THÈSE

Pour obtenir le grade de

**DOCTEUR DE L'UNIVERSITÉ GRENOBLE ALPES**

École doctorale : MSTII - Mathématiques, Sciences et technologies de l'information, Informatique
Spécialité : Mathématiques et Informatique
Unité de recherche : Laboratoire Jean Kuntzmann

# Modèles génératifs à variables latentes profondes pour le traitement multimédia

# Deep latent-variable generative models for multimedia processing

Présentée par :

**Xiaoyu LIN**

Direction de thèse :

| | |
|---|---|
| **Xavier ALAMEDA-PINEDA**<br>CHARGE DE RECHERCHE HDR, CENTRE INRIA UNIVERSITE GRENOBLE ALPES | Directeur de thèse |
| **Laurent GIRIN**<br>PROFESSEUR DES UNIVERSITES, GRENOBLE INP - UGA | Co-directeur de thèse |

Rapporteurs :

**GAËL RICHARD**
PROFESSEUR, TELECOM PARIS
**DAVID PICARD**
DIRECTEUR DE RECHERCHE, ECOLE NATIONALE DES PONTS ET CHAUSSEES

Thèse soutenue publiquement le **25 juin 2024**, devant le jury composé de :

| | |
|---|---|
| **JEAN-MARC BROSSIER,**<br>PROFESSEUR DES UNIVERSITES, GRENOBLE INP | Président |
| **XAVIER ALAMEDA PINEDA,**<br>CHARGE DE RECHERCHE HDR, CENTRE INRIA UNIVERSITE GRENOBLE ALPES | Directeur de thèse |
| **LAURENT GIRIN,**<br>PROFESSEUR DES UNIVERSITES, GRENOBLE INP | Co-directeur de thèse |
| **GAËL RICHARD,**<br>PROFESSEUR, TELECOM PARIS | Rapporteur |
| **DAVID PICARD,**<br>DIRECTEUR DE RECHERCHE, ECOLE NATIONALE DES PONTS ET CHAUSSEES | Rapporteur |
| **SHAI BEN-DAVID,**<br>FULL PROFESSOR, UNIVERSITY OF WATERLOO | Examinateur |
| **DOROTHEA KOLOSSA,**<br>FULL PROFESSOR, TECHNISCHE UNIVERSITÄT BERLIN | Examinatrice |

# ABSTRACT

Deep probabilistic generative models hold a crucial position within the realm of machine learning research. They serve as powerful tools for comprehending complex real-world data, such as image, audio, and text, by modeling their underlying distributions. This capability further enables the generation of new data samples. Moreover, these models can be utilized to discover hidden structures and the intrinsic factors of variation within data. The data representations that are learned through this process can be leveraged across a spectrum of downstream prediction tasks, thereby enhancing the decision-making process.

Another research direction involves leveraging the flexibility and robust generalization ability of deep probabilistic generative models for solving intricate scientific and engineering problems. Though supervised deep learning methods applied to sophisticatedly designed neural architectures have achieved state-of-the-art performance across various domains, their practical application to real-world situations remains constrained. These limitations arise from the necessity of extensive volumes of annotated data for training and a shortfall in model interpretability. In this PhD work, we explore an alternative approach using deep probabilistic generative models within an unsupervised or weakly supervised framework to overcome these hurdles. Specifically, the proposed approach involves initially pre-training a deep probabilistic generative model (DPGM) with natural or synthetic signals to embed prior knowledge about the complex data patterns. Subsequently, this pre-trained model is integrated into an extended DPGM to address the specific practical problem.

Our research focuses on a specific type of DPGM designed for sequential data, referred to as dynamical variational auto-encoder (DVAE). DVAEs are a family of deep latent variable models extended from the variational auto-encoder (VAE) for sequential data modeling. They leverage a sequence of latent vectors to depict the intricate temporal dependencies within the sequential observed data. By integrating DVAEs within a DPGM, we address a range of audio and visual tasks, namely multi-object tracking (MOT), single-channel audio source separation (SC-ASS), and speech enhancement (SE). The solutions are derived based on variational inference (VI) methods. Additionally, we also investigate a novel architecture, HiT-DVAE, which incorporates the Transformer architecture within the probabilistic framework of DVAEs. HiT-DVAE and its variant, LigHT-DVAE, both demonstrate excellent performance in speech modeling through robust sequential data handling.

The findings from our experiments confirm the potential of DPGMs to address real-world problems with limited labeled data, offering scalable and interpretable solutions. Furthermore, the introduction of HiT-DVAE represents a significant advancement in the field, combining the strengths of Transformer architectures with probabilistic modeling for enhanced sequential data analysis. These works not only contribute to the theoretical understanding of DPGMs, but also demonstrate their practical applicability across various domains, laying the groundwork for future innovations in machine learning.

# RÉSUMÉ

Les modèles génératifs probabilistes profonds occupent une position crutiale dans le domaine de la recherche en apprentissage automatique. Ces sont des outils puissants pour comprendre des données réelles complexes, telles que les images, les signaux audios et le texte, en modélisant leurs distributions. Cette capacité permet également la génération de nouveaux échantillons de données. De plus, ces modèles peuvent être utilisés pour découvrir des structures cachées et les facteurs intrinsèques de variation au sein des données. Les représentations des données apprises à travers ce processus peuvent être exploitées dans un spectre de tâches de prédiction en aval, améliorant ainsi le processus décisionnel.

Une autre direction de recherche explore comment la flexibilité et la capacité de généralisation robuste des modèles génératifs probabilistes profonds peut être utilisée pour résoudre des problèmes scientifiques et d'ingénierie complexes. Bien que les méthodes d'apprentissage profond supervisées appliquées sur des architectures neuronales sophistiquées obtiennent les performances de l'état de l'art dans divers domaines, leur application pratique aux situations réelles reste limitée par un certain nombre de facteurs. Ces limitations sont par exemple la nécessité d'un immense volume de données annotées et un certain déficit en interprétabilité des modèles. Dans ce travail de thèse, nous explorons une approche alternative utilisant des modèles génératifs probabilistes profonds dans un cadre non supervisé ou faiblement supervisé pour surmonter ces obstacles. Plus précisément, l'approche proposée implique le pré-entrainement initial d'un modèle génératif probabiliste profond (DPGM) avec des signaux naturels ou synthétiques pour intégrer des con-

naissances préalables sur les données complexes. Ensuite, ce modèle pré-entrainé est intégré dans un DPGM plus large, conçu pour traiter le problème pratique considéré.

Notre recherche se concentre sur un type spécifique de DPGM conçu pour les données séquentielles, appelé auto-encodeur variationnel dynamique (DVAE). Les DVAEs sont une famille de modèles à variables latentes profonds, dérivés de l'auto-encodeur variationnel (VAE), et adaptés pour la modélisation de données séquentielles. Ils reposent sur une séquence de vecteurs latents pour capturer les dépendances temporelles complexes au sein des données séquentielles observées. En intégrant les DVAEs dans un DPGM étendu, nous abordons une gamme de traitements audios et videos, qui sont le suivi multi-objets (MOT), la séparation de sources audio mono-canal (SC-ASS) et le réhaussement de la parole dans le bruit (SE). Les solutions sont dérivées en utilisant la méthodologie générale de l'inférence variationnelle (VI). De plus, nous étudions également une nouvelle architecture de DVAE, appelée HiT-DVAE, qui intègre l'architecture Transformer dans le cadre probabiliste des DVAEs. HiT-DVAE et sa variante, LigHT-DVAE, démontrent tous les deux d'excellentes performances dans la modélisation de la parole.

Les résultats de nos expériences confirment le potentiel des DPGMs pour adresser des problèmes réels avec une quantité de données étiquetées limitée, offrant des solutions évolutives et interprétables. De plus, l'introduction de HiT-DVAE représente une contribution au domaine, combinant la force des architectures Transformer avec celle de la modélisation probabiliste pour une meilleure modélisation des données séquentielles. Ce travail ne contribue pas seulement à la compréhension théorique des DPGMs, mais démontre également leur capacité d'application dans divers domaines, jetant les bases pour de futures innovations en apprentissage automatique.

# Declaration of Generative AI and AI-assisted Technologies in the Writing Process

Statement: During the preparation of this thesis I used ChatGPT (GPT 3.5) to refine the contents, aiming to improve readability and language. Besides, I used ChatGPT (GPT 3.5) to generate the dialogues reported in Figure 8.1, Figure 8.2, and Figure 8.3 in Section 8.3, Chapter 8. I also used DALL·E to generate the image in Figure 8.4 in Section 8.3, Chapter 8. Following the use of these services, I reviewed and edited the content as needed and take full responsibility for the content.

# ACKNOWLEDGMENT

Pursuing a PhD is not a easy journey, and it is the support and companionship along the way that make it memorable. I would like to express my deepest gratitude to everyone who has been part of this significant chapter in my life.

I am particularly grateful to my advisors, Xavier Alameda-Pineda and Laurent Girin. I have learned a lot from them, not only in terms of scientific knowledge and research methodologies but also in terms of their attitudes towards science and their invaluable personalities. My scientific research journey has not been smooth sailing, just like everyone else. At the begining of my PhD, I spent one year on my first research work, the MixDVAE model. We have spent a lot of time articulating the research findings during this work into a long paper of around twenty pages. I was so proud of this paper that I viewed it as a work of art that I poured my heart and soul into. Unfortunately, it was rejected after a long review process for the reason that our method did not beat the state-of-the-art supervised models. I was very frustrated with this result. Laurent encouraged me that this does not mean that I have not done a good job. It was just because our targeting journal is a top-level journal, leading to its competitive selection criteria. Laurent told me “Garde le moral et la tête haute. (Keep your spirits up and your head held high.)” Afterwards, we tried another top-level journal. After another long-term review process, it got rejected again. This time, I began to deeply doubt the value of my work. I discussed this with Xavier and asked him what the practical significance of the myriad mathematical equations we had diligently derived was. Xavier did not answer my question directly. He just said that several decades ago, when Kiyoshi Itô established his theory of stochastic differential equations, he never thought about how his theory could inspire the progress of diffusion models today. Finally and fortunately, our paper has been accepted by the Transactions on Machine Learning Research (TMLR), accompanied by highly constructive reviews that greatly contributed to enhancing the quality and depth of its content. Today, I remain without a definitive answer to this question. However, I am certain that the insights shared by Xavier and Laurent will leave a lasting impression on me. I will continue to ponder this inquiry and endeavor to address it through my future research endeavors.

Additionally, I extend my sincere gratitude to my close collaborators: Simon Leglaive, for the countless enlightening scientific exchanges and his encouragement in guiding me along the research journey; and Xiaoyu Bie, for his invaluable assistance with coding and his kindness in addressing my inquiries.

Moreover, I wish to express my sincere gratitude to Gaël Richard and David Picard for graciously accepting to review this thesis. And I am deeply thankful to Dorothea Kolossa, Shai Ben-David and Olivier Michel for their willingness to serve as examiners on the jury panel.

I find myself incredibly fortunate to be a member of the RobotLearn team. I have had the pleasure of encountering plenty of interesting souls here. Gaétan Lepage, he excels in coding and is always warmly receptive to assisting us in resolving bugs. The “Remi” system he developed has proven invaluable in streamlining our cluster server connections and saving time. Anand Ballou, he is very sociable, cares about others, and always proposes organizing after-work activities. Louis Airale, during our time sharing the same office, we talked a lot about the PhD life, social and cultural differences, as well as other interesting topics. Chris Reinke, he gave me a lot of valuable suggestions on my academic career. Space constraints prevent me from recounting all my experiences with each member of the team. Nevertheless, I wish to extend my heartfelt gratitude to all of the former and current members of the RobotLearn team throughout my PhD journey: Radu Horaud, Soraya Arias, Nicolas Turro, Matthieu Py, Alex Auternaud, Kirubakaran Ramamoorthy, Victor Sanchez, Ghazi Shazan Ahmad, Daniel Jost, Andres

Bermeo Marinelli, Wen Guo, Alexandre Duplessis, Lorenzo Vaquero Otal, Basavasagar Kallinath Patil, Yihong Xu, Luis Gomez Cámara, Timothée Wintz, Zhiqi Kang, Guillaume Delorme, David Emukpere, and Natanaël Dubois–Quilici. I would also like to express my special thanks to Nathalie Gillot. She provided invaluable assistance with administrative stuff and reimbursement procedures, allowing me to focus more on scientific research.

I am also grateful to all the Chinese friends I met in Grenoble: Dexiong Chen, Xiaoyu Bie, Wen Guo, Yuming Du, Zhiqi Kang, Yangtao Wang, Yang Xiao and Xi Shen. Their companionship helped me navigate through the difficult time during the Covid lockdown, and we have spent a lot of precious and memorable moments together.

Thanks to my French family, Odile Marquestaut, Yves Tresson, Jacques Marquestaut, Paul Tresson, Marie-Lou Tresson and Emmanuel Tresson, for their consistent invitations during the Christmas festivities and for treating me as a real member of the family. They have offered me immense emotional support, bringing me a sense of joy and comfort from having a family presence.

Thanks to my best friends, Yiyu Wang, Fanwen Sun, Li Fang, Yongxin Zhou, and Linjing Tang. They provide me with unwavering emotional support. Even though some of my friends are far away and we cannot easily meet each other in person, the countless hours spent talking on weekends make me feel as though they were right there with me, standing by my side.

Thanks to my family, Ruihong Feng, Hejun Lin, and Jingbo Lin for their boundless and unwavering love. Words fail to express the depth of my emotions.

Finally, I would also like to thank myself for my courage, fortitude, curiosity and vitality that have enabled me to navigate this challenging yet meaningful journey.

# CONTENTS

1 Introduction 19

1.1 Probabilistic Generative Modeling: General Considerations . . . . . . . . 20

1.2 Applying (D)PGMs for Multimedia Processing . . . . . . . . . . . . . . 22

1.3 Overview of the Thesis . . . . . . . . . . . . . . . . . . . . . . . . . 23

1.4 List of Publications . . . . . . . . . . . . . . . . . . . . . . . . . . 24

2 Methodological Background 25

2.1 Probabilistic Generative Models . . . . . . . . . . . . . . . . . . . . 26

2.2 Latent Variable Models: An Overview . . . . . . . . . . . . . . . . . 30

2.2.1 LVMs for Static Data . . . . . . . . . . . . . . . . . . . . . . 32

2.2.2 LVMs for Sequential Data . . . . . . . . . . . . . . . . . . . 41

2.3 Learning and Inference of the Latent Variable Models . . . . . . . . . . . 48

2.3.1 Posterior Intractablility and Approximate Inference . . . . . . . . 48

2.3.2 A General Introduction to Variational Inference . . . . . . . . . . 49

2.3.3 Exact Inference and EM Algorithms . . . . . . . . . . . . . . . . 51

2.3.4 Free-Form VI and Mean-Field Approximation . . . . . . . . . . 51

2.3.5 Fixed-Form VI and Gradient-Based Optimization . . . . . . . . . 53

2.4 Deep Architectures for Sequential Data Modeling . . . . . . . . . . . . . 55

2.4.1 Recurrent Neural Networks . . . . . . . . . . . . . . . . . . . 56

2.4.2 1D Convolutional Neural Networks . . . . . . . . . . . . . . . . 58

2.4.3 Attention Mechanism and Transformers . . . . . . . . . . . . . . 59

2.5 Examples of Applications to Audio, Image, and Video Processing . . . . 63
2.5.1 Multi-Target Tracking . . . . . . . . . . . . . . . . . . . . . . . . 64
2.5.2 Single-Channel Audio Source Separation . . . . . . . . . . . . . 66
2.5.3 Speech Enhancement . . . . . . . . . . . . . . . . . . . . . . . . 69

**3 Mixture of DVAEs for Multi-Source Trajectory Modeling and Separation 73**
3.1 A DPGM for Multi-Source Data . . . . . . . . . . . . . . . . . . . . . 74
3.2 MixDVAE Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 76
3.2.1 Problem Formulation and Notations . . . . . . . . . . . . . . . . 76
3.2.2 General Principle of the Proposed Model and Solution . . . . . . 77
3.2.3 Generative Model . . . . . . . . . . . . . . . . . . . . . . . . . . 78
3.2.4 Inference Model . . . . . . . . . . . . . . . . . . . . . . . . . . . 80
3.3 MixDVAE Solution: A Variational Expectation-Maximization Algorithm 81
3.3.1 E-S Step . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 82
3.3.2 E-Z Step . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 84
3.3.3 E-W Step . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86
3.3.4 M Step . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 86
3.3.5 MixDVAE Complete Algorithm . . . . . . . . . . . . . . . . . . . 87
3.3.6 Choice of the DVAE Model . . . . . . . . . . . . . . . . . . . . . 89

**4 Application of MixDVAE on MOT 91**
4.1 Application of MixDVAE on MOT . . . . . . . . . . . . . . . . . . . . 92
4.2 Setting MixDVAE in the MOT Configuration . . . . . . . . . . . . . . 93
4.3 DVAE Pre-training . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
4.3.1 Dataset . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 93
4.3.2 Training Details . . . . . . . . . . . . . . . . . . . . . . . . . . . 94
4.4 MixDVAE Evaluation Set-up . . . . . . . . . . . . . . . . . . . . . . . 94
4.4.1 Dataset . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 94
4.4.2 Algorithm Initialization . . . . . . . . . . . . . . . . . . . . . . . 95
4.4.3 Observation Covariance Matrix . . . . . . . . . . . . . . . . . . . 96

4.4.4 Hyperparameters . . . . . . . . . . 96
4.4.5 Baselines . . . . . . . . . . 96
4.4.6 Evaluation Metrics . . . . . . . . . . 98
4.5 Experimental Results . . . . . . . . . . 98
4.5.1 Quantitative Analysis . . . . . . . . . . 98
4.5.2 Qualitative Analysis . . . . . . . . . . 100
4.6 Ablation Studies . . . . . . . . . . 102
4.6.1 Influence of the Pre-trained DVAE Model Quality . . . . . . . . . . 102
4.6.2 Influence of the DVAE Fine-tuning . . . . . . . . . . 103
4.6.3 Influence of the Observation Variance Ratio . . . . . . . . . . 105

**5 Application of MixDVAE on SC-ASS 107**

5.1 Application of MixDVAE on SC-ASS . . . . . . . . . . 108
5.2 Setting MixDVAE in the SC-ASS Configuration . . . . . . . . . . 109
5.3 DVAE Pre-training . . . . . . . . . . 110
5.3.1 Dataset . . . . . . . . . . 110
5.3.2 Pre-processing . . . . . . . . . . 111
5.3.3 Training Details . . . . . . . . . . 111
5.4 MixDVAE Evaluation Set-up . . . . . . . . . . 112
5.4.1 Dataset . . . . . . . . . . 112
5.4.2 Algorithm Initialization . . . . . . . . . . 112
5.4.3 Observation Covariance Matrix . . . . . . . . . . 113
5.4.4 Hyperparameters . . . . . . . . . . 113
5.4.5 Baselines . . . . . . . . . . 113
5.4.6 Evaluation Metrics. . . . . . . . . . . 115
5.5 Experimental Results . . . . . . . . . . 115
5.5.1 Quantitative Analysis . . . . . . . . . . 115
5.5.2 Qualitative Analysis . . . . . . . . . . 117
5.6 Ablation Studies . . . . . . . . . . 118
5.6.1 Influence of the Pre-trained DVAE Model Quality . . . . . . . . . . 118

5.6.2 Influence of the DVAE Fine-tuning . . . . . . . . . . . . . . . . 120
5.7 Discussion on the Computational Complexity . . . . . . . . . . . . . . . 120
5.8 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 122

**6 Unsupervised Speech Enhancement with Deep Dynamical Probabilistic Generative Models 125**
6.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 126
6.2 Dynamical DPGM-based SE Method . . . . . . . . . . . . . . . . . . . . 129
6.2.1 Clean Speech Modeling with an RVAE . . . . . . . . . . . . . . 129
6.2.2 Dynamical DPGM-Based Noise Model . . . . . . . . . . . . . . 131
6.2.3 Speech Enhancement with the Inference Model . . . . . . . . . . 132
6.2.4 Model Optimization . . . . . . . . . . . . . . . . . . . . . . . 133
6.3 Experimental Settings . . . . . . . . . . . . . . . . . . . . . . . . . 134
6.3.1 Datasets and Pre-processing . . . . . . . . . . . . . . . . . . . 134
6.3.2 Implementation Details and Training Settings . . . . . . . . . . . 135
6.3.3 Baselines and Evaluation Metrics . . . . . . . . . . . . . . . . . 136
6.4 Experimental Results . . . . . . . . . . . . . . . . . . . . . . . . . 136
6.4.1 Comparison of Different Variants under Different Configurations 136
6.4.2 Comparison with the Baselines . . . . . . . . . . . . . . . . . . . 138
6.4.3 Discussion on the Computational Time . . . . . . . . . . . . . 140
6.4.4 Qualitative Analysis . . . . . . . . . . . . . . . . . . . . . . . 141
6.4.5 Results on Unmatched Pre-train/Test Set . . . . . . . . . . . . 144
6.5 Conclusion and Further Discussions . . . . . . . . . . . . . . . . . . . 144

**7 Speech Modeling with a Hierarchical Transformer Dynamical VAE 147**
7.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 148
7.2 Probabilistic Model of HiT-DVAE . . . . . . . . . . . . . . . . . . . . 150
7.2.1 Generative Model . . . . . . . . . . . . . . . . . . . . . . . . . 151
7.2.2 Inference Model . . . . . . . . . . . . . . . . . . . . . . . . . 152
7.2.3 Optimization . . . . . . . . . . . . . . . . . . . . . . . . . . . 152

7.3 Model Architecture of HiT-DVAE and LigHT-DVAE . . . . . . . . . . . 153
    7.3.1 HiT-DVAE Encoder . . . . . . . . . . . . . . . . . . . . . . . . 154
    7.3.2 HiT-DVAE Decoder . . . . . . . . . . . . . . . . . . . . . . . . 154
    7.3.3 Distinctions with the Original Transformers Architecture . . . . . 155
    7.3.4 LigHT-DVAE Adaptations . . . . . . . . . . . . . . . . . . . . . 156
7.4 Experimental Settings . . . . . . . . . . . . . . . . . . . . . . . . . . 157
    7.4.1 Datasets and Pre-processing . . . . . . . . . . . . . . . . . . . . 157
    7.4.2 Implementation Details and Training Settings . . . . . . . . . . . 158
    7.4.3 Baselines . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 158
    7.4.4 Evaluation Metrics . . . . . . . . . . . . . . . . . . . . . . . . . 158
7.5 Experimental Results . . . . . . . . . . . . . . . . . . . . . . . . . . 159
    7.5.1 Speech Spectrograms Analysis-Resynthesis Results . . . . . . . . 159
    7.5.2 Ablation Studies on Model Structures . . . . . . . . . . . . . . . 160
    7.5.3 Interpretability of the Global Latent Variable . . . . . . . . . . . 162
    7.5.4 Speech Spectrograms Generation Results . . . . . . . . . . . . . 163
7.6 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 164

**8 Conclusion and Further Discussions 165**

8.1 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 166
8.2 Insights and Limitations . . . . . . . . . . . . . . . . . . . . . . . . . 168
8.3 Towards a Broader Discussion . . . . . . . . . . . . . . . . . . . . . . 172

**A Appendix 179**

A.1 Mixture of DVAEs for Multi-Source Trajectory Modeling and Separation 179
    A.1.1 MixDVAE Algorithm Calculation Details . . . . . . . . . . . . . 179
    A.1.2 SRNN Implementation Details . . . . . . . . . . . . . . . . . . . 185
A.2 Application of MixDVAE on MOT . . . . . . . . . . . . . . . . . . . . 186
    A.2.1 Cascade Initialization in MOT . . . . . . . . . . . . . . . . . . . 186
    A.2.2 MOT Dataset Processing . . . . . . . . . . . . . . . . . . . . . . 188
    A.2.3 MOT Baselines Implementation Details . . . . . . . . . . . . . . 190

A.2.4 More MOT Tracking Examples . . . . . . . . . . . . . . . . . . . 191
A.3 Application of MixDVAE on SC-ASS . . . . . . . . . . . . . . . . . . . . . 193
A.3.1 Formulas for SC-ASS . . . . . . . . . . . . . . . . . . . . . . . . 193
A.3.2 More SC-ASS Examples . . . . . . . . . . . . . . . . . . . . . . . 194
A.4 Unsupervised Speech Enhancement with Deep Dynamical Probabilistic Generative Models . . . . . . . . . . . . . . . . . . . . . . . . . . . 194
A.4.1 Posterior Distributions Derivation . . . . . . . . . . . . . . . . . . 194

**Acronyms** **197**

**List of Figures** **205**

**List of Tables** **209**

**List of Algorithms** **211**

# CHAPTER 1

# INTRODUCTION

*It is not knowledge, but the act of learning,*
*not possession but the act of getting there,*
*which grants the greatest enjoyment.*

*— Carl Friedrich Gauss*

## 1.1 PROBABILISTIC GENERATIVE MODELING: GENERAL CONSIDERATIONS

Machine learning methods can be broadly categorized into two main groups: discriminative models and generative models. Given an observed variable $\mathbf{x}$, the discriminative models aim at learning a direct mapping from the observation to the prediction, which can be formulated as $\mathbf{y} = f(\mathbf{x})$. The predicted variable $\mathbf{y}$ can either be discrete in the case of classification or continuous in the case of regression. While the generative models usually do not have a specific prediction target. They are often employed to discover the underlying data generation mechanism by leveraging statistical analysis tools. The discriminative models are objective-oriented while the generative models are reasoning-oriented. Both of these two kinds of models can be interpreted from a probabilistic perspective. Let $\mathbf{x}$ denote the observed random variable. Let $\mathbf{y}$ denote the target random variable of a discriminative model, which can also be a generation factor of a generative models. For example, imagine that we have a set of pictures of dogs and cats. A discriminative model may be designed to predict the category of a given picture, which could be 'dog' or 'cat'. While this category information can also be used to generate corresponding pictures. Finally, let $\mathbf{h}$ denote other factors that we may also be interested in for the data generation. In the previous example, $\mathbf{h}$ could be, for instance, the color of the animal. The probabilistic discriminative model (PDM) aims at modeling the conditional distribution $p(\mathbf{y}|\mathbf{x})$, while the probabilistic generative model (PGM) aims at modeling the joint distribution $p(\mathbf{x}, \mathbf{y}, \mathbf{h})$. Both of these distributions can be represented by leveraging various mathematical tools. Particularly, the development of deep neural network (DNN) provides a powerful tool for capturing complex dependencies among variables, therefore enabling more accurate distribution modeling. Combining probabilistic models with DNN techniques results in what is commonly known as deep probabilistic models, with a specific subtype termed the deep probabilistic generative model (DPGM).

Just as the physicians understand and interpret the world with physical laws, the data scientists understand and interpret the world with statistical models. Understanding and modeling the whole data generation process in the real world is generally (much) more complicated than making a prediction with the observed data. Therefore, PGMs necessitate to impose stronger assumptions on the data compared to their discriminative counterparts. Nonetheless, exploring PGMs remains an appealing and worthwhile endeavor for various compelling reasons. First, mastering the data generation mechanism enables us not only to generate new data points from the estimated distribution, but also to control the generation process by varying the generative factors. For instance, the PGMs are widely used for text-to-image generation [143, 159, 178], text-to-speech generation [167, 93, 202], audio generation [1, 92] and code completion [35, 144]. Furthermore, they are also applied to tasks such as style transfer [240] and missing data imputation [18]. Second, conceiving PGMs naturally guides us to understand the world through causal reasoning. By conceptualizing the underlying data generation factors and modeling them with unobserved random variables, we force the model to learn interpretable, semantically meaningful and statistically independent data representations [15]. This can be interpreted as integrating inductive biases into the model with human-level intuitive prior knowledge. And this kind of inductive bias is essential to construct more generalizable models [76]. Finally, fitting a discriminative model generally lies under the supervised learning paradigm. This requires a large amount of annotated data with well-defined labels, which often requires costly and resource-intensive efforts. In contrast, PGMs can be applied under the unsupervised or weakly-supervised settings to discover and extract the inner structure and dependencies of the data. The features and representations learned by PGMs can therefore benefit various downstream supervised prediction tasks by reducing the amount of data required for training [15, 87].

In addition to the aforementioned motivations, a noteworthy avenue of research involves the application of generative models to tackle intricate scientific or engineering challenges. In numerous real-world scientific or industrial scenarios, problems often exhibit significant complexity and involve various influential factors. These problems typically cannot be simplified into straightforward mapping tasks that can be solved by directly training a discriminative prediction model. Moreover, in both rigorous scientific research and robust industrial production, relying solely on a single prediction often proves inadequate. For instance, in some critical fields such as medicine and autonomous driving, it is crucial to understand and assess the uncertainty of the prediction, so as to build more robust and reliable data-driven models. Hence, it becomes imperative to quantify uncertainties from a statistical standpoint. Indeed, model uncertainties arise from two main sources. Firstly, during the data collection phase, measuring sensors are typically subject to noise disturbances, resulting in measurement errors to a certain degree. Secondly, during the prediction phase, the model may exhibit errors if the assumption of identically and independently distributed data samples is not met or if the data sample size is limited. Nevertheless, discriminative models struggle to effectively quantify both types of uncertainties. In contrast, probabilistic generative models, combined with Bayesian inference, offer a principled approach for for taking uncertainty into account explicitly. Another important consideration is that, while the development of DNNs, along with stochastic gradient descent optimization, has significantly enhanced model expressiveness and facilitated learning from large datasets, deep learning models have sacrificed interpretability. It remains challenging to establish a theoretical framework explaining the remarkable performance of these models. Conversely, probabilistic generative models are inherently interpretable. Their explicitly defined random variables and generative distributions ensure transparency. Lastly, the flexibility and robust generalization ability of PGMs make them particularly well-suited for scenarios where vast amounts of annotated data are lacking.

## 1.2 APPLYING (D)PGMS FOR MULTIMEDIA PROCESSING

In this thesis, we will explore the utilization of DPGMs, particularly focusing on dynamical versions of these models, to tackle complex audio and visual tasks. Our research is centered on a specific class of dynamical DPGM, known as dynamical variational autoencoder (DVAE) tailored for sequential data modeling. More precisely, we investigate a learning paradigm which involves first pre-training a (several) DVAE model(s) on a (several) single source dataset(s) to capture the dynamical information of the sequential data.

Subsequently, the pre-trained DVAE model(s) will be integrated into another PGM to address complex multimedia processing problems, including multi-object tracking, single-channel audio source separation, and speech enhancement. The complete solutions are developed using principled Bayesian methods and variational inference methodology. We conduct comprehensive studies and experimental examinations to illustrate the effectiveness and limitations of this kind of method.

## 1.3 OVERVIEW OF THE THESIS

In Chapter 2, we provide a brief introduction to the general methodological background of PGMs. Specifically, we begin with an overview of existing generative models. Then, our attention turns to a specific category of generative models, the latent variable model (LVM). We explore different kinds of LVMs, classify them into distinct categories based on various aspects, and also discuss the associated learning and inference methods. Additionally, we present the deep architectures for sequential data modeling and outline the research context for the three tasks addressed in this thesis: multi-object tracking (MOT), single-channel audio source separation (SC-ASS), and speech enhancement (SE) with noise.

In Chapter 3, we tackle a complex and challenging task encountered in multiple technical domains such as computer vision or aeronautics: multi-source trajectory tracking and separation. We propose a DPGM framework and derive the solutions utilizing variational inference (VI). We apply the proposed method to two distinct application scenarios: MOT (Chapter 4) and SC-ASS (Chapter 5).

In Chapter 6, we explore unsupervised speech enhancement with deep dynamical generative models. We employ two distinct DNNs to individually model the distributions of clean speech and noise. Throughout this chapter, we will illustrate how various training configurations enable us to strike a balance between performance and inference efficiency.

In Chapter 7, we introduce a novel DPGM architecture called HiT-DVAE and investigate its applications to speech signal modeling. This architecture integrates a Transformer structure into a DVAE framework, leveraging the strengths of both. We assess the capabilities of this model through comprehensive experimentation.

In Chapter 8, we provide a comprehensive summary of the research conducted in this thesis, highlighting the insights gained and addressing the limitations of the proposed methods. Additionally, we engage in a broader discussion regarding potential future research directions.

## 1.4 LIST OF PUBLICATIONS

The content of this manuscript is based on the following publications:

- Xiaoyu Lin, Laurent Girin, and Xavier Alameda-Pineda. “Mixture of dynamical variational autoencoders for multi-source trajectory modeling and separation.” In Transactions on Machine Learning Research, 2023.
- Xiaoyu Lin, Simon Leglaive, Laurent Girin, and Xavier Alameda-Pineda. “Unsupervised speech enhancement with deep dynamical generative speech and noise models.” In Proceedings Interspeech Conference, pages 5102-5106, 2023.
- Xiaoyu Lin, Xiaoyu Bie, Simon Leglaive, Laurent Girin, and Xavier Alameda-Pineda. “Speech modeling with a hierarchical Transformer dynamical VAE.” In IEEE International Conference on Acoustics, Speech and Signal Processing, pages 1-5, 2023.

# CHAPTER 2

# METHODOLOGICAL BACKGROUND

*To know what you know and what you do*
*not know, that is true knowledge.*
知之为知之，不知为不知，是知也。

*— Confucius*

## 2.1 PROBABILISTIC GENERATIVE MODELS

The current existing generative models can be broadly grouped into two categories, according to if they model implicitly or explicitly the data distribution (i.e. the probability density function (PDF) of the data) [47, 141]. For models in the first category, the data distribution is implicitly modeled via the data sampling process, with no need for explicit formulation of the PDF. A typical example of models of this type is the **generative adversarial network (GAN)** [74]. A GAN comprises two fundamental components: a generator and a discriminator. The generator takes a random vector $\mathbf{z} \in \mathbb{R}^l$ sampled from the standard Gaussian distribution as input, and transforms it into a synthetic data sample $\mathbf{x} = \mathbf{f}_\theta(\mathbf{z}) \in \mathbb{R}^d$ with a function $\mathbf{f}_\theta : \mathbb{R}^l \to \mathbb{R}^d$ parameterized by a DNN. On the other hand, the discriminator takes either the generated data sample $\mathbf{x}$ or a real data sample from the training dataset $\mathbf{y}$ as input, and determine whether the input data is “real” or “fake”. The parameters of the generator and the discriminator are jointly optimized via adversarial training.

**Table 2.1**: The PDF forms of different generative models. In all of the models, $\mathbf{x}$ represents the data and $\theta$ represents the parameters of the model. Other symbols used in a specific model will be explained in the corresponding paragraph.

| Model | PDF form |
|---|---|
| Auto-regressive models | $p_\theta(\mathbf{x}) = \prod_{i=1}^{d} p_\theta(x_i \mid \mathbf{x}_{<i})$ |
| Normalizing flows | $p_\theta(\mathbf{x}) = p_{\mathbf{z}}(f_\theta^{-1}(\mathbf{x})) \lvert \det(\mathbf{J}_{f_\theta^{-1}}(\mathbf{x})) \rvert$ |
| Energy-based models | $p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$ |
| Diffusion model | $p_\theta(\mathbf{x}_0) = \int p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} \mid \mathbf{x}_t) \mathrm{d}\mathbf{x}_{1:T}$ |
| Score-based models | $p_\theta(\mathbf{x}) = \exp(\log p(\mathbf{x}_0) + \int_0^1 s_\theta(\mathbf{x}_0 + t(\mathbf{x} - \mathbf{x}_0))^T (\mathbf{x} - \mathbf{x}_0) \mathrm{d}t$ |
| Latent variable models | $p_\theta(\mathbf{x}) = \int p(\mathbf{z}) p_\theta(\mathbf{x} \mid \mathbf{z}) \mathrm{d}\mathbf{z}$ |

In contrast to the implicit generative models, models of the second category rely on an explicit formulation $p_\theta(\mathbf{x})$ parameterized by a set of parameters $\theta$ to approximate the true data PDF $p(\mathbf{x})$. With an explicit expression of the PDF, this kind of models are typically trained by maximizing the data likelihood function, or a surrogate of it [24, 141]. However, it is important to note that maximum likelihood estimation (MLE) is not the sole approach for learning these models; alternative methods are explored in, for instance, the score based models [94, 185]. We provide some examples of parametrized PDF forms corresponding to different generative models in Table 2.1.

One of the most intuitive and commonly used explicit generative models are the **auto-regressive models** [200, 204, 145, 146]. The auto-regressive models factorize the PDF of data $\mathbf{x} \in \mathbb{R}^d$ with dimension $d$ based on the causal auto-regressive dependencies between variables at different dimensions, as indicated in Table 2.1. Simple as it is in terms of probabilistic modeling, there exists a plethora of choices to model the auto-regressive dependencies so as to enhance the models' expressiveness, including the causal multilayer perceptron (MLP) [200, 66], the recurrent neural network (RNN) [204], the convolutional neural network (CNN) [145, 203], and the current state-of-the-art model, the generative Transformers [146]. The auto-regressive models have achieved very impressive data generation performance in various domains such as image, audio and text. Nevertheless, we need to point out that the auto-regressive models are generally employed in a deterministic way, without taking into account the uncertainty in the generation process.

Instead of directly modeling the complex distribution of the high-dimensional data, the **normalizing flows** [168, 106, 150] convert a simple known distribution into the complex unknown distribution of the data using (a set of cascaded) invertible function(s). Let $\mathbf{z} \in \mathbb{R}^d$ denote a random variable which follows a tractable and easy-to-sample distribution $p_{\mathbf{z}}(\mathbf{z})$. Let $\mathbf{f}_\theta : \mathbb{R}^d \to \mathbb{R}^d$ denote a continuously differentiable invertible function parametrized with DNNs. The normalizing flows transform the random variable $\mathbf{z}$ to the data sample $\mathbf{x}$ with $\mathbf{x} = \mathbf{f}_\theta(\mathbf{z})$. As indicated in Table 2.1, the PDF of $\mathbf{x}$ is obtained using the *change of variable formula*, with $\mathbf{J}_{\mathbf{f}_\theta^{-1}}$ denoting the Jacobian matrix of the function $\mathbf{f}_\theta^{-1}$. Normalizing flows enable the generation of data with complex patterns from vectors randomly sampled from a relatively simple distribution. However, the requirement for an invertible transform function places strong constraints on the choice of neural network architectures for flow-based models.

Inspired by the Boltzmann distribution in statistical physics, the **energy-based models** [196, 114] define the PDF of the data with the help of an energy function $E_\theta(\cdot)$, which is parameterized by DNNs, and a partition function $Z_\theta := \int \exp(-E_\theta(\mathbf{x}))\mathrm{d}\mathbf{x}$, which serves as the normalizing constant of the PDF. The model optimization involves maximizing the data log-likelihood. However, since the partition function $Z_\theta$ is usually intractable, the gradient of $\log Z_\theta$ with respect to $\theta$ needs to be approximated using techniques such as the contrastive divergence algorithm [86] and the Markov Chain Monte Carlo (MCMC) sampling. Alternatively, one can resort to the score matching methods [94, 210] or the noise contrastive estimation (NCE) [81] to circumvent the density normalization issue and learn model parameters.

Inspired by the diffusion process in non-equilibrium statistical physics, the principle of **diffusion models** [183, 88] is to corrupt the original structured data sample $\mathbf{x}_0$ by gradually adding small amounts of Gaussian noise into the data within a Markov chain, and by doing so, transform the unknown complex data distribution into a standard Gaussian distribution (denoted as $p(\mathbf{x}_T)$ in Table 2.1). The data generation process is obtained by reversing the diffusion process. Training the generative process of a diffusion model consists of finding the reverse Markov transition kernel $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ which maximizes a lower bound of the data log-likelihood. The transition kernel is often parametrized by DNNs.

The **score-based models** [185, 186, 188] propose to estimate the gradient of the data distribution *log-density* instead of directly estimating the PDF in order to circumvent the challenging issue of density normalization. Given a PDF $p(\mathbf{x})$, the associated Stein score function is defined as $\mathrm{s}(\mathbf{x}) := \nabla_{\mathbf{x}} \log p(\mathbf{x})$ [123]. The score function is usually modeled and parametrized with DNNs, resulting in a modeled score function $\mathrm{s}_\theta(\mathbf{x})$. The model is trained to minimize the Fisher divergence between the parameterized distribution $p_\theta(\mathbf{x})$ and the true data distribution $p(\mathbf{x})$, which is defined as $F(p, p_\theta) = \frac{1}{2}\mathbb{E}_p[||\mathrm{s}(\mathbf{x}) - \mathrm{s}_\theta(\mathbf{x})||_2^2]$. As the true data score function $\mathrm{s}(\mathbf{x})$ is inaccessible, directly computing the Fisher divergence is infeasible. The current methods resort to various score matching techniques to address this issue [94, 210, 187]. Once the estimated score function $s_\theta(\mathbf{x})$ is obtained, the data points can be sampled using Langevin MCMC sampling technique [152, 78], with no need to explicitly estimate the normalization constant of the PDF. Conversely, to derive the PDF from a given a score function, one can apply the formula provided in Table 2.1

and determine the term $\log p(\mathbf{x}_0)$ using the normalization constraint $\int p(\mathbf{x})d\mathbf{x} = 1$. In particular, the denoising score matching method [188] can be connected with the diffusion models [88] under a more general umbrella of stochastic differential equation (SDE) [194, 242].

Finally, we present the **latent variable model (LVM)**, which forms the central topic that will be explored in this manuscript. LVMs introduce latent variables $\mathbf{z}$ with relatively simple and usually known prior distributions to represent the complicated distribution over the observed variable $\mathbf{x}$. We should note that, although both normalizing flows and the LVMs make use of auxiliary variables with simpler prior distributions to model complex real-world data distributions, there is a distinct difference between these two methods. Normalizing flows depict the relationship between the observed variable $\mathbf{x}$ and the latent variable $\mathbf{z}$ with a deterministic function $\mathbf{f}_\theta$, whereas LVMs formulate the problem in a more principled probabilistic perspective by leveraging the conditional distributions and Bayes' rule.[1] When the conditional dependencies between the random variables are presented graphically, we delve into the world of **graphical models** [100, 213]. Graphical models play a crucial role in uncovering the hidden factors that govern the data generation process and in capturing the causal relationships among these factors [155]. However, it is not mandatory for the latent variables to have any explicit physical interpretation, they can be introduced simply in purpose to construct more complex joint distributions or to describe the supposed inner structure of the data [24]. Computing the marginal likelihood of the observed data $p(\mathbf{x})$ from the joint distribution $p(\mathbf{x}, \mathbf{z})$ requires integration over the latent space, which might be computationally and/or analytically intractable. Therefore, instead of directly maximizing the data (log-)likelihood, a surrogate objective called the Evidence Lower BOund (ELBO) (which will be formally introduced in Section 2.3.2) is optimized. Another essential task in the application of LVMs is the identification of the posterior distribution $p(\mathbf{z}|\mathbf{x})$, which is necessary for the inference of the latent variables $\mathbf{z}$ given the observed variable $\mathbf{x}$. Applying Bayes' rule, we can express this as $p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{z})p(\mathbf{x}|\mathbf{z})}{\int p(\mathbf{z})p(\mathbf{x}|\mathbf{z})\mathrm{d}\mathbf{z}}$. As previously discussed, the denominator term which involves the

---

[1] These two methods can also be combined together to construct more powerful generative models.

integration over the latent space is usually intractable. Therefore, the estimation of the posterior distribution often necessitates the use of various approximation methods. In the following sections, we will make a closer examination of different types of LVMs and present the associated methodologies for learning the model parameters and inferring the latent variables. More precisely, in Section 2.2, we primarily concentrate on the introduction of different LVMs. In Section 2.3, we delve into the details of their *solutions*, explaining how to learn the model parameters and infer the posterior distribution of the latent variables.

## 2.2 LATENT VARIABLE MODELS: AN OVERVIEW

As discussed in the previous subsection, LVMs serve two primary purposes. First, LVMs have the capacity to depict complex high-dimensional real-world data distributions by leveraging (a set of) auxiliary latent variable(s). This is achieved by specifying both the prior knowledge on the latent variable(s) and the structural conditional relationships between observed and latent variables. Second, LVMs provide a potential means to gain insights into a given dataset and to understand the underlying data generation mechanism through probabilistic modeling. This is accomplished by inferring the posterior distribution of the latent variable(s) given the observed variable via Bayesian methods. The generative and interpretative aspects of LVMs are two sides of the same coin. As both the parameters of the generative model and the posterior distribution are unknown, these two problems are often solved together. We will provide further explanations on this point in the next section.

Based on the characteristics of both the observed and latent variables, LVMs can be classified into various categories from different perspectives. To begin with, according to the discrete or continuous nature of the observed and latent variables, LVMs can be divided into four distinct groups. In this PhD work, we constraint our scope to the case where the observed variables are continuous. Within this context, we consider the two cases where the latent variables can be either discrete or continuous. *Discrete latent variables* are generally used to represent categorical factors, inducing a clustering over the data samples. For instance, consider a set of images $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ of different animals: cats, dogs, racoons, and elephants. The animal type inside the image can be defined as a discrete latent variable $\mathbf{z}$, which follows a categorical distribution with four possible values. An LVM can be defined to model this image dataset by specifying the prior distribution of the latent variable $p(\mathbf{z})$ and the conditional distribution $p(\mathbf{x}|\mathbf{z})$. However, in many real-world scenarios, the categorical factors are unknown and may not be easily intuitively pre-defined. To unveil these factors, clustering techniques [133] are frequently utilized. On the other hand, *continuous latent variables* are generally employed to represent continuous underlying generative factors, which is often associated to the process of dimensionality reduction. The underlying hypothesis here is that, the high-dimensional data points are supposed to live on (or close to) a manifold of much lower dimensionality than that of the original data space [24]. For instance[2], given a single $100 \times 100$ pixel image $\mathbf{x}_0$, by simply translating and rotating it, we can generate a dataset $\{\mathbf{x}_1, ...\mathbf{x}_n\}$. Although the original image samples in this dataset lie in a 10,000-dimensional data space, the variations of these data points only comprise three degrees of freedom, corresponding to the vertical and horizontal translations and the rotations respectively. Therefore, we can resort to LVMs to learn a low-dimension representation $\mathbf{z}$ of the high-dimensional image data.

Another perspective to classify the LVMs is to consider the properties of the observed variable. In statistical machine learning, most models operate under the common assumption that the observed data samples are independently and identically distributed (i.i.d.). A single global latent variable can be defined to describe the variations within the data sample. While this assumption demonstrate its effectiveness on a great number of “static” data types, such as images, it shows limited capacities when dealing with more structured data types, such as sequential data. Sequential data, such as time series, audio signals, and DNA sequences, are very common in our daily lives. They are supposed to evolve over time by following certain underlying mechanisms or patterns. However, the temporal evolution patterns within each data sample may not be adequately captured under the

---

[2] This example originates from [24], in the introduction of Chapter 12.

i.i.d. assumption with a single global latent variable. Therefore, introducing a sequence of latent variables can be an effective approach to depict the complex temporal dependencies within the sequential data. This has lead to the development of state space model (SSM) [193] and more sophisticated "dynamical" LVMs for sequential data.

Finally, with the remarkable recent advances in deep learning, it is possible to combine the expressive modeling power of DNNs and the interpretability of LVMs to construct richer and more flexible generative models. These models are commonly referred to as deep latent variable model (DLVM). Just like LVMs, DLVMs can be either static or dynamical according to the nature of the observed/latent variables and the presence of temporal dependencies within the model. Moreover, the DLVMs used for dynamical modeling can vary not only according to the nature of the temporal dependencies across successive observed and/or latent variables, but also according to the way these dependencies are implemented with DNNs. We will come back to this principle in more details in Section 2.2.2.

In the following, we present a selection of representative LVMs based on the classification discussed above. We will begin with LVMs for static data, followed by LVMs for sequential data. The goal is to provide a comprehensive overview of the development of LVMs, although this compilation is certainly not exhaustive. Besides, we should note that in the literature, the name of a model is often confused with the name of the solution (e.g. the methods employed to learn the model parameters and to infer the posterior distribution of the latent variables). More specifically, the name of a model solution as a data analysis method often implies the definition of the underlying model, which has its own name. For instance, the factor analysis (FA) method that will be introduced in Section 2.2.1 is indeed a solution for the linear Gaussian model (LGM). We declare that even if the model solutions are presented in Section 2.3, in this section we may employ both names when introducing the model, as is usual in the literature.

### 2.2.1 LVMs FOR STATIC DATA

#### 1. Discrete latent variable

*a) Mixture models.* We start with LVMs for static data with discrete latent variables. A typical case is the **mixture models** [136, 137]. Mixture models find wide applications in numerous signal processing and pattern recognition tasks, where the goal is to model the distribution of a signal that can take several distinct discrete states. Each state is expected to be encoded by a distinct value of the latent variable. The marginal distribution of the observed data is modeled as a linear combination of a finite set of component distributions. Each component distribution, linked to a specific state, is defined by the conditional distribution given the value of the latent variable. The mixing coefficients, which indicate the relative weight of each component, correspond to the prior distribution of the discrete latent variable.

This can be formulated as follows. Unless otherwise specified, in the following, we will use $\mathbf{x}$ to denote the observed variable and $\mathbf{z}$ to denote the latent variable. In the context of mixture models, we suppose that $\mathbf{x}$ can take on $K$ possible states and $\mathbf{z}$ is a discrete scalar variable, hence renoted $z$, taking values in $\{1, ..., K\}$. $z$ is assumed to follow a categorical prior distribution with $p(z = k) = \pi_k$. The marginal distribution of $\mathbf{x}$ can therefore be defined as

$$p(\mathbf{x}; \theta) = \sum_{k=1}^{K} p(z = k) p(\mathbf{x}|z = k) = \sum_{k=1}^{K} \pi_k p(\mathbf{x}; \theta_k). \tag{2.1}$$

Here we use $\theta$ and $\theta_k$ to represent the parameters of the marginal distribution of $\mathbf{x}$ and that of each component conditional distribution, respectively. The component distributions $p(\mathbf{x}; \theta_k)$ can take various forms. In particular, when it is Gaussian, we obtain the famous **Gaussian mixture model (GMM)**:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{2.2}$$

GMM is a very popular probabilistic model in machine learning and have been broadly employed for signal processing and pattern discovery in various domains, especially before the advent of deep learning [24].

Besides, GMM has a strong connection with the K-means clustering algorithm [133].

In fact, clustering a set of data points $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ amounts to associate each data point $\mathbf{x}_n$ with a discrete latent variable $z_n \in \{1, ..., K\}$ that specifies the category $\mathbf{x}_n$ belongs to. Given the observed data point $\mathbf{x}_n$, we can compute the posterior distribution over the latent variable $z_n$ using Bayes' rule:

$$\gamma(z_{nk}) := p(z_n = k|\mathbf{x}_n) = \frac{p(z = k)p(\mathbf{x}_n|z = k)}{\sum_{j=1}^{K} p(z = j)p(\mathbf{x}_n|z = j)} = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \tag{2.3}$$

The quantity $\gamma(z_{nk})$ is referred to as the *responsibility* that component $k$ takes for explaining $\mathbf{x}_n$ [24]. Computing all responsibilities for a data point $\mathbf{x}_n$ can be viewed as *soft assignment* of this data point to the clusters. Otherwise, we can also compute the *hard assignment* for $\mathbf{x}_n$ by taking the category with the maximum responsibility value :

$$\hat{z}_n = \arg\max_k \gamma(z_{nk}). \tag{2.4}$$

If we consider a uniform prior distribution over $z_n$ ($\forall k \in \{1, ..., K\}, p(z_n = k) = \frac{1}{K}$) and spherical Gaussian distributions with $\boldsymbol{\Sigma}_k = \mathbf{I}$, the hard assignment reduces to

$$\hat{z}_n = \arg\min_k ||\mathbf{x}_n - \boldsymbol{\mu}_k||_2^2. \tag{2.5}$$

This is the common case in the K-means clustering algorithm where the assignment is done by searching the minimum euclidean distance.

*b) Vector-quantised variational autoencoder.* Although the mixture models along with the clustering algorithms provide effective tools to discover discrete and compositional data patterns, applying them directly to the raw high-dimensional data can often result to suboptimal performance. With the advancement of deep learning methods, well-designed DNNs can be employed to extract features from the raw data. Subsequently, clustering algorithms can be applied on these learned features, facilitating the discovery of more abstract and potentially semantically meaningful discrete latent representations. Here we present a representative generative model in this type, known as the **Vector-quantised variational autoencoder (VQ-VAE)** [205]. VQ-VAE originates from the variational auto-encoder (VAE), a LVM designed for continuous variables which will be presented in Section 2.2.1. In the original paper [205], VQ-VAE is formulated as a model that leverages vector quantization (VQ) techniques [77] to discretize the latent representations learned by VAEs. In this section, we aim to provide an alternative interpretation of VQ-VAE and establish the connections to GMMs.

VQ-VAE comprises three essential components: an encoder, responsible for encoding the raw input data by reducing its dimension and creating a continuous latent space; a quantizer, designed to discretize the continuous latent space learned with the encoder by learning a codebook of discrete vectors; and a decoder, tasked with reconstructing the input data from the discrete latent embeddings. Both encoder and decoder are based on DNNs. To illustrate the basic conception of VQ-VAE, let us first consider a VQ-VAE designed for vector data, with the encoder encoding the input into a single latent vector. Given an observed data point $\mathbf{x}_n \in \mathbb{R}^d$, the encoder will learn a continuous function $f_\phi(\cdot) : \mathbb{R}^d \to \mathbb{R}^l$ to encode $\mathbf{x}_n$ into a lower dimensional latent space. The continuous encodings of the encoder $f_\phi(\mathbf{x}_n)$ will then be assigned to the nearest discrete latent vector in the codebook by computing the euclidean distance between $f_\phi(\mathbf{x}_n)$ and all vectors in the codebook $\{\mathbf{e}_1, ..., \mathbf{e}_K\}$. This process resembles K-means clustering with hard assignment in the continuous space learned by the encoder. Finally, the assigned latent vector $\mathbf{e}_k$ will be fed into the decoder to reconstruct the original signal: $\hat{\mathbf{x}_n} = g_\theta(\mathbf{e}_k)$.

This process can be interpreted alternatively from a probabilistic perspective with discrete latent variables. We first clarify several notations. Let $\mathbf{x} \in \mathbb{R}^d$ denote the continuous observed data. Let $\mathbf{z} \in \mathbb{R}^K$, which is a K-dimensional one-hot vector with a particular element $z_k$ set to 1 and other elements set to 0, denote the latent assignment variable. Given a data point $\mathbf{x}_n$ and the corresponding assignment variable $\mathbf{z}_n$, $z_{nk} = 1$ means that the continuous encoding $f_\phi(\mathbf{x}_n)$ is assigned to the discrete latent vector $\mathbf{e}_k$. Finally, let $\mathbf{A} \in \mathbb{R}^{K \times l}$ denote all of the codebook vectors. The assigned discrete latent vector can therefore be represented as $\mathbf{e}_k = \mathbf{z}^T \mathbf{A}$. Applying the GMM, the marginal distribution of

$\mathbf{x}$ can be written as:

$$p(\mathbf{x};\theta) = \sum_{\mathbf{z}} p_\theta(\mathbf{z}) p_\theta(\mathbf{x}|\mathbf{z}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{2.6}$$

As we discussed before, in the case of hard-assignment K-means clustering algorithm, the prior of $\mathbf{z}$ is supposed to be uniform while the conditional distributions are supposed to be spherical Gaussians with identity covariance matrix. And the mean vectors can be expressed with the decoder neural networks, giving rise to:

$$p(\mathbf{x};\theta) = \frac{1}{K} \sum_{k=1}^{K} \mathcal{N}(\mathbf{x}|g_\theta(\mathbf{e}_k), \mathbf{I}). \tag{2.7}$$

In that case, the elements of the assignment variable $\mathbf{z}$ can be determined analytically by applying 2.5.

However, in practice, situations are a bit more complicated. As the VQ-VAE is generally applied to image or audio data, the inputs are often encoded into a series of latent variables instead of a single latent variable, achieved through the use of convolutional layers. For instance, consider an image input represented by $\mathbf{x} \in \mathbb{R}^{W \times H \times C}$, where $W$, $H$, $C$ denote the width, height, and channels (dimension) of the image, respectively. The encoder function $f_\phi$ encodes $\mathbf{x}$ into a latent space, yielding $f_\phi(\mathbf{x}) \in \mathbb{R}^{W' \times H' \times l}$, where $W'$, $H'$, $l$ represent the width, height, and channels (dimension) of the resulting latent feature map. Instead of directly applying the quantizer to the entire feature map $f_\phi(\mathbf{x})$, VQ-VAE applies the quantizer to vectors at individual positions of the feature map $f_\phi(\mathbf{x})_{i,j}$. The joint distribution of $\mathbf{x}$ is therefore:

$$p(\mathbf{x};\theta) = \sum_{\mathbf{z}_{1,1},\ldots,\mathbf{z}_{W'H'}} p_\theta(\mathbf{z}_{1,1},\ldots,\mathbf{z}_{W'H'}) p_\theta(\mathbf{x}|\mathbf{z}_{1,1},\ldots,\mathbf{z}_{W'H'}). \tag{2.8}$$

Let $\mathbf{e}_{i,j} = \mathbf{z}_{i,j}^T \mathbf{A}$ denote the assigned discrete latent vector at position $(i,j)$, the conditional distribution can be written as:

$$p(\mathbf{x}|\mathbf{z}_{1,1},\ldots,\mathbf{z}_{W'H'}) = \mathcal{N}(\mathbf{x}|g_\theta([\mathbf{e}_{1,1},\ldots,\mathbf{e}_{W'H'}]), \mathbf{I}). \tag{2.9}$$

The posterior distribution of the assignment variables $\mathbf{z}_{1,1}, ..., \mathbf{z}_{W'H'}$ can no longer be determined analytically. In VQ-VAE, the elements of each assignment variable $\mathbf{z}_{i,j}$ are approximated by:

$$\forall k \in \{1, ..K\}, \hat{z}_{i,j,k} = \begin{cases} 1 & \text{if } k = \arg\min_k ||f_\phi(\mathbf{x})_{i,j} - \mathbf{e}_k||_2^2; \\ 0 & \text{otherwise.} \end{cases} \tag{2.10}$$

During the training phase for both the encoder and the decoder, the prior distribution $p_\theta(\mathbf{z}_{1,1}, ..., \mathbf{z}_{W'H'})$ is treated as factorized uniform distribution, allowing it to be disregarded initially. However, in subsequent stages detailed in VQ-VAE [205, 163], an auto-regressive prior neural network is suggested to be fitted. This step aids the data generation process.

VQ-VAE demonstrated its powerful data analysis-resynthesis and generation capabilities on various data formats, including images, audio signals and videos. And it can also be used to learn interpretable representations by imposing further regularizations on the discrete latent space [54].

### 2. Continuous latent variable

*a) Factor analysis & probabilistic PCA.* When considering a continuous latent space, one straightforward scenario is the linear Gaussian model (LGM). Specifically, we delve into a class of continuous latent space LVMs, where the prior distribution $p(\mathbf{z})$ is Gaussian, along with a Gaussian conditional distribution $p(\mathbf{x}|\mathbf{z})$. The dependency between the observed variable $\mathbf{x}$ and the latent variable $\mathbf{z}$ is supposed to be linear. The solutions to LGMs are usually referred to as **factor analysis (FA)**.

Let $\mathbf{x} \in \mathbb{R}^d$ denote the observed variable and let $\mathbf{z} \in \mathbb{R}^l$ denote the continuous latent variable with latent dimension $l$. Usually, we have $l < d$. In LGM, the prior distribution of the latent variable $\mathbf{z}$ and the conditional distribution $p(\mathbf{x}|\mathbf{z})$ are defined as following:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}), \tag{2.11}$$

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{z} + \boldsymbol{\mu}, \boldsymbol{\Psi}), \tag{2.12}$$

where $\mathbf{W} \in \mathbf{R}^{l \times d}$ is a linear projection matrix, $\boldsymbol{\mu} \in \mathbb{R}^d$ is a translation vector, and $\boldsymbol{\Psi} \in \mathbb{R}^{d \times d}$ is a diagonal covariance matrix. We note that in LGM models, the generative process of the observed variable $\mathbf{x}$ can be viewed as an additive Gaussian noise model. In particular, let $\boldsymbol{\epsilon} \in \mathbb{R}^d$ be a zero-mean Gaussian-distributed noise variable with covariance matrix $\boldsymbol{\Psi}$. And suppose that $\boldsymbol{\epsilon}$ is independent with $\mathbf{W}\mathbf{z} + \boldsymbol{\mu}$. Then, we can obtain $\mathbf{x}$ with:

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}. \tag{2.13}$$

As both the pior and conditional distributions are Gaussian, the marginal distribution of $\mathbf{x}$ is also Gaussian, and can be computed as:

$$p(\mathbf{x}; \theta) = \int p(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) \mathrm{d}\mathbf{z} = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \mathbf{C}), \tag{2.14}$$

with

$$\mathbf{C} = \mathbf{W}\mathbf{W}^T + \boldsymbol{\Psi}, \tag{2.15}$$

and $\theta$ refers to the parameters set $\{\mathbf{W}, \boldsymbol{\mu}, \boldsymbol{\Psi}\}$. We should note that given a marginal distribution $p(\mathbf{x})$, this parametrization is not unique. In fact, it is rotation-invariant in the latent space. To illustrate this, let us consider a matrix $\tilde{\mathbf{W}} = \mathbf{W}\mathbf{R}$, where $\mathbf{R}$ is an arbitrary orthogonal rotation matrix, satisfying $\mathbf{R}\mathbf{R}^T = \mathbf{I}$. The term $\tilde{\mathbf{W}}\tilde{\mathbf{W}}^T$ that appears in the covariance matrix $\mathbf{C}$ satisfies $\tilde{\mathbf{W}}\tilde{\mathbf{W}}^T = \mathbf{W}\mathbf{R}\mathbf{R}^T\mathbf{W}^T = \mathbf{W}\mathbf{W}^T$. Multiplying $\mathbf{W}$ by an orthogonal matrix can be interpreted as rotating the latent variable $\mathbf{z}$ before generating $\mathbf{x}$. As $\mathbf{z}$ follows an isotropic Gaussian prior, the marginal distribution of $\mathbf{x}$ is invariant to this rotation. Therefore, the LGM models are *unidentifiable*.[3]

The posterior distribution $p(\mathbf{z}|\mathbf{x})$ is also Gaussian, and can be derived in closed-form as:

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_{\mathbf{z}|\mathbf{x}}, \boldsymbol{\Sigma}_{\mathbf{z}|\mathbf{x}}), \tag{2.16}$$

with

$$\boldsymbol{\Sigma}_{\mathbf{z}|\mathbf{x}} = (\mathbf{I} + \mathbf{W}^T \boldsymbol{\Psi}^{-1} \mathbf{W})^{-1}. \tag{2.17}$$

---

[3] In statistics, a parametric model $\mathcal{P} = \{P_\theta : \theta \in \Theta\}$ with parameter space $\Theta$ is said to be identifiable if the mapping $\theta \mapsto P_\theta$ is one-to-one: $\forall \theta_1, \theta_2 \in \Theta, P_{\theta_1} = P_{\theta_2} \Rightarrow \theta_1 = \theta_2$.

$$\boldsymbol{\mu}_{\mathbf{z}|\mathbf{x}} = \boldsymbol{\Sigma}_{\mathbf{z}|\mathbf{x}} \mathbf{W}^T \boldsymbol{\Psi}^{-1} (\mathbf{x} - \boldsymbol{\mu}), \tag{2.18}$$

The **probabilistic principal components analysis (PPCA)** [198] is a special case of FA where the additive noise in Equation 2.13 is supposed to be isotropic, which means that the covariance matrix $\boldsymbol{\Psi}$ becomes $\sigma^2\mathbf{I}$. PPCA is the probabilistic version of the principal components analysis (PCA). PCA is a technique that is widely applied across various data processing domains, including dimensionality reduction, data compression, feature extraction and data visualization [99]. It was originally defined as an orthogonal linear projection of the data into a reduced-dimensional space, referred to as the *principal subspace*, with the aim of maximizing the variance of the projected data [90]. While as we have discussed, PCA can also be formulated in a probabilistic perspective within the LVM framework. In fact, the PPCA can be interpreted as the solution to the LGM when the number of free parameters is significantly reduced while still allowing the model to capture the dominant correlations in the data set [24].

*b) Independent component analysis.* We may also consider models with linear dependency between the observed and latent variables but non-Gaussian latent distributions. An important class of solutions for such models are the **independent component analysis (ICA)** [39, 32, 95, 12]. ICA is typically used to solve the *blind source separation* problem. Consider a situation where two people speak at the same time and we record their voices with two distinct microphones. The signals recorded by each of the microphone at each time step $t$ can be gathered in the observed vector $\mathbf{x}_t$, and represented as a linear combination of the (unknown) amplitudes of the 'source' signals emitted by the two speakers at the same time step $t$, and gathered in the latent vector $\mathbf{z}_t$:

$$\mathbf{x}_t = \mathbf{A}\mathbf{z}_t, \tag{2.19}$$

with $\mathbf{A} \in \mathbb{R}^{2\times 2}$ an invertible mixing matrix. ICA applies to the case where the number of source signals is equal to or greater than the number of recorded signals, as is verified in our example. Under these assumptions, if we can uniquely estimate the entries of $\mathbf{A}$ from the observed data samples, then we can invert the mixing process and obtain the unobserved source signals.

We note that Equation 2.19 can be obtained from Equation 2.13 by setting both $\boldsymbol{\mu}$ and $\boldsymbol{\epsilon}$ to the zero vector. If we choose to use Gaussian prior and conditional distributions, we fall into the LGMs and their corresponding solutions FA. However, as we discussed previously, the FA solutions are unidentifiable with the isotropic Gaussian prior due to the rotation invariant property. This implies that we are unable to uniquely determine the values of $\mathbf{A}$. Therefore, ICA requires that the latent variable $\mathbf{z}$ follows an independent, non-Gaussian prior distribution. In ICA models, we generally consider a distribution that factorizes over the latent variables, such that

$$p(\mathbf{z}) = \prod_{j=1}^{d} p(z_j), \tag{2.20}$$

and each of the factorized latent variable distribution is chosen as a heavy-tailed distribution, for example a Laplace distribution. This can be shown to enable the identifiability of the model [39].

*c) Variational autoencoder.* The LVMs with continuous latent variables that we have discussed so far focus on applying linear dependencies between the observed and latent variables, which allows for easy derivation of model solutions. For instance, we can easily compute tractable marginal and posterior distributions for the LGMs. Nonetheless, linear models might be too simple and may not be able to efficiently capture the complexity of real-world data distributions. The advent of deep learning offers the potential to extend these models into non-linear structures with the help of DNNs. The **variational autoencoder (VAE)** [107, 169] presents a good example of this type of models.

In the VAE, the latent vector $\mathbf{z} \in \mathbb{R}^l$ is generally assumed to follow a standard Gaussian prior distribution:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I}). \tag{2.21}$$

The conditional distribution can be chosen according to the nature of the observed variable $\mathbf{x} \in \mathbb{R}^d$. It can be, for instance, a multivariate Gaussian distribution with diagonal covariance matrix:

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}\big(\mathbf{x}|\boldsymbol{\mu}_\theta(\mathbf{z}), \operatorname{diag}(\boldsymbol{\sigma}^2_\theta(\mathbf{z}))\big). \tag{2.22}$$

The parameters of this distribution, here the mean vector $\boldsymbol{\mu}_\theta(\cdot)$ and the standard deviation vector $\boldsymbol{\sigma}_\theta(\cdot)$, are non-linear functions of $\mathbf{z}$ modeled by a DNN called the *decoder network* and parameterized with parameters $\theta$.

The posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ corresponding to the above model does not have an analytical expression. Following the variational inference principles [213], an approximate posterior distribution $q_\phi(\mathbf{z}|\mathbf{x})$ is introduced. A common choice for $q_\phi$ is to use a multivariate Gaussian distribution with diagonal covariance matrix:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}\big(\mathbf{z}|\boldsymbol{\mu}_\phi(\mathbf{x}), \mathrm{diag}(\boldsymbol{\sigma}^2_\phi(\mathbf{x}))\big), \tag{2.23}$$

where the mean and standard deviation vectors $\boldsymbol{\mu}_\phi(\cdot)$ and $\boldsymbol{\sigma}_\phi(\cdot)$ are non-linear functions of $\mathbf{x}$ implemented by another DNN called the *encoder network* and parameterized by $\phi$.

As the VAE allows to extract non-linear embeddings from the observed data, it can be employed to learn more complex data representations. Actually, different variants of VAE have been explored to enhance its ability in acquiring richer and more expressive latent representations, see, e.g., [85, 236, 105].

### 2.2.2 LVMs for Sequential Data

As mentioned before, sequential data involve recordings of observed signals over time. As we mentioned before, directly treating the entire sequences as i.i.d. samples and learning a single global latent variable for each sequence can lead to suboptimal model performance, especially for tasks involving temporal predictions. In this subsection, we present a branch of LVMs that leverage a sequence of latent variables $\{\mathbf{z}_1, ..., \mathbf{z}_T\}$ to model the temporal dependencies within a given observed sequence $\{\mathbf{x}_1, ..., \mathbf{x}_T\}$, resulting in dynamical LVMs. We start by considering dynamical LVMs with discrete latent variables, and further extend our exploration to dynamical LVMs with continuous latent variables.

**1. Discrete latent variable**

*a) Hidden Markov models.* One of the simplest way to model the dynamical properties of an observed sequence $\{\mathbf{x}_1, ..., \mathbf{x}_T\}$ via a sequence of discrete latent variables $\{\mathbf{z}_1, ..., \mathbf{z}_T\}$ is to employ a **hidden Markov model (HMM)** [157]. Let us consider here a scalar latent variable $z_t \in \{1, ..., K\}$ (the approach can be generalized to a vector latent variable). The latent variables $z_{1:T}$ are supposed to follow a first-order Markov transition model. The observations can either be discrete with $\mathbf{x}_t \in \{1, ..., N_{\mathbf{x}}\}$, or continuous with $\mathbf{x}_t \in \mathbb{R}^d$. At each time step $t$, the model effectively corresponds to a mixture model with component distributions given by $p(\mathbf{x}_t|z_t)$. The difference with the mixture models mentioned in Section 2.2.1 lies in the fact that the latent variables $z_t$ are no longer independent over time; instead, they exhibit temporal correlations modeled with specific conditional distributions conditioned on $z_{t-1}$.

Let us consider the equations formalizing the above principles. In the HMM, the joint distribution of both observed and latent variable sequences can be expressed as follows:

$$p(\mathbf{x}_{1:T}, z_{1:T}) = p(z_1) \prod_{t=2}^{T} p(z_t|z_{t-1}) \prod_{t=1}^{T} p(\mathbf{x}_t|z_t). \tag{2.24}$$

The initial latent variable is assumed to follow a K-categorical distribution, with $p(z_1 = k) = \pi_k$. The *transition model* between successive latent variables is usually defined by a conditional probability table:

$$p(z_t = i|z_{t-1} = k) = A_{ik}, \tag{2.25}$$

where $A_{ik}$ is the entry of the *transition matrix* $\mathbf{A}$, satisfying $0 \leq A_{ik} \leq 1$ and $\sum_{k=1}^{K} A_{ik} = 1$. The conditional distribution $p(\mathbf{x}_t|z_t)$, often referred to as the *observation model* in the signal processing / control theory literature, can take various forms depending on the nature of the observed data. In the case of continuous data, a common choice is to use the Gaussian distribution:

$$p(\mathbf{x}_t|z_t = k) = \mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{2.26}$$

Besides, we can also consider more flexible observation models. For example, we can use a M-component GMM:

$$p(\mathbf{x}_t|z_t = k) = \sum_{m=1}^{M} w_{km} \mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_{km}, \boldsymbol{\Sigma}_{km}). \tag{2.27}$$

This model is usually mentioned as the HMM-GMM model [157, 156].

The posterior distributions of the latent variables $p(z_t|\mathbf{x}_{1:T})$ can be analytically computed with the forward-backward algorithm [156] or the Baum-Welch algorithm [11].

**2. Continuous latent variable**

*a) Linear dynamical systems.* If we consider that the observed sequential data are governed by continuously varying hidden states (or generative factors) rather than discrete ones, we can define sequential LVMs with continuous latent variables. The simplest and most commonly-used models of this type is the **linear(-Gaussian) dynamical system (LDS)** [68]. Similar to HMM, the LDS also imposes a first-order Markov chain on the latent variables. However, the transition distributions are no longer depicted by a discrete transition table; instead, they are defined by an LGM (see Section 2.2.1 for the introduction of LGM in this manuscript). Additionally, an LDS generally assumes that the observed variable at each time step $t$ follows a conditional Gaussian distribution, with the mean being a linear function of the latent variable at the same time step, i.e. another LGM similar to the one described in Section 2.2.1. The same way as a HMM can be viewed as a sequential extension of the mixture models, a LDS can be viewed as a sequential extension of the "static" LGM.

Let us formulate the above principles mathematically. The transition and observation models of LDS are generally defined as following:

$$p(\mathbf{z}_t|\mathbf{z}_{t-1}) = \mathcal{N}(\mathbf{z}_t|\mathbf{A}_t\mathbf{z}_{t-1}, \mathbf{\Gamma}_t), \tag{2.28}$$

$$p(\mathbf{x}_t|\mathbf{z}_t) = \mathcal{N}(\mathbf{x}_t|\mathbf{C}_t\mathbf{z}_t, \mathbf{\Sigma}_t), \tag{2.29}$$

where $\mathbf{A}_t$ defines the *transition matrix*, $\mathbf{\Gamma}_t$ represents the *process noise*, $\mathbf{C}_t$ defines the *observation matrix*, and $\mathbf{\Sigma}_t$ represents the *observation noise*. If we consider a stationary model, the transition and observation models are assumed to be constant over time, which implies $\mathbf{A}_t = \mathbf{A}$, $\mathbf{\Gamma}_t = \mathbf{\Gamma}$, $\mathbf{C}_t = \mathbf{C}$ and $\mathbf{\Sigma}_t = \mathbf{\Sigma}$. The initial latent variable is also assumed to follow a Gaussian distribution:

$$p(\mathbf{z}_1) = \mathcal{N}(\mathbf{z}_1|\boldsymbol{\mu}_0, \mathbf{V}_0). \tag{2.30}$$

Similar to the LGM, we can express this generative process in an equivalent form with additive Gaussian noises:

$$\mathbf{z}_t = \mathbf{A}_t \mathbf{z}_{t-1} + \boldsymbol{\epsilon}_t, \quad \text{with} \quad \boldsymbol{\epsilon}_t \sim \mathcal{N}(\boldsymbol{\epsilon}_t | \mathbf{0}, \boldsymbol{\Gamma}_t), \tag{2.31}$$

$$\mathbf{x}_t = \mathbf{C}_t \mathbf{z}_t + \boldsymbol{\eta}_t, \quad \text{with} \quad \boldsymbol{\eta}_t \sim \mathcal{N}(\boldsymbol{\eta}_t | \mathbf{0}, \boldsymbol{\Sigma}_t), \tag{2.32}$$

$$\mathbf{z}_1 = \boldsymbol{\mu}_0 + \mathbf{u}, \quad \text{with} \quad \mathbf{u} \sim \mathcal{N}(\mathbf{u} | \mathbf{0}, \mathbf{V}_0). \tag{2.33}$$

In that case where all distributions are Gaussian with linear dependencies, the LDS has an analytical solution, i.e. there is a closed-form expression of the posterior distribution of the latent variables, known as Kalman filter [101] (for the causal solution) and Kalman smoother (for the non-causal solution) [162].

However, the linear-Gaussian assumption can be a significant limitation for real-world signals with complex dynamics. A widely-used generalization of the Kalman filter to sequential data with non-linear dynamics is the extended Kalman filter [50, 232], which is a first-order Gaussian approximation to the Kalman filter based on local linearization using the Taylor series expansion. Another interesting extension is to combine a set of LDSs with an HMM, resulting in the switching state space model (also named switching Kalman filter) [139, 69]. This model segments the sequential data into different regimes, each regime being modeled by a LDS, and the succession of regimes is ruled by the HMM.

*b) Dynamical variational autoencoders.* Recently, DNNs, and in particular RNNs, have been used within DPGM structures to model sequential data. In this line, the **dynamical variational auto-encoder (DVAE)** are a family of powerful DPGMs that extend the VAE to model complex non-linear temporal dependencies within a sequence of observed data vectors and corresponding (continuous) latent vectors [71]. Roughly speaking, DVAE models can be viewed as combining a VAE with temporal models such as RNNs and/or SSMs.[4] However, it is important to highlight the fundamental distinction between DVAEs and conventional SSMs. DVAEs break the Markovian dependencies

---

[4] The distinction between RNNs and SSMs mainly lies in the preservation of Markovian properties. Due to their definition, SSMs maintain the Markovian properties while RNNs, with their sophisticated designs, have the potential to surpass Markovian limitations, enabling the modeling of longer temporal correlations.

within the latent states and the single-frame latent-observation dependencies, thereby facilitating more flexible relationships between latent and observed variables.

Using the chain rule, the most general DVAE generative distribution can be written as the following causal generative process:

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}), \tag{2.34}$$

where $p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})$ and $p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})$ are arbitrary generative distributions, the parameters of which are provided sequentially by RNNs taking the respective conditioning variables as inputs (and $\theta_\mathbf{x}$ and $\theta_\mathbf{z}$ denote the respective parameters of these RNNs). A common choice is to use Gaussian distributions with diagonal covariance matrices:

$$p_{\theta_\mathbf{x}}(\mathbf{x}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}) = \mathcal{N}\big(\mathbf{x}_t; \boldsymbol{\mu}_{\theta_\mathbf{x}}(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}), \mathrm{diag}(\boldsymbol{v}_{\theta_\mathbf{x}}(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}))\big), \tag{2.35}$$

$$p_{\theta_\mathbf{z}}(\mathbf{z}_t|\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}) = \mathcal{N}\big(\mathbf{z}_t; \boldsymbol{\mu}_{\theta_\mathbf{z}}(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}), \mathrm{diag}(\boldsymbol{v}_{\theta_\mathbf{z}}(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t-1}))\big). \tag{2.36}$$

It can be noted that the prior distribution of $\mathbf{z}_t$ is more complex than the standard Gaussian used in the vanilla VAE. Also, the different models belonging to the DVAE class differ in the possible conditional independence assumptions that can be made in (2.34).

Similarly to the VAE, the exact posterior distribution $p_\theta(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$ corresponding to the DVAE generative model is not analytically tractable. Again, an inference model $q_{\phi_\mathbf{z}}(\mathbf{z}_{1:T}|\mathbf{x}_{1:T})$ is defined to approximate the exact posterior distribution. This inference model factorises as:

$$q_{\phi_\mathbf{z}}(\mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} q_{\phi_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}), \tag{2.37}$$

Again, the Gaussian distribution with a diagonal covariance matrix is generally used:

$$q_{\phi_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}) = \mathcal{N}\big(\mathbf{z}_t; \boldsymbol{\mu}_{\phi_\mathbf{z}}(\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}), \mathrm{diag}(\boldsymbol{v}_{\phi_\mathbf{z}}(\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}))\big), \tag{2.38}$$

where the mean and variance vectors are provided by RNNs (the encoder networks) taking $(\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T})$ as input and parameterized by $\phi_\mathbf{z}$. With the most general generative model defined in Equation 2.34, the conditional distribution in Equation 2.37 cannot be simplified. However, if conditional independence assumptions are made in (2.34), the dependencies in $q_{\phi_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T})$ can be simplified using the D-separation method [24, 64], see [71] for details. In addition, we can force the inference model to be causal by replacing $\mathbf{x}_{1:T}$ with $\mathbf{x}_{1:t}$ in Equation 2.37 and Equation 2.38. This is particularly suitable for online processing.

*c) Gaussian process variational autoencoder & latent stochastic differential equations.* All of the LVMs for sequential data that have been discussed so far fall within the category of discrete-time systems. By discretizing the time index into uniform intervals, these models commonly assume regular sampling rates for sequences. However, in real-world scenarios, typically with multi-sensors, the collected sequential data may be sampled at irregular time intervals and may include missing values due to the asynchrony of the sensors. Directly applying discrete-time models to these data may not accurately depict the temporal variations of the variables. In such cases, it becomes crucial to consider the continuously varying underlying dynamical process. Here, we rapidly present two prominent LVMs designed for modeling sequential data with continuous-time latent states: the **Gaussian process variational autoencoder (GP-VAE)** [53, 239] and the **latent stochastic differential equations (Latent SDE)** [120, 180].

Let $\mathbf{x}_{\mathcal{T}} \in \mathbb{R}^{T\times d} = \left\{\mathbf{x}_\tau | \tau \in \mathcal{T} = \{t_0, t_1, ..., t_T\}\right\}$ denote a sequence of observed variables measured at timestamps $\mathcal{T} = \{t_0, t_1, ..., t_T\}$ which may consist of irregular time intervals, and we have $t_0 < t_1 < ... < t_T$. Let $\mathbf{z}_{\mathcal{T}} \in \mathbb{R}^{T\times l} = \left\{\mathbf{z}_\tau | \tau \in \mathcal{T} = \{t_0, t_1, ..., t_T\}\right\}$ denote the sequence of corresponding latent variables. The joint distribution $p(\mathbf{x}_{\mathcal{T}}, \mathbf{z}_{\mathcal{T}})$ is supposed to be decomposed as:

$$p_\theta(\mathbf{x}_{\mathcal{T}}, \mathbf{z}_{\mathcal{T}}) = p_{\theta_{\mathbf{z}}}(\mathbf{z}_{\mathcal{T}}) \prod_{\tau\in\mathcal{T}} p_{\theta_{\mathbf{x}}}(\mathbf{x}_\tau|\mathbf{z}_\tau). \tag{2.39}$$

The conditional distribution $p(\mathbf{x}_\tau|\mathbf{z}_\tau)$ is the observation model and is supposed to be a multivariante Gaussian distribution:

$$p_{\theta_{\mathbf{x}}}(\mathbf{x}_\tau|\mathbf{z}_\tau) = \mathcal{N}(\mathbf{x}_\tau|\mathbf{g}_{\theta_{\mathbf{x}}}(\mathbf{z}_\tau), \sigma^2\mathbf{I}), \tag{2.40}$$

with $\mathbf{g}_{\theta_\mathbf{x}}(\cdot) : \mathbb{R}^l \rightarrow \mathbb{R}^d$ a non-linear function parameterized by $\theta_\mathbf{x}$ that maps the latent variable to the observation. Similarly to the VAE, it is usually implemented by the decoder neural network. In parallel, the distribution $p_{\theta_\mathbf{z}}(\mathbf{z}_\mathcal{T})$ is designed to integrate prior knowledge about the dynamic nature of the sequence, which is expected to adhere to a continuous-time dynamical system. More specifically, the latent variables $\mathbf{z}_\mathcal{T}$ are assumed to follow a Gaussian process prior with respect to time $t$ [161]: $\mathbf{z}(t) \sim \mathcal{GP}(\mathbf{m}(t), \mathbf{k}(t, t'))$.[5] This Gaussian process involves a mean function vector $\mathbf{m}(\cdot) : \mathbb{R} \rightarrow \mathbb{R}^l$ [6] and a covariance function vector $\mathbf{k}(\cdot, \cdot)$ with pre-defined positive-definite kernel functions. More specifically, the covariance function vector can either be designed as $k \otimes \mathbf{I}$, which means that different dimensions of $\mathbf{z}_\mathcal{T}$ share the same kernel; or as a vector of kernels $\mathbf{k} = (k_1, ..., k_l)^T$, meaning that different kernels are applied to different dimensions of $\mathbf{z}_\mathcal{T}$ [97, 23]. The parameters of the mean and covariance functions are denoted as $\theta_\mathbf{z}$. The true posterior distribution $p(\mathbf{z}_\mathcal{T}|\mathbf{x}_\mathcal{T})$ is often analytically intractable due to the non-linear dependency between $\mathbf{x}_\mathcal{T}$ and $\mathbf{z}_\mathcal{T}$. Therefore, similarly to the VAE and DVAEs, an inference model $q_\phi(\mathbf{z}_\mathcal{T}|\mathbf{x}_\mathcal{T})$ needs to be defined to approximate the exact posterior distribution. With the aim of increasing approximation accuracy and decreasing computational complexity, various approximation methods are proposed [33, 53, 96]. For instance, [53] proposed an approximated inference model as follows:

$$q_\phi(\mathbf{z}_\mathcal{T}|\mathbf{x}_\mathcal{T}) = \prod_{i=1}^{l} q_\phi(\mathbf{z}^i_\mathcal{T}|\mathbf{x}_\mathcal{T}), \tag{2.41}$$

with

$$q_\phi(\mathbf{z}^i_\mathcal{T}|\mathbf{x}_\mathcal{T}) = \mathcal{N}(\mathbf{z}^i_\mathcal{T}|\boldsymbol{\mu}^i_\phi(\mathbf{x}_\mathcal{T}), \mathbf{\Lambda}^{i^{-1}}_{\phi}(\mathbf{x}_\mathcal{T})). \tag{2.42}$$

The precision matrix $\mathbf{\Lambda}^i$ is chosen to be parametrized as a product of bi-diagonal matrices:

$$\mathbf{\Lambda}^i := \mathbf{B}^{i^T}\mathbf{B}^i, \text{with } \mathbf{B}^i_{tt'} = \begin{cases} b^i_{tt'} & \text{if } t' \in \{t, t+1\} \\ 0 & \text{otherwise} \end{cases}. \tag{2.43}$$

---

[5] According to the definition of Gaussian process, the finite latent sequence $\mathbf{z}_\mathcal{T}$ is supposed to follow a joint Gaussian distribution.

[6] Without loss of generality, the mean function vector $\mathbf{m}(\cdot)$ is usually set to $\mathbf{0}$.

This approximation breaks the dependencies across different dimensions of $\mathbf{z}$ while maintaining a Markov-like temporal dependency. Furthermore, both the mean vector and the precision matrix are implemented with an encoder network parameterized by $\phi$.

Another approach to continuous-time dynamical modeling involves applying a SDE on the latent space. In this context, the latent variables $\mathbf{z}(t)$ are supposed to follow a prior distribution governed by an SDE [120]:

$$d\mathbf{z} = \mathbf{f}_\theta(\mathbf{z}, t)dt + \mathbf{L}(\mathbf{z}, t)d\boldsymbol{\beta}, \tag{2.44}$$

where $\mathbf{f}_\theta(\mathbf{z}, t)$ is the drift function parameterized by neural networks, $\mathbf{L}(\mathbf{z}, t)$ is the dispersion matrix, $\boldsymbol{\beta}$ is a Brownian motion with diffusion matrix $\mathbf{Q}$. The initial condition is $\mathbf{z}(t_0) = \mathbf{z}_0$. As the observation model $p_{\theta_\mathbf{x}}(\mathbf{x}_\tau|\mathbf{z}_\tau)$ is parameterized by neural networks, the posterior distributions of $\mathbf{z}(t)$ are usually intractable. They can be approximated, for instance, by another SDE. Furthermore, if the SDEs are linear time-invariant, the latent SDE models can be connected to the GP-VAE models with Markovian kernels [239]. In that case, the latent states can be derived using Kalman filtering and Kalman smoothing, which can largely reduce the computation complexity of the GP-based methods from $\mathcal{O}(T^3)$ to $\mathcal{O}(T)$.

## 2.3 LEARNING AND INFERENCE OF THE LATENT VARIABLE MODELS

In the previous section, we have presented a branch of LVMs that employ different types of latent variables to model different types of data. We recall that one of the primary goals of LVMs is to infer the posterior distributions of the latent variables. While some LVMs allow for analytical solutions, others do not, as briefly mentioned earlier. In this section, we will systematically address this aspect and introduce the general principles for learning and inference in LVMs.

### 2.3.1 POSTERIOR INTRACTABLILITY AND APPROXIMATE INFERENCE

Given a set of datapoints $\mathcal{D} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ and a parametric probabilistic model $p_\theta$ with parameters $\theta \in \Theta$, we would like to find the optimal values of the parameters $\theta$ that best fit the model with the dataset. In statistical machine learning, this is usually done by maximizing the loglikelihood $p_\theta(\mathcal{D})$. In the context of LVMs, computing the marginal likelihood $p_\theta(\mathbf{x})$ involves integrating the joint distribution $p_\theta(\mathbf{x}, \mathbf{z})$ over the latent variable $\mathbf{z}$. This is often a difficult task. As we have seen in the LVM examples in the previous section, the joint distribution can be factorized as $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})$. When the latent variable $\mathbf{z}$ is continuous, analytical computation of the integration is feasible in certain specific scenarios with conjugate priors, particularly when the dependencies between $\mathbf{x}$ and $\mathbf{z}$ are linear [141]. Conjugate prior means that both the prior $p_\theta(\mathbf{z})$ and the likelihood $p_\theta(\mathbf{x}|\mathbf{z})$ share the same functional form. This often occurs when both distributions belong to the exponential family. Otherwise, we say that the marginal likelihood is *analytically intractable*. When the latent variable $\mathbf{z}$ is discrete, computing the sum across all potential values of $\mathbf{z}$ is theoretically feasible. However, in practice, the summation may scale exponentially and be extremely expensive. In such a case, we say that the marginal likelihood is *computationally intractable*.

As we mentioned in the previous sections, in addition to estimating the model parameters, we are also interested in inferring the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$. According to Bayes' rule: $p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{x})}$, if the marginal likelihood $p_\theta(\mathbf{x})$ is intractable, the posterior distribution is not tractable either. Therefore, we need to resort to **approximate inference** methods to solve the problem. Various approximation approaches exist and differ from each other in their trade-off between accuracy, speed, and applicability. These methods generally fall into two categories: **variational inference (VI)** [213, 40] and **Markov Chain Monte Carlo (MCMC)** sampling [82, 65, 174]. In this dissertation, we primarily focus on the variational inference methods.

### 2.3.2 A GENERAL INTRODUCTION TO VARIATIONAL INFERENCE

Variational methods represent a very broad umbrella encompassing diverse techniques that aim at addressing functional[7] optimization problems. In the realm of variational inference, the fundamental concept involves seeking a distribution $q(\mathbf{z})$ that minimizes

---

[7] A functional is defined as a mapping from a space $\mathbf{x}$ into the field of real or complex numbers.

the discrepancy between this proposed distribution and the exact posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ [213, 40]. The typically used measure of discrepancy for optimization is the Kullback-Leibler (KL) divergence $D_{\text{KL}}\big(q(\mathbf{z})||p_\theta(\mathbf{z}|\mathbf{x})\big)$, defined by

$$\begin{aligned} D_{\text{KL}}\big(q(\mathbf{z})||p_\theta(\mathbf{z}|\mathbf{x})\big) &= \mathbb{E}_{q(\mathbf{z})}\big[\log q(\mathbf{z}) - \log p_\theta(\mathbf{z}|\mathbf{x})\big] \\ &= \mathbb{E}_{q(\mathbf{z})}\big[\log q(\mathbf{z}) - \log p_\theta(\mathbf{x},\mathbf{z}) + \log p_\theta(\mathbf{x})\big]. \end{aligned} \tag{2.45}$$

As the term $\log p_\theta(\mathbf{x})$ is independent with $q(\mathbf{z})$, we can take it out of the expectation. By moving the $D_{\text{KL}}$ term to the right side of the equation and the expectation term to the left side, we obtain

$$\mathbb{E}_{q(\mathbf{z})}\big[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z})\big] = \log p_\theta(\mathbf{x}) - D_{\text{KL}}\big(q(\mathbf{z})||p_\theta(\mathbf{z}|\mathbf{x})\big). \tag{2.46}$$

As the KL divergence is always greater than zero, the expectation term on the left-hand side is always smaller than the marginal log-likelihood (which is also called ‘evidence’ in Bayes’ formula). Therefore, the expectation term is a lower bound of the evidence and is named the Evidence Lower BOund (ELBO). On the one hand, since $\log p_\theta(\mathbf{x})$ is independent of $q(\mathbf{z})$, minimizing the KL divergence w.r.t. $q$ is equivalent to maximizing the ELBO w.r.t. $q$. On the other hand, searching for the optimal parameters $\theta$ that maximize the intractable marginal log-likelihood $\log p_\theta(\mathbf{x})$ can be transformed into searching for $\theta$ that maximizes the ELBO. Therefore, the objective of fitting the model is to find the optimal $\theta$ and $q$ that jointly maximize the ELBO:

$$\mathcal{L}(\theta, q; \mathbf{x}) = \mathbb{E}_{q(\mathbf{z})}\big[\log p_\theta(\mathbf{x},\mathbf{z}) - \log q(\mathbf{z})\big]. \tag{2.47}$$

When the exact posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ can be expressed in closed-form and is computationally tractable, the maximum of the ELBO w.r.t. $q$ is achieved when $q(\mathbf{z})$ equals to this exact posterior distribution. In that case, the KL divergence vanishes, allowing for *exact inference*. In such a case, we can solve the optimization problem over $\theta$ by applying the **expectation-maximization (EM)** algorithm [43, 24]. Otherwise, we need to propose a restricted distribution family for $q(\mathbf{z})$ and seek the member of this family that minimizes the KL divergence. The key point of success for VI is to find a family of $q(\mathbf{z})$ functions that is tractable, easy to sample from, and flexible enough so that it can provide a good approximation to the exact posterior distribution. Based on the constraints that we impose on this family of functions, the VI approaches can be broadly categorized into two groups: the free-form VI with mean-field approximation and the fixed-form VI employing gradient-based optimization. In the following subsections, we first rapidly present the EM algorithm, and then we present the free-form and fixed-form VI methods in more detail.

### 2.3.3 EXACT INFERENCE AND EM ALGORITHMS

The EM algorithm is an iterative optimization procedure that maximizes the ELBO w.r.t. the inference distribution $q$ (E-step) and the model parameters $\theta$ (M-step) respectively [43, 24]. It generally starts with an arbitrary value $\theta^{\text{old}}$. During the E-step, by setting $q(\mathbf{z}) = p_{\theta^{\text{old}}}(\mathbf{z}|\mathbf{x})$, it cancels out the KL term of Equation 2.46 with $\theta = \theta^{\text{old}}$. As a consequence, $\mathcal{L}(\theta, q; \mathbf{x})$ becomes a tight lower bound of the log-likelihood to be maximized. As for the M-step, the ELBO becomes:

$$\mathcal{L}(\theta, q; \mathbf{x}) = \mathbb{E}_{p_{\theta^{\text{old}}}(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}, \mathbf{z})\big] - \mathbb{E}_{p_{\theta^{\text{old}}}(\mathbf{z}|\mathbf{x})}\big[p_{\theta^{\text{old}}}(\mathbf{z}|\mathbf{x})\big]. \tag{2.48}$$

The second term in the right-hand side of Equation 2.48 is the negative entropy of $p_{\theta^{\text{old}}}(\mathbf{z}|\mathbf{x})$ and is independent of $\theta$. Therefore, maximizing the ELBO amounts to maximizing the first term in the right-hand side, which is often referred to as "the Q function" in the EM literature and is denoted $\mathcal{Q}(\theta, \theta^{\text{old}})$. In summary, the exact EM algorithm starts by setting $\theta = \theta^{\text{old}}$ and then alternates between the two following steps.

- E-Step: Set $q(\mathbf{z}) = p_{\theta^{\text{old}}}(\mathbf{z}|\mathbf{x})$ and then compute $\mathcal{Q}(\theta, \theta^{\text{old}}) = \mathbb{E}_{p_{\theta^{\text{old}}}(\mathbf{z}|\mathbf{x})}\big[\log p_\theta(\mathbf{x}, \mathbf{z})\big]$,
- M-Step: Estimate $\theta^{\text{new}} = \arg\max_\theta \mathcal{Q}(\theta, \theta^{\text{old}})$ and then set $\theta^{\text{old}} = \theta^{\text{new}}$.

### 2.3.4 FREE-FORM VI AND MEAN-FIELD APPROXIMATION

One way to approximate the posterior distribution is to break the dependencies between sub-groups of latent variables and factorize the approximate posterior distribution into the product of corresponding terms:

$$q(\mathbf{z}) = \prod_{j=1}^{J} q_j(\mathbf{z}_j). \tag{2.49}$$

The factorized form of variational inference is often referred to as the mean-field approximation [153], which was first developed in physics. We should emphasize that in the mean-field approximation, the sole assumption is the independence between the subgroups of latent variables. There is no necessity to make further assumptions regarding the specific functional form for each $q_j$. In fact, the optimal distribution form can be derived directly by maximizing the ELBO w.r.t. each distribution group. Therefore, the mean-field approximation methods are also referred to as free-form VI [141]. In the following, we will see how to iteratively obtain the optimal solution for each $q_j$.

Let us substitute Equation 2.49 into Equation 2.47, resulting in:

$$\mathcal{L}(\theta, q_1, ..., q_J; \mathbf{x}) = \mathbb{E}_{\prod_j q_j(\mathbf{z}_j)}\big[\log p_\theta(\mathbf{x}, \mathbf{z})\big] - \sum_{j=1}^{J} \mathbb{E}_{q_j(\mathbf{z}_j)}\big[\log q_j(\mathbf{z}_j)\big]. \tag{2.50}$$

When optimizing $\mathcal{L}(\theta, q_1, ..., q_J; \mathbf{x})$ w.r.t. $q_j$, we only need to keep the terms related with $q_j$ and write the other terms as constants:

$$\mathcal{L}(q_j; \mathbf{x}) = \mathbb{E}_{q_j(\mathbf{z}_j)}\Big[\mathbb{E}_{\prod_{i\neq j} q_i(\mathbf{z}_i)}\big[\log p_\theta(\mathbf{x}, \mathbf{z})\big]\Big] - \mathbb{E}_{q_j(\mathbf{z}_j)}\Big[\log q_j(\mathbf{z}_j)\Big] + const. \tag{2.51}$$

If we introduce a new distribution $\tilde{p}(\mathbf{x}, \mathbf{z}_j) = \frac{\exp(\mathbb{E}_{\prod_{i\neq j} q_i(\mathbf{z}_i)}\big[\log p_\theta(\mathbf{x},\mathbf{z})\big])}{Z_j}$, with $Z_j$ the normalization constant, then, Equation 2.51 can be written as the negative KL divergence between $q_j(\mathbf{z}_j)$ and $\tilde{p}(\mathbf{x}, \mathbf{z}_j)$:

$$\mathcal{L}(q_j; \mathbf{x}) = -D_{\mathrm{KL}}(q_j(\mathbf{z}_j)||\tilde{p}(\mathbf{x}, \mathbf{z}_j)) + const. \tag{2.52}$$

The optimal value is obtained when the KL divergence equals zero, which means:

$$q_j^*(\mathbf{z}_j) = \frac{\exp(\mathbb{E}_{\prod_{i\neq j} q_i(\mathbf{z}_i)}\big[\log p_\theta(\mathbf{x}, \mathbf{z})\big])}{Z_j}, \tag{2.53}$$

with

$$Z_j = \int \exp(\mathbb{E}_{\prod_{i \neq j} q_i(\mathbf{z}_i)}\big[\log p_\theta(\mathbf{x}, \mathbf{z})\big])d\mathbf{z}_j. \tag{2.54}$$

We note that the functional form of each $q_j$ is not pre-defined, but is obtained by computing the exponential of the expectation of the joint log-distribution over all other latent variables. Further, we remark that the solution for the factor $q_j$ indeed depends on the other factors $q_i$. Therefore, the global solution is obtained by iteratively estimating each of the $q_j$ [24]. We first have to initialize all the factors to an appropriate form and then update them with Equation 2.53, each in turn, cyclically, and in an arbitrary order. This provides the basis of the E-steps of the variational EM algorithm, summarized below.

The **variational expectation-maximization (VEM)** algorithm combining the general VI principle and the mean-field factorization can be summarized as iterating between the two following steps (with initialization with $\theta = \theta^{\text{old}}$).

- Variational E-Steps: For $j \in \{1, ..., J\}$, compute $q_j(\mathbf{z}_j)$ with Equation 2.53.
- M-Step: Estimate $\theta^{\text{new}} = \arg\max_{\theta}(\mathcal{L}(\theta, q; \mathbf{x}))$, with all factors $q_j(\mathbf{z}_j)$ being fixed as the output of the E-steps, and then set $\theta = \theta^{\text{new}}$ in Equation 2.53.

### 2.3.5 FIXED-FORM VI AND GRADIENT-BASED OPTIMIZATION

An alternative approximation method involves selecting a suitable functional form for $q(\mathbf{z})$ and optimizing the ELBO using gradient-based techniques. Typically, $q(\mathbf{z})$ is chosen from a parametric distribution family, denoted as $q_\phi(\mathbf{z})$ [141]. The commonly used distribution forms for continuous latent variables are the exponential families.

Gradient-based optimization methods require the computation of gradients w.r.t. both the generative parameters $\theta$ and the inference parameters $\phi$. The gradients w.r.t. $\theta$ are easy to compute since we can directly move the gradient operation inside the expectations:

$$\begin{aligned}\nabla_\theta \mathcal{L}(\theta, \phi; \mathbf{x}) &= \nabla_\theta \mathbb{E}_{q_\phi(\mathbf{z})}\big[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z})\big], \\ &= \mathbb{E}_{q_\phi(\mathbf{z})}\big[\nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z})\big]. \end{aligned} \tag{2.55}$$

Nevertheless, the expectations w.r.t. $q_\phi(\mathbf{z})$ are usually analytically intractable. It can be approximated with one-sample Monte Carlo estimation:

$$\nabla_\theta \mathcal{L}(\theta, \phi; \mathbf{x}) \approx \nabla_\theta \log p_\theta(\mathbf{x}, \mathbf{z}^i), \tag{2.56}$$

with $\mathbf{z}^i \sim q_\phi(\mathbf{z})$. Conversely, the gradients w.r.t. $\phi$ are more complicated to compute as the parameters $\phi$ are involved within the expectations. To address this problem, various methods have been developed. One approach is to employ the log-derivative trick and derive the REINFORCE gradient estimator [56]. This gives rise to the **blackbox variational inference (BBVI)** method [160]. Limited by space, we will not delve into further details about this method. Another kind of solution is the **reparametrization trick** [107], which proposes to rewrite the latent variable $\mathbf{z}$ as some differentiable and invertible transformations $g$ of another random variable $\boldsymbol{\epsilon}$ that follows a known distribution $p(\boldsymbol{\epsilon})$, which means that $\mathbf{z} = g(\boldsymbol{\epsilon}, \phi)$. By simply applying the change of variable formula, we can write the gradient of the ELBO w.r.t. $\phi$ as follows:

$$\begin{aligned} \nabla_\phi \mathcal{L}(\theta, \phi; \mathbf{x}) &= \nabla_\phi \mathbb{E}_{q_\phi(\mathbf{z})}\big[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z})\big], \\ &= \nabla_\phi \mathbb{E}_{p(\boldsymbol{\epsilon})}\big[\log p_\theta(\mathbf{x}, g(\boldsymbol{\epsilon}, \phi)) - \log q_\phi(g(\boldsymbol{\epsilon}, \phi))\big], \\ &= \mathbb{E}_{p(\boldsymbol{\epsilon})}\Big[\nabla_\phi\big(\log p_\theta(\mathbf{x}, g(\boldsymbol{\epsilon}, \phi)) - \log q_\phi(g(\boldsymbol{\epsilon}, \phi))\big)\Big]. \end{aligned} \tag{2.57}$$

Again, the expectation w.r.t. $p(\boldsymbol{\epsilon})$ can be approximated with the one-sample Monte Carlo estimation:

$$\nabla_\phi \mathcal{L}(\theta, \phi; \mathbf{x}) \approx \nabla_\phi\big(\log p_\theta(\mathbf{x}, g(\boldsymbol{\epsilon}^i, \phi)) - \log q_\phi(g(\boldsymbol{\epsilon}^i, \phi))\big), \tag{2.58}$$

with $\boldsymbol{\epsilon}^i \sim p(\boldsymbol{\epsilon})$.

Another challenge for gradient-based optimization techniques is that for local latent variables, we need to estimate the inference parameters $\phi_n$ of the approximate posterior distribution $q_{\phi_n}(\mathbf{z}_n|\mathbf{x}_n)$ for each data sample $\mathbf{x}_n$. This process is very time-consuming and makes it hard to scale up when dealing with large datasets. An alternative approach is to use neural networks for parameterization, which take the data sample $\mathbf{x}_n$ as input and predict the parameters $\phi_n$ as $\phi_n = f_\phi(\mathbf{x}_n)$, with $\phi$ the parameters of the neural networks.

This technique is known as **amortized variational inference** [67]. The key idea behind amortized variational inference is to "amortize" the computational cost of inference across the entire dataset. Instead of performing costly optimization for each data point individually, the inference network learns to generalize across the dataset, providing fast and approximate posterior inference for new data points.

By integrating the reparametrization trick and the amortized variational inference techniques, VAEs leverage deep neural networks to parameterize both the generative and inference models. The optimization of the ELBO is performed concurrently with respect to both $\theta$ and $\phi$ using stochastic gradient descent (SGD) techniques [173]. This enables the effective training of VAEs with large datasets.

## 2.4 DEEP ARCHITECTURES FOR SEQUENTIAL DATA MODELING

The advent of **deep neural network (DNN)** [73, 234] in the last decade has pushed traditional statistical machine learning one step forward, opening up the possibilities to scale up statistical methods for handling very large datasets. By stacking multiple non-linear layers together, DNNs offer a prominent means to improve model expressivity, enabling the construction of models depicting intricate (non-linear) dependency structures within the data [73]. Additionally, the **stochastic gradient descent (SGD)** technique provides an effective approach to optimizing the learning objective and estimating model parameters. This is very important because it allows us to attain a good enough approximation of the model parameters even when no analytical solution exists. In particular, the success of the VAE demonstrates the huge potential of integrating DNNs with probabilistic generative models. As discussed in Section 2.1, we refer to these models as deep probabilistic generative model (DPGM). In this manuscript, we primarily focus on applications of DPGMs to sequential data. Hence, this section gives a concise presentation of deep architectures designed for modeling sequential data.

Before diving into the details of different model architectures, we first make a distinction between different types of sequential data modeling tasks that are commonly encountered in machine learning. Specifically, we can identify three scenarios, see [140], Section

15.2: Seq2Vec, Vec2Seq and Seq2Seq. Seq2Vec means that the input of the model is a sequence, while the output is a vector. This kind of model is usually used for sequence classification, such as sentence sentiment analysis. Vec2Seq indicates that the input of the model is a vector, while the output is a sequence. This kind of model is generally used for sequence generation, such as image captioning. Finally, Seq2Seq refers to the models with both sequential input and output. The Seq2Seq models are often implemented with an encoder-decoder architecture. In this section, we focus our discussion on Seq2Seq models.

### 2.4.1 RECURRENT NEURAL NETWORKS

The **recurrent neural networks (RNNs)** [122] stand as widely adopted deep learning models when dealing with sequential data. These models specialize in capturing sequential patterns and preserving historical context within a sequence by utilizing a hidden state mechanism. The hidden states are computed in a recurrent way so that at each time step, they can absorb new information from the input at the same time step while conserving a trace of all the past information. More precisely, let $\mathbf{x}_{1:T} \in \mathbb{R}^{T \times d}$ denote the input sequence, and let $\mathbf{h}_{t-1} \in \mathbb{R}^h$ denote the hidden state at time $t-1$. The hidden state at time t is computed as follows:

$$\mathbf{h}_t = \alpha(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h), \tag{2.59}$$

with $\mathbf{W}_{xh} \in \mathbb{R}^{d \times h}$ and $\mathbf{W}_{hh} \in \mathbb{R}^{h \times h}$ the weights of the model, $\mathbf{b}_h \in \mathbb{R}^h$ the bias of the model, and $\alpha$ a non-linear activation function.

A Seq2Seq model implemented with RNNs was proposed in [192] to generate sequences. It comprises an RNN encoder that encodes the input sequence into a context vector $\mathbf{c}$, typically derived from the hidden state vector, computed using Equation 2.59, at the last time step of the sequence, denoted as $\mathbf{c} = \mathbf{h}_T$. This vector encapsulates information from the entire sequence. Then, the decoder utilizes this context vector to generate the output sequence with another RNN. Specifically, the output vector of the decoder at time $t$ is generated as

$$\mathbf{y}_t = \mathbf{W}_{hy}\mathbf{h}_t^{dec} + \mathbf{b}_y, \tag{2.60}$$

with $\mathbf{W}_{hy}$ and $\mathbf{b}_y$ the weights and bias of the model and $\mathbf{h}_t^{dec}$ the hidden state at time $t$ of the decoder RNN. $\mathbf{h}_t^{dec}$ is computed as

$$\mathbf{h}_t^{dec} = \alpha'(\mathbf{W}_{ch}[\mathbf{c}, \mathbf{y}_{t-1}] + \mathbf{W}_{hh}^{dec}\mathbf{h}_{t-1}^{dec} + \mathbf{b}_h^{dec}), \tag{2.61}$$

with $\mathbf{W}_{ch}$, $\mathbf{W}_{hh}^{dec}$ and $\mathbf{b}_h^{dec}$ the weights and bias of the model and $[\mathbf{c}, \mathbf{y}_{t-1}]$ the concatenation of the two vectors.

We can make several remarks about this kind of Seq2Seq model. First, as the hidden state is computed by recurrently multiplying the hidden state at the previous time step with the model weights, the RNNs are known to suffer from the gradient exploding and vanishing problems. In fact, when training very deep models with many stacked layers, the gradients tend to become either very large (corresponding to the gradient exploding issue) or very small (corresponding to the gradient vanishing issue). These tendencies can significantly impede the effective training of RNNs [108]. To remedy this issue, more sophisticated RNN architectures such as the **long short-term memory (LSTM)** [89] and **gated recurrent units (GRU)** [37] have been proposed and were prominent tools for sequential data modeling tasks for years. Second, by inspecting Equation 2.60 and Equation 2.61, we find them similar to Equation 2.32 and Equation 2.31 of the LDS. Nevertheless, we need to point out three main differences between RNNs and LDS. The first is that the RNN is a deterministic model, while the LDS is a probabilistic model with stochasticity modeling of noise. The second is that RNN enables non-linear dependencies within the hidden states, while the original LDS only permits linear dependencies. And the third is that Equation 2.60 and Equation 2.61 show that each generated vector $\mathbf{y}_t$ not only depends on the previously generated vector $\mathbf{y}_{t-1}$, but on all past vectors via $\mathbf{h}_{t-1}$. This helps RNNs to overcome the limitations of standard Markov assumptions in LDS, and enables them to have unlimited memory in theory. We observe that $\mathbf{y}_t$ is generated in an auto-regressive way by using the past values $\mathbf{y}_{1:t-1}$. In practice, the model is trained by passing the ground truth values. And this training strategy is called *teacher* *forcing* [219]. However, at test time, we no longer have the ground truth past values, and previously generated past values are used instead. This can lead to a significant decrease in the performance of the model (this is a particular case of train/test mismatch). A common solution to this problem is to resort to *scheduled sampling*, which consists in gradually replacing the ground truth past values with the generated ones during training [14]. We will make some investigations on this problem in Chapter 7. Finally, we can further enhance the performance of the RNNs by applying the bidirectional RNNs, which encompass information from both the past and future [181].

### 2.4.2 1D CONVOLUTIONAL NEURAL NETWORKS

The **convolutional neural network (CNN)** is a very popular type of DNNs that were initially designed for modeling 2D inputs such as images [115]. However, they can also be adapted to 1D inputs for sequential data modeling. Typically, the input sequence $\mathbf{x}_{1:T} \in \mathbb{R}^{T\times d}$ is convolved with several 1D kernels. More precisely, we convolve each dimension of the successive input vectors with a different kernel separately and add up the results to obtain one dimension of the output vector: $y_{t,j} = \alpha(\sum_{i=1}^{d} \mathbf{x}_{t:t+k,i}^{T} \mathbf{w}_{i,j})$. $\mathbf{w}_{i,j}$ is the kernel for input dimension $i$ and output dimension $j$, $t$ is the temporal index, $k$ is the kernel size and $\alpha$ is a non-linear activation function. By applying $J$ times convolution over the entire sequence, we obtain a sequence of output vectors $\mathbf{y}_{1:T'} \in \mathbb{R}^{T'\times J}$. We should note that the output sequence length $T'$ does not necessarily equal the input sequence length $T$. In contrast, we usually have $T' < T$. Indeed, in convolutional computations, several techniques exist to handle output sequence length. One effective method is known as **strided convolution**, which involves skipping every $s$ frames of input when computing the convolutions, where $s$ represents the stride size [49]. If we choose a stride size greater than 1 during the convolution, we can get a much shorter output sequence than the input sequence. Another technique is to pad the input data at the beginning and the end of the sequence with pre-defined values. If we would like to obtain an output sequence with the same length as the input, we need to appropriately choose the padding size. An alternative way to make the output sequence length equal to the input sequence is to design an encoder-decoder architecture. The encoder will embed the long input sequence into a much shorter sequence, or even into a vector, with convolutional layers, while the decoder will decode the obtained sequence to its original length with deconvolutional layers [49].

When leveraging 1D CNNs for sequential data analysis or generation, causal CNNs are frequently employed to enable real-time processing. This involves applying a mask during convolution, ensuring that the output vector accesses only past input values. Additionally, in order to capture long-term temporal dependencies effectively, dilated convolutions can be applied to increase the receptive field of the convolutional layers [230]. The dilated convolution method computes the convolutions by taking every $r$'s input frame, with $r$ referred to as the dilation factor. These techniques are successfully employed in the Wavenet model [203] for text-to-speech (TTS) synthesis. Furthermore, the authors of [7] conducted experiments to compare the performance of convolutional networks and recurrent networks across various sequence modeling tasks. Their findings led to the conclusion that simple convolutional architectures outperform recurrent networks, such as LSTMs, across a diverse range of tasks. Today, 1D CNNs are widely used across various machine learning domains, particularly in tasks related to raw (time-domain) audio signals modeling [131, 42].

### 2.4.3 ATTENTION MECHANISM AND TRANSFORMERS

In Section 2.4.1, we introduced the Seq2Seq model with RNNs. In this model, the information of the entire input sequence is embedded into a single vector $\mathbf{h}_T$ and fed into the decoder. When the decoder generates the output sequence, each output vector $\mathbf{y}_t$ has access to the same information coming from the input sequence. However, output vectors at different time steps may have different temporal dependencies on the information from the input sequence. Thus, it is desirable to design an architecture so that each $\mathbf{y}_t$ has dynamic access to information from the input vectors at different time steps. This is the seminal idea of the **attention mechanism** proposed in [6]. Concretely, the attention mechanism is based on three primary components: queries $\mathbf{Q} \in \mathbb{R}^{T \times d}$, keys $\mathbf{K} \in \mathbb{R}^{T \times d}$ and values $\mathbf{V} \in \mathbb{R}^{T \times v}$. For instance, in the Seq2Seq approach, the queries can be the right-shifted hidden state vectors of the decoder RNN: $\mathbf{Q} = \mathbf{h}^d_{0:T-1}$, while the keys and values can be the hidden state vectors of the encoder RNN: $\mathbf{K} = \mathbf{V} = \mathbf{h}^e_{1:T}$. To obtain the output of the attention module at a certain time step $t$, we first need to compute the similarity between the query vector at time $t$, $\mathbf{Q}_t$, and each of the key vectors $\mathbf{K}_{t'}$ with a certain scoring function: $a_{tt'} = a(\mathbf{Q}_t, \mathbf{K}_{t'})$. Note that the similarity scores have the following properties: $0 < a_{tt'} < 1$, and $\sum_{t'=1}^{T} a_{tt'} = 1$. These properties are typically guaranteed by applying the softmax function. Then, the output vector $\mathbf{y}_t$ is computed as a weighted sum of the values: $\mathbf{y}_t = \sum_{t'=1}^{T} a(\mathbf{Q}_t, \mathbf{K}_{t'})\mathbf{V}_{t'}$. This architecture enables the output vectors to have more flexible dependencies on the input vectors. The similarity score function $a(\mathbf{Q}_t, \mathbf{K}_{t'})$ is usually defined in terms of a kernel function, such as the Gaussian kernel function.[8] In practice, the **scaled dot-product attention** is commonly employed [207]:[9]

$$a(\mathbf{Q}_t, \mathbf{K}_{t'}) = \frac{\mathbf{Q}_t \mathbf{K}_{t'}^T}{\sqrt{d}}. \tag{2.62}$$

The output of the scaled dot-product attention is therefore:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\Big(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\Big)\mathbf{V}. \tag{2.63}$$

Incorporating the attention mechanism to link the encoder and decoder within the RNN-based Seq2Seq model significantly enhanced its performance, establishing it as the state-of-the-art model for machine translation tasks during that period [6, 132].

As the attention mechanism makes itself a sequence-to-sequence mapping architecture, we can wonder if it is possible to replace all the RNN modules in a Seq2Seq model with the attention mechanism and thus construct a Seq2Seq model solely based on attention. This basic idea led to the creation of the powerful **Transformers** [207]. Similar to the RNN-based Seq2Seq model, the Transformer is also composed of an encoder and a decoder. (See Figure 2.1 for an overview of the Transformer architecture.) The encoder transforms an input sequence $\mathbf{x}_{1:T}$ into a representation sequence $\mathbf{z}_{1:T}$ using several

---

[8] The Gaussian kernel function is defined as $k(\mathbf{x}, \mathbf{x}') = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(\mathbf{x}-\mathbf{x}')^T(\mathbf{x}-\mathbf{x}')}{2\sigma^2}}$.

[9] Note that the division by $\sqrt{d}$ is to ensure that the variance of the dot product remains unchanged regardless of the size of the inputs.

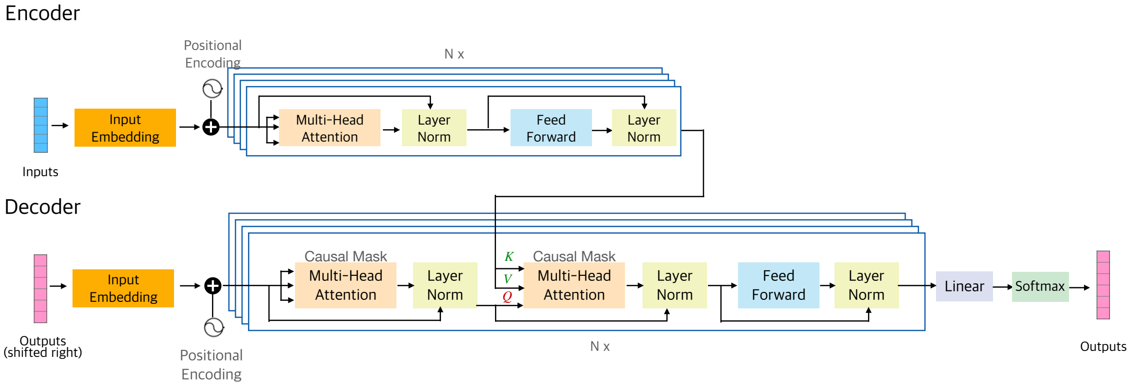


**Figure 2.1**: The model architecture of Transformer. This figure is adapted from Figure 1 of paper [207].

stacked encoder layers. Each of the encoder layers is composed of two sub-layers: a self-attention layer and a feed-forward layer. The queries, keys, and values of the self-attention layer are typically different linear projections of the input sequence. This design aims at extracting the inner temporal correlations within the sequence. The fully connected feed-forward layer is applied at each time step separately and identically, indicating that vectors at different time steps share the same weights. The feed-forward layer can also be understood as convolutions with kernel size 1, and it is designed to aggregate information across the different dimensions for each temporal position in parallel. Residual connections and layer normalization are applied to both of these two sub-layers to avoid gradient vanishing. The decoder generates the output sequence $\mathbf{y}_{1:T}$ in an auto-regressive manner with stacked decoder layers. Each of the decoder layers contains three sub-layers: a self-attention layer, a cross-attention layer and a feed-forward layer. The input of the decoder is the right-shifted (previously) generated sequence $\mathbf{y}_{0:T-1}$. To ensure that the output vector at each time $t$ only has access to the generated vectors up to time $t-1$, a causal mask is applied to the self-attention layer. Between the self-attention layer and the feed-forward layer, there is a cross-attention layer to connect the encoder and the decoder. The queries of the cross-attention layer are the outputs of the self-attention layer of the decoder, while the keys and values are the outputs of the encoder, i.e. the representation sequence $\mathbf{z}_{1:T}$. Similar to the encoder, residual connection and layer normalization are applied to each of the sub-layers of the decoder.

All of the attention modules employed in the Transformer are the scaled dot-product attention that we mentioned above. Furthermore, in order to capture different kinds of temporal dependencies, **multi-head attention (MHA)** is applied. Specifically, in the MHA module, the original queries, keys and values are linearly projected $h$ times to different sets of lower-dimensional vectors, using $h$ different sets of learned linear projections, called ‘heads’. The attention function is applied to each of these projected versions of the (queries, keys, values) sets, resulting in $h$ heads functioning in parallel. Finally, the output of the $h$ heads are concatenated together to give the final output of the MHA module. Let $\mathbf{Q} \in \mathbb{R}^{T\times d}$, $\mathbf{K} \in \mathbb{R}^{T\times d}$ and $\mathbf{V} \in \mathbb{R}^{T\times d}$ [10] denote the queries, keys, and values, respectively. The MHA module can be expressed as follows:

$$\text{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{Concat}(\text{head}_1, \text{head}_2, ..., \text{head}_h)\mathbf{W}^O, \tag{2.64}$$

$$\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V), \tag{2.65}$$

with $\mathbf{W}_i^Q \in \mathbb{R}^{d\times d_k}$, $\mathbf{W}_i^K \in \mathbb{R}^{d\times d_k}$ and $\mathbf{W}_i^V \in \mathbb{R}^{d\times d_v}$ the projection weights for queries, keys and values, respectively for head$_i$, and $\mathbf{W}^O \in \mathbb{R}^{hd_v\times d}$ the projection weights for the output.

One problem with the attention mechanism is that it inherently disregards the order of vectors in a sequence, although this order may be of great importance in dynamical data sequences. Therefore it becomes necessary to incorporate into the model some information on the absolute or relative position of the successive data vectors. In the Transformer architecture, positional information is encoded into a sequence of vectors with the same dimensions as the input sequence. These positional encodings are then added to the input sequence before entering the self-attention layer of the encoder. One way of encoding the

---

[10] To keep consistency with the original paper [207], we suppose that the queries, keys and values have the same dimension $d$, which is also called the model dimension.

positional information is to use the cosinus and sinus function of different frequencies:

$$p_{i,2j} = sin(\frac{i}{10000^{\frac{2j}{d}}}), \tag{2.66}$$

$$p_{i,2j+1} = cos(\frac{i}{10000^{\frac{2j}{d}}}), \tag{2.67}$$

where $i$ is the index of sequential position and $j$ is the index of dimension in the vector [207]. This positional encoding allows the model to learn to attend to relative positions.

The Transformer architecture has achieved remarkable success in modeling sequential data. One of the most important contributions of the Transformers is that it overcome the sequential computation constraints of RNN-based models. [11] It makes every position in the input sequence directly accessible to every position in the output sequence. As a result, the model no longer relies on remembering the entire sequence. These characteristics enable Transformers to easily scale up for large datasets with large model sizes. This architecture has demonstrated compelling success across various domains, such as natural language processing [30, 45], computer vision [48], speech processing [80], and reinforcement learning [34]. The scalability of Transformers has also revolutionized generative models for sequential data. Particularly, the **Generative Pre-trained Transformer (GPT)** models [30, 158, 146] and the ChatGPT product of OpenAI have showcased very impressive language generation performance. On the other hand, a new trend in the deep learning development pathway is to pre-train large-scale models using vast amounts of data via self-supervised training objectives. These resulting models are sometimes called **foundation models** [28]. The development of Transformers also oscillates with this trend.

## 2.5 EXAMPLES OF APPLICATIONS TO AUDIO, IMAGE, AND VIDEO PROCESSING

To conclude this chapter, let us discuss a few examples of practical applications of the above-mentioned techniques for audio, image, and video processing tasks. The goal is

---

[11] The sequential computation constraints of RNN-based models make them poorly suitable for parallelism, consequently slowing down the training process.

both to illustrate the interest and power of the presented models on practical tasks and to rapidly introduce the tasks we have considered in our own PhD work (knowing that more technical details will be given in the corresponding chapters).

In a general manner, probabilistic generative models have found extensive applications across various domains, even before the booming of deep learning approaches. For example, in the field of natural language processing, **latent Dirichlet allocation (LDA)**, a three-level hierarchical Bayesian model designed for discrete data, has become a very classical approach for topic modeling [27]. In the pre-deep-learning era of speech processing, the HMM-GMM model has been the state-of-the-art in automatic speech recognition for about a decade [228]. In the field of computer vision, probabilistic graphical models have been widely used for a branch of classical tasks such as image labeling, segmentation, and denoising, object detection, recognition, and tracking [98]. Over the past decade, the rapid development of deep learning methods has revolutionized these fields. Training sophisticatedly designed DNN architectures has resulted in remarkable performance, often largely surpassing previous achievements. Designing strategies for integrating deep learning approaches and statistical machine learning models to ensure a balance between interpretability and high performance has emerged as a critical challenge. In this section and in our developments described in the next chapters, our focus will be on three specific tasks that illustrate this challenge: multi-target tracking, single-channel audio source separation, and speech enhancement in noise. In this section, we will briefly define each task and review several remarkable techniques developed in these fields.

### 2.5.1 MULTI-TARGET TRACKING

**Multi-target tracking (MTT)**, in its general sense, refers to the problem of detecting, identifying, and recovering the trajectories of multiple moving objects given a series of noisy measurements from the sensor(s) [212]. Originating from aerospace applications in the 1960's, MTT has a long history spanning over 60 years. It finds wide applications across diverse fields, encompassing video surveillance, defence, air traffic control, biomedical research, autonomous driving, and human-robot interaction. MTT is a complex task that requires a blend of mathematical modeling and engineering techniques. Indeed, MTT can be segmented into several sub-tasks: (i) extracting source observations (also called detections) from the raw sensor measurements at every time frame; (ii) associating observations to sources consistently over time; (iii) modeling the dynamics of the sources' movements and filtering the obtained object trajectories; and (iv) accounting for the birth and death process of source trajectories (i.e. the fact that sources may appear and disappear from the observed scene along time). In the field of computer vision, cameras are the most commonly used sensors. The main object of MTT consists here of estimating the positions of all objects of interest in the video, assigning a unique identity to each object, and keeping the same identity for all the frames in which that object appears. In such a case, MTT is also referred to as **multi-object tracking (MOT)** [128, 38], and we will use this term hereinafter.

In MOT, sub-task (i) (referred to as object detection) is often processed separately from (and previously to) the other sub-tasks. Tracking approaches under this paradigm are referred to as tracking-by-detection methods [4, 221, 38, 128]. In the tracking-by-detection paradigm, a detector is initially employed to extract the positions of objects of interest in each frame, commonly represented as detection bounding boxes (DBBs). Recent advancements in powerful detection algorithms have notably improved performance [166, 165, 83]. Utilizing these DBBs, diverse methodologies have been developed to link and construct the corresponding trajectories. This process is sometimes called data association. Before the widespread application of deep learning, statistical methods leveraging Bayesian filtering, such as the joint probabilistic data association filter (JPDAF) [10], the multiple hypothesis tracking (MHT) [26], and multitarget filters based on random finite sets (RFSs) [134], had achieved great success. The booming development of deep learning in the field of computer vision facilitates the integration of appearance information into tracking algorithms, leading to new research directions [38]. Furthermore, recent studies within the computer vision community suggest that the two-stage tracking-by-detection algorithms may exhibit inferior performance because the accuracy of the tracking algorithm heavily relies on the quality of the detections [16]. New trends in the community tend to combine the detection and tracking models together so as to jointly improve the detection and tracking performance [16, 237, 235]. However, tracking methods solely based on appearance information may have several limitations. For instance, in scenarios involving moving cameras, long-term occlusions, non-linear motion dynamics, poor illumination conditions, and similar object appearances, these approaches might encounter problems such as false negatives and identity switchs. These issues can hinder robust long-term tracking performance.

On the other hand, the dynamics of the objects' motion encapsulate rich information for tracking. Properly integrating motion information can help enhance overall tracking performance, as effectively demonstrated in recent studies [5, 16, 179]. A widely adopted method to integrate motion information into tracking algorithm involves refining tracking trajectories with the Kalman filter [19, 221, 16], which is based on a linear dynamic motion model (see Section 2.2.2). While a linear dynamical model is suitable in cases of high sampling rates and reasonable object velocity, challenging tracking scenarios with low sampling rates, moving camera, and high object velocity are still difficult to tackle with such simple models. To address these issues, more sophisticated nonlinear dynamical models using RNNs have been proposed [177, 5, 138, 222, 214, 225, 121, 46]. Nevertheless, the tracking ability of deterministic dynamical models is still limited when dealing with complex and challenging object motions. An alternative approach involves the application of DPGMs. Rather than modeling trajectories deterministically, deep probabilistic generative motion models describe the trajectory distributions, offering enhanced accuracy and flexibility. Additionally, they can also be used to generate reasonable bounding boxes when the detections are missing, which will greatly increase the tracking performance in scenarios of long-term occlusions. A new proposed MOT method based on DPGMs, in the present case DVAEs, will be presented in detail in Chapter 3.

### 2.5.2 SINGLE-CHANNEL AUDIO SOURCE SEPARATION

**Audio source separation (ASS)** is a fundamental task in speech/audio signal processing [209]. It focuses on isolating each source of interest from a mixture of audio signals, which typically include speech, music, and noise. Dating back to the 1950s, when it was initially framed as the "cocktail party problem"[12] [36], this challenging problem has been studied for over 70 years. According to the number of recording microphones, ASS methods can be categorized into single-channel and multi-channel methods. Moreover, the multi-channel setting can also be of various types and complexity [190]. In this manuscript, we focus on the **single-channel audio source separation (SC-ASS)** methods.

Audio signals are typically high-dimensional. For instance, a one-second audio clip, recorded at 16kHz, produces a signal represented as a vector of dimension 16,000. Handling raw (time-domain) waveform signals directly poses significant challenges in audio processing tasks. To circumvent this problem, and also to exploit the inherent structure of audio signals, a common approach consists of converting the waveform signals into a time-frequency representation using methods like the short-time Fourier transform (STFT). In the pre-era of deep learning, numerous statistical approaches emerged for separating audio sources in the time-frequency domain. One of the most commonly employed methods is time-frequency masking and filtering. The fundamental concept involves applying a filter (mask) to the STFT of the mixed signal to obtain an estimation of the individual source signal. The time-frequency mask estimation relies on signal statistics, leading to different types of estimators. For instance, the optimal solution for a linear minimum mean-square-error (LMMSE) estimator is the famous Wiener filter [209]. Another commonly used mask is the ideal binary mask (IBM), which allocates each time-frequency bin of the STFT of the mixed signal to a specific source. The STFT is based on the W-disjoint assumption, asserting that in each time-frequency bin, only one audio source is dominant, meaning that it has significantly higher energy compared to the other sources [227]. A prominent method that extends the principles of the IBM is computational auditory scene analysis (CASA) [216]. CASA leverages the perceptual principles of auditory scene analysis to segregate mixed signals [29]. It utilizes auditory

---

[12] This term originates from the scenario where, in a noisy cocktail party setting, an individual tries to focus his attention on a single speaker despite the various interfering sounds around.

perceptual cues to group the time-frequency bins of the mixture signal both simultaneously and sequentially, enabling the extraction of individual sound sources from complex audio mixtures. LVMs can also be applied to solve the ASS problem. For example, the authors of [148] proposed to use factorized hidden Markov models (factorized HMMs) [70]. The factorized HMMs is extended from an HMM with the purpose of disentangling hidden states of different dynamics. It is therefore well-suited to tackle the ASS problem. Another notable statistical approach that has been very popular for SC-ASS is based on non-negative matrix factorization (NMF) [52]. In this approach, the NMF model is used to decompose the power spectrogram of audio signals into low-dimensional spectral power patterns modulated along time by a coefficient called temporal activation. To apply the NMF model for ASS, the process begins by fitting the NMF model individually on a set of single-source datasets to derive a dictionary of spectral patterns for each source. Subsequently, these patterns are gathered in a dictionary matrix that is kept fixed. Then the NMF model is fitted to the mixture signal to separate. Identifying the spectral patterns used in this decomposition and their corresponding temporal activations enables the extraction of the separated signal. Many variants of this general principle have been applied and have demonstrated success in SC-ASS [149, 112, 113], as well as in multi-channel ASS [147].

Much like in other domains, the evolution of deep learning has significantly transformed research in ASS. Approaches based on deep learning formulate ASS as a supervised learning problem, most of which aim at learning the separation masks by training the model on a large dataset (of aligned individual source signals and mixtures of them). Various kinds of DNN architectures, input acoustic features, and training targets have been explored [217, 84, 229]. A remarkable progress made by [130, 131] proposes to train an end-to-end deep learning framework directly in the time domain. This approach overcame the subopitmality of approaches based on time-frequency representations by inherently leveraging the phase information, and demonstrated to surpass several ideal time-frequency magnitude masks such as the oracle IBM and oracle Wiener filter mask. Based on the signal embedding architecture of [131], many well-designed and more powerful models employing RNNs and Transformers have been proposed [199, 129, 191].

Supervised ASS methods have made substantial advancements and achieved remarkable success. However, a key limitation of supervised approaches lies in their heavy reliance on a substantial amount of paired and aligned mixture and individual source data for training. In practice, it is difficult to obtain such pairs in a real acoustic environment with noise and reverberations. Typically, supervised methods are designed from synthetic mixtures by simply adding up the isolated clean source signals (and possibly the noise). This can limit the model's performance in scenarios involving noise and reverberations. In contrast, unsupervised or semi-supervised approaches might offer better generalization capabilities. However, beyond traditional unsupervised statistical signal processing approaches, exploration of deep learning-based unsupervised methods has been limited. A remarkable example is the work presented in [220], which introduces a mixture invariant training framework (MixIT). MixIT takes a mixture of mixtures as input and separates it into a variable number of latent sources. These estimated separation sources are then remixed to approximate the original mixtures. This architecture can be trained in either semi-supervised or unsupervised modes. In this thesis, we explore a weakly supervised SC-ASS method, which involves pretraining DVAE models on each single source data and solve the source separation problem within a complete DPGM framework. This work will be presented in Chapter 5.

### 2.5.3 SPEECH ENHANCEMENT

Speech enhancement, although closely related to the ASS task discussed in the previous subsection, concentrates on recovering the clean speech signal from a noisy audio recording [13, 125]. In ASS, the interferences are competing audio sources, that can be themselves of interest, whereas in speech enhancement, the interference is generally considered as a 'global' background noise (hubbub), that we simply want to remove. The deep learning based methods outlined earlier for SC-ASS can be more or less easily adapted to address the speech enhancement problem by treating clean speech and noise as two audio sources. And similarly to the supervised SC-ASS methods, supervised approaches to speech enhacement are powerful and can obtain impressive results. However, they also confront challenges in generalizing across diverse noise types and acoustic conditions not encountered during training. Recent research has explored a novel avenue that utilizes deep probabilistic models to tackle this challenge. For instance, the works presented in [154, 57, 58] propose to employ GANs to learn a conditional distribution, predicting the clean speech signal given the noisy speech. Another type of method proposes to use diffusion models for speech enhancement [127, 218, 170]. Specifically, a diffusion model is used to transform a clean signal into a noisy one by gradually adding small-step Gaussian noise (see Section 2.1). Subsequently, speech enhancement is accomplished by applying the inverse diffusion process, conditioned on the input noisy speech signal. DPGMs have shown better generalization abilities compared to methods based on direct noisy-to-clean mapping. Nevertheless, these approaches remain supervised and demand a large amount of paired clean-noisy data for training, which, again, is challenging to obtain in real-life scenarios.

To overcome the limitations of supervised methods, approaches using *non-parallel* clean-noisy data have been investigated. Concretely, three types of methods have been developed: those employing only clean speech data [9, 117, 22], those relying solely on noisy speech data [2, 103, 61, 59] and those utilizing non-paired clean-noisy speech data [223, 231]. A representative solution of the first type can be found in [9]. This work proposes to first learn a prior distribution over clean speech by pre-training a DPGM, here a VAE, on a clean speech dataset. Then, at test time, the learned VAE model is combined with a low-rank NMF-based noise model to infer the variances of speech and noise, and finally build a Wiener filter for speech enhancement. Methods of the second type employ diverse model training strategies. In [2] and [103], the denoising model is trained to map two different noisy versions of the same clean speech signal. The motivation behind this approach lies in the assumption that, when the noises introduced to the input and target noisy speech signals are zero-mean and uncorrelated, deep neural networks are able to learn an averaged denoised representation of the signal. The approach presented in [61] involves corrupting noisy speech signals by introducing additive noise. The model is trained to reconstruct the original noisy speech from its corrupted version. During the test, the trained model is expected to recover the clean speech signal from a noisy speech input. Though we cannot find rigorous theoretical arguments to support this training strategy, it has been shown to achieve good performance in practice. The model proposed in [59] is an unsupervised extension of the supervised MetricGAN method [57, 58]. The MetricGAN method proposes to train the discriminator of the GAN model to mimic the behavior of the target evaluation function. In [59], the supervised evaluation metric (e.g., PESQ [172] utilized in the MetricGAN model) is replaced with an unsupervised evaluation metric, DNSMOS [164]. This adjustment eliminates the requirement for ground truth clean speech data for training. Finally, methods of the third type are typically based on the cycle-consistent generative adversarial network (CycleGAN) [240] architecture. These models usually employ two or more generators and discriminators to learn the cycle consistency between clean and noisy speech signals. In this thesis, we investigate a speech enhancement method of the first type, which entails first pre-training a DVAE model on clean speech signal, followed by integrating this pre-trained DVAE model with another DPGM-based noise model for speech enhancement. This work will be presented in Chapter 6.

# CHAPTER 3

# MIXTURE OF DVAEs FOR MULTI-SOURCE TRAJECTORY MODELING AND SEPARATION

---

> *I basically know of two principles for treating complicated systems in simple ways: the first is the principle of modularity and the second is the principle of abstraction.*
>
> *— Michael I. Jordan*

This chapter is based on the following publication:

Xiaoyu Lin, Laurent Girin, and Xavier Alameda-Pineda. "Mixture of dynamical variational autoencoders for multi-source trajectory modeling and separation." In Transactions on Machine Learning Research, 2023.

## 3.1 A DPGM FOR MULTI-SOURCE DATA

All the DPGMs discussed in Section 2.2 have been used to model the distribution of single-source data, the source being either static and having several possible states (in the case of mixture models) or sequential with different types of underlying dynamics (e.g., locally linear). In real-life scenarios, we often encounter situations where a number of sequential source signals appear concurrently in a natural scene for a certain period of time and are observed jointly. Each underlying source can have its own dynamics, and the problem is to obtain an estimation of the content and/or the position along time of each source separately, which includes consistently recovering the identity of each source over time. In short, we want to estimate and separate each source's trajectory from a set of mixed-up observations. For instance, in the context of multi-object tracking (MOT) as discussed in Section 2.5.1, several moving targets are observed concurrently, and our aim is to recover the trajectories of each object. Another scenario is the single-channel audio source separation (SC-ASS) problem, as discussed in Section 2.5.2, where a mixture of audio signals is recorded, and we would like to separate each of the audio sources.

In this chapter, we propose to tackle this problem within a DPGM framework, with a model combining the following two bricks: (a) a DPGM for modeling the dynamics of each source independently; in this work, we propose to use a model from the DVAE family (presented in Section 2.2.2) to model each individual source. The DVAE-generated random vector represents the source vector, i.e. the source content/position that we want to track over time, and the latent random vector represents the underlying (continuous) hidden state/factor that governs the source dynamics; (b) A discrete latent assignment variable which assigns each observation in the set of (mixed-up) observations to a source.

We name the resulting model as Multi-Source Mixture of DVAEs (MixDVAE). In addition to the MixDVAE model, we propose a multi-source trajectory estimation algorithm (i.e., a solution to the MixDVAE model). Importantly, this estimation method does not require a massive multi-source annotated dataset for model training. Instead, we first pre-train the DVAE model on an unlabeled (synthetic or natural) single-source trajectory dataset, to capture the dynamics of an individual source type. Afterwards, the pre-trained DVAE is plugged into the MixDVAE model together with the observation-to-source assignment latent variable to solve the problem for each multi-source test data sequence to process. For each test data sequence, the (approximate) posterior distributions of both the observation-to-source assignment variable and the source vector of each source are derived using the VI methodology (presented in Section 2.3.2) – more specifically, we propose a VEM algorithm [100, 24, 213].

The proposed model and method are versatile in essence. They can be easily adapted and applied to a variety of estimation problems with multiple dynamical sources with different configurations. For example, if all sources are assumed to have similar dynamics, a single DVAE model can be used to model all sources (more specifically, a different instance of the same DVAE model can be used for each source) and only one pre-training is made on a single single-source dataset. If different types of sources are considered, with different dynamics, one can use different instances of the same DVAE model, but pre-trained on different single-source datasets, or one can use (different instances of) different DVAE models, also pre-trained on different single-source datasets. In any case, as stated above, there is no need for a massive dataset containing annotated mixtures of simultaneous sources, as would be the case with a fully-supervised approach. Labeled multi-source datasets must be much larger than single-source datasets, since the mixture process intrinsically multiplies the content diversity, and thus can be very costly and difficult to obtain. Therefore, our method can be considered as data-frugal and weakly supervised compared to a fully-supervised method. One limitation of the proposed model, though, is that the VEM algorithm applied at test time is relatively costly in computation (this point is investigated in the experimental part of our study). Another limitation is the fact that all sources are assumed to behave independently. In other words, the proposed MixDVAE does not explicitly model the possible interactions between the different sources. This is planned for future work.

In this PhD work, in addition to the methodological developments on the MixDVAE model and solution, which will be the core of the present chapter, we illustrate the versatility of MixDVAE by applying it to two notably different tasks, in computer vision and in audio processing, namely the MOT and SC-ASS tasks that have been introduced in Sections 2.5.1 and 2.5.2, respectively, and that we have mentioned again at the beginning of this chapter. The application of MixDVAE to the MOT problem will be presented in detail in Chapter 4 and the application to the SC-ASS problem will be presented in detail in Chapter 5. In each case, we will report and discuss corresponding experimental results.

## 3.2 MIXDVAE MODEL

### 3.2.1 PROBLEM FORMULATION AND NOTATIONS

Let us consider a sequence containing $N$ sources or targets that we observe over time. Let $n \in \{1, ..., N\}$ denote the source index and let $\mathbf{s}_{tn} \in \mathbb{R}^S$ be here the true (unknown) $n$-th *source vector* at time frame $t$. At every time frame $t$, we gather $K_t$ observations, and this number can vary over time. We denote by $\mathbf{o}_{tk} \in \mathbb{R}^O$, $k \in \{1, ..., K_t\}$, the $k$-th observation at frame $t$. The problem tackled in this paper consists in estimating the sequence of hidden source vectors $\mathbf{s}_{1:T,n} = \{\mathbf{s}_{tn}\}_{t=1}^T$, for each source $n$, from the complete set of observations $\mathbf{o}_{1:T,1:K_t} = \{\mathbf{o}_{tk}\}_{t=1,k=1}^{T,K_t}$.

To solve this problem, we define two additional sets of latent variables. First, for each source $n$ at time frame $t$, we define a latent variable $\mathbf{z}_{tn} \in \mathbb{R}^L$ associated with $\mathbf{s}_{tn}$ through a DVAE model. This DVAE model, which might be identical for all sources or not, is used to model the dynamics of each individual source and is plugged into the proposed probabilistic MixDVAE model. Second, for each observation $\mathbf{o}_{tk}$, we define a discrete observation-to-source assignment variable $w_{tk}$ taking its value in $\{1, ..., N\}$. $w_{tk} = n$ means that observation $k$ at time frame $t$ is assigned to/was generated by source $n$. This results in per-source sequences of assigned observations.

**Table 3.1**: Summary of the variable notations.

| Variable notation | Definition |
|---|---|
| $T, t \in \{1, \dots, T\}$ | Sequence length and frame index |
| $N, n \in \{1, \dots, N\}$ | Total number of sources and source index |
| $K_t, k \in \{1, \dots, K_t\}$ | Number of observations at $t$, and obs. index |
| $\mathbf{s}_{tn} \in \mathbb{R}^S$ | True position/content of source $n$ at time $t$ |
| $\mathbf{z}_{tn} \in \mathbb{R}^L$ | Latent variable of source $n$ at time $t$ |
| $\mathbf{o}_{tk} \in \mathbb{R}^O$ | Observation $k$ at time $t$ |
| $w_{tk} \in \{1, \dots, N\}$ | Assignment variable of observation $k$ at time $t$ |
| $\mathbf{s}_{:,n} = \mathbf{s}_{1:T,n}$ | Source vector sequence for source $n$ |
| $\mathbf{s}_{t,:} = \mathbf{s}_{t,1:N}$ | Set of all source vectors at time $t$ |
| $\mathbf{s} = \mathbf{s}_{1:T,1:N}$ | Set of all source vectors |
| $\mathbf{z}_{:,n}$, $\mathbf{z}_{t,:}$, $\mathbf{z}$ | Analogous for the latent variable |
| $\mathbf{o} = \mathbf{o}_{1:T,1:K_t}$ | Set of all observations |
| $\mathbf{w} = \mathbf{w}_{1:T,1:K_t}$ | Set of all assignment variables |

Hereinafter, to simplify the notations, we use ":" as a shortcut subscript for the set of all values of the corresponding index. For example, $\mathbf{s}_{:,n} = \mathbf{s}_{1:T,n}$ is the complete trajectory of source $n$ and $\mathbf{s}_{t,:} = \mathbf{s}_{t,1:N}$ is the set of all source vectors at time frame $t$. All notations are summarized in Table 3.1.

### 3.2.2 GENERAL PRINCIPLE OF THE PROPOSED MODEL AND SOLUTION

The general methodology of MixDVAE is to define a parametric joint distribution of all variables $p_\theta(\mathbf{o}, \mathbf{s}, \mathbf{z}, \mathbf{w})$, then estimate its parameters $\theta$ and (an approximation of) the corresponding posterior distribution $p_\theta(\mathbf{s}, \mathbf{z}, \mathbf{w}|\mathbf{o})$, from which we can deduce an estimate of $\mathbf{s}_{1:T,n}$ for each source $n$. The proposed MixDVAE generative model $p_\theta(\mathbf{o}, \mathbf{s}, \mathbf{z}, \mathbf{w})$ is presented in Section 3.2.3. As briefly stated above, it integrates the DVAE generative model defined by Equations (2.34)–(2.36) for modeling the sources dynamics and does not use any human-annotated data for training. As is usually the case in (D)VAE-based generative models, both the exact posterior distribution $p_\theta(\mathbf{s}, \mathbf{z}, \mathbf{w}|\mathbf{o})$ and the marginalization of the joint distribution $p_\theta(\mathbf{o}, \mathbf{s}, \mathbf{z}, \mathbf{w})$ w.r.t. the latent variables are analytically intractable. Therefore we cannot directly use an exact EM algorithm and we resort to VI. We propose the following strategy, inspired by the structured mean-field method that we summarized

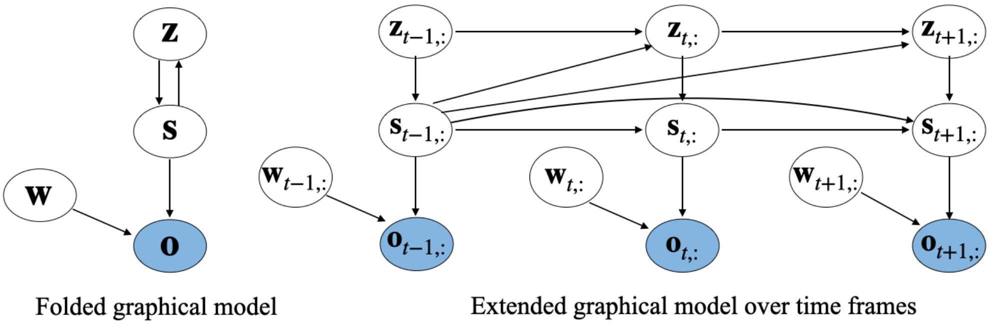


**Figure 3.1**: Graphical representation of the proposed MixDVAE model.

in Section 2.3.4, with $\mathbf{h}$ being here equal to $\{\mathbf{s}, \mathbf{z}, \mathbf{w}\}$. In Section 3.2.4, we define an approximate posterior distribution $q_\phi(\mathbf{s}, \mathbf{z}, \mathbf{w}|\mathbf{o})$ that partially factorizes over $\{\mathbf{s}, \mathbf{z}, \mathbf{w}\}$. Just like the proposed MixDVAE generative model includes the DVAE generative model, the approximate posterior distribution includes the DVAE inference model as one of the factors. This factorization makes possible the derivation of a model solution in the form of a VEM algorithm, as detailed in Section 3.3.

### 3.2.3 GENERATIVE MODEL

Let us now specify the joint distribution of observed and latent variables $p_\theta(\mathbf{o}, \mathbf{w}, \mathbf{s}, \mathbf{z})$. We assume that the observation variable $\mathbf{o}$ only depends on $\mathbf{w}$ and $\mathbf{s}$, while the assignment variable $\mathbf{w}$ is a priori independent of the other variables. The graphical representation of MixDVAE is shown in Fig. 3.1. Applying the chain rule and these conditional dependency assumptions, the joint distribution can be factorised as follows:

$$p_\theta(\mathbf{o}, \mathbf{w}, \mathbf{s}, \mathbf{z}) = p_{\theta_\mathbf{o}}(\mathbf{o}|\mathbf{w}, \mathbf{s}) p_{\theta_\mathbf{w}}(\mathbf{w}) p_{\theta_{\mathbf{sz}}}(\mathbf{s}, \mathbf{z}). \tag{3.1}$$

**Observation model.** We assume that the observations are conditionally independent through time and independent of each other, that is to say, at any time frame $t$, the observation $\mathbf{o}_{tk}$ only depends on its corresponding assignment $w_{tk}$ and source vector at the same time frame. The observation model $p_{\theta_{\mathrm{o}}}(\mathbf{o}|\mathbf{w}, \mathbf{s})$ can thus be factorised as:[1]

$$p_{\theta_{\mathrm{o}}}(\mathbf{o}|\mathbf{w}, \mathbf{s}) = \prod_{t=1}^{T} \prod_{k=1}^{K_t} p_{\theta_{\mathrm{o}}}(\mathbf{o}_{tk}|w_{tk}, \mathbf{s}_{t,:}). \tag{3.2}$$

Given the value of the assignment variable, the distribution $p(\mathbf{o}_{tk}|w_{tk}, \mathbf{s}_{t,:})$ is modeled by a Gaussian distribution:

$$p_{\theta_{\mathrm{o}}}(\mathbf{o}_{tk}|w_{tk} = n, \mathbf{s}_{tn}) = \mathcal{N}(\mathbf{o}_{tk}; \mathbf{s}_{tn}, \mathbf{\Phi}_{tk}). \tag{3.3}$$

This equation models only the observation noise via the covariance $\mathbf{\Phi}_{tk} \in \mathbb{R}^{O \times O}$ and thus assumes that the assigned observation lies close to the true source vector.[2]

**Assignment model.** Similarly, we assume that, a priori, the assignment variables are independent across time and observations:

$$p_{\theta_{\mathrm{w}}}(\mathbf{w}) = \prod_{t=1}^{T} \prod_{k=1}^{K_t} p_{\theta_{\mathrm{w}}}(w_{tk}). \tag{3.4}$$

For each time frame $t$ and each observation $k$, the assignment variable $w_{tk}$ is assumed to follow a uniform prior distribution:

$$p_{\theta_{\mathrm{w}}}(w_{tk}) = \frac{1}{N}. \tag{3.5}$$

**Dynamical model.** Finally, $p_{\theta_{\mathrm{sz}}}(\mathbf{s}, \mathbf{z})$ is modeled with a DVAE. The different sources are assumed to be independent of each other. This implies that in the present work we do not consider possible interactions among sources. More complex dynamical models including source interaction are beyond the scope of this study. With this assumption, the joint distribution of all source vectors and corresponding latent variable $p_{\theta_{\mathrm{sz}}}(\mathbf{s}, \mathbf{z})$ can be

---

[1] In this equation, we use $\mathbf{s}_{t,:}$ and not $\mathbf{s}_{tn}$, since the value of $w_{tk}$ is not specified.

[2] For simplicity of presentation, we state the case in which the observation and source vector dimensions are the same, i.e. $O = S$. In a more general case where $O \neq S$, we can consider the use of a projection matrix $\mathbf{P}_k \in \mathbb{R}^{O \times S}$ and define $p_{\theta_{\mathrm{o}}}(\mathbf{o}_{tk}|w_{tk} = n, \mathbf{s}_{tn}) = \mathcal{N}(\mathbf{o}_{tk}; \mathbf{P}_k\mathbf{s}_{tn}, \mathbf{\Phi}_{tk})$. Again, for simplicity, we consider $\mathbf{P}_k = \mathbf{I}$ in the rest of the paper. All derivations and results are generalizable to $\mathbf{P}_k \neq \mathbf{I}$.

factorized across sources as:

$$p_{\theta_{\mathbf{sz}}}(\mathbf{s}, \mathbf{z}) = \prod_{n=1}^{N} p_{\theta_{\mathbf{sz}}}(\mathbf{s}_{:,n}, \mathbf{z}_{:,n}), \tag{3.6}$$

where $p_{\theta_{\mathbf{sz}}}(\mathbf{s}_{:,n}, \mathbf{z}_{:,n})$ is the DVAE model defined in Equation 2.34–Equation 2.36 and applied to $\mathbf{s}_{:,n}$ and $\mathbf{z}_{:,n}$ (defining $\mathbf{p}_{t,n} = \{\mathbf{s}_{1:t-1,n}, \mathbf{z}_{1:t-1,n}\}$).[3] As mentioned before, the DVAE model can be either the same architecture for all sources, pre-trained on a unique single-source dataset, or the same architecture but pre-trained on different single-source datasets for different sources, or completely different architectures for each source.

Overall, the parameters in the generative model to be estimated are $\theta = \{\theta_{\mathbf{o}} = \{\mathbf{\Phi}_{tk}\}_{t,k=1}^{T,K_t}, \theta_{\mathbf{s}}, \theta_{\mathbf{z}}\}$ (note that $\theta_{\mathbf{w}} = \emptyset$).

### 3.2.4 INFERENCE MODEL

The exact posterior distribution corresponding to the MixDVAE generative model described in Section 3.2.3 is neither analytically nor computationally tractable. Therefore, we propose the following factorized approximation that leads to a computationally tractable inference model:

$$q_{\phi}(\mathbf{s}, \mathbf{z}, \mathbf{w}|\mathbf{o}) = q_{\phi_{\mathbf{w}}}(\mathbf{w}|\mathbf{o}) q_{\phi_{\mathbf{z}}}(\mathbf{z}|\mathbf{s}) q_{\phi_{\mathbf{s}}}(\mathbf{s}|\mathbf{o}), \tag{3.7}$$

where $q_{\phi_{\mathbf{z}}}(\mathbf{z}|\mathbf{s})$ corresponds to the inference model of the DVAE and the optimal distributions $q_{\phi_{\mathbf{s}}}(\mathbf{s}|\mathbf{o})$ and $q_{\phi_{\mathbf{w}}}(\mathbf{w}|\mathbf{o})$ are derived below in the E-steps of the MixDVAE algorithm. The factorization Equation 3.7 is inspired by the structured mean-field method [153] that we summarized in Section 2.3.4, since we break the posterior dependency between $\mathbf{w}$ and $\{\mathbf{s}, \mathbf{z}\}$. However, we keep the dependency between $\mathbf{s}$ and $\mathbf{z}$ at inference time since it is the essence of the DVAE. In addition, we assume that the posterior distribution of the DVAE

---

[3] Here we denote the DVAE parameters by $\theta_{\mathbf{sz}}$ instead of $\theta$, to differentiate the DVAE parameters from the other parameters.

latent variable is independent for each source, so that we have:

$$q_{\phi_{\mathbf{z}}}(\mathbf{z}|\mathbf{s}) = \prod_{n=1}^{N} q_{\phi_{\mathbf{z}}}(\mathbf{z}_{:,n}|\mathbf{s}_{:,n}), \tag{3.8}$$

where $q_{\phi_{\mathbf{z}}}(\mathbf{z}_{:,n}|\mathbf{s}_{:,n})$ is given by Equation 2.37 and Equation 2.38 applied to $\{\mathbf{z}_{:,n}, \mathbf{s}_{:,n}\}$. This is coherent with the generative model, where we assumed that the dynamics of the various sources are independent of each other.

## 3.3 MIXDVAE SOLUTION: A VARIATIONAL EXPECTATION-MAXIMIZATION ALGORITHM

Let us now present the proposed algorithm for jointly deriving the terms of the inference model (other than the DVAE terms) and estimating the parameters of the complete MixDVAE model, based on the maximization of the corresponding ELBO. The inference is done directly on each multi-source test sequence to process and does not require previous supervised training with a labeled multi-source dataset. It only requires to pre-train the DVAE model on synthetic or natural single-source sequences.

As discussed in Section 2.3.4 and Section 2.3.5, in many latent variable models, the optimization of the ELBO is done either following the structured mean-field method Equation 2.53 or using gradient based methods as in (D)VAEs. In our case, we cannot directly use the generic structured mean-field inference procedure, since the proposed approximation Equation 3.7 does not factorize completely in a set of disjoint latent variables (e.g., $q_{\phi_{\mathbf{z}}}(\mathbf{z}|\mathbf{s})$ is conditioned on $\mathbf{s}$). Alternatively, one could resort to purely amortized inference and conceive a deep encoder that approximates the distributions in Equation 3.7, leading to a looser approximation bound. We propose a strategy that is a middle ground between these two worlds. We use the structured mean-field principles that provide a tighter bound since they do not impose a distribution family for $q_{\phi_{\mathrm{w}}}$ and $q_{\phi_{\mathbf{s}}}$, and we use the philosophy of amortized inference for $q_{\phi_{\mathbf{z}}}$ so as to exploit the pre-trained DVAE encoder.

To do so, we have to go back to the fundamentals of VI and iteratively maximize the

MixDVAE model ELBO defined by:

$$\mathcal{L}(\theta, \phi; \mathbf{o}) = \mathbb{E}_{q_\phi(\mathbf{s},\mathbf{z},\mathbf{w}|\mathbf{o})}[\log p_\theta(\mathbf{o}, \mathbf{s}, \mathbf{z}, \mathbf{w}) - \log q_\phi(\mathbf{s}, \mathbf{z}, \mathbf{w}|\mathbf{o})]. \tag{3.9}$$

By injecting Equation 3.1 and Equation 3.7 into Equation 3.9, we can develop $\mathcal{L}(\theta, \phi; \mathbf{o})$ as follows:

$$\begin{aligned}\mathcal{L}(\theta, \phi; \mathbf{o}) = {} & \mathbb{E}_{q_{\phi_\mathbf{w}}(\mathbf{w}|\mathbf{o})q_{\phi_\mathbf{s}}(\mathbf{s}|\mathbf{o})}\big[\log p_{\theta_\mathbf{o}}(\mathbf{o}|\mathbf{w}, \mathbf{s})\big] + \mathbb{E}_{q_{\phi_\mathbf{w}}(\mathbf{w}|\mathbf{o})}\big[\log p_{\theta_\mathbf{w}}(\mathbf{w}) - \log q_{\phi_\mathbf{w}}(\mathbf{w}|\mathbf{o})\big] \\ & + \mathbb{E}_{q_{\phi_\mathbf{s}}(\mathbf{s}|\mathbf{o})}\Big[\mathbb{E}_{q_{\phi_\mathbf{z}}(\mathbf{z}|\mathbf{s})}\big[\log p_{\theta_\mathbf{sz}}(\mathbf{s}, \mathbf{z}) - \log q_{\phi_\mathbf{z}}(\mathbf{z}|\mathbf{s})\big]\Big] - \mathbb{E}_{q_{\phi_\mathbf{s}}(\mathbf{s}|\mathbf{o})}\big[\log q_{\phi_\mathbf{s}}(\mathbf{s}|\mathbf{o})\big].\end{aligned} \tag{3.10}$$

The ELBO maximization is done by alternatively and iteratively maximizing the different terms corresponding to the various posterior and generative distributions. In our case, we obtain a series of variational E and M steps. While the E steps associated to $q_{\phi_\mathbf{w}}$ and $q_{\phi_\mathbf{s}}$ follow the structured mean-field principle, the E step associated to $q_{\phi_\mathbf{z}}$ is based on the principle of amortized inference commonly used in (D)VAEs.

### 3.3.1 E-S Step

We first consider the computation of the optimal posterior distribution $q_{\phi_\mathbf{s}}(\mathbf{s}|\mathbf{o})$. To this aim, we first select the terms in (3.10) that depend on $\mathbf{s}$, the other terms being here considered as a constant:

$$\begin{aligned}\mathcal{L}_\mathbf{s}(\theta, \phi; \mathbf{o}) = {} & \mathbb{E}_{q_{\phi_\mathbf{s}}(\mathbf{s}|\mathbf{o})}\Big[\mathbb{E}_{q_{\phi_\mathbf{w}}(\mathbf{w}|\mathbf{o})}\big[\log p_{\theta_\mathbf{o}}(\mathbf{o}|\mathbf{w}, \mathbf{s})\big] \\ & + \mathbb{E}_{q_{\phi_\mathbf{z}}(\mathbf{z}|\mathbf{s})}\big[\log p_{\theta_\mathbf{sz}}(\mathbf{s}, \mathbf{z}) - \log q_{\phi_\mathbf{z}}(\mathbf{z}|\mathbf{s})\big] - \log q_{\phi_\mathbf{s}}(\mathbf{s}|\mathbf{o})\Big].\end{aligned} \tag{3.11}$$

Let us define:

$$\tilde{p}(\mathbf{s}|\mathbf{o}) = \mathcal{C}' \exp\Big(\mathbb{E}_{q_{\phi_\mathbf{w}}(\mathbf{w}|\mathbf{o})}\big[\log p_{\theta_\mathbf{o}}(\mathbf{o}|\mathbf{w}, \mathbf{s})\big] + \mathbb{E}_{q_{\phi_\mathbf{z}}(\mathbf{z}|\mathbf{s})}\big[\log p_{\theta_\mathbf{sz}}(\mathbf{s}, \mathbf{z}) - \log q_{\phi_\mathbf{z}}(\mathbf{z}|\mathbf{s})\big]\Big), \tag{3.12}$$

where $\mathcal{C}' > 0$ is the appropriate normalisation constant. Equation 3.11 rewrites:

$$\mathcal{L}_{\mathbf{s}}(\theta, \phi; \mathbf{o}) = -D_{\mathrm{KL}}\big(q_{\phi_{\mathbf{s}}}(\mathbf{s}|\mathbf{o}) \parallel \tilde{p}(\mathbf{s}|\mathbf{o})\big) + \mathcal{C}, \tag{3.13}$$

where $D_{\mathrm{KL}}(\cdot|\cdot)$ denotes the KL divergence. Therefore, the optimal distribution is the one minimising the above KL divergence:

$$q_{\phi_{\mathbf{s}}}(\mathbf{s}|\mathbf{o}) = \tilde{p}(\mathbf{s}|\mathbf{o}) \propto \exp\Big(\mathbb{E}_{q_{\phi_{\mathbf{w}}}(\mathbf{w}|\mathbf{o})}\big[\log p_{\theta_{\mathbf{o}}}(\mathbf{o}|\mathbf{w}, \mathbf{s})\big] + \mathbb{E}_{q_{\phi_{\mathbf{z}}}(\mathbf{z}|\mathbf{s})}\big[\log p_{\theta_{\mathbf{sz}}}(\mathbf{s}, \mathbf{z}) - \log q_{\phi_{\mathbf{z}}}(\mathbf{z}|\mathbf{s})\big]\Big). \tag{3.14}$$

Since for any pair $(t, k)$, the assignment variable $w_{tk}$ follows a discrete posterior distribution, we can denote the corresponding probability values by $\eta_{tkn} = q_{\phi_{\mathbf{w}}}(w_{tk} = n|\mathbf{o}_{tk})$. These values will be computed in the E-W step below. The expectation with respect to $q_{\phi_{\mathbf{w}}}(\mathbf{w}|\mathbf{o})$ in Equation 3.14 can be calculated using these values. However, the expectation with respect to $q_{\phi_{\mathbf{z}}}(\mathbf{z}|\mathbf{s})$ cannot be calculated in closed form. As usually done in the (D)VAE methodology, it is thus replaced by a Monte Carlo estimate using sampled sequences drawn from the DVAE inference model at the previous iteration (this point will be specified in Section 3.3.5). Replacing the distributions in Equation 3.14 with Equation 3.2, Equation 3.6, and Equation 3.8, and calculating the expectations with respect to $q_{\phi_{\mathbf{w}}}(\mathbf{w}|\mathbf{o})$ and $q_{\phi_{\mathbf{z}}}(\mathbf{z}|\mathbf{s})$, we find that $q_{\phi_{\mathbf{s}}}(\mathbf{s}|\mathbf{o})$ factorizes with respect to $n$ as follows:

$$q_{\phi_{\mathbf{s}}}(\mathbf{s}|\mathbf{o}) = \prod_{n=1}^{N} q_{\phi_{\mathbf{s}}}(\mathbf{s}_{:,n}|\mathbf{o}). \tag{3.15}$$

Each of these factors corresponds to the posterior distribution of the $n$-th source vector. Given Equation 3.14 and the DVAE generative and inference models, we see that at a given time $t$, the distribution over $\mathbf{s}_{tn}$ has non-linear dependencies w.r.t. the previous and current DVAE latent variables $\mathbf{z}_{1:t,n}$ and the previous source vectors $\mathbf{s}_{1:t-1,n}$. These non-linear dependencies impede to obtain an efficient closed-form solution. We resort to point sample estimates obtained using samples of $\mathbf{z}_{1:t,n}$ and of $\mathbf{s}_{1:t-1,n}$, at the current iteration, denoted $\mathbf{z}^{(i)}_{1:t,n}$ and $\mathbf{s}^{(i)}_{1:t-1,n}$. Using these samples, the posterior distribution is approximated with (details can be found in Appendix A.1.1):

$$q_{\phi_{\mathbf{s}}}(\mathbf{s}_{:,n}|\mathbf{o}) \approx \prod_{t=1}^{T} q_{\phi_{\mathbf{s}}}(\mathbf{s}_{tn}|\mathbf{s}^{(i)}_{1:t-1,n}, \mathbf{z}^{(i)}_{1:t,n}, \mathbf{o}), \tag{3.16}$$

where each term of the product is shown to be a Gaussian:

$$q_{\phi_{\mathbf{s}}}(\mathbf{s}_{tn}|\mathbf{s}^{(i)}_{1:t-1,n}, \mathbf{z}^{(i)}_{1:t,n}, \mathbf{o}) = \mathcal{N}(\mathbf{s}_{tn}; \mathbf{m}_{tn}, \mathbf{V}_{tn}), \tag{3.17}$$

with covariance matrix and mean vector given by:

$$\mathbf{V}_{tn} = \Big( \sum_{k=1}^{K_t} \eta_{tkn} \mathbf{\Phi}_{tk}^{-1} + \text{diag}(\boldsymbol{v}^{(i)}_{\theta_{\mathbf{s}},tn})^{-1} \Big)^{-1}, \tag{3.18}$$

$$\mathbf{m}_{tn} = \mathbf{V}_{tn} \Big( \sum_{k=1}^{K_t} \eta_{tkn} \mathbf{\Phi}_{tk}^{-1} \mathbf{o}_{tk} + \text{diag}(\boldsymbol{v}^{(i)}_{\theta_{\mathbf{s}},tn})^{-1} \boldsymbol{\mu}^{(i)}_{\theta_{\mathbf{s}},tn} \Big), \tag{3.19}$$

where $\boldsymbol{v}^{(i)}_{\theta_{\mathbf{s}},tn}$ and $\boldsymbol{\mu}^{(i)}_{\theta_{\mathbf{s}},tn}$ are simplified notations for $\boldsymbol{v}_{\theta_{\mathbf{s}}}(\mathbf{s}^{(i)}_{1:t-1,n}, \mathbf{z}^{(i)}_{1:t,n})$ and $\boldsymbol{\mu}_{\theta_{\mathbf{s}}}(\mathbf{s}^{(i)}_{1:t-1,n}, \mathbf{z}^{(i)}_{1:t,n})$, respectively denoting the variance and mean vector provided by the DVAE decoder network for source $n$ at time frame $t$. As we have to sample both $\mathbf{s}_{:,n}$ and $\mathbf{z}_{:,n}$, we need to pay attention to the sampling order. This will be discussed in detail in Section 3.3.5. Importantly, in practice, $\mathbf{m}_{tn}$ is used as the estimate of $\mathbf{s}_{tn}$.

Equation 3.19 shows that the estimated $n$-th source vector is obtained by combining the observations $\mathbf{o}_{tk}$ and the mean source vector $\boldsymbol{\mu}^{(i)}_{\theta_{\mathbf{s}},tn}$ predicted by the DVAE generative model. The balance between these two terms depends on the assignment variable $\eta_{tkn}$, the observation model covariance matrix $\mathbf{\Phi}_{tk}$ and the source variance predicted by the DVAE generative model $\boldsymbol{v}^{(i)}_{\theta_{\mathbf{s}},tn}$. Ideally, the model should be able to appropriately balance these two terms so as to optimally exploit both the observations and the DVAE predictions.

### 3.3.2 E-Z STEP

In the E-Z step, we consider the DVAE inference model $q_{\phi_{\mathbf{z}}}(\mathbf{z}|\mathbf{s})$, defined by Equation 3.8, Equation 2.37 and Equation 2.38.

In Equation 3.10, the corresponding term is the third one, which we denote by $\mathcal{L}_{\mathbf{z}}(\theta_{\mathbf{s}}, \theta_{\mathbf{z}}, \phi_{\mathbf{z}}; \mathbf{o})$

and which factorizes across sources as follows (see Appendix A.1.1):

$$\mathcal{L}_{\mathbf{z}}(\theta_{\mathbf{s}}, \theta_{\mathbf{z}}, \phi_{\mathbf{z}}; \mathbf{o}) = \sum_{n=1}^{N} \mathcal{L}_{\mathbf{z},n}(\theta_{\mathbf{s}}, \theta_{\mathbf{z}}, \phi_{\mathbf{z}}; \mathbf{o}), \tag{3.20}$$

with

$$\begin{aligned}\mathcal{L}_{\mathbf{z},n}(\theta_{\mathbf{s}}, \theta_{\mathbf{z}}, \phi_{\mathbf{z}}; \mathbf{o}) = \mathbb{E}_{q_{\phi_{\mathbf{s}}}(\mathbf{s}_{:,n}|\mathbf{o})}\Big[&\mathbb{E}_{q_{\phi_{\mathbf{z}}}(\mathbf{z}_{:,n}|\mathbf{s}_{:,n})}\big[\log p_{\theta_{\mathbf{sz}}}(\mathbf{s}_{:,n}, \mathbf{z}_{:,n})\big] \\ &- \mathbb{E}_{q_{\phi_{\mathbf{z}}}(\mathbf{z}_{:,n}|\mathbf{s}_{:,n})}\big[\log q_{\phi_{\mathbf{z}}}(\mathbf{z}_{:,n}|\mathbf{s}_{:,n})\big]\Big].\end{aligned} \tag{3.21}$$

Inside the expectation $\mathbb{E}_{q_{\phi_{\mathbf{s}}}(\mathbf{s}_{:,n}|\mathbf{o})}[\cdot]$, we recognize the DVAE ELBO, which is defined as:

$$\mathcal{L}_{\mathbf{z},n}(\theta_{\mathbf{s}}, \theta_{\mathbf{z}}, \phi_{\mathbf{z}}; \mathbf{s}_{1:T}) = \mathbb{E}_{q_{\phi_{\mathbf{z}}}(\mathbf{z}_{1:T}|\mathbf{s}_{1:T})}\big[\log p_{\theta_{\mathbf{sz}}}(\mathbf{s}_{1:T}, \mathbf{z}_{1:T}) - \log q_{\phi_{\mathbf{z}}}(\mathbf{z}_{1:T}|\mathbf{s}_{1:T})\big], \tag{3.22}$$

applied to source $n$. This suggests the following strategy. Previously to and independently of the MixDVAE algorithm, we pre-train the DVAE model on a dataset of synthetic or natural unlabeled single-source sequences.[4] This is done only once, and the resulting DVAE is then plugged into the MixDVAE algorithm to process multi-source sequences. This provides the E-Z step with very good initial values of the DVAE parameters $\theta_{\mathbf{s}}$, $\theta_{\mathbf{z}}$ and $\phi_{\mathbf{z}}$. As for the following of the E-Z step, the expectation over $q_{\phi_{\mathbf{s}}}(\mathbf{s}_{:,n}|\mathbf{o})$ in Equation 3.21 is not analytically tractable. A Monte Carlo estimate is thus used instead, using samples of both $\mathbf{z}$ and $\mathbf{s}$, similarly to what was done in the E-S step. Finally, SGD is used to maximize (the Monte Carlo estimate of) $\mathcal{L}_{\mathbf{z}}(\theta_{\mathbf{s}}, \theta_{\mathbf{z}}, \phi_{\mathbf{z}}; \mathbf{o})$, jointly updating $\theta_{\mathbf{s}}$, $\theta_{\mathbf{z}}$ and $\phi_{\mathbf{z}}$; that is, we fine-tune the DVAE model within the MixDVAE algorithm, using the observations $\mathbf{o}$. Note that in our experiments presented in the two next chapters, we also consider the case where we neutralize the fine-tuning, i.e. we remove the E-Z step and use the DVAE model as provided by the pre-training phase.

---

[4] When we apply MixDVAE to MOT and SC-ASS in the next chapters, this pre-training will be detailed in Sections 4.3 and 5.3, respectively.

### 3.3.3 E-W STEP

Thanks to the separation of $\mathbf{w}$ from the two other latent variables in Equation 3.7, the posterior distribution $q_{\phi_\mathbf{w}}(\mathbf{w}|\mathbf{o})$ can be calculated in closed form by directly applying the optimal structured mean-field update equation Equation 2.53 to our model. It can be shown that this is equivalent to maximizing Equation 3.10 w.r.t. $q_{\phi_\mathbf{w}}(\mathbf{w}|\mathbf{o})$. We obtain (see Appendix A.1.1 for details):

$$q_{\phi_\mathbf{w}}(\mathbf{w}|\mathbf{o}) \propto \prod_{t=1}^{T}\prod_{k=1}^{K_t} q_{\phi_\mathbf{w}}(w_{tk}|\mathbf{o}), \tag{3.23}$$

with

$$q_{\phi_\mathbf{w}}(w_{tk}=n|\mathbf{o}) = \eta_{tkn} = \frac{\beta_{tkn}}{\sum_{i=1}^{N}\beta_{tki}}, \tag{3.24}$$

where

$$\beta_{tkn} = \mathcal{N}(\mathbf{o}_{tk}; \mathbf{m}_{tn}, \mathbf{\Phi}_{tk}) \exp\Big(-\frac{1}{2}\mathrm{Tr}\big(\mathbf{\Phi}_{tk}^{-1}\mathbf{V}_{tn}\big)\Big). \tag{3.25}$$

The parameters $\mathbf{m}_{tn}$ and $\mathbf{V}_{tn}$ in the above equation have been defined in Equation 3.19 and Equation 3.18, respectively.

### 3.3.4 M STEP

As discussed in Section 2.3.2, the maximization step generally consists in estimating the parameters $\theta$ of the generative model by maximizing the ELBO over $\theta$. We recall that $\theta = \{\theta_\mathbf{o} = \{\mathbf{\Phi}_{tk}\}_{t,k=1}^{T,K_t}, \theta_\mathbf{s}, \theta_\mathbf{z}\}$. In this work, the parameters of the DVAE decoder $\theta_\mathbf{s}$ and $\theta_\mathbf{z}$ are first estimated (offline) during the pre-training of the DVAE and then fine-tuned in the E-Z step in an amortized way, all this jointly with the parameters of the encoder $\phi_\mathbf{z}$. Therefore, in the M-step, we only need to estimate the observation model covariance matrices $\theta_\mathbf{o} = \{\mathbf{\Phi}_{tk}\}_{t,k=1}^{T,K_t}$. In Equation 3.10, only the first term depends on $\theta_\mathbf{o}$. Setting its derivative with respect to $\mathbf{\Phi}_{tk}$ to zero, we obtain (see Appendix A.1.1 for details):

$$\mathbf{\Phi}_{tk} = \sum_{n=1}^{N} \eta_{tkn}\Big((\mathbf{o}_{tk}-\mathbf{m}_{tn})(\mathbf{o}_{tk}-\mathbf{m}_{tn})^T + \mathbf{V}_{tn}\Big). \tag{3.26}$$

In practice, it is difficult to obtain a reliable estimation using only a single observation. We address this issue in Sections 4.4.3 and 5.4.3.

### 3.3.5 MIXDVAE COMPLETE ALGORITHM

As already mentioned in Section 3.3.1, we must pay attention to the sampling order of $\mathrm{s}$ and $\mathbf{z}$ when running the iterations of the E-S and E-Z steps. As indicated in the pseudo-code of Algorithm 1, in practice, the E-S and E-Z steps are processed jointly. We start with the initial source vectors sequence $\mathrm{s}^{(0)}_{1:T,1:N}$ and initial mean source vectors sequence $\mathbf{m}^{(0)}_{1:T,1:N}$. At any iteration $i$ of the E-Z and E-S steps, for each source $n$ and each time frame $t$, we sample in the following order:

1. Compute the parameters $\boldsymbol{\mu}^{(i)}_{\phi_{\mathbf{z}},tn}$ and $\boldsymbol{v}^{(i)}_{\phi_{\mathbf{z}},tn}$[5] of the posterior distribution of $\mathbf{z}_t$ using the DVAE encoder network with inputs $\mathrm{s}^{(i-1)}_{1:T,n}$ sampled at the previous iteration and $\mathbf{z}^{(i)}_{1:t-1,n}$ sampled at the current iteration. Then, sample $\mathbf{z}^{(i)}_{tn}$ from $q_{\phi_{\mathbf{z}}}(\mathbf{z}_{tn}|\mathrm{s}^{(i-1)}_{1:T,n}, \mathbf{z}^{(i)}_{1:t-1,n})$.

2. Compute the parameters $\boldsymbol{\mu}^{(i)}_{\theta_{\mathbf{z}},tn}$ and $\boldsymbol{v}^{(i)}_{\theta_{\mathbf{z}},tn}$[6] of the generative distribution of $\mathbf{z}_t$ using the corresponding DVAE decoder network with inputs $\mathrm{s}^{(i)}_{1:t-1,n}$ and $\mathbf{z}^{(i)}_{1:t-1,n}$, both sampled at the current iteration.

3. Compute the parameters $\boldsymbol{\mu}^{(i)}_{\theta_{\mathrm{s}},tn}$ and $\boldsymbol{v}^{(i)}_{\theta_{\mathrm{s}},tn}$ of the generative distribution of $\mathrm{s}_t$ using the corresponding DVAE decoder network with inputs $\mathrm{s}^{(i)}_{1:t-1,n}$ and $\mathbf{z}^{(i)}_{1:t,n}$, both sampled at the current iteration. Compute the parameters $\mathbf{m}^{(i)}_{tn}$ and $\mathbf{V}^{(i)}_{tn}$ of the posterior distribution of $\mathrm{s}_t$ with Equation 3.18 and Equation 3.19, and sample $\mathrm{s}^{(i)}_{tn}$ from it.

Note that with the above sampling order, the Monte Carlo estimate of the ELBO term maximized in the E-Z step Equation 3.21 is given by (for source $n$):

$$\begin{aligned}\widehat{\mathcal{L}}_{\mathbf{z},n}(\theta_{\mathrm{s}}, \theta_{\mathbf{z}}, \phi_{\mathbf{z}}; \mathbf{o}) = & \sum_{t=1}^{T} \log p_{\theta_{\mathrm{s}}}(\mathrm{s}^{(i)}_{tn}|\mathrm{s}^{(i)}_{1:t-1,n}, \mathbf{z}^{(i)}_{1:t,n}) \\ & - \sum_{t=1}^{T} D_{\mathrm{KL}}\big(q_{\phi_{\mathbf{z}}}(\mathbf{z}_{tn}|\mathrm{s}^{(i-1)}_{1:T,n}, \mathbf{z}^{(i)}_{1:t-1,n})||p_{\theta_{\mathbf{z}}}(\mathbf{z}_{tn}|\mathrm{s}^{(i)}_{1:t-1,n}, \mathbf{z}^{(i)}_{1:t-1,n})\big). \quad (3.27)\end{aligned}$$

---

[5] $\boldsymbol{\mu}^{(i)}_{\phi_{\mathbf{z}},tn}$ and $\boldsymbol{v}^{(i)}_{\phi_{\mathbf{z}},tn}$ are shortcuts for $\boldsymbol{\mu}_{\phi_{\mathbf{z}}}\big(\mathrm{s}^{(i-1)}_{1:T,n}, \mathbf{z}^{(i)}_{1:t-1,n}\big)$ and $\boldsymbol{v}_{\phi_{\mathbf{z}}}\big(\mathrm{s}^{(i-1)}_{1:T,n}, \mathbf{z}^{(i)}_{1:t-1,n}\big)$ respectively.

[6] Analogous definitions hold.

**Algorithm 1** MixDVAE algorithm

```
Input:
    Observation vectors $\mathbf{o} = \mathbf{o}_{1:T,1:K_t}$;
Output:
    Parameters of $q_{\phi_\mathbf{s}}(\mathbf{s})$ : $\{\mathbf{m}_{tn}^{(I)}, \mathbf{V}_{tn}^{(I)}\}_{t,n=1}^{T,N}$ (the estimated $n$-th source vector at time frame $t$ is $\mathbf{m}_{tn}$);
    Values of the assignment variable $\{\eta_{tkn}^{(I)}\}_{t,n,k=1}^{T,N,K_t}$;
Initialization
    See Sections 4.4.2 and 5.4.2
for $i \leftarrow 1$ to $I$ do
    E-W Step
    for $n \leftarrow 1$ to $N$ do
        for $t \leftarrow 1$ to $T$ do
            for $k \leftarrow 1$ to $K_t$ do
                Compute $\eta_{tkn}^{(i)}$ using (3.24) and (3.25);
            end for
        end for
    end for
    E-Z and E-S Step
    for $n \leftarrow 1$ to $N$ do
        for $t \leftarrow 1$ to $T$ do
            Encoder;
            Compute $\boldsymbol{\mu}_{\phi_\mathbf{z},tn}^{(i)}$, $\boldsymbol{v}_{\phi_\mathbf{z},tn}^{(i)}$ with input $\mathbf{s}_{1:T,n}^{(i-1)}$ and $\mathbf{z}_{1:t-1,n}^{(i)}$;
            Sample $\mathbf{z}_{tn}^{(i)}$ from $q_{\phi_\mathbf{z}}(\mathbf{z}_{tn}|\mathbf{s}_{1:T,n}^{(i-1)}, \mathbf{z}_{1:t-1,n}^{(i)}) = \mathcal{N}\big(\mathbf{z}_{tn}; \boldsymbol{\mu}_{\phi_\mathbf{z},tn}^{(i)}, \text{diag}(\boldsymbol{v}_{\phi_\mathbf{z},tn}^{(i)})\big)$;
            Decoder;
            Compute $\boldsymbol{\mu}_{\theta_\mathbf{z},tn}^{(i)}$ and $\boldsymbol{v}_{\theta_\mathbf{z},tn}^{(i)}$ with input $\mathbf{s}_{1:t-1,n}^{(i)}$ and $\mathbf{z}_{1:t-1,n}^{(i)}$;
            Compute $\boldsymbol{\mu}_{\theta_\mathbf{s},tn}^{(i)}$ and $\boldsymbol{v}_{\theta_\mathbf{s},tn}^{(i)}$ with input $\mathbf{s}_{1:t-1,n}^{(i)}$ and $\mathbf{z}_{1:t,n}^{(i)}$;
            E-S update;
            Compute $\mathbf{m}_{tn}^{(i)}$, $\mathbf{V}_{tn}^{(i)}$ using (3.19) and (3.18);
            Sample $\mathbf{s}_{tn}^{(i)}$ from $\mathcal{N}(\mathbf{s}_{tn}; \mathbf{m}_{tn}^{(i)}, \mathbf{V}_{tn}^{(i)})$;
        end for
        E-Z update;
        Compute $\widehat{\mathcal{L}}_n(\theta_\mathbf{s}, \theta_\mathbf{z}, \phi_\mathbf{z}; \mathbf{o})$ using (3.27);
    end for
    Compute $\widehat{\mathcal{L}}(\theta_\mathbf{s}, \theta_\mathbf{z}, \phi_\mathbf{z}; \mathbf{o}) = \sum_{n=1}^{N} \widehat{\mathcal{L}}_n(\theta_\mathbf{s}, \theta_\mathbf{z}, \phi_\mathbf{z}; \mathbf{o})$;
    Fine-tune the DVAE parameters $\{\theta_\mathbf{s}, \theta_\mathbf{z}, \phi_\mathbf{z}\}$ by applying SGD on $\widehat{\mathcal{L}}(\theta_\mathbf{s}, \theta_\mathbf{z}, \phi_\mathbf{z}; \mathbf{o})$;
    M Step
    Compute $\mathbf{\Phi}_{tk}^{(i)}$ using (3.26) or following Sections 4.4.3 and 5.4.3;
end for
```

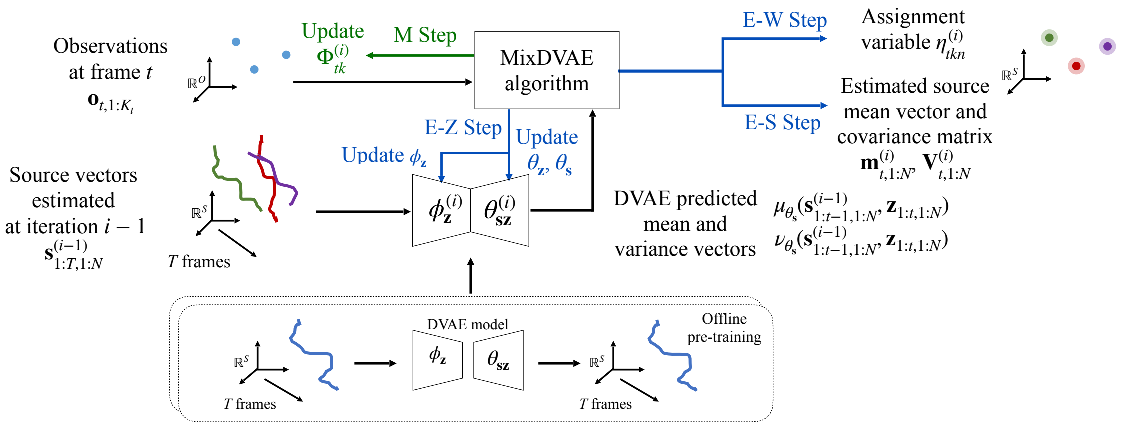


**Figure 3.2**: Overview of the proposed MixDVAE algorithm at a given time frame $t$. The DVAE model is pretrained offline using a (synthetic or natural) single-source dataset. It takes as input the sequence of source vectors, encodes them into a sequence of latent vectors, which are then decoded into the reconstructed sequence of source vectors. For a given time frame $t$, the MixDVAE algorithm takes as input the observations at time $t$ as well as the mean and variance vectors estimated by the DVAE model. By iterating the E-S, E-Z, E-W and M steps, we obtain estimates of the assignment variable and of each source vector.

The whole MixDVAE algorithm, taking into account these practical aspects, is summarized in the form of pseudo-code in Algorithm 1.[7] In addition, Fig. 3.2 shows a schematic overview of the algorithm.

### 3.3.6 CHOICE OF THE DVAE MODEL

We recall that the DVAE is a general class of models that differ by adopting different conditional independence assumptions for the generative distributions in the right-hand-side of Equation 2.34. In [71], seven DVAE models from the literature have been extensively discussed, and six of them have been benchmarked on the analysis-resynthesis task (on speech signals and 3D human motion data). We chose to use here the stochastic recurrent neural network (SRNN) model initially proposed in [55], because it was shown in [71] to provide a very good trade-off between model complexity and modeling power. The

---

[7] As illustrated in Sections 4.4.2 and 5.4.2, in practice, we can choose different VEM step orders. Here we present the algorithm with the order E-S/E-Z Step, E-W Step and M Step.

probabilistic dependencies of the SRNN generative model are defined as follows:

$$p_{\theta_{\mathbf{sz}}}(\mathbf{s}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_{\theta_{\mathbf{s}}}(\mathbf{s}_t|\mathbf{s}_{1:t-1}, \mathbf{z}_t) p_{\theta_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{s}_{1:t-1}, \mathbf{z}_{t-1}). \tag{3.28}$$

To perform online estimation, we use the following causal SRNN inference model:

$$q_{\phi_{\mathbf{z}}}(\mathbf{z}_{1:T}|\mathbf{s}_{1:T}) = \prod_{t=1}^{T} q_{\phi_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{s}_{1:t}, \mathbf{z}_{t-1}). \tag{3.29}$$

The implementation details of the SRNN model can be found in Appendix A.1.2.

# CHAPTER 4

# APPLICATION OF MIXDVAE ON MOT

*Knowledge is the beginning of practice;*
*doing is the completion of knowing.*
知是行之始，行是知之成。

*— Wang Yangming*

This chapter is based on the following publication:

Xiaoyu Lin, Laurent Girin, and Xavier Alameda-Pineda. “Mixture of dynamical variational autoencoders for multi-source trajectory modeling and separation.” In Transactions on Machine Learning Research, 2023.

## 4.1 APPLICATION OF MIXDVAE ON MOT

In the previous chapter, we presented the general principles and solutions of MixDVAE . In this chapter, we introduce the application of the MixDVAE model and the corresponding algorithm to the MOT problem. This requires to first specify the data and model configurations in this context.

As discussed in Section 2.5.1, a classical solution to the MOT problem is the ‘tracking-by-detection’ paradigm. In this paradigm, a set of DBBs are given at each time step by a front-end detection algorithm, each of them potentially corresponding to one of the targets. These DBBs are then used as the observations. In Chapter 4, we apply the MixDVAE model to the MOT problem in this configuration (i.e., using the set of DBBs as observations). We do that in a simplified scenario where the number of objects is assumed known and constant during the observation measurements. A complete and fully-operational MOT system would require to include a module managing the ‘birth’ and ‘death’ of target tracks (i.e., objects disappearing of the scene and new objects appearing in the scene). We do not address this problem since the purpose of this work is not to propose a fully-operational MOT system but rather to focus on the problem of multi-source dynamics modeling with DVAEs. It must be noted however that even if we assume that the actual number of objects present in the scene is known and does not vary across the modeled sequence, at any time step $t$, it is not necessarily equal to the number of DBBs, since occlusions (leading to missed detections) may occur. We will see in our reported experiments that MixDVAE is able to deal with these difficulties.

## 4.2 SETTING MIXDVAE IN THE MOT CONFIGURATION

As mentioned in Section 4.1, under the tracking-by-detection configuration, the objective of the MOT task is to estimate the trajectories of moving objects from a set of given DBBs. In this case, the source vector $\mathbf{s}_{tn}$ represents the position of object $n$ at time frame $t$, which is given in practice by the coordinates of the (top-left and bottom-right points of the) "true" corresponding bounding box, i.e. $\mathbf{s}_{tn} = (s_{tn}^{\text{L}}, s_{tn}^{\text{T}}, s_{tn}^{\text{R}}, s_{tn}^{\text{B}}) \in \mathbb{R}^4$. The observation vector $\mathbf{o}_{tk} = (o_{tk}^{\text{L}}, o_{tk}^{\text{T}}, o_{tk}^{\text{R}}, o_{tk}^{\text{B}}) \in \mathbb{R}^4$ contains the coordinates of the (top-left and bottom-right points of the) $k$-th DBB at frame $t$. In a VAE or DVAE, the dimension $L$ of the latent vector $\mathbf{z}_{tn}$ is usually smaller than the dimension of the observed vector, in order to obtain a compact data representation. Since in the MOT task the data dimension is already small ($O = S = 4$), we also set $L = 4$. The sequence of estimated source position vectors is given directly by Equation 3.19, for $n = 1$ to $N$ and $t = 1$ to $T$, directly forming source trajectories, with no further post-processing.

## 4.3 DVAE PRE-TRAINING

### 4.3.1 DATASET

We consider pedestrian tracking for the MOT task and assume that all the moving sources have similar dynamical patterns. We thus pre-train a single DVAE model on a synthetic single-source trajectory dataset. This dataset contains synthetic bounding box trajectories in the form of $T$-frame sequences ($T = 60$) of 4D vectors $\{(x_t^{\text{L}}, x_t^{\text{T}}, x_t^{\text{R}}, x_t^{\text{B}})\}_{t=1}^{T}$. These trajectories are generated using piece-wise combinations of several elementary functions, namely: static $a(t) = a_0$, constant velocity $a(t) = a_1 t + a_0$, constant acceleration $a(t) = a_2 t^2 + a_1 t + a_0$, and sinusoidal (allowing for circular trajectories) $a(t) = a \sin(\omega t + \phi_0)$. The parameters $a_1$, $a_2$, $\omega$, and $\phi_0$ are sampled from some pre-defined distributions, whose parameters are estimated from the detections on the training subset of the MOT17 dataset [44], which is a widely-used pedestrian tracking dataset (rapidly described it in the next subsection). The two remaining parameters, $a_0$ and $a$, are set to the values that ensure continuous trajectories. More details about the single-source synthetic trajectories generation can be found in Appendix A.2.2. Overall, we generated $12{,}105$ sequences for the training dataset and $3{,}052$ sequences for the validation dataset.

### 4.3.2 TRAINING DETAILS

The SRNN model used in our experiments is an auto-regressive model, i.e., it uses the past source vectors $\mathbf{s}_{1:t-1}$ to predict the current one $\mathbf{s}_t$. In practice, the estimated past vectors are used for this prediction, rather than the ground-truth past vectors. To make the model robust to this problem, we trained the model in the scheduled sampling mode [14]. This means that during training, we gradually replace the ground-truth past values with the previously generated ones to predict the current value (see [71, Chapter 4] for a discussion on this issue). The model was trained using the Adam optimizer [104] with a learning rate set to $0.001$ and a batch size set to $256$. An early-stopping strategy was adopted, with a patience of $50$ epochs.

## 4.4 MIXDVAE EVALUATION SET-UP

### 4.4.1 DATASET

For the evaluation of the proposed MixDVAE algorithm, we used the training set of MOT17. MOT17 contains pedestrian scenes filmed in different places such as in a shopping mall or in a street, with static or moving cameras. The motion patterns of the pedestrians in these videos are quite diverse and challenging. The MOT17 training set contains seven sequences with length varying from twenty seconds to one minute, with different frame rates (14, 25, and 30 fps). The ground-truth bounding boxes are provided, as well as the detection results obtained with three customized detectors, namely DPM [51], Faster-RCNN [166], and SDP [226]. As briefly stated in the introduction, we focus our study on modeling the source dynamics for multiple-source tasks. Therefore, we leave aside the problem of appearing/disappearing sources (usually referred to as birth/death processes) and consider a fixed number of $N = 3$ tracks. We have thus designed a new dataset from the MOT17 training set, which we call the MOT17-3T dataset. The MOT17-3T dataset uses the publicly-released DBBs of the MOT17 dataset. We split a complete video sequence into subsequences of sequence length $T$. Three values of $T$ are evaluated in our experiments: 60, 120, and 300 frames (respectively corresponding to 2, 4, and 10 seconds at 30 fps). Each test sequence contains three source trajectories with possible occlusions and detection absences, see an example in Fig. 4.1. More details on the design of the MOT17-3T dataset can be found in Appendix A.2.2. We have finally created $1{,}712$, $1{,}161$, and $1{,}137$ 3-source test sequences of length $T = 60, 120$, and 300 frames, respectively. Notice that the pre-trained DVAE is not fine-tuned on these test sequences.

### 4.4.2 ALGORITHM INITIALIZATION

Before starting the iterations of the proposed VEM algorithm as described in Algorithm 1, we need to initialize the values of several parameters and variables. Theoretically, there is no preference in the order of the three E-steps. In practice, however, for initialization convenience, we followed the order E-W Step, E-Z/E-S Step. Indeed, starting with E-W Step requires the initialization of the mean vector and covariance matrix of the source vector posterior distribution $\mathbf{m}_{tn}, \mathbf{V}_{tn}$, the input vectors of the DVAE encoder $\mathbf{s}_{1:T,n}$ and the observation covariance matrices $\mathbf{\Phi}_{tk}$. For MOT, $\mathbf{m}_{tn}$ can be easily initialised over a short sequence by assuming that the source does not move too much. Indeed, the initial values of $\mathbf{m}_{tn}$ can be set to the value of the observed bounding box at the beginning of the sequence $\mathbf{m}_{0n}$. While this strategy is very straightforward to implement, it is too simple for many tracking scenarios, especially for long sequences. We thus propose to split a long sequence into sub-sequences. For each sub-sequence, we initialise $\mathbf{m}_{tn}$ to the value at the beginning of the sub-sequence. After this initialisation, we run a few iterations of the VEM algorithm over the sub-sequence, allowing us to have an estimate of the source position at the end of the sub-sequence. This value is then used to provide a constant initialisation for the next sub-sequence. At the end, all these initializations are concatenated, providing a piece-wise constant initialization for $\mathbf{m}_{tn}$ over the entire long sequence. More implementation details, as well as the pseudo-code of this cascade initialization strategy, are provided in Appendix A.2.1. The input vectors of the DVAE encoder are initialized with the same values as the ones used for $\mathbf{m}_{tn}$.

### 4.4.3 OBSERVATION COVARIANCE MATRIX

In our MOT experiments, we observed that the estimated values of both $\mathbf{\Phi}_{tk}$ and $\boldsymbol{v}_{\theta_{\mathbf{s}},tn}$ in Equation 3.19 increased very quickly with the VEM algorithm iterations. This caused instability and unbalance between these two terms, which finally conducted the whole model to diverge. To solve this problem, we set $\mathbf{\Phi}_{tk}$ to a given fixed value, which is constant on the whole analyzed $T$-frame sequence and not updated during the VEM iterations. Specifically, for the MOT task, $\mathbf{\Phi}_{tk}$ is set to a diagonal matrix, and the diagonal entries are set to $r^2_{\mathbf{\Phi}}\left[(o^{\mathrm{R}}_{1k} - o^{\mathrm{L}}_{1k})^2, (o^{\mathrm{T}}_{1k} - o^{\mathrm{B}}_{1k})^2, (o^{\mathrm{R}}_{1k} - o^{\mathrm{L}}_{1k})^2, (o^{\mathrm{T}}_{1k} - o^{\mathrm{B}}_{1k})^2\right]$, where $r_{\mathbf{\Phi}}$ is a factor lower than 1. In common terms, $\mathbf{\Phi}_{tk}$ is set to a fraction of the (squared) size of the corresponding observation at frame 1. The covariance matrices $\mathbf{V}_{tn}$ are initialized with the same values as $\mathbf{\Phi}_{tk}$.

### 4.4.4 HYPERPARAMETERS

The VEM algorithm of MixDVAE has four hyperparameters to be set. The observation covariance matrix ratio $r_{\mathbf{\Phi}}$ is set to $0.04$, the initialization subsequence length $J$ is set to 30, and the initialization iteration number $I_0$ is set to 20. The MixDVAE algorithm itself is run for $I = 70$ iterations, which was experimentally shown to lead to convergence.

### 4.4.5 BASELINES

We compare our model with two recent state-of-the-art probabilistic MOT methods: The Autoregressive Tracklet Inpainting and Scoring for Tracking (ArTIST) model of [179] and the Variational Kalman Filter (VKF) of [8]. In addition to that, in order to demonstrate the advantage of using a DVAE model for modeling the dynamics of single-source trajectories, we consider replacing the DVAE model with a simpler deep auto-regressive (Deep AR) model. ArTIST is a supervised stochastic autoregressive model that learns the discretized multi-modal distribution of human motion using annotated MOT sequences. It can assign detections to tracks by scoring tracklet[1] proposals with their likelihood. And

---

[1] A tracklet indicates a sequence of estimated position vectors consistent over time and assigned to the same object.

it can also generate continuations of the source trajectories and inpaint those containing missing detections. We have reused the trained models as well as the tracklet scoring and inpainting code provided by the authors[2] and reimplemented the object tracking part according to the paper, as this part was not provided. Implementation details can be found in Appendix A.2.3. Alike the proposed MixDVAE algorithm, the VKF algorithm for MOT [8] is also based on the VI methodology to combine source position estimation and detection-to-source assignment. However, a basic one-step linear dynamical model is used in VKF instead of the DVAE model in the proposed MixDVAE algorithm. In short, the dynamical model we use in VKF is $p_{\theta_{\mathbf{s}}}(\mathbf{s}_t|\mathbf{s}_{t-1}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{s}_{tn}; \mathbf{D}\mathbf{s}_{t-1,n}, \mathbf{\Lambda}_{tn})$, where $\mathbf{D}$ is assumed to be the identity matrix and $\mathbf{\Lambda}_{tn}$ is estimated in the M step. Hence, the VKF MOT algorithm is a combination of VI and Kalman filter update equations. In [8], the method was proposed in an audiovisual set-up. The observations contain not only the DBBs coordinates, but also appearance features and multichannel audio recordings. For a fair comparison with MixDVAE, we use here the same observations, i.e., we simplified VKF by using only the DBBs coordinates. For both ArTIST and VKF, the tracked sequences are initialized using the DBBs at the first frame, as what we have done for MixDVAE. For VKF, similarly to MixDVAE, we need to provide initial values for $\mathbf{m}_{tn}$ and $\mathbf{V}_{tn}$. For a fair comparison, we applied the same cascade initialization as the one presented above, except that a linear dynamical model is used in place of the DVAE to ensure the transition between two consecutive subsequences. The covariance matrices $\mathbf{V}_{tn}$ are initialized with pre-defined values that stabilize the EM algorithm. The covariance matrices $\mathbf{\Phi}_{tk}$ are fixed to the same values as for MixDVAE. The covariance matrices of the linear dynamical model (denoted $\mathbf{\Lambda}_{tn}$ in [8]) are initialized with the same values as $\mathbf{V}_{tn}$. Finally the simpler Deep AR baseline model is a deep generative model without stochastic latent variables. In this baseline, the dynamical model becomes $p_{\theta_{\mathbf{s}}}(\mathbf{s}_t|\mathbf{s}_{t-1}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{s}_{t,n}; \boldsymbol{\mu}_{\theta_{\mathbf{s}}}(\mathbf{s}_{1:t-1,n}), \text{diag}(\mathbf{v}_{\theta_{\mathbf{s}}}(\mathbf{s}_{1:t-1,n})))$. In practice, the Deep AR model is implemented with an LSTM layer. The hidden dimension of the LSTM layer is set to match that of the LSTM layers employed in the DVAE model, i.e. it is equal to 8.

---

[2] available at https://github.com/fatemeh-slh/ArTIST

### 4.4.6 EVALUATION METRICS

We use the standard MOT metrics [17, 171] to evaluate the tracking performance of MixDVAE and compare it to the baselines, namely: multi-object tracking accuracy (MOTA), multi-object tracking precision (MOTP), identity F1 score (IDF1), number of identity switches (IDS), mostly tracked (MT), mostly lost (ML), false positives (FP) and false negatives (FN). The three test subsets contain a different number of test sequences, with a different sequence length $T$. Therefore, for IDS, FP and FN, we report both the number of occurrences and the corresponding percentage. Among them, MOTA is considered to be the most representative metric. It is defined by aggregating the frame-wise versions of the metrics $\mathrm{FP}_t$, $\mathrm{FN}_t$, and $\mathrm{IDS}_t$ over frames:

$$\mathrm{MOTA} = 1 - \frac{\sum_t (\mathrm{FN}_t + \mathrm{FP}_t + \mathrm{IDS}_t)}{\sum_t \mathrm{GT}_t}, \tag{4.1}$$

where $\mathrm{GT}_t$ denotes the number of ground-truth tracks at frame $t$. Higher MOTA values imply less errors (in terms of FPs, FNs, and IDS), and hence better tracking performance. MOTP defines the averaged overlap between all correctly matched sources and their corresponding ground truth. Higher MOTP implies more accurate position estimations. IDF1 is the ratio of correctly identified detections over the average number of ground-truth and computed detections. IDS reflects the capability of the model to preserve the identity of the tracked sources, especially in case of occlusion and track fragmentation. MT and ML represent how much the trajectory is recovered by the tracking algorithm. A source track is mostly tracked (resp. mostly lost) if it is covered by the tracker for at least $80\%$ (resp. not more than $20\%$) of its life span.

## 4.5 EXPERIMENTAL RESULTS

### 4.5.1 QUANTITATIVE ANALYSIS

We now present and discuss the tracking results obtained with the proposed MixDVAE algorithm and compare them with those obtained with the baselines. In these experiments, the value of the observation variance ratio $r_{\mathbf{\Phi}}$ is set to 0.04 and no fine-tuning is applied

**Table 4.1**: MOT results for short ($T = 60$), medium ($T = 120$), and long ($T = 300$) sequences.

| Dataset | Method | MOTA$\uparrow$ | MOTP$\uparrow$ | IDF1$\uparrow$ | #IDS$\downarrow$ | %IDS$\downarrow$ | MT$\uparrow$ | ML$\downarrow$ | #FP$\downarrow$ | %FP$\downarrow$ | #FN$\downarrow$ | %FN$\downarrow$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Short | ArTIST | 63.7 | **84.1** | 48.7 | 86371 | 28.0 | **4684** | **0** | **9962** | **3.2** | **15525** | **5.0** |
|  | VKF | 56.0 | 82.7 | 77.3 | 5660 | 1.8 | 3742 | 761 | 64945 | 21.1 | 64945 | 21.1 |
|  | Deep AR | 67.4 | 76.1 | 83.1 | 5248 | 1.7 | 3670 | 129 | 49595 | 16.0 | 49595 | 16.0 |
|  | MixDVAE | **79.1** | 81.3 | **88.4** | **4966** | **1.6** | 4370 | 50 | 29808 | 9.7 | 29808 | 9.7 |
| Medium | ArTIST | 61.0 | **84.2** | 43.9 | 102978 | 24.6 | **2943** | **0** | **25388** | **6.1** | **34812** | **8.3** |
|  | VKF | 57.5 | 83.3 | 77.6 | 7657 | 1.8 | 2563 | 487 | 85053 | 20.3 | 85053 | 20.3 |
|  | Deep AR | 65.3 | 76.0 | 81.8 | **5387** | **1.3** | 2435 | 149 | 71775 | 17.0 | 71775 | 17.0 |
|  | MixDVAE | **78.6** | 82.2 | **88.0** | 6107 | 1.5 | 2907 | 120 | 41747 | 9.9 | 41747 | 9.9 |
| Long | ArTIST | 53.5 | 84.5 | 40.7 | 205263 | 20.1 | 2513 | **4** | 135401 | 13.2 | 135401 | 13.2 |
|  | VKF | 74.4 | **86.2** | 84.4 | 30069 | 2.9 | 2756 | 100 | 116160 | 11.4 | 116160 | 11.4 |
|  | Deep AR | 75.5 | 76.6 | 87.1 | 26506 | 2.6 | 2555 | 18 | 123262 | 12.1 | 123262 | 12.1 |
|  | MixDVAE | **83.2** | 82.4 | **90.0** | **23081** | **2.3** | **2890** | 12 | **74550** | **7.3** | **74550** | **7.3** |

to SRNN in the E-Z step. Ablation study on these factors is presented in Section 4.6.

The values of the MOT metrics obtained on short, medium and long sequence subsets ($T = 60$, $120$, and $300$ frames, respectively) are shown in Table 4.1. We see that the proposed MixDVAE algorithm obtains the best MOTA scores for the three subsets (i.e., for the three different sequence length values). This is remarkable given that ArTIST was trained on the MOT17 training dataset, whereas MixDVAE never saw the ground-truth sequences before the test. Furthermore, we notice that both VKF and MixDVAE have much less IDS and much higher IDF1 scores than ArTIST, which implies that the observation-to-source assignment based on the VI method is more efficient than direct estimation of the position likelihood distribution to preserve the correct source identity during tracking. Besides, the MixDVAE model also has better scores than the VKF model for these two metrics, which implies that the DVAE-based dynamical model performs better on identity preservation than the linear dynamical model of VKF. For the 60- and 120-frame sequences, the ArTIST model has lower FP and FN percentages and higher MOTP scores (though the MOTP scores of all three algorithms are quite close for every value of $T$). This is reasonable because, again, ArTIST was trained on the same dataset using the ground-truth sequences while our model is unsupervised. Overall, the adverse effect caused by frequent identity switches is much greater than the positive effect of lower FP and FN for the ArTIST model. That explains why MixDVAE has much better MOTA scores than ArTIST. For the long (300-frame) sequences, MixDVAE obtains an overall much better performance than the ArTIST model, since it obtains here the best scores for 6 metrics out of 8, including FP and FN. This shows that MixDVAE is particularly good at tracking objects on the long term (we recall that $T = 300$ represents 10 s of video at 30 fps).

Besides, MixDVAE also globally exhibits notably better performance than VKF on all of the three datasets. This clearly indicates that the modeling of the sources dynamics with a DVAE model outperforms the use of a simple linear-Gaussian dynamical model and can greatly improve the tracking performance. We can also notice that the VKF algorithm globally performs much better on 300-frame sequences than on 60- and 120-frame sequences. One possible explanation for this phenomenon is that the dynamical patterns of long sequences are simpler than those of short and medium sequences. In fact, the data statistics show that the average velocity in long sequences is much lower than that in short and medium sequences. In this case, the linear dynamical model can perform quite well –although not as well as the DVAE.

Finally, we can see in Table 4.1 that MixDVAE with SRNN as dynamical model has an overall significantly better performance than MixDVAE with the baseline Deep AR dynamical model. This demonstrates the important role of the latent variables in SRNN for the dynamical modeling of sequential data. We remind that the latent vector $\mathbf{z}_{:,n}$ is assumed to efficiently encode the generative factors of source $n$'s trajectory.

### 4.5.2 QUALITATIVE ANALYSIS

To illustrate the behavior of MixDVAE and the baseline models, we present an example of tracking result in Fig. 4.1. More examples can be found in Appendix A.2.4. In the example of Fig. 4.1, the detection for Source 3 ($o_3$ in the figure) is absent from $t = 2$ and reappears after $t = 20$. But we limit the plot to $t = 10$ for a better visualization. This is a case of long-term detection absence. An immediate identity switch occurs at $t = 2$ for the ArTIST model. Then, the track obtained by ArTIST is no longer stable. We speculate the reason for the frequent identity switches made by ArTIST is that the estimated distributions do not correspond well to the true sequential position distributions, which is possibly

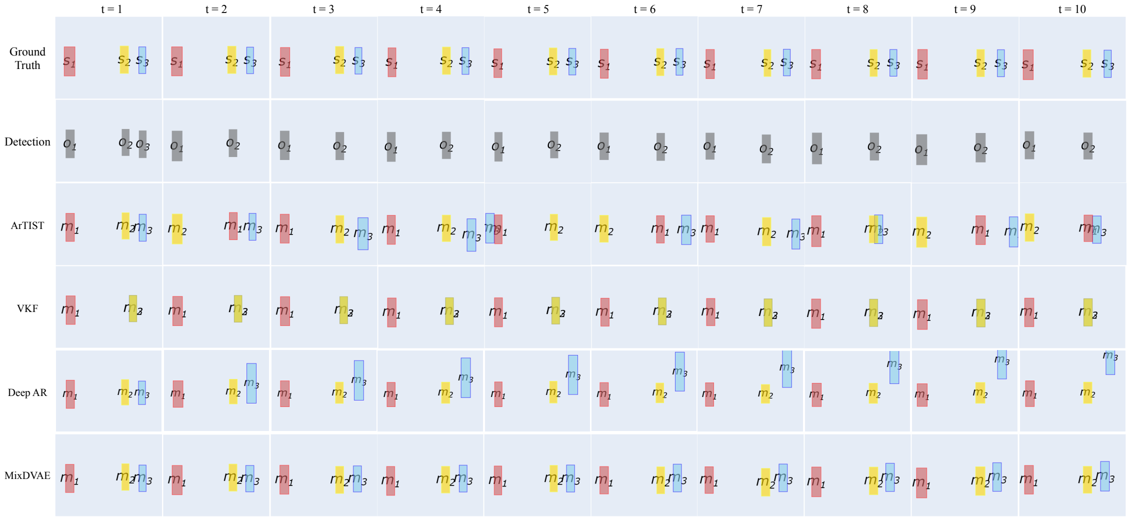


**Figure 4.1**: Example of tracking result obtained with the proposed MixDVAE algorithm and the two baselines. For clarity of presentation, the simplified notations $s_1$, $o_1$, and $m_1$ denote the ground-truth source position, the observation, and the estimated source position, respectively (for Source 1, and the same for the two other sources). Best seen in color.

due to the way these distributions are discretized. In addition to the identity switches, the estimations generated by ArTIST at $t = 5$, $8$, and $10$ are not accurate. This causes a decrease of the tracking performance. For the VKF model, the estimated bounding boxes for Sources 2 and 3 ($m_2$ and $m_3$ in the figure) overlap each other. This means that the two observations are both assigned to the same source, which is Source 2. From Equation 3.24 and Equation 3.25, we know that the value of the assignment variable depends on the posterior mean and variance vectors $\mathbf{m}_{tn}$ and $\mathbf{V}_{tn}$, which themselves depend on the dynamical model. With a linear dynamical model, VKF is not able to correctly predict distinct $m_2$ and $m_3$ trajectories. The Deep AR model succeed to predict distinct $m_2$ and $m_3$ trajectories. However, the trajectory $m_3$ is not accurate due to the absence of $o_3$. In contrast, the very good dynamical modeling capacity of the DVAE makes MixDVAE able to keep tracking despite of the long-term detection absence and generate reasonable $m_3$ estimations, which correspond well to the ground-truth trajectory of Source 3 ($s_3$ in the figure).

**Table 4.2**: Capacity of the SRNN model pre-trained at three data scales of the synthetic trajectories dataset. SRNN-full, SRNN-half, and SRNN-quarter stand for SRNN pre-trained on the totality, half of and quarter of our original training set, respectively.

| Model name | Training loss | Validation loss |
|---|---|---|
| SRNN-full | -40.77 | -40.15 |
| SRNN-half | -39.36 | -38.86 |
| SRNN-quarter | -36.55 | -35.27 |

## 4.6 ABLATION STUDIES

In order to better understand the MixDVAE model, we conducted ablation studies on the influence of the pre-trained DVAE model quality, the influence of fine-tuning the DVAE, and the influence of the observation variation matrix ratio $r_{\mathbf{\Phi}}$.

### 4.6.1 INFLUENCE OF THE PRE-TRAINED DVAE MODEL QUALITY

To understand the influence of the pre-trained DVAE model quality on the performance of MixDVAE , we have pre-trained the DVAE model at different data scales and tested the performance of MixDVAE using these different pre-trained models. Specifically, the SRNN model is pre-trained on three separate datasets with different scales: the full synthetic trajectories training set used in Section 4.3, consisting of $12{,}105$ trajectories, a dataset containing half of these synthetic trajectories, randomly selected ($6{,}052$ trajectories), and another dataset with a quarter of these synthetic trajectories, randomly selected ($3{,}026$ trajectories). We use the ELBO loss to represent the quality of the resulting pre-trained SRNN models. The ELBO loss values are reported in Table 4.2. As expected, we observe that by decreasing the training data size, the performance of the SRNN model drops (with higher training and validation loss).

We run the MixDVAE inference algorithm with the three pre-trained SRNN models on the short sequence test subset ($T = 60$ frames), and the obtained results are reported in Table 4.3. We can see that the overall performance of the MixDVAE algorithm with SRNN-half and SRNN-quarter drops compared to that with SRNN-full, but this drop

**Table 4.3**: MOT results obtained by MixDVAE with SRNN pre-trained at the three data scales. The results are reported for the short sequence test subset ($T = 60$ frames).

| Model name | MOTA↑ | MOTP↑ | IDF1↑ | #IDs↓ | %IDs↓ | MT↑ | ML↓ | #FP↓ | %FP↓ | #FN↓ | %FN↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| SRNN-full | **79.1** | 81.3 | **88.4** | **4966** | **1.6** | **4370** | **50** | **29808** | **9.7** | **29808** | **9.7** |
| SRNN-half | 74.7 | **84.4** | 86.6 | 5624 | 1.8 | 4039 | 94 | 38153 | 12.4 | 38153 | 12.4 |
| SRNN-quarter | 75.2 | **84.4** | 86.9 | 5598 | 1.8 | 4040 | 91 | 37443 | 12.2 | 37443 | 12.2 |

**Table 4.4**: MOT results obtained by MixDVAE with and without the fine-tuning of SRNN. The results are reported for the short, medium and long sequence test subsets ($T = 60$, $120$, and $300$ frames, respectively).

| Dataset | Fine-tuning | MOTA↑ | MOTP↑ | IDF1↑ | #IDs↓ | %IDs↓ | MT↑ | ML↓ | #FP↓ | %FP↓ | #FN↓ | %FN↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Short | Yes | 75.1 | **83.5** | 86.7 | **2862** | **0.9** | 4067 | 64 | 36990 | 11.9 | 36990 | 11.9 |
| | No | **79.1** | 81.3 | **88.4** | 4966 | 1.6 | **4370** | **50** | **29808** | **9.7** | **29808** | **9.7** |
| Medium | Yes | 73.1 | **84.0** | 85.9 | **3044** | **0.7** | 2705 | 136 | 54604 | 13.1 | 54604 | 13.1 |
| | No | **78.6** | 82.2 | **88.0** | 6107 | 1.5 | **2907** | **120** | **41747** | **9.9** | **41747** | **9.9** |
| Long | Yes | 65.6 | **84.9** | 81.6 | **8670** | **0.8** | 2286 | 67 | 171515 | 13.8 | 171515 | 13.8 |
| | No | **83.2** | 82.4 | **90.0** | 23081 | 2.3 | **2890** | **12** | **74550** | **7.3** | **74550** | **7.3** |

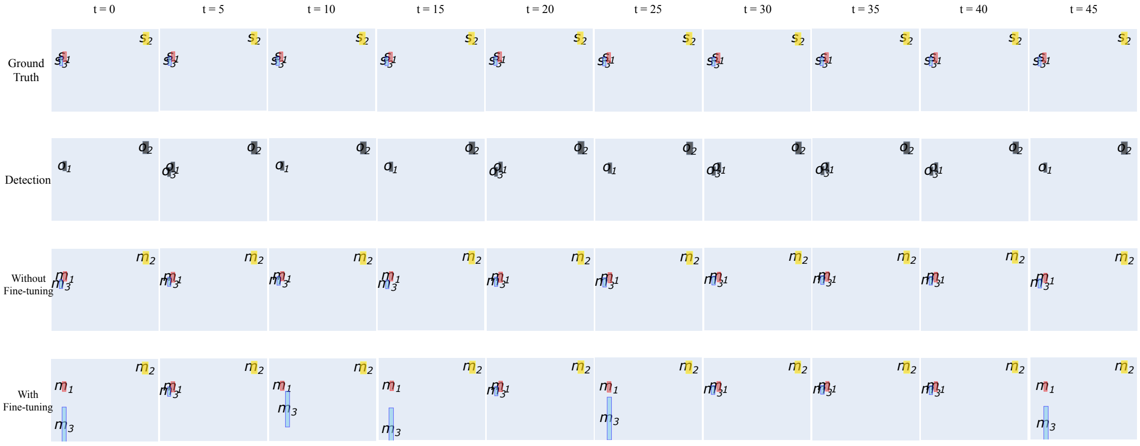
is relatively limited, at least for some of the metrics, including the key MOTA metric. Moreover, the difference between the performance of MixDVAE with SRNN-half and with SRNN-quarter is quite small. Therefore, even if it is hard to draw a general conclusion from a single experiment with three dataset sizes, this seems to indicate some robustness of MixDVAE w.r.t. the DVAE training dataset size, and confirm its interest as a data-frugal weakly supervised method (here for the MOT application).

### 4.6.2 INFLUENCE OF THE DVAE FINE-TUNING

As mentioned in Section 3.3.2, the DVAE model can either be fine-tuned or not in the MixDVAE algorithm. We have studied the effect of fine-tuning SRNN.

Table 4.4 shows the MOT scores obtained by MixDVAE on the three test subsets with and without the fine-tuning of SRNN in the E-Z step. We observe that for all three datasets, not fine-tuning the DVAE model leads to the best overall performance (as measured by MOTA in particular). Though fine-tuning the DVAE model can indeed increase the MOTP score and decrease the number of identity switches, it does not improve the overall tracking performance. Indeed, fine-tuning increases the FP and FN numbers/proportions, and thus decreases the MOTA scores. Especially on the long sequence dataset, the MOTA score drops from $83.2$ to $65.6$.

Example 1



Example 2

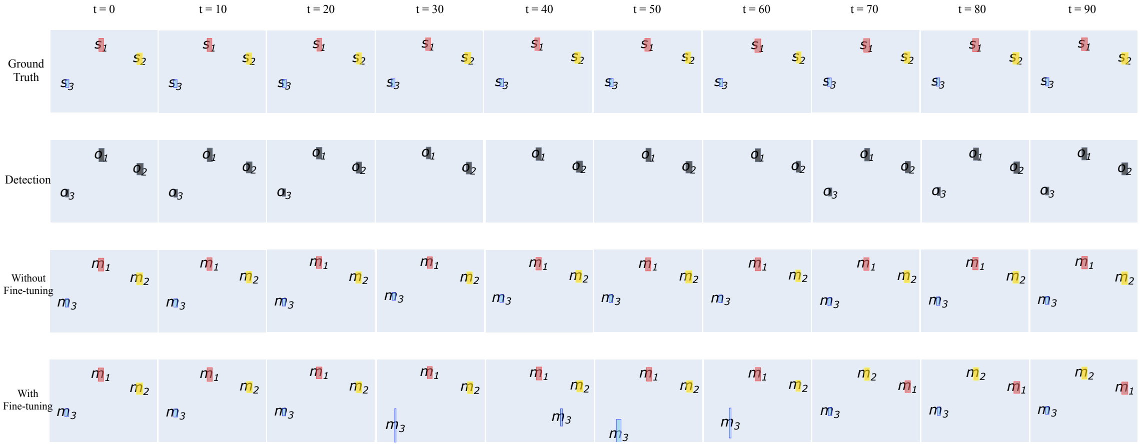


**Figure 4.2**: Two examples of tracking result obtained with the proposed MixDVAE algorithm, with and without fine-tuning during the E-Z step. For clarity of presentation, the simplified notations $s_1$, $o_1$, and $m_1$ denote the ground-truth source position, the observation, and the estimated position, respectively (for Source 1, and the same for the two other sources). Best seen in color.

We therefore observe that fine-tuning the DVAE model results in performance degradation. The possible reason is that fine-tuning could make the model more sensible to observation noise, and lead to a generative model with worse performance. To verify this conjecture and to better understand the effect of fine-tuning, we have plotted in Fig. 4.2 two tracking examples, extracted from the long sequence test subset ($T = 300$ frames).

**Table 4.5**: Results obtained by MixDVAE on MOT17-3T (short sequences subset) for different values of $r_{\mathbf{\Phi}}$. The values on the left (resp. right) side of the slashes are obtained without (resp. with) the fine-tuning of SRNN in the E-Z Step.

| $r_{\mathbf{\Phi}}$ | MOTA↑ | MOTP↑ | IDF1↑ | #IDs↓ | %IDs↓ | MT↑ | ML↓ | #FP↓ | %FP↓ | #FN↓ | %FN↓ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.01 | 35.9/32.8 | **84.5/84.8** | 66.6/65.5 | **4914**/3216 | **1.6**/1.0 | 2946/2714 | 916/913 | 96438/102062 | 31.3/33.1 | 96438/102062 | 31.3/33.1 |
| 0.02 | 65.5/61.8 | 84.2/84.7 | 81.3/79.8 | 5319/3073 | 1.7/1.0 | 3932/3652 | 407/379 | 50596/57291 | 16.4/18.6 | 50596/57291 | 16.4/18.6 |
| 0.03 | 74.9/70.0 | 83.1/84.3 | 86.1/84.4 | 5088/**2853** | 1.7/**0.9** | 4232/3931 | 158/160 | 36165/43777 | 11.7/14.2 | 36165/43777 | 11.7/14.2 |
| 0.04 | **79.1**/75.1 | 81.3/83.5 | **88.4**/86.7 | 4966/2862 | 1.6/0.9 | **4370**/4067 | 50/64 | **29808**/36990 | **9.7**/11.9 | **29808**/36990 | **9.7**/11.9 |
| 0.05 | 76.4/**75.6** | 79.2/82.6 | 87.1/**87.1** | 4982/2919 | 1.6/0.9 | 4268/**4066** | **42/53** | 33924/**36088** | 11.0/**11.7** | 33924/**36088** | 11.0/**11.7** |
| 0.06 | 69.2/70.1 | 76.9/82.0 | 83.5/84.4 | 5297/3005 | 1.7/1.0 | 3978/3845 | 73/137 | 44793/44598 | 14.5/14.5 | 44793/44598 | 14.5/14.5 |
| 0.07 | 59.8/66.8 | 74.8/80.3 | 78.9/82.9 | 5146/3000 | 1.7/1.0 | 3688/3775 | 188/285 | 59348/49646 | 19.2/16.1 | 59348/49646 | 19.2/16.1 |
| 0.08 | 48.5/60.6 | 73.1/79.4 | 73.3/79.9 | 5097/3119 | 1.7/1.0 | 3303/3637 | 337/432 | 76865/59220 | 24.9/19.2 | 76865/59220 | 24.9/19.2 |

To make possible the display of a long sequence in a limited space, the first example is plotted every 5 frames, whereas the second example is plotted every 10 frames. In Example 1, we observe that the detection for source $\mathrm{s}_3$ is missed for $t = 0$, $t = 10$, $t = 15$, $t = 25$ and $t = 45$. At these frames, MixDVAE without SRNN fine-tuning can still make a good estimation of $\mathrm{s}_3$'s position, whereas MixDVAE with SRNN fine-tuning can not make an accurate prediction. In the latter case, this caused a large error between the estimated source position and the ground truth. We can see a similar phenomena in Example 2. At frame $t = 30$, $t = 40$, $t = 50$ and $t = 60$, when the detection bounding box for source $\mathrm{s}_3$ is absent, the estimation obtained by MixDVAE with SRNN fine-tuning is bad (it is particularly bad for $t = 40$). This phenomenon confirms our conjecture that the observation noise, particularly the lack of observations, can introduce unforeseen effects during fine-tuning, resulting in a model with degraded performance.

### 4.6.3 INFLUENCE OF THE OBSERVATION VARIANCE RATIO

Table 4.5 reports the MOT scores obtained with MixDVAE as a function of $r_{\mathbf{\Phi}}$. These experiments are conducted on the subset of short sequences. We report the results for both with and without fine-tuning SRNN in the E-Z step. Apart from the value of $r_{\mathbf{\Phi}}$ and the fine-tuning option, all other conditions are exactly the same across experiments.

Table 4.5 shows that, whether fine-tuning SRNN in the E-Z step or not, the MOT scores first globally increase with $r_{\mathbf{\Phi}}$,[3] reach their optimal values for $r_{\mathbf{\Phi}} = 0.04$ or $0.05$ (for most

---

[3] Except for the MOTP score, which continually decreases with the increase of $r_{\mathbf{\Phi}}$. This can be explained as follows. MOTP measures the precision of the position estima-

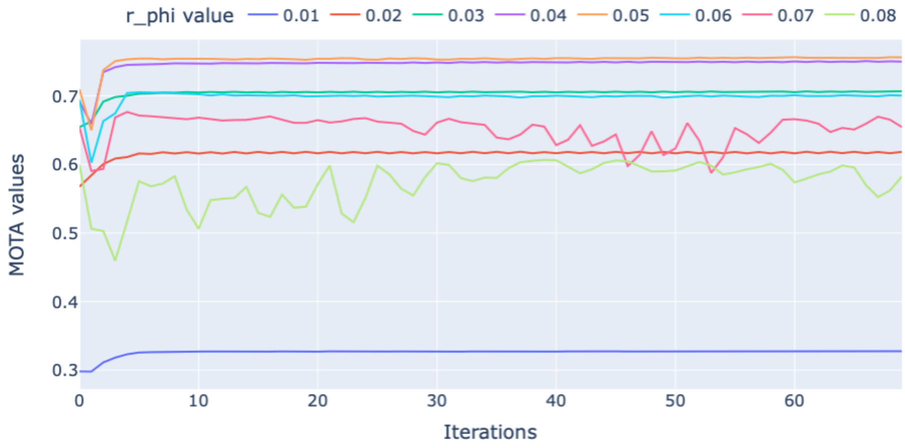


**Figure 4.3**: MOTA score obtained by MixDVAE as a function of the number of VEM iterations, for different values of $r_{\mathbf{\Phi}}$.

metrics), and then decrease for greater $r_{\mathbf{\Phi}}$ values. For confirmation, we have also computed the averaged empirical ratio $\hat{r}_{\mathbf{\Phi}}$ of the detected bounding boxes (with the SDP detector), which is calculated as $\frac{1}{4T}\sum_{t=1}^{T}\frac{1}{K_t}\sum_{k=1}^{K_t}\big(\frac{|s^{\mathrm{L}}_{tk}-o^{\mathrm{L}}_{tk}|}{o^{\mathrm{R}}_{tk}-o^{\mathrm{L}}_{tk}}+\frac{|s^{\mathrm{T}}_{tk}-o^{\mathrm{T}}_{tk}|}{o^{\mathrm{T}}_{tk}-o^{\mathrm{B}}_{tk}}+\frac{|s^{\mathrm{R}}_{tk}-o^{\mathrm{R}}_{tk}|}{o^{\mathrm{R}}_{tk}-o^{\mathrm{L}}_{tk}}+\frac{|s^{\mathrm{B}}_{tk}-o^{\mathrm{B}}_{tk}|}{o^{\mathrm{T}}_{tk}-o^{\mathrm{B}}_{tk}}\big)$.[4] This value equals to $0.053$, $0.053$ and $0.047$ respectively for the short, medium and long sequence dataset. These values, which are close to each other because we used the same detector, correspond well to the $r_{\mathbf{\Phi}}$ value for the best performing model in Table 4.5. We can conclude that the model has better performance if the value of $r_{\mathbf{\Phi}}$ corresponds (empirically) to the detector performance. Besides, we have also observed that the value of $r_{\mathbf{\Phi}}$ has an impact on the convergence of the MixDVAE algorithm. Fig. 4.3 displays the MOTA score as a function of the number of MixDVAE iterations (here with the fine-tuning of the DVAE model). It appears clearly that for too high values of $r_{\mathbf{\Phi}}$, the model exhibits a lower and more hectic performance than for the optimal value.

---

tion for the matched bounding boxes. The estimated position $\mathbf{m}_{tn}$ in Equation 3.19 is a weighted combination of the observation and the DVAE prediction. When $\mathbf{\Phi}_{tk}$ increases, the contribution of the observation decreases and $\mathbf{m}_{tn}$ is closer to the DVAE prediction. Since the error of the DVAE prediction may accumulate over time, this finally decreases the position estimation accuracy.

[4] Note that here $s_{tk}$ denotes the position of the target matched with the observation $o_{tk}$ at time frame $t$. We omit the target positions that are not matched with any observation.

# CHAPTER 5

# APPLICATION OF MIXDVAE ON SC-ASS

*Mathematics compares the most diverse*
*phenomena and discovers the secret*
*analogies that unite them.*

*— Joseph Fourier*

This chapter is based on the following publication:

Xiaoyu Lin, Laurent Girin, and Xavier Alameda-Pineda. “Mixture of dynamical variational autoencoders for multi-source trajectory modeling and separation.” In Transactions on Machine Learning Research, 2023.

## 5.1 APPLICATION OF MIXDVAE ON SC-ASS

In this chapter, we present the application of MixDVAE on the SC-ASS taks. Before diving into the details of the application configurations, we would like to make clear that the goal of applying MixDVAE to audio source separation is not to compete with supervised state-of-the-art methods on this task, but it is rather to illustrate its versatility, i.e. deliver a proof-of-concept for its applicability to tasks that are as diverse as MOT in computer vision and audio source separation. Besides, one major benefit of the MixDVAE is that, as explained in Section 3.1, it can be considered as data-frugal and weakly supervised since it only uses one or several single-source(-type) dataset(s) of moderate size for training, in contrast to supervised state-of-the-art models, which use massive dataset of parallel/aligned mixture and single-source samples.

In SC-ASS, when applying the time-frequency masking approach and the W-disjoint assumption (as discussed in Section 2.5.2), each time-frequency (TF) bin of the spectrogram of the observed mixture signal is assumed to belong to a dominant source. This mask can be modeled in a probabilistic way using a discrete assignment variable. Therefore, we can apply the proposed MixDVAE model to solve the SC-ASS problem by combining this assignment variable with a set of DVAEs modeling the dynamics of the audio sources in the TF domain. The main difference compared to the MOT problem is that we have to consider the frequency dimension in addition to time, and at a given TF bin, we have here only one single observation to assign to a source, instead of a set of observations. We apply this principle and illustrate the use of MixDVAE for SC-ASS in Chapter 5. It can be noted that the dynamics of different types of audio source signals (speech, musical instruments, noises, etc.) can be very different. So, this is a typical use-case where we can pre-train different DVAE models on different single-source datasets to capture the dynamics of different types of source.

## 5.2 SETTING MIXDVAE IN THE SC-ASS CONFIGURATION

When applying MixDVAE to the SC-ASS task, we work in the STFT domain. This implies that both the source and observation vectors are complex-valued. More precisely, the $n$-th source vector $\mathbf{s}_{tn} = \{s_{tn,f}\}_{f=1}^{F} \in \mathbb{C}^F$ is the short-time spectrum of audio source $n$ at time frame $t$ ($f$ denotes the frequency bin and $\mathbf{s} \in \mathbb{C}^{T \times F}$ is the complete STFT spectrogram). The number of frequency bins, $F$, is typically set to $256$, $512$ or $1024$ (a power of 2 is preferred to use the fast Fourier transform). As is usually adopted in audio processing, $\mathbf{s}_{tn}$ is assumed to follow a zero-mean circularly-symmetric complex Gaussian prior distribution [62, 124, 72], i.e. Equation 2.35 becomes $p_{\theta_{\mathbf{s}}}(\mathbf{s}_t|\mathbf{s}_{1:t-1}, \mathbf{z}_{1:t}) = \mathcal{N}_c\big(\mathbf{s}_t; \mathbf{0}, \mathrm{diag}(\boldsymbol{v}_{\theta_{\mathbf{s}},t}(\mathbf{s}_{1:t-1}, \mathbf{z}_{1:t}))\big)$. More specifically, the complex Gaussian distributions are used to characterize complex random variables whose real and imaginary parts are jointly Gaussian distributed [75]. Given a multivariate complex random variable $\mathbf{Z} = \mathbf{X} + i\mathbf{Y} \in \mathbb{C}^d$ that follows a complex Gaussian distribution, the distribution is characterized by three parameters: the mean vector $\mu = \mathbb{E}[\mathbf{Z}] \in \mathbb{C}^d$, the covariance matrix $\boldsymbol{\Gamma} = \mathbb{E}[(\mathbf{Z} - \boldsymbol{\mu})(\mathbf{Z} - \boldsymbol{\mu})^H] \in \mathbb{C}^{d \times d}$, and the relation matrix $\mathbf{C} = \mathbb{E}[(\mathbf{Z} - \boldsymbol{\mu})(\mathbf{Z} - \boldsymbol{\mu})^T] \in \mathbb{C}^{d \times d}$. In the special case of central (zero-mean) circularly-symmetric Gaussian distribution, both $\boldsymbol{\mu}$ and $\mathbf{C}$ equal zero, and the PDF of $\mathbf{Z}$ only depends on its magnitude but not on its phase [111]. In such cases, the squared magnitude $|\mathbf{Z}|^2$ follow a exponential distribution whereas the phase is uniformly distributed between $[-\pi, \pi]$. As in practice, the DNNs can only take real values as input. When processing the STFT spectrograms, the DVAE model will input the squared magnitude of the STFT coefficients, instead of the complex values. By injecting the exponential conditional distributions into the ELBO, it becomes

$$\mathcal{L}(\theta_{\mathbf{s}}, \phi_{\mathbf{z}}; \mathbf{s}_{1:T}) = -\sum_{t=1}^{T} \mathbb{E}_{q_{\phi_{\mathbf{z}}}} \Big[ d_{\mathrm{IS}}(|\mathbf{s}_t|^2, \mathbf{v}_{\theta_{\mathbf{s}},t}) + D_{\mathrm{KL}}\big(q_{\phi_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{s}_{1:T}) || p_{\theta_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{s}_{1:t-1}, \mathbf{z}_{1:t-1})\big) \Big],$$

where $d_{\text{IS}}(\cdot, \cdot)$ is the Itakura-Saito (IS) divergence [62], and $D_{\text{KL}}(\cdot||\cdot)$ is the KL divergence.

The latent space dimension $L$ is typically set to a value significantly lower than $F$. Also, here, $\mathbf{o} = \{o_{tf}\}_{t,f=1}^{TF} \in \mathbb{C}^{F \times T}$ denotes the STFT spectrogram of the observed mixture signal. We define the $k$-th observation variable at frame $t$ as $\mathbf{o}_{tk} = o_{tf} \in \mathbb{C}$, which is the STFT coefficient of the mixture signal at TF bin $(t, f)$. In other words, the observation index $k$ is identified with the frequency bin/index $f$, and the total number of observations at any frame $t$ equals the number of frequency bins, i.e. $K_t = F$ for each $t$. We note that in this case, $\mathbf{s}_{tn}$ and $\mathbf{o}_{tk}$ do not have the same dimension, even though $\mathbf{s}_{tn}$ and $\mathbf{o}_{t,:}$ do. Therefore, as mentioned in Footnote 2 of Chapter 3, we need to define a projection matrix $\mathbf{P}_k \in \mathbb{C}^{1 \times F}$, which is here the transposed one-hot vector activated at the $k$-th index. Finally, the observation $\mathbf{o}_{tk} = o_{tf}$ is modeled with a conditional circularly-symmetric complex Gaussian, centered at the corresponding source coefficient, and Equation 3.3 becomes $p_{\theta_{\mathbf{o}}}(\mathbf{o}_{tk}|w_{tk} = n, \mathbf{s}_{tn}) = \mathcal{N}_c(\mathbf{o}_{tk}; \mathbf{P}_k \mathbf{s}_{tn}, \mathbf{\Phi}_{tk})$. We can see that the assignment variable $w_{tk}$ associates each TF-bin of the observed mixture spectrogram to one of the sources, and thus implicitly defines a TF mask. All these adaptations yield minimal changes in the MixDVAE derivation and solution. These changes are provided in Appendix A.3.1, including the final source vector estimate. Note that the estimated $n$-th source waveform is obtained by applying the inverse STFT on $\mathbf{m}_{1:T,n}$.

## 5.3 DVAE PRE-TRAINING

### 5.3.1 DATASET

We illustrate the application of MixDVAE to the SC-ASS problem with the separation of a speech signal and a musical instrument, in the present case the Chinese bamboo flute (CBF). These two audio sources have very different spectral and dynamical patterns. So, we choose here to pre-train two instances of the same DVAE model separately on two single-source datasets, a speech dataset and a CBF dataset. For the speech dataset, we used the Wall Street Journal (WSJ0) dataset [63], which is composed of 16-kHz monophonic speech signals, with three subsets: *si_tr_s*, *si_dt_05* and *si_et_05*, used for model training, validation and test, and containing 24.9, 2.2 and 1.5 hours of speech recordings, respectively. For the musical instrument dataset, we used the CBF dataset of [215], which contains CBF performances recorded by 10 professional CBF performers. The dataset comprises recordings of both isolated playing techniques and full-length pieces. We only used the full pieces recordings in our experiments. The original recordings are stereo and at a sampling frequency of 44.1kHz. In our experiments, we used only one channel and downsampled the signals to 16-kHz, to match the speech signals rate. We selected the second half pieces recordings of player 1 and 2 as the validation set, the second half pieces recordings of player 3, 4 and 5 as the test set, and use all other recordings for DVAE pre-training. The total duration of the training, validation and test sets are 2.1, 0.2 and 0.3 hours respectively. For both datasets, we used the training set for DVAE pre-training and the validation set for early stopping.

### 5.3.2 PRE-PROCESSING

For both the speech and CBF dataset, we pre-processed the raw audio signals in the following way. First, the silence at the beginning and end of each signal are trimmed with a voice activity detection threshold of 30 dB. Then, the waveform signals are normalized by dividing their absolute maximum value. The STFT coefficients are computed with a 64-ms sine window (1024 samples) and a 75%-overlap (256-sample hop length), resulting in sequences of 513-dimensional discrete Fourier coefficient vectors (for positive frequencies). Note that in practice, the DVAE model will input speech power spectrograms, i.e., the squared modulus of s, instead of the complex-valued STFT spectrograms [71, Chapter 13], and these STFT power spectrograms are split into smaller sequences of length 50 frames (corresponding to audio segments of 0.8 s) for training.

### 5.3.3 TRAINING DETAILS

We also used the SRNN model with scheduled sampling training (see Section 4.3). The model was trained with the Adam optimizer with a learning rate set to 0.002 and a batch size set to 256. The latent space dimension $L$ was set to 16. The early-stopping patience was set to 50 epochs for the WSJ0 dataset and 200 epochs for the CBF dataset.

## 5.4 MIXDVAE EVALUATION SET-UP

### 5.4.1 DATASET

To generate the test mixture signals, we first randomly selected two signals from the WSJ0 test set and the CBF test set, respectively. Then, we removed the silence at the beginning and end in the same way as for the pre-processing. The clipped speech and CBF signals were mixed together with several different speech-to-music (power) ratios, namely $-10$, $-5$, $0$ and $5$ dB. The waveform mixture signals were then normalized and transformed to STFT spectrograms in the same way as in the pre-processing. Similar to the MOT scenario, we tested MixDVAE with different test sequence length values. To this aim, the mixed signal STFT spectrograms were split into subsequences of length 50, 100 and 300 frames (respectively corresponding to audio segments of 0.8, 1.6 and 4.8 s). Overall, we generated $878$, $491$, and $372$ mixed test signals of length $T = 50, 100$ and $300$ frames, respectively.

### 5.4.2 ALGORITHM INITIALIZATION

As for MOT, we need to initialize the values of several parameters and variables. For SC-ASS, it is more difficult to obtain a reasonable initialization for $\mathbf{m}_{tn}$ (complex-valued) using directly the observed mixture signal. We thus choose the following VEM iteration order: E-Z/E-S Step, E-W Step. In this case, we have to initialize the posterior distribution of the assignment variable (i.e. all the values of $\eta_{tkn}$), the input vectors of the DVAE encoder $\mathrm{s}_{1:T,n}$ (for the two sources), and the observation model covariance matrices $\mathbf{\Phi}_{tk}$. We initialize $\eta_{tkn}$ with a discrete uniform distribution. As for the DVAE encoder, we first input the power spectrogram of the mixture signal (recall that the two DVAEs were pre-trained on different natural single-source datasets). We then use the reconstructed output power spectrograms as the initialization of the DVAE encoders input.

### 5.4.3 OBSERVATION COVARIANCE MATRIX

Similar to the MOT case, $\mathbf{\Phi}_{tk}$ is not estimated in the M Step, but fixed to $r_{\mathbf{\Phi}}^2 |o_{tk}|^2$. In plain words, $\mathbf{\Phi}_{tk}$ is set to a fraction of the observation power. This setting turned out to stabilize the VEM iteration process and finally led to very satisfying estimation results.

### 5.4.4 HYPERPARAMETERS

Regarding the hyperparameters of the MixDVAE VEM algorithm for the SC-ASS task, the observation covariance matrix ratio $r_{\mathbf{\Phi}}$ is set to 0.01. The total number of iterations $I$ is set to 70. And the DVAE model is not fine-tuned in the E-Z step for the reported experiments.

### 5.4.5 BASELINES

As mentioned in Section 2.5.2, the state-of-the-art SC-ASS methods are mostly fully supervised, thus requiring a very large amount of paired (aligned) mixture signals and individual source signals for training. Very few methods are under the weakly-supervised or unsupervised settings. Thus, it is difficult to find a fairly comparable baseline model. In the presented experiments, we have compared the proposed MixDVAE method with the unsupervised audio source separation method called MixIT [220] and with two weakly-supervised methods based on NMF, namely a vanilla NMF model [52] and an NMF model with temporal extensions [211]. MixIT is a deep-learning-based unsupervised single-channel source separation method. It is trained on a dataset constructed by mixing up the existing mixture audio signals. The model separates them into a variable number of latent source signals that can be remixed to approximate the original mixtures. In a totally unsupervised setting, MixIT does not require having the separated source signals for training. For the implementation of the MixIT model, we have reused the code provided by the authors and adapted it for the speech-CBF source separation task. In the weakly-supervised NMF baseline methods, an NMF model is first pre-trained on each single-source dataset separately, resulting in a dictionary of non-negative spectral templates $\mathbf{W}_n$ for each of the sources to separate. Such pre-training is similar in spirit to the pre-training stage of the

MixDVAE method. After that, the obtained spectral template dictionaries of all sources are fixed and concatenated together so as to learn the temporal activation matrix $\mathbf{H}$ for the test mixture signal. Then the $\mathbf{H}$ entries corresponding to the spectral templates in $\mathbf{W}_n$ are used to separate source $n$ (in practice, Wiener filters are build to separate the sources in the STFT domain, see [52] and [211] for details). We have re-implemented both NMF-based baselines according to the formula given in the corresponding papers. The latent dimension $K$ of the NMF model for both speech and CBF data is set to 128, which is determined by grid search. To demonstrate the interest of using a DVAE model for modeling the audio source dynamics in MixDVAE, we made additional experiments with replacing the DVAE model in MixDVAE with two other dynamical models: a linear-Gaussian dynamical model and a deep auto-regressive dynamical model, which results in baseline models similar to the VKF model and the Deep AR model that we have already used in our MOT experiments (see Section 4.4). For the VKF model, we initialize the values of $\eta_{tkn}$ in two ways: the ground-truth assignment mask, which is also named as ideal binary mask (IBM) in the audio source separation literature (we call the resulting model VKF-oracle) and the mask defined from the outputs of the pre-trained DVAEs when inputing the mixture signal spectrogram (we call the resulting model VKF-DVAE-init). Note that VKF-oracle provides an (unrealistic) upper bound of separation performance with a linear dynamical model, whereas VKF-DVAE-init uses the same initial information as MixDVAE. $\mathbf{\Phi}_{tk}$ are fixed to the same values as for MixDVAE and $\mathbf{\Lambda}_{tn}$ are initialized with the identity matrix multiplied by a scalar. For the Deep AR model, it is implemented with an LSTM layer, with the hidden dimension set equal to that of the LSTM layers employed in the DVAE model. Finally, to investigate the effects of the VEM algorithm in the MixDVAE model, we also compared our model with the direct reconstruction of the source signals from the output of the pre-trained DVAEs when using the mixture spectrogram as the input, i.e. the information used to intialize both MixDVAE and VKF-DVAE-init (we call this baseline method DVAE-init). As these output spectrograms are power spectrograms, we combined their square root (amplitude spectrogram) with the phase spectrogram of the mixture signal to reconstruct the waveform of the baseline separated signals.

### 5.4.6 EVALUATION METRICS.

We used four source separation performance metrics widely-used in speech/audio processing. The root mean squared error (RMSE), the scale-invariant signal-to-distortion ratio (SI-SDR) [175] in dB, and the perceptual evaluation of speech quality (PESQ) score [172] (values in $[-0.5, 4.5]$).[1] For all metrics, the higher the better.

## 5.5 EXPERIMENTAL RESULTS

### 5.5.1 QUANTITATIVE ANALYSIS

We report the speech-CBF separation results on the short, medium and long test sequence subsets ($T = 50$, $100$, $300$ frames, respectively) in Table 5.1. In addition to the results obtained by the different models, we also report the values of the evaluation metrics when applied on the mixture signal, for reference.

We observe that on the short and medium sequence subsets, MixDVAE achieves the best performance for all of the evaluation metrics. While on the long sequences subsets, MixIT obtains slightly better results than MixDVAE for the speech. This demonstrates that the proposed method works well on the SC-ASS task. Unsurprisingly, VKF-Oracle obtains the best scores on all metrics because it was initialized with the ground-truth mask. When comparing MixDVAE with the methods of different dynamical models, we find that MixDVAE obtains overall better performance than both VKF-DVAE-init and Deep AR on all of the metrics for all of the three subsets. It is clear that the non-linear DVAE model with stochastic latent variables is much more efficient than the linear-Gaussian model and the deep auto-regressive model without latent variables for modeling the audio source dynamics. Besides, with the increase of the sequence length, the performance of MixDVAE dropped quite moderately (less than $0.6$ dB and less than $0.3$ dB in SI-SDR gain decrease on speech and CBF respectively), while the performance of VKF dropped

---

[1] The PESQ objective measure was developed mostly for evaluating the quality of speech signals, but since it is largely based on a model of Human auditory perception, we assume we can also use it on the CBF sounds to avoid to complicate the evaluation protocol.

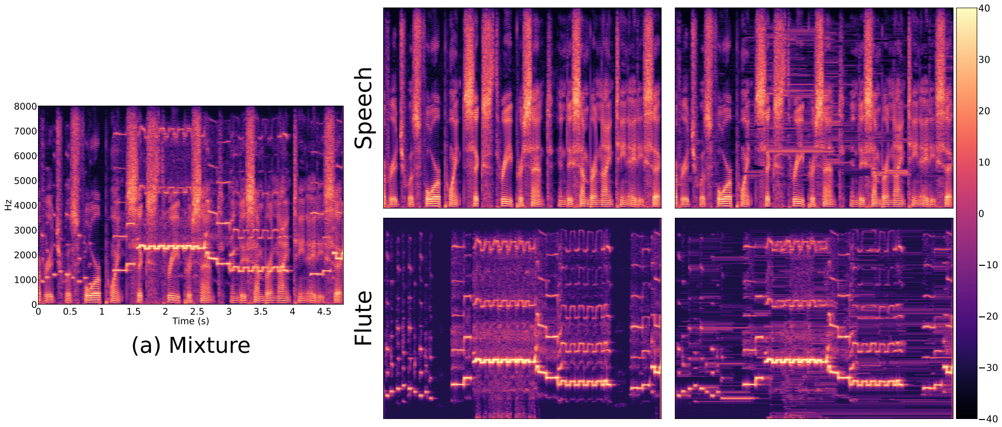
**Table 5.1**: SC-ASS results for short ($T = 50$), medium ($T = 100$), and long ($T = 300$) sequences.

| Dataset | Method | Speech |  |  | Chinese bamboo flute |  |  |
|---|---|---|---|---|---|---|---|
|  |  | RMSE ↓ | SI-SDR ↑ | PESQ ↑ | RMSE ↓ | SI-SDR ↑ | PESQ ↑ |
| Short | Mixture | 0.016 | -4.94 | 1.22 | 0.016 | 4.93 | 1.09 |
|  | VKF-Oracle | 0.004 | 14.83 | 2.00 | 0.004 | 20.15 | 2.33 |
|  | DVAE-init | 0.013 | -0.51 | 1.20 | 0.019 | 3.04 | 1.44 |
|  | VKF-DVAE-init | 0.012 | 2.24 | 1.21 | 0.012 | 8.06 | 1.33 |
|  | Deep AR | 0.009 | 5.32 | 1.29 | 0.018 | 5.19 | 1.48 |
|  | MixIT | 0.011 | 3.26 | - | 0.009 | 7.15 | - |
|  | Vanilla NMF | 0.011 | 3.01 | 1.40 | 0.012 | 9.09 | 1.37 |
|  | Temporal NMF | 0.009 | 4.99 | 1.53 | 0.011 | 10.26 | 1.53 |
|  | MixDVAE | **0.006** | **9.23** | **1.73** | **0.007** | **13.50** | **2.30** |
| Medium | Mixture | 0.016 | -4.44 | 1.17 | 0.016 | 4.44 | 1.08 |
|  | VKF-Oracle | 0.004 | 14.88 | 1.88 | 0.003 | 20.24 | 2.41 |
|  | DVAE-init | 0.014 | 0.10 | 1.15 | 0.020 | 2.42 | 1.27 |
|  | VKF-DVAE-init | 0.013 | 1.25 | 1.12 | 0.013 | 7.42 | 1.26 |
|  | Deep AR | 0.010 | 4.88 | 1.21 | 0.017 | 5.17 | 1.35 |
|  | MixIT | 0.009 | 4.75 | - | 0.009 | 8.74 | - |
|  | Vanilla NMF | 0.011 | 3.28 | 1.41 | 0.011 | 8.88 | 1.35 |
|  | Temporal NMF | 0.010 | 5.12 | 1.48 | 0.011 | 9.96 | 1.44 |
|  | MixDVAE | **0.007** | **9.32** | **1.65** | **0.007** | **13.05** | **2.16** |
| Long | Mixture | 0.016 | -4.52 | 1.19 | 0.016 | 4.53 | 1.10 |
|  | VKF-Oracle | 0.004 | 14.65 | 1.89 | 0.003 | 20.45 | 2.60 |
|  | DVAE-init | 0.013 | 0.20 | 1.15 | 0.020 | 2.29 | 1.22 |
|  | VKF-DVAE-init | 0.013 | 0.34 | 1.10 | 0.013 | 7.35 | 1.24 |
|  | Deep AR | 0.010 | 3.87 | 1.17 | 0.017 | 4.74 | 1.27 |
|  | MixIT | **0.006** | **10.2** | - | 0.007 | 11.76 | - |
|  | Vanilla NMF | 0.011 | 3.31 | 1.40 | 0.011 | 8.98 | 1.35 |
|  | Temporal NMF | 0.010 | 5.01 | 1.47 | 0.011 | 10.06 | 1.42 |
|  | MixDVAE | 0.007 | 9.06 | **1.64** | **0.007** | **12.92** | **2.06** |

by $2.32$ dB on speech and by $0.71$ dB on CBF, and the performance of Deep AR dropped by $1.45$ dB on speech and by $0.45$ dB on CBF.

Compared to DVAE-init, both MixDVAE and VKF-DVAE-init exhibit better separation performance (at least in terms of SI-SDR for VKF-DVAE-init). This indicates that the multi-source dynamical model with the observation-to-source assignment latent variable plays an important role in separating the content of different audio sources.

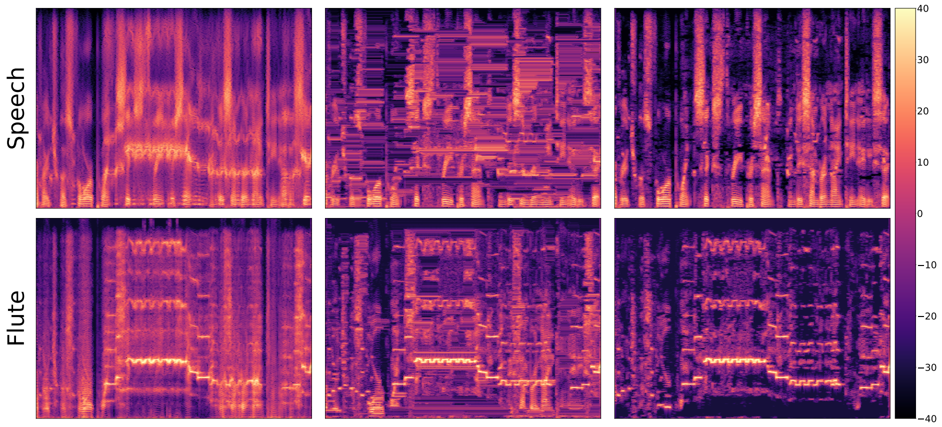
Although MixIT obtains slightly better performance than MixDVAE for speech on the long sequences subsets, its performance on short and medium sequences subsets is quite bad (in terms of SI-SDR, only $3.26$ dB for speech and $7.15$ dB for CBF on the



(a) Mixture

(b) Ground Truth

(c) VKF-Oracle



(d) DVAE-init

(e) VKF-DVAE-init

(f) MixDVAE

**Figure 5.1**: An example of audio source separation result obtained with the proposed MixDVAE algorithm and the baselines (speech and CBF power spectrograms). Best seen in color.

short sequences subset, and $4.75$ dB for speech and $8.74$ dB for CBF on the medium sequences subset). As for the NMF based models, though adding temporal extensions to the vanilla NMF model indeed improves the model performance, the obtained results remain significantly inferior to that obtained by MixDVAE.

### 5.5.2 QUALITATIVE ANALYSIS

To illustrate the behavior of the different models, we selected an audio source separation example and plotted the spectrograms in Fig. 5.1. More examples can be found in Appendix A.3.2. In the given example, the sequence length of the spectrograms is 300 frames. It is obvious that the ground-truth spectrograms of both the speech and CBF have spectral components with non-linear trajectories over time. Even though VKF-Oracle achieves the best performance, we observe in Fig. 5.1(c) that there are several stationary traces (artifactual horizontal spectral lines) in the spectrograms caused by the inappropriate linear dynamics hypothesis. This phenomenon becomes even worse when VKF is initialized with the mask defined by the outputs of the pre-trained DVAEs (VKF-DVAE-init). In Fig. 5.1(d), we clearly see the stationary traces, especially for the separated speech spectrogram. We believe that this is the reason why VKF-DVAE-init showed poor separation performance in general. When looking at the outputs of the pre-trained DVAE models, we find that the pre-trained DVAE models can provide a relatively good initialization for the VEM algorithm, even if we are still far from separated sources. In fact, since in the SC-ASS task, we pre-trained separately two DVAE models on the speech dataset and on the CBF dataset, the pre-trained DVAE models already have some prior information about the single-source dynamics. Though we only give the mixture spectrogram as input, the pre-trained DVAE models can, to some extent, enhance the information of the source used in pre-training and attenuate the information of the other source. However, this kind of filtering is not very efficient. As we can see in the top figure of Fig. 5.1(e), the output spectrogram provided by the DVAE model pre-trained on the speech dataset still keeps a significant amount of information on the CBF. Finally, even if the initialization is not that accurate, we see in Fig. 5.1(f) that MixDVAE achieved a good separation of the two sources after running the VEM iterations.

## 5.6 ABLATION STUDIES

Similar to the MOT task, we conduct ablation studies for the SC-ASS task on the influence of the pre-trained DVAE model quality and the influence of fine-tuning the DVAE.

### 5.6.1 INFLUENCE OF THE PRE-TRAINED DVAE MODEL QUALITY

Similar to the MOT task in Section 4.6.1, we also generated two additional subsets of the training data for both the WSJ0 and CBF datasets, comprising half and a quarter of our original training dataset (used in Section 5.3.1), randomly selected. The two new

**Table 5.2**: Capacity of the SRNN model pre-trained at three data scales of the WSJ0 and the CBF datasets. SRNN-full, SRNN-half, and SRNN-quarter stand for SRNN pre-trained on the totality, half of and quarter of our original training set, respectively.

| Model name | WSJ0 | | CBF | |
|---|---|---|---|---|
| | Training loss | Validation loss | Training loss | Validation loss |
| SRNN-full | 353.89 | 373.61 | 521.76 | 779.69 |
| SRNN-half | 358.13 | 389.58 | 489.53 | 949.11 |
| SRNN-quarter | 361.58 | 383.64 | 646.55 | 1106.27 |

**Table 5.3**: SC-ASS results obtained by MixDVAE with SRNN pre-trained at the three data scales. The results are reported for the short sequence test subset ($T = 50$).

| Model name | Speech | | | Chinese bamboo flute | | |
|---|---|---|---|---|---|---|
| | RMSE ↓ | SI-SDR ↑ | PESQ ↑ | RMSE ↓ | SI-SDR ↑ | PESQ ↑ |
| SRNN-full | **0.006** | 9.23 | 1.73 | **0.007** | **13.50** | **2.30** |
| SRNN-half | **0.006** | **9.66** | **1.82** | 0.009 | 12.29 | 2.28 |
| SRNN-quarter | 0.007 | 8.83 | 1.79 | 0.011 | 10.29 | 2.13 |

subsets of WSJ0 contains $12.45$ and $6.29$ hours of speech recordings respectively. And the two new subsets of CBF contains $1.07$ and $0.55$ hours of CBF recordings respectively. The performance of the SRNN model pre-trained on these different datasets is reported in Table 5.2. For the WSJ0 dataset, we observe that the training and validation losses are relatively close to each other, and both increase when decreasing the training data size, but the increase is moderate. Therefore, the capacity of SRNN drops, but quite slightly. However, for the CBF dataset, the gap between the training and validation losses is higher, and the training loss of SRNN-half decreases compared to SRNN-full while the validation loss increases significantly, increasing the gap. Both the training loss and the validation loss of SRNN-quarter are higher than that of SRNN-full and SRNN-half, and the gap between training and validation is also relatively large. This shows that the size of the (full) CBF dataset may be a bit too limited, and reducing this dataset may harm the generalization capacity of SRNN.

The SC-ASS results obtained by MixDVAE with SRNN pre-trained at the three different data scales are reported in Table 5.3. The experiments are conducted on the short se-

**Table 5.4**: SC-ASS results obtained by MixDVAE with and without the fine-tuning of SRNN. The results are reported for the short ($T = 50$), medium ($T = 100$) and long ($T = 300$) test sequence subsets.

| Dataset | Finetuning | Speech | | | Chinese bamboo flute | | |
|---|---|---|---|---|---|---|---|
| | | RMSE ↓ | SI-SDR ↑ | PESQ ↑ | RMSE ↓ | SI-SDR ↑ | PESQ ↑ |
| Short | Yes | 0.007 | 8.00 | 1.63 | 0.007 | 12.73 | 2.15 |
| | No | **0.006** | **9.23** | **1.73** | **0.007** | **13.50** | **2.30** |
| Medium | Yes | 0.008 | 8.00 | 1.55 | 0.008 | 12.23 | 2.02 |
| | No | **0.007** | **9.32** | **1.65** | **0.007** | **13.05** | **2.16** |
| Long | Yes | 0.008 | 7.02 | 1.49 | 0.008 | 11.40 | 1.88 |
| | No | **0.007** | **9.06** | **1.64** | **0.007** | **12.92** | **2.06** |

quence subset ($T = 50$). We find that, surprisingly, the separation performance of MixDVAE with SRNN-half on the speech signals has been slightly improved over SRNN-full, whereas (much less surprisingly) the performance on the CBF signals has decreased. This may be caused by the lower generalization ability of SRNN-half on the CBF dataset. For SRNN-quarter, the performance of MixDVAE on both the speech and the CBF decrease, but the decrease for the speech is quite moderate ($0.4$ dB SI-SDR w.r.t. SRNN-full; the PESQ value is even slightly better), whereas the CBF is loosing about $3.2$ dB SI-SDR. Again, even if it is difficult to draw a general conclusion from this single experiment, those results seem to indicate a relative robustness of MixDVAE to the limitation of the DVAE training dataset size, provided that the DVAE keeps a sufficient generalization capability.

### 5.6.2 INFLUENCE OF THE DVAE FINE-TUNING

Table 5.4 shows the performance of MixDVAE on the three test subsets with and without fine-tuning SRNN in the E-Z step. Similar to the MOT task, for all three datasets, not fine-tuning SRNN leads to the best overall source separation performance (on all of the evaluation metrics).

## 5.7 DISCUSSION ON THE COMPUTATIONAL COMPLEXITY

In this section, we discuss the computational complexity of the MixDVAE algorithm for both of these two tasks. The proposed method is based on two parts: (i) the pre-training of a DVAE model on a single-source dataset, and (ii) the MixDVAE VEM algorithm for source tracking. The computational cost for the pre-training stage mainly depends on the data type and data size of the single-trajectory dataset. To give a general idea, we measured the average training time required for a single epoch (iteration over the whole training set) on both the synthetic trajectories dataset for MOT and the WSJ0 dataset for SC-ASS. The measurement is conducted on an NVIDIA Quadro RTX 8000, in a machine with an Intel(R) Xeon(R) Gold 5218R CPU @ 2.10GHz. The obtained results for different data scales as mentioned in Section 4.6.1 is reported in Table 5.6. We have observed that doubling the size of the training data results in almost a doubling of the training time. On the other hand, the computation complexity of the MixDVAE algorithm mainly depends on three factors: the number of VEM iterations, the number of sources to track and separate, and the sequence length. Typically, the performance of MixDVAE exhibits an initial rapid increase over the VEM iterations, followed by stabilization towards a plateau. In Fig. 5.2, we plot the evolution of the averaged performance of MixDVAE over the medium sequence test dataset as a function of the number of VEM iteration (the performance is represented by the MOTA score for the MOT task and by the SI-SDR score for the SC-ASS task). We observe that for the MOT task, the performance of MixDVAE has been stabilized from around 10 iterations, whereas for the SC-ASS task, the performance has been stabilized from around 20 iterations. In practice, we run the algorithm for more iterations to guarantee the convergence. Taking computational time optimization into account, it is possible to identify an optimal number of iterations by applying a grid search, for a specific task and dataset. To quantify the computational time of the MixDVAE algorithm, we compute the averaged processing time for one sequence on the MOT task, for the three considered values of the sequence length, and for the case of 3 and 6 sources. This average processing time is measured on an NVIDIA Quadro RTX 4000 GPU, in a machine with an Intel(R) Xeon(R) W-2145 CPU@3.70GHz, and it is averaged on 10 test sequences. The results are reported in Table 5.5. We observe a linear increase of the computation time as a function of the sequence length. As the number of sources to track doubles, the computation time exhibits more than a twofold increase. The com-

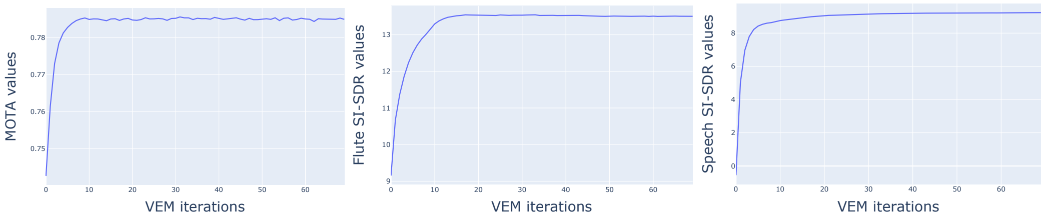


**Figure 5.2**: Evolution of the performance of MixDVAE as a function of the number of VEM iterations (MOTA score for the MOT task and SI-SDR scores for the SC-ASS task).

**Table 5.5**: Averaged processing time per sequence for the MOT task.

| Sequence length (frames) | 60 | | 120 | | 300 | |
|---|---|---|---|---|---|---|
| # sources | 3 | 6 | 3 | 6 | 3 | 6 |
| Computation time per sequence (s) | 23.01 | 57.29 | 45.05 | 110.41 | 112.93 | 272.94 |

**Table 5.6**: Pre-training computational cost on different datasets at different scales.

| Task | Data set | Data scale | One epoch training time (s) |
|---|---|---|---|
| MOT | Synthetic trajectories | Full | 15 |
| | | Half | 7.8 |
| | | Quarter | 4.8 |
| SC-ASS | WSJ0 | Fall | 190.8 |
| | | Half | 121.2 |
| | | Quarter | 63 |

putation complexity can be a bottleneck for the MixDVAE especially for long sequences with a large number of sources. However, further algorithm and code optimization might be possible, since we did not focus on this aspect of the problem so far.

## 5.8 CONCLUSION

In this section, we make a general conclusion of Chapter 3, Chapter 4, and Chapter 5. In Chapter 3, we introduce MixDVAE, an DPGM designed to model the dynamics of multiple, jointly observed, sources. MixDVAE involves two main modules: A DVAE model for capturing the dynamics of each individual source and a discrete latent assignment variable that assign observations to sources, thus enabling us to form complete trajectories. The model learning process consists of two stages. During the first stage, the same or different DVAE model(s) is/are pre-trained on the synthetic or natural single-source trajectory dataset(s) to obtain prior information about the sources dynamics. During the second stage, the pre-trained DVAE model(s) is/are integrated into the general MixDVAE model. The entire MixDVAE model is solved using the VI framework with a VEM algorithm that combines the structured mean-field approximation and the amortized inference principles. The VEM algorithm is run directly on each multi-source test data sequence to process and the entire method does not require massive multi-source annotated datasets for training, which are difficult to obtain, especially for natural data. Hence, we consider it as weakly-supervised, as opposed to the fully-supervised approaches most commonly used in many multi-source processing applications. We illustrate the versatility of MixDVAE by applying it to two distinct scenarios: the MOT task (in Chapter 4) and the SC-ASS task (in Chapter 5). Experimental results demonstrate that MixDVAE performs well on both tasks. Specifically, thanks to the strong dynamical modeling capacity of DVAE, MixDVAE shows to be more efficient than the combination of a linear dynamical model with the assignment variable. In addition, MixDVAE can generate reasonable predictions of the source vector even in the absence of observations, resulting in smooth and robust trajectories, as demonstrated in the MOT task. Our experiments demonstrate the generalization capability of MixDVAE trained on a synthetic single-target dataset, and evaluated in a multiple-target dataset. Finally, we believe that MixDVAE has a very strong potential for modeling the dynamics of multiple-source systems in general, and can be applied to various other tasks. However, we acknowledge that MixDVAE also has certain limitations, such as the assumption that each source behaves independently and the lack of consideration for interactions among them. We leave this as a challenging topic for future research.

CHAPTER 6

# UNSUPERVISED SPEECH ENHANCEMENT WITH DEEP DYNAMICAL PROBABILISTIC GENERATIVE MODELS

*Progress imposes not only new possibilities*
*for the future but new restrictions.*

*— Norbert Wiener*

This chapter is based on the following publication:

Xiaoyu Lin, Simon Leglaive, Laurent Girin, and Xavier Alameda-Pineda. "Unsupervised speech enhancement with deep dynamical generative speech and noise models." In Proceedings Interspeech Conference, pages 5102-5106, 2023.

## 6.1 INTRODUCTION

In Section 2.5.3, we provided a brief overview of the background in speech enhancement (SE) research. In this chapter, we will concentrate on the unsupervised SE methods, delving deeper into the challenges in this field and presenting a novel unsupervised SE method.

In the first place, we clarify several terminologies adopted in this dissertation so that we can make our discussions more efficient. As introduced in Section 2.5.2 and Section 2.5.3, conventional *supervised* SE methods typically rely on a substantial corpus of paired clean-noisy speech data for training. These datasets are commonly synthetically generated by summing up pre-recorded clean speech signals with noise signals. However, as previously highlighted, this training paradigm poses challenges for supervised SE methods to effectively generalize to real-world scenarios. In our research context, we define the *unsupervised* SE methods as approaches that abstain from using aligned parallel clean-noisy speech data during training. This definition may be arguable, as the methods presented in Section 2.5.3, which rely solely on clean speech data or utilize unpaired clean-noisy speech data, still necessitate a clean speech dataset for model training. Acquiring a clean speech dataset, which needs to be recorded in an anechoic chamber, generally requires more efforts than obtaining the noisy speech dataset. Nevertheless, the underlying argument, akin to the one we articulated in Section 3.1 of Chapter 3, is that we aim to utilize a modest amount (considerably smaller than what supervised SE methods necessitate) of clean speech dataset for training, while still preserving the robust generalization ability of the model. In several literature sources [182, 142], this kind of methods are referred to as *semi-supervised*. Nonetheless, we find this term also inappropriate as, in machine learning, *semi-supervised* learning typically involves using a small amount of labeled data (paired clean-noisy speech in the case of SE) alongside a large amount of unlabeled data during training [241]. Another concept worth discussing is that of *weakly supervised* learning methods. Weakly supervised learning is a broader umbrella that mainly encompass three learning configurations with weak supervision: incomplete supervision, where only a subset of training data is labeled (including the semi-supervised learning configuration); inexact supervision, where only coarse-grained labels are provided; and inaccurate supervision, where the provided labels may not always correspond to the ground-truth [238]. For SE methods that solely rely on clean speech dataset or unpaired clean-noisy speech dataset for training, we find them on the borderline of being characterized into the second category of weakly supervised learning methods. After careful consideration, we opt to categorize all methods that do not use paired clean-noisy speech data for training as unsupervised methods.

Additionally, as we are interested in the generalization ability of the proposed methods on test data encompassing noise types and acoustic conditions not encountered during training, we further divide the unsupervised SE methods into two categories, following the terminology defined in [22]: the unsupervised *noise-dependent* (ND) methods and the unsupervised *noise-agnostic* (NA) methods. ND methods incorporate noise or noisy samples in their training process, enabling the model to acquire noise information about different noise types before testing. In contrast, NA methods exclusively utilize clean speech signals for training, relying on estimating the noise characteristics at test time for each noisy speech sequence to be processed. A typical unsupervised NA approach that has been discussed in 2.5.3 uses a pre-trained VAE as a prior distribution of the clean speech signal and a NMF model for the noise variance [9, 117, 151, 176]. The NMF parameters and the VAE latent vector are estimated at test time from the noisy signal and combined to build a denoising Wiener filter. Further developments in this general line were proposed in, e.g., [119, 31, 118, 224].

Recently, it was proposed in [22] to replace the VAE by a DVAE [71, 20], yielding better clean speech modeling capacities by considering the temporal dependencies across successive spectrogram frames. The algorithm proposed in [22] was shown to achieve very competitive performance even when compared to supervised approaches. However, this algorithm has two main drawbacks. First, the NMF may be a too simple model for many real-world noise signals, which are poorly described in the spectrogram domain as a non-negative linear combination of a few spectral templates. Second, at test time, the inference algorithm, which needs to execute the VEM algorithm for several iterations on each noisy sequence independently, is highly computation- and time-consuming.

In this part of our PhD work, we aim at both increasing the modeling power of the noise model and accelerating the inference process. To achieve this, we build on [22] and propose to replace the NMF noise model with a dynamical DPGM. We have seen in Chapter 2 that the dynamical DPGMs represent a versatile class of dynamical models for sequential data generation by leveraging DNNs.[1] This proposition is motivated by two key factors. First, DNNs with purposefully designed architectures are supposed to offer greater expressiveness compared to the NMF model, enabling improved estimation for complex noise types, such as those coming from cars, music, baby crying, etc. Second, employing DNNs as the noise model facilitates both training the noise model and fine-tuning the clean speech model efficiently on a large dataset using the SGD method. As we will see, during test time, the inference can be streamlined through a simple forward pass, bypassing the need for a time-consuming sequence-by-sequence EM algorithm estimation, as was the case in [22].

Besides the fact that the proposed method can be seen as an extension of [22], it is worth noting that it has some connections with MixDVAE, as applied to the SC-ASS problem in Chapter 5. The main differences lie in the fact that, in this new SE model, we only pre-train a single DVAE model on a clean speech dataset, as opposed to the MixDVAE model, which would pre-train two instances of the DVAE model (or two different DVAE models) — one on the clean dataset and another on the noise dataset. As we will see, the parameters of the noise model are directly learned from the noisy speech dataset. Moreover, this method does not rely on a discrete assignment variable to separate the

---

[1] Note that the DVAE used for clean speech modeling also falls within the dynamical DPGM family, hence the choice of the chapter title.

clean speech signal from the noise. Instead, the clean speech signal estimate is obtained by applying a Wiener filter, build from the DPGM outputs, on the noisy speech signal.

We implement and test the dynamical DPGM noise model with different inputs: the DVAE latent variable (LV), or the noisy observations (NO), or both (NOLV). Different model variants employ different architectures; the implementation details will be specified in 6.3.2. Moreover, these three variants are implemented and tested in both ND (using a large noisy speech dataset) or NA configurations. Even further, the models trained in ND configuration can then be fine-tuned on each noisy speech test sequence to process, in order to get adapted to specific noise types in the test set (i.e., ND followed by noise adaptation). Experimental results show that the proposed method obtains performance that is comparable to that of [22], while in the ND configuration, it requires much less computation time during inference.

## 6.2 DYNAMICAL DPGM-BASED SE METHOD

In this section, we present our unsupervised SE method using dynamical DPGM-based speech and noise models.

### 6.2.1 CLEAN SPEECH MODELING WITH AN RVAE

Similarly to the MixDVAE applied to the SC-ASS problem in Chapter 5, and similarly to a vast amount of speech enhancement and separation methods, we work in the STFT domain. Let $\mathbf{s}_{1:T} = \{\mathbf{s}_t\}_{t=1}^T \in \mathbb{C}^{F\times T}$ denote the STFT spectrogram of the clean speech. Each vector $\mathbf{s}_t = \{s_{t,f}\}_{f=1}^F \in \mathbb{C}^F$ is the short-time spectrum at time frame $t$, and $f$ denotes the frequency bin. Let $\mathbf{z}_{1:T} \in \mathbb{R}^{L\times T}$ denote the associated latent vector sequence, with latent dimension $L \ll F$. As reported in [22], among several tested DVAE models, the Recurrent Variational AutoEncoder (RVAE) model [116] worked best on the SE task. So, we also use this model in this work. The RVAE generative model is defined as:

$$p_{\theta_{\mathbf{s}}}(\mathbf{s}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_{\theta_{\mathbf{s}}}(\mathbf{s}_t|\mathbf{z}_{1:t})p(\mathbf{z}_t). \tag{6.1}$$

For each time frame $t$, and conditionally to $\mathbf{z}_{1:t}$, $\mathbf{s}_t$ is assumed to follow a circularly-symmetric zero-mean complex Gaussian distribution [62, 124], same as discussed in Chapter 5, Section 5.2:

$$p_{\theta_\mathbf{s}}(\mathbf{s}_t|\mathbf{z}_{1:t}) = \mathcal{N}_c\big(\mathbf{s}_t; \mathbf{0}, \mathrm{diag}(\mathbf{v}_{\theta_\mathbf{s},t})\big). \tag{6.2}$$

In this work, all covariance matrices are assumed to be diagonal and are represented by the vector of diagonal entries. Here, $\mathbf{v}_{\theta_\mathbf{s},t} \in \mathbb{R}_+^F$ is a function of $\mathbf{z}_{1:t}$ and is modeled with the RVAE decoder (i.e., it is generated with the RVAE decoder DNN, which is an RNN parametrized by $\theta_\mathbf{s}$). The latent vector $\mathbf{z}_t$ is assumed to follow a standard Gaussian prior distribution:

$$p(\mathbf{z}_t) = \mathcal{N}(\mathbf{z}_t; \mathbf{0}, \mathbf{I}). \tag{6.3}$$

The inference model, i.e. the approximated posterior distribution of RVAE, is defined as [116]:

$$q_{\phi_\mathbf{z}}(\mathbf{z}_{1:T}|\mathbf{s}_{1:T}) = \prod_{t=1}^{T} q_{\phi_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{s}_{t:T}), \tag{6.4}$$

with $\mathbf{z}_t$ assumed to follow a (real-valued) Gaussian distribution:

$$q_{\phi_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{s}_{t:T}) = \mathcal{N}\big(\mathbf{z}_t; \boldsymbol{\mu}_{\phi_\mathbf{z},t}, \mathrm{diag}(\mathbf{v}_{\phi_\mathbf{z},t})\big), \tag{6.5}$$

where $\boldsymbol{\mu}_{\phi_\mathbf{z},t} \in \mathbb{R}^L$ and $\mathbf{v}_{\phi_\mathbf{z},t} \in \mathbb{R}_+^L$ are both a function of $\mathbf{z}_{1:t-1}$ and $\mathbf{s}_{t:T}$, which is modeled with the RVAE encoder (i.e., it is generated with the RVAE encoder DNN, which is an RNN parametrized by $\phi_\mathbf{z}$; note that in practice, the squared modulus of the $\mathbf{s}_{t:T}$ entries are send to the encoder input instead of the complex-valued STFT coefficients, as discussed in Chapter 5, Section 5.2).

The RVAE model is pre-trained on a clean speech dataset by maximizing the following ELBO [116]:

$$\begin{aligned}\mathcal{L}(\theta_\mathbf{s}, \phi_\mathbf{z}; \mathbf{s}_{1:T}) = -\sum_{t=1}^{T} \mathbb{E}_{q_{\phi_\mathbf{z}}}\Big[& d_{\mathrm{IS}}(|\mathbf{s}_t|^2, \mathbf{v}_{\theta_\mathbf{s},t}) \\ &+ D_{\mathrm{KL}}\big(q_{\phi_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{s}_{t:T})||p(\mathbf{z}_t)\big)\Big],\end{aligned} \tag{6.6}$$

where modulus and exponentiation are element-wise, $d_{\text{IS}}(\cdot, \cdot)$ is the IS divergence [62] and $D_{\text{KL}}(\cdot||\cdot)$ is the KL divergence, which were both already defined in the preceding chapters.

### 6.2.2 DYNAMICAL DPGM-BASED NOISE MODEL

Let $\mathbf{x}_{1:T} = \{\mathbf{x}_t\}_{t=1}^T \in \mathbb{C}^{F\times T}$ and $\mathbf{n}_{1:T} = \{\mathbf{n}_t\}_{t=1}^T \in \mathbb{C}^{F\times T}$ denote respectively the complex-valued STFT spectrogram of the noisy speech and the noise, which is assumed additive:

$$\mathbf{x}_{1:T} = \mathbf{s}_{1:T} + \mathbf{n}_{1:T}. \tag{6.7}$$

At each time frame $t$, $\mathbf{n}_t$ is assumed to follow a circularly-symmetric zero-mean complex Gaussian distribution:

$$p_{\theta_{\mathbf{n}}}(\mathbf{n}_t) = \mathcal{N}_c\big(\mathbf{n}_t; \mathbf{0}, \text{diag}(\mathbf{v}_{\theta_{\mathbf{n}},t})\big), \tag{6.8}$$

where $\mathbf{v}_{\theta_{\mathbf{n}},t} \in \mathbb{R}_+^F$ is the noise variance vector. In several previous works, $\mathbf{v}_{\theta_{\mathbf{n}},t}$ was modeled with NMF, i.e. factorized into the product of two low-rank non-negative matrices. In this work, we model $\mathbf{v}_{\theta_{\mathbf{n}},t}$ with a dynamical DPGM. We propose three different noise model dependencies: (i) **DVAE latent variable (LV)**, in which $\mathbf{v}_{\theta_{\mathbf{n}},t}$ is a function of the whole sequence of the DVAE latent vectors, i.e. $\mathbf{v}_{\theta_{\mathbf{n}},t} = \mathbf{v}_{\theta_{\mathbf{n}},t}(\mathbf{z}_{1:T})$. In other words, in this version, the latent variables are shared between the clean speech and the noise models; (ii) **noisy observations (NO)**, in which $\mathbf{v}_{\theta_{\mathbf{n}},t}$ is a function of all the past values of the noisy speech, i.e. $\mathbf{v}_{\theta_{\mathbf{n}},t} = \mathbf{v}_{\theta_{\mathbf{n}},t}(\mathbf{x}_{1:t-1})$; and (iii) **both noisy observations and DVAE latent variables (NOLV)**, in which $\mathbf{v}_{\theta_{\mathbf{n}},t}$ is a function of all the past values of the noisy speech as well as the past and present values of the DVAE latent vectors, i.e. $\mathbf{v}_{\theta_{\mathbf{n}},t} = \mathbf{v}_{\theta_{\mathbf{n}},t}(\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t})$. For clarity of presentation, let $\mathbf{p}_t$ denote the input of the noise model, i.e. $\mathbf{p}_t = \mathbf{z}_{1:T}$ in LV, $\mathbf{p}_t = \mathbf{x}_{1:t-1}$ in NOLV, and $\mathbf{p}_t = \{\mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}\}$ in NOLV. For all model dependencies (NOLV, NOLV, or NOLV), the noise variance $\mathbf{v}_{\theta_{\mathbf{n}},t}$ is a function of $\mathbf{p}_t$ that is implemented by a DNN parametrized by $\theta_{\mathbf{n}}$.

Applying the chain rule and taking into account the conditional dependencies, the generative model over the set of variables $\{\mathbf{x}_{1:T}, \mathbf{s}_{1:T}, \mathbf{z}_{1:T}\}$ is given by:

$$p_\theta(\mathbf{x}_{1:T}, \mathbf{s}_{1:T}, \mathbf{z}_{1:T}) = \prod_{t=1}^{T} p_{\theta_\mathbf{n}}(\mathbf{x}_t|\mathbf{s}_t, \mathbf{p}_t) p_{\theta_\mathbf{s}}(\mathbf{s}_t|\mathbf{z}_{1:t}) p(\mathbf{z}_t), \tag{6.9}$$

where

$$p_{\theta_\mathbf{n}}(\mathbf{x}_t|\mathbf{s}_t, \mathbf{p}_t) = \mathcal{N}_c\big(\mathbf{x}_t; \mathbf{s}_t, \mathrm{diag}(\mathbf{v}_{\theta_\mathbf{n},t}(\mathbf{p}_t))\big) \tag{6.10}$$

is deduced from Equation 6.7 and Equation 6.8, $p_{\theta_\mathbf{s}}(\mathbf{s}_t|\mathbf{z}_{1:t})$ and $p(\mathbf{z}_t)$ are defined in Equation 6.2 and Equation 6.3, and $\theta = \theta_\mathbf{s} \cup \theta_\mathbf{n}$.

### 6.2.3 SPEECH ENHANCEMENT WITH THE INFERENCE MODEL

Applying the D-separation principles [24], the posterior distribution corresponding to the generative model Equation 6.9 factorizes as follows:

$$p_\theta(\mathbf{s}_{1:T}, \mathbf{z}_{1:T}|\mathbf{x}_{1:T}) = \prod_{t=1}^{T} p_\theta(\mathbf{s}_t|\mathbf{z}_{1:t}, \mathbf{x}_t, \mathbf{p}_t) p_\theta(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}). \tag{6.11}$$

For each time frame $t$, $p_\theta(\mathbf{s}_t|\mathbf{z}_{1:t}, \mathbf{x}_t, \mathbf{p}_t)$ can be computed in closed form as a complex Gaussian distribution $p_\theta(\mathbf{s}_t|\mathbf{z}_{1:t}, \mathbf{x}_t, \mathbf{p}_t) = \mathcal{N}_c(\mathbf{s}_t; \boldsymbol{\mu}_{\theta,t}, \mathrm{diag}(\mathbf{v}_{\theta,t}))$ (see Appendix A.4.1 for more details), with

$$\boldsymbol{\mu}_{\theta,t} = \frac{\mathbf{v}_{\theta_\mathbf{s},t}(\mathbf{z}_{1:t})}{\mathbf{v}_{\theta_\mathbf{s},t}(\mathbf{z}_{1:t}) + \mathbf{v}_{\theta_\mathbf{n},t}(\mathbf{p}_t)} \mathbf{x}_t, \tag{6.12}$$

$$\mathbf{v}_{\theta,t} = \frac{\mathbf{v}_{\theta_\mathbf{s},t}(\mathbf{z}_{1:t}) \mathbf{v}_{\theta_\mathbf{n},t}(\mathbf{p}_t)}{\mathbf{v}_{\theta_\mathbf{s},t}(\mathbf{z}_{1:t}) + \mathbf{v}_{\theta_\mathbf{n},t}(\mathbf{p}_t)}, \tag{6.13}$$

where vector multiplication and division are element-wise. Equation 6.12 provides the clean speech signal linear minimum mean-square-error (LMMSE) estimate $\hat{\mathbf{s}}_t = \boldsymbol{\mu}_{\theta,t}$, which corresponds to the Wiener filter output (and the corresponding time-domain waveform is obtained using the inverse STFT).

The distribution $p_\theta(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T})$ is intractable and cannot be used directly to recursively provide the $\mathbf{z}_{1:T}$ estimate. We thus approximate it with the RVAE inference model

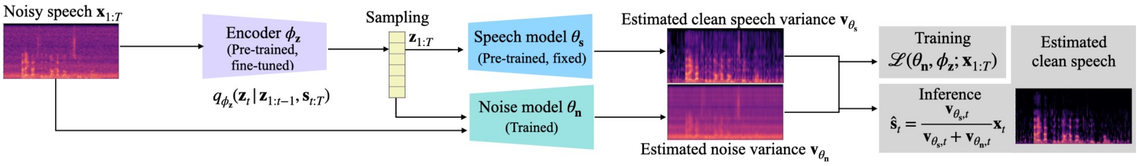

Noisy speech $\mathbf{x}_{1:T}$
Encoder $\phi_\mathbf{z}$
(Pre-trained, fine-tuned)
$q_{\phi_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{s}_{t:T})$
Sampling
$\mathbf{z}_{1:T}$
Speech model $\theta_\mathbf{s}$
(Pre-trained, fixed)
Noise model $\theta_\mathbf{n}$
(Trained)
Estimated clean speech variance $\mathbf{v}_{\theta_\mathbf{s}}$
Estimated noise variance $\mathbf{v}_{\theta_\mathbf{n}}$
Training
$\mathscr{L}(\theta_\mathbf{n}, \phi_\mathbf{z}; \mathbf{x}_{1:T})$
Inference
$\hat{\mathbf{s}}_t = \frac{\mathbf{v}_{\theta_\mathbf{s},t}}{\mathbf{v}_{\theta_\mathbf{s},t} + \mathbf{v}_{\theta_\mathbf{n},t}} \mathbf{x}_t$
Estimated clean speech

**Figure 6.1**: Schematic view of the proposed SE method. The training and test configurations correspond to the two gray boxes depicted on the right side of the figure.

(defined in Section 6.2.1):

$$p_\theta(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}) \approx q_{\phi_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{t:T}). \tag{6.14}$$

Here, the RVAE encoder, pre-trained on a clean speech signal dataset, takes as input the noisy speech signal, and must thus be adapted to such kind of input (see the next subsection). In the following, we inject Equation 6.14 into Equation 6.11, and the resulting approximate joint posterior is denoted by $p_{\theta,\phi_\mathbf{z}}(\mathbf{s}_{1:T}, \mathbf{z}_{1:T}|\mathbf{x}_{1:T})$.

### 6.2.4 Model Optimization

We recall that the parameters $\{\theta_\mathbf{s}, \phi_\mathbf{z}\}$ are learned by pre-training the RVAE on a clean speech dataset. $\theta_\mathbf{s}$ is then fixed during the SE stage, whereas $\phi_\mathbf{z}$ has to be fine-tuned on the noisy signal(s), and we also have to estimate the noise model parameters $\theta_\mathbf{n}$. Following the principles of variational inference presented in Section 2.3, the parameters are optimized by maximizing the ELBO, which is here defined as:

$$\mathcal{L}(\theta_\mathbf{n}, \phi_\mathbf{z}; \mathbf{x}_{1:T}) = \mathbb{E}_{p_{\theta,\phi_\mathbf{z}}}\left[\log \frac{p_\theta(\mathbf{x}_{1:T}, \mathbf{s}_{1:T}, \mathbf{z}_{1:T})}{p_{\theta,\phi_\mathbf{z}}(\mathbf{s}_{1:T}, \mathbf{z}_{1:T}|\mathbf{x}_{1:T})}\right]. \tag{6.15}$$

Given the factorizations Equation 6.9 and Equation 6.11, and the fact that all involved distributions are Gaussian, Equation 6.15 can be developed as:

$$\begin{aligned}\mathcal{L}(\theta_\mathbf{n}, \phi_\mathbf{z}; \mathbf{x}_{1:T}) = -\sum_{t=1}^{T} \mathbb{E}_{q_{\phi_\mathbf{z}}}\Big[& d_{\text{IS}}(|\mathbf{x}_t|^2, \mathbf{v}_{\theta_\mathbf{s},t} + \mathbf{v}_{\theta_\mathbf{n},t}) \\ &+ D_{\text{KL}}\big(q_{\phi_\mathbf{z}}(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{x}_{t:T})||p(\mathbf{z}_t)\big)\Big].\end{aligned} \tag{6.16}$$

As mentioned before, the model can be trained in either NA or ND configuration.

When trained in NA configuration, the parameters $\{\theta_{\mathbf{n}}, \phi_{\mathbf{z}}\}$ are estimated directly from the test noisy speech sequence to be enhanced. This is done by optimizing the ELBO Equation 6.16 independently on each single noisy speech sequence for a certain number of iterations. Afterwards, the clean speech estimate is computed with Equation 6.12, using the optimal parameters and latent vectors sampled from the encoder. This configuration allows the model to adapt to the specific noise patterns of each test sequence, without the need for any prior knowledge or training data on the noise type. This makes it suitable for scenarios where the noise type is unknown. When trained in ND configuration, the model parameters are estimated by optimizing the ELBO Equation 6.16 on a large noisy speech training set using SGD optimization (we recall that no parallel noisy-clean data is thus used). Then at test time, the clean speech is computed using Equation 6.12 with a single forward pass of the model on the noisy test sequence. This results in a much more time-efficient inference than methods based on an NMF noise model, while still achieving competitive performance. A schematic view of the proposed method is shown in Fig. 6.1.

## 6.3 EXPERIMENTAL SETTINGS

### 6.3.1 DATASETS AND PRE-PROCESSING

We used two datasets to evaluate the proposed method: the WSJ0-QUT dataset introduced in [116] and reused in [22], and the publicly available VoiceBank-DEMAND (VB-DMD) dataset [201]. WSJ0-QUT is obtained by mixing clean signals from the Wall Street Journal (WSJ0) dataset [63] with various types of noise signals from the QUT-NOISE dataset [41] with three different signal-to-noise ratio (SNR) values: $-5$, 0 and 5 dB. It contains 12,765 utterances from 101 speakers, 1,026 utterances from 10 speakers and 651 utterances from 8 speakers for model training, validation and test, respectively. VB-DMD is obtained by mixing clean signals from the VoiceBank (VB) corpus [208] with ten types of noise from the DEMAND noise dataset [197]. Following [59], we used 10,802 utterances from 26 speakers for training, 770 utterances from 2 other speakers for validation, and 824 utterances from 2 other speakers for test. The SNR values used for the training set are 15, 10, 5 and 0 dB, while the SNR values used for the test set are 17.5, 12.5, 7.5, and

2.5 dB. For each dataset, we first pre-trained the RVAE model on the clean speech dataset, i.e. WSJ0 or VB; then we estimated the noise model parameters using the noisy speech data, either in the NA or ND configuration (see Section 6.2.4 ).

Before being input into the neural networks, the audio signals are pre-processed as follows. We compute the STFT with a 64-ms sine window (1,024 samples) and a 75%-overlap (256-sample shift), resulting in a sequence of 513-dimensional discrete Fourier coefficients (for positive frequencies). The squared modulus of the STFT coefficients is computed afterwards. For the RVAE pre-training and the SE model trained in ND configuration, we first use a voice activity detection threshold of 30 dB to remove silence portions at the beginning and the end of the signals, and rescale the waveforms in $[-1, 1]$ before computing the STFT coefficients. And we also split the training utterances into smaller sequences of length $T = 100$ frames. At test time, the model is evaluated on the complete noisy test utterances, which can be of variable length. The SE model in NA configuration is trained and evaluated directly on each single complete noisy test utterances.

### 6.3.2 IMPLEMENTATION DETAILS AND TRAINING SETTINGS

The RVAE architecture closely follows the one used in [22], with the exception of replacing the bidirectional LSTM (BLSTM) layers in both the encoder and decoder with standard LSTM layers, since we use here the causal version of RVAE [116]. The latent vector dimension was set to $L = 16$.

The NO noise model is implemented using an LSTM layer that takes as input at time $t$ the past noisy speech vectors $\mathbf{x}_{1:t-1}$, followed by a MLP layer with a *tanh* activation function, except for the output layer, which is linear, and which provides the noise log-variance vector $\log \mathbf{v}_{\theta_{\mathbf{n}},t}(\mathbf{p}_t)$. The architecture of the NOLV and LV noise model are similar to that of the NO noise model, except that the NOLV model uses two LSTM layers, one to encode information from the past noisy speech vectors $\mathbf{x}_{1:t-1}$ and another one to process the past and present latent vectors $\mathbf{z}_{1:t}$, and the LV noise model uses a single LSTM layer to encode information from the complete latent vector sequence $\mathbf{z}_{1:T}$.

For all training processes, we used the Adam optimizer [104] with parameters $\beta_1 = 0.9, \beta_2 = 0.99, \epsilon = 10^{-9}$. For RVAE pre-training and ND training configuration, we decayed the learning rate (from $5 \times 10^{-4}$ to $10^{-8}$) with a cosine annealing scheduler [126]. The models are trained in maximum 500 epochs and the validation set is used to select the best models. During the RVAE pre-training, we applied linear warm-up to the KL term in Equation 6.6 during the first 20 epochs [184].

### 6.3.3 BASELINES AND EVALUATION METRICS

We compare our method with both supervised and unsupervised SE baselines. For supervised baselines, we considered Open-Unmix (UMX) [189] and MetricGAN+ [58], which are LSTM-based methods, and CDiffuSE [127] and SGMSE+ [170], which are diffusion-based methods. For unsupervised baselines, we compared to MetricGAN-U [59], NyTT [61], and RVAE-VEM [116, 22] (note that RVAE-VEM is the combination of clean speech RVAE and noise NMF models optimized with a VEM algorithm that was the starting point of the present work).

As for the SE performance metrics, we used the scale-invariant scale-invariant signal-to-distortion ratio (SI-SDR) [175] (in dB), the perceptual evaluation of speech quality (PESQ) score [172] (in $[-0.5, 4.5]$), and the extended short-time objective intelligibility (ESTOI) score [195] (in $[0, 1]$). We also evaluated the computational efficiency of the inference (denoising algorithm) for RVAE-VEM, SGMSE+ and the proposed method (in different configurations) using the average real-time factor (RTF), which is the time required to process 1 second of audio.[2]

## 6.4 EXPERIMENTAL RESULTS

### 6.4.1 COMPARISON OF DIFFERENT VARIANTS UNDER DIFFERENT CONFIGURATIONS

The SE results for different variants of the dynamical DPGM model are reported in Table 6.1. We observe that on both of these two datasets, models trained under the U-NA

---

[2] All of the RTF values are computed on NVIDIA Quadro RTX 4000 GPU, in a machine with an Intel(R) Xeon(R) W-2145 CPU @ 3.70GHz and averaged on 10 sequences.

**Table 6.1**: SE results for different variants of the dynamical DPGM model. U-NA stands for unsupervised noise-agnostic and U-ND stands for unsupervised noise-dependent, U-NDA stands for U-ND training followed by noise adaptation fine-tuning. The best scores are in bold and the second best scores are underlined.

| Pre-train set | Train set | Test set | Model | Configuration | SI-SDR ↑ | $\text{PESQ}_{\text{MOS}}$ ↑ | $\text{PESQ}_{\text{WB}}$ ↑ | $\text{PESQ}_{\text{NB}}$ ↑ | ESTOI ↑ |
|---|---|---|---|---|---|---|---|---|---|
| - | - | WSJ0-QUT test | Noisy mix. | - | -2.6 | 1.83 | 1.14 | 1.57 | 0.50 |
| WSJ0 train (clean) | WSJ0-QUT train (noisy) | | RVAE-LV | U-NA | 5.4 | 2.31 | 1.53 | 2.01 | **0.65** |
| | | | | U-ND | 5.3 | 2.25 | 1.53 | 1.95 | 0.60 |
| | | | | U-NDA | **6.2** | **2.38** | **1.56** | **2.07** | 0.62 |
| | | | RVAE-NO | U-NA | 6.0 | 2.33 | **1.56** | 2.04 | **0.65** |
| | | | | U-ND | 3.7 | 2.11 | 1.37 | 1.81 | 0.58 |
| | | | | U-NDA | 5.8 | 2.31 | 1.54 | 2.02 | 0.63 |
| | | | RVAE-NOLV | U-NA | 5.5 | 2.31 | 1.53 | 2.01 | **0.65** |
| | | | | U-ND | 4.9 | 2.11 | 1.42 | 1.83 | 0.60 |
| | | | | U-NDA | **6.2** | 2.29 | **1.56** | 2.00 | 0.62 |
| - | - | VB-DMD test | Noisy mix. | - | 8.4 | 3.02 | 1.97 | 2.88 | 0.79 |
| VB train (clean) | VB-DMD train (noisy) | | RVAE-LV | U-NA | 17.5 | 3.23 | 2.39 | 3.15 | **0.82** |
| | | | | U-ND | 17.4 | 3.24 | **2.40** | 3.17 | 0.81 |
| | | | | U-NDA | **17.8** | 3.22 | 2.38 | 3.14 | 0.81 |
| | | | RVAE-NO | U-NA | 17.3 | **3.25** | **2.40** | **3.18** | **0.82** |
| | | | | U-ND | 16.7 | 3.03 | 2.12 | 2.89 | 0.79 |
| | | | | U-NDA | 17.2 | 3.06 | 2.18 | 2.93 | 0.80 |
| | | | RVAE-NOLV | U-NA | 17.5 | **3.25** | **2.40** | **3.18** | **0.82** |
| | | | | U-ND | 16.9 | 3.04 | 2.14 | 2.90 | 0.79 |
| | | | | U-NDA | 17.4 | 3.17 | 2.30 | 3.07 | 0.81 |

configuration achieve good performance. This shows the ability of the proposed model to adapt to the noise characteristics in the NA configuration. More specifically, in the NA configuration, the RVAE-NO model performs slightly better than the other two models on the WSJ0-QUT dataset while the three noise model variants (NOLV, NOLV, NOLV) lead to very similar performance on the VB-DMD dataset. However, the RVAE-NO model and RVAE-NOLV model have a slight drop of performance when trained in U-ND configuration. This may be due to the mismatch between train and test data in this U-ND configuration (different types of noise being used for training and test). In contrast, the RVAE-LV model reveals very robust when used in the U-ND configuration. This may be because estimating the noise variance only from the latent vectors, without using the noisy speech vectors, helps to alleviate the training/test data mismatch issue. It also shows that by conducting noise adaptation fine-tuning after the U-ND training configuration, the performance of all the three model variants was greatly improved (over the U-ND configuration) on both of these two datasets. And on the WSJ0-QUT dataset, models trained under the U-NDA configuration achieve similar or even better (for the NOLV and NOLV models) performance than those directly trained under the U-NA configuration.

**Table 6.2**: SE results compared with baselines. S stands for supervised, U-NA stands for unsupervised noise-agnostic and U-ND stands for unsupervised noise-dependent. The best scores are in bold and the second best scores are underlined.

| Pre-train set | Train set | Test set | Model | Configuration | SI-SDR ↑ | PESQ$_{\text{MOS}}$ ↑ | PESQ$_{\text{WB}}$ ↑ | PESQ$_{\text{NB}}$ ↑ | ESTOI ↑ |
|---|---|---|---|---|---|---|---|---|---|
| - | - | WSJ0-QUT test | Noisy mix. | - | -2.6 | 1.83 | 1.14 | 1.57 | 0.50 |
| - | - | WSJ0-QUT test | Wiener filter (Oracle) | - | 12.1 | 3.14 | 2.56 | 3.03 | 0.88 |
| - | WSJ0-QUT train | WSJ0-QUT test | UMX | S | 5.7 | 2.16 | 1.38 | 1.83 | **0.63** |
| - | WSJ0-QUT train | WSJ0-QUT test | MetricGAN+ | S | 3.6 | **2.83** | **2.18** | **2.61** | 0.60 |
| WSJ0 train | - | WSJ0-QUT test | RVAE-VEM | U-NA | **5.8** | 2.27 | 1.54 | 1.98 | 0.62 |
| WSJ0 train | WSJ0-QUT train | WSJ0-QUT test | RVAE-LV | U-ND | 5.3 | 2.25 | 1.53 | 1.95 | 0.60 |
| - | - | VB-DMD test | Noisy mix. | - | 8.4 | 3.02 | 1.97 | 2.88 | 0.79 |
| - | - | VB-DMD test | Wiener filter (Oracle) | - | 21.5 | 3.70 | 3.39 | 3.79 | 0.93 |
| - | VB-DMD train | VB-DMD test | UMX | S | 14.0 | 3.18 | 2.35 | 3.08 | 0.83 |
| - | VB-DMD train | VB-DMD test | MetricGAN+ | S | 8.5 | **3.59** | **3.13** | **3.63** | 0.83 |
| - | VB-DMD train | VB-DMD test | CDiffuSE | S | 12.6 | - | 2.46 | - | 0.79 |
| - | VB-DMD train | VB-DMD test | SGMSE+ | S | 17.3 | - | 2.93 | - | **0.87** |
| - | VB-DMD train + Extra noise | VB-DMD test | NyTT Xtra | U-ND | **17.7** | - | 2.30 | - | - |
| - | VB-DMD train | VB-DMD test | MetricGAN-U | U-ND | 8.2 | 3.20 | 2.45 | 3.11 | 0.77 |
| VB train | - | VB-DMD test | RVAE-VEM | U-NA | 17.1 | 3.23 | 2.48 | 3.15 | 0.81 |
| VB train | VB-DMD train | VB-DMD test | RVAE-LV | U-ND | 17.4 | 3.24 | 2.40 | 3.17 | 0.81 |

### 6.4.2 COMPARISON WITH THE BASELINES

As depicted in Table 6.1, the RVAE-LV model exhibits overall better performance compared to the other two variants across all three training configurations. Consequently, we opt to use the results achieved by the RVAE-LV model trained under the U-ND configuration for comparison with other baseline models. The results are reported in Table 6.2.

Before comparing the performance of different approaches, we make several remarks about the datasets and the evaluation metrics. First, we observe that all of the methods achieve higher score on the VB-DMD dataset than on the WSJ0-QUT dataset. This is reasonable because the test noisy sequences of the VB-DMD dataset have higher SNR than that of the WSJ0-QUT dataset, and the patterns of the added noise are also simpler. In short, VB-DMD is a “easier” dataset than the WSJ0-QUT dataset. Second, we notice that there exists a slight inconsistency among these evaluation metrics. For instance, the MetricGAN+ model achieves remarkable high PESQ scores (the highest among the evaluated models) on both of these two datasets. However, its SI-SDR scores are notably low. This inconsistency arises from the distinction between the metrics. Specifically, the SI-SDR score assesses the time-to-time signal estimation quality, whereas the PESQ score gauges the global perceptual quality. These two scores are not always correlated. Given that the MetricGAN+ model directly use the PESQ score as its training objective, it is unsurprising that it attains a very good PESQ score and a relatively lower SI-SDR score. Lastly, we included the evaluation scores for the noisy mixture and the oracle Wiener filter. The term “noisy mixture” denotes the direct assessment of the quality of the test noisy speech sequences using the corresponding clean speech as reference. On the other hand, “oracle Wiener filter” signifies the computation of the Wiener filter with the true clean speech and noise variances and applied to the noisy speech STFT spectrogram, with the evaluation based on the clean speech as reference. Noisy mixture scores provide a benchmark for gauging the improvements brought about by the models, while the oracle Wiener filter can be viewed as an upper bound for methods based on STFT spectrogram masking.

In terms of model performance, we observe that both the proposed RVAE-LV model and the RVAE-VEM model [22] achieve very competitive results, especially when compared to the supervised SE methods. Note that both of these two methods have never used pairs of aligned noisy-clean speech data for training. In comparison to the RVAE-VEM model, RVAE-LV demonstrates slightly improved results on the VB-DMD dataset but exhibits slightly inferior performance on WSJ0-QUT dataset. However, the significant advantage of RVAE-LV over RVAE-VEM lies in its ability to deliver a rapid inference process in the U-ND configuration, with only minimal performance degradation. Further discussion on this aspect will be presented in the next subsection. Besides, the ND + noise

**Table 6.3**: RTF of different models during inference. The best scores are in bold.

| Test set | Model | Configuration | # Iter. ↓ | RTF ↓ |
|---|---|---|---|---|
| WSJ0-QUT te. | RVAE-VEM | U-NA | 300 | 27.91 |
| | RVAE-LV | U-NA | 1000 | 89.42 |
| | | U-ND | - | **0.02** |
| | | U-NDA | 190 | 17.42 |
| VB-DMD te. | SGMSE+ | S | - | 3.39 |
| | RVAE-VEM | U-NA | 100 | 9.55 |
| | RVAE-LV | U-NA | 900 | 81.62 |
| | | U-ND | - | **0.02** |
| | | U-NDA | 25 | 2.32 |

adaptation configuration enables the model to adapt to specific noise types and improve performance.

### 6.4.3 DISCUSSION ON THE COMPUTATIONAL TIME

To evaluate the computational efficiency of different models at test time, we calculated the RTF for RVAE-VEM, SGMSE+ and the proposed RVAE-LV model across different configurations. The corresponding values are provided in Table 6.3. Both the RVAE-VEM model and the RVAE-LV model in the U-ND configuration employ an EM-based optimization method, with the computational time of these models being proportional to the required number of iterations during the EM algorithm. This iteration count is also reported in the table. In general, achieving good performance in the U-NA configuration is at the price of very high RTF values. In fact, the iteration number of the EM algorithm should be considered as a hyperparameter of the model. And in our experiments, this value is determined by grid search. It is noteworthy that, to achieve optimal performance, the RVAE-LV model requires more iterations than the RVAE-VEM model. This observation suggests that the LSTM-based noise model is more challenging to converge than the NMF-based noise model. In contrast, in the U-ND configuration, the inference process only requires a single forward pass of the trained model, resulting in a much lower RTF value of 0.02. Additionally, by conducting the noise adaptation fine-tuning for several iterations (190 iterations on the WSJ0-QUT dataset and 25 iterations on the VB-DMD dataset), the model's performance can be further enhanced. This can be viewed as a compromise between computational efficiency and model performance. On the other hand, due to the time-consuming inverse diffusion process, the state-of-the-art supervised baseline SGMSE+ has an RTF value that is much higher than the proposed unsupervised model in the U-ND configuration, while the two methods have similar performance in terms of SI-SDR (however, the RTF of SGMSE+ remains much lower than the proposed model in the U-NA and U-NDA configurations).

### 6.4.4 QUALITATIVE ANALYSIS

By substituting the NMF-based noise model with the DNN-based noise model, one of our expectations was that the DNN-based noise model could capture more complex noise patterns and consequently improve the SE performance. However, the results reported in Table 6.2 indicate that this objective is not achieved. To further investigate the potential underlying reasons, we conducted a qualitative analysis over two examples.

To gain a deeper understanding of the impact of the choice of the noise model and training configurations, we conducted a comparative analysis of the SE results. Specifically, we compared the outcomes achieved by the RVAE-LV method trained under both U-NA and U-ND configurations with those obtained by the RVAE-VEM method. In Figure 6.2, we present a challenging example featuring noise from real-life recordings and a relatively low SNR of 0 dB in the noisy speech. In this case, all three methods—RVAE-LV under the U-ND configuration, RVAE-LV under the U-NA configuration, and RVAE-VEM under the U-NA configuration—yield suboptimal results, with SI-SDR values of 4.52 dB, 4.42 dB and 6.85 dB respectively. Notably, the RVAE-LV model under both configurations performs less effectively than the RVAE-VEM model. Upon inspecting the estimated noise variance, we observe that the noise patterns learned by the RVAE-LV model trained under the U-ND configuration are much smoother than those learned by the NMF model and the RVAE-LV model trained under the U-NA configuration. The possible reason is that when training on a large dataset, the DNN-based noise model tends to learn averaged patterns over the whole dataset. When comparing the estimated noise variance obtained by the RVAE-LV model trained under the U-NA configuration with that

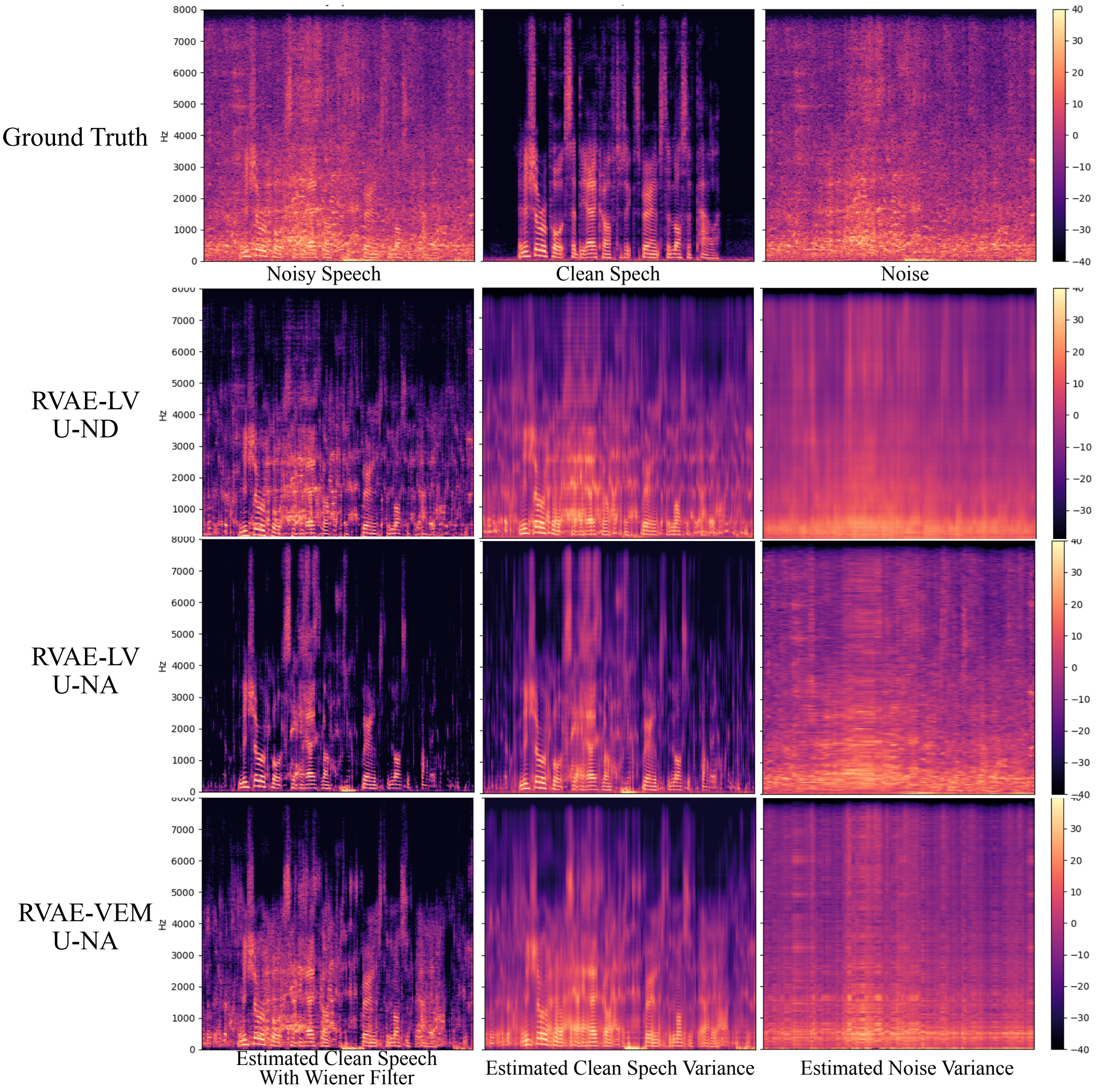


**Figure 6.2**: Example of SE results obtained with different methods. In this example, the noise type is “restaurant” and the SNR of the noisy speech is 0 dB. Best seen in color.

obtained by the RVAE-VEM model, it seems that the RVAE-LV model learned a more accurate estimation. However, despite this seemingly improved noise estimation, RVAE-LV under the U-NA configuration exhibits lower SI-SDR scores than the RVAE-VEM model. This discrepancy may be attributed to the possibility that when the noise model is too powerful, it retains less information about the clean speech in the estimated result.

In Figure 6.3 we present another simpler example with artificially synthesised speech-

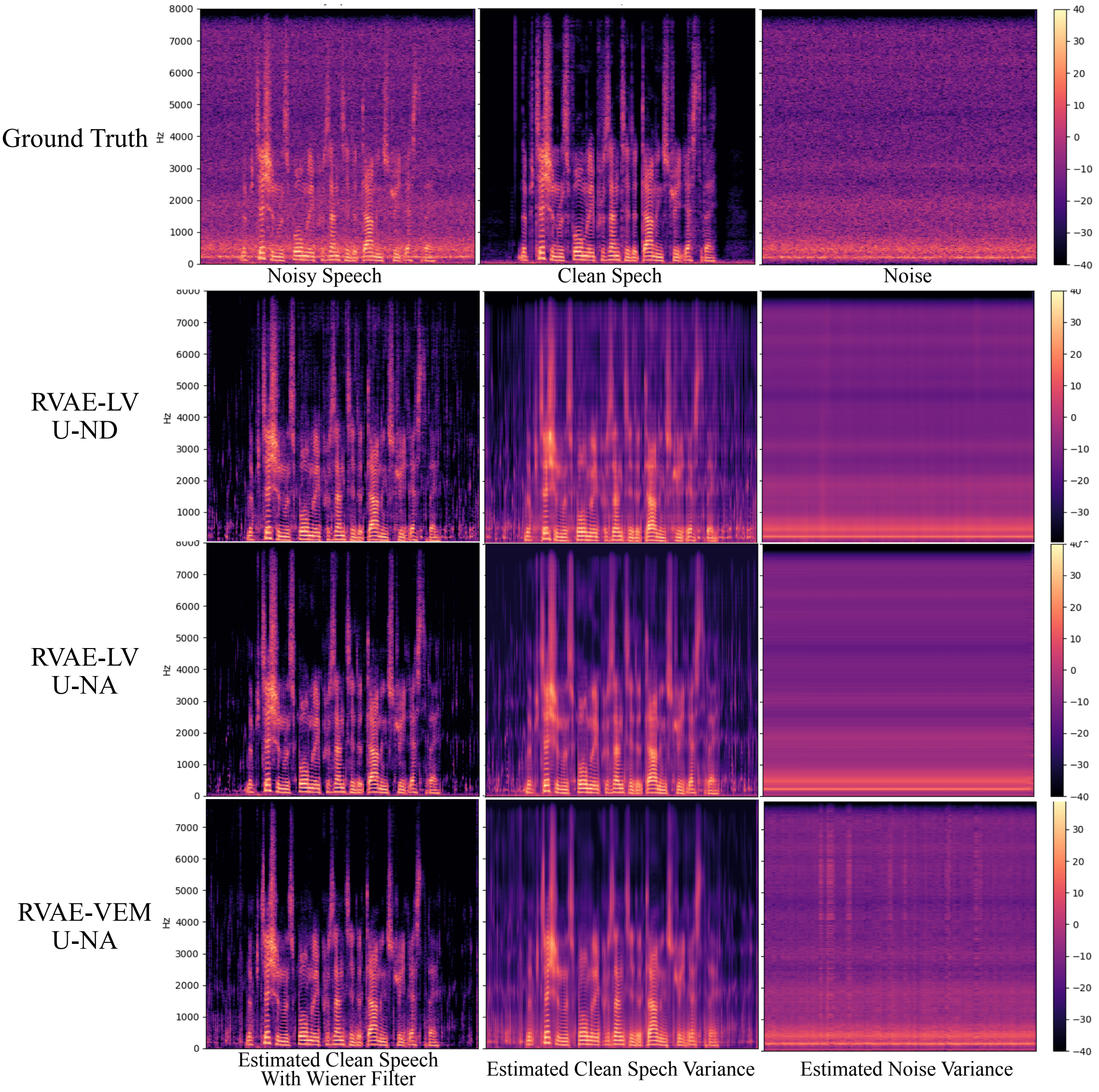


**Figure 6.3**: Example of SE results obtained with different methods. In this example, the noise type is "speech-shaped noise" and the SNR of the noisy speech is 5 dB. Best seen in color.

shaped noise and a higher SNR of 5 dB in the noisy speech. In this case, all three methods obtain good SE performance, with SI-SDR values of 13.7 dB, 14.8 dB and 11.2 dB respectively. Particularly, RVAE-LV models trained under both NA and ND configurations achieve higher SI-SDR scores than the RVAE-VEM method.

**Table 6.4**: SE results compared with baselines. U-NA stands for unsupervised noise-agnostic and U-ND stands for unsupervised noise-dependent. The best scores are in bold and the second best scores are underlined.

| Pre-train set | Train set | Test set | Model | Configuration | SI-SDR $\uparrow$ | $\text{PESQ}_{\text{MOS}} \uparrow$ | $\text{PESQ}_{\text{WB}} \uparrow$ | $\text{PESQ}_{\text{NB}} \uparrow$ | ESTOI $\uparrow$ |
|---|---|---|---|---|---|---|---|---|---|
| VB train | - | WSJ0-QUT test | RVAE-VEM | U-NA | **4.3** | 2.12 | 1.37 | 1.84 | 0.57 |
| VB train | WSJ0-QUT train | WSJ0-QUT test | RVAE-LV | U-ND | **4.3** | **2.17** | **1.41** | **1.85** | **0.58** |
| WSJ0 train | - | VB-DMD test | RVAE-VEM | U-NA | 17.3 | **3.21** | **2.41** | **3.13** | **0.81** |
| WSJ0 train | VB-DMD train | VB-DMD test | RVAE-LV | U-ND | **17.4** | 3.18 | 2.21 | 3.09 | 0.80 |

### 6.4.5 Results on Unmatched Pre-train/Test Set

Another motivation for the proposed method is its ability to adapt to noise characteristics and acoustic conditions not encountered during training. To explore this aspect, we conducted experiments involving unmatched pre-training and testing datasets. Specifically, we firstly pre-train the RVAE model on the WSJ0 clean speech dataset, and then, train the noise model and fine-tune the encoder on the VB-DMD training set. Subsequently, testing was carried out on the VB-DMD test set. We also conducted reverse experiments (RVAE model pre-trained on the VB dataset, then noise model and fine-tuning of the encoder trained and tested on the WSJ0-QUT dataset), and the results are reported in Table 6.4. Observing the performance on the WSJ0-QUT dataset, we note a slight drop in performance for both the RVAE-VEM and the RVAE-LV models compared to the scenario with matched pre-training and testing datasets. These two models achieve relatively similar performance, with the RVAE-LV model exhibiting slightly better perceptual scores. On the VB-DMD dataset, RVAE-VEM maintained its performance, while the perceptual scores of the RVAE-LV model has slightly dropped.

## 6.5 Conclusion and Further Discussions

In this chapter, we present a new unsupervised SE model that uses a dynamical DPGM for both speech and noise. We tested three different dependencies for the noise model (NO, LV, NOLV), as well as three ‘training/testing’ configurations (NA, ND, and ND +

noise adaptation). Experiments show that in the NA configuration, our model outperforms several unsupervised baselines (including RVAE+NMF), and competes well with the supervised baselines. In the ND configuration, our model provides a fast inference process with minimal performance degradation (especially for RVAE-LV). Furthermore, the ND + noise adaptation configuration enables the model to adapt to specific noise characters and further improve performance with much fewer iterations than in the ND configuration. Additionally, we conducted further analysis from various perspectives to provide deeper insights into the strengths and limitations of the proposed method.

Here, we make additional discussions about the limitations of this method and potential directions for improvement. In general, this method is based on the idea of first pre-training a dynamical DPGM over a clean speech dataset, then leverage the learned prior information over the clean speech dataset to realize unsupervised SE. The pre-trained clean speech model is combined with an additional noise model to estimate the clean speech variance and the noise variance, respectively. These estimated variances are used to compute the estimated Wiener filter, which is applied to the noisy speech to obtain the enhanced speech. The parameters of the noise model are learned directly from the noisy speech. Meanwhile, the encoder of the pre-trained model is fine-tuned to get adapted to the noisy speech input. The convergence of the training process is guaranteed by fixing the decoder of the pre-trained model. In the first place, the framework itself has an inherent limitation: Since the Wiener filter is a real-valued mask, this method completely ignores the clean speech phase information (the clean speech estimate retains the same phase values as the noisy speech; note that this limitation is not specific of our unsupervised DVAE-based approach, it is common to all methods based on the estimation of the speech and noise power spectrograms to build a Wiener filter). However, it is observed that the phase information may play an important role in improving the audio quality, and DNN-based methods using raw waveform audios as input can even surpass the oracle Wiener filter results [131]. Therefore, one potential direction for improvement is to apply this SE framework directly in the temporal domain, rather than applying it to STFT spectrograms. For instance, we can implement the DVAE models with 1D CNNs [203], enabling us to operate directly within the temporal domain. Secondly, the modeling capability and generalization ability of the pre-trained clean speech model are crucial for the success of this method. While DVAEs serve as powerful deep probabilistic generative models for sequential data, leading to satisfactory SE results, there may be room for improvement by exploring alternative encoder-decoder based speech modeling architectures. Thirdly, the selection of the noise model and the approach to learn its parameters are also essential aspects. In this work, we propose to replace the NMF-based noise model, learned using the VEM algorithm, with the DNN-based noise model trained with SGD method. We observe that while the proposed method can accelerate the inference time, it does not significantly contribute to improve the SE performance. A plausible explanation is that the training loss defined in Equation 6.16 may not be sufficiently effective in training the noise model over a large noisy speech dataset. Exploring a more effective unsupervised learning strategy is also a promising avenue for further research.

# CHAPTER 7

# SPEECH MODELING WITH A HIERARCHICAL TRANSFORMER DYNAMICAL VAE

*A computer would deserve to be called intelligent if it could deceive a human into believing that it was human.*

*— Alan Turing*

This chapter is based on the following publication:

Xiaoyu Lin, Xiaoyu Bie, Simon Leglaive, Laurent Girin, and Xavier Alameda-Pineda. "Speech modeling with a hierarchical Transformer dynamical VAE." In IEEE International Conference on Acoustics, Speech and Signal Processing, pages 1-5, 2023.

A part of this part of my PhD work has been conducted in collaboration with Xiaoyu Bie, who was another PhD student in the RobotLearn team at Inria at the time of this work, and is now a post-doc researcher at ParisTech Telecom. For the purpose of transparency, I clarify the contributions as follows. Xiaoyu Bie conceived the seminal idea of HiT-DVAE and developed the initial implementation code. In his own PhD work, he applied HiT-DVAE to 3D human motion data modeling. I re-implemented the code completely, conceived the idea of LigHT-DVAE, conducted the experiments on speech signals presented in the above paper and in this chapter, obtained all the experimental results reported in Table 7.1, Table 7.2, Figure 7.3 and Table 7.3, drafted the initial version of the above paper, participated in the paper refinement and went through all the process related to the publication. As a complement of information, I can add that Xiaoyu Bie has incorporated a portion of the results presented in Table 7.1, Figure 7.3, and Table 7.3, as well as some of the content of the above paper, into his PhD thesis manuscript.

## 7.1 INTRODUCTION

In Section 2.2.2, we introduced the DVAEs, which are a family of DPGMs that extends the VAE to model a sequence of observed data with the help of a sequence of latent vectors. In Chapter 3, 4, 5, and 6, we presented the applications of DVAEs on various unsupervised audio and video tasks. Owing to their flexibility and high generalization ability, DVAEs have demonstrated significant success in handling audio and visual tasks involving sequential data. In the majority of the DVAEs presented in the literature, temporal dependencies within each sequence and across two sequences are modeled using RNNs. RNNs are very classical deep neural network architectures for capturing temporal dependencies within sequential data. They have achieved considerable success across various tasks. Nevertheless, the use of RNNs in DVAEs also introduces certain challenges, with a common issue being the mismatch between training and test conditions. Indeed, the commonly adopted configuration for RNN training is to use the ground-truth past observed vectors $\mathbf{s}_{1:t-1}$ in the generative model, a training strategy often referred to as *teacher-forcing* [219]. However, at test/generation time, we can only use the previously generated values $\hat{\mathbf{s}}_{1:t-1}$ to generate the current one. This generally results in large accumulated prediction errors along the sequence. Directly training an RNN in the generation mode is also difficult. To remedy this problem, *scheduled sampling* can be adopted, i.e., the ground truth past vectors $\mathbf{s}_{1:t-1}$ are progressively replaced with the previously generated ones $\hat{\mathbf{s}}_{1:t-1}$ along the training iterations [14]. This strategy was successfully adopted for the training of autoregressive (AR) DVAEs, i.e. DVAEs with a recursive structure on the observed sequence $\mathbf{s}_{1:T}$ [71]. However, this requires a well designed sampling scheduler to guarantee the prediction performance. Additionally, RNNs are also known to be poorly suited for parallel computation and are prone to issues like exploding and vanishing gradients, particularly when dealing with long sequences [108]. The sequential computation characteristics of RNNs also limit their capabilities to be trained on large datasets.

Replacing RNNs with Transformers [207] for sequential modeling has become a new trend in both natural language processing, e.g., [45], and computer vision, e.g., [48]. In contrast to the recursive computation in RNNs, the Transformers architecture uses self-attention modules to capture temporal correlations within the input sequences of both the encoder and the decoder. Additionally, the correlations between the output sequence of the encoder and the input sequence of the decoder are modeled using a cross-attention module (See Section 2.4.3 for explanations of the attention mechanism). The incorporation of attention mechanisms empowers Transformers to effectively learn more flexible dependencies within the sequence, facilitating the modeling of complex and long-term correlations. Furthermore, the holistic attention architecture of Transformers aids in resolving the non-parallel computation constraints and addressing challenges related to exploding and vanishing gradients encountered in RNNs. In principle, replacing RNNs with Transformers in DVAEs can also benefit the probabilistic modeling of temporal dependencies between the observed and latent sequences. Following this idea, the hierarchical Transformer dynamical variational auto-encoder (HiT-DVAE) model proposed in [21] integrates a DVAE with Transformers. In [21], this model has been applied to 3D human motion generation. The HiT-DVAE model demonstrates robust sequential data modeling capabilities and attained competitive performance in this challenging task.

In the present PhD work, we apply the HiT-DVAE model for speech signal modeling. Additionally, we propose an improved version of HiT-DVAE, named LigHT-DVAE. The LigHT-DVAE model shares the same probabilistic model with HiT-DVAE but differs in the implementation of the dependencies. More specifically, LigHT-DVAE reduces the number of parameters by sharing the parameters of the decoders of the original HiT-DVAE model. Experimental results show that we achieve very competitive results in the speech analysis-resynthesis task. Furthermore, we investigate the modeling capacity of LigHT-DVAE compared to other DVAEs through extensive ablation studies on the model structure. In particular, we show that the special structure of both HiT-DVAE and LigHT-DVAE makes them robust to using teacher-forcing at training time, which largely simplifies the training procedure compared with other AR DVAEs.

## 7.2 PROBABILISTIC MODEL OF HIT-DVAE

The HiT-DVAE model is a DVAE in which the temporal dependencies between and within the observed and latent sequences are implemented with Transformers instead of RNNs [21]. In this section, we present the probabilistic model of HiT-DVAE. We first present the most general formulas and subsequently provide specific distributions when applying the HiT-DVAE model for speech signal modeling. In the context of speech modeling, we work with the STFT of speech waveforms, denoted as $\mathrm{s}_{1:T} \in \mathbb{C}^{F \times T}$. Each vector $\mathrm{s}_t = \{s_{f,t}\}_{f=1}^{F}$ of the sequence is the short-time complex-valued spectrum at time frame $t$, and $f$ denotes the frequency bin.

### 7.2.1 GENERATIVE MODEL

In the present context of speech signal modeling, $\mathbf{s}_{1:T}$ is the observed data sequence (as opposed to a latent vector sequence in Chapter 5, where it was the speech signal to track and separate from an observed mixture). In the HiT-DVAE model, two kinds of latent variables are introduced: a sequence of latent vectors $\mathbf{z}_{1:T} \in \mathbb{R}^{L_z \times T}$ used to capture the dynamical properties of the observed sequence, and a time-independent latent variable $\mathbf{w} \in \mathbb{R}^{L_w}$, defined for a whole speech utterance (typically a complete sentence from a given speaker), intended to encapsulate high-level features. These features may include, for example, the speaker ID or emotional state.[1]

The generative model of HiT-DVAE writes:

$$p_\theta(\mathbf{s}_{1:T}, \mathbf{z}_{1:T}, \mathbf{w}) = p_{\theta_\mathbf{w}}(\mathbf{w}) \prod_{t=1}^{T} p_{\theta_\mathbf{s}}(\mathbf{s}_t | \mathbf{s}_{1:t-1}, \mathbf{z}_{1:t}, \mathbf{w}) p_{\theta_\mathbf{z}}(\mathbf{z}_t | \mathbf{s}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{w}), \tag{7.1}$$

where $\theta = \theta_\mathbf{s} \cup \theta_\mathbf{z} \cup \theta_\mathbf{w}$. In general, the observed vector at each time step is supposed to follow a Gaussian conditional distribution with a diagonal covariance matrix:

$$p_{\theta_\mathbf{s}}(\mathbf{s}_t | \mathbf{s}_{1:t-1}, \mathbf{z}_{1:t}, \mathbf{w}) = \mathcal{N}\big(\mathbf{s}_t; \boldsymbol{\mu}_{\theta_\mathbf{s},t}, \mathrm{diag}(\mathbf{v}_{\theta_\mathbf{s},t})\big). \tag{7.2}$$

As already discussed and implemented in chapter 5, in the context of statistical audio/speech processing, the STFT coefficient at each time-frequency bin $s_{t,f}$ are assumed to follow a circularly-symmetric zero-mean complex Gaussian distribution [62, 124, 72]. The STFT coefficients at different frequency bins are assumed to be independent. The conditional distribution of $\mathbf{s}_t$ in Equation 7.1 thus writes:

$$p_{\theta_\mathbf{s}}(\mathbf{s}_t | \mathbf{s}_{1:t-1}, \mathbf{z}_{1:t}, \mathbf{w}) = \mathcal{N}_c\big(\mathbf{s}_t; \mathbf{0}, \mathrm{diag}(\mathbf{v}_{\theta_\mathbf{s},t})\big). \tag{7.3}$$

In this work, all covariance matrices are assumed diagonal and are represented by the vector of diagonal entries. Here, $\mathbf{v}_{\theta_\mathbf{s},t} \in \mathbb{R}_+^F$ depends on the variables $\{\mathbf{s}_{1:t-1}, \mathbf{z}_{1:t}, \mathbf{w}\}$. Note that in practice, HiT-DVAE will input speech power spectrogram, i.e., the squared

---

[1] In the above-mentioned human motion generation application, it could be the motion class.

modulus of $\mathbf{s}_{1:t-1}$, instead of the complex-valued STFT coefficients themselves, also as what we have discussed in Chapter 5. The latent vector $\mathbf{z}_t$ is assumed to follow a (real-valued) Gaussian distribution:

$$p_{\theta_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{s}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{w}) = \mathcal{N}\big(\mathbf{z}_t; \boldsymbol{\mu}_{\theta_{\mathbf{z}},t}, \text{diag}(\mathbf{v}_{\theta_{\mathbf{z}},t})\big). \tag{7.4}$$

The mean and variance vectors $\boldsymbol{\mu}_{\theta_{\mathbf{z}},t} \in \mathbb{R}^{L_z}$ and $\mathbf{v}_{\theta_{\mathbf{z}},t} \in \mathbb{R}_+^{L_z}$ both depend on $\{\mathbf{s}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{w}\}$. As for the latent vector $\mathbf{w}$, it is assumed to follow a standard Gaussian prior:

$$p_{\theta_{\mathbf{w}}}(\mathbf{w}) = \mathcal{N}(\mathbf{w}; \mathbf{0}, \mathbf{I}). \tag{7.5}$$

### 7.2.2 INFERENCE MODEL

The HiT-DVAE inference model approximates the intractable exact posterior distribution of the latent sequence and is defined as:

$$q_\phi(\mathbf{z}_{1:T}, \mathbf{w}|\mathbf{s}_{1:T}) = q_{\phi_{\mathbf{w}}}(\mathbf{w}|\mathbf{s}_{1:T}) \prod_{t=1}^{T} q_{\phi_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{s}_{1:T}, \mathbf{w}), \tag{7.6}$$

with $\phi = \phi_{\mathbf{w}} \cup \phi_{\mathbf{z}}$. The conditional distributions of $\mathbf{w}$ and $\mathbf{z}_t$ are defined as:

$$q_{\phi_{\mathbf{w}}}(\mathbf{w}|\mathbf{s}_{1:T}) = \mathcal{N}\big(\mathbf{w}; \boldsymbol{\mu}_{\phi_{\mathbf{w}}}, \text{diag}(\mathbf{v}_{\phi_{\mathbf{w}}})\big), \tag{7.7}$$

$$q_{\phi_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{s}_{1:T}, \mathbf{w}) = \mathcal{N}\big(\mathbf{z}_t; \boldsymbol{\mu}_{\phi_{\mathbf{z}},t}, \text{diag}(\mathbf{v}_{\phi_{\mathbf{z}},t})\big), \tag{7.8}$$

and where $\boldsymbol{\mu}_{\phi_{\mathbf{w}}} \in \mathbb{R}^{L_w}$ and $\mathbf{v}_{\phi_{\mathbf{w}}} \in \mathbb{R}_+^{L_w}$ depend on $\mathbf{s}_{1:T}$, and $\boldsymbol{\mu}_{\phi_{\mathbf{z}},t} \in \mathbb{R}^{L_z}$ and $\mathbf{v}_{\phi_{\mathbf{z}},t} \in \mathbb{R}_+^{L_z}$ depend on $\{\mathbf{s}_{1:T}, \mathbf{w}\}$.

### 7.2.3 OPTIMIZATION

The parameters of both the encoder $\phi = \phi_{\mathbf{w}} \cup \phi_{\mathbf{z}}$ and the decoder $\theta = \theta_{\mathbf{s}} \cup \theta_{\mathbf{z}} \cup \theta_{\mathbf{w}}$ are optimized by maximizing the ELBO, which is defined as

$$\mathcal{L}(\theta, \phi; \mathbf{s}_{1:T}) = \mathbb{E}_{q_\phi(\mathbf{z}_{1:T}, \mathbf{w}|\mathbf{s}_{1:T})}\Big[\log \frac{p_\theta(\mathbf{s}_{1:T}, \mathbf{z}_{1:T}, \mathbf{w})}{q_\phi(\mathbf{z}_{1:T}, \mathbf{w}|\mathbf{s}_{1:T})}\Big], \tag{7.9}$$

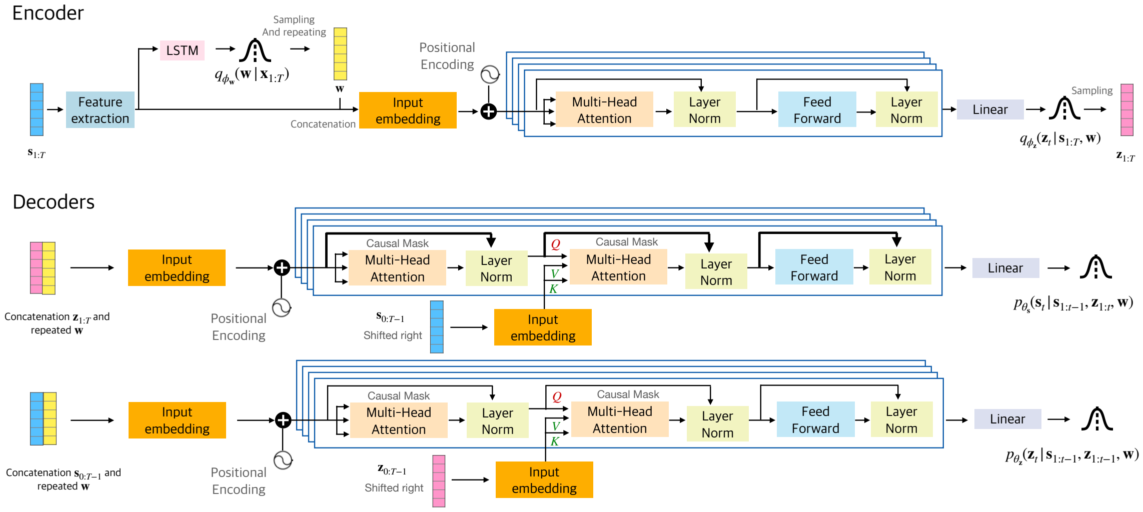


**Figure 7.1**: HiT-DVAE model architecture.

over a training speech dataset. In the context of speech modeling, by injecting Equation 7.1, Equation 7.3, Equation 7.4, Equation 7.5, Equation 7.6, Equation 7.7 and Equation 7.8 into Equation 7.9, the ELBO can be developed as:

$$\begin{aligned}
\mathcal{L}(\theta, \phi; \mathbf{s}_{1:T}) &= D_{\mathrm{KL}}(q_{\phi_{\mathbf{w}}}(\mathbf{w}|\mathbf{s}_{1:T}) \parallel p_{\theta_{\mathbf{w}}}(\mathbf{w})) \\
&- \sum_{t=1}^{T} \mathbb{E}_{q_{\phi_{\mathbf{z}}} q_{\phi_{\mathbf{w}}}} \Big[ d_{\mathrm{IS}}(|\mathbf{s}_t|^2, \mathbf{v}_{\theta_{\mathbf{s}},t}) \\
&+ D_{\mathrm{KL}}(q_{\phi_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{s}_{1:T}, \mathbf{w}) \parallel p_{\theta_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{s}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{w})) \Big]
\end{aligned} \tag{7.10}$$

where $d_{\mathrm{IS}}(\cdot,\cdot)$ is the IS divergence [62], $D_{\mathrm{KL}}(\cdot||\cdot)$ is the KL divergence, as previously defined in Chapter 5. And the modulus square function is applied element-wise.

## 7.3 Model Architecture of HiT-DVAE and LigHT-DVAE

We note that there exist various ways to implement the probabilistic model defined in Section 7.2 using the Transformer architecture. In this section, we present the specific implementation of HiT-DVAE and the adaptations made in LigHT-DVAE. The overall architecture of HiT-DVAE is illustrated in Figure 7.1.

### 7.3.1 HIT-DVAE ENCODER

The HiT-DVAE model is constructed based on the original Transformer architecture [207], containing both an encoder and a decoder. The encoder is used to compute the parameters of the inference model distributions. It takes the observed sequence $\mathbf{s}_{1:T}$ as input. After passing through the feature extraction layers, it is then processed by an LSTM to compute $\boldsymbol{\mu}_{\phi_\mathbf{w}}$ and $\mathbf{v}_{\phi_\mathbf{w}}$, which are the parameters of the distribution $q_{\phi_\mathbf{w}}(\mathbf{w}|\mathbf{x}_{1:T})$. We only retain the LSTM output at the last time index $T$, since it contains all information on the complete sequence $\mathbf{s}_{1:T}$. As discussed in Section 2.3.5, the expectations with respect to the approximated posterior distribution in the ELBO are estimated using the Monte-Carlo method. The latent variable $\mathbf{w}$ is sampled employing the reparametrization trick. The obtained $\mathbf{w}$ is repeated for $T$ times and concatenated with the extracted observed sequence features. After passing through the input embedding layer and incorporating positional encodings [207], the inputs traverse the Transformer encoder layers. Each encoder layer consists of a multi-head self-attention (MHSA) sub-module and a feed-forward (FF) sub-module, see Figure 7.1 top. Layer normalization (LN) and residual connections are also applied, as in the original Transformer introduced in Section 2.4.3. To enhance the model capacity, multiple encoder layers can be stacked. The MHSA module is the essential component for extracting temporal correlations within the input sequence. It is important to note that in the MHSA module of the encoder layers, no masks are added. Consequently, at any time frame $t$, the query has access to the keys and values of the entire sequence, consistent with the temporal dependencies specified in Equation 7.8. In short, the HiT-DVAE encoder in non-causal. The output of the last Transformer encoder layer will finally pass through a linear layer to obtain the parameters $\boldsymbol{\mu}_{\phi_\mathbf{w},t}$ and $\mathbf{v}_{\phi_\mathbf{w},t}$ for the distributions $q_{\phi_\mathbf{z}}(\mathbf{z}_t|\mathbf{x}_{1:T}, \mathbf{w})$, from which we can sample the latent variables $\mathbf{z}_{1:T}$ using cascaded reparametrization tricks.

### 7.3.2 HIT-DVAE DECODER

HiT-DVAE includes two decoders: one for computing the parameters of the distributions $p_{\theta_\mathbf{s}}(\mathbf{s}_t|\mathbf{s}_{1:t-1}, \mathbf{z}_{1:t}, \mathbf{w})$ and the other for computing the parameters of the distributions

$p_{\theta_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{s}_{1:t-1}, \mathbf{z}_{1:t-1}, \mathbf{w})$, see Figure 7.1 bottom. For the decoder to generate the conditional distributions of $\mathbf{s}_t$, the model takes as input the concatenation of $\mathbf{z}_{1:T}$ and the replicated sampled value of $\mathbf{w}$. Input embedding layers and positional encodings are employed similarly to the encoder. Following the standard Transformer decoder structure, in each of the stacked decoder layers, three sub-modules are implemented, namely an MHSA, a multi-head cross-attention (MHCA) module, and an FF module. In the MHSA module, the queries are the outputs of the MHSA module applied over the input sequence, i.e., the concatenated vectors of $\mathbf{z}_{1:T}$ and $\mathbf{w}$, while the keys and values are the embeddings of the right-shifted observed sequence $\mathbf{s}_{0:T-1}$. The MHSA module is used to extract the inner correlations within the input sequence of the decoder (concatenation of $\mathbf{z}_{1:T}$ and $\mathbf{w}$), while the MHCA module is used to exploit the dependencies between the input sequence of the decoder and the sequence $\mathbf{s}_{0:T-1}$. In order to respect the temporal dependencies in Equation 7.3, causal masks are applied in both the MHSA module and the MHCA module to prevent access to information from the future, and preserve the causal nature of the generation process.

For the decoder to generate the conditional distributions of $\mathbf{z}_t$, the model takes as input the concatenation of the right-shifted sequence $\mathbf{s}_{0:1-T}$ and the replicated sampled value of $\mathbf{w}$. The same model structure is adopted as in the decoder for $\mathbf{s}_t$, except that in the MHCA module, the queries are the outputs of the MHSA module applied over the input sequence, i.e., the concatenated vectors of $\mathbf{s}_{0:1-T}$ and $\mathbf{w}$, while the keys and values are the embeddings of the right-shifted latent vectors $\mathbf{z}_{0:T-1}$.

### 7.3.3 DISTINCTIONS WITH THE ORIGINAL TRANSFORMERS ARCHITECTURE

We emphasize several notable differences in the HiT-DVAE architecture compared to the original Transformers: (a) There are two decoders working in parallel: one to compute the parameters of Equation 7.2 and one to compute the parameters of Equation 7.4; (b) The output of the encoder and decoders are distribution parameters instead of ‘deterministic’ data representations in the conventional use of the Transformers; (c) The output target of the first decoder is the input data $\mathbf{s}_{1:T}$, indicating that the model is trained to reconstruct

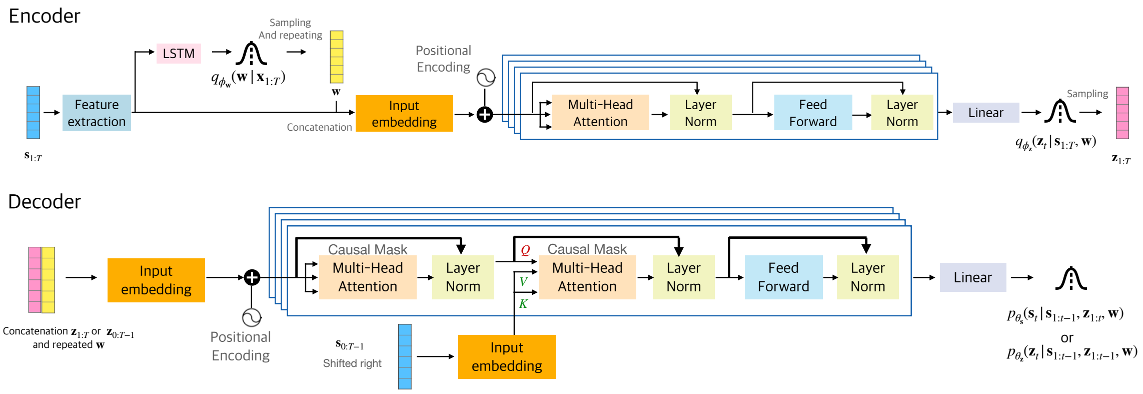


**Figure 7.2**: LigHT-DVAE model structure.

the input sequence rather than generate a different sequence, as commonly seen in regression or machine translation problems; (d) A global latent variable $\mathbf{w}$ is introduced to extract sequence-level features; (e) In the MHCA module of the decoder, to generate the distribution parameters of $\mathrm{s}_t$, the role of $\mathbf{z}_{1:T}$ (encoder output) and $\mathrm{s}_{0:T-1}$ (decoder output) as source information for the queries and keys/values, respectively, is inverted compared to the conventional Transformer as presented in Section 2.4.3. Points (a), (b) and (c) are inherent to the DVAE framework. Point (d) is a model design choice (see Section 7.2.1). Point (e) stands as a crucial choice to mitigate the accumulated error issues arising from the teacher-forcing training strategy, as introduced at the beginning of this chapter. A comprehensive justification of this choice will be provided in Section 7.5.2.

### 7.3.4 LIGHT-DVAE ADAPTATIONS

We observe from Equation 7.2 and Equation 7.4 that the generative distributions of $\mathrm{s}_t$ and $\mathbf{z}_t$ share very similar temporal dependencies across the sequences of $\mathrm{s}$ and $\mathbf{z}$. The key distinction lies in the fact that the generation of $\mathrm{s}_t$ is contingent upon both the past and current values of $\mathbf{z}$, whereas the generation of $\mathbf{z}_t$ relies solely on the past values of $\mathbf{z}$. In HiT-DVAE, two decoders are employed to model the sequential dependencies of these two generative distributions separately. Nevertheless, we would like to explore if these two distributions can share the sequential dependency implementations so that we can use a single decoder for the generative model. This can help to reduce the overall number of model parameters and make the model more compact and easy to train. Therefore, we propose the LigHT-DVAE model, which uses a single decoder instead of two decoders to generate the parameters of both Equation 7.2 and Equation 7.4. The decoder takes as input either the concatenation of $\mathbf{z}_{1:T}$ and the repeated values of $\mathbf{w}$ (for the generation of $\mathbf{s}_t$) or the concatenation of $\mathbf{z}_{0:T-1}$ and the repeated values of $\mathbf{w}$ (for the generation of $\mathbf{z}_t$). The queries of the MHCA module of the decoder is the output of the MHSA module applied over the input sequence, while the keys and values are the embeddings of $\mathbf{s}_{0:T-1}$. The model architecture of LigHT-DVAE is illustrated in Figure 7.2.

## 7.4 EXPERIMENTAL SETTINGS

We conducted a set of experiments to evaluate the efficiency of HiT-DVAE and LigHT-DVAE for speech signal modeling. In this section, we first describe the experimental settings that we used, and in the next section we will provide the experimental results.

### 7.4.1 DATASETS AND PRE-PROCESSING

The experiments are conducted on two datasets: the Wall Street Journal (WSJ0) dataset [63] and the Voice Bank (VB) corpus [208]. The WSJ0 dataset is composed of 16-kHz speech signals, with three subsets: *si_tr_s*, *si_dt_05* and *si_et_05*, used for model training, validation and test, and containing 12,765, 1,026 and 651 utterances respectively. The VB dataset contains a training set with 11,572 utterances performed by 28 speakers and a test set with 824 utterances performed by 2 speakers. We follow [60] to choose two speakers (p226 and p287) from the training set for validation, which contains 770 utterances, and use the leftover 26 speakers for training.

In all experiments, the raw audio signals are preprocessed in the following way. First, the silence at the beginning and end of the signals is cropped by using a voice activity detection threshold of 30 dB. Then, the waveform signals are normalized so that their maximum absolute value is one. The STFT is computed with a 64-ms sine window (1024 samples) and a 75%-overlap (256 samples hop length), resulting in a sequence of 513-dimensional discrete Fourier coefficients (for positive frequencies). Finally, the power magnitude of the STFT coefficients is computed. We set the sequence length of each

STFT spectrogram to $T = 150$ (corresponding to speech segments of 2.4s) for WSJ0 and $T = 100$ (corresponding to speech segments of 1.6s) for VB. At test time, the model is evaluated on the complete test utterances, which can be of variable length.

### 7.4.2 IMPLEMENTATION DETAILS AND TRAINING SETTINGS

The Transformer-based encoder and decoders in both HiT-DVAE and LigHT-DVAE are composed of 4 identical layers as described in Section 7.3.1 and Section 7.3.2. All input vectors are embedded into vectors of dimension $d_{\text{model}} = 256$, which is the size for all multi-head attention blocks. In practice, we applied single-head attention because we found that using multi-head attention decreased the performance in our experiments. All feed-forward blocks in the Transformer layers consist of two dense layers with size 1024 and 256. The latent dimension $L_z$ and $L_w$ are set to $16$ and $32$, respectively.

### 7.4.3 BASELINES

We compare HiT-DVAE and LigHT-DVAE with the basic VAE and three other DVAE models: The Deep Kalman Filter (DKF) [109], the Recurrent Variational AutoEncoder (RVAE) [116], and the stochastic recurrent neural network (SRNN) [55]. As HiT/LigHT-DVAE, SRNN is an AR model that uses $\mathbf{s}_{1:t-1}$ to generate $\mathbf{s}_t$. DKF and RVAE are not AR. DKF generates $\mathbf{s}_t$ from $\mathbf{z}_t$ and there is a dynamical model on $\mathbf{z}_t$ only, whereas RVAE generates $\mathbf{s}_t$ 'directly' from $\mathbf{z}_{1:t}$ (we use the causal version of RVAE; see [116]). As mentioned in the introduction, an AR model can be trained either in *teacher-forcing* (TF) mode or in *scheduled-sampling* (SS) mode. In our experiments, SRNN is trained in both modes, while HiT/LigHT-DVAE are only trained in the teacher-forcing mode. Except for that, all models are trained with the same settings described in Section 7.4.2.

### 7.4.4 EVALUATION METRICS

We evaluate the average speech analysis-resynthesis performance on the test set (in generation mode for AR models) using four evaluation metrics: The root mean squared error (RMSE), the scale-invariant signal-to-distortion ratio (SI-SDR) [175] in dB, the perceptual evaluation of speech quality (PESQ) score [172] (in $[-0.5, 4.5]$), and the extended

**Table 7.1**: Speech spectrograms analysis-resynthesis results.

| Dataset | Model | RMSE ↓ | SI-SDR ↑ | PESQ ↑ | ESTOI ↑ |
|---|---|---|---|---|---|
| WSJ0 | VAE | 0.040 | 7.4 | 3.28 | 0.88 |
| | DKF | 0.037 | 8.3 | 3.51 | **0.91** |
| | RVAE | 0.034 | 8.9 | 3.53 | **0.91** |
| | SRNN (SS) | 0.036 | 8.7 | **3.57** | **0.91** |
| | SRNN (TF) | 0.061 | 2.6 | 2.53 | 0.76 |
| | HiT-DVAE | 0.031 | 10.0 | 3.52 | **0.91** |
| | LigHT-DVAE | **0.030** | **10.1** | 3.55 | **0.91** |
| VB | VAE | 0.052 | 8.4 | 3.24 | 0.89 |
| | DKF | 0.048 | 9.3 | 3.44 | 0.91 |
| | RVAE | 0.050 | 8.9 | 3.39 | 0.90 |
| | SRNN (SS) | 0.044 | 10.1 | 3.42 | 0.91 |
| | SRNN (TF) | 0.102 | -0.1 | 2.15 | 0.75 |
| | HiT-DVAE | 0.039 | 11.4 | **3.60** | **0.93** |
| | LigHT-DVAE | **0.038** | **11.6** | 3.58 | **0.93** |

short-time objective intelligibility (ESTOI) score [195] (in $[0, 1]$). We also evaluate the average generation performance via the Fréchet Deep Speech Distance (FDSD) proposed in [25]. FDSD is a quantitative metric for speech signals generative models. It relies on speech features extracted with the pre-trained speech recognition model DeepSpeech2 [3].

## 7.5 EXPERIMENTAL RESULTS

### 7.5.1 SPEECH SPECTROGRAMS ANALYSIS-RESYNTHESIS RESULTS

The speech analysis-resynthesis task aims at reconstruct the speech signal from the learned latent representations. In our experimental context, the HiT-DVAE and LigHT-DVAE models will input the squared magnitude of the speech STFT spectrogram, as is done in Chapter 5 and Chapter 6, and output the reconstructed squared magnitude of the spectrogram. Nevertheless, to compute the PESQ and the ESTOI scores, we need to reconstruct the waveform of the speech. To achieve this, we combined the reconstructed magnitude of the spectrogram with the phase of the original speech signal.

The speech analysis-resynthesis results on both the WSJ0 dataset and the VB dataset are reported in Table 7.1. On the WSJ0 dataset, HiT-DVAE and LigHT-DVAE outperform the other DVAE models for all metrics, except for PESQ (SRNN (SS) is slightly better) and ESTOI (the five models are on par). On the VB dataset, HiT/LigHT-DVAE outperform all other DVAE models on all metrics. As AR models, even if HiT/LigHT-DVAE are trained in teacher-forcing mode, they still keep very robust performance when tested in generation mode. In contrast, SRNN (TF) trained in teacher-forcing mode leads to a notable drop of performance compared to SRNN (SS) trained in scheduled-sampling mode. Finally, the experimental results show that sharing the parameters of the decoders for $\mathrm{s}$ and $\mathbf{z}$ in LigHT-DVAE slightly increases the performance compared to HiT-DVAE. At the same time, it reduces the number of model parameters from 21.75 M to 17.46 M.

### 7.5.2 ABLATION STUDIES ON MODEL STRUCTURES

To try to explain why HiT/LigHT-DVAE are robust to the teacher-forcing training mode, we performed several ablation studies on the models structure on the VB dataset. As explained in Section 7.3.3, different to the original Transformer decoder, the queries to decode $\mathrm{s}_t$ in HiT/LigHT-DVAE are computed from $\mathbf{z}_{1:t}$ instead of $\mathrm{s}_{1:t-1}$. In a first ablation experiment, we invert the role of the queries and the keys/values when decoding $\mathrm{s}_t$, i.e., we swap the decoder inputs $\mathbf{z}_{1:t}$ and $\mathrm{s}_{1:t-1}$. the resulting models are denoted with “Inv-s”. The models are always trained in the teacher-forcing mode. Table 7.2 reports the evaluation results when using generated or ground-truth $\mathrm{s}_{1:t-1}$ to decode $\mathrm{s}_t$, in the “GEN” and “GT” rows respectively.

Although the models in the Inv-s configuration achieve very good results when trained in TF mode and evaluated with GT values of $\mathrm{s}_{1:t-1}$ (i.e. no train/test mismatch), their performance notably drops when evaluating in GEN mode (i.e. train/test mismatch; for example, we observe a drop from 11.4 to 3.8 dB SI-SDR for HiT-DVAE-Inv-s). We believe that this lack of robustness to a mismatch between the training and test modes is mainly caused by the residual connections on the queries in the original Transformer decoder architecture (which are highlighted in bold in Figure 7.1 and Figure 7.2). Indeed, the presence of residual connections in the decoders emphasizes the reliance of the output on information from the queries rather than that from the keys and values in the MHCA mod-

**Table 7.2**: Speech spectrograms analysis-resynthesis results: Ablation studies on the HiT/LigHT-DVAE models structure.

| Test $\mathrm{s}_{1:t-1}$ | Model | RMSE ↓ | SI-SDR ↑ | PESQ ↑ | ESTOI ↑ |
|---|---|---|---|---|---|
| GEN | HiT-DVAE | 0.039 | 11.4 | 3.60 | 0.93 |
| | HiT-DVAE-Inv-s | 0.079 | 3.8 | 2.61 | 0.75 |
| | HiT-DVAE-Inv-s-NRC | 0.067 | 5.8 | 2.68 | 0.78 |
| | LigHT-DVAE | 0.038 | 11.6 | 3.58 | 0.93 |
| | LigHT-DVAE-Inv-s | 0.079 | 3.9 | 2.58 | 0.75 |
| | LigHT-DVAE-Inv-s-NRC | 0.068 | 5.7 | 2.63 | 0.78 |
| GT | HiT-DVAE | 0.038 | 11.5 | 3.60 | 0.93 |
| | HiT-DVAE-Inv-s | 0.038 | 11.4 | 3.32 | 0.90 |
| | HiT-DVAE-Inv-s-NRC | 0.067 | 5.8 | 2.68 | 0.78 |
| | LigHT-DVAE | 0.038 | 11.7 | 3.59 | 0.93 |
| | LigHT-DVAE-Inv-s | 0.040 | 10.9 | 3.29 | 0.89 |
| | LigHT-DVAE-Inv-s-NRC | 0.068 | 5.7 | 2.63 | 0.78 |

ule. In the Inv-s configuration, $\mathrm{s}_{1:t-1}$ is employed to compute the queries and the residual connections make the prediction of $\mathrm{s}_t$ rely directly on the preceding vectors $\mathrm{s}_{1:t-1}$. This is not the case in the 'regular/original' HiT/LigHT-DVAE models where $\mathbf{z}_{1:t-1}$ is used to compute the queries from which residual connections start. This forces the prediction of $\mathrm{s}_t$ to rely more on $\mathbf{z}_{1:t-1}$ than $\mathrm{s}_{1:t-1}$, which explains the better generalization of HiT/LigHT-DVAE compared to HiT/LigHT-DVAE-Inv-s. This structural aspect is very important in the present context of speech modeling, because adjacent speech spectrogram frames are much more correlated than discrete tokens in natural language processing, the original application domain of Transformers. To confirm this interpretation, we further removed the residual connections in the decoder of $\mathrm{s}_t$ in the Inv-s configuration (this new configuration is referred to as Inv-s-NRC, NRC standing for 'no residual connection'). As a result, the performance slightly increased compared to the Inv-s configuration in the GEN test mode, but it remains much below that of the original HiT/LigHT-DVAE. Also, the performance in the Inv-s-NRC configuration is the same when evaluated in GEN mode and GT mode, confirming that the generalization problem in the Inv-s configuration was due to the use of $\mathrm{s}_{1:t-1}$ to compute the queries from which residual connections start. Overall, this ablation study experimentally showed the importance of the architectural design made in the HiT/LigHT-DVAE models for the modeling of highly-correlated sequences of continuous

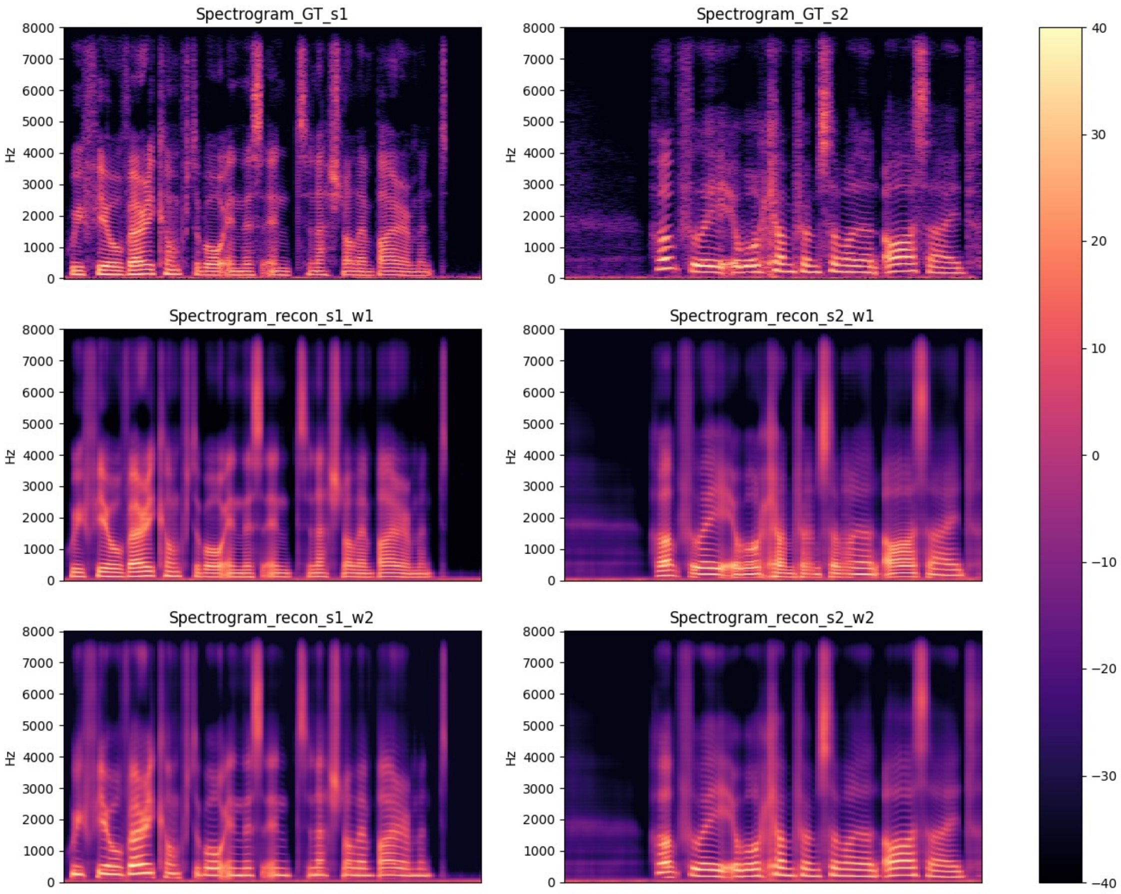


**Figure 7.3**: Speech re-synthesis results for two example speech signals with and without swapping $\mathbf{w}$. In the first column, GT_s1, recon_s1_w1, recon_s1_w2 represent the ground truth spectrogram of $\mathbf{s}_1$, the spectrogram re-synthesised from $\mathbf{z}_1$, $\mathbf{w}_1$, and the spectrogram re-synthesised from $\mathbf{z}_1$, $\mathbf{w}_2$, respectively. Similar notations apply to $\mathbf{s}_2$ in the second column.

data such as speech spectrograms.

### 7.5.3 INTERPRETABILITY OF THE GLOBAL LATENT VARIABLE

As described in Section 7.2.1, the latent variable $\mathbf{w}$ is designed to capture sequence-level features such as the speaker identity. We conducted tests to better understand the information actually embedded in $\mathbf{w}$. Specifically, we swapped the $\mathbf{w}$ value encoded from two different speech sequences –one by a female speaker and one by a male speaker– before resynthesis, similarly to what was done in [91]. The results are shown in Figure 7.3.

**Table 7.3**: Speech spectrograms generation results.

| Model | FDSD ↓ |
|---|---|
| VAE | 70.92 ± 0.44 |
| DKF | 32.78 ± 0.28 |
| RVAE | 45.75 ± 0.11 |
| SRNN (SS) | 25.28 ± 0.19 |
| SRNN (TF) | 25.53 ± 0.13 |
| HiT-DVAE | **22.50 ± 0.26** |
| LigHT-DVAE | 29.22 ± 0.26 |
| VB Test (exact phase) | 4.11 ± 0.14 |
| VB Test (Griffin-Lim) | 4.11 ± 0.15 |

We observe that the F0 range of the two spectrograms also swapped, indicating that $\mathbf{w}$ encodes the speaker or at least the speaker genre.

### 7.5.4 SPEECH SPECTROGRAMS GENERATION RESULTS

In addition to the above speech analysis-resynthesis experiments, we also evaluated the models performance on the unconditional speech spectrogram generation task via FDSD scores. To compute these scores, we generated 10,240 1.6-s speech signal samples from each model. The DVAE models generate power spectrograms, from which we deduced magnitude spectrograms. We then used the Griffin-Lim algorithm [79] to reconstruct the corresponding phase spectrograms, and then generate the waveform signals with inverse STFT. We used 1.6-s utterances from the VB training set as the reference set to compute the FDSD scores. Finally, in order to have a reference FDSD score, we also computed the FDSD on the VB test set, with exact original phase or phase reconstructed by the Griffin-Lim algorithm. As shown in Table 7.3, HiT-DVAE outperforms the other DVAE models on this generation task. This is a new result, since HiT-DVAE was never used before for speech modeling and generation. It can also be seen that the AR DVAE models (HiT/LigHT-DVAE and SRNN) generally have a better generation performance than the non-AR DVAE models.

Besides, we observe that despite the similarity in analysis-resynthesis between LigHT-DVAE and HiT-DVAE results across all metrics, LigHT-DVAE exhibits inferior unconditional speech generation abilities compared to the HiT-DVAE model. We hypothesize that this discrepancy may arise because, during speech analysis-resynthesis, the generative model of $\mathbf{z}$ was not utilized (the signals were re-synthesised from the $\mathbf{z}$ values obtained from the encoder). Given that the generative model of $\mathbf{s}$ is well-learned for both HiT-DVAE and LigHT-DVAE, they exhibit similar performance. In contrast, for speech generation, $\mathbf{z}$ is derived from the generative model of $\mathbf{z}$. In LigHT-DVAE, this portion of the networks is shared with the generative model of $\mathbf{s}$, potentially impacting its performance compared to networks trained separately.

## 7.6 CONCLUSION

In this work, we employed the HiT-DVAE model from [21] and its new variant LigHT-DVAE for the analysis-resynthesis and unconditional generation of speech signals. Our experiments showcase the robust modeling capabilities of HiT/LigHT-DVAE on speech spectrograms, emphasizing their resilience to the mismatch between the teacher-forcing mode used during training and the generation mode used during testing, a notable improvement over previous LSTM-based autoregressive DVAE models like SRNN. This allows for a more straightforward training procedure. Furthermore, comparing with HiT-DVAE, the newly proposed LigHT-DVAE model achieves competitive performance with approximately $20\%$ fewer parameters. Additionally, we have investigated the significance of the global latent variable $\mathbf{w}$. Experimental results indicate that $\mathbf{w}$ can effectively encode certain time-independent information across the entire sequence.

We anticipate that HiT/LigHT-DVAE holds significant promise in the realm of unsupervised speech representation learning and dynamical modeling. Such advancements can potentially benefit various downstream speech processing tasks, including but not limited to speech enhancement, speech separation, and speech inpainting.

## CHAPTER 8

# CONCLUSION AND FURTHER DISCUSSIONS

*All that is real is reasonable, and all that is reasonable is real.*
*Was vernünftig ist, das ist wirklich; und was wirklich ist, das ist vernünftig.*

*— Georg Wilhelm Friedrich Hegel*

## 8.1 CONCLUSION

In this dissertation, we have investigated the applications of DPGMs in diverse audio and visual tasks within the framework of unsupervised or weakly supervised learning configurations. More precisely, our focus revolves around a specialized family of DPGMs tailored for sequential data, specifically the DVAE models. We leverage these models to address intricate engineering challenges, including multi-source trajectory modeling and separation, with further applications for MOT and SC-ASS, as well as SE and speech modeling.

In Chapter 2, we offered a concise overview of contemporary research on DPGMs. Our focus then shifted to a specific type of generative models – the LVMs. We delved into the probabilistic modeling structure of various LVMs and discussed the learning and inference methods of LVMs. Afterwards, we presented diverse deep neural network architectures for sequential data modeling, including RNNs, 1D CNNs and Transformers. At the end of this chapter, we provided a brief introduction to the research backgrounds of the three primary tasks addressed in this dissertation: MOT, SC-ASS, and SE.

In Chapter 3, we proposed a versatile LVM framework designed to address the complex task of modeling and separating trajectories from multiple sources. Our proposed solution harnesses the capabilities of a DVAE model for effectively capturing the dynamic information of each individual source. Additionally, we incorporate a discrete latent assignment variable to facilitate the assignment of observations to specific sources, enabling the formation of coherent trajectories. The DVAE model(s) undergo(es) initial pre-training on a single-source dataset. Subsequently, the pre-trained DVAE model(s) is/are seamlessly integrated into the proposed MixDVAE model. The holistic learning and inference processes of the entire MixDVAE model are executed through the application of the VI framework, employing a VEM algorithm. This algorithm effectively combines a structured mean-field approximation with amortized inference techniques. MixDVAE exhibits versatility across various application scenarios. It is adeptly applied to a vision task, specifically MOT (Chapter 4), as well as an audio task, focusing on SC-ASS (Chapter 5). The inherent advantage of the MixDVAE model lies in its ability to operate effectively without the need for an extensive amount of annotated data during training. Through pre-training on either a synthetic or natural single-source dataset, DVAE swiftly adjusts to the multi-source context within the MixDVAE framework. The implementation of MixDVAE in MOT showcases its robustness in scenarios involving frequent object occlusions and detection gaps. This model serves as a noteworthy complementary solution to the vision feature-based tracking models within the computer vision community. On the other hand, while the SC-ASS outcomes achieved by MixDVAE may not currently match the performance of the state-of-the-art supervised methods [191, 233], its application in this task demonstrates the versatility of its usage across various scenarios. Furthermore, it represents a significant stride towards weakly supervised and/or unsupervised SC-ASS.

In Chapter 6, we introduced a novel unsupervised SE framework that utilizes dynamical DPGMs to model both speech and noise signals. This model is developed based on a previous approach [22], where the initial step involves pre-training a DVAE model on a clean speech dataset. In [22], the pre-trained DVAE model is subsequently combined with a NMF noise model, employing the VEM algorithm to estimate clean speech from the noisy speech signal. In our work, we propose to replace the NMF noise model with a dynamical DPGM. Our proposed model can be trained in various training/testing configurations (NA, ND and ND + noise adaptation). Specifically, training under the ND configuration allows the model to circumvent the time-consuming VEM iterations, enabling rapid and efficient inference. We have additionally examined the model's performance under unmatched pre-training/testing dataset. The proposed method demonstrates a relatively robust performance with only a slight decline.

In Chapter 7, we employed a novel DVAE architecture known as HiT/LigHT-DVAE for modeling speech data. The HiT/LigHT-DVAE models leverage the robust sequential modeling capabilities of Transformers, seamlessly integrating them into the principled probabilistic modeling framework of DVAE families. These models encompass two types of random variables: a global latent variable $\mathbf{w}$ designed to capture temporally invariant information, and a sequence of latent variables $\mathbf{z}_{1:T}$ intended to encode dynamical information. The special structure of the HiT/LigHT-DVAE models demonstrate robustness to mismatches between the teacher-forcing mode used during training and the generation mode used during testing. Moreover, they achieve superior performance in speech modeling compared to other DVAE models. In short, HiT/LigHT-DVAE models exhibit promising potential for downstream applications in speech processing tasks.

## 8.2 INSIGHTS AND LIMITATIONS

DPGMs indeed constitute a cornerstone of contemporary machine learning research. They offer a versatile framework for understanding complex data distributions and generating new data samples. On the one hand, by harnessing the expressive power and scalability of deep neural networks, various deep generative models, including GANs and recently developed diffusion models discussed in Section 2.1 demonstrate remarkable proficiency in generating high-quality data samples.[1] On the other hand, DPGMs also provide a prominent means of discovering unknown underlying factors of data variations. This aspect enables practical applications such as anomaly detection, data imputation, and feature disentanglement, fostering deeper insights into the underlying structures of complex datasets. The discovered data representations learned from large amounts of data can also benefit plenty of downstream tasks.

In this dissertation, our primary focus has been on employing the capabilities of DPGMs to address intricate scientific and engineering challenges. It is worth noting that while supervised learning methods have achieved state-of-the-art performance across various engineering domains and scientific disciplines, the widespread implementation of artificial intelligence systems in real-life production settings remains a challenging task with significant hurdles yet to overcome. Firstly, annotating data in specific domains often requires experts with skilled domain knowledge. For instance, in health care, obtaining the data labels for predicting organ failure from physiological signals usually necessitates the involvement of experienced clinicians. Therefore, data annotation is indeed a very expensive endeavor. Secondly, the application of AI systems to real-life scenarios necessitates

---

[1] See https://openai.com/dall-e-3 and https://huggingface.co/spaces/facebook/MusicGen for high quality text-to-image and text-to-music generation demos.

the reliability and interpretability of the system. This entails ensuring that the system's decisions can be trusted and understood to some extent, particularly in critical domains such as healthcare, finance, or education, where transparency and accountability are of paramount importance. Additionally, interpretability is crucial for gaining insights into a model's behavior, correcting model biases and disfunctions, and facilitating collaboration between AI systems and human experts.

In this PhD work, we investigated a novel framework for unsupervised or weakly-supervised learning. In general, our approach involves initially pre-training a DPGM using either natural or synthetic signals to imbue it with prior knowledge about the complex data distribution. Subsequently, the pre-trained DPGM is integrated into a well-defined LVM to address practical problems. Notably, our method prioritizes interpretability, as all factors are defined within a principled probabilistic framework. Additionally, it proves to be data-frugal, as it does not rely on vast amounts of paired annotated data for training. Experimental results demonstrate the promising potential of this learning framework. However, it is important to acknowledge several limitations of this approach, particularly in its current stage of development. First, complete reliance on unsupervised or weakly-supervised learning often results in subpar performance compared to fully-supervised learning setups. Exploring a semi-supervised learning framework, where model parameters are learned with a small amount of labeled data, could serve as a potential future research direction to further enhance model performance. Second, the solutions derived from this approach using free-form VI and EM-based optimisation methods can be inherently time-consuming. In Chapter 6, we have proposed a method to address this challenge by replacing the statistical NMF noise model with a DNN-based noise model for speech enhancement and optimizing the model parameters using the SGD method. While this approach effectively accelerates inference time while maintaining comparable performance, the contribution of the DNN to enhancing model performance is not as significant as expected, despite its increasing expressiveness. We speculate that this may be attributed to the limitations of unsupervised learning settings in providing effective training signals. This observation also underscores the necessity for the development of semi-supervised training frameworks.

At the end of this section, I would like to discuss some of the observations, reflections and concerns from my personal perspective as a young researcher in the machine learning field.

In the first place, concerning the main method that I have explored over the past three years, I find it to be an elegant and promising approach rooted in the principled Bayesian philosophy, which, to some extent, guarantees the interpretability and mathematical rigorous. However, when we are trying to integrate DNNs into the elegant PGM framework, I always sense that the theories of PGMs and VI presented in Chapter 2, Section 2.1, Section 2.2, and Section 2.3 are not sufficient. Two observations are given as follows. The first one is that, although this two stage un-/weakly supervised learning paradigm achieves satisfying empirical results, the underlying reasons for its success remains unclear. This necessitates further and more rigorous investigations. Actually, the robustness and generalization ability of the pre-trained DPGM significantly influence the performance of this method. During the second integration/fine-tuning stage, we would like the pre-trained model to retain the learned patterns of the single source in the case of source separation (of the clean speech in the case of SE), while remaining robust to the noisy input. Nevertheless, it seems that we do not have a mathematical tool to explain and measure this property. Moreover, I think this learning paradigm is closely related to current trend of employing foundation models, which involves first pre-training a model over a large dataset and then adapting it for specific tasks. In my personal opinion, it is imperative to establish a rigorous and more profound theory to aid in our understanding of the empirical phenomena. Another observation comes from the fact that, despite the significant success of deep LVMs such as VAEs and DVAEs across various application scenarios, our understanding of the latent space of these models remains limited. An important research direction in the field of machine learning, known as representation learning [15], is dedicated to addressing this question. Nonetheless, similarly, no principled and unified theories have been established yet.

Another aspect that I would like to highlight and discuss originates from a more general and high-level point of view. Throughout my PhD studies, the ongoing discussions and debates between theory versus practice and academia versus industry have never ceased. And these questions have continuously weighted on my mind from the beginning to the end. After completing the draft of most parts of this PhD thesis, I started to think about the "true problems" within the machine learning and artificial intelligence field. As Vladimir N. Vapnik mentioned in the preface of his book *The Nature of Statistical Learning Theory* [206]:

> Understanding the nature of the problem is extremely important because it leads to searching in the right direction for results and prevents searching in wrong directions.

To understand the nature of the problem, the first questions that I start to ask are: What are the definitions of machine learning and artificial intelligence? And what precisely constitutes the research object in this field? An inspiring opinion shared by Josh Tenenbaum at NeurIPS 2021, and quoted by Kevin P. Murphy in the Introduction chapter of his book *Probabilistic Machine Learning: Advanced Topics* [141] is that:

> Intelligence is not just about pattern recognition and function approximation. It's about modeling the world.

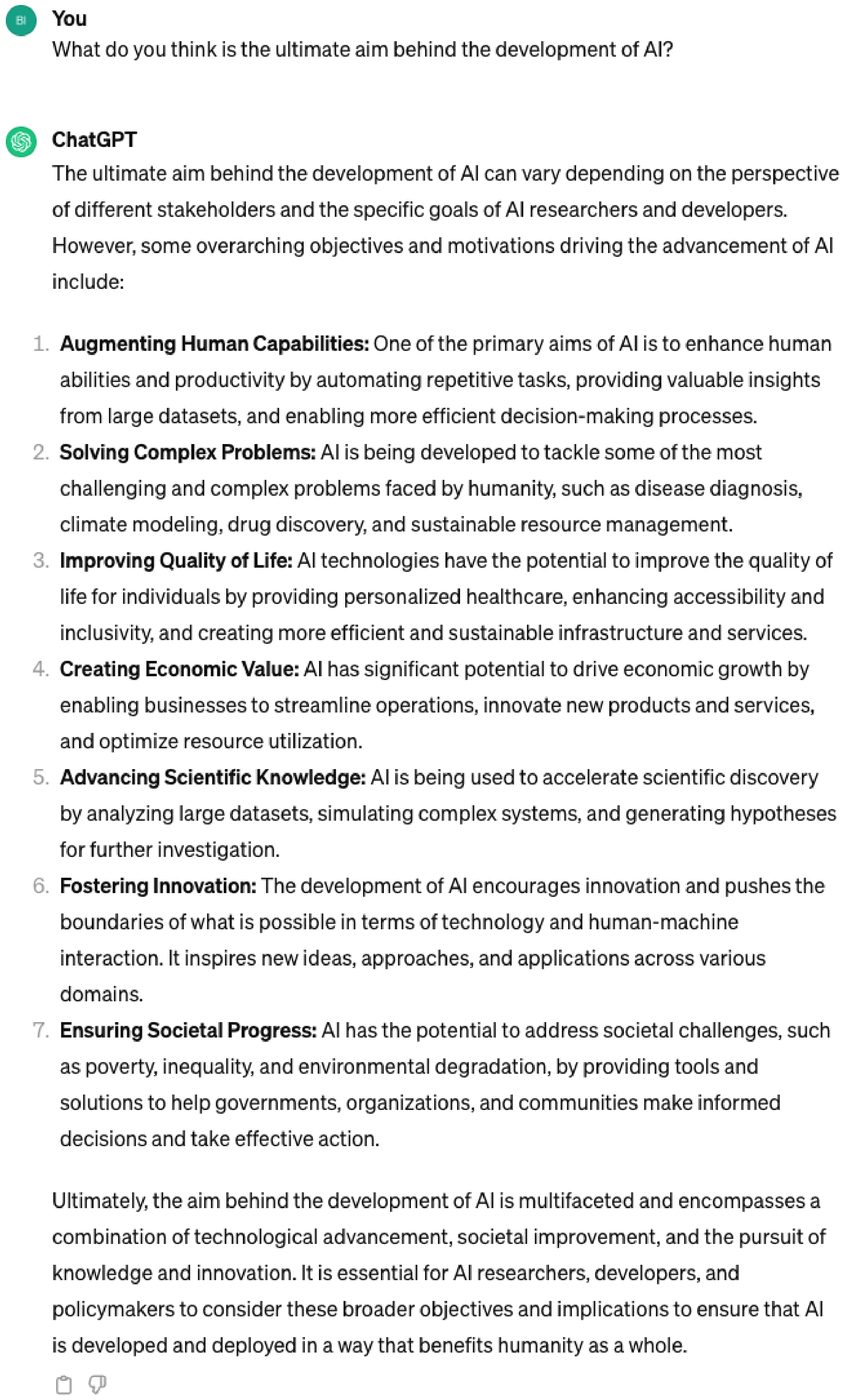
If we interpret the progression of natural sciences, such as physics, chemistry, biology, etc., as humanity's effort to model the world through quantitative analysis and mathematical tools, machine learning and artificial intelligence can be understood as endeavors to model and quantify the human intelligence and learning process, and subsequently translate these models into mathematical formulations and computer algorithms. We should note that this is only one perspective to interpret machine learning and artificial intelligence. And this perspective is closer to the computational cognitive science. Another perspective, from a more practical standpoint, is that we endeavor to model the world, uncover patterns and dependencies, and/or achieve specific application goals, by harnessing the formidable computational capability of computers and access to (large) datasets. The research field of machine learning and artificial intelligence is so vast that it is not only concerned with well-designed models, architectures and algorithms that achieve superior performance on specific tasks, but also concerned with fundamental theories and principles. However, it seems that the recent explosive growth of deep learning methods in several application fields such as computer vision, natural language processing and audio processing, alongside the discovery of scaling laws for neural language models [102] and development of foundation models [28], has raised challenges about traditional statistical learning theories. And our theoretical understanding seems to lag far behind the advancements seen in practical applications.

## 8.3 TOWARDS A BROADER DISCUSSION

Reflecting on history, from Alain Turing's initial conception of "learning machines" to the introduction of the term "artificial intelligence" at the Dartmouth Conference in 1956, the field of AI has witnessed an extraordinary evolution over the past seven decades. Generations after generations of brilliant researchers have dedicated their intelligence, diligence, and passion to propel this research trajectory forward. Humanity's power to achieve such remarkable strides within such a relatively brief period never ceases to astonish me. Undoubtedly, the rise of deep learning, characterized by the utilization of deep neural networks with multiple non-linear layers, has sparked a revolution across various domains of AI research, including computer vision, natural language processing, and speech recognition. Moreover, the advent of large-scale pre-trained models such as BERT [45] and GPT [146], which are also recognized as foundation models [28], has brought us into an entirely new realm of advancement. The launch of the ChatGPT product alongside the GPT-4 model [146] instilled a growing belief that the foundation models truly possess a remarkable semblance of human intelligence. Nevertheless, it is important to acknowledge that our comprehension and mastery of these models still significantly trail the pace of their development.

In crafting this dissertation, one critical question has persistently engaged my thoughts: What is the ultimate aim behind the development of AI? Through numerous conversations with colleagues and friends, and after thoughtful reflection, I have arrived at an insight grounded in my present understanding. If AI is fated to revolutionize societal norms, refine how we think, and transform our cultural landscapes, my aspiration is that it promotes deeper empathy and understanding among individuals while mitigating tendencies towards arrogance, prejudice, and exploitation. This would enable people to forge stronger and more meaningful connections with one another. Moreover, I envision AI liberating individuals from tedious work, thereby affording them greater opportunities to explore their creative potentials with newfound time and energy. Nonetheless, achieving this ultimate objective entails a significant journey ahead. It necessitates that each individual fully leverage their wisdom, creativity, and rational thinking to sculpt the forthcoming future landscape. The required efforts are twofold. On one side, this asks the researchers and developers of AI systems to incorporate ethical considerations and societal impacts at the heart of the AI's development, governance, and deployment. On the other side, it calls upon every individual, as end-users of AI technologies, to cultivate our critical thinking abilities and figure out the most effective ways to deploy AI, thereby ensuring we master these tools rather than become their slaves. Interestingly, I have also discussed this question with ChatGPT. Its answers are reported in Figure 8.1, 8.2 and 8.3. I have also asked ChatGPT to generate a picture of how people will live with AI in the future. The generated picture is shown in Figure 8.4. In this picture, one human-like robot pushes a stroller while another human-like robot carries a basket of vegetables. A couple with a child walk casually alongside them. It is a very beautiful picture, and just as ChatGPT describes it, "it showcases a harmonious coexistence between humans and robots in a technologically advanced, sustainable urban environment." This picture makes me believe that AI can indeed make great contributions to gender equality by reducing housework burdens and redistributing roles and tasks traditionally influenced by gender norms.

In conclusion, I deeply believe that humanity will eventually master the art of leveraging AI, just as we have learned to harness the power of machines. As ChatGPT suggests, the ideal path forward lies in fostering a collaborative relationship between humans and AI, rather than a competitive one. This partnership holds the promise of unlocking unprecedented potential and advancing our collective capabilities.



**You**

What do you think is the ultimate aim behind the development of AI?

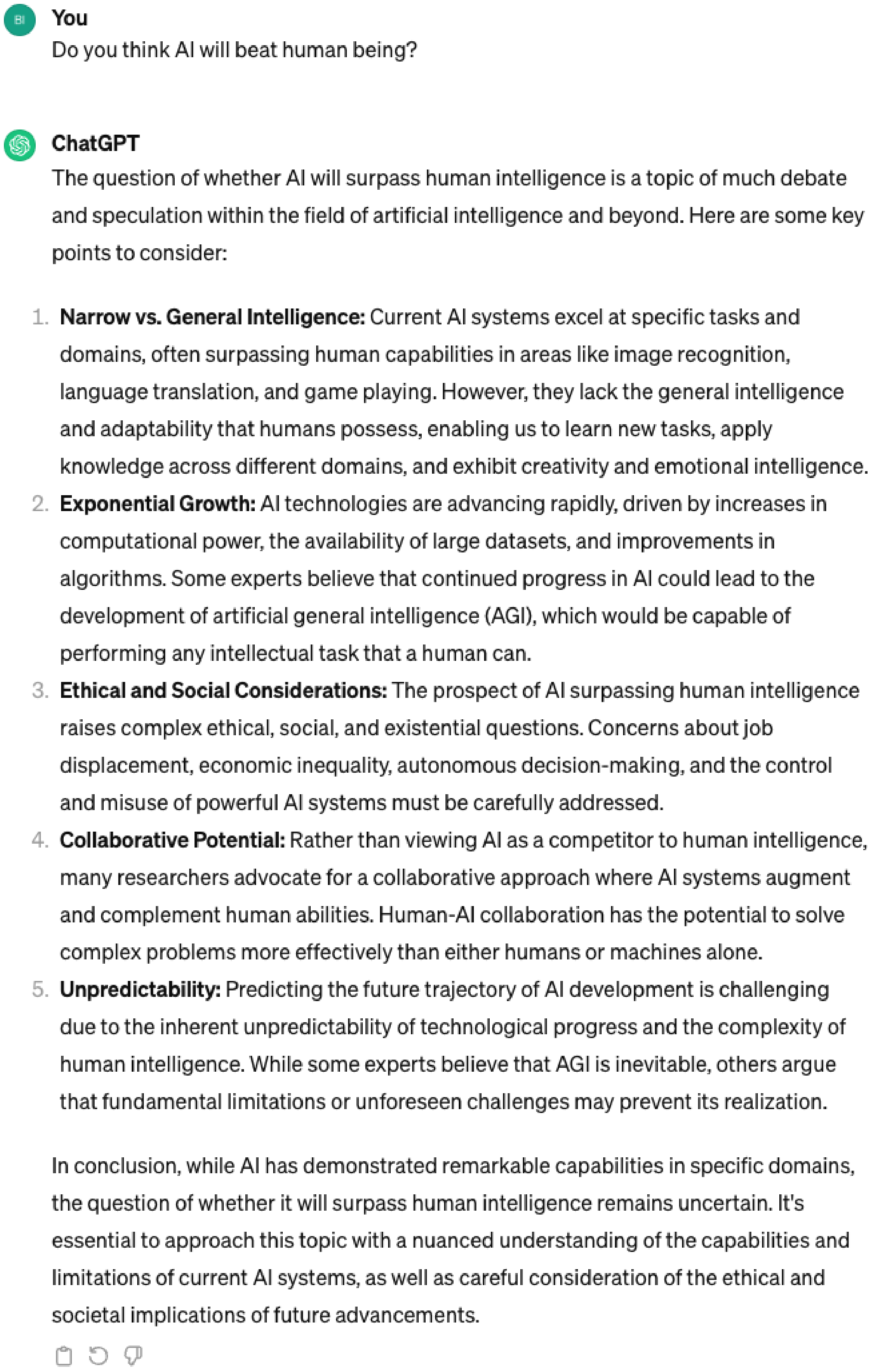
**ChatGPT**

The ultimate aim behind the development of AI can vary depending on the perspective of different stakeholders and the specific goals of AI researchers and developers. However, some overarching objectives and motivations driving the advancement of AI include:

1. **Augmenting Human Capabilities:** One of the primary aims of AI is to enhance human abilities and productivity by automating repetitive tasks, providing valuable insights from large datasets, and enabling more efficient decision-making processes.
2. **Solving Complex Problems:** AI is being developed to tackle some of the most challenging and complex problems faced by humanity, such as disease diagnosis, climate modeling, drug discovery, and sustainable resource management.
3. **Improving Quality of Life:** AI technologies have the potential to improve the quality of life for individuals by providing personalized healthcare, enhancing accessibility and inclusivity, and creating more efficient and sustainable infrastructure and services.
4. **Creating Economic Value:** AI has significant potential to drive economic growth by enabling businesses to streamline operations, innovate new products and services, and optimize resource utilization.
5. **Advancing Scientific Knowledge:** AI is being used to accelerate scientific discovery by analyzing large datasets, simulating complex systems, and generating hypotheses for further investigation.
6. **Fostering Innovation:** The development of AI encourages innovation and pushes the boundaries of what is possible in terms of technology and human-machine interaction. It inspires new ideas, approaches, and applications across various domains.
7. **Ensuring Societal Progress:** AI has the potential to address societal challenges, such as poverty, inequality, and environmental degradation, by providing tools and solutions to help governments, organizations, and communities make informed decisions and take effective action.

Ultimately, the aim behind the development of AI is multifaceted and encompasses a combination of technological advancement, societal improvement, and the pursuit of knowledge and innovation. It is essential for AI researchers, developers, and policymakers to consider these broader objectives and implications to ensure that AI is developed and deployed in a way that benefits humanity as a whole.

**Figure 8.1**: Conversation with ChatGPT about the ultimate aim behind the development of AI. This conversation is generated by GPT-3.5.



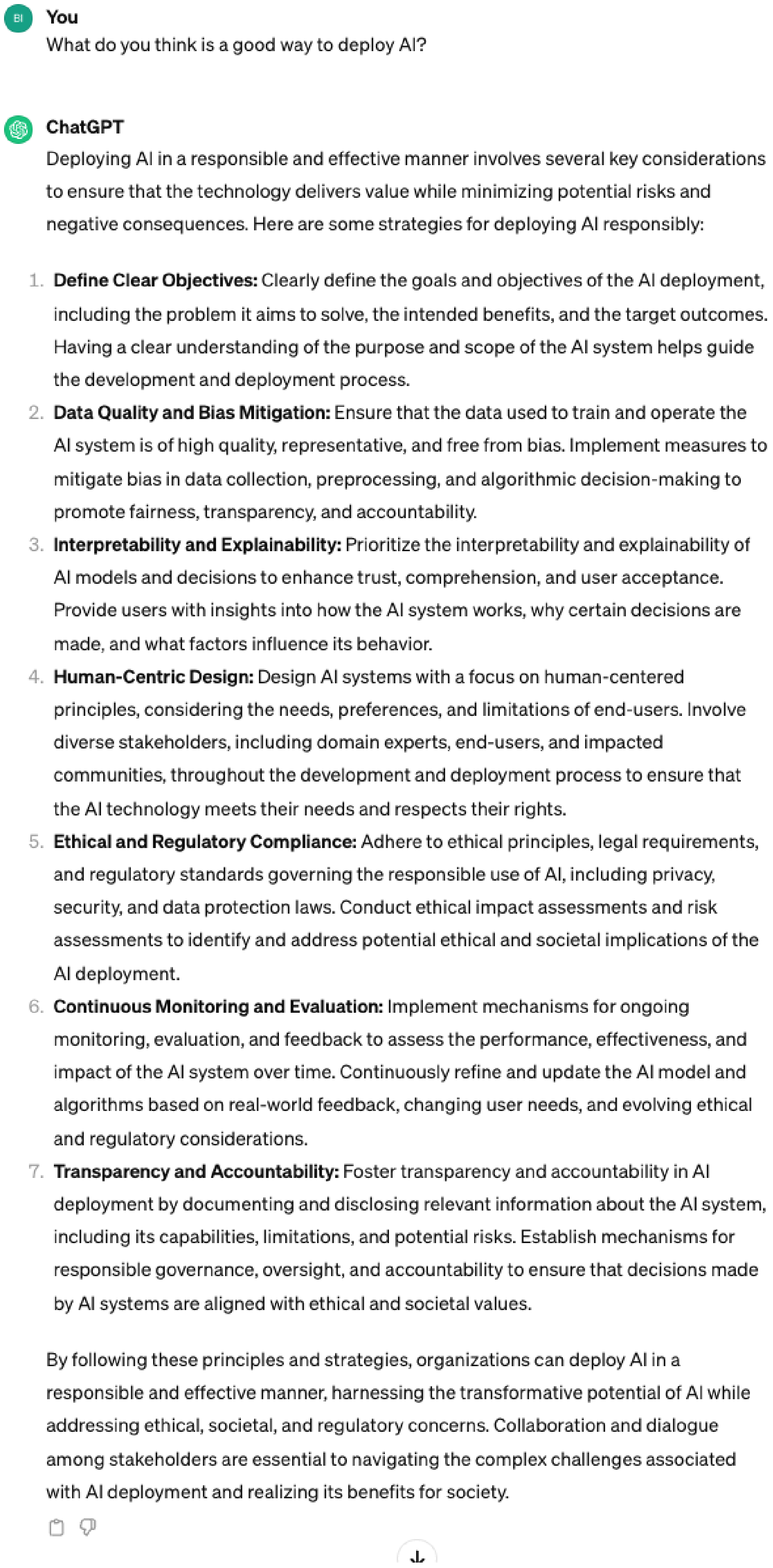
**You**
Do you think AI will beat human being?

**ChatGPT**
The question of whether AI will surpass human intelligence is a topic of much debate and speculation within the field of artificial intelligence and beyond. Here are some key points to consider:

1. **Narrow vs. General Intelligence:** Current AI systems excel at specific tasks and domains, often surpassing human capabilities in areas like image recognition, language translation, and game playing. However, they lack the general intelligence and adaptability that humans possess, enabling us to learn new tasks, apply knowledge across different domains, and exhibit creativity and emotional intelligence.
2. **Exponential Growth:** AI technologies are advancing rapidly, driven by increases in computational power, the availability of large datasets, and improvements in algorithms. Some experts believe that continued progress in AI could lead to the development of artificial general intelligence (AGI), which would be capable of performing any intellectual task that a human can.
3. **Ethical and Social Considerations:** The prospect of AI surpassing human intelligence raises complex ethical, social, and existential questions. Concerns about job displacement, economic inequality, autonomous decision-making, and the control and misuse of powerful AI systems must be carefully addressed.
4. **Collaborative Potential:** Rather than viewing AI as a competitor to human intelligence, many researchers advocate for a collaborative approach where AI systems augment and complement human abilities. Human-AI collaboration has the potential to solve complex problems more effectively than either humans or machines alone.
5. **Unpredictability:** Predicting the future trajectory of AI development is challenging due to the inherent unpredictability of technological progress and the complexity of human intelligence. While some experts believe that AGI is inevitable, others argue that fundamental limitations or unforeseen challenges may prevent its realization.

In conclusion, while AI has demonstrated remarkable capabilities in specific domains, the question of whether it will surpass human intelligence remains uncertain. It's essential to approach this topic with a nuanced understanding of the capabilities and limitations of current AI systems, as well as careful consideration of the ethical and societal implications of future advancements.

**Figure 8.2**: Conversation with ChatGPT about the relationship between humans and AI. This conversation is generated by GPT-3.5.



**You**

What do you think is a good way to deploy AI?

**ChatGPT**

Deploying AI in a responsible and effective manner involves several key considerations to ensure that the technology delivers value while minimizing potential risks and negative consequences. Here are some strategies for deploying AI responsibly:

1. **Define Clear Objectives:** Clearly define the goals and objectives of the AI deployment, including the problem it aims to solve, the intended benefits, and the target outcomes. Having a clear understanding of the purpose and scope of the AI system helps guide the development and deployment process.
2. **Data Quality and Bias Mitigation:** Ensure that the data used to train and operate the AI system is of high quality, representative, and free from bias. Implement measures to mitigate bias in data collection, preprocessing, and algorithmic decision-making to promote fairness, transparency, and accountability.
3. **Interpretability and Explainability:** Prioritize the interpretability and explainability of AI models and decisions to enhance trust, comprehension, and user acceptance. Provide users with insights into how the AI system works, why certain decisions are made, and what factors influence its behavior.
4. **Human-Centric Design:** Design AI systems with a focus on human-centered principles, considering the needs, preferences, and limitations of end-users. Involve diverse stakeholders, including domain experts, end-users, and impacted communities, throughout the development and deployment process to ensure that the AI technology meets their needs and respects their rights.
5. **Ethical and Regulatory Compliance:** Adhere to ethical principles, legal requirements, and regulatory standards governing the responsible use of AI, including privacy, security, and data protection laws. Conduct ethical impact assessments and risk assessments to identify and address potential ethical and societal implications of the AI deployment.
6. **Continuous Monitoring and Evaluation:** Implement mechanisms for ongoing monitoring, evaluation, and feedback to assess the performance, effectiveness, and impact of the AI system over time. Continuously refine and update the AI model and algorithms based on real-world feedback, changing user needs, and evolving ethical and regulatory considerations.
7. **Transparency and Accountability:** Foster transparency and accountability in AI deployment by documenting and disclosing relevant information about the AI system, including its capabilities, limitations, and potential risks. Establish mechanisms for responsible governance, oversight, and accountability to ensure that decisions made by AI systems are aligned with ethical and societal values.

By following these principles and strategies, organizations can deploy AI in a responsible and effective manner, harnessing the transformative potential of AI while addressing ethical, societal, and regulatory concerns. Collaboration and dialogue among stakeholders are essential to navigating the complex challenges associated with AI deployment and realizing its benefits for society.

**Figure 8.3**: Conversation with ChatGPT about what is a good way to deploy AI. This conversation is generated by GPT-3.5.

**Figure 8.4**: A picture of how people will live with AI in the future generated by DALL·E. The prompt used to generate this picture is “Please draw me a picture of how people will live with AI in the future.”

# CHAPTER A

# APPENDIX

## A.1 MIXTURE OF DVAES FOR MULTI-SOURCE TRAJECTORY MODELING AND SEPARATION

### A.1.1 MIXDVAE ALGORITHM CALCULATION DETAILS

**E-S Step**

Here we detail the calculation of the posterior distribution $q_{\phi_\mathbf{s}}(\mathbf{s}|\mathbf{o})$. Using Equation 3.2, the first expectation term in Equation 3.14 can be developed as:

$$\begin{aligned}\mathbb{E}_{q_{\phi_\mathbf{w}}(\mathbf{w}|\mathbf{o})}\big[\log p_{\theta_\mathbf{o}}(\mathbf{o}|\mathbf{w},\mathbf{s})\big] &= \mathbb{E}_{q_{\phi_\mathbf{w}}(\mathbf{w}|\mathbf{o})}\Bigg[\sum_{t=1}^{T}\sum_{k=1}^{K_t}\log p_{\theta_\mathbf{o}}(\mathbf{o}_{tk}|w_{tk},\mathbf{s}_{t,1:N})\Bigg] \\ &= \sum_{t=1}^{T}\sum_{k=1}^{K_t}\mathbb{E}_{q_{\phi_\mathbf{w}}(w_{tk}|\mathbf{o}_{tk})}\big[\log p_{\theta_\mathbf{o}}(\mathbf{o}_{tk}|w_{tk},\mathbf{s}_{t,1:N})\big]. \qquad (A.1)\end{aligned}$$

Since for any pair $(t,k)$, the assignment variable $w_{tk}$ follows a discrete posterior distribution, we can denote its values by

$$q_{\phi_\mathbf{w}}(w_{tk}=n|\mathbf{o}_{tk}) = \eta_{tkn}, \qquad (A.2)$$

which will be calculated later in the E-W Step. With this notation, we have:

$$\mathbb{E}_{q_{\phi_{\mathbf{w}}}(\mathbf{w}|\mathbf{o})}\big[\log p_{\theta_{\mathbf{o}}}(\mathbf{o}|\mathbf{w},\mathbf{s})\big] = \sum_{t=1}^{T}\sum_{k=1}^{K_t}\sum_{n=1}^{N}\eta_{tkn}\log p_{\theta_{\mathbf{o}}}(\mathbf{o}_{tk}|w_{tk}=n,\mathbf{s}_{tn}). \tag{A.3}$$

The second expectation in Equation 3.14 cannot be computed analytically as a distribution on s because of the non-linearity in the decoder and in the encoder. In order to avoid a tedious sampling procedure and obtain a computationally efficient solution, we further approximate this term by assuming $q_{\phi_{\mathbf{z}}}(\mathbf{z}|\mathbf{s}) \approx q_{\phi_{\mathbf{z}}}(\mathbf{z}|\mathbf{s}=\mathbf{m}^{(i-1)})$, where $\mathbf{m}^{(i-1)}$ is the mean value of the posterior distribution of s estimated at the previous iteration. By using this approximation, the term $\mathbb{E}_{q_{\phi_{\mathbf{z}}}(\mathbf{z}|\mathbf{s})}\big[\log q_{\phi_{\mathbf{z}}}(\mathbf{z}|\mathbf{s})\big]$ is now considered as a constant.

In addition, we observe that the second term of Equation 3.14 can be rewritten as:

$$\mathbb{E}_{q_{\phi_{\mathbf{z}}}(\mathbf{z}|\mathbf{s})}\big[\log p_{\theta_{\mathbf{sz}}}(\mathbf{s},\mathbf{z})\big] = \sum_{n=1}^{N}\mathbb{E}_{q_{\phi_{\mathbf{z}}}(\mathbf{z}_{:,n}|\mathbf{m}_{:,n}^{(i-1)})}\big[\log p_{\theta_{\mathbf{sz}}}(\mathbf{s}_{:,n},\mathbf{z}_{:,n})\big], \tag{A.4}$$

since both the DVAE joint distribution and posterior distribution factorise over the sources, as formalized in Equation 3.6 and Equation 3.8. As a consequence, the posterior distribution of s factorises over the sources too:

$$q_{\phi_{\mathbf{s}}}(\mathbf{s}|\mathbf{o}) = \prod_{n=1}^{N} q_{\phi_{\mathbf{s}}}(\mathbf{s}_{:,n}|\mathbf{o}), \tag{A.5}$$

and therefore:

$$q_{\phi_{\mathbf{s}}}(\mathbf{s}_{:,n}|\mathbf{o}) \propto \exp\Big(\mathbb{E}_{q_{\phi_{\mathbf{z}}}(\mathbf{z}_{:,n}|\mathbf{m}_{:,n}^{(i-1)})}\big[\log p_{\theta_{\mathbf{sz}}}(\mathbf{s}_{:,n},\mathbf{z}_{:,n})\big]\Big) \prod_{t=1}^{T}\prod_{k=1}^{K_t}\exp\big(\eta_{tkn}\log p_{\theta_{\mathbf{o}}}(\mathbf{o}_{tk}|w_{tk}=n,\mathbf{s}_{tn})\big). \tag{A.6}$$

In the above equation, the expectation term cannot be calculated in closed form. As usually done in the DVAE methodology, it is thus replaced by a Monte Carlo estimate using sampled sequences drawn from the DVAE inference model. Let us denote by $\mathbf{z}_{:,n}^{(i)} \sim q_{\phi_{\mathbf{z}}}(\mathbf{z}_{:,n}|\mathbf{m}_{:,n}^{(i-1)})$ such a sampled sequence. In the present work, we use single point estimate, thus obtaining:

$$\begin{aligned}
q_{\phi_{\mathbf{s}}}(\mathbf{s}_{:,n}|\mathbf{o}) &\propto p_{\theta_{\mathbf{sz}}}(\mathbf{s}_{:,n}, \mathbf{z}^{(i)}_{:,n}) \prod_{t=1}^{T}\prod_{k=1}^{K_t} \exp\Big(\eta_{tkn} \log p_{\theta_{\mathbf{o}}}(\mathbf{o}_{tk}|w_{tk}=n, \mathbf{s}_{tn})\Big) \\
&\propto \prod_{t=1}^{T}\Big(p_{\theta_{\mathbf{s}}}(\mathbf{s}_{tn}|\mathbf{s}_{1:t-1,n}, \mathbf{z}^{(i)}_{1:t,n}) p_{\theta_{\mathbf{z}}}(\mathbf{z}^{(i)}_{tn}|\mathbf{s}_{1:t-1,n}, \mathbf{z}^{(i)}_{1:t-1,n}) \\
&\qquad \prod_{k=1}^{K_t} \exp\Big(\eta_{tkn} \log p_{\theta_{\mathbf{o}}}(\mathbf{o}_{tk}|w_{tk}=n, \mathbf{s}_{tn})\Big)\Big).
\end{aligned} \tag{A.7}$$

We observe that the $t$-th element of the previous factorisation is a distribution over $\mathbf{s}_{tn}$ conditioned by $\mathbf{s}_{1:t-1,n}$. As for $q_{\phi_{\mathbf{z}}}(\mathbf{z}_{:,n}|\mathbf{s}_{:,n})$, the dependency with $\mathbf{s}_{1:t-1,n}$ is non-linear and therefore would impede to obtain a computationally efficient closed-form solution. In the same attempt of avoiding costly sampling strategies, we approximate the previous expression replacing $\mathbf{s}_{1:t-1,n}$ with $\mathbf{s}^{(i)}_{1:t-1,n}$, obtaining:

$$q_{\phi_{\mathbf{s}}}(\mathbf{s}_{:,n}|\mathbf{o}) \approx \prod_{t=1}^{T} q_{\phi_{\mathbf{s}}}(\mathbf{s}_{tn}|\mathbf{s}^{(i)}_{1:t-1,n}, \mathbf{o}), \tag{A.8}$$

with

$$q_{\phi_{\mathbf{s}}}(\mathbf{s}_{tn}|\mathbf{s}^{(i)}_{1:t-1,n}, \mathbf{o}) \propto p_{\theta_{\mathbf{s}}}(\mathbf{s}_{tn}|\mathbf{s}^{(i)}_{1:t-1,n}, \mathbf{z}^{(i)}_{1:t,n}) \prod_{k=1}^{K_t} \exp\Big(\eta_{tkn} \log p_{\theta_{\mathbf{o}}}(\mathbf{o}_{tk}|w_{tk}=n, \mathbf{s}_{tn})\Big), \tag{A.9}$$

since the term $p_{\theta_{\mathbf{z}}}(\mathbf{z}^{(i)}_{tn}|\mathbf{s}^{(i)}_{1:t-1,n}, \mathbf{z}^{(i)}_{1:t-1,n})$ becomes a constant.

Another interesting consequence of sampling $\mathbf{s}_{1:t-1,n}$ is that the dependency with the future observations of $q_{\phi_{\mathbf{s}}}(\mathbf{s}_{tn}|\mathbf{s}^{(i)}_{1:t-1,n}, \mathbf{o})$ disappears. Indeed, since we are sampling *at every time step*, the future posterior distributions $q_{\phi_{\mathbf{s}}}(\mathbf{s}_{t+k,n}|\mathbf{s}^{(i)}_{1:t+k-1,n}, \mathbf{o})$ do not depend on $\mathbf{s}_{tn}$, and therefore the posterior distribution of $\mathbf{s}_{tn}$ will not depend on the future observations.

The two distributions in the above equation are Gaussian distributions defined in (2.35), and (3.3). Therefore, it can be shown that the variational posterior distribution of $\mathbf{s}_{tn}$ is a Gaussian distribution: $q_{\phi_{\mathbf{s}}}(\mathbf{s}_{tn}|\mathbf{s}^{(i)}_{1:t-1,n}, \mathbf{o}) = \mathcal{N}(\mathbf{s}_{tn}; \mathbf{m}_{tn}, \mathbf{V}_{tn})$ with covariance matrix and mean vector provided in (3.18) and (3.19) respectively, and recalled here for completeness:

$$\mathbf{V}_{tn} = \Big( \sum_{k=1}^{K_t} \eta_{tkn} \mathbf{\Phi}_{tk}^{-1} + \mathrm{diag}(\boldsymbol{v}^{(i)}_{\theta_\mathbf{s},tn})^{-1} \Big)^{-1}, \tag{A.10}$$

$$\mathbf{m}_{tn} = \mathbf{V}_{tn} \Big( \sum_{k=1}^{K_t} \eta_{tkn} \mathbf{\Phi}_{tk}^{-1} \mathbf{o}_{tk} + \mathrm{diag}(\boldsymbol{v}^{(i)}_{\theta_\mathbf{s},tn})^{-1} \boldsymbol{\mu}^{(i)}_{\theta_\mathbf{s},tn} \Big), \tag{A.11}$$

where $\boldsymbol{v}^{(i)}_{\theta_\mathbf{s},tn}$ and $\boldsymbol{\mu}^{(i)}_{\theta_\mathbf{s},tn}$ are simplified notations for $\boldsymbol{v}_{\theta_\mathbf{s}}(\mathbf{s}^{(i)}_{1:t-1,n}, \mathbf{z}^{(i)}_{1:t,n})$ and $\boldsymbol{\mu}_{\theta_\mathbf{s}}(\mathbf{s}^{(i)}_{1:t-1,n}, \mathbf{z}^{(i)}_{1:t,n})$ respectively, denoting the variance and mean vector estimated by the DVAE for source $n$ at time frame $t$.

### E-Z Step

Here we detail the calculation of the ELBO term Equation 3.20.

$$\begin{aligned}
\mathcal{L}(\theta_\mathbf{s}, \theta_\mathbf{z}, \phi_\mathbf{z}; \mathbf{o}) &= \mathbb{E}_{q_{\phi_\mathbf{s}}(\mathbf{s}|\mathbf{o})} \Big[ \mathbb{E}_{q_{\phi_\mathbf{z}}(\mathbf{z}|\mathbf{s})} \big[ \log p_{\theta_{\mathbf{sz}}}(\mathbf{s}, \mathbf{z}) - \log q_{\phi_\mathbf{z}}(\mathbf{z}|\mathbf{s}) \big] \Big] \\
&= \mathbb{E}_{\prod_{n=1}^{N} q_{\phi_\mathbf{s}}(\mathbf{s}_{:,n}|\mathbf{o})} \Big[ \mathbb{E}_{\prod_{n=1}^{N} q_{\phi_\mathbf{z}}(\mathbf{z}_{:,n}|\mathbf{s}_{:,n})} \Big[ \sum_{n=1}^{N} \log p_{\theta_{\mathbf{sz}}}(\mathbf{s}_{:,n}, \mathbf{z}_{:,n}) \Big] \\
&- \mathbb{E}_{\prod_{n=1}^{N} q_{\phi_\mathbf{z}}(\mathbf{z}_{:,n}|\mathbf{s}_{:,n})} \Big[ \sum_{n=1}^{N} \log q_{\phi_\mathbf{z}}(\mathbf{z}_{:,n}|\mathbf{s}_{:,n}) \Big] \Big] \\
&= \sum_{n=1}^{N} \mathbb{E}_{q_{\phi_\mathbf{s}}(\mathbf{s}_{:,n}|\mathbf{o})} \Big[ \mathbb{E}_{q_{\phi_\mathbf{z}}(\mathbf{z}_{:,n}|\mathbf{s}_{:,n})} \big[ \log p_{\theta_{\mathbf{sz}}}(\mathbf{s}_{:,n}, \mathbf{z}_{:,n}) \big] - \mathbb{E}_{q_{\phi_\mathbf{z}}(\mathbf{z}_{:,n}|\mathbf{s}_{:,n})} \big[ \log q_{\phi_\mathbf{z}}(\mathbf{z}_{:,n}|\mathbf{s}_{:,n}) \big] \Big] \\
&= \sum_{n=1}^{N} \mathcal{L}_n(\theta_\mathbf{s}, \theta_\mathbf{z}, \phi_\mathbf{z}; \mathbf{o}),
\end{aligned} \tag{A.12}$$

with

$$\mathcal{L}_n(\theta_\mathbf{s}, \theta_\mathbf{z}, \phi_\mathbf{z}; \mathbf{o}) = \mathbb{E}_{q_{\phi_\mathbf{s}}(\mathbf{s}_{:,n}|\mathbf{o})} \Big[ \mathbb{E}_{q_{\phi_\mathbf{z}}(\mathbf{z}_{:,n}|\mathbf{s}_{:,n})} \big[ \log p_{\theta_{\mathbf{sz}}}(\mathbf{s}_{:,n}, \mathbf{z}_{:,n}) \big] - \mathbb{E}_{q_{\phi_\mathbf{z}}(\mathbf{z}_{:,n}|\mathbf{s}_{:,n})} \big[ \log q_{\phi_\mathbf{z}}(\mathbf{z}_{:,n}|\mathbf{s}_{:,n}) \big] \Big]. \tag{A.13}$$

### E-W Step

Here we detail the calculation of the posterior distribution $q_{\phi_\mathbf{w}}(\mathbf{w}|\mathbf{o})$. Applying the optimal update equation Equation 2.53 to $\mathbf{w}$, we have:

$$q_{\phi_\mathbf{w}}(\mathbf{w}|\mathbf{o}) \propto \exp \Big( \mathbb{E}_{q_{\phi_\mathbf{s}}(\mathbf{s}|\mathbf{o}) q_{\phi_\mathbf{z}}(\mathbf{z}|\mathbf{s})} \big[ \log p_\theta(\mathbf{o}, \mathbf{w}, \mathbf{s}, \mathbf{z}) \big] \Big). \tag{A.14}$$

Using Equation 3.1, we derive:

$$q_{\phi_{\mathbf{w}}}(\mathbf{w}|\mathbf{o}) \propto p_{\theta_{\mathbf{w}}}(\mathbf{w}) \exp\Big(\mathbb{E}_{q_{\phi_{\mathbf{s}}}(\mathbf{s}|\mathbf{o})}\big[\log p_{\theta_{\mathbf{o}}}(\mathbf{o}|\mathbf{w},\mathbf{s})\big]\Big). \tag{A.15}$$

Using Equation 3.2, the expectation term can be developed as:[1]

$$\begin{aligned}\mathbb{E}_{q_{\phi_{\mathbf{s}}}(\mathbf{s}|\mathbf{o})}\big[\log p_{\theta_{\mathbf{o}}}(\mathbf{o}|\mathbf{w},\mathbf{s})\big] &= \mathbb{E}_{q_{\phi_{\mathbf{s}}}(\mathbf{s}|\mathbf{o})}\Big[\sum_{t=1}^{T}\sum_{k=1}^{K_t}\log p_{\theta_{\mathbf{o}}}(\mathbf{o}_{tk}|w_{tk},\mathbf{s}_{t,:})\Big] \\ &= \sum_{t=1}^{T}\sum_{k=1}^{K_t}\mathbb{E}_{q_{\phi_{\mathbf{s}}}(\mathbf{s}_{t,:}|\mathbf{o})}\big[\log p_{\theta_{\mathbf{o}}}(\mathbf{o}_{tk}|w_{tk},\mathbf{s}_{t,:})\big].\end{aligned} \tag{A.16}$$

Combining Equation 3.4 and the previous result, we have:

$$q_{\phi_{\mathbf{w}}}(\mathbf{w}|\mathbf{o}) \propto \prod_{t=1}^{T}\prod_{k=1}^{K_t} p_{\theta_{\mathbf{w}}}(w_{tk}) \exp\Big(\mathbb{E}_{q_{\phi_{\mathbf{s}}}(\mathbf{s}_{t,:}|\mathbf{o})}\big[\log p_{\theta_{\mathbf{o}}}(\mathbf{o}_{tk}|w_{tk},\mathbf{s}_{t,:})\big]\Big), \tag{A.17}$$

which we can rewrite

$$q_{\phi_{\mathbf{w}}}(\mathbf{w}|\mathbf{o}) \propto \prod_{t=1}^{T}\prod_{k=1}^{K_t} q_{\phi_{\mathbf{w}}}(w_{tk}|\mathbf{o}), \tag{A.18}$$

with

$$q_{\phi_{\mathbf{w}}}(w_{tk}|\mathbf{o}) = p_{\theta_{\mathbf{w}}}(w_{tk}) \exp\Big(\mathbb{E}_{q_{\phi_{\mathbf{s}}}(\mathbf{s}_{t,:}|\mathbf{o})}\big[\log p_{\theta_{\mathbf{o}}}(\mathbf{o}_{tk}|w_{tk},\mathbf{s}_{t,:})\big]\Big). \tag{A.19}$$

The assignment variable $w_{tk}$ follows a discrete distribution and we denote:

$$\eta_{tkn} = q_{\phi_{\mathbf{w}}}(w_{tk}=n|\mathbf{o}) \propto p_{\phi_{\mathbf{w}}}(w_{tk}=n) \exp\Big(\mathbb{E}_{q_{\phi_{\mathbf{s}}}(\mathbf{s}_{tn}|\mathbf{o})}\big[\log p_{\theta_{\mathbf{o}}}(\mathbf{o}_{tk}|w_{tk}=n,\mathbf{s}_{tn})\big]\Big). \tag{A.20}$$

Using the fact that both $p_{\theta_{\mathbf{o}}}(\mathbf{o}_{tk}|w_{tk} = n, \mathbf{s}_{tn})$ and $q_{\phi_{\mathbf{s}}}(\mathbf{s}_{tn}|\mathbf{o})$ are multivariate Gaussian distributions (defined in Equation 3.3 and Equation 3.17–Equation 3.19, respectively),

---

[1] In fact, the posterior distribution $q_{\phi_{\mathbf{s}}}(\mathbf{s}_{t,:}|\mathbf{o})$ is also conditioned on $\mathbf{s}_{1:t-1,:}$ and $\mathbf{z}_{1:t,:}$. We use this abuse of notation for concision.

the previous expectation can be calculated in closed form:

$$\begin{aligned}\mathbb{E}_{q_{\phi_{\mathbf{s}}}(\mathbf{s}_{tn}|\mathbf{o})}\big[\log p_{\theta_{\mathbf{o}}}(\mathbf{o}_{tk}|w_{tk}=n,\mathbf{s}_{tn})\big] &= \int_{\mathbf{s}_{tn}} \mathcal{N}(\mathbf{s}_{tn};\mathbf{m}_{tn},\mathbf{V}_{tn})\log\mathcal{N}(\mathbf{o}_{tk};\mathbf{s}_{tn},\boldsymbol{\Phi}_{tk})d\mathbf{s}_{tn}\\ &= -\frac{1}{2}\Big[\log|\boldsymbol{\Phi}_{tk}| + (\mathbf{o}_{tk}-\mathbf{m}_{tn})^T\boldsymbol{\Phi}_{tk}^{-1}(\mathbf{o}_{tk}-\mathbf{m}_{tn})\\ &\quad + \mathrm{Tr}\big(\boldsymbol{\Phi}_{tk}^{-1}\mathbf{V}_{tn}\big)\Big]. \end{aligned} \tag{A.21}$$

By using Equation 3.5, the previous result, and normalizing to 1, we finally get:

$$\eta_{tkn} = \frac{\beta_{tkn}}{\sum_{i=1}^{N}\beta_{tki}}, \tag{A.22}$$

where

$$\beta_{tkn} = \mathcal{N}(\mathbf{o}_{tk};\mathbf{m}_{tn},\boldsymbol{\Phi}_{tk})\exp\Big(-\frac{1}{2}\mathrm{Tr}\big(\boldsymbol{\Phi}_{tk}^{-1}\mathbf{V}_{tn}\big)\Big). \tag{A.23}$$

### M Step

Here we detail the calculation of $\boldsymbol{\Phi}_{tk}$. In the ELBO expression Equation 3.10, only the first term depends on $\theta_{\mathbf{o}}$:

$$\begin{aligned}\mathcal{L}(\theta_{\mathbf{o}};\mathbf{o}) &= \mathbb{E}_{q_{\phi_{\mathbf{w}}}(\mathbf{w}|\mathbf{o})q_{\phi_{\mathbf{s}}}(\mathbf{s}|\mathbf{o})}\big[\log p_{\theta_{\mathbf{o}}}(\mathbf{o}|\mathbf{w},\mathbf{s})\big]\\ &= \sum_{n=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{K_t}\eta_{tkn}\int_{\mathbf{s}_{tn}}\mathcal{N}(\mathbf{s}_{tn};\mathbf{m}_{tn},\mathbf{V}_{tn})\log\mathcal{N}(\mathbf{o}_{tk};\mathbf{s}_{tn},\boldsymbol{\Phi}_{tk})d\mathbf{s}_{tn}\\ &= -\frac{1}{2}\sum_{n=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{K_t}\eta_{tkn}\Big[\log|\boldsymbol{\Phi}_{tk}| + (\mathbf{o}_{tk}-\mathbf{m}_{tn})^T\boldsymbol{\Phi}_{tk}^{-1}(\mathbf{o}_{tk}-\mathbf{m}_{tn}) + \mathrm{Tr}(\boldsymbol{\Phi}_{tk}^{-1}\mathbf{V}_{tn})\Big]. \end{aligned} \tag{A.24}$$

By computing the derivative of $\mathcal{L}(\theta_{\mathbf{o}};\mathbf{o})$ with respect to $\boldsymbol{\Phi}_{tk}$ and setting it to 0, we find the optimal value of $\boldsymbol{\Phi}_{tk}$ that maximizes the ELBO:

$$\boldsymbol{\Phi}_{tk} = \sum_{n=1}^{N}\eta_{tkn}\Big((\mathbf{o}_{tk}-\mathbf{m}_{tn})(\mathbf{o}_{tk}-\mathbf{m}_{tn})^T + \mathbf{V}_{tn}\Big). \tag{A.25}$$

### A.1.2 SRNN IMPLEMENTATION DETAILS

The SRNN generative model is implemented with a forward LSTM network, which embeds all the past information of the sequence s. Then, a dense layer with the *tanh* activation function plus a linear layer provide the parameters $\boldsymbol{\mu}_{\theta_{\mathbf{s}}}, \boldsymbol{v}_{\theta_{\mathbf{s}}}$. Similarly, the parameters $\boldsymbol{\mu}_{\theta_{\mathbf{z}}}, \boldsymbol{v}_{\theta_{\mathbf{z}}}$ are computed with two dense layers with tanh activation function plus a linear layer appended to the LSTM as well. The inference model shares the hidden variables of the forward LSTM network of the generative model and uses two dense layers with the *tanh* activation function plus a linear layer to compute the parameters $\boldsymbol{\mu}_{\phi_{\mathbf{z}}}, \boldsymbol{v}_{\phi_{\mathbf{z}}}$.

In the MOT set-up, both $\mathbf{s}_t$ and $\mathbf{z}_t$ are of dimension 4. While in the SC-ASS set-up, $\mathbf{s}_t$ is of dimension 513 and $\mathbf{z}_t$ is of dimension 16. The SRNN generative distributions in the right-hand side of Equation 3.28 are implemented as:

$$\mathbf{h}_t = d_h(\mathbf{s}_{t-1}, \mathbf{h}_{t-1}), \tag{A.26}$$

$$\left[\boldsymbol{\mu}_{\theta_{\mathbf{z}}}, \boldsymbol{v}_{\theta_{\mathbf{z}}}\right] = d_z(\mathbf{h}_t, \mathbf{z}_{t-1}), \tag{A.27}$$

$$p_{\theta_{\mathbf{z}}}(\mathbf{z}_t | \mathbf{s}_{1:t-1}, \mathbf{z}_{t-1}) = \mathcal{N}\big(\mathbf{z}_t; \boldsymbol{\mu}_{\theta_{\mathbf{z}}}, \mathrm{diag}(\boldsymbol{v}_{\theta_{\mathbf{z}}})\big), \tag{A.28}$$

$$\left[\boldsymbol{\mu}_{\theta_{\mathbf{s}}}, \boldsymbol{v}_{\theta_{\mathbf{s}}}\right] = d_s(\mathbf{h}_t, \mathbf{z}_t), \tag{A.29}$$

$$p_{\theta_{\mathbf{s}}}(\mathbf{s}_t | \mathbf{s}_{1:t-1}, \mathbf{z}_t) = \mathcal{N}\big(\mathbf{s}_t; \boldsymbol{\mu}_{\theta_{\mathbf{s}}}, \mathrm{diag}(\boldsymbol{v}_{\theta_{\mathbf{s}}})\big), \tag{A.30}$$

where the function $d_h$ in Equation A.26 is implemented by a forward RNN and $\mathbf{h}_t$ denotes the RNN hidden state vector, the dimension of which is set to 8 for MOT and 128 for SC-ASS. In practice, LSTM networks are used. The function $d_s$ in Equation A.29 is implemented by a dense layer of dimension 16 for MOT and of dimension 256 for SC-ASS, with the tanh activation function, followed by a linear layer, which outputs are the parameters $\boldsymbol{\mu}_{\theta_{\mathbf{s}}}, \boldsymbol{v}_{\theta_{\mathbf{s}}}$. The function $d_z$ in Equation A.27 is implemented by two dense layers of dimension 8, 8 respectively for MOT and of dimension 64, 32 respectively for SC-ASS, with the *tanh* activation function, followed by a linear layer, which outputs are the parameters $\boldsymbol{\mu}_{\theta_{\mathbf{z}}}, \boldsymbol{v}_{\theta_{\mathbf{z}}}$.

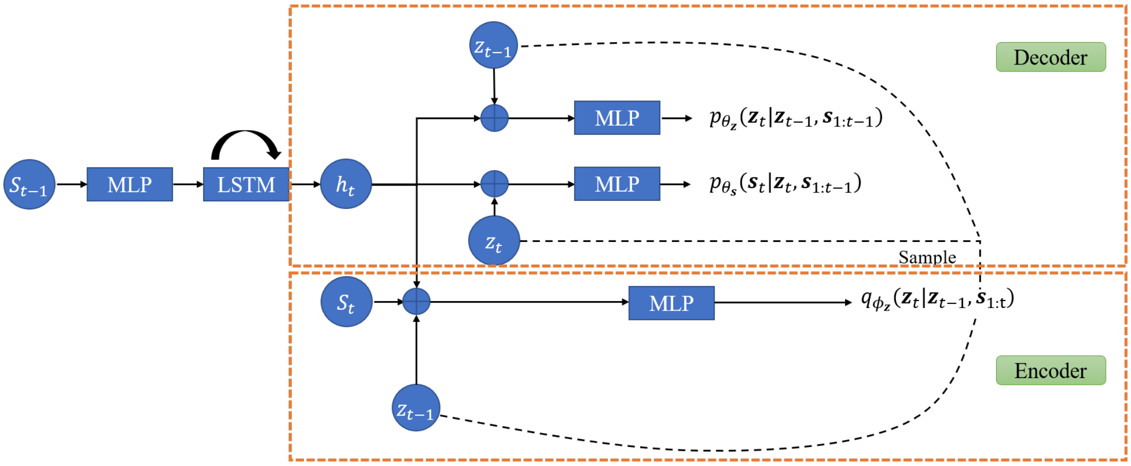

Decoder
$z_{t-1}$
MLP
$p_{\theta_z}(z_t|z_{t-1}, s_{1:t-1})$
$s_{t-1}$
MLP
LSTM
$h_t$
MLP
$p_{\theta_s}(s_t|z_t, s_{1:t-1})$
$z_t$
Sample
$s_t$
MLP
$q_{\phi_z}(z_t|z_{t-1}, s_{1:t})$
Encoder
$z_{t-1}$

**Figure A.1**: Schema of the SRNN model architecture. The “plus” symbol represents the concatenation of the input vectors.

The SRNN inference model in the right-hand side of Equation 3.29 is implemented as:

$$\left[\boldsymbol{\mu}_{\phi_{\mathbf{z}}}, \boldsymbol{v}_{\phi_{\mathbf{z}}}\right] = e_{\mathbf{z}}(\mathbf{h}_t, \mathbf{s}_t, \mathbf{z}_{t-1}), \tag{A.31}$$

$$q_{\phi_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{s}_{1:t}) = \mathcal{N}\big(\mathbf{z}_t; \boldsymbol{\mu}_{\phi_{\mathbf{z}}}, \mathrm{diag}(\boldsymbol{v}_{\phi_{\mathbf{z}}})\big), \tag{A.32}$$

where the function $e_{\mathbf{z}}$ in Equation A.31 is implemented by two dense layers of dimension 16 and 8 respectively for MOT and of dimension 64 and 32 respectively for SC-ASS, with the *tanh* activation function, followed by a linear layer, which outputs are the parameters $\boldsymbol{\mu}_{\phi_{\mathbf{z}}}, \boldsymbol{v}_{\phi_{\mathbf{z}}}$.

The SRNN architecture is schematized in Figure A.1. It can be noted that the RNN internal state $\mathbf{h}_t$ cumulating the information on $\mathbf{s}_{1:t-1}$ is shared by the encoder and the decoder, see [71, Chapter 4] for a discussion on this issue.

## A.2 APPLICATION OF MIXDVAE ON MOT

### A.2.1 CASCADE INITIALIZATION IN MOT

For the initialization of the source (position) vector, we first split the long sequence indexed by $t \in \{1, 2, ..., T\}$ into $J$ smaller sub-sequences indexed by $\{\{1, ..., t_1\}, \{t_1 + 1, ..., t_2\}, ..., \{t_{J-1} + 1, ..., T\}\}$. For the first sub-sequence, the mean vector sequence

**Algorithm 2** Cascade initialization of the position vector sequence

```
Input:
    Detected bounding boxes at the first frame $\mathbf{o}_{1,1:K_1}$;
    Pre-trained DVAE parameters $\{\theta_{\mathbf{s}}, \theta_{\mathbf{z}}, \phi_{\mathbf{z}}\}$;
    Initialized observation model covariance matrices $\{\mathbf{\Phi}_{tk}^{(0)}\}_{t,k=1}^{T,K_t}$;
    Initialized covariance matrices $\{\mathbf{V}_{tn}^{(0)}\}_{t,n=1}^{T,N}$;
Output:
    Initialized mean position vector sequence $\{\mathbf{m}_{tn}^{(0)}\}_{t,n=1}^{T,N}$;
    Initialized sampled position vector sequence $\{\mathbf{s}_{tn}^{(0)}\}_{t,n=1}^{T,N}$;
Split the whole observation sequence $\mathbf{o}$ into $J$ sub-sequences indexed by $\{t_0 = 1, ..., t_1\}$, $\{t_1 + 1, ..., t_2\}$, ..., $\{t_{J-1} + 1, ..., t_J = T\}$;
for $j == 1$ do
    for $k \leftarrow 1$ to $K_1$ do
        $n \leftarrow k$;
        for $t \leftarrow 1$ to $t_1$ do
            $\mathbf{m}_{tn}^{(0)}, \mathbf{s}_{tn}^{(0)} = \mathbf{o}_{1k}$;
        end for
    end for
end for
for $j \leftarrow 2$ to $J$ do
    for $n \leftarrow 1$ to $N$ do
        for $t \leftarrow t_{j-1} + 1$ to $t_j$ do
            $\mathbf{m}_{tn}^{(0)}, \mathbf{s}_{tn}^{(0)} = \mathbf{m}_{t_{j-1}n}^{(I_0)}$;
        end for
        $\{\mathbf{m}_{tn}^{(I_0)}\}_{t=t_{j-1}+1}^{t_j}$ = MixDVAE($I_0$, $\{\{\theta_{\mathbf{s}}, \theta_{\mathbf{z}}, \phi_{\mathbf{z}}\}$,
        $\{\mathbf{\Phi}_{tk}^{(0)}, \mathbf{m}_{tn}^{(0)}, \mathbf{V}_{tn}^{(0)}, \mathbf{s}_{tn}^{(0)}\}_{t=t_{j-1}+1,k=1,n=1}^{t_j,K_t,N}\}$);
    end for
end for
$\mathbf{m}_{1:T,1:N}^{(0)} = \left[\mathbf{m}_{1:t_1,1:N}^{(0)}, ..., \mathbf{m}_{t_{J-1}+1:T,1:N}^{(0)}\right]$;
$\mathbf{s}_{1:T,1:N}^{(0)} = \left[\mathbf{s}_{1:t_1,1:N}^{(0)}, ..., \mathbf{s}_{t_{J-1}+1:T,1:N}^{(0)}\right]$;
```

$\mathbf{m}_{1:t_1,n}$ is initialized as the detected vector at the first frame $\mathbf{o}_{1k}$ repeated for $t_1$ times with a arbitrary order of assignment. Thus, there are as many tracked sources as initial detections, i.e., this implicitly sets $N = K_1$. The subsequence of source position vectors $\mathbf{s}_{1:t_1,n}$ is initialized with the same values as for the mean vector. Then, we run the MixDVAE algorithm on the first subsequence for $I_0$ iterations. Next, we initialize the mean vector sequence $\mathbf{m}_{t_1+1:t_2,n}$ of the second subsequence with $\mathbf{m}_{t_1n}$ repeated for $t_2 - t_1$ times (and the same for $\mathbf{s}_{t_1+1:t_2,n}$). And so on for the following subsequences. Finally, the initial-

**Algorithm 3** Synthetic trajectories generation

```
Input:
    Total sequence length $T$;
    Maximum sub-sequence number $s_{max}$;
    Distribution parameters $\mu_{b_0}$, $\sigma_{b_0}$, $\mu_r$, $\sigma_r$, $\mu_{a_1}$, $\sigma_{a_1}$, $\mu_{a_2}$, $\sigma_{a_2}$, $\mu_\omega$, $\sigma_\omega$, $\mu_{\phi_0}, \sigma_{\phi_0}$;
    Discrete probability distribution of different elementary trajectory function types $p = [p_1, p_2, p_3, p_4]$;
Output:
    Synthetic bounding box position sequence $gen\_seq = \{(x_t^{\mathrm{L}}, x_t^{\mathrm{T}}, x_t^{\mathrm{R}}, x_t^{\mathrm{B}})\}_{t=1}^{T}$;
function GenSeq($x_0$, $s$, $t_{split}$, $params\_prob$, $p$)
    $start = x_0$;
    for $i \leftarrow 0$ to $s$ do
        Sample $function\_type$ using $p$;
        Sample trajectory function parameters $params\_list$ using $params\_prob$;
        $t_i = t_{split}[i]$;
        $x\_sub_i$ = GenTraj($start$, $func\_type$,
        $params\_list$);
        $start = x\_sub_i[t_i]$;
    end for
    $x = [x\_sub_0, ..., x\_sub_{s-1}]$;
    return x;
end function
Sample $x_0$, $y_0$ from $\mathcal{U}(0, 1)$;
Sample $b_0$ from $\log \mathcal{N}(\mu_{b_0}, \sigma_{b_0})$;
Sample $r_{ab}$ from $\mathcal{N}(\mu_r, \sigma_r)$;
Randomly sample $s$ in $\{0, ..., s_{max}\}$;
Randomly sample $t_{split} = \{t_0, ..., t_{s-1}\}$ in $\{1, ..., T\}$;
$x =$ GenSeq($x_0$, $s$, $t_{split}$, $params\_prob$, $p$);
$y =$ GenSeq($y_0$, $s$, $t_{split}$, $params\_prob$, $p$);
$b =$ GenSeq($w_0$, $s$, $t_{split}$, $params\_prob$, $p$);
$a = b * r_{ab}$;
$gen\_seq = [x, y, x + b, y - a]$;
```

ized subsequences are concatenated together to form the initialized whole sequence. The pseudo-code of the cascade initialization can be found in Algorithm 2.

### A.2.2 MOT DATASET PROCESSING

#### Synthetic trajectory dataset generation

To generate bounding boxes with reasonable size, we generate the coordinates of the top-left point (noted as $x_t^{\mathrm{L}}$ and $x_t^{\mathrm{T}}$) plus the height (noted as $a_t$) and width (noted as

$b_t$) of the bounding boxes and deduce the coordinates of the bottom-right point. The width-height ratio is sampled randomly, and kept constant during the trajectory. While the trajectory of one coordinate is generated using piece-wise combinations of elementary functions, which are: static $a(t) = a_0$, constant velocity $a(t) = a_1 t + a_0$, constant acceleration $a(t) = a_2 t^2 + a_1 t + a_0$, and sinusoidal (allowing for circular trajectories) $a(t) = a\sin(\omega t + \phi_0)$. That is to say, we split the whole sequence into several segments, and each segment is dominated by a certain elementary function. An example of a 3-segment combination could be:

$$a(t) = \begin{cases} a_0 & 1 \leq t < t_1, \\ a_2 t^2 + a_1 t + a_0' & t_1 \leq t < t_2, \\ a_3 \sin(\omega t + \phi_0) & t_1 \leq t \leq T, \end{cases} \tag{A.33}$$

where the segments length is sampled from some pre-defined distributions to generate reasonable and continuous trajectories. The number of segments $s$ is first uniformly sampled in the set $\{1, \ldots, s_{\max}\}$. We then sample $s$ segment lengths that sum up to $T$. This defines the segment boundaries $t_1, \ldots, t_{s-1}$. For each segment, one of the four elementary functions is randomly selected. The function parameters are sampled as follow: $a_1 \sim \mathcal{N}(\mu_{a_1}, \sigma^2_{a_1})$, $a_2 \sim \mathcal{N}(\mu_{a_2}, \sigma^2_{a_2})$, $\omega \sim \mathcal{N}(\mu_\omega, \sigma^2_\omega)$ and $\phi_0 \sim \mathcal{N}(\mu_{\phi_0}, \sigma^2_{\phi_0})$. The two remaining parameters, $a_0$ and $a$, are set to the values needed to ensure continuous trajectories, thus initialising the trajectories at every segment, except for the first one. The very initial trajectory point is sampled randomly from $\mathcal{U}(0, 1)$. And the initial width is sampled from a log-normal distribution $b_0 \sim \log\mathcal{N}(\mu_{b_0}, \sigma^2_{b_0})$. Finally, the ratio between the height and width is supposed to be constant with respect to time. It is sampled from a log-normal distribution $r_{ab} = \frac{a}{b} \sim \log\mathcal{N}(\mu_r, \sigma^2_r)$ and the height is obtained by multiplying the width and the ratio. More implementation details can be found in Algorithm 3.

In our experiments, the total sequence length of the generated trajectories for DVAE pre-training equals to $T = 60$ frames. And the maximum number of segments is set to $s_{max} = 3$. The parameters of the $a_1$, $a_2$, $\omega$, $\phi_0$, $w_0$, and $r_{hw}$ distributions are determined by estimating the statistical characteristics of publicly published detections of the MOT17 training dataset. More precisely, we estimated the empirical mean and standard deviation of the speed and acceleration for all matched detection sequences (i.e., the first and second order differentiation of the position sequences).

### MOT17-3T dataset construction

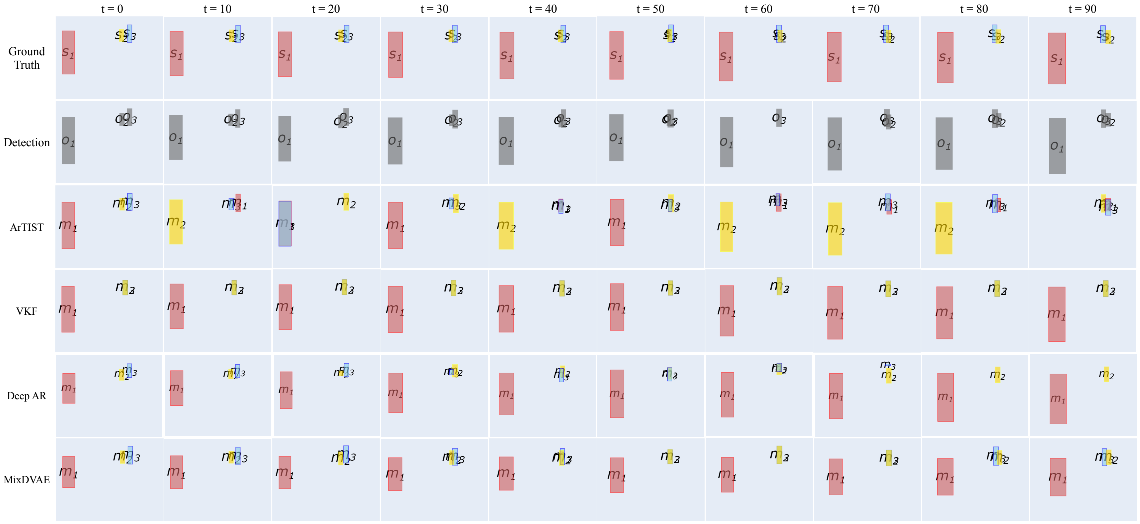
To construct the MOT17-3T dataset, first, we matched the detected bounding boxes to the ground-truth bounding boxes using the Hungarian algorithm [110] and retained only the matched detected bounding boxes (i.e., the detected bounding boxes that were not matched to any ground-truth bounding boxes were discarded). The cost matrix were computed according to the the Intersection-over-Union (IoU) distance between bounding boxes. We split each complete video sequence into subsequences of length $T$ (three different values of $T$ are tested in our experiments, as detailed below) and only kept the tracks with a length no shorter than $T$. For each subsequence, we randomly chose three tracks that appeared in this subsequence from the beginning to the end. The detected bounding boxes of these three tracks form one test data sample. We have tested three values for the sequence length $T$ to evaluate its influence on the tracking performance of our algorithm: 60, 120, and 300 frames (respectively corresponding to 2, 4, and 10 seconds at 30 fps). Among the three public detection results provided with the MOT17 dataset, SDP has the best detection performance. So, we used the detection results of SDP to create our dataset.

## A.2.3 MOT BASELINES IMPLEMENTATION DETAILS

ArTIST [179] is a probabilistic auto-regressive model which consists of two main blocks: MA-Net and the ArTIST model. MA-Net is a recurrent autoencoder that is trained to learn a representation of the dynamical interaction between all agents in the scene. ArTIST is an RNN that takes as input a 4D velocity vector of the current frame for one object as well as the corresponding 256-dimensional interaction representation learned by MA-Net, and outputs a probability distribution for each dimension of the motion velocity for the next frame. As indicated in [179], the models are trained on the MOT17 training set and the PathTrack [135] dataset. We have reused the trained models as well as the tracklet scoring and inpainting code provided by the authors and reimplemented the object tracking part according to the paper, as this part was not provided. The tracklets are initialized with the bounding boxes detected in the first frame. For any time frame $t$, the score of assigning a detection $\mathbf{o}_{tk}$ to a tracklet $n$ is obtained by evaluating the likelihood of this detection under the distribution estimated by the ArTIST model. The final assignment is computed using the Hungarian algorithm. For any tentatively alive tracklet whose last assignment is prior to $t-1$ with a non-zero gap (implying that there exists a detection absence), the algorithm first performs tracklet inpainting to fill the gap up to $t-1$, then computes the assignment score with the inpainted tracklet. As described in [179], the inpainting is done with multinominal sampling, and a tracklet rejection scheme (TRS) is applied to select the best inpainted trajectory. In order to eliminate possible inpainting ambiguities, the Hungarian algorithm is run twice, once only for the full sequences without gaps and the second time for the inpainted sequences. The number of candidates for multinominal sampling is set to 50. For the TRS, the IoU threshold used in [179] is $0.5$. In our test scenario, there are less tracklets and the risk of false negative is much greater than that of false positive. So, we decreased the threshold to $0.1$, which provided better results than the original value.

### A.2.4 MORE MOT TRACKING EXAMPLES

The first example plotted in Fig. A.2 illustrates the case where two persons cross each other. This is one of the most complicated situations that may cause an identity switch and even lead to tracking loss. Considering the limited space for the figure, we display the bounding boxes every ten frames to view the whole process of crossing. For $t = 60$, when the ground-truth bounding boxes of Sources 2 and 3 ($s_2$ and $s_3$ in the figure) strongly overlap, Detection $o_2$ disappears. Again, ArTIST exhibits frequent identity switches. Besides, at $t = 20$, the estimated bounding box $m_1$ is totally overlapped with that of $m_3$. And at $t = 90$, the estimated bounding boxes for all of the three sources are getting very close to each other. This indicates that the identity switches can cause unreasonable trajectories estimation. For VKF, the observations for both Sources 2 and 3 are assigned to the same target $s_3$ all along the sequence, due to $s_2$ and $s_3$ being close to each other, Example 1: Crossing sources.



Example 2: Crossing sources with frequent detection absence.

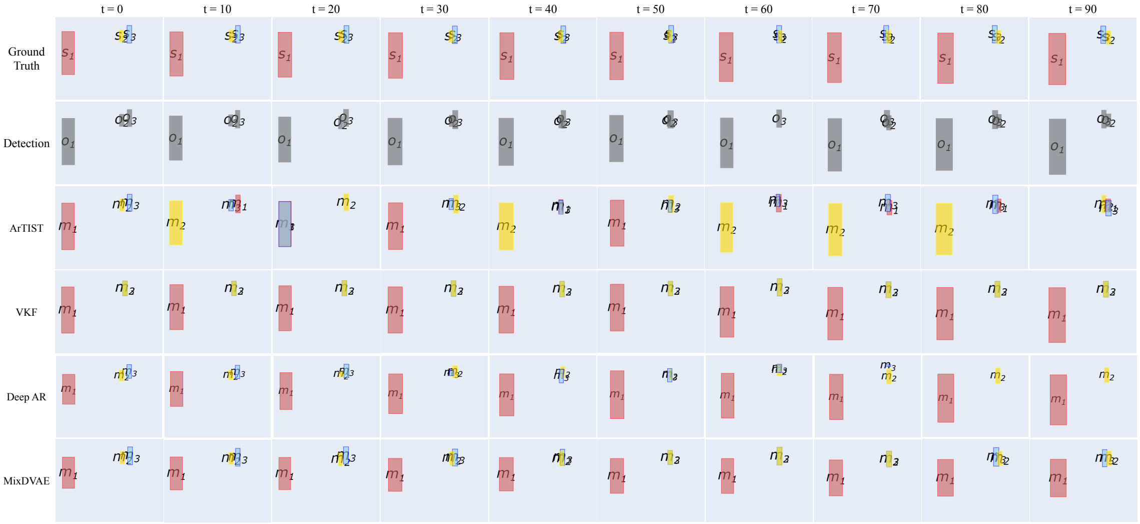


**Figure A.2**: Examples of tracking result obtained with the proposed MixDVAE algorithm and the two baselines. For clarity of presentation, the simplified notations $s_1$, $o_1$, and $m_1$ denote the ground-truth source position, the observation, and the estimated position, respectively (for Source 1, and the same for the two other sources). Best seen in color.

so that the estimated bounding boxes $m_2$ and $m_3$ overlap completely. For the Deep AR, the estimation of $m_3$ becomes inaccurate from $t = 70$ and it disappears at $t = 80$ and $t = 90$ (the estimation is out of the frame). In contrast, MixDVAE displays a consistent tracking of the three sources. For $t = 50$, $60$, and $70$, the estimated bounding boxes $m_2$

and $m_3$ overlap due to the ground-truth bounding boxes $s_2$ and $s_3$ strongly overlap each other. However, the tracking is correctly resumed at $t = 80$, with no identity switch (i.e., the crossing of Sources 2 and 3 is correctly captured by the model).

The second example displayed in Fig. A.2 is another more complicated situation with two sources very close to each other and frequent detection absence. At $t = 20$ when observation $o_3$ disappears, both ArTIST and VKF lose one of the tracks, whereas MixDVAE keeps a reasonable tracking of the three tracks. From $t = 60$ to $80$, both $o_2$ and $o_3$ are absent. The tracks inpainted by ArTIST are not consistent anymore and VKF still misses one track. The estimations of Deep AR are inaccurate when the detections are absent. However, even in this difficult scenario, MixDVAE keeps on providing three reasonable trajectories.

## A.3 APPLICATION OF MIXDVAE ON SC-ASS

### A.3.1 FORMULAS FOR SC-ASS

With the adaptations in the model mentioned at the beginning of Chapter 5 for the SC-ASS task, the solution formulas are as following. In the E-S Step, Equation 3.18 and Equation 3.19 become:

$$\mathbf{V}_{tn} = \Big( \sum_{k=1}^{K_t} \eta_{tkn} \mathbf{P}_k^T \mathbf{\Phi}_{tk}^{-1} + \text{diag}(\boldsymbol{v}_{\theta_\mathbf{s},tn}^{(i)})^{-1} \Big)^{-1}, \tag{A.34}$$

$$\mathbf{m}_{tn} = \mathbf{V}_{tn} \Big( \sum_{k=1}^{K_t} \eta_{tkn} \mathbf{P}_k^T \mathbf{\Phi}_{tk}^{-1} \mathbf{o}_{tk} \Big). \tag{A.35}$$

The E-Z Step is not changed. In the E-W Step, Equation 3.25 become:

$$\beta_{tkn} = \mathcal{N}_c(\mathbf{o}_{tk}; \mathbf{P}_k \mathbf{m}_{tn}, \mathbf{\Phi}_{tk}) \exp\Big( -\frac{1}{2} \text{Tr}\big(\mathbf{P}_k^T \mathbf{\Phi}_{tk}^{-1} \mathbf{P}_k \mathbf{V}_{tn}\big) \Big). \tag{A.36}$$

Finally, in the M Step, Equation 3.26 become:

$$\mathbf{\Phi}_{tk} = \sum_{n=1}^{N} \eta_{tkn} \Big( (\mathbf{o}_{tk} - \mathbf{P}_k \mathbf{m}_{tn})(\mathbf{o}_{tk} - \mathbf{P}_k \mathbf{m}_{tn})^T + \mathbf{P}_k \mathbf{V}_{tn} \mathbf{P}_k^T \Big). \tag{A.37}$$

### A.3.2 MORE SC-ASS EXAMPLES

In Fig.A.3 we plot two other SC-ASS examples.

## A.4 UNSUPERVISED SPEECH ENHANCEMENT WITH DEEP DYNAMICAL PROBABILISTIC GENERATIVE MODELS

### A.4.1 POSTERIOR DISTRIBUTIONS DERIVATION

Applying the D-Separation [24] method, the posterior distribution corresponding to the generative model Equation 6.9 can be decomposed as follows:

$$p_{\phi}(\mathbf{s}_{1:T}, \mathbf{z}_{1:T} | \mathbf{x}_{1:T}) = \prod_{t=1}^{T} p_{\phi_{\mathbf{s}}}(\mathbf{s}_t | \mathbf{z}_{1:t}, \mathbf{x}_t, \mathbf{p}_t) p_{\phi_{\mathbf{z}}}(\mathbf{z}_t | \mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}). \tag{A.38}$$

For each time frame $t$, by applying the Bayesian formula, the terms related to $\mathbf{s}_t$ can be developed as:

$$\begin{aligned} p_{\phi_{\mathbf{s}}}(\mathbf{s}_t | \mathbf{z}_{1:t}, \mathbf{x}_t, \mathbf{p}_t) &= \frac{p_{\theta_{\mathbf{n}}}(\mathbf{x}_t | \mathbf{s}_t, \mathbf{p}_t) p_{\theta_{\mathbf{s}}}(\mathbf{s}_t | \mathbf{z}_{1:t}) p(\mathbf{z}_{1:t}, \mathbf{x}_{1:t-1})}{p(\mathbf{z}_{1:t}, \mathbf{x}_{1:t})}, \\ &\propto p_{\theta_{\mathbf{n}}}(\mathbf{x}_t | \mathbf{s}_t, \mathbf{p}_t) p_{\theta_{\mathbf{s}}}(\mathbf{s}_t | \mathbf{z}_{1:t}), \\ &= \mathcal{N}_c(\mathbf{x}_t; \mathbf{s}_t, \mathrm{diag}(\mathbf{v}_{\theta_{\mathbf{n}},t}(\mathbf{p}_t))) \mathcal{N}_c(\mathbf{s}_t; \mathbf{0}, \mathrm{diag}(\mathbf{v}_{\theta_{\mathbf{s}},t})). \end{aligned} \tag{A.39}$$

Therefore, $p(\mathbf{s}_t | \mathbf{z}_{1:t}, \mathbf{x}_{1:t})$ is a complex Gaussian distribution $p(\mathbf{s}_t | \mathbf{z}_{1:t}, \mathbf{x}_{1:t}) = \mathcal{N}_c(\boldsymbol{\mu}_{\phi_{\mathbf{s}},t}, \mathbf{v}_{\phi_{\mathbf{s}},t})$, with

$$\boldsymbol{\mu}_{\phi_{\mathbf{s}},t} = \frac{\mathbf{v}_{\theta_{\mathbf{s}},t}(\mathbf{z}_{1:t})}{\mathbf{v}_{\theta_{\mathbf{s}},t}(\mathbf{z}_{1:t}) + \mathbf{v}_{\theta_{n},t}(\mathbf{x}_{1:t-1})} \mathbf{x}_t, \tag{A.40}$$

$$\mathbf{v}_{\phi_{\mathbf{s}},t} = \frac{\mathbf{v}_{\theta_{\mathbf{s}},t}(\mathbf{z}_{1:t}) \mathbf{v}_{\theta_{n},t}(\mathbf{x}_{1:t-1})}{\mathbf{v}_{\theta_{\mathbf{s}},t}(\mathbf{z}_{1:t}) + \mathbf{v}_{\theta_{n},t}(\mathbf{x}_{1:t-1})}. \tag{A.41}$$

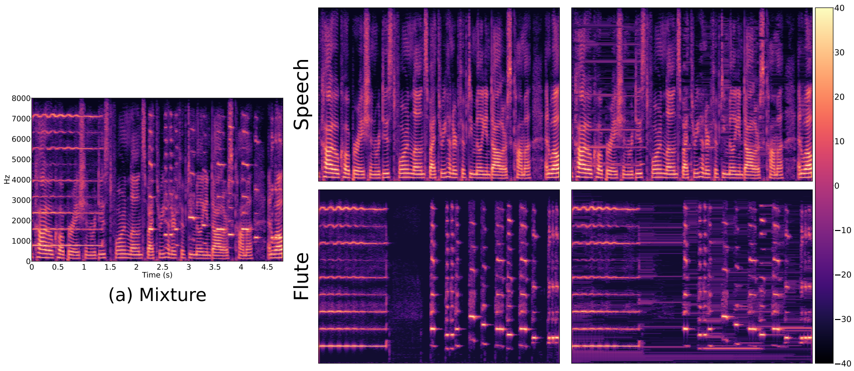
While the distribution $p_{\phi_{\mathbf{z}}}(\mathbf{z}_t | \mathbf{z}_{1:t-1}, \mathbf{x}_{1:T})$ is computationally intractable, we approximate it with the RVAE inference model:

$$p_{\phi_{\mathbf{z}}}(\mathbf{z}_t | \mathbf{z}_{1:t-1}, \mathbf{x}_{1:T}) \approx q_{\phi_{\mathbf{z}}}(\mathbf{z}_t | \mathbf{z}_{1:t-1}, \mathbf{x}_{t:T}), \tag{A.42}$$

with

$$q_{\phi_{\mathbf{z}}}(\mathbf{z}_t|\mathbf{z}_{1:t-1}, \mathbf{s}_{t:T}) = \mathcal{N}(\mathbf{z}_t; \boldsymbol{\mu}_{\phi_{\mathbf{z}},t}, \text{diag}(\mathbf{v}_{\phi_{\mathbf{z}},t})). \tag{A.43}$$

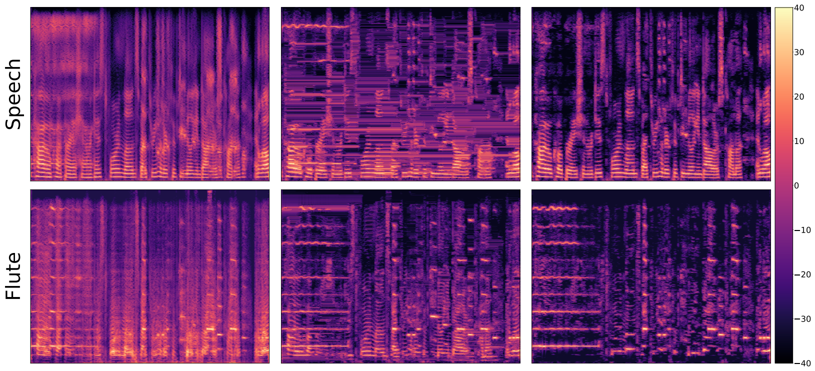
The parameters to be optimized in the inference model are: $\phi = \{\phi_{\mathbf{z}}\}$.



(a) Mixture

(b) Ground Truth

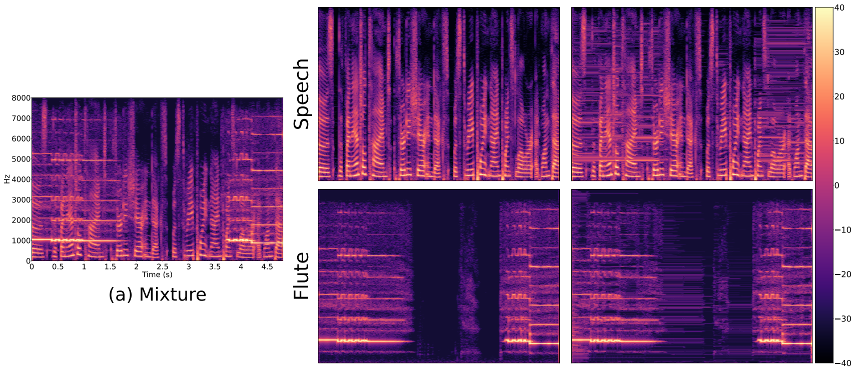
(c) VKF-Oracle



(d) DVAE-init

(e) VKF-DVAE-init

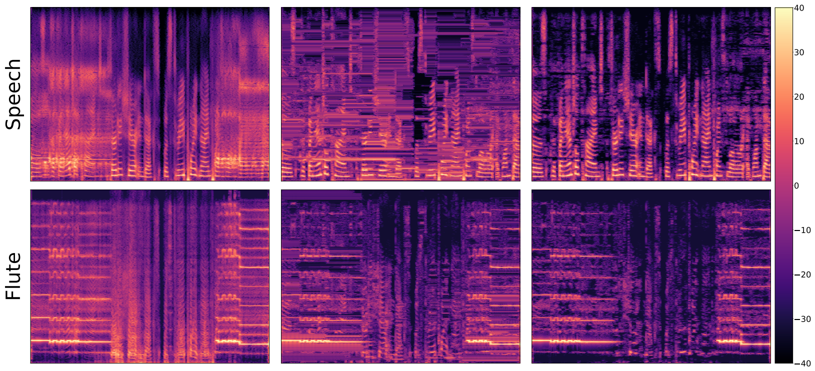
(f) MixDVAE



(a) Mixture

(b) Ground Truth

(c) VKF-Oracle



(d) DVAE-init

(e) VKF-DVAE-init

(f) MixDVAE

**Figure A.3**: Examples of audio source separation results obtained with the proposed MixDVAE algorithm and the baselines. Best seen in color.

# ACRONYMS

**AR** autoregressive. 149, 150, 158, 160, 163

**ASS** Audio source separation. 66–69

**BBVI** blackbox variational inference. 54

**BLSTM** bidirectional LSTM. 135

**CASA** computational auditory scene analysis. 67

**CBF** Chinese bamboo flute. 110–120, 204, 208

**CNN** convolutional neural network. 27, 58, 59, 145, 166

**DBBs** detection bounding boxes. 65, 92–94, 97

**DKF** Deep Kalman Filter. 158, 159, 163

**DLVM** deep latent variable model. 32

**DNN** deep neural network. 20, 22, 23, 26–28, 32, 34, 35, 40, 41, 44, 55, 58, 64, 68, 109, 128, 130, 131, 141, 145, 146, 169, 170

**DPGM** deep probabilistic generative model. 3, 4, 20, 22, 23, 44, 55, 66, 69–71, 74, 122, 128, 129, 131, 136, 137, 144, 145, 148, 166–170, 208

**DVAE** dynamical variational auto-encoder. 4, 22, 23, 44, 45, 47, 66, 69, 71, 74–81, 83–89, 92, 93, 95–97, 99–104, 106, 108–116, 118, 120–123, 127–129, 131, 145, 146, 148–150, 156, 158, 160, 163, 164, 166–168, 170, 180, 182, 187, 189, 203

**ELBO** Evidence Lower BOund. 29, 50–55, 81, 82, 85–87, 102, 109, 130, 133, 134, 154, 182, 184

**EM** expectation-maximization. 50, 51, 53, 77, 97, 128, 140, 169

**ESTOI** extended short-time objective intelligibility. 136, 158–160

**FA** factor analysis. 32, 37, 39, 40

**factorized HMMs** factorized hidden Markov models. 68

**FDSD** Fréchet Deep Speech Distance. 159, 163

**FF** feed-forward. 154, 155

**FN** false negatives. 98–100, 103, 105

**FP** false positives. 98–100, 103, 105

**GAN** generative adversarial network. 26, 70, 71

**GMM** Gaussian mixture model. 33, 35, 42

**GP-VAE** Gaussian process variational autoencoder. 46, 48

**GPT** Generative Pre-trained Transformer. 63

**GRU** gated recurrent units. 57

**HiT-DVAE** hierarchical Transformer dynamical variational auto-encoder. 4, 23, 148, 150–161, 163, 164, 204

**HMM** hidden Markov model. 41–44, 68

**HMM-GMM** hidden Markov model - gaussian mixture model. 43, 64

**i.i.d.** independently and identically distributed. 31, 32, 41

**IBM** ideal binary mask. 67, 68, 114

**ICA** independent component analysis. 39, 40

**IDF1** identity F1 score. 98, 99, 103, 105

**IDS** number of identity switches. 98, 99

**IS** Itakura-Saito. 110, 131, 153

**JPDAF** joint probabilistic data association filter. 65

**KL** Kullback-Leibler. 50–52, 83, 110, 131, 136, 153

**Latent SDE** latent stochastic differential equations. 46

**LDA** latent Dirichlet allocation. 64

**LDS** linear(-Gaussian) dynamical system. 43, 44, 57

**LGM** linear Gaussian model. 32, 37–40, 43, 44

**LMMSE** linear minimum mean-square-error. 67, 132

**LN** Layer normalization. 154

**LSTM** long short-term memory. 57, 59, 97, 114, 135, 136, 140, 154, 164

**LV** latent variable. 129, 131, 135, 144

**LVM** latent variable model. 23, 29–33, 35, 37, 39–41, 43, 46, 48, 49, 68, 166, 169, 170

**MCMC** Markov Chain Monte Carlo. 28, 49

**MHA** multi-head attention. 62

**MHCA** multi-head cross-attention. 155–157, 160

**MHSA** multi-head self-attention. 154, 155, 157

**MHT** multiple hypothesis tracking. 65

**ML** mostly lost. 98, 99, 103, 105

**MLE** maximum likelihood estimation. 26

**MLP** multilayer perceptron. 27, 135

**MOT** multi-object tracking. 4, 23, 65, 66, 74, 76, 85, 92, 93, 95–99, 103, 105, 108, 112–114, 118, 120–123, 166, 167, 204, 207, 208

**MOTA** multi-object tracking accuracy. 98, 99, 103–106, 121, 122, 204

**MOTP** multi-object tracking precision. 98, 99, 103, 105

**MT** mostly tracked. 98, 99, 103, 105

**MTT** Multi-target tracking. 64, 65

**NA** noise-agnostic. 127, 129, 133–135, 137, 143–145, 167

**NCE** noise contrastive estimation. 28

**ND** noise-dependent. 127, 129, 133–136, 139, 143–145, 167

**NMF** non-negative matrix factorization. 68, 70, 113, 114, 116, 117, 127, 128, 131, 134, 136, 140, 141, 145, 146, 167, 169

**NO** noisy observations. 129, 131, 135, 144

**NOLV** noisy observations and latent variables. 129, 131, 135, 137, 144

**PCA** principal components analysis. 39

**PDF** probability density function. 26–28, 109, 207

**PDM** probabilistic discriminative model. 20

**PESQ** perceptual evaluation of speech quality. 115, 116, 119, 120, 136, 139, 158–160

**PGM** probabilistic generative model. 20–23, 170

**PPCA** probabilistic principal components analysis. 39

**RFSs** random finite sets. 65

**RMSE** root mean squared error. 115, 116, 119, 120, 158

**RNN** recurrent neural network. 27, 44, 45, 56–60, 63, 66, 69, 130, 148–150, 166, 185

**RNNs** recurrent neural networks. 56

**RTF** real-time factor. 136, 140, 141

**RVAE** Recurrent Variational AutoEncoder. 129, 130, 132, 133, 135, 136, 144, 145, 158, 159, 163

**SC-ASS** single-channel audio source separation. 4, 23, 67–69, 74, 76, 85, 108–110, 112, 113, 115, 116, 118–123, 128, 129, 166, 167, 193, 194, 204, 208

**SDE** stochastic differential equation. 29, 48

**SE** speech enhancement. 4, 23, 126–129, 133, 135–139, 141–146, 166, 167, 170, 204, 208, 209

**Seq2Seq** sequence to sequence. 56, 57, 59, 60

**Seq2Vec** sequence to vector. 56

**SGD** stochastic gradient descent. 55, 85, 88, 128, 134, 146, 169

**SI-SDR** scale-invariant signal-to-distortion ratio. 115, 116, 119–122, 136, 139, 141–143, 158, 160, 204

**SNR** signal-to-noise ratio. 134, 139, 141, 143

**SRNN** stochastic recurrent neural network. 89, 90, 94, 99, 100, 102, 103, 105, 111, 119, 120, 158–160, 163, 164, 185, 186, 205, 207, 208

**SS** scheduled-sampling. 158–160, 163

**SSM** state space model. 32, 44

**STFT** short-time Fourier transform. 67, 109–112, 114, 129–132, 135, 139, 145, 150–152, 157–159

**TF** teacher-forcing. 158–160, 163

**TF** time-frequency. 108, 110

**TTS** text-to-speech. 59

**U-NA** unsupervised noise-agnostic. 136–138, 140–142, 144, 208, 209

**U-ND** unsupervised noise-dependent. 137–141, 144, 208, 209

**U-NDA** unsupervised noise-dependent with noise adaptation. 137, 140, 141

**VAE** variational auto-encoder. 4, 34, 35, 40, 41, 44, 45, 47, 55, 70, 93, 127, 148, 158, 163, 170

**VB** Voice Bank. 157–160, 163

**Vec2Seq** vector to sequence. 56

**VEM** variational expectation-maximization. 53, 75, 78, 89, 95, 96, 112–114, 118, 121–123, 128, 136, 146, 166, 167, 204

**VI** variational inference. 4, 23, 49, 51–53, 75, 77, 81, 97, 99, 123, 166, 169, 170

**VQ** vector quantization. 35

**VQ-VAE** Vector-quantised variational autoencoder. 34–37

**WSJ0** Wall Street Journal. 110–112, 118, 119, 121, 122, 157–159, 208

# LIST OF FIGURES

2.1 The model architecture of Transformer. This figure is adapted from Figure 1 of paper [207]. . . . . . . . . . . . . . . . . . . . . . . . . . . . 61

3.1 Graphical representation of the proposed MixDVAE model. . . . . . . . . 78

3.2 Overview of the proposed MixDVAE algorithm at a given time frame $t$. The DVAE model is pretrained offline using a (synthetic or natural) single-source dataset. It takes as input the sequence of source vectors, encodes them into a sequence of latent vectors, which are then decoded into the reconstructed sequence of source vectors. For a given time frame $t$, the MixDVAE algorithm takes as input the observations at time $t$ as well as the mean and variance vectors estimated by the DVAE model. By iterating the E-S, E-Z, E-W and M steps, we obtain estimates of the assignment variable and of each source vector. . . . . . . . . . . . . . . . . 89

4.1 Example of tracking result obtained with the proposed MixDVAE algorithm and the two baselines. For clarity of presentation, the simplified notations $s_1$, $o_1$, and $m_1$ denote the ground-truth source position, the observation, and the estimated source position, respectively (for Source 1, and the same for the two other sources). Best seen in color. . . . . . . . . 101

4.2 Two examples of tracking result obtained with the proposed MixDVAE algorithm, with and without fine-tuning during the E-Z step. For clarity of presentation, the simplified notations $s_1$, $o_1$, and $m_1$ denote the ground-truth source position, the observation, and the estimated position, respectively (for Source 1, and the same for the two other sources). Best seen in color. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 104

4.3 MOTA score obtained by MixDVAE as a function of the number of VEM iterations, for different values of $r_{\mathbf{\Phi}}$. . . . . . . . . . . . . . . . . . . . . 106

5.1 An example of audio source separation result obtained with the proposed MixDVAE algorithm and the baselines (speech and CBF power spectrograms). Best seen in color. . . . . . . . . . . . . . . . . . . . . . . . . 117

5.2 Evolution of the performance of MixDVAE as a function of the number of VEM iterations (MOTA score for the MOT task and SI-SDR scores for the SC-ASS task). . . . . . . . . . . . . . . . . . . . . . . . . . . . . 122

6.1 Schematic view of the proposed SE method. The training and test configurations correspond to the two gray boxes depicted on the right side of the figure. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 133

6.2 Example of SE results obtained with different methods. In this example, the noise type is “restaurant” and the SNR of the noisy speech is 0 dB. Best seen in color. . . . . . . . . . . . . . . . . . . . . . . . . . . . 142

6.3 Example of SE results obtained with different methods. In this example, the noise type is “speech-shaped noise” and the SNR of the noisy speech is 5 dB. Best seen in color. . . . . . . . . . . . . . . . . . . . . . . . . 143

7.1 HiT-DVAE model architecture. . . . . . . . . . . . . . . . . . . . . . . . 153

7.2 LigHT-DVAE model structure. . . . . . . . . . . . . . . . . . . . . . . . 156

7.3 Speech re-synthesis results for two example speech signals with and without swapping $\mathbf{w}$. In the first column, GT_s1, recon_s1_w1, recon_s1_w2 represent the ground truth spectrogram of $\mathbf{s}_1$, the spectrogram re-synthesised from $\mathbf{z}_1$, $\mathbf{w}_1$, and the spectrogram re-synthesised from $\mathbf{z}_1$, $\mathbf{w}_2$, respectively. Similar notations apply to $\mathbf{s}_2$ in the second column. . . . . . . . . 162

8.1 Conversation with ChatGPT about the ultimate aim behind the development of AI. This conversation is generated by GPT-3.5. . . . . . . . . . . 174
8.2 Conversation with ChatGPT about the relationship between humans and AI. This conversation is generated by GPT-3.5. . . . . . . . . . . . . . 175
8.3 Conversation with ChatGPT about what is a good way to deploy AI. This conversation is generated by GPT-3.5. . . . . . . . . . . . . . . . . . . . 176
8.4 A picture of how people will live with AI in the future generated by DALL·E. The prompt used to generate this picture is "Please draw me a picture of how people will live with AI in the future." . . . . . . . . . . 177

A.1 Schema of the SRNN model architecture. The "plus" symbol represents the concatenation of the input vectors. . . . . . . . . . . . . . . . . . . . 186
A.2 Examples of tracking result obtained with the proposed MixDVAE algorithm and the two baselines. For clarity of presentation, the simplified notations $s_1$, $o_1$, and $m_1$ denote the ground-truth source position, the observation, and the estimated position, respectively (for Source 1, and the same for the two other sources). Best seen in color. . . . . . . . . . . . . 192
A.3 Examples of audio source separation results obtained with the proposed MixDVAE algorithm and the baselines. Best seen in color. . . . . . . . . 196

# LIST OF TABLES

2.1 The PDF forms of different generative models. In all of the models, $\mathrm{x}$ represents the data and $\theta$ represents the parameters of the model. Other symbols used in a specific model will be explained in the corresponding paragraph. . . . . . . . . . . . . . . . . . . . . . . . . . . . 26

3.1 Summary of the variable notations. . . . . . . . . . . . . . . . . . . . . . . . 77

4.1 MOT results for short ($T = 60$), medium ($T = 120$), and long ($T = 300$) sequences. . . . . . . . . . . . . . . . . . . . . . . . . . . . 99

4.2 Capacity of the SRNN model pre-trained at three data scales of the synthetic trajectories dataset. SRNN-full, SRNN-half, and SRNN-quarter stand for SRNN pre-trained on the totality, half of and quarter of our original training set, respectively. . . . . . . . . . . . . . . . . . . . . . . . . 102

4.3 MOT results obtained by MixDVAE with SRNN pre-trained at the three data scales. The results are reported for the short sequence test subset ($T = 60$ frames). . . . . . . . . . . . . . . . . . . . . . . . . . . . 103

4.4 MOT results obtained by MixDVAE with and without the fine-tuning of SRNN. The results are reported for the short, medium and long sequence test subsets ($T = 60$, $120$, and $300$ frames, respectively). . . . . . . . . . 103

4.5 Results obtained by MixDVAE on MOT17-3T (short sequences subset) for different values of $r_{\mathbf{\Phi}}$. The values on the left (resp. right) side of the slashes are obtained without (resp. with) the fine-tuning of SRNN in the E-Z Step. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 105

5.1 SC-ASS results for short ($T = 50$), medium ($T = 100$), and long ($T = 300$) sequences. . . . . . . . . . . . . . . . . . . . . . . . . . . . . 116
5.2 Capacity of the SRNN model pre-trained at three data scales of the WSJ0 and the CBF datasets. SRNN-full, SRNN-half, and SRNN-quarter stand for SRNN pre-trained on the totality, half of and quarter of our original training set, respectively. . . . . . . . . . . . . . . . . . . . . . . . . . 119
5.3 SC-ASS results obtained by MixDVAE with SRNN pre-trained at the three data scales. The results are reported for the short sequence test subset ($T = 50$). . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 119
5.4 SC-ASS results obtained by MixDVAE with and without the fine-tuning of SRNN. The results are reported for the short ($T = 50$), medium ($T = 100$) and long ($T = 300$) test sequence subsets. . . . . . . . . . . . . . . 120
5.5 Averaged processing time per sequence for the MOT task. . . . . . . . . 122
5.6 Pre-training computational cost on different datasets at different scales. . 122

6.1 SE results for different variants of the dynamical DPGM model. U-NA stands for unsupervised noise-agnostic and U-ND stands for unsupervised noise-dependent, U-NDA stands for U-ND training followed by noise adaptation fine-tuning. The best scores are in bold and the second best scores are underlined. . . . . . . . . . . . . . . . . . . . . . . . . . . 137
6.2 SE results compared with baselines. S stands for supervised, U-NA stands for unsupervised noise-agnostic and U-ND stands for unsupervised noise-dependent. The best scores are in bold and the second best scores are underlined. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 138
6.3 RTF of different models during inference. The best scores are in bold. . . 140

6.4 SE results compared with baselines. U-NA stands for unsupervised noise-agnostic and U-ND stands for unsupervised noise-dependent. The best scores are in bold and the second best scores are underlined. . . . . . . . 144

7.1 Speech spectrograms analysis-resynthesis results. . . . . . . . . . . . . . . 159
7.2 Speech spectrograms analysis-resynthesis results: Ablation studies on the HiT/LigHT-DVAE models structure. . . . . . . . . . . . . . . . . . . . . . . 161
7.3 Speech spectrograms generation results. . . . . . . . . . . . . . . . . . . . 163

# LIST OF ALGORITHMS

1 MixDVAE algorithm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 88
2 Cascade initialization of the position vector sequence . . . . . . . . . . . 187
3 Synthetic trajectories generation . . . . . . . . . . . . . . . . . . . . . . . 188

# BIBLIOGRAPHY

[1] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325*, 2023.

[2] Nasim Alamdari, Arian Azarang, and Nasser Kehtarnavaz. Improving deep speech denoising by noisy2noisy signal mapping. *Applied Acoustics*, 172:107631, 2019.

[3] Dario Amodei, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Qiang Cheng, Guoliang Chen, Jie Chen, Jingdong Chen, Zhijie Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Ke Ding, Niandong Du, Erich Elsen, Jesse Engel, Weiwei Fang, Linxi Fan, Christopher Fougner, Liang Gao, Caixia Gong, Awni Hannun, Tony Han, Lappi Johannes, Bing Jiang, Cai Ju, Billy Jun, Patrick LeGresley, Libby Lin, Junjie Liu, Yang Liu, Weigao Li, Xiangang Li, Dongpeng Ma, Sharan Narang, Andrew Ng, Sherjil Ozair, Yiping Peng, Ryan Prenger, Sheng Qian, Zongfeng Quan, Jonathan Raiman, Vinay Rao, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Kavya Srinet, Anuroop Sriram, Haiyuan Tang, Liliang Tang, Chong Wang, Jidong Wang, Kaifu Wang, Yi Wang, Zhijian Wang, Zhiqian Wang, Shuang Wu, Likai Wei, Bo Xiao, Wen Xie, Yan Xie, Dani Yogatama, Bin Yuan, Jun Zhan, and Zhenyao Zhu. Deep speech 2 : End-to-end speech recognition in english and mandarin. In *Proceedings of the International Conference on Machine Learning*, pages 173–182. PMLR, 2016.

[4] Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. People-tracking-by-detection and people-detection-by-tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[5] Maryam Babaee, Zimu Li, and Gerhard Rigoll. Occlusion handling in tracking multiple people using rnn. In *IEEE International Conference on Image Processing*, pages 2715–2719, 2018.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2016.

[7] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[8] Yutong Ban, Xavier Alameda-Pineda, Laurent Girin, and Radu Horaud. Variational bayesian inference for audio-visual tracking of multiple speakers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1761–1776, 2021.

[9] Yoshiaki Bando, Masato Mimura, Katsutoshi Itoyama, Kazuyoshi Yoshii, and Tatsuya Kawahara. Statistical speech enhancement based on probabilistic integration of variational autoencoder and non-negative matrix factorization. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018.

[10] Yaakov Bar-Shalom, Peter K. Willett, and Xin Tian. *Tracking and Data Fusion: A Handbook of Algorithms*. YBS Publishing, 2011.

[11] Leonard E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. In *Inequalities III: Proceedings of the Third Symposium on Inequalities*, pages 1–8. Academic Press, 1972.

[12] Anthony J. Bell and Terrence J. Sejnowski. An information-maximization approach

to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[13] Jacob Benesty, Shoji Makino, and Jingdong Chen. *Speech Enhancement*. Springer Berlin, Heidelberg, 2005.

[14] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2015.

[15] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[16] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixé. Tracking without bells and whistles. In *IEEE/CVF International Conference on Computer Vision*, pages 941–951, 2019.

[17] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The CLEAR MOT metrics. *EURASIP Journal on Image and Video Processing*, 2008:10, 2008.

[18] Dimitris Bertsimas, Colin Pawlowski, and Ying Daisy Zhuo. From predictive methods to missing data imputation: An optimization approach. *Journal of Machine Learning Research*, 18(196):1–39, 2018.

[19] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *IEEE International Conference on Image Processing*, pages 3464–3468, 2016.

[20] Xiaoyu Bie, Laurent Girin, Simon Leglaive, Thomas Hueber, and Xavier Alameda-Pineda. A benchmark of dynamical variational autoencoders applied to speech spectrogram modeling. In *Proceedings Interspeech Conference*, 2021.

[21] Xiaoyu Bie, Wen Guo, Simon Leglaive, Laurent Girin, Francesc Moreno-Noguer, and Xavier Alameda-Pineda. HiT-DVAE: Human motion generation via Hierarchical Transformer Dynamical VAE. *arXiv preprint arxiv:2204.01565*, 2022.

[22] Xiaoyu Bie, Simon Leglaive, Xavier Alameda-Pineda, and Laurent Girin. Unsupervised speech enhancement using dynamical variational autoencoders. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:2993–3007, 2022.

[23] Simon Bing, Vincent Fortuin, and Gunnar Ratsch. On disentanglement in gaussian process variational autoencoders. In *Symposium on Advances in Approximate Bayesian Inference*, 2022.

[24] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer New York, NY, 1 edition, 2006.

[25] Mikołaj Bińkowski, Jeff Donahue, Sander Dieleman, Aidan Clark, Erich Elsen, Norman Casagrande, Luis C. Cobo, and Karen Simonyan. High fidelity speech synthesis with adversarial networks. In *International Conference on Learning Representations*, 2020.

[26] Samuel S. Blackman and Robert Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House radar library. Artech House, 1999.

[27] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[28] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu,

Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2022.

[29] Albert S. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound*. The MIT Press, 1990.

[30] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2020.

[31] Guillaume Carbajal, Julius Richter, and Timo Gerkmann. Guided variational autoencoder for speech enhancement with a supervised classifier. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2021.

[32] Jean-Francois Cardoso and Antoine Souloumiac. Blind beamforming for non-gaussian signals. *IEE Proceedings F (Radar and Signal Processing)*, 140(6):362–370, December 1993.

[33] Francesco Paolo Casale, Adrian Dalca, Luca Saglietti, Jennifer Listgarten, and Nicolo Fusi. Gaussian process prior variational autoencoders. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018.

[34] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In *Advances in Neural Information Processing Systems*, pages 15084–15097. Curran Associates, Inc., 2021.

[35] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, S. Arun Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[36] E. Colin Cherry. Some experiments on the recognition of speech, with one and with two ears. *Journal of the Acoustical Society of America*, 25:975–979, 1953.

[37] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of*

*the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2014.

[38] Gioele Ciaparrone, Francisco Luque Sánchez, Siham Tabik, Luigi Troiano, Roberto Tagliaferri, and Francisco Herrera. Deep learning in video multi-object tracking: A survey. *Neurocomputing*, 381:61–88, 2020.

[39] Pierre Comon and Christian Jutten. *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic Press, 2010.

[40] Alp Kucukelbir David M. Blei and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[41] David Dean, Ahilan Kanagasundaram, Houman Ghaemmaghami, Md. Hafizur Rahman, and Sridha Sridharan. The QUT-NOISE-SRE protocol for the evaluation of noisy speaker recognition. In *Proceedings Interspeech Conference*, 2015.

[42] Alexandre Défossez, Jade Copet, Gabriel Synnaeve, and Yossi Adi. High fidelity neural audio compression. *Transactions on Machine Learning Research*, 2023. Featured Certification, Reproducibility Certification.

[43] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[44] Patrick Dendorfer, Aljoša Ošep, Anton Milan, Konrad Schindler, Daniel Cremers, Ian Reid, Stefan Roth, and Laura Leal-Taixé. MOTChallenge: A benchmark for single-camera multiple target tracking. *International Journal of Computer Vision*, 129:845–881, 2021.

[45] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for*

*Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 2019.

[46] Caglayan Dicle, Octavia I. Camps, and Mario Sznaier. The way they move: Tracking multiple targets with similar appearance. In *IEEE International Conference on Computer Vision*, pages 2304–2311, 2013.

[47] Peter J. Diggle and Richard J. Gratton. Monte carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 46(2):193–227, 1984.

[48] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[49] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[50] Garry A Einicke and Langford B White. Robust extended Kalman filtering. *IEEE Transactions on Signal Processing*, 47(9):2596–2599, 1999.

[51] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[52] Cédric Févotte, Emmanuel Vincent, and Alexey Ozerov. *Single-Channel Audio Source Separation with NMF: Divergences, Constraints and Algorithms*, pages 1–24. Springer International Publishing, 2018.

[53] Vincent Fortuin, Dmitry Baranchuk, Gunnar Raetsch, and Stephan Mandt. Gp-vae: Deep probabilistic time series imputation. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 1651–1661. PMLR, 2020.

[54] Vincent Fortuin, Matthias Hüser, Francesco Locatello, Heiko Strathmann, and Gunnar Rätsch. Som-vae: Interpretable discrete representation learning on time series. In *International Conference on Learning Representations*, 2019.

[55] Marco Fraccaro, Søren Kaae Sø nderby, Ulrich Paquet, and Ole Winther. Sequential neural models with stochastic layers. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2016.

[56] Michael C Fu. *Handbook of Simulation Optimization*. Springer New York, NY, 2014.

[57] Szu-Wei Fu, Chien-Feng Liao, Yu Tsao, and Shou-De Lin. MetricGAN: Generative adversarial networks based black-box metric scores optimization for speech enhancement. In *Proceedings of the International Conference on Machine Learning*, 2019.

[58] Szu-Wei Fu, Cheng Yu, Tsun-An Hsieh, Peter William VanHarn Plantinga, Mirco Ravanelli, Xugang Lu, and Yu Tsao. MetricGAN+: An improved version of MetricGAN for speech enhancement. In *Proceedings Interspeech Conference*, 2021.

[59] Szu-Wei Fu, Cheng Yu, Kuo-Hsuan Hung, Mirco Ravanelli, and Yu Tsao. MetricGAN-U: Unsupervised speech enhancement / dereverberation based only on noisy / reverberated speech. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2022.

[60] Szu-Wei Fu, Cheng Yu, Kuo-Hsuan Hung, Mirco Ravanelli, and Yu Tsao. MetricGAN-U: Unsupervised speech enhancement-dereverberation based only on noisy/reverberated speech. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2022.

[61] Takuya Fujimura, Yuma Koizumi, Kohei Yatabe, and Ryoichi Miyazaki. Noisy-target training: A training strategy for DNN-based speech enhancement without clean speech. In *Proceedings European Signal Processing Conference*, 2021.

[62] 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.

[63] John S. Garofolo, David Graff, Doug Paul, and David Pallett. CSR-I (WSJ0) Sennheiser LDC93S6B. *Philadelphia: Linguistic Data Consortium*, 1993.

[64] Dan Geiger, Thomas Verma, and Judea Pearl. Identifying independence in Bayesian networks. *Networks*, 20(5):507–534, 1990.

[65] Alan E. Gelfand and Adrian F. M. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85(410):398–409, 1990.

[66] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *Proceedings of the International Conference on Machine Learning*, volume 37, pages 881–889. PMLR, 2015.

[67] Samuel J. Gershman and Noah D. Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, 2014.

[68] Zoubin Ghahramani and Geoffrey E. Hinton. Parameter estimation for linear dynamical systems. *Technical Report, University of Toronto*, 1996.

[69] Zoubin Ghahramani and Geoffrey E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):831–864, 2000.

[70] Zoubin Ghahramani and Michael I. Jordan. Factorial hidden markov models. In *Advances in Neural Information Processing Systems*. MIT Press, 1995.

[71] Laurent Girin, Simon Leglaive, Xiaoyu Bie, Julien Diard, Thomas Hueber, and Xavier Alameda-Pineda. Dynamical variational autoencoders: A comprehensive review. *Foundations and Trends in Machine Learning*, 15(1-2):1–175, 2021.

[72] Laurent Girin, Fanny Roche, Thomas Hueber, and Simon Leglaive. Notes on the use of variational autoencoders for speech and audio spectrogram modeling. In *Proceedings International Conference on Digital Audio Effects (DAFx)*, 2019.

[73] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[74] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2014.

[75] Nathaniel R. Goodman. Statistical Analysis Based on a Certain Multivariate Complex Gaussian Distribution (An Introduction). *The Annals of Mathematical Statistics*, 34(1):152 – 177, 1963.

[76] Anirudh Goyal and Yoshua Bengio. Inductive biases for deep learning of higher-level cognition. *Proc. R. Soc. A*, 478:20210068, 2022.

[77] Robert Gray. Vector quantization. *IEEE ASSP Magazine*, 1(2):4–29, 1984.

[78] Ulf Grenander and Michael I. Miller. Representations of knowledge in complex systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 56(4):549–603, 1994.

[79] Daniel W. Griffin and Jae S. Lim. Signal estimation from modified short-time Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.

[80] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. Conformer: Convolution-augmented Transformer for Speech Recognition. In *Proceedings Interspeech Conference*, pages 5036–5040, 2020.

[81] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Interna-*

*tional Conference on Artificial Intelligence and Statistics*, pages 297–304. PMLR, 2010.

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

[83] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[84] John R. Hershey, Zhuo Chen, Jonathan Le Roux, and Shinji Watanabe. Deep clustering: Discriminative embeddings for segmentation and separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 31–35, 2016.

[85] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

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

[87] Geoffrey E. Hinton. Learning multiple layers of representation. *Trends in Cognitive Sciences*, 11(10):428–34, 2007.

[88] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, pages 6840–6851. Curran Associates, Inc., 2020.

[89] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[90] Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933.

[91] Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017.

[92] Rongjie Huang, Jiawei Huang, Dongchao Yang, Yi Ren, Luping Liu, Mingze Li, Zhenhui Ye, Jinglin Liu, Xiang Yin, and Zhou Zhao. Make-an-audio: Text-to-audio generation with prompt-enhanced diffusion models. In *Proceedings of the International Conference on Machine Learning*, pages 13916–13932. PMLR, 2023.

[93] Rongjie Huang, Max W. Y. Lam, Jun Wang, Dan Su, Dong Yu, Yi Ren, and Zhou Zhao. Fastdiff: A fast conditional diffusion model for high-quality speech synthesis. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 4157–4163, 2022.

[94] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005.

[95] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4):411–430, 2000.

[96] Metod Jazbec, Matt Ashman, Vincent Fortuin, Michael Pearce, Stephan Mandt, and Gunnar Rätsch. Scalable gaussian process variational autoencoders. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, pages 3511–3519. PMLR, 2021.

[97] Metod Jazbec, Michael Arthur Leopold Pearce, and Vincent Fortuin. Factorized gaussian process variational autoencoders. In *Symposium on Advances in Approximate Bayesian Inference*, 2021.

[98] Qiang Ji. *Probabilistic Graphical Models for Computer Vision*. Academic Press, Oxford, 2020.

[99] Ian T. Jolliffe. *Principal Component Analysis*. Springer New York, NY, 2002.

[100] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

[101] Rudolf Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[102] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[103] Madhav Mahesh Kashyap, Anuj Tambwekar, Krishnamoorthy Manohara, and S. Natarajan. Speech denoising without clean training data: a noise2noise approach. In *Proceedings Interspeech Conference*, 2021.

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

[105] Diederik P. Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2014.

[106] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, page 4743–4751. Curran Associates Inc., 2016.

[107] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *International Conference on Learning Representations*, 2014.

[108] John F. Kolen and Stefan C. Kremer. *Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies*, pages 237–243. IEEE, 2001.

[109] Rahul Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. In *Advances in Approximate Bayesian Inference*, December 2015.

[110] Harold W Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[111] Amos Lapidoth. *A Foundation in Digital Communication*. Cambridge University Press, Cambridge, UK, second edition, February 2017.

[112] Clément Laroche, Matthieu Kowalski, Hélène Papadopoulos, and Gaël Richard. A structured nonnegative matrix factorization for source separation. In *European Signal Processing Conference*, pages 2033–2037, 2015.

[113] Clément Laroche, Hélène Papadopoulos, Matthieu Kowalski, and Gaël Richard. Drum extraction in single channel audio signals using multi-layer non negative matrix factor deconvolution. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 46–50, 2017.

[114] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. A tutorial on energy-based learning. *Predicting Structured Data*, 2:9, 2006.

[115] Yann LeCun, Larry D. Jackel, Leon Bottou, A. Brunot, Corinna Cortes, John S. Denker, Harris Drucker, Isabelle Guyon, Urs A. Muller, Eduard Sackinger, Patrice Simard, and Vladimir Vapnik. Comparison of learning algorithms for handwritten digit recognition. In *International Conference on Artificial Neural Networks*, pages 53–60, 1995.

[116] Simon Leglaive, Xavier Alameda-Pineda, Laurent Girin, and Radu Horaud. A recurrent variational autoencoder for speech enhancement. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020.

[117] Simon Leglaive, Laurent Girin, and Radu Horaud. A variance modeling framework based on variational autoencoders for speech enhancement. In *Proceedings Workshop on Machine Learning for Signal Processing*, 2018.

[118] Simon Leglaive, Laurent Girin, and Radu Horaud. Semi-supervised multichannel speech enhancement with variational autoencoders and non-negative matrix factor-

ization. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019.

[119] Simon Leglaive, Umut Şimşekli, Antoine Liutkus, Laurent Girin, and Radu Horaud. Speech enhancement with variational autoencoders and alpha-stable distributions. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019.

[120] Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 3870–3882. PMLR, 2020.

[121] Yiming Liang and Yue Zhou. LSTM multiple object tracker combining multiple cues. In *IEEE International Conference on Image Processing*, pages 2351–2355, 2018.

[122] Zachary C. Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[123] Qiang Liu, Jason Lee, and Michael Jordan. A kernelized stein discrepancy for goodness-of-fit tests. In *Proceedings of the International Conference on Machine Learning*, pages 276–284. PMLR, 2016.

[124] Antoine Liutkus, Roland Badeau, and Gäel Richard. Gaussian processes for underdetermined source separation. *IEEE Transactions on Signal Processing*, 59(7):3155–3167, 2011.

[125] Philipos C Loizou. *Speech enhancement: Theory and practice*. CRC press, 2007.

[126] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*, 2017.

[127] Yen-Ju Lu, Zhong-Qiu Wang, Shinji Watanabe, Alexander Richard, Cheng Yu, and Yu Tsao. Conditional diffusion probabilistic model for speech enhancement. In

*IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 7402–7406, 2022.

[128] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. Multiple object tracking: A literature review. *Artificial Intelligence*, 293:103448, 2021.

[129] Yi Luo, Zhuo Chen, and Takuya Yoshioka. Dual-path rnn: Efficient long sequence modeling for time-domain single-channel speech separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 46–50, 2020.

[130] Yi Luo and Nima Mesgarani. Tasnet: Time-domain audio separation network for real-time, single-channel speech separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 696–700, 2018.

[131] Yi Luo and Nima Mesgarani. Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(8):1256–1266, 2019.

[132] Thang Luong, Hieu Pham, and Christopher D. Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2015.

[133] James MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Berkeley symposium on mathematical statistics and probability*, pages 281–297, 1967.

[134] Ronald P. S. Mahler. *Statistical Multisource-Multitarget Information Fusion*. Artech House, Inc., 2007.

[135] Santiago Manen, Michael Gygli, Dengxin Dai, and Luc Van Gool. Pathtrack: Fast trajectory annotation with path supervision. In *IEEE International Conference on Computer Vision*, pages 290–299, 2017.

[136] Geoff J. McLachlan and Kaye E. Basford. *Mixture models: Inference and applications to clustering*. Marcel Dekker, 1988.

[137] Geoffrey J. McLachlan, Sharon X. Lee, and Suren I. Rathnayake. Finite mixture models. *Annual Review of Statistics and Its Application*, 6(Volume 6, 2019):355–378, 2019.

[138] Anton Milan, S. Hamid Rezatofighi, Anthony Dick, Ian Reid, and Konrad Schindler. Online multi-target tracking using recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, page 4225–4232. AAAI Press, 2017.

[139] Kevin P. Murphy. *Switching Kalman Filters*. Citeseer, 1998.

[140] Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[141] Kevin P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023.

[142] Gautham J. Mysore and Paris Smaragdis. A non-negative approach to semi-supervised separation of speech from noise with the use of temporal dynamics. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 17–20, 2011.

[143] Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob Mcgrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards photorealistic image generation and editing with text-guided diffusion models. In *Proceedings of the International Conference on Machine Learning*, pages 16784–16804. PMLR, 2022.

[144] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Haiquan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. A conversational paradigm for program synthesis. *arXiv preprint arXiv:2203.13474*, 2022.

[145] Aäron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, page 4797–4805. Curran Associates Inc., 2016.

[146] OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[147] Alexey Ozerov and Cédric Fevotte. Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(3):550–563, 2010.

[148] Alexey Ozerov, Cédric Févotte, and Maurice Charbit. Factorial scaled hidden markov model for polyphonic audio representation and source separation. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 121–124, 2009.

[149] Alexey Ozerov, Antoine Liutkus, Roland Badeau, and Gaël Richard. Coding-based informed source separation: Nonnegative tensor factorization approach. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(8):1699–1712, 2013.

[150] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

[151] Manuel Pariente, Antoine Deleforge, and Emmanuel Vincent. A statistically principled and computationally efficient approach to speech enhancement using variational autoencoders. In *Proceedings Interspeech Conference*, Graz, Austria, 2019.

[152] Giorgio Parisi. Correlation functions and computer simulations. *Nuclear Physics B*, 180(3):378–384, 1981.

[153] Giorgio Parisi. *Statistical Field Theory*. Addison-Wesley, 1988.

[154] Santiago Pascual, Antonio Bonafonte, and Joan Serrà. SEGAN: Speech enhancement generative adversarial network. In *Proceedings Interspeech Conference*, 2017.

[155] Jonas Peters, Dominik Janzing, and Bernhard Schlkopf. *Elements of Causal Inference: Foundations and Learning Algorithms*. The MIT Press, 2017.

[156] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[157] Lawrence R. Rabiner and Biinghwang Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.

[158] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

[159] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[160] Rajesh Ranganath, Sean Gerrish, and David Blei. Black Box Variational Inference. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 814–822. PMLR, 2014.

[161] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[162] H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, 1965.

[163] Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

[164] Chandan K A Reddy, Vishak Gopal, and Ross Cutler. Dnsmos: A non-intrusive perceptual objective speech quality metric to evaluate noise suppressors. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6493–6497, 2021.

[165] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[166] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2015.

[167] Yi Ren, Yangjun Ruan, Xu Tan, Tao Qin, Sheng Zhao, Zhou Zhao, and Tie-Yan Liu. Fastspeech: Fast, robust and controllable text to speech. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

[168] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

[169] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the International Conference on Machine Learning*, pages 1278–1286. PMLR, 2014.

[170] Julius Richter, Simon Welker, Jean-Marie Lemercier, Bunlong Lay, and Timo Gerkmann. Speech enhancement and dereverberation with diffusion-based generative models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:2351–2364, 2023.

[171] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *Proceedings of the European Conference on Computer Vision*, pages 17–35. Springer, 2016.

[172] Antony Rix, John Beerends, Michael Hollier, and Andries Hekstra. Perceptual evaluation of speech quality (PESQ) - A new method for speech quality assessment

of telephone networks and codecs. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2001.

[173] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951.

[174] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer New York, NY, 2 edition, 2004.

[175] Jonathan Le Roux, Scott Wisdom, Hakan Erdogan, and John R. Hershey. SDR – Half-baked or well done? In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019.

[176] Mostafa Sadeghi and Romain Serizel. Fast and efficient speech enhancement with variational autoencoders. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1–5, 2023.

[177] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. In *IEEE/CVF International Conference on Computer Vision*, pages 300–311, 2017.

[178] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *Advances in Neural Information Processing Systems*, pages 36479–36494. Curran Associates, Inc., 2022.

[179] Fatemeh Saleh, Sadegh Aliakbarian, Hamid Rezatofighi, Mathieu Salzmann, and Stephen Gould. Probabilistic tracklet scoring and inpainting for multiple object tracking. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.

[180] Mona Schirmer, Mazin Eltayeb, Stefan Lessmann, and Maja Rudolph. Modeling

irregular time series with continuous recurrent units. In *Proceedings of the International Conference on Machine Learning*, pages 19388–19405. PMLR, 2022.

[181] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[182] Paris Smaragdis, Bhiksha Raj, and Madhusudana Shashanka. Supervised and semi-supervised separation of sounds from single-channel mixtures. In *Proceedings of the International Conference on Independent Component Analysis and Signal Separation*, page 414–421. Springer-Verlag, 2007.

[183] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[184] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, 2016.

[185] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

[186] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In *Advances in Neural Information Processing Systems*, pages 12438–12448. Curran Associates, Inc., 2020.

[187] Yang Song, Sahaj Garg, Jiaxin Shi, and Stefano Ermon. Sliced score matching: A scalable approach to density and score estimation. In *Uncertainty in Artificial Intelligence*, pages 574–584, 2020.

[188] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[189] Fabian-Robert Stöter, Stefan Uhlich, Antoine Liutkus, and Yuki Mitsufuji. Open-Unmix – A reference implementation for music source separation. *Journal of Open Source Software*, 4(41):1667, 2019.

[190] Nicolas Sturmel, Antoine Liutkus, Jonathan Pinel, Laurent Girin, Sylvain Marchand, Gaël Richard, Roland Badeau, and Laurent Daudet. Linear mixing models for active listening of music productions in realistic studio conditions. In *Audio Engineering Society Convention 132*, 2012.

[191] Cem Subakan, Mirco Ravanelli, Samuele Cornell, Mirko Bronzi, and Jianyuan Zhong. Attention is all you need in speech separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 21–25, 2021.

[192] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2014.

[193] Simo Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

[194] Simo Särkkä and Arno Solin. *Applied Stochastic Differential Equations*. Cambridge University Press, 2019.

[195] Cees H. Taal, Richard C. Hendriks, Richard Heusdens, and Jesper Jensen. An algorithm for intelligibility prediction of time–frequency weighted noisy speech. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2125–2136, 2011.

[196] Yee Whye Teh, Max Welling, Simon Osindero, and Geoffrey E. Hinton. Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4:1235–1260, 2003.

[197] Joachim Thiemann, Nobutaka Ito, and Emmanuel Vincent. The diverse environments multi-channel acoustic noise database (DEMAND): A database of multi-

channel environmental noise recordings. *The Journal of the Acoustical Society of America*, 133:3591–3591, 2013.

[198] 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.

[199] Efthymios Tzinis, Shrikant Venkataramani, Zhepei Wang, Cem Subakan, and Paris Smaragdis. Two-step sound source separation: Training on learned latent targets. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 31–35, 2020.

[200] Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *Journal of Machine Learning Research*, 17(205):1–37, 2016.

[201] Cassia Valentini-Botinhao, Xin Wang, Shinji Takaki, and Junichi Yamagishi. Investigating RNN-based speech enhancement methods for noise-robust text-to-speech. In *Proceedings Speech Synthesis Workshop*, 2016.

[202] Rafael Valle, Kevin J. Shih, Ryan Prenger, and Bryan Catanzaro. Flowtron: an autoregressive flow-based generative network for text-to-speech synthesis. In *International Conference on Learning Representations*, 2021.

[203] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. In *Proceedings of Speech Synthesis Workshop*, page 125, 2016.

[204] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 1747–1756. PMLR, 2016.

[205] Aäron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural discrete repre-

sentation learning. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017.

[206] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Information Science and Statistics. Springer New York, NY, 2 edition, 2000.

[207] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017.

[208] Christophe Veaux, Junichi Yamagishi, and Simon King. The Voice Bank corpus: Design, collection and data analysis of a large regional accent speech database. In *Proceedings of International Committee for Co-ordination and Standardisation of Speech Databases*, 2013.

[209] Emmanuel Vincent, Sharon Gannot, and Tuomas Virtanen. *Audio Source Separation and Speech Enhancement*. John Wiley & Sons, Ltd, 2018.

[210] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Computation*, 23(7):1661–1674, 2011.

[211] Tuomas Virtanen. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(3):1066–1074, 2007.

[212] Ba-ngu Vo, Mahendra Mallick, Yaakov Bar shalom, Stefano Coraluppi, Richard Osborne III, Ronald Mahler, and Ba-tuong Vo. *Multitarget Tracking*, pages 1–15. John Wiley & Sons, Ltd, 2015.

[213] Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

[214] Xingyu Wan, Jinjun Wang, and Sanping Zhou. An online and flexible multi-object tracking framework using long short-term memory. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1311–13118, 2018.

[215] Changhong Wang, Emmanouil Benetos, Vincent Lostanlen, and Elaine Chew. Adaptive scattering transforms for playing technique recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:1407–1421, 2022.

[216] DeLiang Wang and Guy J. Brown. *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*. Wiley-IEEE Press, 2006.

[217] DeLiang Wang and Jitong Chen. Supervised speech separation based on deep learning: An overview. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(10):1702–1726, 2018.

[218] Simon Welker, Julius Richter, and Timo Gerkmann. Speech Enhancement with Score-Based Generative Models in the Complex STFT Domain. In *Proceedings Interspeech Conference*, pages 2928–2932, 2022.

[219] R. Jonald Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989.

[220] Scott Wisdom, Efthymios Tzinis, Hakan Erdogan, Ron J. Weiss, Kevin Wilson, and John R. Hershey. Unsupervised sound separation using mixture invariant training. In *Proceedings of the International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2020.

[221] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *IEEE International Conference on Image Processing*, pages 3645–3649, 2017.

[222] Jun Xiang, Guoshuai Zhang, and Jianhua Hou. Online multi-object tracking based on feature representation and Bayesian filtering within a deep learning architecture. *IEEE Access*, pages 27923–27935, 2019.

[223] Yang Xiang and Changchun Bao. A parallel-data-free speech enhancement method using multi-objective learning cycle-consistent generative adversarial network. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:1826–1838, 2020.

[224] Yang Xiang, Jesper Lisby Højvang, Morten Højfeldt Rasmussen, and Mads Græsbøll Christensen. A bayesian permutation training deep representation learning method for speech enhancement with variational autoencoder. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 381–385, 2022.

[225] Bo Yang and Ram Nevatia. Multi-target tracking by online learning of non-linear motion patterns and robust appearance models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1918–1925, 2012.

[226] Fan Yang, Wongun Choi, and Yuanqing Lin. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2129–2137, 2016.

[227] Ozgur Yilmaz and Scott Rickard. Blind separation of speech mixtures via time-frequency masking. *IEEE Transactions on Signal Processing*, 52(7):1830–1847, 2004.

[228] Dong Yu and Li Deng. *Automatic speech recognition.* Springer, 2016.

[229] Dong Yu, Morten Kolbæk, Zheng-Hua Tan, and Jesper Jensen. Permutation invariant training of deep models for speaker-independent multi-talker speech separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 241–245, 2017.

[230] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations*, 2016.

[231] Guochen Yu, Yutian Wang, Chengshi Zheng, Hui Wang, and Qin Zhang. CycleGAN-based non-parallel speech enhancement with an adaptive attention-in-attention mechanism. In *Asia-Pacific Signal and Information Processing Association Annual Conference*, 2021.

[232] Paul Zarchan. *Progress in astronautics and aeronautics: fundamentals of Kalman filtering: a practical approach*, volume 208. Aiaa, 2005.

[233] Neil Zeghidour and David Grangier. Wavesplit: End-to-end speech separation by speaker clustering. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:2840–2849, 2021.

[234] Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola. Dive into Deep Learning. *arXiv preprint arXiv:2106.11342*, 2023.

[235] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision*, 129(11):3069–3087, 2021.

[236] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Infovae: Balancing learning and inference in variational autoencoders. *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5885–5892, 2019.

[237] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *Proceedings European Conference on Computer Vision*, 2020.

[238] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 08 2017.

[239] Harrison Zhu, Carles Balsells-Rodas, and Yingzhen Li. Markovian Gaussian process variational autoencoders. In *Proceedings of the International Conference on Machine Learning*, pages 42938–42961. PMLR, 2023.

[240] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision*, pages 2242–2251, 2017.

[241] Xiaojin (Jerry) Zhu. Semi-supervised learning literature survey. Technical Report TR1530, University of Wisconsin-Madison Department of Computer Sciences, 2005.

[242] Bernt Øksendal. *Stochastic Differential Equations*. Springer Berlin, Heidelberg, 6 edition, 2003.