UNIVERSITY OF CALIFORNIA

Los Angeles

Unsupervised Learning and Understanding of Deep Generative Models

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Statistics

by

Tian Han

2019

ABSTRACT OF THE DISSERTATION

Unsupervised Learning and Understanding of Deep Generative Models

by

Tian Han
Doctor of Philosophy in Statistics
University of California, Los Angeles, 2019
Professor Ying Nian Wu, Chair

Probabilistic generative models, especially ones that are parametrized by convolutional neural network (ConvNet), are compact representation tools towards knowledge understanding and can be crucial in statistics as well as artificial intelligence. The generator model and the energy-based model are two notable examples. Yet the learning and understanding of such models can be challenging because of the high dimensionality of the input and the high non-linearity of the network. In this dissertation, we pay particular attention to the generator model, and study its learning algorithm and the behavior of the learned model. We also develop the joint learning scheme for both the generator model and the energy-based model.

To learn the generator model, we view it in the lens of non-linear generalization of factor analysis and propose an alternating back-propagation algorithm for learning. The alternating back-propagation algorithm iterates the following two steps: (1) *Inferential back-propagation*, which infers the latent factors by Langevin dynamics or gradient descent. (2) *Learning back-propagation*, which updates the parameters given the inferred latent factors by gradient descent. The gradient computations in both steps are powered by back-propagation, and they share most of their code in common. We show that the alternating back-propagation algorithm can learn realistic generator models of natural images, video sequences, and sounds. Moreover, it can also be used to learn from incomplete or indirect training data.

The generator model can be naturally extended for multi-view representation learning where we build separate generator model for each domain but share their latent variables.

The proposed multi-view generator model can be easily learned through alternating back-propagation. Our experiments show that the proposed method is powerful in both generation, prediction and recognition. Specifically, we demonstrate our model can accurately rotate and complete faces as well as predict missing modalities. We also show our model can achieve state-of-art or competitive recognition performance through quantitative comparisons.

Further, the generator model can be jointly learned with the energy-based model. We propose the probabilistic framework, called divergence triangle, as a compact and symmetric (anti-symmetric) objective function that seamlessly integrates variational learning, adversarial learning, wake-sleep algorithm, and contrastive divergence. This unification makes the processes of sampling, inference, energy evaluation readily available without the need for costly Markov chain Monte Carlo methods. Our experiments demonstrate that the divergence triangle is capable of learning (1) an energy-based model with well-formed energy landscape, (2) direct sampling in the form of a generator model, and (3) feed-forward inference that faithfully reconstructs observed as well as synthesized data. The divergence triangle is also a robust training method that can learn from incomplete data.

The last but not the least, we take the inspiration from recent discovery in neuroscience which states that for the face stimuli generated by a pre-trained active appearance model (AAM), the responses of neurons in the selected areas of the primate brain exhibit strong linear relationship with the shape and appearance variables of the AAM that generates the face stimuli. We show that this behavior can be replicated by a generator model. Specifically, we learn the generator model from the face images generated by a pre-trained AAM model using variational auto-encoder, and we show that the inferred latent variables of the learned generator model have strong linear relationship with the shape and appearance variables of the AAM model that generates the face images. Unlike the AAM model that has an explicit shape model where the shape variables generate the landmarks, the generator model has no such shape model and shape variables. Yet the generator model can learn the shape knowledge in the sense that some of the latent variables of the learned generator network capture the shape variations in the face images generated by AAM.

The dissertation of Tian Han is approved.

Luminita Aura Vese

Frederic Paik Schoenberg

Hongquan Xu

Ying Nian Wu, Committee Chair

University of California, Los Angeles

2019

To my family . . .

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Firstly, I would like to express my deepest gratitude to my advisor, prof. Ying Nian Wu, who guide me through the Ph.D study and enlighten me the beauty of research. Prof. Wu is very thoughtful and is truly a deep thinker. During my Ph.D study, I'm always amazed by his insights toward many hard-core learning problems and the most important lesson I learned is that we should do the research in principled and elegant way. The gratitude also goes to my thesis committee members: prof. Hongquan Xu, prof. Frederic Paik Schoenberg and prof. Luminita Aura Vese for the consistent help and support.

Many gratitude goes to prof. Song-Chun Zhu who took me to UCLA in the first place and taught me to keep strong and solve big problems, prof. Long Quan who give me valuable career and research advices and prof. Tianfu Wu who give me useful support for my academic career.

I also feel very fortunate to work with many excellent colleagues in UCLA. I truly learned a lot from the discussion and collaboration with them. Specially thanks goes to: Dr. Lu Yang, Dr. Xiaohan Xie, Dr. Xinglei Xing, Erik Nijkamp, Siyuan Huang, Jiawen Wu, Xiaolin Fang, Shuai Zhu, Bo Pang, Yuan Tao ...

Most importantly, none of this would be possible without the constant love and support of my family. Specially, Janice Kuang, thank you for being by my side all the time, encouraging and supporting me through the darkness. I'm so grateful to have you in my life.

VITA

2010	B.S. (Applied Mathematics), Hefei University of Technology.
2013	M.Phil. (Computer Science), The Hong Kong University of Science and Technology (HKUST).
2014–2019	Teaching Assistant, Statistics Department, UCLA.
2016–2019	Research Assistant, Statistics Department, UCLA.

PUBLICATIONS

[1] Tian Han*, Lu Yang*, Song-Chun Zhu and Ying Nian Wu. *“Alternating Back-Propagation for Generator Network”*. AAAI, 2017

[2] Tian Han, Lu Yang, Jiawen Wu, Xianglei Xing and Ying Nian Wu. *“Learning Generator Networks for Dynamic Patterns”*. WACV, 2019

[3] Tian Han*, Xianglei Xing* and Yian Nian Wu. “Learning Multi-view Generator Network for Shared Representation”. ICPR, 2018

[4] Tian Han*, Erik Nijkamp*, Xiaolin Fang, Mitch Hill, Song-Chun Zhu and Yian Nian Wu. “Divergence Triangle for Joint Training of Generator Model, Energy-based Model, and Inferential Model”. CVPR, 2019

[5] Tian Han, Jiawen Wu, Ying Nian Wu. *“Replicating Active Appearance Model by Generator Network”*. IJCAI, 2018

[6] Xianglei Xing, Tian Han, Ruiqi Gao, Song-Chun Zhu and Yian Nian Wu. “Unsupervised Disentangling of Appearance and Geometry by Deformable Generator Network”. CVPR, 2019

[7] Ying Nian Wu, Ruiqi Gao, Tian Han and Song-Chun Zhu. “A Tale of Three Probabilistic Families: Discriminative, Descriptive and Generative Models”. QAM, 2018

CHAPTER 1

Introduction

1.1 Background and Motivation

Initially developed by Grenander in the 1970s, the pattern theory [Gre70, GM07] is a unified mathematical framework for representing, learning and recognizing patterns that arise in science and engineering. The objects in pattern theory are usually of high complexity or dimensionality, defined in terms of the constituent elements and the bonds between them. The patterns of these objects are characterized by both the algebraic structures governed by local and global rules, as well as the probability distributions of the associated random variables. Such a framework encompasses most of the probability models in various disciplines. Mumford [MD10] later advocated such probabilistic theoretical framework for computer vision, so that learning and inference can be based on probability models.

The discriminative model is one of the most successful developed models in the past decade. A discriminative model is in the form of a classifier and specifies the conditional probability of the output class label given the input signal. Fueled by the large scale datasets, such as ImageNet [DDS09] and LSUN [YSZ15], and powered by the graphical processing units (GPUs), discriminative models that are parametrized by convolutional neural networks have achieved state-of-the-art results in the tasks such as classification and recognition. Such a model can be learned in the supervised setting where a training dataset of input signals and the corresponding output labels are provided. Discriminative models have been widely used in the areas of artificial intelligence including computer vision, natural language processing, speech recognition etc.

Despite its success, the discriminative model only specifies the conditional probability and can only tell apart between two objects. It does not provide any clues about how objects are created and what they look like. Such information is essential towards knowledge understanding as in the famous quote from Richard Feynman ——"If I cannot create, I do not understand", and is the primary focus of generative models. A generative model directly specifies the probability distribution on the input signal and can be considered as a compact representation of knowledge. Once learned, they have potentials for solving a wide range of problems including density estimation, data completion, representation learning as well as downstream inferences.

Despite its generality, learning and understanding of probabilistic generative models remain challenging tasks. Unlike discriminative model learning, generative models are usually learned in the unsupervised setting where the training dataset only consists of input signals without the corresponding output labels. Both the input and the output are of extremely high dimensionality (e.g., $\sim$10000 dims for image) and their hidden structures are also of high complexities. Models parametrized by convolutional neural networks, i.e., deep probabilistic models, can be effective because of their approximation capacity, but learning and understanding such models can be even more difficult because of the highly non-linear and entangled nature of deep networks. In this dissertation, we study how to effectively and efficiently learn deep probabilistic generative models and investigate their underlying behaviors.

1.2 Two probabilistic Generative Models

Deep probabilistic generative models are powerful frameworks for representing complex input data distributions. The goal is to build rich and flexible models to fit complex, multi-modal input data distributions as well as to be able to generate samples with high realism. The family of generative models can be roughly divided into two classes: the first class is the energy-based model (a.k.a undirected graphical model) and the second class is the generator model (a.k.a directed graphical model). In this dissertation, we study and focus on the models that are parametrized by convolutional neural network (ConvNet) which consists of multilayer linear convolution operations followed by non-linear activation functions (see

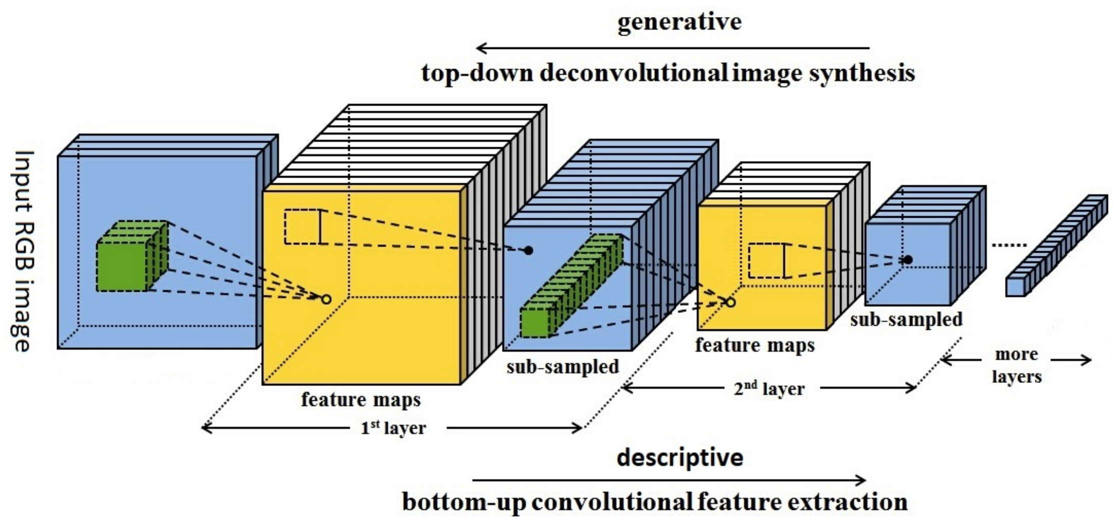

Figure 1.1: Convolutional neural network (ConvNet) mapping. The bottom-up mapping starts from input image and mainly used for feature/statistics extraction. The top-down mapping builds upon latent variable (or categories) and used for generation.

Figure 1.1). We will briefly describe two type of models in this section and leave more detailed analysis in the following chapters. The models described in this section serve as the basic building blocks of our thesis work.

$$\begin{array}{cc} \text{Top-down mapping} & \text{Bottom-up mapping} \\ \text{latent variable } z & \text{energy } -f_\alpha(x) \\ \Downarrow & \Uparrow \\ \text{signal } x \approx g_\theta(z) & \text{signal } x \\ \text{(a) Generator model} & \text{(b) Energy-based model} \end{array}$$

1.2.1 The Energy-based Model

The energy-based model specifies the following probability model on the observed data:

$$\pi_\alpha(x) = \frac{1}{Z(\alpha)} \exp\left[f_\alpha(x)\right], \tag{1.1}$$

where $-f_\alpha(x)$ defines the energy of x, and a low energy x is assigned a high probability, and $Z(\alpha)$ is the normalizing constant. If $f_\alpha(x)$ is linear in α, the model becomes the familiar exponential family model in statistics or the Gibbs distribution in statistical physics. The energy function $f_\alpha(x)$ can be generalized to hierarchical form with multiple layers of features using ConvNet with parameters α and can be computed in a bottom-up manner. In the literature, such model is also called descriptive model [Zhu03]. See Figure 1.1 and above diagram (b) for illustration.

The energy-based model can be interpreted as associative memory as observed by Hopfield [Hop82]. Take image space for example, if π_α is an accurate model of the given dataset, then images whose appearance is consistent with the observed data should have low energy while images which differ greatly from the observed data should have high energy. The major modes of energy function represents the most typical image appearances for observed data. The energy-based model π_α defines explicit log-likelihood of input x via $f_\alpha(x)$, even though $Z(\alpha)$ is intractable. It has no latent variables involved and therefore has no explicit inference.

The energy-based model can be learned through maximum likelihood estimation (MLE) which aims at matching the statistical features between observed data and the data from the model distribution π_α. The learning proceeds by first sampling/synthesising data from the current model, then updating the model (i.e., energy function) in order to decrease the energy for observed data and increase the energy for model samples. It is essentially a "analysis-by-synthesis" learning scheme. However, the learning can be challenging since it is difficult to sample from model distribution π_α due to the intractable $Z(\alpha)$. Therefore, Markov Chain Monte Carlo (MCMC) is needed to get approximate samples from the model [LZW16, XLZ16].

1.2.2 The Latent Variable Model

Instead of explicitly specifying probability distribution on the observed data, the latent variable model generates the data through latent factors z in the following way:

$$z \sim \mathrm{N}(0, I_d),\ x = g_\theta(z) + \epsilon,\ \epsilon \sim \mathrm{N}(0, \sigma^2 I_D) \tag{1.2}$$

where g_θ is a mapping function that maps the d-dimensional latent vector z to the D-dimensional signal x. If $g_\theta(z)$ is linear in θ, then the model becomes the factor analysis which is a prototype model in unsupervised learning of distributed representations. Such mapping can be generalized to non-linear form using ConvNet with parameters θ and essentially a top-down mapping (see Figure 1.1 and above diagram (a)). The ConvNet parametrization version of the model is also known as *the generator model* in the literature [GPM14, RMC15, KW13]. The generator model specifies the complete data distribution $p_\theta(z, x)$ through prior distribution $p(z)$ and the conditional distribution $p_\theta(x|z)$, i.e., $p_\theta(z, x) = p(z)p_\theta(x|z)$. Therefore, the observed data distribution is implicitly defined by integrating out the latent variables, i.e., $p_\theta(x) = \int p_\theta(z, x)dz$.

The generator model is a fundamental representation of knowledge, and it has the following properties: (1) *Analysis*: The model disentangles the variations in the observed signals into independent variations of latent factors. (2) *Synthesis*: The model can synthesize new signals by sampling the factors from the known prior distribution and transforming the factors into the signal. (3) *Embedding*: The model embeds the high-dimensional non-Euclidean manifold formed by the observed signals into the low-dimensional Euclidean space of the latent factors, so that linear interpolation in the low-dimensional factor space results in non-linear interpolation in the data space.

In the unsupervised setting, latent variables z are unknown and need to be inferred during learning. The general learning scheme follows "analysis-by-inference" mechanism where the latent variables are inferred using the current model and the model is updated using the obtained latent variables. The generator model can be learned directly using maximum likelihood, however, the learning can be challenging since the inference is based on the intractable posterior distribution $p_\theta(z|x)$ of the generator model.

1.3 Overview of the Dissertation

The research work in this dissertation aims at learning and understanding deep probabilistic generative models. Particularly, for learning, we focus on the unsupervised setting where only observed data is available and we discuss the model and learning algorithms for different data modalities (e.g., image, video etc). For understanding, we focus on the generator model and examine its behavior through latent factors. We overview the research works and their contributions below.

In chapter 2, we work on unsupervised learning of the generator model. The existing training algorithms for such models involve an assisting network with a separate set of parameters, and they cannot perform accurate inference on the latent factors. We have developed an efficient algorithm called **Alternating Back-propagation (ABP)** that trains the generator model without any assisting network and infers the latent factors explicitly and accurately. Precisely, it iterates the following two steps: (1) inferential step where the latent factors are inferred through Langevin dynamics based on exact likelihood, and (2) learning step where the generator model is learned by gradient descent based on the inferred latent factors. The gradient computations in the two steps can be powered by back-propagation and share the same chain rule, which makes the proposed algorithm very efficient. Experiments show that our approach can learn realistic generator model on various data types, including audio signals, texture/object images and video sequences. It could also be successfully applied to data completion problems.

In chapter 3, we extend the original generator model to multi-view data. The proposed multi-view generator model uses separate generator for each domain but share their underlying latent factors. The shared latent factors can encode common knowledge between different domains while the separate generator model can encode domain specific information. The model is learned using extended alternating back-propagation, and can be effective in multi-view learning (e.g. multi-view gait generation and recognition), as well as audio-video common knowledge extraction.

In chapter 4, we consider to jointly learn the generator model with the energy-based model. Separate learning of the two models are in general difficult for two reasons: (1) The energy-based model is learned through an "analysis-by-synthesis" manner in which the negative samples need to be draw from the model distribution, because the underlying expectation is analytically intractable. (2) A generator model is learned through an "analysis-by-inference" manner in which latent factors need to be inferred which requires integrating out latent variables that is also analytical intractable. Learning such models usually requires expensive MCMC sampling, and few existing work has been proposed to efficiently and effectively bridge these two classes of models. We propose the **divergence triangle** scheme to jointly train these two models. While using the inference model, we make the joint model learning analytically tractable. The learning of the generator model, the energy-based model and the inferential model can be seamlessly integrated within an unified probabilistic framework. It can also be seen as a probabilistic framework to unify the variational learning and adversarial learning schemes. Experiments showed that the proposed work can learn a well-behaved energy-based model with meaningful energy landscape, a realistic generator model with sharp sample quality, and an accurate inferential model with faithful reconstruction.

In chapter 5, we study the behavior of the generator model. The work in this chapter is inspired by the recent *Cell* paper [CT17] which shows how specific neurons in the primate brain respond to and encode the given face stimuli. The face stimuli used in [CT17] are generated by a pre-trained Active Appearance Model (AAM) which is explainable and contains explicit shape and appearance variables. Interestingly, the work in [CT17] on the specific neuron responses exhibits strong linear relationship with shape and appearance variables of AAM. In our work, we have learnt the generator model by variational autoencoder (VAE) using the images generated by pre-trained AAM. The most important finding is that latent factors in the learned generator model exhibit strong linear relationship with the AAM shape/appearance variables. This finding is non-trivial and surprising, because when presented with only synthesized face stimuli, the VAE training of the highly non-linear generator network can automatically learn the code that is linearly related to the underlying AAM code that generates the given face stimuli. The experiments also show that we are able to linearly decode face images and separate the shape and appearance parts given the inferred latent factors from the generator model. Moreover the AAM model can be fully replicated by the generator model.

1.4 Prior Art

In this section, we review some of the representative works on unsupervised learning and understanding of probabilistic generative models that are most related to our work.

1.4.1 Maximum Likelihood Learning for Generative Model

The maximum likelihood learning of the energy-based model requires expectation with respect to the current model, while the maximum likelihood learning of the generator model requires expectation with respect to the posterior distribution of the latent variables. Both expectations can be approximated by MCMC, such as Gibbs sampling [GG84], Langevin dynamics, or Hamiltonian Monte Carlo (HMC) [Nea11]. [LZW16, XLZ16] used Langevin dynamics for learning the energy-based models, and we propose to use Langevin dynamics for learning the generator model in this dissertation. In both cases, MCMC sampling introduces an inner loop in the training procedure, posing a computational expense.

1.4.2 The Energy-based Model

An early version of the energy-based model is the FRAME (Filters, Random field, And Maximum Entropy) model [ZWM97, WZL00]. [ZM98] used gradient-based method such as Langevin dynamics to sample from the model. Other forms of energy-based model also include product of experts [Hin02a], product of t-distribution [WOH03], field of experts [RB05] as well as restricted Boltzmann machine (RBM) [Hin10]. The energy-based models are also called descriptive models as in [Zhu03]. [LZW16, XLZ16] generalized such models to deep versions.

For learning the energy-based model [LCH06], to reduce the computational cost of MCMC sampling, contrastive divergence (CD) [Hin02a] initializes a finite step MCMC from the observed data. The resulting learning algorithm follows the gradient of the difference between two Kullback-Leibler divergences, thus the name contrastive divergence. In this dissertation, we shall use the term "contrastive divergence" in a more general sense than [Hin02a]. Persistent contrastive divergence [Tie08] initializes MCMC sampling from the samples of the previous learning iteration.

Generalizing [Tu07], [JLT17] developed an introspective learning method where the energy function is discriminatively learned, and the energy-based model is both a generative model and a discriminative model.

1.4.3 The Generator Model

For learning the generator model, the variational auto-encoder (VAE) [KW13, RMW14a, MG14] approximates the posterior distribution of the latent variables by an explicit inference model. In VAE, the inference model is learned jointly with the generator model from the observed data. A precursor of VAE is the wake-sleep algorithm [HDF95], where the inference model is learned from the dream data generated by the generator model in the sleep phase.

The generator model can also be learned jointly with a discriminator model, as in the generative adversarial networks (GAN) [GPM14], as well as deep convolutional GAN (DC-GAN) [RMC15], energy-based GAN (EB-GAN) [ZML16], Wasserstein GAN (WGAN) [ACB17]. GAN does not involve an inference model.

1.4.4 Joint Learning of Probabilistic Models

The generator model can be learned jointly with an energy-based model [KB16, DAB17]. We can interpret the learning scheme as an adversarial version of contrastive divergence. While in GAN, the discriminator model eventually becomes a confused one, in the joint learning of the generator model and the energy-based model, the learned energy-based model becomes a well-defined probability distribution on the observed data. The joint learning bares some similarity to WGAN, but unlike WGAN, the joint learning involves two complementary probability distributions.

To bridge the gap between the generator and the energy-based model, the cooperative learning method of [XLG18] introduces finite-step MCMC sampling of the energy-based model with the MCMC initialized from the samples generated by the generator model. Such finite-step MCMC produces synthesized examples closer to the energy-based model, and the generator model can learn from how the finite-step MCMC revises its initial samples.

Adversarially learned inference (ALI) [DBP16, DKD16] combines the learning of the generator model and inference model in an adversarial framework. ALI can be improved by adding conditional entropy regularization, resulting in the ALICE [LLC17] model. The recently proposed method [CDP18] shares the same spirit. They lack an energy-based model on observed data.

1.4.5 Other Forms

Other forms of probabilistic generative models also exist in the literature. Notably, deep belief net (DBN) [HOT06] uses the directed sigmoid belief net, with the RBM on the top layer, to model observed data. Deep Boltzmann machine (DBM) [SH09] instead uses the undirected connection for the modeling. Flow-based generative models, on the other hand, explicitly learn the data distribution through sequence of invertible transformations, examples include recently proposed NICE [DKB14], Real-NVP [DSB16] and GLOW model [KD18].

1.4.6 Disentangled Representation

Disentangled representations behind the observed data[BCV13, MZZ16] is fundamental towards understanding and controlling modern deep models. Among others, deep generative models, e.g., the generator model, have shown great promise in learning representation of images in recent years. However, the learned representation, i.e., latent factor z, is often entangled and not interpretable. We review some of the recent attempts towards studying and learning disentangled latent factors below.

On the one hand, disentangled representation can be obtained implicitly through generative learning. InfoGAN [CDH16], a representative of the GAN family, is proposed under the principle of maximization of the mutual information between the observations and a subset of latent factors. However, its disentangling performance is sensitive to the choice of the prior and the number of latent vectors. β-VAE [HMP16], from the VAE family, learns disentangled representations by utilizing a VAE objective but with strong penalty on prior latent distribution to make it to be independent as much as possible.

On the other hand, disentangled representation can also be obtained explicitly through model. Active Appearance Models (AAM) [CET01, KTP17] learn the appearance and geometric information explicitly. Particularly, AAM uses principal component analysis (PCA) to obtain the independent sets of appearance and geometry latent factors.

CHAPTER 2

The Generator Model and Learning

The generator model described in chapter 1.2.2 is a non-linear generalization of the factor analysis. We shall first describe the basic factor analysis model based on which the non-linear version is developed as the generator model, then we shall present the alternating back-propagation algorithm for model learning. Then we analyze the algorithm and present the experiments both qualitatively and quantitatively on various tasks.

2.1 Factor Analysis and Beyond

Factor analysis is a prototype model in unsupervised learning of distributed representations. Let X be a D-dimensional observed data vector, such as an image. Let Z be the d-dimensional vector of continuous latent factors, $Z = (z_k, k = 1, ..., d)$. The traditional factor analysis model is $X = WZ + \epsilon$, where W is $D \times d$ matrix, and ϵ is a D-dimensional error vector or the observational noise. We assume that $Z \sim \mathrm{N}(0, I_d)$, where I_d stands for the d-dimensional identity matrix. We also assume that $\epsilon \sim \mathrm{N}(0, \sigma^2 I_D)$, i.e., the observational errors are Gaussian white noises. There are three perspectives to view W. (1) *Basis vectors.* Write $W = (W_1, ..., W_d)$, where each W_k is a D-dimensional column vector. Then $X = \sum_{k=1}^{d} z_k W_k + \epsilon$, i.e., W_k are the basis vectors and z_k are the coefficients. (2) *Loading matrix.* Write $W = (w_1, ..., w_D)^\top$, where $w_j^\top$ is the j-th row of W. Then $x_j = \langle w_j, Z \rangle + \epsilon_j$, where x_j and ϵ_j are the j-th components of X and ϵ respectively. Each x_j is a loading of the d factors where w_j is a vector of loading weights, indicating which factors are important for determining x_j. W is called the loading matrix. (3) *Matrix factorization.* Suppose we observe $\mathbf{X} = (x_1, ..., x_n)$, whose factors are $\mathbf{Z} = (z_1, ..., z_n)$, then $\mathbf{X} \approx W\mathbf{Z}$.

The factor analysis model can be learned by the Rubin-Thayer EM algorithm, which involves alternating regressions of Z on X in the E-step and of X on Z in the M-step, with both steps powered by the sweep operator [RT82, LRW98].

The factor analysis model is the prototype of many subsequent models that generalize the prior model of Z.

- *Independent component analysis* [OF97], $d > D$, and Z is assumed to be a redundant but sparse vector, i.e., only a small number of z_k are non-zero or significantly different from zero.
- *Sparse coding* [OF97], $d > D$, and Z is assumed to be a redundant but sparse vector, i.e., only a small number of z_k are non-zero or significantly different from zero.
- *Non-negative matrix factorization* [LS01], it is assumed that $z_k \geq 0$.
- *Recommender system* [KBV09], Z is a vector of a customer's desires in different aspects, and w_j is a vector of product j's desirabilities in these aspects.

2.2 The Generator Model as Non-linear Factor Analysis

2.2.1 ConNet Mapping

In addition to generalizing the prior model of the latent factors Z, we can also generalize the mapping from Z to X. In this dissertation, we consider the generator model [GPM14] that retains the assumptions that $d < D$, $Z \sim \mathrm{N}(0, I_d)$, and $\epsilon \sim \mathrm{N}(0, \sigma^2 I_D)$ as in traditional factor analysis, but generalizes the linear mapping WZ to a non-linear mapping $g_\theta(Z)$, where g is a ConvNet, and θ collects all the connection weights and bias terms of the ConvNet. Then the model becomes (recall Eqn. 1.2 in chapter 1.2.2):

$$\begin{aligned} &X = g_\theta(Z) + \epsilon, \\ &Z \sim \mathrm{N}(0, I_d),\ \epsilon \sim \mathrm{N}(0, \sigma^2 I_D),\ d < D. \end{aligned} \tag{2.1}$$

The reconstruction error is $||X - g_\theta(Z)||^2$. We may assume more sophisticated models for ϵ, such as colored noise or non-Gaussian texture. If X is binary, we can emit X by a probability map $P = 1/[1 + \exp(-g_\theta(Z)))]$, where the sigmoid transformation and Bernoulli sampling are carried out pixel-wise. If X is multi-level, we may assume multinomial logistic emission model or some ordinal emission model.

Although $g_\theta(Z)$ can be any non-linear mapping, the ConvNet parametrization of $g_\theta(Z)$ makes it particularly close to the original factor analysis. Specifically, we can write the top-down ConvNet as follows:

$$Z^{(l-1)} = g_l(W_l Z^{(l)} + b_l), \tag{2.2}$$

where g_l is element-wise non-linearity at layer l, W_l is the matrix of connection weights, b_l is the vector of bias terms at layer l, and $\theta = (W_l, b_l, l = 1, ..., L)$. $Z^{(0)} = g_\theta(Z)$, and $Z^{(L)} = Z$. The top-down ConvNet (2.2) can be considered a recursion of the original factor analysis model, where the factors at the layer $l - 1$ are obtained by the linear superposition of the basis vectors or basis functions that are column vectors of W_l, with the factors at the layer l serving as the coefficients of the linear superposition. In the case of ConvNet, the basis functions are shift-invariant versions of one another, like wavelets.

2.2.2 ReLU and Piecewise Factor Analysis

The element-wise non-linearity g_l in modern ConvNet is usually the two-piece linearity, such as rectified linear unit (ReLU) [KSH12] or the leaky ReLU [MHN13, XWC15]. Each ReLU unit corresponds to a binary switch. For the case of non-leaky ReLU, following the analysis of [PMB13], we can write $Z^{(l-1)} = \delta_l(W_l Z^{(l)} + b_l)$, where $\delta_l = \mathrm{d}iag(1(W_l Z^{(l)} + b_l > 0))$ is a diagonal matrix, $1()$ is an element-wise indicator function. For the case of leaky ReLU, the 0 values on the diagonal are replaced by a leaking factor (e.g., 0.2).

$\delta = (\delta_l, l = 1, ..., L)$ forms a classification of Z according to the network θ. Specifically, the factor space of Z is divided into a large number of pieces by the hyperplanes $W_l Z^{(l)} + b_l = 0$, and each piece is indexed by an instantiation of δ. We can write $\delta = \delta(Z; \theta)$ to make explicit its dependence on Z and θ. On the piece indexed by δ, $g_\theta(Z) = W_\delta Z + b_\delta$.

Assuming $b_l = 0, \forall l$, for simplicity, we have $W_\delta = \delta_1 W_1 ... \delta_L W_L$. Thus each piece defined by $\delta = \delta(Z; \theta)$ corresponds to a linear factor analysis $X = W_\delta Z + \epsilon$, whose basis W_δ is a multiplicative recomposition of the basis functions at multiple layers $(W_l, l = 1, ..., L)$, and the recomposition is controlled by the binary switches at multiple layers $\delta = (\delta_l, l = 1, ..., L)$. Hence the top-down ConvNet amounts to a reconfigurable basis W_δ for representing X, and the model is a piecewise linear factor analysis. If we retain the bias term, we will have $X = W_\delta Z + b_\delta + \epsilon$, for an overall bias term that depends on δ. So the distribution of X is essentially piecewise Gaussian.

The generator model can be considered an explicit implementation of the local linear embedding [RS00], where Z is the embedding of X. In local linear embedding, the mapping between Z and X is implicit. In the generator model, the mapping from Z to X is explicit. With ReLU ConvNet, the mapping is piecewise linear, which is consistent with local linear embedding, except that the partition of the linear pieces by $\delta(Z; \theta)$ in the generator model is learned automatically.

2.3 Alternating Back-Propagation

The factor analysis model can be learned by the Rubin-Thayer EM algorithm [RT82, DLR77], where both the E-step and the M-step are based on multivariate linear regression. Inspired by this algorithm, we propose an alternating back-propagation algorithm for learning the generator network that iterates the following two-steps:

(1) *Inferential back-propagation*: For each training example, infer the continuous latent factors by Langevin dynamics or gradient descent.

(2) *Learning back-propagation*: Update the parameters given the inferred latent factors by gradient descent.

The Langevin dynamics [Nea11] is a stochastic sampling counterpart of gradient descent. The gradient computations in both steps are powered by back-propagation. Because of the ConvNet structure, the gradient computation in step (1) is actually a by-product of the gradient computation in step (2) in terms of coding.

Given the factors, the learning of the ConvNet is a supervised learning problem [DSB15] that can be accomplished by the learning back-propagation. With factors unknown, the learning becomes an unsupervised problem, which can be solved by adding the inferential back-propagation as an inner loop of the learning process. The alternating back-propagation algorithm follows the tradition of alternating operations in unsupervised learning, such as alternating linear regression in the EM algorithm for factor analysis, alternating least squares algorithm for matrix factorization [KBV09, KP08], and alternating gradient descent algorithm for sparse coding [OF97]. All these unsupervised learning algorithms alternate an inference step and a learning step, as is the case with alternating back-propagation.

Explaining away inference. The inferential back-propagation solves an inverse problem by an explaining-away process, where the latent factors compete with each other to explain each training example. The following are the advantages of the explaining-away inference of the latent factors:

(1) The latent factors may follow sophisticated prior models. For instance, in textured motions [WZ03] or dynamic textures [DCW03], the latent factors may follow a dynamic model such as vector auto-regression. By inferring the latent factors that explain the observed examples, we can learn the prior model.

(2) The observed data may be incomplete or indirect. For instance, the training images may contain occluded objects. In this case, the latent factors can still be obtained by explaining the incomplete or indirect observations, and the model can still be learned as before.

The algorithm can be stated in more formal way. Suppose we observe a training set of data vectors $\{X_i, i = 1, ..., n\}$, then each X_i has a corresponding Z_i, but all the X_i share the same ConvNet θ. Intuitively, we should infer $\{Z_i\}$ and learn θ to minimize the reconstruction error $\sum_{i=1}^{n} ||X_i - g_\theta(Z_i)||^2$ plus a regularization term that corresponds to the prior on Z.

More formally, the model can be written as $Z \sim p(Z)$ and $[X|Z, \theta] \sim p_\theta(X|Z)$. Adopting the language of the EM algorithm [DLR77], the complete-data model is given by

$$\begin{aligned}\log p_\theta(X, Z) &= \log\left[p(Z)p_\theta(X|Z)\right] \\ &= -\frac{1}{2\sigma^2}\|X - g_\theta(Z)\|^2 - \frac{1}{2}\|Z\|^2 + \text{const.}\end{aligned} \tag{2.3}$$

The observed-data model is obtained by integrating out Z: $p_\theta(X) = \int p(Z)p_\theta(X|Z)dZ$. The posterior distribution of Z is given by $p_\theta(Z|X) = p_\theta(X, Z)/p_\theta(X) \propto p(Z)p_\theta(X|Z)$ as a function of Z.

For the training data $\{X_i\}$, the complete-data log-likelihood is $L(\theta, \{Z_i\}) = \sum_{i=1}^n \log p_\theta(X_i, Z_i)$, where we assume σ^2 is given. Learning and inference can be accomplished by maximizing the complete-data log-likelihood, which can be obtained by the alternating gradient descent algorithm that iterates the following two steps: (1) Inference step: update Z_i by running l steps of gradient descent. (2) Learning step: update θ by one step of gradient descent.

A more rigorous method is to maximize the observed-data log-likelihood, which is $L(\theta) = \sum_{i=1}^n \log p_\theta(X_i) = \sum_{i=1}^n \log \int p_\theta(X_i, Z_i)dZ_i$. The observed-data log-likelihood takes into account the uncertainties in inferring Z_i.

The gradient of $L(\theta)$ can be calculated according to the following well-known fact that underlies the EM algorithm:

$$\begin{aligned}\frac{\partial}{\partial\theta}\log p_\theta(X) &= \frac{1}{p_\theta(X)}\frac{\partial}{\partial\theta}\int p_\theta(X, Z)dZ \\ &= \mathrm{E}_{p_\theta(Z|X)}\left[\frac{\partial}{\partial\theta}\log p_\theta(X, Z)\right].\end{aligned} \tag{2.4}$$

The expectation with respect to $p_\theta(Z|X)$ can be approximated by drawing samples from $p_\theta(Z|X)$ and then computing the Monte Carlo average.

The Langevin dynamics for sampling $Z \sim p_\theta(Z|X)$ iterates

$$\begin{aligned}&Z_{\tau+1} = Z_\tau + sU_\tau + \\ &\frac{s^2}{2}\left[\frac{1}{\sigma^2}(X - g_\theta(Z_\tau))\frac{\partial}{\partial Z}g_\theta(Z_\tau) - Z_\tau\right],\end{aligned} \tag{2.5}$$

where τ denotes the time step for the Langevin sampling, s is the step size, and U_τ denotes a random vector that follows $\mathrm{N}(0, I_d)$. The Langevin dynamics (2.5) is an explain-away process, where the latent factors in Z compete to explain away the current residual $X - g_\theta(Z_\tau)$.

To explain Langevin dynamics, its continuous time version for sampling $\pi(x) \propto \exp[-\mathcal{E}(x)]$ is $x_{t+\Delta t} = x_t - \Delta t \mathcal{E}'(x_t)/2 + \sqrt{\Delta t} U_t$. The dynamics has π as its stationary distribution, because it can be shown that for any well-behaved testing function h, if $x_t \sim \pi$, then $\mathrm{E}[h(x_{t+\Delta t})] - \mathrm{E}[h(x_t)] \to 0$, as $\Delta t \to 0$, so that $x_{t+\Delta t} \sim \pi$. Alternatively, given $x_t = x$, suppose $x_{t+\Delta t} \sim K(x, y)$, then $[\pi(y)K(y, x)]/[\pi(x)K(x, y)] \to 1$ as $\Delta t \to 0$, where $K(x, y)$ is the transition probability.

The stochastic gradient algorithm of [You99] can be used for learning, where in each iteration, for each Z_i, only a single copy of Z_i is sampled from $p_\theta(Z_i|X_i)$ by running a finite number of steps of Langevin dynamics starting from the current value of Z_i, i.e., the warm start. With $\{Z_i\}$ sampled in this manner, we can update the parameter θ based on the gradient $L'(\theta)$, whose Monte Carlo approximation is:

$$\begin{aligned} L'(\theta) &\approx \sum_{i=1}^{n} \frac{\partial}{\partial \theta} \log p_\theta(X_i, Z_i) \\ &= -\sum_{i=1}^{n} \frac{\partial}{\partial \theta} \frac{1}{2\sigma^2} \|X_i - g_\theta(Z_i)\|^2 \\ &= \sum_{i=1}^{n} \frac{1}{\sigma^2} (X_i - g_\theta(Z_i)) \frac{\partial}{\partial \theta} g_\theta(Z_i). \end{aligned} \tag{2.6}$$

If the Gaussian noise U_τ in the Langevin dynamics (2.5) is removed, then the algorithm becomes the alternating gradient descent algorithm. It is possible to update both θ and $\{Z_i\}$ simultaneously by joint gradient descent.

Both the inferential back-propagation and the learning back-propagation are guided by the residual $X_i - g_\theta(Z_i)$. The inferential back-propagation is based on $\partial g_\theta(Z)/\partial Z$, whereas the learning back-propagation is based on $\partial g_\theta(Z)/\partial \theta$. Both gradients can be efficiently computed by back-propagation. The computations of the two gradients share most of their steps. Specifically, for the top-down ConvNet defined by (2.2), $\partial g_\theta(Z)/\partial \theta$ and $\partial g_\theta(Z)/\partial Z$ share the same code for the chain rule computation of $\partial Z^{(l-1)}/\partial Z^{(l)}$ for $l = 1, ..., L$. Thus, the code for $\partial g_\theta(Z)/\partial Z$ is part of the code for $\partial g_\theta(Z)/\partial \theta$.

Algorithm 1 below describes the details of the learning and sampling algorithm. The Langevin dynamics samples from a gradually changing posterior distribution $p_\theta(Z_i|X_i)$ be-

Algorithm 1 Alternating back-propagation

```
Require:
    (1) training examples $\{X_i, i = 1, ..., n\}$
    (2) number of Langevin steps $l$
    (3) number of learning iterations $T$
Ensure:
    (1) learned parameters $\theta$
    (2) inferred latent factors $\{Z_i, i = 1, ..., n\}$

1: Let $t \leftarrow 0$, initialize $\theta$.
2: Initialize $Z_i$, for $i = 1, ..., n$.
3: repeat
4:     Inferential back-propagation: For each $i$, run $l$ steps of Langevin dynamics to
       sample $Z_i \sim p_\theta(Z_i|X_i)$ with warm start, i.e., starting from the current $Z_i$, each step
       follows equation (2.5).
5:     Learning back-propagation: Update $\theta \leftarrow \theta + \gamma_t L'(\theta)$, where $L'(\theta)$ is computed
       according to equation (2.6), with learning rate $\gamma_t$.
6:     Let $t \leftarrow t + 1$
7: until $t = T$
```

cause θ keeps changing. The updating of both Z_i and θ collaborate to reduce the reconstruction error $||X_i - g_\theta(Z_i)||^2$. The parameter σ^2 plays the role of annealing or tempering in Langevin sampling. If σ^2 is very large, then the posterior is close to the prior $\mathrm{N}(0, I_d)$. If σ^2 is very small, then the posterior may be multi-modal, but the evolving energy landscape of $p_\theta(Z_i|X_i)$ may help alleviate the trapping of the local modes. In practice, we tune the value of σ^2 instead of estimating it. The Langevin dynamics can be extended to Hamiltonian Monte Carlo [Nea11] or more sophisticated versions [GC11].

2.4 EM, Density Mapping, and Density Shifting

Suppose the training data $\{X_i, i = 1, ..., n\}$ come from a data distribution $P_{\text{data}}(X)$. To understand how the alternating back-propagation algorithm or its EM idealization maps the prior distribution of the latent factors $p(Z)$ to the data distribution $P_{\text{data}}(X)$ by the learned $g_\theta(Z)$, to ease the presentation, we use the notation $g(Z;\theta)$ here, the same applies for probability distribution. We define

$$\begin{aligned} P_{\text{data}}(Z, X;\theta) &= P_{\text{data}}(X)p(Z|X;\theta) \\ &= P_{\text{data}}(Z;\theta)P_{\text{data}}(X|Z;\theta), \end{aligned} \tag{2.7}$$

where $P_{\text{data}}(Z;\theta) = \int p(Z|X;\theta)P_{\text{data}}(X)dX$ is obtained by averaging the posteriors $p(Z|X;\theta)$ over the observed data $X \sim P_{\text{data}}$. That is, $P_{\text{data}}(Z;\theta)$ can be considered the data prior. The data prior $P_{\text{data}}(Z;\theta)$ is close to the true prior $p(Z)$ in the sense that

$$\begin{aligned} \text{KL}(P_{\text{data}}(Z;\theta)|p(Z)) &\leq \text{KL}(P_{\text{data}}(X)|p(X;\theta)) \qquad (2.8) \\ &= \text{KL}(P_{\text{data}}(Z, X;\theta)|p(Z, X;\theta)). \end{aligned}$$

The right hand side of (2.8) is minimized at the maximum likelihood estimate $\hat{\theta}$, hence the data prior $P_{\text{data}}(Z;\hat{\theta})$ at $\hat{\theta}$ should be especially close to the true prior $p(Z)$. In other words, at $\hat{\theta}$, the posteriors $p(Z|X;\hat{\theta})$ of all the data points $X \sim P_{\text{data}}$ tend to pave the true prior $p(Z)$.

2.5 Experiments

We test the proposed alternating back-propagation algorithm for generator learning on wide range of tasks. The training images, sounds and videos are scaled so that the intensities are within the range $[-1, 1]$. We adopt the structure of the generator network of [RMC15, DSB15], where the top-down network consists of multiple layers of deconvolution by linear superposition, ReLU non-linearity, and up-sampling, with tanh non-linearity at the bottom-layer [RMC15] to make the signals fall within $[-1, 1]$. For spatial-temporal generator network, we use 3D deconvolutional layers as in [VPT16] but with different top-layer structures for different models. We also adopt batch normalization [IS15].

2.5.1 Qualitative Experiments

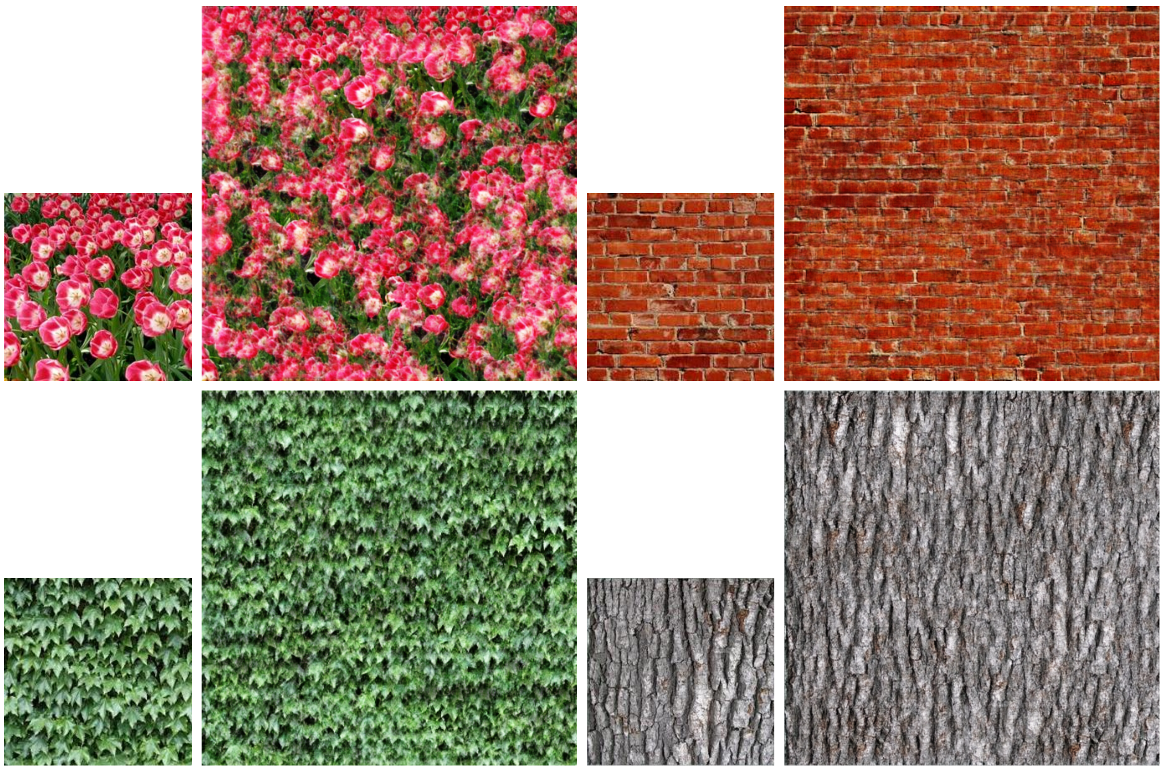

Figure 2.1: Modeling texture patterns. For each example, *Left:* the 224×224 observed image. *Right:* the 448×448 generated image.

Experiments 1: Modeling Texture Patterns. We learn a separate model from each texture image. The images are collected from the Internet, and then resized to $224\times$ 224. The synthesized images are 448×448. Figure 2.1 shows four examples.

The factors Z at the top layer form a $\sqrt{d} \times \sqrt{d}$ image, with each pixel following $\mathrm{N}(0, 1)$ independently. The $\sqrt{d} \times \sqrt{d}$ image Z is then transformed to X by the top-down ConvNet. We use $d = 7^2$ in the learning stage for all the texture experiments. In order to obtain the synthesized image, we randomly sample a 14×14 Z from $\mathrm{N}(0, I)$, and then expand the learned network θ to generate the 448×448 synthesized image $g_\theta(Z)$.

The training network is as follows. Starting from 7×7 image Z, the network has 5 layers of deconvolution with 5×5 kernels (i.e., linear superposition of 5×5 basis functions), with an up-sampling factor of 2 at each layer (i.e., the basis functions are 2 pixels apart). The number of channels in the first layer is 512 (i.e., 512 translation invariant basis functions), and is decreased by a factor 2 at each layer. The Langevin steps $l = 10$ with step size $s = 0.1$.

Experiments 2: Modeling Sound Patterns. A sound signal can be treated as a one-dimensional texture image [MS11]. The sound data are collected from the Internet. Each training signal is a 5 second clip with the sampling rate of 11025 Hertz and is represented as a 1×60000 vector. We learn a separate model from each sound signal.

The latent factors Z form a sequence that follows $\mathrm{N}(0, I_d)$, with $d = 6$. The top-down network consists of 4 layers of deconvolution with kernels of size 1×25, and up-sampling factor of 10. The number of channels in the first layer is 256, and decreases by a factor of 2 at each layer. For synthesis, we start from a longer Gaussian white noise sequence Z with $d = 12$ and generate the synthesized sound by expanding the learned network. Figure 2.2 shows the waveforms of the observed sound signal in the first row and the synthesized sound signal in the second row.

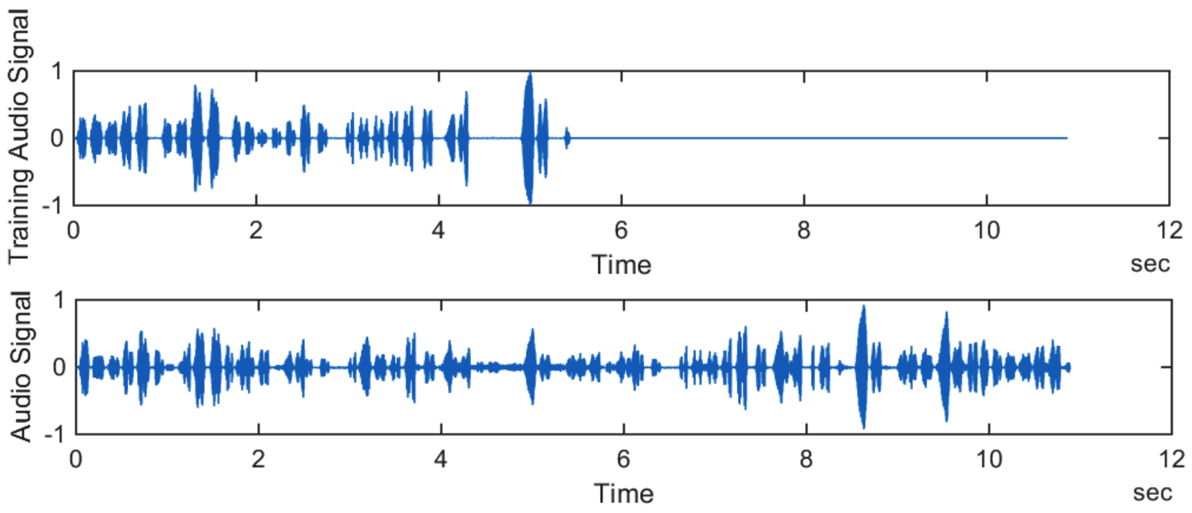

Figure 2.2: Modeling sound patterns. *Row 1:* the waveform of the training sound (the range is 0-5 seconds). *Row 2:* the waveform of the synthesized sound (the range is 0-11 seconds).

Experiments 3: Modeling Object Patterns. We model object patterns using the

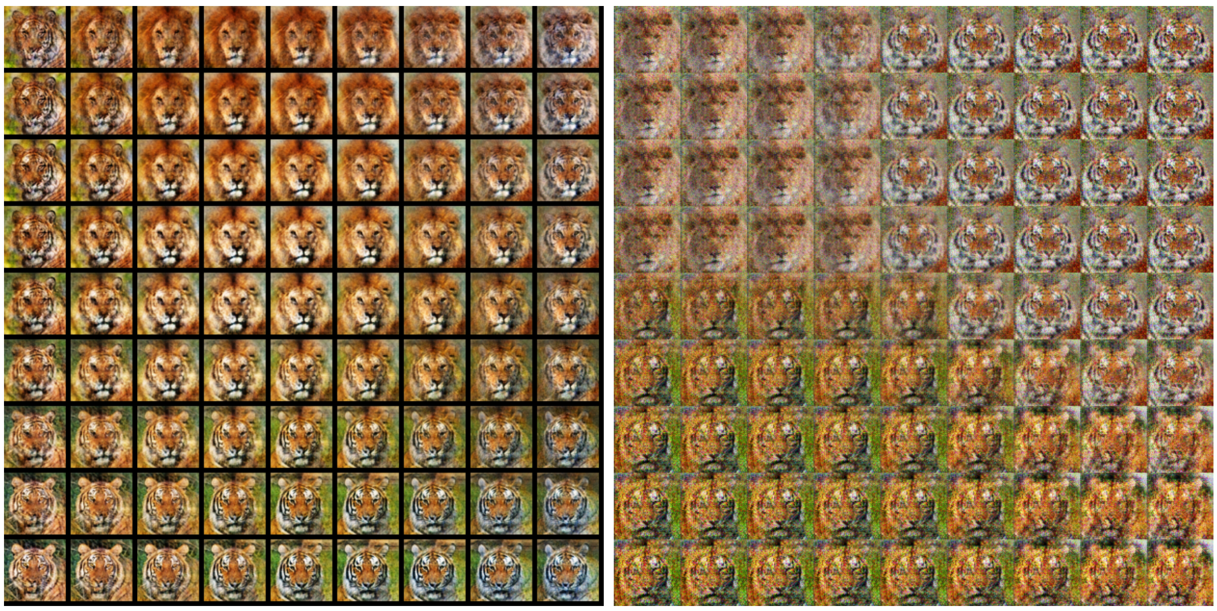

Figure 2.3: Modeling object patterns. *Left:* the synthesized images generated by our method. They are generated by $g_\theta(Z)$ with the learned θ, where $Z = (z_1, z_2) \in [-2, 2]^2$, and Z is discretized into 9×9 values. *Right:* the synthesized images generated using Deep Convolutional Generative Adversarial Net (DCGAN). Z is discretized into 9×9 values within $[-1, 1]^2$.

network structure that is essentially the same as the network for the texture model, except that we include a fully connected layer under the latent factors Z, now a d-dimensional vector. The images are 64×64. We use ReLU with a leaking factor 0.2 [MHN13, XWC15]. The Langevin steps $l = 30$ with step size $s = 0.3$.

In the first experiment, we learn a model where Z has two components, i.e., $Z = (z_1, z_2)$, and $d = 2$. The training data are 11 images of 6 tigers and 5 lions. After training the model, we generate images using the learned top-down ConvNet for $(z_1, z_2) \in [-2, 2]^2$, where we discretize both z_1 and z_2 into 9 equally spaced values. The left panel of Figure 2.3 displays the synthesized images on the 9×9 panel.

In the second experiment, we learn a model with $d = 100$ from 1000 face images randomly selected from the CelebA dataset [LLW15a]. The left panel of Figure 2.4 displays the images generated by the learned model. The middle panel displays the interpolation results. The images at the four corners are generated by the Z vectors of four images randomly selected

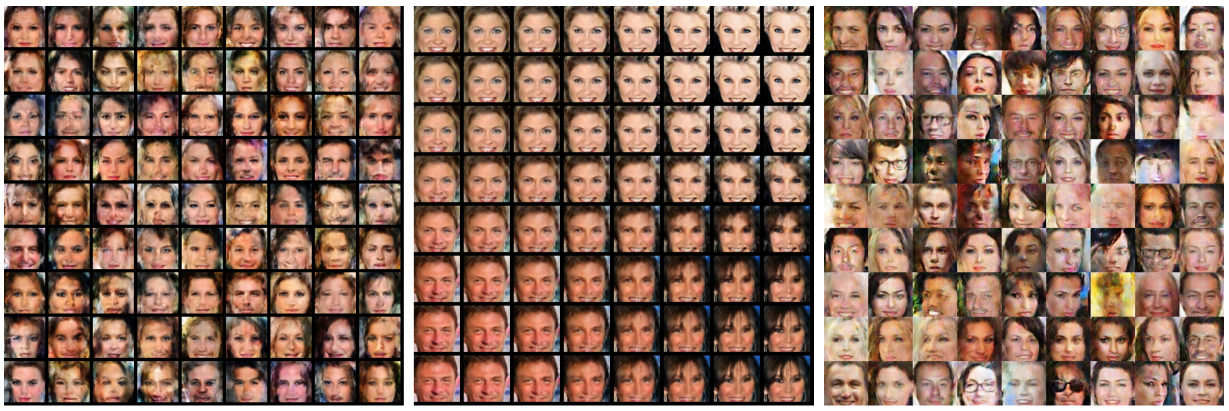

Figure 2.4: Modeling object patterns. *Left:* each image generated by our method is obtained by first sampling $Z \sim \mathrm{N}(0, I_{100})$ and then generating the image by $g_\theta(Z)$ with the learned θ. *Middle:* interpolation. The images at the four corners are reconstructed from the inferred Z vectors of four images randomly selected from the training set. Each image in the middle is obtained by first interpolating the Z vectors of the four corner images, and then generating the image by $g_\theta(Z)$. *Right:* the synthesized images generated by DCGAN, where Z is a 100 dimension vector sampled from uniform distribution.

from the training set. The images in the middle are obtained by first interpolating the Z's of the four corner images using the sphere interpolation [DSB16] and then generating the images by the learned ConvNet.

We also provide qualitative comparison with Deep Convolutional Generative Adversarial Net (DCGAN) [GPM14, RMC15]. The right panel of Figure 2.3 shows the generated results for the lion-tiger dataset using 2-dimensional Z. The right panel of Figure 2.4 displays the generated results trained on 1000 aligned faces from celebA dataset, with $d = 100$. We run $T = 600$ iterations as in our method.

Experiments 4: State-space Dynamic Textures Modeling. We train the generator model on on a wide range of dynamic patterns randomly selected from the DynTex [PFH10] database, action database [GBS07], and from the Internet. In this experiment, we model a textured motion [WZ03] or a dynamic texture [DCW03] by a non-linear state-space dynamic system $X_t = g_\theta(Z_t) + \epsilon_t$, and $Z_{t+1} = AZ_t + \eta_t$, where we assume the latent factors follow a vector auto-regressive model, where A is a $d \times d$ matrix, and $\eta_t \sim \mathrm{N}(0, Q)$ is the innovation. This model is a direct generalization of the linear dynamic system of [DCW03], where X_t is reduced to Z_t by principal component analysis (PCA) via singular value decomposition (SVD). We learn the model in two steps. (1) Treat $\{X_t\}$ as independent examples and learn θ and infer $\{Z_t\}$ as before. (2) Treat $\{Z_t\}$ as the training data, learn A and Q as in [DCW03]. After that, we can synthesize a new dynamic texture. We start from $Z_0 \sim \mathrm{N}(0, I_d)$, and then generate the sequence according to the learned model (we discard a burn-in period of 15 frames). Figure 2.5 shows some experiments, where we set $d = 20$. The first row is a segment of the sequence generated by our model, and the second row is generated by the method of [DCW03], with the same dimensionality of Z. It is possible to generalize the auto-regressive model of Z_t to recurrent network. We may also treat the video sequences as 3D images, and learn generator models with 3D spatial-temporal filters or basis functions as shown in following experiment.

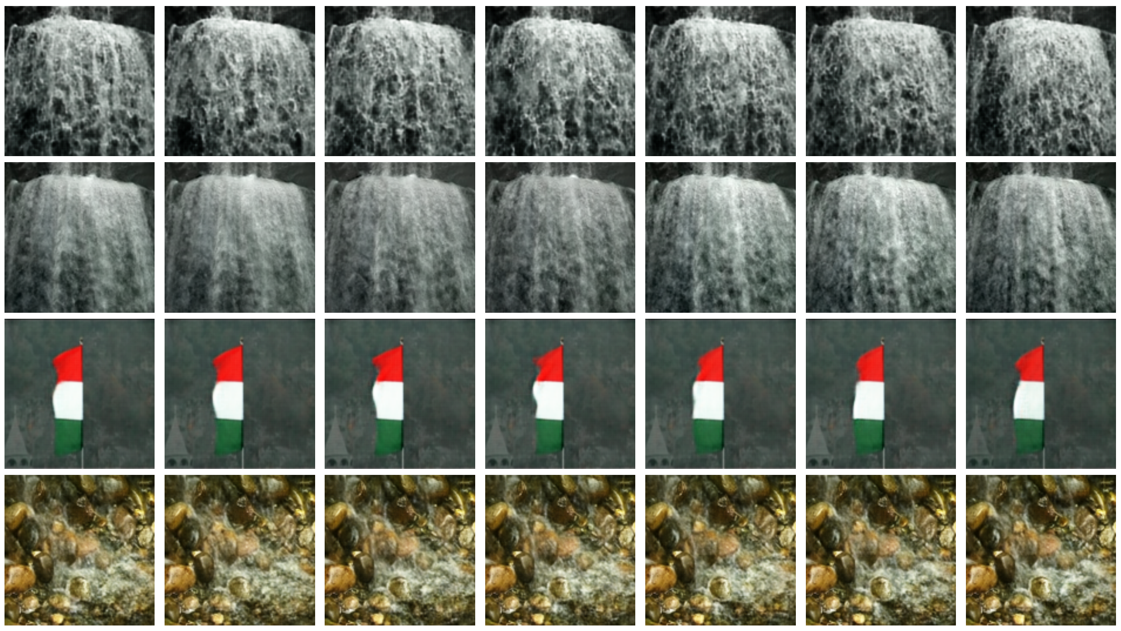

Figure 2.5: State-space dynamic textures modeling. *Row 1:* a segment of the synthesized sequence by our method. *Row 2:* a sequence by the method of [DCW03]. *Rows 3 and 4:* two more sequences by our method.

Experiments 5: Spatial-temporal Dynamic Patterns Modeling. Instead of separately modeling the individual images in the video sequence, we can also seek to capture the dynamic patterns directly by spatial-temporal filters. To be more precise, let $\mathbf{I}(x, y, t)$ be an image sequence, where $(x, y) \in \mathcal{S}$ indexes the pixel in the spatial domain, and $t \in \mathcal{T}$ indexes the frame in the temporal domain. We assume $\mathbf{I}$ is generated by 3D spatial-temporal generator model and we consider the following three versions:

(1) The dynamic pattern in $\mathbf{I}(x, y, t)$ is stationary in both the spatial and temporal domains, such as water waves, rain, snow etc. In that case, we assume $Z = Z(x, y, t, m)$, where $(x, y) \in \mathcal{S}^{(0)}$, $t \in \mathcal{T}^{(0)}$, and $m = 1, ..., M$, where $\mathcal{S}^{(0)}$ and $\mathcal{T}^{(0)}$ are sub-sampled spatial and temporal domains, i.e., the index k in $Z = (Z_k, k = 1, ..., d)$ takes the form $k = (x, y, t, m)$, so that at the top-layer Z consists of M Gaussian white noise image sequences. Specifically, we learn a 4-layer ConvNet, where there are 256, 128, 64 and 3 filters with the size of $4 \times 4 \times 4$ and a stride of 2 in 4 layers respectively. We use $8 \times 8 \times 4 \times 2$ latent factors to generate dynamic patterns.

(2) The dynamic pattern in $\mathbf{I}(x, y, t)$ is stationary in the temporal domain, but non-stationary in the spatial domain, such as dynamic textures that exhibit stochastic repetitiveness in the temporal domain. In that case, we assume $Z = Z(m, t)$, where $t \in \mathcal{T}^{(0)}$, and $m = 1, ..., M$. This model corresponds to the model in (1) where $\mathcal{S}^{(0)}$ is 1×1. Specifically, we learn a 6-layer ConvNet, where the first layer has 512 filters with a size of $4 \times 4 \times 1$ and a stride of 1. The second layer has 64 filters with a size of $7 \times 7 \times 7$ and a stride of 1. There are 256, 128, 64 and 3 filters with a size of $4 \times 4 \times 4$ and a stride of 2 from layer 3 to layer 6. We use $1 \times 1 \times 8 \times 20$ latent factors to generate the dynamic patterns of the double length.

(3) The dynamic pattern in $\mathbf{I}(x, y, t)$ is non-stationary in both spatial and temporal domains, such as actions and movements. In that case, we assume $Z = Z(m)$, $m = 1, ..., M$. The model corresponds to the model in (1) where $\mathcal{S}^{(0)}$ is a single pixel, and $\mathcal{T}^{(0)}$ is a single frame. Specifically, we learn a 6-layer ConvNet, where the first layer has 512 filters with the size of $4 \times 4 \times 1$ and stride of 1. The second layer has 384 filters with the size of $7 \times 7 \times 7$ and stride of 1. There are 256, 128, 64 and 3 filters with the size of $4 \times 4 \times 4$ and stride of 2 from layer 3 to layer 6. We use $1 \times 1 \times 1 \times 20$ latent factors to generate dynamic patterns.

In our experiments, each video sequence from the selected category is partitioned into segments of 32 frames and such segments are used for training and testing. Figure 2.6 shows some synthesis results for different model versions.

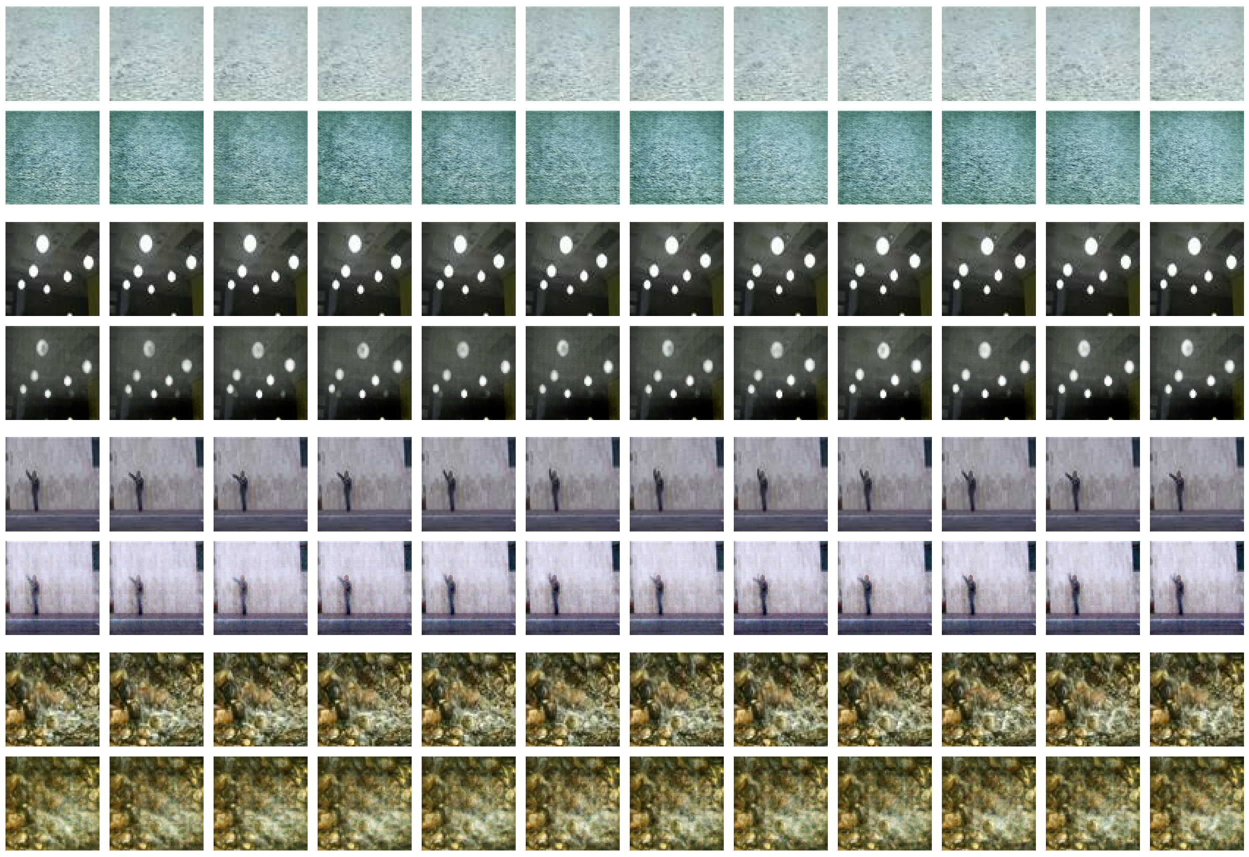

Figure 2.6: Synthesized dynamic sequences using three versions of the generator model. For each category, the first row displays frames of the training sequence, and the second row displays the frames of the generated sequence. The first category shows the synthesized sequence generated by model-1 with the original length. Category 2, category 3 display the synthesized sequences generated by model-2 with doubled length. The last category displays the generated sequence by model-3.

We also compare our method with state space auto-regressive models [DCW03] for dynamic textures. Since their methods are based on frame-wise modeling, we set the latent dimension to 20 for each frame as suggested in their papers. Figure 2.7 shows one comparison result. It can be seen that frame-wise modeling can accumulate noises, rendering less sharp synthesis results as time evolves.

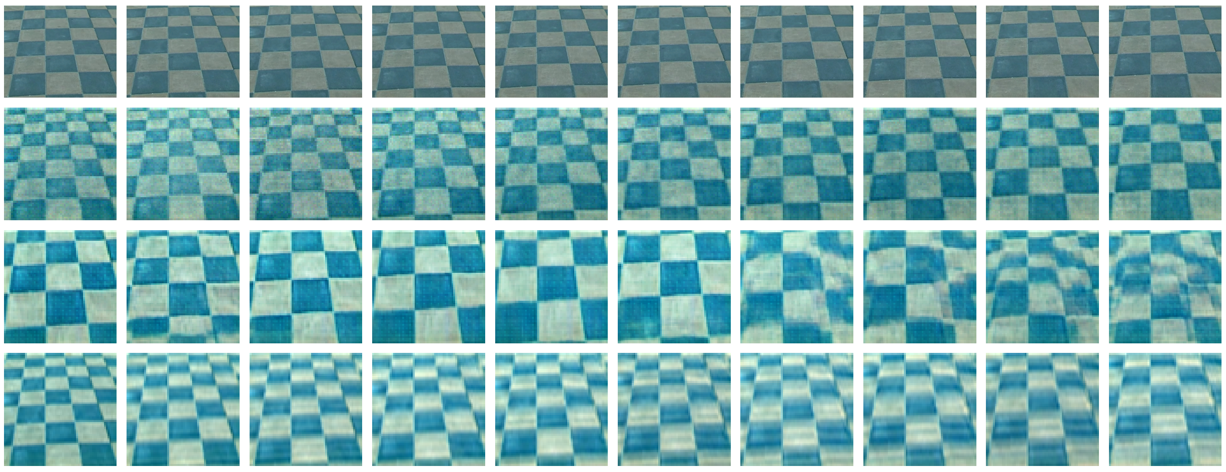

Figure 2.7: The comparison results for moving grid sequence. First row: 8 frames of the training sequence. Second row: 8 frames of the generated sequence using our method. Third row: 8 frames of the generated sequence using state-space generator model in experiment 4. Fourth row: 8 frames of generated sequence using [DCW03].

2.5.2 Quantitative Experiments

Experiments 6: Model Evaluation by Reconstruction Error on Testing Data. After learning the model from the training images (now assumed to be fully observed), we can evaluate the model by the reconstruction error on the testing images. We randomly select 1000 face images for training and 300 images for testing from CelebA dataset. After learning, we infer the latent factors Z for each testing image using inferential back-propagation, and then reconstruct the testing image by $g_\theta(Z)$ using the inferred Z and the learned θ. In the inferential back-propagation for inferring Z, we initialize $Z \sim \mathrm{N}(0, I_d)$, and run 300 Langevin steps with step size 0.05. Table 2.1 shows the reconstruction errors of alternating back-propagation learning (ABP) as compared to PCA learning for different latent dimensions d. Figure 2.8 shows some reconstructed testing images. For PCA, we learn the d eigenvectors from the training images, and then project the testing images on the learned eigenvectors for reconstruction.

experiment	$d = 20$	$d = 60$	$d = 100$	$d = 200$
ABP	0.0810	0.0617	0.0549	0.0523
PCA	0.1038	0.0820	0.0722	0.0621

Table 2.1: Reconstruction errors on testing images, after learning from training images using our method (ABP) and PCA.

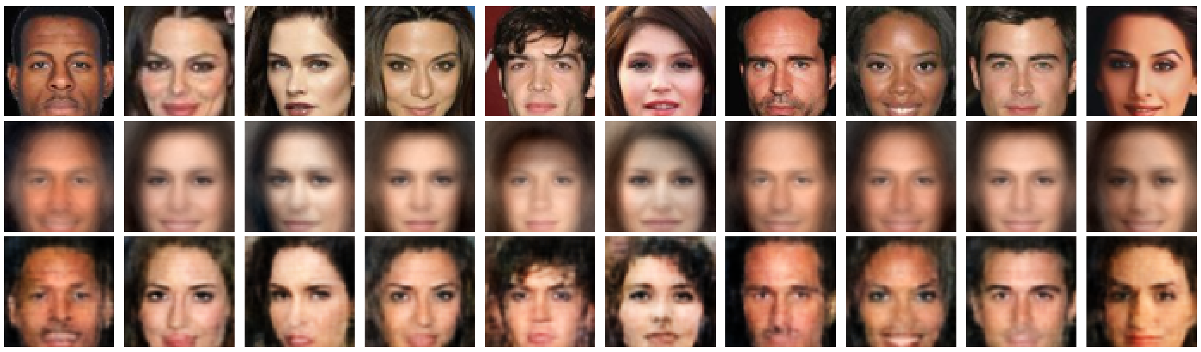

Figure 2.8: Comparison between our method and PCA. *Row 1:* original testing images. *Row 2:* reconstructions by PCA eigenvectors learned from training images. *Row 3:* reconstructions by the generator learned from training images. $d = 20$ for both methods.

Experiments 7: Learning from Indirect Data. We can learn the model from the compressively sensed data [CRT06]. We generate a set of white noise images as random projections. We then project the training images on these white noise images. We can learn the model from the random projections instead of the original images. We only need to replace $\|X - g_\theta(Z)\|^2$ by $\|SX - Sg_\theta(Z)\|^2$, where S is the given white noise sensing matrix, and SX is the observation. We can treat S as a fully connected layer of known filters below $g_\theta(Z)$, so that we can continue to use alternating back-propagation to infer Z and learn θ, thus recovering the image by $g_\theta(Z)$. In the end, we will be able to (T1) Recover the original images from their projections during learning. (T2) Synthesize new images from the learned model. (T3) Recover testing images from their projections based on the learned model. Our experiments are different from traditional compressed sensing, which is task (T3), but not tasks (T1) and (T2). Moreover, the image recovery in our work is based on non-linear dimension reduction instead of linear sparsity.

We evaluate our method on 1000 face images randomly selected from CelebA dataset. These images are projected onto $K = 1000$ white noise images with each pixel randomly sampled from $\mathrm{N}(0, 0.5^2)$. After this random projection, each image of size $64 \times 64 \times 3$ becomes a K-dimensional vector. We show the recovery errors for different latent dimensions d in Table 2.2, where the recovery error is defined as the per pixel difference (relative to the range of the pixel values) between the original image and the recovered image. Figure 2.9 shows some recovery results.

experiment	$d = 20$	$d = 60$	$d = 100$
error	0.0795	0.0617	0.0625

Table 2.2: Recovery errors in 3 experiments of learning from compressively sensed images.

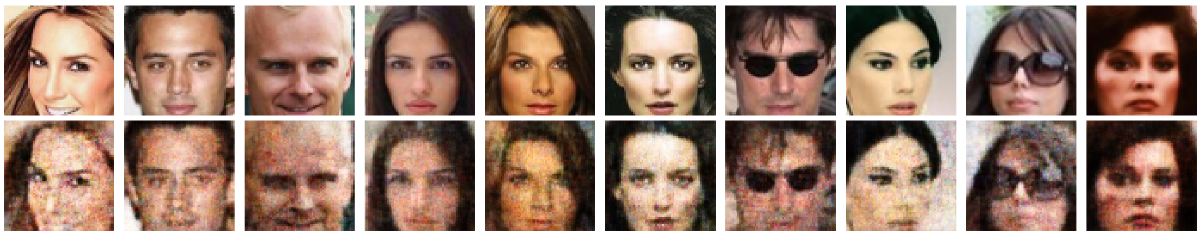

Figure 2.9: Learning from indirect data. *Row 1:* the original $64 \times 64 \times 3$ images, which are projected onto 1,000 white noise images. *Row 2:* the recovered images during learning.

Experiments 8: Learning from Incomplete Images. Our method can learn from images with occluded pixels. This task is inspired by the fact that most of the images contain occluded objects. It can be considered a non-linear generalization of matrix completion in recommender system.

Our method can be adapted to this task with minimal modification. The only modification involves the computation of $\|X - g_\theta(Z)\|^2$. For a fully observed image, it is computed by summing over all the pixels. For a partially observed image, we compute it by summing over only the observed pixels. Then we can continue to use the alternating back-propagation algorithm to infer Z and learn θ. With inferred Z and learned θ, the image can be automatically recovered by $g_\theta(Z)$. In the end, we will be able to accomplish the following tasks: (T1) Recover the occluded pixels of training images. (T2) Synthesize new images from the learned model. (T3) Recover the occluded pixels of testing images using the learned model.

experiment	M0.5	M0.7	M0.9	P20	P30
error	0.0571	0.0662	0.0771	0.0773	0.1035

Table 2.3: Recovery errors in 5 experiments of learning from occluded images.

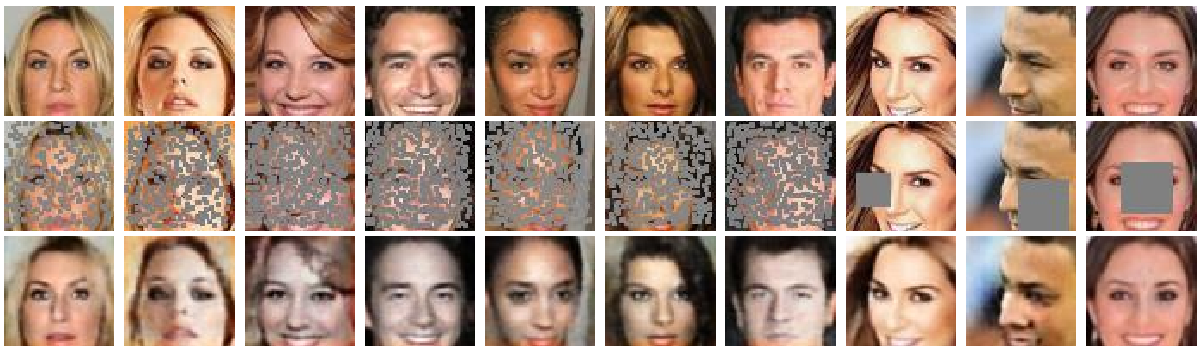
Figure 2.10: Learning from incomplete data. The 10 columns belong to experiments M0.5, M.07, M0.9, M0.9, M0.9, M0.9, M0.9, P20, P30, P30 respectively. *Row 1:* original images, not observed in learning. *Row 2:* training images. *Row 3:* recovered images during learning.

We want to emphasize that in our experiments, all the training images are partially occluded. Our experiments are different from (1) de-noising auto-encoder [VLB08], where the training images are fully observed, and noises are added as a matter of regularization, (2) in-painting or de-noising, where the prior model or regularization has already been learned or given. (2) is about task (T3) mentioned above, but not about tasks (T1) and (T2).

Learning from incomplete data can be difficult for GAN and VAE, because the occluded pixels are different for different training images.

We evaluate our method on 10,000 images randomly selected from CelebA dataset. We design 5 experiments, with two types of occlusions: (1) 3 experiments are about salt and pepper occlusion, where we randomly place 3×3 masks on the 64×64 image domain to cover

Methods	Consecutive Block		Random Block	
	LHD	ours	LHD	ours
flag	0.1359	**0.0392**	0.1111	**0.0300**
waterfall	0.2720	**0.2558**	0.2666	**0.2393**
waterstone	0.1899	**0.1321**	0.1717	**0.1251**
light	0.0728	**0.0725**	0.0586	**0.0400**
elevator	0.1043	**0.0768**	0.0735	**0.0463**

Table 2.4: Interpolation errors for various training videos with different occlusion masks.

roughly 50%, 70% and 90% of pixels respectively. These 3 experiments are denoted M0.5, M0.7, and M0.9 respectively. (2) 2 experiments are about single region mask occlusion, where we randomly place a 20×20 or 30×30 mask on the 64×64 image domain. These 2 experiments are denoted P20 and P30 respectively. We set $d = 100$. Table 2.3 displays the recovery errors of the 5 experiments, where the error is defined as per pixel difference (relative to the range of the pixel values) between the original image and the recovered image on the occluded pixels. We emphasize that the recovery errors are not training errors, because the intensities of the occluded pixels are not observed in training. Figure 2.10 displays recovery results. In experiment M0.9, 90% of pixels are occluded, but we can still learn the model and recover the original images.

Experiments 9: Interpolation of time-stationary patterns. For video sequences with repetitive patterns in the temporal domain, our method with spatial-temporal generator can efficiently learn such patterns by assuming $Z = Z(m, t)$ even on a single training sequence. We evaluate our method by interpolation. We consider two types of missing patterns. The first type is the consecutive block, in which we entirely block 8 consecutive frames in the middle of the training sequence. The second type is the random block, in which we entirely block 3 consecutive frames in 3 randomly chosen positions in the training sequence. Each interpolation experiment is performed on only one video training sequence. Note that the current 3D GAN model [VPT16] cannot easily do this, because the spatial-temporal

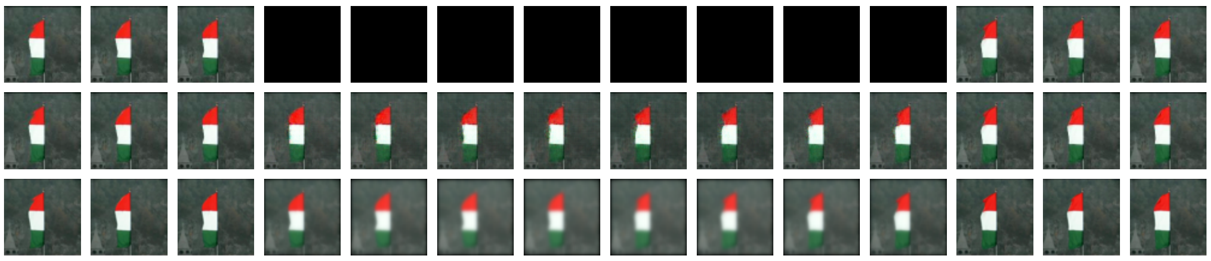

Figure 2.11: Interpolation results. The first row displays the occluded training sequence. The second row displays the interpolation results by our method. The third row shows the interpolation results by using Laplacian heat diffusion.

non-stationary structure is used in their paper and it lacks the inference mechanism. We compare our approach with the widely used Laplacian Heat Diffusion method (LHD) where we extend the 2D Laplacian kernel to 3D version, and we use 500 sweeps for the video interpolation with learning rate 0.5. Table 2.4 shows the recovery errors for various kinds of dynamic patterns and different missing patterns. The recovery error is defined as the per-pixel absolute difference between the ground truth video and the recovered video on the occluded pixels. Figure 2.11 shows the interpolation results on the consecutive block pattern. It can be seen that our method can get sharper and more realistic results.

Experiments 10: Learning from Incomplete Videos. The task of learning from incomplete data is in general challenging and is infeasible for models that do not have inference mechanism, because such models cannot easily borrow strength from other training data. We have evaluated our generator model for this task on images as well as single video with time-stationary pattern. We now consider the spatial-temporal generator model for multiple videos under various blocking patterns. For fair comparison, we adopt the model-3 structure which assumes non-stationary in both the spatial and temporal domains. We compare with the strong baseline model, i.e., 3D variational auto-encoder (3D VAE), based on the original variational auto-encoder [KW13, RMW14a]. The 3D VAE share the same generator model structure as our model. For the inference network for 3D VAE, it has the "mirror" structure as its generator model, with convolution and LeakyReLU (with ratio 0.2) to replace the deconvolution and typical ReLU used in the generator. For all models, we set the latent factor dimension to 100.

We again experiment on three types of occlusions for this task, including salt and pepper noise, 2D patch occlusion and 3D block occlusion as described above. Table 2.5 displays the recovery errors on various dynamic patterns for 3 different occlusion types. It can be seen that the inference-based methods are well suited for this task, and our methods are competitive and outperform the baseline method in terms of recovery errors.

Methods	M0.5		P20		B25x25x15	
	3D VAE [KW13]	3D ABP	3D VAE [KW13]	3D ABP	3D VAE [KW13]	3D ABP
elevator	0.0421	**0.0415**	0.0422	**0.0415**	0.0525	**0.0476**
windmill	0.0442	**0.0412**	0.0464	**0.439**	0.0773	**0.0769**
flag1	0.0296	**0.0283**	0.0375	**0.0363**	0.0400	**0.0389**
flamingo	0.0478	**0.0471**	0.0467	**0.0441**	0.0603	**0.0598**
flag2	0.0775	**0.0772**	0.9746	**0.0947**	0.1601	**0.1592**

Table 2.5: Recover error for the various incomplete training videos based on different occlusion masks.

Experiments 11: Video Recovery. Besides directly learning the model from incomplete videos, we could also learn the spatial-temporal generator on clean videos in the training stage, then try to recover the occluded patterns during testing stage. For this task, we split the training/testing video sequences by about 4:1 in proportion, and randomly block the testing videos using different occlusion patterns. For adversarial trained network using 3D GAN [VPT16], we apply the inferential mechanism powered by Langevin dynamic for direct comparison with our method. For 3D VAE, we iteratively impute the occluded pixels during inference stages [RMW14a].

We experiment on three types of occlusions: (1) salt and pepper mask (M) which covers roughly 50% (M0.5) of the pixels per-frame. (2) 2D patch occlusion (P) where we randomly place a 20×20 (P20) mask on each frame. (3) 3D block occlusion (B) where we randomly place a 35×35×20 block in the whole video. For the inference stage on occluded testing videos, we run the inference step for 200 iterations with step size 0.05 for Langevin dynamic.

Table 2.6 shows recovery errors for various kinds of dynamic patterns and different occluding masks. It can be seen that our proposed ABP based method shows competitive or superior performance over baseline models.

Videos		flag	elevator	grid	windmill	traffic
M0.5	3D GAN [VPT16]	0.3385	0.1679	0.3696	0.2151	0.1697
	3D VAE [KW13]	0.1603	0.0485	0.2516	0.1170	0.0943
	3D ABP	**0.1370**	**0.0457**	**0.2106**	**0.1157**	**0.0903**
P20	3D GAN [VPT16]	0.3380	0.1796	0.3542	0.2483	0.1780
	3D VAE [KW13]	0.1511	0.0591	**0.1981**	0.1643	0.1037
	3D ABP	**0.1459**	**0.0562**	0.2044	**0.1621**	**0.1001**
B35	3D GAN [VPT16]	0.3382	0.1803	0.4505	0.2626	0.1740
	3D VAE [KW13]	0.1801	0.0591	**0.2082**	0.1851	0.1012
	3D ABP	**0.1778**	**0.0577**	0.2159	**0.1779**	**0.0991**

Table 2.6: Recovery errors for various incomplete testing videos with different occlusion masks.

2.6 Summary

In this chapter, we recognize the generator model as a non-linear generalization of the factor analysis model, and propose an alternating back-propagation algorithm for learning. The alternating back-propagation algorithm iterates the inferential back-propagation for inferring the latent factors and the learning back-propagation for updating the parameters. Both back-propagation steps share most of their computing steps in the chain rule calculations. The experiments show the proposed algorithm can be effective in generating different form of data (e.g., audio, image, video etc) as well as completing occluded data.

CHAPTER 3

Multi-view Generator Model

We have discussed the generator model and its learning algorithm in chapter 2. Such model can be effective for data that shares the similar properties, however, the data online are collected in various forms and in different "view points" which present difficulty for the original generator model. In this chapter, we shall extend the generator model and the learning for multi-view data.

3.1 Introduction and Motivation

Multi-view data have become increasingly accessible in many areas of scientific analysis including video surveillance, social computing, and environmental science, where data are collected from diverse domains, or described by different feature sets. The representations from different domains or different feature sets for a common identity are referred to as different"views" of the data. For example, a document can be described using both images and audios, or be described in multiple languages. Human's identity can be represented by multiple biometric features, such as face, gait, fingerprint, iris etc. Human faces or gait sequences captured by multiple cameras from different viewpoints can be utilized together for person identification in uncontrolled environment. Therefore, multi-view representation learning has wide applicability.

Consensus and complementarity are two main principles behind the multi-view representation learning [XTX13, LYZ16]. Consensus principle aims to maximize the agreement on the representations learned from multiple distinct views. For example, among unconstrained face recognition and gait recognition, pose variations or viewpoint variations are the bottleneck for real-world applications. In such applications, differences between intra-subjects under two views are usually much larger than the differences between inter-subjects under the same view, which makes the multi-view learning problem challenging. Consensus principle is commonly employed by various multi-view representation learning methods to obtain a common representation for multiple heterogeneous spaces. This usually comes in two types: (1) finding common subspace across different views, and (2) transforming non-normal views to normal view. For the first type, Canonical Correlation Analysis (CCA) based algorithms [HSS04, SC07, SSY14, XWY16] and coupled projections (CP) based algorithms [LCS10, WXY14, XW16] are two representative methods which tend to find a common subspace such that the heterogeneous attributes can be eliminated in this consensus subspace. Specifically, CCA based algorithms aim to maximize the correlations (or the principal angles) of variables among different views, while CP based algorithms aim to minimize the distances between the projecting point pairs that have similar relations in the original heterogeneous sets. For the second type of consensus principle enforcement, there are several notable models which transform the data under different views into a normal view. Specifically, view transformation based model (VTM) [KWL09, ZZH11, KWZ12] performs a linear transformation from non-normal views to normal view. Stacked progressive auto-encoders (SPAE) [KSC14] models further extend VTM by using deep neural network to model complex non-linear transformations from the non-frontal face images to frontal ones in a progressive way. Recently, generative model based methods, such as generative adversarial networks (GAN) [GPM14], have been employed to rotate images from non-normal views to the normal view. TP-GAN [HZL17] is proposed to recover a frontal face in a data-driven way, while GaitGAN [YCR17] is proposed to generate the side view gait images as invariant gait features for multi-view recognition task.

Different views of data usually contain complementary information, therefore the complementarity principle states that multiple views can be employed to comprehensively and accurately represent the data. The complementarity principle has also been employed by many multi-view representation learning methods in different ways. For the task of human identification at a distance, face and gait features are combined to obtain better performance than the methods that only use the face or gait feature alone. For example, Zhou and Bhanu [ZB06] conducts feature concatenation after normalization and dimension reduction to fuse face and gait information. Xing and Wang [XWL15] fuses the gait and face features by computing the weighted mean of the two projecting features in the coupled subspace. For the task of human pose inference, the image features and different pose information are connected with one another in a latent space, which integrates the complementary information underlying different views [SMF09].

3.2 Model and Algorithm

The generator model described in chapter 2 can be treated as a single-view generator model in which only one generator model is used for all observed data that is potentially mixed:

$$\begin{aligned} &X = g_\theta(Z) + \epsilon, \\ &Z \sim \mathrm{N}(0, I_d),\ \epsilon \sim \mathrm{N}(0, \sigma^2 I_D),\ d < D. \end{aligned} \tag{3.1}$$

Such generator model can be effective in generation as well as completion as described in chapter 2. However, if X is heterogeneous or inhomogeneous, the original generator model can be less effective. The model tends to encode all the variations in the data in the latent factors in a highly non-linear and non-interpretable way. Although such Z can be used to generate sharp images, they hardly contain any useful information for downstream vision tasks. Therefore, we introduce a multi-view generator model to solve this problem.

Suppose X contains signals that come from m different source domains, i.e., $X = \{X^{(1)}, X^{(2)}, ..., X^{(m)}\}$. The number of domains can be determined by the specific application and we assume the signals are obtained across m domains. To effectively model the shared representations, we consider the following model (3.2):

$$\begin{cases} X^{(1)} = g_{\theta_1}(Z) + \epsilon_1 \\ X^{(2)} = g_{\theta_2}(Z) + \epsilon_2 \\ \ldots \\ X^{(m)} = g_{\theta_m}(Z) + \epsilon_m \\ Z \sim \mathrm{N}(0, \mathbf{I}_d) \; \epsilon_v \sim \mathrm{N}(0, \sigma^2 \mathbf{I}_D) \end{cases} \tag{3.2}$$

where the vector of the latent factors Z is shared across signals from different domains, and g_{θ_v} denotes the generator sub-network corresponding the v-th view, $v \in \{1, \ldots, m\}$. In this way, the domain or view specific variation is encoded through its corresponding view-specific generator sub-network, while the latent factors are forced to represent the common features among all the observations.

To learn from this multi-view generator model, we introduce a multi-view alternating back-propagation algorithm based on chapter 2. The generator model can be learned from training examples $\{X_i, i = 1, ..., n\}$ by the maximum likelihood estimation (MLE). Since the error term ϵ is normally distributed, the MLE is equivalent to L_2 reconstruction error $\sum_{i=1}^{n} \|X_i - g_\theta(Z_i)\|^2$. The basic idea of learning is to iteratively optimize $\{Z_i\}$ and θ until convergence. More rigorously, since the latent factors are random variables, sampling method is employed to account for the uncertainty in Z_i.

Specifically, for the training examples from m domains, i.e., $X = \{X^{(1)}, X^{(2)}, ..., X^{(m)}\}$, the model can be written as $Z \sim p(Z)$ and $[X^{(v)}|Z, \theta_v] \sim p_{\theta_v}(X^{(v)}|Z)$, where $v \in \{1, ...m\}$. The complete data log-likelihood is thus

$$\begin{aligned} \log p_\theta(X, Z) &= \log \left[p(Z) \prod_{v=1}^{m} p_{\theta_v}(X^{(v)}|Z) \right] \\ &= -\sum_{v=1}^{m} \frac{1}{2\sigma^2} \|X^{(v)} - g_{\theta_v}(Z)\|^2 - \frac{1}{2}\|Z\|^2 + \mathrm{C}, \end{aligned} \tag{3.3}$$

where C denotes the constant w.r.t Z and θ. The network parameter $\theta = \{\theta_1, \theta_2, ..., \theta_m\}$ is learned by maximizing the observed-data log-likelihood $L(\theta)$ which integrates out the unknown latent factors Z. More precisely, the gradient of $L(\theta)$ can be obtained from:

$$\begin{aligned}\frac{\partial}{\partial \theta_v} L(\theta) &= \frac{\partial}{\partial \theta_v} \log p_\theta(X) \\ &= \mathrm{E}_{p_\theta(Z|X)}\left[\frac{\partial}{\partial \theta_v} \log p_{\theta_v}(X^{(v)} \mid Z)\right]. \end{aligned} \tag{3.4}$$

The expectation can be approximated by Monte Carlo samples drawn from the posterior distribution $p_\theta(Z|X) \propto p_\theta(X, Z) = p(Z) \prod_{v=1}^{m} p_{\theta_v}(X^{(v)}|Z)$. Specifically, the latent factors are inferred using Langevin sampling method and they are updated as follows:

$$Z_{t+1} = Z_t + \frac{s^2}{2} \frac{\partial}{\partial Z} \log p_\theta(X, Z_t) + sU_t \tag{3.5}$$

where the Gaussian standard random noise term U is added to prevent the stochastic gradient step to be trapped by the local modes, and s denotes the step size of the Langevin dynamics. It can be shown that given sufficient transition steps, the obtained Z follows the posterior distribution (see chapter 2). In this chapter, the transition of Z also starts from the updated latent factors from the previous learning iteration, so that the persistent updating results in a sufficiently long chain to sample from the posterior distribution while greatly reducing the computational burden in that we only need to use a small number of transition steps in each learning iteration.

For each training example X_i, we run Langevin dynamics Eqn.(3.5) to get the corresponding posterior sample Z_i, then this sample is used for gradient computation in Eqn.(3.4). More precisely, the parameter θ is learned through Monte Carlo approximation:

$$\frac{\partial}{\partial \theta_v} L(\theta) \approx \frac{1}{n} \sum_{i=1}^{n} \left[\frac{\partial}{\partial \theta_v} \log p_{\theta_v}(X_i^{(v)} \mid Z_i)\right]. \tag{3.6}$$

The whole algorithm iterates through two steps: (1) inferential step that infers the latent factors through the Langevin dynamics, and (2) learning step that updates the network parameter θ by stochastic gradient descent. Gradient computations in both steps are powered by the efficient back-propagation. This is essentially alternating back-propagation algorithm stated in chapter 2, but with domain specific generator model and the shared latent space. The left part of Figure 3.1 illustrates the structure and the training process of the proposed model.

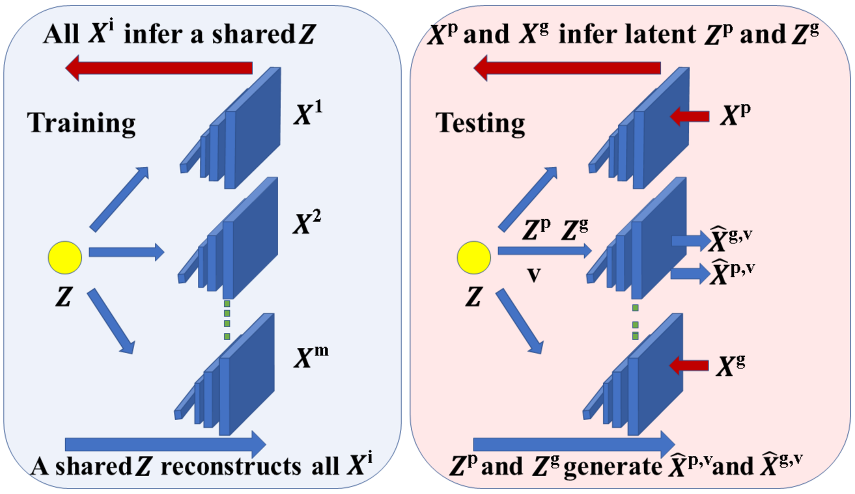
All X^i infer a shared Z
Training
X^1
X^2
X^m
Z
A shared Z reconstructs all X^i
X^p and X^g infer latent Z^p and Z^g
Testing
X^p
Z^p Z^g
v
$\hat{X}^{g,v}$
$\hat{X}^{p,v}$
Z
X^g
Z^p and Z^g generate $\hat{X}^{p,v}$ and $\hat{X}^{g,v}$

Figure 3.1: Overview of our method. The left panel illustrates the training process of the proposed method. The right panel illustrates our method for multi-view recognition task. Suppose the probe dataset X^p and the gallery dataset X^g come from two different views p and g. Our method first infers their latent factors Z^g and Z^p, and then generates $\hat{X}^{p,v}$ and $\hat{X}^{g,v}$ by feeding the inferred latent factors Z^p and Z^g to a sub-network under some common view. The complementary information for generating $\hat{X}^v$ under all views ($v \in \{1, \ldots, m\}$) and the latent factors Z are employed by our method.

3.3 Shared Representation and Recognition by Generation

The proposed model can be used to learn the shared representation (the latent factors Z) across multiple views. On the one hand, our proposed multi-view generator learning scheme is able to rotate or predict data to different views and recover the incomplete data from multiple views in the training stage. On the other hand, the shared representation obtained from our proposed scheme contains some identity information and can be employed in recognition problem.

Specifically, for multi-view recognition problem, during testing stage, suppose we have the gallery dataset X^g and the probe dataset X^p coming from the normal view and the test view, respectively. We can infer their latent factors Z^g and Z^p, which are in the shared distribution, and perform classification in this common subspace. Specifically, in the testing phase, when given data from any view v, we can infer the corresponding latent vector Z^v

using Eqn.(3.5) and we change $\log p_\theta(X,Z)$ by using only the given view v as:

$$\begin{aligned}\log p_\theta(X,Z) &= \log\left[p(Z)p_{\theta_v}(X^{(v)}|Z)\right]\\ &= -\frac{1}{2\sigma^2}\|X^{(v)} - g_{\theta_v}(Z)\|^2 - \frac{1}{2}\|Z\|^2 + \mathrm{C}.\end{aligned} \tag{3.7}$$

Moreover, we can improve the multi-view recognition by generating data $\hat{X}^{g,v}$ and $\hat{X}^{p,v}$ by feeding the inferred latent factors Z^g and Z^p to the sub-generators for a common view $v \in \{1,\dots,m\}$. Figure 3.1 illustrates this multi-view recognition by generation scheme. It is worth noting that although X^g and X^p are from different domains or views, the generated $\hat{X}^{g,v}$ and $\hat{X}^{p,v}$ obey the same distribution, and can be effective in recognition problem. Furthermore, we can perform the match-score fusion [ZB07] which combines both the latent factors Z and the generated data $\hat{X}^v$ in all common views $v \in \{1,\dots,m\}$. This scheme has the advantage of encoding both the common attributes (in Z) and view-specific discriminating information (in $\hat{X}^v$) through a unified way. Let $\mathcal{D}(\cdot,\cdot)$ denote the L_2 distances between two matrices, and define $\mathcal{D}_Z \equiv \mathcal{D}(z^p, Z^g)$, and $\mathcal{D}_X^v \equiv \mathcal{D}(\hat{x}^{p,v}, \hat{X}^{g,v})$, where $\mathcal{D}_Z$ and $\mathcal{D}_X^v$ are both vectors which calculate the L_2 distances between a probe identity and the gallery set containing N_g identities. The normalized match-score of latent factors z^p and Z^g can be computed as:

$$S_Z(z^p, Z^g) = \frac{\exp\{-\mathcal{D}_Z\}}{\sum_{i=1}^{N_g}\exp\{-\mathcal{D}_Z\}(i)}, \tag{3.8}$$

and the normalized match-score of the generated $\hat{X}^{p,v}$ and $\hat{X}^{g,v}$ can be computed as:

$$S_X^v(\hat{x}^{p,v}, \hat{X}^{g,v}) = \frac{\exp\{-\mathcal{D}_X^v\}}{\sum_{i=1}^{N_g}\exp\{-\mathcal{D}_X^v\}(i)}. \tag{3.9}$$

Finally, we can fuse the match score of the latent factors Z and the generated data $\hat{X}^v$ in all common views $v \in \{1,\dots,m\}$ to obtain the fusing match score S_F, defined as:

$$S_F = \sum_{v=1}^{m} w_i S_X^v + w_{m+1} S_Z, \tag{3.10}$$

where $w_i, i \in \{1,\cdots,m+1\}$ are the fusing coefficients which can be optimized on a validation set. The probe identity will be classified to the class for which the fusing match score is the largest. Utilizing the information from all the views helps the recognition problem, since different identities may be difficult to recognize in a particular view but may be easier in some other views. We shall elaborate on these points in the following experiment section.

3.4 Experiments

We test our model for data that are under different view points or from different domains. We cropped images to 64×64 for all experiments and data are normalized to be $[-1, 1]$, except for the last gait recognition experiment where data are scaled to be $[0, 1]$. We use Adam optimizer [KB14] with initial learning rate 0.0002 and we run Langevin dynamics for 20 steps in each iteration with step size 0.1.

3.4.1 Rotation and Completion on the Multi-PIE face database

Multi-PIE database [GMC10] is one of the most widely used face dataset which has a wide range of pose and illusion conditions. Specifically, it consists of 3 sessions of images of 249 identities under 15 poses and 20 different illumination conditions. We train our model on a selected subset which covers 5 poses, i.e., $\{-60°, -30°, 0°, 30°, 60°\}$, and 50 randomly selected subjects under all illuminations. The subjects under odd numbers of illumination conditions are used for training and those under even numbers of illumination conditions are used for testing. We show qualitatively that our model can not only rotating faces, but also learn to recover the incomplete faces under different poses.

We build up two generator nets, one for standard pose 0° (g_{θ_1}) and one for another rotated pose (g_{θ_2}). For each generator net, we adopt the structure similar to [RMC15]. Specifically, we learn a 5-layer convNet which has deconvolutional kernel of size 4×4 and of stride 2, and has 512, 256, 128, 64, 3 filters from top to bottom. Each deconvolution layer is followed by ReLU non-linearity and batch normalization [IS15] except the last layer where tanh is used.

Rotation: We first consider generating faces with the same identity and illumination condition but with the desired pose. The input data to our model are the image pairs which consist of the same subjects under the standard pose and the desired pose. In the testing stage, we have image pairs of 10 testing subjects under these two poses, and we use images under the standard pose as our gallery set and the other images as our testing set. After training, given the gallery set, we first run the Langevin dynamics for 200 steps based on g_{θ_1} to get the inferred latent factors. These factors are then fed into the other learned generator

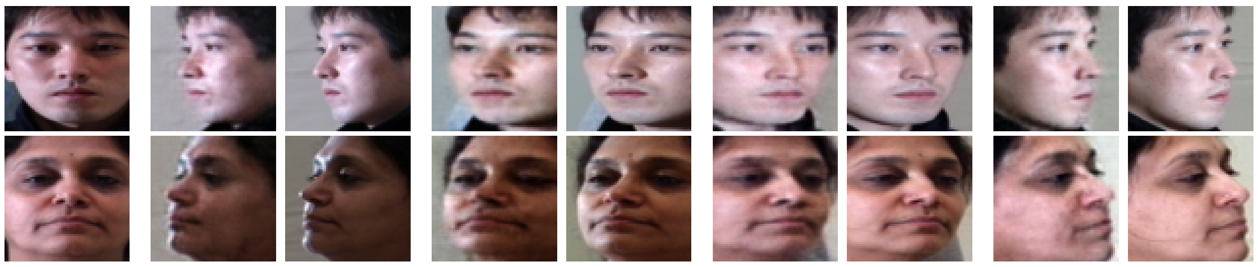

Figure 3.2: Face rotation results for different subjects. First column: face image under standard pose (0°). Second to fifth column pairs: each pair shows the rotated face by our method (left) and the ground truth target (right).

net g_{θ_2} to get the rotated images under the desired pose.

Figure 3.2 shows the qualitative results of rotating face images to $-60°$, $-30°$, $30°$, and $60°$. It can be seen that our shared generator model can accurately generate face images under different poses and they are visually similar to the ground truth testing images.

Completion: The proposed multi-view generator model can also be adapted to the task of image completion in which we only consider the observed or visible pixels for inference.

We experiment with two types of noise patterns: one is the random pixel occlusion where we randomly block 50% or 90% pixels of the training images. The other is random patch occlusion where we randomly place a 20×20 block on each training image. We denote these random patterns as M0.5, M0.9 and P20 respectively.

Figure 3.3 shows completion results. It is clear that even under severe occlusion, the proposed method can still get sharp results. We compute the average per-pixel difference between the recovered images and the corresponding ground truth images to measure the recovery quality. For M0.5, we have recover error 0.256, for M0.9 and P20, we have recover errors 0.404 and 0.328 respectively. Note that we tested the single view generator in the chapter 2, however, the generator is only applied to one front view, whereas the problem of how to efficiently learn multi-views remains unexplored. Our proposed model directly shares the latent factors across different views. Therefore, each sub-domain is forced to communicate with other domains to agree upon the common knowledge. Besides, each sub-domain only contains the images from one view, thus a better generator net can be learned.

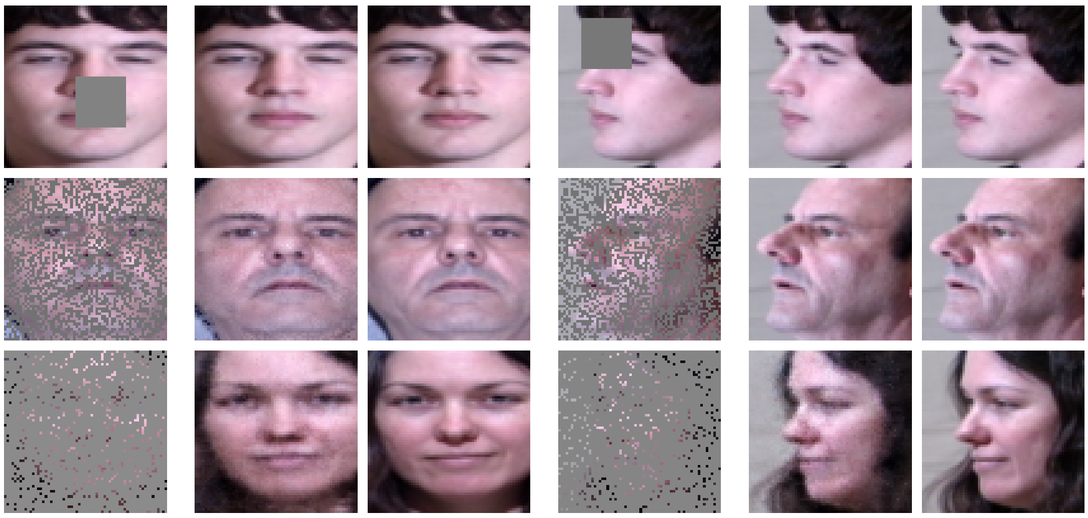

Figure 3.3: Face completion results for different noise patterns. Each row represents the completion results under P20, M0.5 and M0.9 patterns respectively. In each row, the first and fourth columns represent the masked images. The second and fifth columns represent the images after filling in the missing parts using our method. The third and sixth columns represent the ground truth images.

3.4.2 Missing Modality Prediction on OuluVS data

We further evaluate the shared representation learned for both video and audio modalities. Most of the existing methods [NKK11, SS12, BAM17], used spectrogram for audio representation, and require labels for both modalities, so they could hardly generate raw signals. In this dissertation, we directly use *raw* audio and video sequences as inputs and focus on the generation of the missing modality. One feasible and well-known approach for this task is canonical correlation analysis (CCA) [HSS04, XWY16] which finds the *linear* transformations of audio and video data to form a shared representation. Our model can be naturally seen as a *non-linear* version of CCA where two generator nets are trained to find non-linear transformations and latent factors Z are shared across both modalities during inference.

We experiment on OuluVS dataset [ZBP09] which includes audio and pre-extracted mouth ROI sequences for 20 persons speaking 10 phrases 5 times. We circularly add or delete mouth images to make each video clip contains 32 frames, each of size [64, 64]. We also fix the length of audios to be $48,000$ which equals the sample rate, and only keep the first sound channel for simplicity. We use verison 3 of spatial-temporal generator structure for videos (see experiment 5 in section 2.5.1), and 4-layer 1D ConvNet for audios. For audio net, there are 512, 256, 128, 1 filters for each deconvolution layer, and the filter size is 20×1 of stride 10 for all layers. We also utilize batch normalization and ReLU non-linearity except the last layer where tanh is used. For training, we randomly choose 10 persons, each saying 10 phrases 5 times, and use corresponding audios and videos to learn our model and CCA. For testing, we randomly select 3 new persons to: (1) predict the mouth movement based on their audios only ($A \to V$), (2) predict the audio track based on their mouth movements only ($V \to A$). We run 200 steps of Langevin dynamics with stepsize 0.1 for the inference of our model. Table 3.1 shows the mean absolute per-pixel/signal prediction error on 10 different phrases. A lower error indicates a more powerful model for preserving the common information between two modalities.

Experiments		Execuse	Goodby	Hello	How	Nice	Seeyou	Sorry	Thank	Time	Welcome
$V \to A$	CCA	0.1779	0.1828	0.1878	0.1894	0.1870	0.1889	0.1943	0.1842	0.1818	0.1821
	ours	**0.0179**	**0.0243**	**0.0199**	**0.0379**	**0.0210**	**0.0184**	**0.0306**	**0.0164**	**0.0301**	**0.0348**
$A \to V$	CCA	0.4269	0.4468	0.4326	0.6225	0.4549	0.4811	0.5945	0.4472	0.6044	0.5892
	ours	**0.1771**	**0.1860**	**0.1758**	**0.1735**	**0.1791**	**0.1747**	**0.1788**	**0.1713**	**0.1846**	**0.1743**

Table 3.1: Average prediction errors on missing modality for different phrases.

3.4.3 Multi-view Gait Recognition on the CASIA gait database B

The CASIA gait database B [YTT06] is one of the largest multi-view gait databases. It contains 124 subjects captured from 11 viewing angles, namely 0°, 18°, 36°, 54°, 72°, 90°, 108°, 126°, 144°, 162°, and 180°. Under each view, there are ten gait sequences for each person, including six sequences of normal walking, two sequences of walking with a bag, and two sequences of walking with a coat. We experiment on the gait sequences of normal

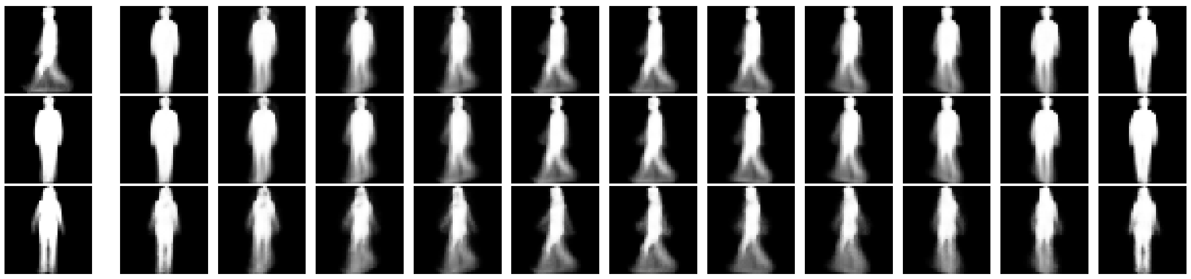

Figure 3.4: Multi-view gait generation results for different subjects. The first column shows the original GEIs, and the second to twelfth columns show the generated GEIs under all the 11 views by our multi-view generator model. Specifically, the first row consists of one subject's original GEI under 90° view, and 11 generated GEIs by forward-propagating the inferred factors into the 11 sub-networks of our multi-view generator model. The second row consists of the same subject's original GEI under 0° view, and 11 generated GEIs under all views. The third row consists of another subject's original GEI under 0° view, and 11 generated GEIs under all views.

walking. Since Gait Energy Image (GEI) [LXW15] is a popular gait feature, which is efficient for computation and is robust to noise, we employ it for our method's input and target images. For this experiment, we build up eleven generator sub-nets, where for each sub-net, we use the similar network structure as in first experiment except that we do not use batch normalization and we use the sigmoid in the last layer.

According to the experimental protocol defined in [YCW17, YCR17], the database is divided into two groups: the first group of the 62 subjects is used for constructing the training set and the remaining 62 subjects are used for performance evaluation. Suppose the gallery view and the probe view are denoted by ω_G and ω_P, respectively. In the training phase, we use the gait sequences from the training group under all the 11 views to alternatively infer the common identity factors and learn the deconvolutional sub-network under each view. In the recognition phase, the sequences from the evaluating group under the gallery view ω_G are utilized to construct the gallery set and the sequences from the evaluating group under the probe view ω_P are utilized to form the probe set. Our method can utilize two kinds of information learned from the proposed model to perform multi-view gait recognition: (1) The

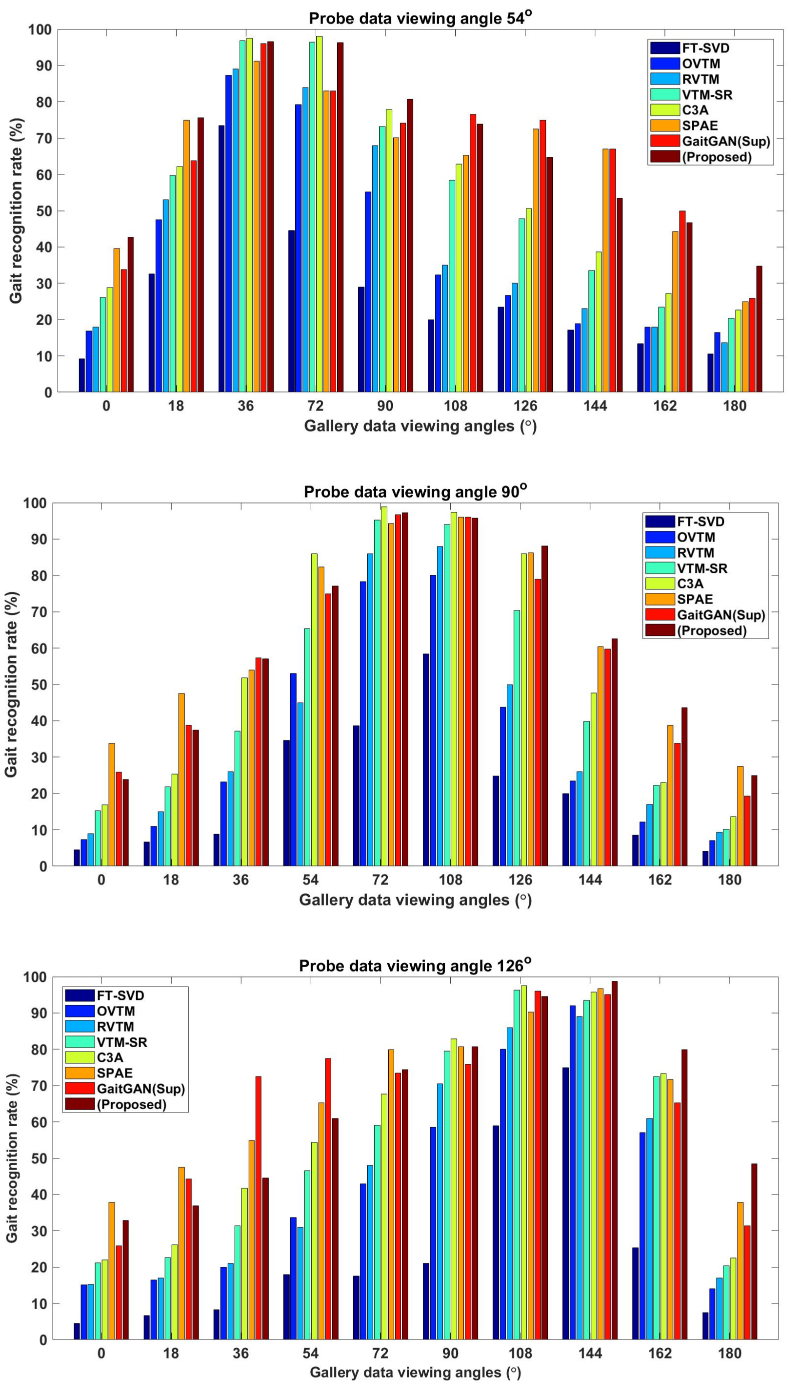

Figure 3.5: Comparisons of gait recognition rates for multi-view gait recognition using different methods. The viewing angles of the probe data in subfigures (a)-(c) are 54°, 90° and 126°, respectively. The viewing angles of the gallery data are the rest 10 viewing angles except the corresponding probe viewing angle.

shared factors (the latent factors Z in Figure 3.1). For both the gallery and probe GEIs, we infer their shared factors and use these factors for recognition. (2) The generation of GEIs. For both the gallery and probe GEIs, we first infer their shared factors, and then generate new GEIs by feeding the inferred latent factors into the sub-networks under some common views. Figure 3.4 shows the generated GEIs through our multi-view generator model. We employ these generated GEIs for further classification. Our method performs a match-score fusion of the above two kinds of information for the final multi-view gait recognition (see section 3.3 for detail).

To evaluate the performance of our algorithm, we compared the proposed model with the following state-of-art methods: FT-SVD [MSM06], OVTM [KWL09], RVTM [ZZH11],VTM-SR [KWZ12], C3A [XWY16], SPAE [YCW17], and GaitGAN [YCR17]. As in experiments of those methods, the probe view ω_P is selected as 54°, 90° or 126°. For each selected probe view, we test on the gallery view ω_G from the rest 10 viewing angles except the corresponding probe view.

The experimental results are plotted in Figure 3.5, which reveals several interesting observations. (1) Most of the algorithms can obtain a high recognition rate when the viewing angle difference between the gallery and probe is small. The reason behind this high recognition rate is that gaits between two closer views share more common information, therefore only simple transformation of gaits is needed to have a good performance. (2) When the angle difference between the gallery and the probe is large, the deep learning gait recognition methods, including SPAE, GaitGAN and the proposed multi-view generator model, perform better than the traditional linear methods, such as FT-SVD, OVTM, RVTM, VTM-SR, and C3A. This is because the deep learning methods can approximate highly non-linear transformations, which are important for this task with large viewpoint variation. (3) Generally speaking, the proposed method performs better then the other two deep learning methods in most of the views. The proposed method can not only improve the recognition rate when the viewpoint variation is not large, but can also handle large viewpoint variation quite well.

In Table 3.2, we further summarize the experimental results of three unsupervised methods, including C3A, SPAE, and the proposed model, which obtain good performances as

Methods\Probe angles	$\theta_P = 54°$	$\theta_P = 90°$	$\theta_P = 126°$	Average
C3A	56.64%	54.65%	58.38%	56.56%
ViDP (Supervised)	64.20%	60.40%	65.00%	63.20%
CNN (Supervised)	77.80%	64.90%	76.10%	72.90%
SPAE	63.31%	62.10%	66.29%	63.90%
GaitGAN (Supervised)	64.52%	58.15%	65.73%	62.80%
Ours	66.53%	60.78%	65.19%	64.17%

Table 3.2: Comparisons of the average recognition accuracy under the probe views $\omega_P = 54°$, $\omega_P = 90°$, $\omega_P = 126°$, where the gallery view are the rest 10 viewing angles except the corresponding probe viewing angle. The values in the right most column are the averages rates at the three probe views $\omega_P = 54°$, $\omega_P = 90°$, $\omega_P = 126°$.

shown in Figure 3.5, where we also include three supervised methods, ViDP [HWZ13], CNN [WHW17], and GaitGAN. It is worth noting that it is generally unfair to directly compare the unsupervised method with the supervised one, since the latter one needs to utilize the label or other auxiliary information for training. As we can observe from Table 3.2, our method, as an unsupervised generative model, obtains comparable results with these state-of-art methods, and even performs better than some supervised methods, such as ViDP and GaitGAN.

3.5 Summary

In this chapter, we propose to learn multi-view generator model through shared latent factors. The learned shared representation can effectively encode the common knowledge across different views, and the domain specific generator networks can accurately obtain the domain related information. We show that the proposed model can be effective in tasks including multi-view rotation and generation, multi-modality prediction as well as multi-view gait recognition.

CHAPTER 4

Joint Learning of Generator Model and Energy-based Model

The generator model can be derived from the view of non-linear factor analysis and learned by alternating back-propagation under maximum likelihood criterion (chapter 2). The model and the algorithm are general and can be effectively adapted to different data forms as well as various tasks (chapter 2 and chapter 3). In this chapter, we shall discuss a principled probabilistic framework that could jointly and tractably learn both the generator model and the energy-based model.

4.1 Learning Deep Probabilistic Models

We shall first briefly review the two probabilistic models, namely the generator model and the energy-based model, both of which are parametrized by convolutional neural networks (ConvNet) [LBB98, KSH12]. Then, we shall present in detail the maximum likelihood learning algorithms for training these two models, respectively. The presentation of the two maximum likelihood learning algorithms in this chapter is unconventional. We seek to derive both algorithms based on the Kullback-Leibler divergence using the same scheme. This will set the stage for the divergence triangle.

We shall use the same notations as defined in section 1.2 of chapter 1. To be more specific, the generator model [GPM14, RMC15, KW13, RMW14a, MG14] is a generalization of the factor analysis model [RT82],

$$z \sim \mathrm{N}(0, I_d),\ x = g_\theta(z) + \epsilon, \tag{4.1}$$

where g_θ is a top-down mapping parametrized by a deep network with parameters θ. It maps the d-dimensional latent vector z to the D-dimensional signal x. $\epsilon \sim \mathrm{N}(0, \sigma^2 I_D)$ and is independent of z. In general, the model is defined by the prior distribution $p(z)$ and the conditional distribution $p_\theta(x|z)$. The complete-data model is $p_\theta(z, x) = p(z)p_\theta(x|z)$. The observed-data model is $p_\theta(x) = \int p_\theta(z, x)dz$. The posterior distribution is $p_\theta(z|x) = p_\theta(z, x)/p_\theta(x)$.

The energy-based model [NCK11, DLW14, LZW16, XLZ16] defines the energy of x through $-f_\alpha(x)$, and assigns a high probability to low energy x. Specifically, we have the following probability model:

$$\pi_\alpha(x) = \frac{1}{Z(\alpha)} \exp\left[f_\alpha(x)\right], \tag{4.2}$$

where $f_\alpha(x)$ is parametrized by a bottom-up deep network with parameters α, and $Z(\alpha)$ is the normalizing constant. We may consider π_α an evaluator, where f_α assigns the value to x, and π_α evaluates x by a normalized probability distribution.

The energy-based model π_α defines explicit log-likelihood via $f_\alpha(x)$, even though $Z(\alpha)$ is intractable. However, it is difficult to sample from π_α. The generator model p_θ can generate x directly by first generating $z \sim p(z)$, and then transforming z to x by $g_\theta(z)$. But it does not define an explicit log-likelihood of x.

In the context of inverse reinforcement learning [ZMB08, AN04] or inverse optimal control, x is action and $-f_\alpha(x)$ defines the cost function or $f_\alpha(x)$ defines the value function or the objective function.

Let $q_{\mathrm{data}}(x)$ be the true distribution that generates the training data. Both the generator p_θ and the energy-based model π_α can be learned by maximum likelihood. For large sample, the maximum likelihood amounts to minimizing the Kullback-Leibler divergence $\mathrm{KL}(q_{\mathrm{data}} \| p_\theta)$ over θ, and minimizing $\mathrm{KL}(q_{\mathrm{data}} \| \pi_\alpha)$ over α, respectively. The expectation $\mathrm{E}_{q_{\mathrm{data}}}$ can be approximated by sample average.

4.1.1 EM-type learning of generator model

To learn the generator model p_θ, we seek to minimize $\mathrm{KL}(q_{\mathrm{data}}(x)\|p_\theta(x))$ over θ. Suppose in an iterative algorithm, the current θ is θ_t. We can fix θ_t at any place we want, and vary θ around θ_t.

We can write

$$\mathrm{KL}(q_{\mathrm{data}}(x)p_{\theta_t}(z|x)\|p_\theta(z,x)) = \mathrm{KL}(q_{\mathrm{data}}(x)\|p_\theta(x)) + \mathrm{KL}(p_{\theta_t}(z|x)\|p_\theta(z|x)). \tag{4.3}$$

In the EM algorithm [DLR77], the left hand side is the surrogate objective function. This surrogate function is more tractable than the true objective function $\mathrm{KL}(q_{\mathrm{data}}(x)\|p_\theta(x))$ because $q_{\mathrm{data}}(x)p_{\theta_t}(z|x)$ is a distribution of the complete data, and $p_\theta(z,x)$ is the complete-data model.

We can write (4.3) as

$$S(\theta) = K(\theta) + \tilde{K}(\theta). \tag{4.4}$$

The geometric picture is that the surrogate objective function $S(\theta)$ is above the true objective function $K(\theta)$, i.e., S majorizes (upper bounds) K, and they touch each other at θ_t, so that $S(\theta_t) = K(\theta_t)$ and $S'(\theta_t) = K'(\theta_t)$. The reason is that $\tilde{K}(\theta_t) = 0$ and $\tilde{K}'(\theta_t) = 0$. See Figure 4.1.

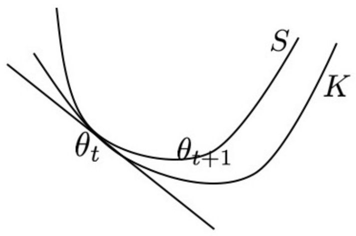

Figure 4.1: The surrogate S majorizes (upper bounds) K, and they touch each other at θ_t with the same tangent.

$q_{\mathrm{data}}(x)p_{\theta_t}(z|x)$ gives us the complete data. Each step of EM fits the complete-data model $p_\theta(z,x)$ by minimizing the surrogate $S(\theta)$,

$$\theta_{t+1} = \arg\min_\theta \mathrm{KL}(q_{\mathrm{data}}(x)p_{\theta_t}(z|x)\|p_\theta(z,x)), \tag{4.5}$$

which amounts to maximizing the complete-data log-likelihood. By minimizing S, we will reduce $S(\theta)$ relative to θ_t, and we will reduce $K(\theta)$ even more, relative to θ_t, because of the majorization picture.

We can also use gradient descent to update θ. Because $S'(\theta_t) = K'(\theta_t)$, and we can place θ_t anywhere, we have

$$-\frac{\partial}{\partial\theta}\mathrm{KL}(q_{\mathrm{data}}(x)\|p_\theta(x)) = \mathrm{E}_{q_{\mathrm{data}}(x)p_\theta(z|x)}\left[\frac{\partial}{\partial\theta}\log p_\theta(z,x)\right]. \tag{4.6}$$

To implement the above updates, we need to compute the expectation with respect to the posterior distribution $p_\theta(z|x)$. It can be approximated by MCMC such as Langevin dynamics or HMC [Nea11]. Both require gradient computations that can be efficiently accomplished by back-propagation. We have learned the generator using such learning method in chapter 2.

4.1.2 Self-critic learning of energy-based model

To learn the energy-based model π_α, we seek to minimize $\mathrm{KL}(q_{\mathrm{data}}(x)\|\pi_\alpha(x))$ over α. Suppose in an iterative algorithm, the current α is α_t. We can fix α_t at any place we want, and vary α around α_t.

Consider the following contrastive divergence

$$\mathrm{KL}(q_{\mathrm{data}}(x)\|\pi_\alpha(x)) - \mathrm{KL}(\pi_{\alpha_t}(x)\|\pi_\alpha(x)). \tag{4.7}$$

We can use the above as surrogate function, which is more tractable than the true objective function, since the $\log Z(\theta)$ term is canceled out. Specifically, we can write (4.7) as

$$\begin{aligned} S(\alpha) &= K(\alpha) - \tilde{K}(\alpha) & (4.8)\\ &= -(\mathrm{E}_{q_{\mathrm{data}}}[f_\alpha(x)] - \mathrm{E}_{\pi_{\alpha_t}}[f_\alpha(x)]) + \text{const.} & (4.9)\end{aligned}$$

The geometric picture is that the surrogate function $S(\alpha)$ is below the true objective function $K(\alpha)$, i.e., S minorizes (lower bounds) K, and they touch each other at α_t, so that $S(\alpha_t) = K(\alpha_t)$, and $S'(\alpha_t) = K'(\alpha_t)$. The reason is that $\tilde{K}(\alpha_t) = 0$ and $\tilde{K}'(\alpha_t) = 0$. See Figure 4.2.

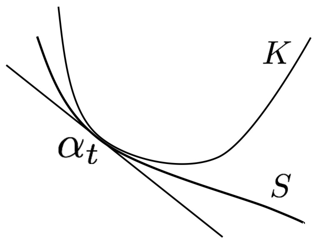

Figure 4.2: The surrogate S minorizes (lower bounds) K, and they touch each other at α_t with the same tangent.

Because S minorizes K, we do not have a EM-like update. However, we can still use gradient descent to update α, where the derivative is

$$K'(\alpha_t) = S'(\alpha_t) = -(\mathrm{E}_{q_{\mathrm{data}}}[f'_{\alpha_t}(x)] - \mathrm{E}_{\pi_{\alpha_t}}[f'_{\alpha_t}(x)]), \tag{4.10}$$

where

$$f'_{\alpha_t}(x) = \frac{\partial}{\partial \alpha} f_\alpha(x)\Big|_{\alpha_t}. \tag{4.11}$$

Since we can place α_t anywhere, we have

$$-\frac{\partial}{\partial \alpha}\mathrm{KL}(q_{\mathrm{data}}(x)\|\pi_\alpha(x)) = \mathrm{E}_{q_{\mathrm{data}}}\left[\frac{\partial}{\partial \alpha} f_\alpha(x)\right] - \mathrm{E}_{\pi_\alpha}\left[\frac{\partial}{\partial \alpha} f_\alpha(x)\right]. \tag{4.12}$$

To implement the above update, we need to compute the expectation with respect to the current model π_{α_t}. It can be approximated by MCMC such as Langevin dynamics or HMC that samples from π_{α_t}. It can be efficiently implemented by gradient computation via back-propagation. [LZW16, XLZ16] trained the energy-based model using such learning method.

The above learning algorithm has an adversarial interpretation. Updating α_t to α_{t+1} by following the gradient of $S(\alpha) = \mathrm{KL}(q_{\mathrm{data}}(x)\|\pi_\alpha(x)) - \mathrm{KL}(\pi_{\alpha_t}(x)\|\pi_\alpha(x)) = -(\mathrm{E}_{q_{\mathrm{data}}}[f_\alpha(x)] - \mathrm{E}_{\pi_{\alpha_t}}[f_\alpha(x)]) + \text{const}$, we seek to decrease the first KL-divergence, while we will increase the second KL-divergence, or we seek to shift the value function $f_\alpha(x)$ toward the observed data and away from the synthesized data generated from the current model. That is, the model π_α criticizes its current version π_{α_t}, i.e., the model is its own adversary or its own critic.

4.1.3 Similarity and difference

In both models, at θ_t or α_t, we have $S = K$, $S' = K'$, because $\tilde{K} = 0$ and $\tilde{K}' = 0$.

The difference is that in the generator model, $S = K + \tilde{K}$, whereas in energy-based model, $S = K - \tilde{K}$.

In the generator model, if we replace the intractable $p_{\theta_t}(z|x)$ by the inference model $q_\phi(z|x)$, we get VAE.

In energy-based model, if we replace the intractable $\pi_{\alpha_t}(x)$ by the generator $p_\theta(x)$, we get adversarial contrastive divergence (ACD). The negative sign in front of $\tilde{K}$ is the root of the adversarial learning.

4.2 Divergence Triangle

In this section, we shall first present the divergence triangle, emphasizing its compact symmetric and anti-symmetric form. Then, we shall show that it is an re-interpretation and integration of existing methods, in particular, VAE [KW13, RMW14a, MG14] and ACD [KB16, DAB17].

Suppose we observe training examples $\{x_{(i)} \sim q_{\text{data}}(x)\}_{i=1}^n$ where $q_{\text{data}}(x)$ is the unknown data distribution. $\pi_\alpha(x) \propto \exp[f_\alpha(x)]$ with energy function $-f_\alpha$ denotes the energy-based model with parameters α. The generator model $p(z)p_\theta(x|z)$ has parameters θ and latent vector z. It is trivial to sample the latent distribution $p(z)$ and the generative process is defined as $z \sim p(z)$, $x \sim p_\theta(x|z)$.

The maximum likelihood learning algorithms for both the generator and energy-based model require MCMC sampling. We modify the maximum likelihood KL-divergences by proposing a divergence triangle criterion, so that the two models can be learned jointly without MCMC. In addition to the generator p_θ and energy-based model π_α, we also include an inference model $q_\phi(z|x)$ in the learning scheme. Such an inference model is a key component in the variational auto-encoder [KW13, RMW14a, MG14]. The inference model $q_\phi(z|x)$ with parameters ϕ maps from the data space to latent space. In the context of EM, $q_\phi(z|x)$ can be considered an imputor that imputes the missing data z to get the complete data (z, x).

The three models above define joint distributions over z and x from different perspectives.

The two marginals, i.e., empirical data distribution $q_{\text{data}}(x)$ and latent prior distribution $p(z)$, are known to us. The goal is to harmonize the three joint distributions so that the competition and cooperation between different loss terms improves learning.

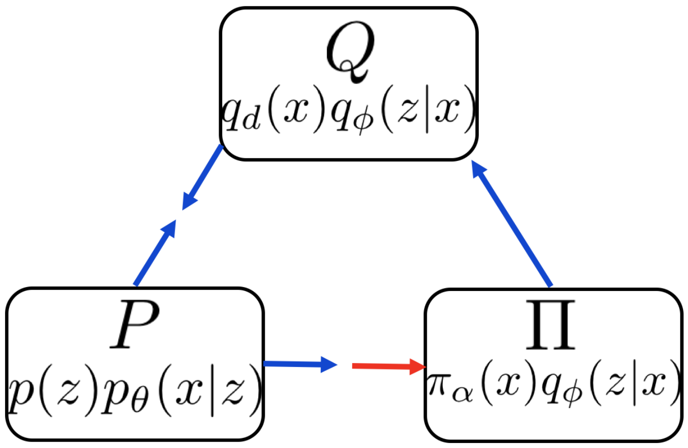

Figure 4.3: Divergence triangle is based on the Kullback-Leibler divergences between three joint distributions of (z, x). The blue arrow indicates the “running toward” behavior and the red arrow indicates the “running away” behavior.

The divergence triangle involves the following three joint distributions on (z, x):

1. Q-distribution: $Q(z, x) = q_{\text{data}}(x)q_\phi(z|x)$.

2. P-distribution: $P(z, x) = p(z)p_\theta(x|z)$.

3. Π-distribution: $\Pi(z, x) = \pi_\alpha(x)q_\phi(z|x)$.

We propose to learn the three models p_θ, π_α, q_ϕ by the following divergence triangle loss functional $\mathcal{D}$

$$\begin{aligned}
&\max_\alpha \min_\theta \min_\phi \mathcal{D}(\alpha, \theta, \phi), \\
&\mathcal{D} = \text{KL}(Q\|P) + \text{KL}(P\|\Pi) - \text{KL}(Q\|\Pi).
\end{aligned} \tag{4.13}$$

See Figure 4.3 for illustration. The divergence triangle is based on the three KL-divergences between the three joint distributions on (z, x). It has a symmetric and anti-symmetric form, where the anti-symmetry is due to the negative sign in front of the last

KL-divergence and the maximization over α. The divergence triangle leads to the following dynamics between the three models: (1) Q and P seek to get close to each other. (2) P seeks to get close to Π. (3) π seeks to get close to q_{data}, but it seeks to get away from P, as indicated by the red arrow. Note that $\text{KL}(Q\|\Pi) = \text{KL}(q_{\text{data}}\|\pi_\alpha)$, because $q_\phi(z|x)$ is canceled out. The effect of (2) and (3) is that π gets close to q_{data}, while inducing P to get close to q_{data} as well, or in other words, P chases π_α toward q_{data}.

4.2.1 Unpacking the Loss Function

The divergence triangle integrates variational and adversarial learning methods, which are modifications of maximum likelihood.

4.2.1.1 Variational Learning

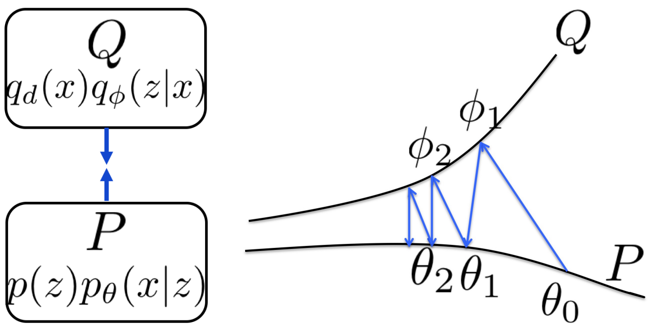

Figure 4.4: Variational auto-encoder (VAE) as joint minimization by alternating projection. Left: Interaction between the models. Right: Alternating projection. The two models run toward each other.

First, $\min_\theta \min_\phi \text{KL}(Q\|P)$ captures the variational auto-encoder (VAE).

$$\text{KL}(Q\|P) = \text{KL}(q_{\text{data}}(x)\|p_\theta(x)) + \text{KL}(q_\phi(z|x)\|p_\theta(z|x)), \tag{4.14}$$

Recall $S = K + \tilde{K}$ in (4.4), if we replace the intractable $p_{\theta_t}(z|x)$ in (4.4) by the explicit $q_\phi(z|x)$, we get (4.14), so that we avoid MCMC for sampling $p_{\theta_t}(z|x)$.

We may interpret VAE as alternating projection between Q and P. See Figure 4.4 for illustration. If $q_\phi(z|x) = p_\theta(z|x)$, the algorithm reduces to the EM algorithm. The wake-sleep algorithm [HDF95] is similar to VAE, except that it updates ϕ by $\min_\phi \text{KL}(P\|Q)$ instead of $\min_\phi \text{KL}(Q\|P)$, so that the wake-sleep algorithm does not have a single objective function. The VAE $\min_\theta \min_\phi \text{KL}(Q\|P)$ defines a cooperative game, with the dynamics that q_ϕ and p_θ run toward each other.

4.2.1.2 Adversarial Learning

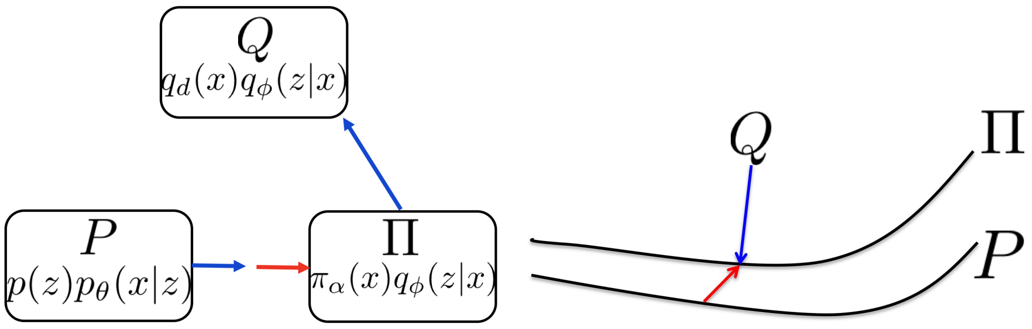

Figure 4.5: Adversarial contrastive divergence (ACD). Left: Interaction between the models. Red arrow indicates a chasing game, where the generator model chases the energy-based model, which runs toward the data distribution. Right: Contrastive divergence.

Next, consider the learning of the energy-based model model [KB16, DAB17]. Recall $S = K - \tilde{K}$ in (4.8), if we replace the intractable $\pi_{\alpha_t}(x)$ in (4.8) by $p_\theta(x)$, we get

$$\min_\alpha \max_\theta [\text{KL}(q_{\text{data}}(x)\|\pi_\alpha(x)) - \text{KL}(p_\theta(x)\|\pi_\alpha(x))], \tag{4.15}$$

or equivalently

$$\max_\alpha \min_\theta [\text{KL}(p_\theta(x)\|\pi_\alpha(x)) - \text{KL}(q_{\text{data}}(x)\|\pi_\alpha(x))], \tag{4.16}$$

so that we avoid MCMC for sampling $\pi_{\alpha_t}(x)$, and the gradient for updating α becomes

$$\frac{\partial}{\partial\alpha}[\text{E}_{q_{\text{data}}}(f_\alpha(x)) - \text{E}_{p_\theta}(f_\alpha(x))]. \tag{4.17}$$

Because of the negative sign in front of the second KL-divergence in (4.15), we need $\max_\theta$ in (4.15) or $\min_\theta$ in (4.16), so that the learning becomes adversarial. See Figure 4.5 for illustration. Inspired by [Hin02b], we call (4.15) the adversarial contrastive divergence (ACD). It underlies [KB16, DAB17].

The adversarial form (4.15) or (4.16) defines a chasing game with the following dynamics: the generator p_θ chases the energy-based model π_α in $\min_\theta \text{KL}(p_\theta \| \pi_\alpha)$, the energy-based model π_α seeks to get closer to q_{data} and get away from p_θ. The red arrow in Figure 4.5 illustrates this chasing game. The result is that π_α lures p_θ toward q_{data}. In the idealized case, p_θ always catches up with π_α, then π_α will converge to the maximum likelihood estimate $\min_\alpha \text{KL}(q_{\text{data}} \| \pi_\alpha)$, and p_θ converges to π_α.

The above chasing game is different from VAE $\min_\theta \min_\phi \text{KL}(Q\|P)$, which defines a cooperative game where q_ϕ and p_θ run toward each other.

Even though the above chasing game is adversarial, both models are running toward the data distribution. While the generator model runs after the energy-based model, the energy-based model runs toward the data distribution. As a consequence, the energy-based model guides or leads the generator model toward the data distribution. It is different from GAN [GPM14]. In GAN, the discriminator eventually becomes a confused one because the generated data become similar to the real data. In the above chasing game, the energy-based model becomes close to the data distribution.

The updating of α by (4.17) bears similarity to Wasserstein GAN (WGAN) [ACB17], but unlike WGAN, f_α defines a probability distribution π_α, and the learning of θ is based on $\min_\theta \text{KL}(p_\theta(x) \| \pi_\alpha(x))$, which is a variational approximation to π_α. This variational approximation only requires knowing $f_\alpha(x)$, without knowing $Z(\alpha)$. However, unlike $q_\phi(z|x)$, $p_\theta(x)$ is still intractable, in particular, its entropy does not have a closed form. Thus, we can again use variational approximation, by changing the problem to $\min_\theta \min_\phi \text{KL}(p(z)p_\theta(x|z) \| \pi_\alpha(x) q_\phi(z|x))$, i.e., $\min_\theta \min_\phi \text{KL}(P\|\Pi)$, which is analytically tractable and which underlies [DAB17]. In fact,

$$\text{KL}(P\|\Pi) = \text{KL}(p_\theta(x) \| \pi_\alpha(x)) + \text{KL}(p_\theta(z|x) \| q_\phi(z|x)). \tag{4.18}$$

Thus, we can modify (4.16) into $\max_\alpha \min_\theta \min_\phi [\text{KL}(P\|\Pi) - \text{KL}(Q\|\Pi)]$, because again we have $\text{KL}(Q\|\Pi) = \text{KL}(q_{\text{data}} \| \pi_\alpha)$.

Fitting the above together, we have the divergence triangle (4.13), which has a compact symmetric and anti-symmetric form.

4.2.2 Gap and Two-side of KL-divergences

We can write the objective function $\mathcal{D}$ as

$$\begin{aligned}\mathcal{D} &= (\mathrm{KL}(q_{\mathrm{data}}(x)\|p_\theta(x)) + \mathrm{KL}(q_\phi(z|x)\|p_\theta(z|x))) \\ &\quad - (\mathrm{KL}(q_{\mathrm{data}}(x)\|\pi_\alpha(x)) - \mathrm{KL}(p(z)p_\theta(x|z)\|\pi_\alpha(x)q_\phi(z|x))) \\ &= ((\mathrm{KL}(q_{\mathrm{data}}(x)\|p_\theta(x)) - \mathrm{KL}(q_{\mathrm{data}}(x)\|\pi_\alpha(x))) \\ &\quad + \mathrm{KL}(q_\phi(z|x)\|p_\theta(z|x)) + \mathrm{KL}(p(z)p_\theta(x|z)\|\pi_\alpha(x)q_\phi(z|x)).\end{aligned}$$

Thus $\mathcal{D}$ is an upper bound of the difference between the log-likelihood of the energy-based model and the log-likelihood of the generator model.

In the divergence triangle, the generator model appears on the right side of $\mathrm{KL}(Q\|P)$, and it also appears on the left side of $\mathrm{KL}(P\|\Pi)$. The former tends to interpolate or smooth the modes of Q, while the later tends to seek after major modes of Π while ignoring minor modes. As a result, the learned generator model tends to generate sharper images. As to the inference model $q_\phi(z|x)$, it appears on the left side of $\mathrm{KL}(Q\|P)$, and it also appears on the right side of $\mathrm{KL}(P\|\Pi)$. The former is variational learning of the real data, while the latter corresponds to the sleep phase of wake-sleep learning, which learns from the dream data generated by P. The inference model thus can infer z from both observed x and generated x.

In fact, if we define

$$\mathcal{D}_0 = \mathrm{KL}(q_{\mathrm{data}}\|p_\theta) + \mathrm{KL}(p_\theta\|\pi_\alpha) - \mathrm{KL}(q_{\mathrm{data}}\|\pi_\alpha), \tag{4.19}$$

we have

$$\mathcal{D} = \mathcal{D}_0 + \mathrm{KL}(q_\phi(z|x)\|p_\theta(z|x)) + \mathrm{KL}(p_\theta(z|x)\|q_\phi(z|x)). \tag{4.20}$$

(4.19) is the divergence triangle between the three marginal distributions on x, where p_θ appears on both sides of KL-divergences. (4.20) is the variational scheme to make the marginal distributions into the joint distributions, which are more tractable. In (4.20), the two KL-divergences have reverse orders.

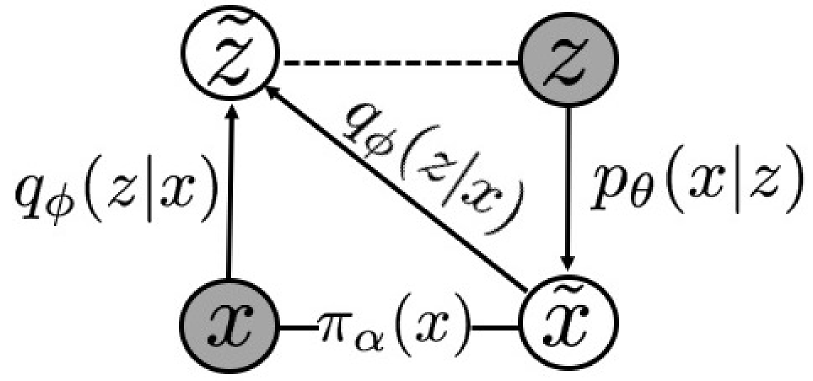

Figure 4.6: Joint learning of three models. The shaded circles z and x represent variables that can be sampled from the true distributions, i.e., $\mathrm{N}(0, I_d)$ and empirical data distribution, respectively. $\tilde{x}$ and $\tilde{z}$ are generated samples using the generator model and the inference model, respectively. The solid line with arrow represents the conditional mapping and dashed line indicates the matching loss is involved.

4.2.3 Training

The three models are each parameterized by convolutional neural networks. The joint learning under the divergence triangle can be implemented by stochastic gradient descent, where the expectations are replaced by the sample averages. Algorithm 2 describes the procedure which is illustrated in Figure 4.6.

4.3 Experiments

In this section, we demonstrate not only that the divergence triangle is capable of successfully learning an energy-based model with a well-behaved energy landscape, a generator model with highly realistic samples, and an inference model with faithful reconstruction ability, but we also show competitive performance on four tasks: image generation, test image reconstruction, energy landscape mapping, and learning from incomplete images. For image generation, we consider spatial stationary texture images, temporal stationary dynamic textures, and general object categories. We also test our model on large-scale datasets and high-resolution images.

Algorithm 2 Joint Training for Divergence Triangle Model

```
Require:
    training images $\{x_{(i)}\}_{i=1}^{n}$,
    number of learning iterations $T$,
    $\alpha$, $\theta$, $\phi \leftarrow$ initialized network parameters.
Ensure:
    estimated parameters $\{\alpha, \theta, \phi\}$,
    generated samples $\{\tilde{x}_{(i)}\}_{i=1}^{\tilde{n}}$.

1:  Let $t \leftarrow 0$.
2:  repeat
3:      $\{z_{(i)} \sim p(z)\}_{i=1}^{\tilde{M}}$.
4:      $\{\tilde{x}_{(i)} \sim p_\theta(x|z_{(i)})\}_{i=1}^{\tilde{M}}$.
5:      $\{x_{(i)} \sim q_{\text{data}}(x))\}_{i=1}^{M}$.
6:      $\{\tilde{z}_{(i)} \sim q_\phi(z|x_{(i)})\}_{i=1}^{M}$.
7:      $\alpha$-step: Given $\{\tilde{x}_{(i)}\}_{i=1}^{\tilde{M}}$ and $\{x_{(i)}\}_{i=1}^{M}$,
        update $\alpha \leftarrow \alpha + \eta_\alpha \frac{\partial}{\partial \alpha}\mathcal{D}$ with learning rate $\eta_\alpha$.
8:      $\phi$-step: Given $\{(z_{(i)}, \tilde{x}_{(i)})\}_{i=1}^{\tilde{M}}$ and $\{(\tilde{z}_{(i)}, x_{(i)})\}_{i=1}^{M}$,
        update $\phi \leftarrow \phi - \eta_\phi \frac{\partial}{\partial \phi}\mathcal{D}$, with learning rate $\eta_\phi$.
9:      $\theta$-step: Given $\{(z_{(i)}, \tilde{x}_{(i)})\}_{i=1}^{\tilde{M}}$ and $\{(\tilde{z}_{(i)}, x_{(i)})\}_{i=1}^{M}$,
        update $\theta \leftarrow \theta - \eta_\theta \frac{\partial}{\partial \theta}\mathcal{D}$, with learning rate $\eta_\theta$
        (optional: multiple-step update).
10:     Let $t \leftarrow t + 1$.
11: until $t = T$
```

The images are resized and scaled to $[-1, 1]$, no further pre-processing is needed. The network parameters are initialized with zero-mean Gaussian with standard deviation 0.02 and optimized using Adam [KB14]. Network weights are decayed with rate 0.0005, and batch normalization [IS15] is used.

4.3.1 Generation

In this experiment, we evaluate the visual quality of generator samples from our divergence triangle model. If the generator model is well-trained, then the obtained samples should be realistic and match the visual features and contents of training images.

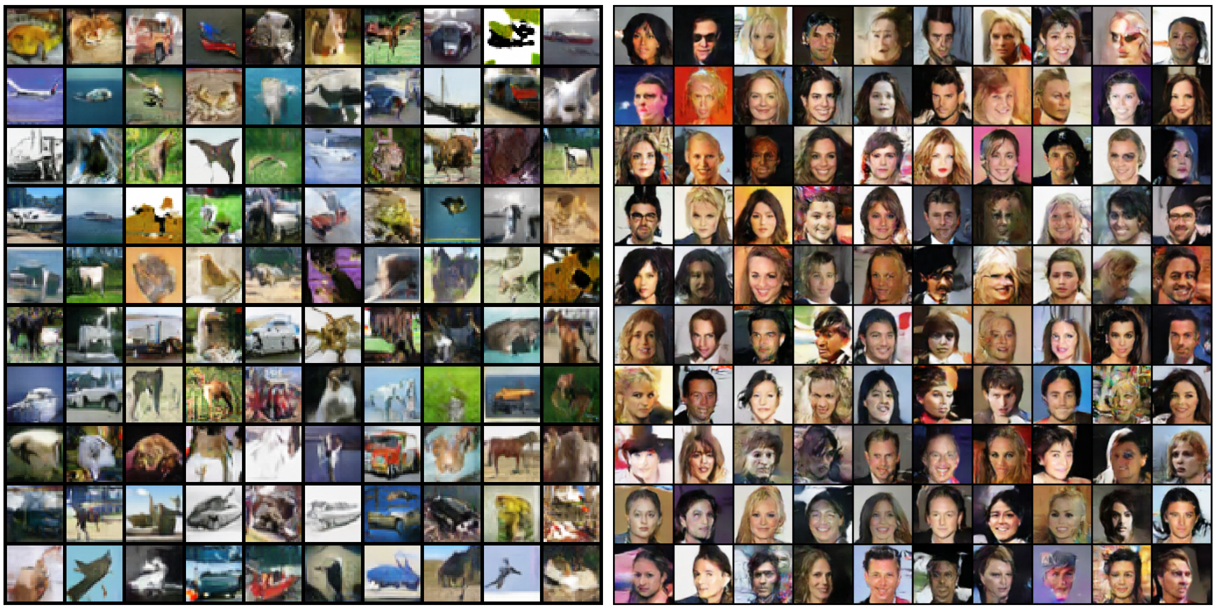
Figure 4.7: Generated samples. Left: generated samples on CIFAR-10 dataset. Right: generated samples on CelebA dataset.

Object Generation. For object categories, we test our model on two commonly-used datasets of natural images: CIFAR-10 and CelebA [LLW15b]. For CelebA face dataset, we randomly select 9,000 images for training and another 1,000 images for testing in reconstruction task. The face images are resized to 64×64 and CIFAR-10 images remain 32×32.

Model	VAE	DCGAN	WGAN	CoopNet	CEGAN	ALI	ALICE	Ours
CIFAR-10 (IS)	4.08	6.16	5.76	6.55	7.07	5.93	6.02	**7.23**
CelebA (FID)	99.09	38.39	36.36	56.57	41.89	60.29	46.14	**31.92**

Table 4.1: Sample quality evaluation of VAE [KW13], WGAN [ACB17], CoopNet [XLG18], CEGAN [DAB17], ALI [DBP16], and ALICE [LLC17]. Row 1: Inception scores for CIFAR-10. Row 2: FID scores for CelebA.

For CelebA data, we use 5 layer generator model start from 100-dim latent factor. We use kernel size 4 with stride 1 for the first layer and stride 2 for the following layers. Each layer have 512, 512, 256, 128, 3 filters respectively. The inference net has the "mirror" structure as generator model. We build the last layer separately to model the posterior mean and variance. For the energy-based model, we use the same structure as the inference net. For the CIFAR-10 data, we use the same network structure as for CelebA, but we use stride 1 for the bottom layer of generator, top layer of inference net as well as energy-based model, and we do not use batch normalization for inference and energy-based model.

The qualitative results of generated samples for objects are shown in Figure 4.7. We further evaluate our model using quantitative evaluations which are based on the Inception Score (IS) [SGZ16] for CIFAR-10 and Frechet Inception Distance (FID) [LKM17] for CelebA faces. We generate 50,000 random samples for the computation of the inception score and 10,000 random samples for the computation of the FID score. Table 4.1 shows the IS and FID scores of our model compared with VAE [KW13], DCGAN [RMC15], WGAN [ACB17], CoopNet [XLG18], CEGAN [DAB17], ALI [DBP16], ALICE [LLC17].

Note that for the Inception Score on CIFAR-10, we borrowed the scores from relevant papers, and for FID score on 9,000 CelebA faces, we re-implemented or used the available code with the similar network structure as our model. It can be seen that our model achieves the competitive performance compared to recent baseline models.

Large-scale Dataset. We also train our model on large scale datasets including down-sampled 32×32 version of ImageNet [OKK16, RDS15] (roughly 1 million images) and

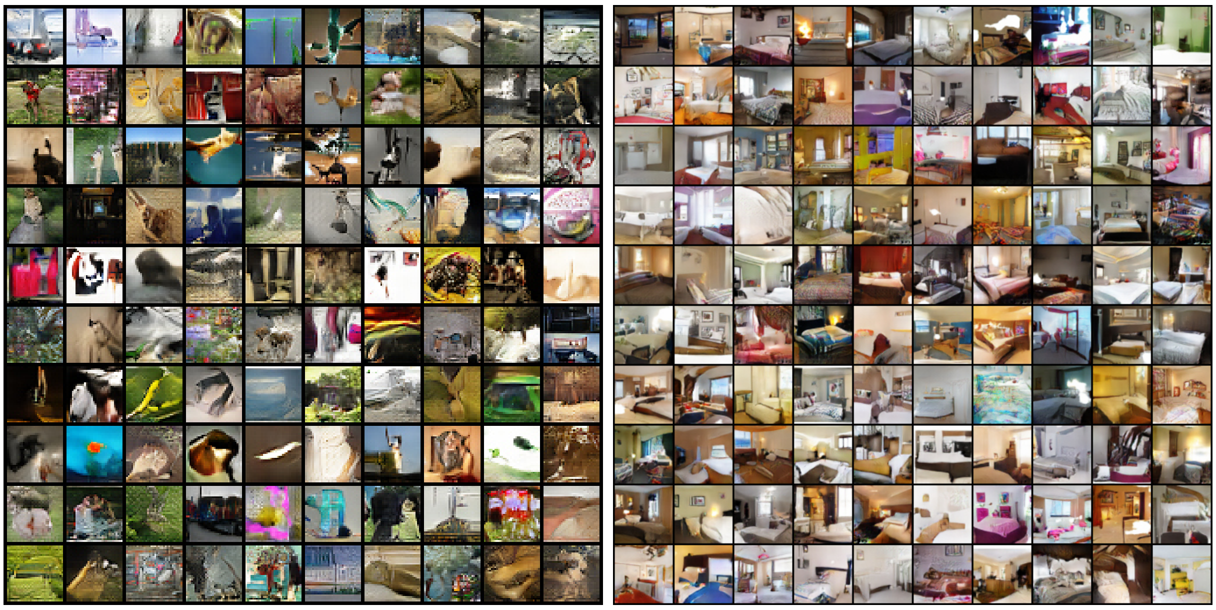
Figure 4.8: Generated samples. Left: 32×32 ImageNet. Right: 64×64 LSUN (bedroom).

Large-scale Scene Understand (LSUN) dataset [YSZ15]. For the LSUN dataset, we consider the *bedroom*, *tower* and *Church ourdoor* categories which contains roughly 3 million, 0.7 million and 0.1 million images and were re-sized to 64×64. The network structures are similar with the ones used in object generation with twice the number of channels and batch normalization is used in all three models. Generated samples are shown on Figure 4.8.

High-resolution Synthesis.

In this section, we recruit a layer-wise training scheme to learn models on CelebA-HQ [KAL17] with resolutions of up to $1,024 \times 1,024$ pixels. Layer-wise training dates back to initializing deep neural networks by Restricted Boltzmann Machines to overcome optimization hurdles [HS06, BLP07] and has been resurrected in progressive GANs [KAL17], albeit the order of layer transitions is reversed such that top layers are trained first. This resembles a Laplacian Pyramid [DCF15] in which images are generated in a coarse-to-fine fashion.

As in [KAL17], the training starts with down-sampled images with a spatial resolution of 4×4 while progressively increasing the size of the images and number of layers. All three models are grown in synchrony where 1×1 convolutions project between RGB and feature. In contrast to [KAL17], we do not require mini-batch discrimination to increase variation of

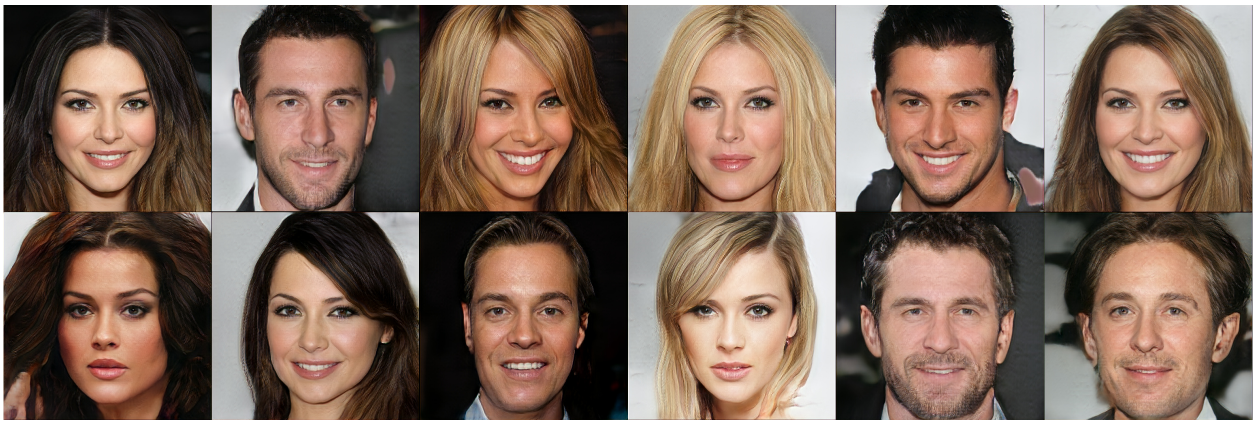

Figure 4.9: Generated samples with $1,024 \times 1,024$ resolution drawn from $g_\theta(z)$ with 512-dimensional latent vector $z \sim N(0, I_d)$ for Celeba-HQ.

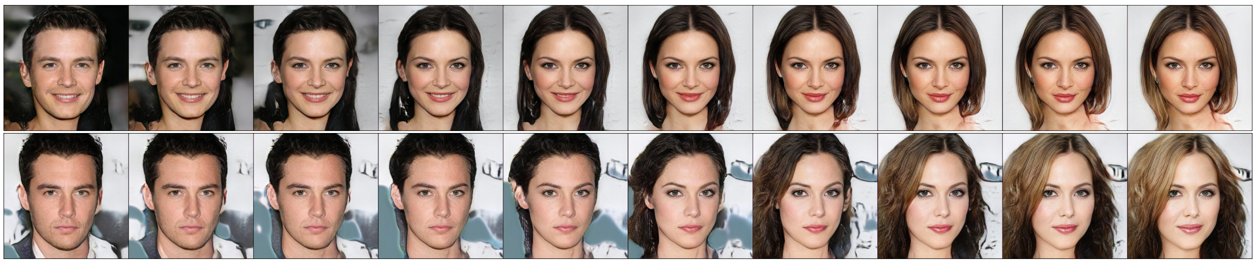

Figure 4.10: High-resolution synthesis from the generator model $g_\theta(z)$ with linear interpolation in latent space (i.e., $(1-\alpha) \cdot z_0 + \alpha \cdot z_1$) for Celeba-HQ.

$g_\theta(\cdot)$ nor gradient penalty to preserve 1-Lipschitz continuity of $f_\alpha(\cdot)$.

Figure 4.9 depicts high-fidelity synthesis in a resolution of $1,024 \times 1,024$ pixels sampled from the generator model $g_\theta(z)$ on CelebA-HQ. Figure 4.10 illustrates linear interpolation in latent space (i.e., $(1-\alpha) \cdot z_0 + \alpha \cdot z_1$), which indicates diversity in the samples.

Therefore, the joint learning in the triangle formulation is not only able to train the three models with stable optimization, but it also achieves synthesis with high fidelity.

Texture Synthesis. We consider texture images, which are spatial stationary and contain repetitive patterns. The texture images are resized to 224×224. Separate models are trained on each image. We start from the latent factor of size $7 \times 7 \times 5$ and use five convolutional-transpose layers with kernel size 4 and up-sampling factor 2 for the generator network. The

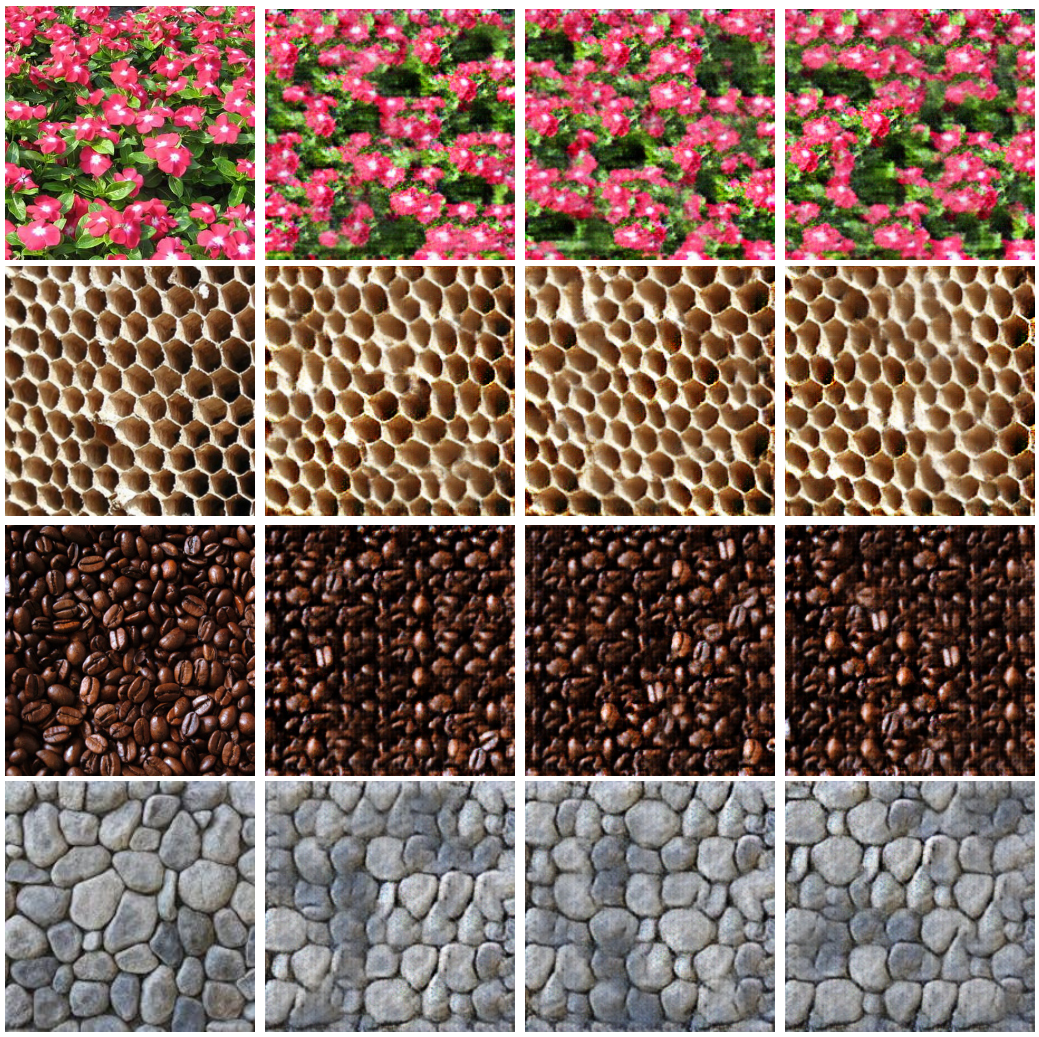

Figure 4.11: Generated texture patterns. For each row, the left one is the training texture, the remaining images are 3 textures generated by divergence triangle.

layers have 512, 512, 256, 128 and 3 filters, respectively, and ReLU non-linearity between each layer is used. The inference model has the inverse or "mirror" structure of generator model except that we use convolutional layers and ReLU with leak factor 0.2. The energy-based model has three convolutional layers. The first two layers have kernel size 7 with stride 2 for 100 and 70 filters respectively, and the last layer has 30 filters with kernel size 5 and stride 1.

The representative examples are shown in Figure 4.11. Three texture synthesis results are obtained by sampling different latent factors from prior distribution $p(z)$. Notice that although we only have one texture image for training, the proposed triangle divergence model can effectively utilize the repetitive patterns, thus generating realistic texture images with different configurations.

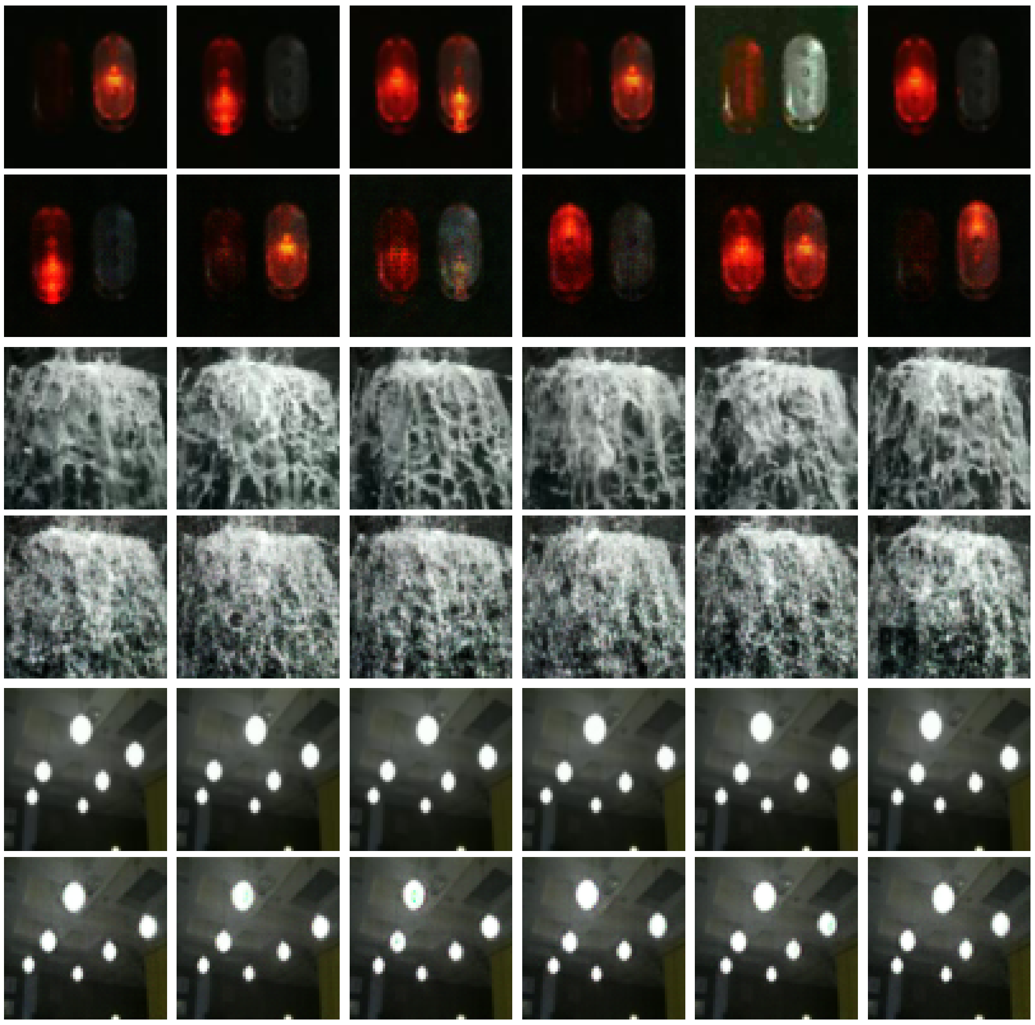

Figure 4.12: Generated dynamic texture patterns. The top row shows the frames from the training video, the bottom row represents the frames for the generated video.

Dynamic texture synthesis. Our model can also be used for dynamic patterns which exhibit stationary regularity in the temporal domain. The training video clips are selected from Dyntex database [PFH10] and resized to 64 pixels $\times$ 64 pixels $\times$ 32 frames. We adopt spatial-temporal models for dynamic patterns that are stationary in the temporal domain but non-stationary in the spatial domain. Specifically, we start from 10 latent factors of size

$1 \times 1 \times 2$ for each video clip and we adopt the same spatial-temporal convolutional transpose generator network as described in chapter 2 except we use kernel size 5 for the second layer. For the inference model, we use 5 spatial-temporal convolutional layers. The first 4 layers have kernel size 4 with upsampling factor 2 and the last layer is fully-connected in spatial domain but convolutional in the temporal domain, yielding re-parametrized μ_ϕ and σ_ϕ which have the same size the as latent factors. For the energy-based model, we use three spatial-temporal convolutional layers. The first two layers have kernel size 4 with up-sample factor 2 in all directions, but the last layer is fully-connected in the spatial domain but convolutional with kernel size 4 and upsample by 2 in the temporal domain. Each layer has 64, 128 and 128 filters, respectively. Some of the synthesis results are shown in Figure 4.12. Note, we sub-sampled 6 frames of the training and generated video clips and we only show them in the first batch for illustration.

4.3.2 Test Image Reconstruction

.

Model	WS	VAE	ALI	ALICE	Ours
CIFAR-10	0.058	0.037	0.311	0.034	**0.028**
CelebA	0.152	0.039	0.519	0.046	**0.030**

Table 4.2: Test reconstruction evaluation for WS [HDF95], VAE [KW13], ALI [DBP16], ALICE [LLC17]. Row 1: MSE for CIFAR-10 test set. Row 2: MSE for 1,000 hold out set from CelebA.

In this experiment, we evaluate the reconstruction ability of our model for a hold-out testing image dataset. This is a strong indicator for the accuracy of our inference model. Specifically, if our divergence triangle model $\mathcal{D}$ is well-learned, then the inference model should match the true posterior of generator model, i.e., $q_\phi(z|x) \approx p_\theta(z|x)$. Therefore, given test signal x_{te}, its reconstruction $\tilde{x_{te}}$ should be close to x_{te}, i.e., $x_{te} \xrightarrow{q_\phi} z_{te} \xrightarrow{p_\theta} \tilde{x_{te}} \approx x_{te}$. Figure 4.13 shows the testing images and their reconstructions on CIFAR-10 and CelebA.

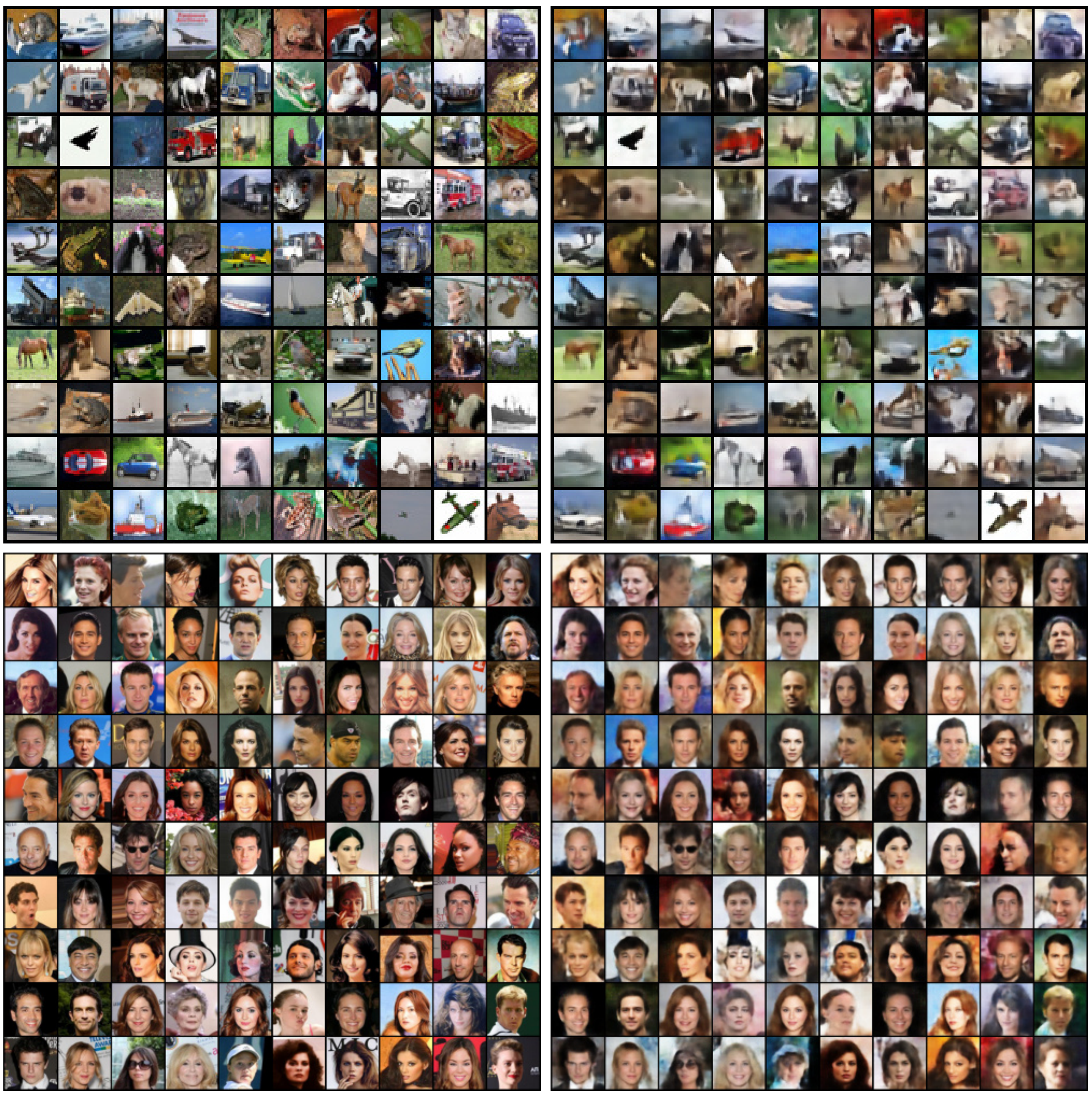

Figure 4.13: Test image reconstruction. Top: CIFAR-10. Bottom: CelebA. Left: test images. Right: reconstructed images.

For CIFAR-10, we use its own 10,000 test images while for CelebA, we use the hold-out 1,000 test images as stated above. The reconstruction quality is further measured by per-pixel mean square error (MSE). Table 4.2 shows the per-pixel MSE of our model compared to WS [HDF95], VAE [KW13], ALI [DBP16], ALICE [LLC17].

Note, we do not consider methods without inference models on training data, including variants of GANs and cooperative training, since it is infeasible to test such models using image reconstruction.

4.3.3 Energy Landscape Mapping

In the following, we evaluate the learned energy-based model by mapping the macroscopic structure of the energy landscape. When following a MLE regime by minimizing $\mathrm{KL}(q_{\text{data}} \| \pi_\alpha)$, we expect the energy-function $-f_\alpha(x)$ to encode $x \sim q_{\text{data}}(x)$ as local energy minima. Moreover, $-f_\alpha(x)$ should form minima for unseen images and macroscopic landscape structure in which basins of minima are distinctly separated by energy barriers. Hopfield observed that such landscape is a model of associative memory [Hop82].

In order to learn a well-formed energy-function, in Algorithm 2, we perform multiple θ-steps such that the samples $\{\tilde{x}_i\}_{i=1}^{\tilde{M}}$ are sufficiently "close" to the local minima of $-f_\alpha(x)$. This avoids the formation of energy minima not resembling the data. The variational approximation of entropy of the marginal generator distribution $H(p_\theta(x))$ preserves diversity in the samples avoiding mode-collapse.

To verify that (i) local minima of $-f_\alpha(x)$ resemble $\{x_i\}$ and (ii) minima are separated by significant energy barriers, we shall follow the approach used in [HNZ18]. When clustering with respect to energetic barriers, the landscape is partitioned into Hopfield basins of attraction whereby each point $\{x_i\}$ on the landscape $-f_\alpha(x)$ is mapped onto a local minimum $\{\hat{x_i}\}$ by a steepest-descent path $x_i^{t+1} = x_i^t + \eta \nabla f_\alpha(x_i^t)$. The similarity measure used for hierarchical clustering is the barrier energy that separates any two regions. Given a pair of local minima $\{\hat{x}_i, \hat{x}_j\}$, we estimate the barrier $b_{i,j} = \max\{-f_\alpha(x_k) : x_k \in \hat{x}_i \xrightarrow{\gamma} \hat{x}_j\}$ as the highest energy along a linear interpolation $x \xrightarrow{\gamma} y = \{x + \gamma(y - x) : \gamma \subseteq [0, 1]\}$. If $b_{i,j} < \epsilon$ for some energy threshold ϵ, then $\{x_i, x_j\}$ belong to the same basin. The clustering is repeated recursively until all minima are clustered together. Such graphs have come to be referred as disconnectivity graphs (DG) [WMW98].

We conduct energy landscape mapping experiments on the MNIST [LC10] and Fashion-MNIST [XRV17] datasets, each containing $70,000$ grayscale images of size 28×28 pixels depicting handwritten digits and fashion products from 10 categories, respectively. We use 4

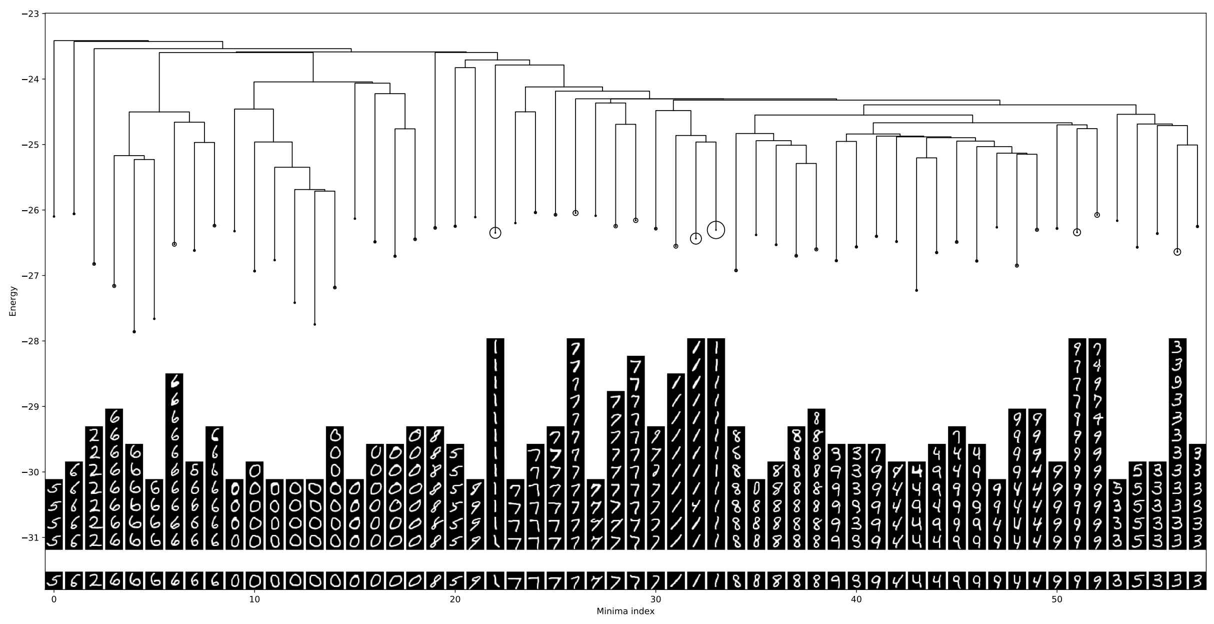

Figure 4.14: Illustration of the disconnectivity-graph depicting the basin structure of the learned energy-function $f_\alpha(x)$ for the MNIST dataset. Each column represents the set of at most 12 basins members ordered by energy where circles indicate the total number of basin members. Vertical lines encode minima depth in terms of energy and horizontal lines depict the lowest known barrier at which two basins merge in the landscape. Basins with less than 4 members were omitted for clarity.

layer generator model that has 100-dim latent factor. The kernel size is 3 for first layer and 4 for the following layers, and the number of channels are 1024, 512, 256, 1 respectively. The inference and energy-based model has the similar "mirror" structure as generator model.

The energy landscape mapping is not without limitations, because it is practically impossible to locate all local modes. Based on the local modes located by our algorithm, see Figure 4.14 for the MNIST dataset, it suggests that the learned energy function is well-formed which not only encodes meaningful images as minima, but also forms meaningful macroscopic structure. Moreover, within basins the local minima have a high degree of purity (i.e. digits within a basin belong to the same class), and, the energy barrier between basins seem informative (i.e. basins of ones and sixes form pure super-basins). Figure 4.15 depicts the energy landscape mapping on Fashion-MNIST.

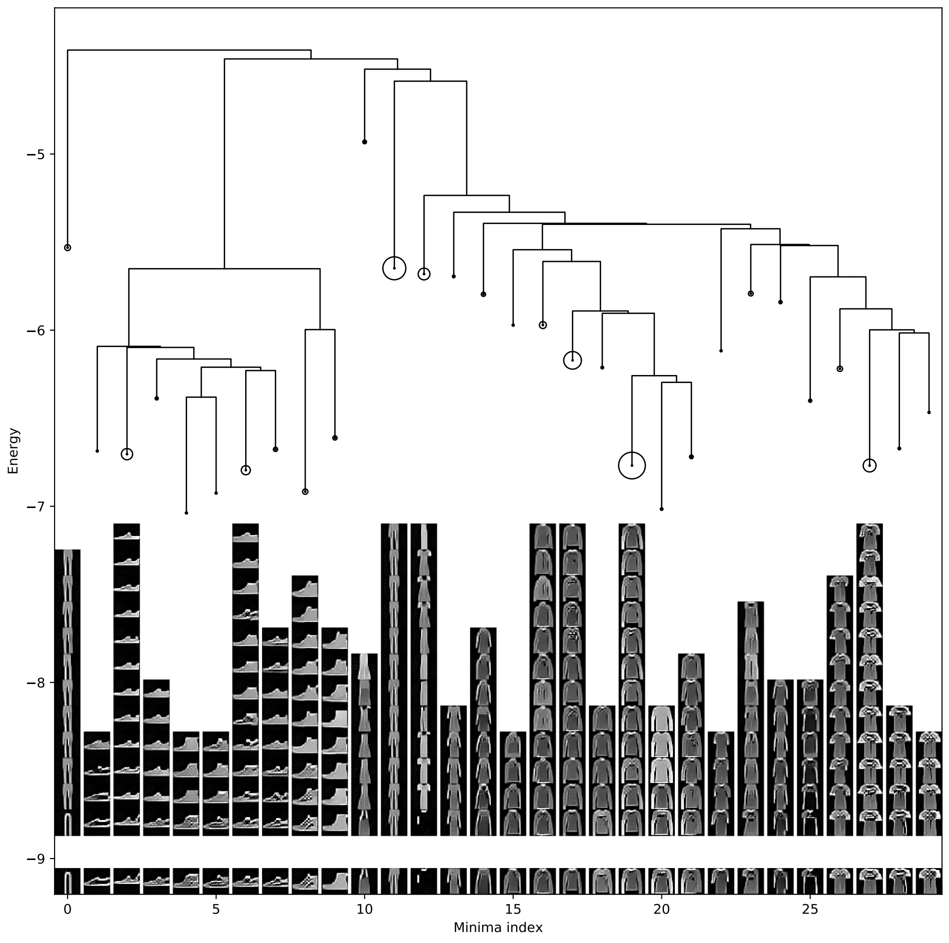

Figure 4.15: Illustration of the disconnectivity-graph depicting the basin structure of the learned energy-function for the Fashion-MNIST dataset. Each column represents the set of at most 12 basins members ordered by energy where circles indicate the total number of basin members. Vertical lines encode minima depth in terms of energy and horizontal lines depict the lowest known barrier at which two basins merge in the landscape. Basins with less than 4 members were omitted for clarity.

Potential applications include unsupervised classification in which energy barriers act as a geodesic similarity measure which captures perceptual distance (as opposed to e.g. ℓ_2 distance), weakly-supervised classification with one label per basins, or, reconstruction of incomplete data (i.e. Hopfield content-addressable memory or image inpainting).

4.3.4 Learning from Incomplete Images

The divergence triangle can also be used to learn from occluded images. This task is challenging as described in chapter 2, because only parts of the images are observed, thus the model needs to learn sufficient information to recover the occluded parts. Notably, chapter 2 of this dissertation proposed to recover incomplete images using alternating back-propagation (ABP) which has a MCMC based inference step to refine the latent factors and perform reconstruction iteratively. VAEs [RMW14b, KW13] build the inference model on occluded images, and can also be adapted for this task. It proceeds by filling the missing parts with average pixel intensity in the beginning, then iteratively re-update the missing parts using reconstructed values. Unlike VAEs, which only consider the un-occluded parts of training data, the proposed model utilizes the generated samples which become gradually recovered during training, resulting in improved recovery accuracy and sharp generation. Note that learning from incomplete data can be difficult for variants of GANs [GPM14, DAB17, RMC15, ACB17] and cooperative training [XLG18], since inference cannot be performed directly on the occluded images.

We evaluate our model on 10,000 images randomly chosen from CelebA dataset. Then, selected images are further center cropped as in chapter 2. Similar to VAEs, we zero-fill the occluded parts in the beginning, then iterative update missing values using reconstructed images obtained from the generator model. Three types of occlusions are used: (1) salt and pepper noise which randomly covers 50% (P0.5) and 70% (P0.7) of the image. (2) Multiple block occlusion which has 10 random blocks of size 10×10 (MB10). (3) Singe block occlusion where we randomly place a large 20×20 and 30×30 block on each image, denoted by B20 and B30 respectively. Table 4.3 shows the recovery errors using VAE [KW13], ABP and

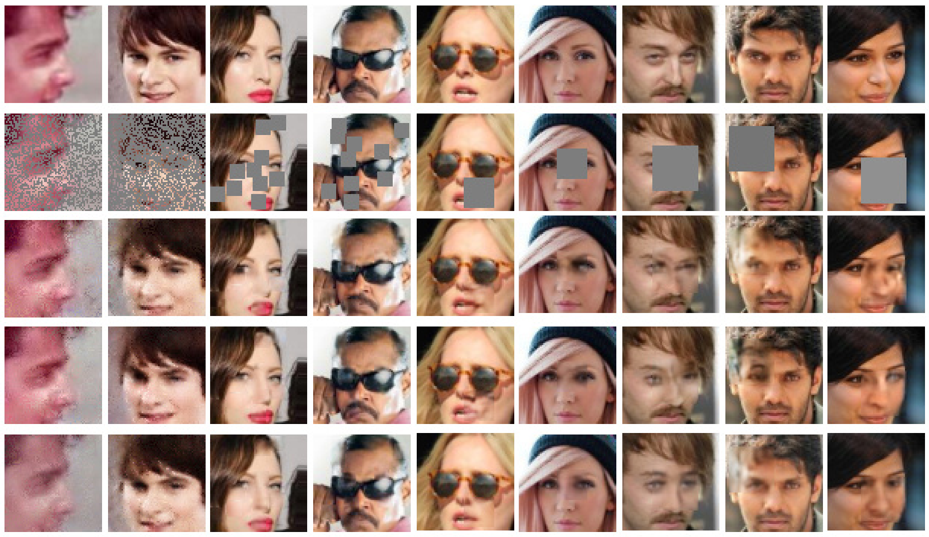

Figure 4.16: Learning from incomplete data from the CelebA dataset. The 9 columns belong to experiments P0.5, P0.7, MB10, MB10, B20, B20, B30, B30, B30 respectively. Row 1: original images, not observed in learning stage. Row 2: training images. Row 3: recovered images using VAE [KW13]. Row 4: recovered images using ABP. Row 5: recovered images using our method.

the triangle model where the error is defined as per-pixel absolute difference (relative to the range of pixel values) between the recovered image on the occluded pixels and the ground truth image.

EXP	P0.5	P0.7	MB10	B20	B30
VAE [KW13]	0.0446	0.0498	0.1169	0.0666	0.0800
ABP	**0.0379**	**0.0428**	0.1070	0.0633	0.0757
Ours	0.0380	0.0430	**0.1060**	**0.0621**	**0.0733**

Table 4.3: Recovery errors for different occlusion masks for 10,000 images from CelebA.

It can be seen that our model consistently out-performs the VAE model for different occlusion patterns. For structured occlusions (i.e., multiple and single blocks), the un-occluded parts contain more meaningful configurations that will improve learning of the generator

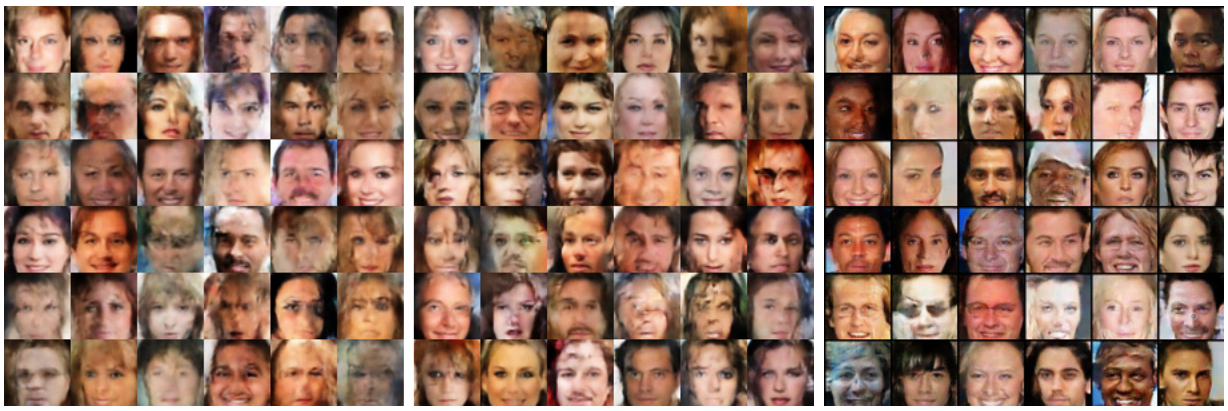

Figure 4.17: Image generation from different models learned from training images of the CelebA dataset with 30×30 occlusions. Left: images generated from VAE model [KW13]. Middle: images generated from ABP model. Right: images generated from our proposed model.

through the energy-based model, which will, in turn, generate more meaningful samples to refine our inference model. This could be verified by the superior results compared to ABP. While for unstructured occlusions (i.e., salt and pepper noise), ABP achieves improved recovery, a possible reason being that un-occluded parts contain less meaningful patterns which offer limited help for learning the generator and inference model. Our model synthesizes sharper and more realistic images from the generator on occluded images. See Figure 4.17 in which images are occluded with 30×30 random blocks.

4.4 Summary

In this chapter, we propose an unified probabilistic framework, called divergence triangle, to jointly and tractably learn the generator model and the energy-based model. The divergence triangle forms the compact learning functional that integrates the maximum likelihood learning, variational learning and adversarial learning. Our experiments show that we could learn the well-formed energy-based model, the realistic generator model as well as faithful inference model. The model can be effective in learning from incomplete data as well.

CHAPTER 5

Understanding Generator Model

In the previous chapters, we discussed probabilistic deep generative models, namely the generator model and the energy-based model, and their unsupervised learning methods. Though good performance can be obtained, the understanding of the learned models remains largely unexplored, especially for deep generative models. The existing models with deep neural network parameterizations are in general difficult to interpret because of their high non-linearity and entangled nature. In this chapter, we venture to study the behavior of the learned deep generative models.

5.1 Introduction and Motivation

Recently, a paper published in Cell [CT17] reports an interesting discovery about the neurons in the middle lateral (ML)/middle fundus (MF) and anterior medial (AM) areas of the primate brain that are responsible for face recognition. Specifically, the paper is concerned with how these neurons respond to and encode the face stimuli generated by a pre-trained Active Appearance Model (AAM) [CET01, CRB15]. In AAM, there are explicit shape variables and appearance variables that generate the positions of the control points and the nominal face image respectively, and the output image is then generated by wrapping the nominal face image using the control points. [CT17] discovers that the responses of the aforementioned neurons to the face image generated by the AAM exhibits strong linear relationship with the shape and appearance variables of the AAM that generates the face image. In fact, the shape and appearance variables of the AAM can be recovered from the neuron responses so that the face image can be reconstructed by the AAM using the recovered shape and appearance variables.

In this chapter, we investigate whether the above phenomenon can be replicated by deep generative models. In particular, we focus on a popular generator model [GPM14], which can be considered a non-linear generalization of the factor analysis model. In generator model, the mapping from the latent variables to the observed signal is modeled by a convolutional neural network (ConvNet), which has proven to be an exceedingly powerful approximator of high-dimensional non-linear mappings.

Both the AAM and the generator model are latent variable models where the signal is obtained by transforming the latent variables. In the AAM, the latent variables consist of explicit shape variables and appearance variables, which generate the control points and the appearance image by linear mappings learned by principal component analysis (PCA). The output image is generated by a highly non-linear but known warping function of the control points and the nominal image. In contrast, the generator model is more generic, in that it does not assume any prior knowledge about shape and deformation, and it does not have any explicit shape variables and shape model. We are interested in whether the generator model can replicate the AAM in the sense that whether the generator model can learn from the images generated by a pre-trained AAM, so that the latent variables of the learned generator model are closely related to the latent shape and appearance variables of the AAM, and the non-linear mapping from the latent variables to the output image in the generator network accounts for the highly non-linear warping function of the AAM. As it is impossible for the latent variables of the learned generator to be the same as the latent variables of the AAM, a strong linear relationship between the two sets of latent variables (or codes) is the best we can hope for. We shall show that such a linear relationship indeed exists, thus qualitatively reproducing the behavior of the neuron responses (or neural code) observed by [CT17].

The generator model can be trained by various methods, including the wake-sleep algorithm [HDF95], variational autoencoder (VAE) [KW13, RMW14a, SKW15], generative adversarial networks (GAN) [GPM14, RMC15, DCF15], moment matching networks [LSZ15] as well as alternating back-propagation (ABP) and divergence triangle which are described in the previous chapters. They have led to impressive results in a wide range of applications, such as image/video synthesis [DSB15], disentangled feature learning [CDH16, HMP16] and pattern completion etc.

In this dissertation, we shall adopt the VAE method to train the generator model. Unlike GAN, the VAE complements the generator model with an inference model that transforms the observed image to the latent variables. The inference network seeks to approximate the posterior distribution of the latent variables given the observed image. The inference model and the generator model form an auto-encoder, where the inference model plays the role of the encoder that encodes the signal into the latent variables (or latent code), and the generator model plays the role of the decoder that decode the latent variables (or latent code) back to the signal. The parameters of the two networks can be learned by maximizing a variational lower bound of the log-likelihood [BKM17] (see chapter 4). We show that the latent variables computed by the inference network from the observed face image are highly correlated with the latent variables of the AAM that generates the face image. This work is phenomenological in nature. It is our hope that the study is of interest to both the neuroscience community and the deep learning community.

5.2 Active Appearance Model (AAM)

The active appearance model [CET01, CRB15] is a generative model for representing face images. It has a shape model and an appearance model. Both models are learned by principal component analysis (PCA).

Shape model: The shape model is based on a set of landmarks or control points. In the training stage, the control points are given for each training image. Let y denote the coordinates of all the control points. The shape model is

$$y = \bar{y} + P_s b_s, \tag{5.1}$$

where $\bar{y}$ is the average shape, P_s is the matrix of eigenvectors, and b_s is the vector of shape variables. $(\bar{y}, P_s)$ are shared across all the training examples, while y and b_s are different for different examples. The model can be learned from the given control points of the training images by PCA, where the number of eigenvectors is determined empirically.

Appearance model: The appearance model generates the nominal image before shape deformation. To learn the model, we can wrap each training image to the shape-normalized image so that its control points match those of the mean shape $\bar{y}$. Then a PCA is performed on the shape-normalized training images. Let g denote the vector of the grey-level image. The appearance model is

$$g = \bar{g} + P_a b_a, \tag{5.2}$$

where $\bar{g}$ is the mean normalized grey-level image, P_a is the matrix of eigenvectors, and b_a is the vector of appearance variables. $(\bar{g}, P_a)$ are shared by all the training examples, while g and b_a are different for different examples. For colored images with RGB channels, we then concatenate three channels into a single vector, then perform the PCA as in grey-level images.

We can learn (P_s, P_a) from the training images with given control points. We concatenate the shape and appearance variables to form the face representation or the latent code, i.e., $Z_{AAM} = [b_s, b_a]$. Given Z_{AAM}, we can generate face image X by generating y and g first, and then warping g according to y using a warping function to output the image $X = h(g, y)$. The warping function h is given and is highly non-linear in g and y.

5.3 Generator Model and Variational Learning

We have discussed the generator model and the variational learning in chapter 2 and chapter 4. We shall briefly review them below.

The generator model is a deep generative model of the following form:

$$Z \sim \mathrm{N}(0, I_d), \tag{5.3}$$

$$X = g_\theta(Z) + \epsilon_i, \tag{5.4}$$

where Z is the vector of latent variables (or latent code) and generates the output image X by a non-linear mapping g_θ, which is modeled by a top-down convolutional neural network

(ConvNet), where θ collects all the weight and bias parameters of the top-down ConvNet. ϵ is the noise vector whose elements are independent $\mathrm{N}(0, \sigma^2)$ random variables. Even though Z follows a simple distribution, the model can generate X with very complex distribution and with very rich patterns because of the expressiveness of g_θ.

Compared to the AAM, the generator network has no explicit shape model such as (5.1) with control points y and shape variables b_s, nor does it have the explicit non-linear warping function $X = h(g, y)$. The generator network relies on the highly expressive ConvNet g_θ to account for the linear shape model and the non-linear warping function. Even though no prior knowledge of shape and warping is built into the generator network, it can learn such knowledge by itself.

Specifically, we shall use a pre-trained AAM as a teacher model, and we let the generator network be the student model. The AAM generates training images, and the generator network learns from the training images. We shall show that the inferred Z from the face image X has a strong linear relationship with the corresponding Z_{AAM} that the AAM uses to generate X.

Given a set of N training images $\{X_i, i = 1, ..., N\}$ generated by AAM, we train the generator model by variational method, i.e., variational auto-encoder (VAE) [KW13, RMW14a, SKW15]. The basic idea of VAE is to approximate the posterior distribution $p_\theta(Z|X)$ by a tractable inference model $q_\phi(Z|X)$ with a separate set of parameters ϕ, such as a Gaussian distribution with independent components $\mathrm{N}(\mu_\phi(X), \sigma_\phi^2(X))$, where $\mu_\phi(X)$ is the vector of means of the components of Z, and $\sigma_\phi^2(X)$ is the vector of variances of the components of Z. Both $\mu_\phi(X)$ and $\sigma_\phi(X)$ can be modeled by bottom-up ConvNets.

The parameters (θ, ϕ) can be learned by jointly maximizing the variational lower bound of the log-likelihood which is computationally tractable as long as the inference model $q_\phi(Z|X)$ is tractable. See [KW13] for more details. $q_\phi(Z|X)$ is the encoder, and $p_\theta(X|Z)$ is the decoder. After learning (θ, ϕ), we can estimate Z from X by the learned posterior mean vector $Z_G = \mu_\phi(X)$. In our work, we use Z_G as the code of X.

The VAE can be viewed as minimization of two joint distributions, namely $Q(Z, X)(=$

$q_{\text{data}}(X)q_\phi(Z|X))$ and $P(Z,X)(=p(Z)p_\theta(X|Z))$ due to the following (see Eqn. 4.14 in chapter 4):

$$\mathrm{KL}(Q\|P) = \mathrm{KL}(q_{\text{data}}(X)\|p_\theta(X)) + \mathrm{KL}(q_\phi(Z|X)\|p_\theta(Z|X)) \tag{5.5}$$

Such joint minimization, i.e., $\min_Q \min_P \mathrm{KL}(Q\|P)$, can be accomplished by alternating projection. Specifically, $\min_Q \mathrm{KL}(Q\|P)$ is a variational projection that minimizes over the first argument, while $\min_P \mathrm{KL}(Q\|P)$ is a model fitting projection that minimizes over the second argument. As is commonly known, the former has mode seeking behavior while the latter has moment matching behavior. Adversarial based methods, e.g., GANs[GPM14], can also be used for training. However, they do not have an inference model or an encoder, which is crucial for our work.

5.4 Experiments

We conduct experiments to investigate whether the generator model can replicate or imitate the AAM, where the AAM serves as the teacher model and the generator model plays the role of the student model. In the learning stage, the generator model only has access to the images generated by the AAM. It does not have access to the shape and appearance variables (latent code) used by the AAM to generate the images. After learning the generator model, we investigate the relationship between the latent code of the learned generator model and the latent code of the AAM.

5.4.1 Experiment Setting

Data Generation. We consider models on both grey-scaled and colored face images. For grey-scaled images, we pre-train the AAM using approximately 200 frontal face images with given landmarks or control points. Coordinates of the landmarks are first averaged, then PCA is performed where the first 10 principal components (PCs) for shape (see Eqn 5.1) are retained. The landmarks of each training image are then smoothly morphed into the average shape, so that the resulting image only carries shape-free appearance information.

Another PCA is then performed on the shape-normalized training images, where the first 10 PCs for appearance (see Eqn 5.2) are retained. This results in a 20-dimensional latent face space. Similarly, for color images, we pre-train the AAM on 800 frontal images with given landmarks. We retain 50 PCs for the shape and another 50 PCs for the appearance which results in a 100-dimensional latent space. Every face has a corresponding AAM code denoted as Z_{AAM}, which encodes its shape and appearance variables. Z_{AAM} has 20 dimension for grey-scaled images and 100 dimension for colored ones.

To generate face stimuli for our experiments, we randomly generate $20,000$ grey-scaled face images of size 256×256 and $10,000$ colored ones of size 128×128 from the above pre-trained AAMs. Specifically, for each dimension of the latent code, we record the standard deviation of the training responses of that dimension, and sample the variable from the Gaussian distribution with the same standard deviation as the real training faces. After that, these sampled variables are combined with the learned eigenvectors P_s and P_a to generate the synthesized images. The obtained images are then used as our training data for the generator network. Figure 5.1 shows some examples of training images to pre-train the AAM, and the synthesized face images generated by the trained AAM.

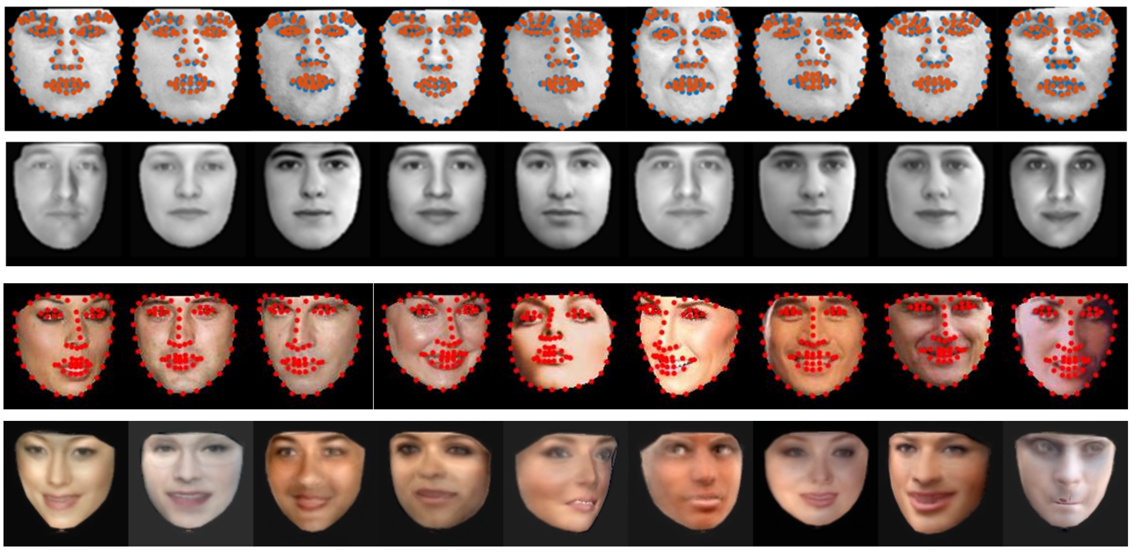

Figure 5.1: Top and third row: training images with landmarks labeled for AAM. Second and bottom row: synthesized AAM images for training the generator network.

VAE Training. The training images obtained above are scaled so that the intensities are within the range $[-1, 1]$. No further pre-processing steps are needed.

For the generator model, we adopt the structure similar to [RMC15, DSB15]. The network consists of multiple deconvolution (a.k.a convolution-transpose) layers interleaved with ReLU non-linearity and batch normalization [IS15]. For grey-scaled images, we learn a 7 layer top-down convNet. The first deconvolutional layer has 512 filters with kernel size 4×4 and stride 1. There are $512, 384, 256, 128, 64, 1$ filters with kernel size 4×4 and stride 2 for the following deconvolution layers respectively. For colored images, we learn a 6 layer top-down convNet where the first layer has 512 filters with kernel size 4×4 of stride 1, and the rest layers have $512, 256, 128, 64, 3$ filters which have the same kernel size as the first layer but with stride 2. Each deconvolution layer is followed by ReLU non-linearity and batch normalization except the last deconvolution layer which is instead followed by the tanh non-linearity.

For the inference model or the encoder network of VAE, we utilize the mirror structure of the generator network (which is the decoder network) where we use convolutional layers instead of deconvolutional ones. Besides, we use the ReLU with leaky factor 0.2 as our non-linearity. The mean and variance networks of the inference model share the same network structure except the top fully-connected layer. We also adopt the batch normalization in the inference model as in the generator model.

We tried different dimensionalities for the latent code Z, including 20, 100 and 200 dimensions for grey-scaled images as well as 100, 200 dimensions for colored ones. We used Adam optimizer [KB14] with initial learning rate 0.0002 for 500 iterations. The outputs of the mean network of the inference model are used as the learned latent code and are denoted as Z_G. Realistic synthesized images can be generated by the trained generator network. See Figure 5.2 for some examples. We design four experiments to examine the relationship between the AAM code Z_{AAM} for generating the face images and the code learned by the generator network, Z_G.

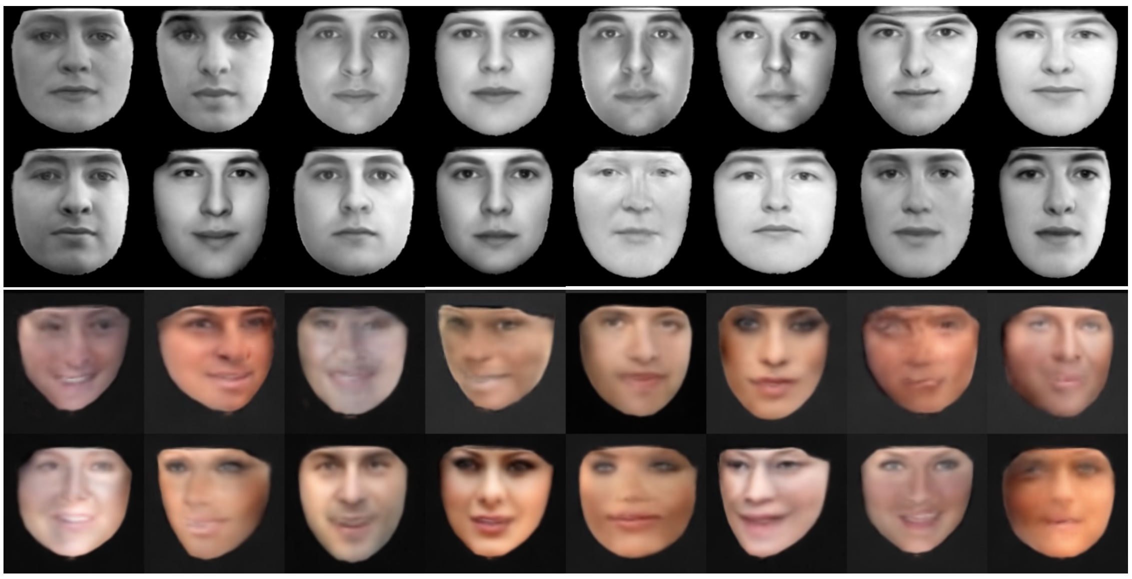

Figure 5.2: Synthesized images generated by the trained generator network.

5.4.2 Linear Relationship

[CT17] discovered that if a neuron has ramp-shaped tuning to different facial features, then its neural response can be approximated by a linear combination of the facial features. That is, the neural code for face patches ML/MF and AM has linear relationship with the AAM code of the presented face stimuli. In our first experiment, we check the strength of linearity between the code learned by the generator model and the underlying AAM code.

The codes for AAM, i.e., Z_{AAM}, are used to predict the corresponding codes learned by generator network, i.e., Z_G, and vice versa. Specifically, we fit linear model A and linear model B respectively:

$$Z_G \approx AZ_{AAM}, \tag{5.6}$$

$$Z_{AAM} \approx BZ_G. \tag{5.7}$$

We also include interception terms in both models. The goodness of fit of the model is determined by the percentage of variance in data that is explained by the fitted linear model, i.e., the so-called R-square (R^2):

$$R^2 = 1 - \frac{\sum_i \|Z_i - \hat{Z}_i\|^2}{\sum_i \|Z_i - \bar{Z}\|^2}, \tag{5.8}$$

where Z_i is the given code for image i, $\hat{Z}_i$ denotes the fitted value, and $\bar{Z}$ is the average of the code. Higher R^2 indicates stronger linear strength.

The R^2 values for different dimensionalities of Z_G are shown in Table 5.1 and Table 5.2. It can be seen that models show strong linear relations on both grey-scaled and colored images. This is non-trivial and surprising, because when presented with only synthesized face stimuli, the VAE training of the highly non-linear generator network [MPC14] can automatically learn the code that is linearly related to the underlying AAM code that generates the given face stimuli. That is, the learned generator network shares similar behavior as the face patch systems ML/MF and AM in the primate brain.

dimension d for Z_G	d=20	d=100	d=200
R^2 (A)	0.9749	0.9332	0.9641
R^2 (B)	0.9728	0.9926	0.9951

Table 5.1: Strength of linearity (R^2) for models A and B on grey-scaled face images.

dimension d for Z_G	d=100	d=200
R^2 (A)	0.6578	0.6627
R^2 (B)	0.7604	0.7983

Table 5.2: Strength of linearity (R^2) for models A and B on colored face images.

5.4.3 Decoding

As argued in [CT17], we should be able to linearly decode the facial features from the neural responses if there is a linear relationship between them. If so, we can accurately predict what the primate brain sees by knowing only the neural responses of the brain. Knowing that our learned code of the generator network Z_G shows strong linear relationship with the facial features Z_{AAM} from the above experiment, we expect that our automatically learned code

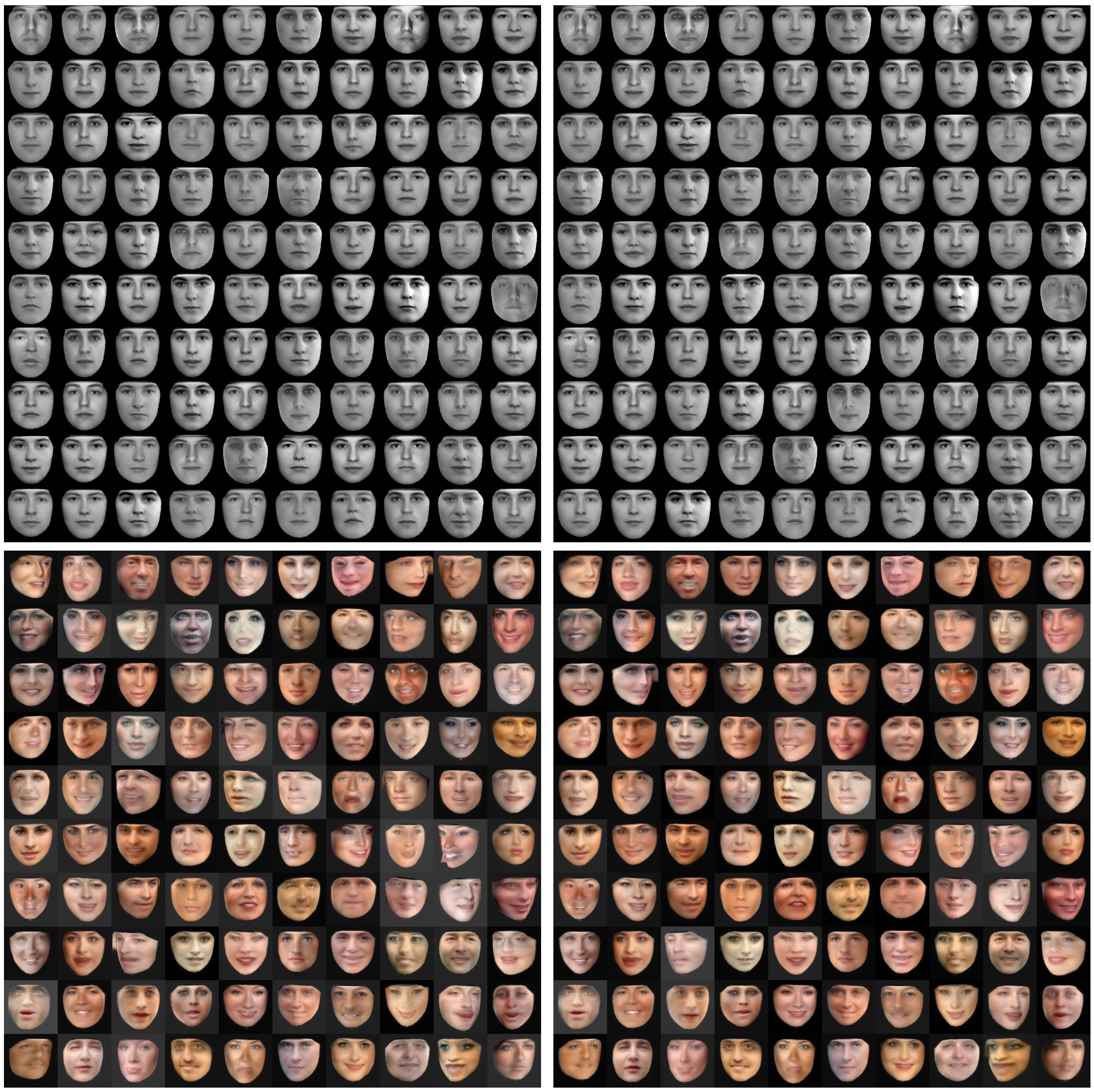

Figure 5.3: Left: test faces. Right: reconstructed faces using linear decoding. Top: results for grey-scaled face images. Bottom: results for colored images.

Z_G can accurately predict the facial features Z_{AAM}, which can then be used to reconstruct the input face image via the AAM. Therefore we further examine the decoding quality in this section.

To proceed, for training, we use Z_{AAM} and Z_G obtained during the learning process to fit model B as described above. Denote the estimated coefficients as $B^\star$. To test the decoding quality, we carry out the following two steps: (1) randomly sample a new set of AAM generated face images, which are used as the testing set. Then use the trained encoder network, i.e., mean network, of VAE to get point estimate of latent code of the generator network, i.e., Z_G^{test}. (2) Use the optimal $B^\star$ to get the predicted AAM code:

$$\hat{Z}_{AAM}^{test} = B^\star Z_G^{test}. \tag{5.9}$$

The predicted AAM code is then projected onto the previously learned AAM eigenvectors P_s and P_a to get the reconstructed image.

For model fitting for $B^\star$, we use the obtained $20,000$ Z_{AAM} during training for grey-scaled images and use $10,000$ Z_{AAM} for colored ones. $2,000$ newly generated testing images are used for both cases. Figure 5.3 shows some testing images and the reconstructed ones. It can be seen that the linear model between the learned code by VAE and the AAM code gives us high decoding quality. In this experiment as well as the subsequent experiments, we set the dimensionality of Z_G to be 100. Other dimensionalities give similar results.

5.4.4 Shape/Appearance Separation

The latent code Z_G learned by the deep generative model is mixed with shape and appearance information. It would be useful to separate the shape and appearance parts of Z_G. In this experiment, we further identify the strengths of shape and appearance parts of the learned code Z_G.

From the first two experiments, we show that Z_G is linearly related to the AAM code Z_{AAM}, which contains both the shape and appearance parts. We can identify these two parts by projecting each dimension of Z_G onto the shape code b_s and the appearance code b_a. We can then obtain the relative R^2 for each part. A higher R^2 for one part indicates the stronger response for this part. Recall that $Z_{AAM} = [b_s, b_a]$. We fit the linear models on the

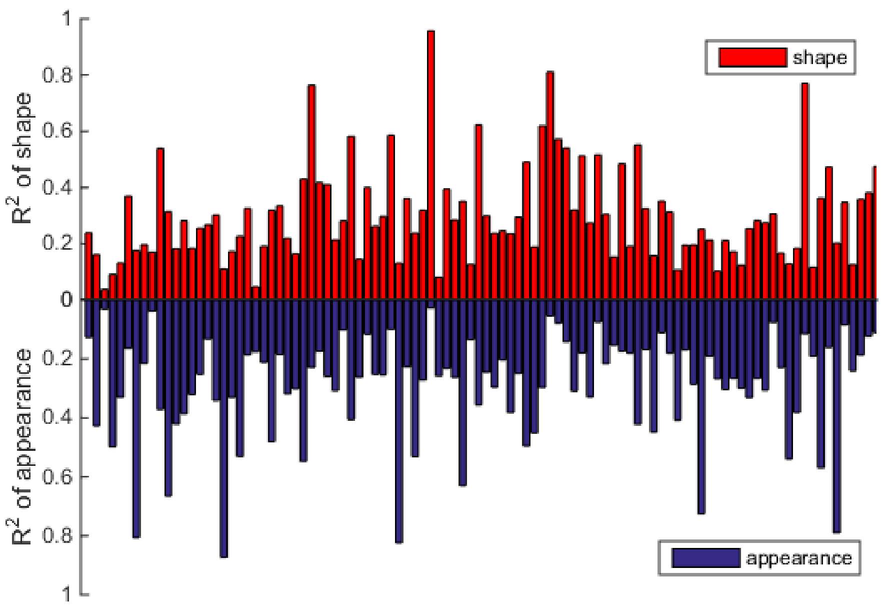

Figure 5.4: Bar plot for R^2 values of shape and appearance. Each vertical bar corresponds to a dimension of the latent code.

shape part b_s and the appearance part b_a respectively:

$$Z_G \approx A_s b_s, \tag{5.10}$$

$$Z_G \approx A_a b_a. \tag{5.11}$$

Figures 5.4 and 5.5 show the R^2 values for each dimension of Z_G. We only show plots for grey-scaled images here to illustrate the idea, note that the similar plots can be obtained for colored images as well. It shows that each dimension of Z_G responds differently to shape and appearance. To further verify and visualize our analysis, we first choose four dimensions with the top R^2 for shape and four dimensions with the top R^2 for appearance. Then we visualize the generated images using the trained generator network by varying (± 3 sd) the four chosen dimensions of the learned code while keeping the other three dimensions fixed. Figure 5.6 shows the result. It is clear that if we only vary the shape dimensions of the code (horizontally in the figure), the generated images mainly change their shapes while the appearances tend to remain similar. On the other hand, if we only vary the

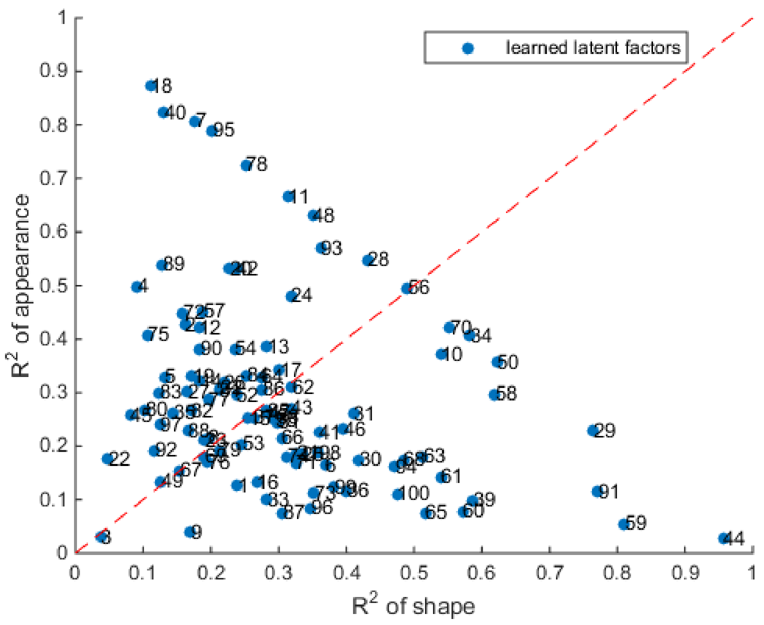

Figure 5.5: Scatter plot for R^2 values of shape and appearance. Each point corresponds to a dimension of the latent code. Red dashed line indicates the equal R^2 values for shape and appearance.

appearance dimensions of the code (vertically in the figure), the generated images mainly change their appearances instead of shapes. Similarly, for colored images, we also estimate the corresponding shape and appearance strength for each dimension of their latent factors Z_G. We select the dimensions which have relatively high R^2 value for appearance and low R^2 for shape, and vary such dimension by ± 3 sd while keeping other dimensions fixed. The first two rows of Figure 5.7 show that such dimension mainly controls the appearance variations. For dimensions that have high R^2 value for shape but low R^2 for appearance mainly controls the shape variations. The last two rows of Figure 5.7 show such examples.

5.4.5 Replicating AAM by Supervised Learning

So far the generator model learns from the AAM in the unsupervised manner, where the generator model only has access to the training images but not the latent code of the AAM. We now examine whether the generator model has enough expressive power to replicate the

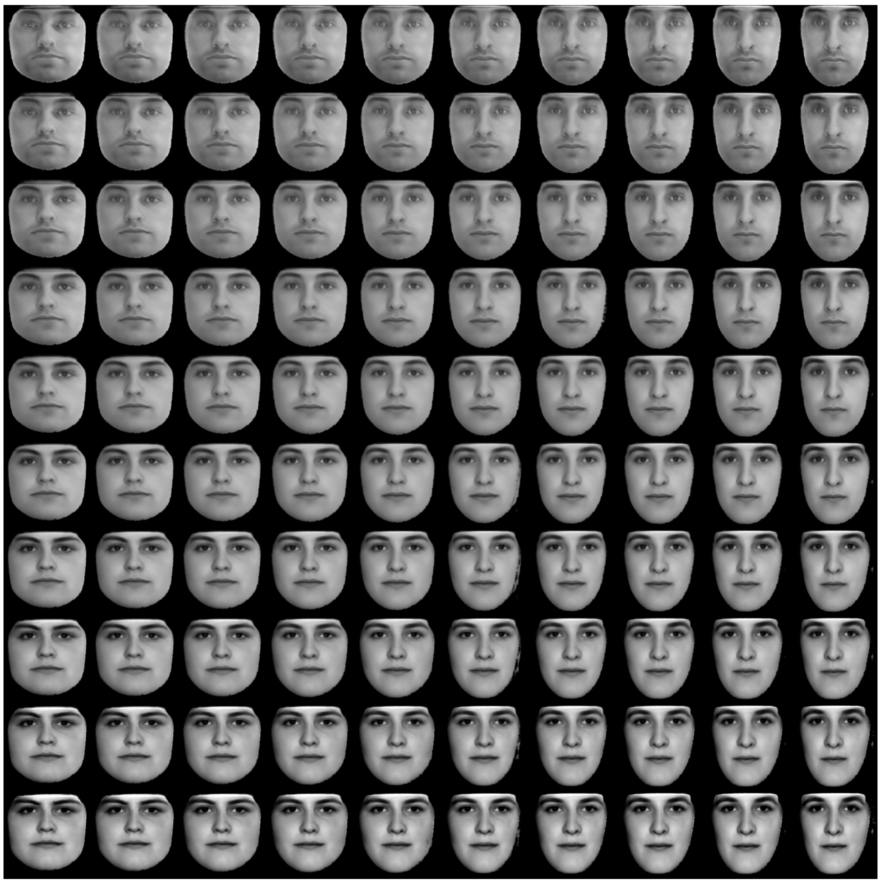
Figure 5.6: Vertical: appearance variation. Horizontal: shape variation.

AAM in the supervised setting where we also provide the latent code of the AAM to the generator.

In this experiment, the synthesized face images and their AAM codes are given, and we use these pairs to learn the generator model. To be more specific, we also consider two cases which include the grey-scaled images and more realistic colored images. we first train the generator network using the $20,000$ grey-scaled images (or $10,000$ colored images) and their codes. Let us denote the trained generator as $G^\star$. Then, we prepare a new set of $2,000$ synthesized images Y_{test} and their AAM code Z_{AAM}^{test} as our testing set. Z_{AAM}^{test} is then fed into $G^\star$ to get $\hat{X}_{test}$. If the generator network is capable of replicating the AAM, then $\hat{X}_{test}$ should be close to X_{test}, that is, the generated images by the trained generator model should be similar to the testing face images.

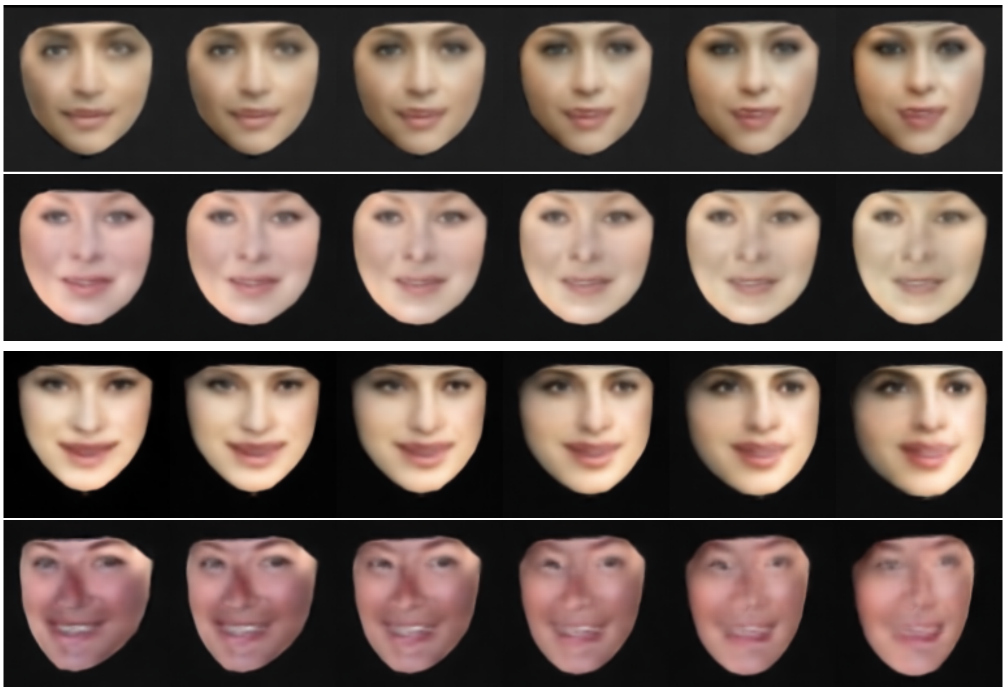

Figure 5.7: Top two rows: appearance variations (First: identity and expression changes. Second: color changes). Bottom two rows: shape variations. (First: the width of the face changes. Second: view points changes.)

We use the same generator model structure as in the VAE training. We use Adam optimizer to train the generator model for supervised learning. The learning rate is 0.0002 with 800 epochs. Figure 5.8 shows the ground-truth testing images generated by the AAM and the reconstructed images generated by the trained generator network for both grey-scaled and colored face images. We also calculate the per-pixel ℓ_1 reconstruction error, which is 0.0137 for grey-scaled images and 0.0840 for colored faces.

5.5 Summary

In this chapter, we design and conduct experiments to examine the relationship between the AAM code that generates the face stimuli and the automatically learned code by the generator model. Through the linearity analysis and the decoding quality analysis, we find that the biological observations made in [CT17] can be qualitatively reproduced by the

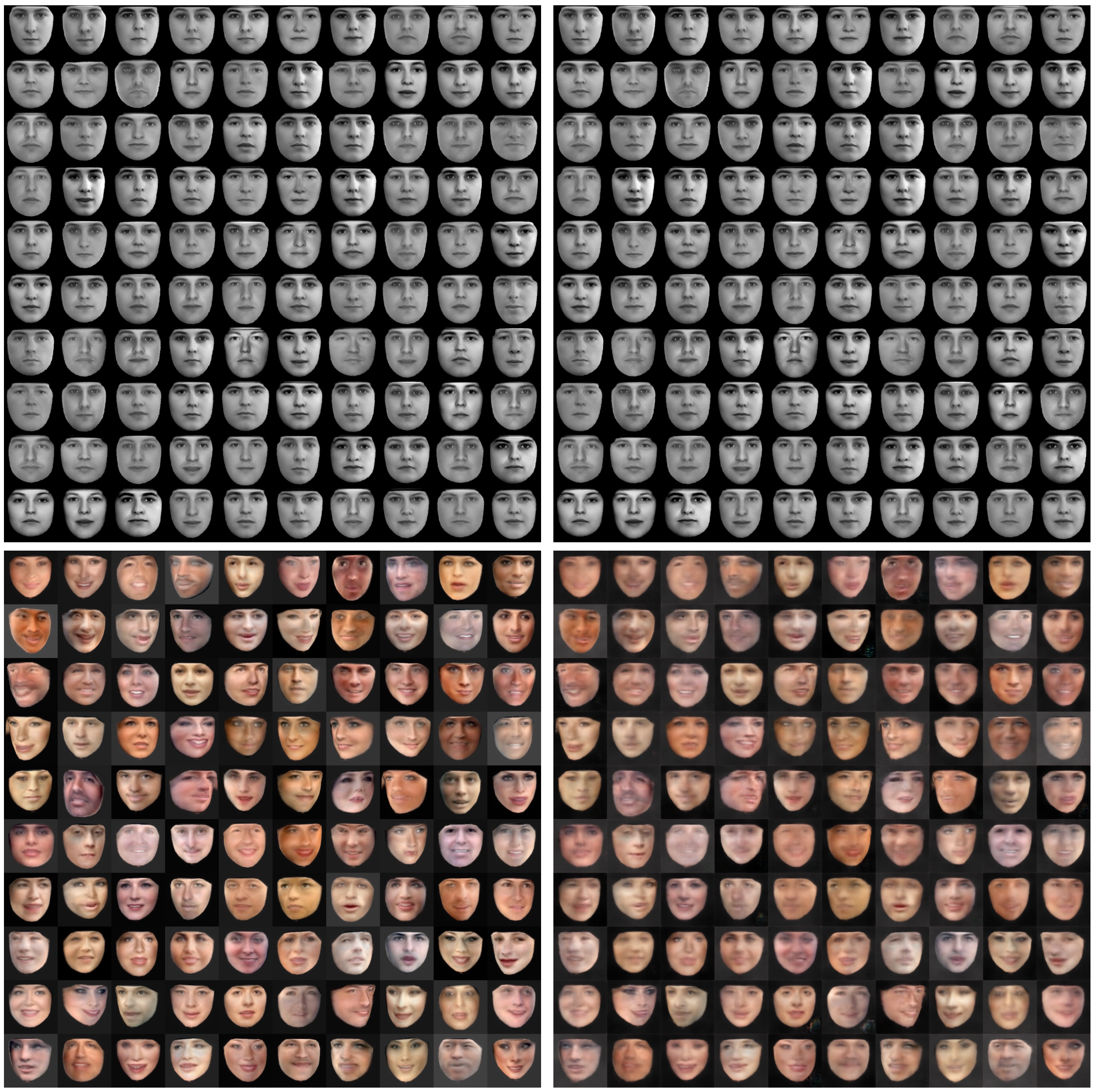
Figure 5.8: Left: test face images generated by AAM. Right: reconstructed face images by the generator network trained by supervised learning.

generator model, i.e., the learned code shows a strong linear relationship with the AAM code. Additionally, we use this relationship to further separate the shape and appearance parts of the learned code. Furthermore, we show that knowledge of AAM can be distilled into the generator model through supervised learning.

CHAPTER 6

Conclusion

In artificial intelligence and machine learning, probabilistic generative models play an important role towards knowledge representation and understanding especially for deep generative models that are parameterized by ConvNet. Such deep models have high approximation capacity, but the learning and understanding can be challenging. In this dissertation, we develop unsupervised learning algorithms for deep generative models and study their behaviors.

We first propose an alternating back-propagation (ABP) algorithm for training the generator model. We recognize that the generator model is a non-linear generalization of the factor analysis model, and develop the alternating back-propagation algorithm as the non-linear generalization of the alternating regression scheme of the Rubin-Thayer EM algorithm for fitting the factor analysis model. The alternating back-propagation algorithm iterates the inferential back-propagation for inferring the latent factors and the learning back-propagation for updating the parameters. Both back-propagation steps share most of their computing steps in the chain rule calculations.

Our learning algorithm is perhaps the most canonical algorithm for training the generator model. It is based on maximum likelihood, which is theoretically the most accurate estimator. The maximum likelihood learning seeks to explain and charge the whole dataset uniformly, so that there is little concern of under-fitting or biased fitting.

As an unsupervised learning algorithm, the alternating back-propagation algorithm is a natural generalization of the original back-propagation algorithm for supervised learning. It adds an inferential back-propagation step to the learning back-propagation step, with minimal overhead in coding and affordable overhead in computing. The inferential back-propagation seeks to perform accurate explaining-away inference of the latent factors. It can be worthwhile for tasks such as learning from incomplete or indirect data. The inferential back-propagation may also be used to evaluate the generators learned by other methods on tasks such as reconstructing or completing testing data. The proposed method can be effective for various data types including audio, image, video etc.

We then proposes to learn multi-view generator model through sharing the latent factors. We argue that the shared representation learned could effectively encode the common knowledge across different views, and the domain specific generator model can accurately obtain the domain related information. Therefore, the proposed method can naturally enforce the consensus and complementary principles of which a good shared representation should possess. We conduct qualitative experiment on face rotation and completion as well as audio-video common knowledge extraction, demonstrating that our method can be effectively utilized for generation and reconstruction tasks. We also conduct extensive quantitative comparisons with existing works in gait recognition, showing that our method, though trained unsupervisely, is competitive or even become state-of-the-art in many cases.

Besides the generator learning, in this dissertation, we also propose to jointly and tractably learn both the generator model and the energy-based model through the inference model. The proposed probabilistic framework, namely divergence triangle, forms the compact learning functional for three models and naturally unifies aspects of maximum likelihood estimation, variational learning, adversarial learning, contrastive divergence, and the wake-sleep algorithm. An extensive set of experiments demonstrated learning of a well-behaved energy-based model, realistic generator model as well as an accurate inference model. Moreover, experiments showed that the proposed divergence framework can also be effective in learning directly from incomplete data.

In terms of understanding of probabilistic models. We take the inspiration from the recent work in neuroscience [CT17] which shows that the face images can be reconstructed using the cell responses from face patches ML/MF and AM. To investigate whether the widely used generator model has the similar property, we design and conduct experiments to examine the relationship between the AAM code that generates the face stimuli and the automatically learned code by the generator model. Through the linearity analysis and the decoding quality analysis, we find that the biological observations made in [CT17] can be qualitatively reproduced by the generator model, i.e., the learned code shows a strong linear relationship with the AAM code. Additionally, we can also use this relationship to further separate the shape and appearance parts of the learned code. Again this is similar to the neural system as it is found that ML/MF and AM carry complementary information about shape and appearance. Furthermore, we show that the generator model is capable of replicating AAM and we demonstrate this through supervised learning. In other words, the knowledge of a pre-trained AAM can be distilled into the generator model.

Future Directions. The learning algorithm proposed or its variants can be applied to non-linear matrix factorization and completion which can be useful in recommender system. It can also be applied to problems where some components or aspects of the factors are supervised such as semi-supervised learning scenario.

We distill the knowledge of a pre-trained AAM to the generator network in this dissertation. It will also be interesting to distill the knowledge of a learned generator model to an AAM in order to interpret the generator model. Furthmore, we shall also extend the algorithm to learn interpretable generator and energy-based models with multiple layers of sparse or semantically meaningful latent variables or features.

REFERENCES

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks." In *International Conference on Machine Learning*, pp. 214–223, 2017.

[AN04] Pieter Abbeel and Andrew Y Ng. "Apprenticeship learning via inverse reinforcement learning." In *Proceedings of the twenty-first international conference on Machine learning*, p. 1, 2004.

[BAM17] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. "Multimodal Machine Learning: A Survey and Taxonomy." *arXiv preprint arXiv:1705.09406*, 2017.

[BCV13] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives." *IEEE transactions on pattern analysis and machine intelligence*, **35**(8):1798–1828, 2013.

[BKM17] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. "Variational inference: A review for statisticians." *Journal of the American Statistical Association*, (just-accepted), 2017.

[BLP07] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. "Greedy layer-wise training of deep networks." In *Advances in neural information processing systems*, pp. 153–160, 2007.

[CDH16] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets." In *Advances in Neural Information Processing Systems*, pp. 2172–2180, 2016.

[CDP18] Liqun Chen, Shuyang Dai, Yunchen Pu, Erjin Zhou, Chunyuan Li, Qinliang Su, Changyou Chen, and Lawrence Carin. "Symmetric variational autoencoder and connections to adversarial learning." In *International Conference on Artificial Intelligence and Statistics*, pp. 661–669, 2018.

[CET01] Timothy F Cootes, Gareth J Edwards, and Christopher J Taylor. "Active appearance models." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):681–685, 2001.

[CRB15] TF Cootes, MG Roberts, KO Babalola, and CJ Taylor. "Active shape and appearance models." In *Handbook of Biomedical Imaging*, pp. 105–122. Springer, 2015.

[CRT06] Emmanuel J Candès, Justin Romberg, and Terence Tao. "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information." *IEEE Transactions on information theory*, **52**(2):489–509, 2006.

[CT17] Le Chang and Doris Y Tsao. "The Code for Facial Identity in the Primate Brain." *Cell*, **169**(6):1013–1028, 2017.

[DAB17] Zihang Dai, Amjad Almahairi, Philip Bachman, Eduard Hovy, and Aaron Courville. "Calibrating energy-based generative adversarial networks." *arXiv preprint arXiv:1702.01691*, 2017.

[DBP16] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. "Adversarially learned inference." *arXiv preprint arXiv:1606.00704*, 2016.

[DCF15] Emily L Denton, Soumith Chintala, Rob Fergus, et al. "Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks." In *NIPS*, pp. 1486–1494, 2015.

[DCW03] G. Doretto, A. Chiuso, Y. Wu, and S. Soatto. "Dynamic textures." *IJCV*, **51**(2):91–109, 2003.

[DDS09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. IEEE, 2009.

[DKB14] Laurent Dinh, David Krueger, and Yoshua Bengio. "NICE: Non-linear independent components estimation." *arXiv preprint arXiv:1410.8516*, 2014.

[DKD16] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. "Adversarial feature learning." *arXiv preprint arXiv:1605.09782*, 2016.

[DLR77] Arthur P Dempster, Nan M Laird, and Donald B Rubin. "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the royal statistical society. Series B (methodological)*, pp. 1–38, 1977.

[DLW14] Jifeng Dai, Yang Lu, and Ying-Nian Wu. "Generative modeling of convolutional neural networks." *arXiv preprint arXiv:1412.6296*, 2014.

[DSB15] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. "Learning to generate chairs with convolutional neural networks." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1538–1546, 2015.

[DSB16] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using Real NVP." *arXiv preprint arXiv:1605.08803*, 2016.

[GBS07] Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. "Actions as Space-Time Shapes." *Transactions on Pattern Analysis and Machine Intelligence*, **29**(12):2247–2253, December 2007.

[GC11] Mark Girolami and Ben Calderhead. "Riemann manifold langevin and hamiltonian monte carlo methods." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **73**(2):123–214, 2011.

[GG84] Stuart Geman and Donald Geman. "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images." *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

[GM07] Ulf Grenander and Michael I Miller. *Pattern theory: from representation to inference.* Oxford University Press, 2007.

[GMC10] Ralph Gross, Iain Matthews, Jeffrey Cohn, Takeo Kanade, and Simon Baker. "Multi-PIE." *Image Vision Comput.*, **28**(5):807–813, May 2010.

[GPM14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets." In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[Gre70] Ulf Grenander. "A unified approach to pattern analysis." *Advances in Computers*, **10**:175–216, 1970.

[HDF95] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. "The" wake-sleep" algorithm for unsupervised neural networks." *Science*, **268**(5214):1158–1161, 1995.

[Hin02a] Geoffrey Hinton. "Training products of experts by minimizing contrastive divergence." *Neural Computation*, pp. 1771–1800, 2002.

[Hin02b] Geoffrey E Hinton. "Training products of experts by minimizing contrastive divergence." *Neural Computation*, **14**(8):1771–1800, 2002.

[Hin10] Geoffrey E. Hinton. "A practical guide to training restricted Boltzmann machines." *Tech. Rep. UTML TR 2010-003, Dept. Comp. Sci., Univ. Toronto*, 2010.

[HMP16] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. "beta-vae: Learning basic visual concepts with a constrained variational framework." 2016.

[HNZ18] Mitch Hill, Erik Nijkamp, and Song-Chun Zhu. "Building a Telescope to Look Into High-Dimensional Image Spaces." *arXiv preprint arXiv:1803.01043*, 2018.

[Hop82] John Hopfield. "Neural networks and physical systems with emergent collective computational abilities." *IProceedings of the National Academy of Sciences of the United States of America*, **79**(8):2554–2558, 1982.

[HOT06] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." *Neural computation*, **18**(7):1527–1554, 2006.

[HS06] Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks." *science*, **313**(5786):504–507, 2006.

[HSS04] David Hardoon, Sandor Szedmak, and John Shawe-Taylor. "Canonical correlation analysis: An overview with application to learning methods." *Neural computation*, **16**(12):2639–2664, 2004.

[HWZ13] Maodi Hu, Yunhong Wang, Zhaoxiang Zhang, James J Little, and Di Huang. "View-invariant discriminative projection for multi-view gait-based human identification." *IEEE Transactions on Information Forensics and Security*, **8**(12):2034–2045, 2013.

[HZL17] Rui Huang, Shu Zhang, Tianyu Li, and Ran He. "Beyond Face Rotation: Global and Local Perception GAN for Photorealistic and Identity Preserving Frontal View Synthesis." *arXiv preprint arXiv:1704.04086*, 2017.

[IS15] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift." *arXiv preprint arXiv:1502.03167*, 2015.

[JLT17] Long Jin, Justin Lazarow, and Zhuowen Tu. "Introspective Learning for Discriminative Classification." In *Advances in Neural Information Processing Systems*, 2017.

[KAL17] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. "Progressive growing of gans for improved quality, stability, and variation." *arXiv preprint arXiv:1710.10196*, 2017.

[KB14] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*, 2014.

[KB16] Taesup Kim and Yoshua Bengio. "Deep Directed Generative Models with Energy-Based Probability Estimation." *arXiv preprint arXiv:1606.03439*, 2016.

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." *Computer*, (8):30–37, 2009.

[KD18] Durk P Kingma and Prafulla Dhariwal. "Glow: Generative flow with invertible 1x1 convolutions." In *Advances in Neural Information Processing Systems*, pp. 10236–10245, 2018.

[KP08] Hyunsoo Kim and Haesun Park. "Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method." *SIAM Journal on Matrix Analysis and Applications*, **30**(2):713–730, 2008.

[KSC14] Meina Kan, Shiguang Shan, Hong Chang, and Xilin Chen. "Stacked progressive auto-encoders (spae) for face recognition across poses." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1883–1890, 2014.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks." In *NIPS*, pp. 1097–1105, 2012.

[KTP17] Jean Kossaifi, Georgios Tzimiropoulos, and Maja Pantic. "Fast and exact newton and bidirectional fitting of active appearance models." *IEEE transactions on image processing*, **26**(2):1040–1053, 2017.

[KW13] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes." *arXiv preprint arXiv:1312.6114*, 2013.

[KWL09] Worapan Kusakunniran, Qiang Wu, Hongdong Li, and Jian Zhang. "Multiple views gait recognition using view transformation model based on optimized gait energy image." In *IEEE 12th International Conference on Computer Vision Workshops*, pp. 1058–1064. IEEE, 2009.

[KWZ12] Worapan Kusakunniran, Qiang Wu, Jian Zhang, and Hongdong Li. "Gait recognition under various viewing angles based on correlated motion regression." *IEEE Transactions on Circuits and Systems for Video Technology*, **22**(6):966–980, 2012.

[LBB98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, **86**(11):2278–2324, 1998.

[LC10] Yann LeCun and Corinna Cortes. "MNIST handwritten digit database." 2010.

[LCH06] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and F Huang. "A tutorial on energy-based learning." *Predicting structured data*, **1**(0), 2006.

[LCS10] Bo Li, Hong Chang, Shiguang Shan, and Xilin Chen. "Low-resolution face recognition via coupled locality preserving mappings." *IEEE Signal processing letters*, **17**(1):20–23, 2010.

[LKM17] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. "Are gans created equal? a large-scale study." *arXiv preprint arXiv:1711.10337*, 2017.

[LLC17] Chunyuan Li, Hao Liu, Changyou Chen, Yuchen Pu, Liqun Chen, Ricardo Henao, and Lawrence Carin. "Alice: Towards understanding adversarial learning for joint distribution matching." In *Advances in Neural Information Processing Systems*, pp. 5495–5503, 2017.

[LLW15a] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Deep learning face attributes in the wild." In *ICCV*, pp. 3730–3738, 2015.

[LLW15b] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Deep Learning Face Attributes in the Wild." In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[LRW98] Chuanhai Liu, Donald B Rubin, and Ying Nian Wu. "Parameter expansion to accelerate EM: The PX-EM algorithm." *Biometrika*, **85**(4):755–770, 1998.

[LS01] Daniel D Lee and H Sebastian Seung. "Algorithms for non-negative matrix factorization." In *Advances in neural information processing systems*, pp. 556–562, 2001.

[LSZ15] Yujia Li, Kevin Swersky, and Rich Zemel. "Generative moment matching networks." In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1718–1727, 2015.

[LXW15] Zhuowen Lv, Xianglei Xing, Kejun Wang, and Donghai Guan. "Class energy image analysis for video sensor-based gait recognition: A review." *Sensors*, **15**(1):932–964, 2015.

[LYZ16] Yingming Li, Ming Yang, and Zhongfei Zhang. "Multi-view representation learning: A survey from shallow methods to deep methods." *arXiv preprint arXiv:1610.01206*, 2016.

[LZW16] Yang Lu, Song-Chun Zhu, and Ying Nian Wu. "Learning FRAME Models Using CNN Filters." In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[MD10] David Mumford and Agnès Desolneux. *Pattern theory: the stochastic analysis of real-world signals*. CRC Press, 2010.

[MG14] Andriy Mnih and Karol Gregor. "Neural Variational Inference and Learning in Belief Networks." In *International Conference on Machine Learning*, pp. 1791–1799, 2014.

[MHN13] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. "Rectifier nonlinearities improve neural network acoustic models." In *Proc. ICML*, volume 30, 2013.

[MPC14] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. "On the number of linear regions of deep neural networks." In *Advances in Neural Information Processing Systems*, pp. 2924–2932, 2014.

[MS11] Josh H McDermott and Eero P Simoncelli. "Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis." *Neuron*, **71**(5):926–940, 2011.

[MSM06] Yasushi Makihara, Ryusuke Sagawa, Yasuhiro Mukaigawa, Tomio Echigo, and Yasushi Yagi. "Gait recognition using a view transformation model in the frequency domain." In *Computer Vision-ECCV 2006*, pp. 151–163. Springer, 2006.

[MZZ16] Michael F Mathieu, Junbo Jake Zhao, Junbo Zhao, Aditya Ramesh, Pablo Sprechmann, and Yann LeCun. "Disentangling factors of variation in deep representation using adversarial training." In *Advances in Neural Information Processing Systems*, pp. 5040–5048, 2016.

[NCK11] Jiquan Ngiam, Zhenghao Chen, Pang W Koh, and Andrew Y Ng. "Learning Deep Energy Models." In *International Conference on Machine Learning*, pp. 1105–1112, 2011.

[Nea11] Radford M Neal. "MCMC using Hamiltonian dynamics." *Handbook of Markov Chain Monte Carlo*, **2**, 2011.

[NKK11] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. "Multimodal deep learning." In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 689–696, 2011.

[OF97] Bruno A Olshausen and David J Field. "Sparse coding with an overcomplete basis set: A strategy employed by V1?" *Vision Research*, **37**(23):3311–3325, 1997.

[OKK16] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel recurrent neural networks." *arXiv preprint arXiv:1601.06759*, 2016.

[PFH10] Renaud Péteri, Sándor Fazekas, and Mark J Huiskes. "DynTex: A comprehensive database of dynamic textures." *Pattern Recognition Letters*, **31**(12):1627–1632, 2010.

[PMB13] Razvan Pascanu, Guido Montufar, and Yoshua Bengio. "On the number of response regions of deep feed forward networks with piece-wise linear activations." *arXiv preprint arXiv:1312.6098*, 2013.

[RB05] Stefan Roth and Michael J Black. "Fields of experts: A framework for learning image priors." In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pp. 860–867. IEEE, 2005.

[RDS15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. "Imagenet large scale visual recognition challenge." *International Journal of Computer Vision*, **115**(3):211–252, 2015.

[RMC15] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks." *arXiv preprint arXiv:1511.06434*, 2015.

[RMW14a] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic Backpropagation and Approximate Inference in Deep Generative Models." In *International Conference on Machine Learning*, pp. 1278–1286, 2014.

[RMW14b] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic backpropagation and approximate inference in deep generative models." *arXiv preprint arXiv:1401.4082*, 2014.

[RS00] Sam T Roweis and Lawrence K Saul. "Nonlinear dimensionality reduction by locally linear embedding." *Science*, **290**(5500):2323–2326, 2000.

[RT82] Donald B Rubin and Dorothy T Thayer. "EM algorithms for ML factor analysis." *Psychometrika*, **47**(1):69–76, 1982.

[SC07] Tingkai Sun and Songcan Chen. "Locality preserving CCA with applications to data visualization and pose estimation." *Image and Vision Computing*, **25**(5):531–543, 2007.

[SGZ16] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. "Improved techniques for training gans." In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

[SH09] Ruslan Salakhutdinov and Geoffrey E Hinton. "Deep boltzmann machines." In *International Conference on Artificial Intelligence and Statistics*, 2009.

[SKW15] Tim Salimans, Diederik Kingma, and Max Welling. "Markov Chain Monte Carlo and Variational Inference: Bridging the Gap." In *Proceedings of The 32nd International Conference on Machine Learning*, pp. 1218–1226, 2015.

[SMF09] Leonid Sigal, Roland Memisevic, and David J Fleet. "Shared kernel information embedding for discriminative inference." In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 2852–2859. IEEE, 2009.

[SS12] Nitish Srivastava and Ruslan R Salakhutdinov. "Multimodal learning with deep boltzmann machines." In *Advances in neural information processing systems*, pp. 2222–2230, 2012.

[SSY14] XiaoBo Shen, QuanSen Sun, and YunHao Yuan. "A unified multiset canonical correlation analysis framework based on graph embedding for multiple feature extraction." *Neurocomputing*, **148**(19):397–408, 2014.

[Tie08] Tijmen Tieleman. "Training restricted boltzmann machines using approximations to the likelihood gradient." *ICML*, pp. 1064–1071, 2008.

[Tu07] Zhuowen Tu. "Learning generative models via discriminative approaches." In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2007.

[VLB08] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. "Extracting and composing robust features with denoising autoencoders." In *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103. ACM, 2008.

[VPT16] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. "Generating videos with scene dynamics." In *Advances In Neural Information Processing Systems*, pp. 613–621, 2016.

[WHW17] Zifeng Wu, Yongzhen Huang, Liang Wang, Xiaogang Wang, and Tieniu Tan. "A comprehensive study on cross-view gait based human identification with deep cnns." *IEEE transactions on pattern analysis and machine intelligence*, **39**(2):209–226, 2017.

[WMW98] David J Wales, Mark A Miller, and Tiffany R Walsh. “Archetypal energy landscapes.” *Nature*, **394**(6695):758, 1998.

[WOH03] Max Welling, Simon Osindero, and Geoffrey E Hinton. “Learning sparse topographic representations with products of student-t distributions.” In *Advances in neural information processing systems*, pp. 1383–1390, 2003.

[WXY14] Kejun Wang, Xianglei Xing, Tao Yan, and Zhuowen Lv. “Couple metric learning based on separable criteria with its application in cross-view gait recognition.” In *Chinese Conference on Biometric Recognition*, pp. 347–356. Springer, 2014.

[WZ03] Yizhou Wang and Song-chun Zhu. “Modeling textured motion: Particle, wave and sketch.” In *ICCV*, pp. 213–220, 2003.

[WZL00] Ying Nian Wu, Song Chun Zhu, and Xiuwen Liu. “Equivalence of Julesz ensembles and FRAME models.” *International Journal of Computer Vision*, **38**(3):247–265, 2000.

[XLG18] Jianwen Xie, Yang Lu, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. “Cooperative Training of Descriptor and Generator Networks.” *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 2018.

[XLZ16] Jianwen Xie, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. “A theory of generative ConvNet.” In *International Conference on Machine Learning*, pp. 2635–2644, 2016.

[XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.” *arXiv preprint arXiv:1708.07747*, 2017.

[XTX13] Chang Xu, Dacheng Tao, and Chao Xu. “A survey on multi-view learning.” *arXiv preprint arXiv:1304.5634*, 2013.

[XW16] Xianglei Xing and Kejun Wang. “Couple manifold discriminant analysis with bipartite graph embedding for low-resolution face recognition.” *Signal Processing*, **125**:329–335, 2016.

[XWC15] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. “Empirical Evaluation of Rectified Activations in Convolutional Network.” *CoRR*, **abs/1505.00853**, 2015.

[XWL15] Xianglei Xing, Kejun Wang, and Zhuowen Lv. “Fusion of gait and facial features using coupled projections for people identification at a distance.” *IEEE Signal Processing Letters*, **22**(12):2349–2353, 2015.

[XWY16] Xianglei Xing, Kejun Wang, Tao Yan, and Zhuowen Lv. “Complete canonical correlation analysis with application to multi-view gait recognition.” *Pattern Recognition*, **50**:107–117, 2016.

[YCR17] Shiqi Yu, Haifeng Chen, Edel B Garcıa Reyes, and P Norman. “Gaitgan: invariant gait feature extraction using generative adversarial networks.” In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 30–37, 2017.

[YCW17] Shiqi Yu, Haifeng Chen, Qing Wang, Linlin Shen, and Yongzhen Huang. “Invariant feature extraction for gait recognition using only one uniform model.” *Neurocomputing*, **239**:81–93, 2017.

[You99] Laurent Younes. “On the convergence of Markovian stochastic algorithms with rapidly decreasing ergodicity rates.” *Stochastics: An International Journal of Probability and Stochastic Processes*, **65**(3-4):177–228, 1999.

[YSZ15] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. “Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop.” *arXiv preprint arXiv:1506.03365*, 2015.

[YTT06] Shiqi Yu, Daoliang Tan, and Tieniu Tan. “A framework for evaluating the effect of view angle, clothing and carrying condition on gait recognition.” In *18th International Conference on Pattern Recognition*, volume 4, pp. 441–444. IEEE, 2006.

[ZB06] Xiaoli Zhou and Bir Bhanu. “Feature fusion of face and gait for human recognition at a distance in video.” In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 4, pp. 529–532. IEEE, 2006.

[ZB07] Xiaoli Zhou and Bir Bhanu. “Integrating face and gait for human recognition at a distance in video.” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, **37**(5):1119–1137, 2007.

[ZBP09] Guoying Zhao, Mark Barnard, and Matti Pietikainen. “Lipreading with local spatiotemporal descriptors.” *IEEE Transactions on Multimedia*, **11**(7):1254–1265, 2009.

[Zhu03] Song-Chun Zhu. “Statistical modeling and conceptualization of visual patterns.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **25**(6):691–712, 2003.

[ZM98] Song-Chun Zhu and David Mumford. “GRADE: Gibbs Reaction And Diffusion Equations.” In *International Conference on Computer Vision*, pp. 847–854, 1998.

[ZMB08] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. “Maximum Entropy Inverse Reinforcement Learning.” 2008.

[ZML16] Junbo Zhao, Michael Mathieu, and Yann LeCun. “Energy-based generative adversarial network.” *arXiv preprint arXiv:1609.03126*, 2016.

[ZWM97] Song-Chun Zhu, Ying Nian Wu, and David Mumford. “Minimax entropy principle and its application to texture modeling.” *Neural Computation*, **9**(8):1627–1660, 1997.

[ZZH11] Shuai Zheng, Junge Zhang, Kaiqi Huang, Ran He, and Tieniu Tan. “Robust view transformation model for gait recognition.” In *18th IEEE International Conference on Image Processing*, pp. 2073–2076. IEEE, 2011.