

UCLA

UCLA Electronic Theses and Dissertations

Title

Real-world High-dimensional Data with Multimodal Distributions: Mode Discovery and Mode-preserving Generative Models

Permalink

<https://escholarship.org/uc/item/3j43x6hz>

Author

Lu, Qiuqing

Publication Date

2022

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Los Angeles

Real-world High-dimensional Data with Multimodal Distributions: Mode Discovery and
Mode-preserving Generative Models

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Electrical and Computer Engineering

by

Qiuqing Lu

2022

© Copyright by
Qiuqing Lu
2022

ABSTRACT OF THE DISSERTATION

Real-world High-dimensional Data with Multimodal Distributions: Mode Discovery and
Mode-preserving Generative Models

by

Qiuqing Lu

Doctor of Philosophy in Electrical and Computer Engineering

University of California, Los Angeles, 2022

Professor Vwani P. Roychowdhury, Chair

Recent advances in Artificial Intelligence (AI) have demonstrated extraordinary performances in tasks relating to our daily life activities. However, the “intelligence” of the machine is still far below our expectations, especially when it concerns the ability to model the observable world in a reliable and predictive fashion. This is a challenging task, as real world data is not only high dimensional but has distributions with multiple modes and different modalities. Identifying such hidden modes and developing succinct generative models that can generalize to unseen scenarios are difficult problems. If one can solve such versatile data modeling problems, then one could take effective actions to achieve desired goals, enabling AI to display true cognitive abilities.

In this thesis, we model real world high-dimensional data as having an inherently low dimensional generative mechanism: a set of computational laws – indexed by a hierarchical mixture of contexts– project low-dimensional data into the observable high-dimensional multi-modal data sets. While the ideal goal of reverse engineering such a universal generative mechanism –purely based on data– is a seemingly impossible task, a spectrum of techniques

are available for handling cases where partial information about the underlying contexts and modes are given. The objective of the thesis is to address the problem of discovering modes and characterizing them along this spectrum of prior information, and demonstrate the effectiveness of our techniques in several real world applications. For example, as a step towards discovering low-dimensional modes of high-dimensional data, we propose an efficient and novel algorithm called eNMF for the non-negative matrix factorization (NMF) problem, which is used widely for interpretable dimensionality reduction tasks. Then we address a problem to refine the modes in the Electroencephalogram (EEG) data where ground truth labels – as assumed in a strictly supervised approach – are not available and only partial and noisy labels can be inferred. A class of events in EEG data, referred to as High-frequency oscillations (HFOs), have been found to be a promising biomarker of the epileptogenic zone in the brains of patients with epilepsy. HFOs, however, can be generated by healthy tissues as well, and further differentiation of the HFOs into epileptogenic HFOs (eHFOs) and non-eHFOs is needed for more accurate localization of the problematic zone. Going further along the spectrum, we address the domain of creating generative models when only the high-level contexts are given and demonstrate that it facilitates the fine-level discovery of modes within the given contexts. In particular, we construct context-aware 3D human motion generation models that enable on-demand sampling of discovered and interpolated modes within different categories of action types, and customization of motion trajectories not present in the training data. Finally, we tackle the problem of creating generative models for scenarios where the observable data is created by an agent that interacts sequentially with real physical environments. We show how one can create generative models that can not only imitate diverse behavioral data, but can do so robustly while interacting with an external environment.

The dissertation of Qiuqing Lu is approved.

Arash A. Amini

Jonathan Chau-Yan Kao

Lieven Vandenberghe

Vwani P. Roychowdhury, Committee Chair

University of California, Los Angeles

2022

To my parents . . .

TABLE OF CONTENTS

1	Introduction	1
1.1	Overview	1
1.2	Data models	3
1.3	Determining and representing multiple modes in data	5
1.3.1	What are modes?	5
1.3.2	Different methods for determining modes	8
1.4	Problem Statement and Contributions	12
1.4.1	Interpretable Dimensionality Reduction: An Efficient Algorithm for Non-Negative Matrix Factorization	14
1.4.2	Mode Separation with Weak Supervision: Discovering Epileptogenic HFOs in the Electroencephalogram (EEG) of Epilepsy Patients . . .	15
1.4.3	Context Aware and Mode Preserving Generative Models (I): Action- type Conditioned 3D Motion Generation, Mode Discovery, and Cus- tomization	17
1.4.4	Context Aware and Mode Preserving Generative Models (II): Imita- tion Learning on a Physical Platform	18
1.5	Organization of the rest of the thesis	19
2	Background and Related Work	21
2.1	Latent dimension reduction	21
2.2	Weakly Supervised Learning	24
2.2.1	A toy example for weighted loss function	26

2.3	Generative models	30
2.3.1	Introduction	30
2.3.2	VAE and VAE variants	31
2.4	Imitation learning	37
2.4.1	Reinforcement learning(RL)	37
2.4.2	Behavior Cloning (BC)	38
2.4.3	Inverse Reinforcement Learning (IR)	38
3	Interpretable Dimensionality Reduction: An Efficient Algorithm for Non-	
	Negative Matrix Factorization	40
3.1	Introduction	40
3.2	Related work	41
3.3	Motivations	42
3.3.1	Complexity	42
3.3.2	Going from SVD to known Exact NMF factors	43
3.3.3	Going from SVD to unknown Exact NMF factors	44
3.3.4	General NMF: Going from SVD to the Positive Orthant	46
3.4	Exterior NMF Algorithm	47
3.4.1	Optimal rotation of unconstrained global optimum	47
3.4.2	Exterior penalty method for attaining feasibility	49
3.4.3	Descent to the local minima	51
3.5	Exterior NMC algorithm	52
3.6	Experiments and results	53
3.6.1	Datasets and algorithms	53

3.6.2	Reconstruction error and run time comparison	55
3.6.3	Structural analysis of minima	63
3.7	Concluding remarks	68
4	Mode Separation with Weak Supervision: Discovering Epileptogenic HFOs in the Electroencephalogram (EEG) of Epilepsy Patients	69
4.1	Introduction	69
4.2	Related work	70
4.3	Overview	72
4.4	Method	74
4.4.1	Dataset	74
4.4.2	Feature preprocessing	75
4.4.3	Network architecture	76
4.4.4	Label assignment for training	77
4.4.5	The Deep Learning Architecture and Training for REVerse engineering (DLATREV)	78
4.4.6	Training and performance analysis	79
4.5	Experiments and results	80
4.5.1	Relationship between epileptogenic HFOs and HFOs with spikes . . .	80
4.5.2	Spatial distribution of epileptogenic and non-epileptogenic HFOs . . .	81
4.5.3	Perturbation analysis to investigate salient features of epileptogenic and non-epileptogenic HFOs	82
4.5.4	Robust choice of the weight term	89
4.6	Discussion	90

5	Context Aware and Mode Preserving Generative Models (I): Action-type Conditioned 3D Motion Generation, Mode Discovery, and Customization .	94
5.1	Introduction	94
5.2	Related work	97
5.3	Algorithm	99
5.4	Method	101
5.4.1	Notations	101
5.4.2	Model Architecture and Loss Functions	101
5.4.3	Sampling the Latent Code Space	103
5.5	Experiment and result	104
5.5.1	Dataset	104
5.5.2	Metrics based on classifiers	105
5.5.3	Comparison to the state of the art	106
5.5.4	Ablation study	106
5.5.5	Customization	108
5.6	Concluding Remarks	112
5.7	Appendix	113
5.7.1	Overview	113
5.7.2	More results	114
5.7.3	Implementation Details	119
5.7.4	Datasets	124
5.7.5	Self organization effects in the latent space	128
5.7.6	Contrastive mechanism in the latent space	130

6	Context Aware and Mode Preserving Generative Models (II): Imitation Learning on a Physical Platform	137
6.1	Introduction	137
6.2	Related Work	138
6.3	Approach	138
6.4	Experiment and Result	139
7	Summary of contributions and future work	143
	References	146

LIST OF FIGURES

1.1	Data generation process.	5
1.2	A real-world example: 3D human motion data	9
1.3	Different approaches for discovering modes in sequential high-dimensional data, and generating samples of such discovered modes on demand.	13
2.1	Demonstration of a PCA transformation: The principal axes found by PCA are annotated with black arrows and the original data is shown as blue dots.	24
2.2	A toy example for denoising wrong labels with a robust loss function.	27
2.3	Performance of various models described in Section 2.2.1.	28
2.4	A schematic of the VAE architecture.	33
2.5	Graphical models for the Gaussian mixture variational autoencoder(GMVAE) showing the generative model (left) and the variational family (right).	35
3.1	Distance to the positive orthant is plotted for synthetic exact datasets with var- ious dimensions and sparsity levels.	45
3.2	Given enough run time, all the algorithms converge to local minima with similar reconstruction error.	56
3.3	Exact factorization.	57
3.4	Convergence speed (Synthetic datasets)	58
3.5	Equal-time constrained NMF reconstruction errors (Synthetic datasets)	60
3.6	Equal-error constrained time complexity (Real datasets)	61
3.7	Equal-time constrained NMC errors:Relative reconstruction error (RE) $\frac{\ P_{\Omega}(X-UV^T)\ _F}{\ P_{\Omega}X\ _F}$ and Relative prediction error $\frac{\ X_{test}-UV^T\ _F}{\ X_{test}\ _F}$ is plotted as a function of the number of latent dimensions (r).	62

3.8	Factor matrices are permutation of each other at the global minimum (Exact factorization scenario).	65
4.1	Upper: demonstration of our objective. Bottom: Traditional framework of HFO classification.	71
4.2	Processing workflow. Our study’s overall data processing workflow is shown as a flowchart.	73
4.3	Network input images: constructed from a one-second raw EEG segment with a detected HFO in the center.	75
4.4	CNN architecture of the model	76
4.5	Spatial distribution of epileptogenic and non-epileptogenic HFOs	81
4.6	Continued on the following page.	84
4.6	The time-frequency plot characteristics of eHFOs and non-eHFOs and model’s responses to such characteristics.	85
4.7	(Continued on the following page.)	87
4.7	The model’s responses to injecting a spike-like feature into the amplitude-coding plot	88
4.8	Find the optimal weight (w) term for the Binary Cross Entropy Loss.	89
5.1	ODMO motion generation.	95
5.2	Overview of our model architecture.	99
5.3	Mode discovery in Warm Up and Intra-Action Category Interpolations (Human-Act12 dataset)	100
5.4	Inter-Action category interpolations (HumanAct12 and UESTC datasets	108
5.5	Examples of Motion customization	109
5.8	Inter-action type interpolation for HumanAct12 Dataset	117

5.9	Inter-action type interpolation for UESTC Dataset	119
5.10	Human skeleton in the MoCap dataset (top left), HumanAct12 dataset (top right), and UESTC dataset (bottom) with the joint number annotated in red color.	124
5.11	Introducing attractive and repulsive forces among latent codes belonging to the same or different categories in the dataset.	132
5.12	Energy for two different spring systems.	133
5.6	Mode discovery in dumbbell-one-arm-shoulder-pressing and intra-action interpolation	135
5.7	End location customization on MoCap dataset	136
6.1	Example of humanoid in Humanoid-Dir-6 environment simulated via MuJoCo simulator [1].	141
6.2	Trajectories of the imitated policies for the walking task	142
6.3	Sampled frames of Humanoid walking in three different directions (generated by our trained agent).	142

LIST OF TABLES

1.1	Different Methods for identifying and describing multiple modes in data.	8
3.1	Equal-time constrained NMF reconstruction errors (Real datasets)	59
3.2	Factor matrices are generalized rotations of each other at the unconstrained global minimum (Synthetic dataset).	66
3.3	Factor matrices are permutation of each other only at local minima (Real datasets).	67
3.4	eNMF satisfies the KKT conditions for all latent dimensions and across both the datasets.	68
5.1	Benchmark on HumanAct12 & UESTC Dataset based on classifier.	106
5.2	Benchmark on CMU MoCap Dataset based on classifier	106
5.3	HumanAct12 Dataset Benchmark	110
5.4	Motion improvement	112
5.5	Benchmark for MM on three Dataset based on classifier.	114
5.6	Architecture of ODMO	121
5.7	Architecture of MoCap and HumanAct12 motion classifiers	121
5.8	Architecture of our UESTC classifier	121
5.9	MoCap Dataset	124
5.11	UESTC Dataset	124
5.10	HumanAct12 Dataset	127

ACKNOWLEDGMENTS

I would like to thank my respectable advisor Professor Vwani Roychowdhury for his mentorship and support. Deeply generous of his time and talents, he spent a lot of time discussing with me, enlightening me, and figuring out things with me. Without his wise and patient guidance, I can't go this far. I am deeply grateful for his countless help throughout my PhD time.

I would also like to express my gratitude to the members of my doctoral committee for taking time to serve on this committee and providing valuable feedbacks. Specifically, I would like to thank Professor Lieven Vandenberghe, with whom I had wonderful collaborations and was inspired a lot by his insightful suggestions. I would like to thank Professor Arash Amini for offering his sharp observations and valuable feedbacks. I would like to thank Professor Jonathan Kao for sharing his critical thinking and thorough opinions. I would like to thank Professor Kung Yao for making time and kindly being in my committee for the qualification exam.

I want to thank my labmates for discussing problems with me in the late nights and help me with their best efforts in all perspectives. Their audacity, ingenuity and optimism encourage me in the past and will continue to do so in the future. I can always learn from them and never felt alone when surrounded by them.

I would like to thank my dear friends for discussing research problems with me even if we are in different domains and exploring this rapidly changing world with excitement and curiosity together.

I would like to thank my parents and my family for their unconditional support and accompany. My life is full of laughs and loves because of them.

I want to thank LA for the bright sunshine which always shares lots of energy with me.

Lastly, I want to thank myself for walking through this journey, I am glad I have had my flight.

VITA

- 2011–2015 B.S. (Automation), Tsinghua University, China.
- 2015-2017 M.S. (Electrical Engineering), University of California, Los Angeles.
- 2017-2022 Ph.D. student (Electrical and computer engineering),University of California, Los Angeles

CHAPTER 1

Introduction

1.1 Overview

Artificial intelligence (AI) already has found wide applications in our daily life, such as the ubiquitous penetration of search engines provided by Google, recommendation systems guiding our interactions with Amazon, and increasingly, self-driving cars operated by Waymo. At the cognitive level, we expect the machine to do what we (as thinking entities) can do as stated by Alan Turing in his paper “Computing Machinery and Intelligence” [2]. A central characteristic of human intelligence is the ability to model the observable world in a reliable and predictive fashion, so that reasonable actions can be taken to achieve certain desired goals. To move towards such a human-like intelligence in machines, well-defined tasks and effective learning algorithms are essential components. One formal way to define the learning process for a machine is aptly defined in [3] as, “a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T , as measured by P , improves with experience E ”. For current main stream methods, the experience E mainly consists of data and feedback given by its associated metric space.

A concrete approach towards achieving these goals is to adopt and replicate a core assumption that has been responsible for the success of the physical sciences in the past couple of centuries: Assume that there exists underlying low-dimensional generative models that are indexed by a hierarchical set of contexts to generate high dimensional data with multiple

different modes. Context controls the content and granularity of the generation process. The goal is to jointly estimate a generative model and a set of contexts that models the data to achieve the best predictive outcomes. Depending on the exact real world scenario and how data is collected, context can be defined in countless ways. Therefore, in order to achieve such a grand challenge task, given a particular environment, a machine must be able to perform the following minimal set of tasks: (i) mode discovery (ii) mode recognition, and (iii) mode generation. Functionality wise, the machine must be able to (i) Identify sufficiently many exemplars from each mode in the environment, and (ii) Develop a succinct discriminative description of the modes from these exemplars.

Following these assumptions and observations, and instead of solving one particular problem, we applied our core idea—automatically discover modes and train mode-preserving generative models from real-world high-dimensional data with multimodal distributions – to different settings in the real world scenario and demonstrated how the corresponding mode recognition and description tasks can be addressed. We also demonstrate how such mode discovery and modeling techniques can help in improving the recognition capacities of machines.

In Section 1.2, we introduce a general computational framework for the underlying generative models that result in creating the real-world data with multiple modes. In Section 1.3, we give some more examples of modes present in real-world data and list existing methods and frameworks for identifying and characterizing modes from data. In Section 1.4, we list several real world applications with representative scenarios covering a spectrum of cases, for which we developed our techniques that address the core problem of discovering and characterizing modes. Finally, in Section 1.5, we give an outline of the contents in the following chapters.

1.2 Data models

The recent developments in artificial intelligence (AI) -spurred by unprecedented growth in computing power, storage, sensor technology, and their ubiquitous deployment– have enabled machines to understand the world better and interact with the real world at deeper levels. Data, “a fact given or granted [4],” is what the world gives or generates when one interacts with the world, and is thus one of the key ingredients in learning how the world works. We peek into the mysteries of the universe through the lens of data.

In order to be able to plan actions and strategize and control the environment, we also need to learn succinct generative models for the data that generalize to unseen situations. Data contains a huge amount of information and undistilled knowledge about how the world, and the right generative models can serve as the bridge between the real world and a potential simulated world. The human view has been that there is a small set of universal laws (which can be stated succinctly) that are responsible for generating the diverse and complex world we see. The diversity in data statistics is due to different contexts, where a context can be loosely defined as a set of parameters that govern the spatio-temporal dynamics of a real-world system. The contexts themselves are expected to be arranged in a hierarchical fashion: for example, in physics only a certain set of dynamics are permitted when a particle is confined within a given field distribution; the given potential acts as the general context, and the initial conditions, and hyper parameters, such as mass and initial velocity, act as sub-contexts. The behaviors displayed under different settings of the various contexts and sub-contexts lead to observed data with multiple modes or structured patterns at different spatio-temporal scales.

A significant outcome of such a generative view of data is that while the observed data –such as images, videos, flow of liquids and gases, and geographical features – might be very high dimensional, their generative or description dimensionality –when suitably defined– is low. For example, one could define the description dimensionality of a data set from

a modified perspective of the well-known concept of Kolmogorov complexity [5] of data sequence. Consider a computational unit, $\mathbf{x} = f(\mathbf{z})$, where $f(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^n$, maps inputs lying in a d -dimensional space to outputs in n -dimensional space, where $n \geq d$. Let a given dataset, $\mathcal{D}$, be described by a distribution $\mathcal{P}_{\mathbf{x}}$, where $\mathbf{x}$ is a random variable in $\mathbb{R}^n$. Let us also suppose that there exists a random variable, $\mathbf{z} \in \mathbb{R}^d$, with distribution $\mathcal{P}_{\mathbf{z}}$, such that $P_{f(\mathbf{z})} \sim \mathcal{P}_{\mathbf{x}}$, where $\sim$ stands for the equality relationship of two distributions under some well-defined metric. We can then say that the computational unit $f(\cdot)$ along with the distribution $\mathcal{P}_{\mathbf{z}}$ is a generative model of $\mathcal{D}$, and d is its corresponding latent or description dimension. The minimum latent dimension of $\mathcal{D}$, $d_{\mathcal{D}}$, is the minimum d for which such a generative model exists. Usually, $d < n$ and hence the support set of the distribution $\mathcal{P}_{\mathbf{x}}$, i.e., the set of points where it is non-zero, can only have measure zero in $\mathbb{R}^n$, and the data $\mathcal{D}$ can only lie on a d -dimensional manifold in $\mathbb{R}^n$.

Physics based models aim to define only a few such computational units, formulated mostly as Partial Differential Equations (PDE's), and a set of universal constants and a few additional parameters, such that a hierarchical generative model can be constructed to explain or emulate all measured data under all possible conditions. Building such an all-inclusive and constructive generative model, however, is an undecidable problem, and in recent literature a data-driven black-box approach is adopted. Such an approach adopts a parameterized deep neural network, $f_{\theta}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^n$, and in general, a parameterized distribution function $\mathcal{P}_{\mathbf{z}}(\delta)$, and aims to learn the parameters (θ, δ) such that $D(\mathcal{P}_{\mathbf{x}}, \mathcal{P}_{f_{\theta}(\mathbf{z}, (\delta))})$ is minimized, where $D(p, q)$ is a distance metric between any two given distributions p and q .

In this thesis, we adopt this latter data-driven approach, and simplify our modeling approach into a standard pipeline with black-box and assume that exist underlying generative models that can be indexed by context to generate high-dimensional data with multiple modes. Depending on the real world scenarios and observation perspectives, context can be defined in countless ways controlling the content and granularity of the generation process.

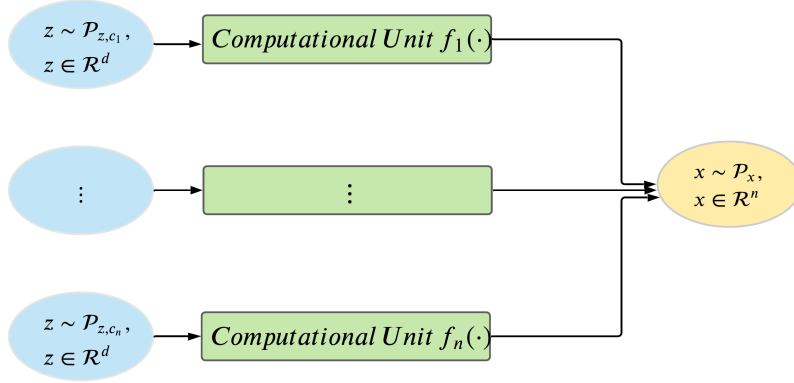


Figure 1.1: Data generation process.

The generation process might contain pure randomness which can be included as input to the generation mode; or be deterministic, which can be totally predicted once the generator is available. Both mechanisms can be captured by a black-box generator model. Specifically, we denote the context as $c \in \{c_1, \dots, c_K\}$, we assume that under each given context c , there would be a different mixture model containing multiple components. For example, data $x \sim p(c, \theta) = \sum_{i=1}^K \phi_i(c) F_i(\beta)$, where $\sum_{i=1}^K \phi_i(c) = 1$ and $\phi_i(c) = 1$ for $c = c_i$. $F_i(\beta)$ is a distribution modeled by parameter β . $F_i(\beta) = \sum_{j=1}^M \psi_j M(\alpha_j)$, where $\sum_{j=1}^M \psi_j = 1$ and $M(\alpha_j)$ capturing the distribution of the mode j . Most of the time, x has higher dimension than the parameter space. It is following our intuition that the world is governed by simplified laws but produce diverse and ample formats. On the other hand, the essence of science is via introducing the fundamental laws of the universe to describe and predict it.

1.3 Determining and representing multiple modes in data

1.3.1 What are modes?

As shown in Figure 1.1, data with multiple modes can be generated according to the context. The concept of modes itself varies depending on the context: it can be the diverse shapes of

sunflower or the diverse types of flowers. Once we specify the context that we are going to inspect on, data with multiple modes under this context can be collected accordingly. For example, consider a 3D human motion dataset called HumanAct12[6, 7] which has labeled example motion data from 12 different action categories. 3D human motion data represented by skeleton sequences is collected from the real world, as different subjects perform different types of activities. For each motion sequence, it can be viewed as a $N \times J \times 3$ array with N frames J joints in the skeleton, (x, y, z) coordinates for each joint. An example skeleton with 24 joints is shown in Figure 1.2. In this set up, the known context c is a specific activity type, for example, the category Warm UP. Different subjects, however, follow different warming up routines; moreover, different sets of joints are warmed up by following different sequences of motion. Thus, the set of motion sequences gathered from multiple subjects and labeled as Warm UP, comprises a distribution with multiple modes, where each mode corresponds to particular routines followed for stretching different sets of joints and muscles.

In a generative model introduced in Chapter 5, we used the activity type labels to construct a generative model with a 20-dimensional latent space, and a multi-modal prior distribution $\mathcal{P}_X$ for the HumanAct12 dataset. If one considers the latent codes corresponding to all the Warm UP motion sequences, then it is again a multi-modal distribution, comprising of several modes. We used a Gaussian Mixture Model (GMM) with K Gaussians to model the latent codes, and determined an optimal K using the Silhouette score [8]. Our analysis yielded 7 such modes, and the time-lapsed visualizations of sample motion sequences in different modes are shown in Figure 1.1. Clearly, the majority of the motion sequences in each mode corresponds to a particular type of warming up routine. Incidentally, and fortunately, there are also sub-labels available for the Warm Up action type in the HumanAct12 dataset, and we can evaluate our modes with respect to the labeled ground truth data. It turns out that the number of labeled sub-categories is also 7, matching our unsupervised discovery. There is, however, a couple of differences between the human labels and machine discovered modes: (i) In human labels, the categories of Wrist and Ankle warm up routines are

grouped together, whereas in our automated discovery they are recognized as two separate modes; and (ii) the warm up routines corresponding to Bow-Right and Bow-Left are grouped together by our automated mode estimation algorithm, but are grouped into two separate modes in human labeling.

Humans constantly learn multiple modes present in real world data and are able to recreate representations of these various modes in the form of textual descriptions and audio-visual renditions. Furthermore, humans have invented succinct and low-dimensional tools like physical laws to mimic and predict the diversity of the world we experience. In this vein while the ultimate goal of Artificial Intelligence (AI) could be difficult to define – for example, can it be conscious, – the functional goals of AI are much more concrete. One of them is to be able to model the observable world in a reliably predictive fashion, so that reasonable actions can be taken to achieve a certain goal. In order to achieve such a grand challenge task, given a particular environment, a machine must be able to perform the following minimal set of tasks: (i) mode discovery (ii) mode recognition, and (iii) mode generation.

A related challenge is that real world data is not only characterized by distributions with multiple modes, but also by the different modalities or types of data. For example, sea waves not only have a distribution in physical space that defines its shape, and a temporal evolution of this shape, but it also has characteristic sound, color, temperature and so on. Often, data from any of these modalities is sufficient to detect the presence of the object with high confidence, and can predict how the data would look like for the other modalities if they were to be collected. For example, a loud and roaring sea, heard from a distance, immediately generates a visual mental picture of the shape and height of the waves that one might witness when one is within the line of sight. With the rapid developments in sensor technology and storage device, more and more modalities of data such as images, video, 3D maps, and text can be collected and stored in ever larger scales –corresponding to the same object or concepts or situations– and how they interact with each other. Although we always

observe such data with both multi-modality (as in images, videos, audio, and other sensory formats) and distributions with multiple modes, how we discover the different modes from a given set of data is far beyond the obvious.

1.3.2 Different methods for determining modes

In Table 1.1 we provide schematics of different mainstream methods used for determining and describing modes in datasets.

Scenario Conditions	Supervised	Weak supervised	Unsupervised	
	Full	Partial/Noisy	None	
Strategy	Data Annotation+ Supervised classification	Weakly-supervised Classification	Representation Learning + Clustering	
Key Technique	Human annotation	Soft-labeling	Matrix Factorization	VAE

Table 1.1: Different Methods for identifying and describing multiple modes in data.

Understanding and characterizing multiple modes conceptually includes two minimal parts: (i) Identify sufficiently many exemplars from each mode, and (ii) Develop a succinct discriminative description of the modes from these exemplars, e.g. via a classifier, such that each mode has sufficient generalization capability and one can compute the likelihood that a given data point belongs to any particular mode. Many efforts have been put in the machine learning community towards this goal. In the current supervised learning framework, human labeling is commonly adopted for annotating the data or providing exemplars for the different modes in the data. Then the discriminative descriptions of the modes are constructed by training a classifier which learns the decision boundaries around the data points belonging

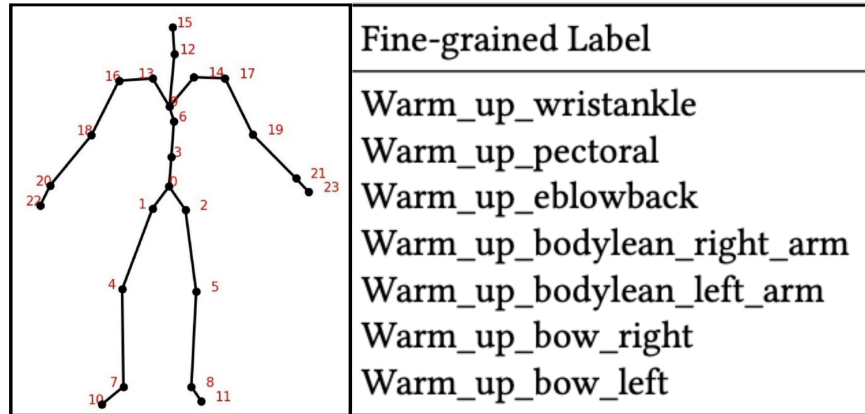
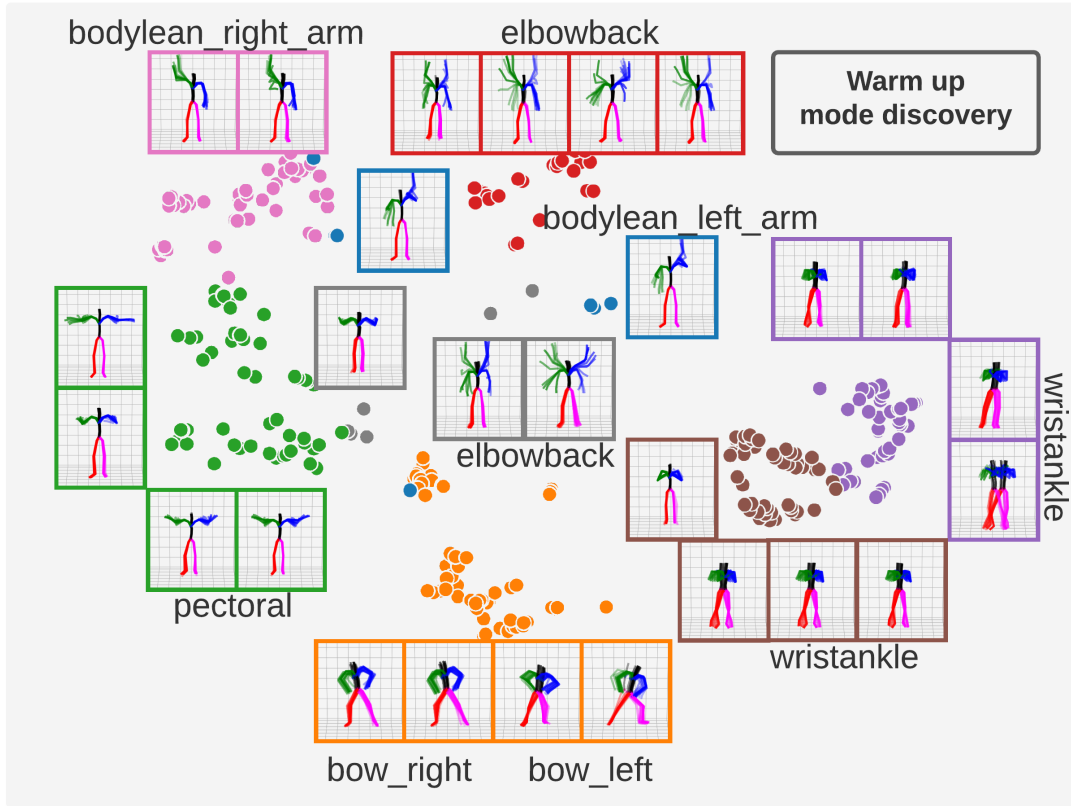


Figure 1.2: Upper: t-SNE projection of encoded human motion data for Warm up activity in HumanAct12 Dataset. Bottom: Skeleton in HumanAct 12 Dataset(Left). Manually annotated fine-grained labels inside Warm Up Activity in HumanAct 12 Dataset(Right).

to the different modes. There are several classes of classifiers in the ML literature, each with its own advantages and disadvantages.

For supervised learning, all the training data points are manually labeled with the correct modes, and models are trained to discriminate the modes. The learnt classifiers are expected to recognize the mode for new incoming data. In other words, the classifiers are designed to learn the mode by setting up categorical boundaries in the feature space. However, to be able to generalize well in complex recognition tasks, like classifying 1000 categories of everyday objects, large amount of labeled exemplars are needed and the pipeline of solving such problems is relatively mature in the deep learning community. Thus this type of tasks is not our main concern for now. Moreover, due to the strict set up of the classification task in a supervised learning framework—for example, the requirement of the correct and exact label for each data point—it can only solve a limited number of real world problems. In most real world scenarios (also, as indicated by our simplified world model), data are not well separated and might come from overlapping regions of different modes, so the exact labels for data points are not available. In such situations, one might have weakly labeled data sets, which cover a spectrum of scenarios ranging from probabilistic or noisy labels, to only partially labeled data sets. Learning modes while denoising the labels fall under the category of a task broadly referred to as weak supervision. In Chapter 2 the case of weak supervision is discussed in more detail.

Data-driven methods are more promising when large labeled datasets are unavailable, as they attempt to systematically define modes according to patterns in the distribution of observed data. Typically modes can be discovered by clustering algorithms in an unsupervised way: breaking up the unlabeled data into component distributions, such that each such component distribution corresponds to a mode. Then, the data samples from each such mode can be used as exemplars and a discriminative classifier can be designed for all the modes. In reality, collecting sufficient amount of data from each mode is not guaranteed, which compounds the difficulty in identifying each mode purely from data without bias. Fur-

thermore, high-dimensional data such as images, document and 3d human motion sequences contain sparse and structured information, which are hard to cluster directly. To address this problem in unsupervised learning, researchers have developed several efficient dimensionality reduction algorithms to reduce the dimension of the raw data. This is based on the hypothesis that the original high-dimensional data in $\mathcal{R}^n$ can often be modeled as lying on or near a low-dimensional, nonlinear manifold [9]. Thus, once the underlying low dimensional manifolds can be identified, one can find distributions that have support sets only withing the manifolds, and reduce potential problems introduced by the curse of dimensionality[].

As already introduced in Section 1.2 there is an alternative generative modeling approach for data analysis. Instead of projecting data into a low dimensional space with the least loss of information, one searches for low dimensional distributions which can be projected into a high-dimensional space -using a mapping function– that can accurately approximate the given high-dimensional data distribution. The framework of variational autoencoder (VAE) discussed in Chapter 2 is one such approach. It has an encoder-decoder architecture that is trained jointly: The encoder projects data into a low-dimensional distribution, and the decoder then projects samples from the low-dimensional distribution back to higher dimension to approximate the original data point. Once the decoder mapping function is trained, the data distribution can be reconstructed by sampling the latent space and then using the decoder to get a generated data point in the original high-dimensional space. There are several generalizations of the basic VAE model. For example, instead of latent code distributions that are dense in the latent space, e.g. Gaussian or mixtures of Gaussians, researchers have introduced models where the latent codes themselves occupy nonlinear manifolds in the low-dimensional latent space [10].

In Figure 1.3, we list the common scenarios where a model is trained to generate data with the same distribution as the observed data, while identifying the underlying modes in the data. As introduced previously, a VAE like framework is able to generate samples of observed training data with multiple modes. Once the latent codes are determined, these

can serve as low-dimensional representations of the original data points, and can be clustered into different modes. In any real world application of generative models – especially when training data is obtained from spatio-temporal real-world dynamics– being realistic is an essential requirement. Usually, these are applications where the generative model simulates or emulates behavior displayed in the training data, without following the actual physical laws. A physics based simulation is usually computationally much more expensive in such scenarios. Physics based constraints on the generative model, however, is needed so that the model generates data that is close to being physically realistic. For example, in generated motion sequences imitating walking or climbing, it is common to observe foot sliding (instead of definite steps) and levitating artifacts. In our action-type conditioned motion generation framework, we added a causal mechanism in the generation process to increase the fidelity of the generated data. The observed data in such example applications, only lists the spatial locations of the different joints of a human body as a function of time. There is no physical environment given for the model to interact with; neither do we have a physical model of how the joints are being moved. We, thus, extended our work to include scenarios, where the generative model actually interacts with a real-world environment. Here, the model has access to a controllable robot’s current state and the actions taken that result in the next state of the machine, after interaction with the environment. We introduced the framework of Self Organized Generative (SOG) model that has the potential to imitate expert behavior and generate robust multi-modal action-conditioned sequential motion while interacting with a realistic environment.

1.4 Problem Statement and Contributions

The preceding chapter lays out a spectrum of scenarios for mode discovery and computational descriptions of such underlying modes. Each such scenario, as outlined in Table 1.1 and Figure 1.3, makes specific assumptions about partial information available about the

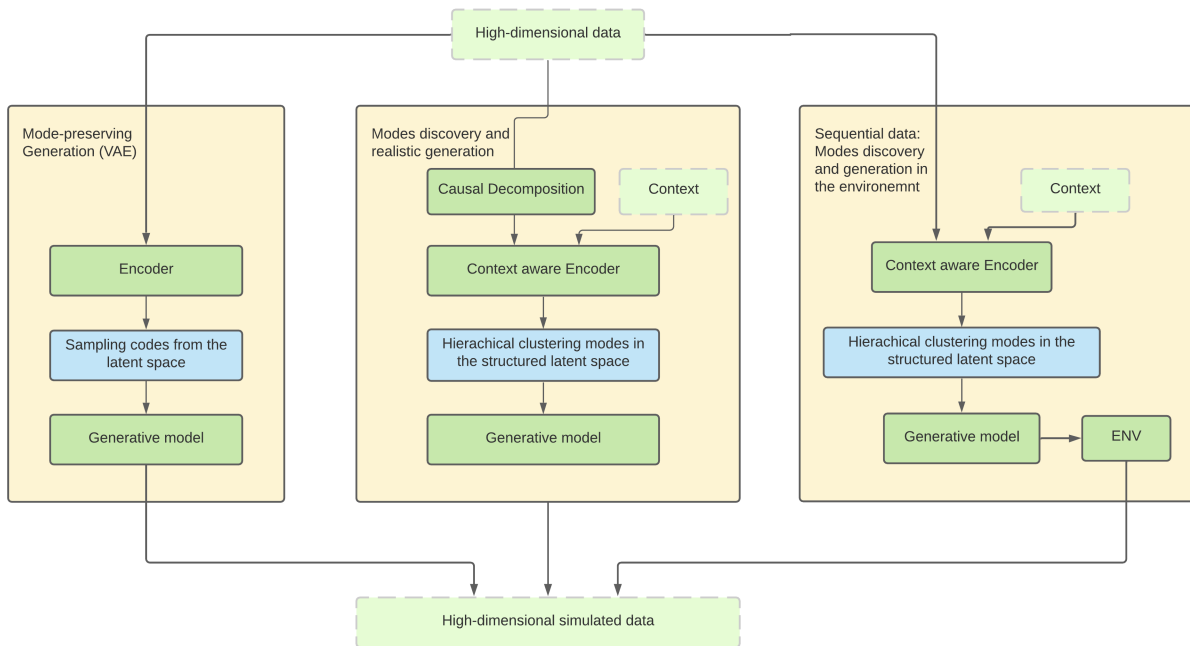


Figure 1.3: Different approaches for discovering modes in sequential high-dimensional data, and generating samples of such discovered modes on demand.

contexts and modes in the dataset. The objective of this thesis is to address the problem of discovering modes and characterizing them along this spectrum of prior information, and demonstrate the effectiveness of our techniques in several real world applications. In the following subsections, we enumerate the application areas and topics addressed, including the principles of dimensionality reduction, weak supervision, context-aware generative models with mode discovery but without a physical environment, and finally, generative models for scenarios with active interactions with a physical system involving an active environment.

1.4.1 Interpretable Dimensionality Reduction: An Efficient Algorithm for Non-Negative Matrix Factorization

Extracting low dimensional representations of a data matrix is one of the fundamental steps for the following data analysis task. Standard methods for finding a low dimensional representation include Principal Component Analysis (PCA) [Jolliffe, 2011], Independent Component Analysis (ICA) [Boscolo et al., 2004], and singular value decomposition (SVD). In many applications, however, the data has special structure, and the factors are more meaningful and interpretable if they satisfy certain constraints; for example, in Network Component Analysis (NCA) [Liao et al., 2003], [Boscolo et al., 2005] the zero entries of one of the factors are predetermined based on a priori information. Non-negative matrix factorization (NMF) is the problem of finding good approximations of a given non-negative matrix as a low-rank product of two non-negative matrices. This linear dimensionality reduction technique has been used very successfully for a large variety of machine learning and data mining tasks, including hyperspectral imaging [Gillis, 2017], topic modeling [Arora et al., 2012b], audio denoising [Schmidt et al., 2007], and video structuring [Berry et al., 2007], communication complexity ([Lee et al., 2009]), probability ([Carlini and Rapallo, 2010]), and computational geometry ([Gillis and Glineur, 2012b]).

Since NMF has been shown to be useful in a number of applications, various NMF algorithms have been developed by researchers from different communities. However, all these

algorithms are interior methods in the sense that they either start with a feasible point (i.e., all the entries of the initial factor matrices are non-negative) or obtain the feasibility of the factors after each iteration. In this thesis, we propose an exterior method that constructs the nonnegative factor matrices using optimal matrix factors of the unconstrained matrix factorization problem (e.g., obtained from SVD) which are not necessarily nonnegative. The proposed algorithm matches or performs better than existing state-of-the-art (SOTA) algorithms in terms of reconstruction error and time complexity, when tested on multiple real data sets. The boost in performance is significant for synthetic datasets of various signal to noise ratio (SNR) levels with the performance gap widening with lower SNR levels. The speed up and low reconstruction errors afforded by our algorithm enabled extensive numerical explorations of the structure and uniqueness of NMF solutions, as well.

1.4.2 Mode Separation with Weak Supervision: Discovering Epileptogenic HFOs in the Electroencephalogram (EEG) of Epilepsy Patients

We next address a problem to refine the modes in the Electroencephalogram (EEG) data where ground truth labels – as assumed in a strictly supervised approach – are not available and only partial and noisy labels can be inferred. Epilepsy is a common disease around the world. More than one-third of individuals with epilepsy are medication-resistant, making them potential surgical candidate. Currently, surgery is primarily guided by neuroimaging and neurophysiology (such as, interictal spikes and seizure onset zone (SOZ), where interictal refers to the segment of EEG signal between seizures). However, the seizure-freedom rate of surgery is suboptimal, ranging from 50 to 85%. Identifying a biomarker that can accurately delineate the spatial extent of the epileptogenic zone (EZ: brain areas responsible for generating seizures) will be groundbreaking. The intracranially-recorded interictal high-frequency oscillations (HFOs) on electroencephalogram (EEG) is one such promising biomarkers. However, to effectively utilize HFOs to guide epilepsy surgery, one needs to differentiate HFOs that are generated from the EZ (epileptogenic HFOs: eHFOs) and HFOs that are generated

from other areas (non-epileptogenic HFOs: non-eHFOs). Although highly desirable, there has been no systematical way to differentiate eHFOs and non-eHFOs yet. Currently the most common practice is quite time consuming and subjective as the doctors inspect each detected HFO visually and make decisions based on personal experience.

We build a data driven model to discover the modes within detected HFOs, i.e. epileptogenic vs non-epileptogenic HFOs via training a binary classifier, but with weak labels inferred from surgical outcomes. In particular, considering that no ground truth annotation for eHFOs is available, we use the same channel-wise golden information previously utilized by the doctors to make manual decisions. For example, in patients where a surgery has been successful, one can make the following assumptions: (i) Most HFOs detected in the SOZ channels, and resected during surgery, are eHFOs, (ii) The rest of HFOs in the resected channels most likely have a significant number of eHFOs; label assignments to HFOs in these channels are likely to be most noisy ones; and (iii) Almost all the HFOs detected in the preserved channels are non-eHFOs. Clearly, any labels assigned following this partial information is noisy. However, with a proper selection of weights and assuming that the eHFOs and non-eHFOs indeed have different morphological features, one can use the framework of weak supervision to identify the different classes. As shown in our work, one can then verify that the discovered eHFOs do indeed have many of the properties that have been manually observed to be true by physicians, and that in some of the interpretable features (such as the time frequency plots) the eHFOs and non-eHFOs have distinctly different distributions. Moreover, the eHFOs do not capture the characteristics of resected vs preserved channels; significant number of non-eHFOs are also present in the resected regions, when certain regions are resected because of surgical considerations and not because they are suspected to be epileptogenic in nature.

1.4.3 Context Aware and Mode Preserving Generative Models (I): Action-type Conditioned 3D Motion Generation, Mode Discovery, and Customization

Human motion generation has wide applications in machine vision, robotics, video games and computer animation. While considerable progress has been made in pixel-based video generation methods [11, 12, 13], the resulting human appearance is still unsatisfactory [6]. Using 3D human skeleton sequences to guide such video generation is considered to be a fundamental step to reduce distortions in human appearance. In recent years, forecasting missing skeleton motion sub-sequences or performing motion completion tasks have been attempted. However, initial or intermediate motion sub-sequences are still required as inputs, which limit these models’ applications.

We address the fundamental problem of 3D motion generation without any initial motion as inputs. Given only the activity type, chosen from among a pool of different activities, and its characteristic features, the generative model should generate motion sequences that are physically realistic and diverse in style, reflecting variations across the human population.

Our work has the following distinctive features: **1) Internal causal motion representation:** Any 3D sample motion is decomposed into two parts: spatial joint movements relative to a body center (root) – referred to as local movement profile (LMP) and the resulting trajectory of the body center. This key step is rooted in the kinematics of body movement: We move our body parts relative to each other (as captured in the LMP), and these movements after interaction with the environment cause our body center to have spatial movement resulting in a particular trajectory. We, thus, first encode the primary causal factor (the LMP) into a low-dimensional latent code and then train a decoder (that learns the causal factor of the environment) to reconstruct the resulting trajectory and motion of body parts. **2) Multi-Modal Latent Code Space and Novel sampling Strategies for Customization:** Additional regularization terms are added to the VAE-type encoder loss function to enable contrastive learning and separate latent codes belonging to different

action types (and styles within an action type) into distinct clusters. This preserves the different modes and styles present in real motion, enabling novel sampling strategies to support motion customization.

1.4.4 Context Aware and Mode Preserving Generative Models (II): Imitation Learning on a Physical Platform

In the 3D human motion generation scenario, the fidelity of the generated motion sequences is measured by metrics given by the classifiers trained on ground truth demonstration data. Although the fidelity metrics can capture some of the relevant quantitative measures for physical realism, there is no way to check whether the whole motion sequence follows a set of physical laws in the real world. A more advanced description/understanding capability would be: the ability to interact with the real environment and generate reasonable outputs when following a particular mode. An agent in a reinforcement learning (RL) set up can be considered to be such a generative model.

We narrow ourselves to the domain of imitation learning, where a number of experts provide exemplar trajectories, where each trajectory is a sequence of (state, action) pairs. In each trajectory, a sequence of tuples are given - (state of a physical system at t , action taken at time t), - where the state of the system at time $(t + 1)$ results from the interaction with the environment, given the action and the state at time t . An expert usually picks a mode, and then follows a policy, i.e., a function that maps the current state and the chosen mode into an action, which in turn results into a new state in the next time instant. Given such expert trajectories where experts follow different sets of strategies or modes, one wants to design a policy function that generates the action to be taken for any given state and an inferred mode, such that the generated trajectories are close to the expert trajectories. In an inverse RL (IRL) approach the algorithms attempt to estimate jointly the reward and the policy functions, often complicating the convergence process and yielding poor results.

Imitation learning is a particular approach, and is a more general task, as it learns the

policy function without the requirement of any well defined reward functions. Concretely, Imitation learning is the task of replicating expert policy from demonstrations, without access to a reward function. This task becomes particularly challenging when the expert exhibits a mixture of behaviors. Prior work has introduced latent variables to model the variations of the expert policy. However, our experiments show that the existing works do not exhibit appropriate imitation of individual modes. To tackle this problem, we adopt an encoder-free generative model for behavior cloning (BC) to accurately distinguish and imitate different modes. Then, we integrate it with a Generative Adversarial Imitation Learning (GAIL) framework to make the learning robust towards compounding errors at unseen states. We show that our method significantly outperforms the state of the art across multiple experiments.

1.5 Organization of the rest of the thesis

The rest of the thesis is organized as follows:

- In chapter 2, we introduce some of the background material and related work for a variety of topics, such as latent dimension reduction, learning with noisy label and typical generative frameworks. These concepts will allow the reader to better follow the technical material in the rest of the chapters.
- In chapter 3, we present an efficient dimension reduction algorithm to project high-dimensional data into low dimension. We show that it works with high efficiency and low reconstruction error for various types of data.
- In chapter 4, we propose a weakly supervised method which uses clinical clues to refine noisy clustering in discovered HFOs.
- In chapter 5, we introduce a generative framework for action-type conditioned 3d human motion generation, which produces realistic and diverse motion sequences, and

provides abilities such as interpolation between different action types and trajectory customization.

- In chapter 6, we introduce a generative model in an Reinforcement learning set up for imitating sequential data while discovering modes automatically.
- Finally, in Chapter 7 we summarize our contributions and discuss several avenues to extend the work reported in this thesis.

CHAPTER 2

Background and Related Work

2.1 Latent dimension reduction

With the development of better sensors and storage devices, more and more high-dimensional datasets are being aggregated spanning multiple domains, such as biological genome sequences, video surveillance, and document repositories for search engines. In many cases, the dimension of variables can be even higher than the number of observations. As already reviewed in Chapter 1, for most cases the data points reside on a much lower dimensional manifold, and have a generative process that maps low dimensional causal factors into a much higher dimensional space. Thus, it is desirable to first reduce the dimension of any given dataset, and then model the reduced dimensional data to perform predictive tasks. There are two mainstream methods to reduce input dimensions [14]: 1) feature selection, where only informative variables are selected, and 2) dimensionality reduction, where the redundancy in original variables is eliminated to generate fewer new variables from the input data.

Here we mainly focus on the dimensionality reduction framework using linear transformations. Given a set of N M -dimensional data points, $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, where $\mathbf{x}_i \in \mathbb{R}^M$, it determines a matrix $W = [\mathbf{w}_1 \mathbf{w}_2 \dots \mathbf{w}_m] \in \mathbb{R}^{M \times m}$, where $m < M$ and $\mathbf{w}_i \in \mathbb{R}^M$, such that $\mathbf{y}_i = W^T \mathbf{x}_i$, $\mathbf{y}_i \in \mathbb{R}^m$ is a lower dimensional representation of every data point, $\mathbf{x}_i$. Different criteria on the projected data points are imposed to obtain the linear transformation matrix, W . Principal Component Analysis (PCA) is one of the most popular algorithms in this

genre, and one of the constraints it imposes is that of orthogonality: $W^T W = I_{m \times m}$. There are numerous equivalent ways of describing the PCA optimization framework, and one of them is to consider $\mathbf{w}_i, i = 1, \dots, m$ as defining an m -dimensional subspace in $\mathbb{R}^M$. Then the projection of any $\mathbf{x}_i$ onto this subspace is given as $\hat{\mathbf{x}}_i = \sum_{j=1}^m (\langle \mathbf{w}_j, \mathbf{x}_i \rangle \mathbf{w}_j)$, where the inner product is defined as $\langle \mathbf{w}_j, \mathbf{x}_i \rangle = \mathbf{x}_i^T \mathbf{w}_j$. Then, the optimal projection matrix W^* is determined such that the reconstruction error is minimized: $W^* = \arg \max_{(W^T W = I_{m \times m})} \sum_{i=1}^N \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2$.

An equivalent way to formulate the PCA problem that leads to a computational method for solving for W , is to consider $\mathbf{X}$ as a zero-mean random variable, i.e., $\mathbb{E}[\mathbf{X}] = 0$, and the N data points are samples drawn from the underlying distribution. Then, the projected random variable, $\mathbf{Y} = W^T \mathbf{X}$, also has zero mean, $\mathbb{E}[\mathbf{Y}] = 0$. Next, W is computed by maximizing the sum of the variances of the m components of $\mathbf{Y}$:

$$W^* = \arg \max_{(W^T W = I_{m \times m})} \text{Tr}(\mathbb{E}[\mathbf{Y}\mathbf{Y}^T]) = \arg \max_{(W^T W = I_{m \times m})} \text{Tr}(W^T R_{\mathbf{X}} W),$$

where $R_{\mathbf{X}} = \mathbb{E}[\mathbf{X}\mathbf{X}^T] \in \mathbb{R}^{M \times M}$, is the covariance matrix of $\mathbf{X}$. The solution to this version of the optimization problem is well-known [15]: $\mathbf{w}_i$ is an eigen-vector corresponding to the i^{th} largest eigen value of $R_{\mathbf{X}}$. Thus, PCA is equivalent to computing an eigen-decomposition of the covariance matrix $R_{\mathbf{X}}$, which can be empirically estimated as $\frac{1}{N} \sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^T = \frac{1}{N} \mathbf{X}\mathbf{X}^T$, where $\mathbf{X} = [\mathbf{x}_1 \mathbf{x}_2 \dots \mathbf{x}_N] \in \mathbb{R}^{M \times N}$ is the data matrix.

Variants of PCA have also been proposed: incremental PCA (I-PCA) [16] for handling streamed data; online algorithms for PCA, where at any time t the PCA projection vectors are computed for the data observed until time t [17, 18]; Probabilistic PCA (P-PCA) [19] based on probability model handling missing values in the data; and, Sparse PCA (S-PCA) [20] for sparse interpretable results.

A generalization of PCA involves using Truncated Singular Value Decomposition (T-SVD). It subsumes PCA and is particularly useful for data where both the columns, $\mathbf{x}_i$'s $\in \mathbb{R}^M$, and the rows, $\mathbf{z}_i$'s $\in \mathbb{R}^N$, of the data matrix, $\mathbf{X} = [\mathbf{x}_1 \mathbf{x}_2 \dots \mathbf{x}_N] = [\mathbf{z}_1 \mathbf{z}_2 \dots \mathbf{z}_M]^T \in \mathbb{R}^{M \times N}$

can be considered as two correlated sets of data points. For example, given a corpus of text documents, a Term-Document matrix is a representation where each column corresponds to a "Term" or a "Word" and the term's distribution over documents, and each row corresponds to a document and the distribution of terms in it. The SVD of a matrix can be represented : $\mathbf{X}_{M \times N} = \mathbf{U}_{M \times M} \mathbf{\Sigma}_{M \times N} \mathbf{V}_{N \times N}^T$, where $\mathbf{U}^T \mathbf{U} = \mathbf{I}_{M \times M}$, $\mathbf{V}^T \mathbf{V} = \mathbf{I}_{N \times N}$, and $\mathbf{\Sigma} \geq 0$ is in general a rectangular, $M \times N$, non-negative diagonal matrix, i.e., $\Sigma(i, j) = \sigma_i \delta_{(i, j)}$ and $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\min(M, N)} \geq 0$. It follows that $\mathbf{X} \mathbf{X}^T = \mathbf{U} \mathbf{\Sigma}^2 \mathbf{U}^T$, and $\mathbf{X}^T \mathbf{X} = \mathbf{V} \mathbf{\Sigma}^2 \mathbf{V}^T$. Thus $\mathbf{U}$ are the eigen-vectors of the covariance matrix of the columns of $\mathbf{X}$, and can be used for PCA and projections to low-dimensional representations. Similarly, $\mathbf{V}$ are the eigen-vectors of the covariance matrix of the rows of $\mathbf{X}$ and can be used for PCA of the data represented by the rows. The columns of $\mathbf{U}$ and $\mathbf{V}$ represent orthogonal directions of maximum variances in the respective data sets.

A different perspective, however, emerges when one considers the T-SVD as an approximate matrix factorization technique. Let $\widehat{\mathbf{X}}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T$, where all the matrices have been truncated to the first k columns. Then $\widehat{\mathbf{X}}_k \approx \mathbf{X}$, especially when k is large. In general, one can define an optimal matrix factorization problem for any given rank $k \leq \min(M, N)$: Let $\widehat{\mathbf{X}}_k = \mathbf{A} \mathbf{B}^T$, then determine $\mathbf{A}, \mathbf{B}$ such that $\|\mathbf{X} - \widehat{\mathbf{X}}_k\|_F$ is minimized. It turns out that the T-SVD is an optimal solution to this matrix factorization problem. Often, one wants to impose certain constraints on $\mathbf{A}, \mathbf{B}$, especially when the data matrix $\mathbf{X}$ has a certain structure to it. For example, the Term-Document matrix, introduced above, has only non-negative entries. One particular case considered in Chapter 3 is that of the Non-Negative Matrix Factorization (NMF) problem [21], where $\mathbf{X} \geq 0$, and $\mathbf{A}, \mathbf{B}$ are also constrained to be non-negative. Such NMF solutions provide interpretable factors, which can be used as features for the following ML tasks.

A widely used method that adopts a nonlinear transformation approach is referred to as the t-Stochastic Neighborhood Embedding (t-SNE)[22] algorithm. The goal of t-SNE is to project data points such that the local information for each data point is preserved in the

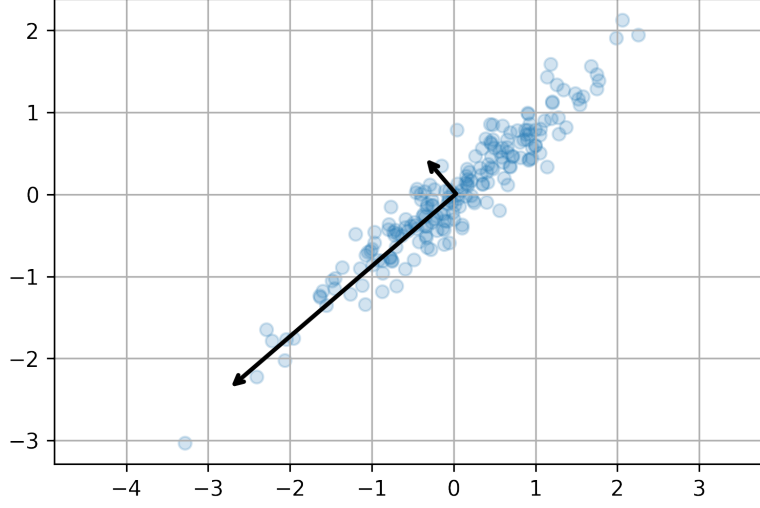


Figure 2.1: Demonstration of a PCA transformation: The principal axes found by PCA are annotated with black arrows and the original data is shown as blue dots.

projected spaces. The approach translates the pair-wise Euclidean distances between data points into conditional probabilities: $p_{ji} = \frac{p_{i|j} + p_{j|i}}{2n}$ where $p_{j|i} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)}$ (for high-dimensional original data with n data points) and $q_{ji} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l} (1 + \|y_k - y_l\|^2)^{-1}}$ (for map points in low dimension). By minimizing the cost function Equation (2.1), low-dimensional data representations y_i (for $i = 1 \dots n$) can be found to minimize the mismatch between the two methods of computing similarities.

$$C = \sum_i KL(P_i || Q_i) = \sum_i \sum_j p_{ji} \log \frac{p_{ji}}{q_{ji}} \quad (2.1)$$

2.2 Weakly Supervised Learning

Supervised learning with deep neural network has achieved great success with the help of large scale datasets. However, strong supervision information like the ground truth label for each data point is required under this framework, which is difficult to obtain or does not

exist for many real world scenarios. Therefore, learning frameworks with weak supervision signals are highly desirable.

Weakly supervised learning could refer to a number of situations[23]: 1) Incomplete supervision, where a large-size training dataset is provided, but only a small subset of the data points are labeled; 2) Inexact supervision, where some supervised labeling information is given, but not exact, for example, multi-instance learning [24, 25, 26]; and 3) Inaccurate supervision where the supervision information is not always the ground truth ones, for example, data with noisy labels. These situations are not mutually exclusive and can potentially occur together in one real world scenario.

For the inaccurate supervision case, it usually refers to the noisy label problem. Formally, we define a dataset $D = \{(x_1, \hat{y}_1) \cdots (x_n, \hat{y}_n)\}$ with n instances and their corresponding labels. Inside this dataset, $\hat{y}_i \in \{Y, N\}$ might not be the same as the ground truth label y_i and the corruption probability, $p(\hat{y}_i \neq y_i | x_i, y_i)$, is usually modeled as instance-dependent label noise, which depends on the features of the instance, x_i , and the ground truth label.

Many efforts have been made to improve the robustness of learning from noisy labels, and here we mainly introduce the ones from the deep learning community due to their wide applications in the daily life. As summarized in [27], different modifications such as sample selection, robust architecture, robust regularization, and loss adjustments have been proposed in the deep learning framework to mitigate the noisy label problem. We will focus on a loss reweighting approach, which is one of the most effective methods of loss adjustments. Inspired by importance reweighing [28], the key idea of loss reweighting is to assign smaller weights to samples with less confidence in their labels during training phase. Formally, the final reweighted loss for i th instance can be written as $w(x_i, \hat{y}_i)l(f(x_i; \theta), \hat{y}_i)$, where w is the weight term and $l(\cdot)$ is a label mismatch loss.

It depends on the reasonable estimation of label confidence and suitable design of weight, which are not straightforward. Considering our assumption of the instance-dependent label noise, data features and label information can be utilized to estimate the confidence of

each label. For example, confidence can be estimated via clustering over instances [29] or Bayesian methods applied on graphical models to combine prior information [30]. For reweighting the loss terms, active bias proposed in [31] assigns the weight as the prediction variances for instances with inconsistent predictions to emphasize that the labels for such data points have more uncertainty. DualGraph [32] utilizes graph neural network to generate distribution-level relation to reweight instance-level loss term. Finally, setting the weight of loss term for uncertain instances as a hyperparameter and sweeping over candidates to select the best one is also a common practice.

For inexact supervision, multi-instance learning is one of the typical cases. Formally, we define a bag as a collection of instances, for example, $X_i = \{x_{i1} \cdots x_{im}\}$ is a bag with m instances. Then, a dataset is defined as a collection of bags and their corresponding labels, i.e. $D = \{(X_1, \hat{y}_1) \cdots (X_n, \hat{y}_n)\}$ is a dataset with n bags, and $\hat{y}_i \in \{Y, N\}$ for $i = 1 \cdots n$. For i th bag, its label $\hat{y}_i = Y$ if there is at least one positive instance inside bag X_i .

The weakly supervised case, introduced in detail in Chapter 4, is a combination of inexact and inaccurate supervision, i.e. multi-instance learning and noisy label problems. In our case, the biomarkers detected inside one particular channel is a bag of instances; the clinical evidence only says whether the channel as whole (which might contain both types of biomarkers, epileptogenic and non-epileptogenic) is a problematic one or healthy one. A problematic channel can have various percentages of epileptogenic and non-epileptogenic biomarkers. Making the problem more challenging, the labels we are given to each channel are themselves noisy, since we can't really define which channels are purely problematic due to the limitation of current clinic domain knowledge.

2.2.1 A toy example for weighted loss function

A toy example is used here to demonstrate how partial information about noisy labels can be used to pick a good set of weights in the loss function to reduce the effect of such noise induced degradation in performance. The particular noise model and the associated terminologies

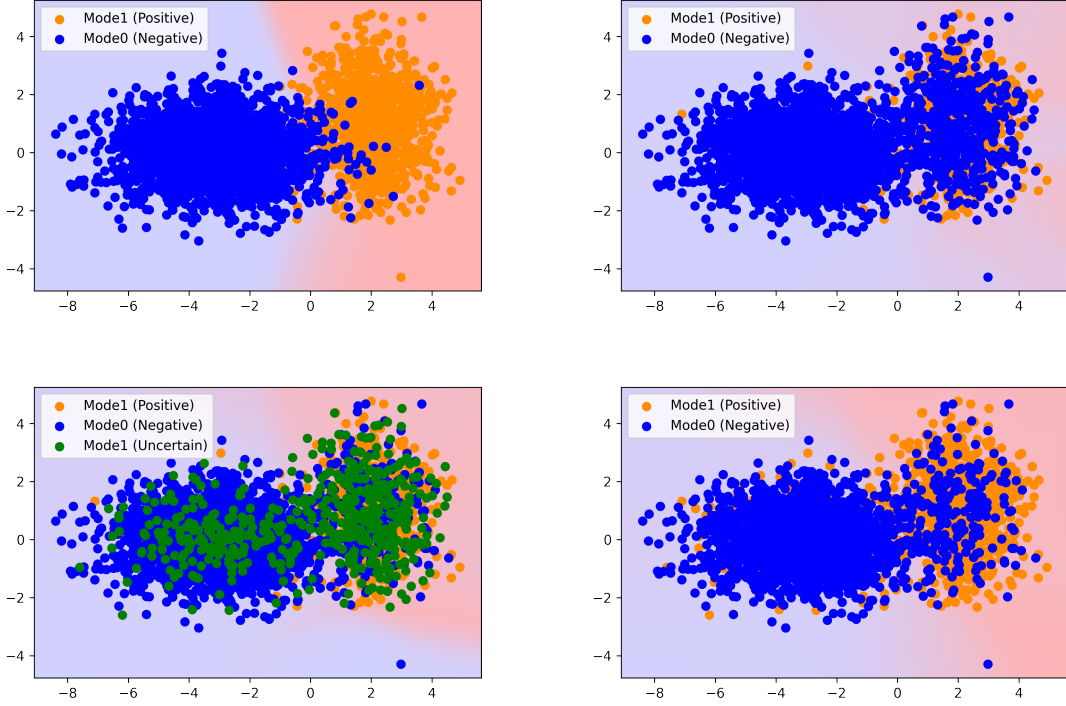


Figure 2.2: A toy example for denoising wrong labels with a robust loss function. *Upper Left* (GT-model): Samples are drawn from a two mode GMM with probabilities $2/3$ and $1/3$, respectively. The prediction probabilities of a supervised classifier based on these ground-truth labels are shown as a mixture of two colors in the background. Next, as discussed in Section 2.2.1, the GMM data samples are divided into four channels (or sets of data points), named as preserved, SOZ, non-SOZ but resected, and resected, respectively. Each such channel is a mixture of ground truth labeled samples, and all data points in a channel can be assigned, as a unit, only one label. This leads to weak labels for the ground truth data. *Upper Right* (SOZ-model): Only the SOZ channel samples are labeled as 1 and both the non-SOZ resected and preserved channels are labeled as 0. *Bottom Left* (WT-model): The samples from both the SOZ channels and non-SOZ resected channels (denoted as green) are labeled as 1 and preserved channels are labeled as 0. Different classifiers are trained by varying the weights associated with the samples in green. *Bottom right* (Resect-Model): All samples from the resected channels (SOZ and non-SOZ in the resected region) are labeled as 1 and all samples in preserved channels are labeled as 0. The results for the different scenarios, when evaluated with respect to the ground truth labels, are shown in Figure 2.3.

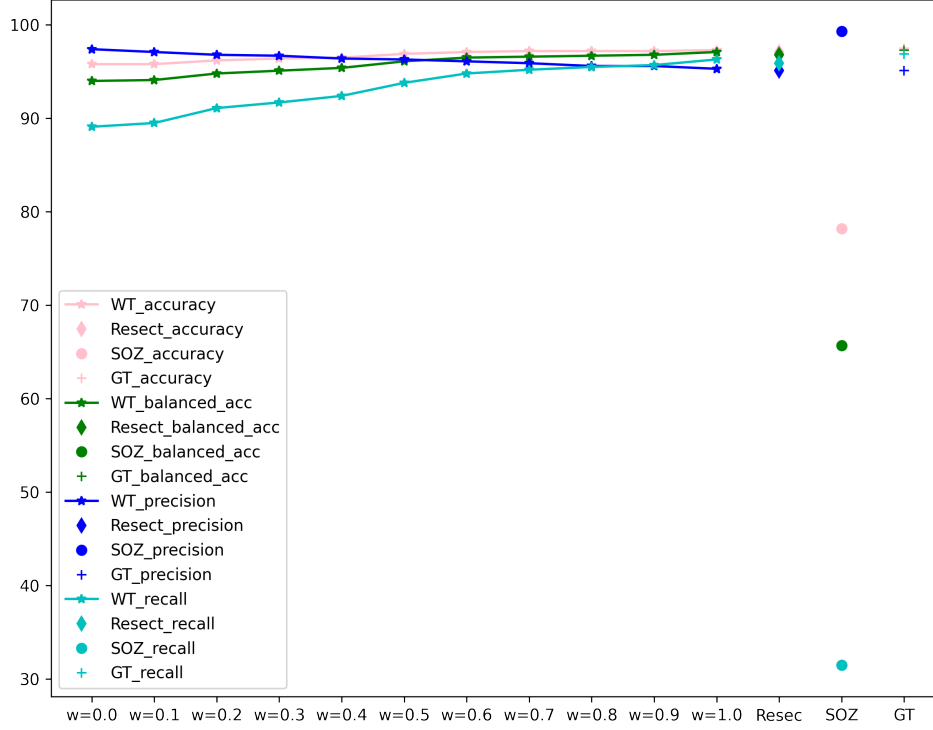


Figure 2.3: Performance of various models described in Section 2.2.1. The GT-model, where no labeling error is introduced, has a precision of 95%, and a recall of 97%. The WT-Model with weight 0.8, has a balanced precision and recall of 96% each. At weight=1 where the WT-model becomes the same as the Resect-model has slightly worse performance: precision 95% and recall 96%. On the other hand, and as expected, the SOZ-Model has an overall poor performance: precision 99%, and recall 32%. Thus even with considerable mislabeling of ground truth data, classifiers can be designed to purify labels.

are based on a real-life example as discussed in Chapter 4. We firstly consider two gaussian distributions with relatively small overlap in 2D space as two modes. To be more precise, samples are drawn from a two mode GMM. We call the ground truth mode-0 samples as healthy samples and the ground truth mode-1 as unhealthy samples. See Chapter 4 for some of the motivations behind using such terms. We further assume that there are three different kinds of channels that generate ground truth healthy and unhealthy samples in different proportions. All samples belonging to the same channel can be given either the label of 1 or 0, thus introducing noise into the ground truth labels. These channels are:

- (i) Preserved channels (containing 2,000 samples) where 90% of the samples are healthy and 10% are unhealthy;
- (ii) SOZ channel (containing 500 samples) where 90% samples are unhealthy and 10% samples are healthy;
- (iii) Uncertain or Resected-but-non-SOZ channel (500 samples) where 60% of the sample are unhealthy and 40% are healthy; and finally,
- (iv) Resected channel, is the union of the SOZ and the Resected-but-non-SOZ channels.

Different classification tasks can be set up by assigning different labels to these channels, and then performing weighting of the loss terms, when possible. The different cases are illustrated in Figure 2.2, and are as follows:

1. *GT-Model*: In this scenario a classifier is trained based purely on the correct ground truth labels. The performance of this classifier is the best one expect and is an upper bound on the performance of the other models;
2. *SOZ-Model*: In this labeling scenario, only the samples in the SOZ channels are assigned a label of 1 and the rest are assigned a label of 0. This is the most noisy case, as a lot of the ground truth modes are mislabeled; for example 60% of the 500 samples (Around 300 samples) in the non-SOZ channels are actually from ground truth mode-1 and yet is labeled as 0.
3. *WT-Model*: In this scenario, both the SOZ and non-SOZ-but-resected channels are given a label 1 and the preserved channel samples are assigned a label of 0. This gives us the opportunity to vary the weights of the loss terms in the uncertain channel, assigned a label of 1, from 0 to 1.
- r. *Resect-Model*: Here, all the samples in the resection channel are treated equally and assigned a weight of 1 in the loss function. Thus, this case corresponds to the WT-model, where the

weights of the samples in the uncertain channel are assigned a weight of 1.

For each model (including the case of WT-model with varying weights), we trained using a simple classifier comprising a three layer neural network. The decision boundary inferred from the classifier can be observed as the boundary of background color in Figure 2.2. The relative performance of different models is also given in Figure 2.3. As the figure shows, for the GT-model, the precision and recall are 95% and 97% respectively. For the WT-model, there are 250 samples that are from mode-0 (healthy) but labeled as 1, where the majority are unhealthy; similarly 200 samples are unhealthy (mode-1) but labeled as 0, where the majority is healthy. But as the weight is varied then it starts with high precision but lower recall, and then reaches a balanced value of 96% for both precision and recall. Thus even with considerable mislabeled samples (450 in total), a classifier with performance almost similar to the case where no labeling errors are introduced can be achieved.

2.3 Generative models

2.3.1 Introduction

Compared to a discriminative model which predicts the probability of a label assigned to the data, generative models learn the training data’s distribution and thus can not only generate new samples by sampling from the learnt distribution but also give out the estimated likelihood of each data point. Many generative models have been developed: Gaussian Mixture Model (GMM), Hidden Markov Model (HMM), Latent Dirichlet Allocation (LDA) and so on. These traditional generative models usually have explicit mathematical expressions and are learnt with carefully designed algorithms. Therefore, they are not scalable and with limited learning capacity. With the development of deep learning, neural networks, as a universal function approximators, have been applied in generative models to model high-dimensional complex data distributions. Different frameworks have been proposed to train these deep generative models: Generative Adversarial Network (GAN), Variational

AutoEncoder (VAE), Normalizing Flows (NF) and so on. Deep generative models have wide applications in physics, simulations and several real-world applications [33, 34, 35].

Even with the potential modeling capacity of neural networks, the task of training a deep generative model is not easy. The objective function of the training process is not straightforward due to the lack of one-to-one correspondence between generative samples and ground truth data distribution. GAN has shown promising results in generating realistic images via the novel training mechanism, i.e. training the generator in an adversarial way and using the discriminator to provide a measure of differences between the distribution of the generated data and that of the ground truth data, but is notorious for unstable training and mode collapse problems. NF and VAE train their models in a nonadversarial way, and thus mostly avoid such training problems. NF, however, assumes an invertible data sampling and density estimation process, which rarely holds in real world scenarios and greatly limits the scale of data it can apply to. Therefore, in the following sections, we focus more on VAE and the variants of VAE considering their wide applications and powerful modeling capacity. Chapter 5, uses and generalizes some of these concepts for the 3D human motion generation problem.

2.3.2 VAE and VAE variants

Variational Autoencoder was first proposed in [36]. As introduced in Section 1.2, VAE falls in the category of generative models where samples from a lower dimensional space are mapped to the higher dimensional data points. Such an approach adopts a parameterized deep neural network, $f_\theta(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^n$, often referred to as the decoder. In general, a parameterized distribution function $\mathcal{P}_z(\delta)$ (where $\mathbf{z}$ is the latent random variable of dimension d), and aims to learn the parameters (θ, δ) such that $D(\mathcal{P}_x, \mathcal{P}_{f_\theta(\mathbf{z}, \delta)})$ is minimized, where $D(p, q)$ is a distance metric between any two given distributions p and q .

It is instructive to define a few shorthand notations for the rest of this section. Let $p_\theta(\mathbf{x})$ be the density function of data, given the parameters θ of the decoder. Also, let $p_\theta(\mathbf{x}, \mathbf{z})$ be

the joint distribution of data $\mathbf{x}$ and latent variable $\mathbf{z}$. This joint distribution can be factored as the product of a prior density function, $p_\delta(\mathbf{z})$, and the conditional distribution of the data given the latent variable: $p_\theta(\mathbf{x}|\mathbf{z})$. In a standard VAE formulation one assumes that $p_\delta(\mathbf{z})$ is zero-mean identity-variance normal distribution and thus has no functional dependence on the parameter δ . As discussed later, the prior distribution itself can be parameterized in alternate formulations of VAE. For the rest of the discussions on the original VAE framework, we shall omit any reference to δ . Let, $p_\theta(\mathbf{z}|\mathbf{x})$ be the posterior distribution of the latent variable, given the data. In order to learn the decoder function, $f_\theta(\cdot)$, it is important that we compute this posterior distribution, so that one can train the network parameters, θ , by sampling from this posterior density. It is, however, a computationally intractable task, and one approach is to adopt a variational method. That is, $p_\theta(\mathbf{z}|\mathbf{x})$ is approximated by a parameterized density function, $q_\phi(\mathbf{z}|\mathbf{x})$. The parameters of $q_\phi(\mathbf{z}|\mathbf{x})$ – for example, if $q_\phi(\mathbf{z}|\mathbf{x})$ is assumed to be normal distribution then mean $\mu(\mathbf{x}) \in \mathbb{R}^d$ and covariance $\sigma(\mathbf{x})^2 \mathbf{I}_{d \times d}$ would be the required parameters – are obtained by feeding $\mathbf{x}$ through an encoder neural network with parameter set ϕ . Such an encoder-decoder architecture is shown in Figure 2.4.

Using a standard variational technique one can express the log-likelihood of the data given parameters θ as follows:

$$\log p_\theta(\mathbf{x}) = \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} [\log p_\theta(\mathbf{x})] \quad (2.2)$$

$$= \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \left(\frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_\theta(\mathbf{z} | \mathbf{x})} \right) \right] \quad (2.3)$$

$$= \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \left(\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} | \mathbf{x})} \cdot \frac{q_\phi(\mathbf{z} | \mathbf{x})}{p_\theta(\mathbf{z} | \mathbf{x})} \right) \right] \quad (2.4)$$

$$= \underbrace{\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \left(\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} | \mathbf{x})} \right) \right]}_{ELBO} + \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \left[\log \left(\frac{q_\phi(\mathbf{z} | \mathbf{x})}{p_\theta(\mathbf{z} | \mathbf{x})} \right) \right] \quad (2.5)$$

The first term is the *evidence lower bound* (ELBO), which is now a function of both θ and ϕ , and second term is the Kullback-Leibler (KL) divergence between estimated posterior $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{z}|\mathbf{x})$. The idea in a variational approach is to maximize the ELBO, and since $\log p_\theta(\mathbf{x})$ is independent of ϕ , the KL divergence term will go to zero, if the neural networks

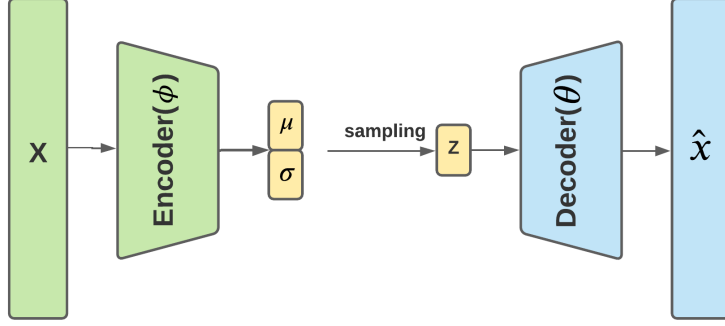


Figure 2.4: A schematic of the VAE architecture.

have enough expressive power.

To train a VAE for a given dataset D , the loss function, which needs to be minimized, is the negative of the ELBO, and can be rewritten as

$$-l_{\theta,\phi}(D) = \sum_{x \in D} \mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x})} [\log(p_{\theta}(\mathbf{x}, \mathbf{z})) - \log(q_{\phi}(\mathbf{z} | \mathbf{x}))] \quad (2.6)$$

$$= \sum_{x \in D} (\mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x})} [\log(p_{\theta}(\mathbf{x} | \mathbf{z}))] + D_{KL}(q_{\phi}(\mathbf{z} | \mathbf{x}) \parallel p_{\theta}(\mathbf{z}))). \quad (2.7)$$

For the standard VAE, we assume the prior $p(\mathbf{z})$ is a Normal distribution with zero mean and unit variance, and the estimated posterior $q_{\phi}(\mathbf{z}|\mathbf{x})$ is assumed to be a normal distribution with mean $\mu(\mathbf{x}) \in \mathbb{R}^d$ and covariance $\sigma^2(\mathbf{x})\mathbf{I}_{d \times d}$. Taking the powerful transformation capacity of neural networks into consideration, this assumption generally works. With this choice of $q_{\phi}(\mathbf{z}|\mathbf{x})$, we have:

$$\begin{aligned}
-l_{\theta,\phi}(D) &= \sum_{x \in D} (\mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x})} [\log(p_{\theta}(\mathbf{x} | \mathbf{z}))] + D_{KL}(q_{\phi}(\mathbf{z} | \mathbf{x}) \parallel p_{\theta}(\mathbf{z}))) \\
&= \sum_{x \in D} (\mathbb{E}_{\mathbf{z} \sim q_{\phi}(\mathbf{z}|\mathbf{x})} [\log(p_{\theta}(\mathbf{x} | \mathbf{z}))] + \frac{1}{2} \left[- \sum_i^d (\log \sigma_i^2(\mathbf{x}) + 1) + \sum_i^d \sigma_i^2(\mathbf{x}) + \sum_i^d \mu_i^2(\mathbf{x}) \right]).
\end{aligned} \tag{2.8}$$

To derive the second term in the above equation, we note that given two Gaussian distributions in $\mathbb{R}^d$, P_1 and P_2 , with mean vectors (μ_1, μ_2) and covariance matrices (Σ_1, Σ_2) , their KL divergence can be expressed as:

$$D_{KL}(P_1 \parallel P_2) = \frac{1}{2} \left(\log \frac{\det \Sigma_2}{\det \Sigma_1} - n + \text{tr}(\Sigma_2^{-1} \Sigma_1) + (\mu_2 - \mu_1)^T \Sigma_2^{-1} (\mu_2 - \mu_1) \right) [37].$$

Applying this formula to our special case we get:

$$\begin{aligned}
D_{KL}[q_{\phi}(\mathbf{z} | \mathbf{x}) \parallel p_{\theta}(\mathbf{z})] &= \frac{1}{2} \left[\log \frac{|\Sigma_2|}{|\Sigma_1|} - n + \text{tr} \{ \Sigma_2^{-1} \Sigma_1 \} + (\mu_2 - \mu_1)^T \Sigma_2^{-1} (\mu_2 - \mu_1) \right] \\
&= \frac{1}{2} \left[\log \frac{|I|}{|\Sigma|} - n + \text{tr} \{ I^{-1} \Sigma \} + (0 - \mu)^T I^{-1} (0 - \mu) \right] \\
&= \frac{1}{2} [-\log |\Sigma| - n + \text{tr} \{ \Sigma \} + \mu^T \mu] \\
&= \frac{1}{2} \left[-\log \prod_i \sigma_i^2 - n + \sum_i \sigma_i^2 + \sum_i \mu_i^2 \right] \\
&= \frac{1}{2} \left[-\sum_i \log \sigma_i^2 - n + \sum_i \sigma_i^2 + \sum_i \mu_i^2 \right] \\
&= \frac{1}{2} \left[-\sum_i (\log \sigma_i^2 + 1) + \sum_i \sigma_i^2 + \sum_i \mu_i^2 \right].
\end{aligned}$$

The encoder and decoder networks with parameters, ϕ and θ , respectively can be trained jointly with the loss function in Equation (2.8).

Instead of assuming that the prior distribution $p(\mathbf{z})$ is a fixed zero-mean unit-variance Gaussian, researchers have argued that a more flexible model of the prior is required since it plays an important role training the encoder and decoder pair: 1) mismatch between prior and the data manifold in a ground truth generative model will lead to the failure of

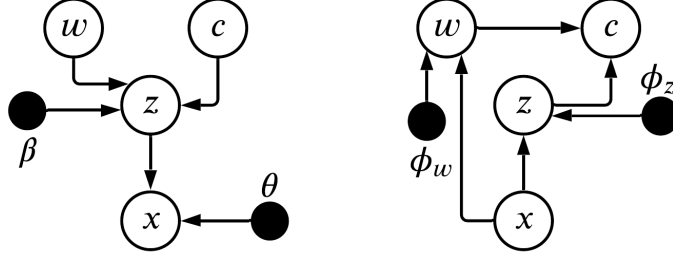


Figure 2.5: Graphical models for the Gaussian mixture variational autoencoder (GMVAE) showing the generative model (left) and the variational family (right).

generative modeling of the real data distribution. 2) Using a non-parameterized prior could make the latent space less interpretable. For example, it will encourage latent codes to have a much higher density around the origin, leading to the difficulty of clustering and sampling from codes belonging to the same mode, when the data contains multiple modes.

Mixture of Gaussian prior with learnable hyperparameters (MoG) was proposed in [38, 39] to provide more modeling capacity in the latent space. It models the latent space as a mixture of Gaussian which is generated by random variable $\mathbf{w}$ and categorical variable $\mathbf{c}$. Figure 2.5 shows the generative process and its generative model follows Equation (2.9):

$$\begin{aligned}
\mathbf{w} &\sim \mathcal{N}(0, \mathbf{I}) \\
\mathbf{c} &\sim \text{Mult}(\boldsymbol{\pi}) \\
\mathbf{z} \mid \mathbf{c}, \mathbf{w} &\sim \prod_{k=1}^K \mathcal{N}(\boldsymbol{\mu}_{c_k}(\mathbf{w}; \beta), \text{diag}(\boldsymbol{\sigma}_{c_k}^2(\mathbf{w}; \beta)))^{c_k} \\
\mathbf{x} \mid \mathbf{z} &\sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{z}; \theta), \text{diag}(\boldsymbol{\sigma}^2(\mathbf{z}; \theta))) \text{ or } \mathcal{B}(\boldsymbol{\mu}(\mathbf{z}; \theta)).
\end{aligned} \tag{2.9}$$

Similarly as the ELBO in the Equation (2.5), the ELBO for MoG with variational family shown in Figure 2.5 can be rewritten as Equation (2.10). Following the decomposition

in $q(\mathbf{z}, \mathbf{w}, \mathbf{c} \mid \mathbf{x}) = \prod_i q_{\phi_z}(\mathbf{z}_i \mid \mathbf{x}_i) q_{\phi_w}(\mathbf{w}_i \mid \mathbf{x}_i) p_{\beta}(\mathbf{c}_i \mid \mathbf{z}_i, \mathbf{w}_i)$, it can be further derived as shown in Equation (2.11) with reconstruction term, conditional prior term, w prior term and categorical variable c prior term. More details can be found in [38].

$$\mathcal{L}_{ELBO} = \mathbb{E}_q \left[\frac{p_{\beta, \theta}(\mathbf{x}, \mathbf{z}, \mathbf{w}, \mathbf{c})}{q(\mathbf{z}, \mathbf{w}, \mathbf{c} \mid \mathbf{x})} \right]. \quad (2.10)$$

$$\begin{aligned} \mathcal{L}_{ELBO} = & \mathbb{E}_{q(\mathbf{z} \mid \mathbf{x})} [\log p_{\theta}(\mathbf{x} \mid \mathbf{z})] - \mathbb{E}_{q(\mathbf{w} \mid \mathbf{x}) p(\mathbf{c} \mid \mathbf{z}, \mathbf{w})} [KL(q_{\phi_z}(\mathbf{z} \mid \mathbf{x}) \parallel p_{\beta}(\mathbf{z} \mid \mathbf{w}, \mathbf{c}))] \\ & - KL(q_{\phi_w}(\mathbf{w} \mid \mathbf{x}) \parallel p(\mathbf{w})) - \mathbb{E}_{q(\mathbf{z} \mid \mathbf{x}) q(\mathbf{w} \mid \mathbf{x})} [KL(p_{\beta}(\mathbf{c} \mid \mathbf{z}, \mathbf{w}) \parallel p(\mathbf{c}))]. \end{aligned} \quad (2.11)$$

However, as mentioned in [40], although it boosts the performance of vanilla VAE, it influences the estimated posterior in the same manner as vanilla VAE and the gradient of ELBO is also similar. Instead, [40] proposed a mixture of Gaussian prior given by variational posteriors conditioned on learnable pseudo-inputs, which couples the prior and posterior. In this setting, instead of using standard normal distribution, the prior is parameterized by $p_{\lambda}(\mathbf{z})$ and the ELBO can be rewritten with three terms: reconstruction term, posterior term, and prior term, as shown in Equation (2.12). The prior that optimize this ELBO can be solved via maximizing the Lagrange function Equation (2.13) and the solution can be written as $p_{\lambda}^*(\mathbf{z}) = \frac{1}{N} \sum_{n=1}^N q_{\phi}(\mathbf{z} \mid \mathbf{x}_n)$, i.e, the aggregated posterior. To avoid overfitting issues, it can be further approximated by a mixture of posterior from pseudo-inputs, i.e $p_{\lambda}(\mathbf{z}) = \frac{1}{K} \sum_{k=1}^K q_{\phi}(\mathbf{z} \mid \mathbf{u}_k)$, where inputs $\mathbf{u}_i$ are learnable. More details can be found in [40].

$$\begin{aligned} \mathcal{L}(\phi, \theta, \lambda) = & \mathbb{E}_{\mathbf{x} \sim q(\mathbf{x})} [\mathbb{E}_{q_{\phi}(\mathbf{z} \mid \mathbf{x})} [\ln p_{\theta}(\mathbf{x} \mid \mathbf{z})]] + \\ & + \mathbb{E}_{\mathbf{x} \sim q(\mathbf{x})} [\mathbb{H}[q_{\phi}(\mathbf{z} \mid \mathbf{x})]] + \\ & - \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z})} [-\ln p_{\lambda}(\mathbf{z})] \end{aligned} \quad (2.12)$$

$$\max_{p_{\lambda}(\mathbf{z})} -\mathbb{E}_{\mathbf{z} \sim q(\mathbf{z})} [-\ln p_{\lambda}(\mathbf{z})] + \beta \left(\int p_{\lambda}(\mathbf{z}) d\mathbf{z} - 1 \right) \quad (2.13)$$

Instead of modeling prior with simple distributions, [10] models the latent space as a learnable manifold that defined by transport operator to better match the data manifold.

2.4 Imitation learning

In real life applications, a lot of data present in the format of time-series, such as signals from the domain of health-care, daily conversations, stock prices and so on. Reinforcement learning (RL) provides a mathematical framework to model an intelligent agent to learn from those kinds of sequential data and make wise decisions. However, in the typical RL setting, it requires well defined reward function which greatly limits the application scenarios and the learning capacity of the agents since the reward function is not always available. As a feasible solution to such problems, imitation learning (IL), doesn't require any reward functions and learns from expert demonstrations. The goal of imitation learning is to mimic the expert behaviors in a given task by learning from expert demonstrations without a reward function. There are two mainstream methods to tackle this problem: behavior cloning (BC) and Inverse Reinforcement learning (IR) [41].

2.4.1 Reinforcement learning(RL)

Reinforcement learning is concerned with how the intelligent agent takes actions in an interactive environment to achieve a particular goal defined in reward. It is one of the three machine learning frameworks besides supervised learning and unsupervised learning. It differs from the other twos in that the training set is not static and updated via the agent's interaction with environments.

Formally, it can be modeled as a Markov Decision Process (MDP), which includes several key components: a set of environment and agent states S , a set of agent actions A and transition probability which models the probability of transiting from state $s_t \in S$ at time t to $s_{t+1} \in S$ at time $t+1$ under action $a_t \in A$ at time t as $p(s_{t+1}|s_t, a_t)$, immediate reward for

transitioning from state s_t to s_{t+1} under action a_t as $R(s_t, a_t, s_{t+1})$. The goal of reinforcement learning is to find an optimal policy $\pi: A \times S \rightarrow [0, 1]$, i.e. $\pi(a, s) = p(a_t = a | s_t = s)$ which maximizes the expected accumulated rewards along the trajectories.

2.4.2 Behavior Cloning (BC)

Behavior Cloning focuses on learning the policy from the expert demonstrations using supervised learning [42]. However, it has unavoidable compounding errors which leads unreasonable actions for unseen states in the training dataset. It fails to model the long-range dependencies in the interactions and lacks generalization ability when the training data is limited.

2.4.3 Inverse Reinforcement Learning (IR)

Inverse reinforcement learning focuses on recovering the reward function when the expert demonstrations are available [43]. Once the reward function is available, the optimal policy can also be inferred for imitation learning task. However, to optimize for the suitable reward function, it usually involves computing the optimal policy inside the loop, which is computationally expensive [44]. Generative Adversarial imitation learning [45] is proposed with better sample efficiency in terms of the number of expert demonstrations required. It learns the policy and reward function with generative adversarial framework and thus improves the robustness of the unseen states also.

Although GAIL provides an efficient framework, imitation learning with diverse expert demonstrations is still a challenging task. This diversity or multiple modes varies according to the scenarios: such as multiple human experts performing the same task in different ways or pursuing multiple unlabeled objectives, e.g. a robotic arm moving towards different goals. In such scenarios, neither BC nor GAIL enjoys any explicit mechanism to capture different modes of behavior. Particularly, BC averages out actions from different modes. On the

other hand, GAIL learns a policy that captures only a subset of control behaviors, because adversarial training is prone to mode collapse [46, 47].

CHAPTER 3

Interpretable Dimensionality Reduction: An Efficient Algorithm for Non-Negative Matrix Factorization

3.1 Introduction

Extracting low dimensional representations of a data matrix is one of the fundamental steps for many data analysis tasks. Standard methods for finding a low dimensional representation include Principal Component Analysis (PCA)[48], Independent Component Analysis (ICA)[49, 50], and singular value decomposition (SVD). In many applications, however, the data has special structure, and the factors are more meaningful and interpretable if they satisfy certain constraints; for example, in Network Component Analysis (NCA)[51, 52] the zero entries of one of the factors are predetermined based on a priori information. Non-negative matrix factorization (NMF) is the problem of finding good approximations of a given non-negative matrix as a low-rank product of two non-negative matrices. This linear dimensionality reduction technique has been used very successfully for a large variety of machine learning and data mining tasks, including hyperspectral imaging [53], topic modeling [54], audio denoising [55], and video structuring [56, 57], communication complexity [58], probability [59], and computational geometry [60].

A standard optimization framework for computing the nonnegative factors uses the Frobenius norm, and a non-convex formulation is given by

$$\underset{U \geq 0, V \geq 0}{\text{minimize}} \quad f(U, V) = \frac{1}{2} \|X - UV^T\|_F^2. \quad (3.1)$$

There exist other variants of the NMF depending on the distance measure used to quantify the quality of the approximation. For example, in [21] both the Frobenius norm (as in 3.1)) and the generalized Kullback-Leibler divergence measure are proposed to measure approximation error, in [61] the author utilizes Itakura-Saito divergence for music analysis and in [62] total variation norm is designed to extract image patterns. In yet another variation (see Section 3.5), the approximation error minimization in 3.1 is done only over a subset of the entries of the data matrix X . In such a case it is referred to as the *matrix completion factorization problem*.

3.2 Related work

Several algorithms have been developed to find good local minima of (3.1) and they can be divided into three general classes: multiplicative update algorithms, gradient descent algorithms, and alternating least squares algorithms. The prototypical multiplicative algorithm (Mult) originated with Lee and Seung [21] and the algorithm is initialized with non-negative factor matrices and they remain non-negative throughout the iterations. However, it is notoriously slow to converge and requires many more iterations than alternatives such as the gradient descent and alternating least squares algorithms. To overcome such shortcomings accelerated versions of the multiplicative algorithm have been proposed [63].

NMF algorithms of the second class are based on gradient descent methods (Grad-Mult)[64]. These algorithms always take a step in the direction of the negative gradient, the direction of steepest descent. In order to ensure non-negativity of the factor matrices after the descent step, a projection step is employed by these algorithms.

NMF algorithms of the third class are based on alternating optimization framework. This framework exploits the fact that, while the optimization problem of (3.1) is not convex in both U and V , it is convex in either U or V . Therefore, these algorithms update the factor matrices in an alternating fashion by solving simpler subproblem. The subproblem that

arises for each factor is non-negative linear least squares (NNLS). Various algorithms have been developed to solve NNLS and we list the most popular ones below:

- ALS [65]: Solve unconstrained least squares followed by a simple projection to the non-negative orthant
- AO-ADMM [66]: Alternating direction method of multipliers (ADMM) is used to solve NNLS. The algorithm uses cholesky decomposition of the mixing matrix $U^T U$ for an efficient implementation
- Active set [67]: This algorithm searches iteratively for optimal set of variables to optimize over by exchanging a variable between active and passive sets
- Hierarchical ALS (HALS) [68]: Each column of left factor matrix (U) or each row of the right factor matrix (V) are updated by solving an unconstrained least squares and then projecting them to the non-negative orthant.

The three classes of NMF algorithms have one thing in common: *these iterative algorithms either start with a feasible point* or obtain the feasibility of the iterates after each iteration.

3.3 Motivations

3.3.1 Complexity

The complexity of a special case of the optimization problem in (3.1) where one asks whether there is an *exact* non-negative solution with a cost function value of 0, has been studied extensively in the literature. More precisely, the *Exact NMF* problem takes as input a matrix $X \geq 0 \in \mathbb{R}^{n \times m}$ with rank equal to r , $r \geq 1$, and asks whether there is a pair of matrices (U, V) , where $U \geq 0 \in \mathbb{R}^{n \times r}$ and $V \geq 0 \in \mathbb{R}^{m \times r}$, such that $X = UV^T$. Vavasis [69] and [70] have shown that this problem is NP hard and there is no $(nm)^{o(r)}$ time algorithm for *Exact NMF*, unless $P = NP$. For some special cases of the Exact NMF problem,

namely the separable case, first Donoho [71] and then [70] have developed polynomial time algorithms in n , m , and r to compute the exact factorization if one exists. Since the exact factorization scenario is a special case of the general NMF problem, it implies that the NMF problem is also NP-hard, and we cannot find any provably efficient algorithm unless $P = NP$. *In order to develop an alternate paradigm for solving (3.1) one will have to define optimization problems that judiciously exploit geometry or structure inherent in the landscape of the underlying optimization problem.* We next provide one such geometrical framework.

3.3.2 Going from SVD to known Exact NMF factors

Let $X = U_0 V_0^T$, where $X \geq 0 \in \mathbb{R}^{n \times m}$, $U_0 \geq 0 \in \mathbb{R}^{n \times r}$, $V_0 \geq 0 \in \mathbb{R}^{m \times r}$, and let all the matrices have rank $= r$. Let X admit the following SVD: $X = U \Sigma V^T$. We define the scaled SVD factor matrices U^* and V^* as $U^* = U \Sigma^{\frac{1}{2}}$, and $V^* = V \Sigma^{\frac{1}{2}}$. Then, as shown in Vavasis [69], there must exist a matrix $A \in \mathbb{R}^{r \times r}$ such that $U_0 = U^* A$ and $V_0 = V^* A^{-T}$. Thus by considering an SVD of A , $A = R_1 \Lambda R_2^T$, we find that there exist rotation matrices $R_1, R_2 \in \mathbb{R}^{r \times r}$ ($R_1 R_1^T = I = R_2 R_2^T$) and diagonal scaling matrix Λ such that

$$U_0 = U^* R_1 \Lambda R_2^T, V_0 = V^* R_1 \Lambda^{-1} R_2^T. \quad (3.2)$$

We can solve for R_1 and Λ using the norm balancing constraint given by

$$U_0 U_0^T = U^* R_1 \Lambda^2 R_1^T U^{*T}, V_0 V_0^T = V^* R_1 \Lambda^{-2} R_1^T V^{*T}. \quad (3.3)$$

(3.3) can be solved by extracting the submatrices of U_0 and U^* consisting of r linearly independent rows

$$U_{0_r} U_{0_r}^T = U_r^* R_1 \Lambda^2 R_1^T U_r^{*T}. \quad (3.4)$$

Linear system in (3.4) has full rank and thus has unique R_1 and Λ . After computing R_1 and Λ , the rotation matrix R_2 can be computed by solving the optimization problem,

$$R_2 = \arg \min_{R_2^T R_2 = I} \left\| \begin{bmatrix} U_0 \\ V_0 \end{bmatrix} - \begin{bmatrix} U^* R_1 \Lambda \\ V^* R_1 \Lambda^{-1} \end{bmatrix} R_2 \right\|_F. \quad (3.5)$$

The problem in (3.5) is known as the Orthogonal Procrustes Problem [72] and admits a closed form solution given by $R_2 = CD^T$, where C and D are the factors obtained from the SVD of $\begin{pmatrix} \begin{bmatrix} U^* R_1 \Lambda \\ V^* R_1 \Lambda^{-1} \end{bmatrix}^T \begin{bmatrix} U_0 \\ V_0 \end{bmatrix} \end{pmatrix}$.

3.3.3 Going from SVD to unknown Exact NMF factors

To solve the Exact NMF problem, we can first compute the scaled SVD factors, (U^*, V^*) , of a nonnegative X , and then solve for $R_1, R_2 \in \mathbb{R}^{r \times r}$ ($R_1 R_1^T = I = R_2 R_2^T$), and diagonal scaling matrix Λ , such that $U^* R_1 \Lambda R_2 \geq 0$, and $V^* R_1 \Lambda^{-1} R_2 \geq 0$. If X has an exact NMF solution, then this problem is feasible. One can define the following non-convex optimization problem to solve for the rotation and scaling matrices:

$$\begin{aligned} & \underset{R_1, R_2, \Lambda}{\text{minimize}} && \sum_{i=1}^n \sum_{j=1}^r h((U^* R_1 \Lambda R_2)_{ij}) + \sum_{i=1}^m \sum_{j=1}^r h((V^* R_1 \Lambda^{-1} R_2)_{ij}) \\ & \text{subject to} && R_1^T R_1 = I, R_2^T R_2 = I, \Lambda \geq 0 \end{aligned} \quad (3.6)$$

where $h(q) = \max\{0, -q\}$. *Clearly the optimization problem in (3.6) has a minimum value = 0 if and only if X admits an Exact solution.* This, however, is an NP Hard problem, and our ADMM implementations for solving (3.6) (a generalized version of the eNMF algorithm presented in Section 3.4, and also presented in Appendix) did yield exact non-negative solutions in reasonable time, only when X has low dimensions. For example for synthetic datasets (for data generation procedure read Section 3.6.1) with dimensions $n = 50, m = 40, r = 10, s = 0.1$ and $n = 100, m = 80, r = 20, s = 0.1$ (where s is the sparsity factor) the ADMM implementation yielded exact non-negative solutions. However, for an X when it's dimensions n, m are large, the ADMM implementations *yielded factors that are only close to the positive orthant, which are still external infeasible points* (see Figure 3.1 for details) *This computational intractability of finding optimal R_1, Λ, R_2 to move the SVD solutions to the feasible region* (even when such solutions exist), *prompted a more approximate approach, as described next in the context of the general NMF problem.*

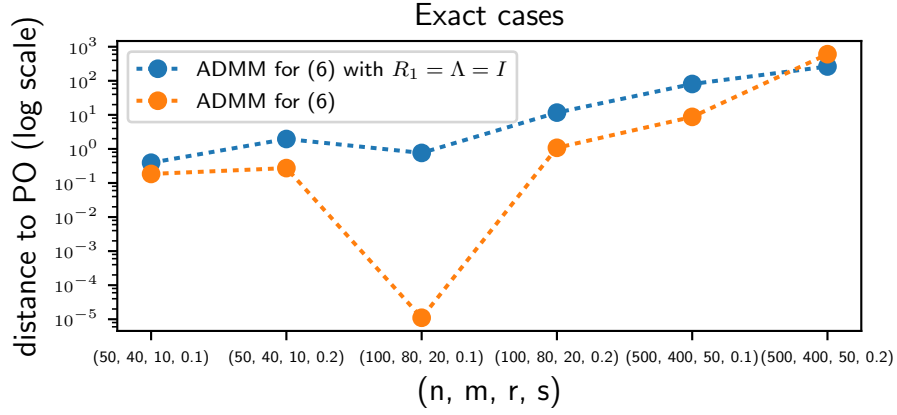


Figure 3.1: Distance to the positive orthant (as defined by the sum of the absolute values of the negative entries of the factor matrices post optimization; see Equations (3.6) and (3.8)) is plotted for synthetic exact datasets with various dimensions and sparsity levels. For lower dimensions, ADMM implementation of (6) generates exact non-negative solutions. For larger dimensions, ADMM implementations for both the general and relaxed version (involving only a single rotation matrix R_2 , and setting $R_1 = \Lambda = I$) generate factors that are only close to the positive orthant (PO), thus requiring further processing.

3.3.4 General NMF: Going from SVD to the Positive Orthant

The unconstrained optimization version of the problem in (3.1) has a well known global optimum (the minimum value, however, is no longer necessarily $= 0$ as in the case of Exact NMF), given by the truncated singular value decomposition (SVD) of the data matrix X , (U^*, V^*) , as defined in the previous section. In general, the optimal value of (3.1) is bounded below by the optimal value of the unconstrained problem. Therefore, solving (3.6) exactly would still generate factor matrices that are outside the positive orthant and we will have to find a way to move to the positive orthant from this close exterior point. Hence, *to make the problem of getting an approximate solution to (3.6) more tractable, we restrict ourselves, to the case where $R_1 = I$, and $\Lambda = I$.* From Figure 3.1 it can be observed that this relaxation produces factor matrices that are comparable (in terms of distance to the positive orthant (PO)) to the ones obtained by solving the general problem in (3.6). In the relaxed version, we search for a closest point to the positive orthant (as defined by the sum of the absolute values of the negative elements of the point) from the set:

$$\mathcal{X}^* = \{(U^*R, V^*R) \mid R^T R = I, R \in \mathbb{R}^{r \times r}\}. \quad (3.7)$$

Next, to explain further why such a close point in $\mathcal{X}^*$ is a “good” point to start at, we consider the following scenario: If we do a descent on $f(U, V)$ starting from any local minimum of the *NMF* problem, then we will converge to a pair of factor matrices (U, V) and *this external unconstrained global minimum is in some sense “closest” to the boundary with respect to the local minimum we started at.* This external globally-optimal (U, V) is in the same level-set as (U^*, V^*) and as shown in [73] this level set of globally optimal solutions is the set $\mathcal{X}^*$ (modulo any scaling of the (U, V)). In light of this geometric intuition, *we can “reverse” the above descent process: start with an element of $\mathcal{X}^*$ that is close to the boundary (based on suitable distance measures) and then use simple exterior methods to find a local minimum of the *NMF* problem.* We next present such an algorithm for the *NMF*.

3.4 Exterior NMF Algorithm

The main block of our algorithm consists of finding a suitably rotated globally optimal point which can then be used as an initial point to an exterior method to construct a solution to the *NMF* problem. In the following subsections, we describe the stages in detail.

3.4.1 Optimal rotation of unconstrained global optimum

An orthogonal rotation that minimizes the sum of the negative entries in U^*R and V^*R can be found by solving the optimization problem in (3.6) with $R_1 = \Lambda = I$. Although the problem is non-convex, we can use ADMM as a heuristic method to solve it [74]. In order to aid the ADMM formulation, we introduce the following notation

$$W = \begin{bmatrix} U^* \\ V^* \end{bmatrix} \in \mathbb{R}^{(n+m) \times r}, Z \in \mathbb{R}^{(n+m) \times r},$$

where Z is the splitting variable. Then, with the above notation we can rewrite the optimization problem in standard ADMM form given by

$$\begin{aligned} & \underset{Z, R}{\text{minimize}} && \sum_{i=1}^{n+m} \sum_{j=1}^r h(Z_{ij}) \\ & \text{subject to} && Z - WR = 0 \\ & && R^T R = I. \end{aligned} \tag{3.8}$$

The ADMM for solving (3.8) is given in Algorithm 1.

Algorithm 1 Rotation

Input: U, V, R^0, Y^0, ρ

Stack U, V vertically to get W

Repeat until convergence:

$$\begin{aligned} Z^{(k+1)} &\leftarrow \underset{Z}{\operatorname{argmin}} \left(\sum_{i=1}^{n+m} \sum_{j=1}^r h(Z_{ij}) + \right. \\ &\quad \left. \frac{\rho}{2} \|Z - WR^{(k)} + \frac{1}{\rho} Y^{(k)}\|_F^2 \right) \\ R^{(k+1)} &\leftarrow \underset{R^T R = I}{\operatorname{argmin}} \left(\frac{\rho}{2} \|Z^{(k+1)} - WR + \frac{1}{\rho} Y^{(k)}\|_F^2 \right) \\ Y^{(k+1)} &\leftarrow Y^{(k)} + \rho(Z^{(k+1)} - WR^{(k+1)}) \\ k &\leftarrow k + 1 \end{aligned}$$

Output: UR, VR

3.4.1.1 Updates for Z

We can update each entry of Z independently. The updates can be obtained by solving a simple quadratic problem:

$$Z_{ij}^{(k+1)} = \begin{cases} b_{ij} & b_{ij} > 0 \\ 0 & -\frac{1}{\rho} \leq b_{ij} \leq 0 \\ b_{ij} + \frac{1}{\rho} & b_{ij} < -\frac{1}{\rho}, \end{cases} \quad (3.9)$$

where $b_{ij} = (WR^{(k)} - \frac{1}{\rho} Y^{(k)})_{ij}$.

3.4.1.2 Updates for R

We can update R by solving a non-convex problem known as Orthogonal Procrustes Problem [72]. If we define $B = Z^{(k+1)} + \frac{1}{\rho} Y^{(k)}$, then the update for R is given by

$$R^{(k+1)} = CD^T, \quad (3.10)$$

where C and D are obtained from the SVD $W^T B = C \Sigma D^T$.

3.4.2 Exterior penalty method for attaining feasibility

The rotated point returned by Algorithm 1 is expected to be close to the positive orthant but not necessarily inside it (unless there is an NMF solution which is also globally optimal for the unconstrained problem; see Figure 3.5 for an example). We use an exterior penalty method to attain feasibility of the NMF factors starting from the rotated point:

$$\min_{U,V} \frac{1}{2} \|X - UV^T\|_F^2 + \delta_u \sum_{i,j} h(U_{ij}) + \delta_v \sum_{i,j} h(V_{ij}),$$

where $h(q) = \max\{0, -q\}$, and δ_u, δ_v are the penalty parameters. We solve the optimization problem by iteratively updating U and V using gradient descent.

3.4.2.1 Updating U

Assuming δ_u is large, the partial derivatives of the cost function for negative entries of U are approximately $-\delta_u$, and the negative entries can be updated as,

$$U_{ij} = U_{ij} + \rho_u \delta_u, \quad \forall (i, j) \in I_-, \quad (3.11)$$

where I_- denotes the set of negative entries and ρ_u is a fixed step size.

Updating the non-negative entries of U in any given row does not affect the gradient of the cost function for non-negative entries of U in other rows and we can leverage this to update non-negative entries in the rows of U independently. The updates for all the nonnegative entries in the i^{th} row can be computed as

$$U_{ij} = (U_{ij} - d_i^* (\nabla_U f)_{ij})_+, \quad \forall (i, j) \in I_+, \quad (3.12)$$

where I_+ denotes the set of non-negative coordinates, $\nabla_U f = (UV^T - X)V$, and the projection operator, $(x)_+ = \max\{0, x\}$, retains their nonnegativity, and d_i^* is the optimal step size. In order to find the optimal step size, we need to formulate an optimization problem. To aid the formulation, let's introduce some notation

- $E^{(k+1)} = X - U^{(k+1)}V^{(k+1)T}$

where $E^{(k+1)}$ is the error matrix at $(k+1)$ iteration. Then with the above definitions, we can define the optimization problem as $\min_{d_i} L = \|E^{(k+1)}(i, :)\|^2$, or

$$\min_{d_i} \|X(i, :) - [U^{(k)}(i, :) - d_i \{M_{U_+} \circ \nabla_U f\}(i, :)] V^{(k)T}\|^2 \quad (3.13)$$

where M_{U_+} is the binary vector mask over the negative entries of U . The problem given by (3.13) is an unconstrained quadratic problem and has a global minimum. The global minimum can be computed by setting the gradient with respect to d_i to zero

$$\nabla_{d_i} L = \sum_{j=1}^n \{2E^{(k)}(i, j)a_j + 2a_j^2 d_i\}$$

where $a_j = \{M_{U_+} \circ \nabla_U f\}(i, :) V^{(k)T}(:, j)$. Solving $\nabla_{d_i} L = 0$, we get

$$d_i^* = \frac{-\sum_{j=1}^n E^{(k)}(i, j)a_j}{\sum_{j=1}^n a_j^2}$$

The expression for d_i^* can be simplified further to give

$$d_i^* = \frac{\|\{M_{U_+} \circ \nabla_U f\}(i, :)\|^2}{\|\{M_{U_+} \circ \nabla_U f\}(i, :) V^{(k)T}\|^2} \quad (3.14)$$

3.4.2.2 Updating V

We follow a similar routine to update V . First we update the negative entries of V as,

$$V_{ij} = V_{ij} + \rho_v \delta_v, \quad \forall (i, j) \in I_- , \quad (3.15)$$

The non-negative entries in the rows of V are updated independently. The updates for all the non-negative entries in the i^{th} row can be computed as

$$V_{ij} = (V_{ij} - t_i^* (\nabla_V f)_{ij})_+, \quad \forall (i, j) \in I_+, \quad (3.16)$$

where t_i^* is the optimal step size, and $\nabla_V f = (UV^T - X)^T U$. The optimal step size computation is similar as before and is given by

$$t_i^* = \frac{\|\{M_{V_+} \circ \nabla_V f\}(i, :)\|^2}{\|\{M_{V_+} \circ \nabla_V f\}(i, :) U^{(k+1)T}\|^2} \quad (3.17)$$

The pseudocode for updating the nonnegative entries is given below:

Algorithm 2 Projected Block Coordinate Descent (PBCD)

- 1: **Input:** $X, U, V, \epsilon, \text{max_iter}, M_{U_+}, M_E$
 - 2: $\nabla_{U_{\text{init}}} f \leftarrow M_{U_+} \circ ((M_E \circ (UV^T - X))V)$
 - 3: $n \leftarrow \text{number of rows in } U$
 - 4: **Repeat until** $\text{projected_grad} < \epsilon \times \|\nabla_{U_{\text{init}}} f\|_F$ or max_iter :
 - 5: **for** $i = 1..n$
 - 6: $d_i^* \leftarrow \text{Using Eq. 3.14}$
 - 7: $U_{i,:} \leftarrow \text{Using Eq. 3.12}$
 - 8: $\text{projected_grad} \leftarrow \|M_{U_+} \circ ((M_E \circ (UV^T - X))V)\|_F$
 - 9: **Output:** U
-

We refer to this attaining feasibility stage as the ascent stage because it leads to a reconstruction error which is higher than the one attained by the rotated SVD factors.

3.4.3 Descent to the local minima

The ascent stage described in Section 3.4.2 returns factors that are very close to a local minima and any standard descent algorithm will move the feasible factors to a local minima of the NMF problem. The PBCD algorithm outlined in the previous section is sufficient to move the feasible factors to a local minima in a reasonable time complexity but we use hierarchical alternating least squares (HALS) for the descent stage. The descent stage terminates once the factor matrices U and V converge to a local minima. The convergence to a local minima is verified by checking for the KKT optimality conditions of (3.1) given below:.

- $\nabla W, W \geq 0$
- $\nabla W \odot W = 0$

where $W = \begin{bmatrix} U \\ V \end{bmatrix}$. Having defined the three main blocks of our exterior NMF algorithm (eNMF), we are now ready to provide the pseudocode for eNMF

Algorithm 3 eNMF

- 1: **Input:** $X, r, \rho, \epsilon, \text{max_iter}$
 - 2: $U^*, V^* \leftarrow SVD(X, r)$
 - 3: $(U^*R, V^*R) \leftarrow \text{Rotation}(U^*, V^*, R^0, Y^0, \rho)$
 - 4: $(U, V) \leftarrow (U^*R, V^*R)$
 - 5: **Repeat until U,V non negative:**
 - 6: Update negative elements in U using Eq. 3.11
 - 7: Compute mask M_{U+}
 - 8: $U \leftarrow PBCD(X, U, V, \epsilon, \text{max_iter}, M_{U+}, \mathbf{1})$
 - 9: Update negative elements in V using Eq. 3.15
 - 10: Compute mask M_{V+}
 - 11: $V \leftarrow PBCD(X^T, V, U, \epsilon, \text{max_iter}, M_{V+}, \mathbf{1})$
 - 12: $U, V \leftarrow HALS(X, U, V)$
 - 13: **Output:** U, V
 - 14: **Note:** $\mathbf{1}$ is the matrix of all 1's
-

3.5 Exterior NMC algorithm

The eNMF algorithm is extended to the case where the data matrix X has missing entries. We compute a local minimum of the unconstrained problem

$$\min_{U, V} \hat{f}(U, V) = \frac{1}{2} \|M_E \circ (X - UV^T)\|_F^2,$$

where M_E is the binary mask over the unknown entries of X , using an alternating least squares algorithm (softImpute-ALS) proposed in [75]. Once we obtain a local minimum, then we follow the same sequence (i.e. rotation of (U, V) closest to the positive orthant,

followed by an exterior penalty method) to obtain a solution to the nonnegative matrix completion problem. The gradient computations in the exterior penalty method are updated accordingly: $\nabla_U \hat{f} = (M_E \circ (UV^T - X))V$, $\nabla_V \hat{f} = (M_E^T \circ (UV^T - X)^T)U$. The pseudocode for eNMC is given below:

Algorithm 4 eNMC

- 1: **Input:** $X, r, M_E, \epsilon, \text{max_iter}$
 - 2: $U_0, V_0 \leftarrow \text{softImpute}(X, r, M_E)$
 - 3: $(U_0 R, V_0 R) \leftarrow \text{Rotation}(U_0, V_0, R^0, Y^0, \rho)$
 - 4: $(U, V) \leftarrow (U_0 R, V_0 R)$
 - 5: **Repeat until convergence:**
 - 6: Update negative elements in U using Eq. 3.11
 - 7: Compute mask M_{U+}
 - 8: $U \leftarrow \text{PBCD}(X, U, V, \epsilon, \text{max_iter}, M_{U+}, M_E)$
 - 9: Update negative elements in V using Eq. 3.15
 - 10: Compute mask M_{V+}
 - 11: $V \leftarrow \text{PBCD}(X^T, V, U, \epsilon, \text{max_iter}, M_{V+}, M_E^T)$
 - 12: **Output:** U, V
-

3.6 Experiments and results

In this section, we run experiments on synthetic and real datasets and demonstrate the superiority of the proposed algorithm in terms of reconstruction error and time complexity.

3.6.1 Datasets and algorithms

Exact factorization dataset: Although we developed an algorithm for the *NMF* problem, we would also like to evaluate the performance of eNMF when applied to the *exact NMF* problem. For this evaluation process, we created a synthetic dataset that admits an exact

factorization. The dataset was created by following the procedure: **i)** generate the entries of the ground truth factor matrices U and V by sampling randomly from $Uniform(0, 1)$, **ii)** a binary mask with sparsity s is applied to the ground truth factors and **iii)** generate the data matrix using $X = UV^T$.

Synthetic dataset (*Dense and High Dimensional X*): It is standard in the NMF literature to use randomly generated matrices to compare the relative performance of algorithms. We generated X as follows: **i)** generate the entries of two matrices $W \in \mathbb{R}^{5000 \times 1000}$, and $H \in \mathbb{R}^{1000 \times 5000}$ by sampling randomly from Uniform $(0, 1)$, **ii)** generate the entries of $N \in \mathbb{R}^{5000 \times 5000}$ by sampling randomly from $\mathcal{N}(0, \tau^2)$, where τ controls the signal to noise ratio (SNR) level, and **iii)** generate the data matrix using $X = WH + N$. The dense and high dimensional X allows one to compare both the scalability and accuracy of different algorithms at various SNR levels.

Verb dataset (*Highly Sparse and Moderate Dimensional X*): This dataset is comprised of entity-verb relationships extracted from a corpus of text data [76], where $X(i, j)$ is the number of times an entity i appears as the object of verb j ; $X \in \mathbb{R}^{282 \times 1528}$ and has 97.3% sparsity.

Face dataset (*Highly skewed and High Dimensional X*): The data matrix with shape 32256×64 contains the pixel values of a person’s image under various levels of illumination [77]. Part based decomposition of images is widely used in image processing and sparse factor matrices help to extract the various parts of an image like nose, eyes, etc. This dataset is useful for comparing the trade offs between reconstruction error and sparsity of the factor matrices obtained from different algorithms.

Movielens dataset (*A popular dataset with 95.5% Missing entries in X*): This dataset contains 1 million ratings by 6040 users on 3706 movies [78]. The rating matrix, $X \in \mathbb{R}^{6040 \times 3706}$, has 95.5% missing (user, rating) entries. The quality of recommendation systems are often benchmarked based on their performance on this dataset.

Six well-known *NMF* algorithms were used for performance comparison: accelerated

multiplicative algorithm (Mult) [63], gradient based algorithm (Grad-Mult) [64], alternating non-negative least squares (ALS) [65], alternating optimization with ADMM (AO-ADMM) [66], hierarchical alternating least squares (HALS) [68], and alternating direction method of multipliers (NMF_ADMM) [79]. However, in [79] a *nested ADMM* approach (where subproblems are themselves solved using ADMM) was used and it had difficulty converging. To alleviate this, we introduced a formulation

$$\begin{aligned} & \underset{U, V, A, B}{\text{minimize}} \quad \|X - AB^T\|_F^2 + I_+(A) + I_+(B) \\ & \text{subject to} \quad U = A, \quad V = B, \end{aligned} \tag{3.18}$$

where I_+ is the indicator function for element-wise nonnegative matrix, and solved (3.18) using ADMM [74] by alternately minimizing the blocks (U, B) and (V, A) . The subproblems in this ADMM formulation turned out to be simple non-negative least squares problems, which can be solved efficiently. This NMF_ADMM algorithm performed much better than the one proposed in [79].

The performance of eNMC algorithm was benchmarked against three widely used NMC algorithms: multiplicative updates (Mult)[80], sequential coordinate descent (SCD) [80], and alternating direction method (ADM) [81].

3.6.2 Reconstruction error and run time comparison

We observed an interesting phenomenon: if we run the competing algorithms for a long enough time (often for days) then they achieve a similar reconstruction error as eNMF (Figure 3.2).

Therefore, for a meaningful comparison of the relative performance of the algorithms we perform two experiments: (i) **Comparison of equal-time constrained reconstruction errors** (see Figure 3.5, Table 3.1): For any dataset and choice of r , we first run eNMF until it reaches a local minimum (verified by checking for the KKT conditions, see Table 3.4), and obtain the resulting reconstruction error and note the run time. Then we run the

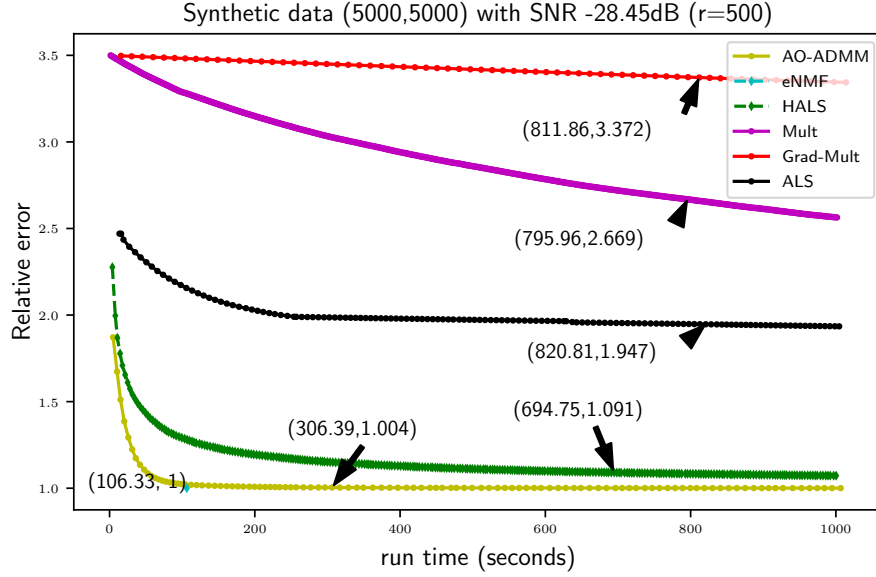


Figure 3.2: **Given enough run time, all the algorithms converge to local minima with similar reconstruction error** (across all the datasets we used): The above figure illustrates it for the Synthetic dataset with $n = m = 5000, r = 500$ and SNR of $-28dB$. Relative error = $\frac{\|X - U_{NMF}V_{NMF}^T\|_F}{\|X - U_{SVD}V_{SVD}^T\|_F}$ is plotted as a function of run time (secs). *eNMF achieves the minimum (i.e. unconstrained global minimum in this case) in 106 seconds, compared to hours of run time for some of the competing algorithms* Competitors like AO-ADMM have a fast convergence initially but slows down as it approaches the minimum, as shown by the long tail.

competing algorithms for a time that is at least as long as eNMF's (hence, *they have a better chance of lowering their reconstruction error*) by interrupting their running iterations right after the execution time exceeds the run time of eNMF. (ii) **Comparison of equal-error constrained time complexity** (see Figures 3.4 and 3.6): We set the reconstruction error to that of eNMF, and then run the rest of competing algorithms until they reach eNMF's reconstruction error.

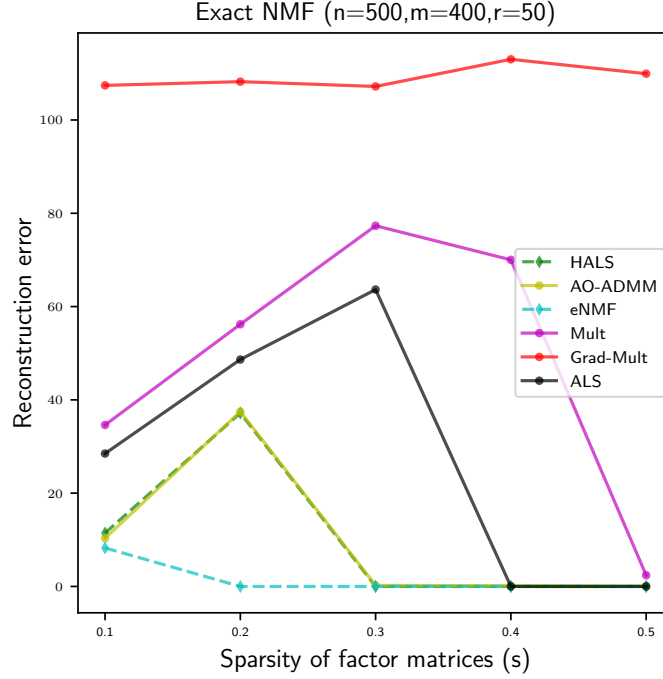


Figure 3.3: **Exact factorization:** eNMF achieves the global minimum for sparsity levels 0.2 through 0.5. Although AO-ADMM and HALS achieves the global minimum for sparsity levels 0.3 through 0.5, but its far away from the global minimum for sparsity level 0.2. For sparsity level 0.1, all the algorithms fail to converge to the global minimum with eNMF achieving the lowest reconstruction error. Although, competitors like AO-ADMM have a fast initial convergence (for sparsity ≥ 0.3) but slows down considerably as it approaches the global minimum (Please see Figure 8, where we had to run them for a long time to get to the global minimum). **Note:** NMF-ADMM had a very high reconstruction error and hence was dropped from the plot.

3.6.2.1 Performance of eNMF in exact factorization scenarios

The performance of eNMF, in terms of reconstruction error, was benchmarked against the six existing algorithms listed previously for various sparsity levels. As before, we do an equal-time constrained experiment by letting the competing algorithms run for at least as

long as eNMF. Figure 3.3 plots the reconstruction error as a function of the sparsity levels s . From the plot, it is clear that eNMF outperforms its competitors by converging to the exact factors for sparsity levels 0.2 through 0.5 and achieving the lowest reconstruction error for sparsity level 0.1.

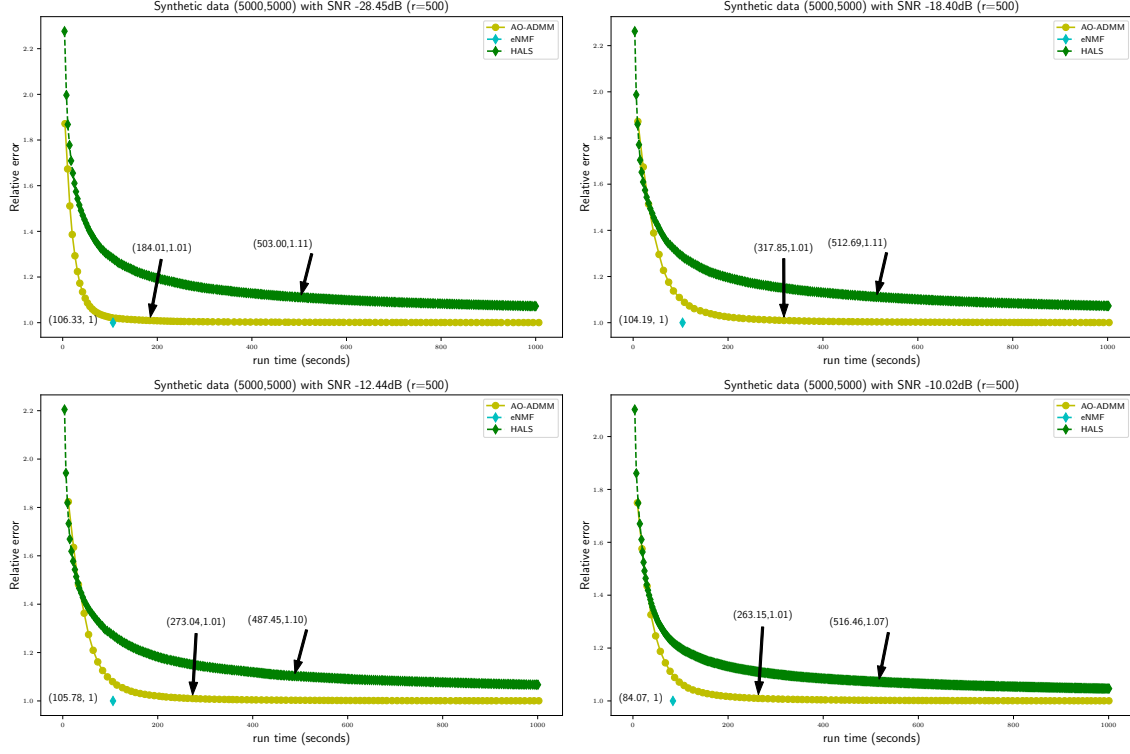


Figure 3.4: **Convergence speed (Synthetic datasets):** Relative error $= \frac{\|X - U_{\text{NMF}} V_{\text{NMF}}^T\|_F}{\|X - U_{\text{SVD}} V_{\text{SVD}}^T\|_F}$ is plotted as a function of run time (secs) for various SNR levels. *eNMF converges to the global minimum at the fastest rate for all SNR levels. As the noise level increases (lower SNR), the gap in the convergence rates of eNMF and AO-ADMM (its closest competitor) widens.* **Note:** For a zoomed in picture of the scenario, we have dropped the convergence curves of algorithms with a very slow convergence.

Dataset	r	eNMF	HALS	NMF-ADMM
Face	5	(18708.23 , 10^{-24} , 18708.23, 18708.23)	(18708.23, 10^{-24} , 18708.23, 18708.23)	(21264.2, 346.43, 21230.2, 21283.9)
	10	(12557.48 , 10^{-24} , 12557.48, 12557.48)	(12583.28, 166.49, 12557.48, 12589.74)	(17317.94, 6347671.75, 15033.09, 22161.31)
	15	(9796.13 , 10^{-2} , 9795.97, 9796.23)	(9856.23, 2533.55, 9795.97, 9916.6)	(18037.35, 33831261.3, 13829.63, 29479.96)
	20	(7916.49 , 10^{-4} , 7916.46, 7916.53)	(7976.74, 1421.07, 7916.46, 8014.39)	(29791.24, 51894946.9, 18871.56, 38436.03)
	25	(6371.27 , 321.47, 6359.1, 6396.98)	(6374.11, 348.67, 6359.64, 6405.24)	(27566.53, 54187136.4, 14638.89, 36614.78)
Verb	10	(14.39 , 10^{-5} , 14.39, 14.4)	(14.39, 10^{-5} , 14.39, 14.41)	(14.49, 10^{-4} , 14.46, 14.51)
	20	(13.58 , 10^{-6} , 13.58, 13.58)	(13.6, 10^{-4} , 13.58, 13.62)	(13.98, 10^{-3} , 13.94, 14.05)
	40	(12.32 , 10^{-5} , 12.31, 12.33)	(12.33, 10^{-4} , 12.31, 12.35)	(14.31, 0.1, 14.03, 15.17)
	80	(10.47 , 10^{-5} , 10.46, 10.49)	(10.56, 10^{-3} , 10.48, 10.69)	(17.68, 2.67, 15.53, 20.06)
	100	(9.68 , 10^{-5} , 9.66, 9.69)	(9.84, 0.01, 9.73, 10.08)	(18.48, 1.71, 16.15, 20.08)

Dataset	r	Grad-Mult	ALS	AO-ADMM
Face	5	(19510.79, 292669.04, 18903.9, 20286.99)	(18708.23, 10^{-24} , 18708.23, 18708.23)	(18708.23,0,0,0)
	10	(14051.48, 67444.72, 13782.29, 14503.72)	(12605.34, 2979.82, 12558.41, 12709.72)	(12580.3,0,0,0)
	15	(11529.95, 47398.35, 11227.89, 11893.61)	(9954.2, 768.18, 9922.16, 9992.92)	(9823.4,0,0,0)
	20	(10330.96, 38336.15, 10095.18, 10622.77)	(7975.07, 1146.44, 7926.35, 8032.71)	(7960.5,0,0,0)
	25	(8912.64, 295901.1, 7971.0, 9493.08)	(6465.98, 1294.33, 6398.2, 6506.12)	(6372.65,0,0,0)
Verb	10	(14.41, 10^{-5} , 14.39, 14.42)	(14.4, 10^{-4} , 14.39, 14.42)	(14.39,0,0,0)
	20	(13.61, 10^{-5} , 13.6, 13.62)	(13.6, 10^{-5} , 13.59, 13.6)	(13.59,0,0,0)
	40	(12.39, 10^{-4} , 12.37, 12.47)	(12.33, 10^{-4} , 12.32, 12.35)	(12.34,0,0,0)
	80	(10.68, 10^{-4} , 10.64, 10.71)	(10.49, 10^{-4} , 10.47, 10.51)	(10.53,0,0,0)
	100	(9.95, 10^{-4} , 9.91, 9.99)	(9.71, 10^{-4} , 9.69, 9.73)	(9.77,0,0,0)

Table 3.1: **Equal-time constrained NMF reconstruction errors (Real datasets)**: We tabulated the reconstruction error statistics (mean, variance, minimum, maximum) for 5 runs of each algorithm. From the table, it is evident that eNMF achieves the lowest mean reconstruction error with least variance for all latent dimensions and across both the datasets. **Note**: We didn't tabulate the error statistics for Mult because it has a very high reconstruction error.

3.6.2.2 Performance of eNMF on synthetic datasets

The performance of eNMF, in terms of reconstruction error and run time, was benchmarked against the six existing algorithms listed previously on synthetic dataset with four SNR levels. Figure 3.5 plots the equal-time constrained relative reconstruction error and as a function of the latent dimension for various SNR levels. From the plot, it is evident that eNMF achieves the unconstrained global minimum for all latent dimensions and across all the four SNR levels. For higher latent dimensions, the gap in relative error of eNMF and its closest competitor AO-ADMM widens for lower SNR levels (more noise). Figure 3.2 plots the relative error as a function of run time for various SNR levels. eNMF converges to the global minimum at the fastest rate for all SNR levels and as the noise level increases the gap

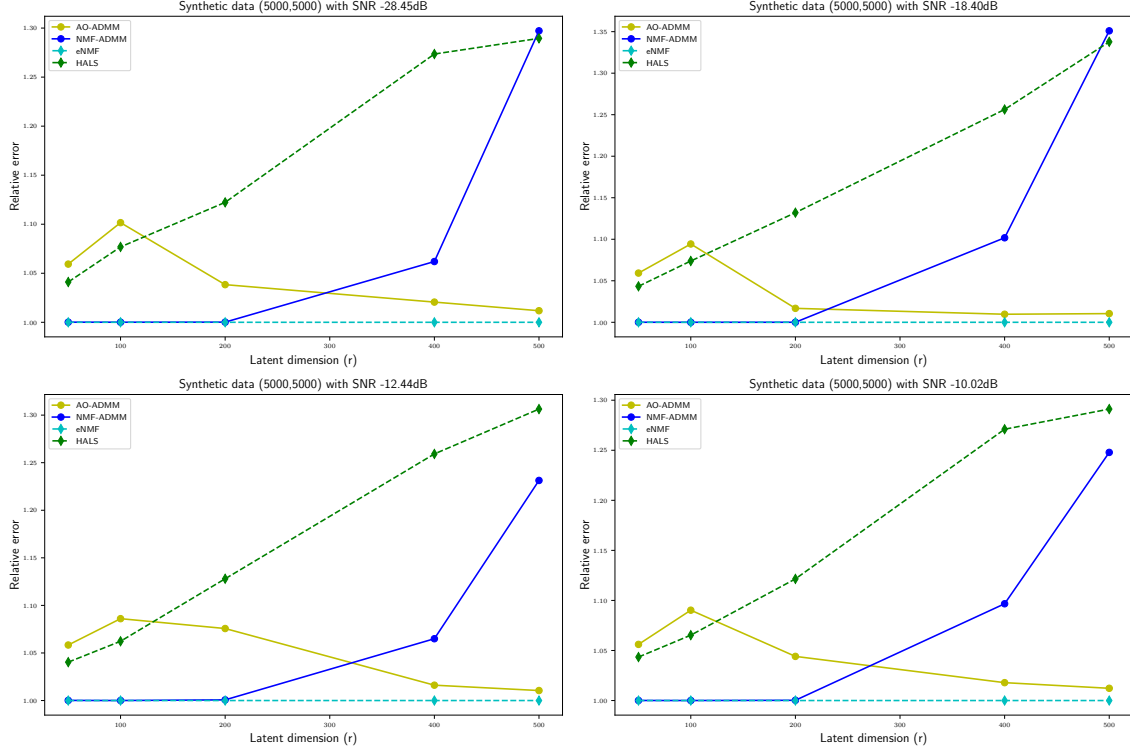


Figure 3.5: **Equal-time constrained NMF reconstruction errors (Synthetic datasets):**

Relative Reconstruction Error (RE) $\frac{\|X - U_{\text{NMF}} V_{\text{NMF}}^T\|_F}{\|X - U_{\text{SVD}} V_{\text{SVD}}^T\|_F}$ is plotted as a function of the number of latent dimensions (r) for various SNR levels. *eNMF achieves the unconstrained global minimum for all latent dimensions and across all the four SNR levels.* Among its competitors, only NMF-ADMM attains the global minimum for lower latent dimensions. For higher latent dimensions, the gap in relative error of eNMF and its closest competitor AO-ADMM widens for lower SNR levels (more noise). **Note:** For a zoomed in picture of the scenario, we have dropped the curves corresponding to the algorithms with a high relative error.

in the convergence rates of eNMF and its closest competitor widens.

3.6.2.3 Performance of eNMF on real datasets

The performance of eNMF, in terms of reconstruction error and run time, was benchmarked against the six existing algorithms listed previously on two real datasets: **Verb** and **Face**.

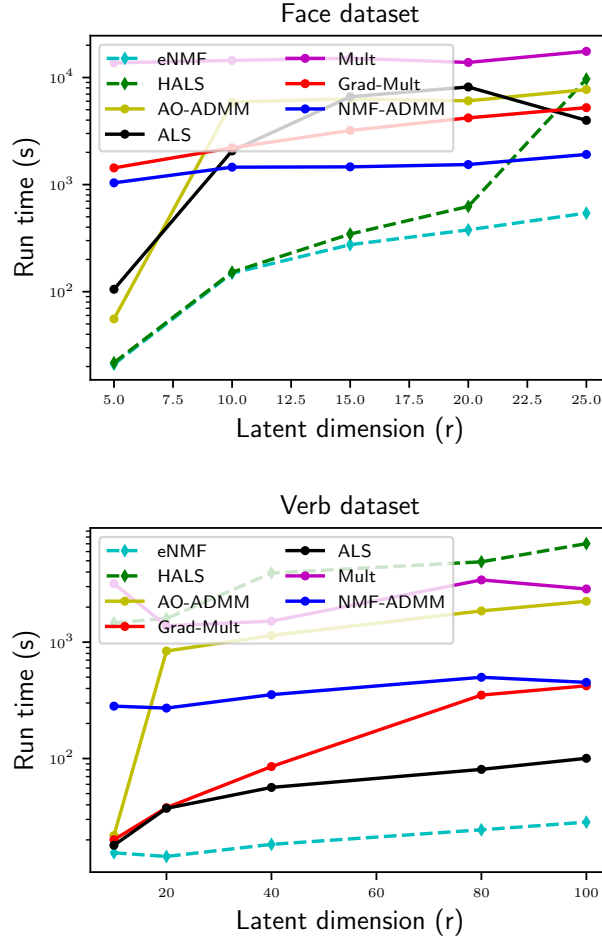


Figure 3.6: **Equal-error constrained time complexity (Real datasets)**: Run time is plotted on a log scale as a function of the number of latent dimensions (r). *eNMF performs significantly better than in its competitors in terms of time complexity with the margin widening for higher latent dimensions.*

Table. 3.1 reports the reconstruction error statistics (over 5 runs) as a function of the latent dimension. From the table, it is clear that eNMF achieves the lowest mean reconstruction error with least variance for all latent dimensions and across both the datasets. Figure 3.6 plots the equal-error constrained time complexity as a function of the latent dimensions. eNMF attains the local minima fastest for all latent dimensions with the gap in convergence

rates of eNMF and its closest competitor widening for higher latent dimensions.

3.6.2.4 Performance of eNMC on movielens dataset

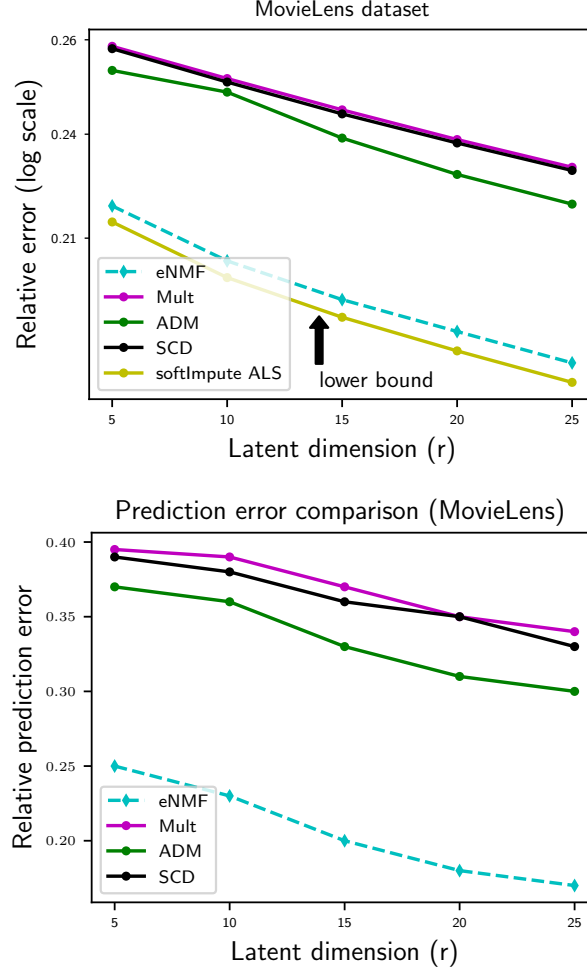


Figure 3.7: **Equal-time constrained NMC errors:** Relative reconstruction error (RE) $\frac{\|P_{\Omega}(X-UV^T)\|_F}{\|P_{\Omega}X\|_F}$ and Relative prediction error $\frac{\|X_{test}-UV^T\|_F}{\|X_{test}\|_F}$ is plotted as a function of the number of latent dimensions (r). *eNMC achieves the lowest reconstruction and prediction error for all latent dimensions.*

NMC is used to predict the ratings that an user would give on an item. The predicted ratings are then used to make recommendations to the user. Thus, the accuracy of the pre-

dicted ratings plays a crucial role in recommendation engines. We evaluated the performance of eNMC in terms of the prediction error: we trained the eNMC algorithm using 80% ratings of the movielens dataset and then used the factors to predict 20% ratings of the movielens dataset. Figure 3.7 plots the equal-time constrained relative reconstruction and prediction error as a function of the latent dimensions. It is evident from the figure that eNMC has a significant advantage over its competitors for all latent dimensions.

3.6.3 Structural analysis of minima

In the exact factorization scenario, the competing algorithms converged to the global minimum and attained a reconstruction error of zero for various sparsity levels. We wanted to explore the relationships between the factors obtained from various algorithms at the global minimum. The most intuitive relationship to check for is permutation. If A and B are two NMF algorithms, then there exists a diagonal scaling matrix Σ and a permutation matrix P such that

$$\begin{bmatrix} U_A \\ V_A \end{bmatrix} = \begin{bmatrix} U_B \Sigma P \\ V_B \Sigma^{-1} P \end{bmatrix}. \quad (3.19)$$

We use two steps to determine if the factor matrices U_A and U_B are permutations of each other. In the first step, we find the candidate mappings between the columns of U_A and U_B . For each column of U_A we repeat the following:

- Compute the cosine similarity between the column of U_A and all columns of U_B
- If the column of U_B with the highest cosine similarity has a similarity value greater than a certain threshold then we consider it as a candidate mapping and store the corresponding indices of the column of U_A and U_B in an array $U_{A_{Candidates}}$ and $U_{B_{Candidates}}$ respectively

In the second step we validate the candidate mappings. For the validation process we only consider the columns of U_A and U_B that are in the array $U_{A_{Candidates}}$ and $U_{B_{Candidates}}$. We

create the submatrices consisting of those columns and denote them as $U_{A_{Sub}}$ and $U_{B_{Sub}}$. Then we repeat the following steps for all columns of $U_{A_{Sub}}$:

- Compute the cosine similarity between the column of $U_{A_{Sub}}[:, i]$ and all columns of $U_{A_{Sub}}$ and store the sorted and rounded to one decimal place result in a vector x_i
- Compute the cosine similarity between $U_{A_{Sub}}[:, i]$ and all columns of $U_{B_{Sub}}$ and store the sorted and rounded to one decimal place result in a vector y_i
- Compute the fraction of the entries that are same in x_i and y_i and store it as a scalar (rounded to one decimal place) z_i
- If z_i is greater than a certain threshold then the candidate mapping is validated and we say that i^{th} column of U_A has converged to a unique column of U_B and the mapping is given in the array $U_{A_{Candidates}}$ and $U_{B_{Candidates}}$

Once we have the permutation matrix mapping U_A to U_B , we check that the same permutation matrix P maps V_A to V_B . In Figure 3.8, we plotted the percentage of the columns of the factor matrices that have converged to a unique column of ground truth factors and the eNMF factors as a function of the sparsity levels. It can be observed from the figure that when the algorithms converge to the global minimum then the factor matrices obtained from different algorithms are permuted versions of both the ground truth factors and the eNMF factors.

The above observation in the context of exact factorization motivated us to check for permutations in the context of *NMF* problems. eNMF attained the unconstrained global minimum on the synthetic dataset for both latent dimensions ($r = 100, 500$) and for all 4 SNR levels. Since the eNMF attained the unconstrained global minimum so the manifold of orthogonal rotations intersect the positive orthant. In this special case, the eNMF solution (U_{eNMF}, V_{eNMF}) and the solution from other NMF algorithms (U_{alg}, V_{alg}) are related by the following generalized rotation:

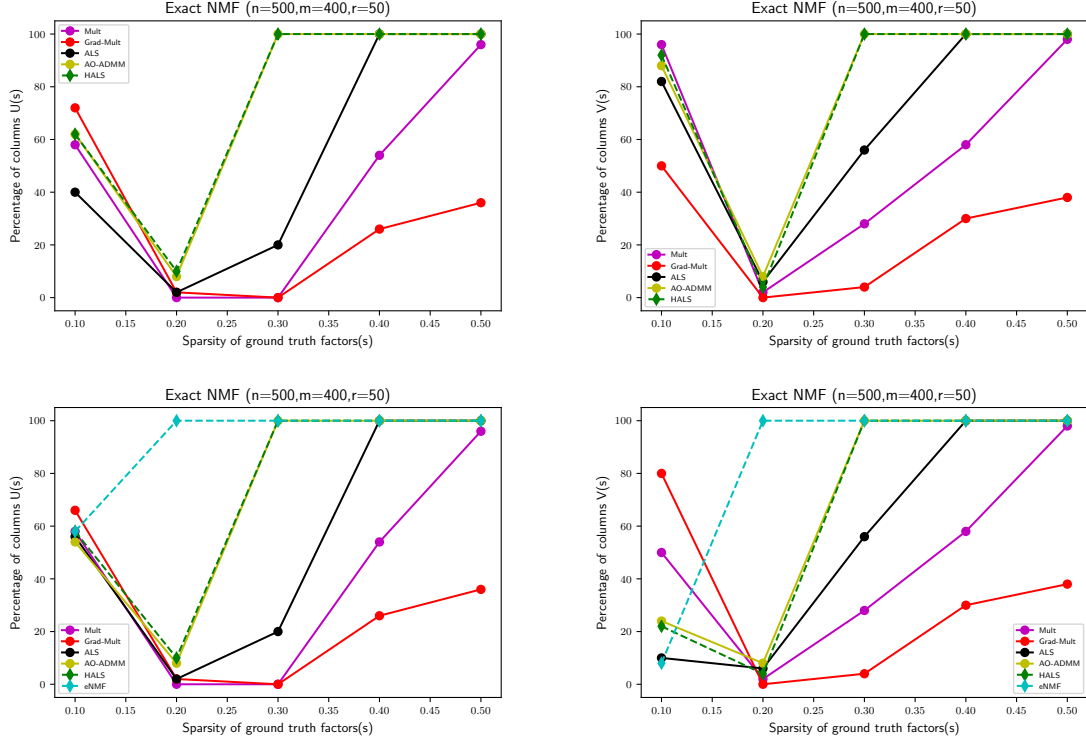


Figure 3.8: **Factor matrices are permutation of each other at the global minimum (Exact factorization scenario):** **Top figure** Percentage of the columns of U_B (where B is any competing algorithm) that have converged to an unique column of ground truth factor U_{gt} is plotted as a function of the sparsity levels (the percentage comparison between V_B and V_{gt} is also shown). **Bottom figure** Percentage of the columns of U_B (where B is any competing algorithm) that have converged to an unique column of U_A (where A is eNMF) is plotted as a function of the sparsity levels (the percentage comparison between V_B and V_A is also shown).

$$U_{alg} = U_{eNMF} R_1 \Lambda R_2^T$$

$$V_{alg} = V_{eNMF} R_1 \Lambda^{-1} R_2^T$$

where R_1 and R_2 are rotation matrices and Λ is a diagonal scaling matrix. Table II validates

r	SNR	eNMF vs AO-ADMM	eNMF vs HALS	eNMF vs ALS
100	0.00143	$(10^{-10}, 10^{-10})$	$(N/A, N/A)$	$(N/A, N/A)$
	0.01447	$(10^{-10}, 10^{-10})$	$(N/A, N/A)$	$(N/A, N/A)$
	0.05705	$(10^{-10}, 10^{-10})$	$(N/A, N/A)$	$(N/A, N/A)$
	0.09950	$(10^{-10}, 10^{-9})$	$(N/A, N/A)$	$(N/A, N/A)$
500	0.00143	$(10^{-5}, 10^{-4})$	$(N/A, N/A)$	$(N/A, N/A)$
	0.01447	$(10^{-9}, 10^{-9})$	$(N/A, N/A)$	$(N/A, N/A)$
	0.05705	$(10^{-7}, 10^{-7})$	$(N/A, N/A)$	$(N/A, N/A)$
	0.09950	$(10^{-5}, 10^{-5})$	$(N/A, N/A)$	$(N/A, N/A)$

Table 3.2: **Factor matrices are generalized rotations of each other at the unconstrained global minimum (Synthetic dataset):** We tabulated the tuple (δ_u, δ_v) , where $\delta_u = \|U_{alg} - U_{eNMF}R_1\Lambda R_2^T\|_F$ and $\delta_v = \|V_{alg} - V_{eNMF}R_1\Lambda^{-1}R_2^T\|_F$. The factors from the competing algorithms are related to the eNMF factors by a generalized rotation ($\delta_u \approx 0$, $\delta_v \approx 0$).

the relationship using the synthetic dataset of latent dimensions 100 and 500 and for 4 different SNR levels. Please see the paper appendix for the solution to the above system of equations.

Having verified the observation on the synthetic dataset, we wanted to validate it on the real datasets. We checked for permutations between the factors obtained from equal-error constrained experiments on verb and face dataset. That is, we obtain an eNMF solution, and then let the other algorithms run until the error either becomes the same as that of eNMF or their reconstruction error doesn't improve for a long enough time. For permutation checking purpose, we fix the algorithm A to be eNMF and then compare them with the factors obtained from the competing algorithms. In Table. 3.3 we compared the percentage of the columns of the factor matrices that have converged to an unique column of the eNMF factors along with the error ratio as a function of the latent dimension for the Verb and Face dataset. From the table , we can make the following claim: the factors from the NMF algorithms converge to a pair of factor matrices that are unique modulo permutation and scaling only at the local minima (error ratio = 1), and even a small deviation from the minima leads to

Permutation results: (% of converged U columns, % of converged V columns, reconstruction error difference, equivalence of local minima)

Dataset	r	eNMF vs HALS	eNMF vs AO-ADMM	eNMF vs ALS
Face	5	(100,100,0,E)	(100,100,0,E)	(100,100,0,E)
	10	(100,100,0,E)	(10,10,32.3,NE)	(100,100,0,E)
	15	(67,47,0.25,TE)	(100,100,0,E)	(67,47,0.25,TE)
	20	(95,95,0.02,TE)	(25,35,78.8,TE)	(25,10,33.1,TE)
	25	(32,32,63.6,TE)	(52,12,19.2,TE)	(24,24,67.5,TE)
Verb	10	(70,70,0.01,TE)	(100,100,0,E)	(80,70,0.0096,TE)
	20	(95,95,0.004,TE)	(80,80,0.01,TE)	(65,65,0.016,TE)
	40	(87.5,87.5,-0.0004,TE)	(82.5,82.5,0.01,TE)	(80,77.5,0.009,TE)
	80	(71.25,71.25,0.069,TE)	(81.25,81.25,0.01,TE)	(80,78.75,0.02,TE)
	100	(69,72,-0.00003,TE)	(73,76,0.01,TE)	(70,73,2.79,TE)

Table 3.3: **Factor matrices are permutation of each other only at local minima (Real datasets):** We tabulated the triplet (u_B, v_B, e_d) , where u_B is the percentage of the columns of U_B (where B is any competing algorithm) that have converged to a unique column of U_{eNMF} and v_B is the percentage of the columns of V_B (where B is any competing algorithm) that have converged to a unique column of V_{eNMF} . e_d is the error difference defined as $e_d = \|X - U_B V_B^T\|_F - \|X - U_{eNMF} V_{eNMF}^T\|_F$. (E: equivalent local minima, TE: tending towards equivalent local minima, NE: non-equivalent local minima). From the table it can be observed that the factors from the competing algorithms are approaching to a permuted version of the eNMF factors ($u_B \approx 100, v_B \approx 100$) in an asymptotic manner due to the slow convergence of the competing algorithms. **Note:** The competing algorithms were run until they either reached eNMF's reconstruction error or their reconstruction error didn't improve much for 40,000 iterations. For the exact run time, please see Figure 3.6.

a large reduction in the number of columns that have converged to a unique column of the eNMF.

KKT optimality conditions: $(\max(\nabla W \cdot W), \sum_{i,j}(\nabla W \cdot W)_{ij}, \min(W))$					
Dataset	r	eNMF	HALS	AO-ADMM	ALS
Face	5	$(10^{-6}, 10^{-5}, 10^{-22})$	$(0.06, 5.8, 10^{-22})$	$(0.087, 5.2, 0)$	$(10^{-4}, 4.7, 0)$
	10	$(10^{-7}, 10^{-5}, 10^{-21})$	$(0.01, 1.1, 10^{-21})$	$(10^{-7}, 10^{-6}, 0)$	$(0.0001, 1.1, 0)$
	15	$(10^{-6}, 10^{-3}, 10^{-21})$	$(3.38, 220.1, 10^{-21})$	$(10^{-7}, 10^{-6}, 0)$	$(0.06, 158.4, 0)$
	20	$(0.34, 1.52, 10^{-21})$	$(33.3, 1548.98, 10^{-21})$	$(0.02, 0.54, 0)$	$(10^{-7}, 10^{-5}, 0)$
	25	$(0.98, 1.72, 10^{-21})$	$(0.0009, 0.02, 10^{-21})$	$(185.6, 9675, 0)$	$(10^{-7}, 10^{-5}, 0)$
Verb	10	$(10^{-16}, 10^{-14}, 0)$	$(10^{-12}, 10^{-10}, 10^{-13})$	$(10^{-9}, 10^{-7}, 0)$	$(10^{-10}, 10^{-8}, 0)$
	20	$(10^{-15}, 10^{-14}, 0)$	$(10^{-12}, 10^{-10}, 10^{-13})$	$(10^{-15}, 10^{-14}, 0)$	$(10^{-10}, 10^{-9}, 0)$
	40	$(10^{-15}, 10^{-14}, 0)$	$(0.02, 0.2, 10^{-13})$	$(10^{-15}, 10^{-14}, 0)$	$(10^{-10}, 10^{-8}, 0)$
	80	$(10^{-9}, 10^{-7}, 0)$	$(10^{-12}, 10^{-9}, 10^{-13})$	$(10^{-15}, 10^{-13}, 0)$	$(0.0001, 0.008, 0)$
	100	$(10^{-6}, 10^{-5}, 0)$	$(0.006, 0.16, 10^{-13})$	$(10^{-15}, 10^{-13}, 0)$	$(0.83, 42.3, 0)$

Table 3.4: **eNMF satisfies the KKT conditions for all latent dimensions and across both the datasets:** We tabulated the triplet $(\delta_W, \sigma_W, \Delta_W)$, where $\delta_W = \max(\nabla W \cdot W)$, $\sigma_W = \sum_{i,j}(\nabla W \cdot W)_{ij}$ and $\Delta_W = \min(W)$. As before W is defined as the stacked NMF factors. eNMF satisfies the KKT conditions $(\delta_W \approx 0, \sigma_W \approx 0, \Delta_W \geq 0)$ for both the datasets and all the latent dimensions.

3.7 Concluding remarks

We developed a general framework for the *NMF* problem that performs significantly better than the existing ones: From Figure 3.5 and Figure 3.4, it is evident that, eNMF’s performance is significantly better than the existing algorithms on both the reconstruction error and run time metrics with the performance gap widening for lower SNR levels (more noise). Also, from Figure 3.4 it can be observed that eNMF’s closest competitors (AO-ADMM and HALS) has a very slow convergence as it approaches the local minimum, depicted by its very long tail. In the context of real datasets, eNMF achieves the lowest reconstruction error (Table 3.1) but the performance gap is lower. The long tail property of the competing algorithms gives eNMF a significant advantage in terms of run time metrics as shown in Figure 3.6.

CHAPTER 4

Mode Separation with Weak Supervision: Discovering Epileptogenic HFOs in the Electroencephalogram (EEG) of Epilepsy Patients

4.1 Introduction

More than one-third of individuals with epilepsy are medication-resistant, making them potential surgical candidates [82]. Currently, surgery is primarily guided by neuroimaging and neurophysiology (interictal spikes and seizure onset zone). However, the seizure-freedom rate of surgery is suboptimal, ranging from 50 to 85% [83, 84, 85, 86]. Identifying a biomarker that can accurately delineate the spatial extent of the epileptogenic zone (EZ: brain areas responsible for generating seizures) will be groundbreaking. Human and animal studies of epilepsy have suggested that intracranially-recorded interictal high-frequency oscillations (HFOs) on electroencephalogram (EEG) is a promising spatial neurophysiological biomarker of the epileptogenic zone. Many retrospective studies demonstrated that the removal of brain regions producing HFOs correlated with post-operative seizure-freedom. However, a recent prospective study [87] could not reproduce these results. One of the most challenging issues is the presence of HFOs generated in healthy brain regions (physiological HFOs), which means seizure-freedom may be achieved despite leaving behind some areas displaying HFOs [88, 89, 90]. In short, to utilize HFOs as a spatial biomarker to guide epilepsy surgery, one needs to establish a methodology to differentiate HFOs that are generated from the EZ (epileptogenic HFOs: eHFOs) and HFOs that are generated from other areas (non-

epileptogenic HFOs: non-eHFOs).

A hypothesis-driven approach to look for eHFOs is quite challenging in practice because one needs to consider numerous and yet-to-be-identified features. Visual classification of HFOs with or without spikes along with artificial removal is commonly performed, because HFOs with spike-wave discharges are considered representative of eHFOs. However, this task is time-consuming and exhibits poor inter-rater reliability. Semi-automated computational methods to evaluate HFO characteristics (frequency, amplitude, and duration) do not appear to be useful for differentiating between eHFOs and non-eHFOs. Fast ripples (250-500 Hz) might more specifically localize epileptogenic zones than ripples (80-250 Hz), but their detection rate is much lower than ripples [91, 92, 93]. Correcting the HFO detection rate with region-specific normative values seems a reasonable approach [94, 95] but this does not determine each HFO event as an eHFO or non-eHFO.

4.2 Related work

Therefore, automated computational methods, such as those developed in the fields of artificial intelligence, would ideally discover eHFOs, guided purely by large samples of HFOs and clinical outcomes. Once trained, an ideal model should work robustly for any future patient. Moreover, these automated models should be interpretable to enable clinical decisions with high confidence and guide further scientific explorations of biological mechanisms. Indeed, machine learning has been successfully applied to the problem of classifying HFOs based on a priori manual engineering of event-wise features, which includes: linear discriminant analysis [96], support vector machines, decision trees, and clustering [97]. More recently, the deep learning(DL) framework has been adopted, which directly works with raw data (avoiding any a priori feature engineering) and yields better performance in the field of neuroimaging. Leveraging DL’s revolutionary success in the field of computer vision using Convolutional Neural Networks (CNNs), prior studies [98, 99, 100] explored the use of CNNs in EEG

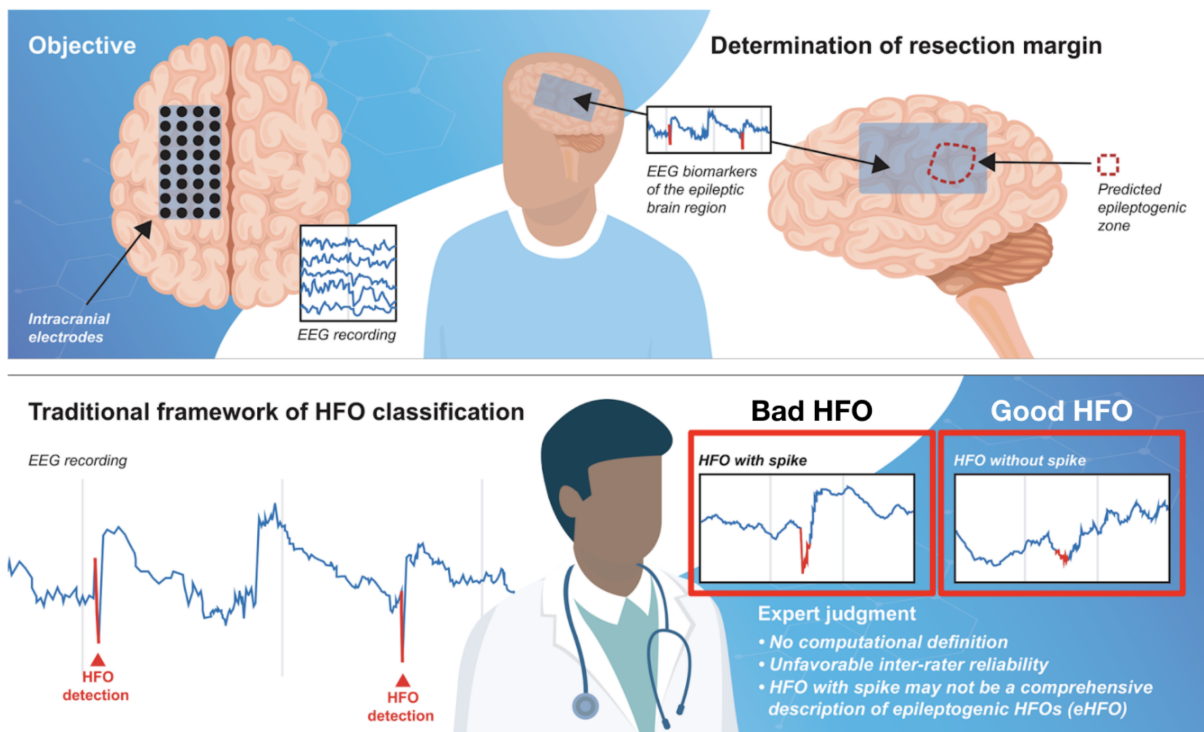


Figure 4.1: Upper: demonstration of our objective. Bottom: Traditional framework of HFO classification.

analysis, especially converting one-dimensional EEG signal into a two-dimensional image for CNNs input. The previous DL approaches conducted the HFO classification in a supervised manner, requiring human annotated labels which constrains the spectrum of usage of their methods, especially the needs of human expert labeling. In the context of medical image analysis, recent work [101] has shown that optimized model architectures and loss functions could mitigate data labeling errors, thus making the DL framework even more applicable.

4.3 Overview

The present study employed innovative analytic approaches to address several challenges expected in applying DL frameworks to the HFO classification task. As noted above, no direct observation of eHFOs is currently possible, making the most widely used supervised framework of DL impractical for our problem. Even if one were to solve this challenging problem, one still needs to make the DL models interpretable, which is yet another difficult task. In this study, we first proved that DL models could reliably emulate experts’ visual annotations in classifying the HFOs into artifacts, HFOs with spikes, or HFOs without spikes, without any a priori feature extractions. To mitigate potential labeling errors, we then generalized this approach to our central task of discovering eHFOs by replacing experts’ inputs with inexact weak labels implied by clinical outcomes and by using the “purification power” of DL to automatically distill eHFOs. Furthermore, (i) we proved the generalizability of this approach by using patient-wise cross-validation, implying a DL algorithm trained by EEG data from a large and diverse enough retrospective cohort is likely applicable to future patients; and (ii) we reverse engineered interpretable salient features of the DL-discovered eHFOs and showed that they aligned with current expert knowledge.

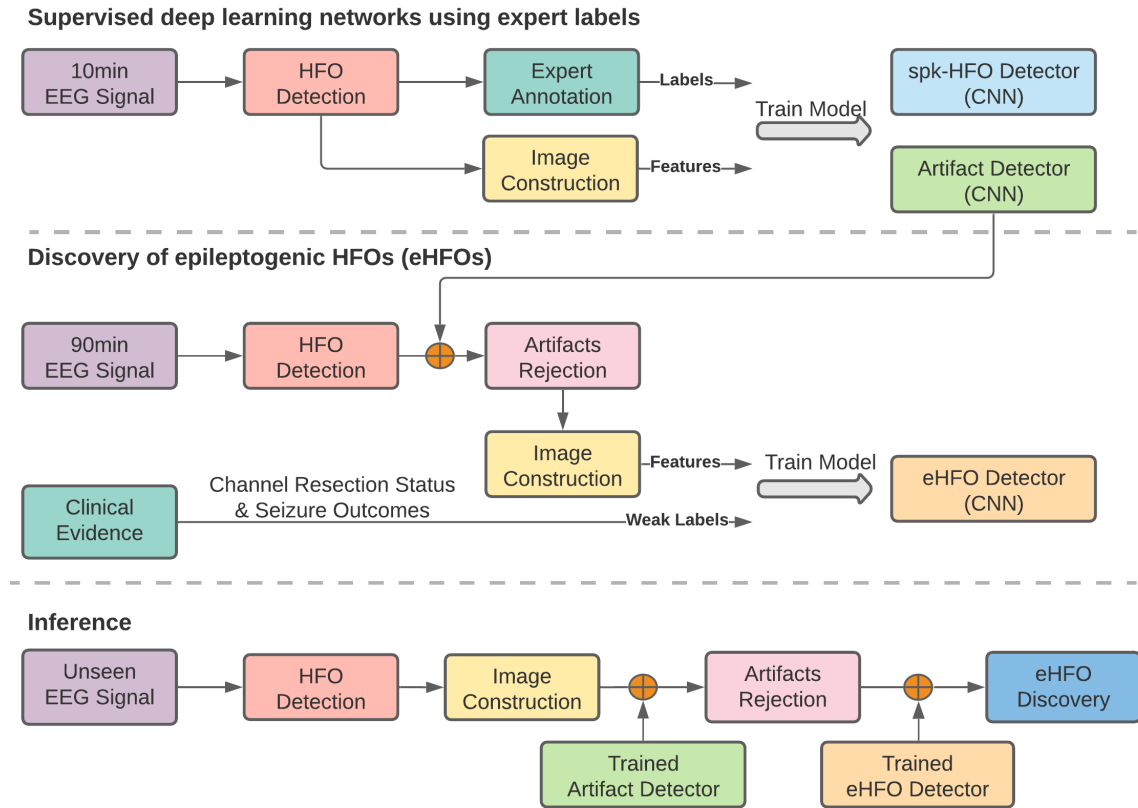


Figure 4.2: Processing workflow. Our study's overall data processing workflow is shown as a flowchart.

4.4 Method

4.4.1 Dataset

Patient cohort: This was a retrospective cohort study. Children (below age 21) with medically refractory epilepsy (typically with monthly or greater seizure frequency and failure of more than three first-line anti-seizure medications) who had intracranial electrodes implanted for the planning of epilepsy surgery with anticipated cortical resection with the Pediatric Epilepsy Program at UCLA were consecutively recruited between August 2016 and August 2018. **HFO detection:** Candidate interictal high frequency oscillations (HFOs) were identified by an automated short-term energy detector (STE) [102, 103]. This detector considers HFOs as oscillatory events with at least six peaks and a center frequency occurring between 80-500 Hz. The root mean square (RMS) threshold was set at five standard deviations (SD), and the peak threshold was set at three SD. The HFO events are segments of EEG signals with durations ranging from 60 to 200 ms (see SI for duration distribution). We referred to these detected events as candidate HFOs (c-HFOs).

Human expert classification of HFOs: A human expert (HN: board certified in clinical neurophysiology and epilepsy, with experience in HFO analysis) classified c-HFOs in each patient’s 10-minute EEG segments into three classes: HFOs with spikes (spk-HFO), HFOs without spikes (non-spk-HFO), and artifacts using RippleLabs graphic user interface, based on three images (unfiltered EEG tracing, filtered EEG tracing [80-500 Hz], and time-frequency plot). Artifacts are false-positive events, including ringing (filtering of sharp transients), as well as muscle and background fluctuations. Another expert with similar qualifications (SH) independently scored c-HFOs from two representative patients, and inter-rater reliability was examined using Cohen’s kappa statistics.

4.4.2 Feature preprocessing

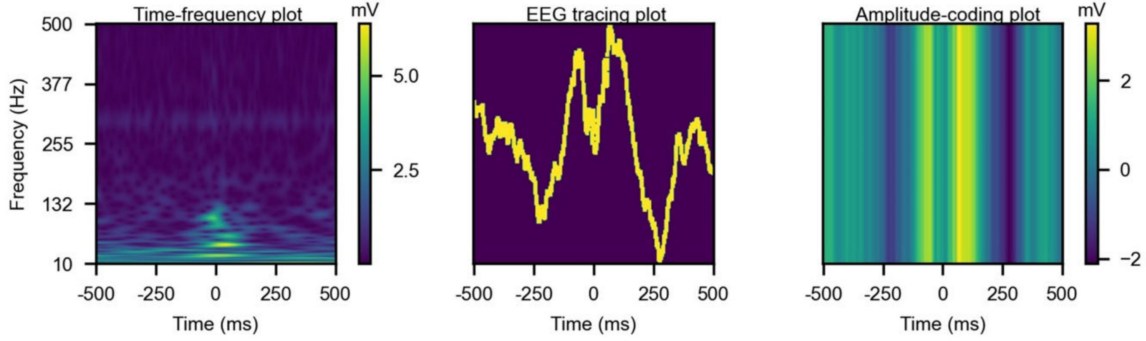


Figure 4.3: Network input images. The network input includes three images constructed from a one-second raw EEG segment with a detected HFO in the center (500ms before and after). Left: The time-frequency plot was generated by continuous Gabor Wavelets ranging from 10Hz to 500Hz. Middle: EEG tracing plot was generated on a 2000 x 2000 image by scaling the time-series signal into the 0 to 2000 range. Right: amplitude-coding plot contains the amplitude at every time point; a column of the image represented the signal’s actual value rescaled with a color gradient. These three images were resized into 224 x 224 in order to fit into the neural network.

Each c-HFO was represented by a one-second window, with the c-HFO was located at the center (0 ms), and including 500ms of EEG signal before and after. To utilize the power of CNN, we captured the time-frequency domain features as well as signal morphology information of the c-HFO window via three images in Figure 4.3. The time-frequency plot (scalogram) was generated by continuous Gabor Wavelets ranging from 10Hz to 500Hz. The EEG tracing plot was generated on a 2000 x 2000 image by scaling the time-series signal into the 0 to 2000 range to represent the EEG waveform’s morphology. The amplitude-coding plot was generated to represent the relative amplitude of the time-series signal: for every time point, the pixel intensity of a column of the image represented the signal’s raw value

at that time. These three images were resized into the standard size (224 x 224), serving as the input to the neural network.

4.4.3 Network architecture

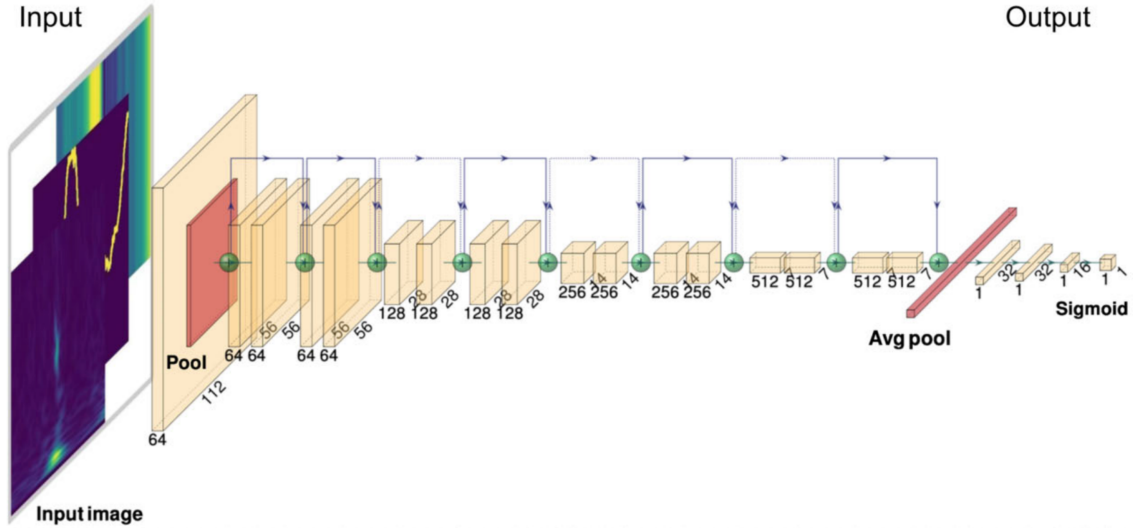


Figure 4.4: CNN architecture. The architecture of the model was adapted from Resnet-18. The last layer of the resnet18 was modified to be three fully-connected layers with LeakyReLU, BatchNorm, and 10 % dropout in between. The output of the model was fed into a sigmoid function to bound the output between 0 to 1, representing the probability of each task. For task 2 (HFOs with spikes vs. HFOs without spikes), the input consisted of the three images (time-frequency plot, EEG tracing plot, and amplitude-coding plot). Meanwhile, for task 1 (Real HFOs vs. artifacts), only the time-frequency plot information was found to be sufficient, and hence three same time-frequency plots were concatenated together as the input of the model.

Given the expert labeling with three labels (artifacts, spk-HFO, and non-spk-HFO), the training of the deep neural network (DNN) model was formulated as two binary classifica-

tion steps. Step1 (artifact detector): we differentiated between artifacts and “Real HFOs”, defined as the union of spk-HFOs and non-spk-HFOs. All “Real HFOs” were labeled as the positive samples and the artifacts were the negative samples. Step2 (spk-HFO detector) classified the “Real HFOs” into spk-HFO and non-spk-HFO; the spk-HFO were defined as positive samples, and the non-spk-HFO were defined as negative sample. See Figure 4.2 for the detailed demonstration of the processing workflow.

The artifact detector and spike detector’s architectures are identical and adapted from ResNet18 [104] with a modification in the last few layers to accommodate the binary classification tasks. Specifically, the last layer of the Resnet18 was modified to be three fully-connected layers with LeakyReLU, BatchNorm, and 10 % dropout in between. The output of the model was fed into a sigmoid function to bound the output between 0 to 1, representing the probability of each task. The input comprising three image channels and the architecture of the networks are shown in Figure 4.4. For the artifact detector, only the time-frequency information was used. Hence time-frequency plots were repeated three times and concatenated together as the input to the artifact detector. For the spk-HFO detector, concatenation of the three feature-representing images (time-frequency plot, EEG tracing plot, and amplitude-coding plot) served as input.

4.4.4 Label assignment for training

We generated a weakly labeled training set with the following assumptions. For those patients that became seizure-free after resection (9 patients), we assumed that all epileptogenic tissue was contained in the resection. Similarly, the preserved regions in post-surgery seizure-free patients could not contain epileptogenic tissue, leading us to assume that the HFOs in the preserved regions are non-epileptogenic. We thus labeled all of the HFOs from resected channels as 1 and all HFOs from preserved channels as 0. By doing such weak supervision, we introduced potential errors in the data labels. Specifically, (1) the epileptogenic zone may generate non-eHFOs in addition to eHFOs; (2) some tissue was resected based on anatomical

location (e.g. the channel was located within the same gyrus as the SOZ) and was likely not epileptogenic, leading to false-positive labels; and (3) in the non-resected region, while the majority of the HFOs were indeed non-epileptogenic, a few HFOs that are morphologically similar to epileptogenic HFOs could have been introduced by phenomena such as propagation. To address (1) and (3), we believed the automatic geometric purification property of the neural network could denoise the errors introduced by this mislabeling since the clinical outcomes constrain the number of mislabels to be few. To address (2), we introduced a weight term in the network’s loss function to reduce the noise introduced by the false positives, which is described in the following section. Note that for patients that were not seizure-free after surgery, the assumption that all epileptogenic tissue lay within the resected boundary did not hold. Thus, they were not used for any training. To maximize the training set size, we used 90 minutes of data from each patient for training. In order to ensure that our proposed framework generalizes across patients, a patient-specific model was designed for each patient without using any of its data. Thus, for a post-operative seizure-free patient, a patient-specific model was trained on data from other patients who became seizure-free after surgery. For post-operative non-seizure-free patients, a patient-specific model was trained on data from all post-operative seizure-free patients. Specifically, for training, we used 90 minutes of EEG data from 8 patients (if the target patient is post-surgery seizure-free) or 9 patients (if the target patient is not post-surgery seizure-free). All predictions (inference) used for the rest of the analysis were generated following this strategy.

4.4.5 The Deep Learning Architecture and Training for REVerse engineering (DLATREV)

The training details are as follows. We first used the artifact detector trained on all patients’ 10 min EEG data to filter out the potential artifacts in the c-HFOs from the 90 min EEG data. For the sake of convenience, any HFOs that passed this filtering, and hence are ”Real HFOs,” would simply be referred to as HFOs going forward. The total number of HFOs

differed considerably among the patients, and a data balancing process was required to balance the information introduced from each patient. For each patient, if the number of HFOs was smaller than a threshold of 2,500 (the median of the HFO distribution among patients), all of the HFOs were used. Otherwise, 2,500 events were uniformly sampled from that patient without replacement. There could potentially be many non-eHFOs in the non-seizure-onset but resected channels that were labeled incorrectly as eHFOs in our weak supervision process. For each input with a label of y and prediction of x , we introduced a weight term, w , in the Binary Cross Entropy (BCE) loss. (Loss = $w * \text{BCE Loss}$, where BCE Loss = $-[y \log(x) + (1 - y) \log(1 - x)]$). If $y = 0$ (the HFO is from a preserved channel), then $w = 1$. If $y = 1$ (the HFO is from a resected channel) and the HFO is from a SOZ channel, then also $w = 1$. However, if $y = 1$ and the HFO is not from a SOZ channel, we used a value of $w = 0.5$, which was found to be optimal for our dataset. For the model architecture and training, we used the same setup as the spike detector, except the CNN’s convolution layers were frozen to only act as feature extractors.

4.4.6 Training and performance analysis

There were two types of training conducted: patient-wise cross-validation and all-patients training. For patient-wise cross-validation, one patient was selected at a time as the test set, and the remaining patients were used for model training. All events were pooled across the rest of the patients, with 10% randomly sampled to serve as a validation set and the remaining 90% used for training. In all-patients training, five-fold cross-validation was conducted across the pooled data across the full patient cohort. For each fold, 20% of the dataset was selected as the test set, 70% was selected as the training set, and the remaining 10% was used for validation.

Since the optimization goal of both detectors is binary classification, we adopted binary cross-entropy as the loss function and the Adam optimizer with a learning rate of 0.0003. All of the training was conducted using 15 epochs (training iterations) and validation loss

was plotted with respect to the number of epochs completed. For the artifact detector, to improve generalization, we picked the model in the epoch that corresponds to the first local minima in the plot; this technique is also known as an early stopping regularization. For the spike detector, we directly picked the model corresponding to the global minima over 15 iterations, i.e., the lowest validation loss.

We calculated the precision, recall, and accuracy of the classification results. To measure the model performance on an unbalanced dataset like ours, we also calculated the F1-score. For the patient-wise cross-validation task, we averaged the performance statistics across patients. For the all-patients training task with five-fold cross-validation, we reported average model performance over cross-validation folds.

4.5 Experiments and results

4.5.1 Relationship between epileptogenic HFOs and HFOs with spikes

In order to understand the correspondence between the discovered eHFOs or non-eHFOs and common clinical knowledge about such HFOs (i.e. HFOs with and without spikes), we let the model classify each HFO event on the 10 minutes annotated data using DLATREV. We quantitatively analyzed the correlation of classified eHFOs and non-eHFOs with their corresponding spk-HFO annotation using the chi-square test. We noted 71.1% (4573/6430) of the eHFOs were HFOs with spikes, and 73.7% (2739/3721) of non-eHFOs were HFOs without spikes ($p < 0.0001$, a chi-square test).

4.5.2 Spatial distribution of epileptogenic and non-epileptogenic HFOs

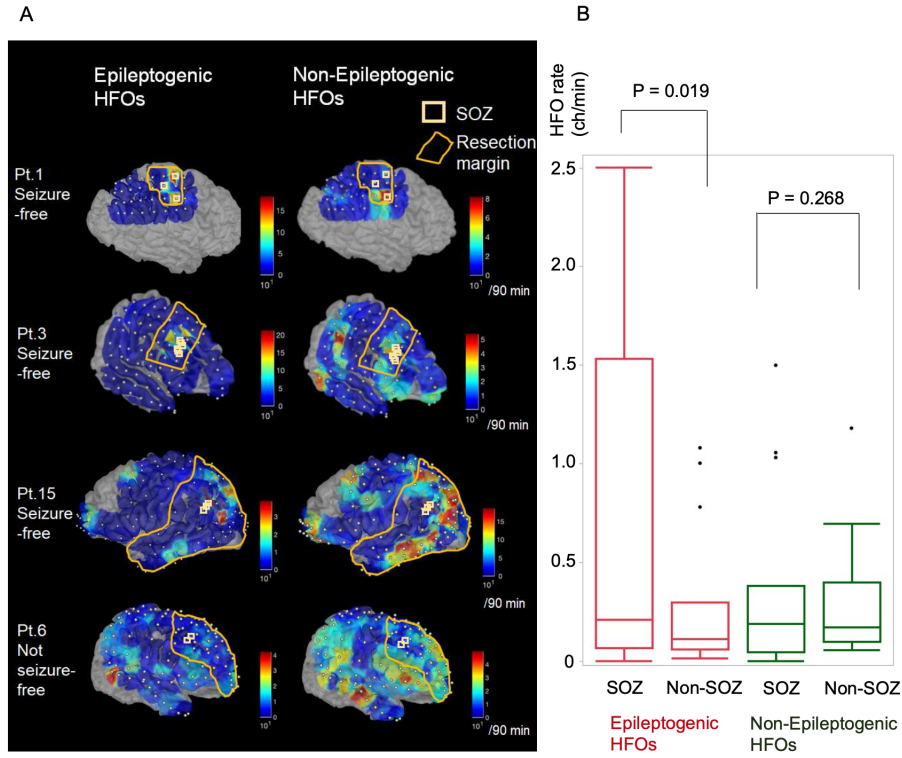


Figure 4.5: (A) In each row, individual 3D-reconstructed MRI with spatially coregistered electrodes is shown. Using a model developed using patient-wise cross-validation, the number of epileptogenic and non-epileptogenic HFOs (number/90 minutes) are projected onto the individual 3D MRI as an inference. Seizure onset zones are marked with white squares, and resected brain regions are marked with orange lines. The first three rows include subjects who achieved seizure-freedom after resection. The last row represents a subject who did not achieve seizure-freedom after resection. Epileptogenic HFOs localize around the seizure onset zones, whereas non-epileptogenic HFOs are localized more diffusely in the entire hemispheres. (B) The rate of HFOs (epileptogenic and non-epileptogenic HFOs) in each patient ($n=15$) is plotted in box plots based on the location (SOZ vs. non-SOZ). The rate of eHFOs (eHFOs/min/channel) was higher in SOZ than that in non-SOZ (mean 0.74 vs. 0.28, $p = 0.02$, paired t-test). The rate of non-eHFOs (non-eHFOs/min/channel) did not differ between the SOZ and non-SOZ (mean 0.36 vs. 0.29, $p = 0.27$, paired t-test).

To visualize the distribution of the eHFOs and non-eHFOs in each patient’s data, we performed inference on HFOs obtained from 90-minute EEG data from 15 patients who underwent resection, using their corresponding patient-specific DLATREV models. Then we plotted the voltage map on the 3D cortical reconstruction to visualize the performance based on the number of eHFOs and non-eHFOs using the FreeSurfer-based cortical modeling. The rate of eHFOs and non-eHFOs was compared in the SOZ and non-SOZ. See Figure 4.5 for more details.

In general, eHFOs clustered around the SOZ, while non-eHFOs distributed diffusely. The rate of eHFOs (eHFOs/min/channel) was higher in SOZ than that in non-SOZ (mean 0.74 vs. 0.28, $p = 0.02$, paired t-test), and the rate of non-eHFOs (non-eHFOs/min/channel) did not differ between the SOZ and non-SOZ (mean 0.36 vs. 0.29, $p = 0.27$, paired t-test)

4.5.3 Perturbation analysis to investigate salient features of epileptogenic and non-epileptogenic HFOs

There exist several potential ways to determine an interpretable function computed by a blackbox classifier such as a DNN, including (i) training an interpretable model with the outputs of the classifier [105] and (ii) adversarial perturbations of the inputs so as to effect maximum changes in the probability of the predicted outcomes such as Grad-CAM [106]. Both of these methods are computationally expensive, and may not yield useful answers without prior knowledge. The knowledge distilled from the time-frequency plot characteristics of eHFOs and non-eHFOs and positive correlation between the eHFOs and spk-HFO was used for the perturbation analysis. We hence conducted the following two perturbations on all classified non-eHFOs in 90 min data to investigate the salient features that are critical to the model’s decision.

Perturbation on time-frequency plot: The $I(x, y)$ image provided a template of the pixels where eHFOs had statistically significantly higher magnitudes than non-eHFOs. Then we designed an inverted T-shaped mask to approximate this template. We hypothesized that

if we perturbed the scalogram of a non-eHFO over the template using the maximum value of the corresponding scalogram (thus making the scalogram similar to that of an eHFO), then the probability of the classifier output should significantly increase, making it look more like an eHFO to the classifier. In practice, each new value within this template equals $0.5 * \text{original value} + 0.5 * \text{maximum value}$. Note that this method perturbed only the scalogram channel input and kept the other channel inputs unchanged. We summarized the results via a histogram of the change in probabilities for each patient across all classified non-eHFOs. Additionally, a one-tailed t-test comparing values of eHFOs and non-eHFOs, was performed on the change of output probability score to ensure that the change was significant and generalized well on the population level. See Figure 4.6 for more details.

Perturbation on amplitude-coding plot: We hypothesized that a spike-like pattern close to an HFO detection in the time domain was a salient feature contributing to the eHFO discovery. We referred to this pattern as the upgoing or downgoing time-domain characteristic pattern (TDCP). There were two channels in which we could inject a TDCP: the EEG tracing plot and the amplitude-coding plot. We picked the amplitude-coding plot as it contained denser information than the EEG tracing plot, so that the model can have a more evident response to the perturbation. We mimicked the insertion of a TDCP by centering it at a particular time step and then replacing the values in the corresponding columns of the amplitude-coding with that of the TDCP. Furthermore, we scaled the TDCP such that for an upgoing TDCP, its peak equals the maximum value in the whole plot, and similarly for the downgoing TDCP, its valley equals the minimum value in the whole plot. In the perturbation, we kept the value of other input channels the same. If the hypothesis were true, we would expect the perturbed amplitude-coding input for a non-eHFO to significantly increase the classifier output probability, making it look more like an eHFO to the classifier. We searched through all possible placements of the TDCP in the amplitude-coding plot to determine the location that led to the most perturbation to the model. The column (where the center of the TDCP was placed) with the maximum change of the probability score

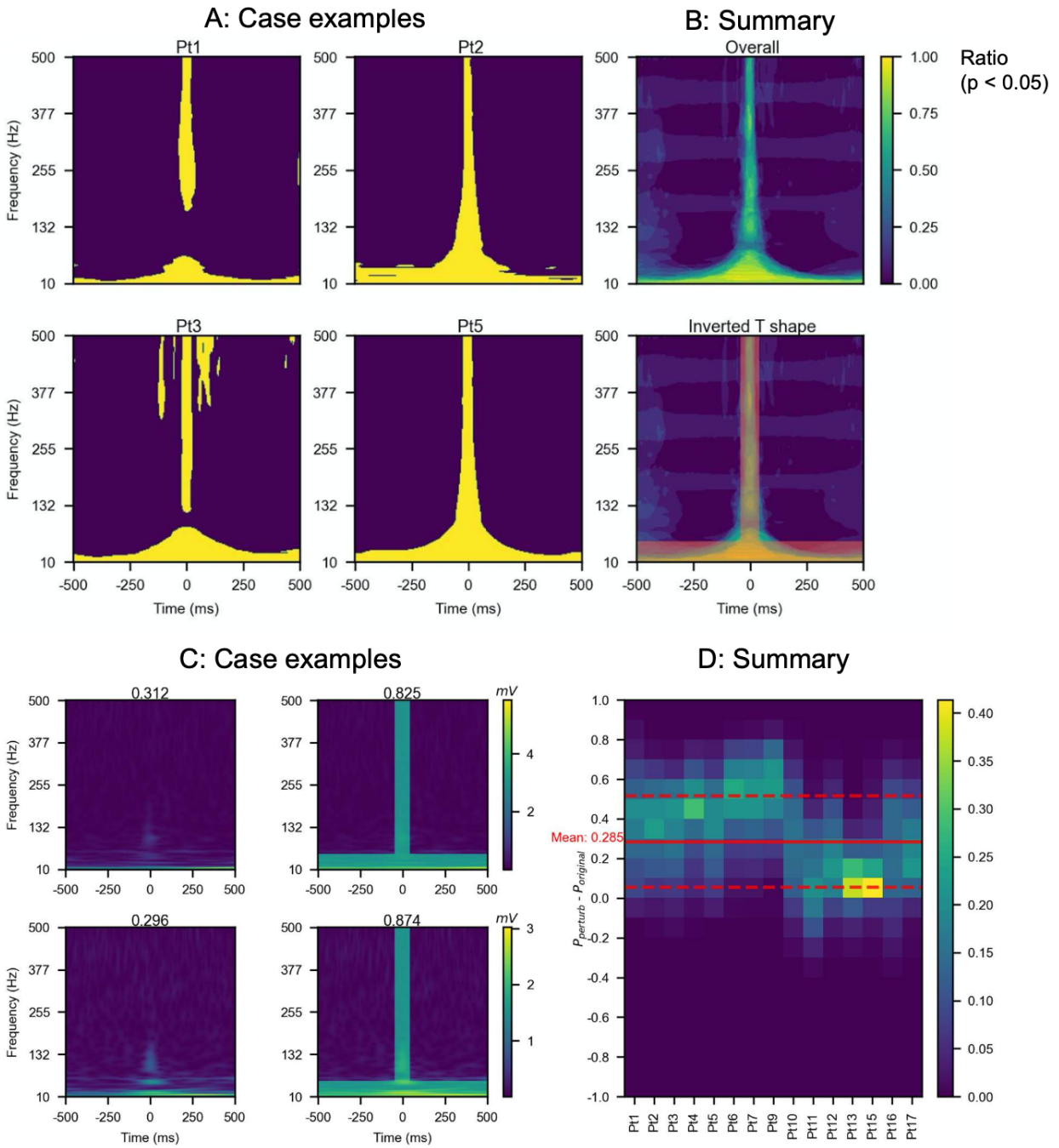


Figure 4.6: Continued on the following page.

Figure 4.6: (A) The time-frequency plot characteristics of epileptogenic and non-epileptogenic HFOs for Pt 1, 2, 3, and 5. The yellow-colored regions in the figure stood for the pixels, where the power spectrum of eHFOs is statistically higher than (p-value below 0.005 from one-tailed t-test) non-eHFOs. Panel (B-top) was generated by taking the average of the individual binary images from each of the 15 patients. It showed the distinguishing features are also significant at the population level. The inverted T-shaped was designed to approximate the differentiating region on the time-frequency plot (B-bottom). (C) The model's response to the inverted T-shaped perturbation on time-frequency plot. We provide two examples for perturbation for non-eHFO events in Pt 12. Each row presents one example and the first column indicates the original time-frequency plot while the second indicates the perturbed time-frequency plot based on the inverted T-shaped perturbation. The prediction value of the model changed from below 0.4 (therefore originally labeling it as non-eHFO) to above 0.8 (thus a change of +0.4) implying that the perturbed HFO would correspond to an eHFO. (D) The change in model confidence in population level. Each column (along the y-axis) is a histogram of the change in confidence for one distinct patient. It shows the frequency distribution of confidence changes after adding the inverted T-shaped perturbation to the time-frequency plot to all classified non-eHFOs for the given patient. The change in confidence level is significant, with an average of 0.285 noted as the red solid line in the histogram (a standard deviation noted as the red dashed line)

was viewed as the optimal time-location of the TDCP. We summarized the results via the histogram of the change in probabilities for each patient across all the classified non-eHFOs. A one-tailed t-test was also performed on the change of that output probability score to ensure that the change was significant and generalized well on the population level.

By utilizing the inverted T-shaped template found in the time-frequency map, we observed that the inverted T-shaped perturbation on the time-frequency plot significantly increased the model prediction probability towards eHFOs (mean probability increase was 0.285, $p < 0.001$) (Figure 6C, D). Based on the positive correlation between eHFOs and spk-HFOs, we set our hypothesized TDCP pattern as a spike. Therefore, we analyzed the effect of introducing a spike-like shape in the amplitude-coding plot. By introducing a downgoing or an upgoing spike close to 0 ms location in a non-eHFOs event, the model confidence increased towards an eHFO event (Figure 7A, D). On the population level, the time step with the highest probability increase was around the center location along the time-axis (Figure 7B, E), where the HFO was detected. Moreover, the prevalent probability increase in non-eHFO events among all 15 patients who underwent resection (Figure 7C, F) demonstrated the non-trivial model response by introducing a spike in the time domain (mean probability increase of 0.438 for a downgoing spike introduction, and 0.465 for an upgoing spike introduction, both with $p < 0.001$).

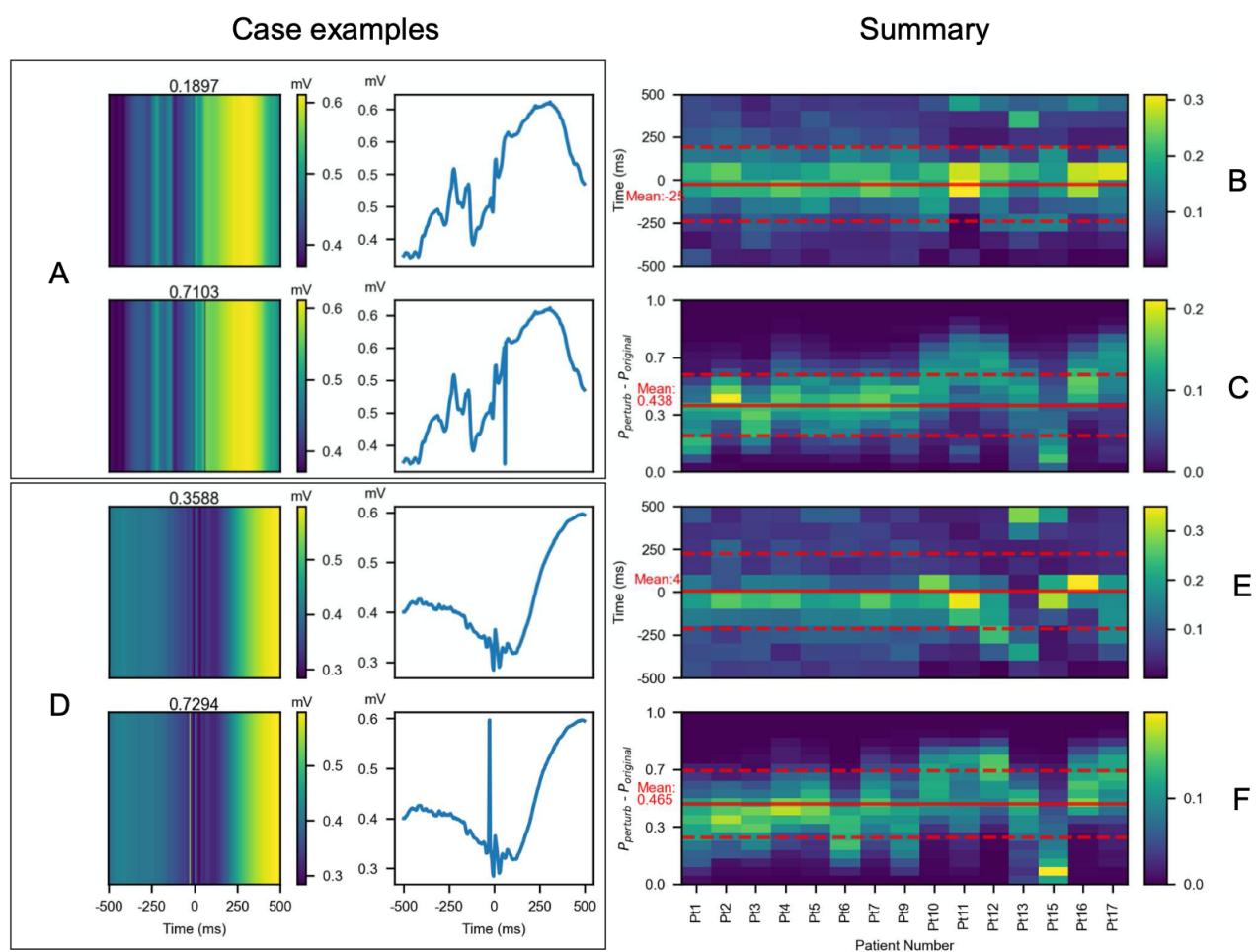


Figure 4.7: (Continued on the following page.)

Figure 4.7: **The model’s responses to injecting a spike-like feature into the amplitude-coding plot.** (A,D) Examples of introducing a downgoing (A) and upgoing (D) spike feature in classified non-eHFO events. These demonstrate that on the introduction of a spike-like perturbation, the model predicts higher confidence toward eHFOs. Subfigure A shows the original amplitude encoding image input and the corresponding time-series signal (top row), and the perturbed amplitude-coding plot, and the corresponding time-series signal with downgoing spike perturbation (bottom row). Similarly, subfigure D shows the same information about a different classified eHFO but with upgoing spike perturbation. (B, C) For each non-eHFO, a downgoing spike perturbation could be introduced at every point in the time interval. For the perturbation, which resulted in the maximum change in confidence, its location (relative to the center, thus in the range of +500 to -500 ms) and the resulting change in confidence (-0.5, 1) are noted. For each patient, we compute a histogram for the distribution of the change in confidence (C) and a separate histogram for the location of the spike over non-eHFOs (B). The same steps are repeated for upgoing spike perturbation, and the results are shown in (E, F). For both up-and-downgoing spikes, the histograms (B, E) show that the spikes located close to the HFO event lead to maximum change in confidence. The change in confidence for both up-and-downgoing spike perturbation is significantly greater than zeros, with means downgoing: 0.438 and upgoing: 0.465, and the locations leading to the max perturbation are close to zeros location with means downgoing: -25ms and upgoing: +4ms. All of these means are noted as red solid lines in each histogram (a standard deviation noted as red dashed lines).

4.5.4 Robust choice of the weight term

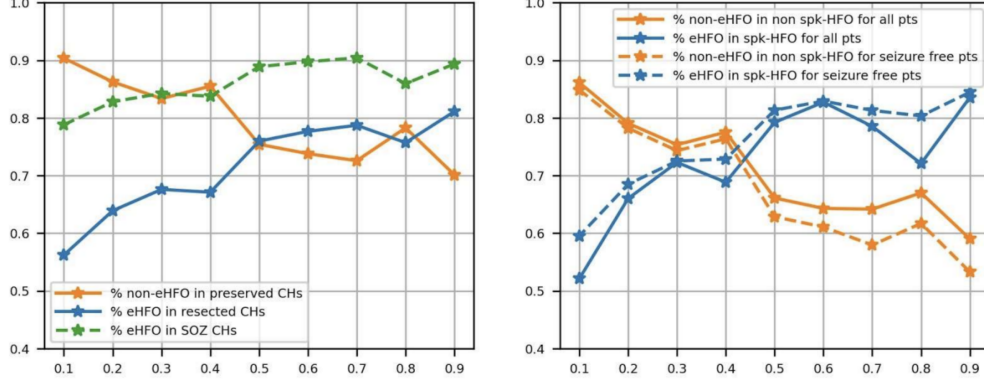


Figure 4.8: Find the optimal weight (w) term for the Binary Cross Entropy Loss.

To find the optimal weight (w) term in the Binary Cross Entropy Loss, we swept the w from 0.1 to 0.9 (x-axis) and compared the performance of the model trained by each w with our hypothesis in patients who had resections with post-operative seizure freedom ($n=9$) in Figure 4.8. The first criterion for selecting an optimal weight w was based on the following hypothesis that requires no expert input: For post-surgery seizure-free patients, the majority of the classified non-eHFOs should be located within the preserved regions, and correspondingly, the majority of the eHFOs should be located within resected regions; moreover, a vast majority of the HFOs in the SOZ channels should be labeled as eHFOs. Thus, an optimal w should balance the first two percentages (i.e., $\%$ of non-eHFOs in preserved regions and the $\%$ of eHFOs in resected regions, shown in y-axis) and maximize the percentage of eHFOs in the SOZ channel. As the Left plot shows, $w = 0.5$ satisfies this criterion: The $\%$ of eHFOs in the resected CHs (orange) equals the $\%$ of non-eHFOs in preserved channels (blue), and the $\%$ of eHFOs in SOZ CHs reaches a maximum value. A second criterion for validating an optimal choice of w was based on the hypothesis that there should be an agreement between the data-driven predictions and expert knowledge: For all patients, the majority of the de-

tected eHFOs should also be labeled by experts as pathological HFOs (i.e., spk-HFOs), and correspondingly, the majority of the detected non-eHFOs should be labeled as physiological by the experts (i.e., non-spk-HFOs). An optimal w would be at a value where the corresponding percentages are balanced. The Right figure shows that the respective percentages (orange vs. blue) become equal (72%) at $w = 0.45$ and remain still balanced at $w = 0.5$ (78% vs. 68%). Given potential inter-rater variations that could change the percentages, we considered $w = 0.5$ to be an optimal choice for our study.

4.6 Discussion

We demonstrated how DL might be used in HFO classification to complement experts at multiple levels. As a first step, we demonstrated that a DL-based algorithm robustly emulated HFO annotations by human experts in rejecting artifacts and determining if an event is associated with a spike. Thus, the associated DL models could help experts with their manual verifications and significantly reduce experts' efforts without compromising performance. More importantly, in the second step, we showed how DL could create novel computational models which could define morphological classes of HFOs indicative of eHFOs and non-eHFOs when guided only by clinical outcomes such as seizure outcomes and resection status of the channels without any expert EEG labeling. We took advantage of our large dataset containing more than 30,000 HFOs within 90 minutes of EEG data from subjects with known post-operative seizure outcomes. The model was trained with a novel weakly supervised approach that utilized the inexact label of resected and non-resected status to discover eHFOs and non-eHFOs. We further showed that the DL-defined eHFOs were clinically relevant and possessed two salient features (HFOs associated with spikes and the inverted T-shaped pattern in scalogram) that experts had identified as characteristics of eHFOs. The removal rate of eHFOs correlated to seizure-freedom after resection, but no such relationship was seen with non-eHFOs. The use of the removal ratio of eHFOs may have an additional value

on the status of complete removal of the SOZ, but the number of patients may be too small to conclude this definitively. By comparing the predicted results with expert annotation, we observed that the most classified eHFOs were spk-HFOs. Similarly, most non-eHFOs were non-spk-HFOs. At SOZ, we observed a higher rate of eHFOs compared to that of non-eHFOs. We next elaborated on how the traits of DL-discovered eHFOs compared with hypothesized traits of so-called pathological HFOs. Pathological HFOs, often seen as fast ripples (FRs: 250 Hz or above), are believed to be excitatory neuronal activities, such as summated action potentials from synchronously bursting neurons [107, 108]. Contrarily, physiological HFOs, which often involve ripples (80-250 Hz), are considered to reflect summated inhibitory postsynaptic potentials [102, 109]. These biological mechanisms are hypothesized to lead to morphological differences between pathological and physiological HFOs, and hence, making them discoverable with computer vision. We used the trained DL model to derive certain signatures of eHFOs in the time-frequency domain that agreed with such common clinical knowledge. In particular, based on the analysis of time-frequency plots of eHFOs and non-eHFOs, eHFOs showed stronger signals throughout ripple and fast ripple bands at the onset of HFOs, which shares the similar observation in HFOs seen at the SOZ [110]. Meanwhile, eHFOs generally showed stronger signals in the low frequency region throughout the time window, which might represent an inhibitory slow-wave postsynaptic component, coupled with out-of-phase excitatory fast firing of HFOs [111]. Taking advantage of these observations, we designed perturbations in the input to probe whether these characteristics were actually the salient features that the model relied on to make a prediction. The group analysis on 90 min data demonstrated that both characteristics led to a significant model confidence increase towards eHFOs. Similarly, we found that the model had automatically learned the salient characteristics of manually labeled spk-HFOs. In addition, the artificial introduction of spike-like activity around the HFO onset also increased model confidence towards eHFOs. Clearly, such expert knowledge was not hard-coded into the DNNs and was automatically inferred only from the partial clinical outcomes. It is noteworthy that we

discovered such salient features of epileptogenic HFOs from exploratory approaches utilizing DL. Clearly, one cannot exhaustively enumerate all the features that the DL model used to make its predictions, and these features are only some of the salient ones that aligned well with expert knowledge. As the study grows in scale and the DL models become accurate and robust in discovering eHFOs, we expect new salient characteristics to be discovered. The prior studies [112, 113] used DL architecture to classify HFOs using human-annotated data, however, such approach has constraints including necessity of human experts and its associated potentially unfavorable inter-rater reliability among experts. This is the first study to demonstrate that DL algorithms can learn from clinical outcomes (such as seizure-freedom after resection) and soft labels (such as resected vs. preserved status) to classify HFOs. A recent DL study [114] using clinical outcomes as soft labels to classify raw EEG signals into focal and non-focal seizures in patients with epilepsy was also proven to be feasible. Our approach expands the spectrum of clinical usage, since there will be no need for expert annotation and easily applied to a larger dataset. In this study, we determined the resection status of channels among human experts, by reviewing pre-and post-resection intraoperative picture and post-operative brain MRI. This subjective approach might provide another type of labeling error, thus automated and objective determination of channel resection status might have been considered.

This work indicates that our deep learning approach can overcome issues including poor inter-rater reliability of HFO classification among human experts and their time constraints of analysis. Although our inter-rater reliability in this study was favorable as previously reported [115], we expect the agreement will likely diminish when raters are from different institutions or have different experience levels. Using HFO analysis, we may identify brain tissue that needs to be removed during surgery without human experts' effort if an algorithm was trained from large enough EEG data from patients with known post-operative outcomes. Notably, validation was performed using patient-wise cross-validation to maintain generalizability across patients so that the results may be applicable to new patients. Based on

our results, the higher standard deviation in the performance of the test set in patient-wise cross-validation than the 5-fold cross-validation implies that the data distribution for some specific patients differs in some aspects from the rest (possibly by recording environment, sex, age, and pathology). This issue will be resolved once we increase the number of patients in the cohort. The already high performance obtained in patient-wise cross-validation, even with $n=19$, is a promising indicator that models trained with retrospective data from a large enough cohort will generalize well to new and unseen patients.

There are several limitations in our study. Regarding the dataset, we have included only 19 patients, and there were only 14 patients who had resective surgery with known clinical outcomes at 24 months. Although we analyzed extended EEG data (90 minutes) from nine patients who achieved seizure-freedom at 24 months after surgery, we did not analyze the entire EEG recordings. It is of interest to include more patients and EEG data to train our DL algorithm and evaluate how much the performance could improve. In fact, we are planning to build a model using EEG data from more than 100 patients who underwent resection. Also, all the data were from pediatric patients from the same institution. With a diversified age range and epilepsy pathology, the morphology of HFOs may change, and the algorithm might be trained differently.

In this work, we proposed an automated tool to analyze HFOs in a large dataset by using DL to simulate a human expert’s visual verification and to predict seizure-free status through an outcome-based reverse engineering approach. Future work to further refine this methodology by examining more datasets from multiple institutions is still needed. By refining a method to distinguish epileptogenic HFOs from others, we will be better positioned to confidently use HFOs to guide resection margin in clinical trials to improve the chance of post-operative seizure-freedom in patients with drug-resistant epilepsy who undergo epilepsy surgery.

CHAPTER 5

Context Aware and Mode Preserving Generative Models (I): Action-type Conditioned 3D Motion Generation, Mode Discovery, and Customization

5.1 Introduction

Human motion generation has wide applications in machine vision, robotics, video games and computer animation. While considerable progress has been made in pixel-based video generation methods [11, 12, 13], the resulting human appearance is still unsatisfactory [6]. Using 3D human skeleton sequences to guide such video generation is considered to be a fundamental step to reduce distortions in human appearance. In recent years, forecasting missing skeleton motion sub-sequences or performing motion completion tasks have been attempted [116, 117, 118, 119, 120, 121, 122]. However, initial or intermediate motion sub-sequences are still required as inputs, which limit these models' applications.

We address the fundamental problem of 3D motion generation without any initial motion as inputs. Given only the activity type, chosen from among a pool of different activities, and its characteristic features, the generative model should generate motion sequences that are physically realistic and diverse in style, reflecting variations across the human population. Amongst many related works such as [123, 124, 125, 126], the most relevant and recent ones are two stage GAN [127], MoCoGAN [128], Action2Motion [6] and ACTOR [129]. [6] in particular is a striking recent work that is among the first to propose automated



Figure 5.1: **ODMO motion generation.** Top: Sample motion sequences generated for multiple activities; Bottom: Diverse motion sequences for Walk activity generated by multi-modal sampling.

generation of 3D pose sequences for multiple activities from the same generator, including complex motion such as dancing and animal behavior. A recent follow up work [129], applies a Transformer-based VAE architecture (instead of being LSTM based) for encoding and decoding, and avoids a time-dependent latent space embedding as done in [6]; it shows considerable performance gain over [6]. In both these frameworks *addressing our problem of motion generation based only on action type*, different action categories share the same latent space, and customization capabilities such as the ability to generate different styles on demand (i.e. in a controllable fashion) within the same category or generating interpolated motion sequences among different categories are missing.

Our work has the following distinctive features:

- **1) Internal causal motion representation:** Any 3D sample motion is decomposed into two parts: spatial joint movements relative to a body center (root) – referred to as local movement profile (LMP) and the resulting trajectory of the body center. This key step is rooted in the kinematics of body movement: We move our body parts relative to each other (as captured in the LMP), and these movements after interaction with the environment cause our body center to have spatial movement resulting in a particular trajectory. We, thus, first encode the primary causal factor (the LMP) into a low-dimensional latent code and then train a decoder (that learns the causal factor of the environment) to reconstruct the resulting trajectory and motion of body parts. *We also believe that this decomposition enables us to perform better than the existing models even with the (x, y, z) representations of the skeleton joints, instead of using more complex representation such as Lie-Group [6] or 6D [129].* This flexibility has an added advantage: We are able to work with rich datasets where only 3-D representations are available and estimation of the rotation representations are unstable.
- **2) Multi-modal latent code space engineering:** Additional regularization terms

are added to the VAE-type encoder loss function to enable contrastive learning and separate latent codes belonging to different action types (and styles within an action type) into distinct clusters. This preserves the different modes and styles present in real motion, enabling novel sampling strategies to support motion customization.

- **3) Multiple mode discovery and multi-modal sampling:** Since the codes for action categories are separated, we do a Gaussian Mixture Model (GMM) fitting of the codes in the same category, where the number of mixture components is determined adaptively, to discover different styles within an action type. These distributions are then sampled to generate diverse motions on demand.
- **4) Both intra- and inter-action type motion interpolation:** Given the multi-modal representations of the latent code space for each action type, one can perform interpolation between modes to generate realistic mixed motion sequences not seen in the training set. Similarly, our framework allows inter-action category interpolations, which can be meaningful for action types that are similar enough.
- **5) Trajectory customization via destination location (end-point) specification:** For action types with significant spatial movement, such as walking, running, and jumping, our framework enables one to generate motion with different styles (by varying the code) while keeping end-point the same. Analogously, by varying the destination location, while keeping the same latent code (hence, style) motion sequences that cover a large spatial domain are generated. Such customized motion can be shown to be realistic by benchmarking motion quality metrics at the ensemble level.

5.2 Related work

Modeling human motion: As already mentioned, various methods have been proposed to model and generate motion with widely varying goals and assumptions about prior informa-

tion. The more traditional models, such as Hidden Markov Models [130], switching linear dynamic systems [131], Gaussian processes [132] and random forests [133] have been proposed. However, some of these methods, like Gaussian processes, are well-known to be computationally intractable, especially for complex datasets. More recently, deep learning frameworks have also been used for deterministic or probabilistic motion generation: deep recurrent neural networks (RNNs) for modeling dynamic temporal behavior [134, 135, 136, 137]; Autoencoder [138], Normalising Flows [139], VAE [140] and GAN [128]. Additionally, embedding of motion in a low dimensional latent space is used implicitly or explicitly by several models to assist their generation tasks, including both time dependent [132, 6] or independent [138, 129] encodings.

Human motion prediction: Similar to the motion generation task, short and long term motion prediction tasks also require modeling of human motion. However, they require additional information as input: the first several motion frames of the desired activity. Moreover, the evaluation metrics are mainly designed to compare the predicted future motion with ground truth motion. In our setup, the generated motion sequences can only be evaluated at the population level with the ground truth data. [119] used Residual Graph Convolutional Network (GCN) blocks to capture spatial-temporal correlation, and focuses on recovering the accurate future motion based on the historical poses. [121] also uses GCN but trains the model in a multitask setting to predict future motion, and recover missing joints in the past at the same time. However, they both rely on history information to generate *deterministic* single future motion. [122] introduced a sampling version of motion prediction, and also considered diversity in the forecast motion. They used learnable mapping functions to map samples from a single random variable (generated by a given historical information sequence) to a set of correlated latent codes for further decoding into a set of correlated future motion sequences. [120] used novel variational autoencoder which preserves the full temporal resolution of the history motion as condition and samples from the latent frequencies explicitly to

introduce high-frequency components into the generated motion. However, in such stochastic prediction tasks, the past motion plays an important role (as input and condition). To illustrate this, *we modify the motion prediction model in [122] to suit our generation task by replacing the past motion information with category label and set it as a baseline algorithm.*

5.3 Algorithm

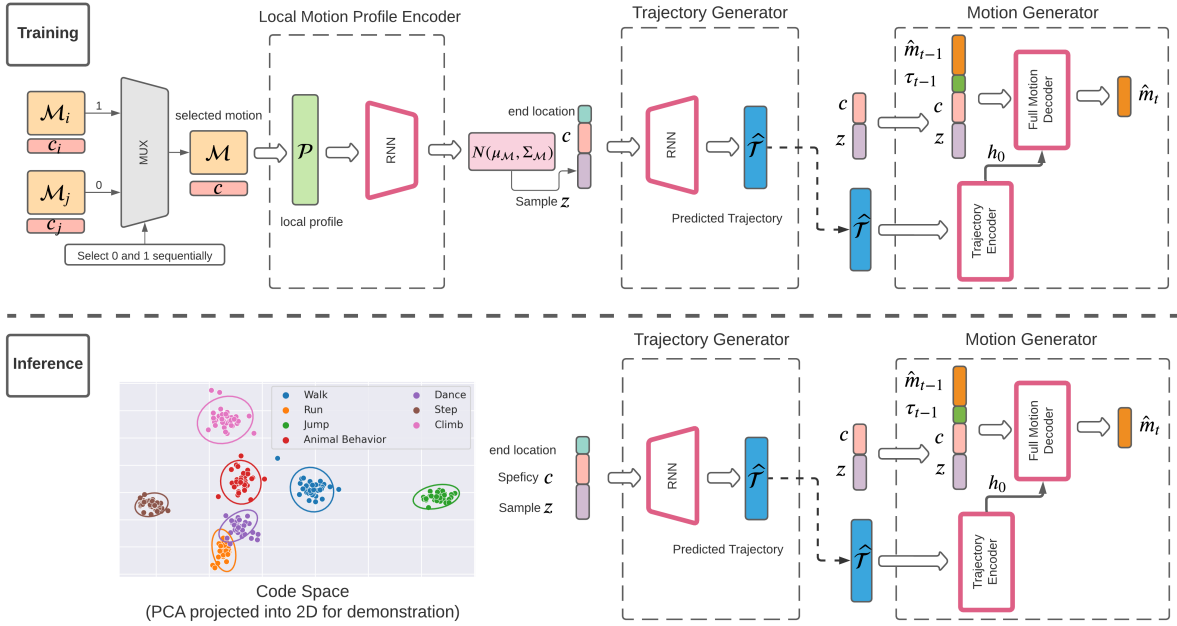


Figure 5.2: Overview of our model architecture.

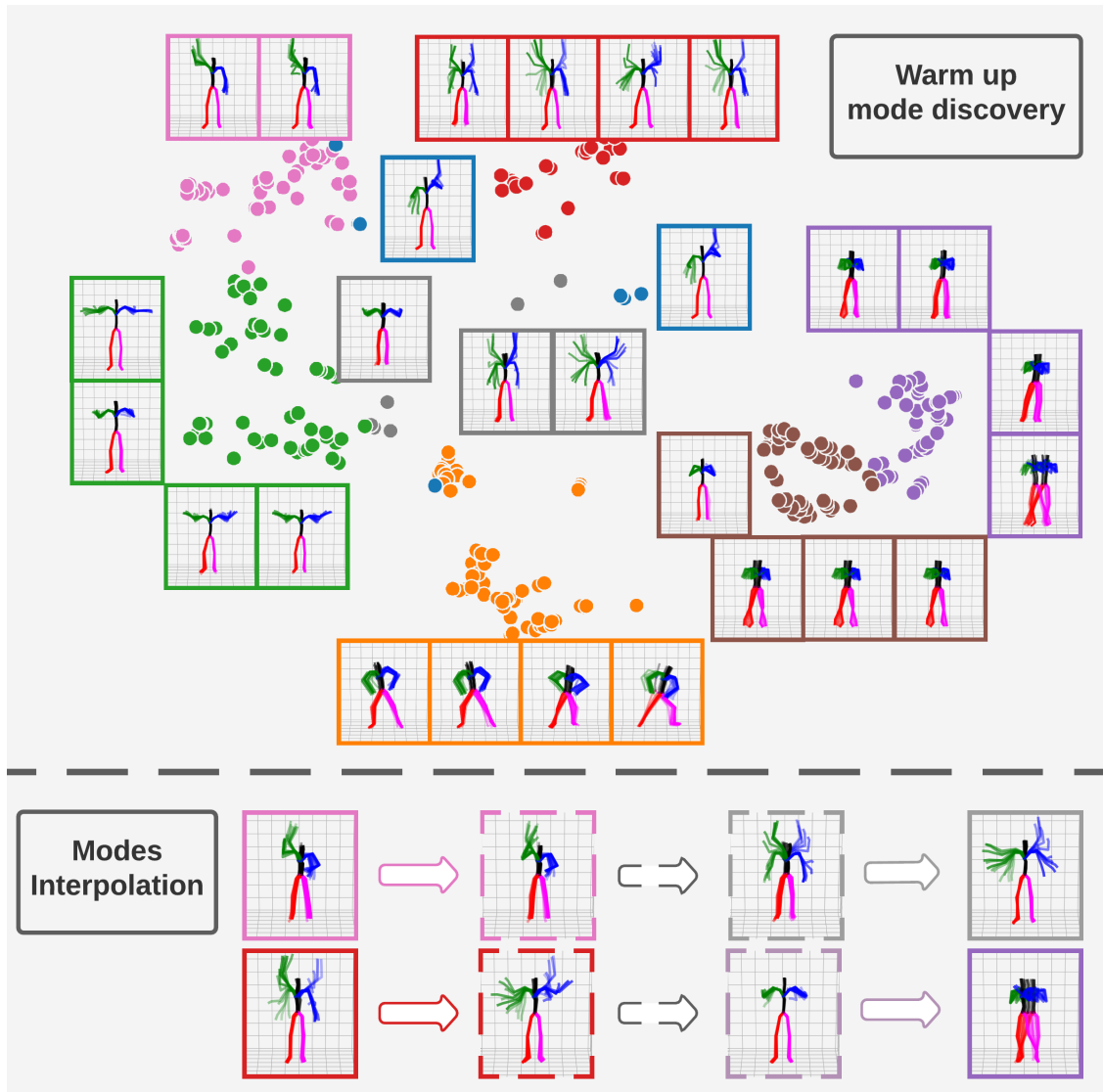


Figure 5.3: **Mode discovery in Warm Up and Intra-Action Category Interpolations** (HumanAct12): Latent codes are projected onto a 2D space using t-SNE [22] to illustrate that the codes can be separated into distinct clusters. For each cluster we sampled a few codes and time-lapsed images summarizing the corresponding generated motions are shown. These time-lapsed images are highly consistent within clusters, providing evidence that each cluster represents a distinct mode of the action type. Together, the modes highlight the different warming up routines present in the training dataset. As the bottom insets show, interpolations between codes from two modes show smooth morphing.

5.4 Method

5.4.1 Notations

Given a set of 3D motion sequences, $\{\mathcal{M}_1, \mathcal{M}_2, \dots\}$ along with their corresponding activity categories, our goal is to train a generative model that is able to produce 3D motion sequences with fixed length T for a specified activity category. Inside a motion sequence $\mathcal{M} = [\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_T]$, the pose $\mathbf{m}_t$ in the t^{th} frame is represented by the locations (e.g. their (x, y, z) coordinates) of J joints: $\mathbf{m}_t = [\mathbf{m}_t(0), \dots, \mathbf{m}_t(J-1)]^T \in \mathbb{R}^{3 \times J}$. One joint, usually the center of the pelvis is referred to as the root joint. We assume that the body center can be approximated by the root joint, and its movement in 3D space along time will be referred as the *trajectory of the motion*, and denoted as $\mathcal{T} = [\mathbf{r}_1, \dots, \mathbf{r}_T]$, where $\mathbf{r}_t \in \mathbb{R}^3$. The end location $\mathbf{r}_T$ of the root joint will be referred to as $\mathbf{r}_e \in \mathbb{R}^3$, and can be used to guide motion generation in our models. Additionally, each activity class c is denoted by a one-hot encoded vector. Furthermore, *local movement profile* (LMP) of a given motion sequence $\mathcal{M}$ is defined as the locations of the joints relative to the root joint position, i.e. $\mathcal{P} = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_T]$, where $\mathbf{p}_t \in \mathbb{R}^{3 \times J}$, and $\mathbf{p}_t(i) = \mathbf{m}_t(i) - \mathbf{r}_t$, $i = 0, \dots, J-1$, $t = 1, \dots, T$. This decomposition of any motion sequence $\mathcal{M}$ into two components, the LMP $\mathcal{P}$ and the resulting trajectory $\mathcal{T}$, is key to the success of our framework, as it enables us to establish a causal relationship between body movement of the human agent, its environment, and the trajectory that ensues.

5.4.2 Model Architecture and Loss Functions

There are three main components in our ODMO (**O**n **D**emand **M**Otion generator): local movement profile encoder (LMP encoder), trajectory generator, and motion generator.¹:

¹For simplicity, the loss functions introduced in this section are defined for a single sampled motion; the only exception is $\mathcal{L}_{es}$ defined in Equation 5.1, which involves a pair of sampled motions, $\mathcal{M}_1$ and $\mathcal{M}_2$, whose corresponding LMPs are $\mathcal{P}_1$ and $\mathcal{P}_2$, and are of class c_1 and c_2

i) LMP encoder Es_θ , parametrized by network parameters θ which learns a representation of local movement profiles ($\mathcal{P}$) in a lower dimensional space $\mathbb{R}^D$. Style codes $\mathbf{z}$ will be sampled from this space, $\mathbf{z} \sim \mathcal{N}(\mu_\theta, \Sigma_\theta)$, where $(\mu_\theta(\mathcal{P}), \Sigma_\theta(\mathcal{P})) = \text{Es}_\theta(\mathcal{P})$. A contrastive loss is applied to a pair of LMPs, $(\mathcal{P}_1, \mathcal{P}_2)$ following Equation 5.1. The contrastive loss term is added to ensure that embeddings of motion sequences belonging to different action categories are well separated and form distinct clusters in the latent space. Moreover, the target variance of posteriors is set to 0.01 in the KLD loss terms so that the embeddings of sequences belonging to the same action category are well separated, preserving the multimodality inherent in real motion. This loss function creates a multi-cluster distribution in the latent space where the clusters themselves can be modeled as GMMs, which are then sampled to generate motion. In Equation (5.1), α is the contrastive margin.

$$\begin{aligned} \mathcal{L}_{es}(\theta) = & \mathbb{1}_{\{c_1=c_2\}} \|\mu_\theta(\mathcal{P}_1) - \mu_\theta(\mathcal{P}_2)\|_2^2 \\ & + \mathbb{1}_{\{c_1 \neq c_2\}} \max\{0, \alpha - \|\mu_\theta(\mathcal{P}_1) - \mu_\theta(\mathcal{P}_2)\|_2\}^2 \\ & - \ln(\det \Sigma_\theta(\mathcal{P}_1)) + 100 * \text{Tr}(\Sigma_\theta(\mathcal{P}_1)) \\ & - \ln(\det \Sigma_\theta(\mathcal{P}_2)) + 100 * \text{Tr}(\Sigma_\theta(\mathcal{P}_2)) \end{aligned} \quad (5.1)$$

ii) Trajectory generator Gr_ϕ , which predicts the resulting 3D trajectory $\mathcal{T}$ via predicting trajectory velocities for T steps, based on the sampled style code $\mathbf{z}$ and one-hot encoded categorical information c . The predicted trajectory $\hat{\mathcal{T}}$ will be reconstructed based on the accumulated velocity predictions, i.e. $\hat{\mathbf{r}}_t = (\text{Gr}_\phi(c, \mathbf{z}))_t + \hat{\mathbf{r}}_{t-1}$, where $\hat{\mathbf{r}}_t$ is the predicted location at time t ($\hat{\mathbf{r}}_0 = \mathbf{0}$). The trajectory generator is trained to predict the velocity of root trajectory $\mathbf{r}_t - \mathbf{r}_{t-1}$ at t^{th} time step ($\mathbf{r}_0 = \mathbf{0}$) with loss:

$$\mathcal{L}_r(\phi) = \sum_{t=1}^T \|(\text{Gr}_\phi(c, \mathbf{z}))_t - (\mathbf{r}_t - \mathbf{r}_{t-1})\|_2^2. \quad (5.2)$$

(iii) Motion generator Gm_ψ , which generates realistic motion sequence $\hat{\mathcal{M}}$ based on style code $\mathbf{z}$, guidance trajectory $\hat{\mathcal{T}}$, category label c . The motion generator is adapted from Seq2Seq [141] which consists of an encoder module and decoder module. The trajectory

encoder encodes the predicted trajectory and sends the compact representation of the trajectory to the full motion decoder (decoder) as the initial hidden state h_0 . The decoder predicts the whole motion sequence $\hat{\mathbf{m}}_1, \hat{\mathbf{m}}_2, \dots, \hat{\mathbf{m}}_T$ sequentially, as in Equation 5.3. At t^{th} step of decoding, the decoder employs the code $\mathbf{z}$, class information c , the time counter τ_t , and the previous step of output $\hat{\mathbf{m}}_{t-1}$ (or $\mathbf{m}_{t-1}$ if teacher forcing is used) to predict the pose $\hat{\mathbf{m}}_t$. In the training process, we apply the reconstruction loss $\mathcal{L}_{mre}$ on the predicted motion,

$$\mathcal{L}_{mre}(\psi) = \sum_{t=1}^T \|\mathbf{m}_t - \text{Gm}_{\psi}(\tilde{\mathbf{m}}_{t-1}, \tau_t, c, \mathbf{z}, \hat{\mathbf{r}})\|_2^2, \quad (5.3)$$

$$\tau_t = \frac{t}{T}, \tilde{\mathbf{m}}_{t-1} = \begin{cases} \mathbf{m}_{t-1}, & \text{if tf} = 1 \\ \hat{\mathbf{m}}_{t-1}, & \text{otherwise.} \end{cases}$$

Finally, we use one combined loss function to jointly optimize the whole network:

$$\mathcal{L}(\theta, \psi, \phi) = \mathcal{L}_{es}(\theta) + \mathcal{L}_r(\phi) + \mathcal{L}_{mre}(\psi) \quad (5.4)$$

5.4.3 Sampling the Latent Code Space

We propose two different sampling strategies for the motion generation task, offering users freedom to customize desired properties of the sampled motion sequences.

Standard GMM sampling: We fit a GMM with the same number of components as the real motion activity categories ($|\mathcal{C}|$), to all the learned latent codes (i.e. the means) across all activity types. Thus, each activity type has a Gaussian distribution assigned to it, which is sampled to generate latent codes for motion generation.

Multi-modal sampling: In real motion, there would be several modalities in each motion category; for example, in the Jump category, one can have humanoids jumping forward, or jumping vertically with little horizontal motion.

Our training formalism encodes the modalities of an action category (present in the data)

as modes in the distribution of latent codes inferred for that category. We fit a GMM with K components to the codes for each category, where K is chosen adaptively by using the silhouette score [8]. One can sample these action specific GMMs to generate multi-modal motion for the same action type.

5.5 Experiment and result

Our base model, ODMO, is trained with the end location information as part of the input to the trajectory generator and the standard GMM sampling strategy is adopted. An end location is generated by a k-nearest neighbors (KNN) trained with codes from real motion and their corresponding end locations. Additionally, we develop another model (ODMO_{NE}) trained with the same architecture but without end location information as input. The implementation details can be found in the supplementary material.

5.5.1 Dataset

We evaluated models on three public motion datasets:

HumanAct12: We used the same HumanAct12 dataset as introduced in [6, 7] which consists of 1,191 motion clips and 12 categories such as: Warm up, Walk and Drink.

UESTC: Compared to HumanAct12 dataset, UESTC [142] uses a finer granularity in defining action categories, resulting in 40 action types performed by 118 subjects; example action types include: Squatting, Punching and Left stretching. There are 25,600 video samples collected from 8 fixed viewpoints. For fair comparison, we used the same preprocessed data as [129]. Motion sequence samples shorter than a given threshold are filtered. This results in 10,629 training sequences and 13,350 testing sequences.

CMU MoCap (MoCap): We identified seven action categories such as Walk, Run and Dance from the CMU MoCap data library consisting of 970 motion sequences in total. For

fair comparison, we used the same data preprocessing steps as [6] but dropped the Wash activity, as most of the motion sequences inside this class are mislabeled.

Among these three datasets, MoCap has real motion with longer overall spatial translations as well as more diversity within each action type making it much more challenging.

5.5.2 Metrics based on classifiers

In this approach, an action-category classifier is designed based on real motion, and then is used to evaluate different ensemble properties of the generated motion. The classifier metrics for the real motion then can be used as ground truth. Adopted from [129, 6, 143], the metrics we used are as follows: Recognition Accuracy (**Acc**) measures the confidence of generated motion belonging to a given category. Higher Acc implies better motion quality. Frechet Inception Distance (**FID**) [144] computes the distance between the distributions of the classifier features of the real motion and the generated motion; hence, *lower FID is desirable*. The **Multimodality (Diversity)** is the expected distance between classifier features of two randomly sampled generated motion sequences belonging to the same action category (across all categories). Thus, *motion samples generated by a better model should have multimodality/diversity closer to that of the real motion*.

We trained such classifiers with real motion sequences as mentioned in [6, 129]. For MoCap, we re-trained the classifier with the same architecture as in [6] but changed the number of outputs to 7. For HumanAct12, we directly employ the same classifier as [6, 129]. For UESTC, since the classifier from [129] is in 6D representation, it is not suitable for our xyz representation. Hence, we followed the train and test split as in [129] and trained our own classifier in xyz representation (the architecture is shown in the supplementary material). The metrics obtained from the classifiers are reported in Tables 5.1 and 5.2 as the mean of 10 sets of samples generated from each model with the 95% confidence interval on the top; each set comprises 400/250/200 samples for each action type in Mocap/HumanAct12/UESTC dataset.

5.5.3 Comparison to the state of the art

Table 5.1: Benchmark on HumanAct12 & UESTC Dataset based on classifier. (Note for UESTC, FID_{tr} and FID_{test} are computed based on train and test split as done in [129].)

Methods	HumanAct12				UESTC				
	Acc $\uparrow$	FID $\downarrow$	Diversity	MModality	Acc $\uparrow$	FID $_{tr}\downarrow$	FID $_{test}\downarrow$	Diversity	MModality
Real motions	99.70 $\pm$.06	0.02 $\pm$.01	7.08 $\pm$.06	2.53 $\pm$.05	99.79 $\pm$.05	0.01 $\pm$.00	0.05 $\pm$.00	7.20 $\pm$.06	1.61 $\pm$.02
DLow-Gen	8.29 $\pm$.16	4.87 $\pm$.03	6.39 $\pm$.09	6.39 $\pm$.06	2.71 $\pm$.11	3.28 $\pm$.06	3.01 $\pm$.06	6.39 $\pm$.06	6.33 $\pm$.03
Act-MoCoGAN	8.29 $\pm$.18	34.77 $\pm$.06	1.64 $\pm$.02	1.63 $\pm$.04	3.87 $\pm$.07	25.48 $\pm$.05	24.76 $\pm$.05	4.22 $\pm$.12	2.28 $\pm$.02
A2M-xyz	69.17 $\pm$.55	1.73 $\pm$.06	6.73 $\pm$.06	4.18 $\pm$.09	90.25 $\pm$.21	0.39 $\pm$.01	0.39 $\pm$.00	7.08 $\pm$.05	1.83 $\pm$.02
A2M-Lie	93.25 $\pm$.47	0.21 $\pm$.01	7.02 $\pm$.05	2.77 $\pm$.07	91.07 $\pm$.21	0.24 $\pm$.01	0.25 $\pm$.00	7.09 $\pm$.06	1.91 $\pm$.04
ACTOR	96.00 $\pm$.23	0.09 $\pm$.00	6.97 $\pm$.08	2.59 $\pm$.07	91.87 $\pm$.18	0.23 $\pm$.01	0.26 $\pm$.01	7.10 $\pm$.09	1.89 $\pm$.04
ODMO	98.57$\pm$.32	0.07$\pm$.00	6.99 $\pm$.07	2.50$\pm$.05	93.72$\pm$.20	0.18$\pm$.03	0.21$\pm$.00	7.15$\pm$.06	1.80$\pm$.03
ODMO _{NE}	96.36 $\pm$.21	0.17 $\pm$.01	7.07 $\pm$.12	2.52 $\pm$.05	91.52 $\pm$.50	0.24 $\pm$.01	0.26 $\pm$.01	7.13 $\pm$.07	1.89 $\pm$.04
$-\mathcal{L}_{es}(\theta)$	75.07 $\pm$ 1.45	0.82 $\pm$.11	6.84 $\pm$.06	4.12 $\pm$.10	84.04 $\pm$.45	0.38 $\pm$.03	0.35 $\pm$.03	6.97 $\pm$.10	2.51 $\pm$.07
$-\text{CMD}$	91.37 $\pm$ 1.17	0.51 $\pm$.10	6.99 $\pm$.07	2.87 $\pm$.08	90.68 $\pm$.27	0.28 $\pm$.01	0.30 $\pm$.01	7.12 $\pm$.07	1.78 $\pm$.04

Table 5.2: Benchmark on CMU MoCap Dataset based on classifier

CMU MoCap				
Methods	Acc $\uparrow$	FID $\downarrow$	Diversity	MModality
Real motions	98.54 $\pm$.14	0.02 $\pm$.00	6.57 $\pm$.11	2.17 $\pm$.08
DLow-Gen	14.35 $\pm$.17	10.88 $\pm$.05	5.13 $\pm$.08	5.06 $\pm$.17
Act-MoCoGAN	14.93 $\pm$.09	37.15 $\pm$.12	0.90 $\pm$.04	0.84 $\pm$.04
A2M-xyz	76.17 $\pm$.67	1.32 $\pm$.05	6.33 $\pm$.07	3.27 $\pm$.08
A2M-Lie	76.33 $\pm$.66	1.42 $\pm$.06	6.28 $\pm$.08	3.35 $\pm$.07
ODMO	97.74$\pm$.17	0.09$\pm$.00	6.63$\pm$.10	2.07$\pm$.10
ODMO _{NE}	87.30 $\pm$.51	0.87 $\pm$.04	6.33 $\pm$.10	2.68 $\pm$.08
$-\mathcal{L}_{es}(\theta)$	69.90 $\pm$ 2.97	2.01 $\pm$.24	6.10 $\pm$.11	3.65 $\pm$.14
$-\text{CMD}$	64.27 $\pm$ 2.47	4.20 $\pm$.36	5.97 $\pm$.08	3.31 $\pm$.11

5.5.4 Ablation study

We conducted two ablation experiments on the ODMO model as shown in the bottom rows of Tables 5.1 and 5.2: (1) $-\mathcal{L}_{es}(\theta)$: removing the contrastive loss. (2) $-\text{CMD}$: removing causal motion decomposition i.e. feeding the original non-decomposed motion into the encoder; All

models are trained using the same hyper-parameters as in ODMO.

Contrastive Learning ($-\mathcal{L}_{es}(\theta)$): Without contrastive loss applied in the latent space, ODMO becomes a classic variational auto-encoder and its performance downgrades. Thus, separating out the codes of different action categories is an essential component of ODMO.

Causal Motion Decomposition ($-CMD$): Causal motion decomposition is one of the key designs in ODMO. The decomposition of motion into local motion profile and trajectory encourages the model to learn the kinematics of body movement. The advantage of causal decomposition should be more pronounced when a human motion has long spatial translation: for un-decomposed motion, the loss function will focus more on the spatial translation while losing details of local movement, leading to unrealistic artifacts like foot skating.

Indeed, HumanAct12 and UESTC lack long spatial translations, and as expected, the models trained with un-decomposed motion can still achieve acceptable performance. However, the advantage of the causal motion decomposition is much more pronounced in the MoCap dataset, where the motions are more spatially dynamic.

5.5.5 Customization

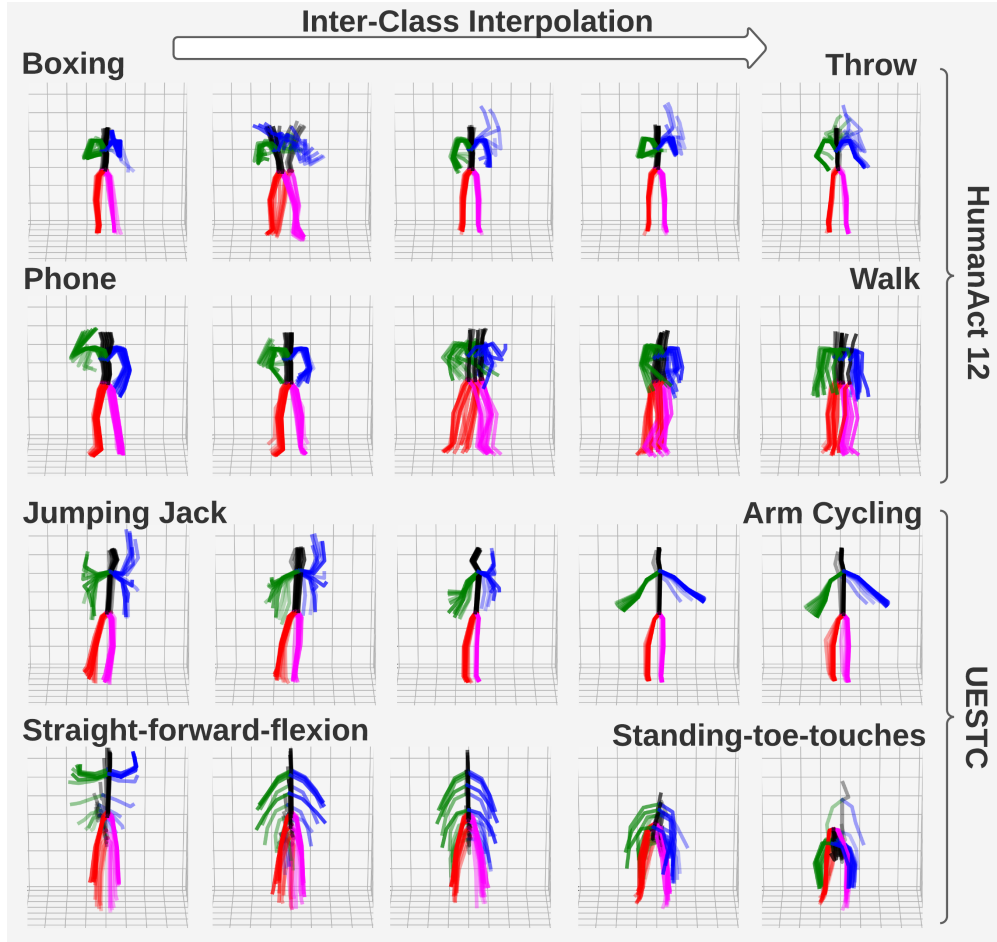


Figure 5.4: **Inter-Action category interpolations** (HumanAct12 and UESTC datasets): For a chosen category pair (A, B) , we sampled two latent codes: one from the category A , and second from the category B , and interpolated codes between these two. For each such category pair, the time-lapsed images of motion sequences generated by ODMO_{NE} from the two originally sampled codes and two other interpolated codes are visualized along a row. Motion transition is clearly visible in the interpolated codes.

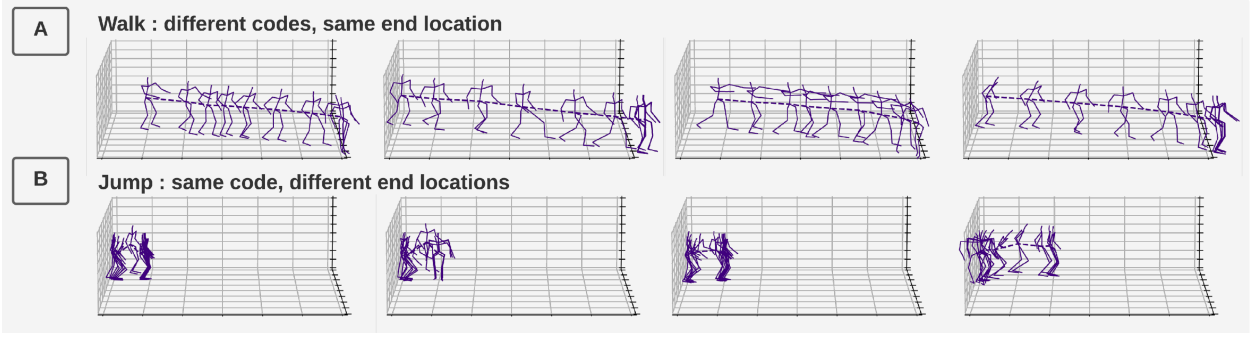


Figure 5.5: **Customization:** (A) Walk sequences generated by the ODMO_{MM} MoCap model with the same end location but with codes sampled from different modes. (B) Generated Jump motion with the same code but with end locations ranging from left to right (horizontally).

5.5.5.1 Mode discovery and multi-modal sampling

The ODMO’s latent space enables us to discover modes within each motion category. Figure 5.3 illustrates a few example modes discovered by our adaptive GMM estimation methodology. Once discovered, each mode can be sampled separately to generate motion sequences of a desired type. In such multi-modal sampling, if we sample the same number of motion in each mode, then we should expect an increase in the diversity of motion within each class. This is because, if the same number of codes are sampled directly from the entire action category (modeled by a single Gaussian) then modes closer to the center of the Gaussian will be over represented and modes further away will be under sampled, leading to less diversity. We name our models using such multi-modal sampling as ODMO_{MM} . The accuracy and diversity using both (i) the multi-modality of this task using classifier, and (ii) a more straightforward measurement, Average Pairwise Distance (APD) [122], are shown in Table 5.3. The APD is computed as the average pairwise Euclidean distance over N samples belonging to the same category: $\frac{1}{N(N-1)} \sum_{i=1}^N \sum_{j \neq i}^N (\sum_{t=1}^T \|\mathbf{m}_t^{(i)} - \mathbf{m}_t^{(j)}\|^2)^{\frac{1}{2}}$ and $\frac{1}{N(N-1)} \sum_{i=1}^N \sum_{j \neq i}^N (\sum_{t=1}^T \|\mathbf{p}_t^{(i)} - \mathbf{p}_t^{(j)}\|^2)^{\frac{1}{2}}$ for undecomposed movements and local movements profile. The APD of the real motion for different categories differ considerably, potentially

reflecting the different intensity levels of corresponding movements. We thus normalize each APD category score with the corresponding APD score of the real motion, and averaged these normalized scores across categories to obtain APD, APD_{LMP} . As shown in Table 5.3, multi-modal sampling model demonstrates a clear gap compared to all other models, including the standard sampling counterpart.

Table 5.3: HumanAct12 Dataset Benchmark: ODMO_{MM} (Multi-modal sampling version of ODMO) has higher accuracy than baselines, while generating highest diversity when measured by both metrics: Multimodality in classifier and APD.

Methods	Acc $\uparrow$	MModality	APD $\uparrow$	APD_{LMP} $\uparrow$
Real Motions	$99.70^{\pm.06}$	$2.53^{\pm.04}$	$1.00^{\pm.00}$	$1.00^{\pm.00}$
A2M-xyz	$69.17^{\pm.55}$	$4.18^{\pm.09}$	$0.79^{\pm.00}$	$0.81^{\pm.00}$
A2M-Lie	$93.25^{\pm.47}$	$2.77^{\pm.07}$	$0.87^{\pm.01}$	$0.84^{\pm.01}$
ACTOR	$96.00^{\pm.23}$	$2.59^{\pm.07}$	$0.86^{\pm.00}$	$0.85^{\pm.00}$
ODMO	$98.57^{\pm.32}$	$2.50^{\pm.05}$	$0.97^{\pm.01}$	$0.92^{\pm.00}$
ODMO_{MM}	$97.65^{\pm.30}$	$2.62^{\pm.05}$	$1.01^{\pm.01}$	$0.96^{\pm.01}$

5.5.5.2 Intra-action category interpolation

Once we discover the modes inside one action category, we can interpolate between codes belonging to different mode clusters to generate new samples not present in the training dataset. As shown in Figure 5.3, we can observe the styles morphing from one type of Warm up to another. Motion style transfer [145] itself is an active research area and requires human labeling of the styles for training. Thus, besides using this customization for animation task, our systematically generated data can be used as data augmentation for style transfer tasks.

5.5.5.3 Inter-action category interpolation

We further demonstrate the utility of having distinct code regions for different action categories by interpolating codes belonging to different categories. Figure 5.4 provides illustrations of such interpolations, where motion transition is clearly visible in the interpolated samples; more examples are shown in the supplementary material.

5.5.5.4 Trajectory customization

Programmable end locations are highly desirable for animation tasks and can be further combined with route planning for more complex motion generation in any given scenario. Here we explore the capability to generate motion sequences with programmable end locations. There are two fundamental requirements for this task: generated motion needs to be realistic and achieves its end goal, i.e., the generated motion can indeed move to the target location. We use classifier’s accuracy to measure the goodness of generated motion. To measure success in reaching the desired end location ($\mathbf{r}_e$), we compute the a goal-error metric: $\frac{\|\mathbf{r}_e - \mathbf{r}_m\|}{\|\mathbf{r}_e\|}$, where $\mathbf{r}_m$ is the end location of the generated motion. To allow some spatial translations, we set $\|\mathbf{r}_e\|$ always greater than 0.2.

We picked the MoCap dataset as it has much longer overall spatial translations than the other two datasets. Qualitative evaluations of two cases of customization are shown in Figure 5.5. For quantitative evaluation at the population level, we randomly sampled 20 codes for each motion category, and for each code we randomly sampled 20 $\mathbf{r}_e$ within that category. These 400 pairs of code and $\mathbf{r}_e$ were sent to the downstream modules to generate corresponding motions. The accuracy of the generated motion sequences is $93.54\% \pm 0.3$ with an average goal-error 0.18 ± 0.004 .

Table 5.4: Motion improvement

Methods	Acc $\uparrow$	FID $\downarrow$
A2M-xyz	$69.17^{\pm.55}$	$1.73^{\pm.06}$
A2M-Lie	$93.25^{\pm.47}$	$0.21^{\pm.01}$
ODMO _{A2M-xyz}	$88.55^{\pm.40}$	$0.88^{\pm.00}$
ODMO _{A2M-Lie}	$98.61^{\pm.30}$	$0.07^{\pm.00}$

5.5.5.5 Motion improvement

ODMO can also be utilized to further improve motion sequences produced by other generative models. By feeding a generated motion through the LMP encoder and the downstream generators, a refined motion is produced. *We conducted a systematic experiment on the motion sequences generated by the Action2Motion (A2M) models (which have room for improvement) with HumanAct12 dataset.* UESTC dataset is not challenging since ODMO, A2M and ACTOR get equally good results, while MoCap dataset is too difficult for A2M to get recognizable refinements over itself. As shown in Table 5.4, the refined motions from both models get higher accuracy and lower FID. Visual inspections show that two common types of artifacts in motion sequences generated by the A2M models –flickering and frozen feet– are diminished. One explanation can be that our trained LMP encoder denoises input motion and maps it into a region in the latent space occupied by codes from ground truth motions that are similar to the noisy input motion. Once mapped, our generator reconstructs a motion from the mapped code with high quality.

5.6 Concluding Remarks

We propose a novel framework, On-Demand Motion Generation (ODMO), for generating realistic, diverse and customised long-term 3D human motion sequences conditioned only on action type. While recent work [6, 129] has focused on this task, customization capabilities such as the following have not been investigated: (i) ability to automatically discover different

modes/styles of motion within each action type in the training set, and then generate samples from these modes on demand; (ii) ability to interpolate modes within an action type and even across action types to generate diverse motions not seen in the training set, and (iii) ability to customize body-center’s trajectories and destination locations of generated motion sequences. Such new capabilities would widen the spectrum of potential applications for 3D human motion generators.

To meet these objectives we introduce several innovations, including: (i) A causal decomposition of any 3D sample motion into two parts: spatial joint movements relative to a body center (root) – referred to as local movement profile (LMP) – and the resulting 3D trajectory of the root node. (ii) Introduction of contrastive learning with a modified VAE loss function that produces distinct clusters of latent codes for different action types in the latent space. With these innovations, ODMO outperforms the SOTA models on three datasets; generates motion sequences with long spatial translations; enables on-demand multi-modal sampling of different styles; generates new types of actions via interpolations; enables customization of trajectories via destination-location manipulations; and improves imperfectly generated motion sequences. *Extensive examples of customization are included in the appendix.*

5.7 Appendix

5.7.1 Overview

In this section, we demonstrate more experimental results in Section 5.7.2. Specifically, mode discovery and mode interpolations are in Section 5.7.2.2; more qualitative results of end location customization are in Section 5.7.2.3; and more qualitative results of interaction type interpolation are in Section 5.7.2.4. Moreover, the model architectures and more implementation details in Section 5.7.3. Dataset details are also introduced in Section 5.7.4. Furthermore, self-organizing and contrastive mechanism in the latent space are introduced in Sections 5.7.5 and 5.7.6.

5.7.2 More results

5.7.2.1 MM sampling for three datasets

Table 5.5: Benchmark for MM on three Dataset based on classifier.

HumanAct12					
Methods	Acc $\uparrow$	FID $\downarrow$	Diversity	MModality	
Real motions	99.70 $\pm$.06	0.02 $\pm$.01	7.08 $\pm$.06	2.53 $\pm$.05	
ODMO	98.57 $\pm$.32	0.07 $\pm$.00	6.99 $\pm$.07	2.50 $\pm$.05	
ODMO _{MM}	97.63 $\pm$.30	0.15 $\pm$.02	7.11 $\pm$.07	2.66 $\pm$.08	
ODMO _{NE}	96.36 $\pm$.21	0.17 $\pm$.01	7.07 $\pm$.12	2.52 $\pm$.05	
ODMO _{NE-MM}	94.98 $\pm$.70	0.16 $\pm$.01	7.05 $\pm$.08	2.71 $\pm$.03	
Mocap					
Methods	Acc $\uparrow$	FID $\downarrow$	Diversity	MModality	
Real motions	98.54 $\pm$.14	0.02 $\pm$.00	6.57 $\pm$.11	2.17 $\pm$.08	
ODMO	97.74 $\pm$.17	0.09 $\pm$.00	6.63 $\pm$.10	2.07 $\pm$.10	
ODMO _{MM}	94.26 $\pm$.33	1.02 $\pm$.06	6.44 $\pm$.05	2.63 $\pm$.09	
ODMO _{NE}	87.30 $\pm$.51	0.87 $\pm$.04	6.33 $\pm$.10	2.68 $\pm$.08	
ODMO _{NE-MM}	85.58 $\pm$.90	1.64 $\pm$.15	6.26 $\pm$.07	3.02 $\pm$.04	
UESTC					
Methods	Acc $\uparrow$	FID _{tr} $\downarrow$	FID _{test} $\downarrow$	Diversity	MModality
Real motions	99.79 $\pm$.05	0.01 $\pm$.00	0.05 $\pm$.00	7.20 $\pm$.06	1.61 $\pm$.02
ODMO	93.72 $\pm$.20	0.18 $\pm$.03	0.21 $\pm$.00	7.15 $\pm$.06	1.80 $\pm$.03
ODMO _{MM}	93.01 $\pm$.32	0.16 $\pm$.00	0.17 $\pm$.00	7.12 $\pm$.05	1.89 $\pm$.04
ODMO _{NE}	91.52 $\pm$.50	0.24 $\pm$.01	0.26 $\pm$.01	7.13 $\pm$.07	1.89 $\pm$.04
ODMO _{NE-MM}	89.86 $\pm$.16	0.21 $\pm$.00	0.22 $\pm$.01	7.07 $\pm$.06	2.12 $\pm$.03

5.7.2.2 Examples for mode discovery

Figure 5.4 provided visualizations of an action type for the HumanAct12 dataset. In this section we visualize the modes discovered inside action category Dumbbell-one-arm-shoulder-

pressing for the UESTC dataset in Figure 5.6. For this action type UESTC has 3 modes; for the Warm Up category for HumanAct12 it had 7 modes. In general, compared to the HumanAct12 dataset, our framework finds that UESTC dataset contains fewer modes per category. This finding is validated by visual inspection that shows: UESTC dataset has fewer variations of how the subjects perform any given activity.

5.7.2.3 End location customization

Figure 5.5 provided end location customization for Jump in MoCap dataset. In this section we visualize, in Figure 5.7, four more examples in different activities, where end locations of generated motion sequences are customized while keeping the code the same.

5.7.2.4 Inter-action type interpolation

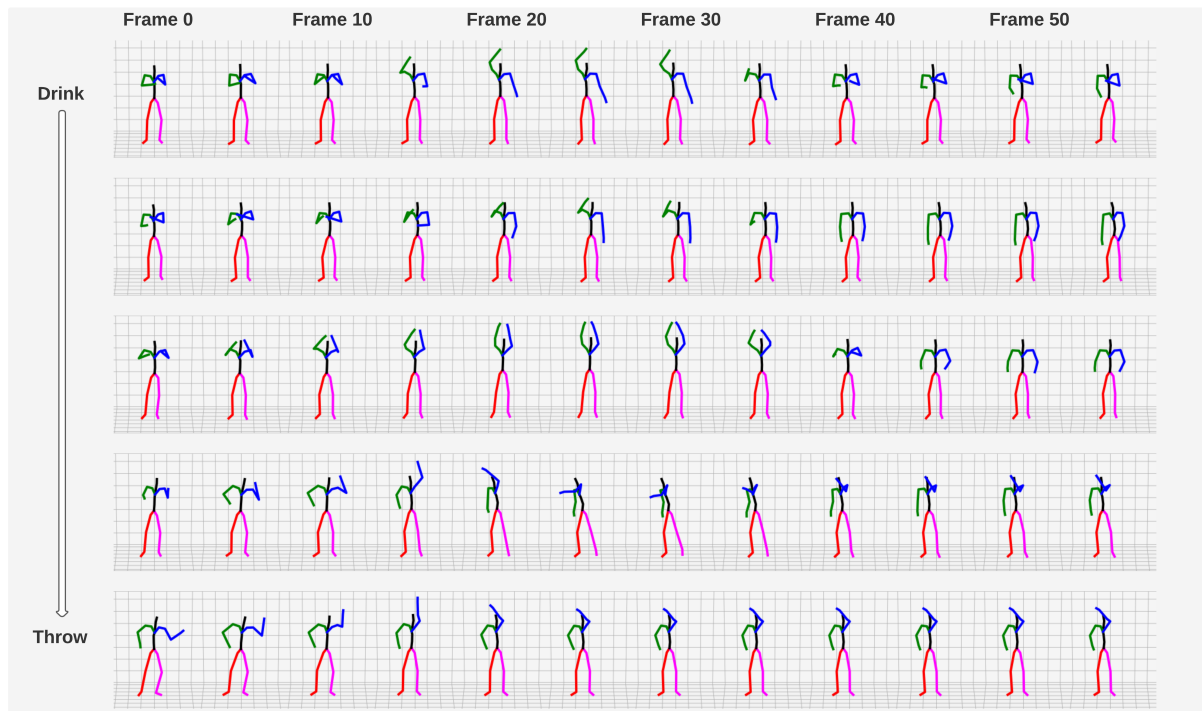
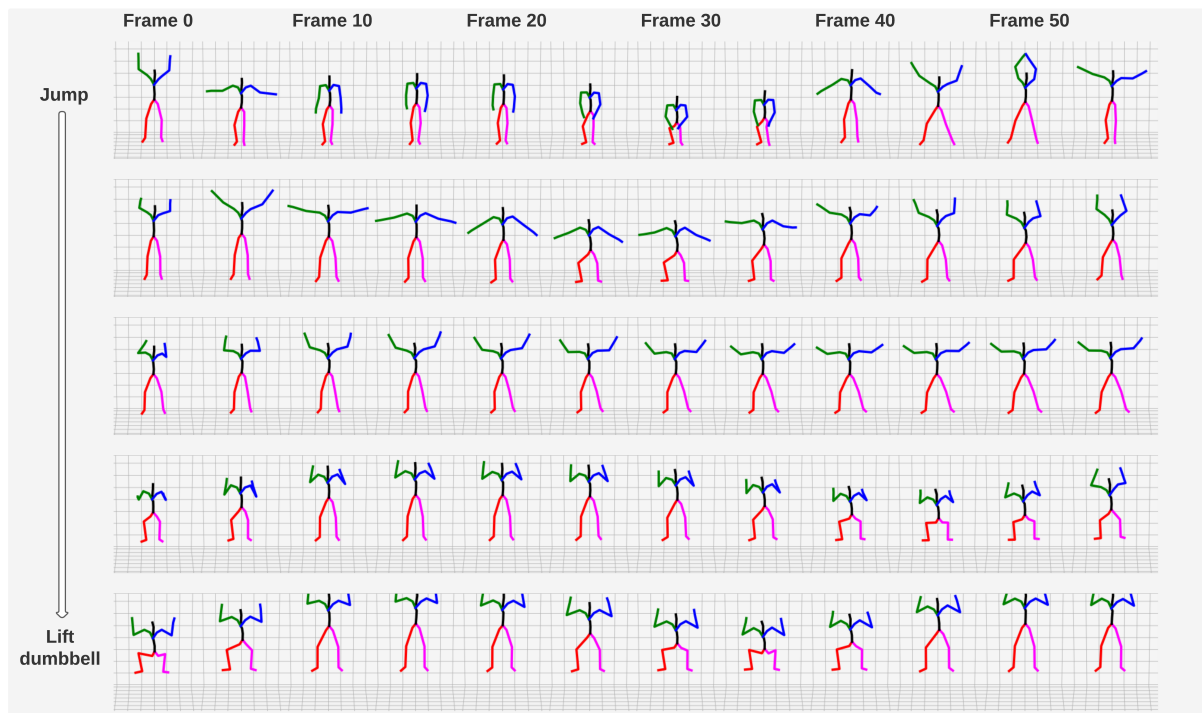




Figure 5.8: Inter-action type interpolation for HumanAct12 Dataset

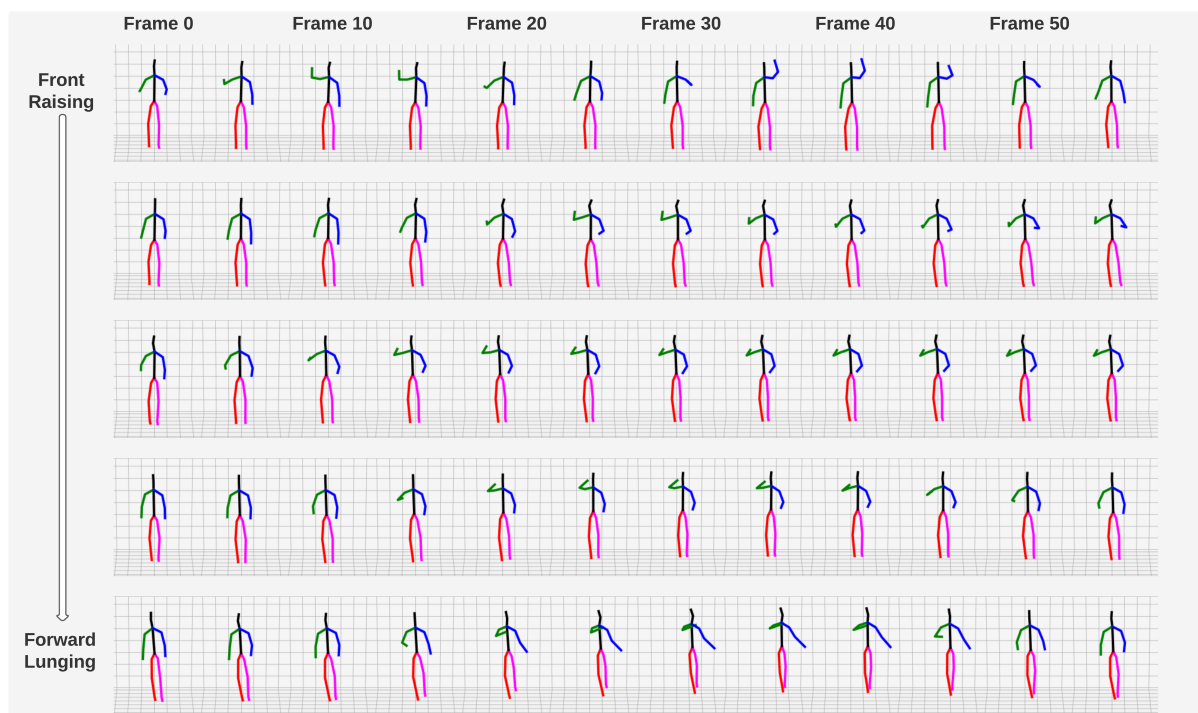




Figure 5.9: Inter-action type interpolation for UESTC Dataset

In Figures 5.8 and 5.9, we demonstrate the inter-action type interpolation examples for both UESTC and HumanAct12 datasets. We took the same procedure as in Figure 5.4: For a chosen category pair (A, B) , we sampled two latent codes: one from the category A , and second from the category B , and interpolated codes between these two. For each such category pair, the evenly sampled frames of motion sequences generated by ODMO_{NE} from the two originally sampled codes and three other interpolated codes are visualized along rows. Motion transition is clearly visible in the interpolated codes.

5.7.3 Implementation Details

In this section, we introduce the implementation details of ODMO and other baseline algorithms, MM sampling, and metrics.

5.7.3.1 ODMO Implementation Detail

We implemented the framework of ODMO using PyTorch and trained the models with TITAN RTX GPU. The detailed architecture for each component is shown in Figure 5.2 and Table 5.6. Note that the latent space dimension (the output of the LMP encoder) is 20 and the decoder only relied on this small code space and the one-hot encoded class information to reconstruct the motion. During training, the contrastive margin α is set to be 5, and the target variance of posteriors is set to 0.01. In the training of the network, we adopted the Adam optimizer with the learning rate 10^{-3} for both dataset. Additionally, teacher forcing (TF) is applied for enforcing the stable and efficiency of the training with the TF rate linear decayed from 1 to 0.3.

5.7.3.2 Multi modal sampling and silhouette score

Our adaptive approach to find the number of modes in activity category is done by using the silhouette score. We fit a GMM model with k mixture components and compute the silhouette score S_k . We vary k from 3 to 11 to get $S_3 \cdots S_{11}$. Starting from S_3 , we determine an i such that $S_i > 0.95 * S_{i+1}$ for the first time. Then the number of modes is selected as i . For UESTC dataset we dropped mixture components whose size is smaller than 20 for denoising propose.

5.7.3.3 Baseline Algorithms

Action2Motion (A2M): We use A2M as one of our major baselines for the three public datasets. We trained the A2M with joints' xyz coordinates representation (A2M xyz) and A2M using Lie group representation (A2M Lie) on MoCap dataset for seven motion categories with their default parameters. On HumanAct12 dataset, we directly employed their released model as baseline. On UESTC dataset, we also directly trained A2M xyz and A2M Lie with their default parameters.

Table 5.6: Architecture of ODMO

Module	Layers	details
LMP encoder	RNN layer	1 LSTM layer with 64 hidden units
	Feature layer	2 fully connected layers followed by BN and PReLU activation
	Output layer	1 fully connected layer
Trajectory generator	Embedding layer	2 fully conencted layers with PReLU activation in between
	RNN layer	2 LSTM layers with 128 hidden units
	Feature layer	3 fully conencted layers with PReLU activation
	Output layer	1 fully connected layer
Motion generator (Trajectory encoder)	Embedding layer	1 fully conencted layer with PReLU activation
	RNN layer	2 LSTM layers with 128 hidden units
Motion generator (Motion decoder)	Embedding layer	1 fully conencted layer with PReLU activation
	RNN layer	2 LSTM layers with 128 hidden units
	Output layer	1 fully connected layer

Table 5.7: Architecture of MoCap and HumanAct12 motion classifiers (same as Action2Motion)

Layers	Details
RNN layer	2 GRU layers with 128 hidden units
Feature layer	fully connected layer with Tanh activation
Output layer	1 fully connected layer

Table 5.8: Architecture of our UESTC classifier

Layers	Details
RNN layer	3 GRU layers with 256 hidden units
Feature layer	fully connected layer with Tanh activation
Output layer	1 fully connected layer

ACTOR: We treat ACTOR as the other strong baselines for UESTC and HumanAct12 dataset. We used their released model for final evaluation.

Other baselines: For implementation of Act-MoCoGAN, we followed their default architecture while replacing the image generator with pose generator and modified the image, video discriminators accordingly, as introduced in A2M paper. We trained the Act-MoCoGAN by tweaking the hyper-parameters so that it achieves the best generative results in the three dataset. As for the Gen-DLow, we adapted it to motion generation task by replacing the motion history condition as the class one-hot encoding. We trained the Gen-DLow (both VAE and DLow) in much more epochs (5000,600) than the default parameter (500, 500) so that it can be better to achieve the generative task.

5.7.3.4 Metrics Computation Details

Our metric computation follows the same procedure as Action2Motion(A2M) which includes sampling of ground truth and generated motion sequences, as well as computing fidelity and diversity (Multimodality) metrics based on them. However, since A2M and ACTOR used slightly different sampling strategies on generating the final evaluation metrics, for fair comparison, we recomputed their metrics with our sampling setup (400/250/200 samples per category for MoCap/HumanAct12/UESTC) instead of using the numbers reported in their papers.

Metrics based on classifier: Frechet Inception Distance(FID), Multimodality and Diversity are computed based on the extracted features, i.e the output of the feature layer (see Tables 5.7 and 5.8) in the classifier.

FID: We randomly sample (400/250/200 for MoCap/HumanAct12/UESTC) samples from ground truth motion and generated motion across all categories indepedently. Then we calculate the sample mean μ and covariance matrix Σ of the activation of the extracted features $x \in \mathcal{R}^{30}$ (the feature dimension in classifier according to Action2Motion) from the

set of real motion sequences as (μ_1, Σ_1) and the set of generated motion sequences as (μ_2, Σ_2) respectively. Lastly, FID is computed as $\|\mu_1 - \mu_2\|^2 + \text{Tr}(\Sigma_1 + \Sigma_2 - 2\sqrt{\Sigma_1 \Sigma_2})$.

Multimodality: To measure diversity inside one category, we randomly sample 20 pairs of motion sequences inside that category and extract their features as set $\mathcal{C}_1$, then compute the average pairwise Euclidean distance among these pairs, i.e. $\frac{1}{20} \sum_{(x_i, x_j) \in \mathcal{C}_1} \|x_i - x_j\|$.

Diversity: To measure the diversity of motion sequences across motion categories, we randomly sample 200 pairs of motion across all categories and extract their features as set $\mathcal{C}_2$, then calculate the average pairwise Euclidean distance, i.e. $\frac{1}{200} \sum_{(x_i, x_j) \in \mathcal{C}_2} \|x_i - x_j\|$.

Motion Classifiers: The metrics based on motion classifiers trained on real motion have been introduced in the A2M paper and we applied the same framework to compare performance of different models. For HumanAct12 dataset, we used the released classifier from A2M to make the results we benchmarked comparable. For MoCap dataset, we used 7 action categories instead of 8 action categories as in the A2M paper (Wash activity is dropped due to the noisy data), and thus we need to train our own classifier for our MoCap dataset. The architecture is shown in Table 5.7. Similarly, for UESTC dataset, a xyz representation based classifier is not available from previous works, so we trained our own classifier with architecture shown in Table 5.8.

5.7.4 Datasets

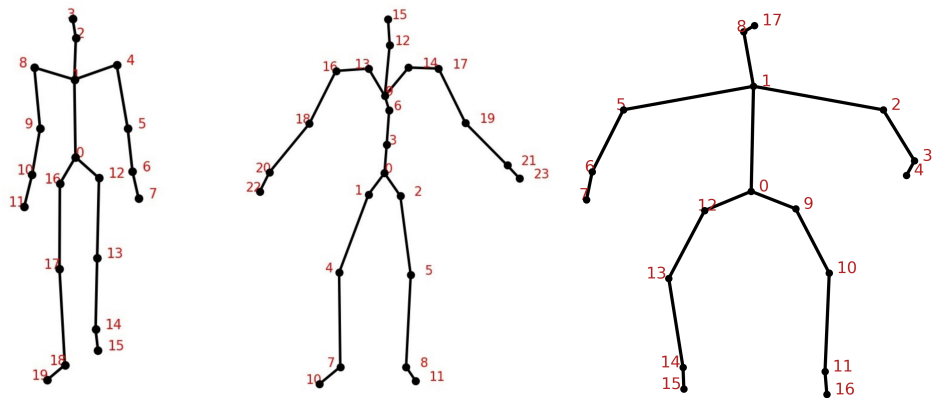


Figure 5.10: Human skeleton in the MoCap dataset (top left), HumanAct12 dataset (top right), and UESTC dataset (bottom) with the joint number annotated in red color.

Table 5.9: MoCap Dataset

Action type	Number of sequences
Walk	474
Run	109
Dance	77
Jump	108
Animal Behavior	101
Step	68
Climb	33
Total	970

Table 5.11: UESTC Dataset

Action type	Number of sequences
standing-gastrocnemius-calf	345

single-leg-lateral-hopping	345
high-knees-running	345
rope-skipping	343
standing-toe-touches	339
front-raising	285
straight-forward-flexion	285
standing-opposite-elbow-to-knee-crunch	285
dumbbell-side-bend	285
shoulder-raising	285
single-dumbbell-raising	285
wrist-circling	285
punching	285
pulling-chest-expanders	284
shoulder-abduction	281
overhead-stretching	270
head-anticlockwise-circling	270
deltoid-muscle-stretching	270
upper-back-stretching	270
spinal-stretching	268
alternate-knee-lifting	255
knee-to-chest	255
knee-circling	255
bent-over-twist	255
standing-rotation	255
pinching-back	240
dumbbell-shrugging	240
dumbbell-one-arm-shoulder-pressing	240

elbow-circling	240
arm-circling	235
raising-hand-and-jumping	225
forward-lunging	225
left-kicking	225
left-lunging	225
punching-and-knee-lifting	225
squatting	225
jumping-jack	225
marking-time-and-knee-lifting	225
rotation-clapping	225
left-stretching	224
Total	10629

We used CMU MoCap dataset, HumanAct 12 dataset, and UESTC dataset, which contain diversified 3D motion sequences with varied lengths for seven, twelve, and forty daily activities respectively. MoCap dataset represents 3D human skeleton as 21 joints and 20 bones; while HumanAct 12 dataset uses 24 joints and 23 bones; For UESTC dataset, there are 18 joints and 17 bones as shown in Figure 5.10. The detailed categories and number of motions belonging to each dataset are shown in Tables 5.9 to 5.11. During the training phase, in order to balance the training samples from different motion categories, we applied the stratified sampling strategy. Furthermore, since each real motion sequence from the training data has different lengths, in each training epoch, we randomly sample a consecutive motion segment with a desired length from each real motion sequence: 100 frames for MoCap and 60 frames for HumanAct12, and 60 frames for UESTC. If one motion sequence is shorter than the desired length, we pad it with the last frame until the desired length is reached. After obtaining the sub-sequences from the original data source, we shift it to

Table 5.10: HumanAct12 Dataset

Action type	Number of sequences
Warm up	215
Walk	47
Run	50
Jump	94
Drink	88
Lift dumbbell	218
Sit	54
Eat	77
Turn steering wheel	56
Phone	61
Boxing	140
Throw	91
Total	1191

ensure that the initial frame's root joint is at the origin.

5.7.5 Self organization effects in the latent space

Here we relax our assumption and assume that we can choose latent code $\mathbf{z}$ freely. Then we study how the latent codes are aligned with data features $\mathbf{x}$ to make the network's output $f(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})$ as close to ground truth $\mathbf{y}$ as possible. Formally, we shall make the assumption that the network f is a Lipschitz continuous function with the constant M . We also define the δ_x - δ_y -neighborhood of a data point $(\mathbf{x}, \mathbf{y})$ as $B(\mathbf{x}_1, \mathbf{y}_1; \delta_x, \delta_y) = \{(\mathbf{x}, \mathbf{y}) \mid \|\mathbf{x} - \mathbf{x}_1\| < \delta_x/M, \|\mathbf{y} - \mathbf{y}_1\| < \delta_y\}$. For later convenience, we define $\delta = \delta_x + \delta_y$. Finally, we denote $d(\mathbf{x}, \mathbf{y}, \mathbf{z}; \boldsymbol{\theta}) := \|f(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}) - \mathbf{y}\|$.

Proposition 1. *For any data point $(\mathbf{x}_1, \mathbf{y}_1)$, if $(\mathbf{x}_2, \mathbf{y}_2) \in B(\mathbf{x}_1, \mathbf{y}_1; \delta_x, \delta_y)$, then $|d(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}; \boldsymbol{\theta}) - d(\mathbf{x}_2, \mathbf{y}_2, \mathbf{z}; \boldsymbol{\theta})| < \delta_x + \delta_y = \delta$*

Proof. From $(\mathbf{x}_2, \mathbf{y}_2) \in B(\mathbf{x}_1, \mathbf{y}_1; \delta_x, \delta_y)$, follows that $\|\mathbf{x}_1 - \mathbf{x}_2\| < \delta_x/M$, $\|\mathbf{y}_1 - \mathbf{y}_2\| < \delta_y$. Therefore:

$$|\|f(\mathbf{x}_1, \mathbf{z}; \boldsymbol{\theta}) - \mathbf{y}_1\| - \|f(\mathbf{x}_2, \mathbf{z}; \boldsymbol{\theta}) - \mathbf{y}_2\|| \leq \|f(\mathbf{x}_1, \mathbf{z}; \boldsymbol{\theta}) - \mathbf{y}_1 - (f(\mathbf{x}_2, \mathbf{z}; \boldsymbol{\theta}) - \mathbf{y}_2)\| \quad (5.5a)$$

$$\leq \|f(\mathbf{x}_1, \mathbf{z}; \boldsymbol{\theta}) - f(\mathbf{x}_2, \mathbf{z}; \boldsymbol{\theta})\| + \|\mathbf{y}_1 - \mathbf{y}_2\| \quad (5.5b)$$

$$\leq M \|\mathbf{x}_1 - \mathbf{x}_2\| + \|\mathbf{y}_1 - \mathbf{y}_2\| \quad (5.5c)$$

$$< \delta_x + \delta_y = \delta. \quad (5.5d)$$

where Equations (5.5a) and (5.5b) follow reverse triangle inequality and triangle inequality, respectively. Besides, Equation (5.5c) follows from the Lipschitz continuity assumption. $\square$

Proposition 2. *Let $(\mathbf{x}_1, \mathbf{y}_1)$ be a data point, and $\mathbf{z}_i, \mathbf{z}_j$ be two choices of the latent code.*

If $\frac{d(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_i; \boldsymbol{\theta})}{d(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_j; \boldsymbol{\theta})} > C > 1$ for a constant C , and $d(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_j; \boldsymbol{\theta})$ is finite, then $\exists \delta_x, \delta_y$ such that

$$\forall (\mathbf{x}_2, \mathbf{y}_2) \in B(\mathbf{x}_1, \mathbf{y}_1; \delta_x, \delta_y), \quad \frac{d(\mathbf{x}_2, \mathbf{y}_2, \mathbf{z}_i; \boldsymbol{\theta})}{d(\mathbf{x}_2, \mathbf{y}_2, \mathbf{z}_j; \boldsymbol{\theta})} > 1$$

Proof. $\forall \eta > 0$, $\exists \delta_x, \delta_y$ such that $\delta_x + \delta_y = \delta < \eta d(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_j; \boldsymbol{\theta})$. Then using proposition 1, the following sequence of inequalities holds

$$d(\mathbf{x}_2, \mathbf{y}_2, \mathbf{z}_i) \geq d(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_i) - \delta \quad (5.6a)$$

$$> Cd(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_j) - \delta \quad (5.6b)$$

$$> Cd(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_j) - \eta d(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_j) \quad (5.6c)$$

$$= (C - \eta)d(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_j) \quad (5.6d)$$

$$> (C - \eta)(d(\mathbf{x}_2, \mathbf{y}_2, \mathbf{z}_j) - \delta) \quad (5.6e)$$

$$> (C - \eta) \left(1 - \frac{\eta}{1 - \eta} \right) d(\mathbf{x}_2, \mathbf{y}_2, \mathbf{z}_j) \quad (5.6f)$$

where the step from (5.6e) to (5.6f) is done by realizing the fact that

$$\eta d(\mathbf{x}_2, \mathbf{y}_2, \mathbf{z}_j) > \eta(d(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}_j) - \delta) > (1 - \eta)\delta$$

and therefore $\delta < \frac{\eta}{1 - \eta} d(\mathbf{x}_2, \mathbf{y}_2, \mathbf{z}_j)$.

It can be easily verified that there always exists a small enough value of $\eta \rightarrow 0^+$ such that

$$(C - \eta) \left(1 - \frac{\eta}{1 - \eta} \right) > 1$$

and therefore

$$d(\mathbf{x}_2, \mathbf{y}_2, \mathbf{z}_i) > d(\mathbf{x}_2, \mathbf{y}_2, \mathbf{z}_j).$$

□

Proposition 3. *Let $(\mathbf{x}_1, \mathbf{y}_1)$ be a data point and let $\mathbf{z}$ be a choice of the latent code, respectively. Also assume an improvement update of the network parameters $\boldsymbol{\theta}_1$ to $\boldsymbol{\theta}_2$ that ensures $d(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}; \boldsymbol{\theta}_1) > d(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}; \boldsymbol{\theta}_2)$.*

Then, there exists δ_x and δ_y such that

$$\forall (\mathbf{x}_2, \mathbf{y}_2) \in B(\mathbf{x}_1, \mathbf{y}_1; \delta_x, \delta_y), \quad d(\mathbf{x}_2, \mathbf{y}_2, \mathbf{z}; \boldsymbol{\theta}_2) < d_{(\delta_x, \delta_y)}^{\max}(\boldsymbol{\theta}_1),$$

where

$$d_{(\delta_x, \delta_y)}^{\max}(\boldsymbol{\theta}) = \max_{(\mathbf{x}, \mathbf{y}) \in B(\mathbf{x}_1, \mathbf{y}_1; \delta_x, \delta_y)} d(\mathbf{x}, \mathbf{y}, \mathbf{z}; \boldsymbol{\theta})$$

is the largest reconstruction distance for data points in the neighborhood of $(\mathbf{x}_1, \mathbf{y}_1)$

Proof. Let $a = d(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}; \boldsymbol{\theta}_1) - d(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}; \boldsymbol{\theta}_2) > 0$, and again $\delta = \delta_x + \delta_y$, then $\forall \delta_x, \delta_y$, and $\forall (\mathbf{x}_2, \mathbf{y}_2) \in B(\mathbf{x}_1, \mathbf{y}_1; \delta_x, \delta_y)$

$$d(\mathbf{x}_2, \mathbf{y}_2, \mathbf{z}; \boldsymbol{\theta}_2) \leq d_{(\delta_x, \delta_y)}^{\max}(\boldsymbol{\theta}_2) \tag{5.7a}$$

$$\leq d(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}; \boldsymbol{\theta}_2) + \delta \tag{5.7b}$$

$$\leq d(\mathbf{x}_1, \mathbf{y}_1, \mathbf{z}; \boldsymbol{\theta}_1) - a + \delta \tag{5.7c}$$

$$\leq d_{(\delta_x, \delta_y)}^{\max}(\boldsymbol{\theta}_1) - a + \delta \tag{5.7d}$$

where Equation (5.7b) follows from Proposition 1.

Since a is finite, $\exists \delta_x, \delta_y$ such that $\delta < a$, and hence

$$d(\mathbf{x}_2, \mathbf{y}_2, \mathbf{z}; \boldsymbol{\theta}_2) < d_{(\delta_x, \delta_y)}^{\max}(\boldsymbol{\theta}_1).$$

□

5.7.6 Contrastive mechanism in the latent space

5.7.6.1 VAE: From variational perspective to prior regularization

As mentioned in 2.3.2, from a Variational Bayesian perspective, the evidence low bound (ELBO) can be derived from data log likelihood $\log p_{\theta}(x)$ minus KL divergence between posterior and prior distribution. If we leave the variational framework aside, ELBO can also be viewed from a quite different optimization perspective: VAE is more like an autoencoder with additional sampling process in the latent space. We want the encoding decoding process have good reconstruction results, so a reconstruction loss term should be imposed. At the same time, we want some regularization on the encoder's output $\mu(x)$ and $\sigma(x)$ such that

there is less overfitting and more generalization. Without regularization of μ , the latent space can potentially be infinite and too sparse and specific for each datapoint. So we can rewrite ELBO as a combination of a reconstruction loss and a regularization term on $\mu(x)$ and $\sigma(x)$, i.e. the negative of log of prior distribution of these parameters.

We assume that for each data point x , the prior for encoded vector μ is independent over all components. For μ , that term would correspond to a prior of μ under normal distribution, i.e.

$$\frac{-1}{2}\mu_i(x)^2 \rightarrow \mu_i(x) \sim \mathcal{N}(\mu_i(x) \mid 0, 1) .$$

Similarly, for σ , its related regularization term would correspond to a prior of gamma distribution,

$$\frac{1}{2} \log (\sigma_i(x)^2) - \sigma_i(x)^2 \rightarrow \sigma_i(x)^2 \sim \Gamma \left(\sigma_i(x)^2 \mid \alpha = \frac{3}{2}, \beta = \frac{1}{2} \right)$$

5.7.6.2 Contrastive mechanism

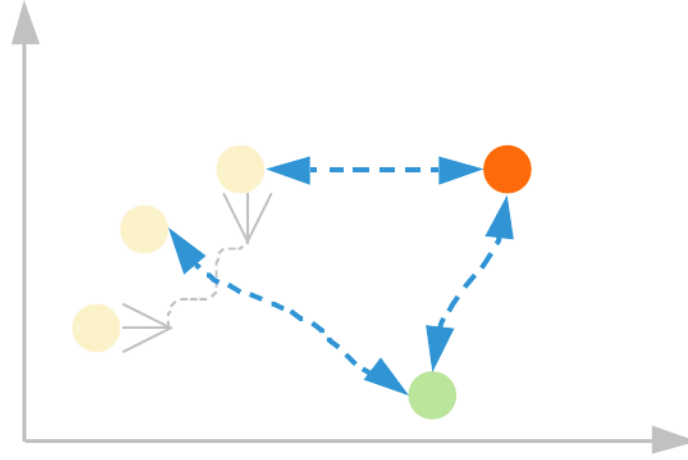


Figure 5.11: Introducing attractive and repulsive forces among latent codes belonging to the same or different categories in the dataset. The nodes in yellow are from the same category and a spring of zero stress length is introduced between them. This generates an attractive force (the outgoing arrows represent the forces bringing them together) encouraging them to cluster together. However, between two nodes belonging to different categories a compression-only spring of a finite zero-stress length is introduced. Thus, to minimize energy, there is a repulsive force when they come close together (the repulsive force is represented by arrows pointing into the nodes). The energy of the entire spring ensemble is used as a regularization term along with reconstruction loss to encourage separate clustering of latent codes for different categories.

Following this point of view, more complex prior of μ 's should be constructed to provide more modeling capacity in the latent space. To automatically form such a structure in the latent space, we deduced some general principles for the algorithm to follow. First, when the motion sequences are similar, their latent codes should be nearby, as shown by the light yellow dots in Figure 5.11. It leads to the assumptions that the latent codes corresponding to the same mode should cluster in the latent space. At the same time, the low-dimensional

representation of different motion sequences are far apart in the latent space, which means that each category should have its own cluster.

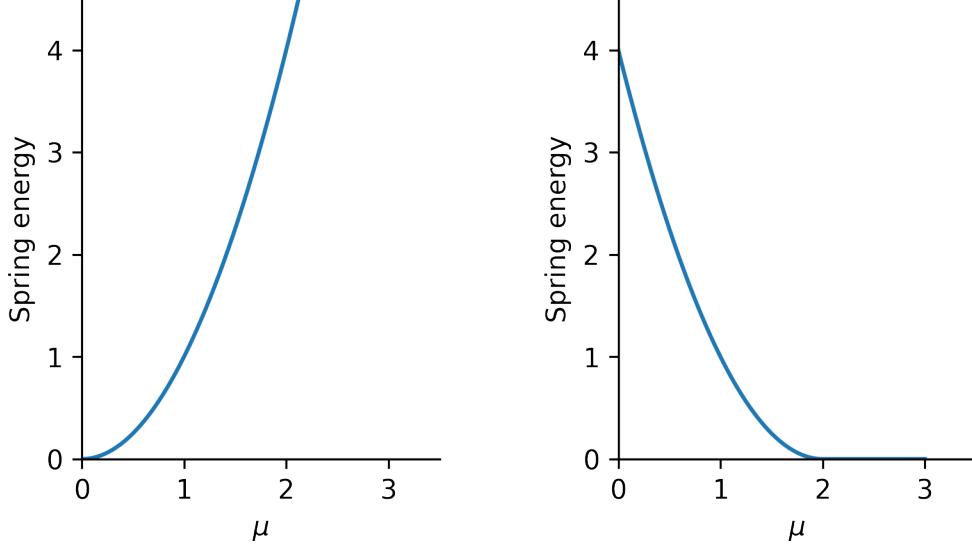


Figure 5.12: Energy for two different spring systems. Left: Spring for pairs of points belonging to the same class. Right: Compression only spring for points belonging to different classes.

Instead of using standard distributions, we form a distribution using self contrasting mechanism. We want to regularize μ jointly for n datapoints. The most convenient way is to view it as a spring ensemble system. The probability of one configuration $x : \mu_1, \dots, \mu_n$ is modeled as Boltzmann distribution, which is proportional to the exponential of the configurations energy, i.e. $p(x) \propto e^{-\beta E}$, where $E = \sum_{i=1}^n \sum_{j=1}^n \frac{1}{2} k(c_i, c_j) (\mu(\mu_i, \mu_j) - \mu_s(c_i, c_j))^2$. In different cases, we specify function k and stress free length μ_s accordingly.

For encoded two data points corresponding to the same activity, μ_i and μ_j , then we assign a spring whose stress free length is $\mu_s = 0$ and the dynamic length is $\mu = \|\mu_i - \mu_j\|$. We

assume the stiffness is 1, so its energy is $\frac{1}{2}(\mu - \mu_s)^2$, as shown in the left figure of Figure 5.12. Higher energy is contained by the pair of points with further distance. A spring like this is assigned to every pair of μ_i and μ_j belonging to the same category for all categories. Thus if we look at energy from one category, minimizing this part of the energy would be equivalent to minimizing the distance of each μ to the category center $\mu_c = \frac{\sum_{i=1}^n \mu_i^2}{n}$, as shown in Equation (5.8).

$$\sum_{i=1}^n \sum_{j=1}^n \frac{1}{2} \|\mu_i - \mu_j\|^2 = 2n \sum_{i=1}^n \frac{1}{2} \|\mu_i - \mu_c\|^2 \quad (5.8)$$

Considering the encoded two data points μ_i and μ_j corresponding to different activities, we define a compression-only spring with stress free length $\mu_s = \alpha$ (margin size), and the spring's dynamic length as $\mu = \|\mu_i - \mu_j\|$, similarly as in the previous case. For this compression only spring, the energy is $\frac{1}{2}(\mu - \mu_s)^2$ only if μ is less than α . The energy with margin $\alpha = 2$ is shown on the right of Figure 5.12.

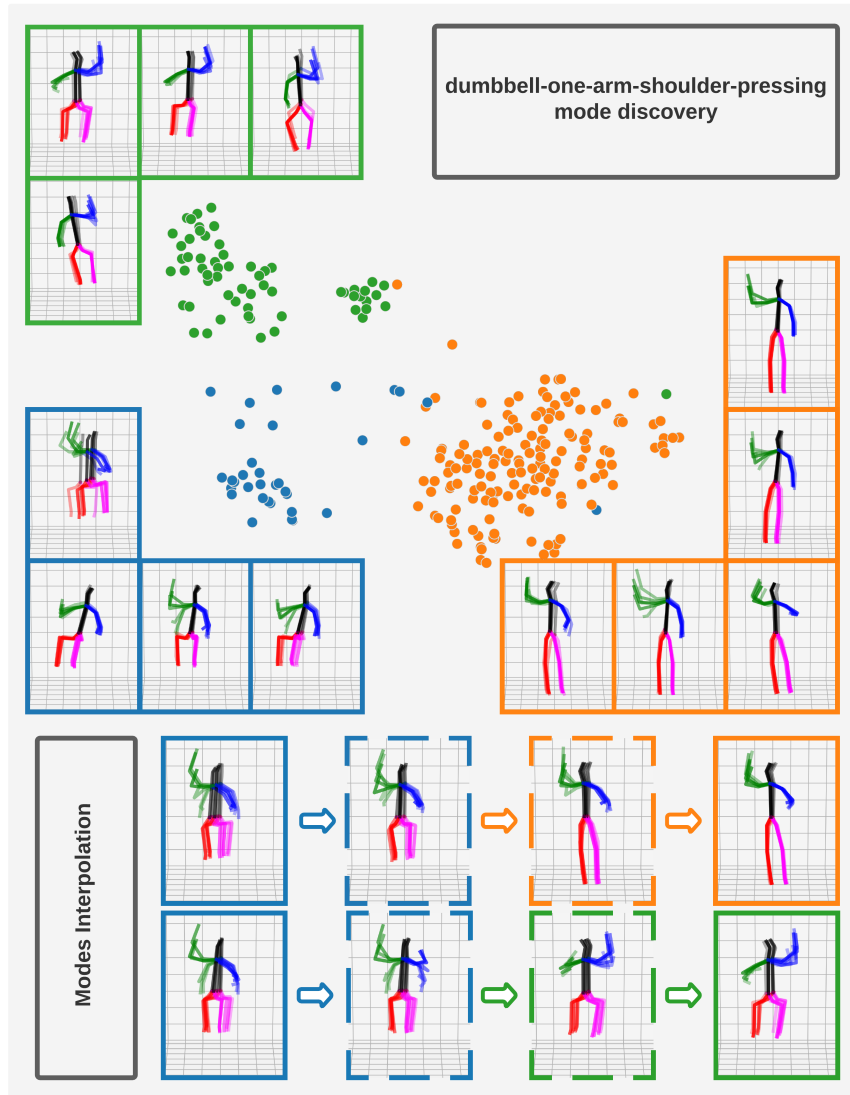


Figure 5.6: **Mode discovery in dumbbell-one-arm-shoulder-pressing and interpolation.** Latent codes are projected onto a 2D space using t-SNE to illustrate that the codes can be separated into distinct clusters. For each cluster we sampled a few codes and time-lapsed images summarizing the corresponding generated motions are shown. These time-lapsed images are highly consistent within clusters, providing evidence that each cluster represents a distinct mode of the action type. Together, the modes highlight the different dumbbell-one-arm-should-pressing routines present in the training dataset. As the bottom insets show, interpolations between codes from two modes show smooth morphing.

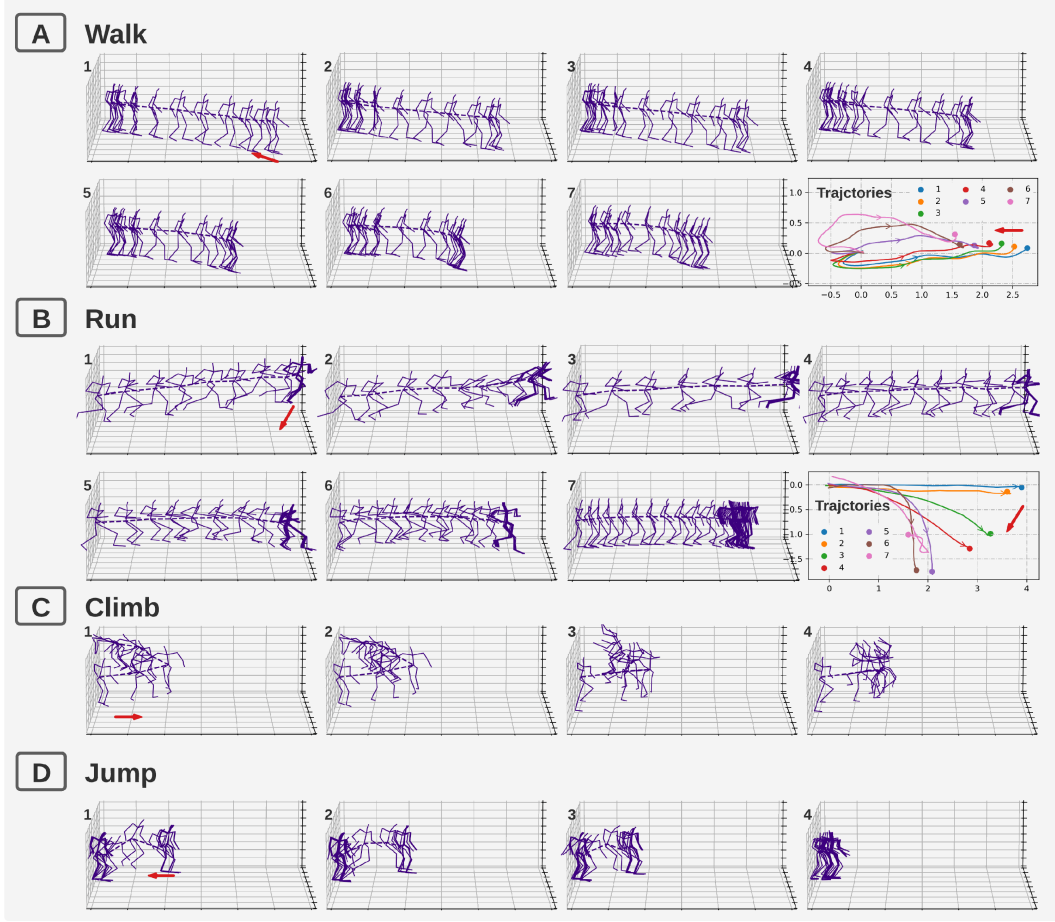


Figure 5.7: **End location customization on MoCap dataset.** In each activity category we sample a code in that category and determine its corresponding end location using KNN and use the ODMO model to generate a motion sequence, rendered in the inset No.1 for that category. Then we linearly vary the end location along the red arrow in the inset No.1. The generated motion sequences for each such specified end location, while keeping code the same, are shown in the following insets. The last inset for Walk and Run categories show the top view of the trajectories corresponding to different end points. It is remarkable to see how the trajectories for the same code got planned accordingly, in response to different end locations for the same latent code. In the Walk category, the trajectories changed course – preferring to split to the right from the humanoid’s perspective – as the end locations were brought nearer.

CHAPTER 6

Context Aware and Mode Preserving Generative Models (II): Imitation Learning on a Physical Platform

6.1 Introduction

Imitation learning is the task of replicating expert policy from demonstrations, without access to a reward function. This task becomes particularly challenging when the expert exhibits a mixture of behaviors. For example, in the case of an expert demonstrations of a humanoid moving in different directions, the action to be taken are highly diverse based on the direction that humanoid intends to move along. In the resulting imitation learning task, the learner has to jointly discover the different modes and the corresponding actions; this can easily lead to a mode collapse phenomenon, where the learner estimates an average model for multiple modes, resulting in poor imitation trajectories. This problem relates to our mode discovery and mode-preserving generation problem, except now it is in a real physical system.

Prior work has introduced latent variables to model the variations of the expert policy. However, our experiments show that the existing works do not exhibit appropriate imitation of individual modes. To tackle this problem, we adopt an encoder-free generative model for behavior cloning (BC) to accurately distinguish and imitate different modes. Then, we integrate it with GAIL to make the learning robust towards compounding errors at unseen states. We show that our method significantly outperforms the state of the art across multiple experiments [146]. Since it’s an ongoing project, we only demonstrate the

most related content here.

6.2 Related Work

Imitation of diverse behaviors is addressed in several prior works. [147, 148] use labels of expert modes in training. These works differ from ours in that we assume expert modes are unlabeled. InfoGAIL [46] and Intention-GAN [149] augment the objective of GAIL with the mutual information between generated trajectories and the corresponding latent codes. This mutual information is evaluated with the help of a posterior network. [47] use a variational autoencoder (VAE) module to encode expert trajectories into a continuous latent variable. These latent codes are supplied as an additional input to both the policy and the discriminator of GAIL. Nevertheless, as we show in our experiments, both InfoGAIL and VAE-GAIL perform poorly in practice.

Proposing a method for multimodal imitation learning requires finding the correspondence between expert trajectories and a latent variable. [47] suggest learning this correspondence with VAEs. However, this imposes the difficulty of designing an isolated (as opposed to end-to-end) recurrent module to encode sequences of state-action pairs. Moreover, VAEs are mainly well suited for continuous latent variables. We propose to address these challenges by removing the recurrent encoder module and instead directly searching for an optimal latent variable. [150, 151, 152] have recently considered similar encoder-free models for non-sequential data.

6.3 Approach

We assume there exist a latent variable $z \sim p(z)$, fully capturing the modes in the humanoid movements in the demonstrations. Following the notations introduced in 2.4.1, we denote agent’s action at time step t as a_t , state as s_t and its policy as $\pi(\cdot|s, z)$ conditioned on the la-

tent variable z , with transition probability controlled by environment as $P(s_{t+1}|a_t, s_t)$. Thus for the expert with policy π_E , the generative process of the trajectories can be formalized as $z \sim p(z)$, $a_t \sim \pi_E(\cdot|s_t, z)$, $s_{t+1} \sim P(s_{t+1}|a_t, s_t)$. For the agent to be trained, we implement its policy π_θ by a neural network parameterized by θ .

Given demonstrations from experts, we train our agent under both behavior cloning and GAIL frameworks. For either approach, inside one training step, we update the agent policy conditioned on the latent code (the trajectory’s mode) which is inferred by an algorithm that we refer to as self organized generative (SOG) model [146] at the same time. Detailed algorithms are shown in Algorithms 5 and 6.

Algorithm 5 our-BC: Multimodal Behavior Cloning

- 1: **Define:** Loss function $\mathcal{L}(\hat{\mathbf{a}}, \mathbf{a}) := ||\hat{\mathbf{a}} - \mathbf{a}||^2$
 - 2: **Input:** Initial parameters of policy network θ_0 ; expert trajectories $\tau_E \sim \pi_E$
 - 3: **for** $iteration = 0, 1, 2, \dots$ **do**
 - 4: Sample state-action pairs $\chi_E \sim \tau_E$ with the same batch size.
 - 5: Sample N_z latent codes $\mathbf{z}_i \sim p(\mathbf{z})$ for each trajectory.
 - 6: Find optimal code $\mathbf{z}_E$ among the sampled N_z latent codes by our SOG algorithm.
 - 7: Calculate $\mathcal{L}_{\text{code}} = \hat{\mathbb{E}}_{\tau_E} [\mathcal{L}(f_\theta(\mathbf{z}_E, \mathbf{s}), \mathbf{a})]$ and its gradients w.r.t. θ .
 - 8: Update f_θ by stochastic gradient descent on θ to minimize $\mathcal{L}_{\text{code}}$.
 - 9: **end for**
-

6.4 Experiment and Result

The humanoid modeled by MuJoCo[1] simulator contains a 376-dimensional state space, describing the position and velocity of each joint, contact forces, and friction of actuator and a 17-dimensional action space. One example can be seen in Figure 6.1. We trained an expert in this MuJoCo environment, which made a humanoid walk in 6 different directions. For each direction, we generate 500 trajectories of 1000 steps. This whole set of trajectories

Algorithm 6 our-GAIL: Multimodal Combination of BC and GAIL

- 1: **Input:** Initial parameters of policy and discriminator networks, $\theta_0, \mathbf{w}_0$; expert trajectories $\tau_E \sim \pi_E$
- 2: **for** $i = 0, 1, 2, \dots$ **do**
- 3: Sample a latent code $\mathbf{z}_i \sim p(\mathbf{z})$, and subsequently a trajectory $\tau_i \sim \pi_{\theta_i}(\cdot | \mathbf{z}_i)$.
- 4: Sample state-action pairs $\chi_i \sim \tau_i$ and $\chi_E \sim \tau_E$ with the same batch size.
- 5: Update the discriminator parameters from $\mathbf{w}_i$ to $\mathbf{w}_{i+1}$ with the gradient

$$\hat{\mathbb{E}}_{\tau_i} [\nabla_{\mathbf{w}} \log D_{\mathbf{w}}(\mathbf{s}, \mathbf{a})] + \hat{\mathbb{E}}_{\tau_E} [\nabla_{\mathbf{w}} \log (1 - D_{\mathbf{w}}(\mathbf{s}, \mathbf{a}))].$$

- 6: Calculate the surrogate loss $\mathcal{L}_{\text{PPO}}$ using the PPO rule with the following objective

$$\hat{\mathbb{E}}_{\tau_i} [\nabla_{\mathbf{w}} \log D_{\mathbf{w}}(\mathbf{s}, \mathbf{a})] - \lambda_H H(\pi_{\theta_i}).$$

- 7: Calculate the loss $\mathcal{L}_{\text{code}}$ per Algorithm 5.
 - 8: Take a policy step from θ_i to θ_{i+1} w.r.t. the objective $\mathcal{L}_{\text{PPO}} + \lambda_S \mathcal{L}_{\text{code}}$.
 - 9: **end for**
-

without any label was made available to our algorithm. In this setting, the task is to learn all the skills in the training data to make the humanoid move like the expert.

We compared the agent trained by our algorithms with the agents trained by InfoGAIL [46] and VAE-GAIL [47], two state-of-the-art benchmarks in multimodal imitation learning. As shown in Figure 6.2, our method is able to recover and distinguish all modes of expert behavior while other two baseline algorithms failed in this challenging task. Additionally, our algorithm replicates the expert’s multimodal behavior robustly as shown in Figure 6.3.

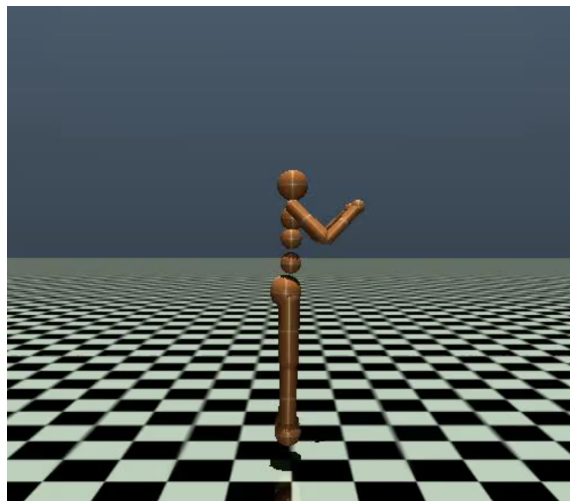


Figure 6.1: Example of humanoid in Humanoid-Dir-6 environment simulated via MuJoCo simulator [1].

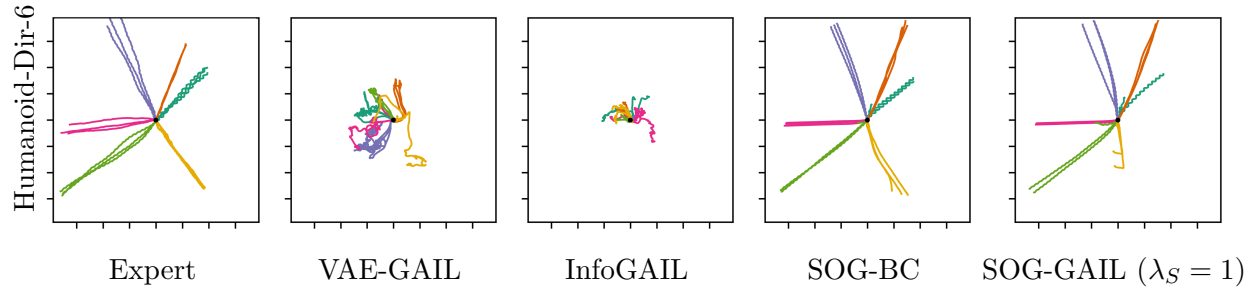


Figure 6.2: **Discrete latent variable: locomotion towards multiple directions.** Trajectories of the imitated policies for the walking task. Different latent codes at inference time are color coded. Number of modes is 6; the number of trajectories per latent code is 3. Both our-BC and our-GAIL precisely separate and imitate the modes in all the experiments. However, VAE-GAIL and InfoGAIL fail to separate the modes and to imitate the expert [146].

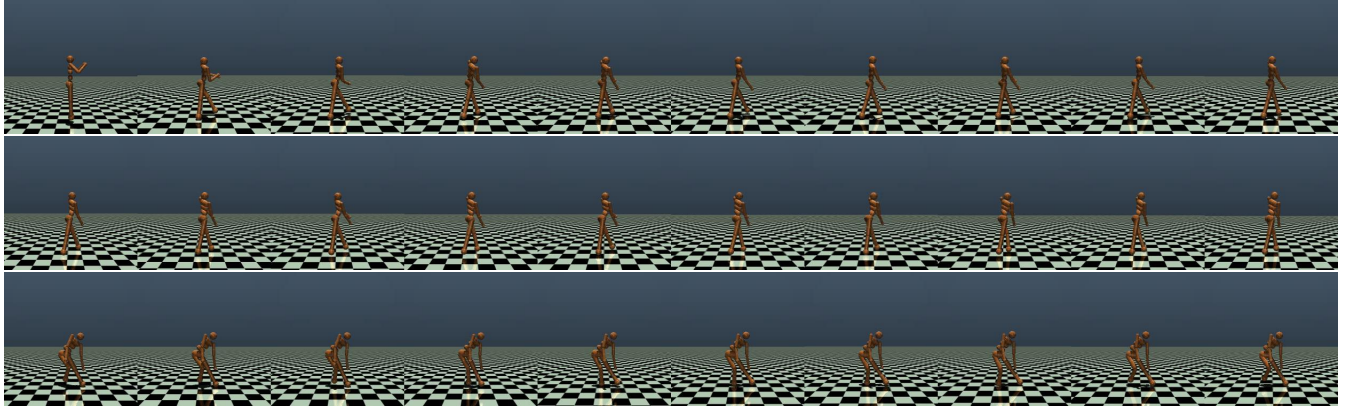


Figure 6.3: Sampled frames of Humanoid walking in three different directions (generated by our trained agent).

CHAPTER 7

Summary of contributions and future work

In this dissertation, we focus on improving the machine’s intelligence by efficiently learning from real-world high-dimensional data. We recognize that a core criterion of having intelligence is the ability to model the observable world in a reliable and predictive fashion such that reasonable actions can be taken to achieve certain desired goals. Humans have been largely successful in this regard, because a core assumption about the observable world we make has turned out to be mostly true: The real-world data –however high in dimension and however many different modes it has– is generated by a set of underlying low-dimensional world models that are indexed by a hierarchical set of contexts. The premise of this thesis is that for machine intelligence to be more robust and efficient, one needs to be able to make machines also think the same way, and one can narrow down the learning tasks for machines into three key components: (i) mode discovery (ii) mode recognition, and (iii) mode generation. This thesis has developed techniques to address these problems in different settings in the real world scenario. Specifically, we focus on automatically discovering modes and training mode-preserving generative models from real-world high-dimensional data with multimodal distributions and demonstrated how such mode discovery and modeling techniques can help in improving the recognition capacities of machines.

In the first part of the dissertation, we introduced the modeling of the underlying generative models that result in creating the real-world data with multiple modes and existing methods and frameworks for identifying and characterizing modes from data. In Chapter 1, we introduced the simplified generative model and definition of modes with examples from

related problems. In Chapter 2, related work for dimensionality reduction techniques, weakly supervised learning and typical generative frameworks is introduced to assist the reader in following the rest of the chapters.

In the second part of the dissertation, we tackle two real-world scenarios: (i) Deriving low-dimensional representations for high-dimensional data: this serves as one of the fundamental steps for the following data analysis and mode discovery task in an ML pipeline; and ii) mode separation when only coarse level and noisy labels are available. Such a scenario is representative of many medical decision making problems currently encountered by clinical experts. In Chapter 3, we proposed an efficient algorithm for interpretable dimensionality reduction and demonstrated its usage in various types of real-world data. In Chapter 4, we performed mode separation in the Electroencephalogram (EEG) of Epilepsy Patients to discover Epileptogenic HFOs with weak supervision and validated our discovery with clinical interpretations.

In the last part of the dissertation, we focus on context aware and mode preserving generative models. We first tackled an action-type conditioned 3D motion generation task which aims for generating realistic and diverse motion sequences that mimics 3D human motion data. It has wide applications in entertainment community. Followed by this, we tackle the problem of creating generative models for scenarios where the observable data is created by an agent that interacts sequentially with real physical environments. It requires the learning of long-range dependencies in the data and dynamic modeling of the environment. In Chapter 5, we proposed a generative model that automatically discovers modes and allows motion customization such as interpolation between different action types and trajectory customization. In Chapter 6, we show how one can create generative models that can not only imitate diverse behavioral data, but can do so robustly while interacting with an external environment.

Clearly, we have only scratched the surface of the general problem of being able to efficiently learn compact and integrated models from data distributions with multiple modes

and under contexts that partially overlap and form a loose hierarchy. Some of the directions for continuing and future work are itemized below:

- Very concrete extensions of the work in this thesis can be phrased as follows: In the case of human motion generation task, can the machine generate reasonable trajectories and movements that are aware of the obstacles of the environment (which is a kind of context)? Can the generative model become collaborative such that two or more persons' trajectories can be planned without obvious conflicts, such as not crashing into each other or coordinating their motions to achieve a collective task of lifting an awkwardly shaped object?
- Such concrete problems are special cases of more general scenarios. For example, one needs to build more complex context models that can capture the organically hierarchical generative structure underlying data observed in real world. For example, one can utilize graphical models to model the hierarchical contexts in multiple scenarios.
- Along similar lines, a key part of natural intelligence is to have an integrated set of models that share sub-models and use environmental cues to piece together a novel strategy from existing models to deal with new scenarios. This new model and strategy then gets its own place in the repertoire of models available to the agent. Thus, instead of building models separately for each task, one needs to integrate models learned from different scenarios and structure them into a framework for storing knowledge. Interpretability, a missing component of the black-box dominated AI/ML scene, can potentially be achieved by such a knowledge representation layer.
- Similarly, building collaborative generative models involving multiple agents is a key area of ongoing investigations. Interactions between independent generative models can be modeled separately to capture the dynamics between agents such that reasonable actions can be taken for all the agents together.

REFERENCES

- [1] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.
- [2] Stevan Harnad. The annotation game: On turing (1950) on computing, machinery, and intelligence. In *The Turing test sourcebook: philosophical and methodological issues in the quest for the thinking computer*. Kluwer, 2006.
- [3] Thomas M Mitchell et al. Machine learning. 1997.
- [4] Douglas R. Harper. data.
- [5] Andrei N Kolmogorov. On tables of random numbers. *Theoretical Computer Science*, 207(2):387–395, 1998.
- [6] Chuan Guo, Xinxin Zuo, Sen Wang, Shihao Zou, Qingyao Sun, Annan Deng, Minglun Gong, and Li Cheng. Action2Motion: Conditioned Generation of 3D Human Motions. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2021–2029, 2020.
- [7] Shihao Zou, Xinxin Zuo, Yiming Qian, Sen Wang, Chi Xu, Minglun Gong, and Li Cheng. Polarization Human Shape and Pose Dataset. *arXiv preprint arXiv:2004.14899*, 2020.
- [8] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.
- [9] Charles Fefferman, Sanjoy Mitter, and Hariharan Narayanan. Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049, 2016.
- [10] Marissa Connor, Gregory Canal, and Christopher Rozell. Variational autoencoder with learned latent structure. In *International Conference on Artificial Intelligence and Statistics*, pages 2359–2367. PMLR, 2021.
- [11] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. *Advances in neural information processing systems*, 29:613–621, 2016.
- [12] Zhongyue Huang, Jingwei Xu, and Bingbing Ni. Human Motion Generation via Cross-Space Constrained Sampling. In *IJCAI*, pages 757–763, 2018.
- [13] Jingwei Xu, Bingbing Ni, and Xiaokang Yang. Progressive Multi-granularity Analysis for Video Prediction. *International Journal of Computer Vision*, 129(3):601–618, 2021.

- [14] Carlos Oscar Sánchez Sorzano, Javier Vargas, and A Pascual Montano. A survey of dimensionality reduction techniques. *arXiv preprint arXiv:1403.2877*, 2014.
- [15] JC Gower. Dijksterhuis, gb: Procrustes problems, 2004.
- [16] Matej Artac, Matjaz Jogan, and Ales Leonardis. Incremental pca for on-line visual learning and recognition. In *Object recognition supported by user interaction for service robots*, volume 3, pages 781–784. IEEE, 2002.
- [17] C. Chatterjee and V.P. Roychowdhury. On self-organizing algorithms and networks for class-separability features. *IEEE Transactions on Neural Networks*, 8(3):663–678, 1997.
- [18] Chanchal Chatterjee, Zhengjiu Kang, and Vwani P Roychowdhury. Algorithms for accelerated convergence of adaptive pca. *IEEE Transactions on Neural Networks*, 11(2):338–355, 2000.
- [19] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.
- [20] Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2):265–286, 2006.
- [21] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.
- [22] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
- [23] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National science review*, 5(1):44–53, 2018.
- [24] Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1-2):31–71, 1997.
- [25] James Foulds and Eibe Frank. A review of multi-instance learning assumptions. *The knowledge engineering review*, 25(1):1–25, 2010.
- [26] Guoqing Liu, Jianxin Wu, and Zhi-Hua Zhou. Key instance detection in multi-instance learning. In *Asian Conference on Machine Learning*, pages 253–268. PMLR, 2012.
- [27] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *arXiv preprint arXiv:2007.08199*, 2020.

- [28] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence*, 38(3):447–461, 2015.
- [29] Umaa Rebbapragada and Carla E Brodley. Class noise mitigation through instance weighting. In *European conference on machine learning*, pages 708–715. Springer, 2007.
- [30] Frederik O Kaster, Bjoern H Menze, Marc-André Weber, and Fred A Hamprecht. Comparative validation of graphical models for learning tumor segmentations from noisy manual annotations. In *International MICCAI Workshop on Medical Computer Vision*, pages 74–85. Springer, 2010.
- [31] Haw-Shiuan Chang, Erik Learned-Miller, and Andrew McCallum. Active bias: Training more accurate neural networks by emphasizing high variance samples. *Advances in Neural Information Processing Systems*, 30:1002–1012, 2017.
- [32] HaiYang Zhang, XiMing Xing, and Liang Liu. Dualgraph: A graph-based method for reasoning about label noise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9654–9663, 2021.
- [33] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91(4):045002, 2019.
- [34] Frank Noé, Simon Olsson, Jonas Köhler, and Hao Wu. Boltzmann generators: Sampling equilibrium states of many-body systems with deep learning. *Science*, 365(6457), 2019.
- [35] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
- [36] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [37] John Duchi. Derivations for linear algebra and optimization. *Berkeley, California*, 3(1):2325–5870, 2007.
- [38] Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016.
- [39] Eric Nalisnick, Lars Hertel, and Padhraic Smyth. Approximate inference for deep latent gaussian mixtures. In *NIPS Workshop on Bayesian Deep Learning*, volume 2, page 131, 2016.

- [40] Jakub Tomczak and Max Welling. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223. PMLR, 2018.
- [41] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *arXiv preprint arXiv:1811.06711*, 2018.
- [42] Dean A Pomerleau. Efficient Training of Artificial Neural Networks for Autonomous Navigation. *Neural computation*, 3(1):88–97, 1991.
- [43] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.
- [44] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum Entropy Inverse Reinforcement Learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.
- [45] Jonathan Ho and Stefano Ermon. Generative Adversarial Imitation Learning. *arXiv preprint arXiv:1606.03476*, 2016.
- [46] Yunzhu Li, Jiaming Song, and Stefano Ermon. Infogail: Interpretable Imitation Learning from Visual Demonstrations. *arXiv preprint arXiv:1703.08840*, 2017.
- [47] Ziyu Wang, Josh Merel, Scott Reed, Greg Wayne, Nando de Freitas, and Nicolas Heess. Robust Imitation of Diverse Behaviors. *arXiv preprint arXiv:1707.02747*, 2017.
- [48] Ian Jolliffe. Principal component analysis. In *International encyclopedia of statistical science*, pages 1094–1096. Springer, 2011.
- [49] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.
- [50] Riccardo Boscolo, Hong Pan, and Vwani P Roychowdhury. Independent component analysis based on nonparametric density estimation. *IEEE Transactions on Neural Networks*, 15(1):55–65, 2004.
- [51] James C Liao, Riccardo Boscolo, Young-Lyeol Yang, Linh My Tran, Chiara Sabatti, and Vwani P Roychowdhury. Network component analysis: reconstruction of regulatory signals in biological systems. *Proceedings of the National Academy of Sciences*, 100(26):15522–15527, 2003.
- [52] Riccardo Boscolo, Chiara Sabatti, James C Liao, and Vwani P Roychowdhury. A generalized framework for network component analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 2(4):289–301, 2005.
- [53] Nicolas Gillis. Introduction to nonnegative matrix factorization. *arXiv preprint arXiv:1703.00663*, 2017.

- [54] Sanjeev Arora, Rong Ge, and Ankur Moitra. Learning Topic Models—Going Beyond SVD. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 1–10. IEEE, 2012.
- [55] Mikkel N Schmidt, Jan Larsen, and Fu-Tien Hsiao. Wind noise reduction using non-negative sparse coding. In *Machine Learning for Signal Processing, 2007 IEEE Workshop on*, pages 431–436. IEEE, 2007.
- [56] Slim Essid and Cédric Févotte. Smooth Nonnegative Matrix Factorization for Un-supervised Audiovisual Document Structuring. *IEEE Transactions on Multimedia*, 15(2):415–425, 2013.
- [57] Michael W Berry, Murray Browne, Amy N Langville, V Paul Pauca, and Robert J Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational statistics & data analysis*, 52(1):155–173, 2007.
- [58] Troy Lee, Adi Shraibman, et al. Lower bounds in communication complexity. *Foundations and Trends® in Theoretical Computer Science*, 3(4):263–399, 2009.
- [59] Enrico Carlini and Fabio Rapallo. Probability matrices, non-negative rank, and parameterization of mixture models. *Linear Algebra and its Applications*, 433(2):424–432, 2010.
- [60] Nicolas Gillis and François Glineur. On the geometric interpretation of the nonnegative rank. *Linear Algebra and its Applications*, 437(11):2685–2712, 2012.
- [61] Cédric Févotte, Nancy Bertin, and Jean-Louis Durrieu. Nonnegative Matrix Factorization with the Itakura-Saito Divergence: With Application to Music Analysis. *Neural computation*, 21(3):793–830, 2009.
- [62] Taiping Zhang, Bin Fang, Weining Liu, Yuan Yan Tang, Guanghui He, and Jing Wen. Total variation norm-based nonnegative matrix factorization for identifying discriminant representation of image patterns. *Neurocomputing*, 71(10-12):1824–1831, 2008.
- [63] Nicolas Gillis and François Glineur. Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization. *Neural computation*, 24(4):1085–1105, 2012.
- [64] Chih-Jen Lin. On the convergence of multiplicative update algorithms for nonnegative matrix factorization. *IEEE Transactions on Neural Networks*, 18(6):1589–1596, 2007.
- [65] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–2779, 2007.
- [66] Kejun Huang, Nicholas D Sidiropoulos, and Athanasios P Liavas. A flexible and efficient algorithmic framework for constrained matrix and tensor factorization. *IEEE Transactions on Signal Processing*, 64(19):5052–5065, 2016.

- [67] Jingu Kim and Haesun Park. Fast nonnegative matrix factorization: An active-set-like method and comparisons. *SIAM Journal on Scientific Computing*, 33(6):3261–3281, 2011.
- [68] Andrzej Cichocki and Anh-Huy Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 92(3):708–721, 2009.
- [69] Stephen A Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377, 2009.
- [70] Sanjeev Arora, Rong Ge, Ravindran Kannan, and Ankur Moitra. Computing a Non-negative Matrix Factorization–Provably. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 145–162. ACM, 2012.
- [71] David Donoho and Victoria Stodden. When Does Non-Negative Matrix Factorization Give a Correct Decomposition into Parts? In *Advances in neural information processing systems*, pages 1141–1148, 2004.
- [72] John C Gower, Garnt B Dijkstra, et al. *Procrustes problems*, volume 30. Oxford University Press on Demand, 2004.
- [73] Zhihui Zhu, Qiuwei Li, Gongguo Tang, and Michael B Wakin. The global optimization geometry of low-rank matrix optimization. *IEEE Transactions on Information Theory*, 67(2):1308–1331, 2021.
- [74] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.
- [75] Trevor Hastie, Rahul Mazumder, Jason D Lee, and Reza Zadeh. Matrix Completion and Low-Rank SVD via Fast Alternating Least Squares. *The Journal of Machine Learning Research*, 16(1):3367–3402, 2015.
- [76] Timothy R Tangherlini, Vwani Roychowdhury, Beth Glenn, Catherine M Crespi, Roja Bandari, Akshay Wadia, Misagh Falahi, Ehsan Ebrahimzadeh, and Roshan Bastani. “Mommy Blogs” and the vaccination exemption narrative: results from a machine-learning approach for story aggregation on parenting social media sites. *JMIR public health and surveillance*, 2(2), 2016.
- [77] K.C. Lee, J. Ho, and D. Kriegman. Acquiring linear subspaces for face recognition under variable lighting. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 27(5):684–698, 2005.
- [78] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):19, 2016.

- [79] Davood Hajinezhad, Tsung-Hui Chang, Xiangfeng Wang, Qingjiang Shi, and Mingyi Hong. Nonnegative matrix factorization using admm: Algorithm and convergence analysis. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4742–4746. IEEE, 2016.
- [80] Xihui Lin and Paul C Boutros. Fast Nonnegative Matrix Factorization and Applications to Pattern Extraction, Deconvolution and Imputation. *bioRxiv*, page 321802, 2018.
- [81] Yangyang Xu, Wotao Yin, Zaiwen Wen, and Yin Zhang. An alternating direction algorithm for matrix completion with nonnegative factors. *Frontiers of Mathematics in China*, 7(2):365–384, 2012.
- [82] Patrick Kwan and Martin J Brodie. Early identification of refractory epilepsy. *New England Journal of Medicine*, 342(5):314–319, 2000.
- [83] Samuel Wiebe, Warren T Blume, John P Girvin, and Michael Eliasziw. A randomized, controlled trial of surgery for temporal-lobe epilepsy. *New England Journal of Medicine*, 345(5):311–318, 2001.
- [84] Jerome Engel, Michael P McDermott, Samuel Wiebe, John T Langfitt, John M Stern, Sandra Dewar, Michael R Sperling, Irenita Gardiner, Giuseppe Erba, Itzhak Fried, et al. Early surgical therapy for drug-resistant temporal lobe epilepsy: a randomized trial. *Jama*, 307(9):922–930, 2012.
- [85] Eishi Asano, Csaba Juhasz, Aashit Shah, Sandeep Sood, and Harry T Chugani. Role of subdural electrocorticography in prediction of long-term seizure outcome in epilepsy surgery. *Brain*, 132(4):1038–1047, 2009.
- [86] Rekha Dwivedi, Bhargavi Ramanujam, P Sarat Chandra, Savita Sapra, Sheffali Gulati, Mani Kalaivani, Ajay Garg, Chandra S Bal, Madhavi Tripathi, Sada N Dwivedi, et al. Surgery for drug-resistant epilepsy in children. *New England Journal of Medicine*, 377(17):1639–1647, 2017.
- [87] Julia Jacobs, Joyce Y Wu, Piero Perucca, Rina Zelman, Malenka Mader, Francois Dubeau, Gary W Mathern, Andreas Schulze-Bonhage, and Jean Gotman. Removing high-frequency oscillations: a prospective multicenter study on seizure outcome. *Neurology*, 91(11):e1040–e1052, 2018.
- [88] Andrew Matsumoto, Benjamin H Brinkmann, S Matthew Stead, Joseph Matsumoto, Michal T Kucewicz, W Richard Marsh, Frederic Meyer, and Gregory Worrell. Pathological and physiological high-frequency oscillations in focal human epilepsy. *Journal of neurophysiology*, 110(8):1958–1964, 2013.

- [89] Federico Melani, Rina Zelman, Francesco Mari, and Jean Gotman. Continuous high frequency activity: a peculiar seeg pattern related to specific brain regions. *Clinical Neurophysiology*, 124(8):1507–1516, 2013.
- [90] Birgit Frauscher, Nicolás von Ellenrieder, Rina Zelman, Christine Rogers, Dang Khoa Nguyen, Philippe Kahane, François Dubeau, and Jean Gotman. High-frequency oscillations in the normal human brain. *Annals of neurology*, 84(3):374–385, 2018.
- [91] J.Y. Wu, R. Sankar, J.T. Lerner, J.H. Matsumoto, H.V. Vinters, and G.W. Mathern. Removing interictal fast ripples on electrocorticography linked with seizure freedom in children. *Neurology*, 75(19):1686–1694, 2010.
- [92] Päivi Nevalainen, Nicolás von Ellenrieder, Petr Klimesš, François Dubeau, Birgit Frauscher, and Jean Gotman. Association of fast ripples on intracranial eeg and outcomes after epilepsy surgery. *Neurology*, 95(16):e2235–e2245, 2020.
- [93] Hiroki Nariai, Shaun A. Hussain, Danilo Bernardo, Aria Fallah, Kristina K. Murata, Jimmy C. Nguyen, Rajsekar R. Rajaraman, Lekha M. Rao, Joyce H. Matsumoto, Jason T. Lerner, Noriko Salamon, David Elashoff, Raman Sankar, and Joyce Y. Wu. Prospective observational study: Fast ripple localization delineates the epileptogenic zone. *Clinical Neurophysiology*, 130(11):2144–2152, 2019.
- [94] Hirotaka Motoi, Jeong-Won Jeong, Csaba Juhasz, Makoto Miyakoshi, Yasuo Nakai, Ayaka Sugiura, Aimee Luat, Sandeep Sood, and Eishi Asano. Quantitative analysis of intracranial electrocorticography signals using the concept of statistical parametric mapping. *Scientific Reports*, 9:17385, 11 2019.
- [95] Naoto Kuroda, Masaki Sonoda, Makoto Miyakoshi, Hiroki Nariai, Jeong-Won Jeong, Hirotaka Motoi, Aimee F Luat, Sandeep Sood, and Eishi Asano. Objective interictal electrophysiology biomarkers optimize prediction of epilepsy surgery outcome. *Brain Communications*, 3(2), 03 2021. fcab042.
- [96] Baotian Zhao, Wenhan Hu, Chao Zhang, Xiu Wang, Yao Wang, Chang Liu, Jiajie Mo, Xiaoli Yang, Lin Sang, Yanshan Ma, Xiaoqi Shao, Kai Zhang, and Jianguo Zhang. Integrated automatic detection, classification and imaging of high frequency oscillations with stereoelectroencephalography. *Frontiers in Neuroscience*, 14:546, 2020.
- [97] Mina Amiri, Jean-Marc Lina, Francesca Pizzo, and Jean Gotman. High frequency oscillations and spikes: Separating real hfos from false oscillations. *Clinical Neurophysiology*, 127:187–196, 2016.
- [98] Jin Jing, Haoqi Sun, Jennifer A. Kim, Aline Herlopian, Ioannis Karakis, Marcus C Ng, Jonathan J. Halford, Douglas Maus, Fonda Chan, Marjan Dolatshahi, Carlos Muñiz, Catherine J. Chu, Valeria Saccá, Jay S. Pathmanathan, Wendong Ge, Justin Dauwels,

- Alice D Lam, Andrew J Cole, Sydney S. Cash, and Michael Brandon Westover. Development of expert-level automated detection of epileptiform discharges during electroencephalogram interpretation. *JAMA neurology*, 2019.
- [99] Baotian Zhao, Wen han Hu, Chao Zhang, Xiu Wang, Yao Wang, Chang Liu, Jiajie Mo, Xiaoli Yang, Lin Sang, Yanshan Ma, Xiao-Qiu Shao, Kai Zhang, and Jianguo Zhang. Integrated automatic detection, classification and imaging of high frequency oscillations with stereoelectroencephalography. *Frontiers in Neuroscience*, 14, 2020.
 - [100] Rui Zuo, Jing Wei, Xiaonan Li, Chunlin Li, Cui Zhao, Zhaohui Ren, Ying Liang, Xinling Geng, Chenxi Jiang, Xiaofeng Yang, and Xu Zhang. Automated detection of high-frequency oscillations in epilepsy based on a convolutional neural network. *Frontiers in Computational Neuroscience*, 13:6, 2019.
 - [101] Deep learning with noisy labels: Exploring techniques and remedies in medical image analysis. *Medical Image Analysis*, 65:101759, 2020.
 - [102] Richard J Staba, Charles L Wilson, Anatol Bragin, Itzhak Fried, and Jerome Engel Jr. Quantitative analysis of high-frequency oscillations (80–500 hz) recorded in human epileptic hippocampus and entorhinal cortex. *Journal of neurophysiology*, 88(4):1743–1752, 2002.
 - [103] Miguel Navarrete, Catalina Alvarado-Rojas, Michel Le Van Quyen, and Mario Valderama. Ripplelab: A comprehensive application for the detection, analysis and classification of high frequency oscillations in electroencephalographic signals. *PloS one*, 11(6):e0158276, 2016.
 - [104] 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*, pages 770–778, 2016.
 - [105] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Gianotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.
 - [106] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.
 - [107] Anatol Bragin, Jerome Engel Jr, Charles L Wilson, Itzhak Fried, and Gary W Mathern. Hippocampal and entorhinal cortex high-frequency oscillations (100–500 hz) in human epileptic brain and in kainic acid-treated rats with chronic seizures. *Epilepsia*, 40(2):127–137, 1999.

- [108] Anatol Bragin, Istvan Mody, Charles L Wilson, and Jerome Engel. Local generation of fast ripples in epileptic brain. *Journal of Neuroscience*, 22(5):2012–2021, 2002.
- [109] Gyorgy Buzsaki, Zolt Horvath, Ronald Urioste, Jammille Hetke, and Kensall Wise. High-frequency network oscillation in the hippocampus. *Science*, 256(5059):1025–1027, 1992.
- [110] Krit Charupanit, Indranil Sen-Gupta, Jack J Lin, and Beth A Lopour. Amplitude of high frequency oscillations as a biomarker of the seizure onset zone. *Clinical Neurophysiology*, 131(11):2542–2550, 2020.
- [111] John GR Jefferys, Liset Menendez de la Prida, Fabrice Wendling, Anatol Bragin, Massimo Avoli, Igor Timofeev, and Fernando H Lopes da Silva. Mechanisms of physiological and epileptic hfo generation. *Progress in neurobiology*, 98(3):250–264, 2012.
- [112] Rui Zuo, Jing Wei, Xiaonan Li, Chunlin Li, Cui Zhao, Zhaohui Ren, Ying Liang, Xinling Geng, Chenxi Jiang, Xiaofeng Yang, et al. Automated detection of high-frequency oscillations in epilepsy based on a convolutional neural network. *Frontiers in computational neuroscience*, 13:6, 2019.
- [113] Baotian Zhao, Wenhan Hu, Chao Zhang, Xiu Wang, Yao Wang, Chang Liu, Jiajie Mo, Xiaoli Yang, Lin Sang, Yanshan Ma, et al. Integrated automatic detection, classification and imaging of high frequency oscillations with stereoelectroencephalography. *Frontiers in Neuroscience*, 14:546, 2020.
- [114] Hisham Daoud and Magdy Bayoumi. Deep learning approach for epileptic focus localization. *IEEE transactions on biomedical circuits and systems*, 14(2):209–220, 2019.
- [115] Hiroki Nariai, Joyce Y Wu, Danilo Bernardo, Aria Fallah, Raman Sankar, and Shaun A Hussain. Interrater reliability in visual identification of interictal high-frequency oscillations on electrocorticography and scalp eeg. *Epilepsia open*, 3:127–132, 2018.
- [116] Emre Aksan, Manuel Kaufmann, and Otmar Hilliges. Structured Prediction Helps 3D Human Motion Modelling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7144–7153, 2019.
- [117] Yujun Cai, Lin Huang, Yiwei Wang, Tat-Jen Cham, Jianfei Cai, Junsong Yuan, Jun Liu, Xu Yang, Yiheng Zhu, Xiaohui Shen, et al. Learning Progressive Joint Propagation for Human Motion Prediction. In *European Conference on Computer Vision*, pages 226–242. Springer, 2020.
- [118] Yongyi Tang, Lin Ma, Wei Liu, and Wei-Shi Zheng. Long-term Human Motion Prediction by Modeling Motion Context and Enhancing Motion Dynamics. In *IJCAI*, 2018.

- [119] Qiongjie Cui, Huaijiang Sun, and Fei Yang. Learning Dynamic Relationships for 3D Human Motion Prediction. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6518–6526, 2020.
- [120] Yan Zhang, Michael J. Black, and Siyu Tang. We are More than Our Joints: Predicting how 3D Bodies Move. In *Proceedings IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2021.
- [121] Qiongjie Cui and Huaijiang Sun. Towards Accurate 3D Human Motion Prediction From Incomplete Observations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4801–4810, June 2021.
- [122] Ye Yuan and Kris Kitani. DLow: Diversifying Latent Flows for Diverse Human Motion Prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.
- [123] Dario Pavllo, David Grangier, and Michael Auli. Quaternet: A quaternion-based recurrent model for human motion. In *British Machine Vision Conference (BMVC)*, 2018.
- [124] Jiashun Wang, Huazhe Xu, Jingwei Xu, Sifei Liu, and Xiaolong Wang. Synthesizing Long-Term 3D Human Motion and Interaction in 3D Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9401–9411, 2021.
- [125] Guiyu Xia, Huaijiang Sun, Qingshan Liu, and Renlong Hang. Learning-based Sphere Nonlinear Interpolation for Motion Synthesis. *IEEE Transactions on Industrial Informatics*, 15(5):2927–2937, 2019.
- [126] Félix G Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. Robust Motion In-betweening. *ACM Transactions on Graphics (TOG)*, 39(4):60–1, 2020.
- [127] Haoye Cai, Chunyan Bai, Yu-Wing Tai, and Chi-Keung Tang. Deep Video Generation, Prediction and Completion of Human Action Sequences. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 366–382, 2018.
- [128] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. MoCoGAN: Decomposing Motion and Content for Video Generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1526–1535, 2018.
- [129] Mathis Petrovich, Michael J. Black, and Gül Varol. Action-conditioned 3D human motion synthesis with transformer VAE. In *ICCV*, 2021.
- [130] Luis Molina Tanco and Adrian Hilton. Realistic synthesis of novel human movements from a database of motion capture examples. In *Proceedings Workshop on Human Motion*, pages 137–142. IEEE, 2000.

- [131] Vladimir Pavlovic, James M Rehg, and John MacCormick. Learning Switching Linear Models of Human Motion. In *NIPS*, volume 2, page 4, 2000.
- [132] Jack M Wang, David J Fleet, and Aaron Hertzmann. Gaussian Process Dynamical Models for Human Motion. *IEEE transactions on pattern analysis and machine intelligence*, 30(2):283–298, 2007.
- [133] Andreas M Lehrmann, Peter V Gehler, and Sebastian Nowozin. Efficient Nonlinear Markov Models for Human Motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1314–1321, 2014.
- [134] Yi Zhou, Zimo Li, Shuangjiu Xiao, Chong He, Zeng Huang, and Hao Li. Auto-conditioned recurrent networks for extended complex human motion synthesis. In *International Conference on Learning Representations*, 2018.
- [135] Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. Recurrent Network Models for Human Dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4346–4354, 2015.
- [136] Julieta Martinez, Michael J Black, and Javier Romero. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2891–2900, 2017.
- [137] Zhiyong Wang, Jinxiang Chai, and Shihong Xia. Combining Recurrent Neural Networks and Adversarial Training for Human Motion Synthesis and Control. *IEEE transactions on visualization and computer graphics*, 27(1):14–28, 2019.
- [138] Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs*, pages 1–4. 2015.
- [139] Gustav Eje Henter, Simon Alexanderson, and Jonas Beskow. MoGlow: Probabilistic and Controllable Motion Synthesis Using Normalising Flows. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020.
- [140] Xinchun Yan, Akash Rastogi, Ruben Villegas, Kalyan Sunkavalli, Eli Shechtman, Sunil Hadap, Ersin Yumer, and Honglak Lee. MT-VAE: Learning Motion Transformations to Generate Multimodal Human Dynamics. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 265–281, 2018.
- [141] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS’14*, page 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

- [142] Yanli Ji, Feixiang Xu, Yang Yang, Fumin Shen, Heng Tao Shen, and Wei-Shi Zheng. A large-scale RGB-D database for arbitrary-view human action recognition. In *Proceedings of the 26th ACM international Conference on Multimedia*, pages 1510–1518, 2018.
- [143] Hsin-Ying Lee, Xiaodong Yang, Ming-Yu Liu, Ting-Chun Wang, Yu-Ding Lu, Ming-Hsuan Yang, and Jan Kautz. Dancing to Music. In *NeurIPS*, 2019.
- [144] 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. *Advances in neural information processing systems*, 30, 2017.
- [145] Kfir Aberman, Yijia Weng, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. Unpaired motion style transfer from video to animation. *ACM Transactions on Graphics (TOG)*, 39(4):64–1, 2020.
- [146] Vahabpour Arash. *Self organizing generative models for diverse imitation learning*. PhD thesis, University of California, Los Angeles, 2022.
- [147] Cong Fei, Bin Wang, Yuzheng Zhuang, Zongzhang Zhang, Jianye Hao, Hongbo Zhang, Xuewu Ji, and Wulong Liu. Triple-gail: a Multi-Modal Imitation Learning Framework with Generative Adversarial Nets. *arXiv preprint arXiv:2005.10622*, 2020.
- [148] Josh Merel, Yuval Tassa, Dhruva TB, Sriram Srinivasan, Jay Lemmon, Ziyu Wang, Greg Wayne, and Nicolas Heess. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017.
- [149] Karol Hausman, Yevgen Chebotar, Stefan Schaal, Gaurav Sukhatme, and Joseph Lim. Multi-Modal Imitation Learning from Unstructured Demonstrations using Generative Adversarial Nets. *arXiv preprint arXiv:1705.10479*, 2017.
- [150] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Anticipating visual representations from unlabeled video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 98–106, 2016.
- [151] Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the Latent Space of Generative Networks. *arXiv preprint arXiv:1707.05776*, 2017.
- [152] Yedid Hoshen, Ke Li, and Jitendra Malik. Non-Adversarial Image Synthesis with Generative Latent Nearest Neighbors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5811–5819, 2019.