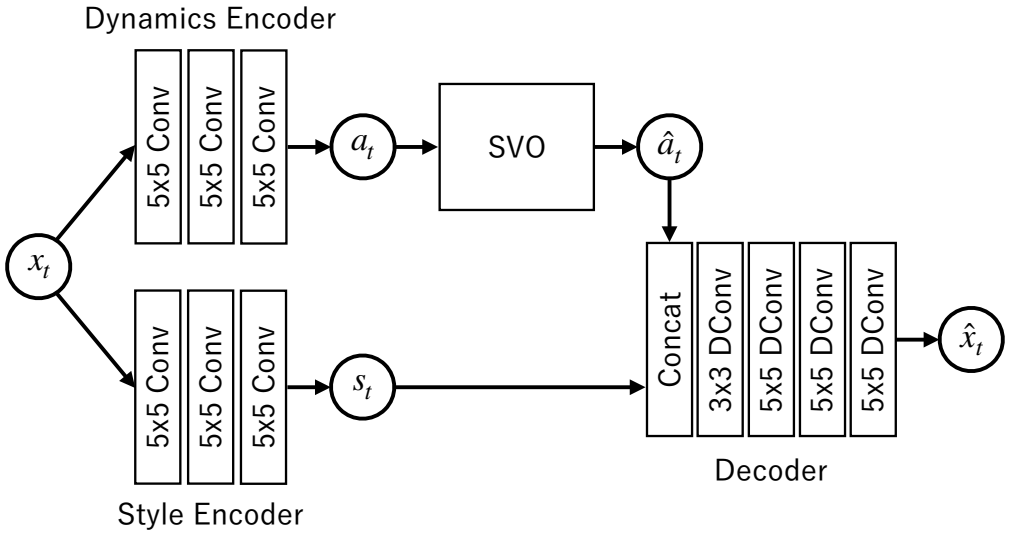


Figure 6.5: The network architecture of outer VAE for rotating MNIST data set. Conv and DConv stand for convolution layer and deconvolution layer, respectively. The encoder consists of a style convolution block and a dynamics convolution block. While the style convolution block extracts style (static) information, such as the shape and edge of the handwritten data, the dynamics convolution block extracts dynamics information, such as rotation dynamics

6.4.4 Rotating MNIST Dataset

We used a rotating MNIST data set consisting of rotated images of handwritten "3" digits [44] to demonstrate that the proposed method can extract latent dynamics from high-dimensional image sequence data. We used all of the rotation angles and divided the data set into 360 for training, 40 for validation, and 642 for testing. We used the hierarchical structure as presented in Figure 6.5 and set the auxiliary dimension to 8 (6 for styles and 2 for dynamics), the latent dimension to 2, and the number of particles to 32.

The prediction MSE is shown in Figure 6.4 (right). The EnKO method consistently provides lower errors than the FIVO and IWAE. Contrary to the walking data set, EnKO without inflation methods provides competitive errors with EnKO with the inflation methods. We introduced auxiliary variables, and the ratio of the substantial observation dimension to the number of particles is less than that of the walking data set. A small inflation factor is more suitable for this rotating handwritten data. The predicted images from the methods are shown in Appendix A.6.

6.4.5 Particle Diversity

We computed cosine similarity for each particle pairs to evaluate particle diversity. The cosine similarity is computed by

$$\mathrm{CS}(\boldsymbol{z}_t^{(i)}, \boldsymbol{z}_t^{(j)}) = \frac{(\boldsymbol{z}_t^{(i)} - \bar{\boldsymbol{z}}_t) \cdot (\boldsymbol{z}_t^{(j)} - \bar{\boldsymbol{z}}_t)}{\|\boldsymbol{z}_t^{(i)} - \bar{\boldsymbol{z}}_t\| \|\boldsymbol{z}_t^{(j)} - \bar{\boldsymbol{z}}_t\|}, \tag{6.18}$$

where $\boldsymbol{v} \cdot \boldsymbol{w}$ represents inner product of vectors $\boldsymbol{v}$ and $\boldsymbol{w}$. Because the cosine similarity is close to 1 when two particles are similar, the higher ratio of particle pairs which provide high cosine similarity is, the less particle diversity the model provides.

The ratio are summarized in Table 6.3. The network trained by IWAE has relatively less diversity for the Lorenz model. The Lorenz model provides chaotic dynamics, and IWAE could not represent the distribution of these dynamics. In contrast, the EnKO methods consistently provide a lower ratio of particle pairs of high cosine similarity than IWAE and FIVO.

Table 6.3: Ratio of particle pairs which provide higher cosine similarity

(a) Fitz-Hugh Nagumo model

System	≥ 0.5	≥ 0.7	≥ 0.9
IWAE	0.329	0.259	0.155
FIVO	0.343	0.276	0.171
EnKO w/o CI	0.327	0.256	0.151
EnKO w/ RTPP	**0.321**	**0.249**	**0.144**
EnKO w/ RTPS	*0.325*	*0.253*	*0.149*

(b) Lorenz model

System	≥ 0.5	≥ 0.7	≥ 0.9
IWAE	0.409	0.341	0.250
FIVO	0.286	0.198	0.0841
EnKO w/o CI	**0.249**	**0.158**	**0.0573**
EnKO w/ RTPP	**0.249**	**0.158**	*0.0574*
EnKO w/ RTPS	*0.251*	*0.160*	0.0581

(c) CMU walking data set

System	≥ 0.5	≥ 0.7	≥ 0.9
IWAE	0.132	0.0435	0.00330
FIVO	0.134	0.0454	0.00364
EnKO w/o CI	**0.124**	**0.0391**	**0.00283**
EnKO w/ RTPP	**0.124**	0.0393	*0.00284*
EnKO w/ RTPS	**0.124**	*0.0392*	**0.00283**

(d) Rotating MNIST data set

System	≥ 0.5	≥ 0.7	≥ 0.9
IWAE	0.336	0.261	0.153
FIVO	0.343	0.272	0.162
EnKO w/o CI	**0.327**	**0.250**	**0.144**
EnKO w/ RTPP	**0.327**	**0.250**	**0.144**
EnKO w/ RTPS	**0.327**	*0.251*	**0.144**

6.5 Discussion

The proposed method demonstrated excellent prediction accuracy in our experiments for three main reasons. The first is that EnKF is suitable for systems that can be approximated linearly in latent space in a time-local manner. The four data sets do not show rapid time-local changes; thus, the EnKF works well for systems that use local linear information. The chaotic nature of the Lorenz model makes long-term prediction difficult; however, the error derived from the linear approximation is small in the short term. On the other hand, downsampling the data every ten times points to a loss of local linearity, destabilizes the EnKF calculation, and reduces the proposed method's superiority. In the Walking dataset, if a sample suddenly starts dancing, the strong nonlinearity at the change point gives the FIVO an advantage.

Second, the phase space of the latent space is well covered by the training dataset, and the temporal evolution of the state is independent of the samples. The Fitz-Hugh Nagumo model exhibits periodic limit-cycle oscillations and does not contain samples that deviate significantly from the cycle. If the training data only includes data that winds from inside the cycle and the test data are given outside the cycle, it will predict incorrectly. The Lorenz model constructs the double-scroll attractor, and training, validation, and testing data were sampled uniformly. For these model systems, the predictions break down for samples obtained from models with different parameters. The Walking data shows behavior similar to periodic limit cycles in the latent space, although individual differences exist. Data with speedy running motion and crab walking may be incorrectly estimated. The Rotating MNIST data exhibit a circular motion in latent space, as the structure of the style and dynamics networks allows for acquiring a dynamics-related latent space. Predictions fail for data with different rotation speeds and opposite rotation data. If the style network is excluded from the embedding, the style evolves as latent states from the dynamics network, making learning difficult.

Third, both the network and the data satisfy the Markov property. Since SVO is used as SVAE, the transition structure of the latent variable is Markov. The Fitz-Hugh Nagumo data is Markov because we generated the data by the explicit Runge-Kutta method. The one-dimensional observation is non-Markov; however, the latent variable is Markov because the encoder constructs a latent space that captures time-delayed information. The Lorenz data is Markov same as the Fitz-Hugh Nagumo data. The data is sensitive at the intersection of the double scrolls. The transition model representing two-peak states is a more appropriate choice for this data. The Walking dataset is Markov because it contains only linear walking motions. The Rotating MNIST dataset is Markov because of the rotational motion at equal intervals. Modeling of the

SVAE needs to be reconsidered for systems with insufficient observation information and non-Markovian state space.

6.6 Conclusion

We have introduced *Ensemble Kalman Variational Objective* (EnKO) to improve time-series model inference by combining sequential VAEs. The proposed method uses the EnKF to infer latent variables and has three advantages over the previous SMC-based methods: particle diversity, the low variance of the gradient estimator, and the unbiasedness of the gradient estimators.

With these advantages, the proposed method outperforms the previous methods regarding the predictive ability for three synthetic and one real-world data set. The inferred 2-dimensional latent dynamics from 1-dimensional observations in the Fitz-Hugh Nagumo model are topologically equivalence to the original 2-dimensional dynamics. The inferred latent trajectories for synthetic Lorenz data could reconstruct the double-scroll attractor.

We provide three future directions for this work. The first direction is the application of the time-delay ODE systems [306, 193, 276, 234]. Our Fitz-Hugh Nagumo experiment demonstrates that the proposed method can reconstruct homomorphism space from one-dimensional observation. The reconstruction is related to time-delay coordinates [280, 348, 320] in the field of dynamics theory. The concept is especially useful for reconstructing the phase space of a dynamical system when we have access to only a single series of observations. The idea is to construct a multidimensional phase space by delaying the time series by a fixed time interval, often called the delay time τ, and stacking these delayed versions as coordinates. The time-delay coordinates can be represented as

$$z_t = \begin{pmatrix} x_t & x_{t+\tau} & \cdots & x_{t+(m-1)\tau} \end{pmatrix}, \tag{6.19}$$

where m is the embedding dimension. Under certain conditions, Takens' Embedding theorem [348] guarantees that the reconstructed phase space using time-delay coordinates is topologically equivalent to the original phase space of the dynamical system. Since this is closely related to our experimental result, we plan to apply the proposal to the time-delay system and compare the results.

The second direction is the applications inferring stochastic differential equations (SDEs). While this work only focuses on the time-series data formulated by the probabilistic time-series model, unequally spaced-time data and continuous-time observed data exist in practical applications. Since the proposed framework is not constrained to a discrete-time state transition, we plan to start with ODE inference such as NeuralODE [50] and ODE2VAE [406] and then extend to SDE.

The third direction is the hybrid algorithm of EnKO and FIVO. The other algorithm could solve the drawbacks of one algorithm. The hybrid of these two algorithms has the potential to solve both drawbacks. This future direction is clear because many hybrid sequential filtering methods of EnKF and PF have been proposed [97, 341].

7 Representation of Protein Dynamics Disentangled by Time-structure-based Prior

Representation learning (RL) is a universal technique for deriving low-dimensional disentangled representations from high-dimensional observations, aiding a multitude of downstream tasks. RL has been extensively applied to various data types, including images and natural language. Here, we analyze molecular dynamics (MD) simulation data of biomolecules in terms of RL. Currently, state-of-the-art RL techniques, mainly motivated by the variational principle, trying to capture slow motions in the representation (latent) space. Here, we propose two methods based on an alternative perspective on the *disentanglement* in the latent space [157]. By disentanglement, we here mean the separation of underlying factors in the simulation data, aiding in detecting physically important coordinates for conformational transitions. The proposed methods introduce a simple prior that imposes temporal constraints in the latent space, serving as a regularization term to facilitate capturing disentangled representations of dynamics. Comparison with other methods via the analysis of MD simulation trajectories for alanine dipeptide and chignolin validates that the proposed methods construct Markov state models (MSMs) whose implied time scales are comparable to state-of-the-art methods. Using a measure based on total variation, we quantitatively evaluated that the proposed methods successfully disentangle physically important coordinates, aiding the interpretation of folding/unfolding transitions of chignolin. Overall, our methods provide good representations of complex biomolecular dynamics for downstream tasks, allowing for better interpretations of conformational transitions.

7.1 Introduction

Molecular dynamics (MD) simulation is one of the most powerful approaches for investigating the conformational dynamics of biomolecules [379, 245]. MD simulations can be used to investigate various dynamic phenomena of biomolecules, such as allosteric transitions [258, 259, 182, 278, 328, 329], molecular docking [243, 98, 124, 153, 178], and structure formation [106, 286, 301, 252, 78]. Raw output data from MD simulations are trajectories and multivariate time-series data containing coordinates of atomic positions. The temporal information contained in trajectory data spans a broad range of time-scales, from atomic vibration in femtoseconds to protein folding in milliseconds. In trajectory data analysis, reducing the high-dimensional data to a small number of collective variables (CVs), which summarize such complicated dynamics, is often challenging. Capturing good CVs is important for various downstream tasks, including conformational clustering, free energy surface for analyzing the thermodynamic stability of states, Markov state modeling for analyzing kinetic behavior, and further simulations with enhanced sampling methods using the captured CVs.

For capturing CVs, dimensionality reduction techniques have been widely used in the studies of MD simulation analysis. Principal component analysis (PCA) captures the CVs with the largest variances by orthogonal transformations of the original coordinates. Relaxation mode analysis (RMA),[347, 255] time-structure-based independent component analysis (tICA),[268, 284, 326, 247, 331] which are often applied for capturing CVs with the slowest relaxations by linear transformations. Advanced nonlinear reduction techniques include isomap[72] and diffusion map[336, 171]. For details, the readers are referred to a recent excellent review on unsupervised learning of MD data by Glielmo et al.[108]

Generally, since the success of such downstream tasks is highly dependent on the choice of CVs, capturing "good" CVs is an important subject for analyzing trajectories. In terms of the field of machine learning, unsupervised learning techniques for obtaining good CVs or features with an emphasis on the performance of various downstream tasks have been studied in the context of representation learning (RL). [27, 28, 335, 49, 377, 351] So far, RL has been developed and successfully applied to various types of data sets besides MD data. For image data, RL extracts informative representations for image classification [373, 376], anomaly detection [181, 45], and object detection [393, 344]. For language data, RL provides low-dimensional embedding for

This Chapter is reprinted (adapted) with permission from *J. Chem. Theory Comput.* 2024, 20, 1, 436—450. `https://doi.org/10.1021/acs.jctc.3c01025`.

sentiment analysis [396, 32], named entity recognition [7, 201], and language translation [279].

The key point of RL is to learn features that *disentangle* many underlying explanatory factors (essential for downstream tasks, such as classification or prediction) hidden in the observed data as possible, discarding as little information about the data.[27, 28] RL to obtain these features include basic dimensionality reduction methods such as PCA and dimensionality expansion methods such as sparse auto-encoder (SAE) [195]. In terms of this view, as shown by Schwantes et al. [327, 121, 342, 246], tICA can be regarded as an optimal embedding of temporal variations in trajectory data by linear transformations. Recently, motivated by recent developments in deep learning technologies, several methods based on neural networks have been developed to disentangle the underlying temporal behavior hidden in trajectory data [381, 131, 343, 17, 76, 242, 53, 52, 364, 378]. An auto-encoder (AE) is a general nonlinear dimensionality reduction method based on neural networks that learn the process of reconstructing the original observed variables after encoding them into low-dimensional latent variables [133].

Various extensions of AE for MD trajectory data have been proposed based on AE. Time-lagged AE (TAE), an extension of AE to the temporal domain, was developed to capture slow CVs by applying time-lagged reconstruction as the loss function in the AE framework [381]. A variational AE (VAE) suppresses overfitting, often occurring in AE, by adding Gaussian noise to the latent variables [174]. Time-lagged VAE (TVAE) learns encoding and time-lagged reconstruction process using VAE [136]. Variational dynamics encoder (VDE) obtains slower CVs by penalizing TVAE's loss function with a negative sample autocorrelation coefficient in the latent space [131]. Gaussian-mixture VAE (GMVAE) simultaneously performs dimensionality reduction, clustering them into macrostates with a Gaussian mixture distribution [364]. Variational approach for Markov processes nets (VAMPnets) and reversible deep MSM (revDMSM) learn macrostates based on a variational approach [242, 241]. State-free reversible VAMPnet (SRV) orthogonalizes the embedding maps with linear VAC[53].

More recently, physically motivated methods have been developed to learn interpretable representations for complex dynamics. For example, Yang et al. made a notable contribution by embedding time series in a latent space coherent with physical simulators, an approach particularly effective in deterministic simulations.[403] Complementing this, Khan and Storkey focused on representation learning that approximately conserves the original Hamiltonian in latent space, an important property in differentiating between static and dynamic representations.[168] From an information-theoretic perspective, Wang and Tiwary innovated by optimizing mutual information in different aspects of state prediction, offering a unique approach to understanding molecular dynamics.[370] Wang et al. explored the integration of physically constrained priors into representation learning, paving the way for more meaningful interpretations of latent dynamics.[371] Wu and Noé employed normalizing flows in latent space embeddings, allowing for a refined separation of reaction coordinates and noise, thereby contributing to a more nuanced approach to the dimensional reduction of molecular kinetics.[388] Together, these studies form a comprehensive framework, informing and inspiring our current research direction toward developing more interpretable and physically consistent models in protein dynamics simulation.

Despite the development of these methods, disentanglement in learning representations of dynamics has not yet been fully explored.

By disentanglement, we here mean the separation of underlying factors in the simulation data, aiding in detecting physically important coordinates for conformational transitions.

Since disentanglement is a crucial property for the success of downstream tasks, methods that can learn disentangled representations would be important for subsequent analyses. In order to explore this point, this study proposes two RL methods, time-structure-based variational autoencoder (tsVAE) and time-structure-based time-lagged variational autoencoder (tsTVAE). Both methods leverage a unique prior to disentangle complex biomolecular dynamics in the latent space. This prior imposes temporal constraints on the transitional motions in the latent space, expected to disentangle transitional motions on an event-by-event basis, not relying on averaged quantities like autocorrelations, which is a typical approach in other existing methods. Consequently, the proposed methods can robustly learn disentangled representations from relatively short MD trajectory data, even when conformational transitions occur infrequently. Through the comparison with other existing methods on the MD simulation data of alanine-dipeptide and chignolin, we show that the proposed methods can extract disentangled CVs that capture conformational transitions well, and the

properties of constructed Markov state models further validate the learned representations (CVs).

This chapter is organized as follows. Section 7.2 describes existing methods, then explains the details of the proposed methods (tsVAE, tsTVAE). Section 7.3 describes the protocols of MD simulations and the properties of MSMs used to assess the qualities of RL methods. Section 7.4 shows the results obtained by applying the existing and proposed methods to MD simulation data for alanine-dipeptide molecule and chignolin folding dynamics. Section 7.5 discusses the limitations of the methods and future directions.

7.2 Theory

7.2.1 Notation

Let $X = \{\boldsymbol{x}_t\}_{t=1}^T$ be the d_x-dimensional T-length observed sequence and $Z = \{\boldsymbol{z}_t\}_{t=1}^T$ be the d_z-dimensional T-length latent sequence. In our neural network model analysis, X is the feature-selected data from the series data obtained by MD simulation. For a vector sequence $V = \{\boldsymbol{v}_t\}_{t=1}^T$, $v_{t,i}$ represents the i-th element of the vector $\boldsymbol{v}_t$. For a vector $\boldsymbol{v} \in \mathbb{R}^d$ and a matrix $M \in \mathbb{R}^{n\times m}$, $\boldsymbol{v}^T \in \mathbb{R}^{1\times d}$ and $M^T \in \mathbb{R}^{m\times n}$ represents the transpose of the vector and the matrix, respectively. For a vector $\boldsymbol{v} = (v_1 \cdots v_d)^T \in \mathbb{R}^d$ and a natural number $n \in \mathbb{N}$, $\boldsymbol{v}^{\circ n} = (v_1^n \cdots v_d^n)^T$ represents the element-wise power of n. For vectors $\boldsymbol{v} = (v_1 \cdots v_d)^T$ and $\boldsymbol{u} = (u_1 \cdots u_d)^T \in \mathbb{R}^d$, $\boldsymbol{v} \odot \boldsymbol{u} = (v_1u_1 \cdots v_du_d)^T$ represents the element-wise product. For a vector $\boldsymbol{v} = (v_1 \cdots v_d)^T$, $\|\boldsymbol{v}\| = \sqrt{\sum_{i=1}^d v_i^2}$ and $\|\boldsymbol{v}\|_1 = \sum_{i=1}^d |v_i|$ represent the L2 norm and the L1 norm, respectively. For a vector $\boldsymbol{v} \in \mathbb{R}^d$, $\mathrm{diag}(\boldsymbol{v}) \in \mathbb{R}^{d\times d}$ represents the diagonalized matrix. For a finite set S, $|S|$ represents the number of elements of the set. For a random variable $X \in \mathcal{X}$, a function $f : \mathcal{X} \to \mathbb{R}$, and a distribution $p : \mathcal{X} \to [0, \infty)$, $\mathbb{E}_{p(X)}[f(X)] \in \mathbb{R}$ represents the expectation of $f(X)$ regarding to the distribution $p(X)$. For random variables $X \in \mathcal{X}$, $Y \in \mathcal{Y}$, a function $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, and a conditional distribution $p(X|Y)$, $\mathbb{E}_{p(X|Y)}[f(X, Y)] : \mathcal{X} \to \mathbb{R}$ represents the conditional expectation of $f(X, Y)$ regarding to the distribution $p(X|Y)$.

7.2.2 Auto-Encoder

Auto-encoder (AE) is a model for obtaining low-dimensional representation by reconstructing an observation from the reduced representation [312, 405] (Figure 7.1(a)). The model embeds an observation $\boldsymbol{x}$ to the latent variable $\boldsymbol{z} = E_{\boldsymbol{\varphi}}(\boldsymbol{x})$ by the encoder $E_{\boldsymbol{\varphi}} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_z}$ and reconstructs the observation $\hat{\boldsymbol{x}} = D_{\boldsymbol{\theta}}(\boldsymbol{z})$ by the decoder $D_{\boldsymbol{\theta}} : \mathbb{R}^{d_z} \to \mathbb{R}^{d_x}$. The encoder $E_{\boldsymbol{\varphi}}$ and the decoder $D_{\boldsymbol{\theta}}$ are neural networks with the model parameters $\boldsymbol{\varphi}$ and $\boldsymbol{\theta}$. The model parameters $\boldsymbol{\varphi}$ and $\boldsymbol{\theta}$ are learned by minimizing the reconstruction loss

$$l_r^{\mathrm{AE}} = \frac{1}{|\mathcal{B}|}\sum_{t\in\mathcal{B}} \|\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t\|^2 = \frac{1}{|\mathcal{B}|}\sum_{t\in\mathcal{B}} \|\boldsymbol{x}_t - D_{\boldsymbol{\theta}} \circ E_{\boldsymbol{\varphi}}(\boldsymbol{x}_t)\|^2, \tag{7.1}$$

where $\mathcal{B}$ is a minibatch whose elements are randomly sampled from $\{1, \cdots, T\}$. Note that the AE does not use temporal information in the observations; thus, it does not learn representations related to dynamic properties.

7.2.3 Variational Auto-Encoder

Variational auto-encoder (VAE) uses probabilistic models for AE's encoder and decoder (Figure 7.1(b)). The model designs a variational posterior distribution $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ and a generative distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ by neural networks [174]. The parameters $\boldsymbol{\varphi}$ and $\boldsymbol{\theta}$ are learned by variational inference [22, 162, 414], which is a method to maximize the evidence lower bound (ELBO) instead of intractable maximization of the log marginal likelihood. The ELBO is defined by

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}, \boldsymbol{x}) := \mathbb{E}_{q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})}{q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}\right] \leq \log p_{\boldsymbol{\theta}}(\boldsymbol{x}), \tag{7.2}$$

where $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})$ represents the joint generative distribution. The inequality in the equation (7.2) is easily proved by Jensen's inequality [160]. The bound is represented by

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}, \boldsymbol{x}) = \mathbb{E}_{q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})\right] - \mathrm{KL}\left(q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x}) \| p_{\boldsymbol{\theta}}(\boldsymbol{z})\right), \tag{7.3}$$

where $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ is a prior distribution, and $\mathrm{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x}) \| p_{\boldsymbol{\theta}}(\boldsymbol{z}))$ represents the Kullback-Leibler (KL) divergence between the posterior $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ and the prior $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ distributions [190, 189, 9]. Resembling the AE, the VAE does not learn the temporal information in the observations.

In a typical VAE, the prior $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ is chosen to be the standard isotropic Gaussian distribution $\mathcal{N}(\boldsymbol{z}_t; \mathbf{0}, I)$ where $\mathcal{N}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ represents the normal distribution with mean vector $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. This choice is computationally convenient (for calculating the KL divergence) and encourages disentangling the representation in the latent space because the isotropic distribution promotes latent variables becoming uncorrelated. At the same time, the prior helps to reduce overfitting to limited sets of training data because the KL term in Eq. 7.3 works as a regularization term to constrain the capacity of information contained in the latent space.[132]

7.2.4 Time-lagged Neural Network Models

The TAE extends the original AE by incorporating time-lagged information to learn the dynamic properties present in observations. The TAE [381] embeds an observation $\boldsymbol{x}_t$ to the latent variable $\boldsymbol{z}_t$ by the encoder network $E_{\boldsymbol{\varphi}} : \mathbb{R}^{d_x} \to \mathbb{R}^{d_z}$ and reconstructs the time-lagged observation $\hat{\boldsymbol{x}}_{t+\tau}$ by the decoder network $D_{\boldsymbol{\theta}} : \mathbb{R}^{d_z} \to \mathbb{R}^{d_x}$ (Figure 7.1(c)). The model parameters $\boldsymbol{\varphi}$ and $\boldsymbol{\theta}$ are learned by minimizing the reconstruction loss

$$l_r^{\mathrm{TAE}} = \frac{1}{|\mathcal{B}|} \sum_{t \in \mathcal{B}} \|\boldsymbol{x}_{t+\tau} - \hat{\boldsymbol{x}}_t\|^2 = \frac{1}{|\mathcal{B}|} \sum_{t \in \mathcal{B}} \|\boldsymbol{x}_{t+\tau} - D_{\boldsymbol{\theta}} \circ E_{\boldsymbol{\varphi}}(\boldsymbol{x}_t)\|^2, \tag{7.4}$$

where $\mathcal{B} \subset \{1, \cdots, T-\tau\}$ and τ are a minibatch and a model lagtime, respectively.

The TVAE [136] is an extension to TAE based on the architecture of the VAE. It introduces the probabilistic model to TAE and designs the variational posterior distribution, the generative distribution, and the prior distribution as follows (Figure 7.1(d)):

$$q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t|\boldsymbol{x}_t) = \mathcal{N}\left(\boldsymbol{z}_t; \boldsymbol{\mu}_{\boldsymbol{\varphi}}(\boldsymbol{x}_t), \mathrm{diag}(\boldsymbol{\sigma}_{\boldsymbol{\varphi}}(\boldsymbol{x}_t))\right), \tag{7.5}$$

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t+\tau}|\boldsymbol{z}_t) = \mathcal{N}\left(\boldsymbol{x}_{t+\tau}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}_t), I\right), \tag{7.6}$$

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_t) = \mathcal{N}(\boldsymbol{z}_t; \mathbf{0}, I). \tag{7.7}$$

The model parameters $\boldsymbol{\varphi}$ and $\boldsymbol{\theta}$ are learned by minimizing the following loss

$$l^{\mathrm{TVAE}} = l_r^{\mathrm{TVAE}} + \beta\, l_p^{\mathrm{TVAE}} \tag{7.8}$$

$$= -\frac{1}{|\mathcal{B}|} \sum_{t \in \mathcal{B}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t+\tau}|\boldsymbol{z}_t) + \beta\, \frac{1}{|\mathcal{B}|} \sum_{t \in \mathcal{B}} \mathrm{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t|\boldsymbol{x}_t) \| p_{\boldsymbol{\theta}}(\boldsymbol{z}_t)), \tag{7.9}$$

where l_r^{TVAE} and l_p^{TVAE} represent the reconstruction loss (negative log-likelihood) and the prior regularized loss, respectively.

In the limit of a single linear hidden layer of the TAE, the tICA solution can be obtained.[381] Also, the TVAE with regularized coefficient $\beta = 0$ reduces to the TAE with latent Gaussian perturbation. These suggest that the TAE and the TVAE are expected to be nonlinear extensions of tICA for learning slow CVs hidden in high-dimensional observation. However, as discussed by Chen et al. [52], just minimizing the reconstruction loss does not ensure that the method captures the slowest CVs in nonlinear ways. Chen et al. theoretically show that, in general, the TAE learns a mixture of slow and maximum variance CVs instead of purely slow CVs.

The VDE[131] is similar to the TVAE, but the problem of the TAE and TVAE is somewhat relaxed. Inspired by the variational approach to conformational dynamics,[272] the VDE introduced the sample autocorrelation loss l_a^{VDE} to the TVAE (Figure 7.1(e)). The VDE loss is defined by

$$l^{\mathrm{VDE}} = l_r^{\mathrm{VDE}} + \beta\, l_p^{\mathrm{VDE}} + \gamma\, l_a^{\mathrm{VDE}} \tag{7.10}$$

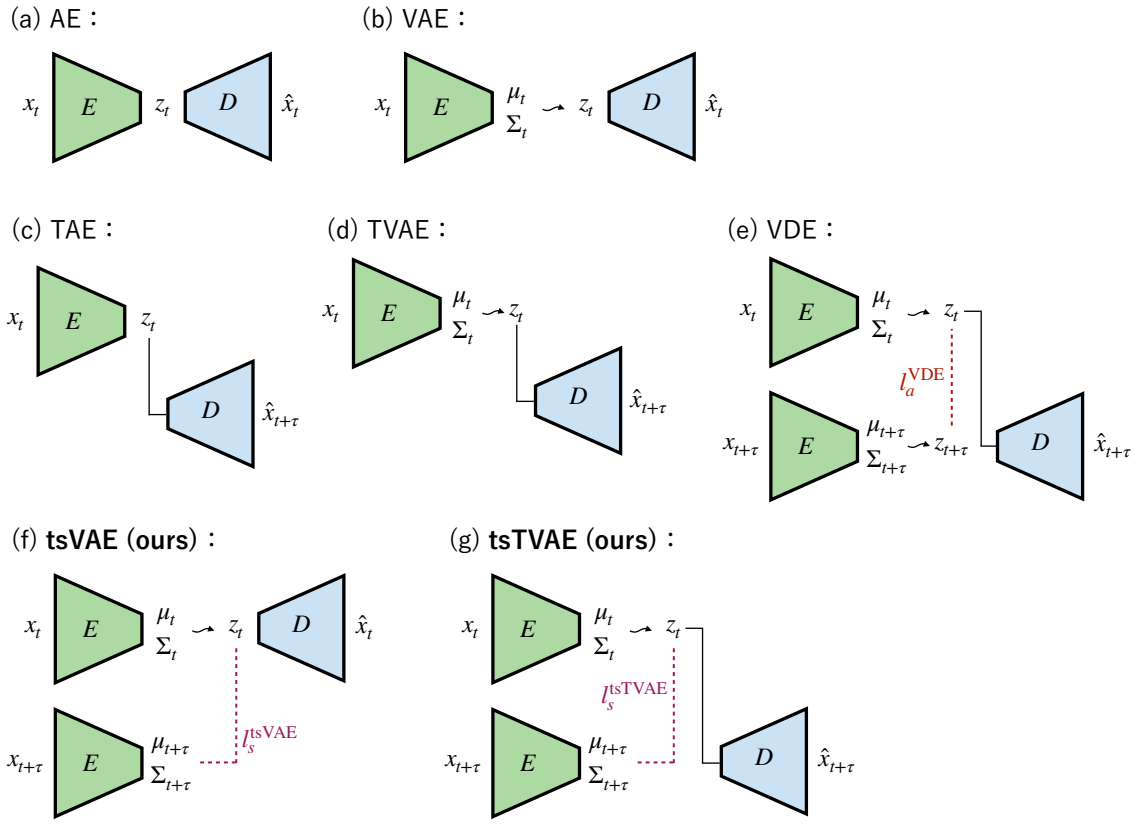


Figure 7.1: **Neural network model structures of AE, VAE, TAE, TVAE, VDE, tsVAE, and tsTVAE**. The green trapezoid E and the blue trapezoid D represent the encoder and the decoder, respectively. The red dashed line l_a^{VDE}, the purple dashed lines l_s^{tsVAE}, and l_s^{tsTVAE} represent the sample autocorrelation loss of VDE and the time-structure-based loss of tsVAE and tsTVAE, respectively. The black lines between a latent variable $\boldsymbol{z}_t$ and the decoder D represent that the $\boldsymbol{z}_t$ is an input of the decoder. The black wavy arrows represent sampling procedures from Gaussian distribution.

$$
\begin{aligned}
&= -\frac{1}{|\mathcal{B}|}\sum_{t\in\mathcal{B}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t+\tau}|\boldsymbol{z}_t) \\
&\quad + \beta\,\frac{1}{|\mathcal{B}|}\sum_{t\in\mathcal{B}} \mathrm{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}_{t+\tau}|\boldsymbol{x}_{t+\tau})\|p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t+\tau})) \\
&\quad - \gamma \sum_{d=1}^{d_z} \frac{\sum_{t\in\mathcal{B}}(z_{t,d}-\bar{z}_{t,d})(z_{t+\tau,d}-\bar{z}_{t+\tau,d})}{\sqrt{\left(\sum_{t\in\mathcal{B}}(z_{t,d}-\bar{z}_{t,d})^2\right)\left(\sum_{t\in\mathcal{B}}(z_{t+\tau,d}-\bar{z}_{t+\tau,d})^2\right)}}.
\end{aligned}
\tag{7.11}
$$

Here, $\bar{\boldsymbol{z}}_t$ represents the mean of $\{\boldsymbol{z}_t\}_{t\in\mathcal{B}}$, and γ is an autocorrelation penalty coefficient. The sample autocorrelation loss l_a^{VDE} calculates the sum of the negative sample autocorrelation in each dimension; the smaller loss means higher sample autocorrelation in the latent space. This loss enhances obtaining slower CVs because higher autocorrelation corresponds to a slower transition in the latent space. While this loss term $\gamma\, l_a^{\mathrm{VDE}}$ mainly depends on the coefficient γ and the minibatch size $|\mathcal{B}|$, the robustnesses are not fully discussed in the original paper [131].

7.2.5 Time-structure-based Neural Network Models

Here, we describe our proposed RL methods, tsVAE and tsTVAE, designed to obtain disentangled representations of protein dynamics (Figure 7.1(f),(g)). The idea behind tsVAE and tsTVAE is based on the VAE applying a prior $p_{\boldsymbol{\theta}}(\boldsymbol{z})$ to encourage the disentanglement of its representation as done by the VAE. Whereas the notion of the VAE's prior is limited to the disentanglement of static distributions, we extend the idea of the prior to temporal domain space.

Following Wang et al. [371], we consider a stochastic process in the latent space. In such a process, slow dynamics are characterized as memoryless and can be described as a Markovian process. A distribution of such a process, after a short time interval $\tau > 0$, can be well approximated as a Gaussian function centered on scaled and/or drifted coordinates of the initial position. Among the various stationary processes with stationary distributions, the Ornstein-Uhlenbeck (OU) process [105] fulfills the above requirements, exhibiting a stationary distribution characterized by a Gaussian distribution. The OU process of one-dimensional stochastic latent variable z_t is defined by

$$\mathrm{d}z_t = -\gamma z_t \mathrm{d}t + \sqrt{D}\mathrm{d}W_t. \tag{7.12}$$

Here, $\gamma > 0$, $D > 0$ are parameters, and W_t denotes the Wiener process. The conditional probability density $p(z_{t+\tau} \mid z_t)$ of the process is given by

$$p\left(z_{t+\tau} \mid z_t\right) = \frac{1}{\sqrt{\frac{2\pi D}{\gamma}\left(1 - e^{-2\gamma\tau}\right)}} \exp\left(-\frac{\left(z_{t+\tau} - z_t e^{-\gamma\tau}\right)^2}{\frac{2D}{\gamma}\left(1 - e^{-2\gamma\tau}\right)}\right) \tag{7.13}$$

$$= \mathcal{N}\left(z_{t+\tau}; z_t e^{-\gamma\tau}, \frac{D}{\gamma}\left(1 - e^{-2\gamma\tau}\right)\right) \tag{7.14}$$

A more general process was described by Wang et al. [371] where they considered an effective potential energy function in the latent space. Here, we do not introduce a potential energy function in the latent space because our objective is not to reduce the energy landscape but to achieve a disentangled representation without any trapping in energy minima. As will be shown below, this simple process can be conceptually related to the VAE prior, and its autocorrelations can be regulated by a single parameter.

When we set $D/\gamma = 1$, the joint probability density of z_t and $z_{t-\tau}$ is given by $p(z_t, z_{t-\tau}) = p(z_{t-\tau})p(z_t|z_{t-\tau})$, where $p(z_{t-\tau})$ is the stationary distribution $\mathcal{N}(z_{t-\tau}; 0, 1)$, and the conditional probability density is given by Eq. 7.14. This results in

$$p(z_t, z_{t-\tau}) = \mathcal{N}(z_{t-\tau}; 0, 1)\mathcal{N}(z_t; \alpha z_{t-\tau}, 1 - \alpha^2), \tag{7.15}$$

where $\alpha = e^{-\gamma\tau}$ $(0 < \alpha < 1)$. Here, the variance of the second distribution on the right-hand side corresponds to the autocorrelation between z_t and $z_{t+\tau}$, explicitly parametrized by α. Consequently, the autocorrelation of the OU process can be easily imposed in the latent space by specifying α, facilitating the capture of slow protein dynamics.

In our proposed RL methods, tsVAE and tsTVAE, we impose the above joint probability density derived from the OU process as a prior in the latent space. Returning to the d_z-dimensional latent space $\boldsymbol{z}_t$ where components are independent of each other, the prior can be described as,

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_t, \boldsymbol{z}_{t-\tau}) = p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t-\tau})p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-\tau}) \tag{7.16}$$

$$= \mathcal{N}(\boldsymbol{z}_{t-\tau}; \boldsymbol{0}, I)\, \mathcal{N}(\boldsymbol{z}_t; \boldsymbol{\alpha} \odot \boldsymbol{z}_{t-\tau}, \mathrm{diag}(1 - \boldsymbol{\alpha}^{\circ 2})), \tag{7.17}$$

where $\boldsymbol{\alpha} \in \mathbb{R}^{d_z}$ is a d_z-dimensional parameter vector. The first normal distribution of the prior $\mathcal{N}(\boldsymbol{z}_t; \boldsymbol{0}, I)$ is the same as that used in the VAE and TVAE. As mentioned, it enhances disentanglement by isotropically distributing the latent variables. The second normal distribution $\mathcal{N}(\boldsymbol{z}_t; \boldsymbol{\alpha} \odot \boldsymbol{z}_{t-\tau}, \mathrm{diag}(1 - \boldsymbol{\alpha}^{\circ 2}))$, which we refer to as *time-structure-based prior*, encourages the position of $\boldsymbol{z}_t$ to be near time-proximal $\boldsymbol{z}_{t-\tau}$ while maintaining a certain covariance around. As noted above, this second prior corresponds to the autocorrelation between $\boldsymbol{z}_{t-\tau}$ and $\boldsymbol{z}_t$. In the limit of $\boldsymbol{\alpha} \to \boldsymbol{0}$, the second prior converges to $\mathcal{N}(\boldsymbol{z}_t; \boldsymbol{0}, I)$ and the whole prior

becomes $p_{\boldsymbol{\theta}}(\boldsymbol{z}_t, \boldsymbol{z}_{t-\tau}) = \mathcal{N}(\boldsymbol{z}_{t-\tau}; \mathbf{0}, I)\mathcal{N}(\boldsymbol{z}_t; \mathbf{0}, I)$, essentially same as that of the VAE, suggesting that the tsVAE and tsTVAE are the natural time-domain extensions of the VAE.

Altogether, the second prior works to make time-proximal samples spatially close to each other, while the first prior makes the other samples isotropically distributed or disentangled. Also, states far apart in time but visited many times in MD trajectories can be brought closer together by the reconstruction loss. These effects complementarily explain state transitions in latent space while collapsing fast modes.

The model parameters $\boldsymbol{\varphi}$ and $\boldsymbol{\theta}$ are learned by minimizing the negative bound

$$l^{\text{ts(T)VAE}} = l_r^{\text{ts(T)VAE}} + \beta\, l_p^{\text{ts(T)VAE}} + l_s^{\text{ts(T)VAE}} \tag{7.18}$$

$$\begin{aligned}
&= -\frac{1}{2|\mathcal{B}|}\sum_{t\in\mathcal{B}}\sum_{i=0}^{1}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t+(i+\delta^{\text{tsTVAE}})\tau}|\boldsymbol{z}_{t+i\tau}) \\
&\quad + \beta\,\frac{1}{2|\mathcal{B}|}\sum_{t\in\mathcal{B}}\mathrm{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t|\boldsymbol{x}_t)\|p_{\boldsymbol{\theta}}(\boldsymbol{z_t})) \\
&\quad + \frac{1}{2|\mathcal{B}|}\sum_{t\in\mathcal{B}}\mathrm{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}_{t+\tau}|\boldsymbol{x}_{t+\tau})\|p_{\boldsymbol{\theta}}(\boldsymbol{z}_{t+\tau}|\boldsymbol{z}_t)))
\end{aligned} \tag{7.19}$$

$$\begin{aligned}
&= \frac{1}{2|\mathcal{B}|}\sum_{t\in\mathcal{B}}\sum_{i=0}^{1}\|\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}_{t+i\tau}) - \boldsymbol{x}_{t+(i+\delta^{\text{tsTVAE}})\tau}\|^2 \\
&\quad + \beta\,\frac{1}{2|\mathcal{B}|}\sum_{t\in\mathcal{B}}\{\boldsymbol{\mu}_{\boldsymbol{\varphi}}(\boldsymbol{x}_t)^T\boldsymbol{\mu}_{\boldsymbol{\varphi}}(\boldsymbol{x}_t) + \|\sigma_{\boldsymbol{\varphi}}(\boldsymbol{x}_t)\|^2 - 2\|\log\sigma_{\boldsymbol{\varphi}}(\boldsymbol{x}_t)\|_1\} \\
&\quad + \frac{1}{2|\mathcal{B}|}\sum_{t\in\mathcal{B}}\sum_{d=1}^{d_z}\left\{\frac{(\mu_{\boldsymbol{\varphi}}(\boldsymbol{x}_{t+\tau})_d - \alpha_d z_{t,i})^2}{1-\alpha_d^2} + \frac{\sigma_{\boldsymbol{\varphi}}(\boldsymbol{x}_{t+\tau})_d^2}{1-\alpha_d^2} - \log\sigma_{\boldsymbol{\varphi}}(\boldsymbol{x}_{t+\tau})_d^2\right\} \\
&\quad + \text{const.},
\end{aligned} \tag{7.20}$$

where $\delta^{\text{tsTVAE}} = 1$ (tsTVAE); 0 (tsVAE) represent whether the model is the tsVAE or the tsTVAE. The equation's third term (from the time-structure-based prior) works as a regularization of transitional motions during τ. The detailed training process for the tsVAE and the tsTVAE is shown in Algorithm 5.

Algorithm 5 Training of tsVAE and tsTVAE

Ensure: model $q_{\boldsymbol{\varphi}}$ and $p_{\boldsymbol{\theta}}$, data $\mathcal{D}$
1: **for** minibatch $\mathcal{B} \subset \mathcal{D}$ **do**
2: Initialize loss $l = 0$
3: **for** $(\boldsymbol{x}_t, \boldsymbol{x}_{t+\tau}, \boldsymbol{x}_{t+2\tau}) \in \mathcal{B}$ **do**
4: **for** $i \in \{0, 1\}$ **do**
5: Samples latent variables $\boldsymbol{z}_{t+i\tau} \sim q_{\boldsymbol{\varphi}}(\boldsymbol{z}_{t+i\tau}|\boldsymbol{x}_{t+i*\tau})$
6: Compute the log-likelihood of the emission $l = l - \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t+(i+\delta^{\text{tsTVAE}})\tau}|\boldsymbol{z}_{t+i\tau})$
7: **end for**
8: Compute the KL divergence of the latent prior $l = l + \mathrm{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t|\boldsymbol{x}_t)\|\mathcal{N}(0,1))$
9: Compute the KL divergence of the latent transition $l = l + \mathrm{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}_{t+\tau}|\boldsymbol{x}_{t+\tau})\|\mathcal{N}(\boldsymbol{\alpha}\odot\boldsymbol{z}_t, \mathrm{diag}(1-\boldsymbol{\alpha}^{\circ 2})))$
10: **end for**
11: Backward loss $\frac{1}{2|\mathcal{B}|}l$ and update model parameters
12: **end for**

7.3 Methods

7.3.1 Molecular Dynamics Simulations

We compared RL methods through the analysis of MD trajectory data sets of alanine-dipeptide and chignolin. MD trajectory data of alanine-dipeptide[123, 274] was taken from a public data (`https://markovmodel.github.io/mdshare/ALA2/#alanine-dipeptide`) provided by the Computational Molecular Biology Group, Freie Universität Berlin. The total MD simulation length of the data is 250 ns performed using ACEMD.[123] Amber ff99SB-ILDN force-field [211] was used for proteins, and the TIP3P model[163] was used for water molecules. All bonds involving hydrogen atoms were constrained.[314] Electrostatic interactions were treated using the smooth particle mesh Ewald method.[71] Temperature was controlled at 300 K by using Langevin dynamics.

MD trajectory data of chignolin was generated by Prof. Matsunaga through performing MD simulations. The total simulation length of 10 μs was performed with NAMD version 3.0 alpha (`http://www.ks.uiuc.edu/Research/namd/`).[285] The input file for the MD simulation of chignolin was prepared by using the CHARMM-GUI web server.[161] The initial structure was taken from the PDB ID of 1UAO.[137] The native structure was used as the initial state for the 200 ns simulation, and data were sampled every ten ps. CHARMM36m[143] was used as the force field, and the TIP3P[163] model was used for water molecules. The covalent bonds, including hydrogen atoms, were constrained.[314] Electrostatic interactions were treated using the smooth particle mesh Ewald method.[71] After equilibrating the system under NPT (300 K and 1 atm), a production run was conducted under NVT condition (300 K). The temperature was controlled by using Langevin dynamics. Five unfolding and four folding events were observed in the total 10 μs length simulation.

7.3.2 Markov State Model Analysis

We here used the Markov state model (MSM)[146] as a downstream task for the embedded representation. By constructing MSMs and assessing their properties, we compared the tsVAE and tsTVAE with PCA, tICA [268, 284, 326], TAE [381], TVAE [136], and VDE [131]. The construction of MSM followed the standard procedure widely used in the community with the PyEMMA package [323]:

1. Feature selection. Aligned heavy-atom Cartesian coordinates were used for the alanine-dipeptide as features ($d_x = 30$). As preprocessing, the coordinates were z-scaled and transformed into triples $\{(\boldsymbol{x}_t, \boldsymbol{x}_{t+\tau}, \boldsymbol{x}_{t+2\tau})\}_{t=1}^{T-2\tau}$. For chignolin, the contact map vector was used as a feature. Specifically, the distance between alpha carbon atoms of non-adjacent residues was extracted from the MD data to obtain $d_x = 28$ data. The data was further transformed by $\exp(-d)$ and whitened to remove correlations between variables.

2. Representation learning. We embedded the selected features into the latent space by the PCA, tICA [268], TAE [381], TVAE [136], VDE [131], SRV [53], tsVAE, and tsTVAE.

3. Clustering. Embedded samples in the latent space by each method were clustered by the k-means method with $k = 100$, and discrete trajectories were obtained.

4. Construction of MSM. We constructed MSM by estimating a transition matrix by counting the number of transition events in the clustered discrete trajectories.

5. Coarse-graining or lumping MSM. The PCCA+[302] algorithm was applied to compute macrostates decomposition of MSM states.

We compared the RL methods by assessing implied timescales (ITS) and eigenvectors of the transition matrix of MSM. The i-th ITS t_i is defined by

$$t_i(\tau_m) = -\frac{\tau_m}{\log \lambda_i(\tau_m)}, \tag{7.21}$$

where τ_m is the MSM lagtime, λ_i is the $(i+1)$-th eigenvalue of the transition matrix.

PCA and tICA were computed using the PyEMMA package [323], and the neural network models were implemented using PyTorch framework [283]. The decoder network $D_{\boldsymbol{\varphi}}$ and the encoder network $E_{\boldsymbol{\theta}}$ are composed of the number of hidden layers 2, respectively. Each layer has 50 units, and the latent dimension is $d_z = 2$. We set the model lagtime $\tau = 50$, the sample autocorrelation coefficient of VDE $\gamma = 1$, and the disentangled regularization coefficient $\beta = 0$. The neural network models were trained using the Adam optimizer [173] of a learning rate 10^{-3}, a batch size 256, and 100 epochs.

The hyperparameters $\boldsymbol{\alpha}$ in the tsVAE and tsTVAE were determined by considering the decaying time τ_d of autocorrelation in the latent space. The decay time τ_d $(d = 1, 2)$ and α_d $(d = 1, 2)$ can be theoretically related by

$$\alpha_d = \left(\frac{\Phi^{-1}(a)}{\sqrt{\tau_d}} \right)^{\frac{\tau}{\tau_d}}, \tag{7.22}$$

where $a = 0.95$ and Φ represent a significance level and the cumulative distribution function of the standard normal distribution, respectively. This relation means that the autocorrelation of lag time τ_d enters the accepted region of the uncorrelated hypothesis. If the sample autocorrelation is sufficiently attenuated, the accepted region is given by $\Phi^{-1}(a)/\sqrt{\tau_d}$ since it does not differ from the sample autocorrelation from the standard normal population. Since the time evolves τ_d/τ times by the decay time, $(\alpha_d)^{\tau_d/\tau}$ is the accepted region. In this study, we chose $\tau_d = 10^5$ $(d = 1, 2)$ sampling steps for both alanine-dipeptide and chignolin and computed α_d $(d = 1, 2)$ using this relation.

7.4 Results

7.4.1 Alanine-dipeptide

Figure 7.2 compares MSM properties obtained from the alanine-dipeptide MD data. Since it is known that the alanine-dipeptide is well characterized by the two backbone dihedral angles (ϕ, ψ), we here investigated whether embedded spaces can achieve close correspondence to reference dihedral space (ϕ, ψ) only from the Cartesian coordinates (features used in MSM analysis). Figure 7.2(a) compares the convergence of the first three ITSs as a function of the MSM lagtime τ_m. The ITSs of MSM using TAE, TVAE, VDE, tsVAE, and tsTVAE converge to the time scales comparable to those of the MSM constructed in the reference space (ϕ, ψ). This means these methods successfully extract slow dynamics mainly determined by the dihedral angles (ϕ, ψ). On the other hand, while the first two ITSs of MSM of tICA and SRV converge to the value comparable to those of the reference space (ϕ, ψ), the third implied timescale is significantly lower. In the case of PCA, the three ITSs of MSM are consistently lower than those of the reference space.

Next, we investigated whether the modes corresponding to the slowest three ITSs are well captured from the input Cartesian coordinates. Figure 7.3 shows the first three scaled right eigenvector maps in the reference space (ϕ, ψ). The reference MSM shows that the first three slowest dynamics of the alanine-dipeptide are the ϕ-rotation, the ψ-rotation with Gauche-negative ϕ, the ψ-rotation with Gauche-positive ϕ (more precisely, this shows the transition to left-handed helix region in the ABEGO-type Ramachandran plot), respectively. The three right eigenvectors of TAE, TVAE, VDE, tsVAE, and tsTVAE are consistent with those of the reference. On the other hand, PCA and tICA fail to capture the correspondent vectors, suggesting the superiority of non-linear transformations over these linear ones in obtaining a mapping from Cartesian coordinates to dihedral angles. SRV fails to capture the third correspondent vector because the method excessively focuses on extracting the two slowest dynamics. These results confirm that the tsVAE and tsTVAE perform comparably with the other state-of-the-art nonlinear methods in this simple system.

7.4.2 Chignolin

We evaluated the RL methods by analyzing the properties of constructed MSMs of chignolin's folding/unfolding dynamics. Figure 7.4(a) and (b) show the representative structures of folded and unfolded states. The MD data for chignolin exhibits a more complex behavior than that of the alanine-dipeptide and is comparatively

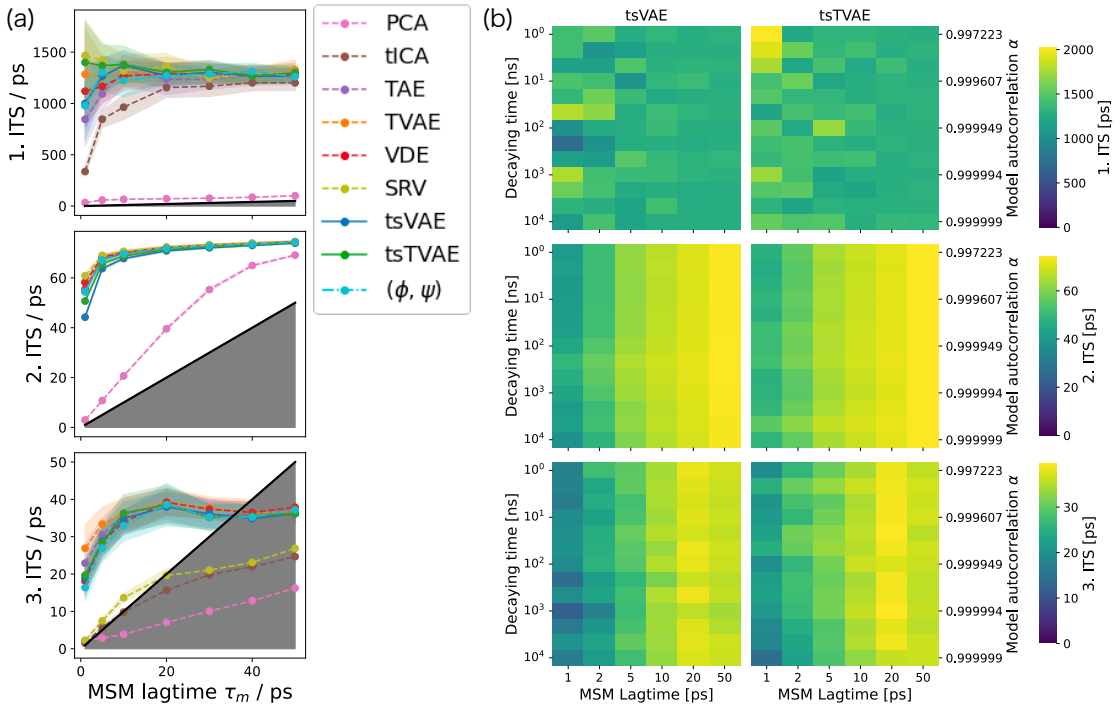


Figure 7.2: **Comparison of Markov state models of alanine-dipeptide trajectory.** (a) The first three ITSs of MSMs constructed in the encoded space. (b) Heatmaps of the first three ITSs against the decaying time τ_d and the MSM lagtime τ_m.

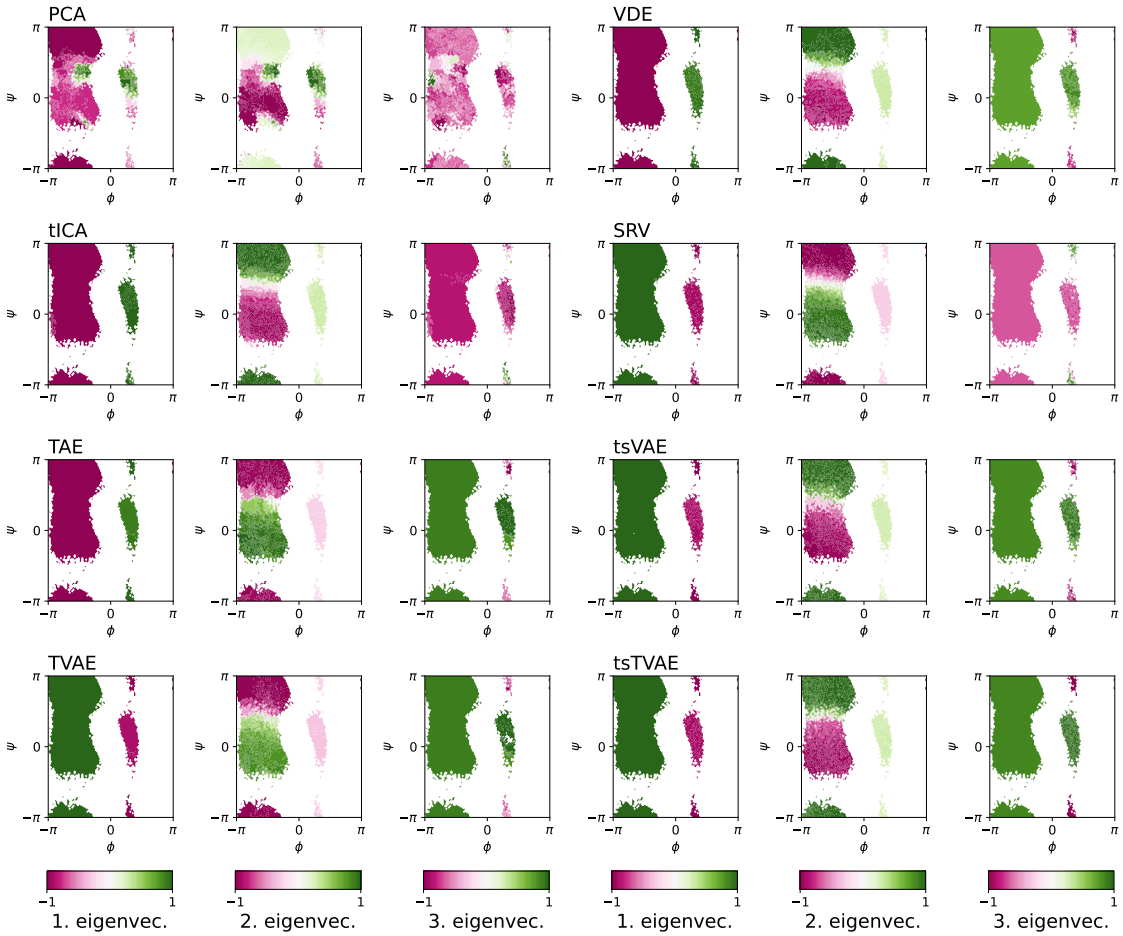


Figure 7.3: The first three scaled right eigenvector maps of MSM with lagtime 50 ps in the reference space (ϕ, ψ) for the alanine-dipeptide trajectories.

short, encompassing only five unfolding and four folding events within 10 μs length simulation. As such, this presents a more stringent comparison of the RL methods under consideration.

Figure 7.4(c) depicts the first two ITSs of the MSMs constructed in the latent space. The figure shows that the ITSs of TAE, TVAE, VDE, SRV, tsVAE, and tsTVAE converge to slow timescales comparable with each other. In contrast, the 2nd ITSs of PCA and tICA do not converge to these time-scales. This indicates that nonlinear transformations are crucial for capturing the conformational transitions of chignolin. As described later, the second slowest dynamics is the dihedral motion, which a linear transformation from the input feature distance cannot obtain. The figure also demonstrates that the 2nd ITSs of SRV, tsVAE, and tsTVAE converge more rapidly than the other methods. This suggests that the three methods can robustly capture transitional motions, even when MD data is not long enough, as in this case. This rapid convergence of tsVAE and tsTVAE could be explained by the inductive bias due to the time-structure-based prior that samples closer in time will also be closer in the latent space.

The macrostates of each MSM are visualized in Figure 7.5. In the figure, samples in the latent space are colored according to three macrostates. Since the native structure of chignolin is well characterized by hydrogen bonds between the backbone amide group of ASP3 and the carbonyl group of THR8 (ASP3N-THR8O), we used this distance as a reference feature. In addition, we used the dihedral angle ψ of ASP as a feature because this angle is found to be coupled to the folding and unfolding transition in our MD data. In the figure, the macrostates of PCA and tICA are considered mixed (i.e., entangled) since the states

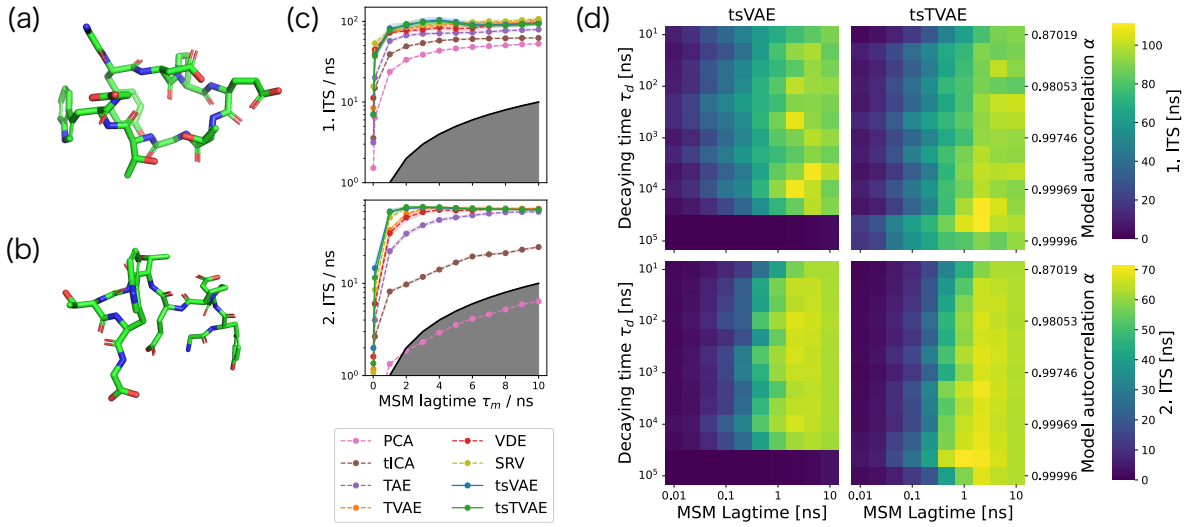


Figure 7.4: **Comparison of Markov state models of chignolin folding and unfolding trajectory.** (a) Representative structure of native states (b) Representative structure of unfolded states (c) The first two ITSs of MSMs constructed in the encoded space. (d) Heatmaps of the first two ITSs against the decaying time τ_d and the MSM lagtime τ_m.

overlap in the reference feature space. On the other hand, the transitions among macrostates of TAE, TVAE, VDE, SRV, tsVAE, and tsTVAE are well correlated with the formation of contact between ASP3 and THR8 and the dihedral angle ψ of ASP. This concludes that these RL methods successfully capture important physical interactions upon the folding and unfolding transitions. Upon closer examination of the figure, it also shows that the SRV and tsVAE clearly disentangle the folding and unfolding transitions without mixing them. Indeed, z_2 in the tsVAE and z_1 in SRV correspond better with the ψ-rotation compared to the other methods. This disentangling of tsVAE could be explained by the property of the method; the time-structure-based prior promotes time-local autocorrelation for each axis while the representations of each axis were disentangled to improve reconstruction accuracy.

We quantitatively evaluated disentanglement using total variation (TV) similarity. The TV similarity between two probability distributions p_1 and p_2 on measurable space $(\mathcal{X}, \mathcal{F}(\mathcal{X}))$ is formulated by

$$\mathrm{TVS}(p_1, p_2) = \int_{\mathcal{X}} \mathrm{d}x \min\{p_1(x), p_2(x)\} \tag{7.23}$$

$$= 1 - \frac{1}{2}\|p_1 - p_2\|_{\mathrm{TV}}, \tag{7.24}$$

where $\|\cdot\|_{\mathrm{TV}}$ represents the TV norm. The lower the similarity, the more divergent the two distributions are. We define the TV similarity of a variable $x \in \mathcal{X}$ (distance between atoms or dihedral angle) along the z_i axis (where $i = 1, \ldots, d_z$) in the latent space by

$$\mathrm{TVS}_i^{\mathcal{X}} := \inf_{z_i^\xi \in [z_i^{\min}, z_i^{\max}]} \mathrm{TVS}(p(x|z_i \leq z_\xi), p(x|z_i > z_\xi)), \tag{7.25}$$

where z_i^ξ is the threshold of z_i, which minimizes the TV similarity of two probability distributions. When $\mathcal{X}$ is a bin set of histograms, the total variation similarity is computed by

$$\mathrm{TVS}(p_1, p_2) = \sum_{i=0}^{N-1} \Delta x \min\{p_1(x + i\Delta x), p_2(x + i\Delta x)\}, \tag{7.26}$$

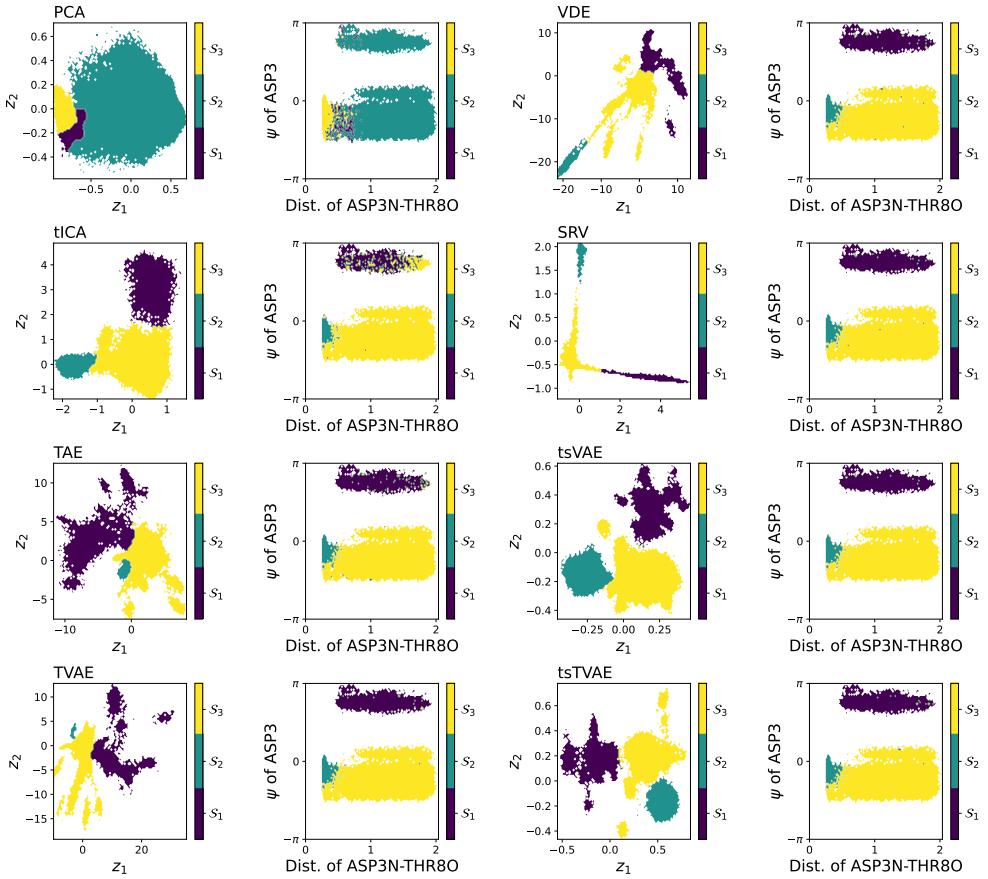


Figure 7.5: Estimated macro-states of MSM with lagtime 1 ns in the embedded and reference space for chignolin trajectories

where Δx is the bin-size and N is the the number of divisions. We created the histograms whose bin edges are estimated by the Freedman Diaconis estimator[96]. We use the space $\mathcal{X}$ as each distance between the no-adjacent pairs of the residues and each dihedral angle of the residues. The exploring range of the threshold is set to be $[\min_{z_i \in \mathcal{D}_{z_i}} z_i + \Delta z_i, \max_{z_i \in \mathcal{D}_{z_i}} z_i - \Delta z_i]$, where $\Delta z_i = (\max_{z_i \in \mathcal{D}_{z_i}} z_i - \min_{z_i \in \mathcal{D}_{z_i}} z_i)/10$ and $\mathcal{D}_{z_i}$ is the trajectory data of z_i.

Figure 7.6 shows the ten lowest TV similarities along the z_1 and z_2 axes. The orange bars correspond to the ASP3-THR8 distance and the dihedral angle ψ of ASP3, which are important for the folding/unfolding transitions. PCA, tICA, and TVAE fail to capture the two features by the minimum TV. TAE captures the dihedral angle as the minimum TVS along z_1; however, it fails to capture the distance by the minimum TVS along z_2. SRV captures the dihedral angle as the minimum TVS along z_1; however, the feature that achieves the minimum TVS along z_2 is also the dihedral angle. The tsVAE and tsTVAE capture the dihedral angle and distance by the minimum TVS along z_1 and z_2, respectively; this means the methods realize interpretable disentanglement in latent space.

7.4.3 Robustness against the choice of hyperparameters

The tsVAE and tsTVAE have the hyperparameters $\boldsymbol{\alpha}$ in their time-structure-based prior, which the user should give in advance. The robustness of results against the choice of the parameters $\boldsymbol{\alpha}$ is important for the practical use of the methods. To evaluate the robustness, we conducted additional MSM analysis for alanine-dipeptide and chignolin trajectories. Figure 7.2(b) and Figure 7.4(d) show ITSs of MSMs constructed

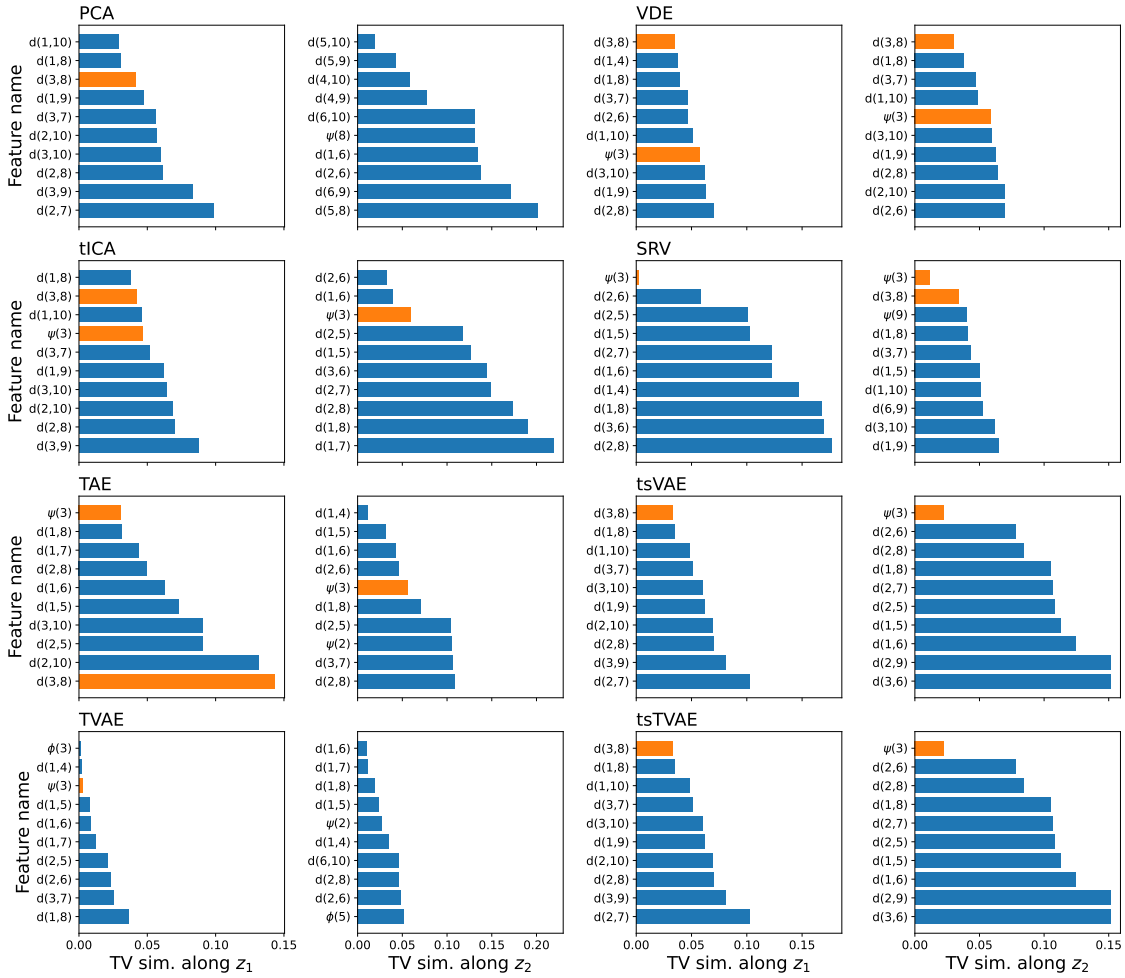


Figure 7.6: Total variation similarity of distances between atom and dihedral angles along the axes of z_1 and z_2 in the latent space. The orange bars indicate the ASP3-THR8 distance and the dihedral angle ψ of ASP3, which are important for the folding/unfolding transitions of chignolin.

in the latent space with different $\boldsymbol{\alpha}$. As described in Eq. (7.22), $\boldsymbol{\alpha}$ can be related to the decay time τ_d of autocorrelation in the latent space as described in Eq. (7.22). The first three ITSs for alanine-dipeptide trajectories have similar values for $\tau_d = 10^3$ ps to $\tau_d = 10^7$ ps. The first two ITSs for chignolin trajectories also have qualitatively almost the same values from $\tau_d = 10^2$ ns to $\tau_d = 10^4$ ns. Overall, these results suggest the robustness of tsVAE and tsTVAE against the choice of the hyperparameters $\boldsymbol{\alpha}$.

Furthermore, to eliminate dependency on $\boldsymbol{\alpha}$, we introduced a prior for $\boldsymbol{\alpha}$ and estimated plausible values of $\boldsymbol{\alpha}$ from the data in Supporting Information.

We evaluated the robustness of each model against the choice of the model lagtime τ by varying τ. Figure 7.7 shows the first three ITSs as a function of the model lagtime τ for alanine-dipeptide trajectories. The ITSs of TAE, TVAE, and VDE are robust to the choice of τ in [1 ps, 10 ps]. The tsVAE and tsTVAE have slight variations similar to those methods, while the second and the third ITS slightly decrease with increasing τ. This is because the tsVAE and tsTVAE are biased toward reducing features on fast time-scales. The dynamics of the second and the third ITSs are too fast for these methods and are subject to the reduction. As shown in Figure 7.8, the ITSs of PCA, tICA, and TAE are significantly lower than those of other methods. In contrast, the ITSs of TVAE, VDE, and SRV remain robustly high across a wide range from

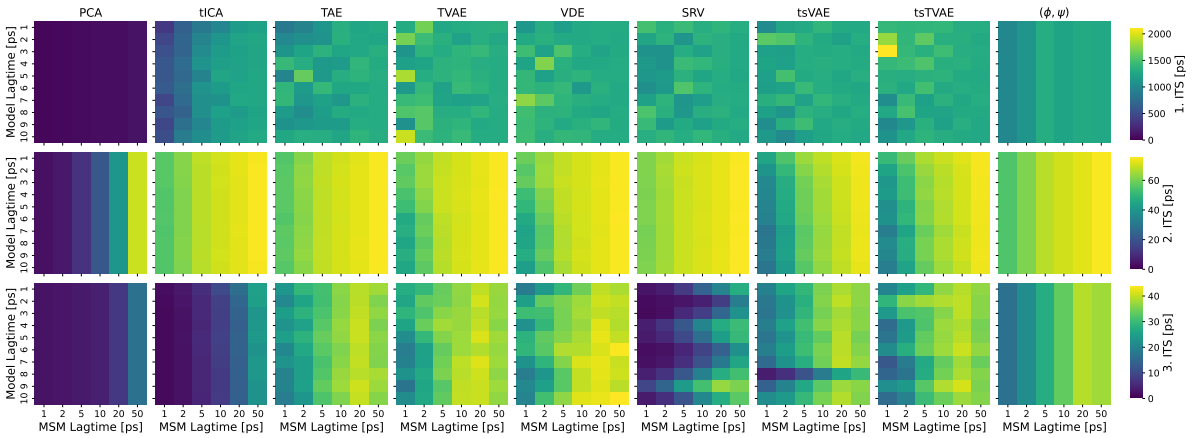


Figure 7.7: Heatmaps of the first three ITSs against the model lagtime τ and the MSM lagtime τ_m for the alanine-dipeptide trajectories.

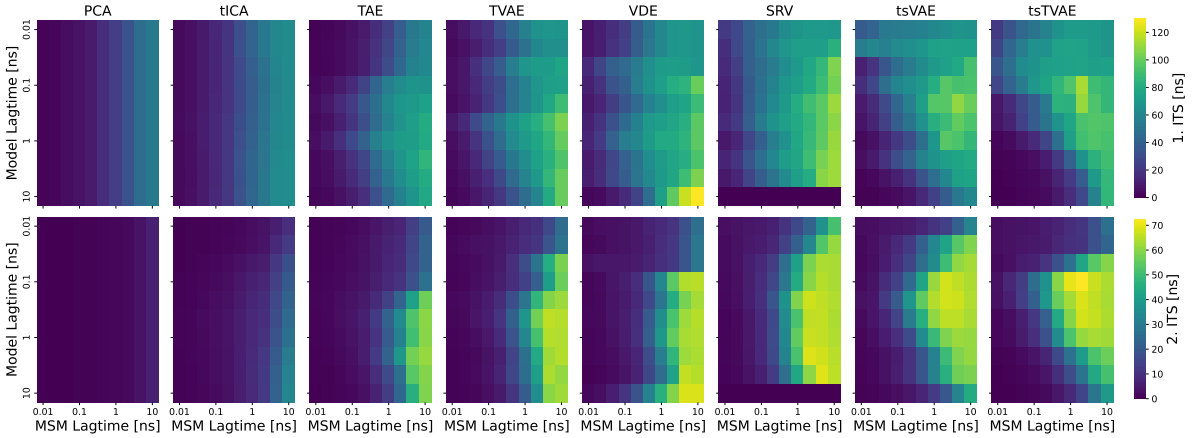


Figure 7.8: Heatmaps of the first two ITSs against the model lagtime τ and the MSM lagtime τ_m for the chignolin trajectories.

$\tau = 10^{-1}$ ns to $\tau = 10$ ns, suggesting the impact of the sampling from the inference distribution on reducing the fast time-scales without overfit to the data. Additionally, the figure demonstrates that the ITSs of tsVAE and tsTVAE are significantly higher than those of the other methods in the surroundings of $\tau = 10^{-1}$ ns, suggesting the impact of the time-structure-based prior. Indeed, the ITSs tend to become high for relatively small τ because small τ increase the impact of the time-structure-based prior, which embeds samples close in the time-to-close position in the latent space. If the scale of the decay time is well chosen, the methods can extract slower dynamics better than other methods.

Another important hyperparameter is the learning rate η, used in the Adam optimizer for neural networks. The ITSs of tsVAE and tsTVAE are robustly higher than the other methods in $\eta \in [10^{-4}, 10^{-2}]$. Detailed results are given in Supporting Information.

7.5 Discussion

In this study, we have proposed two RL methods, tsVAE and tsTVAE, based on a simple time-structure-based prior. Through the comparison of MSM properties constructed from alanine-dipeptide's MD trajectories, we have shown that the proposed methods can capture slow dynamics comparable to state-of-the-art methods.

Furthermore, as shown by the macrostate analysis of coarse-grained MSMs for chignolin's MD trajectories, the tsVAE or tsTVAE can learn disentangled representations that are well correlated with physically important interactions. This suggests that the proposed methods can obtain widely disentangled representations, leading to high interpretability. In the Supporting Information, we show a systematic analysis that can detect physically important interactions for the transitions between macrostates by analyzing the contributions of physical coordinates to the distributions of macro-states.

Although the tsVAE and tsTVAE methods are promising alternatives to the current state-of-the-art methods, we would like to discuss the limitations of these methods. The time-structure-based prior makes time-proximal samples spatially close to each other. Sometimes, this effect introduces a bias into the dynamics in the latent space, leading to incorrect results in downstream tasks, including MSM analysis. For example, suppose the MD simulation is long enough to repeatedly visit the same metastable state in its simulation. Still, the visiting time is far apart from each other and failed to capture with the reconstruction loss. In that case, a bias is introduced by the time-structured-based prior by capturing this state as a different state in the latent space. To avoid this bias, we plan to develop an algorithm that iteratively clusters and learns the embedding, such as SwAV[43] and PCL[202].

There would be two major future directions in this study. The first direction is to apply the methods for enhanced sampling of MD simulation. For example, in the Reweighted Autoencoded Variational Bayes for Enhanced Sampling (RAVE),[305] and Deep-TICA[35], RL methods have been successfully combined with enhanced sampling techniques for biomolecular conformational sampling. Combination with the development version of OpenMM[84] or PLUMED-LibTorch[358] interface would be suitable to apply the biased forces from the latent variables of the tsVAE or tsTVAE as CVs for enhanced sampling.

The second direction is to extend the tsVAE and tsTVAE in the context of contrastive learning. Contrastive learning learns the general features of a dataset without labels by optimizing the embedding model to determine whether data points are similar or dissimilar. Since the time-structured-based prior of the proposed methods brings the latent variable $\boldsymbol{z}_t$ and the time-proximal variable $\boldsymbol{z}_{t+\tau}$ closer together, the tsVAE and tsTVAE can be categorized to time contrastive learning[148]. Thus, introducing the state-of-the-art techniques of contrastive learning in our context will further improve the representation of protein dynamics.

8 Conclusion

This paper describes neural networks and representation learning for time series data. Representation learning automatically extracts meaningful features for explaining observation well. Because the technique saves several costs, such as data engineering and individual model training, representation learning has the potential for many applications. This paper proposes new representation learning methods for specific tasks and algorithms.

In Chapter 5, we propose a QuAD system that detects anomalies and estimates individual cycle time together. The system uses an attention mechanism that measures the relationship between the input query and the existing keys and then outputs the weighted sum of the values. The relationship can be considered a similarity between current and past local patterns. The system uses this explicit representation and outperforms the existing methods in terms of AUPRC for the Doppler radar data. Such explicit representation makes the system easy to inspect for practitioners.

In Chapter 6, we propose EnKO, the learning algorithm for sequential variational auto-encoders (SVAEs), by shedding light on particle diversity. Because the generating process of SVAEs is considered the probabilistic time-series model, sequential Bayes filtering methods can estimate latent state distributions. FIVO [238], an existing work, uses the particle filter for tightening the lower bound of the variational objective; however, the work suffers from particle degeneracy and the biased gradient estimator. The proposal EnKO method overcomes these drawbacks and outperforms the existing works regarding predictive accuracy for several ODE systems, human walking data, and handwritten digit rotating data. The proposed method enhances the inference process, i.e., inferred representations inherited in the original data.

In Chapter 7, we propose representation learning methods called tsVAE and tsTVAE for extracting the slow dynamics of biomolecules. Because the slow dynamics are closely related to biological processes in the human body, extracting the dynamics from trajectory data has been studied. To achieve this, the proposed methods enhance autocorrelation in latent space by using a highly autocorrelated prior related to the physical process described by the Ornstein-Uhlenbeck process. The proposed methods could capture the first several slowest dynamics and disentangle the representations by applying two biomolecule trajectories. The disentanglement means each latent dimension corresponds to a meaningful representation, and the characteristic is crucial for practical applications.

Our proposals center on specific tasks and algorithms due to the difficulty of generalized representation learning for time-series data. There are no applicable foundation models for time-series data because the data depend heavily on domains and cases. In our experiments, fan data in section 5.3 has quasi-periodicity and shaking patterns, Lorenz model in section 6.4 provides chaotic trajectory, Chignolin dynamics in section 7.4 is highly stochastic Markovian. Interpretation of the same local patterns varies from domain to situation. For example, a local pattern is considered noise, chaotic, or highly frequent dynamics. Compensation for these differences by foundation models and fine-tuning is difficult in current techniques.

On the other hand, the dataset space that can be generalized by representation learning is growing. Our research is also contributing to making this wave even more extensive. Time-series representation learning will continue to grow and incorporate the latest technologies, such as large language models (LLMs), graph neural networks (GNNs), optimal transport (OT), and diffusion models (DMs). For example, analyzing system data by LLMs contributes to alleviating domain- and task-specific dependency of time-series data. As described so far, classifying local patterns from only raw time-series data is difficult; however, it can be done with other data, such as system and alternative data. Since time-series relations are considered graphs, graph neural networks [421, 366, 184, 367, 273, 340] and geometric neural networks [291, 37, 20, 36] stimulate new methodologies in the time-series field. For instance, protein trajectories can be considered spatiotemporal graphical data, and the embedding with graph and geometric neural networks is intriguing. Optimal transport, also known as the Wasserstein distance or Earth Mover's Distance, is a concept from mathematics that has become increasingly important in machine learning, especially in tasks like domain adaptation, generative modeling, and clustering. The Wasserstein AE [355] uses Wasserstein distance instead of Kullback-Leibler divergence for evaluating the discrepancy of the variational posterior and the prior. Because OT provides advantages over KL divergence, such as geometric sensitivity, handling disjoint supports, robustness to sample noise, and clear interpretability, advanced methods that combine VAE and OT have

been researched [120, 419, 6]. Diffusion models [134] are a class of generative models in machine learning that have gained significant attention, especially in the generative modeling of images [316, 16, 290, 418], audio [214, 144, 324], and other complex data structures [180, 271, 142]. The models are inspired by the physical process of diffusion, where particles spread out from an area of high concentration to an area of low concentration over time. Because DMs are considered the stacking of VAE, the embedding framework by VAE can potentially be replaced by the models. Representation learning for time series data has many development factors, and buried time series data will continue to be unearthed and utilized.

Acknowledgement

This study benefited from the help of many researchers. I am deeply grateful to my advisor, Professor Kazuyuki Nakamura of Meiji University, for his guidance and encouragement. I am grateful to Professor Tomoyuki Ogawa, Associate Professor Yoshihiro Hirose, and Professor Nobuhiko J Suematsu of Meiji University for their advice at the research planning stage. I thank Prof. Tomoyuki Higuchi of Chuo University for providing the computational environment and guidance through the joint research on Chapters 5 and 6. I thank Associate Professor Kosuke Ohkusa of Chuo University for his help in preparing the data in the collaborative research in Chapter 5. I thank Associate Professor Yasuhiro Matsunaga of Saitama University for preparing the trajectory data in the joint research in Chapter 7 and for deep discussions. I thank Assistant Professor Sotaro Fuchigami of the University of Shizuoka for discussions as an expert in biophysics in the joint research on Chapter 7. I thank Professor Kei Hirose of Kyushu University, Associate Professor Ayaka Sakata and Associate Professor Keisuke Yano of The Institute of Statistical Mathematics, and Dr. Yu Ichida of Meiji University for the opportunity to give talks. I thank Ms. Yoko Yamamoto of Meiji University and Ms. Nobuko Fujiwara of Chuo University for their help in the administrative procedures related to the research. I thank Dr. Yu Ichida, Mr. Ryu Fujiwara, Mr. Yu Han of Meiji University, and Mr. Yu Oshima of Chuo University for many stimulating experiences. Finally, I would like to thank my family for raising me.

A Supplementary Information for EnKO

A.1 Algotirhm

The original EnKF [88] assumes a linear observation model with additive noise

$$\boldsymbol{x}_t = H_t \boldsymbol{z}_t + \boldsymbol{w}_t,\ \boldsymbol{w}_t \sim p_{\boldsymbol{\theta},w}(\boldsymbol{w}_t). \tag{A.1}$$

The EnKF updates the particles by

$$\boldsymbol{z}_t^{(i)} \sim p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}^{(i)}),\ \forall i \in \mathbb{N}_N \tag{A.2a}$$

$$\bar{\boldsymbol{z}}_t = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{z}_t^{(i)}, \tag{A.2b}$$

$$\Sigma_t^z = \frac{1}{N-1}\sum_{i=1}^{N} (\boldsymbol{z}_t^{(i)} - \bar{\boldsymbol{z}}_t)(\boldsymbol{z}_t^{(i)} - \bar{\boldsymbol{z}}_t)^T, \tag{A.2c}$$

$$\boldsymbol{w}_t^{(i)} \sim p_{\boldsymbol{\theta},w}(\boldsymbol{w}_t),\ \forall i \in \mathbb{N}_N, \tag{A.2d}$$

$$\bar{\boldsymbol{w}}_t = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{w}_t^{(i)}, \tag{A.2e}$$

$$\Sigma_t^w = \frac{1}{N-1}\sum_{i=1}^{N} (\boldsymbol{w}_t^{(i)} - \bar{\boldsymbol{w}}_t)(\boldsymbol{w}_t^{(i)} - \bar{\boldsymbol{w}}_t)^T, \tag{A.2f}$$

$$K_t = \Sigma_t^z H_t^T (H_t \Sigma_t^z H_t^T + \Sigma_t^w)^{-1}, \tag{A.2g}$$

$$\boldsymbol{z}_t^{f,(i)} = \boldsymbol{z}_t^{(i)} + K_t(\boldsymbol{x}_t - H_t \boldsymbol{z}_t^{(i)} - \boldsymbol{w}_t^{(i)}),\ \forall i \in \mathbb{N}_N, \tag{A.2h}$$

where $\mathbb{N}_N = \{1, \cdots, N\}$, superscript T indicates vector or matrix transpose, and $\boldsymbol{z}_t^{f,(i)}$ denotes the i-th latent particle after filtering at time t.

The original EnKF is easily applied to the nonlinear observation model

$$\boldsymbol{x}_t \sim p_{\boldsymbol{\theta}}(\boldsymbol{z}_t) \tag{A.3a}$$

$$\overset{\mathbb{E}[\boldsymbol{z}_t]\ exists}{\Longleftrightarrow} \boldsymbol{x}_t = \mathbb{E}[\boldsymbol{z}_t] + (\boldsymbol{x}_t - \mathbb{E}[\boldsymbol{z}_t]),\ \boldsymbol{x}_t \sim p_{\boldsymbol{\theta}}(\boldsymbol{z}_t) \tag{A.3b}$$

$$\Longleftrightarrow \boldsymbol{x}_t = \boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{z}_t) + \boldsymbol{w}_t,\ \boldsymbol{w}_t \sim p_{\boldsymbol{\theta},w}(\boldsymbol{w}_t) \tag{A.3c}$$

by the augmented PTSM

$$\begin{aligned}
\tilde{\boldsymbol{z}}_t &= \begin{pmatrix} \boldsymbol{z}_t \\ \boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{z}_t) \end{pmatrix} \\
&\sim \tilde{p}_{\boldsymbol{\theta}}(\tilde{\boldsymbol{z}}_t|\tilde{\boldsymbol{z}}_{1:t-1}) = p_{\boldsymbol{\theta}}\left(\begin{pmatrix} I_{d_z} & O_{d_x} \end{pmatrix}\tilde{\boldsymbol{z}}_t \middle| \begin{pmatrix} I_{d_z} & O_{d_x} \end{pmatrix}\tilde{\boldsymbol{z}}_{1:t-1}\right), \qquad (4.19a) \\
\boldsymbol{x}_t &= \begin{pmatrix} O_{d_z} & I_{d_x} \end{pmatrix}\tilde{\boldsymbol{z}}_t + \boldsymbol{w}_t = \tilde{H}_t \tilde{\boldsymbol{z}}_t + \boldsymbol{w}_t \\
&\sim \tilde{p}_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\tilde{\boldsymbol{z}}_t) = p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_t \middle| \begin{pmatrix} I_{d_z} & O_{d_x} \end{pmatrix}\tilde{\boldsymbol{z}}_t\right) \qquad (4.19b)
\end{aligned}$$

for the augmented latent states $\tilde{\boldsymbol{z}}_t \in \mathbb{R}^{d_z+d_x}$, where I_d and O_d are the d-dimensional square identity matrix and zero matrix, respectively. By Equation 4.19b, the nonlinear emission is regarded as a linear representation; then, the augmented PTSM can be applied to Equations A.2.

This Chapter is based on "Ensemble Kalman variational objective: a variational inference framework for sequential variational auto-encoders" [154], by the same author, which appeared in the Proceedings of *Nonlinear Theory and Its Applications*,

The first d_z rows of the Kalman gain in the generalized EnKF are obtained by

$$\begin{aligned}
(\tilde{K}_t)_{:d_z} &= \left(\Sigma_t^{\tilde{z}} \tilde{H}_t^T (\tilde{H}_t \Sigma_t^{\tilde{z}} \tilde{H}_t^T + \Sigma_t^w)^{-1}\right)_{:d_z} \\
&= \left(\begin{pmatrix} \Sigma_t^z & \Sigma_t^{z\mu_x} \\ \Sigma_t^{\mu_x z} & \Sigma_t^{\mu_x} \end{pmatrix} \begin{pmatrix} O_{d_z} \\ I_{d_x} \end{pmatrix} \left(\begin{pmatrix} O_{d_z} & I_{d_x} \end{pmatrix} \begin{pmatrix} \Sigma_t^z & \Sigma_t^{z\mu_x} \\ \Sigma_t^{\mu_x z} & \Sigma_t^{\mu_x} \end{pmatrix} \begin{pmatrix} O_{d_z} \\ I_{d_x} \end{pmatrix} + \Sigma_t^w \right)^{-1} \right)_{:d_z} \\
&= \Sigma_t^{z\mu_x} (\Sigma_t^{\mu_x} + \Sigma_t^w)^{-1} \\
&= \Sigma_t^{z\mu_x} (\Sigma_t^x)^{-1},
\end{aligned} \tag{A.5}$$

where $\Sigma_t^{\mu_x}$ and $\Sigma_t^{z\mu_x}$ are sample variance of $\{\boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{z}_t^{(i)})\}$ and sample covariance between $\{\boldsymbol{z}_t^{(i)}\}$ and $\{\boldsymbol{g}_{\boldsymbol{\theta}}(\boldsymbol{z}_t^{(i)})\}$, respectively.

A.2 Proof

Theorem A.1. *A statistics*

$$\hat{p}_N(\boldsymbol{x}_{1:T}) = \frac{1}{N} \sum_{i=1}^{N} \prod_{t=1}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z}_t^{(i)} | \boldsymbol{z}_{1:t-1}^{(i)}) p_{\boldsymbol{\theta}}(\boldsymbol{x}_t | \boldsymbol{z}_t^{(i)})}{q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t^{(i)} | \boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:t-1}^{f,(i)})} \tag{A.6}$$

is an approximately unbiased estimator of the marginal likelihood $p(\boldsymbol{x}_{1:T})$. If the emission distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t)$ and variational distribution $q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}^f, \boldsymbol{x}_{1:T})$ are linear Gaussian, the $\hat{p}_N(\boldsymbol{x}_{1:T})$ is an unbiased estimator of the marginal likelihood.

Proof. EnKO applies the EnKF to the probabilistic time-series model (PTSM)

$$Q_{\boldsymbol{\theta},\boldsymbol{\varphi}}(\boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:T}) = q_{\boldsymbol{\varphi}}(\boldsymbol{z}_1) \prod_{t=2}^{T} q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t | \boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:T}) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_t | \boldsymbol{z}_t). \tag{A.7}$$

With slightly abuse of notation, we denote the predictive distribution at time t by $Q(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1})$ and the filtered distribution by $Q(\boldsymbol{z}_t|\boldsymbol{x}_{1:t})$ in this PTSM. Since the particle $\boldsymbol{z}_t^{f,(i)}$ can be approximated as sampling from the filtered distribution $Q(\boldsymbol{z}_t|\boldsymbol{x}_{1:t})$, the particle $\boldsymbol{z}_t^{(i)}$ can be considered as sampling from predictive distribution $Q(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1})$. The expectation of the estimator is

$$\begin{aligned}
&\mathbb{E}_{Q_{\mathrm{EnKO}}(\boldsymbol{z}_{1:T}^{(1:N)}, \boldsymbol{x}_{1:T}^{(1:N)}|X)}[\hat{p}_N(\boldsymbol{x}_{1:T})] \\
&= \frac{1}{N} \sum_{i=1}^{N} \int \prod_{t=1}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z}_t^{(i)} | \boldsymbol{z}_{1:t-1}^{(i)}) p_{\boldsymbol{\theta}}(\boldsymbol{x}_t | \boldsymbol{z}_t^{(i)})}{q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t^{(i)} | \boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:t-1}^{f,(i)})} \prod_{j=1}^{N} q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t^{(j)} | \boldsymbol{x}_{1:T}, \boldsymbol{z}_{1:t-1}^{f,(j)}) \mathrm{d}\boldsymbol{z}_{1:T}^{(1:N)} \\
&\simeq \frac{1}{N} \sum_{i=1}^{N} \int \prod_{t=1}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z}_t^{(i)} | \boldsymbol{z}_{1:t-1}^{(i)}) p_{\boldsymbol{\theta}}(\boldsymbol{x}_t | \boldsymbol{z}_t^{(i)})}{Q_{\boldsymbol{\theta},\boldsymbol{\varphi}}(\boldsymbol{z}_t^{(i)} | \boldsymbol{x}_{1:t-1})} \prod_{j=1}^{N} Q_{\boldsymbol{\theta},\boldsymbol{\varphi}}(\boldsymbol{z}_t^{(j)} | \boldsymbol{x}_{1:t-1}) \mathrm{d}\boldsymbol{z}_{1:T}^{(1:N)} \\
&= \frac{1}{N} \sum_{i=1}^{N} \int \prod_{t=1}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{z}_t^{(i)} | \boldsymbol{z}_{1:t-1}^{(i)}) p_{\boldsymbol{\theta}}(\boldsymbol{x}_t | \boldsymbol{z}_t^{(i)})}{Q_{\boldsymbol{\theta},\boldsymbol{\varphi}}(\boldsymbol{z}_t^{(i)} | \boldsymbol{x}_{1:t-1})} Q_{\boldsymbol{\theta},\boldsymbol{\varphi}}(\boldsymbol{z}_t^{(i)} | \boldsymbol{x}_{1:t-1}) \mathrm{d}\boldsymbol{z}_{1:T}^{(i)} \\
&= \frac{1}{N} \sum_{i=1}^{N} \int \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_t^{(i)} | \boldsymbol{z}_{1:t-1}^{(i)}) p_{\boldsymbol{\theta}}(\boldsymbol{x}_t | \boldsymbol{z}_t^{(i)}) \mathrm{d}\boldsymbol{z}_{1:T}^{(i)} \\
&= p(\boldsymbol{x}_{1:T}).
\end{aligned}$$

It is noteworthy that in the second line, each particle $\boldsymbol{z}_t^{(i)}$ depends on the other particles through $\boldsymbol{z}_t^{f,(i)}$, while in the third line, the dependence between particles is broken by approximating the predictive distribution $Q(\boldsymbol{z}_t|\boldsymbol{x}_{1:t-1})$.

The approximation equation in the second line is an equality if $p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t)$ and $q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t|\boldsymbol{z}_{1:t-1}, \boldsymbol{x}_{1:T})$ are linear Gaussian due to the general fact of EnKF, and $\hat{p}_N(\boldsymbol{x}_{1:T})$ is an unbiased estimator of the marginal likelihood $p(\boldsymbol{x}_{1:T})$. □

Corollary A.2. *An objective function*

$$\mathcal{L}^N_{\mathrm{EnKO}}(\boldsymbol{\theta}, \boldsymbol{\varphi}, X) = \mathbb{E}_{Q_{\mathrm{EnKO}}(\boldsymbol{z}^{(1:N)}_{1:T}, \boldsymbol{x}^{(1:N)}_{1:T}|X)}[\log \hat{p}_N(\boldsymbol{x}_{1:T})] \tag{A.8}$$

is an approximately lower bound of the log marginal likelihood $\log p(\boldsymbol{x}_{1:T})$*. If the emission distribution* $p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t)$ *and the variational distribution* $q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t|\boldsymbol{x}_{1:T}, \boldsymbol{z}^f_{1:t-1})$ *are linear Gaussian, the objective function is an unbiased estimator of the log marginal likelihood.*

Proof. This is easily proved by Jensen's inequality and Theorem 1, i.e.,

$$\begin{aligned}
\mathcal{L}^N_{\mathrm{EnKO}}(\boldsymbol{\theta}, \boldsymbol{\varphi}, X) &= \mathbb{E}_{Q_{\mathrm{EnKO}}(\boldsymbol{z}^{(1:N)}_{1:T}, \boldsymbol{x}^{(1:N)}_{1:T}|X)}[\log \hat{p}_N(\boldsymbol{x}_{1:T})] \\
&\leq \log \mathbb{E}_{Q_{\mathrm{EnKO}}(\boldsymbol{z}^{(1:N)}_{1:T}, \boldsymbol{x}^{(1:N)}_{1:T}|X)}[\hat{p}_N(\boldsymbol{x}_{1:T})] \\
&\simeq \log p(\boldsymbol{x}_{1:T}).
\end{aligned}$$

□

A.3 Gradient Estimator

The gradient of the objective function is

$$\begin{aligned}
&\nabla_{\boldsymbol{\theta},\boldsymbol{\varphi}} \mathcal{L}_{\mathrm{EnKO}}(\boldsymbol{\theta}, \boldsymbol{\varphi}, X) \\
&\quad = \nabla_{\boldsymbol{\theta},\boldsymbol{\varphi}} \int Q_{\mathrm{EnKO}}(\boldsymbol{z}^{(1:N)}_{1:T}, \boldsymbol{x}^{(1:N)}_{1:T-1}) \log Z_{\mathrm{EnKO}}(\boldsymbol{z}^{(1:N)}_{1:T}, \boldsymbol{x}^{(1:N)}_{1:T-1}) \, \mathrm{d}\boldsymbol{z}^{(1:N)}_{1:T} \mathrm{d}\boldsymbol{x}^{(1:N)}_{1:T-1} \\
&\quad = \nabla_{\boldsymbol{\theta},\boldsymbol{\varphi}} \int \left(\prod_{i=1}^N q_{\boldsymbol{\varphi}}(\boldsymbol{z}^{(i)}_1|\boldsymbol{x}_{1:T}) \right) \cdot \\
&\qquad \left(\prod_{t=2}^T \prod_{i=1}^N q_{\boldsymbol{\varphi}}(\boldsymbol{z}^{(i)}_t|\boldsymbol{x}_{1:T}, \mathrm{EnKF}_{\boldsymbol{\theta}}(\boldsymbol{z}^{(i)}_{t-1}, \boldsymbol{z}^{(1:N)}_{t-1}, \boldsymbol{x}^{(i)}_{t-1}, \boldsymbol{x}^{(1:N)}_{t-1}, \boldsymbol{x}_t)) p_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}_{t-1}|\boldsymbol{z}^{(i)}_{t-1}) \right) \cdot \\
&\qquad \log Z_{\mathrm{EnKO}}(\boldsymbol{z}^{(1:N)}_{1:T}, \boldsymbol{x}^{(1:N)}_{1:T}) \mathrm{d}\boldsymbol{z}^{(1:N)}_{1:T} \mathrm{d}\boldsymbol{x}^{(1:N)}_{1:T-1},
\end{aligned}$$

where $Z_{\mathrm{EnKO}}(\boldsymbol{z}^{(1:N)}_{1:T}, \boldsymbol{x}^{(1:N)}_{1:T-1}) := \hat{p}_N(\boldsymbol{x}_{1:T})$ and $\mathrm{EnKF}_{\boldsymbol{\theta}}(\boldsymbol{z}^{(i)}_{t-1}, \boldsymbol{z}^{(1:N)}_{t-1}, \boldsymbol{x}^{(i)}_{t-1}, \boldsymbol{x}^{(1:N)}_{t-1}, \boldsymbol{x}_t)$ denote the deterministic process of the EnKF as described in Algorithm 1. When reparametrized sampling is applied, including the filtering process, we obtain

$$\begin{aligned}
&\nabla_{\boldsymbol{\theta},\boldsymbol{\varphi}} \mathcal{L}_{\mathrm{EnKO}}(\boldsymbol{\theta}, \boldsymbol{\varphi}, X) \\
&\quad = \int \left(\prod_{t=1}^T \prod_{i=1}^N s^q_t(\boldsymbol{\varepsilon}^{q,(i)}_t) \right) \left(\prod_{t=1}^{T-1} \prod_{i=1}^N s^g_t(\boldsymbol{\varepsilon}^{g,(i)}_t) \right) \cdot \\
&\qquad \nabla_{\boldsymbol{\theta},\boldsymbol{\varphi}} \log Z_{\mathrm{EnKO}}(r^q(\boldsymbol{\varepsilon}^{q,(1:N)}_{1:T}), r^g(\boldsymbol{\varepsilon}^{g,(1:N)}_{1:T-1})) \mathrm{d}\boldsymbol{\varepsilon}^{q,(1:N)}_{1:T} \mathrm{d}\boldsymbol{\varepsilon}^{g,(1:N)}_{1:T-1} \\
&\quad = \mathbb{E}_{S_{\mathrm{EnKO}}(\boldsymbol{\varepsilon}^{q,(1:N)}_{1:T}, \boldsymbol{\varepsilon}^{g,(1:N)}_{1:T-1})} \left[\nabla_{\boldsymbol{\theta},\boldsymbol{\varphi}} \log Z_{\mathrm{EnKO}}(r^q(\boldsymbol{\varepsilon}^{q,(1:N)}_{1:T}), r^g(\boldsymbol{\varepsilon}^{g,(1:N)}_{1:T-1})) \right],
\end{aligned}$$

where $\boldsymbol{\varepsilon}^{q,(i)}_t$ and $\boldsymbol{\varepsilon}^{g,(i)}_t$ follows the distribution $s^q_t(\boldsymbol{\varepsilon}^{q,(i)}_t)$ and $s^g_t(\boldsymbol{\varepsilon}^{g,(i)}_t)$, respectively, and r^q and r^g represent the reparametrization transform from $\boldsymbol{\varepsilon}^{q,(1:N)}_{1:T}$ to $\boldsymbol{z}^{(1:N)}_{1:T}$ and from $\boldsymbol{\varepsilon}^{g,(1:N)}_{1:T}$ to $\boldsymbol{x}^{(1:N)}_{1:T}$, respectively.

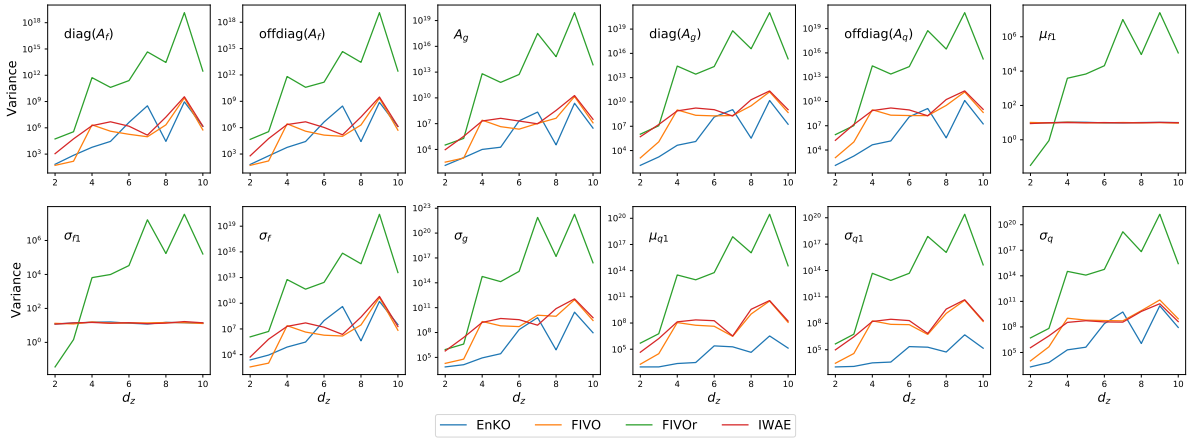


Figure A.1: Variance of the gradient estimates of EnKO, FIVO, FIVOr, and IWAE versus latent dimension d_z, where FIVOr means FIVO with resampling gradient term. For a vector parameter $\boldsymbol{v}$, the variance is computed for each element and averaged over the elements. For a matrix parameter A, diag(A) and offdiag(A) represent the average of the variance of the diagonal elements and the off-diagonal elements, respectively.

A.4 Experiment for Variance of Gradient Estimators

The low variance of the gradient estimator leads to stable learning. In this section, we compute the variance of the gradient estimators of IWAE, FIVO, and EnKO for two toy examples and compare the results among the methods. We set $T = 100$, a number of particles to 16, batch size to 10, number of simulations to 100, and $d_x, d_z \in \{2, 3, \cdots, 10\}$.

A.4.1 Linear Gaussian State Space Model

The first example uses the following variational distribution and SSM

$$\boldsymbol{z}_1 \sim \mathcal{N}(\boldsymbol{\mu}_{q1}, \boldsymbol{\sigma}_{q1}^2), \tag{A.9a}$$

$$\boldsymbol{z}_t \sim q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) = \mathcal{N}(A_q \boldsymbol{z}_{t-1}, \boldsymbol{\sigma}_q^2), \tag{A.9b}$$

$$p_{\boldsymbol{\theta}}(Z, X) = p_{\boldsymbol{\theta}}(\boldsymbol{z}_1) \prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t), \tag{A.9c}$$

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_1) = \mathcal{N}(\boldsymbol{\mu}_{f1}, \boldsymbol{\sigma}_{f1}^2), \tag{A.9d}$$

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) = \mathcal{N}(A_f \boldsymbol{z}_{t-1}, \boldsymbol{\sigma}_f^2), \tag{A.9e}$$

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t) = \mathcal{N}(A_g \boldsymbol{z}_t, \boldsymbol{\sigma}_g^2), \tag{A.9f}$$

where $\boldsymbol{\theta} = \{\boldsymbol{\mu}_{f1}, A_f, A_g, \boldsymbol{\sigma}_{f1}, \boldsymbol{\sigma}_f, \boldsymbol{\sigma}_g\}$ and $\boldsymbol{\varphi} = \{\boldsymbol{\mu}_{q1}, A_q, \boldsymbol{\sigma}_{q1}, \boldsymbol{\sigma}_q\}$. We evaluated the gradient estimators at $\boldsymbol{\mu}_{f1} = \boldsymbol{\mu}_{q1} = \mathbf{0}$, $\boldsymbol{\sigma}_{q1} = \boldsymbol{\sigma}_{f1} = 0.1 \times \mathbf{1}$, $\boldsymbol{\sigma}_q = \boldsymbol{\sigma}_f = \boldsymbol{\sigma}_g = 0.01 \times \mathbf{1}$, $A_q = I_{d_z} + U_q$, $A_f = I_{d_z} + U_f$, $(U_q)_{ij}, (U_f)_{ij} \sim U(-0.05, 0.05)$, and $(A_g)_{ij} \sim U(-0.5, 0.5)$.

Figure A.1 and Figure A.2 show variance of the gradient estimates of EnKO, FIVO, FIVOr, and IWAE versus latent dimension d_z and observation dimension d_x, respectively, where FIVOr means FIVO with resampling gradient term. Figure A.3 shows log relative variance of EnKO to FIVO $\log\{V_{\text{EnKO}}[\nabla\mathcal{L}]/V_{\text{FIVO}}[\nabla\mathcal{L}]\}$. For a vector parameter $\boldsymbol{v} \in \mathbb{R}^d$, the variance is computed for each element and then averaged over the elements

$$V[\boldsymbol{v}] = \frac{1}{d} \sum_{i=1}^{d} V\left[\frac{\partial \mathcal{L}}{\partial v_i}\right]. \tag{A.10}$$

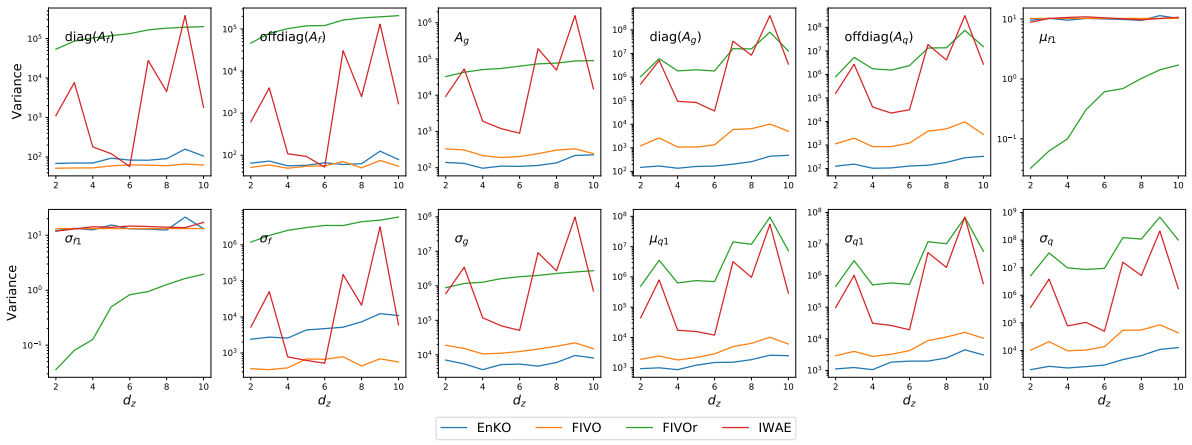


Figure A.2: Variance of the gradient estimates of EnKO, FIVO, FIVOr, and IWAE versus observation dimension d_x, where FIVOr means FIVO with resampling gradient term

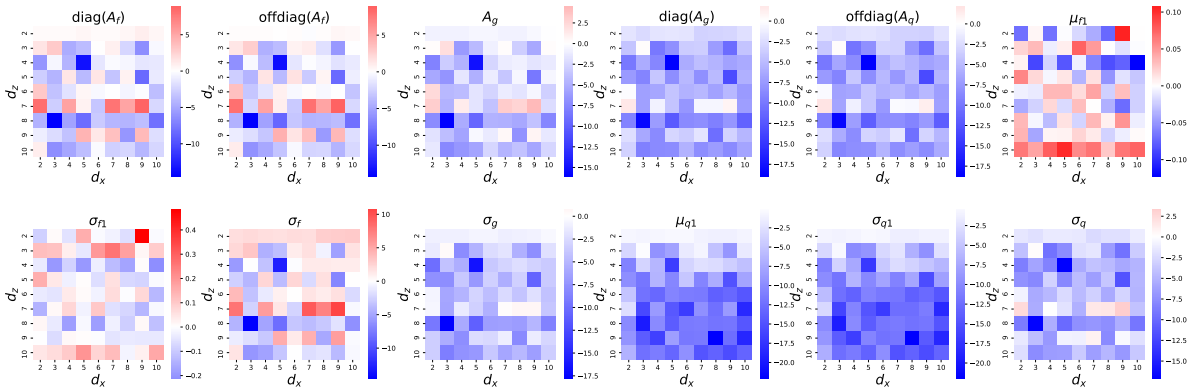


Figure A.3: Log relative variance of EnKO to FIVO $\log\{V_{\text{EnKO}}[\nabla\mathcal{L}]/V_{\text{FIVO}}[\nabla\mathcal{L}]\}$. Positive values (*red*) represent the variance of EnKO is greater than FIVO, and the opposite is true for negative values (*blue*)

For a square matrix parameter $A \in \mathbb{R}^{d\times d}$, a variance of diag(A) and offdiag(A) represent the average of the variance of the diagonal elements and the off-diagonal elements, respectively, i.e.,

$$V[\text{diag}(A)] = \frac{1}{d}\sum_{i=1}^{d} V\left[\frac{\partial\mathcal{L}}{\partial A_{ii}}\right], \tag{A.11}$$

$$V[\text{offdiag}(A)] = \frac{1}{d(d-1)}\sum_{i\neq j, i,j\in\mathbb{N}_d} V\left[\frac{\partial\mathcal{L}}{\partial A_{ij}}\right], \tag{A.12}$$

For a rectangular matrix $A \in \mathbb{R}^{d_1\times d_2}$, the variance is computed same as a vector parameter

$$V[A] = \frac{1}{d^2}\sum_{i=1}^{d_1}\sum_{j=1}^{d_2} V\left[\frac{\partial\mathcal{L}}{\partial A_{ij}}\right]. \tag{A.13}$$

From Figure A.1 and A.2, FIVOr (FIVO with resampling gradient term) has a higher variance than the other methods, and EnKO has a relatively low variance for all parameters. In particular, the variance of FIVOr is rapidly increased when d_z is increased. When $d_x = 2$ is fixed and d_z increases, FIVO and IWAE are competitive, but when $d_z = 2$ is fixed and d_x increases, the variance of IWAE shifts more than that of

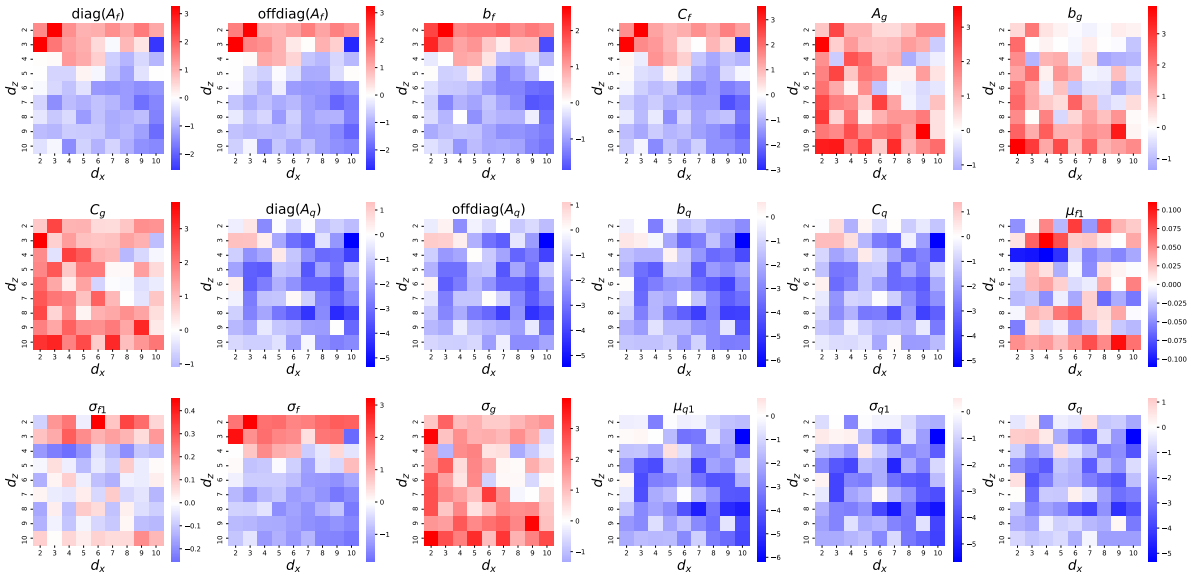


Figure A.4: Log relative variance of EnKO to FIVO $\log\{V_{\text{EnKO}}[\nabla\mathcal{L}]/V_{\text{FIVO}}[\nabla\mathcal{L}]\}$. Positive values (*red*) represent the variance of EnKO is greater than FIVO, and the opposite is true for negative values (*blue*).

FIVO. From Figure A.3, the variance of gradient estimates of EnKO is generally lower than that of FIVO, especially for parameters regarding g and q.

A.4.2 Nonlinear Non-Gaussian State Space Model

The second example uses the following variational distribution and SSM

$$\boldsymbol{z}_1 \sim \mathcal{N}(\boldsymbol{\mu}_{q1}, \boldsymbol{\sigma}_{q1}^2), \tag{A.14a}$$

$$\boldsymbol{z}_t \sim q_{\boldsymbol{\varphi}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) = \mathcal{N}(\tanh(C_q \text{flatten}(\boldsymbol{z}_{t-1}\boldsymbol{z}_{t-1}^T) + A_q\boldsymbol{z}_{t-1} + \boldsymbol{b}_q), \boldsymbol{\sigma}_q^2), \tag{A.14b}$$

$$p_{\boldsymbol{\theta}}(Z, X) = p_{\boldsymbol{\theta}}(\boldsymbol{z}_1)\prod_{t=2}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1})\prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t), \tag{A.14c}$$

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_1) = \mathcal{N}(\boldsymbol{\mu}_{f1}, \boldsymbol{\sigma}_{f1}^2), \tag{A.14d}$$

$$p_{\boldsymbol{\theta}}(\boldsymbol{z}_t|\boldsymbol{z}_{t-1}) = \text{Student}_5(\tanh(C_f \text{flatten}(\boldsymbol{z}_{t-1}\boldsymbol{z}_{t-1}^T) + A_f\boldsymbol{z}_{t-1} + \boldsymbol{b}_f), \boldsymbol{\sigma}_f), \tag{A.14e}$$

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\boldsymbol{z}_t) = \text{Student}_5(\tanh(C_g \text{flatten}(\boldsymbol{z}_t\boldsymbol{z}_t^T) + A_g\boldsymbol{z}_t + \boldsymbol{b}_g), \boldsymbol{\sigma}_g), \tag{A.14f}$$

where $\theta = \{\boldsymbol{\mu}_{f1}, \sigma_{f1}, C_f, A_f, \boldsymbol{b}_f, \boldsymbol{\sigma}_f, C_g, A_g, \boldsymbol{b}_g, \boldsymbol{\sigma}_g\}$, $\varphi = \{\boldsymbol{\mu}_{q1}, \boldsymbol{\sigma}_{q1}, C_q, A_q, \boldsymbol{b}_q, \boldsymbol{\sigma}_q\}$. $\text{Student}_\nu(\rho, \sigma)$ represent student-t distribution with degree of freedom ν, location parameter ρ, and scale parameter σ. For a matrix $A \in \mathbb{R}^{d\times d}$, $\text{flatten}(A) = (a_{11}, a_{12}, \cdots, a_{dd})^T \in \mathbb{R}^{d^2}$.

Figure A.4 shows log relative variance of EnKO to FIVO $\log\{V_{\text{EnKO}}[\nabla\mathcal{L}]/V_{\text{FIVO}}[\nabla\mathcal{L}]\}$. EnKO's gradient estimates for the parameters regarding f and q have lower variance than FIVO, but the opposite is true for g. This indicates that while EnKO can stabilize learning for variational and transition parameters, FIVO has the ability for generative parameters. Since learning transition parameters are often unstable in time-series models, EnKO, which can stabilize the learning transition parameters, is a more effective method.

A.5 Experiment Details

Figure A.5 shows the network architecture of SVO [260] as coded in the authors' GitHub page `https://github.com/amoretti86/PSVO`. Although they trained SVO with the FIVO system, we expanded this

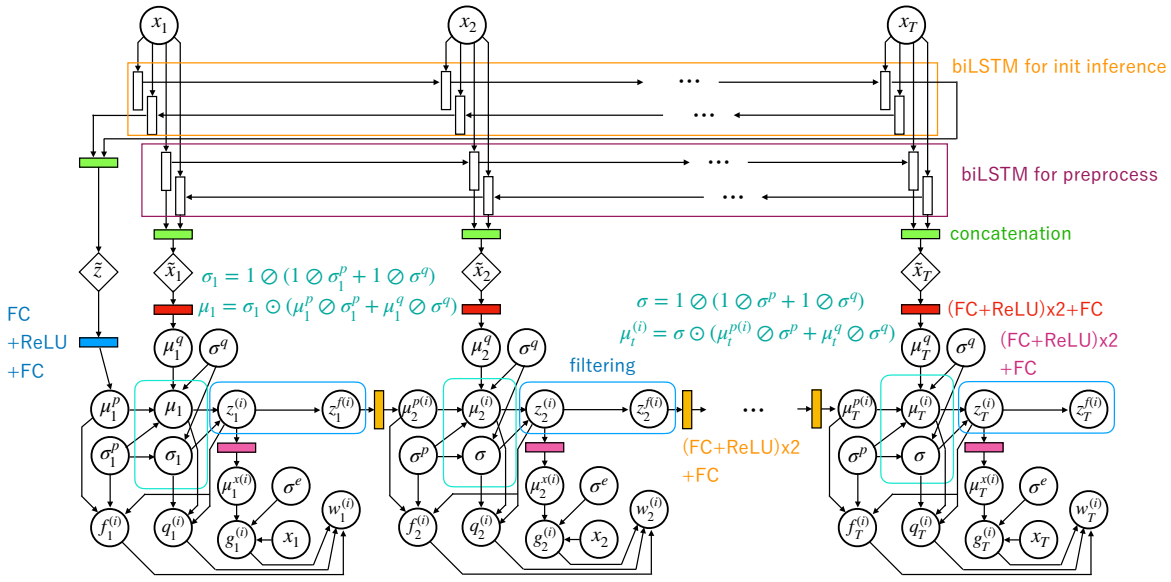


Figure A.5: The network architecture of SVO. FC and biLSTM stand for fully connected layer and bidirectional LSTM, respectively. A filtering step (*blue*) is different by ensemble systems

Table A.1: Chosen hyperparamters and configurations

Parameters	Dataset			
	FHN	Lorenz	Walking	RMNIST
# Neural units per layer	{16, **32**}	{16, **32**}	{50, **100**}	{25, **50**}
Latent dimension	2	3	6	2
Batch size	20	6	4	40
Training epochs	2000	2000	2000	3000
Scaling	abs. div.	abs. div.	-	-

with EnKO and IWAE systems. The chosen hyperparameters of SVO are the same as their experiments. Figure 6.5 shows the network architecture of outer VAE for rotating MNIST data set. The encoder consists of a style convolution block and a dynamics convolution block. While the style convolution block extracts style (static) information, such as the shape and edge of the handwritten data, the dynamics convolution block extracts dynamics information, such as rotation dynamics. The chosen hyperparameters of convolution layers are the same as experiments for the data set in [406]. The dimension of $\boldsymbol{a}_t$ and $\boldsymbol{s}_t$ were set to 2 and 6, respectively. Table A.1 summarizes other hyperparameters and configurations used for generating the results reported in Section 6.4. In this table, the boldfaces are the chosen hyperparameters through our experiments. The more detailed conditions and our experiments are described on our GitHub page `https://github.com/ZoneMS/EnKO`.

We used NVIDIA Tesla V100 GPUs, CUDA 10.2, and PyTorch 1.6.0 for our experiments. Each experiment for the FHN model, the Lorenz model, and the walking data set spent around one day, and each experiment for the rotating MNIST data set spent around two days.

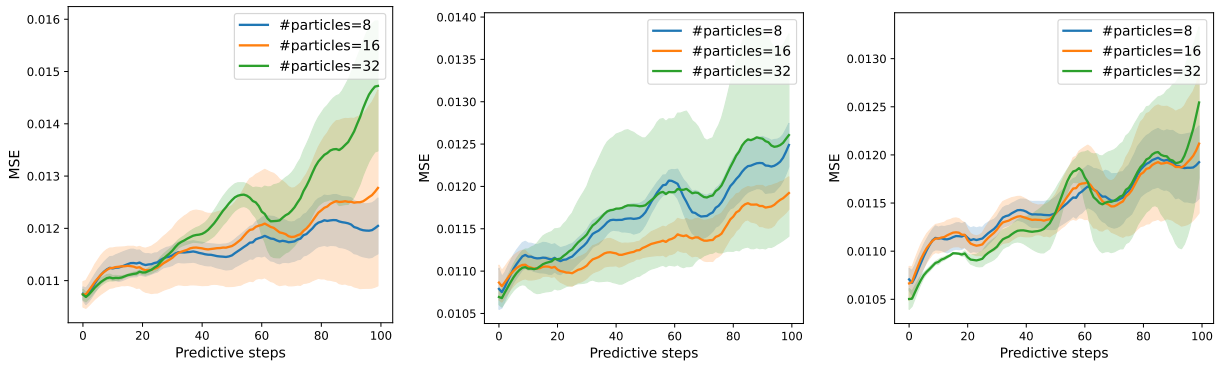


Figure A.6: MSE for synthetic Fitz-Hugh Nagumo data by EnKO without inflation methods (left) and with RTPP (center) and RTPS (right) for various ensemble sizes

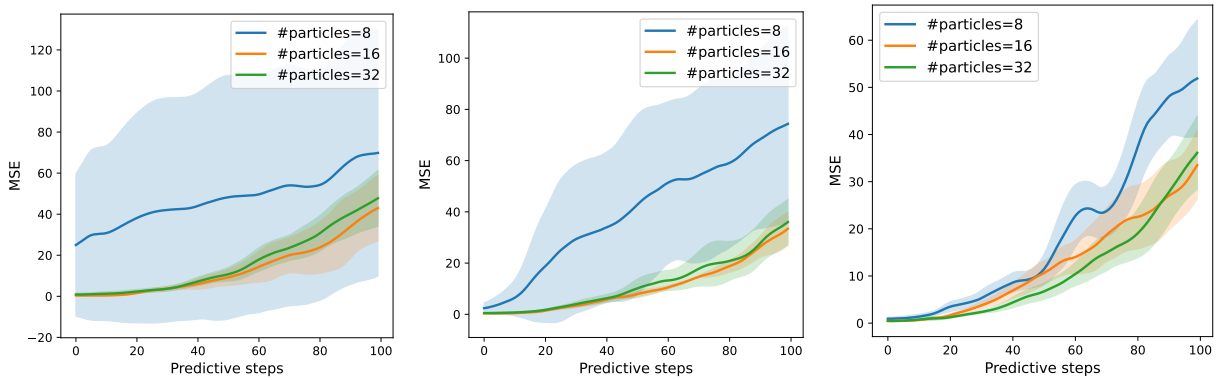


Figure A.7: MSE for synthetic Lorenz data by EnKO without inflation methods (left) and with RTPP (center) and RTPS (right) for various ensemble sizes

A.6 Additional Results

A.6.1 Ensemble Size

To verify that the number of particles changes the prediction ability, we performed experiments with varying numbers of particles. Figure A.6 and Figure A.7 show prediction MSE for Fitz-Hugh Nagumo and Lorenz data, respectively, for various ensemble sizes. An increase in the number of particles improves the prediction accuracy for the Lorenz data. In EnKO, without inflation methods for FHN data, an increase in particles hurts the prediction ability. This result is contrary to the intuition that an increase in the number of particles results in better prediction accuracy; however, it is known that an increase in the number of particles does not necessarily have a positive effect on the results [296]. The Lorenz model is complex and requires a certain number of particles to learn well, while the FHN model is relatively simple and requires only eight particles. It is a future task to calculate the intrinsic dimension of the data in advance and create a framework to appropriately determine the number of particles according to the complexity measure, such as translation error [346].

A.6.2 Inflation Factor

To evaluate the effect of the inflation factor on prediction and to enable appropriate factor selection, we conducted experiments with several factors $\{0.1, 0.2, 0.3\}$. In Section 6.4, the factors that minimize the prediction MSE for the validation data are selected, the results for the test data are shown, and the selected

Table A.2: Chosen inflation factors

Parameters	Dataset			
	FHN	Lorenz	Walking	RMNIST
RTPP	0.2	0.1	0.2	0.1
RTPS	0.1	0.1	0.2	0.3

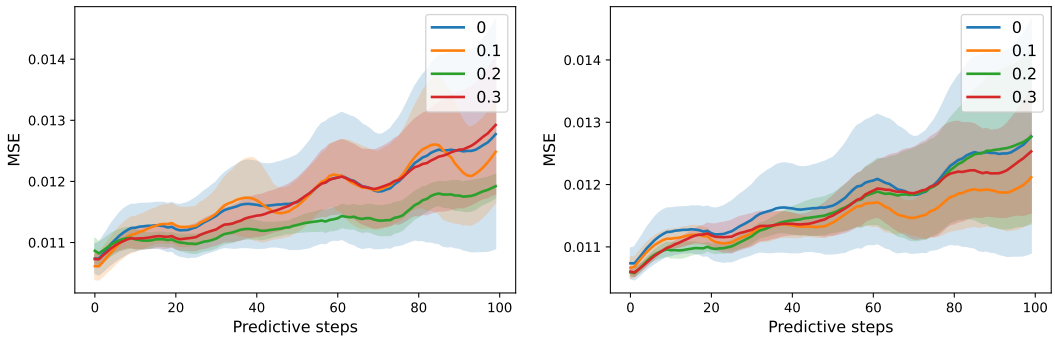


Figure A.8: Prediction MSE for synthetic Fitz-Hugh Nagumo data by EnKO with RTPP (left) and RTPS (right) for various inflation factors

factors are summarized in Table A.2. Figure A.8-A.11 show prediction MSE for the Fitz-Hugh Nagumo, the Lorenz, the walking, and the rotating MNIST data sets, respectively, by EnKO with inflation methods for various inflation factors. For the FHN and Lorenz data, since the errors when the inflation factor is 0.3 are higher, the appropriate factor to improve the inference accuracy is selected from 0.1 to 0.2. For the walking data, a factor of 0.3 further improves the prediction accuracy of the RTPP, indicating that a relatively large factor is appropriate for inference. A small ratio of particles to the observation dimension tends to underestimate the state covariance, as mentioned in Section 4.2.5. Since the ratios were high for FHN and Lorenz, underestimation was not apparent, and a small inflation factor was preferred. In the walking data, the ratio was small, so a large inflation factor was preferred to suppress underestimation. For the rotating MNIST data, the inflation methods did not improve the prediction accuracy, and a factor between 0 and 0.1 is considered appropriate. Since the observations in SVAE correspond to the dimensions of the auxiliary variables, the ratio was kept low, and underestimation did not occur.

If the number of particles is predetermined due to memory or other reasons, the inflation factor can be determined according to the observed or auxiliary dimension. Specifically, we propose that the factor should be selected from 0.01 to 0.2 for low-dimensional systems of 10 or less, 0.2 to 0.3 for medium-dimensional systems of 10 to 100, and according to the dimension of the space of auxiliary variables obtained using outer VAE for high-dimensional systems of 100 or more.

A.6.3 CMU Walking Data

Figure A.12 shows long prediction results for EnKO with RTPS. We inferred the initial latent state and forecasted the values of the observations at all remaining time points according to the learned generative model. Overall, the observation points are within the prediction interval of the mean plus or minus standard deviation, and the proposed method provides excellent predictions over a long period. In particular, it is surprising that the proposed method predicts well even for variables with significant noise effects, such as the neck and the thorax. On the other hand, the proposed method overestimates the prediction interval for the clavicles, which take almost zero values.

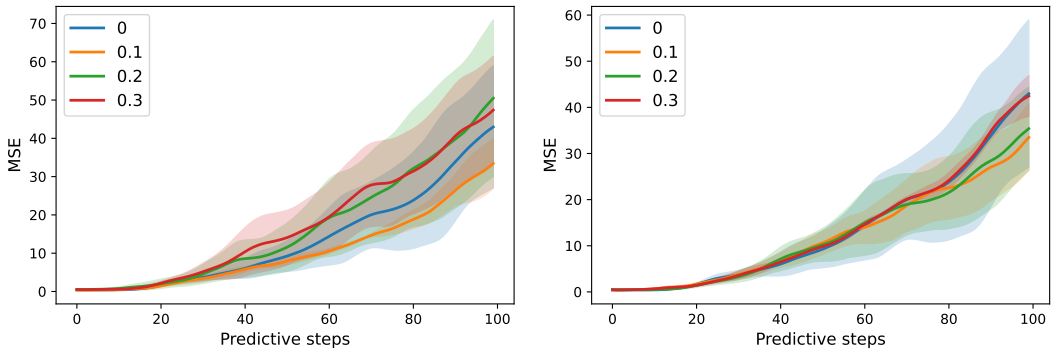


Figure A.9: Prediction MSE for synthetic Lorenz data by EnKO with RTPP (left) and RTPS (right) for various inflation factors

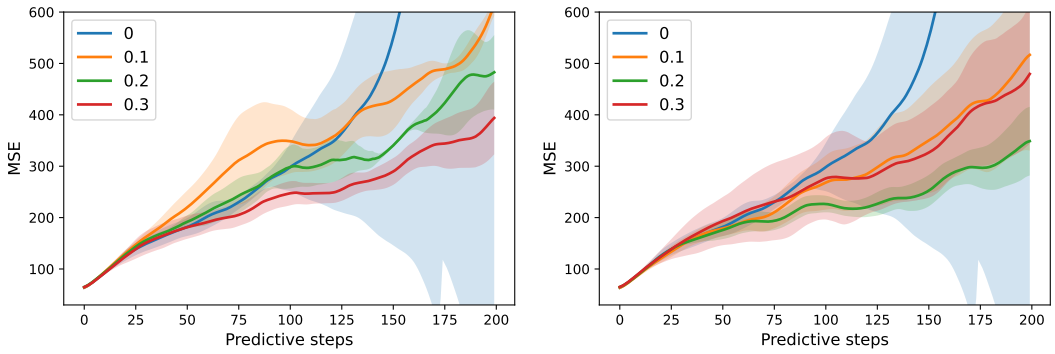


Figure A.10: Prediction MSE for the walking data set by EnKO with RTPP (left) and RTPS (right) for various inflation factors

A.6.4 Rotating MNIST Dataset

Figure A.13 shows true images and prediction results for FIVO, IWAE, and EnKO. We inferred the initial latent state and forecasted the observations at all remaining time points according to the network. It can be seen qualitatively that all the methods produce images close to the observations. However, when we look at the prediction MSE (Figure 6(c)), EnKO outperforms IWAE and FIVO, indicating that EnKO can learn a more appropriate model at a difficult level to distinguish qualitatively.

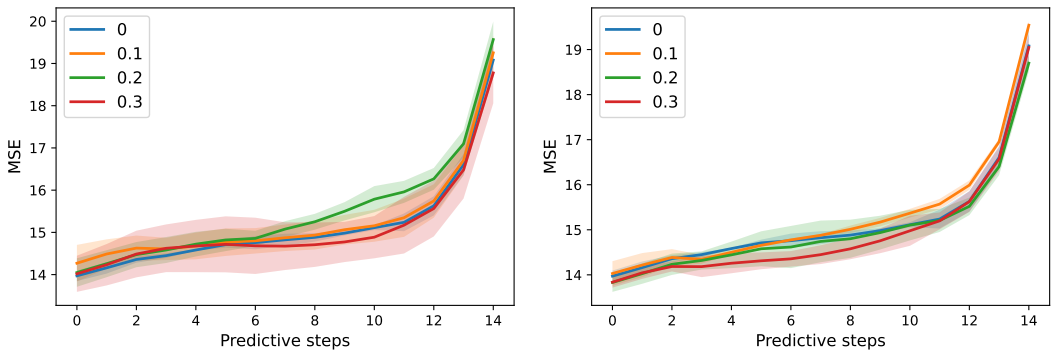


Figure A.11: Prediction MSE for the rotating MNIST data set by EnKO with RTPP (left) and RTPS (right) for various inflation factors

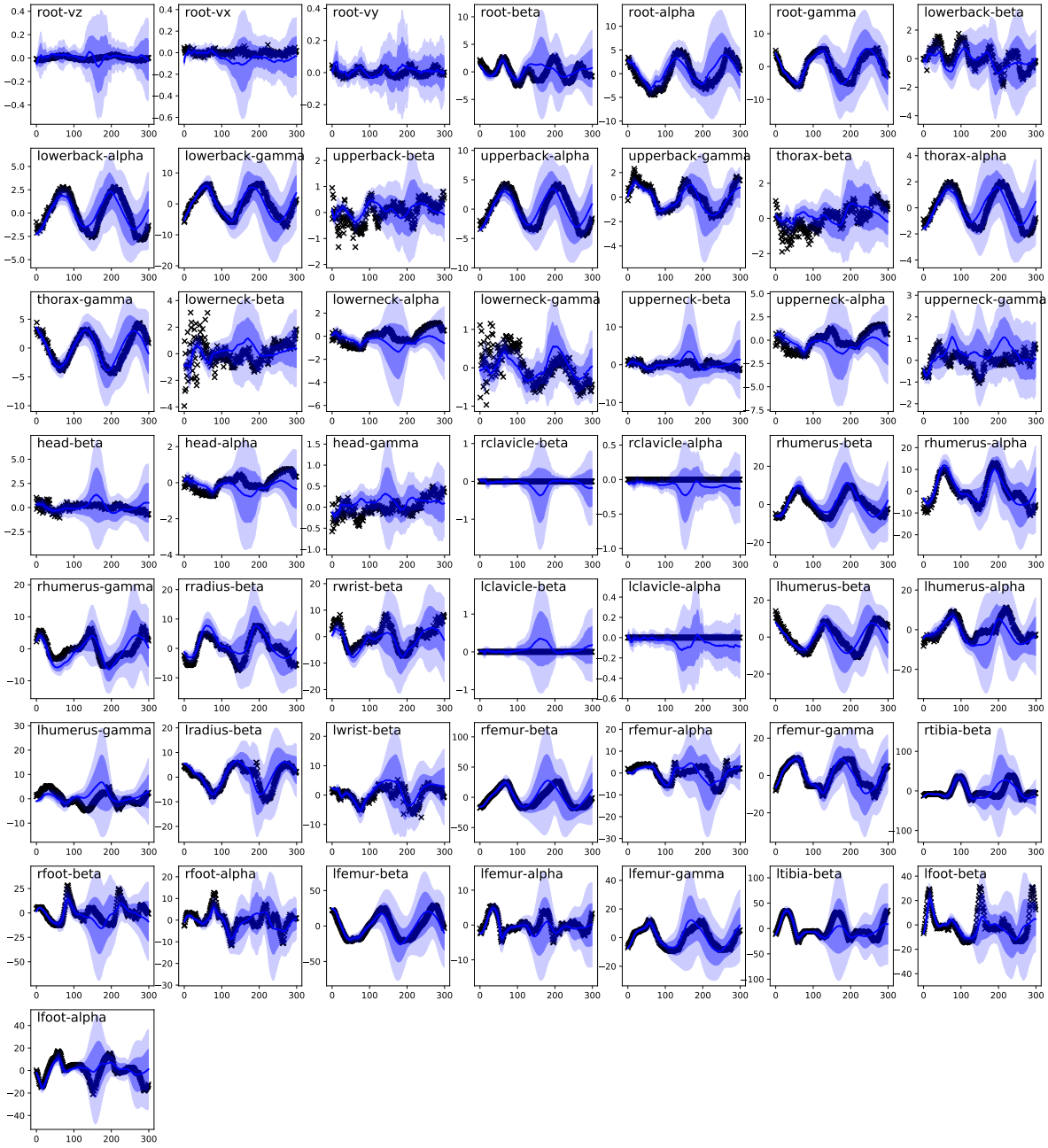


Figure A.12: Long prediction results for EnKO with RTPS. We inferred the initial latent state and predicted the values of the observations at all remaining time points according to the learned generative model. The black times represent the observed points, the solid blue line represents the predicted mean, and the dark and light blue widths represent the predicted mean plus or minus standard deviation and two standard deviations, respectively. The text in the figure shows the variable names. The vz, vx, and vy correspond to velocities, alpha, beta, and gamma correspond to Euler angles, and l and r correspond to left and right

Figure A.13: True images and prediction results for rotating MNIST data set

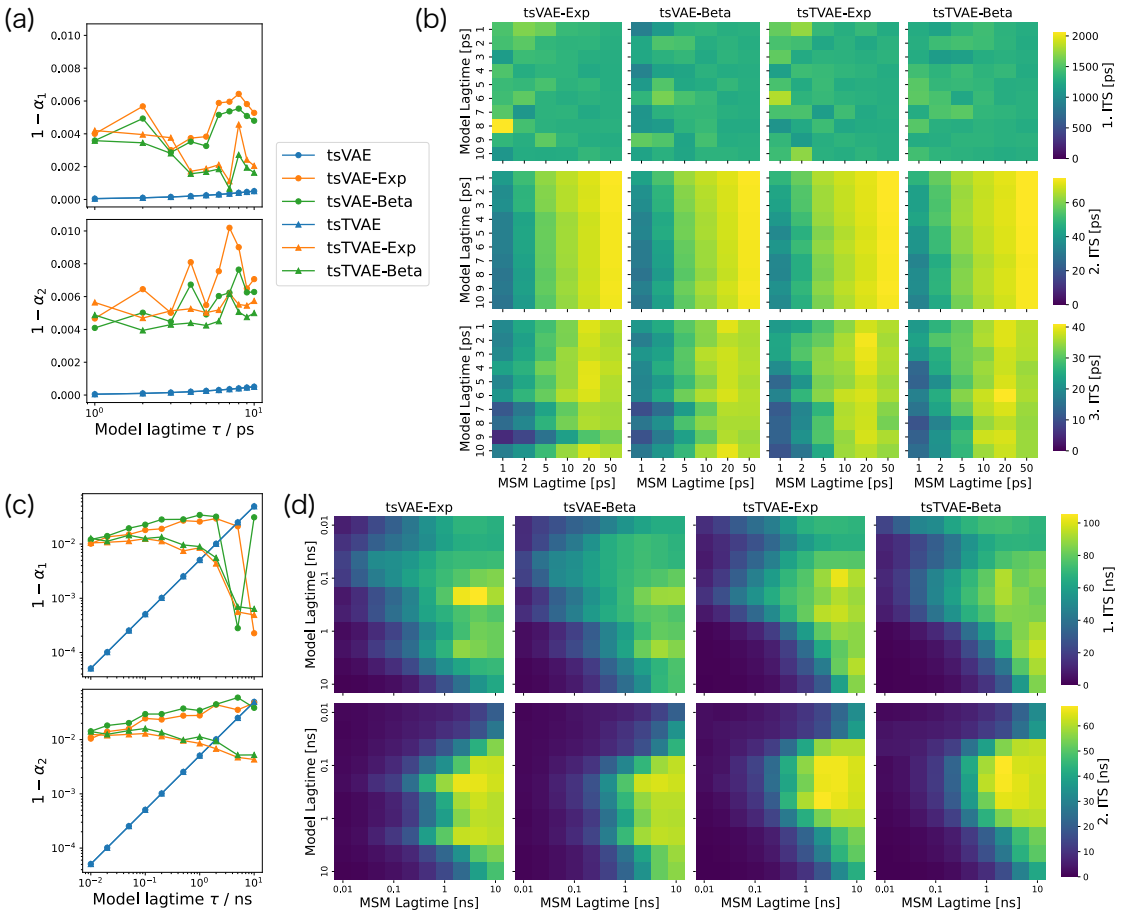


Figure B.1: **Negative autocorrelations and implied timescales (ITS) of MSMs constructed in the latent space**. (a) The negative model autocorrelation $1-\alpha$ versus a model lagtime τ for alanine-dipeptide trajectories. (b) Heatmaps of the first three ITS against the model lagtime τ and the MSM lagtime τ_m for alanine-dipeptide trajectories. (c) The negative model autocorrelation $\boldsymbol{\epsilon} = 1 - \boldsymbol{\alpha}$ versus a model lagtime τ for chignolin trajectories. (d) Heatmaps of the first three ITS against the model lagtime τ and the MSM lagtime τ_m for chignolin trajectories.

B Supplementary Information for tsVAE

B.1 Learning Model Autocorrelation

While we used the fixed hyperparameter α adjusted from the fixed decaying time τ_d in our main experiments, we propose another setting called learned mode. The learned mode of the model means the hyperparameter α is learned together with the neural network parameters. This mode assumes a prior distribution such as the Beta distribution [309, 188] $B(3,1)$. The overall time-structured prior distribution is designed by

$$p_{\boldsymbol{\theta}}(Z, \boldsymbol{\alpha}) = p(\boldsymbol{\alpha}) \prod_{t=1}^{\tau} p_{\boldsymbol{\theta}}(\boldsymbol{z}_t) \prod_{t=\tau+1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{z}_t | \boldsymbol{z}_{t-\tau}, \boldsymbol{\alpha}) \tag{B.1}$$

This Chapter is reprinted (adapted) with permission from *J. Chem. Theory Comput.* 2024, 20, 1, 436—450. `https://doi.org/10.1021/acs.jctc.3c01025`.

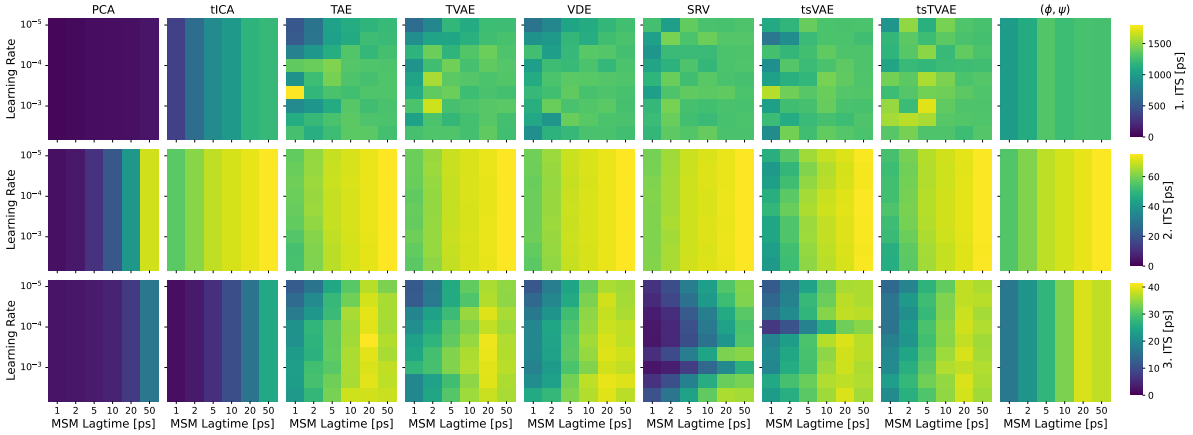


Figure B.2: Heatmaps of the first three ITS against the learning rate η and the MSM lagtime τ_m for alanine-dipeptide trajectories

$$= p(\boldsymbol{\alpha}) \prod_{t=1}^{\tau} \mathcal{N}(\boldsymbol{z}_t; \mathbf{0}, I) \prod_{t=\tau+1}^{T} \mathcal{N}(\boldsymbol{z}_t; \boldsymbol{\alpha} \odot \boldsymbol{z}_{t-\tau}, \mathrm{diag}(1 - \boldsymbol{\alpha}^{\circ 2})). \tag{B.2}$$

In the following experiment, we used the Beta distribution $\mathcal{B}](3, 1)$, and the pseudo-exponential distribution whose probability density function is defined by

$$p(\alpha) = e^{\alpha \log 3} - 1. \tag{B.3}$$

Figure B.1 shows the negative autocorrelations and implied timescales (ITS) of MSMs constructed in the latent space. Figure B.1 (a) and (c) show the negative model autocorrelation $\boldsymbol{\epsilon} = 1 - \boldsymbol{\alpha}$ for alanine-dipeptide and chignolin trajectories, respectively. For the adjusted mode, the parameters were computed using the Eq. (22) of the main article. For the learned mode, the parameters were learned together with neural network parameters. Figure B.1(a) shows that in the learning mode, the autocorrelations are comparable in scale concerning the model lagtime τ for alanine-dipeptide trajectories. Figure B.1(b) shows that the third ITS is lowered as the model lagtime τ increases in the learning and adjusted modes. Regardless of the introduction of model autocorrelation α, it is suggested that a high model lagtime τ can destroy the third slowest dynamics, which is relatively fast. Figure B.1(c) shows that the model autocorrelations learned are destabilized for large model lagtime τ for chignolin trajectories. As the model lagtime τ is increased, ITS decreases significantly, meaning that the model, including α, is not well trained (Figure B.1(d)). The robustness of ITS to the introduction of α means that the presence or absence of α is more beneficial for representation acquisition than the presence or absence of a prior distribution of α.

B.2 Robustness for Learning Rate

A learning rate is an important hyperparameter for optimizing neural network parameters. Figure B.2 and Figure B.3 show implied timescales of MSMs constructed in the latent space with each learning rate. The first three implied timescales for alanine-dipeptide trajectories slightly vary for the learning rate. The first two implied timescales of tsVAE and tsTVAE for chignolin trajectories are higher than the other methods in $\eta \in [10^{-4}, 2 \times 10^{-3}]$. This suggests that the proposed methods can more efficiently obtain slow CVs than the existing methods by exploring the learning rate scale.

B.3 How to Detect Key Variables from MSM Macro-states

Detecting key variables that contributed to slow dynamics is essential. To extract the variables, we used each variable's total variation similarities between histograms of macro-states. The total variation similarity

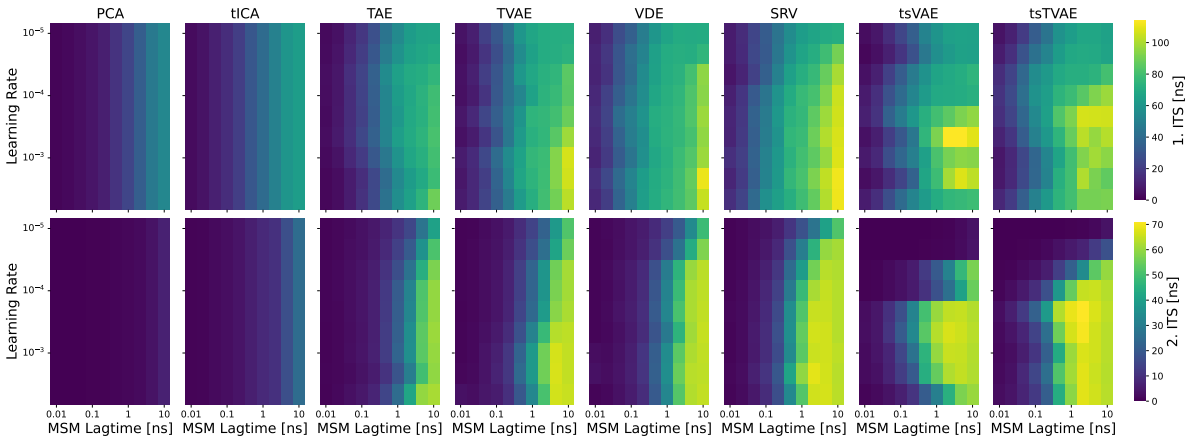


Figure B.3: Heatmaps of the first two ITS against the learning rate η and the MSM lagtime τ_m for chignolin trajectories

between two probability distributions p_1 and p_2 is formulated by

$$\mathrm{TVS}(p_1, p_2) = \int_{\mathcal{X}} \mathrm{d}x \min\{p_1(x), p_2(x)\}, \tag{B.4}$$

where $(\mathcal{X}, \mathcal{F}(\mathcal{X}))$ represents the measurable space. When $\mathcal{X}$ is a bin set of histograms, the total variation distance is computed by

$$\mathrm{TVS}(p_1, p_2) = \sum_{i=0}^{N-1} \Delta x \min\{p_1(x + i\Delta x), p_2(x + i\Delta x)\}, \tag{B.5}$$

where Δx is the bin-size and N is the the number of divisions.

Figure B.4 shows the total variation similarities between the histograms of each variable of macro-states for chignolin trajectories. Let $h_i(x)$ represents histograms of a variable x of the i-th macro-state, i.e.,

$$h_i(x) = \text{histogram of } \{x_t | S_t = s_i,\ t \in \{1, \cdots, T\}\}, \tag{B.6}$$

where S_t and s_i represents the macro-state at a time t and the i-th macro-state. We created the histograms whose bin edges are estimated by the Freedman Diaconis Estimator. We used variable space $\mathcal{X}$ as each distance between the no-adjacent pairs of the residues and each dihedral angle of the residues.

This figure shows critical variables for each transition between macro-states. The total variation similarity between the histogram of the distance $d(3, 8)$ of macro-states s_2 and s_3 is the lowest in TAE, TVAE, VDE, SRV, tsVAE, and tsTVAE. Since the distance $d(3, 8)$ is the critical variable for folding dynamics, these methods can explain the folding dynamics by the transition between s_2 and s_3. The total variation similarities of $\psi(3)$ between s_1 and s_3 of the methods excluding PCA are the lowest. The NN-based methods can construct interpretable embedding space regarding the macro-state transition from these results.

B.4 Relationships between Time-locality and Spacial-locality

We use relative locality to verify whether samples in time proximity are embedded in spatial proximity. The measure is defined by

$$\mathrm{RL}(\Delta t_1 | \Delta t_2) = \frac{1}{T - \Delta t_1} \sum_{t \in [T - \Delta t_1]} r(d(\boldsymbol{z}_t, \boldsymbol{z}_{t+\Delta t_1}); \\ \{d(\boldsymbol{z}_s, \boldsymbol{z}_{s+\Delta t_1})\}_{s \in [T-\Delta t_1]} \cup \{d(\boldsymbol{z}_s, \boldsymbol{z}_{s+\Delta t_2})\}_{s \in [T-\Delta t_2]}), \tag{B.7}$$

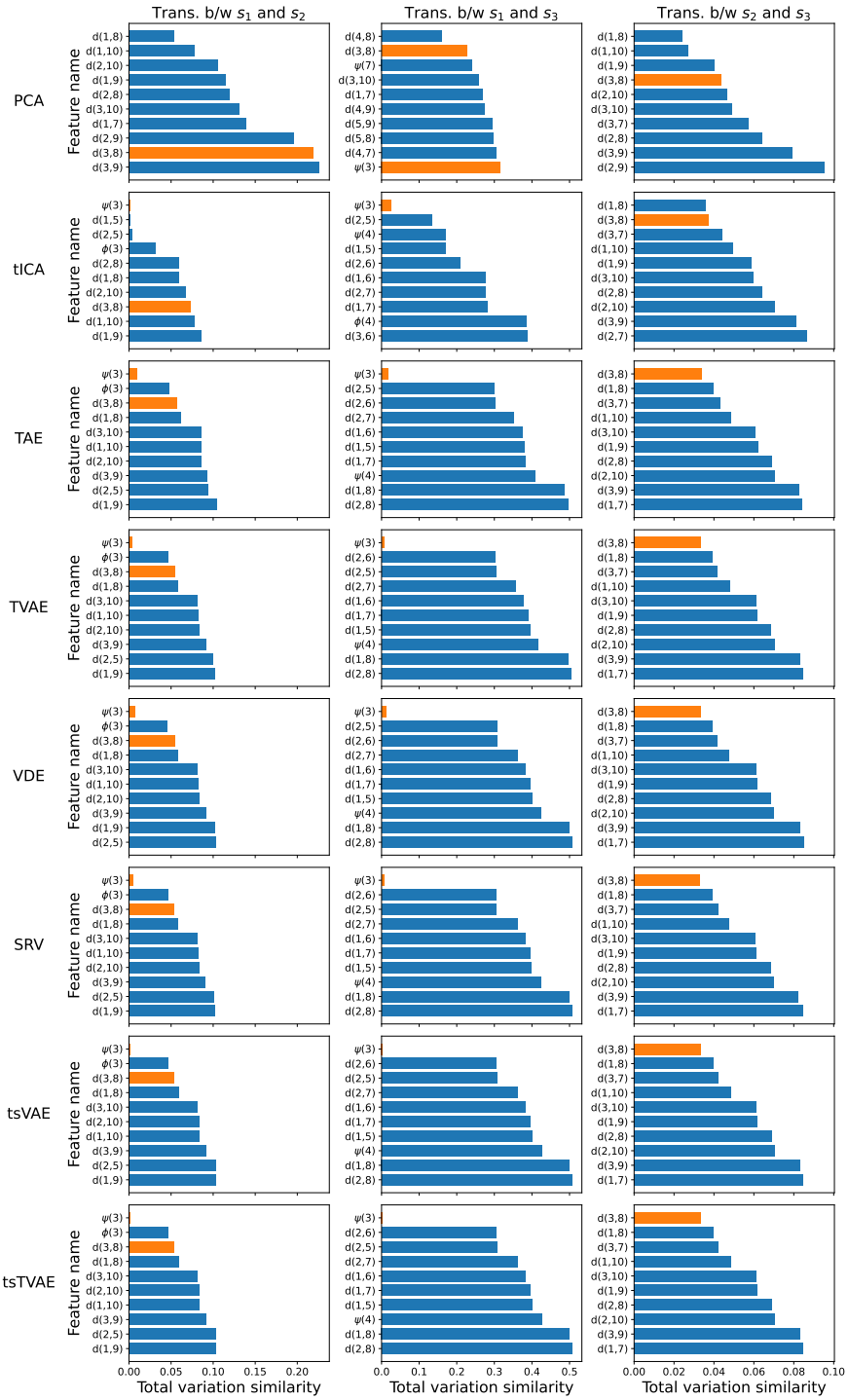


Figure B.4: Total variation similarity between the histograms of each variable of macro-states for chignolin trajectories. The variables include the distances between the no-adjacent residues and the dihedral angles of the residues. The ten smallest total variation distances for the transitions are displayed. The *orange* bar means the specific variables: the distance between ASP3N and THR8O and the dihedral angle ψ of ASP3.

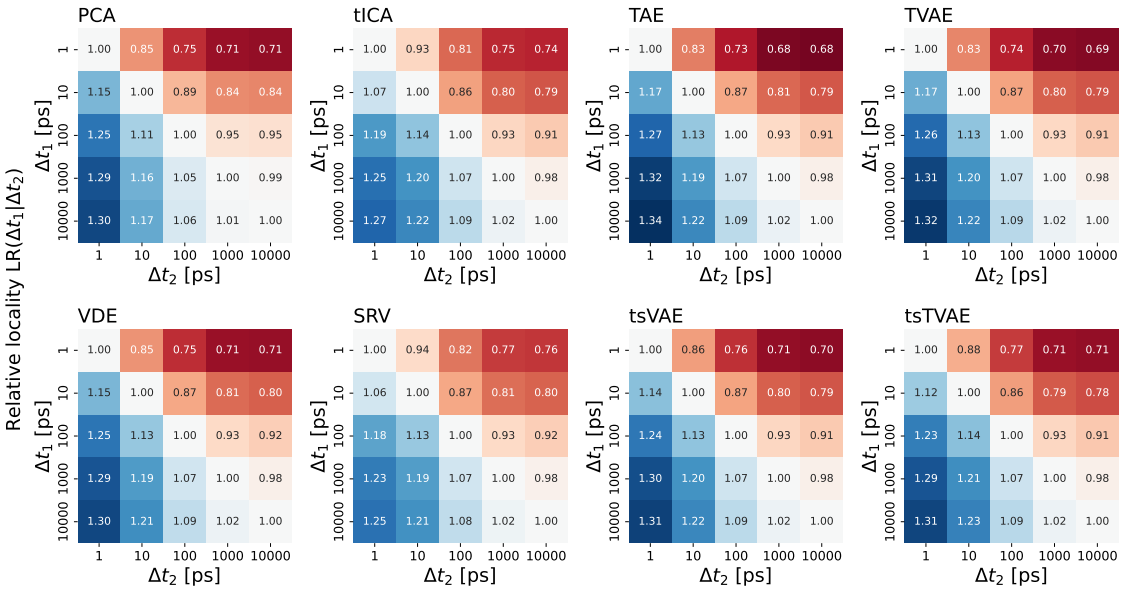


Figure B.5: Relative locality for alanine-dipeptide trajectories

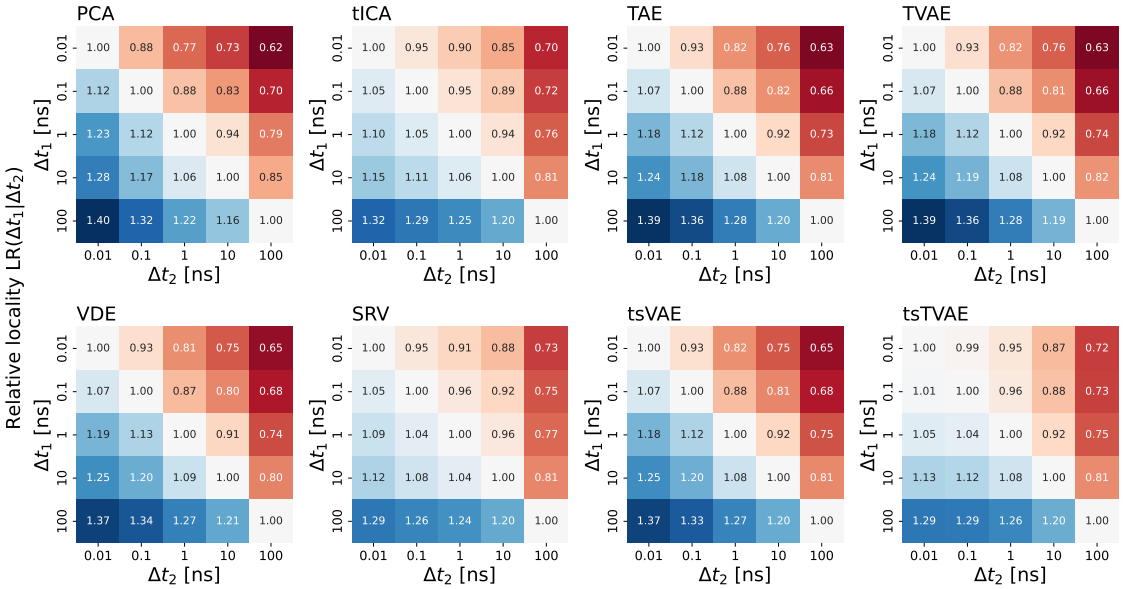


Figure B.6: Relative locality for chignolin trajectories

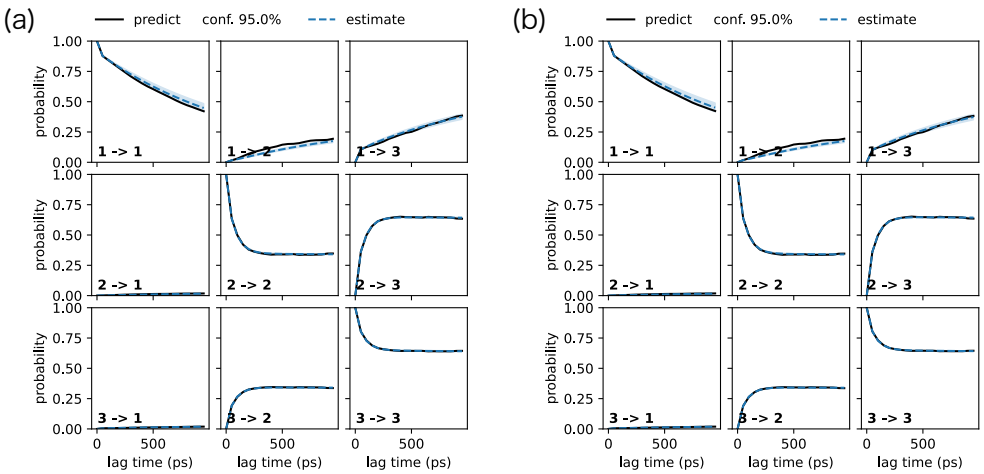


Figure B.7: Chapman-Kolmogorov testing of three macrostates estimated in the encoded space constructed by (a) tsVAE and (b) tsTVAE for alanine-dipeptide trajectories

where T is the time-length, $r(a; A)$ is the rank of the element a among the ordered set A. The lower the value, the closer to space, and the higher the value, the farther away from space.

Figure B.5 and Figure B.6 show the relative locality for the alanine-dipeptide trajectories and the chignolin trajectories, respectively. For all methods, the lower Δt_1 is, the lower the measure is, and the lower Δt_2 is, the higher the measure is. This is because there is an indirect factor called "state" in between. If they are close in time, the structural states of the molecules are similar and consequently embedded in close spatial proximity. The tsVAE and tsTVAE promote spatial proximity in the case of time proximity, but to evaluate this effect, the indirect causality through states must be removed. Since both indirect and direct causality are nonlinear, it is difficult to remove them, and it is also difficult to evaluate the promotion effect from this perspective.

B.5 Other Results for Alanine-dipeptide Trajectories

This chapter briefly describes additional alanine-dipeptide results that were not described before.

Figure B.7 shows the results of the Chapman-Kolmogorov test for the three macrostates of MSM with a lagtime of 50 ps, constructed in latent space by tsVAE and tsTVAE. These methods are consistent with the MSM up to the 1000 ps band, which covers the slowest time scales.

Figure B.8 shows the free energy surface (FES) in latent space and reference space for each method applied in this paper. The FES in the reference space is almost the same for all methods. In the latent space, tICA, tsVAE, and tsTVAE can visually construct an FES that is isomorphic to the reference space. VDE and SRV appear to have spatially collapsed FES, while TAE and TVAE appear to be collapsed in the region $\phi > 0$.

B.6 Additional Results for Chignolin Trajectories

This chapter briefly describes additional chignolin results that were not described before.

Figure B.9 shows the results of the Chapman-Kolmogorov test for the three macrostates of MSM with a lagtime of 1 ns, constructed in latent space by tsVAE and tsTVAE. The configured macrostate can construct a consistent MSM up to 10 ns from that figure.

Figure B.10 shows the free energy surface (FES) in latent space and reference space for each method applied in this paper. tsVAE and tsTVAE find several energy maxima in the potential space.

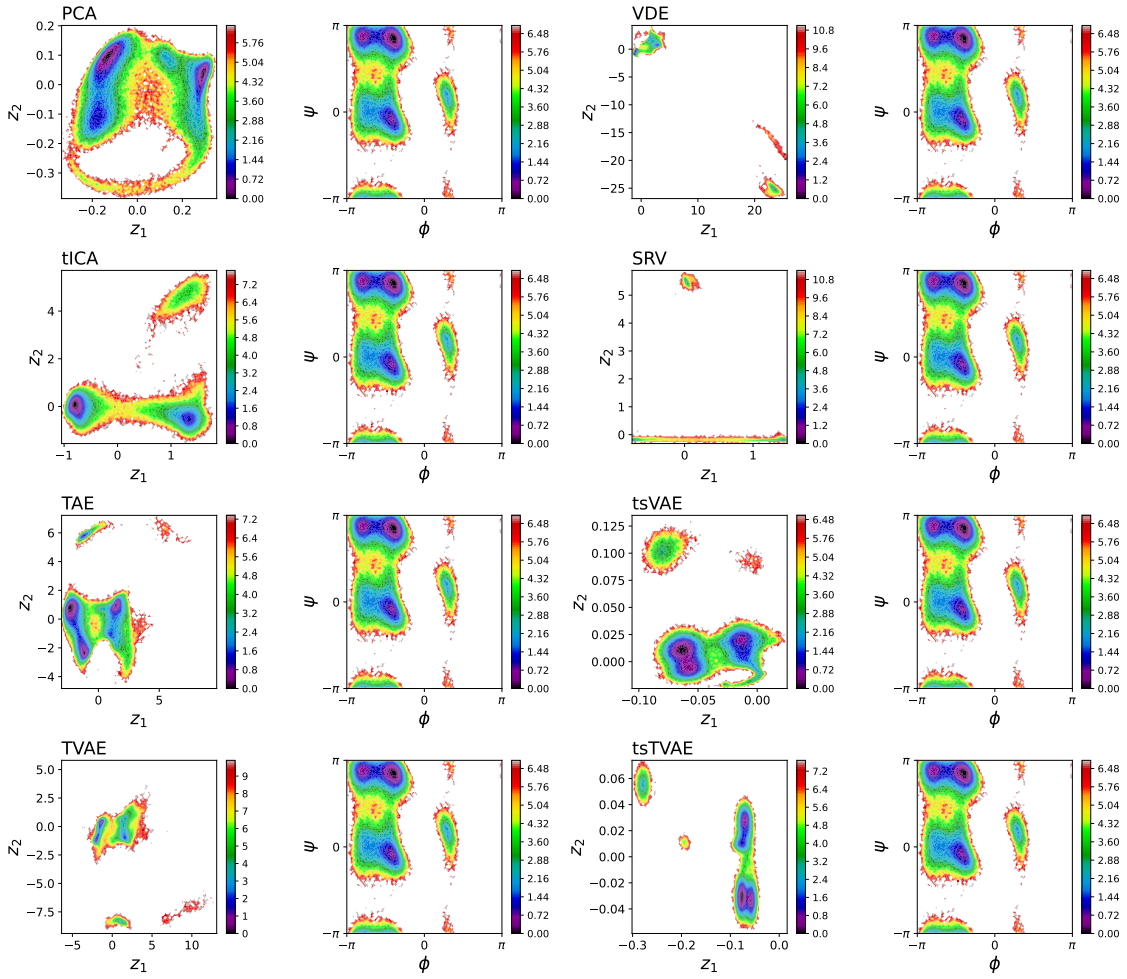


Figure B.8: Free energy surface in the embedded space and the reference space for alanine-dipeptide trajectories

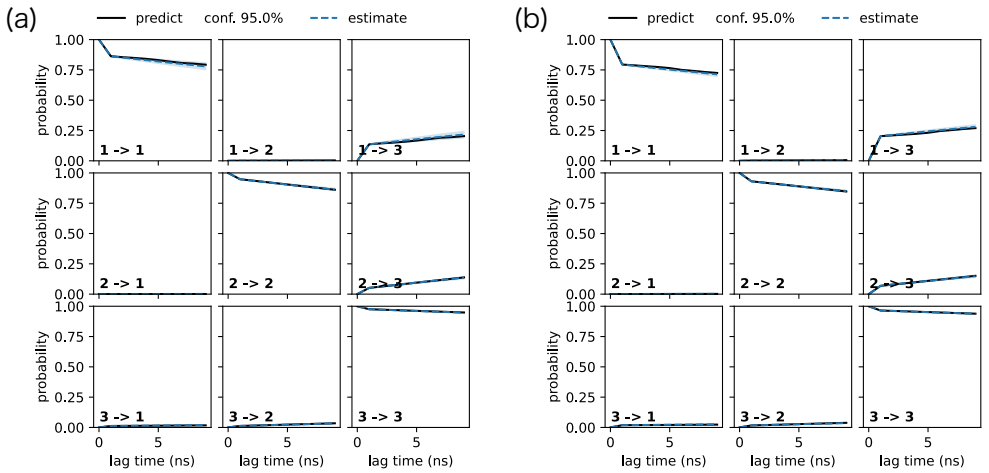


Figure B.9: Chapman-Kolmogorov testing of three macrostates estimated in the encoded space constructed by (a) tsVAE and (b) tsTVAE for chignolin trajectories

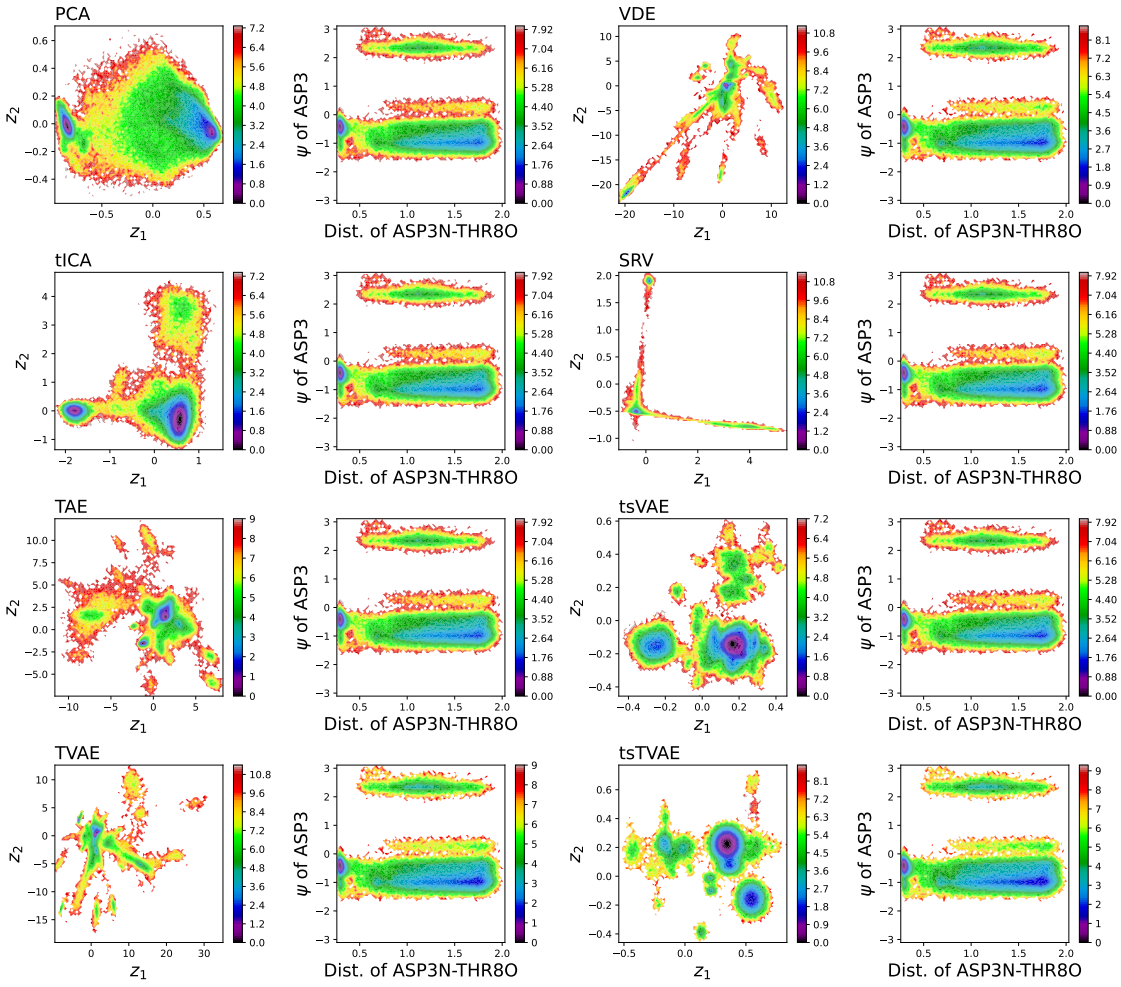


Figure B.10: Free energy surface in the embedded space and the reference space for chignolin trajectories

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[2] M. Ades and P. J. van Leeuwen. An exploration of the equivalent weights particle filter. *Quarterly Journal of the Royal Meteorological Society*, 139(672):820–840, 2013.

[3] A. Agostinelli, T. I. Denk, Z. Borsos, J. Engel, M. Verzetti, A. Caillon, Q. Huang, A. Jansen, A. Roberts, M. Tagliasacchi, M. Sharifi, N. Zeghidour, and C. Frank. MusicLM: Generating music from text. *arXiv preprint arXiv:2301.11325*, 2023.

[4] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[5] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Knowledge Discovery and Data Mining (KDD)*, 2019.

[6] D. Al Chanti and D. Mateus. Olva: Optimal latent vector alignment for unsupervised domain adaptation in medical image segmentation. In M. de Bruijne, P. C. Cattin, S. Cotin, N. Padoy, S. Speidel, Y. Zheng, and C. Essert, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2021*, pages 261–271, Cham, 2021. Springer International Publishing.

[7] T. Al-Moslmi, M. Gallofré Ocaña, A. L. Opdahl, and C. Veres. Named entity extraction for knowledge graphs: A literature overview. *IEEE Access*, 8:32862–32881, 2020.

[8] T. Amarbayasgalan, V. H. Pham, N. Theera-Umpon, and K. H. Ryu. Unsupervised anomaly detection approach for time-series in multi-domains using deep reconstruction error. *Symmetry*, 12(8):1251, 2020.

[9] S. Amari. Information geometry and its applications. *Applied Mathematical Sciences*, 194:374, 2016.

[10] J. L. Anderson and S. L. Anderson. A Monte Carlo implementation of the nonlinear filtering problem to produce ensemble assimilations and forecasts. *Monthly Weather Review*, 127(12):2741–2758, 1999.

[11] A. F. Ansari, K. Benidis, R. Kurle, A. C. Turkmen, H. Soh, A. J. Smola, B. Wang, and T. Januschowski. Deep explicit duration switching models for time series. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 29949–29961. Curran Associates, Inc., 2021.

[12] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.

[13] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[14] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*, 2015.

[15] J. Bai, W. Wang, and C. P. Gomes. Contrastively disentangled sequential variational autoencoder. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 10105–10118. Curran Associates, Inc., 2021.

[16] Y. Balaji, S. Nah, X. Huang, A. Vahdat, J. Song, Q. Zhang, K. Kreis, M. Aittala, T. Aila, S. Laine, B. Catanzaro, T. Karras, and M.-Y. Liu. ediff-i: Text-to-image diffusion models with ensemble of expert denoisers. *arXiv preprint arXiv:2211.01324*, 2022.

[17] S. Bandyopadhyay and J. Mondal. A deep autoencoder framework for discovery of metastable ensembles in biomacromolecules. *J. Chem. Phys.*, 155:114106, 2021.

[18] H. Bao, L. Dong, S. Piao, and F. Wei. BEiT: BERT pre-training of image transformers. In *International Conference on Learning Representations*, 2022.

[19] J. T. Barron. Continuously differentiable exponential linear units. *arXiv preprint arXiv:1704.07483*, 2017.

[20] S. Batzner, A. Musaelian, L. Sun, M. Geiger, J. P. Mailoa, M. Kornbluth, N. Molinari, T. E. Smidt, and B. Kozinsky. E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nat. Commun.*, 13(1):2453, May 2022.

[21] J. Bayer and C. Osendorfer. Learning stochastic recurrent networks. *arXiv preprint arXiv:1411.7610*, 2015.

[22] M. J. Beal. Variational algorithms for approximate Bayesian inference. Technical report, University of London, 2003.

[23] P. Becker, H. Pandya, G. Gebhardt, C. Zhao, C. J. Taylor, and G. Neumann. Recurrent kalman networks: Factorized inference in high-dimensional deep feature spaces. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 544–552. PMLR, 09–15 Jun 2019.

[24] P. Becker-Ehmck, J. Peters, and P. van der Smagt. Switching linear dynamics for variational bayes filtering. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 553–562. PMLR, June 2019.

[25] M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine learning practice and the bias-variance trade-off. In *Proceedings of the National Academy of Sciences*, pages 15849–15854, 2019.

[26] M. Belkin, D. Hsu, and J. Xu. Two models of double descent for weak features. *arXiv preprint arXiv: 1903.07571*, 2019.

[27] Y. Bengio. Deep learning of representations: Looking forward. In *SLSP*, pages 1–37. Springer, 2013.

[28] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE TPAMI*, 35(8), 2013.

[29] B. Berenguel-Baeta, J. Bermudez-Cameo, and J. J. Guerrero. Fredsnet: Joint monocular depth and semantic segmentation with fast fourier convolutions from single panoramas. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6080–6086, 2023.

[30] J. Bergstra, D. Yamins, and D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 115–123, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[31] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. In *neural information processing systems*, pages 2546–2554, 2011.

[32] M. Birjali, M. Kasri, and A. Beni-Hssane. A comprehensive survey on sentiment analysis: Approaches, challenges and trends. *Knowledge-Based Systems*, 226:107134, 2021.

[33] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[34] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(4–5):993–1022, 2003.

[35] L. Bonati, G. Piccini, and M. Parrinello. Deep learning the slow modes for rare events sampling. *Proceedings of the National Academy of Sciences*, 118(44):e2113533118, 2021.

[36] J. Brandstetter, R. Hesselink, E. van der Pol, E. J. Bekkers, and M. Welling. Geometric and physical quantities improve e(3) equivariant message passing. In *International Conference on Learning Representations*, 2022.

[37] J. Brehmer, P. de Haan, S. Behrends, and T. Cohen. Geometric algebra transformer. In *Advances in Neural Information Processing Systems*, volume 37, 2023.

[38] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[39] Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015.

[40] Çaglar Gülçehre, J. Sotelo, M. Moczulski, and Y. Bengio. A robust adaptive stochastic gradient method for deep learning. In *International Joint Conference on Neural Networks (IJCNN)*, 2017.

[41] Y. Cai, Z. Wang, Z. Luo, B. Yin, A. Du, H. Wang, X. Zhou, E. Zhou, X. Zhang, and J. Sun. Learning delicate local representations for multi-person pose estimation. In *ECCV*, 2020.

[42] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9912–9924. Curran Associates, Inc., 2020.

[43] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020.

[44] F. P. Casale, A. Dalca, L. Saglietti, J. Listgarten, and N. Fusi. Gaussian process prior variational autoencoders. In *Neural Information Processing Systems (NeurIPS)*, pages 10369–10380, 2018.

[45] B. Chen, J. Zhang, X. Zhang, Y. Dong, J. Song, P. Zhang, K. Xu, E. Kharlamov, and J. Tang. Gccad: Graph contrastive learning for anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–14, 2022.

[46] C. Chen, X. Lin, Y. Huang, and G. Terejanu. Approximate Bayesian neural network trained with ensemble Kalman filter. In *International Joint Conference on Neural Networks (IJCNN)*, 2019.

[47] C. Chen, X. Lin, and G. Terejanu. An approximate Bayesian long short-term memory algorithm for outlier detection. In *International Conference on Pattern Recognition (ICPR)*, 2018.

[48] C. Chen, C. X. Lu, B. Wang, N. Trigoni, and A. Markham. DynaNet: Neural Kalman dynamical model for motion estimation and prediction. *arXiv preprint arXiv:1908.03918*, 2019.

[49] F. Chen, Y. Wang, B. Wang, and C.-C. J. Kuo. Graph representation learning: A survey. *APSIPA Transactions on Signal and Information Processing*, 9(e15), 2020.

[50] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Neural Information Processing Systems (NeurIPS)*, 2018.

[51] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020.

[52] W. Chen, H. Sidky, and A. L. Ferguson. Capabilities and limitations of time-lagged autoencoders for slow mode discovery in dynamical systems. *J. Chem. Phys.*, 151:064123, 2019.

[53] W. Chen, H. Sidky, and A. L. Ferguson. Nonlinear discovery of slow molecular modes using state-free reversible vampnets. *J. Chem. Phys.*, 150:214114, 2019.

[54] X. Chen, H. Fan, R. Girshick, and K. He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[55] X. Chen and K. He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15750–15758, June 2021.

[56] X. Chen, X. Wang, J. Zhou, Y. Qiao, and C. Dong. Activating more pixels in image super-resolution transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22367–22377, June 2023.

[57] X. Chen, S. Xie, and K. He. An empirical study of training self-supervised vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9620–9629, Los Alamitos, CA, USA, oct 2021. IEEE Computer Society.

[58] Z. Chen, Y. Duan, W. Wang, J. He, T. Lu, J. Dai, and Y. Qiao. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*, 2022.

[59] K. Cheng, S. Aeron, M. C. Hughes, and E. L. Miller. Dynamical wasserstein barycenters for time-series modeling. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 27991–28003. Curran Associates, Inc., 2021.

[60] P. Cheng, M. R. Min, D. Shen, C. Malon, Y. Zhang, Y. Li, and L. Carin. Improving disentangled text representation learning with information-theoretic guidance. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7530–7541, Online, July 2020. Association for Computational Linguistics.

[61] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.

[62] A. J. Chorin and X. Tu. Interpolation and iteration for nonlinear filters. *Communications in Applied Mathematics and Computational Science*, 5:221–240, 2010.

[63] J. Chung, K. Kastner, L. Dinh, K. Goel, A. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. In *Neural Information Processing Systems (NeurIPS)*, 2015.

[64] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020.

[65] D. A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations (ICLR)*, 2016.

[66] J. Collins, J. Sohl-Dickstein, and D. Sussillo. Capacity and trainability in recurrent neural networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[67] M. Corazza, E. Kalnay, D. J. Patil1, S.-C. Yang, R. Morss, M. Cai, I. Szunyogh, B. R. Hunt, and J. A. Yorke. Use of the breeding technique to estimate the structure of the analysis "errors of the day". *Nonlinear Processes in Geophysics*, 10:233–243, 2003.

[68] M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[69] Z. Dai, H. Liu, Q. V. Le, and M. Tan. Coatnet: Marrying convolution and attention for all data sizes. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 3965–3977. Curran Associates, Inc., 2021.

[70] L. S. Dalenogare, G. B. Benitez, N. F. Ayala, and A. G. Frank. The expected contribution of industry 4.0 technologies for industrial performance. *International Journal of Production Economics*, 204:383–394, 2018.

[71] T. Darden, D. York, and L. Pedersen. Particle mesh Ewald: An N·log(N) method for Ewald sums in large systems. *The Journal of Chemical Physics*, 98(12):10089–10092, 06 1993.

[72] P. Das, M. Moll, H. Stamati, L. E. Kavraki, and C. Clementi. Low-dimensional, free-energy landscapes of protein-folding reactions by nonlinear dimensionality reduction. 103(26):9885–9890. Publisher: Proceedings of the National Academy of Sciences.

[73] S. De and S. Smith. Batch normalization biases residual blocks towards the identity function in deep networks. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19964–19975. Curran Associates, Inc., 2020.

[74] E. Denton and R. Fergus. Stochastic video generation with a learned prior. In *International Conference on Machine Learning (ICML)*, pages 1174–1183, 2018.

[75] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[76] S. Doerr, I. Ariz, M. J. Harvey, and G. De Fabritiis. Dimensionality reduction methods for molecular simulations. *arXiv preprint arXiv:1710.10629*, 2017.

[77] J. Domke and D. Sheldon. Importance weighting and variational inference. In *Neural Information Processing Systems (NeurIPS)*, 2018.

[78] M. Dorn, M. E Silva, L. Buriol, and L. Lamb. Three-dimensional protein structure prediction: methods and computational strategies. *Comput. Biol. Chem.*, 53:251–276, 2014.

[79] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[80] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canadian Cartographer*, 10(2):112–122, 1973.

[81] T. Dozat. Incorporating nesterov momentum into adam. In *International Conference on Learning Representations Workshop (ICLR Workshop)*, 2016.

[82] A. Duceo and A. M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Oxford Handbook of Nonlinear Filtering*, 12(654–704):3, 2011.

[83] J. Duchi, E. Hazan, and Y. Singer. Adaptive subradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[84] P. Eastman, J. Swails, J. D. Chodera, R. T. McGibbon, Y. Zhao, K. A. Beauchamp, L.-P. Wang, A. C. Simmonett, M. P. Harrigan, C. D. Stern, R. P. Wiewiora, B. R. Brooks, and V. S. Pande. OpenMM 7: Rapid development of high performance algorithms for molecular dynamics. 13(7):e1005659.

[85] S. Elfwing, E. Uchibe, and K. Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Network*, 107:3–11, 2018.

[86] F. A. Elhaj, N. Salim, A. R. Harris, T. T. Swee, and T. Ahmed. Arrhythmia recognition and classification using combined linear and nonlinear features of ECG signal. *Computer Methods and Programs in Biomedicine*, 127(52–63), 2016.

[87] S. A. Eslami, D. J. Rezende, F. Besse, F. Viola, A. S. Morcos, M. Garnelo, A. Ruderman, A. A. Rusu, I. Danihelka, K. Gregor, D. P. Reichert, L. Buesing, T. Weber, O. Vinyals, D. Rosenbaum, N. Rabinowitz, H. King, C. Hillier, M. Botvinick, D. Wierstra, K. Kavukcuoglu, and D. Hassabis. Neural scene representation and rendering. *Science*, 360(6394):1204–1210, 2018.

[88] G. Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte-Carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, 99(C5):10143–10162, 1994.

[89] G. Evensen. The ensemble Kalman filter: Theoretical formulation and practical implementation. *Ocean dynamics*, 53(4):343–367, 2003.

[90] J. R. Firth. A synopsis of linguistic theory, 1930-55. *The Philological Society*, 1952–1959, 1957.

[91] A. M. Fox, T. J. Hoar, J. L. Anderson, A. F. Arellano, W. K. Smith, M. E. Litvak, N. MacBean, D. S. Schimel, and D. J. P. Moore. Evaluation of a data assimilation system for land surface models using CLM4.5. *Journal of Advances in Modeling Earth Systems*, 10(10):2471–2494, 2018.

[92] M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[93] M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Neural Information Processing Systems (NeurIPS)*, 2017.

[94] M. Fraccaro, S. K. Sønderby, U. Paquet, and O. Winther. Sequential neural models with stochastic layers. In *Neural Information Processing Systems (NeurIPS)*, 2016.

[95] A. G. Frank, L. S. Dalenogare, and N. F. Ayala. Industry 4.0 technologies: Implementation patterns in manufacturing companies. *International Journal of Production Economics*, 210:15–26, 2019.

[96] D. Freedman and P. Diaconis. On the histogram as a density estimator: L2 theory. *Probability Theory and Related Fields*, 57(4):453–476, December 1981.

[97] M. Frei and H. R. Künsch. Bridging the ensemble kalman and particle filters. *Biometrika*, 100(4):781–800, dec 2013.

[98] Y. Fukunishi. Structural ensemble in computational drug screening. *Expert Opin. Drug Metab. Toxicol.*, 6(7):835–849, 2010.

[99] S. Funk. Rmsprop loses to smorms3 - beware the epsilon! `https://sifter.org/~simon/journal/20150420.html`.

[100] P. Gaikwad, A. Mandal, P. Ruth, G. Juve, D. Król, and E. Deelman. Anomaly detection for scientific workflow applications on networked clouds. In *International Conference on High Performance Computing and Simulation*, pages 645–652, 2016.

[101] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

[102] Z. Gan, C. Li, R. Henao, D. E. Carlson, and L. Carin. Deep temporal sigmoid belief networks for sequence modeling. In *Neural Information Processing Systems (NeurIPS)*, pages 2467–2475, 2015.

[103] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1243–1252. PMLR, 06–11 Aug 2017.

[104] F. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with lstm. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pages 850–855 vol.2, 1999.

[105] D. T. Gillespie. Exact numerical simulation of the ornstein-uhlenbeck process and its integral. *Phys. Rev. E*, 54(2):2084–2091, 1996.

[106] K. Ginalski, A. Elofsson, D. Fischer, and L. Rychlewski. 3d-jury: a simple approach to improve protein structure predictions. *Bioinformatics*, 19(8):1015–1018, 2003.

[107] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[108] A. Glielmo, B. E. Husic, A. Rodriguez, C. Clementi, F. Noé, and A. Laio. Unsupervised learning methods for molecular simulation data. 121(16):9722–9758.

[109] S. J. Godsill, A. Doucet, and M. West. Monte Carlo smoothing for nonlinear time series. *Journal of the american statistical association*, 99(465):156–168, 2004.

[110] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[111] N. Gordon, D. Salmond, and A. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140:107–113(6), April 1993.

[112] A. Goyal, A. Sordoni, M.-A. Côté, N. R. Ke, and Y. Bengio. Z-Forcing: Training stochastic recurrent networks. In *Neural Information Processing Systems (NeurIPS)*, 2017.

[113] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28(10):2222–2232, 2017.

[114] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, k. kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent - a new approach to self-supervised learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284. Curran Associates, Inc., 2020.

[115] M. U. Gutmann and A. Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(11):307–361, 2012.

[116] D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[117] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning (ICML)*, 2019.

[118] J. Han, M. R. Min, L. Han, L. E. Li, and X. Zhang. Disentangled recurrent wasserstein autoencoder. In *International Conference on Learning Representations*, 2021.

[119] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, Z. Yang, Y. Zhang, and D. Tao. A survey on vision transformer. In *IEEE transactions on pattern analysis and machine intelligence*, 2022.

[120] X. Hao and P. Shafto. Coupled variational autoencoder. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 12546–12555. PMLR, 23–29 Jul 2023.

[121] M. P. Harrigan and V. S. Pande. Landmark kernel tica for conformational dynamics. *bioRxiv.org e-Print archive*, 2017.

[122] Z. Harris. Distributional structure. *Word*, 10:146–162, 1954.

[123] M. J. Harvey, G. Giupponi, and G. D. Fabritiis§. Acemd: Accelerating biomolecular dynamics in the microsecond time scale. *J. Chem. Theory Compute.*, 5(6):1632–1639, 2009.

[124] P. Hawkins and A. Nicholls. Conformer generation with omega: learning from the data set and the analysis of failures. *J. Chem. Inf. Model.*, 52(11):2919–2936, 2012.

[125] H. Hayashi, J. Koushik, and G. Neubig. Eve: A gradient based optimization method with locally and globally adaptive learning rates. *arXiv preprint arXiv:1611.01505*, 2016.

[126] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735, Los Alamitos, CA, USA, jun 2020. IEEE Computer Society.

[127] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[128] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[129] P. He, X. Liu, J. Gao, and W. Chen. DEBERTA: Decoding-enhanced BERT with isentangled attention. In *International Conference on Learning Representations*, 2021.

[130] T. Head, MechCoder, G. Louppe, I. Shcherbatyi, fcharras, Z. Vinícius, cmmalone, C. Schröder, nel215, N. Campos, T. Young, S. Cereda, T. Fan, rene rex, K. K. Shi, J. Schwabedal, carlosdanielcsantos, Hvass-Labs, M. Pak, SoManyUsernamesTaken, F. Callaway, L. Estève, L. Besson, M. Cherti, K. Pfannschmidt, F. Linzberger, C. Cauet, A. Gut, A. Mueller, and A. Fabisch. scikit-optimize/scikit-optimize: v0.5.2, Mar. 2018.

[131] C. X. Hernández, H. K. Wayment-Steele, M. M. Sultan, B. E. Husic, and V. S. Pande. Variational encoding of complex dynamics. *Physical Review E*, 97(6):062412, 2018.

[132] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

[133] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[134] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc., 2020.

[135] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, nov 1997.

[136] M. Hoffmann, M. K. Scherer, T. Hempel, A. Mardt, B. de Silva, B. E. Husic, S. Klus, H. Wu, J. N. Kutz, S. Brunton, and F. Noé. Deeptime: a Python library for machine learning dynamical models from time series data. *Machine Learning: Science and Technology*, 2021.

[137] S. Honda, K. Yamasaki, Y. Sawada, and H. Morii. 10 residue folded peptide designed by segment statistics. *Structure*, 12:1507–1518, 2004.

[138] M. Horn, M. Moor, C. Bock, B. Rieck, and K. Borgwardt. Set functions for time series. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4353–4363. PMLR, 13–18 Jul 2020.

[139] J. Howard and S. Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 328–339, 2018.

[140] W.-N. Hsu, Y. Zhang, and J. Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[141] X. Hu and C. C. Douglas. Applications of data assimilation methods on a coupled dual porosity stokes model. In V. V. Krzhizhanovskaya, G. Závodszky, M. H. Lees, J. J. Dongarra, P. M. A. Sloot, S. Brissos, and J. Teixeira, editors, *Computational Science – ICCS 2020*, pages 72–85, Cham, 2020. Springer International Publishing.

[142] H. Huang, L. Sun, B. Du, Y. Fu, and W. Lv. Graphgdp: Generative diffusion processes for permutation invariant graph generation. *arXiv preprint arXiv:2212.01842*, 2022.

[143] J. Huang, S. Rauscher, G. Nawrocki, T. Ran, M. Feig, B. L. de Groot, H. Grubmüller, and A. D. MacKerell. CHARMM36m: an improved force field for folded and intrinsically disordered proteins. 14(1):71–73.

[144] R. Huang, J. Huang, D. Yang, Y. Ren, L. Liu, M. Li, Z. Ye, J. Liu, X. Yin, and Z. Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. *arXiv preprint arXiv:2301.12661*, 2023.

[145] A. Hundt, V. Jain, and G. D. Hager. sharpdarts: Faster and more accurate differentiable architecture search. *arXiv preprint arXiv: 1903.09900*, abs/1903.09900, 2019.

[146] B. E. Husic and V. S. Pande. Markov state models: From an art to a science. *Journal of the American Chemical Society*, 140(7):2386–2396, 2018. PMID: 29323881.

[147] A. Hyvarinen and H. Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ica. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[148] A. Hyvarinen and H. Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ICA. In *Neural Information Processing Systems (NIPS)*, 2016.

[149] A. Hyvarinen and H. Morioka. Nonlinear ICA of Temporally Dependent Stationary Sources. In A. Singh and J. Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 460–469. PMLR, 20–22 Apr 2017.

[150] A. Hyvarinen, H. Sasaki, and R. Turner. Nonlinear ica using auxiliary variables and generalized contrastive learning. In K. Chaudhuri and M. Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 859–868. PMLR, 16–18 Apr 2019.

[151] M. I.Jordan. Serial order: A parallel distributed processing approach, ICS report 8604. Technical report, Institute for Cognitive Science, UCSD, La Jolla, 1986.

[152] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. Bach and D. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[153] Y. Ishikawa. A script for automated 3-dimentional structure generation and conformer search from 2-dimentional chemical drawing. *Bioinformatics*, 9(19):988–992, 2013.

[154] T. Ishizone, T. Higuchi, and K. Nakamura. Ensemble kalman variational objective: a variational inference framework for sequential variational auto-encoders. *Nonlinear Theory and Its Applications, IEICE*, 14(4):691–717, 2023.

[155] T. Ishizone, T. Higuchi, and K. Nakamura. Online anomaly detection and cycle-time estimation system for production lines. *SICE*, 59(7):342–352, 2023.

[156] T. Ishizone, T. Higuchi, K. Okusa, and K. Nakamura. An online system of detecting anomalies and estimating cycle times for production lines. In *IECON 2022 – 48th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6, 2022.

[157] T. Ishizone, Y. Matsunaga, S. Fuchigami, and K. Nakamura. Representation of protein dynamics disentangled by time-structure-based prior. *J. Chem. Theory Comput.*, 20:436–450, 1 2024.

[158] H. Jaeger. The" echo state" approach to analysing and training recurrent neural networks-with an erratum note'. *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, 148, 01 2001.

[159] A. Jaiswal, A. R. Babu, M. Z. Zadeh, D. Banerjee, and F. Makedon. A survey on contrastive self-supervised learning. *arXiv preprint arXiv:2011.00362*, 2020.

[160] J. L. W. V. Jensen. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta Mathematica*, 30(none):175 – 193, 1906.

[161] S. Jo, T. Kim, V. G. Iyer, and W. Im. CHARMM-GUI: A web-based graphical user interface for CHARMM. 29(11):1859–1865. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/jcc.20945.

[162] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[163] W. L. Jorgensen, J. Chandrasekhar, J. D. Madura, R. W. Impey, and M. L. Klein. Comparison of simple potential functions for simulating liquid water. 79(2):926–935. Publisher: American Institute of Physics.

[164] R.-Y. Ju, C.-C. Chen, J.-S. Chiang, Y.-S. Lin, and W.-H. Chen. Resolution enhancement processing on low quality images using swin transformer based on interval dense connection strategy. *Multimedia Tools and Applications*, pages 1–17, 2023.

[165] M. S. Junayed, A. Sadeghzadeh, M. B. Islam, L.-K. Wong, and T. Aydın. Himode: A hybrid monocular omnidirectional depth estimation model. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 5208–5217, 2022.

[166] R. Kalman. A new approach to linear filtering and prediction problems. *ASME Journal of Basic Engineering*, 82:35–45, 1960.

[167] T. Karras, M. Aittala, J. Hellsten, S. Laine, J. Lehtinen, and T. Aila. Training generative adversarial networks with limited data. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12104–12114. Curran Associates, Inc., 2020.

[168] A. Khan and A. J. Storkey. Hamiltonian latent operators for content and motion disentanglement in image sequences. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 7250–7263. Curran Associates, Inc., 2022.

[169] I. Khemakhem, D. Kingma, R. Monti, and A. Hyvarinen. Variational autoencoders and nonlinear ica: A unifying framework. In S. Chiappa and R. Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2207–2217. PMLR, 26–28 Aug 2020.

[170] I. Khemakhem, R. Monti, D. Kingma, and A. Hyvarinen. Ice-beem: Identifiable conditional energy-based deep models based on nonlinear ica. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12768–12778. Curran Associates, Inc., 2020.

[171] S. B. Kim, C. J. Dsilva, I. G. Kevrekidis, and P. G. Debenedetti. Systematic characterization of protein folding pathways using diffusion maps: Application to trp-cage miniprotein. 142(8):085101.

[172] T. Kim, S. Ahn, and Y. Bengio. Variational temporal abstraction. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[173] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2014.

[174] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[175] N. Kitaev, L. Kaiser, and A. Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

[176] G. Kitagawa. Monte carlo filtering and smoothing for nonlinear non-gaussian state space model. *Proceedings of the ISCIE International Symposium on Stochastic Systems Theory and its Applications*, 1998:1–6, 1998.

[177] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. In *Advances in Nueral Information Processing Systems (NIPS)*, pages 971–980, 2017.

[178] J. Klett, A. Cortes-Cabrera, R. Gil-Redondo, F. Gago, and A. Morreale. Alfa: Automatic ligand flexibility assignment. *J. Chem. Inf. Model.*, 54(1):314–323, 2014.

[179] I. Kobyzev, S. D. Prince, and M. A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 43(11):3964–3979, nov 2021.

[180] L. Kong, J. Cui, H. Sun, Y. Zhuang, B. A. Prakash, and C. Zhang. Autoregressive diffusion model for graph generation. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17391–17408. PMLR, 23–29 Jul 2023.

[181] O. Kopuklu, J. Zheng, H. Xu, and G. Rigoll. Driver anomaly detection: A dataset and contrastive learning approach. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 91–100, January 2021.

[182] D. Koshland, G. Nemethy, and D. Filmer. Comparison of experimental binding data and theoretical models in proteins containing subunits. *Biochemistry*, 5(1):365–385, 1966.

[183] A. Kosson, D. Fan, and M. Jaggi. Ghost noise for regularizing deep neural networks. *arXiv preprint arXiv: 2305.17205*, 2023.

[184] V. Kothapalli. Randomized schur complement views for graph contrastive learning. In *International Conference on Machine Learning*, 2023.

[185] R. G. Krishnan, U. Shalit, and D. Sontag. Deep Kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

[186] R. G. Krishnan, U. Shalit, and D. Sontag. Structured inference networks for nonlinear state space models. *arXiv preprint arXiv:1609.09869*, 2016.

[187] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Nueral Information Processing Systems (NIPS)*, 2012.

[188] J. Kruschke. *Doing Bayesian data analysis: A tutorial with R and BUGS*. Academic Press / Elsevier, 2011.

[189] S. Kullback. *Information Theory and Statisticcs*. John Wiley & Sons, 1959.

[190] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Sciences*, 22(1):79–86, 1951.

[191] R. Kurle, S. S. Rangapuram, E. de Bézenac, S. Günnemann, and J. Gasthaus. Deep rao-blackwellised particle filters for time series forecasting. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 15371–15382. Curran Associates, Inc., 2020.

[192] W. T. Lai, R.-B. Chen, Y. Chen, and T. Koch. Variational bayesian inference for network autoregression models. *Computational Statistics & Data Analysis*, 169, 2022.

[193] J. Lavaei, S. Sojoudi, and R. M. Murray. Simple delay-based implementation of continuous-time controllers. In *Proceedings of the 2010 American Control Conference*, pages 5781–5788, 2010.

[194] D. Lawson, G. Tucker, C. A. Naesseth, C. Maddison, R. P. Adams, and Y. W. Teh. Twisted variational sequential Monte Carlo. In *Third workshop on Bayesian Deep Learning, NeurIPS*, 2018.

[195] Q. V. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, page 507–514, Madison, WI, USA, 2012. Omnipress.

[196] T. A. Le, M. Igl, T. Rainforth, T. Jin, and F. Wood. Auto-encoding sequential Monte Carlo. In *International Conference on Learning Representations (ICLR)*, 2018.

[197] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[198] J. Lee and K. H. Jin. Local texture estimator for implicit representation function. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1929–1938, June 2022.

[199] J. Li, D. Li, S. Savarese, and S. Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023.

[200] J. Li, D. Li, C. Xiong, and S. Hoi. BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022.

[201] J. Li, A. Sun, J. Han, and C. Li. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):50–70, 2022.

[202] J. Li, P. Zhou, C. Xiong, and S. Hoi. Prototypical contrastive learning of unsupervised representations. In *International Conference on Learning Representations*, 2021.

[203] Q. Li, R. Li, K. Ji, and W. Dai. Kalman filter and its application. In *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, pages 74–77, 2015.

[204] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, and X. Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[205] Y. Li, R. Yuan, G. Zhang, Y. Ma, X. Chen, H. Yin, C. Lin, A. Ragni, E. Benetos, N. Gyenge, R. Dannenberg, R. Liu, W. Chen, G. Xia, Y. Shi, W. Huang, Y. Guo, and J. Fu. MERT: Acoustic music understanding model with large-scale self-supervised training. *arXiv preprint arXiv:2306.00107*, 2023.

[206] J. Liang, J. Cao, G. Sun, K. Zhang, L. Van Gool, and R. Timofte. Swinir: Image restoration using swin transformer. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 1833–1844, 2021.

[207] B. Lim, S. O. Arik, N. Loeff, and T. Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *arXiv preprint arXiv:1912.09363*, 2019.

[208] T. Lin, Y. Wang, X. Liu, and X. Qiu. A survey of transformers. *arXiv preprint arXiv:2106.04554*, 2021.

[209] Y. Lin, I. Koprinska, and M. Rana. Ssdnet: State space decomposition neural network for time series forecasting. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 370–378, 2021.

[210] S. Linderman, M. Johnson, A. Miller, R. Adams, D. Blei, and L. Paninski. Bayesian Learning and Inference in Recurrent Switching Linear Dynamical Systems. In A. Singh and J. Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 914–922. PMLR, 20–22 Apr 2017.

[211] K. Lindorff-Larsen, S. Piana, K. Palmo, P. Maragakis, J. L. Klepeis, R. O. Dror, and D. E. Shaw. Improved side-chain torsion potentials for the amber ff99sb protein force field. *Proteins: Structure, Function, and Bioinformatics*, 78(8):1950–1958, 2010.

[212] F. Lindsten, J. Helske, and M. Vihola. Graphical model inference: Sequential Monte Carlo meets deterministic approximations. In *Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

[213] F. Liu, X. Zhou, J. Cao, and Y. Zhang. Anomaly detection in quasi-periodic time series based on automatic data segmentation and attentional LSTM-CNN. In *IEEE Transactions on Knowledge and Data Engineering*, volume 34, pages 2626–2640, 2020.

[214] H. Liu, Z. Chen, Y. Yuan, X. Mei, X. Liu, D. Mandic, W. Wang, and M. D. Plumbley. AudioLDM: Text-to-audio generation with latent diffusion models. *Proceedings of the International Conference on Machine Learning*, 2023.

[215] H. Liu, L. He, H. Bai, B. Dai, K. Bai, and Z. Xu. Structured inference for recurrent hidden semi-markov model. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 2447–2453. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[216] H. Liu, F. Liu, X. Fan, and D. Huang. Polarized self-attention: Towards high-quality pixel-wise regression. *Arxiv Pre-Print arXiv:2107.00782*, 2021.

[217] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations (ICLR)*, 2020.

[218] S. Liu and W. Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 730–734, 2015.

[219] S. Liu, H. Yu, C. Liao, J. Li, W. Lin, A. X. Liu, and S. Dustdar. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*, 2022.

[220] Y. Liu, T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, H. He, A. Li, M. He, Z. Liu, Z. Wu, L. Zhao, D. Zhu, X. Li, N. Qiang, D. Shen, T. Liu, and B. Ge. Summary of chatgpt-related research and perspective towards the future of large language models. *arXiv preprint arXiv:2304.01852*, 2023.

[221] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[222] Y. Liu, H. Wu, J. Wang, and M. Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. 2022.

[223] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong, F. Wei, and B. Guo. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12009–12019, June 2022.

[224] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, October 2021.

[225] F. Locatello, D. Weissenborn, T. Unterthiner, A. Mahendran, G. Heigold, J. Uszkoreit, A. Dosovitskiy, and T. Kipf. Object-centric learning with slot attention. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 11525–11538. Curran Associates, Inc., 2020.

[226] K. Loh, P. S. Omrani, and R. van der Linden. Deep learning and data assimilation for real-time production prediction in natural gas wells. *arXiv preprint arXiv:1802.05141*, 2018.

[227] I. Loshchilov and F. Hutter. Sgdr: stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

[228] I. Loshchilov and F. Hutter. Fixing weight decay regularization in adam. *arXiv preprint arXiv: 1711.05101*, 2018.

[229] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[230] L. Luo, Y. Xiong, Y. Liu, and X. Sun. Adaptive gradient methods with dynamic bound of learning rate. In *International Conference on Learning Representations (ICLR)*, 2019.

[231] P. Luo, X. Wang, W. Shao, and Z. Peng. Towards understanding regularization in batch normalization. *arXiv preprint arXiv: 1809.00846*, 2018.

[232] Y. Lv. An adaptive real-time outlier detection algorithm based on ARMA model for radar's health monitoring. In *IEEE Autotestcon*, pages 1–7, 2015.

[233] S. Lyu and E. P. Simoncelli. Nonlinear image representation using divisive normalization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[234] H. Ma, S. Leng, K. Aihara, W. Lin, and L. Chen. Randomly distributed embedding making short-term high-dimensional data predictable. *Proceedings of the National Academy of Sciences*, 115(43):E9994–E10002, 2018.

[235] J. Ma, L. Sun, H. Wang, Y. Zhang, and U. Aickelin. Supervised anomaly detection in uncertain pseudoperiodic data streams. *ACM Transactions on Internet Technology*, 16(1):4, 2016.

[236] X. Ma, P. Karkus, D. Hsu, and W. S. Lee. Particle filter recurrent neural networks. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI, 2020*, pages 5101–5108, 2020.

[237] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning (ICML)*, page 3, 2013.

[238] C. J. Maddison, J. Lawson, G. Tucker, N. Heess, M. Norouzi, A. Mnih, A. Doucet, and Y. Teh. Filtering variational objectives. In *Neural Information Processing Systems (NeurIPS)*, 2017.

[239] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations (ICLR)*, 2017.

[240] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal. Long short term memory networks for anomaly detection in time series. In *the 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2015.

[241] A. Mardt and F. Noé. Progress in deep Markov state modeling: Coarse graining and experimental data restraints. *J. Chem. Phys.*, 155(21):1–14, 2021.

[242] A. Mardt, L. Pasquali, H. Wu, and F. Noé. Vampnets for deep learning of molecular kinetics. *Nat. Commun.*, 8(1):1–11, 2018.

[243] E. Maria Novoa, L. Ribas de Pouplana, X. Barril, and M. Orozco. Ensemble docking from homology models. *J. Chem. Theory Compute.*, 6(8):2547–2557, 2010.

[244] V. Masrani, T. A. Le, and F. Wood. The thermodynamic variational objective. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[245] J. McCammon, B. Gelin, and M. Karplus. Dynamics of folded proteins. *Nature*, 267(5612):585–590, 1977.

[246] R. T. McGibbon, B. E. Husic, and V. S. Pande. Identification of simple reaction coordinates from complex dynamics. *J. Chem. Phys.*, 146:044109, 2017.

[247] R. T. McGibbon and V. S. Pande. Learning kinetic distance metrics for markov state models of protein conformational dynamics. *J. Chem. Theory Compute.*, 9:2900–2906, 2013.

[248] S. Mehrang, E. Helander, M. Pavel, A. Chieh, and I. Korhonen. Outlier detection in weight time series of connected scales. In *IEEE International Conference on Bioinformatics and Biomedicine*, pages 1489–1496, 2015.

[249] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR) Workshop*, 2013.

[250] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[251] T. P. Minka. Estimating a Dirichlet distribution. Technical Report 1, MIT, 2000.

[252] V. Mirjalili and M. Feig. Protein structure refinement through structure selection and averaging from molecular dynamics ensembles. *J. Chem. Theory Compute.*, 9(2):1294–1303, 2013.

[253] I. Misra and L. v. d. Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[254] H. L. Mitchell and P. L. Houtekamer. An adaptive ensemble Kalman filter. *Monthly Weather Review*, 128(2):416–433, 2000.

[255] A. Mitsutake and H. Takano. Relaxation mode analysis for molecular dynamics simulations of proteins. 10(2):375–389.

[256] A. Mnih and K. Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[257] D. Molchanov, V. Kharitonov, A. Sobolev, and D. Vetrov. Doubly semi-implicit variational inference. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

[258] J. Monod, J. Changeux, and F. Jacob. Allosteric proteins and cellular control systems. *J. Mol. Biol.*, 6(4):306–329, 1963.

[259] J. Monod, J. Wyman, and J. Changeux. On nature of allosteric transitions – a plausible model. *J. Mol. Biol.*, 12(1):88–118, 1965.

[260] A. Moretti, Z. Wang, L. Wu, and I. Pe'er. Smoothing nonlinear variational objectives with sequential Monte Carlo. In *International Conference on Learning Representations (ICLR)*, 2019.

[261] A. K. Moretti, Z. Wang, L. Wu, I. Drori, and I. Pe'er. Particle smoothing variational objectives. *arXiv preprint arXiv:1909.09734*, 2019.

[262] A. K. Moretti, Z. Wang, L. Wu, I. Drori, and I. Pe'er. Variational objectives for Markovian dynamics with backwards simulation. In *European Conference on Artificial Intelligence (ECAI)*, 2020.

[263] M. Munir, S. A. Siddiqui, M. A. Chattha, A. Dengel, and S. Ahmed. Fusead: Unsupervised anomaly detection in streaming sensors data by fusing statistical and deep learning models. *Sensors*, 19(11), 2019.

[264] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed. Deepant: A deep learning approach for unsupervised anomaly detection in time series. *IEEE Access*, 7:1991–2005, 2019.

[265] C. A. Naesseth, S. W. Linderman, R. Ranganath, and D. M. Blei. Variational sequential Monte Carlo. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.

[266] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*, pages 807–814, 2010.

[267] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. Deep double descent: Where bigger models and more data hurt. *arXiv preprint arXiv: 1912.02292*, 2019.

[268] Y. Naritomi and S. Fuchigami. Slow dynamics in protein fluctuations revealed by time-structure based independent component analysis: The case of domain motions. *J. Chem. Phys.*, 134(6):065101, 2011.

[269] Y. Nesterov. A method of solving a convex programming problem with convergence rate o(1/sqr(k)). *Soviet Mathematcis Doklady*, 27:372–376, 1983.

[270] G. Nguyen-Quynh, P. Becker, C. Qiu, M. Rudolph, and G. Neumann. Switching recurrent kalman networks. *arXiv preprint arXiv:2111.08291*, 2021.

[271] C. Niu, Y. Song, J. Song, S. Zhao, A. Grover, and S. Ermon. Permutation invariant graph generation via score-based generative modeling. In *AISTATS*, pages 4474–4484. PMLR, 2020.

[272] F. Noé and F. Nüske. A variational approach to modeling slow processes in stochastic dynamical systems. *Multiscale Modeling & Simulation*, 11(2):635–655, 2013.

[273] A. Nouranizadeh, M. Matinkia, M. Rahmati, and R. Safabakhsh. Maximum entropy weighted independent set pooling for graph neural networks. *arXiv preprint arXiv:2107.01410*, 2021.

[274] F. Nüske, H. Wu, J.-H. Prinz, C. Wehmeyer, C. Clementi, and F. Noé. Markov state models from short non-equilibrium simulations—analysis and correction of estimation bias. *J. Chem. Phys.*, 146:094104, 2017.

[275] M. Okada, S. Takenaka, and T. Taniguchi. Multi-person pose tracking using sequential monte carlo with probabilistic neural pose predictor. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 10024–10030, 2020.

[276] S. Okuno, K. Aihara, and Y. Hirata. Combining multiple forecasts for multivariate time series via state-dependent weighting. *Chaos*, 29:033128, 2019.

[277] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[278] L. Orellana and M. Orozco. Understanding protein dynamics with coarse- grained models: from structures to disease. *The FEBS Journal*, 279:528–528, 2012.

[279] D. W. Otter, J. R. Medina, and J. K. Kalita. A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 32(2):604–624, 2021.

[280] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw. Geometry from a time series. *Phys. Rev. Lett.*, 45:712–716, Sep 1980.

[281] A. Pal, D. Karkhanis, M. Roberts, S. Dooley, A. Sundararajan, and S. Naidu. Giraffe: Adventures in expanding context lengths in LLMs. *arXiv preprint arXiv:2308.10882*, 2023.

[282] B. Pang, Y. Zhang, Y. Li, J. Cai, and C. Lu. Unsupervised visual representation learning by synchronous momentum grouping. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXX*, page 265–282, Berlin, Heidelberg, 2022. Springer-Verlag.

[283] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[284] G. Pérez-Hernández, F. Paul, T. Giorgino, G. D. Fabritiis, and F. Noé. Identification of slow molecular order parameters for markov model construction. *J. Chem. Phys.*, 139:015102, 2013.

[285] J. C. Phillips, D. J. Hardy, J. D. C. Maia, J. E. Stone, J. V. Ribeiro, R. C. Bernardi, R. Buch, G. Fiorin, J. Hénin, W. Jiang, R. McGreevy, M. C. R. Melo, B. K. Radak, R. D. Skeel, A. Singharoy, Y. Wang, B. Roux, A. Aksimentiev, Z. Luthey-Schulten, L. V. Kalé, K. Schulten, C. Chipot, and E. Tajkhorshid. Scalable molecular dynamics on CPU and GPU architectures with NAMD. 153(4):044130.

[286] S. Piana, K. Lindorff-Larsen, and D. Shaw. Protein folding kinetics and thermodynamics from atomistic simulation. *Proc. Natl. Acad. Sci. U.S.A.*, 109(44):17845–17850, 2012.

[287] L. Prechelt. *Early Stopping — But When?. In: Montavon, G., Orr, G.B., Müller, KR. (eds) Neural Networks: Tricks of the Trade*, volume 7700. Springer, 2012.

[288] X. Qi, K. Hou, T. Liu, Z. Yu, S. Hu, and W. Ou. From known to unknown: Knowledge-guided transformer for time-series sales forecasting in alibaba. *arXiv preprint arXiv:2109.08381*, 2021.

[289] R. Qian, T. Meng, B. Gong, M. Yang, H. Wang, S. Belongie, and Y. Cui. Spatiotemporal contrastive video representation learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6960–6970, Los Alamitos, CA, USA, jun 2021. IEEE Computer Society.

[290] C. Qin, N. Yu, C. Xing, S. Zhang, Z. Chen, S. Ermon, Y. Fu, C. Xiong, and R. Xu. Gluegen: Plug and play multi-modal encoders for x-to-image generation. *arXiv preprint arXiv:2303.10056*, 2023.

[291] Z. Qin, H. Yu, C. Wang, Y. Guo, Y. Peng, and K. Xu. Geometric transformer for fast and robust point cloud registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11143–11152, June 2022.

[292] Y. Raaj, H. Idrees, G. Hidalgo, and Y. Sheikh. Efficient online multi-person 2d pose tracking with recurrent spatio-temporal affinity fields. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4620–4628, 2019.

[293] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021.

[294] A. Radford and K. Narasimhan. Improving language understanding by generative pre-training. 2018.

[295] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

[296] T. Rainforth, A. R. Kosiorek, T. A. Le, C. J. Maddison, M. Igl, F. Wood, and Y. W. Teh. Tighter variational bounds are not necessarily better. In *International Conference on Machine Learning (ICML)*, volume 80, pages 4274–4282, 2018.

[297] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. In *International Conference on Learning Representations Workshop (ICLR Workshop)*, 2018.

[298] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[299] R. Ranftl, A. Bochkovskiy, and V. Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12179–12188, October 2021.

[300] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006.

[301] A. Raval, S. Piana, M. Eastwood, R. Dror, and D. Shaw. Refinement of protein structure homology models via long, all-atom molecular dynamics simulations. *Proteins*, 80(8):2071–2079, 2012.

[302] S. Röblitz and M. Weber. Fuzzy spectral clustering by PCCA+: application to markov state models and data classification. 7(2):147–179.

[303] S. J. Reddi, S. Kale, and S. Kumar. On the convergence of adam and beyond. In *International Conference on Learning Representations (ICLR)*, 2018.

[304] F. Regazzoni, D. Chapelle, and P. Moireau. Combining data assimilation and machine learning to build data-driven models for unknown long time dynamics—applications in cardiovascular modeling. *International Journal for Numerical Methods in Biomedical Engineering*, 37(7):e3471, 2021.

[305] J. M. L. Ribeiro, P. Bravo, Y. Wang, and P. Tiwary. Reweighted autoencoded variational Bayes for enhanced sampling (RAVE). *The Journal of Chemical Physics*, 149(7), 05 2018. 072301.

[306] J.-P. Richard. Time-delay systems: an overview of some recent advances and open problems. *Automatica*, 39(10):1667–1694, 2003.

[307] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. *arXiv preprint arXiv:2112.10752*, 2022.

[308] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[309] C. Rose and D. Murray. *Mathematical Statistics with MATHEMATICA*. Springer, 2002.

[310] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, 1(6088):318–362, 1986.

[311] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–538, 1986.

[312] D. E. Rumelhart and J. L. McClelland. *Learning Internal Representations by Error Propagation*. 1987.

[313] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[314] J.-P. Ryckaert, G. Ciccotti, and H. J. Berendsen. Numerical integration of the cartesian equations of motion of a system with constraints: molecular dynamics of n-alkanes. *Journal of Computational Physics*, 23(3):327–341, 1977.

[315] T. Ryder, A. Golightly, A. S. McGough, and D. Prangle. Black-box variational inference for stochastic differential equations. In *International Conference on Machine Learning (ICML)*, pages 4423–4432, 2018.

[316] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. G. Lopes, B. Karagol Ayan, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*, 2022.

[317] P. Sakov, F. Counillon, L. Bertino, K. A. Lisæter, P. Oke, and A. Korablev. TOPAZ4: An ocean-sea ice data assimilation system for the north atlantic and arctic. *Ocean Science*, 8(4):633–656, 2012.

[318] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry. How does batch normalization help optimization? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[319] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *CVPR*, 2020.

[320] T. Sauer, J. Yorke, and M. Casdagli. Embedology. *J Stat Phys*, 65, 1991.

[321] V. Saxena, J. Ba, and D. Hafner. Clockwork variational autoencoders. *arXiv preprint arXiv:2102.09532*, 2021.

[322] T. Schaul, S. Zhang, and Y. LeCun. No more pesky learning rates. *arXiv preprint arXiv:1206.1106*, 2012.

[323] M. K. Scherer, B. Trendelkamp-Schroer, F. Paul, G. Pérez-Hernández, M. Hoffmann, N. Plattner, C. Wehmeyer, J.-H. Prinz, and F. Noé. PyEMMA 2: A Software Package for Estimation, Validation, and Analysis of Markov Models. *Journal of Chemical Theory and Computation*, 11:5525–5542, 2015.

[324] F. Schneider. Archisound: Audio generation with diffusion. *arXiv preprint arXiv:2301.13267*, 2023.

[325] M. Schuster and K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[326] C. R. Schwantes and V. S. Pande. Improvements in markov state model construction reveal many non-native interactions in the folding of ntl9. *J. Chem. Theory Compute.*, 9(4):2000–2009, 2013.

[327] C. R. Schwantes and V. S. Pande. Modeling molecular kinetics with tica and the kernel trick. *J. Chem. Theory Compute.*, 11(600–608), 2015.

[328] P. Sfriso, A. Emperador, L. Orellana, A. Hospital, G. J. Lluis, and M. Orozco. Finding conformational transition pathways from discrete molecular dynamics simulations. *J. Chem. Theory Compute.*, 8(11):4707–4718, 2012.

[329] P. Sfriso, A. Hospital, A. Emperador, and M. Orozco. Exploration of conformational transition pathways from coarse-grained simulations. *Bioinformatics*, 29(16):1980–1986, 2013.

[330] Z. Shen, C. Lin, K. Liao, L. Nie, Z. Zheng, and Y. Zhao. Panoformer: Panorama transformer for indoor 360° depth estimation. In *European Conference on Computer Vision*, pages 195–211. Springer, 2022.

[331] F. K. Sheong, D.-A. Silva, L. Meng, Y. Zhao, and X. J. Huang. Automatic state partitioning for multibody systems (apm): An efficient algorithm for constructing markov state models to elucidate conformational dynamics of multibody systems. *J. Chem. Theory Compute.*, 11:17–27, 2015.

[332] W. Shi, J. Caballero, L. Theis, F. Huszar, A. Aitken, C. Ledig, and Z. Wang. Is the deconvolution layer the same as a convolutional layer? *arXiv preprint arXiv:1609.07009*, 2016.

[333] S. N. Shukla and B. Marlin. Multi-time attention networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2021.

[334] R. H. Shumway and D. S. Stoffer. *Time Series Analysis and Its Applications With R Examples*. Springer, 2017.

[335] N. Siddharth, B. Paige, J.-W. van de Meent, A. Desmaison, N. D. Goodman, P. Kohli, F. Wood, and P. H. Torr. Learning disentangled representations with semi-supervised deep generative models. In *NIPS*, 2017.

[336] A. Singer, R. Erban, I. G. Kevrekidis, and R. R. Coifman. Detecting intrinsic slow variables in stochastic dynamical systems by anisotropic diffusion maps. 106(38):16090–16095. Publisher: Proceedings of the National Academy of Sciences.

[337] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

[338] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[339] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning (ICML)*, pages 843–852, 2015.

[340] J. Stier and M. Granitzer. Deepgg: a deep graph generator. In *Advances in Intelligent Data Analysis XIX: 19th International Symposium on Intelligent Data Analysis, IDA 2021, Porto, Portugal, April 26–28, 2021, Proceedings*, page 325. Springer Nature.

[341] A. Stordal, H. Karlsen, G. Nævdal, et al. Bridging the ensemble kalman filter and particle filters: the adaptive gaussian mixture filter. *Comput Geosci*, 15:293–305, 2011.

[342] M. M. Sultan and V. S. Pande. tica-metadynamics: accelerating metadynamics by using kinetically selected collective variables. *J. Chem. Theory Compute.*, 13:2440–2447, 2017.

[343] M. M. Sultan, H. K. Wayment-Steele, and V. S. Pande. Transferable neural networks for enhanced sampling of protein dynamics. *arXiv preprint arXiv:1801.00636*, 2018.

[344] B. Sun, B. Li, S. Cai, Y. Yuan, and C. Zhang. Fsce: Few-shot object detection via contrastive proposal encoding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7352–7362, June 2021.

[345] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[346] N. Sviridova and K. Nakamura. Local noise sensitivity: Insight into the noise effect on chaotic dynamics. *Chaos*, 26(12), 2016.

[347] H. Takano and S. Miyashita. Relaxation modes in random spin systems. *Journal of the Physical Society of Japan*, 64(10):3688–3698, 1995.

[348] F. Takens. Detecting strange attractors in turbulence. In D. Rand and L.-S. Young, editors, *Dynamical Systems and Turbulence, Warwick 1980*, pages 366–381, Berlin, Heidelberg, 1981. Springer Berlin Heidelberg.

[349] B. Tang and D. S. Matteson. Probabilistic transformer for time series analysis. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 23592–23608. Curran Associates, Inc., 2021.

[350] D. M. Tax and R. P. Duin. Support vector data description. *Machine learning*, 54(1):45–65, 2004.

[351] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola. What makes for good views for contrastive learning? In *NeurIPS*, 2020.

[352] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop. Technical report, University of Tronto, 2012.

[353] S. Tipirneni and C. K. Reddy. Self-supervised transformer for sparse and irregularly sampled multivariate clinical time-series. *ACM Trans. Knowl. Discov. Data*, 16(6), jul 2022.

[354] M. K. Titsias and F. J. R. Ruiz. Unbiased implicit variational inference. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019.

[355] I. Tolstikhin, O. Bousquet, S. Gelly, and B. Schoelkopf. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018.

[356] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv: 2302.13971*, 2023.

[357] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv: 2307.09288*, 2023.

[358] G. A. Tribello, M. Bonomi, D. Branduardi, C. Camilloni, and G. Bussi. Plumed 2: New feathers for an old bird. *Computer Physics Communications*, 185(2):604–613, 2014.

[359] T. Tsujimura, K. Okusa, K. Maeno, and T. Kamakura. Statistical disturbance rejection of the background noise by microwave Doppler radar-modelization of an electric for noise. In *In Proceedings of the World Congress on Engineering and Computer Science*, 2012.

[360] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[361] S. Vaidya, P. Ambad, and S. Bhosle. Industry 4.0 – a glimpse. *Procedia Manufacturing*, 20:233–238, 2018.

[362] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[363] P. J. van Leeuwen. Nonlinear data assimilation in geosciences: an extremely efficient particle filter. *Quarterly Journal of the Royal Meteorological Society*, 136:1991–1999, 2010.

[364] Y. B. Varolgüneş, T. Bereau, and J. F. Rudzinski. Interpretable embeddings from molecular simulations using gaussian mixture variational autoencoders. *Machine Learning: Science and Technology*, 1(1):015012, 2020.

[365] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[366] S. Verma and Z.-L. Zhang. Learning universal graph neural network embeddings with aid of transfer learning. *arXiv preprint arXiv:1909.10086*, 2019.

[367] C. Vincent-Cuaz, R. Flamary, M. Corneli, T. Vayer, and N. Courty. Template based graph neural network with optimal transport distances. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 11800–11814. Curran Associates, Inc., 2022.

[368] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, 1989.

[369] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus. Regularization of neural networks using DropConnect. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1058–1066, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[370] D. Wang and P. Tiwary. State predictive information bottleneck. *The Journal of Chemical Physics*, 154(13):134111, 04 2021.

[371] D. Wang, Y. Wang, L. Evans, and P. Tiwary. From latent dynamics to meaningful representations. *arXiv preprint arXiv:2209.00905*, 2023.

[372] J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. In *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, volume 30, pages 283–298, 2008.

[373] P. Wang, K. Han, X.-S. Wei, L. Zhang, and L. Wang. Contrastive learning based hybrid networks for long-tailed image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 943–952, June 2021.

[374] P. Wang, S. Wang, J. Lin, S. Bai, X. Zhou, J. Zhou, X. Wang, and C. Zhou. One-peace: Exploring one general representation model toward unlimited modalities. *arXiv preprint arXiv:2305.11172*, 2023.

[375] W. Wang, J. Dai, Z. Chen, Z. Huang, Z. Li, X. Zhu, X. Hu, T. Lu, L. Lu, H. Li, X. Wang, and Y. Qiao. Internimage: Exploring large-scale vision foundation models with deformable convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14408–14419, June 2023.

[376] X. Wang, S. Yang, J. Zhang, M. Wang, J. Zhang, W. Yang, J. Huang, and X. Han. Transformer-based unsupervised contrastive learning for histopathological image classification. *Medical Image Analysis*, 81:102559, 2022.

[377] Y. Wang, Y. Hou, W. Che, and T. Liu. From static to dynamic word representations: a survey. *IJMLC*, 11:1611–1630, 2020.

[378] M. D. Ward, M. I. Zimmerman, S. Swamidass, and G. R. Bowman. Diffnets: Self-supervised deep learning to identify the mechanistic basis for biochemical differences between protein variants. *bioRxiv.org e-Print archive*, 2020.

[379] A. Warshel and M. Levitt. Theoretical studies of enzymic reactions – dielectric, electrostatic and steric stabilization of carbonium-ion in reaction of lysozyme. *J. Mol. Biol.*, 103(2):227–249, 1976.

[380] M. Watter, J. T. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Neural Information Processing Systems (NeurIPS)*, 2015.

[381] C. Wehmeyer and F. Noé. Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics. *The Journal of Chemical Physics*, 148(24):241703, 2018.

[382] Q. Wen, T. Zhou, C. Zhang, W. Chen, Z. Ma, J. Yan, and L. Sun. Transformers in time series: A survey. In *International Joint Conference on Artificial Intelligence(IJCAI)*, 2023.

[383] P. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[384] J. S. Whitaker and T. M. Hamill. Evaluating methods to account for system errors in ensemble data assimilation. *Monthly Weather Review*, 140(9):3078–3089, 2012.

[385] R. J. Williams and D. Zipser. *Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity*, page 433–486. L. Erlbaum Associates Inc., USA, 1995.

[386] G. Woo, C. Liu, D. Sahoo, A. Kumar, and S. C. H. Hoi. Etsformer: Exponential smoothing transformers for time-series forecasting. *arXiv preprint arxiv:2202.01381*, 2022.

[387] M. Wortsman, G. Ilharco, S. Y. Gadre, R. Roelofs, R. Gontijo-Lopes, A. S. Morcos, H. Namkoong, A. Farhadi, Y. Carmon, S. Kornblith, and L. Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR, 17–23 Jul 2022.

[388] H. Wu and F. Noé. Reaction coordinate flows for model reduction of molecular kinetics. *arXiv preprint arXiv:2309.05878*, 2023.

[389] H. Wu, J. Xu, J. Wang, and M. Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 22419–22430. Curran Associates, Inc., 2021.

[390] S. Wu, X. Xiao, Q. Ding, P. Zhao, Y. Wei, and J. Huang. Adversarial sparse transformer for time series forecasting. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17105–17115. Curran Associates, Inc., 2020.

[391] Y. Wu and K. He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[392] Z. Xia, X. Pan, S. Song, L. E. Li, and G. Huang. Vision transformer with deformable attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4794–4803, June 2022.

[393] E. Xie, J. Ding, W. Wang, X. Zhan, H. Xu, P. Sun, Z. Li, and P. Luo. Detco: Unsupervised contrastive learning for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8392–8401, October 2021.

[394] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 12077–12090. Curran Associates, Inc., 2021.

[395] H. xin Tian, X. jun Liu, and M. Han. An outliers detection method of time series data for soft sensor modeling. In *Chinese Control and Decision Conference*, pages 3918–3922, 2016.

[396] G. Xu, Y. Meng, X. Qiu, Z. Yu, and X. Wu. Sentiment analysis of comment texts based on bilstm. *IEEE Access*, 7:51522–51532, 2019.

[397] X. Xu and Y. Chen. Deep switching state space model (ds3m) for nonlinear time series forecasting with regime switching. *arXiv preprint arXiv:2106.02329*, 2021.

[398] Y. Xu, J. Zhang, Q. Zhang, and D. Tao. ViTPose: Simple vision transformer baselines for human pose estimation. In *Advances in Neural Information Processing Systems*, 2022.

[399] Y. Xu, J. Zhang, Q. Zhang, and D. Tao. Vitpose+: Vision transformer foundation model for generic body pose estimation. *arXiv preprint arXiv:2212.04246*, 2022.

[400] Y. Xu, Q. Zhang, J. Zhang, and D. Tao. Vitae: Vision transformer advanced by exploring intrinsic inductive bias. *Advances in Neural Information Processing Systems*, 34, 2021.

[401] M. Yamada, H. Kim, K. Miyoshi, T. Iwata, and H. Yamakawa. Disentangled representations for sequence data using information bottleneck principle. In S. J. Pan and M. Sugiyama, editors, *Proceedings of The 12th Asian Conference on Machine Learning*, volume 129 of *Proceedings of Machine Learning Research*, pages 305–320. PMLR, 18–20 Nov 2020.

[402] S. Yang, Z. Quan, M. Nie, and W. Yang. Transpose: Keypoint localization via transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11802–11812, October 2021.

[403] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge. Learning physics constrained dynamics using autoencoders. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 17157–17172. Curran Associates, Inc., 2022.

[404] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. XLNet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[405] G. H. Yann LeCun, Yoshua Bengio. Deep learning. *Nature*, 521(7553):436–444, 2015.

[406] C. Yildiz, M. Heinonen, and H. Lähdesmäki. ODE2VAE: Deep generative second order ODEs with Bayesian neural networks. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[407] M. Yin and M. Zhou. Semi-implicit variational inference. In *International Conference on Machine Learning (ICML)*, 2018.

[408] L. Yingzhen and S. Mandt. Disentangled sequential autoencoder. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5670–5679. PMLR, 10–15 Jul 2018.

[409] X. Yu, Y. Xue, L. Zhang, L. Wang, T. Liu, and D. Zhu. Noisynn: Exploring the influence of information entropy change in learning systems. *arXiv preprint arXiv: 2309.10625*, 2023.

[410] I. Yun, H.-J. Lee, and C. E. Rhee. Improving 360 monocular depth estimation via non-local dense prediction transformer and joint supervised and self-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3224–3233, 2022.

[411] M. D. Zeiler. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[412] M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv: 1301.3557*, 2013.

[413] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12104–12113, June 2022.

[414] C. Zhang, J. Butepage, H. Kjellstrom, and S. Mandt. Advances in variational inference. 2017.

[415] D. Zhang, F. Huang, S. Liu, X. Wang, and Z. Jin. Swinfir: Revisiting the swinir with fast fourier convolution and improved training for image super-resolution. *arXiv preprint arXiv:2208.11247*, 2022.

[416] F. Zhang, C. Snyder, and J. Sun. Impacts of initial estimate and observation availability on convective-scale data assimilation with an ensemble Kalman filter. *Monthly Weather Review*, 132(5):1238–1253, 2004.

[417] G. Zhang, C. Wang, B. Xu, and R. Grosse. Three mechanisms of weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

[418] L. Zhang, A. Rao, and M. Agrawala. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*, 2023.

[419] O. Zhang, R.-S. Lin, and Y. Gou. Optimal transport based generative autoencoders. *arXiv preprint arXiv:1910.07636*, 2019.

[420] Q. Zhang, Y. Xu, J. Zhang, and D. Tao. Vitaev2: Vision transformer advanced by exploring inductive bias for image recognition and beyond. *arXiv preprint arXiv:2202.10108*, 2022.

[421] Z. Zhang, J. Bu, M. Ester, J. Zhang, Z. Li, C. Yao, H. Dai, Z. Yu, and C. Wang. Hierarchical multi-view graph pooling with structure learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):545–559, 2023.

[422] L. Zhao and L. Wang. Price trend prediction of stock market using outlier data mining algorithm. In *the 5th IEEE International Conference on Big Data and Cloud Computing*, pages 93–98, 2015.

[423] S. Zhao, Z. Liu, J. Lin, J.-Y. Zhu, and S. Han. Differentiable augmentation for data-efficient gan training. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 7559–7570. Curran Associates, Inc., 2020.

[424] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[425] G. Zhou, P. Luo, R. Cao, F. Lin, and B. C. abd Qing He. Mechanism-aware neural machine for dialogue response generation. In *AAAI International Conference on Artificial Intelligence (AAAI))*, 2017.

[426] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(12):11106–11115, May 2021.

[427] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proc. 39th International Conference on Machine Learning (ICML 2022)*, 2022.

[428] M. Zhu, P. J. van Leeuwen, and J. Amezcua. Implicit equal-weights particle filter. *Quarterly Journal of the Royal Meteorological Society*, 142(698):1904–1919, 2016.

[429] Y. Zhu, M. Min, A. Kadav, and H. Graf. S3vae: Self-supervised sequential vae for representation disentanglement and data generation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6537–6546, Los Alamitos, CA, USA, jun 2020. IEEE Computer Society.