Title

Effective Learning of Descriptive and Generator Models and Learning Representations for Grid Cells and V1 Cells

Author

Gao, Ruiqi

Publication Date

2021

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Effective Learning of Descriptive and Generator Models and Learning Representations for Grid Cells and V1 Cells

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Statistics

by

Ruiqi Gao

2021

ABSTRACT OF THE DISSERTATION

Effective Learning of Descriptive and Generator Models and Learning Representations for Grid Cells and V1 Cells

by

Ruiqi Gao
Doctor of Philosophy in Statistics
University of California, Los Angeles, 2021
Professor Song-Chun Zhu, Chair

In recent decades, deep learning has achieved tremendous successes in supervised learning; however, unsupervised learning and representation learning, i.e., learning the hidden structure of the data without requiring expensive and time-consuming human annotation, remains a fundamental challenge, which probably underlies the gap between current artificial intelligence and the intelligence of a biological brain. In this thesis, we propose novel solutions to the problems in this area. Specifically, we work on deep generative modeling, an important approach of unsupervised learning, and representation learning inspired by structures in the brain.

- We propose **efficient algorithms for learning descriptive models**, which are also known as energy-based models (EBMs). Despite an appealing class of generative models with a number of desirable properties, the learning of descriptive models on high-dimensional space remains challenging, which involves computationally expensive Markov chain Monte Carlo (MCMC). To tackle this problem, we propose a **multi-grid modeling and sampling** method, which learns descriptive models at multiple scales or resolutions and the MCMC sampling follows a coarse-to-fine scheme. This approach enables efficient learning and sampling of

descriptive models from large-scale image datasets with small-budget MCMC. Later on, we extend this method to an improved version named **diffusion recovery likelihood**, where a sequence of descriptive models are proposed and learned on increasingly noisy versions of a dataset. Each descriptive model is trained by sampling from the conditional probability of the data at a certain noise level given their noisy versions at a higher noise level, which further releases the burden of MCMC.

- We develop **dynamic and motion-based generator models** which learn semantically meaningful vector representations for spatial-temporal processes such as dynamic textures and action sequences in video data. The models are capable of learning disentangled representations of appearance, trackable motion and intrackable motion in spatial-temporal processes in a fully unsupervised manner. We also propose an efficient learning algorithm named alternating back-propagation through time, which learns the proposed models using online MCMC inference without resorting to auxiliary networks.

- We propose **hybrid generative models** that integrate the advantages of different classes of generative models. Specifically, we propose a training algorithm **flow contrastive estimation** to jointly estimate a descriptive model and a flow-based model, in which the two models are iteratively updated based on a shared adversarial value function. The algorithm is an extension of noise contrastive estimation (NCE) and combines the flexibility of descriptive models and the tractability of flow-based models. We also study another hybrid model where the descriptive model serves as a correction or an exponential tilting of the flow-based model. We show that this model has a particularly simple form in the space of the latent variables of the flow-based model, and MCMC sampling of the descriptive model in the latent space mixes well and traverses modes in the data space.

- We propose an **optimization-based representational model of grid cells**. Grid cells exist in the mammalian medial entorhinal cortex (mEC) and are so named because individual neurons exhibit striking firing patterns that form hexagonal grids when the agent (such as

a rat) navigates in a 2D open field. To understand how grid cells perform path integration, we conduct theoretical analysis of a general representational model of grid cells where the 2D self-position of the agent is represented by a higher-dimensional vector and the 2D self-motion is represented by a general transformation of the vector. We identify two conditions for the general transformation and demonstrate an important geometric property of the general transformation, i.e., local conformal embedding. We further investigate the simplest transformation, i.e., the linear transformation, and uncover its explicit algebraic and geometric structure as a matrix Lie group of rotation. The model learns significant hexagon patterns of grid cells and is capable of accurate path integration.

- We extend the representational model of grid cells to an **optimization-based representational model of V1 simple cells**. V1 stands for the primary visual cortex in the mammalian brain, and V1 simple cells are highly specialized for low-level motion perception and pattern recognition. We propose a representational model of V1 simple cells which couples the following two components: (1) the vector representations of local contents of images and (2) the matrix representations of local pixel displacements caused by the relative motions between the agent and the objects in the 3D scene. The model can learn Gabor-like tunings of V1 simple cells and similar to V1 simple cells, the learned adjacent neurons have quadrature-phase relations.

The dissertation of Ruiqi Gao is approved.

Tao Gao

Hongjing Lu

Ying Nian Wu

Song-Chun Zhu, Committee Chair

University of California, Los Angeles

2021

To my parents and Yutan

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost I'd like to express my deepest gratitude towards my advisor Prof. Song-Chun Zhu and effectively my co-advisor Prof. Ying Nian Wu. Thank Song-Chun for giving me the opportunity to work in a large research lab and communicate and collaborate with lab mates of various research interests. He has always influenced me with his long-term vision and persistence for research. He gives me the courage to stick to difficult but valuable research topics, and reminds me to think about larger scope when working on technical details. Ying Nian is the best advisor I can ever imagine. One can think of many characteristics that make a great advisor: superb breadth and width of knowledge, creative and original research ideas, detailed instructions, generous encouragement, insightful and constructive discussions, strong support in internship and job search. Magically Ying Nian has all those, to the best extent possible. It is my great fortune to work with him for the past five years. What I learned from Ying Nian is much beyond research: the peace of mind, and a consistent set of principles that guide you through all hills and valleys in life. I am deeply impressed and influenced by his taste in research and also his attitude towards life.

I would like to thank Prof. Hongjing Lu and Prof. Tao Gao for serving as my committee members. Thanks for their time on my oral presentations and reviewing my thesis.

Many great minds have guided me through the journey. Jianwen Xie introduced me to the world of generative modeling in the first year of my PhD, and has been my mentor and friend ever since. He is the one who taught me the energy-based model and generator model. He also gave me the first internship opportunity in the summer of 2018. Our extensive collaborations lead to 10 joint research papers. Much gratitude goes to Jianwen for his indispensable role in this thesis. Yang Lu mentored me and gave me the strongest support on my first paper. He taught me a lot about coding and writing and gave me the confidence on doing research. I would like to thank Zhen Xu and Andrew Dai for hosting me at Google Brain during the summer of 2019, which leads to the flow contrastive estimation paper. In the same summer, I was fortunate enough to work with Durk Kingma and begin our fruitful collaborations. Durk hosted me at Google Brain in the summer

of 2020 and we have collaborated on the flow contrastive estimation and the diffusion recovery likelihood papers. I would like to thank Durk's incredibly strong support in my job search, and I cannot wait for continuing our collaborations. I also appreciate Ben Poole for cohosting me in the summer of 2020. I am fortunate enough to work with Xue-Xin Wei, from whom I learn all about neuroscience related to grid cells and place cells. Thank Xue-Xin for giving me career advice and unconditional support in my job search.

Many colleagues have supported me and played a mentorship role through my research, which I sincerely appreciate. Erik Nijkamp taught me the flow-based models and we have collaborated on the two hybrid models. Special thanks to Erik for the mutual support in summer 2019, when both of us interned at Google. Siyuan Huang taught me about 3D vision and perception, which leads to the collaboration on the work about V1 simple cells. I'd like to thank Yaxuan Zhu for close collaborations and for applying the grid cell model to learning representations of 3D camera poses. Thank Yang Song for the discussion and collaboration in summer 2020. Thank Will Grathwohl for stimulating discussions. I do appreciate the support from my other colleagues and collaborators, especially Tian Han, Xianglei Xing, Wenguan Wang, Zilong Zheng, Bo Pang, Pavel Sountsov, Srinivas Vasudevan, Junpei Zhou, and Yufan Ren.

I enjoyed the time spent with my friends. Thanks to Yuejiao Sun, Weinan Song, Pengpeng Ji, Ariel Pan, Haisong Lin, Jaden Tao, and many others, for board games and parties. Thanks to Jerry Xu, Tengyu Liu, Hanlin Zhu, Yixin Zhu, Feng Gao, Yuanlu Xu, Xiaofeng Gao, Yiling Chen, Jingyu Shao, Lifeng Fan, Arjun Akula, Mark Edmonds, Xiaojian Ma, Hangxin Liu, Jonathan Mitchell, Sirui Xie, Yuxing Qiu, Xu Xie, Luyao Yuan, Tianyang Zhao, Chi Zhang, Qing Li, Baoxiong Jia, Muzhi Han, Yixin Chen, Yizhou Zhao, and Shu Wang for chats about research and life.

During my job search, I was incredibly fortunate to have received guidance from researchers in various institutions. Besides those mentioned above, I would like to especially thank Tianfu Wu, Zhuowen Tu, Kevin Murphy, Yanqi Zhou, Han Zhang, Bo Dai, Colin Raffel, Adji Bousso Dieng, Chen Sun, Hang Zhao, Yi Zhu, Arash Vahdat, Jan Kautz, Sifei Liu, and Yuandong Tian, for helping me go through this stressful process.

Finally, thanks to my parents and Yutan Gu, for their continual support and love.

VITA

2016–2021	Graduate Research Assistant, UCLA VCLA, Dr. Song-Chun Zhu
2020	Summer Research Intern, Google Brain, Dr. Diederik P. Kingma
2019	Summer Research Intern, Google Brain, Dr. Zhen Xu
2018	Summer Research Intern, Hikvision Research USA, Dr. Jianwen Xie
2018	Ph.D. Candidate in Statistics, UCLA
2016	B.E. in Statistics, Peking University, China

PUBLICATIONS

$\star$ denotes equal contributions

On path integration of grid cells: group representation and isotropic scaling. **R Gao**, J Xie, XX Wei, SC Zhu, YN Wu. ArXiv, 2021.

Learning neural representation of camera pose with matrix representation of pose shift via view synthesis. Y Zhu, **R Gao**, S Huang, SC Zhu, YN Wu. CVPR, 2021.

Learning energy-based models by diffusion recovery likelihood. **R Gao**, Y Song, B Poole, YN Wu, DP Kingma. ICLR, 2021.

Learning energy-based model with flow-based backbone by neural transport MCMC. E Nijkamp*, **R Gao***, P Sountsov, S Vasudevan, B. Pang, SC Zhu, YN Wu. ArXiv, 2020.

Flow contrastive estimation of energy-based model. **R Gao**, E Nijkamp, DP Kingma, Z Xu, AM Dai, YN Wu. CVPR, 2020.

Learning V1 simple cells with vector representations of local contents and matrix representations

of local motions. **R Gao**, J Xie, S Huang, Y Ren, SC Zhu, YN Wu. ArXiv, 2020.

Generative VoxelNet: learning energy-based models for 3D shape synthesis and analysis. J Xie*, Z Zheng*, **R Gao**, W Wang, SC Zhu, YN Wu. TPAMI, 2020.

Deformable generator networks: unsupervised disentanglement of appearance and geometry. X Xing, **R Gao**, T Han, SC Zhu, and YN Wu. TPAMI, 2020.

Motion-based generator model: unsupervised disentanglement of appearance, trackable and intrackable motions in dynamic patterns. J Xie*, **R Gao***, Z Zheng, SC Zhu, YN Wu. AAAI, 2020.

Representation learning: a statistical perspective. J Xie, **R Gao**, E Nijkamp, SC Zhu, YN Wu. ARSIA, 2020.

Learning grid cells as vector representation of self-position coupled with matrix representation of self-motion. **R Gao***, J Xie*, SC Zhu, YN Wu. ICLR, 2019.

Learning dynamic generator model by alternating back-propagation through time. J Xie*, **R Gao***, Z Zheng, SC Zhu, YN Wu. AAAI, 2019.

Unsupervised disentanglement of appearance and geometry by deformable generator network. X Xing, T Han, **R Gao**, SC Zhu, YN Wu. CVPR, 2019.

A tale of three probabilistic families: discriminative, descriptive and generative models. YN Wu, **R Gao**, T Han, SC Zhu. Quarterly of Applied Mathematics, 2019.

Learning generative ConvNets via multigrid modeling and sampling. **R Gao***, Y Lu*, J Zhou, SC Zhu, YN Wu. CVPR, 2018.

Cooperative Training of Descriptor and Generator Networks. J Xie, Y Lu, **R Gao**, SC Zhu, YN Wu. TPAMI, 2018.

Learning descriptor networks for 3D shape synthesis and analysis. J Xie*, Z Zheng*, **R Gao**, W Wang, SC Zhu, YN Wu. CVPR, 2018.

Cooperative learning of energy-based model and latent variable model via MCMC teaching. J Xie, Y Lu, **R Gao**, YN Wu. AAAI, 2018.

Exploring Generative Perspective of Convolutional Neural Networks by Learning Random Field Models. Y Lu, **R Gao**, SC Zhu, YN Wu. Statistics and Its Interface, 2018.

CHAPTER 1

Introduction

Learning representations of data is an important problem in statistics and machine learning. While the origin of learning representations can be traced back to factor analysis and multidimensional scaling in statistics, it has become a central theme in deep learning with important applications in computer vision and computational neuroscience. However, two challenges remain for representation learning in the field of machine learning and deep learning: (1) data efficiency or label efficiency, the ability to learn from few datapoints or data with few labels; (2) generalization, the ability to transfer the learned knowledge to new tasks or environments. The goal of this thesis is to propose novel solutions to tackle these challenges.

Generative modeling, as an important unsupervised learning approach, has the promise of learning meaningful representations of the input while requiring little or no human annotations. Since such representations can be learned from large-scale unlabeled datasets and are not task-specific, downstream solutions based on the learned representations can potentially be more data-efficient and robust to change of tasks or environments. In this thesis, we work on developing and advancing various types of generative models, with the ultimate goal of learning better representations and intermediate applications such as image/video generations.

On the other hand, human is good at learning from few datapoints and can adapt to new tasks or environments quickly thanks to the biological brain. Understanding structures and functionalities of the brain can potentially inspire more human-like models in machine learning. To this end, in this thesis, we propose generic optimization-based representational models with the same functionality of the certain types of neurons in the brain, with the hope that the artificial neurons

can share similar activities or activation patterns to the biological neurons.

1.1 Generative modeling

One theme of this thesis is deep generative modeling. As the name suggests, generative models can generate examples such as images, texts, and speeches. Equipped with deep neural networks as powerful function approximators, modern generative models can generate highly realistic images and texts. Such generative models are consistent with the brain's ability to imagine and fantasize. In more technical terms, such models provide a unified framework for supervised, unsupervised and semi-supervised learning, the last of which is about learning from a large amount of unlabeled data and a small amount of labeled data. In this thesis, we focus on developing and advancing the descriptive models and generator models, which represent two major directions in the area of deep generative models.

1.1.1 Descriptive model

The descriptive model, also known as the energy-based model (EBM), defines an unnormalized probability density on high dimensional input examples such as images via an energy function so that examples with low energies have high probabilities to be observed. This model can be traced back to the Gibbs distribution in statistical mechanics. Compared to other types of generative models, the descriptive model enjoys several distinct advantages, such as the free-form energy function with almost no constraints, which can be parametrized by any existing network structure, and direct correspondence to discriminative models, which can therefore unify supervised learning and unsupervised learning in an elegant way. The learning of such a model follows an "analysis by synthesis" scheme, which involves sampling data from the current model, and then updating the model parameters based on the difference between the sampled data and the observed data. The challenge lies in the fact that the sampling step usually requires iterative Markov chain Monte Carlo (MCMC), such as Langevin dynamics or Hamiltonian Monte Carlo, which can be time-consuming

and difficult to converge on high-dimensional data space.

To solve this problem, We propose a *multi-grid modeling and sampling* method that achieves efficient learning of descriptive models on large-scale image datasets. The idea is to learn descriptive models at a sequence of resolutions and the sampling follows a progressively coarse-to-fine scheme. The learning scheme has a similar effect to simulated annealing where the temperature of the model decreases gradually, and therefore makes MCMC more efficient.

More recently, we further extend this method to an algorithm named *diffusion recovery likelihood*, where a sequence of descriptive models are proposed and learned on increasingly noisy versions of a dataset. Each marginal descriptive model is trained with recovery likelihood, which maximizes the conditional probability of the data at a certain noise level given their noisy versions at a higher noise level. Optimizing recovery likelihood is more tractable than marginal likelihood, as sampling from the conditional distributions is much easier than sampling from the marginal distributions. That is, we learn marginal descriptive models at different noise levels and use sequential conditional distributions to ease the sampling. Using this method, we push descriptive models to the state-of-the-art of generative models in terms of both sample quality and density estimation.

1.1.2 Generator model

A generator model assumes latent structure behind the input signals, usually in the form of a latent vector, and maps the latent vector to the high-dimensional example such as an image by a non-linear transformation parametrized by a deep neural network. The challenge is to infer such latent variables given observed examples during training. We aim at developing an efficient learning method and meanwhile learning disentangled latent representations for different components of the data, such as the appearance and motion in videos.

To this end, we develop a *dynamic generator model* for spatial-temporal processes, where the latent vector follows a transition model and the transition is parametrized by a feedforward neural network or in general a recurrent neural network, and the latent vector generates the video

frame by a deep network as the emission model. We also propose a maximum likelihood learning algorithm called *alternating back-propagation through time*, which alternates between inferring the latent noise vectors by Langevin dynamics and updating the parameters by gradient descent on a reconstruction error. The gradient computations in both steps can be accomplished by back-propagation through time. Unlike VAE [KW13] or GAN [GPM14], our method can learn generator models without auxiliary networks.

More recently, we further generalize the model to explicitly incorporate trackable and intrackable motion, which can be inferred automatically from the video data in an unsupervised manner. Such models can be useful for learning intuitive physics, and can serve as the dynamic model for model-based reinforcement learning.

1.1.3 Hybrid model

We also work on integrating or jointly training different families of generative models to combine the best of their worlds. Specifically, we develop a *flow contrastive estimation* method to jointly learn a descriptive model and a flow-based model in a discriminative manner, without resorting to MCMC. A flow-based model transforms a latent vector to the image via a sequence or a flow of transformations. Unlike the generator model, the latent vector of the flow-based model is of the same dimension as the image, and the transformations are invertible and the probability density of the data is analytically tractable. However, due to the highly constrained form of the transformations, one usually needs to compose a large number of transformations in order to model complex signals such as images. Our joint training method combines the tractability of the flow-based model and the flexibility of the descriptive model. We also adapt the proposed method to semi-supervised learning and achieve competitive performance on this task.

Besides, we propose a tight coupling of a descriptive model and a flow-based model, by learning a descriptive model with a flow-based model serving as a backbone, so that the descriptive model is a correction or an exponential tilting of the flow-based model. We show that the model

has a particularly simple form in the space of the latent variables of the flow-based model, and MCMC sampling of the descriptive model in the latent space mixes well and traverses modes in the data space. This enables proper sampling and learning of descriptive models. To the best of our knowledge, our work is the first where MCMC sampling is mixing for descriptive models parametrized by modern convolution neural networks.

1.2 Biological plausible representational models

For advancing representation learning in machine learning, it is appealing to borrow inspirations from the biological brain and develop representational models that can meaningfully approximate and explain the structure and activities in the brain. This is the second theme of this thesis. The hope is that by learning and approximating the brain, we would end up with more powerful and meaningful representational models in machine learning. In the first theme of generative modeling, the vector representation in generator models has been widely used and explored especially after the development of deep learning. Sometimes the vector is colloquially called "thought vector". In this theme, we show that it is also of fundamental importance to learn matrix representation for continuous motions or transformations.

1.2.1 Representational models of grid cells

Grid cells, together with place cells, are two types of neurons widely existing in mammalian brains and are in charge of an agent's self-navigation. The discovery of grid cells and place cells was honored by the Nobel prize in physiology and medicine in 2014. Place cells, located in the hippocampus, fire at a single position when the agent (e.g., a rat) moves within an environment, while more surprisingly, grid cells, located in the medial entorhinal cortex (mEC), fire at regular hexagon grids of positions as the agent moves. It is desirable to study representational models that can learn and explain such hexagon grid patterns of grid cells as well as the connection between grid cells and place cells.

To this end, we develop a representational model for grid cells, where the grid cells form a vector representation of the self-position of the agent, and when the agent moves, the vector is rotated by a matrix that represents the displacement. This model gives a natural explanation of the hexagon grid patterns observed in grid cells. Interestingly, it is similar to the mathematical formulation of quantum physics, where the position of a particle is represented by a vector and the displacement is represented by a matrix or an operator acting on the vector. Matrix representation of group is also of fundamental importance in modern mathematics.

More recently, we conduct a theoretical analysis of a more general representation model of path integration by grid cells, where the 2D self-position is encoded as a high dimensional vector, and the 2D self-motion is represented by a general transformation of the vector. We identify two conditions for the transformation. One is a group representation condition that is necessary for path integration. The other is an isotropic scaling condition that ensures locally conformal embedding, so that the error in the vector representation translates proportionally to the error in the 2D self-position. Then we investigate the simplest transformation, i.e., the linear transformation, uncover its explicit algebraic and geometric structure as *matrix Lie group of rotation*, and integrate the path integration model of grid cells and the basis expansion model connecting grid cells and place cells. Finally, with our optimization-based approach, we manage to learn hexagon grid patterns that share similar properties of the grid cells in the rodent brain. The learned model is capable of accurate long-distance path integration.

1.2.2 Representational models of V1 simple cells

V1 stands for the primary visual cortex in the mammalian brain, and it is highly specialized for motion perception and pattern recognition. It is known that V1 simple cells exhibit Gabor-like tunings and adjacent simple cells have quadrature-phase relations (e.g., sine and cosine pairs). The study of V1 simple cells motivates one of the most prominent models in machine learning, sparse coding. Different from sparse coding, we develop a representational model of V1 simple cells by incorporating the representation of perceived motions, where we learn a vector representation

for the local content received from sensory, and a matrix representation for the local displacement caused by the relative motions between the agent and the objects in the 3D scenes. This model naturally learns and explains patterns of V1 simple cells, and can be used for motion inference.

1.2.3 Unified and general framework

Summarizing the representational models of grid cells and V1 simple cells, while vector representation may be considered a "noun", matrix representation may be considered a "verb". In terms of neuroscience, the vector representation corresponds to the activities of neurons, and the matrix representation corresponds to synaptic connections. This general framework may be applicable to more activities in brain and machine intelligence. We also apply this idea to modeling the change of images under ego-motion, which is an important problem in computer vision and human vision. It may shed light on how the brain represents the three-dimensional environment.

1.3 Research questions and contributions

We formulate the research questions and contributions in this thesis as follows.

Research question 1: *Is it possible to achieve efficient learning and proper sampling of deep descriptive models by maximum likelihood type of algorithms?*

In Chapter 2 and [GLZ18], we propose multi-grid modeling and sampling algorithm that achieves efficient learning of deep descriptive models on large image datasets. After training, we can generate realistic samples from the learned models with a small budget MCMC.

Despite being able to learn and generate realistic samples, MCMC is still computationally expensive and generally not converging, in the sense that samples from long-run Markov chains can differ greatly from observed samples. Thus in Chapter 2 and [GSP20], we propose a more efficient training algorithm, diffusion recovery likelihood, to tractably learn and sample from a sequence of descriptive models trained on increasingly noisy versions of a dataset. With this approach, we

push descriptive models to state-of-the-art generative models in terms of both sample quality and likelihood estimation. Moreover, we demonstrate that the long-run MCMC samples do not diverge and still represent realistic images.

To further encourage MCMC mixing, in Chapter 4 and [NGS20], we propose to learn a descriptive model with a flow-based model serving as a backbone. We show that the model has a particularly simple form in the latent space of the flow-based model, and sampling of the descriptive model by neural transport MCMC mixes well and is able to traverse modes in the data space.

Research question 2: *Is it possible to estimate deep descriptive models in a completely MCMC-free manner?*

In Chapter 4 and [GNK20], we answer this question in the affirmative. Specifically, we propose a training method to jointly estimate a descriptive model and a flow-based model, in which the two models are iteratively updated based on a shared value function. The update of the descriptive model is based on noise contrastive estimation (NCE), with the flow-based model serving as an adaptively updated strong noise distribution. With this NCE-based framework, we can totally avoid MCMC sampling. We show the efficacy of the learned descriptive model via unsupervised feature learning. Furthermore, the proposed training method can be easily adapted to semi-supervised learning, using the direct correspondence of the descriptive model to a discriminative model. We achieve competitive results to the state-of-the-art semi-supervised learning methods.

Research question 3: *How can we propose a generator model for modeling spatial-temporal processes and develop the corresponding learning algorithm?*

In Chapter 3 and [XGZ19], we propose a dynamic generator that contains two components: (1) an emission model which is a generator network that maps the state vector to the frame image, and (2) a transition model which is a recurrent neural network in the state space. We also develop an efficient learning algorithm for the dynamic generator model, namely alternating back-propagation through time. This algorithm does not require any auxiliary network. In Chapter 3 and

[XGZ20], we further generalize the model by introducing latent state vectors that explicitly model trackable and intrackable motion. This generalized model can learn disentangled representations of appearance, trackable and intrackable motion in a fully unsupervised manner.

Research question 4: *Can we develop optimization-based representational models with minimally simple assumptions that can meaningfully approximate activities of neurons in biological brains?*

We answer this question by studying two types of neurons in mammalian brains: (1) grid cells in the medial entorhinal cortex (mEC), which is hypothesized to participate in the path integration process, and (2) V1 simple cells in the primary visual cortex, which is essential to motion perception and pattern recognition.

In Chapter 5 and [GXZ18a, GXW20], we conduct a theoretical analysis of a general representational model of path integration by grid cells, where the 2D self-position is encoded as a higher dimensional vector, and the 2D self-motion is represented by a general transformation of the vector. We identify two minimally simple conditions on the transformation, under which we derive the clear local geometric structure of the vector representation in the neural space and a natural explanation to the error correction by grid cells. By investigating the simplest special case of the transformation, i.e., the linear transformation, we uncover its explicit algebraic and geometric structure as a matrix Lie group of rotation. With our optimization-based approach, we can learn significant hexagon grid patterns that share similar properties of the grid cells in the rodent brain. The learned model is capable of accurate long-distance path integration.

In Chapter 6 and [GXZ18b], we propose a representational model that may shed light on V1 simple cells. This model shares a similar idea to the representational model of grid cells in [GXZ18a]. Given an image pair such as consecutive video frames that are related by local displacements, the local contents of images are represented by vectors and the local displacements of the images are represented by matrices operating on the vectors. With this simple model, we are able to learn Gabor-like filter pairs of quadrature phases, which match the profile of Macaque V1 simple cells. We also achieve competitive results on optical flow estimation with the learned

model.

CHAPTER 2

Descriptive models

2.1 Deep descriptive models

This chapter studies the problem of learning deep descriptive models, also known as deep energy-based models (EBMs) [LCH06, Hin02a, HOW06, HOT06, SH09, LGR09, NCK11, LZW16, XLZ16, XZW17a, JLT17] of images [1]. EBMs define an unnormalized density that is the exponential of the negative *energy function*. The energy function is directly defined as a (learned) scalar function of the input, and is often parameterized by a neural network. Specifically, let $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ denote a training example, and $p_\theta(\mathbf{x})$ denote a model's probability density function that aims to approximates $p_{\text{data}}(\mathbf{x})$. An EBM is defined as:

$$p_\theta(\mathbf{x}) = \frac{1}{Z_\theta} \exp(f_\theta(\mathbf{x})), \tag{2.1}$$

where $Z_\theta = \int \exp(f_\theta(\mathbf{x}))d\mathbf{x}$ is the partition function, which is analytically intractable for high-dimensional $\mathbf{x}$. For images, we parameterize $f_\theta(\mathrm{x})$ with a convolutional neural network with a scalar output.

2.1.1 Maximum likelihood estimation

The energy-based model can be estimated from unlabeled data by maximum likelihood estimation (MLE). Suppose we observe training examples $\{x_i, i = 1, ..., n\}$ from unknown true distribution $p_{\text{data}}(x)$. We can view this dataset as forming empirical data distribution, and thus expectation

[1] The main contributions in this chapter were first described in [GLZ18, GSP20].

with respect to $p_{\text{data}}(x)$ can be approximated by averaging over the training examples. In MLE, we seek to maximize the log-likelihood function

$$\mathscr{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log p_\theta(\mathbf{x}_i) \doteq \mathrm{E}_{\mathbf{x}\sim p_{\text{data}}}[\log p_\theta(\mathbf{x})]. \tag{2.2}$$

Maximizing the log-likelihood function is equivalent to minimizing the Kullback-Leibler divergence $\mathrm{KL}(p_{\text{data}}||p_\theta)$ for large n. Its gradient can be written as:

$$-\frac{\partial}{\partial\theta}\mathrm{KL}(p_{\text{data}}||p_\theta) = \mathrm{E}_{p_{\text{data}}}\left[\frac{\partial}{\partial\theta} f_\theta(x)\right] - \mathrm{E}_{p_\theta}\left[\frac{\partial}{\partial\theta} f_\theta(x)\right], \tag{2.3}$$

which is the difference between the expectations of the gradient of $f_\theta(x)$ under p_{data} and p_θ respectively. The expectations can be approximated by averaging over the observed examples and synthesized samples generated from the current model $p_\theta(x)$ respectively. Generating synthesized samples from $p_\theta(\mathbf{x})$ can be done with Markov Chain Monte Carlo (MCMC) such as Langevin dynamics (or Hamiltonian Monte Carlo [GC11]), which iterates

$$\mathbf{x}^{\tau+1} = \mathbf{x}^\tau + \frac{\delta^2}{2}\nabla_{\mathbf{x}} f_\theta(\mathbf{x}^\tau) + \delta\boldsymbol{\epsilon}^\tau, \tag{2.4}$$

where τ indexes the time, δ is the step size, and $\boldsymbol{\epsilon}^\tau \sim \mathscr{N}(0, \boldsymbol{I})$. The MLE of $p_\theta(x)$ seeks to cover all the models of $p_{\text{data}}(x)$. Given the flexibility of model form of $f_\theta(x)$, the MLE of $p_\theta(x)$ has the chance to approximate $p_{\text{data}}(x)$ reasonably well.

Challenge of learning EBMs. The difficulty lies in the fact that the probability distribution or the energy function of the learned model is likely to be multi-modal if the training data are high dimensional and highly varied. The MCMC may have difficulty traversing different modes and may take a long time to converge.

2.1.2 Modifications of maximum likelihood estimation

In this subsection, we review modifications of maximum likelihood estimation (MLE) for learning energy-based models.

Contrastive divergence. The MCMC sampling of p_θ may take a long time to converge, especially if the learned p_θ is multi-modal, which is often the case because p_{data} is usually multi-modal. In order to learn from large datasets, we can only afford small budget MCMC, i.e., within each learning iteration, we can only run MCMC for a small number of steps. To meet such a challenge, [Hin02a] proposed the contrastive divergence (CD) method, where within each learning iteration, we initialize the finite-step MCMC from each $\mathbf{x}$ in the current training batch to obtain a synthesized example $\tilde{\mathbf{x}}$. The parameters are then updated according to the learning gradient (2.3).

Let M_θ be the transition kernel of the finite-step MCMC that samples from $p_\theta(\mathbf{x})$. For any probability distribution $p(\mathbf{x})$ and any Markov transition kernel M, let $Mp(\mathbf{x}') = \int p(\mathbf{x})M(\mathbf{x}, \mathbf{x}')d\mathbf{x}$ denote the marginal distribution obtained after running M starting from p. The learning gradient of CD approximately follows the gradient of the difference between two Kullback-Leibler (KL) divergences:

$$\text{KL}(p_{\text{data}} \| p_\theta) - \text{KL}(M_\theta p_{\text{data}} \| p_\theta), \tag{2.5}$$

thus the name "contrastive divergence". If $M_\theta p_{\text{data}}$ is close to p_θ, then the second divergence is small, and the CD estimate is close to maximum likelihood which minimizes the first divergence. However, it is likely that p_{data} and the learned p_θ are multi-modal. It is expected that p_θ is smoother than p_{data}, i.e., p_{data} is "colder" than p_θ in the language of simulated annealing [KGV83]. If p_{data} is different from p_θ, it is unlikely that $M_\theta p_{\text{data}}$ becomes much closer to p_θ due to the trapping of local modes. This may lead to bias in the CD estimate.

A persistent version of CD [Tie08] is to initialize the MCMC from the observed $\mathbf{x}_i$ in the beginning, and then in each learning epoch, the MCMC is initialized from the synthesized $\tilde{\mathbf{x}}_i$ obtained in the previous epoch. The persistent CD may still face the challenge of traversing and exploring different local energy minima.

Modified and adversarial CDs. The original CD initializes MCMC sampling from the data distribution p_{data}. We may modify it by initializing MCMC sampling from a given distribution p_0,

in the hope that $M_\theta p_0$ is closer to p_θ than $M_\theta p_{\text{data}}$. The learning gradient approximately follows the gradient of

$$\text{KL}(p_{\text{data}} \| p_\theta) - \text{KL}(M_\theta p_0 \| p_\theta). \tag{2.6}$$

That is, we run a finite-step MCMC from a given initial distribution P_0, and use the resulting samples as synthesized examples to approximate the expectation in (2.3). The approximation can be made more accurate using annealed importance sampling [Nea01a]. Following the idea of simulated annealing, p_0 should be a "smoother" distribution than p_θ (the extreme case is to start from white noise p_0). Unlike persistent CD, here the finite-step MCMC is non-persistent, sometimes also referred to as "cold start", where the MCMC is initialized from a given p_0 within each learning iteration, instead of from the examples synthesized by the previous learning epoch. The cold start version is easier to implement for mini-batch learning.

With the multi-grid method (to be introduced in the next section), at each grid, p_0 is the distribution of the images generated by the previous coarser grid. At the smallest grid, p_0 is the one-dimensional histogram of the 1×1 versions of the training images.

Another possibility is to recruit a generator network $q_\alpha(\mathbf{x})$ as an approximated direct sampler [KB16, DAB17], so that p_θ and q_α can be jointly learned by the adversarial CD:

$$\min_{p_\theta} \max_{q_\alpha} \left[\text{KL}(p_{\text{data}} \| p_\theta) - \text{KL}(q_\alpha \| p_\theta)\right]. \tag{2.7}$$

That is, the learning of p_θ is modified CD with q_α supplying synthesized examples, and the learning of q_α is based on $\min_{q_\alpha} \text{KL}(q_\alpha \| p_\theta)$, which is a variational approximation. The adversarial CD is related to Wasserstein GAN [ACB17b], except that the former regularizes the entropy of the generator, while the latter regularizes the critic.

[XLG17] also studied the problem of joint learning of the energy-based model and the generator model. The learning of the energy-based model is based on the modified CD:

$$\text{KL}(p_{\text{data}} \| p_\theta) - \text{KL}(M_\theta q_\alpha \| p_\theta), \tag{2.8}$$

with q_α taking the role of P_0, whereas the learning of the generator is based on how $M_\theta q_\alpha$ modifies q_α, and is accomplished by $a_{t+1} = \arg\min_\alpha \mathrm{KL}(M_\theta q_{\alpha_t} \| q_\alpha)$, i.e., q_α accumulates MCMC transitions to be close to the stationary distribution of M_θ, which is p_θ.

In this chapter, we would not consider recruiting a generator network, so that we do not need to worry about the mismatch between the generator model and the energy-based model. In other words, instead of relying on a learned approximate direct sampler, we endeavor to develop a small budget MCMC for sampling.

2.1.3 Correspondence to discriminative models

The energy-based model corresponds to a discriminative model or a classifier in the following sense [DLW14, XLZ16, Tu07, LJT17, JLT17]. Suppose there are K categories and for each category k, the data is modeled by an EBM $p_{\theta_k}(\mathbf{x})$ with parameters θ_k, in addition to the background category $p_0(\mathbf{x})$. The neural networks $f_{\theta_k}(\mathbf{x})$ for $k = 1, ..., K$ may share common lower layers. Let ρ_k be the prior probability of category k, $k = 0, ..., K$. Then by Bayes rule, the posterior probability for classifying an example $\mathbf{x}$ to the category k is a softmax multi-class classifier

$$p(k|\mathbf{x}) = \frac{\exp(f_{\theta_k}(\mathbf{x}) + b_k)}{\sum_{k=0}^{K} \exp(f_{\theta_k}(\mathbf{x}) + b_k)}, \tag{2.9}$$

where $b_k = \log(\rho_k/\rho_0) - \log Z(\theta_k)$, and for $k = 0$, $f_{\theta_0}(\mathbf{x}) = 0$, $b_0 = 0$. Conversely, if we have the softmax classifier (2.9), then the distribution of each category is $p_{\theta_k}(\mathbf{x})$ of the form (2.1).

That is to say, an EBM can be derived from the commonly used discriminative model, but unlike the discriminative model, an EBM is endowed with the gift of imagination in that it can generate images by sampling from the probability distribution of the model. As a result, the EBM can be learned in an unsupervised setting without requiring class labels. Conversely, a class-conditional EBM can also be turned into a discriminative model for classification.

2.2 Multi-grid modeling and sampling

To address the challenge of learning energy-based models under the constraint of finite budget MCMC, we propose a multi-grid method to learn EBMs at multiple scales or grids [GLZ18]. Specifically, for each training image, we obtain its multi-grid versions by repeated down-scaling. Our method learns a separate EBM at each grid. Within each iteration of our learning algorithm, for each observed training image, we generate the corresponding synthesized images at multiple grids. Specifically, we initialize the finite-step MCMC sampling from the minimal 1×1 version of the training image, and the synthesized image at each grid serves to initialize the finite-step MCMC that samples from the model of the subsequent finer grid. See Figure 2.1 for an illustration, where we sample images sequentially at 3 grids, with 30 steps of the Langevin dynamics at each grid. After obtaining the synthesized images at the multiple grids, the models at the multiple grids are updated separately and simultaneously based on the differences between the synthesized images and the observed training images at different grids.

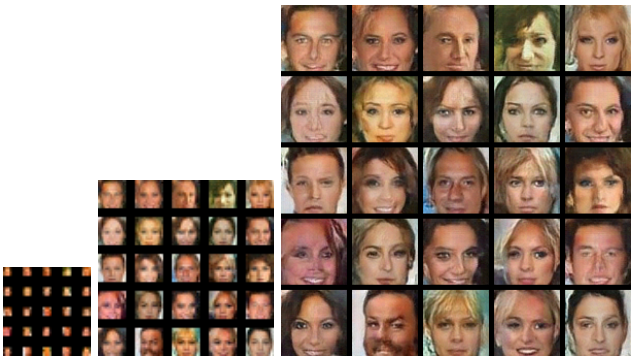


Figure 2.1: Synthesized images at multi-grids. From left to right: 4×4 grid, 16×16 grid and 64×64 grid. Synthesized image at each grid is obtained by 30 step Langevin sampling initialized from the synthesized image at the previous coarser grid, beginning with the 1×1 grid.

The advantages of the proposed method are as follows.

(1) The finite-step MCMC is initialized from the 1×1 version of the observed image, instead of the original observed image. Thus the synthesized image is much less biased by the observed

image compared to the original CD.

(2) The learned models at coarser grids are expected to be smoother than the models at finer grids. Sampling the models at increasingly finer grids sequentially is like a simulated annealing process [KGV83] that helps the MCMC to mix.

(3) Unlike the original CD or persistent CD, the learned models are equipped with a fixed budget MCMC to generate new synthesized images from scratch, because we only need to initialize the MCMC by sampling from the one-dimensional histogram of the 1×1 version of the training images.

We show that the proposed method can learn realistic models of images. The learned models can be used for image processing such as image inpainting. The learned feature maps can be used for subsequent tasks such as classification.

The contributions of this section are as follows. We propose a multi-grid method for learning energy-based models. We show empirically that the proposed method outperforms the original CD, persistent CD, as well as single-grid learning. More importantly, we show that a small budget MCMC is capable of generating diverse and realistic patterns.

2.2.1 Related work

Our method is related to CD [Hin02a] for training energy-based models. In general, both the data distribution of the observed training examples and the learned model distribution can be multi-modal, and the data distribution can be even more multi-modal than the model distribution. The finite-step MCMC of CD initialized from the data distribution may only explore local modes around the training examples, thus the finite-step MCMC may not get close to the model distribution. This can also be the case with persistent CD [Tie08]. In contrast, our method initializes the finite-step MCMC from the minimal 1×1 version of the original image, and the sampling of the model at each grid is initialized from the image sampled from the model at the previous coarser grid. The model distribution at the coarser grid is expected to be smoother than the model

distribution at the finer grid, and the coarse to fine MCMC is likely to generate varied samples from the learned models. As a result, the learned models obtained by our method can be closer to the maximum likelihood estimate than the original CD.

The multi-grid Monte Carlo method originated from statistical physics [GS89]. The motivation for multi-grid Monte Carlo is that reducing the scale or resolution leads to a smoother or less multi-modal distribution. Our work is perhaps the first to apply multi-grid sampling to the learning of deep energy-based models. The difference between our method and the multi-grid MCMC in statistical physics is that in the latter, the distribution of the lower resolution is obtained from the distribution of the higher resolution. In our work, the models at different grids are learned from training images at different resolutions directly and separately.

Our learning method is based on maximum likelihood. Building on the early work of [Tu07], [JLT17, LJT17] have developed an introspective learning method to learn the energy-based model, where the energy function is discriminatively learned. It is possible to apply multi-grid learning and sampling to their method.

2.2.2 Proposed algorithm

For an image $\mathbf{x}$, let $(\mathbf{x}^{(s)}, s = 0, ..., S)$ be the multi-grid versions of $\mathbf{x}$, with $\mathbf{x}^{(0)}$ being the minimal 1×1 version of $\mathbf{x}$, and $\mathbf{x}^{(S)} = \mathbf{x}$. For each $\mathbf{x}^{(s)}$, we can divide the image grid into squared blocks of $d \times d$ pixels. We can reduce each $d \times d$ block into a single pixel by averaging the intensity values of the $d \times d$ pixels. Such a down-scaling operation maps $\mathbf{x}^{(s)}$ to $\mathbf{x}^{(s-1)}$. Conversely, we can also define an up-scaling operation, by expanding each pixel of $\mathbf{x}^{(s-1)}$ into a $d \times d$ block of constant intensity to obtain an up-scaled version $\hat{\mathbf{x}}^{(s)}$ of $\mathbf{x}^{(s-1)}$. The up-scaled $\hat{\mathbf{x}}^{(s)}$ is not identical to the original $\mathbf{x}^{(s)}$ because the high resolution details are lost. The mapping from $\mathbf{x}^{(s)}$ to $\mathbf{x}^{(s-1)}$ is a linear projection onto a set of orthogonal basis vectors, each of which corresponds to a $d \times d$ block. The up-scaling operation is a pseudo-inverse of this linear mapping. In general, d does not even need to be an integer (e.g., $d = 1.5$) for the existence of the linear mapping and its pseudo-inverse.

Let $p^{(s)}_{\theta^{(s)}}(\mathbf{x}^{(s)})$ be the energy-based model at grid s. $p^{(0)}$ can be simply modeled by a one-dimensional histogram of $\mathbf{x}^{(0)}$ pooled from the 1×1 versions of the training images.

Within each learning iteration, for each training image $\mathbf{x}_i$ in the current learning batch, we initialize the finite-step MCMC from the 1×1 image $\mathbf{x}^{(0)}_i$. For $s = 1, ..., S$, we sample from the current $p^{(s)}_{\theta^{(s)}}(\mathbf{x}^{(s)})$ by running l steps of the Langevin dynamics from the up-scaled version of $\tilde{\mathbf{x}}^{(s-1)}_i$ sampled at the previous coaser grid. After that, for $s = 1, ..., S$, we update the model parameters $\theta^{(s)}$ based on the difference between the synthesized $\{\tilde{\mathbf{x}}^{(s)}_i\}$ and the observed $\{\mathbf{x}^{(s)}_i\}$ according to Eq. (2.3).

Algorithm 1 provides the details of the multi-grid method.

In the above sampling scheme, $p^{(0)}$ can be sampled directly because it is a one-dimensional histogram. Each $p^{(s)}$ is expected to be smoother than $p^{(s+1)}$. Thus the sampling scheme is similar to simulated annealing, where we run finite-step MCMC through a sequence of probability distributions that are increasingly multimodal (or cold), in the hope of reaching and exploring major modes of the model distributions. The learning process then shifts these major modes toward the observed examples, while sharpening these modes along the way, in order to memorize the observed examples with these major modes of the model distributions.

Let $p^{(s)}_{\text{data}}$ be the data distribution of $\{\mathbf{x}^{(s)}_i\}$. Let $p^{(s)}_{\theta^{(s)}}$ be the model at grid s. Let $P^{(s)}_{\theta^{(s-1)}}$ be the up-scaled version of the model $p^{(s-1)}_{\theta^{(s-1)}}$. Specifically, let $\mathbf{x}^{(s-1)} \sim p^{(s-1)}_{\theta^{(s-1)}}$ be a random example at grid $s-1$, and let $\hat{\mathbf{x}}^{(s)}$ be the up-scaled version of $\mathbf{x}^{(s-1)}$, then $P^{(s)}_{\theta^{(s-1)}}$ is the distribution of $\hat{\mathbf{x}}^{(s)}$. Let $M^{(s)}_{\theta^{(s)}}$ be the Markov transition kernel of l-step Langevin dynamics that samples $p^{(s)}_{\theta^{(s)}}$. The learning gradient of the multi-grid method at grid s approximately follows the gradient of the difference between two KL divergences:

$$\text{KL}\left(p^{(s)}_{\text{data}} \| p^{(s)}_{\theta^{(s)}}\right) - \text{KL}\left(M^{(s)}_{\theta^{(s)}} P^{(s)}_{\theta^{(s-1)}} \| p^{(s)}_{\theta^{(s)}}\right). \tag{2.10}$$

$P^{(s)}_{\theta^{(s-1)}}$ is smoother than $p^{(s)}_{\theta^{(s)}}$, and $M^{(s)}_{\theta^{(s)}}$ will evolve $P^{(s)}_{\theta^{(s-1)}}$ to a distribution close to $p^{(s)}_{\theta^{(s)}}$ by creating details at the current resolution. If we use the original CD by initializing MCMC from $p^{(s)}_{\text{data}}$, then we are sampling a multi-modal (cold) distribution $p^{(s)}_{\theta^{(s)}}$ by initializing from a presumably even

Algorithm 1 Multi-grid sampling and learning

Input:

(1) training examples $\{\mathbf{x}_i^{(s)}, s = 1, ..., S, i = 1, ..., n\}$,

(2) number of Langevin steps l,

(3) number of learning iterations T.

Output:

(1) estimated parameters $(\theta^{(s)}, s = 1, ..., S)$,

(2) synthesized examples $\{\tilde{\mathbf{x}}_i^{(s)}, s = 1, ..., S, i = 1, ..., n\}$.

1: Let $t \leftarrow 0$, initialize $\theta^{(s)}, s = 1, ..., S$.

2: **repeat**

3: For $i = 1, ..., n$, initialize $\tilde{\mathbf{x}}_i^{(0)} = \mathbf{x}_i^{(0)}$.

4: For $s = 1, ..., S$, initialize $\tilde{\mathbf{x}}_i^{(s)}$ as the up-scaled version of $\tilde{\mathbf{x}}_i^{(s-1)}$, and run l steps of the Langevin dynamics to evolve $\tilde{\mathbf{x}}_i^{(s)}$, each step following Eq. (2.4).

5: For $s = 1, ..., S$, update $\theta_{t+1}^{(s)} = \theta_t^{(s)} + \gamma_t L'(\theta_t^{(s)})$, with step size γ_t, where $L'(\theta_t^{(s)})$ is computed according to equation (2.3).

6: Let $t \leftarrow t + 1$.

7: **until** $t = T$

more multi-modal (or colder) distribution $p_{\text{data}}^{(s)}$, and we may not expect the resulting distribution to be close to the target $p_{\theta^{(s)}}^{(s)}$.

2.2.3 Experiments

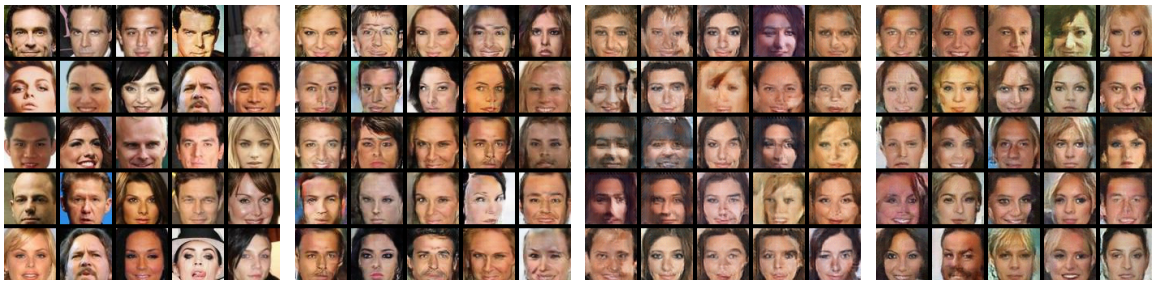
Project page: The code and more results can be found at `http://www.stat.ucla.edu/~ruiqigao/multigrid/main.html`.

We learn the models at 3 grids: 4×4, 16×16 and 64×64, which we refer to as grid1, grid2

and grid3, respectively. That is, we set $S = 3$ (number of grids), $d = 4$ (reducing each 4×4 block to a pixel in the down-scaling operation).

We conduct qualitative and quantitative experiments to evaluate our method with respect to several baseline methods. The first baseline is the single-grid method: starting from a 1×1 image, we directly upscale it to 64×64 and sample a 64×64 image using a single EBM. The other two baselines are CD1 (running 1 step Langevin dynamics from the observed images) and persistent CD. Both CD baselines initialize the MCMC sampling from the observed images. See [GLZ18] for more details.

2.2.3.1 Synthesis



Observed DCGAN single-grid method multi-grid method

Figure 2.2: Synthesized images from models learned from the CelebA dataset. From left to right: observed images, images synthesized by DCGAN [RMC15], single-grid method and multi-grid method. CD1 and persistent CD cannot synthesize realistic images and their results are not shown.

We learn multi-grid models from five datasets: CelebA [LLW15], Large-scale Scene Understanding (LSUN) [SX15], CIFAR-10 [KH09], Street View Housing Numbers (SVHN) [NWC11] and MIT places205 [ZLX14]. In the CelebA dataset, we randomly sample 10,000 images for training. Figure 2.2 show synthesized images generated by models learned from CelebA dataset. We also show synthesized images generated by models learned by DCGAN [RMC15] and the single-grid method. CD1 and persistent CD cannot synthesize realistic images, thus we do not bother

(a) Original images (b) Synthesized images

Figure 2.3: Synthesized images generated by the multi-grid models learned from the LSUN bedrooms dataset.

to show their synthesis results. Compared with the single-grid method, images generated by the multi-grid method are more realistic. The results from multi-grid models are comparable to the results from DCGAN. Figure 2.3 shows synthesized images from models learned from the LSUN bedrooms dataset, which contains more than 3 million training images. The SVHN dataset consists of color images of house numbers collected by Google Street View. The training set consists of 73,257 images and the testing set has 26,032 images. We learn the models in an unsupervised manner. MIT places205 contains images of 205 scene categories. We learn from a single category.

Table 2.1: Inception scores on CIFAR-10.

	Real images	DCGAN	multi-grid method
Inception score	11.237	6.581	6.565

CIFAR-10 includes various object categories and has 50,000 training examples. Figure 2.A.1 shows the synthesized images generated by models learned by the multi-grid method conditional on each category. In this experiment, we run 40 steps of the Langevin dynamics for each grid, and in the final synthesis after learning, we disable the noise term in the Langevin dynamics, which

Figure 2.4: Synthesized images generated by the multi-grid models learned from the CIFAR-10 dataset. Each row illustrates a category, and the multi-grid models are learned conditional on the category. From top to bottom: *airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck*.

slightly improves the synthesis quality. We evaluate the quality of synthesized images quantitatively using the average inception score [SGZ16] in Table 2.1. The multi-grid method gets a comparable inception score as DCGAN as reported in [LW17].

To check the diversity of Langevin dynamics sampling, we synthesize images by initializing the Langevin dynamics from the same 1×1 image. As shown in Figure 2.5, after 90 steps of Langevin dynamics, the sampled images from the same 1×1 image are different from each other.

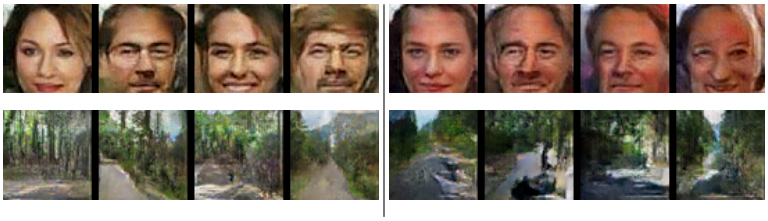


Figure 2.5: Synthesized images by initializing the Langevin dynamics sampling from the same 1×1 image. Each block of 4 images are generated from the same 1×1 image.

2.2.3.2 Unsupervised feature learning for classfication

Table 2.2: Classification error of L2-SVM trained on the features learned from SVHN.

Test error rate with # of labeled images	1,000	2,000	4,000
Persistent CD [Tie08]	45.74	39.47	34.18
One-step CD [Hin02a]	44.38	35.87	30.45
Wasserstein GAN [ACB17b]	43.15	38.00	32.56
Deep directed generative models [KB16]	44.99	34.26	27.44
DCGAN[RMC15]	38.59	32.51	29.37
single-grid method	36.69	30.87	25.60
multi-grid method	**30.23**	**26.54**	**22.83**

To evaluate the features learned by the multi-grid method, we perform a semi-supervised classification experiment by following the same procedure outlined in [RMC15]. That is, we use the multi-grid method as a feature extractor. We first train a multi-grid model on the combination of SVHN training and testing sets in an unsupervised way. Then we train a regularized L2-SVM on the learned representations of grid 3. For a fair comparison, we adopt the discriminator structure of [RMC15] for grid 3, which has 4 convolutional layers of 5×5 filters with 64, 128, 256 and 512 channels respectively. The features from all the convolutional layers are max pooled and concatenated to form a 15,360-dimensional vector. We randomly sample 1000, 2000 and 4000 labeled examples from the training dataset to train the SVM and test on the testing dataset. Within the same setting, we compare the learned features of the multi-grid method with the single-grid method, persistent CD [Tie08], one-step CD [Hin02a], Wasserstein GAN [ACB17b], deep directed generative models [KB16] and DCGAN [RMC15]. Table 2.2 shows the classification results, indicating that the multi-grid method learns strong features.

Table 2.3: Classification error of CNN classifier trained on the features of three grids learned from SVHN.

Test error rate with # of labeled images	1,000	2,000	4,000
DGN [KMR14]	36.02	-	-
Virtual adversarial [MMK15]	24.63	-	-
Auxiliary deep generative model [MSS16]	22.86	-	-
Supervised CNN with the same structure	39.04	22.26	15.24
multi-grid method + CNN classifier	**19.73**	**15.86**	**12.71**

Next, we try to combine the learned features of three grids together. Specifically, we build a two-layer classification CNN on top of the top layer feature maps of three grids. The first layer is a 3×3 stride 1 convolutional layer with 64 channels operated separately on the feature maps of the three grids. Then the outputs from the three grids are concatenated to form a 34,624-dimensional vector. A fully connected layer is added on top of the vector. We train this classifier using 1000, 2000 and 4000 labeled examples that are randomly sampled from the training set. As shown in Table 2.3, our method achieves a test error rate of 19.73% for $1,000$ labeled images. For comparison, we train a classification network from scratch with the same structure (three networks as used in the multi-grid method plus two layers for classification) on the same labeled training data. It has a significantly higher error rate of 39.04% for $1,000$ labeled training images. Our method also outperforms some methods that are specifically designed for semi-supervised learning, such as DGN [KMR14], virtual adversarial [MMK15] and auxiliary deep generative model [MSS16].

2.2.3.3 Image inpainting

We further test our method on image inpainting. In this task, we try to learn the conditional distribution $p_\theta(\mathbf{x}_M|\mathbf{x}_{\bar{M}})$ by our models, where M consists of pixels to be masked, and $\bar{M}$ consists of pixels not to be masked. In the training stage, we randomly place the mask on each training image, but we assume $\mathbf{x}_M$ is observed in training. We follow the same learning and sampling algorithm as in Algorithm 1, except that in the sampling step (i.e., step 4 in Algorithm 1), in each Langevin step, only the masked part of the image is updated, and the unmasked part remains fixed as observed. This is a generalization of the pseudo-likelihood estimation [Bes74], which corresponds to the case where M consists of one pixel. It can also be considered a form of associative memory [Hop82]. After learning $p_\theta(\mathbf{x}_M|\mathbf{x}_{\bar{M}})$ from the fully observed training images, we then use it to inpaint the masked testing images, where the masked parts are not observed.

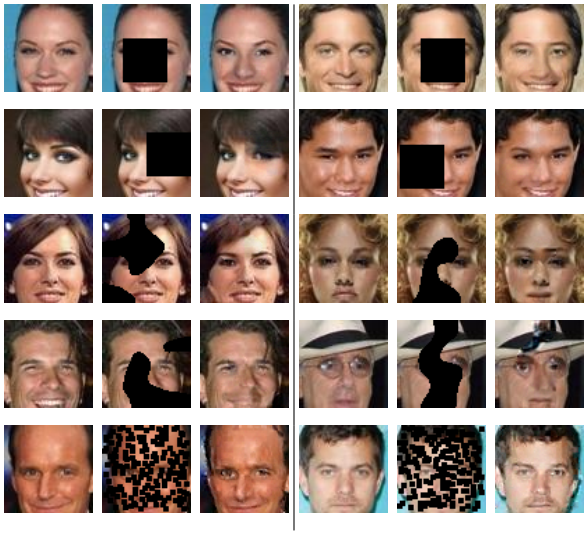


Figure 2.6: Inpainting examples on CelebA dataset. In each block from left to right: the original image, masked input, inpainted image by the multi-grid method.

We use 10,000 face images randomly sampled from CelebA dataset to train the model. We set the mask size at 32×32 for training. During training, the size of the mask is fixed but the position is randomly selected for each training image. Another 1,000 face images are randomly selected

Table 2.4: Quantitative evaluations for three types of masks. Lower values of error are better. Higher values of PSNR are better. PCD, CD1, SG, CE and MG indicate persistent CD, one-step CD, single-grid method, ContextEncoder and multi-grid method, respectively.

	Mask	PCD	CD1	SG	CE	MG
Error	Mask	0.056	0.081	0.066	0.045	**0.042**
	Doodle	0.055	0.078	0.055	0.050	**0.045**
	Pepper	0.069	0.084	0.054	0.060	**0.036**
PSNR	Mask	12.81	12.66	15.97	**17.37**	16.42
	Doodle	12.92	12.68	14.79	15.40	**16.98**
	Pepper	14.93	15.00	15.36	17.04	**19.34**

from CelebA dataset for testing. We find that during the testing, the mask does not need to be restricted to 32×32 square mask. So we test three different shapes of masks: 1) 32×32 square mask, 2) doodle mask with approximately 25% missing pixels, and 3) pepper and salt mask with approximately 60% missing pixels. Figure 2.6 shows some inpainting examples.

We perform quantitative evaluations using two metrics: 1) reconstruction error measured by the per-pixel difference and 2) peak signal-to-noise ratio (PSNR). Metrics are computed between the inpainting results obtained by different methods and the original face images on the masked pixels. We compare with persistent CD, CD1 and the single-grid method. We also compare with the ContextEncoder [PKD16] (CE). We re-train the CE model on 10,000 training face images for a fair comparison. As our tested masks are not in the image center, we use the "inpaintRandom" version of the CE code and randomly place a 32×32 mask in each image during training. The results are shown in Table 2.4. It shows that the multi-grid method works well for the inpainting task.

2.2.4 Conclusion

This section seeks to address the fundamental question of whether we can learn energy-based models purely by themselves without recruiting extra networks such as generator networks. This question is important both conceptually and practically because the energy-based models correspond directly to discriminative classifiers. Being able to learn and sample from such models also provides us a valuable alternative to GAN methods, by relieving us from the concerns with issues such as mismatch between two different classes of models, as well as instability in learning.

To answer the above question, we propose a multi-grid method for learning energy-based models. Our work seeks to facilitate the learning of such models by developing small budget MCMC initialized from a simple distribution for sampling from the learned models. We show that our method can learn realistic models of images and the learned models can be useful for tasks such as image processing and classification.

2.3 Diffusion recovery likelihood

For the previous section, we demonstrate that it is indeed possible to learn deep energy-based models (EBMs) and generate realistic samples on large image datasets with finite budget MCMC. However, learning EBMs on high-resolution image datasets still remains challenging. Moreover, we observe another challenge of learning EBMs. As demonstrated in Figure 2.7, training EBMs with finite budget MCMC results in malformed energy landscapes [NHZ19], **even if the samples from the model look reasonable**. The reason is that the sampling chains with finite budget MCMC is still not converging, leading to biased synthesized samples when estimating gradients for updating the model. For image datasets, the energy potentials learned with non-convergent MCMC even do not have a valid steady-state, in the sense that samples from long-run Markov chains can differ greatly from observed samples, making it difficult to evaluate the learned energy potentials.

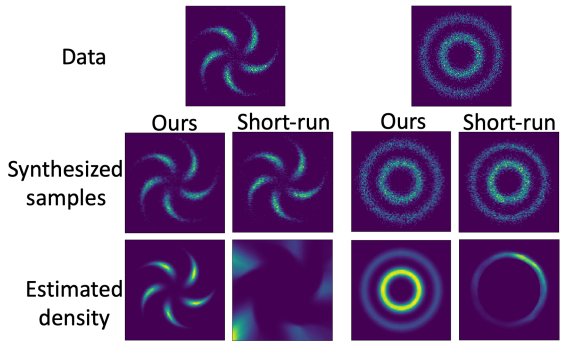


Figure 2.7: Comparison of learning EBMs by diffusion recovery likelihood (Ours) versus marginal likelihood (Short-run).

The root reason for the non-convergent MCMC is still that the energy landscape of the EBMs is too complicated. Hence we would like to answer a question: *can we simplify the energy potentials when doing sampling to make it more MCMC friendly?* This is the motivation of the work in this section.

Our method is inspired by another line of work, diffusion probabilistic models. Originating from [SWM15], such a model is to learn from a diffused version of the data, which are obtained from the original data via a diffusion process that sequentially adds Gaussian white noise. From such diffusion data, one can learn the conditional model of the data at a certain noise level given their noisy versions at the higher noise level of the diffusion process. After learning the sequence of conditional models that invert the diffusion process, one can then generate synthesized images from Gaussian white noise images by ancestral sampling. Building on [SWM15], [HJA20] further developed the method, obtaining strong image synthesis results.

Inspired by [SWM15] and [HJA20], we propose a *diffusion recovery likelihood* method to tackle the challenge of training EBMs directly on a dataset by instead learning a sequence of EBMs for the *marginal* distributions of the diffusion process [GSP20]. The sequence of marginal EBMs is learned with recovery likelihoods that are defined as the conditional distributions that invert the diffusion process. Compared to standard maximum likelihood estimation (MLE) of EBMs, learning marginal EBMs by diffusion recovery likelihood only requires sampling from the conditional

distributions, which is much easier than sampling from the marginal distributions. After learning the marginal EBMs, we can generate synthesized images by a sequence of conditional samples initialized from the Gaussian white noise distribution. Unlike [HJA20] that approximates the reverse process by normal distributions, in our case the conditional distributions are derived from the marginal EBMs, which are more flexible. The framework of recovery likelihood was originally proposed in [BYA13]. In our work, we adapt it to learning the sequence of marginal EBMs from the diffusion data.

We would like to emphasize that our goal is still to learn *marginal* EBMs at different noise levels, and we use sequential conditional distributions to ease the sampling, which is similar to simulated annealing sampling.

Our work is also related to the denoising score matching method of [Vin11], which was further developed by [SE19, SE20] for learning from diffusion data. The training objective used for diffusion probabilistic models is a weighted version of the denoising score matching objective, as revealed by [HJA20]. These methods learn the score functions (the gradients of the energy functions) directly, instead of using the gradients of learned energy functions as in EBMs. On the other hand, [SMS18] parametrizes the score function as the gradient of an MLP energy function, and [SH19] further unifies denoising score matching and neural empirical Bayes.

We demonstrate the efficacy of diffusion recovery likelihood on CIFAR-10, CelebA and LSUN datasets. The generated samples are of high fidelity and comparable to GAN-based methods. On CIFAR-10, we achieve FID 9.58 and inception score 8.30, exceeding existing methods of learning explicit EBMs to a large extent. We also demonstrate that diffusion recovery likelihood outperforms denoising score matching from diffusion data if we naively take the gradients of explicit energy functions as the score functions. More interestingly, by using a thousand diffusion time steps, we demonstrate that even very long MCMC chains from the sequence of conditional distributions produce samples that represent realistic images. With the faithful long-run MCMC samples from the conditional distributions, we can accurately estimate the marginal partition function at zero noise level by importance sampling, and thus evaluate the normalized density of data under the

Figure 2.8: Generated samples on LSUN 128^2 church_outdoor (*left*), LSUN 128^2 bedroom (*center*) and CelebA 64^2 (*right*).

EBM.

2.3.1 From marginal to conditional

Given the difficulty of sampling from the marginal density $p_\theta(\mathbf{x})$, we use the recovery likelihood defined by the density of the observed sample conditional on a noisy sample perturbed by isotropic Gaussian noise. Specifically, let $\tilde{\mathbf{x}} = \mathbf{x} + \sigma\epsilon$ be the noisy observation of $\mathbf{x}$, where $\epsilon \sim \mathcal{N}(0, \boldsymbol{I})$. Suppose $p_\theta(\mathbf{x})$ is defined by the EBM in Eq. (2.1), then the conditional EBM can be derived as

$$p_\theta(\mathbf{x}|\tilde{\mathbf{x}}) = \frac{1}{\tilde{Z}_\theta(\tilde{\mathbf{x}})} \exp\left(f_\theta(\mathbf{x}) - \frac{1}{2\sigma^2}\|\tilde{\mathbf{x}} - \mathbf{x}\|^2\right), \tag{2.11}$$

where $\tilde{Z}_\theta(\tilde{\mathbf{x}}) = \int \exp\left(f_\theta(\mathbf{x}) - \frac{1}{2\sigma^2}\|\tilde{\mathbf{x}} - \mathbf{x}\|^2\right) d\mathbf{x}$ is the partition function of this conditional EBM. Compared to $p_\theta(\mathbf{x})$ (Eq. (2.1)), the extra quadratic term $\frac{1}{2\sigma^2}\|\tilde{\mathbf{x}} - \mathbf{x}\|^2$ in $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ constrains the energy landscape to be localized around $\tilde{\mathbf{x}}$, making the latter less multi-modal and easier to sample from. As we will show later, when σ is small, $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ is approximately a single mode Gaussian

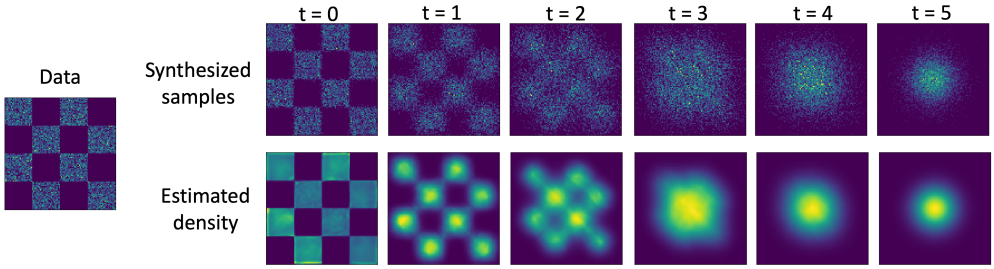


Figure 2.9: Illustration of diffusion recovery likelihood on 2D checkerboard example. *Top*: progressively generated samples. *Bottom*: estimated marginal densities.

distribution, which greatly reduces the burden of MCMC.

A more general formulation is $\tilde{\mathbf{x}} = a\mathbf{x} + \sigma\epsilon$, where a is a positive constant. In that case, we can let $\mathbf{y} = a\mathbf{x}$, and treat $\mathbf{y}$ as the observed sample. Assume $p_\theta(\mathbf{y}) = \frac{1}{Z_\theta}\exp(f_\theta(\mathbf{y}))$, then by *change of variable*, the density function of $\mathbf{x}$ can be derived as $g_\theta(\mathbf{x}) = ap_\theta(a\mathbf{x})$.

2.3.2 Maximizing recovery likelihood

With the conditional EBM, assume we have observed samples $\mathbf{x}_i \sim p_{\text{data}}(\mathbf{x})$ and the corresponding perturbed samples $\tilde{\mathbf{x}}_i = \mathbf{x}_i + \sigma\epsilon_i$ for $i = 1, ..., n$. We define the *recovery log-likelihood function* as

$$\mathscr{J}(\theta) = \frac{1}{n}\sum_{i=1}^{n} \log p_\theta(\mathbf{x}_i|\tilde{\mathbf{x}}_i). \tag{2.12}$$

The term *recovery* indicates that we attempt to recover the clean sample $\mathbf{x}_i$ from the noisy sample $\tilde{\mathbf{x}}_i$. Thus, instead of maximizing $\mathscr{L}(\theta)$ in Eq. (2.2), we can maximize $\mathscr{J}(\theta)$, whose distributions are easier to sample from. Specifically, we generate synthesized samples by K steps of Langevin dynamics that iterates

$$\mathbf{x}^{\tau+1} = \mathbf{x}^\tau + \frac{\delta^2}{2}(\nabla_{\mathbf{x}} f_\theta(\mathbf{x}^\tau) + \frac{1}{\sigma^2}(\tilde{\mathbf{x}} - \mathbf{x}^\tau)) + \delta\epsilon^\tau. \tag{2.13}$$

The model is then updated following the same learning gradients as MLE (Eq. (2.3)), because the quadratic term $-\frac{1}{2\sigma^2}\|\tilde{\mathbf{x}} - \mathbf{x}\|^2$ is not related to θ. Following the classical analysis of MLE, we can show that the point estimate given by maximizing recovery likelihood is an unbiased estimator of

the true parameters, which means that given enough data, a rich enough model and exact synthesis, maximizing the recovery likelihood learns θ such that $p_{\text{data}}(\mathbf{x}) = p_\theta(\mathbf{x})$.

2.3.2.1 Normal Approximation to Conditional

When the variance of perturbed noise σ^2 is small, $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ can be approximated by a normal distribution via a first order Taylor expansion at $\tilde{\mathbf{x}}$. Specifically, the negative conditional energy is

$$-\mathscr{E}_\theta(\mathbf{x}|\tilde{\mathbf{x}}) = f_\theta(\mathbf{x}) - \frac{1}{2\sigma^2}\|\tilde{\mathbf{x}} - \mathbf{x}\|^2 \tag{2.14}$$

$$\doteq f_\theta(\tilde{\mathbf{x}}) + \langle \nabla_{\mathbf{x}} f_\theta(\tilde{\mathbf{x}}), \mathbf{x} - \tilde{\mathbf{x}} \rangle - \frac{1}{2\sigma^2}\|\tilde{\mathbf{x}} - \mathbf{x}\|^2 \tag{2.15}$$

$$= -\frac{1}{2\sigma^2}\left[\|\mathbf{x} - (\tilde{\mathbf{x}} + \sigma^2 \nabla_{\mathbf{x}} f_\theta(\tilde{\mathbf{x}}))\|^2\right] + c, \tag{2.16}$$

where c include terms irrelevant of $\mathbf{x}$ In the above approximation, we do not perform second-order Taylor expansion because σ^2 is small, and $\|\tilde{\mathbf{x}} - \mathbf{x}\|^2/2\sigma^2$ will dominate all the second-order terms from Taylor expansion. Thus we can approximate $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ by a Gaussian approximation $\widetilde{p}_\theta(\mathbf{x}|\tilde{\mathbf{x}})$:

$$\widetilde{p}_\theta(\mathbf{x}|\tilde{\mathbf{x}}) = \mathscr{N}\left(\mathbf{x}; \tilde{\mathbf{x}} + \sigma^2 \nabla_{\mathbf{x}} f_\theta(\tilde{\mathbf{x}}), \sigma^2\right). \tag{2.17}$$

We can sample from this distribution using:

$$\mathbf{x}_{\text{gen}} = \tilde{\mathbf{x}} + \sigma^2 \nabla_{\mathbf{x}} f_\theta(\tilde{\mathbf{x}}) + \sigma\epsilon, \tag{2.18}$$

where $\epsilon \sim \mathscr{N}(0, I)$. This resembles a single step of Langevin dynamics, except that $\sigma\epsilon$ is replaced by $\sqrt{2}\sigma\epsilon$ in Langevin dynamics. This normal approximation has two traits: (1) it verifies the fact that the conditional density $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ can be generally easier to sample from when σ is small; (2) it provides hints of choosing the step size of Langevin dynamics, as discussed in section 2.3.4.

2.3.3 Connection to variational inference and score matching

The normal approximation to the conditional distribution leads to a natural connection to diffusion probabilistic models [SWM15, HJA20] and denoising score matching [Vin11, SE19, SE20,

SMS18, SH19]. Specifically, in diffusion probabilistic models, instead of modeling $p_\theta(x)$ as an energy-based model, it recruits variational inference and directly models the conditional density as

$$p_\theta(\mathbf{x}|\tilde{\mathbf{x}}) \sim \mathcal{N}\left(\tilde{\mathbf{x}} + \sigma^2 s_\theta(\tilde{\mathbf{x}}), \sigma^2\right), \tag{2.19}$$

which is in agreement with the normal approximation (Eq. (2.17)), with $s_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} f_\theta(\mathbf{x})$. On the other hand, the training objective of denoising score matching is to minimize

$$\frac{1}{2\sigma^2}\mathrm{E}_{p(\mathbf{x},\tilde{\mathbf{x}})}[\|\mathbf{x} - (\tilde{\mathbf{x}} + \sigma^2 s_\theta(\tilde{\mathbf{x}}))\|^2], \tag{2.20}$$

where $s_\theta(\mathbf{x})$ is the score of the density of $\tilde{\mathbf{x}}$. This objective is in agreement with the objective of maximizing log-likelihood of the normal approximation (Eq. (2.16)), except that for normal approximation, $\nabla_{\mathbf{x}} f_\theta(\cdot)$ is the score of density of $\mathbf{x}$, instead of $\tilde{\mathbf{x}}$. However, the difference between the scores of density of $\mathbf{x}$ and $\tilde{\mathbf{x}}$ is of $O(\sigma^2)$, which is negligible when σ is sufficiently small. We can further show that the learning gradient of maximizing log-likelihood of the normal approximation is approximately the same as the learning gradient of maximizing the original recovery log-likelihood with one step of Langevin dynamics. It indicates that the training process of maximizing recovery likelihood agrees with the one of diffusion probabilistic models and denoising score matching when σ is small.

As the normal approximation is accurate only when σ is small, it requires many time steps in the diffusion process for this approximation to work well, which is also reported in [HJA20] and [SE20]. In contrast, the diffusion recovery likelihood framework can be more flexible in choosing the number of time steps and the magnitude of σ.

2.3.4 Diffusion recovery likelihood

As we discuss, sampling from $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ becomes simple only when σ is small. In the extreme case when $\sigma \to \infty$, $p_\theta(\mathbf{x}|\tilde{\mathbf{x}})$ converges to the marginal distribution $p_\theta(\mathbf{x})$, which is again highly multimodal and difficult to sample from. To keep σ small and meanwhile equip the model with the ability to generate new samples initialized from white noise, inspired by [SWM15] and [HJA20],

we propose to learn a sequence of recovery likelihoods, on gradually perturbed observed data based on a diffusion process. Specifically, assume a sequence of perturbed observations $\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_T$ such that

$$\mathbf{x}_0 \sim p_{\text{data}}(\mathbf{x}); \;\; \mathbf{x}_{t+1} = \sqrt{1-\sigma_{t+1}^2}\mathbf{x}_t + \sigma_{t+1}\boldsymbol{\epsilon}_{t+1}, \;\; t = 0, 1, ...T-1. \tag{2.21}$$

The scaling factor $\sqrt{1-\sigma_{t+1}^2}$ ensures that the sequence is a spherical interpolation between the observed sample and Gaussian white noise. Let $\mathbf{y}_t = \sqrt{1-\sigma_{t+1}^2}\mathbf{x}_t$, and we assume a sequence of conditional EBMs

$$p_\theta(\mathbf{y}_t|\mathbf{x}_{t+1}) = \frac{1}{\tilde{Z}_{\theta,t}(\mathbf{x}_{t+1})} \exp\left(f_\theta(\mathbf{y}_t, t) - \frac{1}{2\sigma_{t+1}^2}\|\mathbf{x}_{t+1} - \mathbf{y}_t\|^2\right), \;\; t = 0, 1, ..., T-1, \tag{2.22}$$

where $f_\theta(\mathbf{y}_t, t)$ is defined by a neural network conditioned on t.

We follow the learning algorithm in Section 2.3.2. A question is how to determine the step size schedule δ_t of Langevin dynamics. Inspired by the sampling procedure of the normal approximation (Eq. (2.18)), we set the step size $\delta_t = b\sigma_t$, where $b < 1$ is a tuned hyperparameter. This schedule turns out to work well in practice. Thus the K steps of Langevin dynamics iterates

$$\mathbf{y}_t^{\tau+1} = \mathbf{y}_t^\tau + \frac{b^2\sigma_t^2}{2}(\nabla_{\mathbf{y}} f_\theta(\mathbf{y}_t^\tau, t) + \frac{1}{\sigma_t^2}(\mathbf{x}_{t+1} - \mathbf{y}_t^\tau)) + b\sigma_t\boldsymbol{\epsilon}^\tau. \tag{2.23}$$

Algorithm 2 summarizes the training procedure. After training, we initialize the MCMC sampling from Gaussian white noise, and the synthesized sample at each time step serves to initialize the MCMC that samples from the model of the previous time step. See algorithm 3. To show the efficacy of our method, Figures 2.9 and 2.7 display several 2D toy examples learned by diffusion recovery likelihood.

Algorithm 2 Training

repeat

Sample $t \sim \text{Unif}(\{0, ..., T-1\})$.

Sample pairs $(\mathbf{y}_t, \mathbf{x}_{t+1})$.

Set synthesized sample $\mathbf{y}_t^- = \mathbf{x}_{t+1}$.

for $\tau \leftarrow 1$ to K **do**

Update $\mathbf{y}_t^-$ according to Eq. (2.23).

end for

Update θ following the gradients

$\frac{\partial}{\partial \theta} f_\theta(\mathbf{y}_t, t) - \frac{\partial}{\partial \theta} f_\theta(\mathbf{y}_t^-, t)$.

until converged.

Algorithm 3 Progressive sampling

Sample $\mathbf{x}_T \sim \mathcal{N}(0, \boldsymbol{I})$.

for $t \leftarrow T-1$ to 0 **do**

$\mathbf{y}_t = \mathbf{x}_{t+1}$.

for $\tau \leftarrow 1$ to K **do**

Update $\mathbf{y}_t$ according to Eq. (2.23).

end for

$\mathbf{x}_t = \mathbf{y}_t / \sqrt{1 - \sigma_{t+1}^2}$.

end for

return $\mathbf{x}_0$.

2.3.5 Experiments

To show that diffusion recovery likelihood is flexible for diffusion process of various magnitudes of noise, we test the method under two settings: (1) $T = 6$, with $K = 30$ steps of Langevin dynamic per time step; (2) $T = 1000$, with sampling from normal approximation. (2) resembles the noise schedule of [HJA20] and the magnitude of noise added at each time step is much smaller compared to (1). For both settings, we set σ_t^2 to increase linearly. The network structure of $f_\theta(x, t)$ is based on Wide ResNet [ZK16] and we remove weight normalization. t is encoded by Transformer sinusoidal position embedding as in [HJA20]. For (1), we find that including another scaling factor c_t to the step size δ_t helps. Henceforth we simply refer to the two settings as *T6* and *T1k*.

2.3.5.1 Image generation

Figures 2.8 and 2.10 display uncurated samples generated from learned models on CIFAR-10, CelebA 64×64, LSUN 64×64 and 128×128 datasets under *T6* setting. The samples are of high fidelity and comparable to GAN-based methods. See chapter appendix for more generated sam-

ples. Tables 2.5 and 2.7 summarize the quantitative evaluations on CIFAR-10 and CelebA datasets, in terms of Frechet Inception Distance (FID) [HRU17] and inception scores [SGZ16]. On CIFAR-10, our model achieves a FID score of 9.58 and an inception score of 8.30, which outperforms existing methods of learning explicit energy-based models to a large extent, and is superior to a majority of GAN-based methods. On CelebA, our model obtains results comparable with the state-of-the-art GAN-based methods, and outperforms score-based methods [SE19, SE20]. Note that the score-based methods [SE19, SE20] and diffusion probabilistic models [HJA20] directly parametrize and learn the score of data distribution, whereas our goal is to learn explicit energy-based models.

Figure 2.10: Generated samples on unconditional CIFAR-10 (*left*) and LSUN 64^2 church_outdoor (*center*) and LSUN 64^2 bedroom (*right*).

Table 2.5: FID and inception scores on CIFAR-10.

Model	FID↓	Inception↑
GAN-based		
WGAN-GP [GAA17]	36.4	7.86 ± .07
SNGAN [MKK18]	21.7	8.22 ± .05
SNGAN-DDLS [CZS20]	15.42	9.09 ± .10
StyleGAN2-ADA [KAH20]	3.26	**9.74** ± .05
Score-based		
NCSN [SE19]	25.32	8.87 ± .12
NCSN-v2 [SE20]	10.87	8.40 ± .07
DDPM [HJA20]	**3.17**	9.46 ± .11
Explicit EBM-conditional		
CoopNets [XZF19]	-	7.30
EBM-IG [DM19]	37.9	8.30
JEM [GWJ19]	38.4	8.76
Explicit EBM		
Muli-grid [GLZ18]	40.01	6.56
CoopNets [XLG16]	33.61	6.55
EBM-SR [NHZ19]	-	6.21
EBM-IG [DM19]	38.2	6.78
Ours (*T6*)	**9.58**	**8.30** ± .11

Table 2.6: Ablation of training objectives, time steps T and sampling steps K on CIFAR-10. $K = 0$ indicates that we sample from the normal approximation.

Setting / Objective	FID↓	Inception↑
T = 1, K = 180	32.12	6.72 ± 0.12
T = 1000, K = 0	22.58	7.71 ± 0.08
T = 1000, K = 0 (DSM)	21.76	7.76 ± 0.11
T = 6, K = 10	-	-
T = 6, K = 30	9.58	8.30 ± 0.11
T = 6, K = 50	**9.36**	**8.46** ± 0.13

Table 2.7: FID scores on CelebA 64^2.

Model	FID↓
QA-GAN [PC19]	6.42
COCO-GAN [LCC19]	**4.0**
NVAE [VK20]	14.74
NCSN [SE19]	25.30
NCSN-v2 [SE20]	10.23
EBM-SR [NHZ19]	23.02
EBM-Triangle [HNZ20]	24.70
Ours (*T6*)	5.98

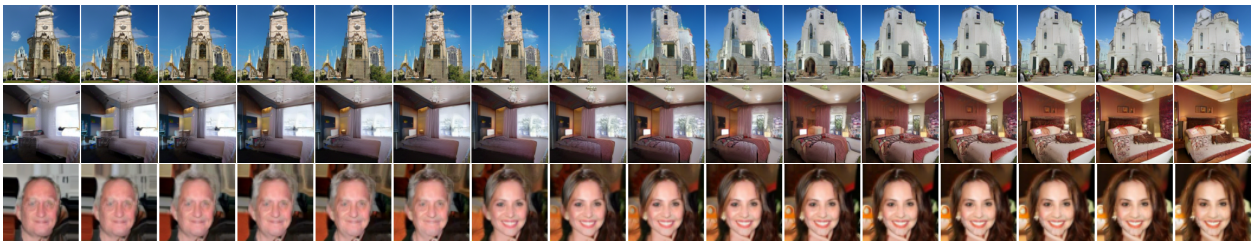


Figure 2.11: Interpolation results between the leftmost and rightmost generated samples. For *top* to *bottom*: LSUN church_outdoor 128^2, LSUN bedroom 128^2 and CelebA 64^2.

Table 2.8: Test bits per dimension on CIFAR-10. † indicates that we estimate the bit per dimension with the approximated log partition function instead of analytically computing it. See section 2.3.5.2.

Model	BPD↓
DDPM [HJA20]	3.70
Glow [KD18]	3.35
Flow++ [HCS19]	3.08
GPixelCNN [OKE16]	3.03
Sparse Transformer [CGR19]	2.80
DistAug [JCC20]	**2.56**
Ours† (*T1k*)	3.18

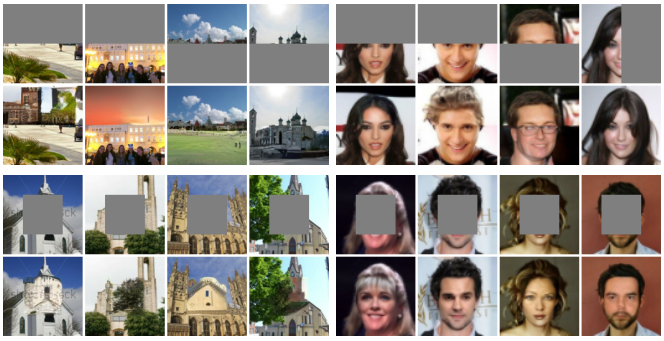


Figure 2.12: Image inpainting on LSUN church_outdoor 128^2 (*left*) and CelebA 64^2 (*right*). With each block, the top row are mask images while the bottom row are inpainted images.

Interpolation. As shown in Figure 2.11, our model is capable of smooth interpolation between two generated samples. Specifically, for two samples $\mathbf{x}_0^{(0)}$ and $\mathbf{x}_0^{(1)}$, we do a sphere interpolation between the initial white noise images $\mathbf{x}_T^{(0)}$ and $\mathbf{x}_T^{(1)}$ and the noise terms of Langevin dynamics $\boldsymbol{\epsilon}_{t,\tau}^{(0)}$ and $\boldsymbol{\epsilon}_{t,\tau}^{(1)}$ for every sampling step at every time step.

Image inpainting. A promising application of energy-based models is to use the learned model as a prior model for image processing, such as image inpainting, denoising and super-resolution [GLZ18, DM19, SE19]. In Figure 2.12, we demonstrate that the learned models by maximizing recovery likelihoods are capable of realistic and semantically meaningful image inpainting. Specifically, given a masked image and the corresponding mask, we first obtain a sequence of perturbed masked images at different noise levels. The inpainting can be easily achieved by running Langevin dynamics progressively on the masked pixels while keeping the observed pixels fixed at decreasingly lower noise levels.

Ablation study. Table 2.6 summarizes the results of ablation study on CIFAR-10. We investigate the effect of changing the numbers of time steps T and sampling steps K. First, to show that it is beneficial to learn by diffusion recovery likelihood, we compare against a baseline approach ($T = 1, K = 180$) where we use only one-time step, so that the recovery likelihood becomes a marginal likelihood. The approach is adopted by [NHZ19] and [DM19]. For a fair comparison, we equip the baseline method with the same budget of MCMC sampling as our *T6* setting (i.e., 180 sampling steps). Our method outperforms this baseline method by a large margin. Also, the models are trained more efficiently as the number of sampling steps per iteration is reduced and amortized by time steps.

Next, we report the sample quality of setting *T1k*. We test two training objectives for this setting: (1) maximizing recovery likelihoods (T = 1000, K = 0) and (2) maximizing the approximated normal distributions (T=1000, K=0 (DSM)). As mentioned in section 2.3.3, (2) is equivalent to the training objectives of denoising score matching [SE19, SE20] and diffusion probabilistic model [HJA20], except that the score functions are taken as the gradients of explicit energy functions. In practice, for a direct comparison, (2) follows the same implementation as in [HJA20], except that the score function is parameterized as the gradients of the explicit energy function used in our method. (1) and (2) achieve similar sample quality in terms of quantitative metrics, where (2) results in a slightly better FID score yet a slightly worse inception score. This verifies the fact

that the training objectives of (1) and (2) are consistent. Both (1) and (2) performs worse than setting T6. A possible explanation is that the sampling error may accumulate with many time steps, so that a more flexible schedule of time steps accompanied by a certain amount of sampling steps is preferred.

Last, we examine the influence of varying the number of sampling steps while fixing the number of time steps. The training becomes unstable when the number of sampling steps is not enough ($T = 6, K = 10$), and more sampling steps lead to better sample quality. However, since $K = 50$ does not gain significant improvement versus $K = 30$, yet of much higher computational cost, we keep $K = 30$ for image generation on all datasets.

2.3.5.2 Long-run chain analysis

Besides achieving high-quality generation, a perhaps equally important aspect of learning EBMs is to obtain a faithful energy potential. A principal way to check the validity of the learned potential is to perform long-run sampling chains and see if the samples still remain realistic. However, as pointed out in [NHH19], almost all existing methods of learning EBMs fail in getting realistic long-run chain samples. In this subsection, we demonstrate that by composing a thousand diffusion time steps (*T1k* setting), we can form steady long-run MCMC chains for the conditional distributions.

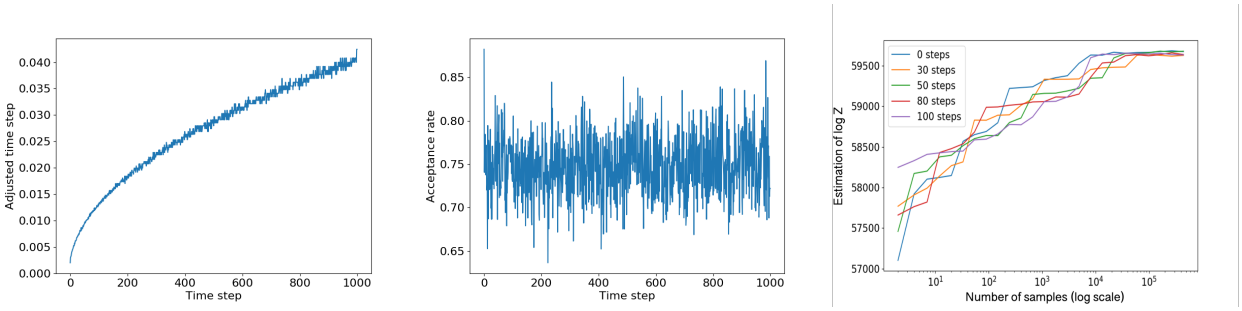


Figure 2.13: *Left*: Adjusted step size of HMC over time step. *Center*: Acceptance rate over time step. *Right*: Estimated log partition function over number of samples with different number of sampling steps per time step. The x axis is plotted in log scale.

First, we prepare a faithful sampler for conducting long-run sampling. Specifically, after training the model under $T1k$ setting by maximizing diffusion recovery likelihood, for each time step, we first sample from the normal approximation and count it as one sampling step, and then use Hamiltonian Monte Carlo (HMC) [Nea11] with 2 leapfrog steps to perform the consecutive sampling steps. To obtain a reasonable schedule of sampling step size, for each time step we adaptively adjust the step size of HMC to make the average acceptance rate range in $[0.6, 0.9]$, which is computed over 1000 chains for 100 steps. Figure 2.13 displays the adjusted step size (*left*) and acceptance rate (*center*) over time step. The adjusted step size increases logarithmically. With this step size schedule, we generate long-run chains from the learned sequence of conditional distributions. As shown in Figure 2.14, images remain realistic for even $100k$ sampling steps in total (i.e., 100 sampling steps per time step), resulting in FID 24.89. This score is close to the one computed on samples generated by $1k$ steps (i.e., sampled from normal approximation), which is 25.12. As a further check, we recruit a No-U-Turn Sampler [HG14] with the same step size schedule as HMC to perform long-run sampling, where the samples also remain realistic.

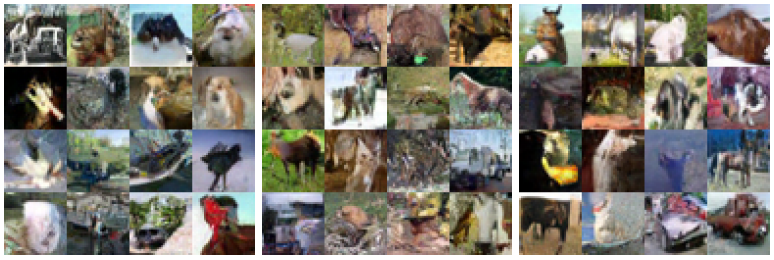


Figure 2.14: Long-run chain samples from model-*T1k* with different total amount of HMC steps. From *left* to *right*: $1k$ steps, $10k$ steps and $100k$ steps.

More interestingly, given the faithful long-run MCMC samples from the *conditional* distributions, we can estimate the log ratio of the partition functions of the *marginal* distributions, and further estimate the partition function of $p_\theta(\mathbf{y}_0)$. The strategy is based on annealed importance sampling [Nea01b]. The right subfigure of Figure 2.13 depicts the estimated log partition function of $p_\theta(\mathbf{y}_0)$ over the number of MCMC samples used. To verify the estimation strategy and again check the long-run chain samples, we conduct multiple runs using samples generated with different

numbers of HMC steps and display the estimation curves. All the curves saturate to values close to each other at the end, indicating the stability of long-run chain samples and the effectiveness of the estimation strategy. With the estimated partition function, by *change of variable*, we can estimate the normalized density of data as $g_\theta(\mathbf{x}_0) = \sqrt{1-\sigma_1^2}\, p_\theta(\sqrt{1-\sigma_1^2}\mathbf{x}_0)$. We report test bits per dimension on CIFAR-10 in Table 2.8. Note that the result should be taken with a grain of salt, because the partition function is estimated by samples and it is a stochastic lower bound of the true value, that will converge to the true value when the number of samples grows large.

2.3.6 Conclusion

In this section, we propose to learn EBMs by diffusion recovery likelihood, a variant of MLE applied to diffusion processes. We achieve high-quality image synthesis, and with a thousand noise levels, we obtain faithful long-run MCMC samples that indicate the validity of the learned energy potentials. Since this method can learn EBMs efficiently with a small budget of MCMC, we are also interested in scaling it up to higher resolution images and investigating this method in other data modalities in the future.

CHAPTER APPENDIX

2.A Additional uncurated samples

Figures 2.A.1, 2.A.2, 2.A.3, 2.A.4, 2.A.5 and 2.A.6 show uncurated samples from the learned models by diffusion recovery likelihood under *T6* setting on CIFAR-10, CelebA 64^2, LSUN church_outdoor 128^2, LSUN bedroom 128^2, LSUN church_outdoor 64^2 and LSUN bedroom 64^2 datasets.

Figure 2.A.1: Generated samples on CIFAR-10.

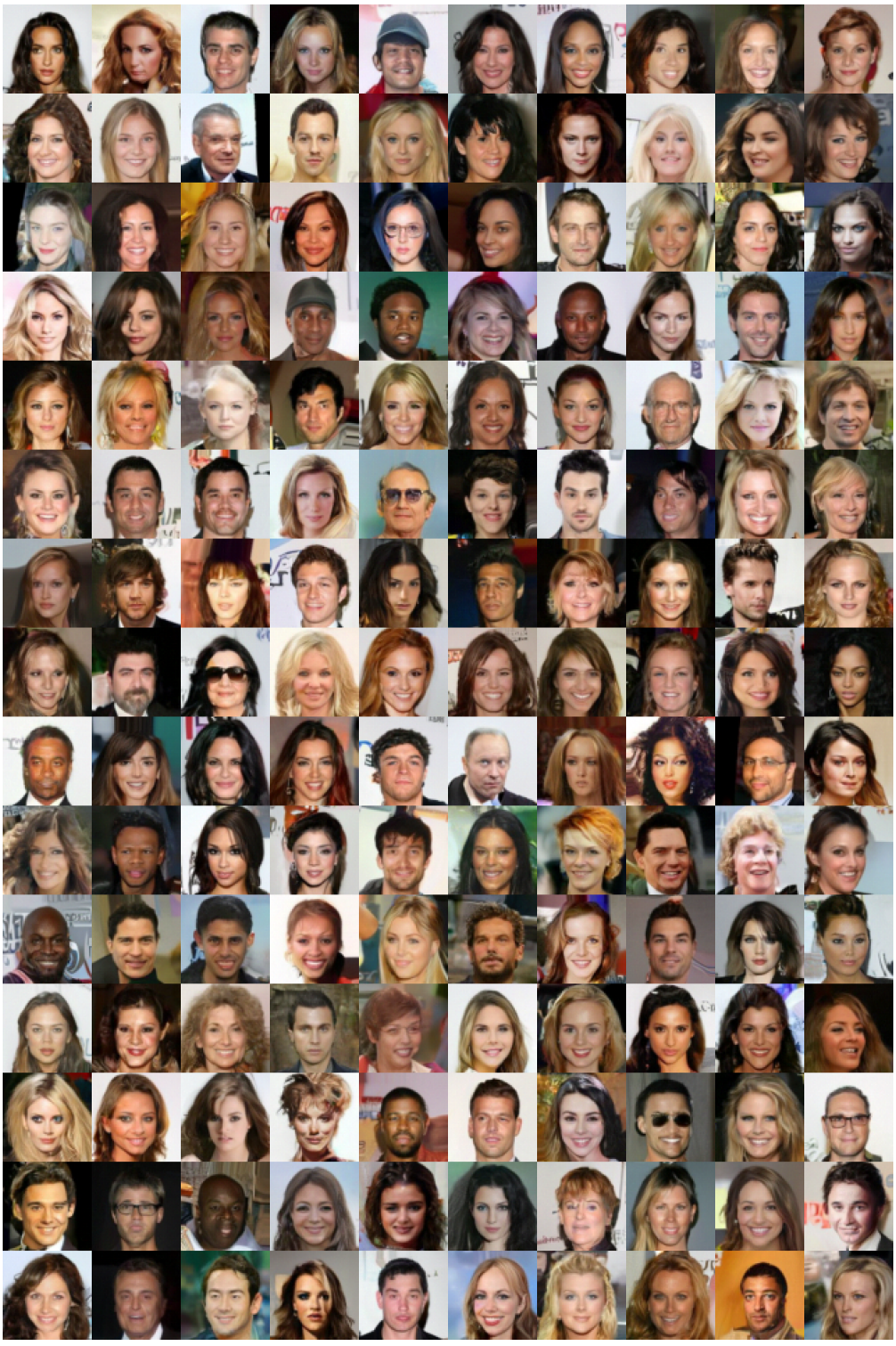


Figure 2.A.2: Generated samples on CelebA 64×64.

Figure 2.A.3: Generated samples on LSUN church_outdoor 128 $\times$ 128. FID=9.76

Figure 2.A.4: Generated samples on LSUN bedroom 128×128. FID=11.27

Figure 2.A.5: Generated samples on LSUN church_outdoor 64×64. FID=7.02

Figure 2.A.6: Generated samples on LSUN bedroom 64×64. FID=8.98

CHAPTER 3

Dynamic generator model

3.1 Introduction

The model

Most physical phenomena in our visual environments are spatial-temporal processes. In this chapter, we study a generative model for spatial-temporal processes such as dynamic textures and action sequences in video data[1]. The model is a non-linear generalization of the linear state-space model proposed by [DCW03] for dynamic textures. The model of [DCW03] is a hidden Markov model, which consists of a transition model that governs the transition probability distribution in the state space, and an emission model that generates the observed signal by a mapping from the state space to the signal space. In the model of [DCW03], the transition model is an auto-regressive model in the d-dimensional state space, and the emission model is a linear mapping from the d-dimensional state vector to the D-dimensional image. In [DCW03], the emission model is learned by treating all the frames of the input video sequence as independent observations, and the linear mapping is learned by principal component analysis via singular value decomposition. This reduces the D-dimensional image to a d-dimensional state vector. The transition model is then learned on the sequence of d-dimensional state vectors by a first-order linear autoregressive model.

Given the high approximation capacity of the modern deep neural networks, it is natural to replace the linear structures in the transition and emission models of [DCW03] by the neural networks. This leads to the following dynamic generator model that has the following two com-

[1]The main contributions in this chapter were first described in [XGZ19, XGZ20].

ponents. (1) The emission model, which is a generator network that maps the d-dimensional state vector to the D-dimensional image via a top-down deconvolution network. (2) The transition model, where the state vector of the next frame is obtained by a non-linear transformation of the state vector of the current frame as well as an independent Gaussian white noise vector that provides randomness in the transition. The non-linear transformation can be parametrized by a feedforward neural network or multi-layer perceptron. In this model, the latent random vectors that generate the observed data are the independent Gaussian noise vectors, also called innovation vectors in [DCW03]. The state vectors and the images can be deterministically computed from these noise vectors.

Learning algorithm

Such dynamic models have been studied in the computer vision literature recently, notably [TLY17]. However, the models are usually trained by the generative adversarial networks (GAN) [GPM14] with an extra discriminator network that seeks to distinguish between the observed data and the synthesized data generated by the dynamic model. Such a model may also be learned by variational auto-encoder (VAE) [KW13] together with an inference model that infers the sequence of noise vectors from the sequence of observed frames. Such an inference model may require a sophisticated design.

In this chapter, we show that it is possible to learn the model on its own using an alternating back-propagation through time (ABPTT) algorithm, without recruiting a separate discriminator model or an inference model. The ABPTT algorithm iterates the following two steps. (1) Inferential back-propagation through time, which samples the sequence of noise vectors given the observed video sequence using the Langevin dynamics, where the gradient of the log posterior distribution of the noise vectors can be calculated by back-propagation through time. (2) Learning back-propagation through time, which updates the parameters of the transition model and the emission model by gradient ascent, where the gradient of the log-likelihood with respect to the model parameters can again be calculated by back-propagation through time.

The alternating back-propagation (ABP) algorithm was originally proposed for the static generator network [HLZ17]. We show that it can be generalized to the dynamic generator model. In our experiments, we show that we can learn the dynamic generator models using the ABPTT algorithm for dynamic textures and action sequences.

Two advantages of the ABPTT algorithm for the dynamic generator models are convenience and efficiency. The algorithm can be easily implemented without designing an extra network. Because it only involves back-propagation through time with respect to a single model, the computation is very efficient.

Disentangled representations

On the other hand, a fundamental challenge in understanding dynamic patterns is learning disentangled representations to separate the underlying factorial components of the observations without supervision [BCV13, MZZ16]. For example, given a video dataset of human facial expressions, a disentangled representation can include the face's appearance attributes (such as color, identity, and gender), the trackable motion attributes (such as movements of eyes, lip, and noise), and the intrackable motion attributes (such as illumination change). A disentangled representation of dynamic patterns is useful in manipulable video generation and calculating video statistics. The goal of this chapter is not only to provide a representational model for video generation, but also for video understanding by disentangling appearance, trackable and intrackable motions in an unsupervised manner. In terms of the underlying physical processes and the perception of the dynamic patterns, they are largely about motions, i.e., movements of pixels or constituent elements, and it is desirable to have a model that is based explicitly on the motions.

To this end, we further generalize the dynamic generator model to a motion-based dynamic generator model. Specifically, in the emission model, we let the hidden state generate the displacement field, which warps the trackable component in the previous image frame to generate the next frame while adding a simultaneously emitted residual image to account for the change that cannot

be explained by the deformation. Thus, each image frame is decomposed into a trackable component that is obtained by warping the previous frame and an intrackable component in the form of the simultaneously generated residual image.

Related work

The proposed learning method is related to the following themes of research.

Dynamic textures. The original dynamic texture model [DCW03] is linear in both the transition model and the emission model. Our work is concerned with a dynamic model with non-linear transition and emission models. See also [TBD18] and references therein for some recent work on dynamic textures.

Chaos modeling. The non-linear dynamic generator model has been used to approximate chaos in a recent work [PLH17]. In the chaos model, the innovation vectors are given as inputs, and the model is deterministic. In contrast, in the model studied in this chapter, the innovation vectors are independent Gaussian noise vectors, and the model is stochastic.

GAN and VAE. The dynamic generator model can also be learned by GAN or VAE. See [TLY17] [SMS17] and [VPT16] for recent video generative models based on GAN. However, GAN does not infer the latent noise vectors. In VAE [KW13], one needs to design an inference model for the sequence of noise vectors, which is a non-trivial task due to the complex dependency structure. Our method does not require an extra model such as a discriminator in GAN or an inference model in VAE.

Models based on spatial-temporal filters or kernels. The patterns in the video data can also be modeled by spatial-temporal filters by treating the data as 3D (2 spatial dimensions and 1 temporal dimension), such as a 3D energy-based model [XZW17b] where the energy function is parametrized by a 3D bottom-up ConvNet, or a 3D generator model [HLX19] where a top-down 3D ConvNet maps a latent random vector to the observed video data. Such models do not have a dynamic structure defined by a transition model, and they are not convenient for predicting future

frames.

Contribution

The main contribution of this chapter lies in the combination of the dynamic generator model and the alternating back-propagation through time algorithm, as well as a novel generalization of the dynamic generator model to disentangle the appearance, trackable and intrackable motions. Both the model and algorithm are simple and natural, and their combination can be very useful for modeling and analyzing spatial-temporal processes. The model is one-piece in the sense that (1) the transition model and emission model are integrated into a single latent variable model. (2) The learning of the dynamic model is end-to-end, which is different from [HLZ17]'s treatment. (3) The learning of our model does not need to recruit a discriminative network (like GAN) or an inference network (like VAE), which makes our method simple and efficient in terms of computational cost and model parameter size. (4) The generalized model can learn disentangled representations in a purely unsupervised settings without ground truth or pre-inferred displacement fields or optical flows.

3.2 Model and learning algorithm

3.2.1 Dynamic generator model

Let $\mathbf{x} = (\mathbf{x}_t, t = 1, ..., T)$ be the observed video sequence, where $\mathbf{x}_t$ is a frame at time t. The dynamic generator model consists of the following two components [XGZ19]:

$$\mathbf{s}_t = \mathbf{f}_\alpha(\mathbf{s}_{t-1}, \boldsymbol{\xi}_t), \tag{3.1}$$

$$\mathbf{x}_t = \mathbf{g}_\beta(\mathbf{s}_t) + \boldsymbol{\epsilon}_t, \tag{3.2}$$

where $t = 1, ..., T$. (3.15) is the transition model, and (3.19) is the emission model. $\mathbf{s}_t$ is the d-dimensional hidden state vector. $\boldsymbol{\xi}_t \sim \mathcal{N}(0, \boldsymbol{I})$ is the noise vector of a certain dimensionality. The Gaussian noise vectors $(\boldsymbol{\xi}_t, t = 1, ..., T)$ are independent of each other. The sequence of

$(\mathbf{s}_t, t = 1, ..., T)$ follows a non-linear auto-regressive model, where the noise vector $\boldsymbol{\xi}_t$ encodes the randomness in the transition from $\mathbf{s}_{t-1}$ to $\mathbf{s}_t$ in the d-dimensional state space. $\mathbf{f}_\alpha$ is a feedforward neural network or multi-layer perceptron, where α denotes the weight and bias parameters of the network. We can adopt a residual form [HZR16a] for $\mathbf{f}_\alpha$ to model the change of the state vector. $\mathbf{x}_t$ is the D-dimensional image, which is generated by the d-dimensional hidden state vector $\mathbf{s}_t$. $\mathbf{g}_\beta$ is a top-down convolutional network (sometimes also called deconvolution network), where β denotes the weight and bias parameters of this top-down network. $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_D)$ is the residual error. We let $\theta = (\alpha, \beta)$ denote all the model parameters.

Let $\boldsymbol{\xi} = (\boldsymbol{\xi}_t, t = 1, ..., T)$. $\boldsymbol{\xi}$ consists of the latent random vectors that need to be inferred from $\mathbf{x}$. Although $\mathbf{x}_t$ is generated by the state vector $\mathbf{s}_t$, $\mathbf{s} = (\mathbf{s}_t, t = 1, ..., T)$ are generated by $\boldsymbol{\xi}$. In fact, we can write $\mathbf{x} = \mathbf{h}_\theta(\boldsymbol{\xi}) + \boldsymbol{\epsilon}$, where $\mathbf{h}_\theta$ composes $\mathbf{f}_\alpha$ and $\mathbf{g}_\beta$ over time, and $\boldsymbol{\epsilon} = (\boldsymbol{\epsilon}_t, t = 1, ..., T)$ denotes the observation errors.

3.2.2 Learning and inference algorithm

Let $p(\boldsymbol{\xi})$ be the prior distribution of $\boldsymbol{\xi}$. Let $p_\theta(\mathbf{x}|\boldsymbol{\xi}) \sim \mathcal{N}(\mathbf{h}_\theta(\boldsymbol{\xi}), \sigma^2 I)$ be the conditional distribution of $\mathbf{x}$ given $\boldsymbol{\xi}$, where $\boldsymbol{I}$ is the identity matrix whose dimension matches that of $\mathbf{x}$. The marginal distribution is $p_\theta(\mathbf{x}) = \int p(\boldsymbol{\xi}) p_\theta(\mathbf{x}|\boldsymbol{\xi}) d\boldsymbol{\xi}$ with the latent variable $\boldsymbol{\xi}$ integrated out. We estimate the model parameter θ by the maximum likelihood method that maximizes the observed-data log-likelihood $\log p_\theta(\mathbf{x})$, which is analytically intractable. In contrast, the complete-data log-likelihood $\log p_\theta(\boldsymbol{\xi}, \mathbf{x})$, where $p_\theta(\boldsymbol{\xi}, \mathbf{x}) = p(\boldsymbol{\xi}) p_\theta(\mathbf{x}|\boldsymbol{\xi})$, is analytically tractable. The following identity links the gradient of the observed-data log-likelihood $\log p_\theta(\mathbf{x})$ to the gradient of the complete-data log-likelihood $\log p_\theta(\boldsymbol{\xi}, \mathbf{x})$:

$$\begin{aligned}\frac{\partial}{\partial\theta}\log p_\theta(\mathbf{x}) &= \frac{1}{p_\theta(\mathbf{x})}\frac{\partial}{\partial\theta}p_\theta(\mathbf{x}) \\ &= \frac{1}{p_\theta(\mathbf{x})}\int\left[\frac{\partial}{\partial\theta}\log p_\theta(\boldsymbol{\xi},\mathbf{x})\right]p_\theta(\boldsymbol{\xi},\mathbf{x})d\boldsymbol{\xi} \\ &= \mathrm{E}_{p_\theta(\boldsymbol{\xi}|\mathbf{x})}\left[\frac{\partial}{\partial\theta}\log p_\theta(\boldsymbol{\xi},\mathbf{x})\right], \end{aligned} \tag{3.3}$$

where $p_\theta(\boldsymbol{\xi}|\mathbf{x}) = p_\theta(\boldsymbol{\xi},\mathbf{x})/p_\theta(\mathbf{x})$ is the posterior distribution of the latent $\boldsymbol{\xi}$ given the observed $\mathbf{x}$. The above expectation can be approximated by Monte Carlo average. Specifically, we sample from the posterior distribution $p_\theta(\boldsymbol{\xi}|\mathbf{x})$ using the Langevin dynamics:

$$\boldsymbol{\xi}^{(\tau+1)} = \boldsymbol{\xi}^{(\tau)} + \frac{\delta^2}{2}\frac{\partial}{\partial\boldsymbol{\xi}}\log p_\theta(\boldsymbol{\xi}^{(\tau)}|\mathbf{x}) + \delta\mathbf{z}_\tau, \tag{3.4}$$

where τ indexes the time step of the Langevin dynamics (not to be confused with the time step of the dynamics model, t), $\mathbf{z}_\tau \sim \mathcal{N}(0,\boldsymbol{I})$ where $\boldsymbol{I}$ is the identity matrix whose dimension matches that of $\boldsymbol{\xi}$, and $\boldsymbol{\xi}^{(\tau)} = (\boldsymbol{\xi}_t^{(\tau)}, t = 1,...,T)$ denotes all the sampled latent noise vectors at time step τ. δ is the step size of the Langevin dynamics. We can correct for the finite step size by adding a Metropolis-Hastings acceptance-rejection step. After sampling $\boldsymbol{\xi} \sim p_\theta(\boldsymbol{\xi}|\mathbf{x})$ using the Langevin dynamics, we can update θ by stochastic gradient ascent

$$\Delta\theta \propto \frac{\partial}{\partial\theta}\log p_\theta(\boldsymbol{\xi},\mathbf{x}), \tag{3.5}$$

where the stochasticity of the gradient ascent comes from the fact that we use Monte Carlo to approximate the expectation in (5.2). The learning algorithm iterates the following two steps. (1) Inference step: Given the current θ, sample $\boldsymbol{\xi}$ from $p_\theta(\boldsymbol{\xi}|\mathbf{x})$ according to (3.4). (2) Learning step: Given $\boldsymbol{\xi}$, update θ according to (3.5). We can use a warm start scheme for sampling in step (1). Specifically, when running the Langevin dynamics, we start from the current $\boldsymbol{\xi}$, and run a finite number of steps. Then we update θ in step (2) using the sampled $\boldsymbol{\xi}$. Such a stochastic gradient ascent algorithm has been analyzed by [You99].

Since $\frac{\partial}{\partial\boldsymbol{\xi}}\log p_\theta(\boldsymbol{\xi}|\mathbf{x}) = \frac{\partial}{\partial\boldsymbol{\xi}}\log p_\theta(\boldsymbol{\xi},\mathbf{x})$, both steps (1) and (2) involve derivatives of

$$\log p_\theta(\boldsymbol{\xi},\mathbf{x}) = -\frac{1}{2}\left[\|\boldsymbol{\xi}\|^2 + \frac{1}{\sigma^2}\|\mathbf{x} - \mathbf{h}_\theta(\boldsymbol{\xi})\|^2\right] + \text{const},$$

where the constant term does not depend on $\boldsymbol{\xi}$ or θ. Step (1) needs to compute the derivative of $\log p_\theta(\boldsymbol{\xi}, \mathbf{x})$ with respect to $\boldsymbol{\xi}$. Step (2) needs to compute the derivative of $\log p_\theta(\boldsymbol{\xi}, \mathbf{x})$ with respect to θ. Both can be computed by back-propagation through time. Therefore the algorithm is an alternating back-propagation through time algorithm. Step (1) can be called inferential back-propagation through time. Step (2) can be called learning back-propagation through time.

To be more specific, the complete-data log-likelihood $\log p_\theta(\boldsymbol{\xi}, \mathbf{x})$ can be written as (up to an additive constant, assuming $\sigma^2 = 1$)

$$L(\theta, \boldsymbol{\xi}) = -\frac{1}{2} \sum_{t=1}^{T} \left[\|\mathbf{x}_t - \mathbf{g}_\beta(\mathbf{s}_t)\|^2 + \|\boldsymbol{\xi}_t\|^2 \right]. \tag{3.6}$$

The derivative with respect to β is

$$\frac{\partial L}{\partial \beta} = \sum_{t=1}^{T} (\mathbf{x}_t - \mathbf{g}_\beta(\mathbf{s}_t)) \frac{\partial \mathbf{g}_\beta(\mathbf{s}_t)}{\partial \beta}. \tag{3.7}$$

The derivative with respect to α is

$$\frac{\partial L}{\partial \alpha} = \sum_{t=1}^{T} (\mathbf{x}_t - \mathbf{g}_\beta(\mathbf{s}_t)) \frac{\partial \mathbf{g}_\beta(\mathbf{s}_t)}{\partial \mathbf{s}_t} \frac{\partial \mathbf{s}_t}{\partial \alpha}, \tag{3.8}$$

where $\frac{\partial \mathbf{s}_t}{\partial \alpha}$ can be computed recursively. To infer $\boldsymbol{\xi}$, for any fixed time point t_0,

$$\frac{\partial L}{\partial \boldsymbol{\xi}_{t_0}} = \sum_{t=t_0+1}^{T} (\mathbf{x}_t - \mathbf{g}_\beta(\mathbf{s}_t)) \frac{\partial \mathbf{g}_\beta(\mathbf{s}_t)}{\partial \mathbf{s}_t} \frac{\partial \mathbf{s}_t}{\partial \boldsymbol{\xi}_{t_0}} - \boldsymbol{\xi}_{t_0}, \tag{3.9}$$

where $\frac{\partial \mathbf{s}_t}{\partial \boldsymbol{\xi}_{t_0}}$ can again be computed recursively.

A minor issue is the initialization of the transition model. We may assume that $s_0 \sim \mathcal{N}(0, \boldsymbol{I})$. In the inference step, we can sample $\mathbf{s}_0$ together with $\boldsymbol{\xi}$ using the Langevin dynamics.

It is worth mentioning the difference between our algorithm and the variational inference. While variational inference is convenient for learning a regular generator network, for the dynamic generator model studied in this chapter, it is not a simple task to design an inference model that infers the sequence of latent vectors $\boldsymbol{\xi} = (\boldsymbol{\xi}_t, t = 1, ..., T)$ from the sequence of $\mathbf{x} = (\mathbf{x}_t, t = 1, ..., T)$. In contrast, our learning method does not require such an inference model and can be easily implemented. The inference step in our model can be done via directly sampling from the posterior

distribution $p_\theta(\boldsymbol{\xi}|\mathbf{x})$, which is powered by back-propagation through time. Additionally, our model directly targets maximum likelihood, while model learning via variational inference is to maximize a lower bound.

3.2.3 Learning from multiple sequences

We can learn the model from multiple sequences of different appearances but of similar motion patterns. Let $\mathbf{x}^{(i)} = (\mathbf{x}_t^{(i)}, t = 1, ..., T)$ be the i-th training sequence, $i = 1, ..., n$. We can use an appearance (or content) vector $a^{(i)}$ for each sequence to account for the variation in appearance. The model is of the following form

$$\mathbf{s}_t^{(i)} = \mathbf{f}_\alpha(\mathbf{s}_{t-1}^{(i)}, \boldsymbol{\xi}_t^{(i)}), \tag{3.10}$$

$$\mathbf{x}_t^{(i)} = \mathbf{g}_\beta(\mathbf{s}_t^{(i)}, \mathbf{a}^{(i)}) + \boldsymbol{\epsilon}_t^{(i)}, \tag{3.11}$$

where $\mathbf{a}^{(i)} \sim \mathcal{N}(0, \boldsymbol{I})$, and $\mathbf{a}^{(i)}$ is fixed over time for each sequence i. To learn from such training data, we only need to add the Langevin sampling of $\mathbf{a}^{(i)}$. If the motion sequences are of different motion patterns, we can also introduce another vector $\mathbf{m}^{(i)} \sim \mathcal{N}(0, \boldsymbol{I})$ to account for the variations of motion patterns, so that the transition model becomes $\mathbf{s}_t^{(i)} = \mathbf{f}_\alpha(\mathbf{s}_{t-1}^{(i)}, \boldsymbol{\xi}_t^{(i)}, \mathbf{m}^{(i)})$ with $\mathbf{m}^{(i)}$ fixed for the sequence i.

Recently [TLY17] studies a similar model where the transition model is modeled by a recurrent neural network (RNN) with another layer of hidden vectors. [TLY17] learns the model using GAN. In comparison, we use a simpler Markov transition model and we learn the model by alternating back-propagation through time. Even though the latent state vectors follow a Markovian model, the observed sequence is non-Markovian.

Algorithm 1 summarizes the learning and inference algorithm for multiple sequences with appearance vectors. If we learn from a single sequence such as dynamic texture, we can remove the appearance vector $a^{(i)}$, or simply fix it to a zero vector.

Algorithm 4 Learning and inference by alternating back-propagation through time (ABPTT)

Require: (1) training sequences $\{\mathbf{x}^{(i)} = (\mathbf{x}_t^{(i)}, t = 1, ..., T), i = 1, ..., n\}$

(2) number of Langevin steps l

(3) number of learning iterations N.

Ensure: (1) learned parameters $\theta = (\alpha, \beta)$

(2) inferred noise vectors $\boldsymbol{\xi}^{(i)} = (\boldsymbol{\xi}_t^{(i)}, t = 1, ..., T)$.

1: Initialize $\theta = (\alpha, \beta)$. Initialize $\boldsymbol{\xi}^{(i)}$ and $\mathbf{a}^{(i)}$. Initialize $k = 0$.

2: **repeat**

3: **Inferential back-propagation through time**: For $i = 1, ..., n$, sample $\boldsymbol{\xi}^{(i)}$ and $\mathbf{a}^{(i)}$ by running l steps of Langevin dynamics according to (3.4), starting from their current values.

4: **Learning back-propagation through time**: Update α and β by gradient ascent according to (3.8) and (3.7).

5: Let $k \leftarrow k + 1$

6: **until** $k = N$

3.3 Experiments

Learn to generate dynamic textures

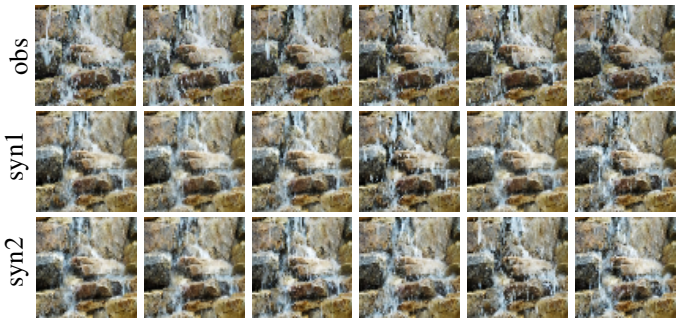
We first learn the model for dynamic textures, which are sequences of images of moving scenes that exhibit stationarity in time. We learn a separate model from each example. The video clips for training are collected from DynTex++ dataset of [GA10] and the Internet. Each observed video clip is prepared to be of the size 64 pixels $\times$ 64 pixels $\times$ 60 frames. Check [XGZ19] for implementation details. We can synthesize dynamic textures from the learned model by firstly randomly initializing the initial hidden state $\mathbf{s}_0$, and then following Equation (3.15) and (3.19) to generate a sequence of images with a sequence of innovation vectors $\{\boldsymbol{\xi}_t\}$ sampled from Gaussian distribution. In practice, we use "burn-in" to throw away some iterations at the beginning of the dynamic process to ensure the transition model enters the high probability region (i.e., the state sequence $\{\mathbf{s}_t\}$ converges to stationarity), no matter where $\mathbf{s}_0$ starts from.

Figure 3.3.1 shows several generation results. For each example, the first row displays 6 frames of the observed 60-frame sequence, while the second and third rows display 6 frames of two synthesized sequences of 120 frames in length, which are generated by the learned model.

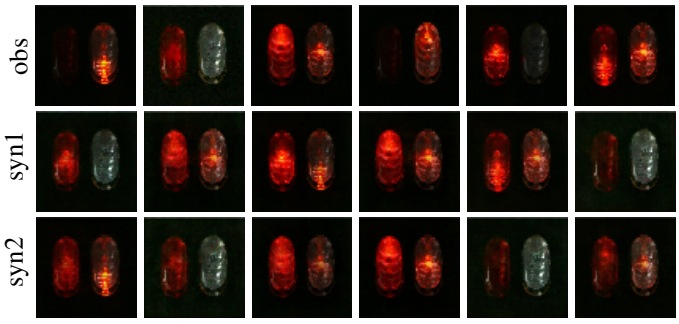
Similar to [TBD18], we perform a human perceptual study to evaluate the perceived realism of the synthesized examples. We randomly select 20 different human users. Each user has sequentially presented a pair of synthesized and real dynamic textures in a random order, and asked to select which one is fake after viewing them for a specified exposure time. The "fooling" rate, which is the user error rate in discriminating real versus synthesized dynamic textures, is calculated to measure the realism of the synthesized results. Higher "fooling" rate indicates more realistic and convincing synthesized dynamic textures. "Perfect" synthesized results correspond to a fooling rate of 50% (i.e., random guess), meaning that the users are unable to distinguish between the synthesized and real examples. The number of pairwise comparisons presented to each user is 36 (12 categories $\times$ 3 examples). The exposure time is chosen from discrete durations between 0.3 and 3.6 seconds.

(a) burning fire heating a pot

(b) flapping flag



(c) waterfall



(d) flashing lights

Figure 3.3.1: Generating dynamic textures. For each category, the first row displays 6 frames of the observed sequence, and the second and third rows show the corresponding frames of two synthesized sequences generated by the learned model.

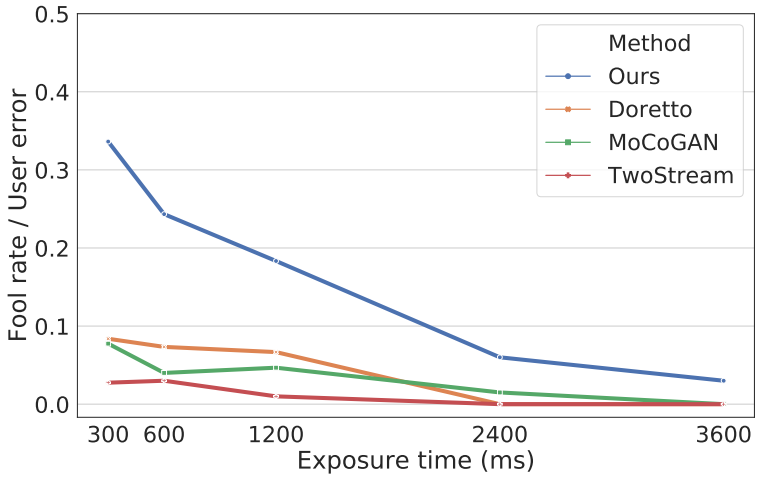


Figure 3.3.2: Limited time pairwise comparison results. Each curve shows the "fooling" rates (realism) over different exposure times.

We compare our model with three baseline methods, such as LDS (linear dynamic system) [DCW03], TwoStream [TBD18] and MoCoGAN [TLY17], for dynamic texture synthesis in terms of "fooling" rate on 12 dynamic texture videos (e.g., waterfall, burning fire, waving flag, etc).

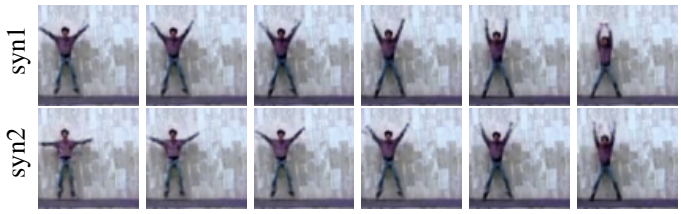
Figure 3.3.2 summarizes the comparative result by showing the "fooling" rate as a function of exposure time across methods. We can find that as the given exposure time becomes longer, it becomes easier for the users to observe the difference between the real and synthesized dynamic textures. More specifically, the "fooling" rate decreases as exposure time increases, and then remains at the same level for longer exposures. Overall, our method can generate more realistic dynamic textures than other baseline methods. The result also shows that the linear model (i.e., LDS) outperforms the more sophisticated baselines (i.e., TwoStream and MoCoGAN). The reason is that when learning from a single example, the MoCoGAN may not fit the training data very well due to the unstable and complicated adversarial training scheme as well as a large number of parameters to be learned, and the TwoStream method has a limitation that it cannot handle dynamic textures that have structured background (e.g., burning fire heating a pot).

Learn to generate action patterns with appearance consistency

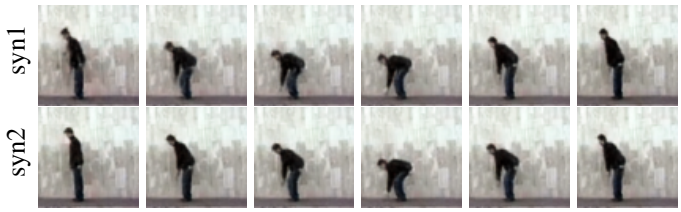
We learn the model from multiple examples with different appearances by using a 100-dimensional appearance vector. We infer the appearance vector and the initial state via a 15-step Langevin dynamics within each iteration of the learning process. We learn the model using the Weizmann action dataset [GBS07], which contains 81 videos of 9 people performing 9 actions, including jacking, jumping, walking, etc, as well as an animal action dataset that includes 20 videos of 10 animals performing running and walking collected from the Internet. Each video is scaled to 64×64 pixels $\times 30$ frames. We adopt the same structure of the model as the one in Section 3.3, except that the emission model takes the concatenation of the appearance vector and the hidden state as input. For each experiment, a single model is trained on the whole dataset without annotations. The dimensions of the hidden state $\mathbf{s}$ and the Gaussian noise $\boldsymbol{\xi}$ are set to be 100 and 50 respectively for the Weizmann action dataset, and 3 and 100 for the animal action dataset.

Figure 3.3.3 shows some synthesized results for each experiment. To synthesize video, we randomly pick an appearance vector inferred from the observed video and generate a new motion pattern for that specified appearance vector by the learned model with a noise sequence of $\{\boldsymbol{\xi}_t, t = 1, .., T\}$ and an initial state $\mathbf{s}_0$ sampled from Gaussian white noise. We show two different synthesized motions for each appearance vector. With a fixed appearance, the learned model can generate diverse motions with a consistent appearance.

Figure 3.3.4 shows two examples of video interpolation by interpolating between appearance vectors of videos at the two ends. We conduct these experiments on some videos selected from categories "blooming" and "melting" in the dataset of [ZB16]. For each example, the videos at the two ends are generated with the appearance vectors inferred from the two observed videos. Each video in the middle is obtained by first interpolating the appearance vectors of the two end videos, and then generating the videos using the dynamic generator. All the generated videos use the same set of noise sequence $\{\boldsymbol{\xi}_t\}$ and $\mathbf{s}_0$ randomly sampled from Gaussian white noise. We observe smooth transitions in contents and motions of all the generated videos and that the intermediate

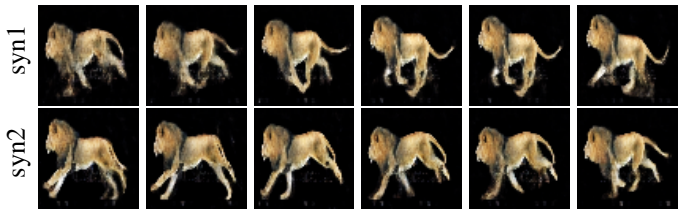


person 1

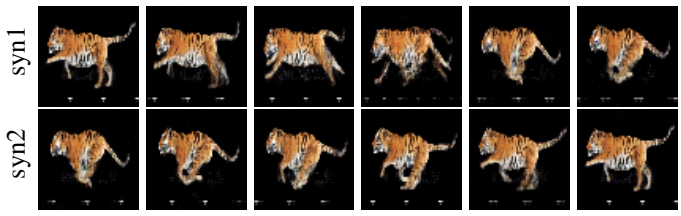


person 2

(a) synthesizing human actions (Weizmann dataset)



lion



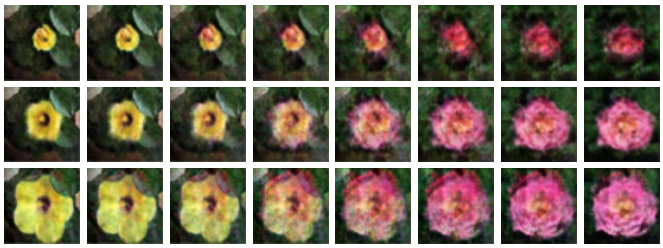
tiger

(b) synthesizing animal actions (animal action dataset)

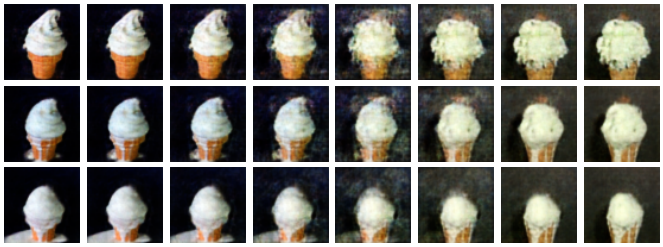
Figure 3.3.3: Generated action patterns. For each inferred appearance vector, two synthesized videos are displayed.

videos are also physically plausible.

We compare with MoCoGAN and TGAN [SMS17] by training on 9 selected categories (e.g., PlayingCello, PlayingDaf, PlayingDhol, PlayingFlute, PlayingGuitar, PlayingPiano, PlayingSi-



(a) blooming



(b) melting

Figure 3.3.4: Video interpolation by interpolating between appearance latent vectors of videos at the two ends. For each example, each column is one synthesized video. We show 3 frames for each video in each column.

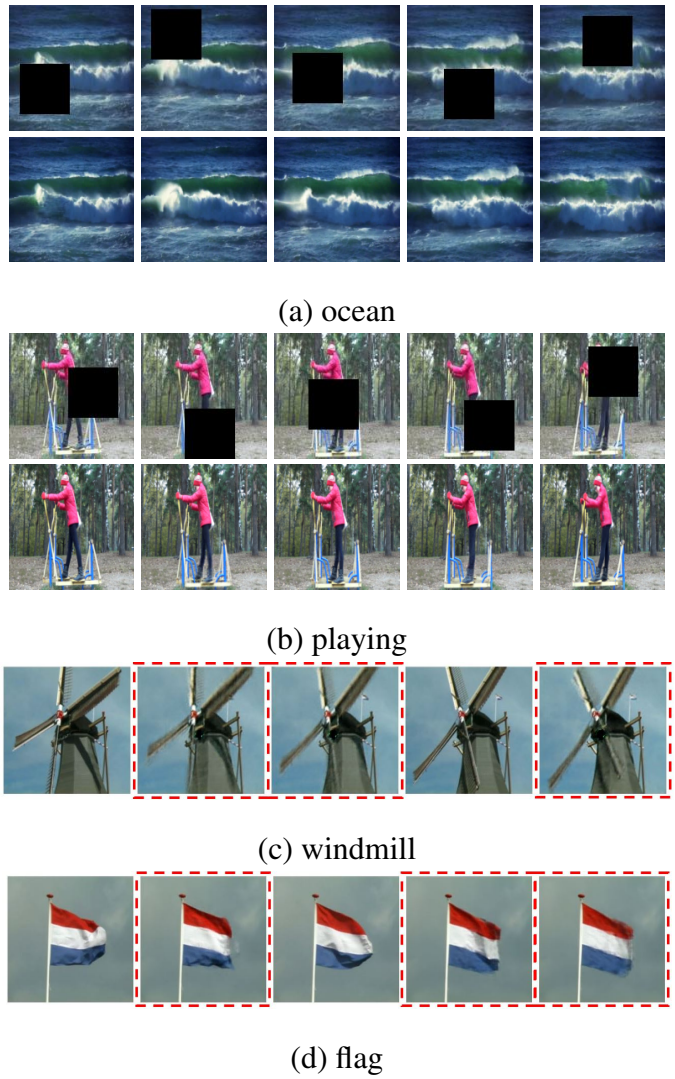
tar, PlayingTabla, and PlayingViolin) of videos in the UCF101 [SZS12] database and following [SMS17] to compute the inception score. Table 3.3.1 shows comparison results. Our model outperforms the MoCoGAN and TGAN in terms of inception score.

Table 3.3.1: Inception score for models trained on 9 classes of videos in UCF101 database.

Reference	ours	MoCoGAN	TGAN
11.05±0.16	**8.21±0.09**	4.40±0.04	5.48±0.06

Learn from incomplete data

Our model can learn from videos with occluded pixels and frames. We adapt our algorithm to this task with minimal modification involving the computation of $\sum_{t=1}^{T} \|\mathbf{x}_t - \mathbf{g}_\beta(\mathbf{s}_t)\|^2$. In the setting of



(a) ocean

(b) playing

(c) windmill

(d) flag

Figure 3.3.5: Learning from occluded videos. (a,b) For each experiment, the first row displays a segment of the occluded sequence with black masks. The second row shows the corresponding segment of the recovered sequence. (c,d) The 3 frames with red bounding box are recovered by the learning algorithm, and they are occluded in the training stage. Each video has 70 frames and 50% frames are randomly occluded.

learning from fully observed videos, it is computed by summing over all the pixels of the video frames, while in the setting of learning from partially visible videos, we compute it by summing over only the visible pixels of the video frames. Then we can continue to use the alternating back-

propagation through time (ABPTT) algorithm to infer $\{\boldsymbol{\xi}_t, t = 1, ..., T\}$ and $\mathbf{s}_0$, and then learn β and α. With inferred $\{\boldsymbol{\xi}_t\}$ and $\mathbf{s}_0$, and learned β and α, the video with occluded pixels or frames can be automatically recovered by $\mathbf{g}_\beta(\mathbf{s}_t)$, where the hidden state can be recursively computed by $\mathbf{s}_t = \mathbf{f}_\alpha(\mathbf{s}_{t-1}, \boldsymbol{\xi}_t)$.

Eventually, our model can achieve the following tasks: (1) recover the occluded pixels of training videos. (2) synthesize new videos by the learned model. (3) recover the occluded pixels of testing videos using the learned model. Different from those inpainting methods where the prior model has already been given or learned from fully observed training data, our recovery experiment is about an unsupervised learning task, where the ground truths of the occluded pixels are unknown in training the model for recovery. It is also worth mentioning that learning from incomplete data can be difficult for GANs (e.g., MoCoGAN), because of their lack of an adaptive inference process in the training stage. Here "adaptive" means the inference can be performed on input images with different sets of occluded pixels. We test our recovery algorithm on 6 video sequences collected from DynTex++ dataset.

We have two types of occlusions: (1) single region mask occlusion, where a 60×60 mask is randomly placed on each 150×150 image frame of each video. (2) missing image frames, where 50% of the image frames are randomly blocked in each video. For each type of occlusion experiment, we measure the recovery errors by the average per-pixel difference between the recovered video sequences and the original ones (The range of pixel intensities is [0, 255]), and compare with STGCN [XZW17b], which is a spatial-temporal deep convolutional energy-based model that can recover missing pixels of videos by synthesis during the learning process. We also report results obtained by generic spatial-temporal Markov random field models with potentials that are ℓ_1 or ℓ_2 difference between pixels of nearest neighbors that are defined in both spatial and temporal domains, and the recovery is accomplished by synthesizing missing pixels via Gibbs sampling. Table 3.3.2 shows the comparison results. Some qualitative results for recovery by our models are displayed in Figure 3.3.5.

Table 3.3.2: Recovery errors in occlusion experiments

(a) single region masks

	ours	STGCN	MRF-ℓ_1	MRF-ℓ_2
flag	**7.8782**	8.1636	10.6586	12.5300
fountain	**5.6988**	6.0323	11.8299	12.1696
ocean	**3.3966**	3.4842	8.7498	9.8078
playing	**4.9251**	6.1575	15.6296	15.7085
sea world	**5.6596**	5.8850	12.0297	12.2868
windmill	**6.6827**	7.8858	11.7355	13.2036
Avg.	**5.7068**	6.2681	11.7722	12.6177

(b) 50% missing frames

	ours	STGCN	MRF-ℓ_1	MRF-ℓ_2
flag	**5.0874**	5.5992	10.7171	12.6317
fountain	**5.5669**	8.0531	19.4331	13.2251
ocean	**3.3666**	4.0428	9.0838	9.8913
playing	**5.2563**	7.6103	22.2827	17.5692
sea world	**4.0682**	5.4348	13.5101	12.9305
windmill	**6.9267**	7.5346	13.3364	12.9911
Avg.	**5.0454**	6.3791	14.7272	13.2065

Learn to animate static image

A conditional version of the dynamic generator model can be used for video prediction given a static image. Specifically, we learn a mapping from a static image frame to the subsequent frames. We incorporate an extra encoder $\mathbf{e}_\gamma$, where γ denotes the weight and bias parameters of the encoder, to map the first image frame $\mathbf{x}_0^{(i)}$ into its appearance or content vector $\mathbf{a}^{(i)}$ and state vector $\mathbf{s}_0^{(i)}$. The dynamic generator takes the state vector $\mathbf{s}_0^{(i)}$ as the initial state and uses the appearance vector $\mathbf{a}^{(i)}$ to generate the subsequent video frames $\{\mathbf{x}_t^{(i)}, t = 1, ..., T\}$ for the i-th video. The conditional

model is of the following form

$$[\mathbf{s}_0^{(i)}, \mathbf{a}^{(i)}] = \mathbf{e}_\gamma(\mathbf{x}_0^{(i)}), \tag{3.12}$$

$$\mathbf{s}_t^{(i)} = \mathbf{f}_\alpha(\mathbf{s}_{t-1}^{(i)}, \boldsymbol{\xi}_t^{(i)}), \tag{3.13}$$

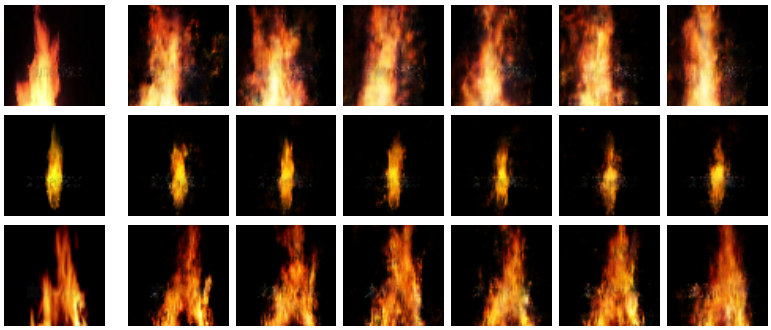
$$\mathbf{x}_t^{(i)} = \mathbf{g}_\beta(\mathbf{s}_t^{(i)}, \mathbf{a}^{(i)}) + \boldsymbol{\epsilon}_t^{(i)}. \tag{3.14}$$

We learn both the encoder and the dynamic generator (i.e., transition model and emission model) together by alternating back-propagation through time. The appearance vector and the initial state are no longer hidden variables that need to be inferred in training. Once the model is learned, given a testing static image, the learned encoder $\mathbf{e}_\gamma$ extracts from it the appearance vector and the initial state vector, which generate a sequence of images by the dynamic generator.

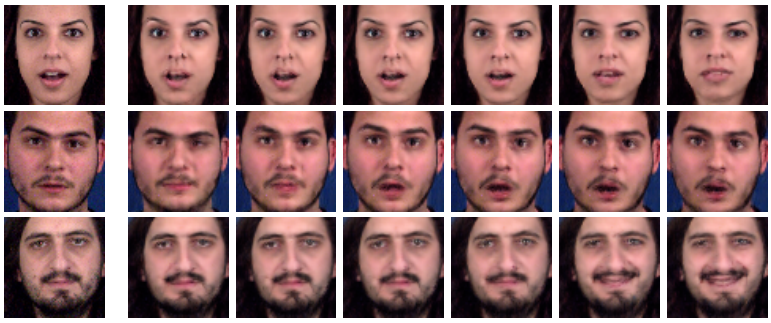
We test our model on burning fire dataset [XZW17b], and MUG Facial Expression dataset [ND10]. The encoder has 3 convolutional layers with numbers of channels $\{64, 128, 256\}$, filter sizes $\{5, 3, 3\}$ and sub-sampling factors $\{2, 2, 1\}$ at different layers, and one fully connected layer with the output size equal to the dimension of the appearance vector (100) plus the dimension of the hidden state (80). The dimension of $\boldsymbol{\xi}$ is 20. The other configurations are similar to what we used in Section 3.3. We qualitatively display some results in Figure 3.3.6, where each row is one example of image-to-video prediction. For each example, the left image is the static image frame for testing, and the rest are 6 frames of the predicted video sequence. The results show that the predicted frames by our method have fairly plausible motions.

3.4 Motion-based dynamic generator model

In this section, we generalize the dynamic generator model to further represent motion explicitly [XGZ20]. Specifically, Let $\mathbf{X} = (\mathbf{X}_t, t = 0, 1, ..., T)$ be the observed video sequence of dynamic pattern, where $\mathbf{X}_t$ is a frame at time t, and $\mathbf{X}_t$ is defined on the 2D rectangle lattice D. The



(a) burning fire



(b) facial expression

Figure 3.3.6: Image-to-video prediction. For each example, the first image is the static image frame, and the rest are 6 frames of the predicted sequence.

motion-based model for the dynamic patterns consists of the following components:

$$\mathbf{s}_t = (\mathbf{s}_t^M, \mathbf{s}_t^R) = \mathbf{f}_1(\mathbf{s}_{t-1}, \mathbf{h}_t), \tag{3.15}$$

$$\mathbf{M}_t = (\delta(x,y), \forall (x,y) \in D) = \mathbf{f}_2(\mathbf{s}_t^M), \tag{3.16}$$

$$\mathbf{R}_t = \mathbf{f}_3(\mathbf{s}_t^R), \tag{3.17}$$

$$\mathbf{I}_t = \mathbf{f}_4(\mathbf{I}_{t-1}, \mathbf{M}_t), \tag{3.18}$$

$$\mathbf{X}_t = \mathbf{I}_t + \mathbf{R}_t + \boldsymbol{\epsilon}_t, \tag{3.19}$$

where $t = 1, ..., T$.

In the above model, $\mathbf{f} = (\mathbf{f}_i, i = 0,1,2,3)$ are neural networks parameterized by $\theta = (\theta_i, i = 0,1,2,3)$.

Equation (3.15) is the transition model, where $\mathbf{s}_t$ is the state vector at time t, $\mathbf{h}_t \sim \mathcal{N}(0, \boldsymbol{I})$ is a hidden Gaussian white noise vector, where $\boldsymbol{I}$ is the identity matrix. $\mathbf{h}_t$ are independent over t. $\mathbf{f}_1$ defines the transition from $\mathbf{s}_{t-1}$ to $\mathbf{s}_t$.

The state vector $\mathbf{s}_t$ consists of two sub-vectors. One is $\mathbf{s}_t^M$ for motion. The other is $\mathbf{s}_t^R$ for residual. While $\mathbf{s}_t^M$ generates the motion of the trackable part of the image frame $\mathbf{X}_{t-1}$, $\mathbf{s}_t^R$ generates the non-trackable part of $\mathbf{X}_t$.

Specifically, in Equation (3.16), $\mathbf{s}_t^M$ generates the field of pixel displacement $\mathbf{M}_t$, which consists of the displacement $\delta(x,y)$ of pixel (x,y) in the image domain D. $\mathbf{M}_t$ is a 2D image, because the displacement $\delta = (\delta_x, \delta_y)$ is 2D. $\mathbf{f}_2$ defines the mapping from $\mathbf{s}_t^M$ to $\mathbf{M}_t$. In Equation (3.18), $\mathbf{M}_t$ is used to warp the trackable part $\mathbf{I}_{t-1}$ of the previous image frame $\mathbf{X}_{t-1}$ by a warping function $\mathbf{f}_4$, which is given by bilinear interpolation. There is no unknown parameter in $\mathbf{f}_4$. In Equation (3.17), $\mathbf{s}_t^R$ generates the residual image $\mathbf{R}_t$. $\mathbf{f}_3$ defines the mapping from $\mathbf{s}_t^R$ to $\mathbf{R}_t$. In Equation (3.19), the image frame $\mathbf{X}_t$ is the sum of the warped image $\mathbf{I}_t$ and the residual image $\mathbf{R}_t$, plus a Gaussian white noise error $\boldsymbol{\epsilon}_t \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$. We assume the variance σ^2 is given. In Equation (3.20), the initial trackable frame $\mathbf{I}_0$ is generated by an generator $\mathbf{f}_0$ from an appearance hidden variable $\mathbf{c}$ that

follows Gaussian distribution. To initialize the first frame $\mathbf{X}_0$, we use the following method:

$$\mathbf{I}_0 = \mathbf{f}_0(\mathbf{c}),\ \mathbf{R}_0 = \mathbf{f}_3(\mathbf{s}_0^R),\ \mathbf{X}_0 = \mathbf{I}_0 + \mathbf{R}_0 + \epsilon_0. \tag{3.20}$$

Multiple sequences. Similar to the dynamic generator, this motion-based dynamic generator model can be easily generalized to handle multiple sequences. We only need to introduce a sequence-specific vector $\mathbf{a}$, sampled from a Gaussian white noise prior distribution. For each video sequence, this vector $\mathbf{a}$ is fixed, and it can be concatenated to the state vector $\mathbf{s}_t$ in both the transition model and the emission model. We may also let $\mathbf{a} = (\mathbf{a}^M, \mathbf{a}^R)$, so that $\mathbf{a}^M$ is concatenated to $\mathbf{s}_t^M$ to generate $\mathbf{M}_t$, and $\mathbf{a}^R$ is concatenated to $\mathbf{s}_t^R$ to generate $\mathbf{R}_t$. This enables us to disentangle motion patterns and appearance patterns in the video sequence.

Intrackability. For the image $\mathbf{X}_t$, we define $\mathbf{I}_t$ to be the trackable part because it is obtained by the movements of pixels, and we define $\mathbf{R}_t$ to be the non-trackable part. The intrackability of the sequence can be defined as the ratio between the average of the ℓ_2 norm of the non-trackable part $\mathbf{R}_t$ and the norm of the image $\mathbf{X}_t$, where the average is over the time frames.

Summarized form. Let $\mathbf{h} = (h_t, t = 1, ..., T)$. h consist of the hidden random vectors that need to be inferred from $\mathbf{X} = (\mathbf{X}_t, t = 0, 1, ..., T)$. We can also include the latent variable $\mathbf{c}$ into $\mathbf{h}$ for notation simplicity. Although $\mathbf{X}_t$ is generated by the state vector $\mathbf{s}_t$, $\mathbf{s} = (\mathbf{s}_t, t = 0, 1, ..., T)$ are generated by $\mathbf{h}$. In fact, we can write $\mathbf{X} = \mathbf{f}_\theta(\mathbf{h}) + \epsilon$, where $\mathbf{f}_\theta$ composes $\mathbf{f}_0$, $\mathbf{f}_1$, $\mathbf{f}_2$, $\mathbf{f}_3$ and $\mathbf{f}_4$ over time t, and $\epsilon = (\epsilon_t, t = 0, 1, ..., T)$ denotes the observation errors.

We use the same alternating back-propagation through time (ABPTT) algorithm to learn this model as for the dynamic generator model.

3.4.1 Unsupervised disentanglement of appearance and motion

To study the performance of the proposed model for disentanglement of appearance and motion, we perform a motion transfer experiment. Figure 3.4.1 demonstrates the results. Figure 3.4.1 (a) displays some image frames of one observed video where a woman is performing a surprise expression. We first learn a model from the observed video. Figure 3.4.1 (b) visualizes the learned

motion. We then fix the inferred appearance latent vector and synthesize new surprise facial expressions by randomly sampling the latent vectors of the learned model. Two new synthesized "surprise" expressions on the same women are shown in Figure 3.4.1 (c). We further study transferring the learned motion to some unseen appearances. We select two unseen faces from the testing set. We apply the learned motion (i.e., the learned warping sequence) to the first frame of each testing video, and generate new image sequences, as shown in Figure 3.4.1 (d). We can also apply the learned motion to some faces from other domains. Figure 3.4.1 (e) shows two examples of transferring the learned motion to the cartoon face images. In each example, the image frame shown in the first column is the input appearance, and the rest image frames are generated when we apply the learned warping sequence to the input appearance. We can even apply the learned human facial expression motion to non-human appearances, such as animal faces (see Figure 3.4.1(f)).

3.4.2 Unsupervised disentanglement of trackable and intrackable components

Intrackability (or trackability) is an important concept of motion patterns, which has been studied in [GZ12]. It was demonstrated in [WZG08, GZ12, HXZ15] that trackability changes over scales, densities, and stochasticity of the dynamics. For example, the trackability of a video of the waterfall will depend on the distance between the observed target and the observer. Besides, the observer's preference for interpreting dynamic motions via tracking appearance details is a subjective factor to affect the perceived trackability of a dynamic pattern in the visual system of the brain.

In the context of our model, we can define intractability as the ratio between the average of ℓ_2 norm of the non-tractable residual image $\mathbf{R}_t$ and the average of the ℓ_2 norm of the observed image $\mathbf{X}_t$. This ratio depends on the penalty parameter λ_1 of the ℓ_2 norm of $\mathbf{R}_t$ used in the learning stage. This penalty parameter corresponds to the subjective preference mentioned above. The larger the preference λ_1 is, the larger extent to which we interpret a video by trackable contents, the less the residuals, and the less intrackability score.

Our model can unsupervised disentangle the trackable and intrackable components of the training videos. The intrackability can be directly obtained as a result of learning the model, where we do not need the ground truth or pre-inferred optical flows. In addition, the intrackability is defined in terms of the coherent motion pattern learned from the whole video sequence by our model.

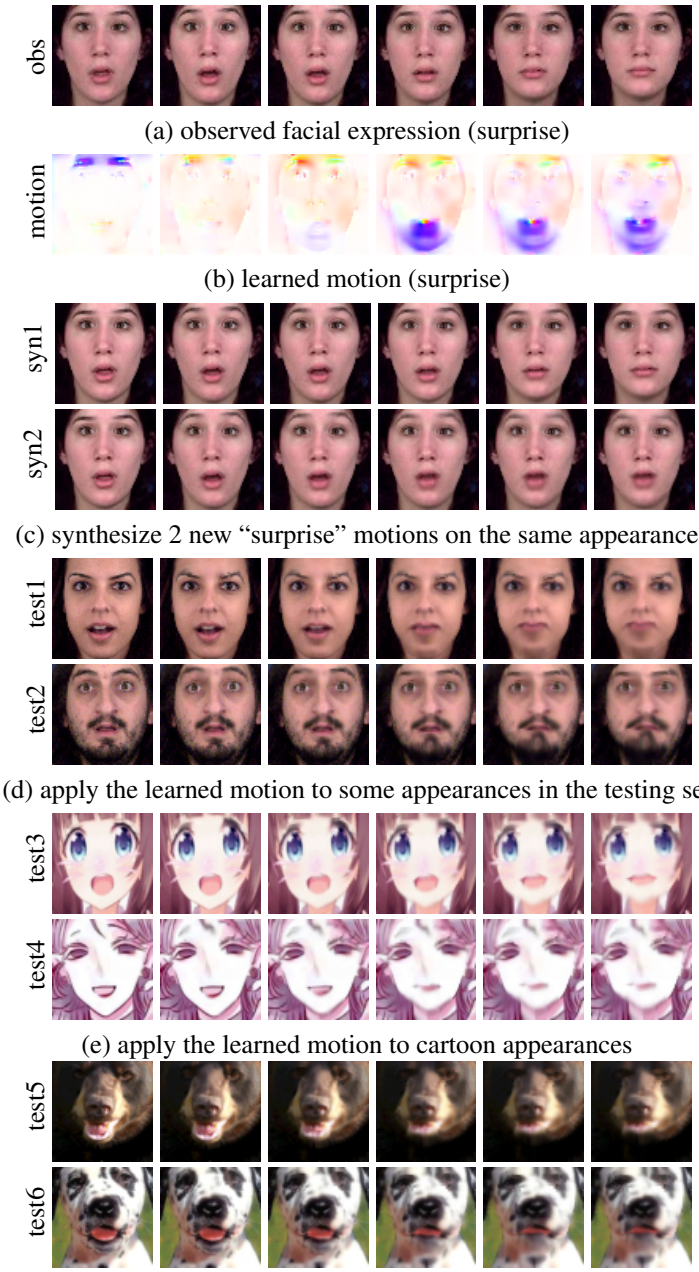
Figure 3.4.2 shows a curve of intrackability scores under different preference rates ($\lambda_1 = 0.5, 1, 2$ and 5) for each of 10 different dynamic patterns. One typical image frame is illustrated for each of the video clips that we used. The penalty parameter for smoothness is fixed to be 0.005. The results are reasonable and consistent with our empirical observations and intuitions. For example, under the same subjective preference, a video with structured background and slow-motion tends to have a lower intrackability score because one can track the elements in motion easily (e.g., a video clip exhibiting boiling water in a static pot), while a video with fast and random motion tends to have a higher intrackability score due to the loss of track of the elements in the video (e.g., a video clip exhibiting burning flame or flowing water). Moreover, we find that as the preference λ_1 increases, the intrackability of all videos decreases, because the model seeks to interpret each video using more trackable motion.

3.5 Conclusion

This chapter studies a dynamic generator model for spatial-temporal processes. The model is a non-linear generalization of the linear state-space model where the non-linear transformations in the transition and emission models are parameterized by neural networks. The model can be conveniently and efficiently learned by an alternating back-propagation through time (ABPTT) algorithm that alternatively samples from the posterior distribution of the latent noise vectors and then updates the model parameters.

The model can be generalized to a motion-based dynamic generator model by including random vectors to account for trackable and intrackable motions explicitly, and the learning algorithm can still apply to the generalized models. The generalized model is capable of disentangling the

image sequence into appearance, trackable and intrackable motions and we can learn the model without ground truth or pre-inference of the movements of the pixels or the optical flows. They are automatically inferred in the learning process.



(a) observed facial expression (surprise)

(b) learned motion (surprise)

(c) synthesize 2 new "surprise" motions on the same appearance

(d) apply the learned motion to some appearances in the testing set

(e) apply the learned motion to cartoon appearances

(f) apply the learned motion to animal appearances

Figure 3.4.1: Transferring motion to new appearance. (a) observed facial motion video. (b) The learned motion. (c) The synthesized new "surprise" motions on the same appearance. (d) Motion transfer to some new appearances in the testing set. (e) Motion transfer to some cartoon appearance collected from Internet. (f) Motion transfer to some animal faces collected from Internet.

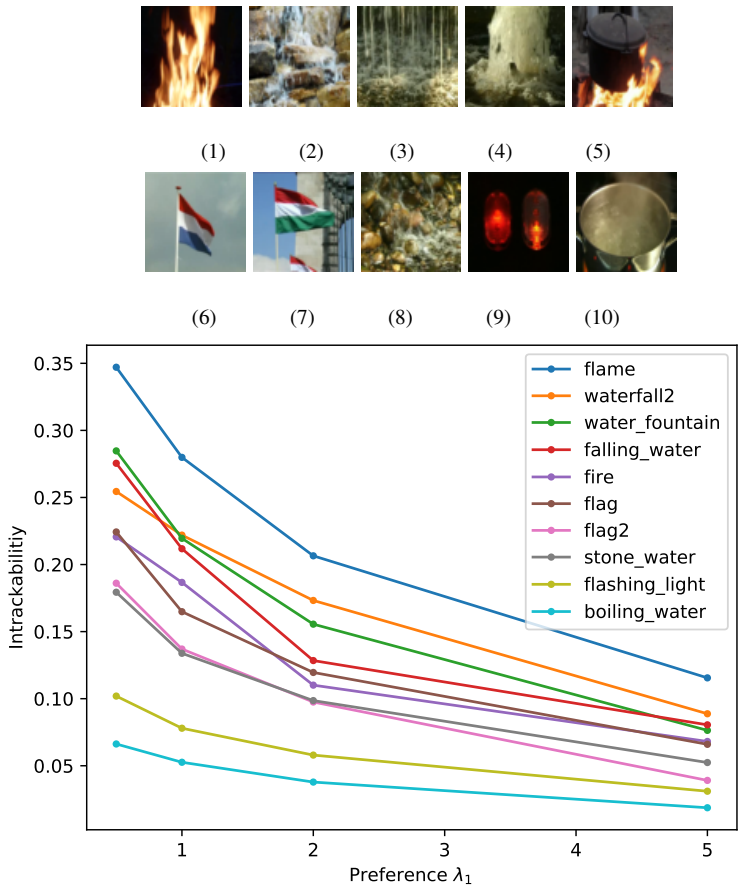


Figure 3.4.2: The intrackability score vs the preference rate λ_1 (the penalty parameter for the norm of the residual image). The penalty parameter for smoothness is 0.005.

CHAPTER 4

Hybrid models

4.1 Introduction

In this chapter, we introduce two hybrid models which couple a descriptive model, also known as an energy-based model (EBM), and a flow-based model to combine the best of two worlds[1]. The first one is a loose coupling where we jointly estimate an energy-based model and a flow-based model based on a shared value function, while the second one is a tight coupling where we learn EBM with a flow-based model serving as a backbone, so that the EBM is a correction or an exponential tilting of the flow-based model. In this section, we begin by introducing the flow-based model and the motivation to combine the two families of models.

4.1.1 Motivation

Recently, *flow-based models* (henceforth simply called *flow models*) have gained popularity as a type of deep generative model [DKB14, DSB16, GCB18, BDJ18, KBE19, TVA19, DBM19] and for use in variational inference [KW13, RM15, KSJ16].

Flow models have two properties that set them apart from other types of deep generative models: (1) they allow for efficient evaluation of the density function, and (2) they allow for efficient sampling from the model. See the next subsection for the detailed formulation of a flow-based model. Efficient evaluation of the log-density allows flow models to be directly optimized towards the log-likelihood objective, unlike variational autoencoders (VAEs) [KW13, RMW14], which are

[1]The main contributions in this chapter were first described in [GNK20, NGS20]

optimized towards a *bound* on the log-likelihood, and generative adversarial networks (GANs) [GPM14]. Auto-regressive models [Gra13, ODZ16, SKC17], on the other hand, are (in principle) inefficient to sample from, since synthesis requires computation that is proportional to the dimensionality of the data.

These properties of efficient density evaluation and efficient sampling are typically viewed as advantageous. However, they have a potential downside: these properties also act as *assumptions* on the true data distribution that they are trying to model. By choosing a flow model, one is making the assumption that the true data distribution is one that is in principle simple to sample from, and is computationally efficient to normalize. In addition, flow models assume that the data is generated by a finite sequence of invertible functions. If these assumptions do not hold, flow-based models can result in a poor fit.

On the other end of the spectrum of deep generative models lies the family of descriptive models, also known as energy-based models (EBMs). As introduced in Chapter 2, energy-based models define an unnormalized density that is the exponential of the negative *energy function*. The energy function is directly defined as a (learned) scalar function of the input, and is often parameterized by a neural network, such as a convolutional network [LBB98, KSH12]. Evaluation of the density function for a given data point involves calculating a normalizing constant, which requires an intractable integral. Sampling from EBMs is expensive and requires approximation as well, such as computationally expensive Markov Chain Monte Carlo (MCMC) sampling. EBMs, therefore, do not make any of the two assumptions above: they do not assume that the density of data is easily normalized, and they do not assume efficient synthesis. Moreover, they do not constrain the data distribution by invertible functions.

Contrasting an EBM with a flow model, the former is on the side of representation where different layers represent features of different complexities, whereas the latter is on the side of learned computation, where each layer, or each transformation, is like a step in the computation. The EBM is like an objective function or a target distribution whereas the flow model is like a finite step iterative algorithm or a learned sampler. Borrowing language from reinforcement learning

[FCA16], the flow model is like an actor whereas the EBM is like a critic or an evaluator. The EBM can be simpler and more flexible in form than the flow model which is highly constrained, and thus the EBM may capture the modes of the data distribution more accurately than the flow model. In contrast, the flow model is capable of direct generation via ancestral sampling, which is sorely lacking in an EBM. It may thus be desirable to couple the two models together, combining the tractability of the flow model and the flexibility of EBM. This is the goal of this chapter.

4.1.2 Flow-based model

A flow model is of the form

$$\mathbf{x} = \mathbf{g}_\alpha(\mathbf{z});\ \mathbf{z} \sim q_0(\mathbf{z}), \tag{4.1}$$

where q_0 is a known noise distribution. $\mathbf{g}_\alpha$ is a composition of a sequence of invertible transformations where the log-determinants of the Jacobians of the transformations can be explicitly obtained. α denotes the parameters. Let $q_\alpha(\mathbf{x})$ be the probability density of the model given a datapoint $\mathbf{x}$ with parameter α. Then under the *change of variables* $q_\alpha(\mathbf{x})$ can be expressed as

$$q_\alpha(\mathbf{x}) = q_0(\mathbf{g}_\alpha^{-1}(\mathbf{x}))|\det(\partial \mathbf{g}_\alpha^{-1}(\mathbf{x})/\partial \mathbf{x})|. \tag{4.2}$$

More specifically, suppose $\mathbf{g}_\alpha$ is composed of a sequence of transformations $\mathbf{g}_\alpha = \mathbf{g}_{\alpha_1} \circ \cdots \circ \mathbf{g}_{\alpha_m}$. The relation between $\mathbf{z}$ and $\mathbf{x}$ can be written as $\mathbf{z} \leftrightarrow \mathbf{h}_1 \leftrightarrow \cdots \leftrightarrow \mathbf{h}_{m-1} \leftrightarrow \mathbf{x}$. And thus we have

$$q_\alpha(\mathbf{x}) = q_0(\mathbf{g}_\alpha^{-1}(\mathbf{x}))\Pi_{i=1}^m|\det(\partial \mathbf{h}_{i-1}/\partial \mathbf{h}_i)|, \tag{4.3}$$

where we define $\mathbf{z} \coloneqq \mathbf{h}_0$ and $\mathbf{x} \coloneqq \mathbf{h}_m$ for conciseness. With carefully designed transformations, as explored in flow-based methods, the determinant of the Jacobian matrix $(\partial \mathbf{h}_{i-1}/\partial \mathbf{h}_i)$ can be incredibly simple to compute. The key idea is to choose transformations whose Jacobian is a triangle matrix, so that the determinant becomes

$$|\det(\partial \mathbf{h}_{i-1}/\partial \mathbf{h}_i)| = \Pi|\mathrm{diag}(\partial \mathbf{h}_{i-1}/\partial \mathbf{h}_i)|. \tag{4.4}$$

The following are the two scenarios for estimating q_α:

(1) Generative modeling by MLE [DKB14, DSB16, KD18, GCB18, BDJ18, KBE19, TVA19], based on $\min_\alpha \mathrm{KL}(p_{\text{data}} \| q_\alpha)$, where again $\mathrm{E}_{p_{\text{data}}}$ can be approximated by average over observed examples.

(2) Variational approximation to an unnormalized target density p [KW13, RM15, KSJ16, KW14, KKH19], based on $\min_\alpha \mathrm{KL}(q_\alpha \| p)$, where

$$\mathrm{KL}(q_\alpha \| p) = \mathrm{E}_{q_\alpha}[\log q_\alpha(\mathbf{x})] - \mathrm{E}_{q_\alpha}[\log p(\mathbf{x})] \tag{4.5}$$

$$= \mathrm{E}_{\mathbf{z}}[\log q_0(\mathbf{z}) - \log |\det(\mathbf{g}'_\alpha(\mathbf{z}))|] - \mathrm{E}_{q_\alpha}[\log p(\mathbf{x})]. \tag{4.6}$$

$\mathrm{KL}(q_\alpha \| p)$ is the difference between energy and entropy, i.e., we want q_α to have low energy but high entropy. $\mathrm{KL}(q_\alpha \| p)$ can be calculated without inversion of $\mathbf{g}_\alpha$.

When q_α appears on the right of KL-divergence, as in (1), it is forced to cover most of the modes of p_{data}, When q_α appears on the left of KL-divergence, as in (2), it tends to chase the major modes of p while ignoring the minor modes [Mur12, FR12]. As shown in the following section, our proposed method learns a flow model by combining (1) and (2).

4.2 Flow contrastive estimation

In this section, we study a training method to jointly estimate an energy-based model and a flow-based model, in which the two models are iteratively updated based on a shared adversarial value function [GNK20]. Our joint training method is inspired by the noise contrastive estimation (NCE) of [GH10], where an EBM is learned discriminatively by classifying the real data and the data generated by a noise model. In NCE, the noise model must have an explicit normalized density function. Moreover, it is desirable for the noise distribution to be close to the data distribution for accurate estimation of the EBM. However, the noise distribution can be far away from the data distribution. The flow model can potentially transform or transport the noise distribution to a distribution closer to the data distribution. With the advent of strong flow-based generative models [DKB14, DSB16, KD18], it is natural to recruit the flow model as the contrast distribution for

noise contrastive estimation of the EBM.

However, even with the flow-based model pre-trained by maximum likelihood estimation (MLE) on the data distribution, it may still not be strong enough as a contrast distribution, in the sense that the synthesized examples generated by the pre-trained flow model may still be distinguished from the real examples by a classifier based on an EBM. Thus, we want the flow model to be a stronger contrast or a stronger training opponent for EBM. To achieve this goal, we can simply use the same objective function of NCE, which is the log-likelihood of the logistic regression for classification. While NCE updates the EBM by maximizing this objective function, we can also update the flow model by minimizing the same objective function to make the classification task harder for the EBM. Such update of the flow model combines MLE and variational approximation, and helps correct the over-dispersion of MLE. If the EBM is close to the data distribution, this amounts to minimizing the Jensen-Shannon divergence (JSD) [GPM14] between the data distribution and the flow model. In this sense, the learning scheme relates closely to GANs [GPM14]. However, unlike GANs, which learns a generator model that defines an implicit probability density function via a low-dimensional latent vector, our method learns two probabilistic models with explicit probability densities (a normalized one and an unnormalized one).

The contributions of this section are as follows. We explore a parameter estimation method that couples the estimation of an EBM and a flow model using a shared objective function. It improves NCE with a flow-transformed noise distribution, and modifies the MLE of the flow model to approximate JSD minimization, and helps correct the over-dispersion of MLE. Experiments on 2D synthetic data show that the learned EBM achieves accurate density estimation with a much simpler network structure than the flow model. On real image datasets, we demonstrate a significant improvement in the synthesis quality of the flow model, and the effectiveness of unsupervised feature learning by the energy-based model. Furthermore, we show that the proposed method can be easily adapted to semi-supervised learning, achieving performance comparable to state-of-the-art semi-supervised methods.

4.2.1 Noise contrastive estimation

Noise contrastive estimation (NCE) can be used to learn the EBM, by including the normalizing constant as another learnable parameter. Specifically, for an energy-based model $p_\theta(\mathbf{x}) = \frac{1}{Z(\theta)}\exp[f_\theta(\mathbf{x})]$, we define $p_\theta(\mathbf{x}) = \exp[f_\theta(\mathbf{x}) - c]$, where $c = \log Z(\theta)$. c is now treated as a free parameter, and is included into θ. Suppose we observe training examples $\{\mathbf{x}_i, i = 1, ..., n\}$, and we have generated examples $\{\tilde{\mathbf{x}}_i, i = 1, ..., n\}$ from a noise distribution $q(\mathbf{x})$. Then θ can be estimated by maximizing the following objective function:

$$J(\theta) = \mathrm{E}_{p_{\text{data}}}\left[\log \frac{p_\theta(\mathbf{x})}{p_\theta(\mathbf{x}) + q(\mathbf{x})}\right] + \mathrm{E}_q\left[\log \frac{q(\mathbf{x})}{p_\theta(\mathbf{x}) + q(\mathbf{x})}\right], \tag{4.7}$$

which transforms estimation of EBM into a classification problem.

The objective function connects to logistic regression in supervised learning in the following sense. Suppose for each training or generated examples we assign a binary class label y: $y = 1$ if $\mathbf{x}$ is from training dataset and $y = 0$ if $\mathbf{x}$ is generated from $q(\mathbf{x})$. In logistic regression, the posterior probabilities of classes given the data $\mathbf{x}$ are estimated. As the data distribution $p_{\text{data}}(\mathbf{x})$ is unknown, the class-conditional probability $p(\cdot|y = 1)$ is modeled with $p_\theta(\mathbf{x})$. And $p(\cdot|y = 0)$ is modeled by $q(\mathbf{x})$. Suppose we assume equal probabilities for the two class labels, i.e., $p(y = 1) = p(y = 0) = 0.5$. Then we obtain the posterior probabilities:

$$p_\theta(y = 1|\mathbf{x}) = \frac{p_\theta(\mathbf{x})}{p_\theta(\mathbf{x}) + q(\mathbf{x})} \coloneqq u(\mathbf{x}, \theta). \tag{4.8}$$

The class-labels y are Bernoulli-distributed, so that the log-likelihood of the parameter θ becomes

$$l(\theta) = \sum_{i=1}^{n} \log u(\mathbf{x}_i; \theta) + \sum_{i=1}^{n} \log(1 - u(\tilde{\mathbf{x}}_i; \theta)), \tag{4.9}$$

which is, up to a factor of $1/n$, an approximation of eqn. 4.7.

The choice of the noise distribution $q(\mathbf{x})$ is a design issue. Generally speaking, we expect $q(\mathbf{x})$ to satisfy the following: (1) analytically tractable expression of normalized density; (2) easy to draw samples from; (3) close to data distribution. In practice, (3) is important for learning a model over high-dimensional data. If $q(\mathbf{x})$ is not close to the data distribution, the classification problem would be too easy and would not require p_θ to learn much about the modality of the data.

4.2.2 Flow contrastive estimation

A natural improvement to NCE is to transform the noise so that the resulting distribution is closer to the data distribution. This is exactly what the flow model achieves. That is, a flow model transforms a known noise distribution $q_0(\mathbf{z})$ by a composition of a sequence of invertible transformations $\mathbf{g}_\alpha(\cdot)$. It also fulfills (1) and (2) of the requirements of NCE. However, in practice, we find that a pre-trained $q_\alpha(\mathbf{x})$, such as learned by MLE, is not strong enough for learning an EBM $p_\theta(\mathbf{x})$ because the synthesized data from the MLE of $q_\alpha(\mathbf{x})$ can still be easily distinguished from the real data by an EBM. Thus, we propose to iteratively train the EBM and flow model, in which case the flow model is adaptively adjusted to become a stronger contrast distribution or a stronger training opponent for EBM. This is achieved by a parameter estimation scheme similar to GAN, where $p_\theta(x)$ and $q_\alpha(\mathbf{x})$ play a minimax game with a unified value function: $\min_\alpha \max_\theta V(\theta, \alpha)$,

$$V(\theta, \alpha) = \mathrm{E}_{p_{\text{data}}}\left[\log \frac{p_\theta(\mathbf{x})}{p_\theta(\mathbf{x}) + q_\alpha(\mathbf{x})}\right] + \mathrm{E}_{\mathbf{z}}\left[\log \frac{q_\alpha(\mathbf{g}_\alpha(\mathbf{z}))}{p_\theta(\mathbf{g}_\alpha(\mathbf{z})) + q_\alpha(\mathbf{g}_\alpha(\mathbf{z}))}\right], \tag{4.10}$$

where $\mathrm{E}_{p_{\text{data}}}$ is approximated by averaging over observed samples $\{\mathbf{x}_i, i = 1, ..., n\}$, while $\mathrm{E}_{\mathbf{z}}$ is approximated by averaging over negative samples $\{\tilde{\mathbf{x}}_i, i = 1, ..., n\}$ drawn from $q_\alpha(\mathbf{x})$, with $\mathbf{z}_i \sim q_0(\mathbf{z})$ independently for $i = 1, ..., n$. In the experiments, we choose Glow [KD18] as the flow-based model. The algorithm can either start from a randomly initialized Glow model or a pre-trained one by MLE. Here we assume equal prior probabilities for observed samples and negative samples. It can be easily modified to the situation where we assign a higher prior probability to the negative samples, given the fact we have access to infinite amount of free negative samples.

The objective function can be interpreted from the following perspectives:

(1) Noise contrastive estimation for EBM. The update of θ can be seen as noise contrastive estimation of $p_\theta(\mathbf{x})$, but with a flow-transformed noise distribution $q_\alpha(\mathbf{x})$ which is adaptively updated. The training is essentially a logistic regression. However, unlike regular logistic regression for classification, for each $\mathbf{x}_i$ or $\tilde{\mathbf{x}}_i$, we must include $\log q_\alpha(\mathbf{x}_i)$ or $\log q_\alpha(\tilde{\mathbf{x}}_i)$ as an example-dependent bias term. This forces $p_\theta(\mathbf{x})$ to replicate $q_\alpha(\mathbf{x})$ in addition to distinguishing between $p_{p_{\text{data}}}(\mathbf{x})$ and $q_\alpha(\mathbf{x})$, so that $p_\theta(\mathbf{x}_i)$ is in general larger than $q_\alpha(\mathbf{x}_i)$, and $p_\theta(\tilde{\mathbf{x}}_i)$ is in general smaller than $q_\alpha(\tilde{\mathbf{x}}_i)$.

(2) Minimization of Jensen-Shannon divergence for the flow model. If $p_\theta(\mathbf{x})$ is close to the data distribution, then the update of α is approximately minimizing the Jensen-Shannon divergence between the flow model q_α and data distribution p_{data}:

$$\text{JSD}(q_\alpha \| p_{\text{data}}) = \text{KL}(p_{\text{data}} \| (p_{\text{data}} + q_\alpha)/2) + \text{KL}(q_\alpha \| (p_{\text{data}} + q_\alpha)/2). \tag{4.11}$$

Its gradient w.r.t. α equals the gradient of $-\text{E}_{p_{\text{data}}}[\log((p_\theta + q_\alpha)/2)] + \text{KL}(q_\alpha \| (p_\theta + q_\alpha)/2)$. The gradient of the first term resembles MLE, which forces q_α to cover the modes of data distribution, and tends to lead to an over-dispersed model, which is also pointed out in [KD18]. The gradient of the second term is similar to reverse Kullback-Leibler divergence between q_α and p_θ, or variational approximation of p_θ by q_α, which forces q_α to chase the modes of p_θ [Mur12, FR12]. This may help correct the over-dispersion of MLE, and combines the two scenarios of estimating the flow-based model q_α as described in subsection 4.1.2.

(3) Connection with GAN. Our parameter estimation scheme is closely related to GAN. In GAN, the discriminator D and generator G play a minimax game: $\min_G \max_D V(G,D)$,

$$V(G,D) = \text{E}_{p_{\text{data}}}[\log D(\mathbf{x})] + \text{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z}_i)))]. \tag{4.12}$$

The discriminator $D(\mathbf{x})$ is learning the probability ratio $p_{\text{data}}(\mathbf{x})/(p_{\text{data}}(\mathbf{x}) + p_G(\mathbf{x}))$, which is about the difference between p_{data} and p_G [FCA16]. In the end, if the generator G learns to perfectly replicate p_{data}, then the discriminator D ends up with a random guess. However, in our method, the ratio is explicitly modeled by p_θ and q_α. p_θ must contain all the learned knowledge in q_α, in addition to the difference between p_{data} and q_α. In the end, we learn two explicit probability distributions p_θ and q_α as approximations to p_{data}.

Henceforth we simply refer to the proposed method as flow contrastive estimation, or FCE.

4.2.3 Semi-supervised learning

A class-conditional energy-based model can be transformed into a discriminative model in the following sense. Suppose there are K categories $k = 1, ..., K$, and the model learns a distinct density

$p_{\theta_k}(\mathbf{x})$ for each k. The networks $f_{\theta_k}(\mathbf{x})$ for $k = 1, ..., K$ may share common lower layers, but with different top layers. Let ρ_k be the prior probability of category k, for $k = 1, ..., K$. Then the posterior probability for classifying $\mathbf{x}$ to the category k is a softmax multi-class classifier

$$P(k|\mathbf{x}) = \frac{\exp(f_{\theta_k}(\mathbf{x}) + b_k)}{\sum_{l=1}^{K} \exp(f_{\theta_l}(\mathbf{x}) + b_l)}, \tag{4.13}$$

where $b_k = \log(\rho_k) - \log Z(\theta_k)$.

Given this correspondence, we can modify FCE to do semi-supervised learning. Specifically, assume $\{(\mathbf{x}_i, y_i), i = 1, ..., m\}$ are observed examples with labels known, and $\{\mathbf{x}_i, i = m+1, ..., m+n\}$ are observed unlabeled examples. For each category k, we can assume that class-conditional EBM is in the form

$$p_{\theta_k}(\mathbf{x}) = \frac{1}{Z(\theta_k)} \exp[f_{\theta_k}(\mathbf{x})] = \exp[f_{\theta_k}(\mathbf{x}) - c_k], \tag{4.14}$$

where $f_{\theta_k}(\mathbf{x})$ share all the weights except for the top layer. And we assume equal prior probability for each category. Let θ denotes all the parameters from class-conditional EBMs $\{\theta_k, k = 1, ..., K\}$. For labeled examples, we can maximize the conditional posterior probability of label y, given $\mathbf{x}$ and the fact that $\mathbf{x}$ is an observed example (instead of a generated example from q_α). By Bayes rule, this leads to maximizing the following objective function over θ:

$$\begin{aligned} L_{\text{label}}(\theta) &= \mathrm{E}_{p_{\text{data}}(\mathbf{x},y)} \left[\log p_\theta(y|\mathbf{x}, y \in \{1, ..., K\})\right] \\ &= \mathrm{E}_{p_{\text{data}}(\mathbf{x},y)} \left[\log \frac{p_{\theta_y}(\mathbf{x})}{\sum_{k=1}^{K} p_{\theta_k}(\mathbf{x})}\right], \end{aligned} \tag{4.15}$$

which is similar to a classifier in the form.

For unlabeled examples, the probability can be defined by an unconditional EBM, which is in the form of a mixture model:

$$p_\theta(\mathbf{x}) = \sum_{i=1}^{K} p_\theta(\mathbf{x}|y = k)p(y = k) = \frac{1}{K} \sum_{i=1}^{K} p_{\theta_k}(\mathbf{x}), \tag{4.16}$$

Together with the generated examples from $q_\alpha(\mathbf{x})$, we can define the same value function $V(\theta, \alpha)$ as eqn. 4.10 for the unlabeled examples. The joint estimation algorithm alternate the following two steps: (1) update θ by $\max_\theta L_{\text{label}}(\theta) + V(\theta, \alpha)$; (2) update α by $\min_\alpha V(\theta, \alpha)$. Due to

the flexibility of EBM, $f_{\theta_k}(\mathbf{x})$ can be defined by any existing state-of-the-art network structures designed for semi-supervised learning.

4.2.4 Experiments

For FCE, we adaptively adjust the numbers of updates for EBM and Glow: we first update EBM for a few iterations until the classification accuracy is above 0.5, and then we update Glow until the classification accuracy is below 0.5. We use *Adam* [KB14] with learning rate $\alpha = 0.0003$ for the EBM and *Adamax* [KB14] with learning rate $\alpha = 0.00001$ for the Glow model. Code and more results can be found at `http://www.stat.ucla.edu/~ruiqigao/fce/main.html`

4.2.4.1 Density estimation on 2D synthetic data

Figure 4.2.1 demonstrates the results of FCE on several 2D distributions, where FCE starts from a randomly initialized Glow. The learned EBM can fit multi-modal distributions accurately, and forms a better fit than Glow learned by either FCE or MLE. Notably, the EBM is defined by a much simpler network structure than Glow: for Glow we use 10 affine coupling layers, which amount to 30 fully-connected layers, while the energy-based model is defined by a 4-layer fully-connected network with the same width as Glow. Another interesting finding is that the EBM can fit the distributions well, even if the flow model is not a perfect contrastive distribution.

For the distribution depicted in the first row of Figure 4.2.1, which is a mixture of eight Gaussian distributions, we can compare the estimated densities by the learned models with the ground truth densities. Figure 4.2.2 shows the mean squared error of the estimated log-density over numbers of training iterations of EBMs. We show the results of FCE either starting from a randomly initialized Glow ('rand') or a Glow model pre-trained by MLE ('trained'), and compare with NCE with a Gaussian noise distribution. FCE starting from a randomly initialized Glow converges in fewer iterations. Both settings of FCE achieve a lower error rate than NCE.

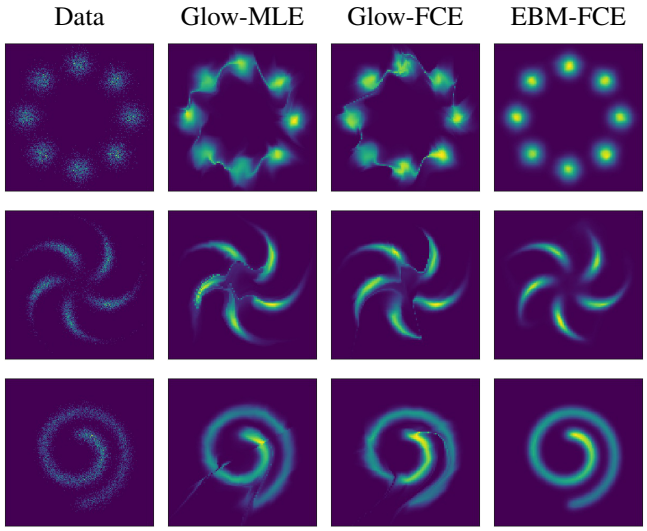


Figure 4.2.1: Comparison of trained EBM and Glow models on 2-dimensional data distributions.

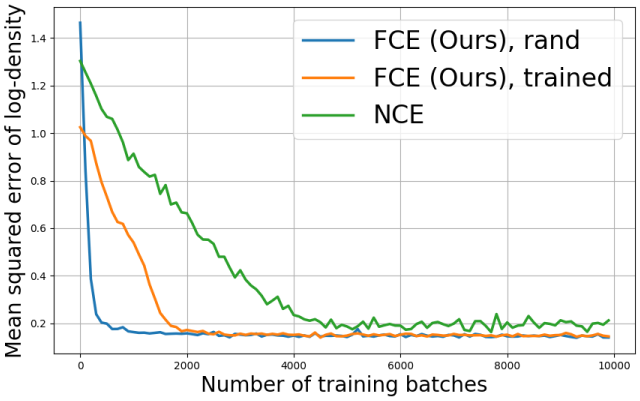


Figure 4.2.2: Density estimation accuracy in 2D examples of a mixture of 8 Gaussian distributions.

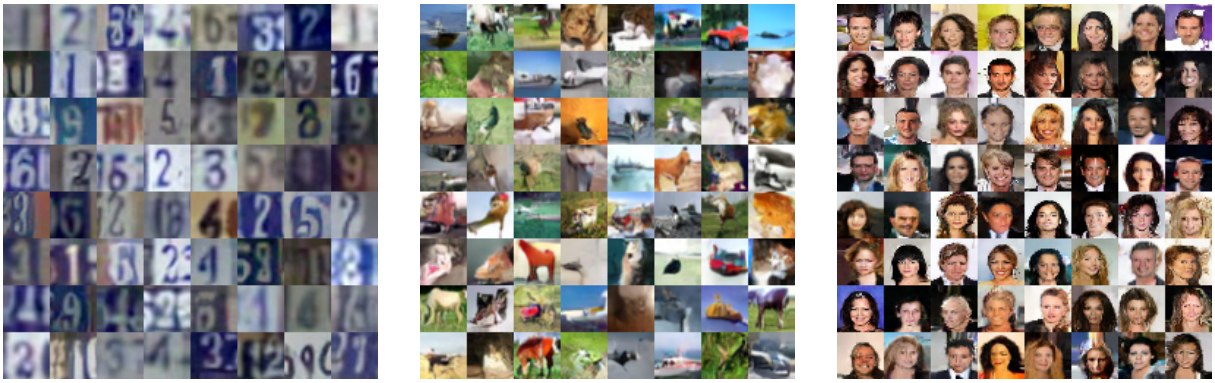


Figure 4.2.3: Synthesized examples from the Glow model learned by FCE. From left to right panels are from SVHN, CIFAR-10 and CelebA datasets, respectively. The image size is 32×32.

4.2.4.2 Learning on real image datasets

We conduct experiments on the Street View House Numbers (SVHN) [NWC11], CIFAR-10 [KH09] and CelebA [LLW18] datasets. We resized the CelebA images to 32×32 pixels, and used $20,000$ images as a test set. We initialize FCE with a pre-trained Glow model, trained by MLE, for the sake of efficiency. We again emphasize the simplicity of the EBM model structure compared to Glow. For Glow, depth per level [KD18] is set as 8, 16, 32 for SVHN, CelebA and CIFAR-10 respectively. Figure 4.2.3 depicts synthesized examples from learned Glow models. To evaluate the fidelity of synthesized examples, Table 4.2.1 summarizes the Fréchet Inception Distance (FID) [HRU17] of the synthesized examples computed with the Inception V3 [SVI16] classifier. The fidelity is significantly improved compared to Glow trained by MLE, and is competitive to the other generative models. In Table 4.2.2, we report the average negative log-likelihood (bits per dimension) on the testing sets. The log-likelihood of the learned EBM is based on the estimated normalizing constant (i.e., a parameter of the model) and should be taken with a grain of salt. For the learned Glow model, the log-likelihood of the Glow model estimated with FCE is slightly lower than the log-likelihood of the Glow model trained with MLE.

Table 4.2.1: FID scores for generated samples. For our method, we evaluate generative samples from the learned Glow model.

Method	SVHN	CIFAR-10	CelebA
VAE [KW13]	57.25	78.41	38.76
DCGAN [RMC15]	21.40	37.70	12.50
Glow [KD18]	41.70	45.99	23.32
FCE (Ours)	**20.19**	**37.30**	**12.21**

Table 4.2.2: Bits per dimension on testing data. † indicates that the log-likelihood is computed based on models with estimated normalizing constant, and should be taken with a grain of salt.

Model	SVHN	CIFAR-10	CelebA
Glow-MLE	2.17	3.35	3.49
Glow-FCE (Ours)	2.25	3.45	3.54
EBM-FCE (Ours)	†2.15	†3.27	†3.40

4.2.4.3 Unsupervised feature learning

To further explore the EBM learned with FCE, we perform unsupervised feature learning with features from a learned EBM. Specifically, we first conduct FCE on the entire training set of SVHN in an unsupervised way. Then, we extract the top layer feature maps from the learned EBM, and train a linear classifier on top of the extracted features using only a subset of the training images and their corresponding labels. Figure 4.2.4 shows the classification accuracy as a function of the number of labeled examples. Meanwhile, we compare our method with a supervised model with the same model structure as the EBM, and is trained only on the same subset of labeled examples each time. We observe that FCE outperforms the supervised model when the number of labeled examples is small (less than 2000).

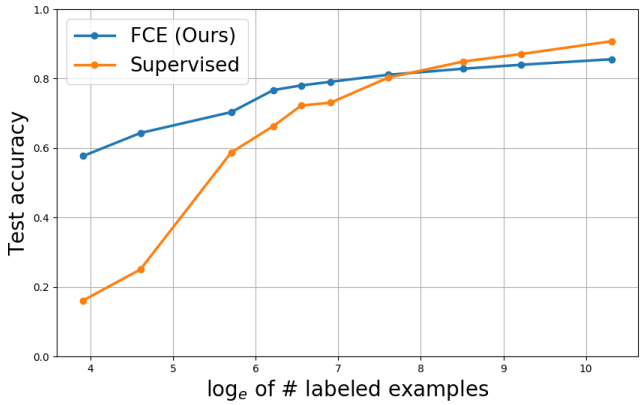


Figure 4.2.4: SVHN test-set classification accuracy as a function of number of labeled examples. The features from top layer feature maps are extracted and a linear classifier is learned on the extracted features.

Next, we try to combine features from multiple layers together. Specifically, following the same procedure outlined in [RMC15], the features from the top three convolutional layers are max pooled and concatenated to form a $14,336$-dimensional vector of feature. A regularized L2-SVM is then trained on these features with a subset of training examples and the corresponding labels.

Table 4.2.3: Test set classification error of L2-SVM classifier trained on the concatenated features learned from SVHN. DDGM stands for Deep Directed Generative Models. For fair comparison, all the energy-based models or discriminative models are trained with the same model structure.

Method	# of labeled data		
	1000	2000	4000
WGAN [ACB17a]	43.15	38.00	32.56
WGAN-GP [GAA17]	40.12	32.24	30.63
DDGM [KB16]	44.99	34.26	27.44
DCGAN [RMC15]	38.59	32.51	29.37
SN-GAN [MKK18]	40.82	31.24	28.69
MMD-GAN-rep [WSH18]	36.74	29.12	25.23
Persistent CD [Tie08]	45.74	39.47	34.18
One-step CD [Hin02b]	44.38	35.87	30.45
Multigrid sampling [GLZ18]	30.23	26.54	22.83
FCE (Ours)	**27.07**	**24.12**	**22.05**

Table 4.2.3 summarizes the results of using $1,000$, $2,000$ and $4,000$ labeled examples from the training set. At the top part of the table, we compare with methods that estimate an EBM or a discriminative model coupled with a generator network. At the middle part of the table, we compare with methods that learn an EBM with contrastive divergence (CD) and modified versions of CD. For a fair comparison, we use the same model structure for the EBMs or discriminative models used in all the methods. The results indicate that FCE outperforms these methods in terms of the effectiveness of learned features.

4.2.4.4 Semi-supervised learning

In section 4.2.3 we show that FCE can be generalized to perform semi-supervised learning. We emphasize that for semi-supervised learning, FCE not only learns a classification boundary or a

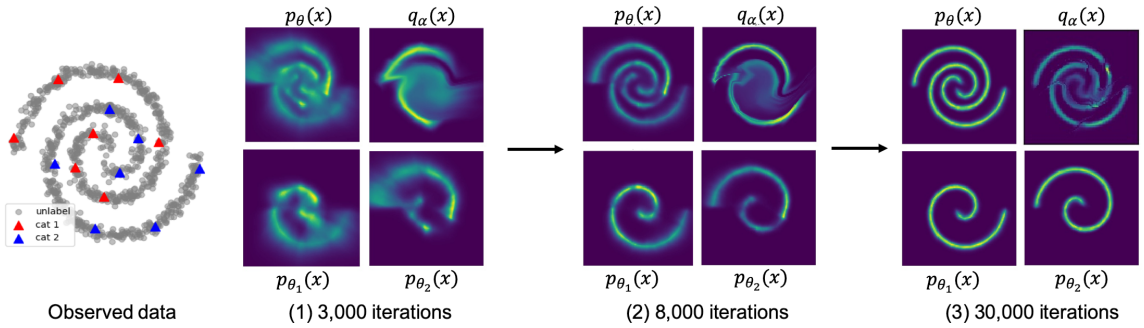


Figure 4.2.5: Illustration of FCE for semi-supervised learning on a 2D example, where the data distribution is two spirals belonging to two categories. Within each panel, the top left is the learned unconditional EBM. The top right is the learned Glow model. The bottom is two class-conditional EBMs. For observed data, seven labeled points are provided for each category.

posterior label distribution $p(y|\mathbf{x})$. Instead, the algorithm ends up with K estimated probabilistic distributions $p(\mathbf{x}|y=k), k=1,...K$ for observed examples belonging to K categories. Figure 4.2.5 illustrates this point by showing the learning process on a 2D example, where the data distribution consists of two twisted spirals belonging to two categories. Seven labeled points are provided for each category. As the training goes, the unconditional EBM $p_\theta(\mathbf{x})$ learns to capture all the modes of the data distribution, which is in the form of a mixture of class-conditional EBMs $p_{\theta_1}(\mathbf{x})$ and $p_{\theta_2}(\mathbf{x})$. Meanwhile, by maximizing the objective function $L_{\text{label}}(\theta)$ (eqn. 4.15), $p_\theta(\mathbf{x})$ is forced to project the learned modes into different spaces, resulting in two well-separated class-conditional EBMs. As shown in Figure 4.2.5, within a single mode of one category, the EBM tends to learn a smoothly connected cluster, which is often what we desire in semi-supervised learning.

Then we test the proposed method on a dataset of real images. Following the setting in [MMK18], we use two types of CNN structures (‘Conv-smallqand ‘Conv-largeq) for EBMs, which are commonly used in state-of-the-art semi-supervised learning methods. Before the joint training starts, EBMs are firstly trained for $50,000$ iterations with the Glow model fixed. In practice, this helps EBMs keep pace with the pre-trained Glow model, and equips EBMs with reasonable classification ability. We report the performance at this stage as ‘FCE-init’. Also, since virtual

adversarial training (VAT) [MMK18] has been demonstrated as an effective regularization method for semi-supervised learning, we consider adopting it as an additional loss for learning the EBMs. More specifically, the loss is defined as the robustness of the conditional label distribution around each input data point against local perturbations. 'FCE + VAT' indicates the training with VAT.

Table 4.2.4 summarizes the results of semi-supervised learning on the SVHN dataset. We report the mean error rates and standard deviations over three runs. All the methods listed in the table belong to the family of semi-supervised learning methods. Our method achieves competitive performance to these state-of-the-art methods. 'FCE + VAT' results show that the effectiveness of FCE does not overlap much with the existing semi-supervised methods, and thus they can be combined to further boost the performance.

4.2.5 Conclusion

This section explores joint training of an energy-based model with a flow-based model, by combining the representational flexibility of the energy-based model and the computational tractability of the flow-based model. We may consider the learned energy-based model as the learned representation, while the learned flow-based model as the learned computation. This method can be considered as an adaptive version of noise contrastive estimation where the noise is transformed by a flow model to make its distribution closer to the data distribution and to make it a stronger contrast to the energy-based model. Meanwhile, the flow-based model is updated adaptively through the learning process, under the same adversarial value function.

In future work, we intend to generalize the joint training method by combining the energy-based model with other normalized probabilistic models, such as auto-regressive models. We also intend to explore other joint training methods such as those based on adversarial contrastive divergence [KB16, DAB17, HNF18] or divergence triangle [HNF18].

4.3 Descriptive model with flow-based backbone

As mentioned in Chapter 2, the maximum likelihood estimation of the descriptive model, also known as the energy-based model (EBM), follows an "analysis by synthesis" scheme. In the synthesis step, synthesized examples are generated by sampling from the current model. In the analysis step, the model parameters are updated based on the statistical difference between the synthesized examples and the observed examples. The synthesis step usually requires Markov chain Monte Carlo (MCMC) sampling, such as Langevin dynamics [Lan08] or Hamiltonian Monte Carlo (HMC) [Nea11].

However, gradient-based MCMC sampling in high dimensional data space, such as image space, generally does not mix (unless we instead sample from conditional distribution as in diffusion recovery likelihood), which is a fundamental issue from a statistical perspective. Without being able to generate fair examples from the model, the estimated gradient of the maximum likelihood learning can be highly biased, and the learned model parameters can be far from the unbiased estimator given by MLE. Even if we can learn the model by other means without resorting to MCMC sampling, e.g., by flow contrastive estimation (FCE) [GH10, GNK19], it is still necessary to be able to draw fair examples from the learned model for the purpose of model checking or downstream applications based on the learned model.

To tackle this problem, we propose to learn an EBM with a flow-based model as a backbone model or base measure, so that the EBM is in the form of a correction, or an exponential tilting, of the flow-based model [NGS20]. Specifically, we learn an EBM by correcting a relatively simple flow-based model with a relatively simple energy function parameterized by a free-form convolutional neural network. We show that the resulting EBM has a particularly simple form in the space of the latent variables of the flow-based model. MCMC sampling of the EBM in the latent space, which is a simple special case of neural transport MCMC [HSD19], mixes well and is able to traverse modes in the data space. This enables proper sampling and learning of EBMs. Please refer to [NGS20] for more details.

Table 4.2.4: Semi-supervised classification error (%) on the SVHN test set. † indicates that we derive the results by running the released code. * indicates that the method uses data augmentation. The other cited results are provided by the original papers. Our results are averaged over three runs.

Method	# of labeled data	
	500	1000
SWWAE [ZMG15]		23.56
Skip DGM [MSS16]		16.61 ($\pm$0.24)
Auxiliary DGM [MSS16]		22.86
GAN with FM [SGZ16]	18.44 ($\pm$4.8)	8.11 ($\pm$1.3)
VAT-Conv-small [MMK18]		6.83 ($\pm$0.24)
on Conv-small used in [SGZ16, MMK18]		
FCE-init	9.42 ($\pm$0.24)	8.50 ($\pm$0.26)
FCE	**7.05** ($\pm$0.28)	**6.35** ($\pm$0.12)
Π model [LA16]	7.05 ($\pm$0.30)	5.43 ($\pm$0.25)
VAT-Conv-large [MMK18]	†8.98 ($\pm$0.26)	5.77 ($\pm$0.32)
Mean Teacher [TV17]	5.45 ($\pm$0.14)	5.21 ($\pm$0.21)
Π model* [LA16]	6.83 ($\pm$0.66)	4.95 ($\pm$0.26)
Temporal ensembling* [LA16]	5.12 ($\pm$0.13)	4.42 ($\pm$0.16)
on Conv-large used in [LA16, MMK18]		
FCE-init	8.86 ($\pm$0.26)	7.60 ($\pm$0.23)
FCE	6.86 ($\pm$0.18)	5.54 ($\pm$0.18)
FCE + VAT	**4.47** ($\pm$0.23)	**3.87** ($\pm$0.14)

CHAPTER 5

A representational model of grid cells

5.1 Introduction

Imagine walking in the darkness. Purely based on the sense of self-motion, one can gain a sense of self-position by integrating the self-motion - a process often referred to as path integration [Dar73, EJ04, HFM05, FBB08, MBJ06]. While the exact neural underpinning of path integration remains unclear, it has been hypothesized that the grid cells [HFM05, FHW08, YWU11, KJB12, JWM13, DBB10] in the mammalian medial entorhinal cortex (mEC) may be involved in this process [GAS18, RWP19, HBZ16]. The grid cells are so named because individual neurons exhibit striking firing patterns that form hexagonal grids when the agent (such as a rat) navigates in a 2D open field [FMW04, HFM05, FT06, BF09, SF11, BWZ07, CWZ13, AIL09, PSR13, AB20]. The grid cells also interact with the place cells in the hippocampus [OK79]. Unlike a grid cell that fires at the vertices of a lattice, a place cell often fires at a single (or a few) locations.

The purpose of this chapter is to understand how the grid cells may perform path integration calculations[1]. We study a general optimization-based representational model in which the 2D self-position is represented by a higher-dimensional vector and the 2D self-motion is represented by a transformation of the vector. The vector representation can be considered position encoding or position embedding, where the elements of the vector may be interpreted as activities of a population of grid cells. The transformation can be realized by a recurrent network that acts on the vector. Our focus is to study the properties of the transformation.

[1]The main contributions in this chapter were first described in [GXZ18a, GXW20]

Specifically, we identify two conditions for the transformation: a group representation condition and an isotropic scaling condition, under which we demonstrate that the local neighborhood around a self-position in the 2D physical space is embedded conformally as a 2D neighborhood around the vector representation of the self-position in the neural space.

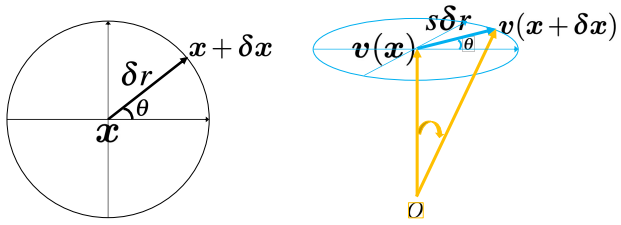
We then investigate the simplest special case of the transformation, i.e., linear transformation, that forms a matrix Lie group of rotation, under which case we show that the isotropic scaling condition is connected to the hexagonal grid patterns of the grid cells. Our numerical experiments demonstrate that our model learns clear hexagon grid patterns of multiple scales which share observed properties of the grid cells in the rodent brain, by optimizing a simple loss function. The learned model is also capable of accurate long-distance path integration.

Contributions. Our work contributes to understanding the grid cells from the perspective of representation learning. We conduct novel theoretical analysis of (1) general transformation for path integration by identifying two key conditions and a local conformal embedding property, (2) linear transformation by revealing the algebraic and geometric structure and connecting the isotropic scaling condition and the hexagon grid patterns, and (3) integration of linear transformation model and basis expansion model. Experimentally we learn clear hexagon grid patterns that are consistent with biological observations, and the learned model is capable of accurate path integration.

5.2 General transformation

5.2.1 Position embedding

Consider an agent (e.g., a rat) navigating within a 2D open field. Let $\boldsymbol{x} = (x_1, x_2)$ be the self-position of the agent. We assume that the self-position $\boldsymbol{x}$ in the 2D physical space is represented by the response activities of a population of d neurons (e.g., $d = 200$), which form a vector $\boldsymbol{v}(\boldsymbol{x}) = (v_i(\boldsymbol{x}), i = 1, ..., d)^\top$ in the d-dimensional "neural space", with each element $v_i(\boldsymbol{x})$ representing the



(a) physical space (b) neural space

Figure 5.2.1: The local 2D polar system around self-position $\boldsymbol{x}$ in the 2D physical space (a) is embedded conformally as a 2D polar system around vector $\boldsymbol{v}(\boldsymbol{x})$ in the d-dimensional neural space (b), with a scaling factor s (so that δr in the physical space becomes $s\delta r$ in the neural space).

firing rate of one neuron when the animal is at location $\boldsymbol{x}$.

$\boldsymbol{v}(\boldsymbol{x})$ can be called position encoding or position embedding. Collectively, $(\boldsymbol{v}(\boldsymbol{x}), \forall \boldsymbol{x})$ forms a *codebook* of $\boldsymbol{x} \in \mathbb{R}^2$, and $(\boldsymbol{v}(\boldsymbol{x}), \forall \boldsymbol{x})$ is a *2D manifold* in the d-dimensional neural space, i.e., globally we embed $\mathbb{R}^2$ as a 2D manifold in the neural space. Locally, we identify two conditions under which the 2D local neighborhood around $\boldsymbol{x}$ is embedded *conformally* as a 2D neighborhood around $\boldsymbol{v}(\boldsymbol{x})$ with a scaling factor. See Fig. 5.2.1. As shown in Section 5.3.3, the conformal embedding is connected to the hexagon grid patterns.

5.2.2 Transformation and path integration

At self-position $\boldsymbol{x}$, if the agent makes a self-motion $\Delta \boldsymbol{x} = (\Delta x_1, \Delta x_2)$, then it moves to $\boldsymbol{x} + \Delta \boldsymbol{x}$. Correspondingly, the vector representation $\boldsymbol{v}(\boldsymbol{x})$ is transformed to $\boldsymbol{v}(\boldsymbol{x} + \Delta \boldsymbol{x})$. The general form of the transformation can be formulated as:

$$\boldsymbol{v}(\boldsymbol{x} + \Delta \boldsymbol{x}) = F(\boldsymbol{v}(\boldsymbol{x}), \Delta \boldsymbol{x}). \tag{5.1}$$

The transformation $F(\cdot, \Delta \boldsymbol{x})$ can be considered a representation of $\Delta \boldsymbol{x}$, which forms a 2D additive group. We call Eq. (5.1) the *transformation model*. It can be implemented by a recurrent network to derive a path integration model: if we start from $\boldsymbol{x}_0$, and make a sequence of moves $(\Delta \boldsymbol{x}_t, t = 1, ..., T)$, then the vector is updated by $\boldsymbol{v}_t = F(\boldsymbol{v}_{t-1}, \Delta \boldsymbol{x}_t)$, where $\boldsymbol{v}_0 = \boldsymbol{v}(\boldsymbol{x}_0)$, and $t = 1, ..., T$.

5.2.3 Group representation condition

The solution to the transformation model (Eq. (5.1)) should satisfy the following condition.

Condition 1. *(Group representation condition)* $(\boldsymbol{v}(\boldsymbol{x}), \forall \boldsymbol{x})$ *and* $(F(\boldsymbol{v}, \Delta\boldsymbol{x}), \forall \Delta\boldsymbol{x})$ *form a representation of the 2D additive Euclidean group* $\mathbb{R}^2$ *in the sense that*

$$F(\boldsymbol{v}(\boldsymbol{x}), \Delta\boldsymbol{x}_1 + \Delta\boldsymbol{x}_2) = F(F(\boldsymbol{v}(\boldsymbol{x}), \Delta\boldsymbol{x}_1), \Delta\boldsymbol{x}_2), \tag{5.2}$$

for any $\Delta\boldsymbol{x}_1$ *and* $\Delta\boldsymbol{x}_2$*, and for any* $\boldsymbol{x}$*.*

The reason is that the agent can move in one step by $\Delta\boldsymbol{x}_1 + \Delta\boldsymbol{x}_2$, or first move by $\Delta\boldsymbol{x}_1$, and then move by $\Delta\boldsymbol{x}_2$. Both paths would end up at the same $\boldsymbol{x} + \Delta\boldsymbol{x}_1 + \Delta\boldsymbol{x}_2$, which is represented by the same $\boldsymbol{v}(\boldsymbol{x} + \Delta\boldsymbol{x}_1 + \Delta\boldsymbol{x}_2)$. It is a necessary self-consistent condition for the transformation model (Eq. (5.1)).

5.2.4 Egocentric self-motion

Self-motion can also parametrized egocentrically as $(\Delta r, \theta)$, where Δr is the displacement along the direction $\theta \in [0, 2\pi]$, so that $\Delta\boldsymbol{x} = (\Delta x_1 = \Delta r \cos\theta, \Delta x_2 = \Delta r \sin\theta)$. The egocentric self-motion could be more biologically plausible where θ is encoded by head direction, and Δr can be interpreted as the speed along direction θ. The transformation model then becomes

$$\boldsymbol{v}(\boldsymbol{x} + \Delta\boldsymbol{x}) = F(\boldsymbol{v}(\boldsymbol{x}), \Delta r, \theta), \tag{5.3}$$

where we continue to use $F(\cdot)$ for the transformation (with slight abuse of notation). $(\Delta r, \theta)$ form a polar coordinate system around $\boldsymbol{x}$.

5.2.5 Infinitesimal self-motion and directional derivative

In this subsection, we derive the transformation model for infinitesimal self-motion. While we use $\Delta\boldsymbol{x}$ or Δr to denote finite (non-infinitesimal) self-motion, we use $\delta\boldsymbol{x}$ or δr to denote infinitesimal

self-motion. At self-position $\boldsymbol{x}$, for an infinitesimal displacement δr along direction θ, $\delta \boldsymbol{x} = (\delta x_1 = \delta r \cos\theta, \delta x_2 = \delta r \sin\theta)$. See Fig. 5.2.1 (a) for an illustration. Given that δr is infinitesimal, a first order Taylor expansion of $F(\boldsymbol{v}(\boldsymbol{x}), \delta r, \theta)$ with respect to δr gives us

$$\begin{aligned} \boldsymbol{v}(\boldsymbol{x}+\delta \boldsymbol{x}) &= F(\boldsymbol{v}(\boldsymbol{x}), \delta r, \theta) = F(\boldsymbol{v}(\boldsymbol{x}), 0, \theta) + F'(\boldsymbol{v}(\boldsymbol{x}), 0, \theta)\delta r + o(\delta r) \\ &= \boldsymbol{v}(\boldsymbol{x}) + f_\theta(\boldsymbol{v}(\boldsymbol{x}))\delta r + o(\delta r), \end{aligned} \tag{5.4}$$

where $F(\boldsymbol{v}(\boldsymbol{x}), 0, \theta) = \boldsymbol{v}(\boldsymbol{x})$ because zero displacement does not change $\boldsymbol{v}(\boldsymbol{x})$, and $f_\theta(\boldsymbol{v}(\boldsymbol{x})) \coloneqq F'(\boldsymbol{v}(\boldsymbol{x}), 0, \theta)$ is the first derivative of $F(\boldsymbol{v}(\boldsymbol{x}), \Delta r, \theta)$ with respect to Δr at $\Delta r = 0$. $f_\theta(\boldsymbol{v}(\boldsymbol{x}))$ is the *directional derivative* of $F(\cdot)$ at self-position $\boldsymbol{x}$ and direction θ.

5.2.6 Isotropic scaling condition

With the directional derivative, we define the second condition as follows, which leads to locally conformal embedding and is connected to hexagon grid pattern.

Condition 2. *(Isotropic scaling condition) For any fixed $\boldsymbol{x}$, $\|f_\theta(\boldsymbol{v}(\boldsymbol{x}))\|$ is constant over θ.*

Let $f_0(\boldsymbol{v}(\boldsymbol{x}))$ denote $f_\theta(\boldsymbol{v}(\boldsymbol{x}))$ for $\theta = 0$, and $f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))$ denote $f_\theta(\boldsymbol{v}(\boldsymbol{x}))$ for $\theta = \pi/2$. Then we have the following theorem: .

Theorem 1. *Assume group representation condition 1 and isotropic scaling condition 2. At any fixed $\boldsymbol{x}$, for the local motion $\delta \boldsymbol{x} = (\delta r \cos\theta, \delta r \sin\theta)$ around $\boldsymbol{x}$, let $\delta \boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x}+\delta \boldsymbol{x}) - \boldsymbol{v}(\boldsymbol{x})$ be the change of vector and $s = \|f_\theta(\boldsymbol{v}(\boldsymbol{x}))\|$, then we have $\|\delta \boldsymbol{v}\| = s\|\delta \boldsymbol{x}\|$. Moreover,*

$$\delta \boldsymbol{v} = f_\theta(\boldsymbol{v}(\boldsymbol{x}))\delta r + o(\delta r) = f_0(\boldsymbol{v}(\boldsymbol{x}))\delta r \cos\theta + f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))\delta r \sin\theta + o(\delta r), \tag{5.5}$$

where $f_0(\boldsymbol{v}(\boldsymbol{x}))$ and $f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))$ are two orthogonal basis vectors of equal norm s.

See chapter appendix for a proof and Fig. 5.2.1(b) for an illustration. Theorem 1 indicates that the local 2D polar system around self-position $\boldsymbol{x}$ in the 2D physical space is embedded conformally as a 2D polar system around vector $\boldsymbol{v}(\boldsymbol{x})$ in the d-dimensional neural space, with a scaling factor s (our analysis is local for any fixed $\boldsymbol{x}$, and s may depend on $\boldsymbol{x}$).

Why isotropic scaling and conformal embedding? The neurons are intrinsically noisy. During path integration, the errors may accumulate in $\boldsymbol{v}$. Moreover, when inferring self-position from a visual image, it is possible the $\boldsymbol{v}$ is inferred first with error, and then $\boldsymbol{x}$ is decoded from the inferred $\boldsymbol{v}$. Due to isotropic scaling and conformal embedding, locally we have $\|\delta \boldsymbol{v}\| = s\|\delta \boldsymbol{x}\|$, which guarantees that the ℓ_2 error in $\boldsymbol{v}$ translates proportionally to the ℓ_2 error in $\boldsymbol{x}$, so that there will not be adversarial perturbations in $\boldsymbol{v}(\boldsymbol{x})$ that cause excessively big errors in $\boldsymbol{x}$. Section 5.3.2 presents explicit analysis on error correction.

5.3 Linear transformation

After studying the general transformation, we now investigate the linear transformation of $\boldsymbol{v}(\boldsymbol{x})$, for the following reasons. (1) It is the simplest transformation for which we can derive explicit algebraic and geometric results. (2) It enables us to connect the isotropic scaling condition to hexagon grid patterns. (3) In Section 5.4, we integrate it with the basis expansion model, which is also linear in $\boldsymbol{v}(\boldsymbol{x})$.

For finite (non-infinitesimal) self-motion, the linear transformation model is:

$$\boldsymbol{v}(\boldsymbol{x}+\Delta \boldsymbol{x}) = F(\boldsymbol{v}(\boldsymbol{x}), \Delta \boldsymbol{x}) = \boldsymbol{M}(\Delta \boldsymbol{x})\boldsymbol{v}(\boldsymbol{x}), \tag{5.6}$$

where $\boldsymbol{M}(\Delta \boldsymbol{x})$ is a matrix. The group representation condition becomes

$$\boldsymbol{M}(\Delta \boldsymbol{x}_1 + \Delta \boldsymbol{x}_2) = \boldsymbol{M}(\Delta \boldsymbol{x}_2)\boldsymbol{M}(\Delta \boldsymbol{x}_1), \tag{5.7}$$

i.e., $\boldsymbol{M}(\Delta \boldsymbol{x})$ is a matrix representation of self-motion $\Delta \boldsymbol{x}$. For egocentric parametrization of self-motion $(\Delta r, \theta)$, we can further write $\boldsymbol{M}(\Delta \boldsymbol{x}) = \boldsymbol{M}_\theta(\Delta r)$ for $\Delta \boldsymbol{x} = (\Delta r \cos\theta, \Delta r \sin\theta)$, and the linear model becomes $\boldsymbol{v}(\boldsymbol{x}+\Delta \boldsymbol{x}) = F(\boldsymbol{v}(\boldsymbol{x}), \Delta r, \theta) = \boldsymbol{M}_\theta(\Delta r)\boldsymbol{v}(\boldsymbol{x})$.

5.3.1 Algebraic structure: matrix Lie algebra and Lie group

For the linear model (Eq. (5.6)), the directional derivative can be expressed as: $f_\theta(\boldsymbol{v}(\boldsymbol{x})) = F'(\boldsymbol{v}(\boldsymbol{x}), 0, \theta) = \boldsymbol{B}(\theta)\boldsymbol{v}(\boldsymbol{x})$, where $\boldsymbol{B}(\theta)$ is a $d \times d$ matrix only depending on the direction θ.

For infinitesimal self-motion, the transformation model in Eq. (5.4) becomes

$$\boldsymbol{v}(\boldsymbol{x}+\delta\boldsymbol{x}) = (\boldsymbol{I}+\boldsymbol{B}(\theta)\delta r)\boldsymbol{v}(\boldsymbol{x})+o(\delta r), \tag{5.8}$$

where $\boldsymbol{I}$ is the identity matrix. It can be considered a linear recurrent network where $\boldsymbol{B}(\theta)$ is the learnable weight matrix. We have the following theorem for the algebraic structure of the linear transformation.

Theorem 2. *Assume the linear transformation model so that for infinitesimal self-motion $(\delta r, \theta)$, the model is in the form of Eq. (5.8), then for finite displacement Δr,*

$$\boldsymbol{v}(\boldsymbol{x}+\Delta\boldsymbol{x}) = \boldsymbol{M}_\theta(\Delta r)\boldsymbol{v}(\boldsymbol{x}) = \exp(\boldsymbol{B}(\theta)\Delta r)\boldsymbol{v}(\boldsymbol{x}). \tag{5.9}$$

Proof. We can divide Δr into N steps, so that $\delta r = \Delta r/N \to 0$ as $N \to \infty$, and

$$\boldsymbol{v}(\boldsymbol{x}+\Delta\boldsymbol{x}) = (\boldsymbol{I}+\boldsymbol{B}(\theta)(\Delta r/N)+o(1/N))^N\boldsymbol{v}(\boldsymbol{x}) \to \exp(\boldsymbol{B}(\theta)\Delta r)\boldsymbol{v}(\boldsymbol{x}) \tag{5.10}$$

as $N \to \infty$. The matrix exponential map is defined by $\exp(A) = \sum_{n=0}^{\infty} A^n/n!$. □

The above math underlies the relationship between matrix Lie algebra and matrix Lie group [Tay02]. For a fixed θ, the set of $\boldsymbol{M}_\theta(\Delta r) = \exp(\boldsymbol{B}(\theta)\Delta r)$ for $\Delta r \in \mathbb{R}$ forms a *matrix Lie group*, which is both a group and a manifold. The tangent space of $\boldsymbol{M}_\theta(\Delta r)$ at identity $\boldsymbol{I}$ is called *matrix Lie algebra*. $\boldsymbol{B}(\theta)$ is the basis of this tangent space, and is often referred to as the *generator matrix*.

Path integration. The path integration model then becomes as follows. If the agent starts from $\boldsymbol{x}_0$, and make a sequence of moves $((\Delta r_t, \theta_t), t = 1, ..., T)$, then the vector representation of self-position is updated by

$$\boldsymbol{v}_t = \exp(\boldsymbol{B}(\theta_t)\Delta r_t)\boldsymbol{v}_{t-1}, \tag{5.11}$$

where $\boldsymbol{v}_0 = \boldsymbol{v}(\boldsymbol{x}_0)$, and $t = 1, ..., T$.

Approximation to exponential map. For a finite but small Δr, $\exp(\boldsymbol{B}(\theta)\Delta r)$ can be approximated by a second-order (or higher-order) Taylor expansion

$$\exp(\boldsymbol{B}(\theta)\Delta r) = \boldsymbol{I}+\boldsymbol{B}(\theta)\Delta r+\boldsymbol{B}(\theta)^2\Delta r^2/2+o(\Delta r^2). \tag{5.12}$$

5.3.2 Geometric structure: rotation, periodicity, metic and error correction

If we assume $\boldsymbol{B}(\theta) = -\boldsymbol{B}(\theta)^\top$, i.e., skew-symmetric, then $\boldsymbol{I} + \boldsymbol{B}(\theta)\delta r$ in Eq. (5.8) is approximately a rotation matrix operating on $\boldsymbol{v}(\boldsymbol{x})$, due to the fact that $(\boldsymbol{I} + \boldsymbol{B}(\theta)\delta r)(\boldsymbol{I} + \boldsymbol{B}(\theta)\delta r)^\top = \boldsymbol{I} + O(\delta r^2)$. For finite Δr, $\exp(\boldsymbol{B}(\theta)\Delta r)$ is also a rotation matrix, as it equals to the product of N matrices $\boldsymbol{I} + \boldsymbol{B}(\theta)(\Delta r/N)$ (Eq. (5.10)). The geometric interpretation is that, if the agent moves along the direction θ in the physical space, the vector $\boldsymbol{v}(\boldsymbol{x})$ is rotated by the matrix $\boldsymbol{B}(\theta)$ in the neural space, while the ℓ_2 norm $\|\boldsymbol{v}(\boldsymbol{x})\|^2$ remains fixed. We may interpret $\|\boldsymbol{v}(\boldsymbol{x})\|^2 = \sum_{i=1}^d v_i(\boldsymbol{x})^2$ as the total energy of grid cells. See Fig. 5.2.1(b).

The angle of rotation is given by $\|\boldsymbol{B}(\theta)\boldsymbol{v}(\boldsymbol{x})\|\delta r/\|\boldsymbol{v}(\boldsymbol{x})\|$, because $\|\boldsymbol{B}(\theta)\boldsymbol{v}(\boldsymbol{x})\|\delta r$ is the arc length and $\|\boldsymbol{v}(\boldsymbol{x})\|$ is the radius. If we further assume the isotropic scaling condition, which becomes that $\|f_\theta(\boldsymbol{v}(\boldsymbol{x}))\| = \|\boldsymbol{B}(\theta)\boldsymbol{v}(\boldsymbol{x})\|$ is constant over θ for the linear model, then the angle of rotation can be written as $\mu\delta r$, where $\mu = \|\boldsymbol{B}(\theta)\boldsymbol{v}(\boldsymbol{x})\|/\|\boldsymbol{v}(\boldsymbol{x})\|$ is independent of θ. Geometrically, μ tells us how fast the vector rotates in the neural space as the agent moves in the physical space. In practice, μ can be much bigger than 1 for the learned model, thus the vector rotates back to itself in a short distance, causing the periodic patterns in the elements of $\boldsymbol{v}(\boldsymbol{x})$. μ captures the notion of metric.

For $\mu \gg 1$, the conformal embedding in Fig. 5.2.1 (b) **magnifies** the local motion in Fig. 5.2.1 (a), and this enables error correction [SF11]. More specifically, we have the following explicit result.

Theorem 3. *Assume the linear transformation model (Eq. (5.8)) and the isotropic scaling condition 2. For any fixed $\boldsymbol{x}$, let $\mu = \|\boldsymbol{B}(\theta)\boldsymbol{v}(\boldsymbol{x})\|/\|\boldsymbol{v}(\boldsymbol{x})\|$. Suppose the neurons are noisy: $\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \tau^2 \boldsymbol{I}_d)$ and $\tau^2 = \alpha^2(\|\boldsymbol{v}(\boldsymbol{x})\|^2/d)$, so that α^2 measures the variance of noise relative to the average magnitude of $(v_i(\boldsymbol{x})^2, i = 1, ..., d)$. Suppose the agent infers its 2D position $\hat{\boldsymbol{x}}$ from $\boldsymbol{v}$ by $\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}'} \|\boldsymbol{v} - \boldsymbol{v}(\boldsymbol{x}')\|^2$, i.e., $\boldsymbol{v}(\hat{\boldsymbol{x}})$ is the projection of $\boldsymbol{v}$ onto the 2D manifold formed by $(\boldsymbol{v}(\boldsymbol{x}'), \forall \boldsymbol{x}')$. Then we have*

$$E\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|^2 = 2\alpha^2/(\mu^2 d). \tag{5.13}$$

See chapter appendix for proof. A similar result can be obtained for the general transformation model. By the above theorem, error correction of grid cells is due to two factors: (1) higher dimensionality d of $\boldsymbol{v}(\boldsymbol{x})$ for encoding 2D $\boldsymbol{x}$, and (2) a magnifying $\mu \gg 1$ (our analysis is local for any fixed $\boldsymbol{x}$, and μ may depend on $\boldsymbol{x}$).

5.3.3 Isotropic scaling condition and hexagon grid patterns

In this subsection, we make the connection between the isotropic scaling condition 2 in the linear model and the emergence of hexagon grid patterns, which can be created by linearly mixing three Fourier plane waves whose directions are $2\pi/3$ apart. We show such linear mixing satisfies the linear model and the isotropic scaling condition.

Theorem 4. *Let* $\boldsymbol{e}(\boldsymbol{x}) = (\exp(i\langle \boldsymbol{a}_j, \boldsymbol{x}\rangle), j = 1,2,3)^\top$*, where* $(\boldsymbol{a}_j, j = 1,2,3)$ *are three 2D vectors of equal norm, and the angle between every pair of them is* $2\pi/3$*. Let* $\boldsymbol{v}(\boldsymbol{x}) = \boldsymbol{U}\boldsymbol{e}(\boldsymbol{x})$*, where* $\boldsymbol{U}$ *is an arbitrary unitary matrix. Let* $\boldsymbol{B}(\theta) = \boldsymbol{U}^* \boldsymbol{D}(\theta)\boldsymbol{U}$*, where* $\boldsymbol{D}(\theta) = diag(i\langle \boldsymbol{a}_j, \boldsymbol{q}(\theta)\rangle, j = 1,2,3)$*, with* $\boldsymbol{q}(\theta) = (\cos\theta, \sin\theta)^\top$*. Then* $(\boldsymbol{v}(\boldsymbol{x}), \boldsymbol{B}(\theta))$ *satisfies the linear transformation model (Eq. (5.8)) and the isotropic scaling condition 2. Moreover,* $\boldsymbol{B}(\theta)$ *is skew-symmetric.*

See chapter appendix for proof. We would like to emphasize that the above theorem is to give a solution to our model, but our optimization-based model **does not assume any superposition of Fourier basis functions** as in the theorem. Our experimental results are learned purely by optimizing a loss function based on the simple assumptions of our model with generic vectors and matrices.

5.3.4 Modules

Biologically, it is well established that grid cells are organized in discrete modules [BHB07, SSS12] or blocks. We thus partition the vector $\boldsymbol{v}(\boldsymbol{x})$ into K blocks, $\boldsymbol{v}(\boldsymbol{x}) = (\boldsymbol{v}_k(\boldsymbol{x}), k = 1, ..., K)$. Correspondingly the generator matrices $\boldsymbol{B}(\theta) = \text{diag}(\boldsymbol{B}_k(\theta), k = 1, ..., K)$ are block diagonal, so that each sub-vector $\boldsymbol{v}_k(\boldsymbol{x})$ is rotated by a sub-matrix $\boldsymbol{B}_k(\theta)$. For the general transformation model,

each sub-vector is transformed by a separate sub-network. By the same argument as in subsection 5.3.2, let $\mu_k = \|\boldsymbol{B}_k \boldsymbol{v}_k(\boldsymbol{x})\|/\|\boldsymbol{v}_k(\boldsymbol{x})\|$, then μ_k is the metric of module k.

5.4 Interaction with place cells

5.4.1 Place cells

For each $\boldsymbol{v}(\boldsymbol{x})$, we need to uniquely decode $\boldsymbol{x}$ globally. This can be accomplished with place cells. Specifically, each place cell fires when the agent is at a specific position. Let $A(\boldsymbol{x}, \boldsymbol{x}')$ be the response map for the place cell associated with position $\boldsymbol{x}'$. It measures the adjacency between $\boldsymbol{x}$ and $\boldsymbol{x}'$. A commonly used form of $A(\boldsymbol{x}, \boldsymbol{x}')$ is the Gaussian adjacency kernel $A(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2/(2\sigma^2))$.

5.4.2 Basis expansion

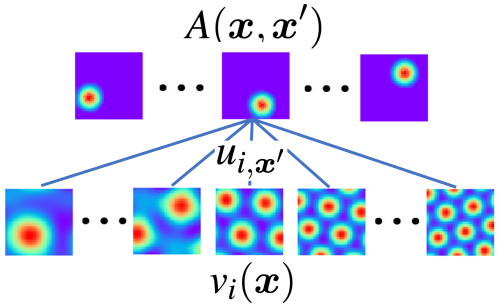


Figure 5.4.1: Illustration of basis expansion model $A(\boldsymbol{x}, \boldsymbol{x}') = \sum_{i=1}^{d} u_{i,\boldsymbol{x}'} v_i(\boldsymbol{x})$, where $v_i(\boldsymbol{x})$ is the response map of i-th grid cell, shown at the bottom, which shows 5 different i. $A(\boldsymbol{x}, \boldsymbol{x}')$ is the response map of place cell associated with $\boldsymbol{x}'$, shown at the top, which shows 3 different $\boldsymbol{x}'$. $u_{i,\boldsymbol{x}'}$ is the connection weight.

A popular model that connects place cells and grid cells is the following basis expansion model (or PCA-based model) [DSM16]:

$$A(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{v}(\boldsymbol{x}), \boldsymbol{u}(\boldsymbol{x}') \rangle = \sum_{i=1}^{d} u_{i,\boldsymbol{x}'} v_i(\boldsymbol{x}), \tag{5.14}$$

where $\boldsymbol{v}(\boldsymbol{x}) = (v_i(\boldsymbol{x}), i = 1, ..., d)^\top$, and $\boldsymbol{u}(\boldsymbol{x}') = (u_{i,\boldsymbol{x}'}, i = 1, ..., d)^\top$. Here $(v_i(\boldsymbol{x}), i = 1, ..., d)$ forms a set of d basis functions (which are functions of $\boldsymbol{x}$) for expanding $A(\boldsymbol{x}, \boldsymbol{x}')$ (which is a function of $\boldsymbol{x}$ for each place $\boldsymbol{x}'$), while $\boldsymbol{u}(\boldsymbol{x}')$ is the read-out weight vector for place cell at $\boldsymbol{x}'$, and needs to be learned. See Fig. 5.4.1 for an illustration. Experimental results on biological brains have shown that the connections from grid cells to place cells are excitatory [ZYM13, ROS18]. We thus assume that $u_{i,\boldsymbol{x}'} \geq 0$ for all i and $\boldsymbol{x}'$.

5.4.3 From group representation to basis functions

The vector representation $\boldsymbol{v}(\boldsymbol{x})$ generated (or constrained) by the linear transformation model (Eq. (5.6)) can serve as basis functions of the PCA-based basis expansion model (Eq. (5.14)), due to the fundamental theorems of Schur [Zee16] and Peter-Weyl [Tay02], which reveal the root of Fourier analysis and generalize it to general groups. Specifically, if $\boldsymbol{M}(\Delta\boldsymbol{x})$ is an irreducible unitary representation of $\Delta\boldsymbol{x}$ that forms a compact Lie group, then the elements $\{M_{ij}(\Delta\boldsymbol{x})\}$ form a set of orthogonal basis functions of $\Delta\boldsymbol{x}$. Let $\boldsymbol{v}(\boldsymbol{x}) = \boldsymbol{M}(\boldsymbol{x})\boldsymbol{v}(0)$ (where we choose the origin 0 as the reference point). The elements of $\boldsymbol{v}(\boldsymbol{x})$, i.e., $(v_i(\boldsymbol{x}), i = 1, ..., d)$, are linear mixings of the basis functions $\{M_{ij}(\boldsymbol{x})\}$, so that they themselves form a new set of basis functions that serve to expand $(A(\boldsymbol{x}, \boldsymbol{x}'), \forall \boldsymbol{x}')$ that parametrizes the place cells.

The basis expansion model (or PCA-based model) (Eq. 5.14) assumes that the basis functions are orthogonal, while in our work, **we do not make the orthogonality assumption**. Interestingly, the learned transformation model generates basis functions that are close to being orthogonal automatically. See chapter appendix for more detailed explanation and experimental result.

5.4.4 Decoding and re-encoding

For a neural response vector $\boldsymbol{v}$, such as $\boldsymbol{v}_t$ in Eq. (5.11), the response of the place cell associated with location $\boldsymbol{x}'$ is $\langle \boldsymbol{v}, \boldsymbol{u}(\boldsymbol{x}') \rangle$. We can decode the position $\hat{\boldsymbol{x}}$ by examining which place cell has

the maximal response, i.e.,

$$\hat{\boldsymbol{x}} = \arg\max_{\boldsymbol{x}'} \langle \boldsymbol{v}, \boldsymbol{u}(\boldsymbol{x}') \rangle. \tag{5.15}$$

After decoding $\hat{\boldsymbol{x}}$, we can re-encode $\boldsymbol{v} \leftarrow \boldsymbol{v}(\hat{\boldsymbol{x}})$ for error correction. Decoding and re-encoding can also be done by directly projecting $\boldsymbol{v}$ onto the manifold $(\boldsymbol{v}(\boldsymbol{x}), \forall \boldsymbol{x})$, which gives similar results. See chapter appendix for more analysis and experimental results.

5.5 Learning

We learn the model by optimizing a loss function defined based on three model assumptions discussed above: (1) the basis expansion model (Eq. (5.14)), (2) the linear transformation model (Eq. (5.9)) and (3) the isotropic scaling condition 2. The input is the set of adjacency kernels $A(\boldsymbol{x}, \boldsymbol{x}'), \forall \boldsymbol{x}, \boldsymbol{x}'$. The unknown parameters to be learned are (1) $(\boldsymbol{v}(\boldsymbol{x}) = (\boldsymbol{v}_k(\boldsymbol{x}), k = 1, ..., K), \forall \boldsymbol{x})$, (2) $(\boldsymbol{u}(\boldsymbol{x}'), \forall \boldsymbol{x}')$ and (3) $(\boldsymbol{B}(\theta), \forall \theta)$. We assume that there are K modules or blocks and $\boldsymbol{B}(\theta)$ is skew-symmetric, so that $\boldsymbol{B}(\theta)$ are parametrized as block-diagonal matrices $(\boldsymbol{B}_k(\theta), k = 1, ..., K), \forall \theta)$ and only the lower triangle parts of the matrices need to be learned. The loss function is defined as a weighted sum of simple ℓ_2 loss terms constraining the three model assumptions: $L = L_0 + \lambda_1 L_1 + \lambda_2 L_2$, where

$$L_0 = \mathrm{E}_{\boldsymbol{x}, \boldsymbol{x}'}[A(\boldsymbol{x}, \boldsymbol{x}') - \langle \boldsymbol{v}(\boldsymbol{x}), \boldsymbol{u}(\boldsymbol{x}') \rangle]^2, \text{ (basis expansion)} \tag{5.16}$$

$$L_1 = \sum_{k=1}^{K} \mathrm{E}_{\boldsymbol{x}, \Delta \boldsymbol{x}} \| \boldsymbol{v}_k(\boldsymbol{x} + \Delta \boldsymbol{x}) - \exp(\boldsymbol{B}_k(\theta) \Delta r) \boldsymbol{v}_k(\boldsymbol{x}) \|^2, \text{ (transformation)} \tag{5.17}$$

$$L_2 = \sum_{k=1}^{K} \mathrm{E}_{\boldsymbol{x}, \theta, \Delta \theta} [\| \boldsymbol{B}_k(\theta + \Delta \theta) \boldsymbol{v}_k(\boldsymbol{x}) \| - \| \boldsymbol{B}_k(\theta) \boldsymbol{v}_k(\boldsymbol{x}) \|]^2. \text{ (isotropic scaling)} \tag{5.18}$$

λ_1 and λ_2 are chosen so that the three loss terms are of similar magnitudes. $A(\boldsymbol{x}, \boldsymbol{x}')$ are given as Gaussian adjacency kernels. For regularization, we add a penalty on $\|\boldsymbol{u}(\boldsymbol{x}')\|^2$, and further assume $\boldsymbol{u}(\boldsymbol{x}') \geq 0$ so that the connections from grid cells to place cells are excitatory [ZYM13, ROS18]. However, note that $\boldsymbol{u}(\boldsymbol{x}') \geq 0$ is not necessary for the emergence of hexagon grid patterns as shown in the ablation studies.

Expectations in L_0, L_1 and L_2 are approximated by Monte Carlo samples. L is minimized by *Adam* [KB14] optimizer. See chapter appendix for implementation details.

5.6 Experiments

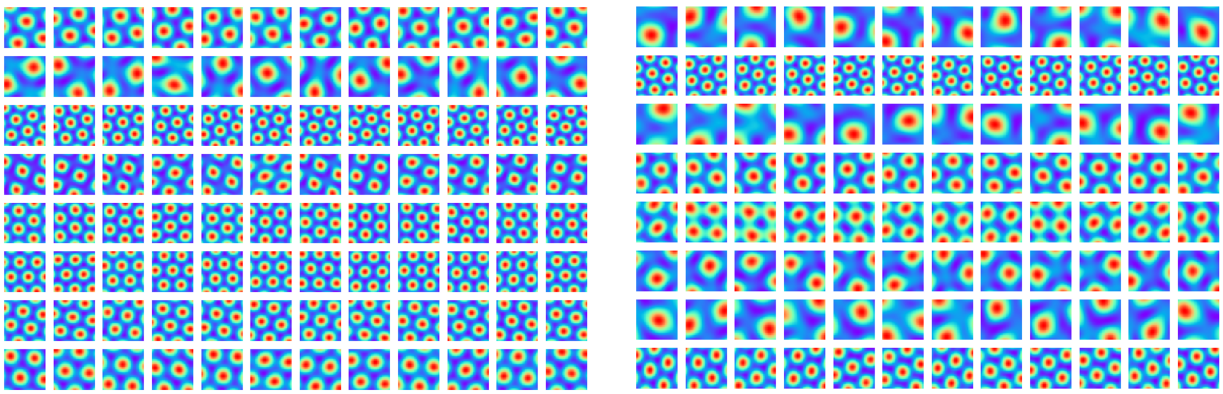


Figure 5.6.1: Hexagonal grid firing patterns emerge in the learned network. Every response map shows the firing pattern of one neuron (i.e, one element of $\boldsymbol{v}$) in the 2D environment. Every row shows the firing patterns of the neurons within the same block or module.

We conduct numerical experiments to learn the representations as described in Section 5.5. Specifically, we use a square environment with size 1m $\times$ 1m, which is discretized into a 40×40 lattice. For direction, we discretize the circle $[0, 2\pi]$ into 144 directions and use nearest neighbor linear interpolations for values in between. We use the second-order Taylor expansion (5.12) to approximate the exponential map $\exp(\boldsymbol{B}(\theta)\Delta r)$. The self displacement Δr are sampled within a small range, i.e., Δr is smaller than 3 grids on the lattice. For $A(\boldsymbol{x}, \boldsymbol{x}')$, we use a Gaussian adjacency kernel with $\sigma = 0.07$. $\boldsymbol{v}(\boldsymbol{x})$ is of $d = 192$ dimensions, which is partitioned into $K = 16$ modules, each of which has 12 cells.

5.6.1 Hexagon grid patterns

Fig. 5.B.10 shows the learned firing patterns of $\boldsymbol{v}(\boldsymbol{x}) = (v_i(\boldsymbol{x}), i = 1, ..., d)$ over the 40×40 lattice of $\boldsymbol{x}$. Every row shows the learned units belonging to the same block or module. Regular hexagon grid patterns emerge. Within each block or module, the scales and orientations are roughly the same, but with different phases or spatial shifts. For the learned $\boldsymbol{B}(\theta)$, each element shows regular sine/cosine tuning over θ. See to chapter appendix for more learned patterns.

We further investigate the characteristics of the learned firing rate patterns (i.e., $\boldsymbol{v}(\boldsymbol{x})$) using measures adopted from the grid cell literature. Specifically, the hexagonal regularity, scale and orientation of grid-like patterns are quantified using the gridness score, grid scale and grid orientation [LAC10, SFH06], which are determined by taking a circular sample of the autocorrelogram of the response map. Table 5.6.1 summarizes the results of gridness scores and comparisons with other optimization-based approaches [BBU18, SMG19]. We apply the same threshold to determine whether a learned neuron can be considered a grid cell as in [BBU18] (i.e., gridness score ¿ 0.37). For our model, 73.10% of the learned neurons exhibit significant hexagonal periodicity in terms of gridness scores. Fig. 5.6.2 shows the histogram of grid scales of learned grid cell neurons (mean 0.33, range 0.21 to 0.49), which follows a multi-modal distribution. The ratio between neighboring modes is roughly 1.52 and 1.51, which closely match the theoretical predictions [WPB15, SMH15] and also the empirical results from rodent grid cells [SSS12]. Collectively, these results reveal striking, quantitative correspondence between the properties of our model neurons and those of the grid cells in the brain.

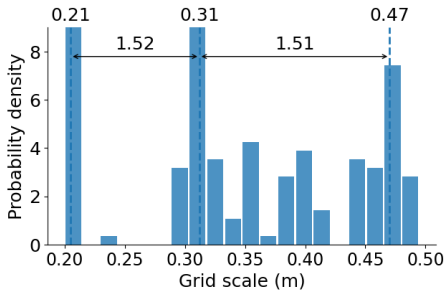


Figure 5.6.2: Multi-modal distribution of grid scales of the learned model grid cells. The scale ratios closely match the real data [SSS12].

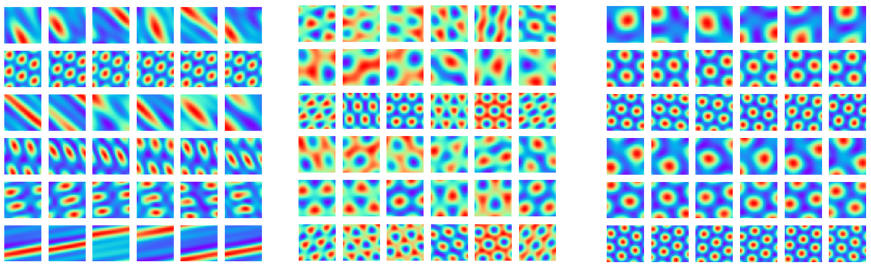
Table 5.6.1: Summary of gridness scores of the units learned from different models. To determine the valid grid cells, we apply the same threshold of gridness score as in [BBU18], i.e., gridness score ¿ 0.37. For our model, we run 5 trials and report the average and standard deviation.

Model	Gridness score	% of grid cells
[BBU18] (LSTM)	0.18	25.20
[SMG19] (RNN)	0.48	56.10
Ours	**0.90** $\pm$ 0.044	**73.10** $\pm$ 1.33

Ablation studies. We conduct ablation studies to examine whether certain model assumptions are empirically important for the emergence of hexagon grid patterns. The conclusions are highlighted as follows: (1) The loss term L_2 in Eq. (5.18) for isotropic scaling is necessary for learning hexagon grid patterns. (2) The constraint $\boldsymbol{u}(\boldsymbol{x}') \geq 0$ is not necessary for learning hexagon patterns, but the activations can be either excitatory or inhibitory without the constraint. (3) The skew-symmetric assumption on $\boldsymbol{B}(\boldsymbol{\theta})$ is not important for learning hexagon grid patterns. (4) Hexagon patterns always emerge regardless of the choice of block size and number of blocks. (5) Multiple blocks or modules are necessary for the emergence of hexagon grid patterns of multiple scales. See Fig. 5.6.3 for several learned patterns and chapter appendix for the full studies.

5.6.2 Path integration

We then examine the ability of the learned model on performing multi-step path integration, which can be accomplished by recurrently updating $\boldsymbol{v}_t$ (Eq. (5.11)) and decoding $\boldsymbol{v}_t$ to $\boldsymbol{x}_t$ for $t = 1, ..., T$ (Eq. (5.15)). Re-encoding $\boldsymbol{v}_t \leftarrow \boldsymbol{v}(\boldsymbol{x}_t)$ after decoding is adopted. Fig. 5.6.4(a) shows an example trajectory of accurate path integration for number of time steps $T = 30$. As shown in Fig. 5.6.4(b), with re-encoding, the path integration error remains close to zero over a duration of 500 time steps



(a) without L_2 (b) without $\boldsymbol{u}(\boldsymbol{x}') \geq 0$ (c) without skew-symmetry

Figure 5.6.3: Learned response maps in ablation studies where a certain model assumption is removed. (a) Remove the loss term L_2. (b) Remove the assumption $\boldsymbol{u}(\boldsymbol{x}') \geq 0$. (c) Remove the skew-symmetric assumption on $\boldsymbol{B}(\theta)$.

(< 0.01 cm, averaged over 1,000 episodes), even if the model is trained with the single-time-step transformation model (Eq. (5.17)). Without re-encoding, the error goes slightly higher but still remains small (ranging from 0.0 to 4.2 cm, mean 1.9 cm in the 1m $\times$ 1m environment). Fig. 5.6.4(c) summarizes the path integration performance by fixing the number of blocks and altering the block size. The performance of path integration would be improved as the block size becomes larger, i.e., more units or cells in each module. When block size is larger than 16, path integration is very accurate for the time steps tested.

Error correction. See chapter appendix for numerical experiments on error correction, which show that the learned model is still capable of path integration when we apply Gaussian white noise errors or Bernoulli drop-out errors to $\boldsymbol{v}_t$.

5.7 Related work

Our work is related to several lines of previous research on modeling grid cells. First, RNN models have been used to model grid cells and path integration. The traditional approach uses simulation-based models with hand-crafted connectivity, known as Continuous Attractor Neural Network (CAN) [AA92, BF09, CWZ13, PSR13, AB20]. On the other hand, more recently two pioneering works [CW18, BBU18] developed optimization-based RNN approaches to learn the

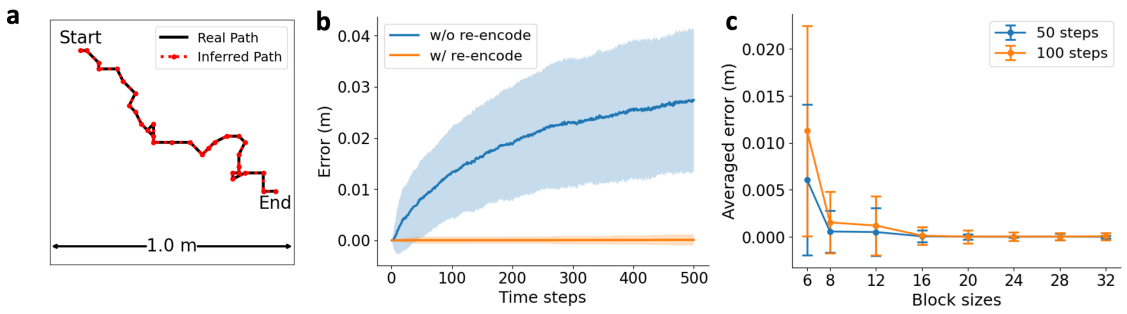


Figure 5.6.4: The learned model can perform accurate path integration. (a) Black: example trajectory. Red: inferred trajectory. (b) Path integration error over the number of time steps, for procedures with re-encoding and without re-encoding. (c) Path integration error with fixed number of blocks and different block sizes, for 50 and 100 time steps. The error band in (b) and error bar in (c) are standard deviations computed over 1,000 episodes.

path integration model and discovered that grid-like response patterns could emerge in the optimized networks. These results are further substantiated in [SMG19, CWC20]. Our work analyzes the properties of the general recurrent model for path integration, and our method belongs to the scheme of optimization-based approaches.

Second, our work differs from the PCA-based basis expansion models [DSM16, SMG19, SBG17] in that, unlike PCA, we make no assumption about the orthogonality between the basis functions, and the basis functions are generated by the transformation model. Furthermore, in previous basis expansion models [DSM16, SMG19], place fields with Mexican-hat patterns (with balanced excitatory center and inhibitory surround) had to be assumed in order to obtain hexagonal grid firing patterns. However, experimentally measured place fields in biological brains were instead well-characterized by Gaussian functions. Crucially, in our model, hexagonal grids emerge from learning with Gaussian place fields, and there is no need to assume any additional surround mechanisms or difference of Gaussian kernels.

In earlier work, we propose matrix representation of 2D self-motion [GXZ18a], while this chapter analyzes general transformation, and our investigation of the special case of the linear model reveals matrix Lie algebra and matrix Lie group structure, and connects the isotropic scaling

condition to hexagon grid patterns. Our work also connects the linear transformation model to the basis expansion model.

5.8 Conclusion and discussion

This chapter analyzes the recurrent model for path integration calculations by grid cells. We identify a group representation condition and an isotropic scaling condition that give rise to the locally conformal embedding of the self-motion. We study a linear prototype model that has a matrix Lie algebra and matrix Lie group structure, and connect the isotropic scaling condition to hexagon grid patterns. In addition to these theoretical investigations, our numerical experiments demonstrate that our model can learn hexagon grid patterns for the response maps of grid cells, and the learned model is capable of accurate path integration.

CHAPTER APPENDIX

5.A Theoretical analysis

5.A.1 Graphical illustrations of key equations

Fig. 5.A.1 illustrates key equations in the main text as well as in the chapter appendix.

5.A.2 Proof of Theorem 1 on conformal embedding

Proof: See Fig. 5.A.1(a) and (b) for an illustration. Consider the self-motion $\delta\boldsymbol{x} = (\delta r\cos\theta, \delta r\sin\theta)$,

$$\boldsymbol{v}(\boldsymbol{x}+\delta\boldsymbol{x}) = F(\boldsymbol{v}(\boldsymbol{x}), \delta r, \theta) = \boldsymbol{v}(\boldsymbol{x}) + f_\theta(\boldsymbol{v}(\boldsymbol{x}))\delta r + o(\delta r). \tag{5.19}$$

We can decompose the self-motion $\delta\boldsymbol{x}$ into two steps. First move along the direction 0 by $\delta r\cos\theta$, and then move along the direction $\pi/2$ by $\delta r\sin\theta$. Then under the **group representation condition**:

$$\begin{aligned}
\boldsymbol{v}(\boldsymbol{x}+\delta\boldsymbol{x}) &= F[F(\boldsymbol{v}(\boldsymbol{x}), \delta r\cos\theta, 0), \delta r\sin\theta, \pi/2)] \\
&= F[\boldsymbol{v}(\boldsymbol{x}) + f_0(\boldsymbol{v}(\boldsymbol{x}))\delta r\cos\theta + o(\delta r), \delta r\sin\theta, \pi/2] \\
&= [\boldsymbol{v}(\boldsymbol{x}) + f_0(\boldsymbol{v}(\boldsymbol{x}))\delta r\cos\theta] + f_{\pi/2}[\boldsymbol{v}(\boldsymbol{x}) + f_0(\boldsymbol{v}(\boldsymbol{x}))\delta r\cos\theta + o(\delta r)]\delta r\sin\theta + o(\delta r) \\
&= \boldsymbol{v}(\boldsymbol{x}) + f_0(\boldsymbol{v}(\boldsymbol{x}))\delta r\cos\theta + f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))\delta r\sin\theta + o(\delta r),
\end{aligned} \tag{5.20}$$

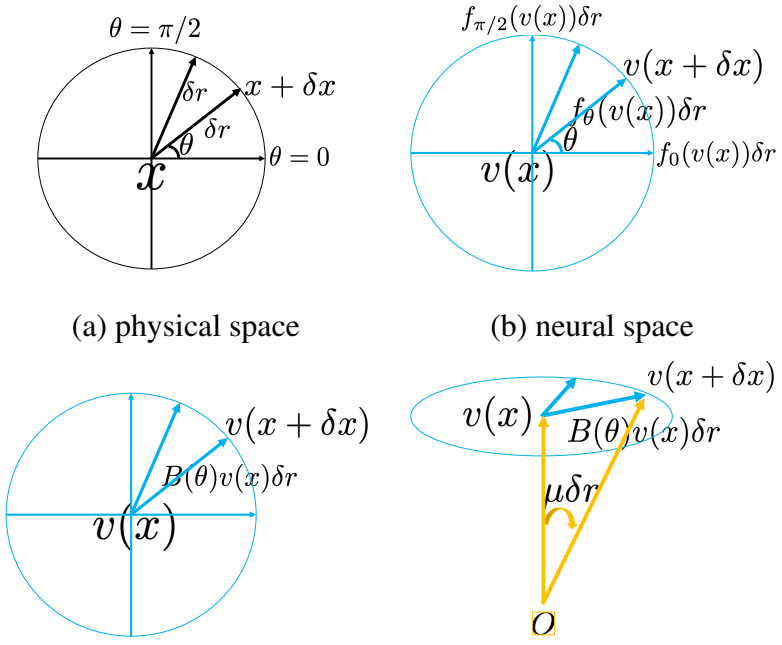
The last equation holds because assuming the derivative $f'_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))$ exists, then by first-order Taylor expansion,

$$f_{\pi/2}[\boldsymbol{v}(\boldsymbol{x}) + f_0(\boldsymbol{v}(\boldsymbol{x}))\delta r\cos\theta + o(\delta r)]\delta r\sin\theta \tag{5.21}$$

$$= [f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x})) + f'_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))f_0(\boldsymbol{v}(\boldsymbol{x}))\delta r\cos\theta + o(\delta r)]\delta r\sin\theta \tag{5.22}$$

$$= f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))\delta r\sin\theta + o(\delta r). \tag{5.23}$$

Since $\boldsymbol{v}(\boldsymbol{x}+\delta\boldsymbol{x}) = \boldsymbol{v}(\boldsymbol{x}) + f_\theta(\boldsymbol{v}(\boldsymbol{x}))\delta r + o(\delta r)$, by Eq. (5.20) we have $f_\theta(\boldsymbol{v}(\boldsymbol{x})) = f_0(\boldsymbol{v}(\boldsymbol{x}))\cos\theta +$



(a) physical space (b) neural space

(c) linear transformation (d) as rotation

Figure 5.A.1: Color-coded illustration. (a) In the 2D physical space, the agent moves from $\boldsymbol{x}$ to $\boldsymbol{x} + \delta\boldsymbol{x}$, where $\delta\boldsymbol{x} = (\delta r \cos\theta, \delta r \sin\theta)$, i.e., the agent moves by δr along the direction θ. We also show a displacement of δr in a different direction. (b) In the d-dimensional neural space, the vector $\boldsymbol{v}(\boldsymbol{x})$ is changed to $\boldsymbol{v}(\boldsymbol{x} + \delta\boldsymbol{x}) = F(\boldsymbol{v}(\boldsymbol{x}), \delta r, \theta) = \boldsymbol{v}(\boldsymbol{x}) + f_\theta(\boldsymbol{v}(\boldsymbol{x}))\delta r + o(\delta r)$, where the displacement is $f_\theta(\boldsymbol{v}(\boldsymbol{x}))\delta r = f_0(\boldsymbol{v}(\boldsymbol{x}))\delta r \cos\theta + f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))\delta r \sin\theta$. Under the isotropic condition that $\|f_\theta(\boldsymbol{v}(\boldsymbol{x}))\|$ is constant over θ, the local 2D self-motion $\delta\boldsymbol{x}$ at $\boldsymbol{x}$ in the 2D physical space is embedded conformally into the neural space as a 2D subspace around $\boldsymbol{v}(\boldsymbol{x})$. (c) Linear transformation, where $f_\theta(\boldsymbol{v}(\boldsymbol{x})) = \boldsymbol{B}(\theta)\boldsymbol{v}(\boldsymbol{x})$. (d) 3D perspective view of linear transformation as a rotation: $\boldsymbol{v}(\boldsymbol{x} + \delta\boldsymbol{x})$ is a rotation of $\boldsymbol{v}(\boldsymbol{x})$, and the angle of rotation is $\mu\delta r$, where $\mu = \|\boldsymbol{B}(\theta)\boldsymbol{v}(\boldsymbol{x})\|/\|\boldsymbol{v}(\boldsymbol{x}))\|$ (μ may depend on $\boldsymbol{x}$).

$f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))\sin\theta$, which is a 2D basis expansion. We are yet to prove that the two basis vectors $f_0(\boldsymbol{v}(\boldsymbol{x}))$ and $f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))$ are orthogonal with equal norm.

For notational simplicity, let $\boldsymbol{v}_1 = f_0(\boldsymbol{v}(\boldsymbol{x}))$ and $\boldsymbol{v}_2 = f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))$. Then under the **isotropic scaling condition**, $\|\boldsymbol{v}_1\| = \|\boldsymbol{v}_2\| = \|f_\theta(\boldsymbol{v}(\boldsymbol{x}))\| = s$, and $f_\theta(\boldsymbol{v}(\boldsymbol{x})) = \boldsymbol{v}_1 \cos\theta + \boldsymbol{v}_2 \sin\theta$ for any θ.

Then we have that for any θ,

$$s^2 = \|f_\theta(\boldsymbol{v}(\boldsymbol{x}))\|^2 = \|\boldsymbol{v}_1 \cos\theta + \boldsymbol{v}_2 \sin\theta\|^2 = s^2 + 2\langle \boldsymbol{v}_1, \boldsymbol{v}_2\rangle \cos\theta \sin\theta. \tag{5.24}$$

Thus $\langle \boldsymbol{v}_1, \boldsymbol{v}_2\rangle = 0$, i.e., $f_0(\boldsymbol{v}(\boldsymbol{x})) \perp f_{\pi/2}(\boldsymbol{v}(\boldsymbol{x}))$. This leads to the conformal embedding of the local 2D polar system in the physical space as a 2D polar system in the d-dimensional neural space, with a scaling factor s (which may depend on $\boldsymbol{x}$). □

5.A.3 Proof of Theorem 3 on error correction

We first prove the following result about the general transformation.

Proposition 1. *Assume the general transformation model (Eq. (5.19)) and the isotropic scaling condition. For any fixed $\boldsymbol{x}$, let $s = \|f_\theta(\boldsymbol{v}(\boldsymbol{x}))\|$, which is independent of θ. Suppose the neurons are noisy: $\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \tau^2 \boldsymbol{I}_d)$, and d is the dimensionality of $\boldsymbol{v}$. Suppose the agent infers its 2D position $\hat{\boldsymbol{x}}$ from $\boldsymbol{v}$ by $\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}'} \|\boldsymbol{v} - \boldsymbol{v}(\boldsymbol{x}')\|^2$, i.e., $\boldsymbol{v}(\hat{\boldsymbol{x}})$ is the projection of $\boldsymbol{v}$ onto the 2D manifold formed by $(\boldsymbol{v}(\boldsymbol{x}'), \forall \boldsymbol{x}')$. Then we have*

$$E\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|^2 = 2\tau^2/s^2. \tag{5.25}$$

Proof: By theorem 1, for a fixed self-position $\boldsymbol{x}$, we embed the 2D local neighborhood around $\boldsymbol{x}$ as a local 2D plane around $\boldsymbol{v}(\boldsymbol{x})$ in the d-dimensional neural space. A local perturbation in self-position, $\delta\boldsymbol{x}$, is translated into a local perturbation in $\boldsymbol{v}(\boldsymbol{v} + \delta\boldsymbol{x})$, so that

$$\|\delta\boldsymbol{v}\|^2 = \|f_\theta(\boldsymbol{v}(\boldsymbol{x}))\delta r + o(\delta r)\|^2 = s^2\|\delta\boldsymbol{x}\|^2, \tag{5.26}$$

where $\delta\boldsymbol{v} = \boldsymbol{v}(\boldsymbol{x} + \delta\boldsymbol{x}) - \boldsymbol{v}(\boldsymbol{x})$.

Suppose the agent infers its 2D position $\hat{\boldsymbol{x}}$ by $\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}'} \|\boldsymbol{v} - \boldsymbol{v}(\boldsymbol{x}')\|^2$, which amounts to projecting $\boldsymbol{v}$ onto the local 2D plane around $\boldsymbol{v}(\boldsymbol{x})$. The projected vector $\boldsymbol{v}(\hat{\boldsymbol{x}})$ on the local 2D plane is $\boldsymbol{v}(\boldsymbol{x}) + \delta\boldsymbol{v}$, where $\delta\boldsymbol{v}$ is the projection of ϵ onto the 2D plane. More specifically, let $(\boldsymbol{v}_1, \boldsymbol{v}_2)$ be an orthonormal basis of the local 2D plane centered at $\boldsymbol{v}(\boldsymbol{x})$. Then $\delta\boldsymbol{v}$ can be written

as $e_1\boldsymbol{v}_1 + e_2\boldsymbol{v}_2$, where

$$\boldsymbol{e} = (e_1, e_2)^\top = (\boldsymbol{v}_1, \boldsymbol{v}_2)^\top \boldsymbol{\varepsilon} \sim \mathcal{N}(0, \tau^2 \boldsymbol{I}_2). \tag{5.27}$$

Let $\delta\boldsymbol{x} = \hat{\boldsymbol{x}} - \boldsymbol{x}$. Due to **isotropic scaling and conformal embedding**, the ℓ_2 squared error translate according to

$$\|\delta\boldsymbol{x}\|^2 = \|\delta\boldsymbol{v}\|^2/s^2 = (e_1^2 + e_2^2)/s^2, \tag{5.28}$$

whose expectation is $2\tau^2/s^2$. Thus $\mathrm{E}\|\hat{\boldsymbol{x}} - \boldsymbol{x}\|^2 = 2\tau^2/s^2$.□

Proof of Theorem 3: It is reasonable to assume $\tau^2 = \alpha^2(\|\boldsymbol{v}(\boldsymbol{x})\|^2/d)$, where α^2 measures the variance of noise relative to $\|\boldsymbol{v}(\boldsymbol{x})\|^2/d$, which is the average of $(v_i(\boldsymbol{x})^2, i = 1, ..., d)$. In other words, α^2 measures the noise level.

In the linear case, the metric is

$$\mu = \|f_\theta(\boldsymbol{v}(\boldsymbol{x}))\|/\|\boldsymbol{v}(\boldsymbol{x})\| = \|\boldsymbol{B}(\theta)\boldsymbol{v}(\boldsymbol{x}))\|/\|\boldsymbol{v}(\boldsymbol{x})\| = s/\|\boldsymbol{v}(\boldsymbol{x})\|, \tag{5.29}$$

which measures how fast $\boldsymbol{v}(\boldsymbol{x})$ rotates in the neural space as $\boldsymbol{x}$ changes. Then

$$\mathrm{E}\|\delta\boldsymbol{x}\|^2 = 2\alpha^2/(\mu^2 d). \tag{5.30}$$

The above scaling shows that error correction depends on two factors. One is the metric μ, and the other is the dimensionality d, i.e., the number of neurons. These correspond to two phases of error correction. One is to project the d-dimensional $\boldsymbol{\epsilon}$ to the 2-dimensional $\delta\boldsymbol{v}$. The bigger d is, the bigger the error correction. The other is to translate $\|\delta\boldsymbol{v}\|^2$ to $\|\delta\boldsymbol{x}\|^2$. The bigger μ is, the bigger the error correction. □

5.A.4 Proof of Theorem 4 on hexagon grid patterns

Proof: Let $\boldsymbol{e}(\boldsymbol{x}) = (\exp(i\langle \boldsymbol{a}_j, \boldsymbol{x}\rangle), j = 1, 2, 3)^\top$, where $(\boldsymbol{a}_j, j = 1, 2, 3)$ are three 2D vectors of equal norm, and the angle between every pair of them is $2\pi/3$. Let $\boldsymbol{v}(\boldsymbol{x}) = \boldsymbol{U}\boldsymbol{e}(\boldsymbol{x})$, where $\boldsymbol{U}$ is an

arbitrary unitary matrix, i.e., $\boldsymbol{U}^*\boldsymbol{U} = \boldsymbol{I}$. Then $\|\boldsymbol{v}(\boldsymbol{x})\|^2 = \|\boldsymbol{e}(\boldsymbol{x})\|^2 = 3\ \forall \boldsymbol{x}$, and $\boldsymbol{e}(\boldsymbol{x}) = \boldsymbol{U}^*\boldsymbol{v}(\boldsymbol{x})$. For self-motion $\delta\boldsymbol{x} = (\delta r\cos\theta, \delta r\sin\theta) = \boldsymbol{q}(\theta)\delta r$, let

$$\begin{aligned}\Lambda(\delta\boldsymbol{x},\theta) &= \text{diag}(\exp(\langle \boldsymbol{a}_j, \delta\boldsymbol{x}\rangle), j = 1,2,3)\\ &= \text{diag}(\exp(\langle \boldsymbol{a}_j, \boldsymbol{q}(\theta)\rangle\delta r), j = 1,2,3)\\ &= \boldsymbol{I} + \text{diag}(i\langle \boldsymbol{a}_j, \boldsymbol{q}(\theta)\rangle), j = 1,2,3)\delta r + o(\delta r)\\ &= \boldsymbol{I} + \boldsymbol{D}(\theta)\delta r + o(\delta r). \end{aligned} \tag{5.31}$$

Then

$$\begin{aligned}\boldsymbol{v}(\boldsymbol{x}+\delta\boldsymbol{x}) &= \boldsymbol{U}\boldsymbol{e}(\boldsymbol{x}+\delta x)\\ &= \boldsymbol{U}\Lambda(\delta\boldsymbol{x},\theta)\boldsymbol{e}(\boldsymbol{x})\\ &= \boldsymbol{U}\Lambda(\delta\boldsymbol{x},\theta)\boldsymbol{U}^*\boldsymbol{v}(\boldsymbol{x})\\ &= (\boldsymbol{I} + \boldsymbol{U}\boldsymbol{D}(\theta)\boldsymbol{U}^*\boldsymbol{v}(\boldsymbol{x})\delta r)\boldsymbol{v}(\boldsymbol{x}) + o(\delta r)\\ &= (\boldsymbol{I} + \boldsymbol{B}(\theta)\delta r)\boldsymbol{v}(\boldsymbol{x}) + o(\delta r), \end{aligned} \tag{5.32}$$

where $\boldsymbol{B}(\theta) = \boldsymbol{U}\boldsymbol{D}(\theta)\boldsymbol{U}^*$, and $\boldsymbol{B}(\theta) = -\boldsymbol{B}(\theta)^*$. For isotropic condition,

$$\begin{aligned}\|\boldsymbol{B}(\theta)\boldsymbol{v}(\boldsymbol{x})\|^2 &= \|\boldsymbol{D}(\theta)\boldsymbol{e}(\boldsymbol{x})\|^2\\ &= \sum_{j=1}^{3}\langle \boldsymbol{a}_j, \boldsymbol{q}(\theta)\rangle^2\\ &= \text{const}\|\boldsymbol{a}_j\|^2\|\boldsymbol{q}(\theta)\|^2 = \text{const}\|\boldsymbol{a}_j\|^2, \end{aligned} \tag{5.33}$$

which is independent of θ, because $(\boldsymbol{a}_j, j = 1,2,3)$ forms a **tight frame** in 2D.

One example of $\boldsymbol{U}$ is the following matrix:

$$\frac{1}{\sqrt{3}}\begin{pmatrix} 1 & 1 & 1\\ 1 & \exp(i2\pi/3) & \exp(-i2\pi/3)\\ 1 & \exp(-i2\pi/3) & \exp(i2\pi/3)\end{pmatrix} \tag{5.34}$$

The resulting $(v_i(x), i = 1,2,3)$ have the same orientation but different phases, i.e., they are spatially shifted versions of each other. □

The limitation of Theorem 4 is that we only show $\boldsymbol{v}(\boldsymbol{x}) = \boldsymbol{U}\boldsymbol{e}(\boldsymbol{x})$ satisfies the linear model and the isotropic scaling condition, but we did not show that linear model with the isotropic condition only has solutions that are hexagon grid patterns.

5.A.5 From group representation to orthogonal basis functions

Group representation is a central theme in modern mathematics. In particular, it leads to a deep understanding and generalization of Fourier analysis or harmonic analysis.

For the set of $\{\Delta\boldsymbol{x}\}$ that form a group, a matrix representation $\boldsymbol{M}(\Delta\boldsymbol{x})$ is equivalent to another representation $\tilde{\boldsymbol{M}}(\Delta\boldsymbol{x})$ if there exists an invertible matrix $\boldsymbol{P}$ such that $\tilde{\boldsymbol{M}}(\Delta\boldsymbol{x}) = \boldsymbol{P}\boldsymbol{M}(\Delta\boldsymbol{x})\boldsymbol{P}^{-1}$ for each $\boldsymbol{x}$. A matrix representation is reducible if it is equivalent to a block diagonal matrix representation, i.e., we can find a matrix $\boldsymbol{P}$, such that $\boldsymbol{P}\boldsymbol{M}(\Delta\boldsymbol{x})\boldsymbol{P}^{-1}$ is block diagonal for every $\Delta\boldsymbol{x}$. Suppose the group is a finite group or a compact Lie group, and $\boldsymbol{M}$ is a unitary representation, i.e., $\boldsymbol{M}(\Delta\boldsymbol{x})$ is a unitary matrix. If $\boldsymbol{M}$ is block-diagonal, $\boldsymbol{M} = \text{diag}(\boldsymbol{M}_k, k = 1, ..., K)$, with non-equivalent blocks, and each block $\boldsymbol{M}_k$ cannot be further reduced, then the matrix elements $(M_{kij}(\Delta\boldsymbol{x}))$ are orthogonal basis functions of $\Delta\boldsymbol{x}$. Such orthogonality relations are proved by Schur [Zee16] for finite group, and by Peter-Weyl for compact Lie group [Tay02]. For our case, theoretically the group of displacements $\Delta\boldsymbol{x}$ in the 2D domain is $\mathbb{R}^2$, but we learn our model within a finite range, and we further discretize the range into a lattice. Thus the above orthogonal relations hold.

In our model, we also assume block diagonal $\boldsymbol{M}$, and we call each block a module. However, we do not assume each module is irreducible, i.e., each module itself may be further diagonalized into a block diagonal matrix of irreducible sub-blocks. Thus the elements within the same module $\boldsymbol{v}_k(\boldsymbol{x})$ may be linear mixings of orthogonal basis functions of the irreducible sub-blocks, and the linear mixings themselves are not necessarily orthogonal.

Fig. 5.A.2 visualizes the correlation between pairs of the learned $\boldsymbol{v}_i(\boldsymbol{x})$ and $\boldsymbol{v}_j(\boldsymbol{x})$, $i, j = 1, ..., d$. For different i and j, the correlations between different $\boldsymbol{v}_i(\boldsymbol{x})$ and $\boldsymbol{v}_j(\boldsymbol{x})$ are close to zero; i.e., they

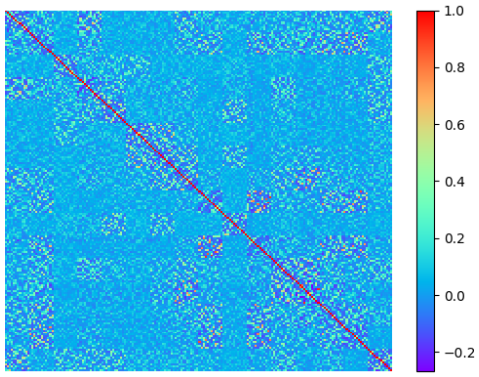


Figure 5.A.2: Correlation heatmap for each pair of the learned $v_i(\boldsymbol{x})$ and $v_j(\boldsymbol{x})$. The correlations are computed over 40×40 lattice of $\boldsymbol{x}$.

are nearly orthogonal to each other. The average absolute value of correlation is 0.09, and the within-block average value is about the same as the between-block average value.

Unlike previous work on learning basis expansion model (or PCA-based model [DSM16]), we **do not constrain the basis functions $\boldsymbol{v}(\boldsymbol{x}) = (v_i(\boldsymbol{x}), i = 1, ..., d)$ to be orthogonal to each other**. Instead, we constrain them by our path integration model via the loss term L_1. Nonetheless, the learned $v_i(\boldsymbol{x})$ are close to being orthogonal in our experiments.

5.A.6 Decoding and re-encoding

In the above analysis, the projection of $\boldsymbol{v}$ onto the local 2D plane around $\boldsymbol{v}(\boldsymbol{x})$ is $\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}'} \|\boldsymbol{v} - \boldsymbol{v}(\boldsymbol{x}')\|^2$, which, for the linear model, amounts to decoding $\boldsymbol{v}$ to $\hat{\boldsymbol{x}}$ via

$$\hat{\boldsymbol{x}} = \arg\max_{x'} \langle \boldsymbol{v}, \boldsymbol{v}(\boldsymbol{x}') \rangle, \tag{5.35}$$

because $\|\boldsymbol{v}(\boldsymbol{x}')\|^2$ is constant. We project $\boldsymbol{v}$ to $\boldsymbol{v}(\hat{\boldsymbol{x}})$, which is an re-encoding of $\boldsymbol{v}$.

We can also perform decoding via the learned $\boldsymbol{u}(\boldsymbol{x}')$:

$$\hat{\boldsymbol{x}} = \arg\max_{x'} \langle \boldsymbol{v}, \boldsymbol{u}(\boldsymbol{x}') \rangle, \tag{5.36}$$

and re-encoding $\boldsymbol{v} \leftarrow \boldsymbol{v}(\hat{\boldsymbol{x}})$. For the above decoding, the heat map

$$\boldsymbol{h}(\boldsymbol{x}') = \langle \boldsymbol{v}, \boldsymbol{u}(\boldsymbol{x}') \rangle = \langle \boldsymbol{v}(\boldsymbol{x}), \boldsymbol{u}(\boldsymbol{x}') \rangle + \langle \boldsymbol{\epsilon}, \boldsymbol{u}(\boldsymbol{x}') \rangle = A(\boldsymbol{x}, \boldsymbol{x}') + \boldsymbol{e}(\boldsymbol{x}'), \tag{5.37}$$

where $\boldsymbol{e}(\boldsymbol{x}') = \langle \boldsymbol{\epsilon}, \boldsymbol{u}(\boldsymbol{x}') \rangle \sim \mathcal{N}(0, \sigma^2 \|\boldsymbol{v}(\boldsymbol{x})\|^2 \|\boldsymbol{u}(\boldsymbol{x}')\|^2 / d)$. For $A(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\|\boldsymbol{x} - \boldsymbol{x}'\|^2/(2\sigma^2)) = \langle \boldsymbol{v}(\boldsymbol{x}), \boldsymbol{u}(\boldsymbol{x}') \rangle$, if σ^2 is small, $A(\boldsymbol{x}, \boldsymbol{x}')$ decreases to 0 quickly, i.e., if $\|\boldsymbol{x}' - \boldsymbol{x}\| > c$, then $A(\boldsymbol{x}, \boldsymbol{x}') < \exp(-c^2/(2\sigma^2))$, and the chance for the maximum of $\boldsymbol{h}(\boldsymbol{x}')$ to be achieved at an $\boldsymbol{x}'$ so that $\|\boldsymbol{x}' - \boldsymbol{x}\| > c$ can be very small. The above analysis also provides a justification for regularizing $\|\boldsymbol{u}(\boldsymbol{x}')\|^2$ in learning.

For error correction, we want to use small σ^2. However, for path planning, we need large σ^2 so that we can assess the adjacency as well as the change of the adjacency between the position on the path and the target position even if they are far apart.

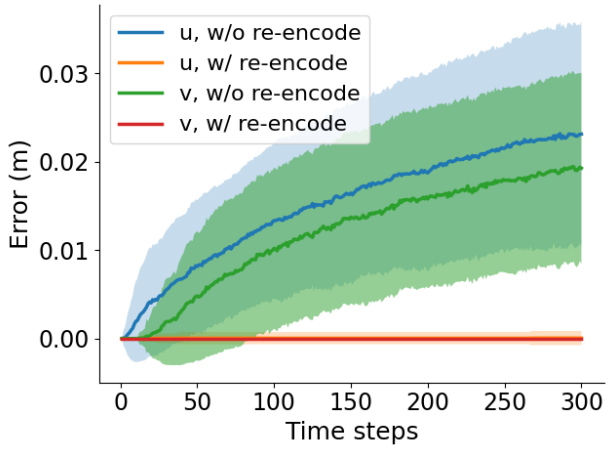


Figure 5.A.3: Path integration error over number of time steps. The mean and standard deviation band is computed over 1,000 episodes. "$\boldsymbol{v}$" means decoding by Eq. (5.35), and "$\boldsymbol{u}$" means decoding by Eq. (5.36). The squared domain is 1m $\times$ 1m.

In the experiments in the main text, we use Eq. (5.36) for decoding. In Fig. 5.A.3, we also show the results of path integration using Eq. (5.35) for decoding, whose performance is even better than Eq. (5.36). Especially the error would remain 0 over 300 time steps and 1,000 episodes using Eq. (5.35) with re-encoding. The advantage of (5.35) is that error correction is achieved within the grid cells system itself without interacting with the place cells.

5.B Experiments

5.B.1 Implementation details

Monte Carlo samples. The expectations in loss terms are approximated by Monte Carlo samples. Here we detail the generation of Monte Carlo samples. For $(\boldsymbol{x}, \boldsymbol{x}')$ used in $L_0 = \mathrm{E}_{\boldsymbol{x},\boldsymbol{x}'}[A(\boldsymbol{x}, \boldsymbol{x}') - \langle \boldsymbol{v}(\boldsymbol{x}), \boldsymbol{u}(\boldsymbol{x}') \rangle]^2$, $\boldsymbol{x}$ is first sampled uniformly within the entire domain, and then the displacement $d\boldsymbol{x}$ between $\boldsymbol{x}$ and $\boldsymbol{x}'$ is sampled from a normal distribution $\mathcal{N}(0, \sigma^2 \boldsymbol{I}_2)$, where $\sigma = 0.48$. This is to ensure that nearby samples are given more emphasis. We let $\boldsymbol{x}' = \boldsymbol{x} + d\boldsymbol{x}$, and those pairs $(\boldsymbol{x}, \boldsymbol{x}')$ within the range of domain (i.e., 1m $\times$ 1m, 40×40 lattice) are kept as valid data. For $(\boldsymbol{x}, \Delta\boldsymbol{x})$ used in $L_1 = \mathrm{E}_{\boldsymbol{x},\Delta\boldsymbol{x}} |\boldsymbol{v}(\boldsymbol{x} + \Delta\boldsymbol{x}) - \exp(\boldsymbol{B}(\theta)\Delta r)\boldsymbol{v}(\boldsymbol{x})|^2$, $\Delta\boldsymbol{x}$ is sampled uniformly within a circular domain with radius equal to 3 grids and $(0, 0)$ as the center. Specifically, Δr^2, the squared length of $\Delta\boldsymbol{x}$, is sampled uniformly from $[0, 3]$ grids, and θ is sampled uniformly from $[0, 2\pi]$. We take the square root of the sampled Δr^2 as Δr and let $\Delta\boldsymbol{x} = (\Delta r \cos\theta, \Delta r \sin\theta)$. Then $\boldsymbol{x}$ is uniformly sampled from the region such that both $\boldsymbol{x}$ and $\boldsymbol{x} + \Delta\boldsymbol{x}$ are within the range of domain. For $(\theta, \Delta\theta)$ used in $L_2 = \sum_{k=1}^{K} \mathrm{E}_{\boldsymbol{x},\theta,\Delta\theta}[\|\boldsymbol{B}_k(\theta + \Delta\theta)\boldsymbol{v}_k(\boldsymbol{x})\| - \|\boldsymbol{B}_k(\theta)\boldsymbol{v}_k(\boldsymbol{x})\|]^2$, we uniformly sample θ and $\theta + \Delta\theta$ from discretized angles, i.e., 144 directions discretized for circle $[0, 2\pi]$. We will study sampling only small $\Delta\theta$ in the future.

Training details. The model is trained for 14,000 iterations. At each iteration, the samples are generated online. For the first 8,000 iterations, we update all learnable parameters, while for the following iterations, we fix the learned $\boldsymbol{v}(\boldsymbol{x})$ and update the other learnable parameters. The initial learning rate is set as 0.003 and is decreased by a factor of 0.5 every 500 iterations after 8,000 iterations. We use Adam [KB14] optimizer. The model is trained on a single Titan XP GPU. We apply the maximum batch size that can fit into the single GPU, which is 90,000. It takes about 3.5 hours to train the model on a single Titan XP GPU.

5.B.2 Learned patterns

Fig. 5.B.1 displays the autocorrelograms of learned patterns of $\boldsymbol{v}(\boldsymbol{x})$.

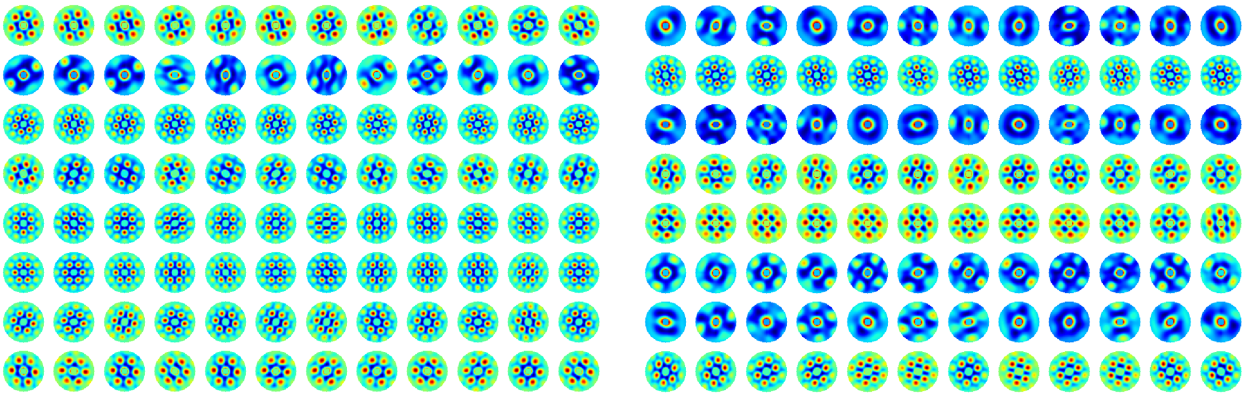


Figure 5.B.1: Autocorrelograms of the learned patterns of $\boldsymbol{v}(\boldsymbol{x})$.

Fig. 5.B.2 shows the learned patterns of $\boldsymbol{u}(\boldsymbol{x})$ with 16 blocks of 12 cells in each block. Regular hexagon patterns also emerge.

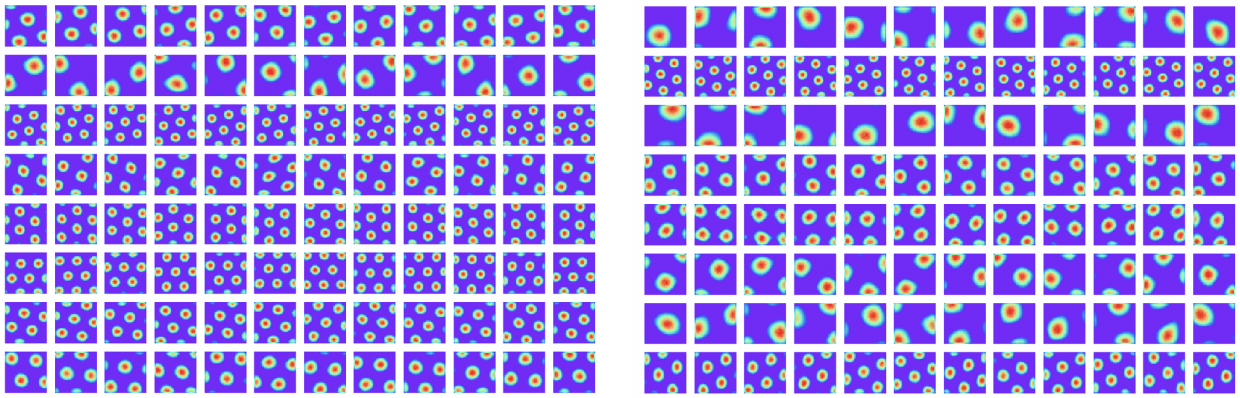


Figure 5.B.2: Learned patterns of $\boldsymbol{u}(\boldsymbol{x})$ with 16 blocks of size 12 cells in each block. Every row shows the learned patterns within the same block.

Because $A(\boldsymbol{x}, \boldsymbol{x}')$ is a sharp Gaussian kernel, it contains a whole range of frequencies in the 2D Fourier domain. The learned response maps of the grid cells span a range of frequencies or scales too. Each module or block focuses on a certain frequency band, or a ring in the 2D Fourier domain,

which corresponds to the metric of the module. The isotropic condition serves to constrain each module to be within a ring of 2D frequencies. Different modules collectively pave the whole range of frequencies.

It is worth noting that, consistent with the experiential observations, we assume individual place field $A(\boldsymbol{x}, \boldsymbol{x}')$ to exhibit a Gaussian shape, rather than a Mexican-hat pattern (with balanced excitatory center and inhibitory surround) as assumed in previous basis expansion models [DSM16, SMG19] of grid cells. The Mexican-hat difference of Gaussians pattern occupies a ring in the 2D Fourier domain. It corresponds to a module in our model. But we use isotropic condition to enforce each module to be within a ring, and we use different modules to pave the whole Fourier domain.

For learned firing patterns of $\boldsymbol{v}(\boldsymbol{x})$, we also display the histogram of grid orientations in Fig. 5.B.3, where we do not observe clear clusters.

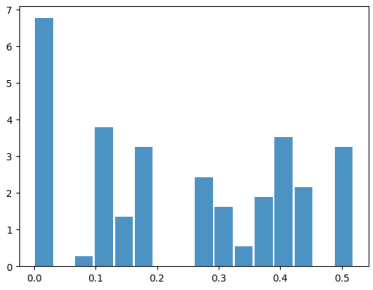


Figure 5.B.3: Histogram of grid orientations of the learned firing patterns of $\boldsymbol{v}(\boldsymbol{x})$.

In Fig. 5.B.4, we show the learned patterns of a block of $\boldsymbol{B}(\theta)$. Each element shows significant sine/cosine tuning over θ. For the other blocks, the patterns are all similar.

5.B.3 Error correction

We begin by aessing the ability of error correction of the learned system following the setting in Theorem 3. Specifically, for a given location $\boldsymbol{x}$, suppose the neurons are perturbed by Gaussian noise: $\boldsymbol{v} = \boldsymbol{v}(x) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \tau^2 \boldsymbol{I}_d)$ and $\tau^2 = \alpha^2(\|\boldsymbol{v}(\boldsymbol{x})\|^2/d)$, so that α^2 measures the

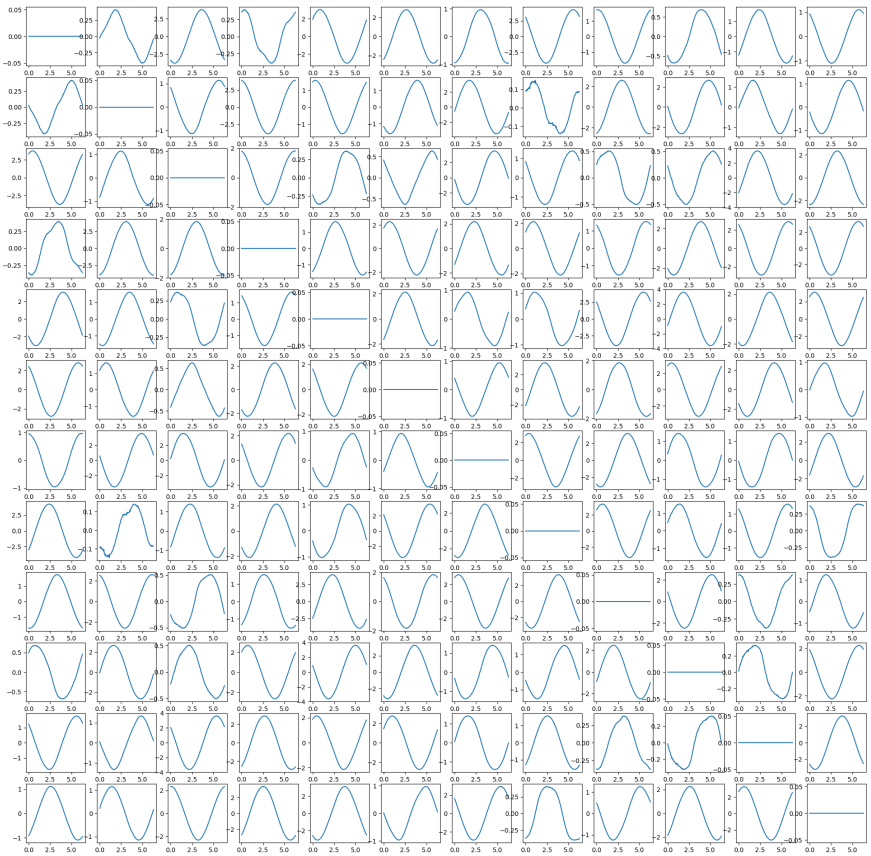


Figure 5.B.4: Learned patterns of a block of $\boldsymbol{B}(\theta)$. Each subfigure shows the value of an element in $\boldsymbol{B}(\theta)$ (vertical axis) over θ (horizontal axis).

variance of noise relative to the average magnitude of $(v_i(\boldsymbol{x})^2, i = 1, ..., d)$ and α measures the relative standard deviation. We infer the 2D position $\hat{\boldsymbol{x}}$ from $\boldsymbol{v}$ by $\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}'} \|\boldsymbol{v} - \boldsymbol{v}(\boldsymbol{x}')\|^2$. Fig. 5.B.5 displays the inference error over the relative standard deviation α of the added Gaussian noise. We also show the results using the learned $\boldsymbol{u}(\boldsymbol{x}')$ for inference (Eq. (5.36)). The system works remarkably well even if $\alpha = 2$.

We further assess the ability of error correction in long-distance path integration. Specifically, along the way of path integration, at every time step t, two types of errors are introduced to $\boldsymbol{v}_t$: (1) Gaussian noise or (2) dropout masks, i.e., a certain percentage of units are randomly set to zero. Fig. 5.B.6 summarizes the path integration performance with different levels of injected errors

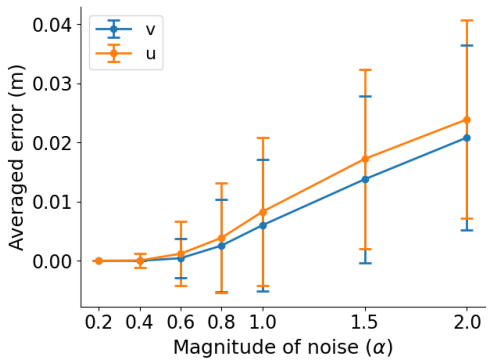


Figure 5.B.5: Error correction results following the setting in Theorem 3. The error bar stands for the standard deviation over 1,000 trials. "$\boldsymbol{v}$" means decoding by Eq. (5.35), and "$\boldsymbol{u}$" means decoding by Eq. (5.36). The squared domain is 1m $\times$ 1m.

for $T = 100$, using $\boldsymbol{v}(\boldsymbol{x}')$ (Eq. (5.35)) or $\boldsymbol{u}(\boldsymbol{x}')$ (Eq. (5.36)) for decoding. The results show that re-encoding at each step helps error correction, especially for dropout masks. For Gaussian noise, even without decoding and re-encoding at each step, decoding at the final step alone is capable of removing much of the noise. Notably, with re-encoding, the path integration works well even if Gaussian noise with $\alpha = 1$ is added or 50% units are randomly dropped out at each step, indicating that the learned system is robust to different sources of errors.

5.B.4 Ablation studies

Isotropic scaling condition is necessary for hexagon grid patterns. A natural question is whether the isotropic scaling condition (condition 2) is important to learn hexagon grid patterns. To verify this, we learn the model by removing the loss term L_2 (equation 14 in the main text) from the loss function, which constrains the model to meet condition 2. As shown in Fig. 5.B.7, more strip-like patterns emerge without L_2, indicating that condition 2 is important for isotropic grid-like patterns to emerge.

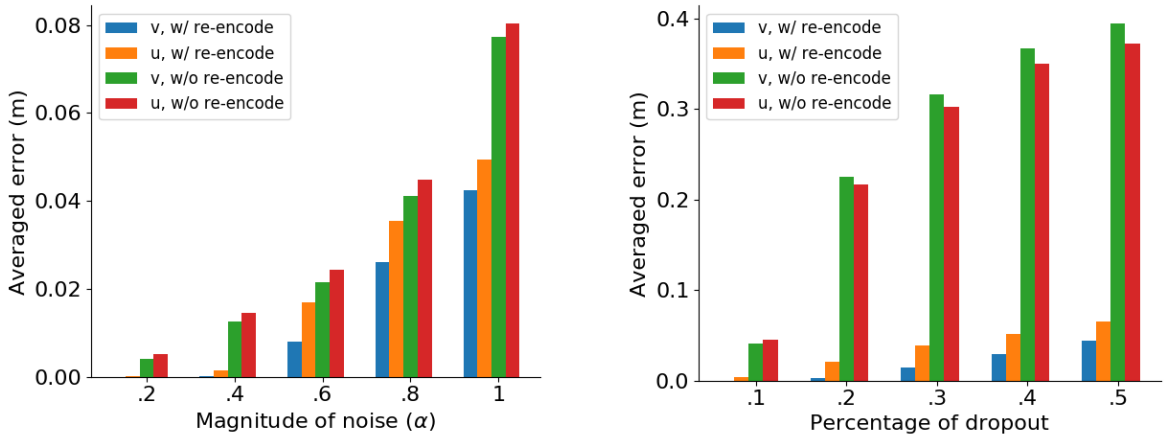


Figure 5.B.6: Path integration results with different levels of injected errors. *Left*: Gaussian noise. The magnitude of noise is measured using the average of the squared magnitudes of the units in $\boldsymbol{v}(\boldsymbol{x})$ as the reference. *Right*: dropout masks. Certain percentage of units are randomly set to zero at each step. "$\boldsymbol{v}$" means decoding by Eq. (5.35), and "$\boldsymbol{u}$" means decoding by Eq. (5.36). The squared domain is 1m $\times$ 1m.

Assumption of $\boldsymbol{u}(\boldsymbol{x}') \geq 0$ is not necessary for hexagon grid patterns. During training, we make an assumption of $\boldsymbol{u}(\boldsymbol{x}') \geq 0$ to make sure the connections from grid cells to place cells are excitatory [ZYM13, ROS18]. However, we want to emphasize this is not a key assumption leading to hexagon patterns in our model. Fig. 5.B.8 demonstrates the learned neurons in the network without assuming $\boldsymbol{u}(\boldsymbol{x}') \geq 0$, where hexagonal grid firing patterns also emerge. The average gridness score is 0.82 and the percentage of grid cells is 87.50%. However, the grid activations could be either positive/excitatory (in red color) or negative/inhibitory (in blue color).

Skew-symmetric assumption of $\boldsymbol{B}(\theta)$ is not important for hexagon grid patterns. To derive the clear geometric structure of the linear transformation model, we have assumed that $\boldsymbol{B}(\theta)$ is skew-symmetric, i.e., $\boldsymbol{B}(\theta) = -\boldsymbol{B}(\theta)^\top$. Nonetheless, this assumption is not important for the emergence of hexagon grid patterns. Fig. 5.B.9 demonstrates the learned neurons without assuming that $\boldsymbol{B}(\theta)$ is skew-symmetric. Hexagon grid firing patterns emerge in most of the neurons, with only one block of square grid firing patterns.

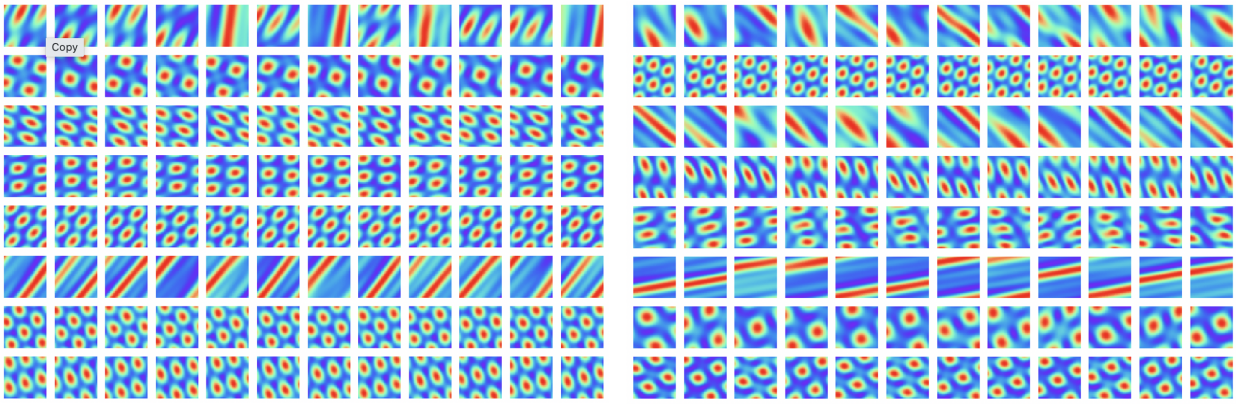


Figure 5.B.7: Learned neurons without loss term L_2, which is the constraint on isotropic scaling condition. More strip-like firing patterns emerge.

Number and sizes of blocks do not matter. It is worthwhile to mention that the emergence of hexagonal grid firing patterns in the learned neurons are not due to the specific design of the block size or the number of blocks. Fig. 5.B.10 visualizes the learned neurons by fixing the total amount of neurons at 192 and altering the block size and number of blocks. Hexagon patterns emerge in all the settings.

Multiple blocks or modules are necessary for learning grid patterns of multiple scales. We further try to fully remove the assumption of blocks or modules; i.e., we learn a single block of $\boldsymbol{B}(\theta)$. Fig. 5.B.11 shows the learned neurons and the corresponding autocorrelograms. All the learned neurons share similar large scales, which indicates that the high-frequency part of $A(\boldsymbol{x}, \boldsymbol{x}')$ may not be fitted very well.

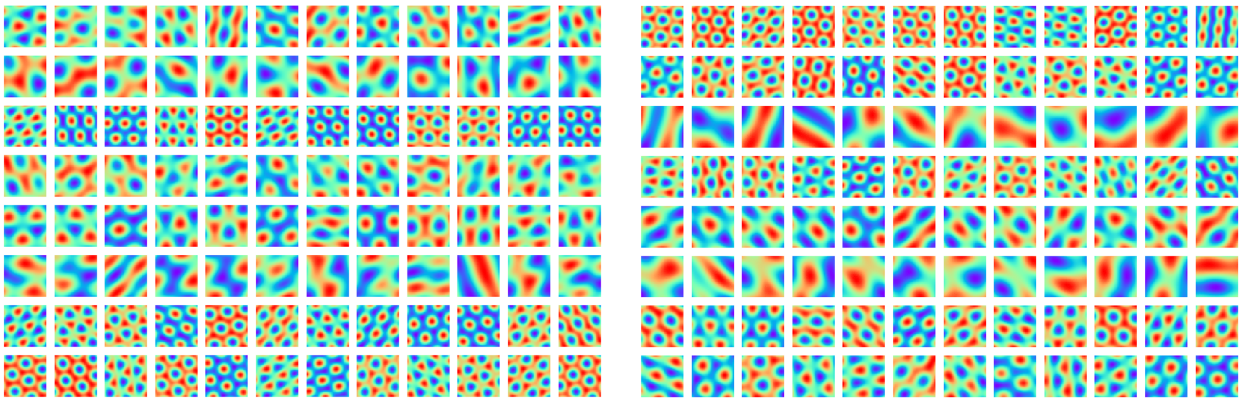


Figure 5.B.8: Learned neurons without the assumption of $\boldsymbol{u}(\boldsymbol{x}') \geq 0$. Hexagonal grid firing patterns also emerge, with the grid activations being either positive/excitatory (in red color) or negative/inhibitory (in blue color).

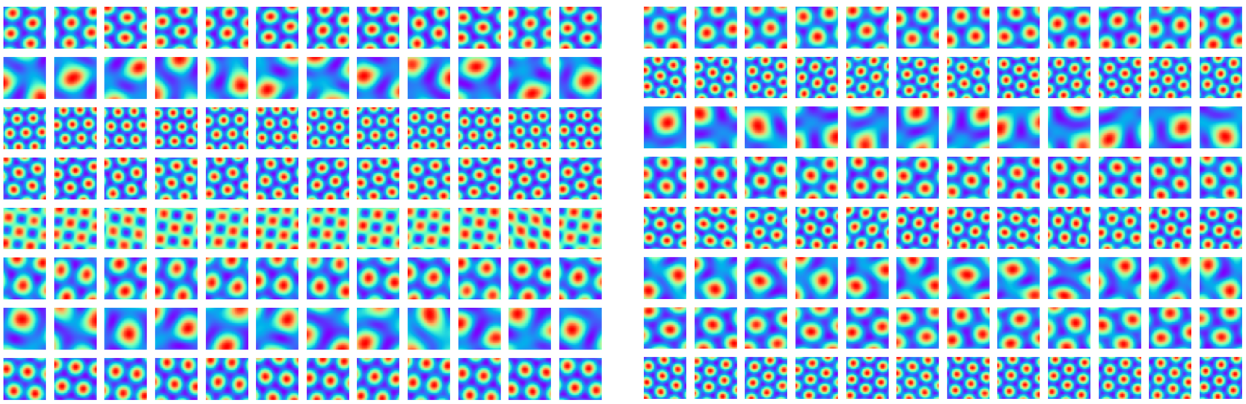
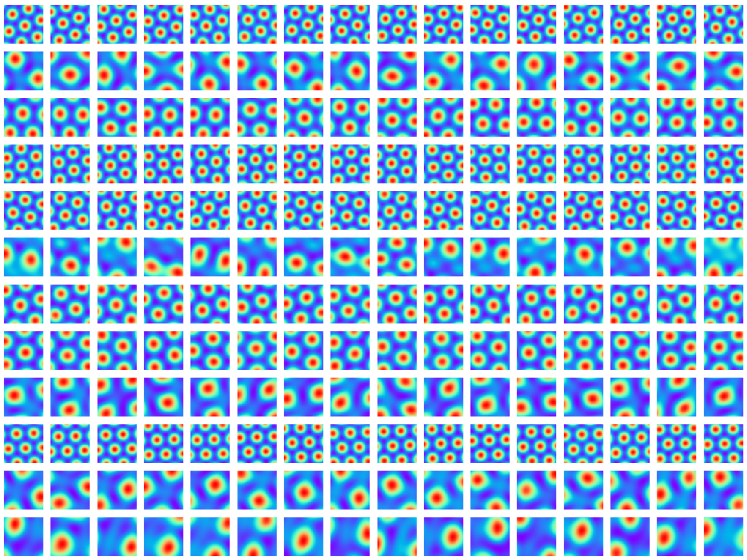
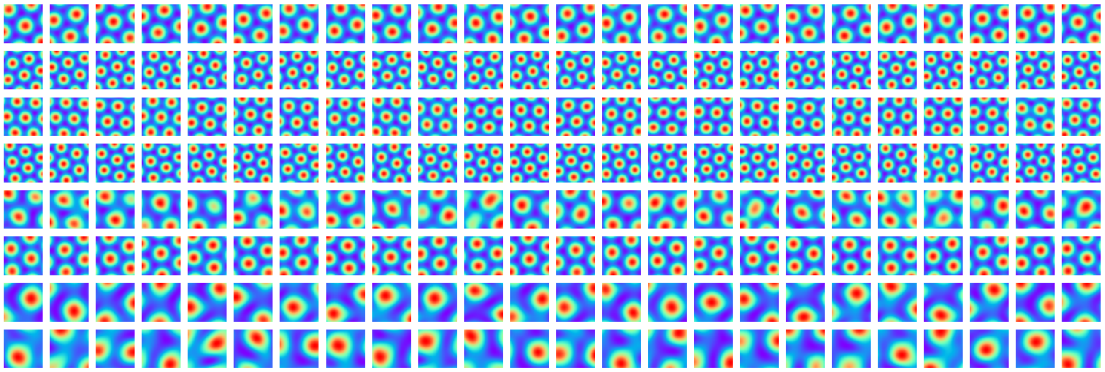


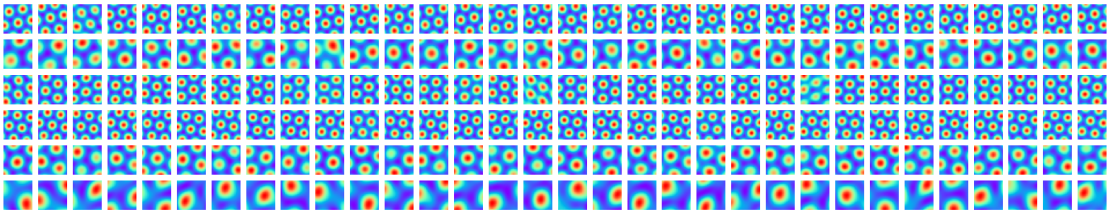
Figure 5.B.9: Learned neurons without skew-symmetric assumption of $\boldsymbol{B}(\theta)$. Hexagonal grid firing patterns emerge in most of the neurons, with a block of square grid firing patterns.



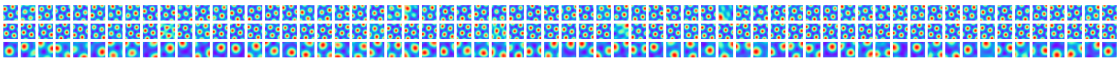
(a) Block size $= 16$



(b) Block size $= 24$



(c) Block size $= 32$



(d) Block size $= 64$

Figure 5.B.10: Learned patterns of $\boldsymbol{v}(\boldsymbol{x})$ with different block sizes. The total number of units is fixed at 192. Every row shows the learned patterns within the same block.

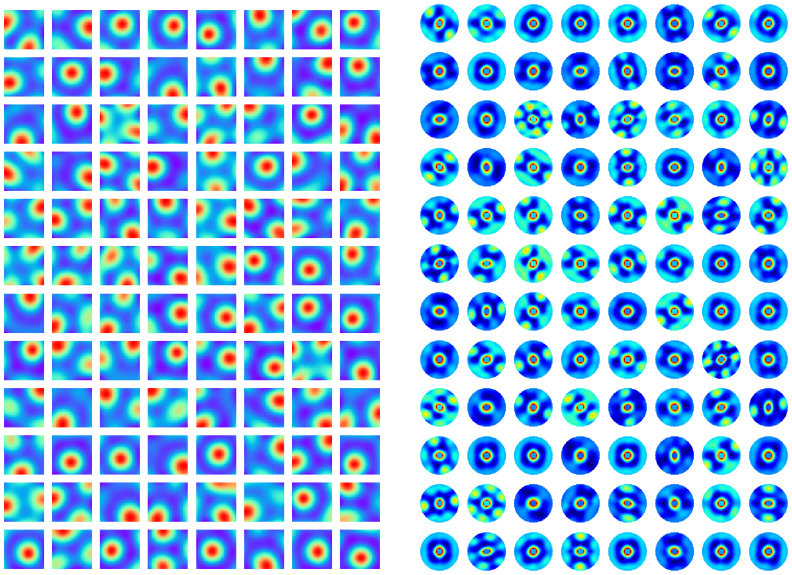


Figure 5.B.11: *Left*: learned neurons with a single block of $\boldsymbol{B}(\theta)$. The firing patterns has a single large scale, meaning that the high frequency part of $A(\boldsymbol{x}, \boldsymbol{x}')$ is not fitted very well. *Right*: autocorrelograms of the learned neurons. Some exhibit clear hexagon grid patterns, while the other do not, probably because the scale of those grid patterns are beyond the scope of the whole area.

CHAPTER 6

A representational model of V1 simple cells

6.1 Introduction

Our understanding of the primary visual cortex or V1 [HW59] is still very limited [OF05]. In particular, mathematical and representational models for V1 are still in short supply. Two prominent examples of such models are sparse coding [OF97] and independent component analysis (ICA) [BS97]. Although such models do not provide detailed explanations of V1 at the level of neuronal dynamics, they help us understand the computational problems being solved by V1.

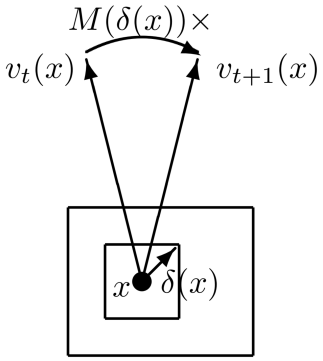


Figure 6.1.1: Scheme of representation

Inspired by the representational model of grid cells in the last chapter, we propose a representational model towards understanding V1, which is to model image pairs that are related by local pixel displacements[1]. The image pair can be consecutive frames of a video sequence, where the

[1]The main contributions in this chapter were first described in [GXZ18b].

local pixel displacements are caused by the relative motions between the agent and the objects in the 3D environment. Perceiving such local motions can be crucial for inferring ego-motion, object motions, and 3D depth information.

As is the case with existing models, we expect our model to explain only limited aspects of V1, some of which are: (1) The receptive fields of V1 simple cells resemble Gabor filters [Dau85]. (2) Adjacent simple cells have quadrature phase relationship [PR81]. (3) The V1 cells are capable of perceiving local motions. While existing models can all explain (1), our model can also account for (2) and (3) naturally. Compared to models such as sparse coding and ICA, our model serves a more direct purpose of perceiving local motions.

Our model consists of the following two components.

(1) Vector representation of local image content. The local content around each pixel is represented by a high-dimensional vector. Each unit in the vector is obtained by a linear filter. These local filters or wavelets are assumed to form a normalized tight frame, i.e., the image can be reconstructed from the vectors using the linear filters as the basis functions.

(2) Matrix representation of local displacement. The change of the image from the current time frame to the next time frame is caused by the displacements of the pixels. Each possible displacement is represented by a matrix that acts on the vector. When the image changes according to the displacements, the vector at each pixel is multiplied by the matrix that represents the local displacement, in other words, the vector at each pixel is rotated by the matrix representation of the displacement of this pixel.

See Figure 6.1.1 for an illustration, where the image is illustrated by the big rectangle. A pixel is illustrated by a dot. The local image content is illustrated by a small square around it. The displacement of the pixel is illustrated by a short arrow, which is within the small square. The vector representation of the local image content is represented by a long vector, which rotates as the image undergoes deformation due to the pixel displacements. Section 6.2 explains the notation.

One motivation of our work comes from Fourier analysis. An image patch $\mathbf{I}$ can be expressed

by the Fourier decomposition $\mathbf{I} = \sum_k c_k e^{i\langle \omega_k, x\rangle}$. Assuming the image patch undergoes a smooth motion so that all the pixels are shifted by a constant displacement dx, the shifted image patch $\mathbf{J}(x) = \mathbf{I}(x - dx) = \sum_k c_k e^{-i\langle \omega_k, dx\rangle} e^{i\langle \omega_k, x\rangle}$. The change from the complex number c_k to $c_k e^{-i\langle \omega_k, dx\rangle}$ corresponds to rotating a 2D vector by a 2×2 matrix. We emphasize that our model does not assume Fourier basis or its localized version such as Gabor filters. The model figures it out with generic vector and matrix representations.

We train this representational model on image pairs where in each pair, the second image is a deformed version of the first image, and the deformation is either known or inferred. We learn the encoding matrices for vector representation and the matrices that represent the pixel displacements from the training data.

Our experiments show that our method learns V1-like units that can be well approximated by Gabor filters with quadrature phase relationships. The profile of learned units matches one of the simple cells in Macaque V1 very well. After learning the encoding matrices for vector representation and the matrix representations of the displacements, we can infer the displacement field using the learned model. Compared to current optical flow estimation methods [DFI15, IMS17], which use complex deep neural networks to predict the optical flow, our model is much simpler and is based on explicit vector and matrix representations. We also demonstrate comparable results to these methods, in terms of the inference of the displacement field.

In terms of biological interpretation, the vectors can be interpreted as activities of groups of neurons, and the matrices can be interpreted as synaptic connections. See subsections 6.3.4 and 6.3.5 for details.

6.2 Representational model

6.2.1 Vector representation

Let $\{\mathbf{I}(\boldsymbol{x}), \boldsymbol{x} \in D\}$ be an image observed at a certain instant, where $\boldsymbol{x} = (x_1, x_2) \in D$ is the 2D coordinates of pixel. D is the image domain (e.g., 128×128). We represent the image $\mathbf{I}$ by vectors $\{\boldsymbol{v}(\boldsymbol{x}), \boldsymbol{x} \in D_-\}$, where each $\boldsymbol{v}(\boldsymbol{x})$ is a vector defined at pixel $\boldsymbol{x}$, and D_- may consist of a sub-sampled set of pixels in D (e.g., sub-sampled every 8 pixels). $\boldsymbol{V} = \{\boldsymbol{v}(\boldsymbol{x}), \boldsymbol{x} \in D_-\}$ forms a vector representation of the whole image.

We assume the vector encoding is linear and convolutional. Specifically, let $\mathbf{I}[\boldsymbol{x}]$ be a squared patch (e.g., 16×16) of $\mathbf{I}$ centered at $\boldsymbol{x}$. We can make $\mathbf{I}[\boldsymbol{x}]$ into a vector (e.g., 256 dimensional). Let

$$\boldsymbol{v}(\boldsymbol{x}) = \boldsymbol{W}\mathbf{I}[\boldsymbol{x}], \ \boldsymbol{x} \in D_-, \tag{6.1}$$

be the linear encoder, where $\boldsymbol{W}$ is the encoding matrix that encodes $\mathbf{I}[\boldsymbol{x}]$ into a vector $\boldsymbol{v}(\boldsymbol{x})$, and $\boldsymbol{W}$ is the same for all $\boldsymbol{x}$, i.e., convolutional. The rows of $\boldsymbol{W}$ are the linear filters and can be displayed as local image patches of the same size as the image patch $\mathbf{I}[\boldsymbol{x}]$. We can write $\boldsymbol{V} = \boldsymbol{W}\mathbf{I}$, if we treat $\mathbf{I}$ as a vector, and the rows of $\boldsymbol{W}$ are the shifted or translated versions of $\boldsymbol{W}$.

Tight frame and isometry. We assume that $\boldsymbol{W}$ is an auto-encoding normalized tight frame. Specifically, let $\boldsymbol{W}(\boldsymbol{x})$ denote the translation of filter $\boldsymbol{W}$ to pixel $\boldsymbol{x}$ and zero-padding the pixels outside the filters. Each column of $\boldsymbol{W}(\boldsymbol{x})$ is of the same dimension as $\mathbf{I}$, then

$$\mathbf{I} = \boldsymbol{W}^\top \boldsymbol{V} = \sum_{\boldsymbol{x}} \boldsymbol{W}^\top(\boldsymbol{x})\boldsymbol{v}(\boldsymbol{x}), \tag{6.2}$$

Thus, the linear filters for bottom-up encoding also serve as basis functions for top-down decoding. Both the encoder and decoder can be implemented by convolutional linear neural networks.

The normalized tight frame assumption can be justified by the fact that for two images $\mathbf{I}$ and $\mathbf{J}$, we have $\langle \boldsymbol{W}\mathbf{I}, \boldsymbol{W}\mathbf{J} \rangle = \mathbf{I}^\top \boldsymbol{W}^\top \boldsymbol{W}\mathbf{J} = \langle \mathbf{I}, \mathbf{J} \rangle$, that is, the vector representation preserves the inner

product. As a result, $\|\boldsymbol{W}\mathbf{I}\| = \|\mathbf{I}\|$, $\|\boldsymbol{W}\mathbf{J}\| = \|\mathbf{J}\|$, thus the vector representation also preserves the angle and has the isometry property.

When the image $\mathbf{I}$ changes from $\mathbf{I}_t$ to $\mathbf{I}_{t+1}$, its vector representation $\boldsymbol{V}$ changes from $\boldsymbol{V}_t$ to $\boldsymbol{V}_{t+1}$, and the angle between $\mathbf{I}_t$ and $\mathbf{I}_{t+1}$ is the same as the angle between $\boldsymbol{V}_t$ and $\boldsymbol{V}_{t+1}$.

Sub-vectors. The vector $\boldsymbol{v}(\boldsymbol{x})$ can be high-dimensional. We further divide $\boldsymbol{v}(\boldsymbol{x})$ into K sub-vectors, $\boldsymbol{v}(\boldsymbol{x}) = (\boldsymbol{v}^{(k)}(\boldsymbol{x}), k = 1, ..., K)$. Each sub-vector is obtained by an encoding sub-matrix $\boldsymbol{W}^{(k)}$, i.e., $\boldsymbol{v}^{(k)}(\boldsymbol{x}) = \boldsymbol{W}^{(k)}\mathbf{I}[\boldsymbol{x}]$, $k = 1, ..., K$, where $\boldsymbol{W}^{(k)}$ consists of the rows of $\boldsymbol{W}$ that correspond to $\boldsymbol{v}^{(k)}$. According to the normalized tight frame assumption, we have

$$\mathbf{I} = \sum_{\boldsymbol{x} \in D_-} \sum_{k=1}^{K} \boldsymbol{W}^{(k)\top} \boldsymbol{v}^{(k)}(\boldsymbol{x}). \tag{6.3}$$

In practice, we find that this assumption is necessary for the emergence of the V1-like receptive field.

6.2.2 Matrix representation

Let $\mathbf{I}_t$ be the image at time frame t. Suppose the pixels of $\mathbf{I}_t$ undergo local displacements, where the displacement at pixel $\boldsymbol{x}$ is $\boldsymbol{\delta}(\boldsymbol{x})$. We assume that $\boldsymbol{\delta}(\boldsymbol{x})$ is within a squared range Δ (e.g., $[-6, 6] \times [-6, 6]$ pixels) that is inside the range of $\mathbf{I}_t[\boldsymbol{x}]$ (e.g., 16×16 pixels). We assume the displacement field $(\boldsymbol{\delta}(\boldsymbol{x}), \forall \boldsymbol{x})$ is locally smooth, i.e., pixels within each local path undergoes similar displacements. Let $\mathbf{I}_{t+1}$ be the resulting image. Let $\boldsymbol{v}_t(\boldsymbol{x})$ be the vector representation of $\mathbf{I}_t[\boldsymbol{x}]$, and let $\boldsymbol{v}_{t+1}(\boldsymbol{x})$ be the vector representation of $\mathbf{I}_{t+1}[\boldsymbol{x}]$. Then $\boldsymbol{v}_t(\boldsymbol{x}) = (\boldsymbol{v}_t^{(k)}(\boldsymbol{x}), k = 1, ..., K)$, and $\boldsymbol{v}_{t+1}(\boldsymbol{x}) = (\boldsymbol{v}_{t+1}^{(k)}(\boldsymbol{x}), k = 1, ..., K)$.

The transition from $\mathbf{I}_t$ to $\mathbf{I}_{t+1}$ is illustrated by the following diagram:

$$\begin{array}{ccc} \boldsymbol{v}_t^{(k)}(\boldsymbol{x}) & \overset{\boldsymbol{M}^{(k)}(\boldsymbol{\delta}(\boldsymbol{x}))\times}{\longrightarrow} & \boldsymbol{v}_{t+1}^{(k)}(\boldsymbol{x}) \\ \boldsymbol{W}^{(k)} \uparrow & \uparrow & \uparrow \boldsymbol{W}^{(k)} \\ \mathbf{I}_t & \overset{\boldsymbol{\delta}(\boldsymbol{x})}{\longrightarrow} & \mathbf{I}_{t+1} \end{array} \tag{6.4}$$

Specifically, we assume that

$$\boldsymbol{v}_{t+1}^{(k)}(\boldsymbol{x}) = \boldsymbol{M}^{(k)}(\boldsymbol{\delta}(\boldsymbol{x}))\boldsymbol{v}_t^{(k)}(\boldsymbol{x}),\ \forall x \in D_-, k = 1, ..., K. \tag{6.5}$$

That is, when $\mathbf{I}$ changes from $\mathbf{I}_t$ to $\mathbf{I}_{t+1}$, $\boldsymbol{v}^{(k)}(\boldsymbol{x})$ undergoes a linear transformation, driven by a matrix $\boldsymbol{M}^{(k)}(\boldsymbol{\delta}(\boldsymbol{x}))$, which depends on the local displacement $\boldsymbol{\delta}(\boldsymbol{x})$. In terms of the whole vector $\boldsymbol{v}(\boldsymbol{x}) = (\boldsymbol{v}^{(k)}(\boldsymbol{x}), k = 1, ..., K)$, we have $\boldsymbol{v}_{t+1}(\boldsymbol{x}) = \boldsymbol{M}(\boldsymbol{\delta}(\boldsymbol{x}))\boldsymbol{v}_t(\boldsymbol{x})$, where $\boldsymbol{M}(\boldsymbol{\delta}(\boldsymbol{x})) = \text{diag}(\boldsymbol{M}^{(k)}(\boldsymbol{\delta}(\boldsymbol{x})), k = 1, ..., K)$ is the matrix representation of the local displacement $\boldsymbol{\delta}(\boldsymbol{x})$.

Disentangled rotations. The linear transformations of the sub-vectors $\boldsymbol{v}^{(k)}(\boldsymbol{x})$ can be considered as rotations. Here we use the word "rotation" in the loose sense without strictly enforcing $\boldsymbol{M}^{(k)}(\boldsymbol{\delta})$ to be orthogonal. $\boldsymbol{v}(\boldsymbol{x})$ is like a multi-arm clock, with each arm $\boldsymbol{v}^{(k)}(\boldsymbol{x})$ rotated by $\boldsymbol{M}^{(k)}(\boldsymbol{\delta}(\boldsymbol{x}))$. The rotations of $\boldsymbol{v}^{(k)}(\boldsymbol{x})$ for different k and $\boldsymbol{x}$ are disentangled. Here disentanglement means that the rotation of a sub-vector does not depend on other sub-vectors. This enables the agent to sense the displacement of a pixel only by sensing the rotations of the sub-vectors at this pixel without having to establish the correspondences between the pixels of consecutive frames.

Parametrization. We can discretize the displacement $\boldsymbol{\delta}(\boldsymbol{x})$ into a finite set of possible values $\{\boldsymbol{\delta}\}$, and we learn a separate $\boldsymbol{M}^{(k)}(\boldsymbol{\delta})$ for each $\boldsymbol{\delta}$. We can also learn a parametric version of $\boldsymbol{M}^{(k)}(\boldsymbol{\delta})$ as the second order Taylor expansion of a matrix-valued function of $\boldsymbol{\delta} = (\delta_1, \delta_2)$, $\boldsymbol{M}^{(k)}(\boldsymbol{\delta}) = \boldsymbol{I} + \boldsymbol{B}_1^{(k)}\delta_1 + \boldsymbol{B}_2^{(k)}\delta_2 + \boldsymbol{B}_{11}^{(k)}\delta_1^2 + \boldsymbol{B}_{22}^{(k)}\delta_2^2 + \boldsymbol{B}_{12}^{(k)}\delta_1\delta_2$, where $\boldsymbol{I}$ is the identity matrix, and $\boldsymbol{B}^{(k)} = (\boldsymbol{B}_1^{(k)}, \boldsymbol{B}_2^{(k)}, \boldsymbol{B}_{11}^{(k)}, \boldsymbol{B}_{22}^{(k)}, \boldsymbol{B}_{12}^{(k)})$ are matrices of coefficients of the same dimensionality as $\boldsymbol{M}^{(k)}(\boldsymbol{\delta})$.

Local mixing. If $\delta(\boldsymbol{x})$ is large, $\boldsymbol{v}_{t+1}^{(k)}(\boldsymbol{x})$ may contain information from adjacent image patches of $\mathbf{I}_t$ in addition to $\mathbf{I}_t[\boldsymbol{x}]$. We can generalize the motion model in Equation (6.5) to allow local mixing of encoded vectors. Let $\mathscr{S}$ be a local support centered at 0. We assume that

$$\boldsymbol{v}_{t+1}^{(k)}(\boldsymbol{x}) = \sum_{\mathrm{d}\boldsymbol{x}\in\mathscr{S}} M^{(k)}(\delta(\boldsymbol{x}), \mathrm{d}\boldsymbol{x})\boldsymbol{v}_t^{(k)}(\boldsymbol{x}+\mathrm{d}\boldsymbol{x}) \tag{6.6}$$

In the learning algorithm, we discretize $\mathrm{d}\boldsymbol{x}$ and learn a separate $M^{(k)}(\delta, \mathrm{d}\boldsymbol{x})$ for each $\mathrm{d}\boldsymbol{x}$.

6.3 Learning and Inference

The input data consist of the triplets $(\mathbf{I}_t, (\delta(\boldsymbol{x}), \boldsymbol{x}\in D_-), \mathbf{I}_{t+1})$, where $(\delta(\boldsymbol{x}))$ is the given displacement field. The learned model consists of matrices $(\boldsymbol{W}^{(k)}, \boldsymbol{M}^{(k)}(\delta), k = 1, ..., K, \delta\in\Delta)$, where Δ is the range of δ. In the case of parametric $\boldsymbol{M}^{(k)}$, we learn the $\boldsymbol{B}$ matrices in the second order Taylor expansion in subsection 6.2.2.

6.3.1 Loss functions for learning

We use the following loss functions:

(1) Rotation loss

$$L_{1,\boldsymbol{x},k} = \left\| \boldsymbol{W}^{(k)}\mathbf{I}_{t+1}[\boldsymbol{x}] - \boldsymbol{M}^{(k)}(\delta(\boldsymbol{x}))\boldsymbol{W}^{(k)}\mathbf{I}_t[\boldsymbol{x}] \right\|^2. \tag{6.7}$$

For local mixing generalization,

$$L_{1,\boldsymbol{x},k} = \left\| \boldsymbol{W}^{(k)}\mathbf{I}_{t+1}[\boldsymbol{x}] - \sum_{\mathrm{d}\boldsymbol{x}\in\mathscr{S}} \boldsymbol{M}^{(k)}(\delta(\boldsymbol{x}), \mathrm{d}\boldsymbol{x})\boldsymbol{W}^{(k)}\mathbf{I}_t(\boldsymbol{x}+\mathrm{d}\boldsymbol{x}) \right\|^2. \tag{6.8}$$

(2) Reconstruction loss

$$L_2 = \left\| \mathbf{I}_t - \sum_{\boldsymbol{x}\in D_-} \boldsymbol{W}^\top\boldsymbol{W}\mathbf{I}_t[\boldsymbol{x}] \right\|^2 + \left\| \mathbf{I}_{t+1} - \sum_{\boldsymbol{x}\in D_-} \boldsymbol{W}^\top\boldsymbol{W}\mathbf{I}_{t+1}[\boldsymbol{x}] \right\|^2. \tag{6.9}$$

In practice, we learn the model by a weighted sum of the expectations of $\sum_{k=1}^K \sum_{\boldsymbol{x}\in D_-} L_{1,\boldsymbol{x},k}$ and L_2, where the expectations are taken over the training pairs of images and the corresponding displacement fields.

6.3.2 Inference of motion

After learning $(W^{(k)}, M^{(k)}(\delta), \forall k, \forall \delta)$, for a testing pair $(\mathbf{I}_t, \mathbf{I}_{t+1})$, we can infer the pixel displacement field $(\delta(\boldsymbol{x}), \boldsymbol{x} \in D_-)$ by minimizing the rotation loss: $\delta(\boldsymbol{x}) = \arg\max_{\delta \in \Delta} L_{1,\boldsymbol{x}}(\delta)$, where

$$L_{1,\boldsymbol{x}}(\delta) = \sum_{k=1}^{K} \left\| \boldsymbol{W}^{(k)} \mathbf{I}_{t+1}[\boldsymbol{x}] - \boldsymbol{M}^{(k)}(\delta) \boldsymbol{W}^{(k)} \mathbf{I}_t[\boldsymbol{x}] \right\|^2 = \| \boldsymbol{W} \mathbf{I}_{t+1}[\boldsymbol{x}] - \boldsymbol{M}(\delta) \boldsymbol{W} \mathbf{I}_t[\boldsymbol{x}] \|^2. \quad (6.10)$$

This algorithm is efficient because it can be parallelized for all $\boldsymbol{x} \in D_-$ and for all $\delta \in \Delta$.

If we learn a parametric model for $(\boldsymbol{M}^{(k)}(\delta))$, we can infer the displacement field $(\delta(\boldsymbol{x}), \forall \boldsymbol{x})$ by minimizing $\sum_{\boldsymbol{x}} L_{1,\boldsymbol{x}}(\delta(\boldsymbol{x}))$ using gradient descent with an initialization of $(\delta(\boldsymbol{x}))$ from random small values. To encourage the smoothness of the displacement field, we add the penalty term $\| \nabla \delta(\boldsymbol{x}) \|^2$.

6.3.3 Unsupervised learning

We can easily adapt the learning of the model to an unsupervised manner, without knowing the pixel displacement field $(\delta(\boldsymbol{x}), \boldsymbol{x} \in D_-)$. Specifically, we can iterate the following two steps: (1) update model parameters by loss functions defined in section 6.3.1; (2) infer the displacement field as described in 6.3.2. To eliminate the ambiguity of the representation of $(\boldsymbol{M}^{(k)}(\delta))$ with respect to δ, we use a parametric model for $(\boldsymbol{M}^{(k)}(\delta))$, and add a regularization term $\| \mathbf{I}_{t+1} - \omega(\mathbf{I}_t, \delta) \|^2$ in the inference step, where $\omega(\cdot, \cdot)$ is a differentiable warping function. To summarize, we infer the displacement field $(\delta(\boldsymbol{x}), \boldsymbol{x} \in D_-)$ by minimizing:

$$\sum_{\boldsymbol{x}} L_{1,\boldsymbol{x}}(\delta) + \| \nabla \delta \|^2 + \| \mathbf{I}_{t+1} - \omega(\mathbf{I}_t, \delta) \|^2. \quad (6.11)$$

In practice, for each image pair at each iteration, we start the inference by running gradient descent on the inferred displacement field from the last iteration.

6.3.4 Biological interpretations of cells and synaptic connections

The learned $(\boldsymbol{W}^{(k)}, \boldsymbol{M}^{(k)}(\delta)), \forall k, \delta)$ can be interpreted as synaptic connections. For each k, $\boldsymbol{W}^{(k)}$ corresponds to one set of connection weights. Suppose $\delta \in \Delta$ is discretized, then for each δ,

$M^{(k)}(\delta)$ corresponds to one set of connection weights, and $(M^{(k)}(\delta), \delta \in \Delta)$ corresponds to multiple sets of connection weights. After computing $v^{(k)}_{t,x} = W^{(k)}\mathbf{I}_t[x]$, $M^{(k)}(\delta)v^{(k)}_{t,x}$ is computed simultaneously for every $\delta \in \Delta$. Then $\delta(x)$ is inferred by max pooling according to Equation (6.10).

$v^{(k)}_{t,x}$ can be interpreted as activities of simple cells, and $\|v^{(k)}_{t,x}\|^2$ can be interpreted as activity of a complex cell. If $M^{(k)}(\delta)$ is close to a rotation matrix, then we have norm stability so that $\|v^{(k)}_{t,x}\| \approx \|v^{(k)}_{t+1,x}\|$, which is related to the slowness property [HHV03, WS02].

6.3.5 Spatiotemporal filters and recurrent implementation

If we enforce norm stability or the orthogonality of $M^{(k)}(\delta)$, then minimizing $\|v_{t+1,x} - M(\delta)v_{t,x}\|^2$ over $\delta \in \Delta$ is equivalent to maximizing $\langle v_{t+1,x}, M(\delta)v_{t,x}\rangle$, which in turn is equivalent to maximizing $\|v_{t+1,x} + M(\delta)v_{t,x}\|^2$ so that $v_{t+1,x}$ and $M(\delta)v_{t,x}$ are aligned. This alignment criterion can be conveniently generalized to multiple consecutive frames, so that we can estimate the velocity at x by maximizing the m-step alignment score $\|u\|^2$, where

$$u = \sum_{i=0}^{m} M(\delta)^{m-i} v_{t+i,x} = \sum_{i=0}^{m} M(\delta)^{m-i} W \mathbf{I}_{t+i}[x] \tag{6.12}$$

consists of responses of spatiotemporal filters, and $\|u\|^2$ corresponds to the energy of motion δ in the motion energy model [AB85] for direction selective cells. Thus our model is connected with the motion energy model. Moreover, our model enables a recurrent network for computing u by $u_i = v_{t+i,x} + M(\delta)u_{i-1}$ for $i = 0, ..., m$, with $u_{-1} = 0$, and $u = u_m$. This recurrent implementation is much more efficient and biologically plausible than the plain implementation of spatiotemporal filtering which requires memorizing all the $\mathbf{I}_{t+i}$ for $i = 0, ..., m$. See [PS17] for a discussion of biological plausibility of recurrent implementation of spatiotemporal filtering in general.

6.4 Experiments

We learn our model $(\boldsymbol{W}^{(k)}, \boldsymbol{M}^{(k)}(\delta), k = 1, ..., K)$ from image pairs $(\mathbf{I}_t, \mathbf{I}_{t+1})$ with its displacement field $(\delta(\boldsymbol{x}))$. The number of sub-vectors $K = 40$, and the number of units in each sub-vector $\boldsymbol{v}^{(k)}(\boldsymbol{x})$ is 2. We use Adam [KB14] with learning rate $lr = 0.0008$ for updating the models. We conduct experiments on two synthetic datasets and two public datasets to demonstrate the efficacy of the proposed model. We also analyze the effects of the sub-vector dimensionality and sub-sampling rate in the ablation study.

Synthetic and public datasets

For the proposed model, we want to learn from local motions within local contents. However, existing datasets such as Flying Chairs [DFI15], FlyingThings3D [MIH16], and KITTI flow [GLU12] contain image pairs with large motions, which are unlikely consecutive frames of motion perception. To this end, we generate two synthetic datasets:

V1Deform. For this dataset, we consider random smooth deformations for natural images. Specifically, We obtain the training data by collecting static images for $(\mathbf{I}_t)$ and simulate the displacement field $(\delta(\boldsymbol{x}))$. The simulated displacement field is then used to transform $\mathbf{I}_t$ to obtain $\mathbf{I}_{t+1}$. We retrieve natural images as $\mathbf{I}_t$ from MIT places365 dataset [ZKL16]. The images are scaled to 128×128. We sub-sample the pixels of images into a $m \times m$ grid ($m = 4$ in the experiments), and randomly generate displacements on the grid points, which serve as the control points for deformation. Then $\delta(\boldsymbol{x})$ for $\boldsymbol{x} \in D$ can be obtained by spline interpolation of the displacements on the control points. We get $\mathbf{I}_{t+1}$ by warping $\mathbf{I}_t$ using $\delta(\boldsymbol{x})$ [JSZ15]. When generating a displacement $\delta = (\delta_1, \delta_2)$, both δ_1 and δ_2 are randomly sampled from a range of $[-6, +6]$. We synthesize $20,000$ pairs for training and $3,000$ pairs for testing.

V1FlyingObjects. For this dataset, we consider separating the displacement field into motions of the background and foreground, to jointly simulate the self-motion of the agent and the motion of the objects in the natural 3D scenes. To this end, we apply affine transformations to

background images collected from MIT places365 [ZKL16] and foreground objects from a public 2D object dataset COIL-100 [NNM96]. The background images are scaled to 128×128, and the foreground images are randomly rescaled. To generate motion, we randomly sample affine parameters of translation, rotation, and scaling for both the foreground and background images. The motions of the foreground objects are relative to the background images, which can be explained as the relative motion between the moving object and agent. We tune the distribution of the affine parameters to keep the range of the displacement fields within $[-6, +6]$, which is consistent with the V1Deform dataset. Together with the mask of the foreground object and the sampled transformation parameters, we render the image pair $(\mathbf{I}_t, \mathbf{I}_{t+1})$ and its displacement field $(\boldsymbol{\delta}(\boldsymbol{x}))$ for each pair of the background image and foreground image.

For the foreground objects, we obtain the estimated masks from [tev06], resulting in 96 objects with 72 views per object available. We generate $14,411$ synthetic image pairs with their corresponding displacement fields and further split $12,411$ pairs for training and $2,000$ pairs for testing. Compared with previous optical flow datasets like Flying Chairs [DFI15] and scene flow datasets like FlyingThings3D [MIH16], the proposed V1FlyingObjects dataset has various foreground objects with more realistic texture and smoother displacement fields, which simulates more realistic environments.

We shall release the two synthetic datasets, which are suitable for studying local motions and perceptions. Besides, we also use two public datasets:

MPI-Sintel. MPI-Sintel [BWS12, WBS12] is a public dataset designed for the evaluation of optical flow derived from rendered artificial scenes, with special attention to realistic image properties. Since MPI-Sintel is relatively small, which contains around a thousand image pairs, we use it only for testing the learned models in the inference of the displacement field. We use the final version of MPI-Sintel and resize each frame into a size of 128×128. We select frame pairs whose motions are within the range of $[-6, +6]$, resulting in 384 frame pairs in total.

MUG Facial Expression. MUG Facial Expression dataset [APD10] records natural facial expression videos of 86 subjects sitting in front of one camera. This dataset has no ground truth

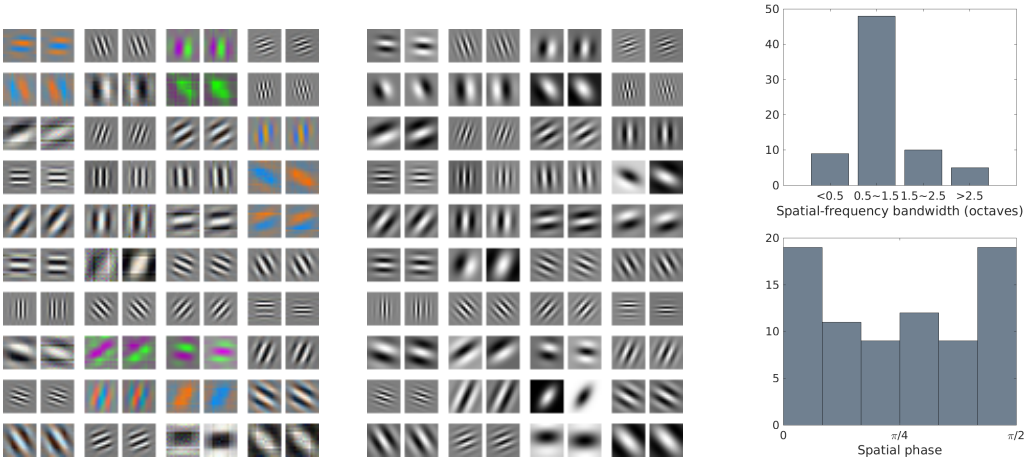
of the displacement field, which we use for unsupervised learning. 200 videos with 30 frames are randomly selected for training, and another 100 videos are sampled for testing.

Learned Gabor-like units with quadrature phase relationship

In this section, we show and analyze the learned units. The size of the filter is 16×16, with a sub-sampling rate of 8 pixels. Figure 6.4.1(a) displays the learned units, i.e., rows of $\boldsymbol{W}^{(k)}$, on V1Deform. The units are learned with non-parametric $M(\delta)$, i.e., we learn a separate $\boldsymbol{M}(\delta)$ for each displacement. $\delta(\boldsymbol{x})$ is discretized with an interval of 0.5. Similar patterns can be obtained by using a parametric version of $\boldsymbol{M}(\delta)$. V1-like patterns emerge from the learned units. Moreover, within each sub-vector, the orientations and frequencies of learned units are similar, while the phases are different.

To further analyze the spatial profile of the learned units, we fit every unit by a two dimensional Gabor function [JP87]: $h(x',y') = A\exp(-(x'/\sqrt{2}\sigma_{x'})^2 - (y'/\sqrt{2}\sigma_{y'}))\cos(2\pi f x' + \phi)$, where (x',y') is obtained by translating and rotating the original coordinate system (x_0,y_0): $x' = (x-x_0)\cos\theta + (y-y_0)\sin\theta, y' = -(x-x_0)\sin\theta + (y-y_0)\cos\theta$. The fitted Gabor patterns are shown in Figure 6.4.1(b), with the average fitting r^2 equal to 0.96 (std = 0.04). The average spatial-frequency bandwidth is 1.13 octaves, with range of 0.12 to 4.67. Figure 6.4.1(c) shows the distribution of the spatial-frequency bandwidth, where the majority falls within range of 0.5 to 2.5. The characteristics are reasonably similar to those of simple-cell receptive fields in the cat [ITS00] (weighted mean 1.32 octaves, range of 0.5 to 2.5) and the macaque monkey [FGN85] (median 1.4 octaves, range of 0.4 to 2.6). To analyze the distribution of the spatial phase ϕ, we follow the method in [Rin02] to transform the parameter ϕ into an effective range of 0 to $\pi/2$, and plot the histogram of the transformed ϕ in Figure 6.4.1(c). The strong bimodal with phases clustering near 0 and $\pi/2$ is consistent with those of the macaque monkey [Rin02].

In the above experiment, we fix the size of the convolutional filters (16×16 pixels). A more reasonable model is to have different sizes of convolutional filters, with small size filters capturing



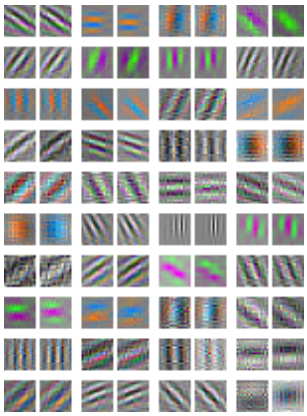
(a) Learned units (b) Fitted Gabors (c) Frequency and phase

Figure 6.4.1: Learned results on V1Deform. (a) Learned units. Each block shows two learned units within the same sub-vector. (b) Fitted Gabor patterns. (c) Distributions of spatial-frequency bandwidth (in octaves) and spatial phase ϕ.

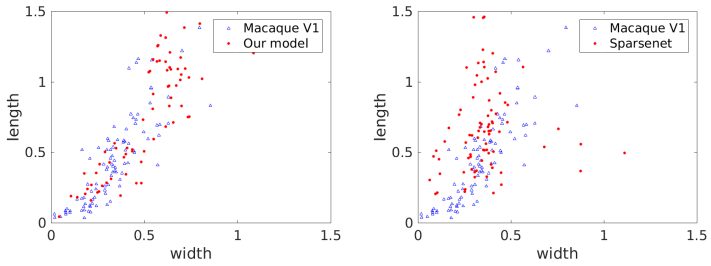
high-frequency content and large size filters capturing low-frequency content. For fixed-size filters, they should only account for the image content within a frequency band. To this end, we smooth every image by two Gaussian smoothing kernels (kernel size 8, $\sigma = 1, 4$), and take the difference between the two smoothed images as the input image of the model. The effect of the two smoothing kernels is similar to a bandpass filter so that the input images are constrained within a certain range of frequencies. The learned filters on V1Deform are shown in Figure 6.4.2(a). Again for every unit, we fit it by a two dimensional Gabor function, resulting in an average fitting $r^2 = 0.83$ (std $= 0.12$). Following the analysis of [Rin02, RS07], a scatter plot of $n_x = \sigma_x f$ versus $n_y = \sigma_y f$ is constructed in Figure 6.4.2(b) based on the fitted parameters, where n_x and n_y represent the width and length of the Gabor envelopes measured in periods of the cosine waves. Compared to Sparsenet [OF96, OF97], the units learned by our model have more similar structure to the receptive fields of simples cells of Macaque monkey.

We also compare the learned units within each sub-vector in Figure 6.4.2(c). Within each sub-vector, the frequency f and orientation θ of the paired units tend to be the same. More importantly,

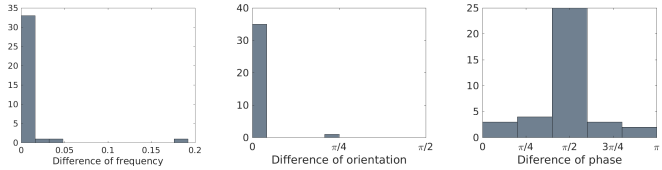
most of the paired units differ in phase ϕ by approximately $\pi/2$, consistent with the quadratic phase relationship between adjacent simple cells [PR81, EH97].



(a) Learned units



(b) Gabor envelope shapes of the learned units



(c) Profile of paired units within each sub-vector

Figure 6.4.2: Learned results on band-pass image pairs from V1Deform. (a) Learned units. Each block shows two learned units within the same sub-vector. (b) Distribution of the Gabor envelope shapes in the width and length 2D-plane. (c) Difference of frequency f, orientation θ and phase ϕ of paired units within each sub-vector.

Inference of displacement field

We test the learned representations in terms of inferring the displacement field $(\delta(\boldsymbol{x}))$ between pairs of frames $(\mathbf{I}_t, \mathbf{I}_{t+1})$. To get valid image patches for inference, we leave out those displacements at image border (8 pixels at each side).

We use non-parametric $M(\delta)$ and the local mixing motion model (Equation (6.6)), where the local support $\mathscr{S}$ is in a range of $[-4, +4]$, and $\mathrm{d}\boldsymbol{x}$ is taken with a sub-sampling rate of 2. After obtaining the inferred displacement field $(\delta(\boldsymbol{x}))$ by the learned model, we also train a CNN model with ResNet blocks [HZR16b] to refine the inferred displacement field. In training this CNN, the input is the inferred displacement field, and the output is the ground truth

displacement field, with least-squares regression loss. This refinement CNN is to approximate the processing in visual areas V2-V6 that integrates and refines the motion perception in V1 [GIM02, LK02, MD85, BB05, AK75]. We learn the models from the training sets of V1Deform and V1FlyingObjects datasets respectively, and test on the corresponding test sets. We also test the model learned from V1FlyingObjects on MPI-Sintel.

Table 6.4.1 summarizes the average endpoint error (AEE) of the inferred results. We compare with several baseline methods, including FlowNet 2.0 and its variants [DFI15, IMS17], for which we test both the trained models ('trained') on our datasets and the released models pre-trained on large scale datasets ('pre-trained'). Note that for MPI-Sintel, a pre-trained baseline model gives better performance compared to the one trained on V1FlyingObjects, probably because these methods train deep and complicated neural networks with a large amount of parameters to predict optical flows in supervised manners, which may require large scale data to fit and transfer to different domains. On the other hand, our model can be treated as a simple one-layer auto-encoder network, accompanied by weight matrices representing motions. As shown in Table 6.4.1, our model has about 88 times fewer parameters than FlowNet 2.0 and 21 times fewer parameters than the light FlowNet2-C model. We achieve competitive performance compared to these baseline methods. Figure 6.4.3 displays several examples of the inferred displacement field. Inferred results from the FlowNet 2.0 models are shown as a qualitative comparison. For each dataset, we show the result of the FlowNet 2.0 model with lower AEE among the pre-trained and trained ones.

Unsupervised learning

We further test the unsupervised learning of the proposed method. For unsupervised learning, we scale the images to size 64×64. The size of the filter is 8×8, with a sub-sampling rate of 4 pixels. Displacements at the image border (4 pixels at each side) are left out. We train the model on MUG Facial Expression and V1FlyingObjects datasets. Figure 6.4.4 shows some examples of inferred displacement fields on the testing set of MUG Facial Expression. The inference results are reasonable, which capture the motions around eyes, eyebrows, chin, or mouth. For the model

Figure 6.4.3: Examples of inferred displacement field on V1Deform, V1FlyingObjects and MPI-Sintel. For each block, from left to right are $\mathbf{I}_t$, $\mathbf{I}_{t+1}$, ground truth displacement field and inferred displacement field by FlowNet 2.0 model and our learned model respectively. For each dataset, we show the result of FlowNet 2.0 model with lower AEE among the pre-trained and trained ones. The displacement fields are color coded [LYT10].

trained on V1FlyingObjects, we test on the testing set of V1FlyingObjects and MPI-Sintel. Table 6.4.2 summarizes the quantitative results. We include comparisons with several baseline methods for unsupervised optical flow estimation: Unsup [JHD16] and UnFlow [MHR18] and its variants, which are also trained on V1FlyingObjects. The proposed model achieves better performance compared with baseline models.

Table 6.4.1: Average endpoint error of the inferred displacement and number of paramters. (Abbreviation FN2 refers to FlowNet 2.0.)

	V1Deform		V1FlyingObjects		MPI-Sintel		# param. (M)
	pre-trained	trained	pre-trained	trained	pre-trained	trained	
FN2-C	1.324	1.130	0.852	1.034	0.363	0.524	39.18
FN2-S	1.316	0.213	0.865	0.261	0.410	0.422	38.68
FN2-CS	0.713	0.264	0.362	0.243	0.266	0.346	77.87
FN2-CSS	0.629	0.301	0.299	0.303	0.234	0.450	116.57
FN2	0.686	0.205	0.285	0.265	0.146	0.278	162.52
Ours	-	0.258	-	0.442	-	0.337	1.82
Ours-ref	-	**0.156**	-	**0.202**	-	**0.140**	1.84

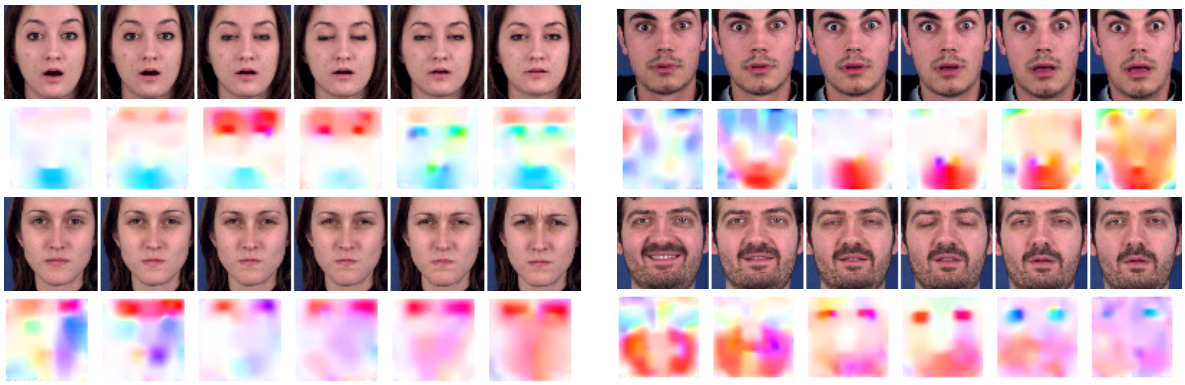


Figure 6.4.4: Examples of inferred displacement fields by unsupervised learning. Within each block, the top row shows the observed image frames, while the bottom row shows the inferred color-coded displacement fields [LYT10].

Ablation Study

We perform ablation studies to analyze the effect of two components of the proposed model: (1) dimensionality of sub-vectors; (2) sub-sampling rate. Besides comparing AEE of motion inference, we also test if the learned model can make multi-step animations of image frames given the sequence of displacement fields, by predicting the vector representation consecutively and decod-

Table 6.4.2: Average endpoint error of the inferred displacement in unsupervised learning.

	Unsup	UnFlow-C	UnFlow-CS	UnFlow-CSS	Ours
V1FlyingObjects (train)	0.379	0.336	0.374	0.347	**0.245**
V1FlyingObjects (test)	0.811	0.399	0.394	0.453	**0.316**
MPI-Sintel	0.440	0.198	0.248	0.202	**0.101**

ing to image frames. Per-pixel mean squared error (MSE) of the predicted next five image frames is reported. Table 6.4.3 summarizes the results learned from V1Deform. The dimensionality of sub-vectors controls the complexity of the motion matrices, which is set to a minimum of 2 in the experiments. As the dimensionality of sub-vectors increases, the error rates of the two tasks decrease first and then increase. We emphasize that with higher dimensionality of sub-vectors, e.g., 4 or 6, Gabor-like patterns still emerge. On the other hand, the sub-sampling rate can be changed to make the adjacent image patches connect with each other more loosely or tightly. As shown in Table 6.4.3, a sub-sampling rate of 8, which is half of the filter size, leads to the optimal performance.

Table 6.4.3: Ablation study measured by AEEs of motion inference and MSEs of multi-step animation, learned from V1Deform dataset.

	Sub-vector dimension				
	2	4	6	8	12
Infer AEE	0.44	0.421	0.406	0.412	0.420
Animation MSE	8.122	7.986	7.125	7. 586	8.017

	Sub-sampling rate		
	4	8	16
Infer AEE	0.654	0.444	0.532
Animation MSE	11.139	8.122	10.293

6.5 Conclusion

In this chapter we propose a simple representational model that couples vector representations of local image contents and matrix representations of local motions. Our model learns Gabor-like

units with quadrature phases. We also give biological interpretations of the learned model and connect it to the spatiotemporal energy model. Our model is novel, and it is our hope that it adds to our understanding of motion perception in V1 in terms of modeling and inference.

CHAPTER 7

Conclusion

Unsupervised learning and representation learning form an important aspect of modern machine learning. With the development of deep learning and advance in computational efficiency, the models in this area become incredibly flexible leading to important applications in computer vision and computational neuroscience. In this thesis, we demonstrate novel models and algorithms to push the development of unsupervised learning and representation learning.

One theme of this thesis is deep generative models, which is an important family of models in unsupervised learning. We are mainly interested in developing efficient and scalable learning algorithms of the descriptive model and generator model. Our contributions include: (1) with multi-grid modeling and sampling, we show that it is possible to train and sample from a deep descriptive model by maximum likelihood type learning algorithm, with small budget MCMC. (2) By optimizing diffusion recovery likelihoods, we further improve the learning of descriptive models by improving the sampling as sequential conditional sampling. Moreover, our long-run MCMC samples do not diverge and still represent realistic images, allowing us to accurately estimate the normalized density of data even for high-dimensional datasets. (3) We propose a dynamic generator model for spatial-temporal processes, and develop an efficient learning algorithm, namely alternating back-propagation through time. Such an algorithm is capable of estimating the dynamic generator model without recruiting any auxiliary model like in VAE or GAN. The model is further generalized to a motion-based dynamic generator model to learn disentangled representations of appearance, trackable and intrackable motion in a fully unsupervised manner. (4) We propose two hybrid models that couple a descriptive model and a flow-based model to combine the best of two

worlds. The first one (flow contrastive estimation) is to jointly estimate a descriptive model and a flow-based model by optimizing a shared adversarial value function. Such a model enables efficient estimation of descriptive models without MCMC, and achieves competitive performance on semi-supervised learning. The second one is to learn a descriptive model with a flow-based model serving as a backbone, so that the descriptive model is a correction or an exponential tilting of the flow-based model. With this model, we show that sampling of the descriptive model can mix well and traverse different modes by neural transport MCMC.

The other theme of this thesis is representation learning that is inspired by and can meaningfully approximate structures and activities in the brain. We are particularly interested in two types of neurons: (1) grid cells in the mammalian medial entorhinal cortex (mEC) which is hypothesized to be involved in mental navigation, or more precisely the path integration process. (2) V1 simple cells in the mammalian primary visual cortex, which is essential for low-level motion perception and pattern recognition. Our contributions include: (1) we conduct theoretical analysis of a general representation model of path integration by grid cells. We identify two conditions on the transformation under which we demonstrate that the local geometry of the neural space is a conformal embedding of the 2D physical space. Then we investigate the simplest transformation, i.e., the linear transformation, and uncover its explicit algebraic and geometric structure as a matrix Lie group of rotation. With our optimization-based approach, we manage to learn hexagon grid patterns that share similar properties of the grid cells in the rodent brain. The learned model is capable of accurate long-distance path integration. (2) We propose a representational model to shed light on V1 simple cells. The local contents of images are presented by vectors, and the local pixel displacements are represented by matrices operated on the vectors. We show that by training on consecutive video frames, our model can learn Gabor-like filter pairs of quadrature phases, which closely match the profile of Macaques V1 simple cells. The learned model achieves competitive performance on optical flow estimation.

REFERENCES

[AA92] Daniel J Amit and Daniel J Amit. *Modeling brain function: The world of attractor neural networks*. Cambridge university press, 1992.

[AB85] Edward H Adelson and James R Bergen. "Spatiotemporal energy models for the perception of motion." *Josa a*, **2**(2):284–299, 1985.

[AB20] Haggai Agmon and Yoram Burak. "A theory of joint attractor dynamics in the hippocampus and the entorhinal cortex accounts for artificial remapping and grid cell field-to-field variability." *eLife*, **9**:e56894, 2020.

[ACB17a] Martin Arjovsky, Soumith Chintala, and Bottou. "Wasserstein gan." *arXiv preprint arXiv:1701.07875*, 2017.

[ACB17b] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein gan." *arXiv preprint arXiv:1701.07875*, 2017.

[AIL09] Licurgo de Almeida, Marco Idiart, and John E Lisman. "The input–output transformation of the hippocampal granule cells: from grid cells to place fields." *Journal of Neuroscience*, **29**(23):7504–7512, 2009.

[AK75] JM Allman and JH Kass. "The dorsomedial cortical visual area: a third tier area in the occipital lobe of the owl monkey (Aotus trivirgatus)." *Brain research*, **100**(3):473–487, 1975.

[APD10] Niki Aifanti, Christos Papachristou, and Anastasios Delopoulos. "The MUG facial expression database." In *11th International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS 10*, pp. 1–4. IEEE, 2010.

[BB05] Richard T Born and David C Bradley. "Structure and function of visual area MT." *Annu. Rev. Neurosci.*, **28**:157–189, 2005.

[BBU18] Andrea Banino, Caswell Barry, Benigno Uria, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin J Chadwick, Thomas Degris, Joseph Modayil, et al. "Vector-based navigation using grid-like representations in artificial agents." *Nature*, **557**(7705):429, 2018.

[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.

[BDJ18] Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. "Invertible residual networks." *arXiv preprint arXiv:1811.00995*, 2018.

[Bes74] Julian Besag. "Spatial interaction and the statistical analysis of lattice systems." *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 192–236, 1974.

[BF09] Yoram Burak and Ila R Fiete. "Accurate path integration in continuous attractor network models of grid cells." *PLoS computational biology*, **5**(2):e1000291, 2009.

[BHB07] Caswell Barry, Robin Hayman, Neil Burgess, and Kathryn J Jeffery. "Experience-dependent rescaling of entorhinal grids." *Nature neuroscience*, **10**(6):682–684, 2007.

[BS97] Anthony J Bell and Terrence J Sejnowski. "The independent components of natural scenes are edge filters." *Vision research*, **37**(23):3327–3338, 1997.

[BWS12] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. "A naturalistic open source movie for optical flow evaluation." In A. Fitzgibbon et al. (Eds.), editor, *European Conf. on Computer Vision (ECCV)*, Part IV, LNCS 7577, pp. 611–625. Springer-Verlag, October 2012.

[BWZ07] Hugh T Blair, Adam C Welday, and Kechen Zhang. "Scale-invariant memory representations emerge from moire interference between grid fields that produce theta oscillations: a computational model." *Journal of Neuroscience*, **27**(12):3211–3229, 2007.

[BYA13] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. "Generalized denoising auto-encoders as generative models." In *Advances in neural information processing systems*, pp. 899–907, 2013.

[CGR19] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. "Generating long sequences with sparse transformers." *arXiv preprint arXiv:1904.10509*, 2019.

[CW18] Christopher J Cueva and Xue-Xin Wei. "Emergence of grid-like representations by training recurrent neural networks to perform spatial localization." *arXiv preprint arXiv:1803.07770*, 2018.

[CWC20] Christopher J Cueva, Peter Y Wang, Matthew Chin, and Xue-Xin Wei. "Emergence of functional and structural properties of the head direction system by optimization of recurrent neural networks." *International Conferences on Learning Representations (ICLR)*, 2020.

[CWZ13] Jonathan J Couey, Aree Witoelar, Sheng-Jia Zhang, Kang Zheng, Jing Ye, Benjamin Dunn, Rafal Czajkowski, May-Britt Moser, Edvard I Moser, Yasser Roudi, et al. "Recurrent inhibitory circuitry as a mechanism for grid formation." *Nature neuroscience*, **16**(3):318–324, 2013.

[CZS20] Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. "Your GAN is Secretly an Energy-based Model and You Should use Discriminator Driven Latent Sampling." *arXiv preprint arXiv:2003.06060*, 2020.

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

[Dar73] Charles Darwin. "Origin of certain instincts.", 1873.

[Dau85] John G Daugman. "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters." *JOSA A*, **2**(7):1160–1169, 1985.

[DBB10] Christian F Doeller, Caswell Barry, and Neil Burgess. "Evidence for grid cells in a human memory network." *Nature*, **463**(7281):657, 2010.

[DBM19] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. "Neural spline flows." In *Advances in Neural Information Processing Systems*, pp. 7509–7520, 2019.

[DCW03] Gianfranco Doretto, Alessandro Chiuso, Ying Nian Wu, and Stefano Soatto. "Dynamic textures." *International Journal of Computer Vision*, **51**(2):91–109, 2003.

[DFI15] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. v.d. Smagt, D. Cremers, and T. Brox. "FlowNet: Learning Optical Flow with Convolutional Networks." In *IEEE International Conference on Computer Vision (ICCV)*, 2015.

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

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

[DM19] Yilun Du and Igor Mordatch. "Implicit generation and generalization in energy-based models." *arXiv preprint arXiv:1903.08689*, 2019.

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

[DSM16] Yedidyah Dordek, Daniel Soudry, Ron Meir, and Dori Derdikman. "Extracting grid cell characteristics from place cell inputs using non-negative principal component analysis." *Elife*, **5**:e10094, 2016.

[EH97] Robert C Emerson and Morgan C Huang. "Quadrature subunits in directionally selective simple cells: counterphase and drifting grating responses." *Visual neuroscience*, **14**(2):373–385, 1997.

[EJ04] Ariane S Etienne and Kathryn J Jeffery. "Path integration in mammals." *Hippocampus*, **14**(2):180–192, 2004.

[FBB08] Ila R Fiete, Yoram Burak, and Ted Brookings. "What grid cells convey about rat location." *Journal of Neuroscience*, **28**(27):6858–6871, 2008.

[FCA16] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. "A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models." *arXiv preprint arXiv:1611.03852*, 2016.

[FGN85] KH Foster, James P Gaska, M Nagler, and DA Pollen. "Spatial and temporal frequency selectivity of neurones in visual cortical areas V1 and V2 of the macaque monkey." *The Journal of physiology*, **365**(1):331–363, 1985.

[FHW08] Marianne Fyhn, Torkel Hafting, Menno P Witter, Edvard I Moser, and May-Britt Moser. "Grid cells in mice." *Hippocampus*, **18**(12):1230–1238, 2008.

[FMW04] Marianne Fyhn, Sturla Molden, Menno P Witter, Edvard I Moser, and May-Britt Moser. "Spatial representation in the entorhinal cortex." *Science*, **305**(5688):1258–1264, 2004.

[FR12] Charles W Fox and Stephen J Roberts. "A tutorial on variational Bayesian inference." *Artificial intelligence review*, **38**(2):85–95, 2012.

[FT06] Mark C Fuhs and David S Touretzky. "A spin glass model of path integration in rat medial entorhinal cortex." *Journal of Neuroscience*, **26**(16):4266–4276, 2006.

[GA10] Bernard Ghanem and Narendra Ahuja. "Maximum margin distance learning for dynamic texture recognition." In *European Conference on Computer Vision (ECCV)*, pp. 223–236, 2010.

[GAA17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. "Improved training of wasserstein gans." In *Advances in neural information processing systems*, pp. 5767–5777, 2017.

[GAS18] Mariana Gil, Mihai Ancau, Magdalene I Schlesiger, Angela Neitz, Kevin Allen, Rodrigo J De Marco, and Hannah Monyer. "Impaired path integration in mice with disrupted grid cell firing." *Nature neuroscience*, **21**(1):81–91, 2018.

[GBS07] Lena Gorelick, Moshe Blank, Eli Shechtman, Michal Irani, and Ronen Basri. "Actions as space-time shapes." *IEEE transactions on pattern analysis and machine intelligence*, **29**(12):2247–2253, 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.

[GCB18] Will Grathwohl, Ricky TQ Chen, Jesse Betterncourt, Ilya Sutskever, and David Duvenaud. "Ffjord: Free-form continuous dynamics for scalable reversible generative models." *arXiv preprint arXiv:1810.01367*, 2018.

[GH10] Michael Gutmann and Aapo Hyvärinen. "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models." In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 297–304, 2010.

[GIM02] Michael S Gazzaniga, RB Ivry, and GR Mangun. "Cognitive Neuroscience, New York: W. W.", 2002.

[GLU12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite." In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[GLZ18] Ruiqi Gao, Yang Lu, Junpei Zhou, Song-Chun Zhu, and Ying Nian Wu. "Learning generative convnets via multi-grid modeling and sampling." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9155–9164, 2018.

[GNK19] Ruiqi Gao, Erik Nijkamp, Diederik P Kingma, Zhen Xu, Andrew M Dai, and Ying Nian Wu. "Flow Contrastive Estimation of Energy-Based Models." *arXiv preprint arXiv:1912.00589*, 2019.

[GNK20] Ruiqi Gao, Erik Nijkamp, Diederik P Kingma, Zhen Xu, Andrew M Dai, and Ying Nian Wu. "Flow contrastive estimation of energy-based models." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7518–7528, 2020.

[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.

[Gra13] Alex Graves. "Generating sequences with recurrent neural networks." *arXiv preprint arXiv:1308.0850*, 2013.

[GS89] Jonathan Goodman and Alan D Sokal. "Multigrid monte carlo method. conceptual foundations." *Physical Review D*, **40**(6):2035, 1989.

[GSP20] Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P Kingma. "Learning Energy-Based Models by Diffusion Recovery Likelihood." *arXiv preprint arXiv:2012.08125*, 2020.

[GWJ19] Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. "Your Classifier is Secretly an Energy Based Model and You Should Treat it Like One." *arXiv preprint arXiv:1912.03263*, 2019.

[GXW20] Ruiqi Gao, Jianwen Xie, Xue-Xin Wei, Song-Chun Zhu, and Ying Nian Wu. "On Path Integration of Grid Cells: Group Representation and Isotropic Scaling." *arXiv preprint arXiv:2006.10259*, 2020.

[GXZ18a] Ruiqi Gao, Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. "Learning grid cells as vector representation of self-position coupled with matrix representation of self-motion." *arXiv preprint arXiv:1810.05597*, 2018.

[GXZ18b] Ruiqi Gao, Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. "Learning Vector Representation of Content and Matrix Representation of Change: Towards a Representational Model of V1." *arXiv preprint arXiv:1902.03871*, 2018.

[GZ12] Haifeng Gong and Song-Chun Zhu. "Intrackability: Characterizing video statistics and pursuing video representations." *International journal of computer vision*, **97**(3):255–275, 2012.

[HBZ16] Aidan J Horner, James A Bisby, Ewa Zotow, Daniel Bush, and Neil Burgess. "Grid-like processing of imagined navigation." *Current Biology*, **26**(6):842–847, 2016.

[HCS19] Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. "Flow++: Improving flow-based generative models with variational dequantization and architecture design." *arXiv preprint arXiv:1902.00275*, 2019.

[HFM05] Torkel Hafting, Marianne Fyhn, Sturla Molden, May-Britt Moser, and Edvard I Moser. "Microstructure of a spatial map in the entorhinal cortex." *Nature*, **436**(7052):801, 2005.

[HG14] Matthew D Hoffman and Andrew Gelman. "The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo." *J. Mach. Learn. Res.*, **15**(1):1593–1623, 2014.

[HHV03] Aapo Hyvärinen, Jarmo Hurri, and Jaakko Väyrynen. "Bubbles: a unifying framework for low-level statistical properties of natural image sequences." *JOSA A*, **20**(7):1237–1252, 2003.

[Hin02a] Geoffrey E Hinton. "Training products of experts by minimizing contrastive divergence." *Neural Computation*, **14**(8):1771–1800, 2002.

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

[HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models." *arXiv preprint arXiv:2006.11239*, 2020.

[HLX19] Tian Han, Yang Lu, Xianglei Xing, and Ying Nian Wu. "Learning Generator Networks for Dynamic Patterns." In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[HLZ17] Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. "Alternating Back-Propagation for generator network." In *31st AAAI Conference on Artificial Intelligence*, 2017.

[HNF18] Tian Han, Erik Nijkamp, Xiaolin Fang, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. "Divergence triangle for joint training of generator model, energy-based model, and inference model." *arXiv preprint arXiv:1812.10907*, 2018.

[HNZ20] Tian Han, Erik Nijkamp, Linqi Zhou, Bo Pang, Song-Chun Zhu, and Ying Nian Wu. "Joint Training of Variational Auto-Encoder and Latent Energy-Based Model." In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7978–7987, 2020.

[Hop82] John J Hopfield. "Neural networks and physical systems with emergent collective computational abilities." *Proceedings of the national academy of sciences*, **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**:1527–1554, 2006.

[HOW06] Geoffrey E Hinton, Simon Osindero, Max Welling, and Yee-Whye Teh. "Unsupervised discovery of nonlinear structure using contrastive backpropagation." *Cognitive Science*, **30**(4):725–731, 2006.

[HRU17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. "Gans trained by a two time-scale update rule converge to a local nash equilibrium." In *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

[HSD19] Matthew Hoffman, Pavel Sountsov, Joshua V Dillon, Ian Langmore, Dustin Tran, and Srinivas Vasudevan. "NeuTra-lizing bad geometry in Hamiltonian Monte Carlo using neural transport." *arXiv preprint arXiv:1903.03704*, 2019.

[HW59] David H Hubel and Torsten N Wiesel. "Receptive fields of single neurones in the cat's striate cortex." *The Journal of physiology*, **148**(3):574–591, 1959.

[HXZ15] Zhi Han, Zongben Xu, and Song-Chun Zhu. "Video Primal Sketch: A Unified Middle-Level Representation for Video." *Journal of Mathematical Imaging and Vision*, **53**(2):151–170, 2015.

[HZR16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[HZR16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Identity mappings in deep residual networks." In *European conference on computer vision*, pp. 630–645. Springer, 2016.

[IMS17] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks." In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[ITS00] Naoum P Issa, Christopher Trepel, and Michael P Stryker. "Spatial frequency maps in cat visual cortex." *Journal of Neuroscience*, **20**(22):8504–8514, 2000.

[JCC20] Heewoo Jun, Rewon Child, Mark Chen, John Schulman, Aditya Ramesh, Alec Radford, and Ilya Sutskever. "Distribution Augmentation for Generative Modeling." In *Proceedings of Machine Learning and Systems 2020*, pp. 10563–10576. 2020.

[JHD16] J Yu Jason, Adam W Harley, and Konstantinos G Derpanis. "Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness." In *European Conference on Computer Vision*, pp. 3–10. Springer, 2016.

[JLT17] Long Jin, Justin Lazarow, and Zhuowen Tu. "Introspective classification with convolutional nets." In *Advances in Neural Information Processing Systems*, pp. 823–833, 2017.

[JP87] Judson P Jones and Larry A Palmer. "An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex." *Journal of neurophysiology*, **58**(6):1233–1258, 1987.

[JSZ15] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. "Spatial transformer networks." In *Advances in neural information processing systems*, pp. 2017–2025, 2015.

[JWM13] Joshua Jacobs, Christoph T Weidemann, Jonathan F Miller, Alec Solway, John F Burke, Xue-Xin Wei, Nanthia Suthana, Michael R Sperling, Ashwini D Sharan, Itzhak Fried, et al. "Direct recordings of grid-like neuronal activity in human spatial navigation." *Nature neuroscience*, **16**(9):1188, 2013.

[KAH20] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. "Training generative adversarial networks with limited data." *arXiv preprint arXiv:2006.06676*, 2020.

[KB14] Diederik P 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.

[KBE19] Manoj Kumar, Mohammad Babaeizadeh, Dumitru Erhan, Chelsea Finn, Sergey Levine, Laurent Dinh, and Durk Kingma. "VideoFlow: A flow-based generative model for video." *arXiv preprint arXiv:1903.01434*, 2019.

[KD18] Diederik P Kingma and Prafulla Dhariwal. "Glow: Generative flow with invertible 1x1 convolutions." In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

[KGV83] Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi, et al. "Optimization by simulated annealing." *science*, **220**(4598):671–680, 1983.

[KH09] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images." Technical report, Citeseer, 2009.

[KJB12] Nathaniel J Killian, Michael J Jutras, and Elizabeth A Buffalo. "A map of visual space in the primate entorhinal cortex." *Nature*, **491**(7426):761, 2012.

[KKH19] Ilyes Khemakhem, Diederik P Kingma, and Aapo Hyvärinen. "Variational Autoencoders and Nonlinear ICA: A Unifying Framework." *arXiv preprint arXiv:1907.04809*, 2019.

[KMR14] Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. "Semi-supervised learning with deep generative models." In *Advances in Neural Information Processing Systems*, pp. 3581–3589, 2014.

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

[KSJ16] Diederik P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. "Improved variational inference with inverse autoregressive flow." In *Advances in neural information processing systems*, pp. 4743–4751, 2016.

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

[KW14] Diederik Kingma and Max Welling. In *International Conference on Machine Learning*, pp. 1782–1790, 2014.

[LA16] Samuli Laine and Timo Aila. "Temporal ensembling for semi-supervised learning." *arXiv preprint arXiv:1610.02242*, 2016.

[LAC10] Rosamund F Langston, James A Ainge, Jonathan J Couey, Cathrin B Canto, Tale L Bjerknes, Menno P Witter, Edvard I Moser, and May-Britt Moser. "Development of the spatial representation system in the rat." *Science*, **328**(5985):1576–1580, 2010.

[Lan08] Paul Langevin. *On the theory of Brownian motion*. 1908.

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

[LCC19] Chieh Hubert Lin, Chia-Che Chang, Yu-Sheng Chen, Da-Cheng Juan, Wei Wei, and Hwann-Tzong Chen. "COCO-GAN: generation by parts via conditional coordinating." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4512–4521, 2019.

[LCH06] Yann LeCun, Sumit Chopra, Rata Hadsell, Mare'Aurelio Ranzato, and Fu Jie Huang. "A tutorial on energy-based learning." 2006.

[LGR09] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations." In *International Conference on Machine Learning*, pp. 609–616, 2009.

[LJT17] Justin Lazarow, Long Jin, and Zhuowen Tu. "Introspective neural networks for generative modeling." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2774–2783, 2017.

[LK02] David C Lyon and Jon H Kaas. "Evidence for a modified V3 with dorsal and ventral halves in macaque monkeys." *Neuron*, **33**(3):453–461, 2002.

[LLW15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Deep learning face attributes in the wild." In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3730–3738, 2015.

[LLW18] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Large-scale celebfaces attributes (celeba) dataset." *Retrieved August*, **15**:2018, 2018.

[LW17] Qiang Liu and Dilin Wang. "Learning Deep Energy Models: Contrastive Divergence vs. Amortized MLE." *arXiv preprint arXiv:1707.00797*, 2017.

[LYT10] Ce Liu, Jenny Yuen, and Antonio Torralba. "Sift flow: Dense correspondence across scenes and its applications." *IEEE transactions on pattern analysis and machine intelligence*, **33**(5):978–994, 2010.

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

[MBJ06] Bruce L McNaughton, Francesco P Battaglia, Ole Jensen, Edvard I Moser, and May-Britt Moser. "Path integration and the neural basis of the'cognitive map'." *Nature Reviews Neuroscience*, **7**(8):663, 2006.

[MD85] Jeffrey Moran and Robert Desimone. "Selective attention gates visual processing in the extrastriate cortex." *Science*, **229**(4715):782–784, 1985.

[MHR18] Simon Meister, Junhwa Hur, and Stefan Roth. "UnFlow: Unsupervised learning of optical flow with a bidirectional census loss." In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[MIH16] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4040–4048, 2016.

[MKK18] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. "Spectral normalization for generative adversarial networks." *arXiv preprint arXiv:1802.05957*, 2018.

[MMK15] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. "Distributional smoothing by virtual adversarial examples." *stat*, **1050**:2, 2015.

[MMK18] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. "Virtual adversarial training: a regularization method for supervised and semi-supervised learning." *IEEE transactions on pattern analysis and machine intelligence*, **41**(8):1979–1993, 2018.

[MSS16] Lars Maaløe, Casper Kaae Sønderby, Søren Kaae Sønderby, and Ole Winther. "Auxiliary deep generative models." *arXiv preprint arXiv:1602.05473*, 2016.

[Mur12] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[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 *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 1105–1112, 2011.

[ND10] C. Papachristou N. Aifanti and A. Delopoulos. "The MUG Facial Expression Database." In *Proceedings of 11th Int. Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, pp. 12–14, 2010.

[Nea01a] Radford M Neal. "Annealed Importance Sampling." *Statistics and Computing*, **11**, 2001.

[Nea01b] Radford M Neal. "Annealed importance sampling." *Statistics and computing*, **11**(2):125–139, 2001.

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

[NGS20] Erik Nijkamp, Ruiqi Gao, Pavel Sountsov, Srinivas Vasudevan, Bo Pang, Song-Chun Zhu, and Ying Nian Wu. "Learning energy-based model with flow-based backbone by neural transport mcmc." *arXiv preprint arXiv:2006.06897*, 2020.

[NHH19] Erik Nijkamp, Mitch Hill, Tian Han, Song-Chun Zhu, and Ying Nian Wu. "On the anatomy of mcmc-based maximum likelihood learning of energy-based models." *arXiv preprint arXiv:1903.12370*, 2019.

[NHZ19] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. "On Learning Non-Convergent Short-Run MCMC Toward Energy-Based Model." *arXiv preprint arXiv:1904.09770*, 2019.

[NNM96] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. "Columbia object image library (coil-20)." 1996.

[NWC11] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. "Reading digits in natural images with unsupervised feature learning." 2011.

[ODZ16] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. "Wavenet: A generative model for raw audio." *arXiv preprint arXiv:1609.03499*, 2016.

[OF96] Bruno A Olshausen and David J Field. "Emergence of simple-cell receptive field properties by learning a sparse code for natural images." *Nature*, **381**(6583):607, 1996.

[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.

[OF05] Bruno A Olshausen and David J Field. "How close are we to understanding V1?" *Neural computation*, **17**(8):1665–1699, 2005.

[OK79] John O'Keefe. "A review of the hippocampal place cells." *Progress in neurobiology*, **13**(4):419–439, 1979.

[OKE16] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. "Conditional image generation with pixelcnn decoders." In *Advances in neural information processing systems*, pp. 4790–4798, 2016.

[PC19] KANCHARLA Parimala and Sumohana and Channappayya. "Quality Aware Generative Adversarial Networks." In *Advances in Neural Information Processing Systems*, pp. 2948–2958, 2019.

[PKD16] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. "Context encoders: Feature learning by inpainting." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2536–2544, 2016.

[PLH17] Jaideep Pathak, Zhixin Lu, Brian R Hunt, Michelle Girvan, and Edward Ott. "Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data." *Chaos: An Interdisciplinary Journal of Nonlinear Science*, **27**(12):121102, 2017.

[PR81] Daniel A Pollen and Steven F Ronner. "Phase relationships between adjacent simple cells in the visual cortex." *Science*, **212**(4501):1409–1411, 1981.

[PS17] Marius Pachitariu and Maneesh Sahani. "Visual motion computation in recurrent neural networks." *bioRxiv*, p. 099101, 2017.

[PSR13] Hugh Pastoll, Lukas Solanka, Mark CW van Rossum, and Matthew F Nolan. "Feedback inhibition enables theta-nested gamma oscillations and grid firing fields." *Neuron*, **77**(1):141–154, 2013.

[Rin02] Dario L Ringach. "Spatial structure and symmetry of simple-cell receptive fields in macaque primary visual cortex." *Journal of neurophysiology*, **88**(1):455–463, 2002.

[RM15] Danilo Jimenez Rezende and Shakir Mohamed. "Variational inference with normalizing flows." *arXiv preprint arXiv:1505.05770*, 2015.

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

[RMW14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. "Stochastic backpropagation and approximate inference in deep generative models." *arXiv preprint arXiv:1401.4082*, 2014.

[ROS18] David C Rowland, Horst A Obenhaus, Emilie R Skytøen, Qiangwei Zhang, Cliff G Kentros, Edvard I Moser, and May-Britt Moser. "Functional properties of stellate cells in medial entorhinal cortex layer II." *Elife*, **7**:e36664, 2018.

[RS07] Martin Rehn and Friedrich T Sommer. "A network that uses few active neurones to code visual input predicts the diverse shapes of cortical receptive fields." *Journal of computational neuroscience*, **22**(2):135–146, 2007.

[RWP19] Thomas Ridler, Jonathan Witton, Keith G Phillips, Andrew D Randall, and Jonathan T Brown. "Impaired speed encoding is associated with reduced grid cell periodicity in a mouse model of tauopathy." *bioRxiv*, p. 595652, 2019.

[SBG17] Kimberly L Stachenfeld, Matthew M Botvinick, and Samuel J Gershman. "The hippocampus as a predictive map." *Nature neuroscience*, **20**(11):1643, 2017.

[SE19] Yang Song and Stefano Ermon. "Generative modeling by estimating gradients of the data distribution." In *Advances in Neural Information Processing Systems*, pp. 11895–11907, 2019.

[SE20] Yang Song and Stefano Ermon. "Improved techniques for training score-based generative models." *arXiv preprint arXiv:2006.09011*, 2020.

[SF11] Sameet Sreenivasan and Ila Fiete. "Grid cells generate an analog error-correcting code for singularly precise neural computation." *Nature neuroscience*, **14**(10):1330, 2011.

[SFH06] Francesca Sargolini, Marianne Fyhn, Torkel Hafting, Bruce L McNaughton, Menno P Witter, May-Britt Moser, and Edvard I Moser. "Conjunctive representation of position, direction, and velocity in entorhinal cortex." *Science*, **312**(5774):758–762, 2006.

[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 *AISTATS*, 2009.

[SH19] Saeed Saremi and Aapo Hyvarinen. "Neural empirical bayes." *Journal of Machine Learning Research*, **20**:1–23, 2019.

[SKC17] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. "Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications." *arXiv preprint arXiv:1701.05517*, 2017.

[SMG19] Ben Sorscher, Gabriel Mel, Surya Ganguli, and Samuel A Ocko. "A unified theory for the origin of grid cells through the lens of pattern formation." 2019.

[SMH15] Martin Stemmler, Alexander Mathis, and Andreas VM Herz. "Connecting multiple spatial scales to decode the population activity of grid cells." *Science Advances*, **1**(11):e1500816, 2015.

[SMS17] Masaki Saito, Eiichi Matsumoto, and Shunta Saito. "Temporal generative adversarial nets with singular value clipping." In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[SMS18] Saeed Saremi, Arash Mehrjou, Bernhard Schölkopf, and Aapo Hyvärinen. "Deep energy estimator networks." *arXiv preprint arXiv:1805.08306*, 2018.

[SSS12] Hanne Stensola, Tor Stensola, Trygve Solstad, Kristian Frøland, May-Britt Moser, and Edvard I Moser. "The entorhinal grid map is discretized." *Nature*, **492**(7427):72, 2012.

[SVI16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[SWM15] Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. "Deep unsupervised learning using nonequilibrium thermodynamics." *arXiv preprint arXiv:1503.03585*, 2015.

[SX15] Fisher Yu Yinda Zhang Shuran Song and Ari Seff Jianxiong Xiao. "Construction of a Large-scale Image Dataset using Deep Learning with Humans in the Loop." *arXiv preprint arXiv:1506.03365*, 2015.

[SZS12] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. "UCF101: A dataset of 101 human actions classes from videos in the wild." *arXiv preprint arXiv:1212.0402*, 2012.

[Tay02] Michael Taylor. "Lectures on Lie groups." *Lecture Notes, available at http://www. unc. edu/math/Faculty/met/lieg. html*, 2002.

[TBD18] Matthew Tesfaldet, Marcus A. Brubaker, and Konstantinos G. Derpanis. "Two-Stream Convolutional Networks for Dynamic Texture Synthesis." In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[tev06] "GroundTruth100-for-COIL : 100 labelled images for object recognition tests.", 2006.

[Tie08] Tijmen Tieleman. "Training restricted Boltzmann machines using approximations to the likelihood gradient." In *Proceedings of the 25th international conference on Machine learning*, pp. 1064–1071. ACM, 2008.

[TLY17] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. "Mocogan: Decomposing motion and content for video generation." *arXiv preprint arXiv:1707.04993*, 2017.

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

[TV17] Antti Tarvainen and Harri Valpola. "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results." In *Advances in neural information processing systems*, pp. 1195–1204, 2017.

[TVA19] Dustin Tran, Keyon Vafa, Kumar Krishna Agrawal, Laurent Dinh, and Ben Poole. "Discrete Flows: Invertible Generative Models of Discrete Data." *arXiv preprint arXiv:1905.10347*, 2019.

[Vin11] Pascal Vincent. "A connection between score matching and denoising autoencoders." *Neural Computation*, **23**(7):1661–1674, 2011.

[VK20] Arash Vahdat and Jan Kautz. "Nvae: A deep hierarchical variational autoencoder." *Advances in Neural Information Processing Systems*, **33**, 2020.

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

[WBS12] J. Wulff, D. J. Butler, G. B. Stanley, and M. J. Black. "Lessons and insights from creating a synthetic optical flow benchmark." In A. Fusiello et al. (Eds.), editor, *ECCV Workshop on Unsolved Problems in Optical Flow and Stereo Estimation*, Part II, LNCS 7584, pp. 168–177. Springer-Verlag, October 2012.

[WPB15] Xue-Xin Wei, Jason Prentice, and Vijay Balasubramanian. "A principle of economy predicts the functional architecture of grid cells." *Elife*, **4**:e08362, 2015.

[WS02] Laurenz Wiskott and Terrence J Sejnowski. "Slow feature analysis: Unsupervised learning of invariances." *Neural computation*, **14**(4):715–770, 2002.

[WSH18] Wei Wang, Yuan Sun, and Saman Halgamuge. "Improving mmd-gan training with repulsive loss function." *arXiv preprint arXiv:1812.09916*, 2018.

[WZG08] Ying Nian Wu, Song-Chun Zhu, and Cheng-En Guo. "From information scaling of natural images to regimes of statistical models." *Quarterly of Applied Mathematics*, **66**:81–122, 2008.

[XGZ19] Jianwen Xie, Ruiqi Gao, Zilong Zheng, Song-Chun Zhu, and Ying Nian Wu. "Learning dynamic generator model by alternating back-propagation through time." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 5498–5507, 2019.

[XGZ20] Jianwen Xie, Ruiqi Gao, Zilong Zheng, Song-Chun Zhu, and Ying Nian Wu. "Motion-Based Generator Model: Unsupervised Disentanglement of Appearance, Trackable and Intrackable Motions in Dynamic Patterns." In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 12442–12451, 2020.

[XLG16] Jianwen Xie, Yang Lu, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. "Cooperative training of descriptor and generator networks." *arXiv preprint arXiv:1609.09408*, 2016.

[XLG17] Jianwen Xie, Yang Lu, Ruiqi Gao, Song-Chun Zhu, and Ying Nian Wu. "Cooperative training of descriptor and generator networks." *arXiv preprint arXiv:1609.09408*, 2017.

[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.

[XZF19] Jianwen Xie, Zilong Zheng, Xiaolin Fang, Song-Chun Zhu, and Ying Nian Wu. "Cooperative Training of Fast Thinking Initializer and Slow Thinking Solver for Multi-Modal Conditional Learning." *arXiv preprint arXiv:1902.02812*, 2019.

[XZW17a] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. "Synthesizing Dynamic Patterns by Spatial-Temporal Generative ConvNet." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7093–7101, 2017.

[XZW17b] Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. "Synthesizing dynamic patterns by spatial-temporal generative convnet." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7093–7101, 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.

[YWU11] Michael M Yartsev, Menno P Witter, and Nachum Ulanovsky. "Grid cells without theta oscillations in the entorhinal cortex of bats." *Nature*, **479**(7371):103, 2011.

[ZB16] Yipin Zhou and Tamara L Berg. "Learning temporal transformations from time-lapse videos." In *European Conference on Computer Vision (ECCV)*, pp. 262–277, 2016.

[Zee16] Anthony Zee. *Group theory in a nutshell for physicists*. Princeton University Press, 2016.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. "Wide residual networks." *arXiv preprint arXiv:1605.07146*, 2016.

[ZKL16] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Antonio Torralba, and Aude Oliva. "Places: An image database for deep scene understanding." *arXiv preprint arXiv:1610.02055*, 2016.

[ZLX14] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. "Learning deep features for scene recognition using places database." In *Advances in Neural Information Processing Systems*, pp. 487–495, 2014.

[ZMG15] Junbo Zhao, Michael Mathieu, Ross Goroshin, and Yann Lecun. "Stacked what-where auto-encoders." *arXiv preprint arXiv:1506.02351*, 2015.

[ZYM13] Sheng-Jia Zhang, Jing Ye, Chenglin Miao, Albert Tsao, Ignas Cerniauskas, Debora Ledergerber, May-Britt Moser, and Edvard I Moser. "Optogenetic dissection of entorhinal-hippocampal functional connectivity." *Science*, **340**(6128), 2013.