EXPLORING AND EXPLOITING STRUCTURE AND SELF-SUPERVISION IN SEQUENCE LEARNING

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Junwen Bai

August 2022

EXPLORING AND EXPLOITING STRUCTURE AND SELF-SUPERVISION IN SEQUENCE LEARNING

Junwen Bai, Ph.D.

Cornell University 2022

Sequence data, which consists of values organized in a certain order, is one of the most commonly seen data types in our everyday lives. For instance, the daily temperature and precipitation measurements throughout a year form a sequence of weather conditions. The crop yields each year over the past several decades depict a trend in agricultural production. These are also known as time series data. Time-indexed data is not the only kind of sequence data. Linguistic data such as speech and texts are sequential in nature. DNA sequences are indexed based on the physical order of the bases and materials' density of states are indexed by energy levels. In fact, any reasonably ordered data can be viewed as a sequence.

Sequence data has been a long-standing area of interest in the artificial intelligence (AI) field, and this class of problems is often called *sequence learning*. Various kinds of sequence learning tasks have been defined, such as predicting the general properties of a sequence, tagging the sequence with labels at each index, or generating a new sequence from the input. Different types of sequence data have unique structures, and it is often challenging to develop a model to encode or decode the inherent sequential relationship, so each sub-field has historically relied on separate sequence learning tools and frameworks. However, recent advances in machine learning (ML) and deep neural networks (DNN) have provided the capacity to handle arbitrarily long sequences and store his-

torical states in a more unified fashion, regardless of the modality of the data. Deep models like recurrent neural networks (RNN), long short-term memory (LSTM), gated recurrent units (GRU) and transformers have become the foundation of most modern sequence learning and feature extraction methods. A new challenge is to efficiently and effectively utilize these deep models to capture intrinsic features from the input sequences.

In this thesis, I will study both supervised and self-supervised sequence learning using deep models. Conventional methods for supervised sequence learning are typically designed to study sequences of scalar values or vectors, and are not suitable for structured data such as graphs. I will illustrate three novel yet challenging scenarios involving graphs and sequences: dynamic node property prediction for a fixed graph, sequence prediction from a graph, and multi-label prediction of sequence inputs. Structured input data can be modeled using a framework that combines graph neural networks (GNN) with sequence models (e.g., GRU and transformer). This framework is validated on several tasks, including crop yield prediction and density of states prediction.

Self-supervised learning is another trending direction in the sequence learning field. Self-supervision obtains supervisory signals from the data itself and leverages the underlying structure in the data. It has the potential to improve the sample efficiency for downstream tasks and contribute to better model interpretability. In recent years, self-supervised sequence learning has been successfully applied to language and acoustic model pretraining. In my thesis, I will demonstrate that self-supervision can enforce latent structure, disentangle static and dynamic factors, and supplement supervised signals in model training, by applying it to speech recognition, video understanding and sequence generation.

In general, I will show in this thesis different methods to capture and exploit

structure from sequence data and diverse explorations of the self-supervision for sequence learning.

BIOGRAPHICAL SKETCH

Junwen Bai works with Professor Carla P. Gomes for his Ph.D. study in the Department of Computer Science at Cornell University. His research covers machine learning, signal processing, and language technology, focusing on sequence representation learning and probabilistic modeling, often under scenarios with low supervision. Junwen Bai has developed scalable and general machine learning methods for real-world problems including automatic speech recognition, computational sustainability, and scientific discovery. His work received an innovative application award at AAAI-17, was featured as one of the top 10 coolest army science and technology advances in 2019, and won the best ML innovation paper at the NeurIPS Climate Change workshop in 2021. Prior to Cornell, Junwen earned his bachelor's degree in computer science from Zhiyuan College, Shanghai Jiao Tong University in 2017.

Dedicated to my parents, sister, nephew and future darling.

ACKNOWLEDGEMENTS

I would like to first express my sincerest gratitude to my advisor, Professor Carla P. Gomes, for her vision, enthusiasm, guidance, kindness and encouragement. Throughout my Ph.D. career, Carla provided strong support for my research, including numerous interesting projects, abundant hardware resources and opportunities to attend many academic events. She encouraged me to explore the machine learning directions that interested me and gave invaluable suggestions and guidance. Also, I want to thank my other committee members, Professor Bart Selman and Professor Austin Benson, for their continuous help and patient mentoring. Furthermore, I appreciate Professor John Hopcroft for his massive contributions to both my research and education, as well as his high-level advice on my career path.

I also wish to thank my internship advisors, Dr. Weiran Wang, Dr. Bo Li and Dr. Yu Zhang, for constructive discussions on cutting-edge research topics. Their profound insights, diverse backgrounds and wide knowledge inspired my research directions and expanded my view of the machine learning field.

My sincere thanks are also extended to my dear colleagues and fantastic collaborators, Shufeng Kong, Di Chen, Joshua Fan, John M. Gregoire, Zhiyun Li, Ariel Ortiz-Bobea, Sebastian Ament, Richard Bernstein, Yu-An Chung, Yexiang Xue, Johan Björck, Brendan Rappazzo, Guillaume Perez, Santosh K. Suram, Dieqiao Feng, Yiwei Bai, Qinru Shi, Wenting Zhao, Andrew Allyn, Zihang Lai, Runzhe Yang, Guandao Yang, Tianyi Zhang, Robert B. van Dover, Ronan Le Bras, Ankur Bapna, Yingbo Zhou, Caiming Xiong and Tara N. Sainath, for helping me through different stages of my Ph.D. studies. My research and thesis would not have been possible without their enlightenment, inspiration and constant support.

Furthermore, I want to thank my parents, sister, brother-in-law, and my lovely nephew, for their understanding and encouragement in the past five years. Due to the pandemic, I haven't been home for years. But I was always backed by my supportive family, which motivated me to pursue my research goals.

My research was supported by NSF Expedition awards for Computational Sustainability (CCF-1522054, CNS-1059284 and CNS-0832782), NSF CRI CNS-1059284, NSF OIA-1936950, MURI FA9550-18-1-0136, DE-SC0020383, USDA Co-operative Agreement 58-6000-9-0041, NIFA Hatch Project 1017421, NSF INSPIRE IIS-1344201 and ARO DURIP W911NF-17-1-0187.

TABLE OF CONTENTS

Biographical Sketch . . . iii
Dedication . . . iv
Acknowledgements . . . v
Table of Contents . . . vii
List of Tables . . . ix
List of Figures . . . x

1 Introduction **1**
1.1 Supervised Sequence Learning for Structured Data . . . 8
1.2 Self-supervised Sequence Representation Learning . . . 12

2 Exploring and Exploiting Structures in Sequence Learning **20**
2.1 Sequence-on-Graph: Harnessing Geospatial and Temporal Information for Crop Yield Prediction . . . 20
2.1.1 Introduction . . . 20
2.1.2 GNN-RNN . . . 23
2.1.3 Prior Works . . . 28
2.1.4 Experiments on Crop Yield Prediction . . . 30
2.1.5 Discussion . . . 47
2.2 Graph-to-Sequence: Crystal to Sequence Learning for Density of States Prediction . . . 48
2.2.1 Introduction . . . 48
2.2.2 Crystal-to-DoS Learning . . . 51
2.2.3 Related Works . . . 58
2.2.4 Experiments on DoS Prediction . . . 60
2.2.5 Discussion . . . 68
2.3 Sequence-to-Graph: Multi-Label Classification on Text Data and Others . . . 69
2.3.1 Introduction . . . 69
2.3.2 Problem Formulation and Methods . . . 72
2.3.3 Existing MLC Works . . . 80
2.3.4 Experiments on Text and Ecology Data . . . 81
2.3.5 Discussion . . . 90

3 Versatile Self-supversion in Sequence Learning **92**
3.1 Self-supervision for Pretraining: Deep Autoencoding Predictive Components for Sequence Representation Learning . . . 92
3.1.1 Introduction . . . 92
3.1.2 Deep Autoencoding Predictive Components . . . 94
3.1.3 Existing Pretraining Strategies . . . 102
3.1.4 Experiments on Physics System, Forecasting and ASR . . . 104
3.1.5 Discussion . . . 115

3.2 Self-supervision for Disentanglement: Contrastively Disentangled Sequential Variational Audoencoder . . . 115
3.2.1 Introduction . . . 115
3.2.2 C-DSVAE for Sequence Disentanglement . . . 117
3.2.3 Prior Efforts for Disentanglement . . . 125
3.2.4 Experiments on Videos and Audios . . . 128
3.2.5 Discussion . . . 143
3.3 Self-supervision with Supervision: Joint Unsupervised and Supervised Training for Multilingual ASR . . . 144
3.3.1 Introduction . . . 144
3.3.2 Related Work . . . 146
3.3.3 JUST Method for Multilingual ASR . . . 148
3.3.4 Experiments . . . 151
3.3.5 Discussion . . . 157

4 Conclusion **158**

A Appendix for Chapter 2 **161**
A.1 Contrastive Learning Module . . . 161
A.1.1 Connection with Triplet Loss . . . 161
A.1.2 Gradients of Contrastive Loss . . . 162

B Appendix for Chapter 3 **165**
B.1 Appendix for Chapter 3.1 . . . 165
B.1.1 Derivation of Predictive Information . . . 165
B.2 Appendix for Chapter 3.2 . . . 166
B.2.1 Derivation of the per-sequence ELBO . . . 166
B.2.2 Proof of Theorem 1 . . . 167
B.2.3 MWS estimation of $I(s; z_{1:T})$ . . . 169
B.2.4 Contrastive estimation of MI . . . 170
B.2.5 Summary of objective function and estimations . . . 171

LIST OF TABLES

2.1 GNN-RNN evaluation results 40
2.2 Early prediction results (2018 corn 42
2.3 Evaluation results on phDoS 62
2.4 Evaluation results on eDoS 63
2.5 Ablation study on phDoS 64
2.6 Ablation study on eDoS 64
2.7 Training time for RNN-based and transformer-based models . . 65
2.8 ex-F1 and mi-F1 scores 82
2.9 ma-F1 and Hamming scores 83
2.10 Precision@1 scores 84
2.11 Dataset Statistics 84
2.12 Ablation study on the contrastive learning module and the Gaussian mixture module 85
2.13 Comparisons between MPVAE and C-GMVAE using partial data 85
2.14 mi-F1 after replacing the contrastive module with a GNN or a covariance matrix 86

3.1 The R^2 scores of recovered 3D trajectories 105
3.2 The R^2 scores for ablation study 107
3.3 The R^2 scores for reconstruction 107
3.4 The R^2 scores for CPC with different temporal lags 108
3.5 The R^2 scores for variations of DAPC 110
3.6 The R^2 scores for CPC on the temperature dataset 111
3.7 Ablation study on different variants of DAPC 113
3.8 WERs on LibriSpeech 114
3.9 Disentanglement metrics on Sprites 128
3.10 Disentanglement metrics on MUG 128
3.11 Disentanglement metrics on SM-MNIST 130
3.12 Disentanglement metrics on TIMIT 130
3.13 Compare MWS and contrastive estimation of $I(s; x_{1:T})$ and $I(z_{1:T}; x_{1:T})$ on MUG 138
3.14 Compare MWS and contrastive estimations of $I(s; x_{1:T})$ and $I(z_{1:T}; x_{1:T})$ on SM-MNIST 140
3.15 Performance with different batch sizes on TIMIT 141
3.16 Ablation study of the augmentations on SMMNIST 143
3.17 Ablation study of the augmentations on TIMIT. 143
3.18 WERs on MLS for different methods 150
3.19 Weight sensitivity study on β 151

LIST OF FIGURES

1.1 Applications of sequence learning 2
1.2 Mainstream sequence models 4
1.3 Three scenarios of supervised sequence learning 9
1.4 Masking and contrasting 13

2.1 Overview of GNN-RNN 25
2.2 Example of aggregating features to county level 33
2.3 NRCS Soil Texture classification 36
2.4 Maps of predicted and true corn yields in 2018 42
2.5 Predicted vs. ground truth corn yields in 2018 43
2.6 2019 corn: predicted vs true 44
2.7 2019 corn: predicted vs true (scattered plot) 44
2.8 2019 soybean: predicted vs true 45
2.9 2019 soybean: predicted vs true (scattered plot) 45
2.10 DoS prediction 51
2.11 RNN decoding for DoS prediction 54
2.12 Chunk RNN decoding for DoS prediction 54
2.13 Chunk RNN decoding with attention for DoS prediction 55
2.14 Multi-Head Attention 56
2.15 Transformer decoding for DoS prediction 56
2.16 Qualitative results on phDoS 66
2.17 Qualitative results on eDoS 67
2.18 C-GMVAE overview 73
2.19 Label-label inner-products from C-GMVAE 88
2.20 Relative performances on 4 metrics on *ebird* dataset 89

3.1 DAPC overview 95
3.2 Lorenz attractor 106
3.3 Predictive information for reconstruction 108
3.4 Qualitative recovery results 109
3.5 Different models' R^2 score improvements over PCA 109
3.6 DAPC on temperature dataset 110
3.7 C-DSVAE overview 118
3.8 Data augmentations on SM-MNIST and Sprites 123
3.9 Manipulate the static and dynamic factors on Sprites 133
3.10 Generations on Sprites 133
3.11 The random generations of contents and motions 134
3.12 Compare C-DSVAE and S3VAE on MUG 134
3.13 Fix either the static or dynamic factors and replace the other . . . 135
3.14 Compare C-DSVAE and R-WAE on SM-MNIST 136
3.15 Generations on SM-MNIST (motion fixed) 137
3.16 Generations on SM-MNIST (content fixed) 138

3.17 Swap the static and dynamic factors on SM-MNIST 139
3.18 Learning curves of the MI terms 139
3.19 Interpolation in the latent space 140
3.20 MI between s and $z_{1:T}$ 142
3.21 MI between s and $x_{1:T}$ 142
3.22 JUST overview 147
3.23 Model inspection for JUST 154

CHAPTER 1

INTRODUCTION

Revolutionary artificial intelligence (AI) advances in the past ten years have brought machine intelligence to an unprecedented level. Modern AI systems can see [1], read [2], hear [3] and even create [4]. Such huge leap has greatly impacted our everyday lives. Autonomous driving used to be a fantasy, but it has now become one of the most promising industries [5]. Machine translation [6] is gradually abridging the gap between languages. A state-of-the-art automatic speech recognition (ASR) model [7] can identify over ten languages simultaneously. Even in the complicated scientific discovery domain, advanced machine learning techniques have been used to accelerate the process [8, 9, 10] and to aid experts' analyses [11, 12, 13].

The big data era and deep learning age [14] emerged as the exploitation of modern graphics and tensor processing units (i.e., GPU and TPU) [15, 16] made the neural network (NN) training possible, and a huge amount of labeled data [17] became widely available. With simplified and modularized neural networks [18, 19], backpropagation [18] can easily improve the performance of models on a variety of tasks such as computer vision [20, 21, 22, 23, 24, 25], natural language processing [26, 27, 28, 29, 30, 31], speech recognition [32, 33, 34, 35, 36, 37] and scientific discovery [13, 38, 39, 8, 40, 41, 9, 10], as the model size grows larger and larger. Most of these works achieve success by combining massive labeled datasets with deep models. Deep models have large enough capacity to approximate extremely complex functions, and massive amounts of labeled data provide strong enough signals to fit and train these deep models.

Most of these inspiring demonstrations can be categorized into *supervised*

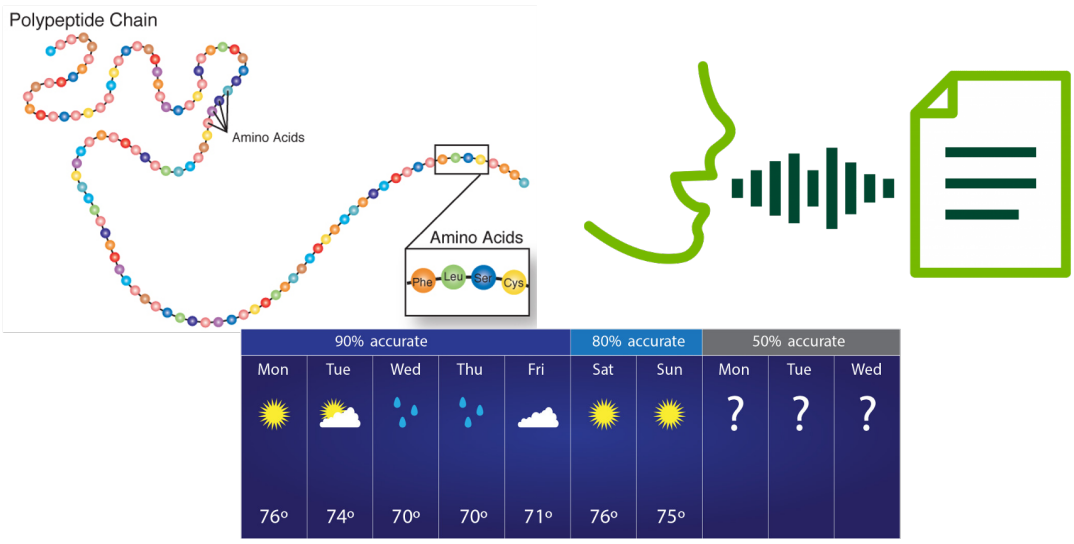


Figure 1.1: Some applications of sequence learning: the prediction of a protein's 3D structure from its amino acid sequence [13] (top left) , speech transcription with automatic speech recognition models [47] (top right) and reliable weather forecasting using deep generative models [48] (bottom).

learning, where the input-output pairs are known in training. Among all kinds of supervised learning tasks, sequence learning is one of the most common and also one of the most challenging supervised learning tasks. For instance, one prevalent and representative type of sequence data is the time series data, where the values of interest are indexed by time [42]. Some examples of time series and their applications include: using annual crop yield or agricultural productivity growth to model the price fluctuation of commodities [43], measuring daily rainfall to understand drought severity [44], studying historic stock price trends to maximize profit and minimize loss [45], and using consecutive healthcare records to infer patients' health condition and predict future medical costs [46].

Time series data is not the only type of sequence data. Video is a sequence of images with strong connections between adjacent frames. Video understanding and generation has become one of the toughest tasks in sequence learning [49, 50]. Speech and text data are also sequential in nature. In the speech domain, the

next phoneme is strongly correlated with the previous phonemes [51]. Similarly, in the text domain, the correlations among subsequent words or sub-words are studied through language models [52]. In scientific discovery, sequence data is also prevalent. In protein interpretation [53], bacterial strain identification [54], and density of states prediction [55], sequence learning constantly plays a critical role, and can be applied to any reasonably ordered data, as illustrated in Figure 1.1.

While sequence data is commonly studied, it remains an arduous task. Compared with the standard machine learning tasks where the inputs have simple fixed patterns and abundant labels [56], sequence data takes various forms in different applications, presents more diverse learning tasks, and often requires more expensive effort to obtain labels. Input-wise, different domains adopt different data types. Speech data are often represented by waveforms [57]. Text data is comprised of sequential tokens [58]. Crystalline materials are often represented using graphs [59]. It is non-trivial to capture the structure of the input data and extract useful features, especially for graph data, and different sequences could have variable lengths [60], which is challenging. Task-wise, given the unique structure of sequence data, one can assign a tag for each word token [61], predict a global property for the whole sequence [62], or even generate another sequence from the input [63]. Label-wise, it is much more costly to collect the labels for sequences. In speech recognition, an audio clip can be transcribed to written texts, but it is tough to assign ground-truth phoneme to each frame [64]. In machine translation, linguistic experts and interpreters are hired to produce paired sentences of different languages [65, 66]. As a result, sequence data often has (1) more complicated input structures, which need special handling, and (2) much less of a supervision signal compared to the dataset size due to the

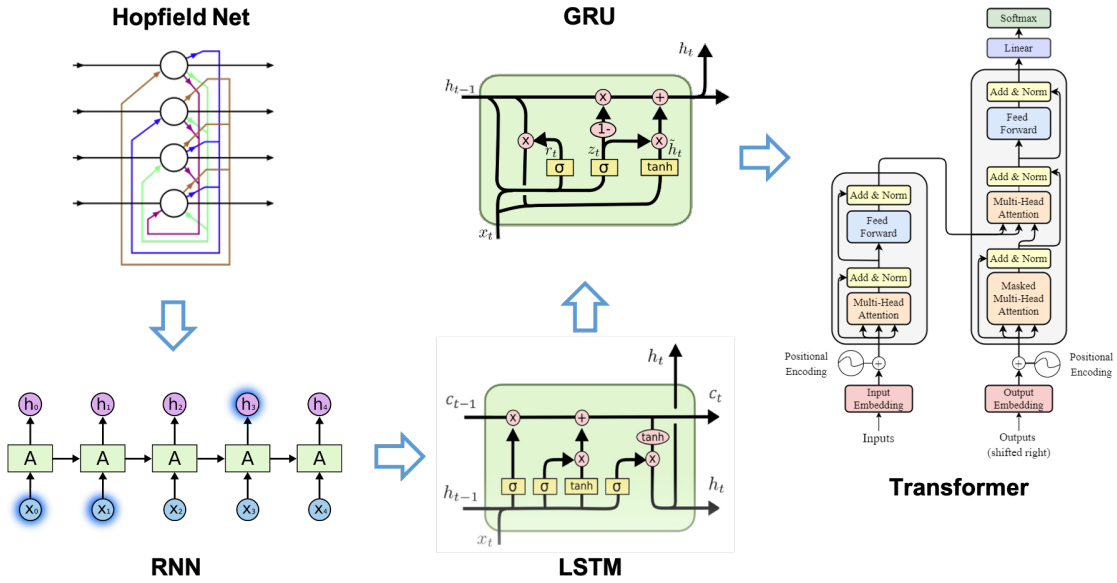


Figure 1.2: The evolution of mainstream sequence models, from the Hopfield net, RNN, LSTM and GRU to the modern transformer[1]. The common idea is to store information found earlier in the sequence.

high-quality requirements in label acquisition.

To handle the sequential structure of the input data, many novel deep learning models have been proposed. The pioneer is the recurrent neural network (RNN) [68]. The earliest RNN prototype is also known as the *Hopfield network* [69], where the core idea is to remember the previous states while updating. However, Hopfields nets only have binary neurons and thus limit capacity. Nevertheless, the loop structure introduced in this work inspired the modern RNN framework. An RNN keeps the output (or hidden state) from the previous step and reads the current input simultaneously to produce the output for the latest state. It is capable of capturing the short-term dependencies on the past states, but not long-term dependencies. It has been shown that gradients would vanish and distant past states get lost as the sequence length grows [70]. Long short-term memory (LSTM) networks are explicitly designed to model long-term dependencies [19].

[1] figures adapted from [67]

To store the information for long periods of time, an LSTM unit includes a separate state *cell state*, besides the standard input and previous hidden state. This cell state flows along the sequence and adds or removes some information at each step. To update this cell state, three neural gates learn how much to forget, how much to add and what to add, according to the input and previous hidden state. LSTM networks have greatly advanced the sequence learning field. Speech processing [71], language understanding [72], video captioning [73], weather forecasting [74] and many other domains have converged on using LSTM networks. A further improvement over LSTM is the gated recurrent unit (GRU) [75]. A GRU merges the cell state and hidden state, to further simplify the sequence model. GRU networks reduce matrix multiplications and sometimes perform better on smaller datasets [76]. But in general, GRU's performance is on par with LSTM.

While these RNN variants alleviate the long-term dependency issues, their memory of state history is still inadequate. For example, the last sentence might be associated with the first sentence in an essay. Such a connection would be almost impossible for LSTM or GRU to model. Furthermore, LSTM and GRU networks do not support parallel processing or computation. If the sequence is long, both the training and inference would take a very long time. Transformers [28] were then proposed to address these two concerns. Transformers do not read a sequence step by step, but as a whole instead. An attention mechanism characterizes the long-term token-to-token relationships [77]. Each token learns a latent representation and the inner products of two token representations denote the similarity. Position embeddings are further added to indicate the sequential order. Unlike RNNs, transformers keep the historical information in each state and visit all the past states when retrieving the memory. RNNs need to update

and maintain the memory bank and cell state, while transformers avoid this burden by simply storing all the past states [78]. The computations for similarity measurements, past state retrieval and aggregation can be represented as the matrix multiplications and are thus fast, parallelizable and scalable [79]. Just like LSTM networks had done before, transformers have dominated the sequence learning field in recent years, including vision, language, speech and multimodal applications [80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 53]. Surprisingly, transformers further facilitate the development of self-supervised learning. Notable self-supervised learning works such as BERT [91] and wav2vec 2.0 [37] are all built on top of transformers. Chapter 3 discusses the self-supervision in sequence learning. Though transformers outperform RNNs on most larger-scale datasets, RNNs are still suitable for many short-length or small-scale applications. The evolution of mainstream sequence models is shown in Figure 1.2.

The continuous refinement of sequence models has laid a solid foundation for extracting useful and compact latent features for sequence learning, allowing researchers to focus on loss design, task configuration and more challenging problems. As these sequence learning techniques reach the peak in standard supervised sequence learning tasks, researchers start to generalize them to a wider range of areas.

One direction is to extend supervised sequence learning to new fields where such techniques were deemed unfit in the past, like agriculture and scientific discovery. Normally, sequence data is comprised of a sequence of values or vectors. Simple RNNs or even multi-layer perceptrons (MLP) can handle them well [92]. But if the sequence data has another structure such as a graph, new frameworks are needed to interpret the new type of data and its structure (e.g.,

graph) [93]. In addition, labels can have other structures besides the plain values. The previously described sequence learning tasks assign a single label to each input, while in other cases, each input can be associated with multiple labels. The connections among these labels also constitute some structure for the label set [94]. Reconstructing this structure helps reveal the label connections and boosts the prediction accuracy. It is important to identify and utilize these structures in the sequence learning when we tackle complicated real-world problems.

Another increasingly popular direction is self-supervised sequence learning. Learning without supervision was believed to be fruitless, in part because (1) the training lacks guidance, (2) inputs are often noisy, and (3) models have limited capacity to distill meaningful representations from the weak self-supervision signals. On the other hand, modern sequence models have huge capacities [95] and the sequential structures can be exploited to devise self-supervised tasks through masking [96] or contrasting [97]. Unlike previous non-neural unsupervised learning methods that focused on the input space, neural methods may operate in the latent space where abstract, cleaner and less noisy features can be obtained [98]. These merits greatly alleviate the previous concerns and make self-supervised sequence learning possible. The primary success of self-supervision is the sequence representation learning. Without any supervision, sequence models can extract a more abstract feature for each input and these features are often more advantageous in the downstream supervised tasks. Additionally, Chapter 3 will show that self-supervision has applications beyond representation learning. Semantic feature disentanglement, latent clustering and training regularization can all benefit from self-supervised sequence learning.

1.1 Supervised Sequence Learning for Structured Data

As mentioned previously, the constant development of sequence models has greatly improved their expressiveness. Advanced sequence models like GRUs and transformers can already excel in most standard supervised learning tasks. In these tasks (e.g., speech and natural language processing (NLP)), sequence data is typically a series of values, vectors or signals. Similar simple-structured data might not always exist in other domains. In materials science, as an example, inorganic materials are arranged in crystals with highly ordered microscopic structure [99, 100]. It is hard to represent the crystals straightforwardly in terms of values, vectors, or signals. A more appropriate depiction is graph. Likewise, the continental US is a collection of 3,243 counties [101]. Vector-based representations are not suitable for encoding the geospatial correlations among counties, while graphs can more naturally represent the relationships between neighboring counties. Chapter 2 describes different ways to encode and comprehend graph-structured data.

Section 2.1 probes into the dynamic node property prediction for a fixed graph. In this problem, the graph is given and fixed; in other words, the adjacency matrix remains the same, while the node property of interest changes over time. Since different nodes are connected on the graph, their variations are also linked. An effective model must therefore incorporate the dynamics and properly embed the adjacency relations. One important application, from an agricultural setting, is crop yield prediction, to estimate the year-end crop production for each county in the US [102]. Multiple factors, especially climate change [103], can significantly impact the growth of crops, including temperature, precipitation, moisture, wind, etc. Initial methods decoded the impacts of these confounding influences [104]

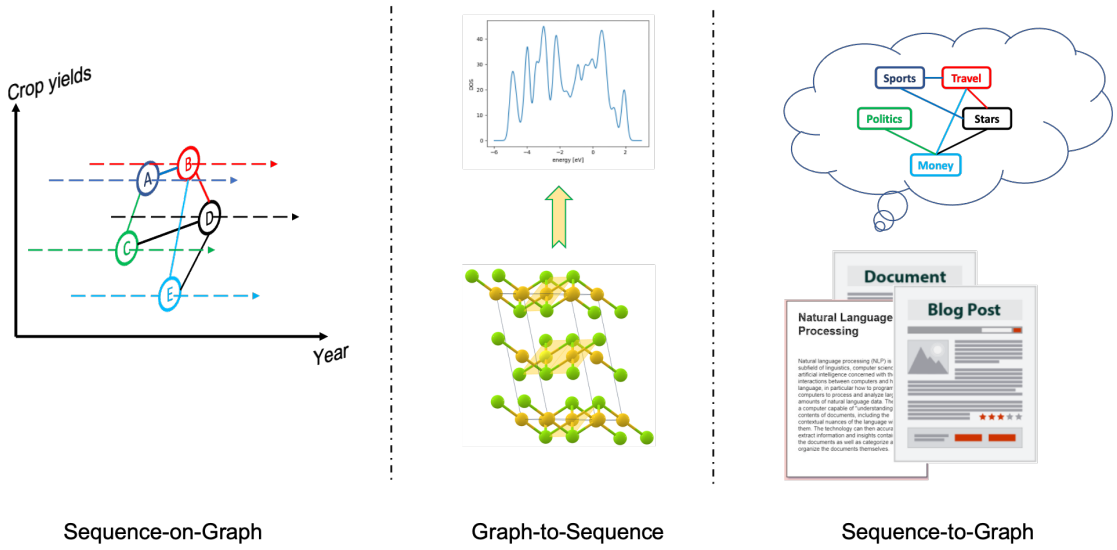


Figure 1.3: Three scenarios of supervised sequence learning for structured data: sequence-on-graph (crop yield prediction), graph-to-sequence (crystals' DoS prediction), sequence-to-graph (multi-label tag prediction).

on the yield outcomes, by deriving a mapping from the environmental predictors to the yield. However, historical yield trends and weather condition fluctuations are also valuable to consider [43]. Sequence models can store historical data to improve forecasts. Recent works adopted RNNs to apply these insights, by looking back several years for each county [105], yet they still miss the important geographical connections among counties. Nearby counties should have strongly correlated crop gains. If one county has bumper harvest, neighboring counties are likely to as well. This real-world problem can be formalized as a dynamic node property prediction task. The graph contains a node for each US county, with an edge connecting each pair of neighboring counties. This graph is fixed, but the crop yield and weather conditions of each county vary every year. I will introduce a new sequence learning method for this task, incorporating both the historical weather data and geospatial structures to estimate the crop yields [106]. LSTM- and GRU-based networks were used because the sequence length is short — at most five years. This method delivered state-of-the-art prediction accuracies

and extended the study to the whole continental US. The paper introducing it won the *Best ML Innovation Paper* award at NeurIPS Climate Change workshop and *AI for Earth* Microsoft Grant in 2021.

Section 2.2 studies a sequence prediction problem for scientific discovery from materials science, called *Density of States (DoS) prediction*. Crop yield prediction considers sequences of features and targets for each node in a graph, while in DoS prediction, the output is a sequence for the whole graph. DoS is an important spectral property of crystalline systems, that describes the proportion of states that are occupied at each energy level in the system. Crystalline systems are often represented with graphs analogous to their crystal structures, with nodes representing atoms, and edges representing interactions between atoms. Unlike scalar properties such as Fermi energy and band gap, DoS is a spectral property, which can be considered a type of sequence. The class of problems mapping graph input to sequential output is called graph-to-sequence learning [93], and is analogous to sequence-to-sequence learning [27], a prevailing learning framework. A sequence-to-sequence (seq2seq) model takes a sequence of items and outputs another sequence of items. Analogously, a graph-to-sequence model reads a graph as input, and outputs a sequence. I will introduce a new graph-to-sequence learning framework, *Xtal2DoS*, to predict the DoS from the graph-structured crystalline system. Xtal2DoS generalizes transformer-based seq2seq models [28] to handle graphs by replacing the transformer encoder with a graph attention network (GAT) [107]. Xtal2DoS greatly outperforms the Euclidean neural networks and multi-target probabilistic regression models on this task, and further simplifies the model architecture.

Section 2.3 presents the multi-label classification for sequence data such as

webpages and posts. Multi-label prediction is much tougher than single-label prediction in that label correlations are more critical [108]. Simply treating them as individual labels would degrade the performance [109]. Label correlations can be weak or strong, positive or negative. Therefore, unveiling the structure among labels contributes fundamentally to the final classification quality. For example, articles webpages are usually associated with multiple tags. *Sports* and *entertainment* topics are generally correlated, while *vacation* is typically uncorrelated with *politics* [110]. From a high-level perspective, this is a sequence-to-graph learning task, to estimate the label correlation graph from the input sequences. Unlike prior methods that explicitly construct neural nets to portray the label-to-label relationships [111, 112], the new method, C-GMVAE, learns label embeddings and encodes the similarity or difference with label-label inner products, inducing an implicit label correlation graph and boosts the classification performance for text sequences and citizen science bird observations from the eBird program [113].

In summary, Chapter 2 introduces three supervised sequence learning scenarios for structured data: sequence-on-graph, graph-to-sequence and sequence-to-graph; an overview is shown in Figure 1.3. Impactful real-world applications will be included as empirical studies to show the practicability of the new methods. The models for these scenarios are introduced through impactful real-world applications, with empirical studies to demonstrate their practicability.

It is also worth noting that most of these works are inspired by the computational sustainability (CompSust) initiative [114]. CompSust emphasizes applying AI techniques to practical problems concerned with the balancing of environmental, economic, and societal needs for a sustainable future. Real applications

pose real challenges. Applications such as crop yield and DoS prediction involve patterns and structures distinct from tasks that machine learning has traditionally addressed, and thus CompSust applications have served to motivate the design of cutting-edge and practical models.

1.2 Self-supervised Sequence Representation Learning

The recent successes in deep learning have been driven largely by the combination of deep models and large manually-annotated datasets [1, 56]. While deeper models can be deployed due to the continuously increasing GPU capacities [115], it is harder to acquire enough manual effort to meet the annotation demand. In computer vision, the amount of labeled images has grown considerably, from CIFAR-100 (100 classes containing 600 images each) [116], ImageNet (1,000 categories and 2M images in total) [56], and COCO (328K images from 91 object types) [117], to Places365 (10M images from 434 scene categories) [118].Similarly, in the speech domain, as an example of sequential data, the total hours per dataset have increased from 5.4 hours (TIMIT [64]), 80 hours (WSJ [119]), and 960 hours (LibriSpeech [120]), to 45K hours (MLS [121]). Though effort can be made to further enlarge the dataset size, some limitations have been emerging. First, even for the largest datasets, the sizes are still small compared to the real-world scope. Actually images can easily be grouped into far more than 1,000 categories, and 45K hours of recorded speech is minuscule considering the amount of variability that exists in across the hundreds of countries and billions of people in the world. Second, for many established fields, experts are willing to spend enough time collecting labeled data. But as other fields begin to apply sequence learning, or the same fields apply it to new settings, it is difficult to collect labeled data for

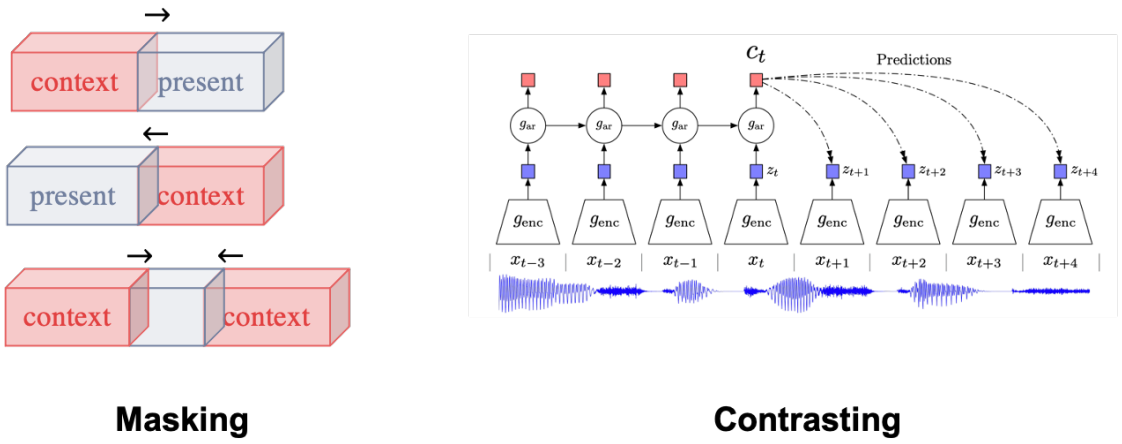


Figure 1.4: Left: Different masking strategies, including masking the future, the past, or the present [122]. Right: The core idea of contrastive learning is to pull together similar samples and push away dissimilar ones [123].

supervised learning. Third, supervised learning often results in models that are highly specialized for a given task, and the learnt representations can rarely be transferred directly and conveniently to new tasks. These issues have motivated the development of self-supervised learning.

Self-supervised representation learning is becoming more and more prominent in the sequence domain [124]. Self-supervised and unsupervised techniques were originally designed for clustering and dimensionality reduction purposes [125, 126]. Contemporary self-supervised methods are primarily intended to learn widely applicable representations for inputs so that downstream supervised tasks can use these representations as inputs instead or further finetune them. BERT is a well-known example [91]. Exploring effective self-supervision from the sequence data and exploiting the powerful sequence models are the keys to learning high-quality representations. The earlier methods explored structure within the input space, but these were often constrained by low encoder capacity [127]. Later approaches used deep sequence models to increase capacity, but could not guide the models towards a meaningful latent space

structures. To date, there have been two mainstream self-supervised sequence learning directions, *masking* and *contrasting*. In masking, some portions of a sequence are masked out with special tokens (or simply zero), and the sequence models are expected to reconstruct these missing parts [128]. The learnt encodings are called *contextual embeddings* because they are aware of the left and right contexts [29, 91, 129]. Figure 1.4 shows several masking schemes. Contrastive learning adopts a slightly different perspective: given an anchor sample, its embedding should be close to similar samples (positive) and far from dissimilar samples (negative) in some learnt embedding space [123]. For sequence data, the future states from the same sequence can be viewed as positive samples, and the states from other sequences may be negative ones. An illustration is presented in Figure 1.4. Contrastive learning is akin to mutual information estimation [123, 130], and essentially extends the idea of triplet loss [131], ranking loss [132], and noise-contrastive estimation [133].

Masking and contrasting generally develop in parallel. Section 3.1 describes an innovative technique to embed the mutual information estimation from contrastive learning into masking, which regularizes the training process and imposes structures onto the latent space. The introduced method, deep autoencoding predictive components (DAPC), uses predictive information [134] as the metric to estimate latent mutual information exactly, rather than contrastive estimation, which is a lower bound of the exact mutual information. The novel self-supervised method (1) learns versatile representations for various downstream tasks with applications to ASR, weather forecasting, physical system recovery, etc., (2) saves sample complexity and manual annotation, and (3) outperforms other self-supervised pretraining models.

While most self-supervised learning methods are designed for pretraining and representation abstraction, self-supervision can do much more, such as disentanglement. Disentangled representation learning is a central topic in interpretable machine learning. Many renowned deep models have been criticized because they lack semantic meaning in the latent space [135] and can only be used as a black box [136]. Hence understanding what the deep models are learning builds confidence in using these models. One strategy is to analyze the model and latent codes as a post-process after a model is trained [137]. Another route is to force the model to gain semantic insights in the latent space during training [138]. Section 3.2 describes a fundamental subject in sequence learning, *disentangling* the time-variant and time-invariant factors. In a self-introduction video, the person's facial expression changes, while the identity remains the same. In an inaugural address, the speaker is invariant while the linguistic contents vary with time. The contrastively disentangled sequential variational autoencoder (C-DSVAE) is a model that uses probabilistic reasoning and contrastive learning without requiring any supervision. It learns dynamic factors for each step and a static factor for the whole sequence. It has been successfully applied to video and audio generation tasks, achieving state-of-the-art results.

In addition to pretraining and disentanglement, self-supervision can be integrated with supervision synergistically. Most works have treated self-supervision and supervision separately [139]. Self-supervised learning trains an encoder to extract general representations in the first stage, and supervised learning finetunes the encoder for specific tasks in the second stage. However, such two-stage scheme also has some shortcomings. For instance, the encoder might catastrophically forget the previously learnt knowledge in the first stage [140]. It is also nontrivial to select the best pre-trained checkpoint, as pre-training for longer is

not necessarily better. For this reason, we describe a method to train the sequence model with both supervised and self-supervised losses jointly, to reconcile the gradients provided by both types of losses and to mutually regularize each other. The resulting model not only reduces the training time, but also outperforms the latest models on a difficult multilingual ASR task.

In summary, Chapter 3 (1) describes a brand-new self-supervised learning method based on masking and mutual information estimation to enforce latent structure, (2) proves the effectiveness of self-supervision in disentangled representation learning, and (3) validates the performance when the supervised and self-supervised signals are fused.

List of Publications

Most of the findings, claims and results in this thesis stem from these publications:

1. **Junwen Bai**, Shufeng Kong, Carla Gomes. Gaussian Mixture Variational Autoencoder with Contrastive Learning for Multi-Label Classification. International Conference on Machine Learning (ICML), 2022.
2. **Junwen Bai**, Bo Li, Yu Zhang, Ankur Bapna, Nikhil Siddhartha, Khe Chai Sim, Tara N. Sainath. Joint Unsupervised and Supervised Training for Multilingual ASR. International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2022.
3. Joshua Fan*, **Junwen Bai***, Zhiyun Li*, Ariel Ortiz-Bobea, Carla Gomes. A GNN-RNN Approach for Harnessing Geospatial and Temporal Informa-

tion: Application to Crop Yield Prediction. AAAI Conference on Artificial Intelligence (AAAI), 2022.

4. Bo Li, Ruoming Pang, Tara N. Sainath, Anmol Gulati, Yu Zhang, James Qin, Parisa Haghani, W. Ronny Huang, Min Ma, **Junwen Bai**. Scaling End-to-End Models for Large-Scale Multilingual ASR. IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), 2021.
5. **Junwen Bai**, Weiran Wang, Carla Gomes. Contrastively Disentangled Sequential Variational Autoencoder. Advances In Neural Information Processing Systems (NeurIPS), 2021.
6. **Junwen Bai**, Weiran Wang, Yingbo Zhou, Caiming Xiong. Representation Learning for Sequence Data with Deep Autoencoding Predictive Components. International Conference on Learning Representations (ICLR), 2021.
7. Wenting Zhao, Shufeng Kong, **Junwen Bai**, Daniel Fink, Carla Gomes. HOT-VAE: Learning High-Order Label Correlation for Multi-Label Classification via Attention-Based Variational Autoencoders. AAAI Conference on Artificial Intelligence (AAAI), 2021.
8. **Junwen Bai**, Shufeng Kong, Carla Gomes. Disentangled Variational Autoencoder based Multi-Label Classification with Covariance-Aware Multivariate Probit Model. International Joint Conference on Artificial Intelligence (IJCAI), 2020.
9. Shufeng Kong, **Junwen Bai**, Jae Hee Lee, Di Chen, Andrew Allyn, Michell Stuart, Malin Pinsky, Kathy Mills, Carla Gomes. Deep Hurdle Networks for Zero-Inflated Multi-Target Regression: Application to Multiple Species Abundance Estimation. International Joint Conference on Artificial Intelligence (IJCAI), 2020.

10. Guandao Yang, Tianyi Zhang, Polina Kirichenko, **Junwen Bai**, Andrew Wilson, Chris De Sa. SWALP: Stochastic Weight Averaging in Low-Precision Training. International Conference on Machine Learning (ICML), 2019.

11. **Junwen Bai**, Zihang Lai, Runzhe Yang, Yexiang Xue, John Gregoire, Carla P. Gomes. Imitation Refinement For X-Ray Diffraction Signal Processing. International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2019.

12. Carla P. Gomes, **Junwen Bai**, Yexiang Xue, Johan Björck, Brendan Rappazzo, Sebastian Ament, Richard Bernstein, Shufeng Kong, Santosh K Suram, R Bruce van Dover, John M Gregoire. CRYSTAL: a multi-agent AI system for automated mapping of materials' crystal structures. In MRS Communications 9 (2) 600-608, 2019.

13. **Junwen Bai**, Yexiang Xue, Johan Bjorck, Ronan Le Bras, Brendan Rappazzo, Richard Bernstein, Santosh K. Suram, R. Bruce van Dover, John M. Gregoire, Carla P. Gomes. Phase Mapper: Accelerating Materials Discovery with AI. In AI Magazine 39 (1), 15-26, 2018.

14. **Junwen Bai**, Sebastian Ament, Guillaume Perez, John M. Gregoire, Carla P. Gomes. An Efficient Relaxed Projection Method for Constrained Nonnegative Matrix Factorization with Application to the Phase-Mapping Problem in Materials Science. International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR), 2018.

15. **Junwen Bai**, Johan Bjorck, Yexiang Xue, Santosh K. Suram, John M. Gregoire, Carla P. Gomes. Relaxation Methods for Constrained Matrix Factorization Problems: Solving the Phase Mapping Problem in Materials

Discovery. International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR), 2017.

16. Yexiang Xue, **Junwen Bai**, Ronan Le Bras, Brendan Rappazo, Richard Bernstein, Johan Bjorck, Liane Longpre, Santosh K. Suram, Robert B. van Dover, John Gregoire, Carla P. Gomes. Phase-Mapper: An AI Platform to Accelerate High Throughput Materials Discovery. AAAI Conference on Artificial Intelligence (AAAI), 2017.

17. Santosh K. Suram, Yexiang Xue, **Junwen Bai**, Ronan Le Bras, Brendan Rappazzo, Richard Bernstein, Johan Bjorck, Lan Zhou, Robert B.van Dover, Carla P. Gomes, John M. Gregoire. Automated Phase Mapping with AgileFD and its Application to Light Absorber Discovery in the V-Mn-Nb Oxide System. In American Chemical Society Combinatorial Science 19(1), 37-46, 2017.

CHAPTER 2

EXPLORING AND EXPLOITING STRUCTURES IN SEQUENCE LEARNING

In this chapter, I will describe three scenarios in the supervised sequence learning for structured data: sequence-on-graph, graph-to-sequence, and sequence-to-graph, with a concrete real-world running example for each topic. I will illustrate how to discover, frame and employ the structures hidden inside the data, for different prediction tasks. Some of the research described in this chapter has been reported in the following publications: Fan et al. (2022) [106] and Bai et al. (2022) [141].

2.1 Sequence-on-Graph: Harnessing Geospatial and Temporal Information for Crop Yield Prediction

2.1.1 Introduction

Climate change [142] has become a real and pressing challenge that poses many threats to our everyday life. Besides the evident extreme events [143], climatic variations also gradually impact the yields of major crops [144]. Crop production is vulnerable and sensitive to fluctuations in climatic factors such as temperature, precipitation, soil moisture and many other factors [145]. As the planet gets warmer, all of these swiftly-changing factors could perturb annual crop yields; climate change has already reduced agricultural productivity growth by 21% [43]. Many recent works urge rethinking crop production practices under climate change [146, 147], which motivates the crop yield prediction problem [148]. Crop

yield prediction can help with food security [149], supply stability [150], seed breeding [151], and economic planning [152].

However, crop yield depends on numerous complex factors including weather, land, water, etc. While there exist specialized process-based models to simulate crop growth [153], they often produce highly-biased predictions, require strong assumptions about management practices, and are computationally expensive [154]. Therefore, in recent years, powerful yet inexpensive machine learning methods have been widely adopted in crop yield prediction and demonstrated impressive results [155, 156, 157, 158, 105, 159]. Machine learning models (especially deep learning models) benefit from the large capacity, sophisticated non-linearity and mature techniques inherited from other application domains.

Despite the enormous amount of machine learning papers for crop yield prediction, many of them share similar methods. Among around 70 papers we surveyed, 48 used neural networks, 10 used tree-based methods (e.g. decision tree, random forest), and 10 used linear regressions (e.g. lasso). These methods often only differ in location (US, Brazil, India), study granularity (province, county, site/farm), crop types (soybean, corn), and time range (weeks to years). Similar findings are also reported in [148]. For many relatively small self-collected datasets, simpler models are preferred. But these models may not perform well on a large and diverse region like the entire US.

In this work, we compare various machine learning techniques on a nation-wide scale, and introduce a novel graph-based framework, GNN-RNN, which integrates both geospatial and temporal knowledge into inference. We train and compare these methods on over 2,000 counties from 41 states in the US mainland, with data covering years 1981 to 2019. There are up to 49 climatic and soil factors

for each county, including precipitation, temperature, wind, soil moisture, soil quality, etc. Most of these factors vary across time within the year, or vary across different soil layers. All features are publicly available from sources such as PRISM, NLDAS, and gSSURGO. Furthermore, although not every county has crops planted, USDA provides corn yield labels from 41 states, and soybean yields from 31 states.

Recent works using machine learning demonstrate promising results for crop yield prediction [105]. Nevertheless, these methods treat each county as an i.i.d. sample in their models, which is plausible in small regions but may not fully utilize the spatial structure of a larger region. For instance, if one county has a splendid harvest, the neighborhood counties are very likely to have high yields as well, which violates the independence assumption. It is also problematic to treat counties in the northern US and southern US as i.i.d. samples, as the distribution of their climatic and soil conditions is very different. We hence introduce **graph neural networks** (GNN) [160] to take into account the geographical relationships among counties. When the model makes a prediction for a county, it can combine the features from neighboring counties with its own features to boost the predictive power. GNN models have been successful in many tasks such as election prediction [161] and COVID forecasting [162]. Additionally, we show that GNNs can work synergistically with RNNs to combine both geospatial and temporal information for prediction. We will show the novel **GNN-RNN** model can achieve superior performance in experiments. As far as we know, our work is the first to incorporate geographical knowledge into crop yield prediction. To further lay a solid foundation in this task, we compare various widely adopted machine learning methods with our method, including lasso [163], gradient boosting tree [164], CNN, RNN, CNN-RNN [105]. These are

also predominant methods among the papers we surveyed. The experimental results show that GNN-based methods consistently outperform these existing models on the nationwide benchmark. On both RMSE and R^2, our GNN-RNN outperforms the state-of-the-art CNN-RNN model by 10%.

2.1.2 GNN-RNN

Problem Formulation

In crop yield prediction, we denote each county's climatic features by $\mathbf{x}_{c,t}$ and ground-truth crop yield (for a particular crop) by $y_{c,t} \in \mathbb{R}$, where c, t represent county and year respectively. Each $\mathbf{x}_{c,t}$ contains four types of features (detailed descriptions of these features can be found in the Experiments section): weather features $\mathbf{x}^w_{c,t} \in \mathbb{R}^{n_w \times 52}$, land surface features $\mathbf{x}^l_{c,t} \in \mathbb{R}^{n_l \times 52}$, soil quality features $\mathbf{x}^s_c \in \mathbb{R}^{n_s \times 6}$, and some extra features (e.g. crop production index) $\mathbf{x}^e_c \in \mathbb{R}^{n_e}$. Namely, $\mathbf{x}_{c,t} = (\mathbf{x}^w_{c,t}, \mathbf{x}^l_{c,t}, \mathbf{x}^s_c, \mathbf{x}^e_c)$. We denote the number of weather, land surface, soil quality, and extra variables as n_w, n_l, n_s, n_e respectively. Among these features, $\mathbf{x}^w_{c,t}, \mathbf{x}^l_{c,t}$ change both spatially and temporally, while $\mathbf{x}^s_c, \mathbf{x}^e_c$ are county-specific and remain stable over time. The goal is to predict $y_{c,t}$ given $\mathbf{x}_{c,t}$. Recent work [105] also showed features from past years can help with the prediction, so we reformulate our task as predicting $y_{c,t}$ with $\{\mathbf{x}_{c,t}, \mathbf{x}_{c,t-1}, ..., \mathbf{x}_{c,t-\Delta t}\}$. Δt is the length of year dependency. If $\Delta t = 0$, the model will not consider features from prior years.

Per-Year Embedding Extraction

Regardless of whether the models use historical features or not, the first step is always to extract an embedding for each year from $\mathbf{x}_{c,t}$. Then a prediction can be made based on the embedding from the current year or the embeddings from the last few years.

The four types of features $\mathbf{x}^w_{c,t}, \mathbf{x}^l_{c,t}, \mathbf{x}^s_c, \mathbf{x}^e_c$ have different structures. Using a uniform neural network to extract the embedding may not effectively exploit the structure in the raw data. For example, weekly features $\mathbf{x}^w_{c,t}, \mathbf{x}^l_{c,t}$ naturally incorporate a temporal order, but county-specific soil features $\mathbf{x}^s_c$ do not change temporally and are measured at different depths underground. Therefore, we use separate neural networks to process the differently structured-parts from $\mathbf{x}_{c,t}$:

$$\begin{aligned} \mathbf{h}^{wl}_{c,t} &= f_{wl}(\mathbf{x}^w_{c,t}, \mathbf{x}^l_{c,t}) \\ \mathbf{h}^s_c &= f_s(\mathbf{x}^s_c) \\ \mathbf{h}_{c,t} &= (\mathbf{h}^{wl}_{c,t}, \mathbf{h}^s_c, \mathbf{x}^e_c) \end{aligned} \tag{2.1}$$

$f_{wl}(\cdot)$ handles the features that vary over time. Since land surface features like soil moisture from $\mathbf{x}^l_{c,t}$ are weekly data closely related to weather, we concatenate $\mathbf{x}^l_{c,t}$ and $\mathbf{x}^w_{c,t}$ before further passing to f_{wl}. Given the temporal order, an RNN or a CNN can be used for f_{wl} to facilitate information aggregation along the time axis. On the other hand, $f_s(\cdot)$ aggregates information along soil depths. We use CNN as the architecture for f_s. $\mathbf{x}^e_c$ only contains six scalar values, so we directly pass it to the output embedding. The final embedding $\mathbf{h}_{c,t}$ is the concatenation of $\mathbf{h}^{wl}_{c,t}, \mathbf{h}^s_c, \mathbf{x}^e_c$.

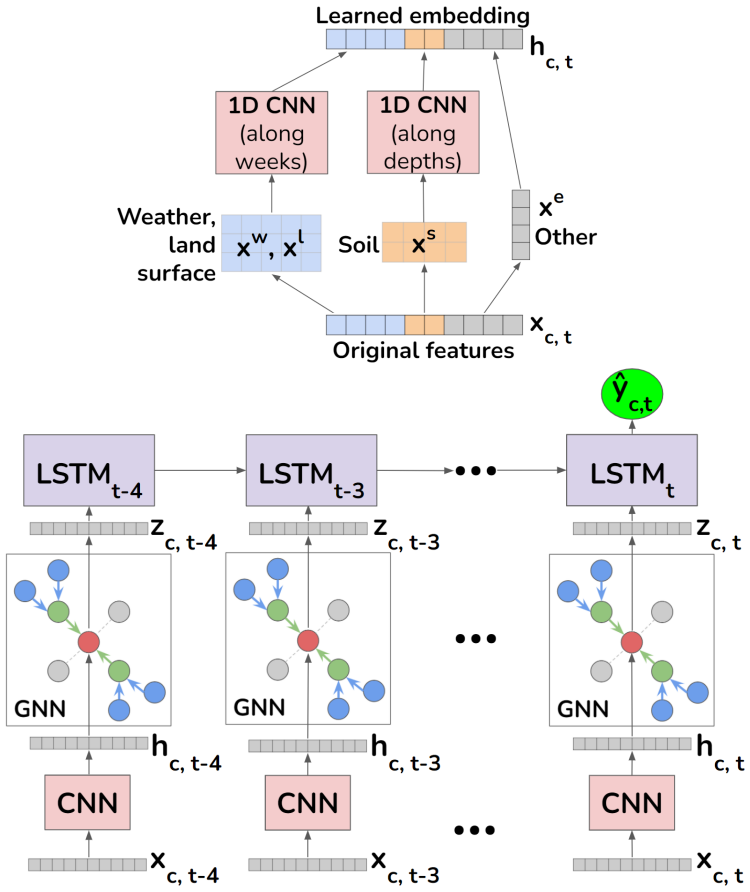


Figure 2.1: Left: The CNN model used for per-year embedding extraction. Right: Our overall GNN-RNN framework. For each county c and year t', the CNN extracts an embedding $\mathbf{h}_{c,t'}$. Then we apply a GNN to refine each year's embedding by aggregating information from neighboring counties, producing a new embedding $\mathbf{z}_{c,t'}$. Finally, an LSTM processes the embeddings from each year and outputs the yield prediction $\widehat{y}_{c,t}$.

Temporal Dependency

Though new crops are planted every year and yields primarily depend on climatic factors within one year, it has been observed that the trend and variations captured by recent history can be very informative for prediction [105]. For example, crop yields have tended to increase over the past few decades due to improvements in technology and genetics [104]. While data on the underlying technological improvement is unavailable [105], we can observe recent trends in crop yield. Our per-year embedding extraction makes it easy to incorporate

historical knowledge. All we need is an RNN that reads the per-year embeddings from the current year and several prior years. The output from the last time step would be our prediction for the crop yield of the current year:

$$\widehat{y}_{c,t} = r(\mathbf{h}_{c,t-\Delta t}, ..., \mathbf{h}_{c,t-1}, \mathbf{h}_{c,t}) \tag{2.2}$$

where $r(\cdot)$ is an RNN, and $\mathbf{h}_{c,t'}$ is the embedding from year t' for county c. The model described so far follows the CNN-RNN framework, which has previously been shown to outperform single-year NN models [105].

Incorporating Geographical Knowledge

Eq. 2.2 shows how one can extend the use of embeddings from Eq. 2.1 temporally. Then a natural question is, Can we take advantage of the embeddings geospatially as well? Intuitively, if a county has good yields, nearby counties tend to have good yields as well. The weather and soil conditions should also transition smoothly across the continent. The additional features from neighboring counties could boost the prediction if used properly. A recent success in COVID-19 forecasting [162] with similar insights could further support incorporating geographical knowledge, where the graph-based representation learning greatly improves case prediction.

Graph Neural Network GNN [165] is a novel type of neural network proposed to unravel the complicated dependencies inherent in graph-structured data sources. Given its strong power in representation learning, GNN has demonstrated prominent applications in chemistry [39], traffic [166], biology [167], and computer vision [168] with sophisticated model architectures [169, 170, 107]. Formally, a graph is denoted by $G = (V, E)$ where V is the set of nodes and E is

the set of edges between nodes. In our crop yield prediction task, each node is a county. E is represented as a symmetric adjacency matrix $A \in \{0,1\}^{N \times N}$ where $A_{i,j} = 1$ if two counties $v_i, v_j \in V$ border and $A_{i,j} = 0$ otherwise. N is the total number of counties. Each node is associated with $\mathbf{x}_{c,t}$ for every year.

GraphSAGE A popular GNN model, GraphSAGE, [170] is a general framework that leverages node feature information and learns node embeddings through aggregation from a node's local neighborhood. Unlike many other methods based on matrix factorization and normalization [161], GraphSAGE simply aggregates the features from a local neighborhood, and is thus less computationally expensive. The features can be aggregated from a different number of hops or search depth. Therefore the model often generalizes better. GraphSAGE is suitable for crop yield prediction because most counties only border a few others and the adjacency matrix is sparse. It also provides flexible aggregation methods.

Formally, for the l-th layer of GraphSAGE,

$$\begin{aligned} \mathbf{a}_{c,t}^{(l)} &= g_l(\{\mathbf{z}_{c',t}^{(l-1)}, \forall c' \in \mathcal{N}(c)\}) \\ \mathbf{z}_{c,t}^{(l)} &= \sigma(\mathbf{W}^{(l)} \cdot (\mathbf{z}_{c,t}^{(l-1)}, \mathbf{a}_{c,t}^{(l)})) \end{aligned} \tag{2.3}$$

where $\mathbf{z}_{c,t}^{(0)} = h_{c,t}$ from Eq. 2.1, and $l \in \{0, 1, ..., L\}$. $\mathcal{N}(c) = \{c', \forall A_{c,c'} = 1\}$ is the set of neighboring counties for c. The aggregation function for the l-th layer is denoted $g_l(\cdot)$, which could be mean, pooling, or graph convolution (GCN) function. In practice, we found mean or pooling are effective and computationally efficient. $\mathbf{a}_{c,t}^{(l)}$ is the aggregated embedding from the bordering counties. We concatenate $\mathbf{a}_{c,t}^{(l)}$ with the last layer's embedding $\mathbf{z}_{c,t}^{(l-1)}$ before the transformation using $\mathbf{W}^{(l)}$. $\sigma(\cdot)$ is a non-linear function.

GNN-RNN The output embedding from GNN's last layer $\mathbf{z}_{c,t}^{(L)}$ thus extracts the information (e.g., weather, soil) from the whole local neighborhood for year t. To

integrate the historical knowledge, we can do the same as in Eq. 2.2, by taking the GNN output embeddings from prior years:

$$\widehat{y}_{c,t} = r(\mathbf{z}^{(L)}_{c,t-\Delta t}, ..., \mathbf{z}^{(L)}_{c,t-1}, \mathbf{z}^{(L)}_{c,t}) \tag{2.4}$$

where $\mathbf{z}^{(L)}_{c,t'}$ is the GNN embedding from year t'.

Loss Function We use log-cosh function as our objective:

$$L(\widehat{y}_{c,t}, y_{c,t}) = \log(\cosh(\widehat{y}_{c,t} - y_{c,t})) \tag{2.5}$$

Log-cosh works similarly to mean square error, but is not as strongly affected by the occasional wildly incorrect prediction. It is also twice differentiable everywhere. Mini-batch training is adopted during optimization. Batch loss is the average log-cosh loss of all samples in a batch.

2.1.3 Prior Works

As one of the early works, [171] attempted a shallow neural network on corn yield prediction. It was shown that neural networks could beat conventional regression algorithms [172]. In recent years, owing to the development of deep learning, neural network-based models have become more prevalent in the crop yield prediction field [156, 173]. As we mentioned in the introduction, 48 out of 70 recent works we surveyed employed neural nets. More than half of the NN-based works adopted CNN and one fourth of them employed RNN.

There are two groups of research directions according to different input sources. The first group of methods read remote sensing data such as satellite images or normalized difference vegetation index (NDVI), and used that to

estimate the yields. [158] applied a deep Gaussian process to predict crop yields from a series of the multi-spectral satellite images. [174] employed deep CNNs to reduce the prediction uncertainty on RGB images. [175] did a case study in the Midwest and compared various AI models on satellite product datasets. 20 out of all the 70 papers primarily or only dealt with remote sensing data. These methods illuminate the use of the widely available remote sensing data, but it is hard to directly model the relations between crop yields and environmental factors that actually affect the yields. Therefore, another line of research aims at collecting environmental factors and directly training models with these factors as inputs. [176] used temperature, rainfall and other meterological parameters to predict wheat production. [177] collected crop genotypes and environments to predict the performance of corn hybrids. More recently, CNN-RNN [105] incorporated historical environment knowledge and proved benefits. Our GNN-RNN takes one more step by further adding neighborhood information.

Dataset-wise, most papers have their own small-scale datasets, which vary largely in different dimensions. Scale-wise, [178] only studied a single zone in Mexico, while [179] studied the whole country of Argentina. Time-wise, [179] spanned over 5 years, while [158] investigated 13 years. It is thus hard to fairly compare models or validate the performance. There have been several efforts to evaluate models on a large and consistent dataset. The closest one to ours is the one from the CNN-RNN paper [105]. However, they only used 13 states from the Corn Belt and used fewer features than us (for example, they did not use land surface data such as soil moisture). By contrast, we evaluate our models at a nationwide scale, using data from 41 states and 39 years. This forces our models to generalize to a diverse range of locations that have very different climatic and geographic conditions, instead of overfitting to a single region.

2.1.4 Experiments on Crop Yield Prediction

We compare 11 representative machine learning models, including GNN and GNN-RNN, on US county-level crop yields for corn and soybean. We evaluate performance on three metrics: RMSE, R^2, and correlation coefficient. Given a test year t, we use year $t-1$ for validation and all the prior years for training. For example, if the test year is 2019, we train on data from years 1981-2017 (inclusive), validate on 2018 crop yields, and test on 2019 crop yields.

Dataset Details

Crop yield labels for corn and soybean are available from the USDA Crop Production Reports [180] for numerous counties in the US. Not all counties report data in every year, but the coverage is still quite comprehensive. For example, for corn, all years between 1981 and 2003 have over 2,000 counties across 41 states reporting data. We train and evaluate our model on all counties where yield data is available. (When computing the loss, we ignore counties that do not have yield labels for that year.)

We use a variety of climate, land surface, and soil quality variables as input features; these features are available for almost all counties in the contiguous 48 US states (3,107 counties in total[1]). We draw 7 weather features from the PRISM climate mapping system [181]: precipitation, min/mean/max temperature, min/max vapor pressure deficit, and mean dewpoint temperature. These features are available at a 4×4 km grid for each day.

[1]The only exception is Nantucket County, Massachusetts, where land surface model data is missing, since it is an offshore island. Also note that some counties have feature data but not label (yield) data. Only the GNN and GNN-RNN models can make use of these unlabeled county features.

We acquire 16 land surface features from the North American Land Data Assimilation System (NLDAS) [182], which is a large-scale land surface model that closely simulates land surface parameters. These features include soil moisture content, moisture availability, and soil temperature (all at various soil depths), as well as observed weather variables such as wind speed and humidity. These variables are available at a 0.125×0.125 degree ($\sim$ 14 km) spatial resolution, every hour.

Soil quality features were acquired from the Gridded Soil Survey Geographic Database (gSSURGO) [183], at a 30×30 meter resolution. These features include available water capacity, bulk density, and electrical conductivity, pH, and organic matter. Unlike the weather and land surface features, the gSSURGO soil quality features are fixed and *do not change over time*. In addition to the raw features, we use the raw sand, silt, and clay percentages to compute the "soil texture type" of each pixel based on the Natural Resources Conservation Service Soil Survey's classification scheme [184], and then compute the fraction of each county occupied by each soil texture type. In total, we have a total of 20 gSSURGO variables that are depth-dependent (so there are values for 6 different soil depth levels), and 6 extra variables which are not depth-dependent (such as crop productivity indices). Finally, as in [105], we use the average crop yield (over all counties) of the previous year as an additional input feature, to capture the increasing trend in crop yield over time. A full list of the features can be found in the following.

All of these datasets were originally available as gridded raster data at a variety of spatial resolutions. We aggregated each feature to the county level by computing the weighted average of the variable over all grid cells that overlap

with the county. Each grid cell is weighted by the percentage of the cell that lies inside the county, multiplied by the percentage of that grid cell which is cropland, pasture, or grassland; the land cover percentages are computed using the National Land Cover Database [185]. In addition, the time-dependent variables (weather and land surface) were aggregated from daily to weekly frequency to make the prediction task more tractable.

List of Features

We provide a list of all of the input features used in the model, grouped by source. Note that all features are spatially aggregated to the county level, using a weighted average (where each grid cell is weighted by the fraction of the cell that lies inside the county, multiplied by the percentage of the grid cell that is cropland/pasture/grassland). An example of this aggregation process is depicted in Figure 2.2. Temporally, all time-dependent features are also aggregated to weekly frequency.

Weather features ($\mathbf{x}_{c,t}^{w}$) come from the PRISM dataset [181], with an original spatial resolution of 4 km and a temporal resolution of daily:

1. Precipitation
2. Mean dewpoint temperature
3. Daily max temperature
4. Daily mean temperature
5. Daily minimum temperature
6. Max vapor pressure deficit

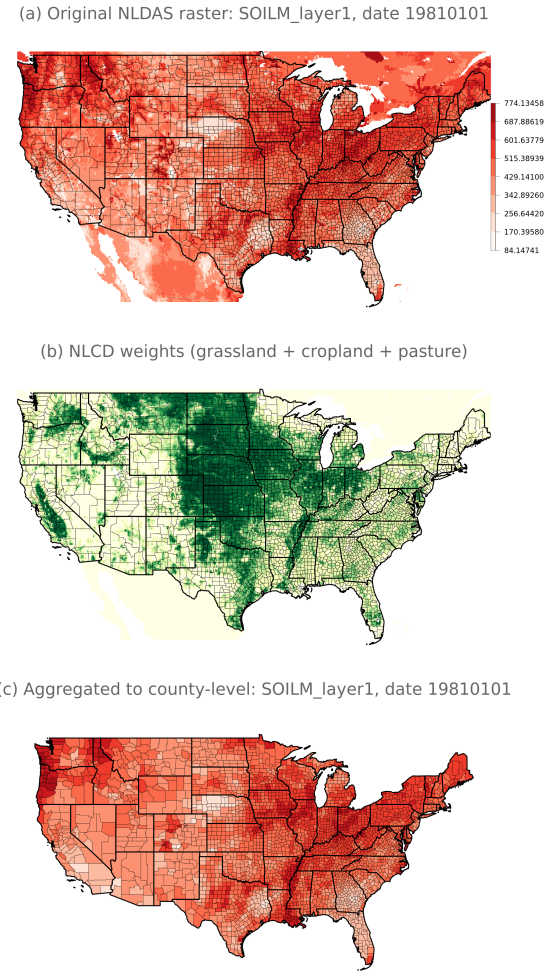


Figure 2.2: Example of aggregating features to county level.
(a) raw raster of soil moisture from NLDAS.
(b) Percentage cropland/grassland/pasture (used to compute grid cell weights).
(c) the county-level values we generated.

7. Min vapor pressure deficit

Land surface features ($\mathbf{x}^l_{c,t}$) come from the NLDAS land surface model [182], with an original spatial resolution of 0.125 degrees (14 km) and a temporal resolution of hourly:

1. Precipitation hourly total (kg/m^2)
2. Moisture availability (%), 0-200 cm

3. Moisture availability (%), 0-100 cm
4. Soil moisture content (kg/m^2), 0-200cm
5. Soil moisture content (kg/m^2), 0-100cm
6. Soil moisture content (kg/m^2), 0-10cm
7. Soil moisture content (kg/m^2), 10-40cm
8. Soil moisture content (kg/m^2), 40-100cm
9. Soil moisture content (kg/m^2), 100-200cm
10. 2-m above ground specific humidity (kg/kg)
11. 2-m above ground temperature (K)
12. Soil temperature (K), 0-10 cm
13. Soil temperature (K), 10-40 cm
14. Soil temperature (K), 40-100 cm
15. Soil temperature (K), 100-200 cm
16. Wind speed (m/s), hourly max

Note that the cm ranges represent depths in the soil.

Soil quality features ($\mathbf{x}_c^s$) come from the Gridded Soil Survey Geographic Database (gSSURGO) [183]. The dataset has a 30-m spatial resolution for the continental U.S. These variables do not change over time. However, they vary with depths, which are measured at 6 soil depth layers (0-5cm, 5-15cm, 15-30cm, 30-60cm, 60-100cm, 100-200cm). Because soil quality at a given point can vary substantially within a county, accounting for the location of agricultural activity can be critical when constructing appropriate county-level soil variables. Thus, the weighted-average technique is especially important here. We aggregate the

fine-scale soil data to the county level based on the percentage of each NLCD Land Cover grid cell that was covered by agricultural land (grassland, pasture, cropland) in 2011.

1. Available water capacity of the dominant soil component
2. Bulk density
3. Electrical conductivity of the dominant soil component
4. Organic matter
5. Average % silt
6. Average % clay
7. Average % sand
8. % area covered by Clay soil type
9. % area covered by Silty Clay soil type
10. % area covered by Sandy Clay soil type
11. % area covered by Clay Loam soil type
12. % area covered by Silty Clay Loam soil type
13. % area covered by Sandy Clay Loam soil type
14. % area covered by Loam soil type
15. % area covered by Silt Loam soil type
16. % area covered by Sandy Loam soil type
17. % area covered by Silt Loam soil type
18. % area covered by Loamy Sand soil type
19. % area covered by Sand soil type

20. pH, which is influenced by chemical reactions between water and the dominant soil component

Note that features 8-19 were not present in the original gSSURGO dataset. Rather, for each pixel, we used the raw silt, clay, and sand percentages to compute the soil texture type of that pixel, based on the National Resources Conservation Service Soil Survey's classification scheme [184]. This classification scheme is depicted in Figure 2.3. After classifying each pixel's soil texture type, we compute the fraction of each county that is occupied by each soil texture type.

Figure 2.3: NRCS Soil Texture classification [184]. The three sides of the triangle represent percentage sand, clay, and silt, and the colored regions are the soil texture types.

Extra features ($\mathbf{x}_c^e$) also come from the gSSURGO dataset [183], but are not depth-dependent. They are listed below:

1. National commodity crop productivity index
2. Depth to any soil restrictive layer

3. NCCPI crop productivity index for small grains, weighted average
4. NCCPI crop productivity index for corn
5. NCCPI crop productivity index for cotton
6. NCCPI crop productivity index for soybean

Compared Methods

We consider two types of methods: **(a) single-year methods** that only use features from year t to predict yield for the same year t, and **(b) 5-year methods** that use features from a 5-year series (years $\{t-4, t-3, \ldots, t\}$) to predict yield for year t.

Single-year methods. We first consider methods that only use a single year of data to make predictions, to provide a fair comparison to the single-year GNN. For non-deep baseline methods, we select lasso, ridge regressor and gradient boosting regressor. For these methods, we flatten all the features from the entire year into a single feature vector, ignoring the temporal and soil-depth structure in the data. Next, we tried three baseline deep learning architectures for $f_{wl}(\cdot)$: LSTM [19], GRU [75], and 1-D CNN [2]. All of these methods process the weekly time-series of weather and land surface data within the year. We compare these methods with our single-year GNN model (Eq. 2.3), which incorporates geospatial context in making predictions.

5-year methods. For history-dependent models, we follow [105] by considering a 5-year dependency for a consistent and fair comparison. Two baseline models using LSTM and GRU respectively handle the raw inputs $\{\mathbf{x}_{c,t-\Delta t}, ..., \mathbf{x}_{c,t}\}$ directly with $r(\cdot)$. Specifically, they flatten the features *for each year* into a single vector (disregarding the weekly structure of the weather data or the depth struc-

ture of the soil data), and then feed the 5 year-vectors into the LSTM or GRU. The most recent CNN-RNN model [105] pre-processes the raw features with a CNN (choosing CNN for $f_{wl}(\cdot)$) and then uses a LSTM to model the sequence embeddings as described in Eq. 2.2. Finally, the GNN-RNN model introduced in this work (Eq. 2.4) still uses a CNN for $f_{wl}(\cdot)$ to encode the raw features into an embedding for each year, then uses the GNN to refine the embeddings using information from the county's spatial context, and then passes those embeddings into an LSTM.

Evaluation Metrics

We evaluate our model across all counties in the test year with data. We use three standard regression metrics: RMSE, R^2, and Pearson correlation coefficient (Corr).

The RMSE is the square root of the mean squared error between the prediction and the true value:

$$RMSE = \sqrt{\frac{\sum_c (y_c - \widehat{y_c})^2}{N}}$$

where y_c is the true yield for county c, $\widehat{y_c}$ is the model's predicted yield for county c, and N is the total number for counties in the test set with yield data. In this work, we further divide RMSE by the standard deviation of the current crop's yield (across all years), in order to make the results for different crops comparable.

R^2 is a measure of how much the variation in the data can be explained by

the model predictions. Formally,

$$R^2 = 1 - \frac{\sum_c (y_c - \widehat{y_c})^2}{\sum_c (y_c - \bar{y})^2}$$

where $\bar{y}$ is the average yield across the entire test dataset. The top of the fraction is the sum of the squared residuals (difference between true yield and model prediction). The bottom is the total sum of squares (of the difference between the true yield and the average yield across the test dataset), which is proportional to the overall variance of the test data.

The Pearson correlation coefficient (Corr) measures the strength of the linear correlation between the true and predicted values. The correlation between two variables x and y is given as

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Again $\bar{x}$ and $\bar{y}$ are the means of x and y respectively. We let x be the model prediction and y be the true yield.

Model Details

For the shallow models (ridge regression, lasso, and gradient boosting regressor), we used scikit-learn's implementations.

For the baseline single-year models, we evaluated using LSTM, GRU, and CNN as $f_{wl}(\cdot)$ to process the weekly weather and land surface data. For CNN, we used a 1-D CNN similar to the one in [105], but we process all weather and

Method	RMSE	R^2	Corr
lasso 1y	0.7846	0.3839	0.7778
ridge 1y	0.9255	0.1428	0.7626
gradient-boosting 1y	0.7402	0.4516	0.7794
gru 1y	0.5938	0.6472	0.8158
lstm 1y	0.6146	0.6220	0.8303
cnn 1y	0.5824	0.6606	0.8235
gnn 1y **(ours)**	**0.4846**	**0.7517**	**0.8759**
(std)	*(0.0097)*	*(0.0100)*	*(0.0019)*
gru 5y	0.6765	0.5419	0.8194
lstm 5y	0.6542	0.5716	0.8060
cnn-rnn 5y	0.5511	0.6936	0.8425
gnn-rnn 5y **(ours)**	**0.4900**	**0.7595**	**0.8731**
(std)	*(0.0191)*	*(0.0186)*	*(0.0092)*

(a) 2018 corn results

Method	RMSE	R^2	Corr
lasso 1y	0.6838	0.3122	0.6715
ridge 1y	0.7081	0.2623	0.6723
gradient-boosting 1y	0.7345	0.2064	0.6857
gru 1y	0.5890	0.4897	0.7381
lstm 1y	0.6245	0.4262	0.7096
cnn 1y	0.5572	0.5432	0.7384
gnn 1y **(ours)**	**0.4930**	**0.6286**	**0.8011**
(std)	*(0.0068)*	*(0.0102)*	*(0.0037)*
gru 5y	0.5279	0.5900	0.7785
lstm 5y	0.5311	0.5849	0.7821
cnn-rnn 5y	0.5212	0.5842	0.7868
gnn-rnn 5y **(ours)**	**0.4677**	**0.6782**	**0.8272**
(std)	*(0.0035)*	*(0.0049)*	*(0.0038)*

(b) 2019 corn results

Method	RMSE	R^2	Corr
lasso 1y	0.6226	0.6090	0.7912
ridge 1y	0.7633	0.4125	0.7550
gradient-boosting 1y	0.6686	0.5492	0.7986
gru 1y	0.6376	0.5932	**0.8356**
lstm 1y	0.6459	0.5825	0.8129
cnn 1y	0.6584	0.5661	0.7988
gnn 1y **(ours)**	**0.5637**	**0.6794**	0.8273
(std)	*(0.0144)*	*(0.0163)*	*(0.0095)*
gru 5y	0.6094	0.6254	0.8218
lstm 5y	0.5430	0.7026	0.8459
cnn-rnn 5y	0.5647	0.6784	**0.8650**
gnn-rnn 5y **(ours)**	**0.5333**	**0.7129**	0.8591
(std)	*(0.0194)*	*(0.0206)*	*(0.0049)*

(c) 2018 soybean results

Method	RMSE	R^2	Corr
lasso 1y	0.5731	0.6137	0.8089
ridge 1y	0.6069	0.5668	0.7944
gradient-boosting 1y	0.6802	0.4558	0.7899
gru 1y	0.5742	0.5150	0.7569
lstm 1y	0.5907	0.4867	0.7195
cnn 1y	0.5699	0.5222	0.7385
gnn 1y **(ours)**	**0.4916**	**0.7148**	**0.8505**
(std)	*(0.0335)*	*(0.0395)*	*(0.0165)*
gru 5y	0.5751	0.6109	0.8158
lstm 5y	0.5512	0.6427	0.8156
cnn-rnn 5y	0.5365	0.6615	0.8423
gnn-rnn 5y **(ours)**	**0.4745**	**0.7349**	**0.8602**
(std)	*(0.0160)*	*(0.0179)*	*(0.0076)*

(d) 2019 soybean results

Table 2.1: Evaluation results. For RMSE, lower is better; for R^2 and Corr, higher is better. We grouped the methods based on whether they use 1 year of data (1y) or 5 years of data (5y) to make predictions.

land surface parameters together. The CNN contains series of 1D convolutions, ReLUs, and average pooling layers; this sequence is repeated four times. For all methods that use LSTM or GRU, we used PyTorch's implementation with 64 hidden states.

The same CNN is used as the encoder for the weekly weather and land surface data in the CNN-RNN, GNN, and GNN-RNN models. (We also tried using an LSTM as the encoder for the weekly data for these models, but this did not improve results.) For all methods except for the 5-year LSTM/GRU, we

processed the soil data using another small 1-D CNN (with three convolutional layers, and without average pooling), where the convolutions operate across 6 different soil depths.

For the simple 5-year baseline models (LSTM and GRU), we fed the flattened feature vectors for each year through an LSTM or GRU, followed by a 2-layer fully connected network.

For the GNN and GNN-RNN models, we used the implementation of GraphSAGE from the dgl library; we used a 2-layer GNN, with edge dropout of 0.1. The adjacency graph of US counties is provided by the US Census Bureau. We used stochastic mini-batch training to train the model, where each layer samples 10 neighbors to receive messages from. We tried different aggregation functions and found that the "pooling" approach generally performed best.

For all methods, we use the Adam optimizer [186], sometimes with a mild cosine or step decay. We tried learning rates between 1e-5 and 1e-3, used a weight decay of 1e-5 or 1e-4, and a batch size of 32, 64, or 128. We trained the model for 100 to 200 epochs (until the validation loss clearly stopped improving). We chose the epoch and hyperparameter setting that produced the lowest RMSE on the validation year (the year before the test year). We ran the GNN and GNN-RNN models 3 times with different random seeds to evaluate the variance in the results.

Crop Yield Prediction Results

We evaluate the model on four test datasets: 2018 corn, 2018 soybean, 2019 corn, and 2019 soybean. These datasets span a wide geographic area, as well

Method	RMSE	R^2	Corr
lstm 1y	0.6347	0.5968	**0.8148**
cnn 1y	0.7253	0.4736	0.7004
gnn 1y **(ours)**	**0.5877**	**0.6543**	0.8124
lstm 5y	0.7004	0.5091	0.7708
cnn-rnn 5y	0.6532	0.5730	0.7732
gnn-rnn 5y **(ours)**	**0.5836**	**0.6591**	**0.8259**

Table 2.2: Early prediction results (2018 corn, after June 1).

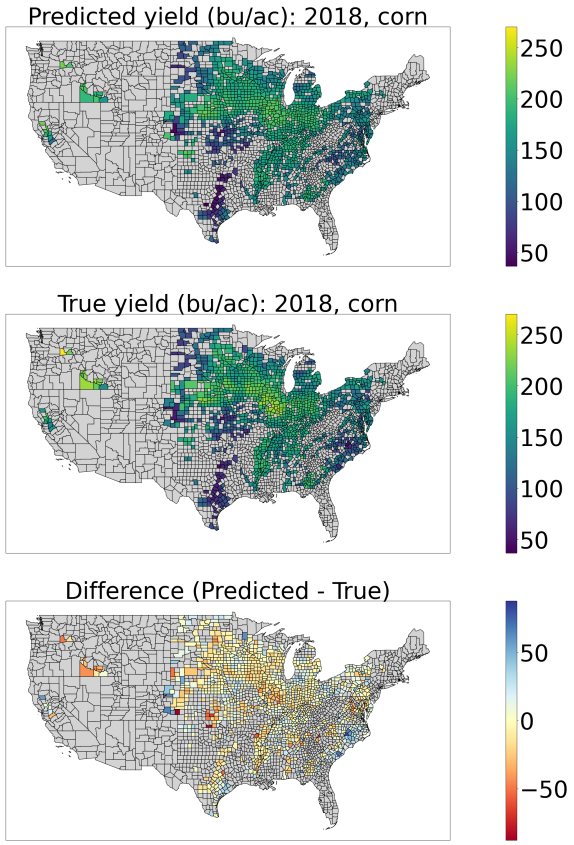


Figure 2.4: Maps of predicted (top) and true (middle) corn yields in 2018, along with the difference (bottom). For the Difference plot, yellow means an accurate prediction, blue means the model predicted too high, and red means the model predicted too low. Gray means no data.

as differing growing conditions (2019 was a bad year due to the wet spring in the Midwest, which caused planting to be delayed). The results on these datasets are shown in Table 2.1. For the methods that only use 1 year when making predictions, our GNN model clearly outperforms comparable baselines

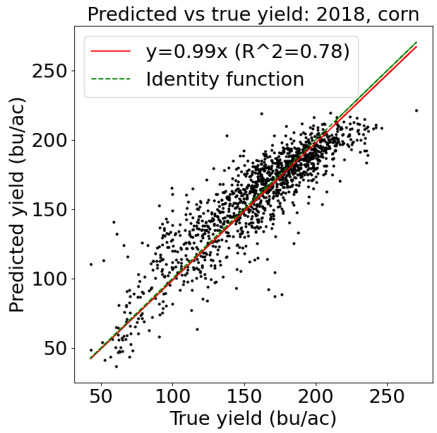


Figure 2.5: Predicted vs. ground truth corn yields in 2018.

across all datasets and metrics (except for 2018 soybean Corr, where it is slightly worse than GRU). For the methods that use a history of 5 years, our GNN-RNN outperforms competing baselines in almost all cases (except for 2018 soybean Corr, where it is slightly worse than CNN-RNN). For example, in 2019, our corn yield prediction on R^2 score is 16% better than the prediction of a state-of-the-art work [105]. On average, we achieve a relative R^2 improvement of 10.44% over the recent CNN-RNN model, 16.16% over the 5-year LSTM, and a relative RMSE improvement of 9.6% over the CNN-RNN model, 13.18% over the 5-year LSTM. These indicate the importance of exploiting geospatial context in these predictions.

Figure 2.5 shows an example scatterplot of true vs. predicted corn yields for the test year 2018 (Figure 2.7 for corn yields in 2019, Figure 2.9 for soybean yields in 2019). The GNN-RNN model is able to capture differences in yield between counties quite well. One minor issue is that the model is not able to capture the counties with very high yields very well; the model almost never predicts a yield higher than 220, but there are actually several counties with a true yield higher than this. This may stem from the fact that such high yields were almost never

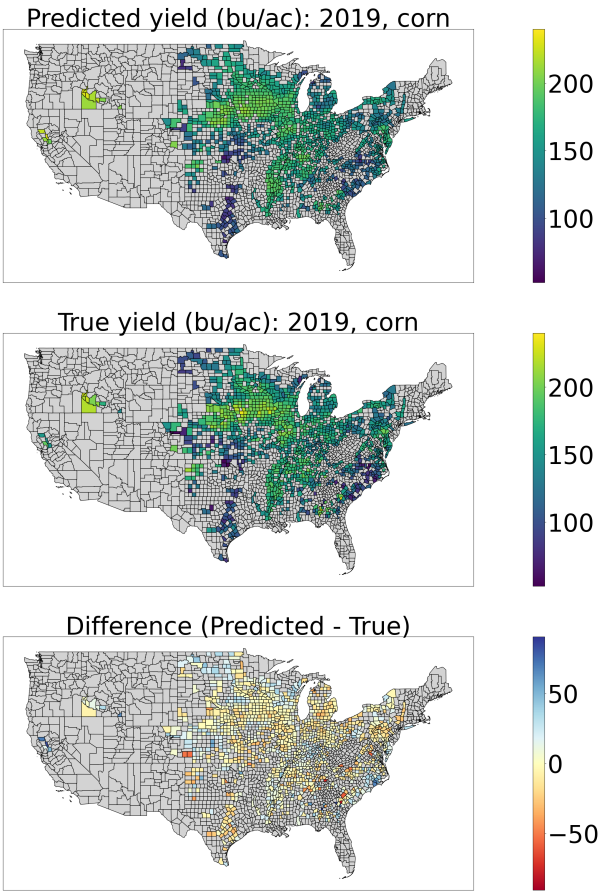


Figure 2.6: 2019 corn: Maps of predicted (top) and true (middle) yields, along with the difference (bottom).

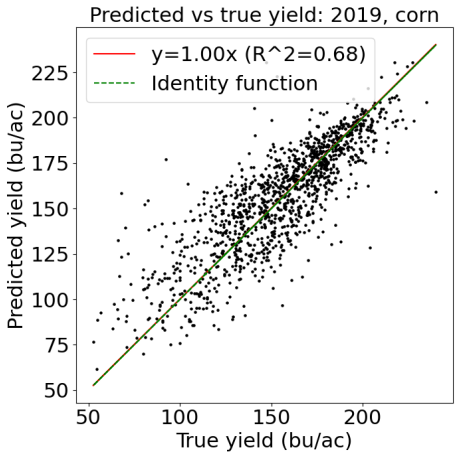


Figure 2.7: 2019 corn: Predicted vs. ground truth yields.

seen before in the training years.

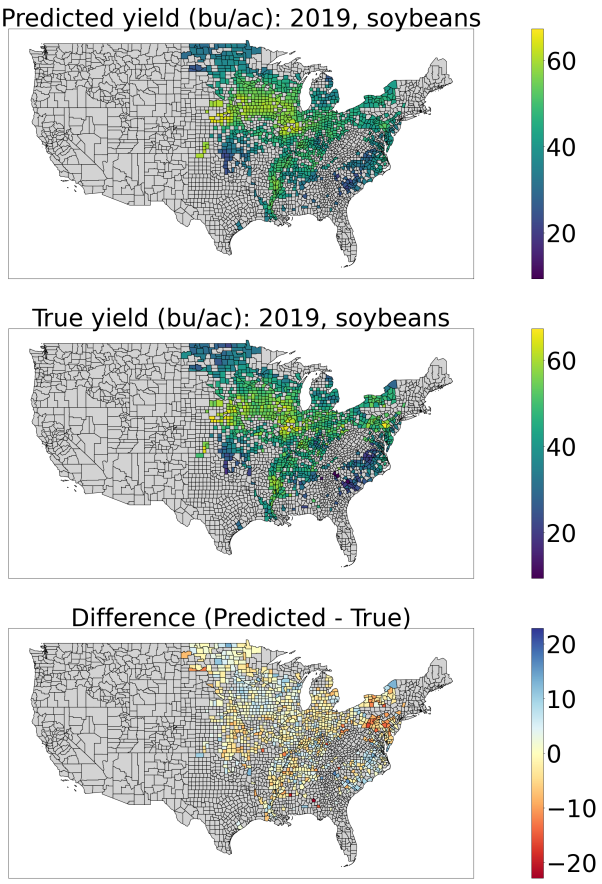


Figure 2.8: 2019 soybeans: Maps of predicted (top) and true (middle) yields, along with the difference (bottom).

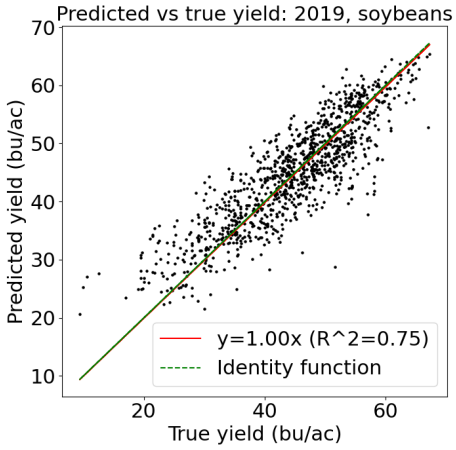


Figure 2.9: 2019 soybeans: Predicted vs. ground truth yields.

We can also see these trends in the map (Figure 2.4, 2.6, 2.8). While the GNN-RNN model captures large-scale trends in crop yield very well, it sometimes

outputs overly smooth predictions within a region, and under-predicts the area of high true yield in the Midwest. Improving the GNN's ability to detect fine-scale variations without smoothing them out is an important area for future work.

We can see that crop yield prediction on a large scale is rather challenging, due to the complexity of the prediction task and the data involved. Each data point (one county/one year) has over 6,000 features, and standard models can easily overfit to noise in the data and fail to generalize. In order for the prediction task to be tractable, a model needs to take advantage of the various forms of structure in the data; temporal structure within a year (to capture weather patterns in different times of the year), temporal structure across years (to capture long-term trends such as technological improvements), and geospatial structure (to capture correlations between nearby county yields). Our GNN-RNN model is the first model to take all of these aspects into account when making crop yield predictions, and achieves superior performance compared with the existing state-of-the-art.

Early Prediction

In practice, crop yield predictions are most useful if they can be made well before harvest, as this gives time for markets to adapt, and humanitarian aid to be organized in cases of famine [158]. To simulate this, at test time only, for each county we mask out all weather and land surface features from after June 1 (week 22) of the test year, and replace them with the average values for that county during the training years. Then we pass the masked features through a pre-trained model to obtain predictions. The results for several methods for 2018

corn are presented in Table 2.2. The graph-based models (GNN and GNN-RNN) clearly outperform competing baselines in this scenario, again illustrating the importance of utilizing geospatial context.

2.1.5 Discussion

In this work, we introduce a novel GNN-RNN framework to innovatively incorporate both geospatial and temporal knowledge into crop yield prediction, through graph-based deep learning methods. To our knowledge, our method is the first to take advantage of the spatial structure in the data when making crop yield predictions, as opposed to previous approaches which assume that neighboring counties are independent samples. We conduct extensive experiments on large-scale datasets covering 41 US states and 39 years, and show that our approach substantially outperforms many existing state-of-the-art machine learning methods across multiple datasets. Thus, we demonstrate that incorporating knowledge about a county's geospatial neighborhood and recent history can significantly enhance the prediction accuracy of deep learning methods for crop yield prediction.

2.2 Graph-to-Sequence: Crystal to Sequence Learning for Density of States Prediction

2.2.1 Introduction

Machine learning for materials discovery has risen to prominence recently, with impressive applications to materials' property predictions [187, 188]. While the performance greatly exceeds the existing models, most of the achievements are confined to individual scalar quantities [189, 190, 191, 192]. For conventional machine learning models, some other structured properties are often hard to deal with, for example spectral properties [193]. Two fundamental spectral properties are phonon density of states (phDoS) [194] and electronic density of states (eDoS) [195]. In a given frequency interval, phDoS describes the number of phonon modes if the density of wave vectors in the Brillouin zone is homogeneously distributed. The phonon density of states is typically normalized to one [196]. eDoS is essentially the number of different states at a particular energy level that electrons are allowed to occupy (namely, the number of electron states per unit volume per unit energy) [197]. Mathematically, the density of states can be regarded as a sequence depicting the fundamental characteristics of particles. Each crystalline material is associated with unique density of states (phDoS, eDoS). The goal of density of states prediction is to estimate this sequence given an input structure.

There are two challenges in this sequence prediction problem. First, the input crystals are 3D structures, which are very different from typical machine learning formats (i.e., vectors or matrices). Featurizing the crystal structures

is the cornerstone of the whole learning process. If useful information cannot be extracted effectively, the follow-up predictions would be severely hampered. Essentially, each crystal structure is a collection of atoms. Each atomic site contains a specific element, and these sites can be seen as nodes on a graph. Atoms also have interactions with each other [198]. Close atoms often have strong bonds. These bonds can be represented by the connecting edges on the graph [189]. Conventional CNN [1] and RNN [19] are not applicable to graphs, while graph neural networks [170] are specifically designed for such structures, and some GNN examples have been seen in Section 2.1. The pioneering work applying GNN networks to materials property prediction was CGCNN [189], where the crystal unit cell was formalized as a graph with atoms as nodes and bonds as edges. MEGNet [199] further introduced a global state to add extra features like temperature, pressure, and entropy. More recently, GATGNN [59] suggested using an attention mechanism to capture the global crystal structure beyond the local atomic environments. In GATGNN, the local augmented graph-attention layers encode the properties of local environments for each atom to obtain an abstract feature, and a global attention layer aggregates these atomic vectors by weighted averaging to produce the global feature vector for prediction. This exploitation of structure markedly facilitated feature extraction from the crystalline systems, and built an extensible foundation for various predictions.

The second challenge in sequence prediction is the target format. Works like GATGNN concentrated on the prediction of scalar properties such as formation energies, band gaps and elastic moduli. The performance of these models degrades when applied to spectral properties like eDoS and phDoS [10]. The simplest way to adapt the previous methods to predict sequences is to expand the last layer to output multiple values instead of a single value. The general

idea is similar to predicting multiple scalar properties simultaneously. However, such parallel prediction overlooks the sequential nature of the target DoS. The smoothness property cannot be enforced onto the predicted targets either. Mat2Spec [10] observes this flaw and re-formulates the problem as a multi-target regression task. Mat2Spec keeps the framework of expanding the last layer to output more values, but additionally uses contrastive learning [97] and a probabilistic variational autoencoder [200] to model the correlations among different locations of the sequence.

In this work, we introduce Crystal-to-DoS (Xtal2DoS) learning, to improve both the graph encoding and the sequence decoding for phDoS and eDoS predictions. First, we inherit the general framework of GATGNN and employ graph attention networks (GAT) [107] to learn both the local and global attention. Second, we adopt the attention-based transformer to decode the target sequence. [28]. The transformer reads the one-hot position embedding together with the global crystal representation, and decodes left-to-right auto-regressively. Differing from the multi-target regression perspective, Xtal2DoS takes the sequence-to-sequence (seq2seq) learning perspective. In seq2seq, the decoding process incorporates both self-attention and source-attention. Self-attention retrieves the historical memory from the previous steps, and source-attention derives the similarity between the decoding sequence and the input sequence. In this DoS prediction task, the input is a graph rather than a sequence, so the source-attention attends each position with all the atom embeddings. The resulting model, Xtal2DoS, thus improves both the encoding and decoding over prior models, for this graph-to-sequence learning problem. Experimental results show our method can outperform the existing state-of-the-art models on all four evaluation metrics by over 15% on average for phDoS, and over 6% on average for eDoS.

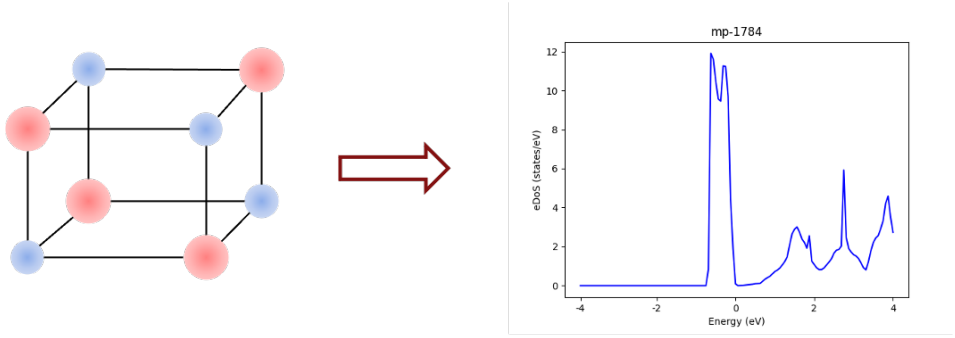


Figure 2.10: An illustration of predicting eDoS from the crystal structure for material *CsF*.

2.2.2 Crystal-to-DoS Learning

Problem Formulation

The input crystal material x is denoted as a graph $G_x = (V, E)$. $V = \{u_i\}_{i=1}^{n}$ is the collection of n atoms where u_i is an atom embedding learnt from CGCNN. $E = \{e_{ij}\}_{1 \leq i,j \leq n}$ denotes the bonds connecting atoms. e_{ij} depicts the distance between two atoms i, j. For a crystalline system, the crystal structure repeats in the 3D space and each atom has a 3D coordinate. Though theoretically, each atom has connections with any other atom, for simplicity and generality, we choose n_{max_nbr} neighbors to be the upper bound for the number of neighbors of each atom. In other words, the sum of each row or column of G_x's adjacency matrix is smaller than or equal to n_{max_nbr}. Our goal is to predict y for each x. y is the desired density of states (phDoS or eDoS), with length l_y. $l_y = 51$ for phDoS and $l_y = 128$ for eDoS. l is predefined for each type of DoS. The learnt model maps x to y ($f : x \rightarrow y$). Figure 2.10 presents an example (caesium fluoride , *CsF*) of DoS prediction.

Graph Attention Encoder

As the prototype of GAT, graph convolutional network (GCN) [169] combines the local graph structure and node-level features to yield impressive node predictions. However, the feature aggregation of GCN depends on structure and is mostly linear, constraining effectiveness of message passing on the graph. In GCN, a graph convolution operation can be written as

$$h_i^{l+1} = \sigma(\sum_{j \in N(i)} \frac{1}{c_{ij}} W^l h_j^l) \tag{2.6}$$

$\sigma(\cdot)$ is a non-linear activation function. W_l is the feature transformation matrix in the l-th layer. h_i^l is the latent node embedding for node i in the l-th layer, with $h_i^0 = u_i$. $c_{ij} = \sqrt{|N(i)||N(j)|}$ is a normalization constant and $N(i)$ is the set of node i's neighbors.

In Xtal2DoS, we use a variant of GAT, *UniMP* [201], to encode the crystalline graph. Compared to GAT, UniMP adds support for edge features. For each layer in UniMP, h_i^l is first linearly transformed to queries, keys and values

$$\begin{aligned} q_i^l &= W_q^l h_i^l + b_q^l, \\ k_i^l &= W_k^l h_i^l + b_k^l, \\ v_i^l &= W_v^l h_i^l + b_v^l. \end{aligned} \tag{2.7}$$

Similarly, we also transform the edge features between node i and node j

$$g_{ij} = W_e e_{ij} + b_e. \tag{2.8}$$

The edge feature transformation is not layer-dependent and W_e, b_e are shared for all edges. The attention coefficients are then computed as follows

$$\alpha_{ij}^l = \frac{\langle q_i^l,\ k_j^l + g_{ij} \rangle}{\sum_{p \in N(i)} \langle q_i^l,\ k_p^l + g_{ip} \rangle} \tag{2.9}$$

where $\langle q, k \rangle = \exp(\frac{q^T k}{\sqrt{d}})$ and d is the dimensionality of q_i^l, k_i^l or v_i^l. Afterwards, the embeddings from neighbors are aggregated, together with the edge embeddings, weighted by attention scores

$$\hat{h}_i^{l+1} = \sum_{j \in N(i)} \alpha_{ij}^l \cdot (v_j^l + e_{ij}). \tag{2.10}$$

From Eq. 2.6 and Eq. 2.10, GAT considers similarity scores and edge information when performing the aggregation to gain more expressiveness.

Finally, motivated by the success of residual neural networks [21], UniMP appends a gated residual connection before passing $\hat{h}_i^{l+1}$ to the next layer

$$\begin{aligned} r_i^l &= W_r^l \, h_i^l + b_r^l, \\ \beta_i^l &= \text{sigmoid}(a^l \, [\hat{h}_i^{l+1} \; ; \; r_i^l \; ; \; \hat{h}_i^{l+1} - r_i^l]) \\ h_i^{l+1} &= \sigma(f_{LN}((1 - \beta_i^l) \, \hat{h}_i^{l+1} + \beta_i^l \, r_i^l)) \end{aligned} \tag{2.11}$$

where ; denotes concatenation, a^l is the learnable weight parameter, and f_{LN} represents the layer norm. Note that for the last layer of UniMP, $\sigma(\cdot)$ and f_{LN} are skipped.

Sequential Decoding

GAT learns embeddings for each node, h_i^L, if there are L layers in total. GATGNN and Mat2Spec simply take the average of all the embeddings as the global feature vector

$$h_x = \frac{1}{n} \sum_{i=1}^{n} h_i^L \tag{2.12}$$

Then the global feature vector is used to decode the target DoS, using MLP or VAE.

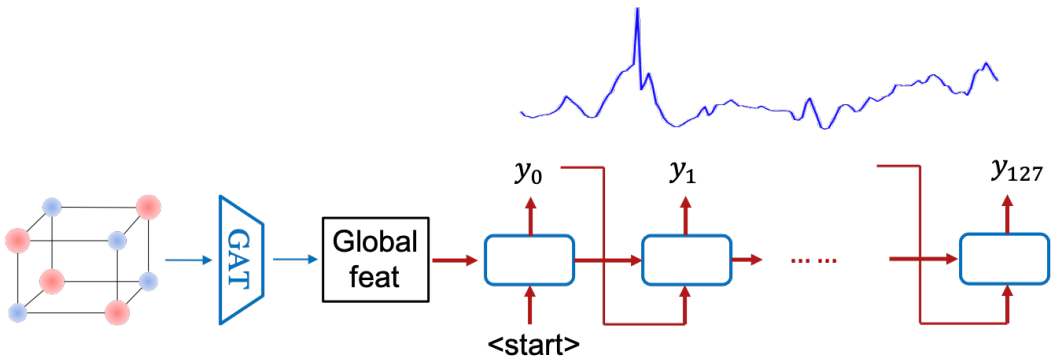


Figure 2.11: The global feature vector is fed into RNN as the initial state. The decoding unrolls step by step.

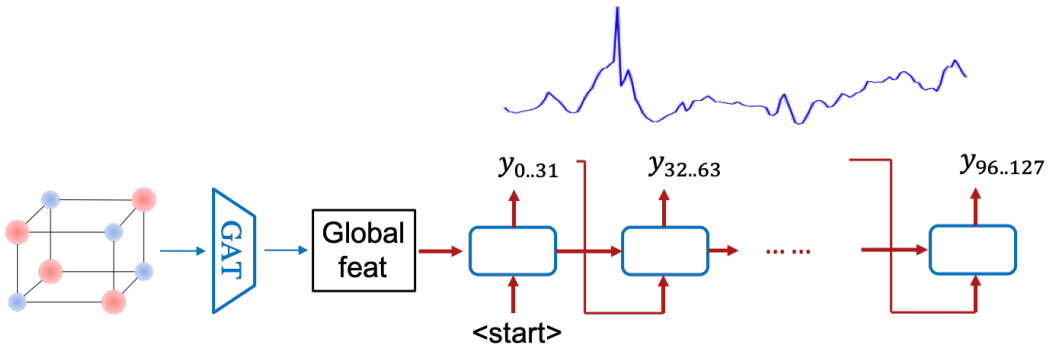


Figure 2.12: Chunk RNN decoder is an RNN decoder that predicts a segment instead of a single value at each step.

We compared various sequence models including RNN and transformer for decoding. By default, we used the GRU model for RNN.

RNN Figure 2.11 sketches the RNN decoding of the GAT-encoded crystal. RNN is naturally suitable for sequence decoding. It reads the global feature vector as the cell state and a special $\langle start \rangle$ symbol to indicate the start of the decoding. The output at each step is also the input for the next step. However, RNN is criticized for lacking long-term memory [202]. When we decode the last step (y_{127}), the memory of the first step (y_0) is probably degraded. This might hurt the prediction quality for long sequences. Another drawback of RNN is the speed. Each step has to wait for the previous output, and thus the decoding process

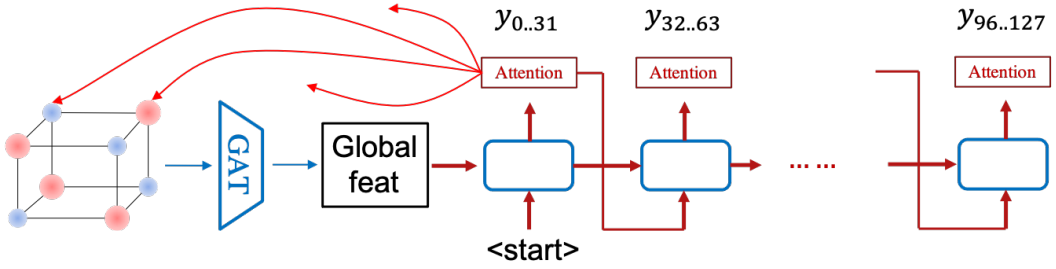


Figure 2.13: Adding an attention mechanism to the chunk RNN decoder.

could take a very long time.

Chunk RNN Figure 2.12 gives an overview of a chunk RNN, which is a simple RNN variant. Chunk RNN predicts a segment rather than a single value at each step. The benefit is that the decoding time could be greatly shortened. For instance, on eDoS, if the segment length is 32, then sequence length can be reduced from 128 to 4. On the other hand, each segment still waits for the generation of the previous segment, and longer segments harm the accuracy gradually.

Chunk RNN + Attention To compensate for the performance degradation brought by chunk RNN, we further introduce the attention mechanism into the sequence model (Figure 2.13). The attention module directly attends to the latent sequence states with the atom embeddings. The latent states can take specific atoms into consideration and build more fine-grained correlations. In practice, we also find including the attention module boosts the evaluation results.

Transformer Given that we have already introduced the attention mechanism, it is intuitive to completely drop the recurrent framework and embrace the fully attentional transformer [28]. The input at each step is the concatenation of the positional embeddings and the global feature vector. A positional embedding is simply a one-hot vector of length l_y indicating the index of the current step. The

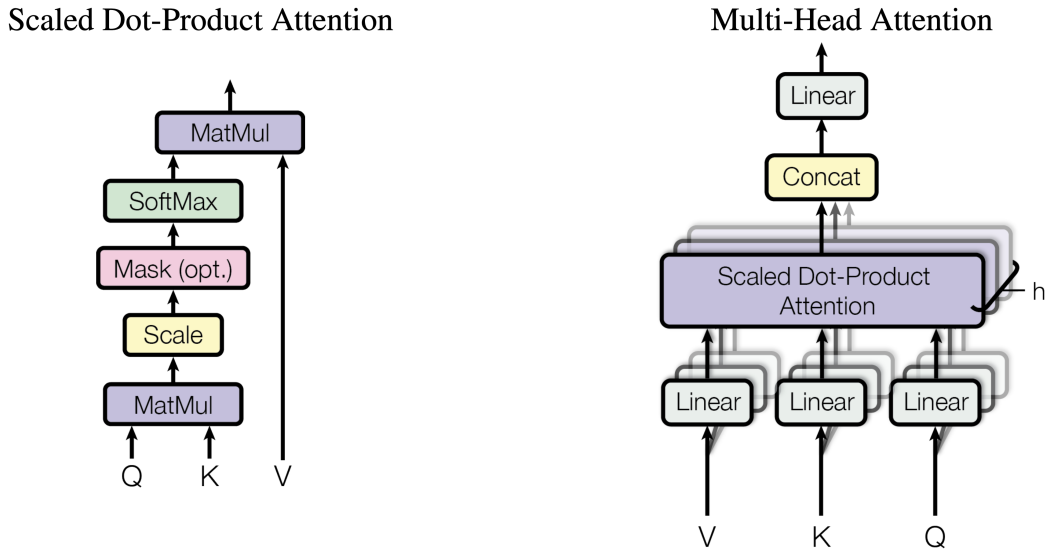


Figure 2.14: Images are from [28]. Left: Illustration of the scaled dot-product attention. Right: Several attention layers can run in parallel.

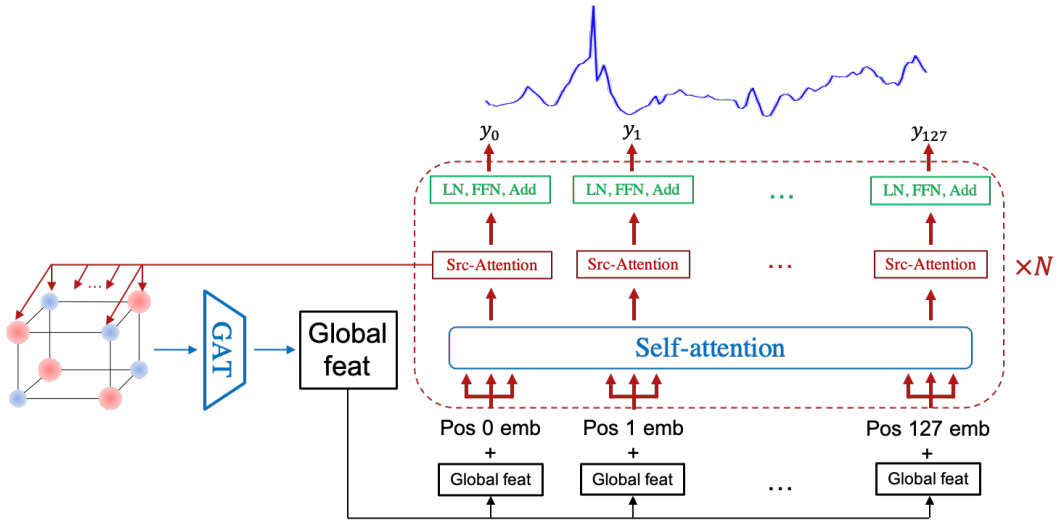


Figure 2.15: Fully attention-based transformer decoding for DoS prediction. All the recurrent components are replaced with self-attention.

global feature vector h_x additionally provides the global context for decoding. Formally, let $z_i = [s_i\ ;\ h_x]$ where s_i is the one-hot position embedding with length l_y. z_i is the initial state for the first self-attention layer. As in UniMP, we derive collections of queries, keys and values for the hidden states of each self-attention layer, denoted by Q, K, and V, respectively.

Transformer adopts dot-product attention to calculate the outputs

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^T / \sqrt{d})\, V \tag{2.13}$$

where d is the latent dimensionality. Multiple sets of Q, K, and V matrices can be learned, and each set is known as a head. Multiple heads (multi-head) can give self-attention greater power of discrimination. The outputs of all the heads are concatenated, layer-normalized and decoded through a fully connected feed-forward network. Figure 2.14 shows an example of h heads in total.

During the decoding phase, each attention layer also computes the source-attention between the output states and the input atom nodes. If we denote the queries, keys, values in decoding as $Q_{dec}, K_{dec}, V_{dec}$, and the ones from the input graph encoding as $Q_{enc}, K_{dec}, V_{dec}$, we have

$$\begin{aligned} \text{Self-Attention}(Q_{dec}, K_{dec}, V_{dec}) &= \text{softmax}(\frac{Q_{dec}K_{dec}^T}{\sqrt{d}})V_{dec}, \\ \text{Source-Attention}(Q_{dec}, K_{enc}, V_{enc}) &= \text{softmax}(\frac{Q_{dec}K_{enc}^T}{\sqrt{d}})V_{enc}, \end{aligned} \tag{2.14}$$

for the sequential decoding. Source-attention usually happens after self-attention. Furthermore, the transformer model adds masking to prevent the current positions from attending to the future positions, to prevent leftward information flow in the decoder, and to preserve the auto-regressive property, ensuring each decoded state only knows the history but not the future. The final generated length-l_y sequence $\hat{y}$ is compared with the ground-truth y through Kullback–Leibler divergence (KL-div) [203] or its generalized version, *Bregman Divergence* (BD) [204], resulting in the loss term, $\mathcal{L} = D_{KL}(y \parallel \hat{y})$. We chose KL-div and BD because they are more suitable than other measures for data containing peaks, which are ubiquitous in DoS.

Our model is a synergistic combination of a GAT (UniMP) encoder and a

transformer decoder. The whole model is entirely based on attentions on graphs and sequences. Figure 2.15 illustrates the main idea of Xtal2DoS.

2.2.3 Related Works

Learning for 3D structure has been a long-standing problem. In materials discovery, two atomic systems can be exactly the same even though the coordinates are totally different in the 3D space if they can be transformed to one another through translation, rotation or inversion. For standard machine learning models, it is very hard to capture such invariance and symmetry [205]. There are in general three strategies to address the concern.

First, the same models can be trained using more data [206]. The structured inputs can be randomly transformed without changing the labels. Some prior works successfully applied data augmentation to exploring materials design spaces [207]. On the other hand, in materials science, even with augmentation, it is not easy for machine learning models to comprehend the crystalline system, let alone the data collection and augmentation require enormous domain knowledge and expert effort [188].

Second, rather than expanding the scale on the dataset side, some researchers have explored the possibility of enforcing the invariance or equivariance inside the model. This kind of model handles the coordinates directly and are often called equivariant neural networks [208]. One notable model for materials science applications is E3NN [209]. E3NN performs training based on radial functions and spherical harmonics [210]. Intuitively, E3NN represents the atoms with the 3D coordinates, mass and velocity, and tells the equivariant convolutional

filters how the crystals might transform. Thus, E3NN can capture the full crystal symmetry, while reducing training set sample complexity. However, radial functions and spherical harmonics may limit the expressiveness of the model. In some scenarios where the data is abundant, E3NN's benefits might become marginal.

Third, it is often helpful to pre-process the data for simplicity [211]. Instead of using the coordinates, we can re-format the crystals to graphs by retaining only the atoms and the bonds between them. Some of the structure information is lost from the input, such as the lattice angles, but this could also reduce the risk of overfitting. Once the inputs are converted to graphs, mature graph neural network techniques [160] can be applied. A representative work is GATGNN [59]. GATGNN is one of the primary methods to encode the atom embeddings and extract the global feature vector. It still leads in the performance of many scalar property prediction tasks including absolute energy, Fermi energy, band gap and Poisson-ratio. Nevertheless, in sequence property predictions, its performance degrades significantly [10]. Mat2Spec [10] uses the same graph encoder as GATGNN, but improves the decoder. GATGNN only uses an MLP, which is incapable of sequence decoding. Mat2Spec not only aligns the inputs and targets in a latent multivariate Gaussian space, but also learns an embedding for each energy level through contrastive learning. Mat2Spec views the DoS prediction as a multi-target regression problem, yet ignores the sequential structure of DoS. Besides, Mat2Spec simply deepens the MLP decoder and plugs contrastive modules into the decoder, while sequence modeling is cleaner and simpler.

Xtal2DoS follows the design of GATGNN and Mat2Spec because the graph encoding is shown to outperform the coordinate encoding [10]. On top of the

existing designs, Xtal2DoS further replaces GAT encoder with a more advanced variant, UniMP, and replaces the decoder with a powerful sequence model, transformer. Evaluation results and ablation studies prove the effectiveness and efficiency of our introduced model.

2.2.4 Experiments on DoS Prediction

We validate the performance of Xtal2DoS on two datasets, phDoS and eDoS. The task of directly predicting phDoS in crystalline solids from atomic species and positions was first proposed in E3NN [209]. As mentioned above, phDoS is normalized to 1, since phDoS gives the number of modes per unit frequency per unit volume of real space and is essentially a distribution. phDoS is critical for calculating properties such as the average phonon frequency and the heat capacity at 300K. The analogous task of predicting eDoS for nonmagnetic materials was proposed in [10]. eDoS is the same as phDoS except that it depicts electrons. Materials scientists are interested in a small energy range between -4 to 4 eV with respect to band edges with 63 meV intervals, including both unoccupied and occupied states. Fermi energy and band edges are all set to 0 eV on this energy grid. Predicting eDoS accurately can be very valuable to bandgap energy estimation. Both phDoS and eDoS are continuous functions in nature. We sample 51 points on phDoS and 128 points on eDoS spanning the frequency or energy range. In other words, $l_y = 51$ for phDoS and $l_y = 128$ for eDoS.

Evaluation Metrics

Four metrics are adopted for evaluation: coefficient of determination (R^2), mean absolute error (MAE), mean squared error (MSE) and Wasserstein distance (WD).

R^2 measures the proportion of the variation in the ground-truth targets y that is predictable from the predictions $\hat{y}$,

$$R^2 = 1 - \frac{\sum_{i=1}^{l_y}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{l_y}(y_i - \bar{y})^2} \tag{2.15}$$

where $\bar{y} = \frac{1}{l_y}\sum_i y_i$. In the best case when y and $\hat{y}$ match exactly, R^2 would reach the maximum possible value, 1. If $y_i = \bar{y}$, $R^2 = 0$. Note that R^2 score can be negative if the predictions are worse.

MAE and MSE are standard metrics for regression. We also adopt them for evaluation:

$$\begin{aligned} \text{MAE} &= \frac{1}{l_y}\sum_{i=1}^{l_y}|y_i - \hat{y}_i| \\ \text{MSE} &= \frac{1}{l_y}\sum_{i=1}^{l_y}(y_i - \hat{y}_i)^2 \end{aligned} \tag{2.16}$$

Wasserstein distance is another metric defined between probability distributions. If we view the two distributions as two piles of earth, WD is the minimum amount of dirt to move in order to turn one pile to be the same as the other. Therefore, WD is also known as earth mover's distance. We define the WD as

$$\text{WD} = \int_{-\infty}^{+\infty}|P - Q| \tag{2.17}$$

for distributions p and q with cumulative distribution functions (CDF) P and Q, respectively. The continuous form of WD can also be adapted for discrete cases.

model	normalized			
	R^2	MAE	MSE	WD
Mat2Spec	0.641	0.00773	0.000264	0.0736
Xtal2DoS	0.732	0.00677	0.000194	0.0634
improvement	14.20%	12.42%	26.52%	13.86%

model	original			
	R^2	MAE	MSE	WD
E3NN	0.480	0.105	0.036	-
GATGNN	0.450	0.105	0.042	-
Mat2Spec	0.620	0.0782	0.0231	0.0736
Xtal2DoS	0.673	0.0659	0.0172	0.0634
improvement	8.55%	15.73%	25.54%	13.86%

Table 2.3: Top: The evaluation results on four metrics in the normalized space. Bottom: The evaluation results on four metrics in the original space. WD is always computed after normalization by the definition.

Implementation

The graph encoder is built with PyTorch Geometric [212] modules. The encoder is composed of three GAT/UniMP layers, each followed by batch normalization [213] and a non-linear activation [214]. The sequence decoder is a stack of six self-attention layers, where each layer has subsequent layer normalization and feed-forward layers. The whole model is trained with the Adam optimizer [186], on Nvidia V100 GPU.

Prediction Results

We evaluate the results on two spaces. One is the sum-normalized space (sum to 1) and the other is the original input space. For phDoS, the highest density value is 1 in the raw input space. For eDoS, the highest density value could exceed 50. Two spaces give different weights to different samples, leading to different

model	normalized			
	R2	MAE	MSE	WD
Mat2Spec	0.351	0.00326	0.000036	0.230
Xtal2DoS	0.385	0.00314	0.000034	0.210
improvement	9.52%	3.50%	5.56%	8.50%

model	original			
	R2	MAE	MSE	WD
E3NN	0.410	5.01	101.9	0.47
GATGNN	0.32	4.89	118.2	0.35
Mat2Spec	0.545	3.8939	76.624	0.230
Xtal2Dos	0.575	3.7353	71.566	0.210
improvement	5.50%	4.07%	6.60%	8.50%

Table 2.4: Top: The evaluation results on four metrics in the normalized space. Bottom: The evaluation results on four metrics in the original space. WD is always computed after normalization according to definition.

metric values.

Xtal2DoS is first evaluated on phDoS, compared with other state-of-the-art baselines, including E3NN, GATGNN, Mat2Spec and some of their variants. Among all the compared existing methods, Mat2Spec gives the best numbers. Our Xtal2DoS can still greatly improve over Mat2Spec. On R^2, MAE, MSE and WD, the relative improvements are as large as 14.20%, 12.42%, 26.52%, 13.86%, respectively. The average improvement is 16.75%. Similarly, in the original space, the average improvement across all four metrics can also reach 15.92%. Table 2.3 presents all the evaluation values for phDoS.

Table 2.4 shows the performance gain achieved by Xtal2DoS over Mat2Spec on the more challenging eDoS dataset. Xtal2DoS outperformed Mat2Spec on the four metrics by 9.52%, 3.50%, 5.56%, 8.50% respectively, in the normalized space. The improvement was 6.77% on average. This is narrower than the one from phDoS dataset, but the over 6% performance boost is still substantial, especially

model	normalized			
	R2	MAE	MSE	WD
RNN	0.663	0.00794	0.000245	0.0797
RNN+Attn	0.668	0.00726	0.000241	0.0664
Chunk RNN+Attn	0.693	0.00720	0.000223	0.0656
Xtal2DoS	0.732	0.00677	0.000194	0.0634
improvement	5.63%	5.97%	13.00%	3.35%

model	original		
	R2	MAE	MSE
RNN	0.588	0.0757	0.0217
RNN+Attn	0.622	0.0698	0.0199
Chunk RNN+Attn	0.643	0.0687	0.0188
Xtal2DoS	0.673	0.0659	0.0172
improvement	4.67%	4.08%	8.51%

Table 2.5: Ablation study on phDoS. The evaluation results for different module combinations in the normalized space (top) and original space (bottom).

model	normalized			
	R2	MAE	MSE	WD
Chunk RNN+Attn	0.371	0.00320	0.000035	0.221
Xtal2DoS	0.385	0.00314	0.000034	0.210
improvement	3.77%	1.88%	2.86%	4.98%

model	original		
	R2	MAE	MSE
Chunk RNN+Attn	0.558	3.9125	75.013
Xtal2DoS	0.575	3.7353	71.566
improvement	3.05%	4.53%	4.60%

Table 2.6: Ablation study on eDoS. The evaluation results for different module combinations in the normalized space (top) and original space (bottom).

in the tough eDoS prediction task. Likewise, the average improvement in the original input space is also as high as 6.17%, which is a significant increment.

model	Time per epoch (s)
Mat2Spec	33
RNN	54
RNN+Attn	69
Chunk RNN+Attn	42
Xtal2DoS	11

Table 2.7: The training time for Mat2Spec, RNN-based sequence models and Xtal2DoS. For fair comparison, we illustrate the training cost for each epoch in seconds.

Ablation Study

To validate the neural module selection of Xtal2DoS, we further compare the performance of RNN, RNN+attention, Chunk RNN+attention, and the transformer (Xtal2DoS). We use the same four metrics to compare these settings. In Table 2.5 and 2.6, transformers always dominate other RNN-based models on all the metrics. The improvements shown in the last row indicate the comparison between Xtal2Dos and the best compared model (usually Chunk RNN+Attn). The average relative improvements are 5.75% and 4.06% in the original space on phDoS and eDoS.

Qualitative Results

We also show some qualitative results on both phDoS and eDoS in Figure 2.16 and 2.17. Both figures demonstrate the accuracy of our predictions. The red and blue lines are always closely correlated.

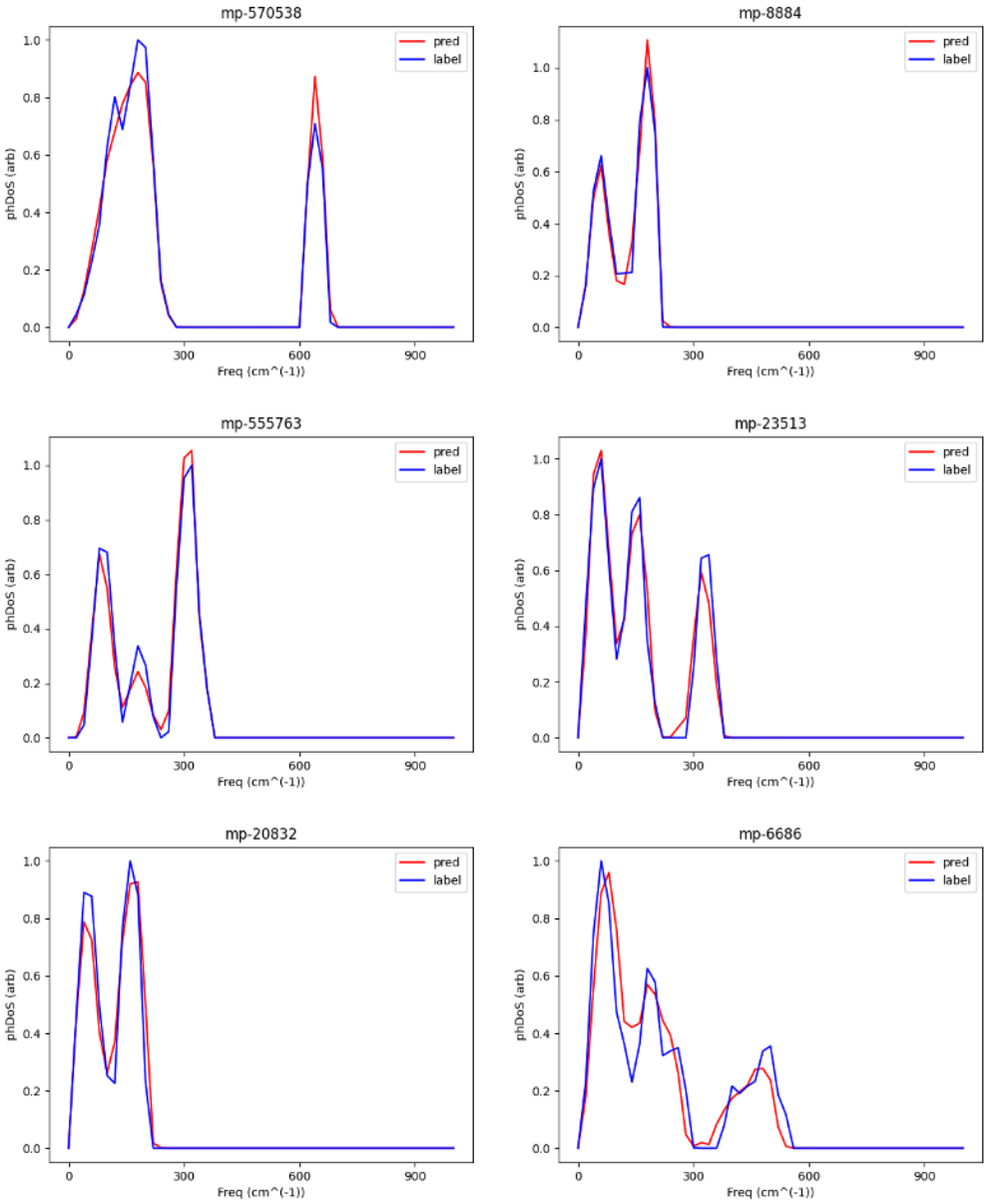


Figure 2.16: Comparison of the phDoS predictions and ground-truth labels on some crystalline systems. The red line denotes the prediction and the blue line denotes the ground-truth.

Training Speed

One major advantage of transformer models is parallelism. Unlike RNNs where each step has to wait for the previous output, transformer computes the out-

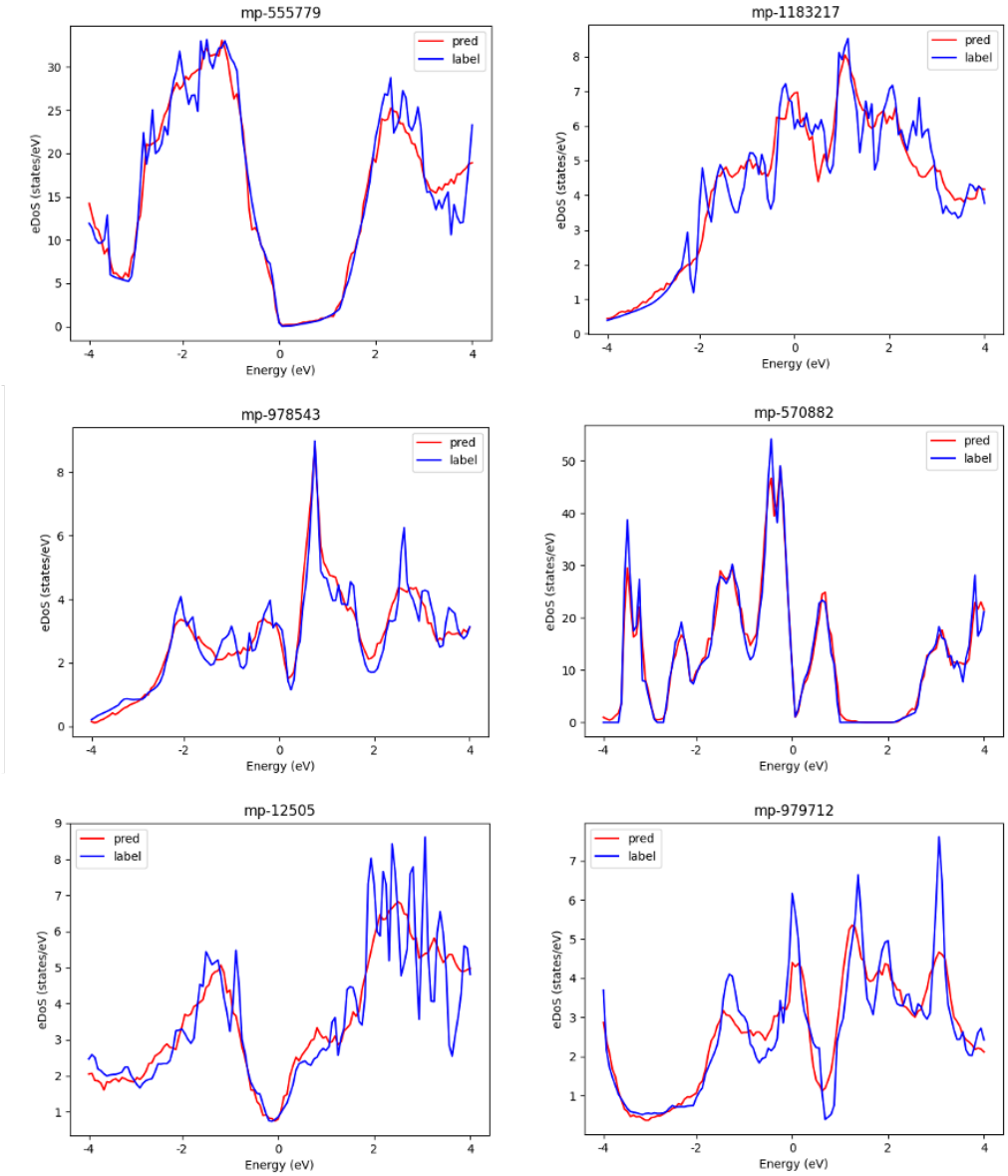


Figure 2.17: The comparison between the eDoS predictions and ground-truth labels on some crystalline systems. The red line denotes the prediction and the blue line denotes the ground-truth.

puts at all the steps simultaneously and enforces auto-regressiveness through masking. These operations can be performed in matrix form, taking advantage of powerful GPUs, and thus substantially reduce the training cost. Table 2.7 shows the training time per epoch for Mat2Spec, RNN-based sequence models and Xtal2DoS. Mat2Spec does not involve any sequential model and generates

outputs through MLP, so it is faster than any RNN-based sequence decoder model. Mat2Spec instead integrates probabilistic sampling and contrastive learning, so it is not surprising that it takes more time than Xtal2DoS. Unsurprisingly, RNN trains more than 50% slower than Mat2Spec. Adding the attention module further prolongs the time cost. RNN+Attention doubles the training cost compared with Mat2Spec. This also motivated us to attempt chunk RNN. As a result, chunk RNN+Attention can complete a training epoch faster than RNN and RNN+Attention, but is still slower than Mat2Spec. Xtal2DoS is the fastest method among all these models. It only takes one third of Mat2Spec's training time, one fifth of RNN's training time, and one fourth of chunk RNN+Attention's training time. Xtal2DoS excels at speed because of its parallel design and its succinct structure to avoid extra neural module. Faster models are often more advantageous in training, tuning and deploying.

2.2.5 Discussion

In this work, we introduce a novel graph-to-sequence learning framework, *Xtal2DoS*, for predicting the spectral properties of materials, *phDoS* and *eDoS*, from the input crystalline structures. Our model adopts an improved graph attention network (to handle edges) as the graph encoder, and a transformer as the decoder for sequential decoding. Xtal2DoS is fully attentional and captures all the atom-atom, atom-sequence and step-step correlations. Therefore, it considerably exceeds other existing state-of-the-art baselines, in quantitative evaluation metrics, qualitative matching results and training speed.

2.3 Sequence-to-Graph: Multi-Label Classification on Text Data and Others

2.3.1 Introduction

In many machine learning tasks, an instance can have several labels. The task of predicting multiple labels is known as multi-label classification (MLC). MLC is common in domains like computer vision [215], natural language processing [216] and biology [217]. Unlike the single-label scenario, label correlations are more important in MLC. Early works capture the correlations through classifier chains [218], Bayesian inference [219], and dimensionality reduction [220].

Thanks to the huge capacity of neural networks (NN), many previous methods can be improved by their neural extensions. For example, classifier chains can be naturally enhanced by recurrent neural networks (RNN) [215]. The non-linearity of NN alleviates the complex design of feature mapping and many deep models can therefore focus on the loss function, feature-label and label-label correlation modeling.

One trending direction is to learn a deep latent space shared by features and labels. The encoded samples from the latent space are then decoded to targets. One typical example is C2AE [221], which learns latent codes for both features and labels. The latent codes are passed to a decoder to derive the target labels. C2AE minimizes an ℓ_2 distance between the feature and label codes, together with a relaxed orthogonality regularization. However, the learnt deterministic latent space lacks smoothness and structures. Small perturbations in this latent space can lead to totally different decoding results. Even if the corresponding feature

and label codes are close, we cannot guarantee the decoded targets are similar. To address this concern, MPVAE [222] proposes to replace the deterministic latent space with a probabilistic space under a variational autoencoder (VAE) framework. The Gaussian latent spaces are aligned with KL-divergence, and the sampling process enforces smoothness. Similar ideas can be found in [223]. However, these methods assume a unimodal Gaussian latent space, which is known to cause over-regularization and posterior collapse [224, 225]. A better strategy would be to learn a multimodal latent space. It is more reasonable to assume the observed data are generated from a multimodal subspace rather than a unimodal one.

Another popular group of methods focuses on better label correlation modeling. Their idea is straightforward: some labels should be more correlated if they co-appear often while others should be less relevant. Existing methods adopt pairwise ranking loss, covariance matrices, conditional random fields (CRF) or graph neural nets (GNN) to this end [109, 226, 227, 111, 228]. These methods often either constrain the learning through a predefined structure (which requires a larger model size), or aren't powerful enough to capture the correlations (such as pairwise ranking loss).

Our idea is simple: we learn embeddings for each label class and the inner products between embeddings should reflect the similarity. We further learn feature embeddings whose inner products with label embeddings correspond to feature-label similarity and can be used for prediction. We assume these embeddings are generated from a probabilistic multimodal latent space shared by features and labels, where we use KL-divergence to align the feature and label latent distributions. On the other hand, one might be concerned that

embeddings alone won't capture both label-label and label-feature correlations, which were usually modeled by extra GNN and covariance matrices in prior works [111, 222]. To this end, we stress on the loss function terms rather than extra structure to capture these correlations. Intuitively, if two labels co-appear often, their embeddings should be close. Otherwise, if two labels seldom co-appear, their embeddings should be distant. A triplet-like loss could be naturally applied in this scenario. Nonetheless, its extension, contrastive loss, has shown to be even more effective than the triplet loss by introducing more samples rather than just one triplet. We show that contrastive loss can pull together correlated label embeddings, push away unrelated label embeddings (see Figure 2.19), and even perform better than GNN-based or covariance-based methods.

Our new model for MLC, contrastive learning boosted Gaussian mixture variational autoencoder (C-GMVAE), alleviates the over-regularization and posterior collapse concerns, and also learns useful feature and label embeddings. C-GMVAE is applied to three text datasets as well as some other biology and ecology datasets, and outperforms the existing methods on five metrics. Moreover, we show that using only 50% of the data, our results can match the full performance of other state-of-the-art methods. Ablation studies and interpretability of learnt embeddings will also be illustrated in the experiments. Our contributions can be summarized in three aspects: **(i)** We adopt contrastive loss instead of triplet or ranking loss to strengthen the label embedding learning. We empirically show that by using a contrastive loss, one can get rid of heavy-duty label correlation modules (e.g., covariance matrices, GNNs) while achieving even better performances. **(ii)** Though contrastive learning is commonly applied in self-supervised learning, our work shows that by properly defining anchor, positive and negative samples, contrastive loss can leverage label information very

effectively in the supervised MLC scenario as well. **(iii)** Unlike prior probabilistic models, C-GMVAE learns a multimodal latent space and integrates the probabilistic modeling (VAE module) with embedding learning (contrastive module) synergistically.

2.3.2 Problem Formulation and Methods

In MLC, given a dataset containing N labeled samples (x, y), where $x \in \mathbb{R}^D$ and $y \in \{0, 1\}^L$, our goal is to find a mapping from x to y. N, D, L are the number of samples, feature length and label set size respectively. The binary coding indicates the labels associated with the sample x. Labels are correlated with each other.

Preliminaries

Gaussian Mixture VAE A standard VAE [229] pulls together the posterior distribution and a parameter-free isotropic Gaussian prior. Two losses are optimized together in training: KL-divergence from the prior to the posterior, and the distance between the reconstructed targets and the real targets. One weakness of this formulation is the unimodality of its latent space, inhibiting the learning of more complex representations. Another concern is over-regularization: if the posterior is exactly the same as the prior, the learnt representations would be uninformative of the inputs. Numerous works extend the prior to be more complex [230, 231, 224]. In our work, we adopt the Gausian mixture prior. The probability density can be depicted as $p(z) = \frac{1}{k}\sum_{i=1}^{k} \mathcal{N}(z|\mu_i, \sigma_i^2)$ where i is the cluster index of k Gaussian clusters with mean μ_i and covariance σ_i^2 [232, 233]. Our intuition is

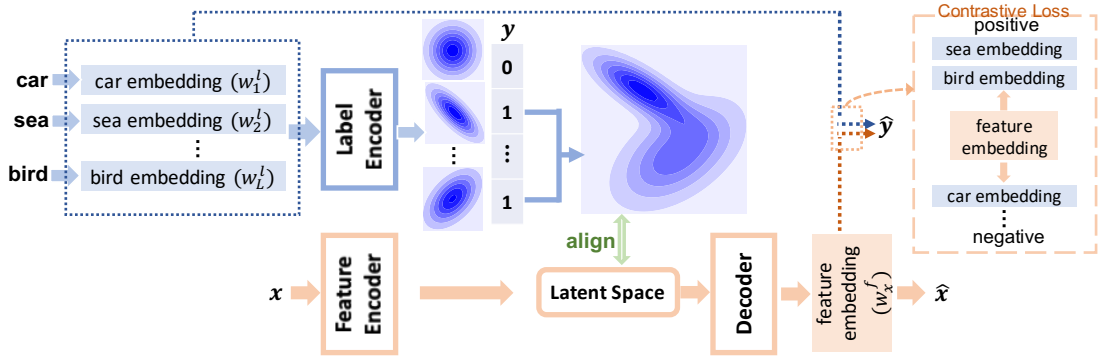


Figure 2.18: The full pipeline of C-GMVAE. Every label category is mapped to a learnable embedding first. The label encoder transforms each embedding w_i^l to a multivariate Gaussian latent space. The sample's associated label set selects the positive latent spaces and forms a Gaussian mixture prior. Each feature is also mapped to a latent space through a feature encoder. The posterior is aligned with the prior via KL-divergence. The decoder takes in a sample from the latent space and produces a feature embedding w_x^f. A contrastive loss is designed to pull together the feature embedding and the positive label embeddings, while separating the feature embedding from the negative label embeddings. Prediction $\hat{y}$ is generated by passing the inner products between the feature embedding w_x^f and the label embeddings w_i^l to the sigmoid functions. A sample with label set {sea, bird} is shown here.

that each label embedding could correlate to a Gaussian subspace. Given a label set, the mixture of the positive Gaussian subspaces forms a unique multimodal prior distribution. The label embeddings also receive the gradients from the contrastive loss and thus the contrastive learning is combined with latent space construction. Our formulation is also related to MVAE [225, 234] which adopts the idea of product-of-experts.

Contrastive Learning We describe how to use contrastive learning to capture the correlations (i.e., feature-label and label-label correlations). Contrastive learning [123, 235, 236] is a novel learning style. The core idea is simple: given an anchor sample, it should be close to similar samples (positive) and far from dissimilar samples (negative) in some learnt embedding space. It differs from triplet loss in the number of negative samples and the loss estimation method. Contrastive loss is largely motivated by noise contrastive estimation (NCE) [133] and its form is

generalizable. The raw contrastive loss formulation only considers the instance-level invariance (multiple views of one instance), but with label information, we can learn category-level invariance (multiple instances per class/category) [237].

In the multi-label scenario, one can regard the feature embedding as the anchor sample, positive label embeddings as the positive samples and negative label embeddings as the negative samples. The formulation can fit the contrastive learning framework naturally and is one of our major contributions. Compared to the pairwise ranking loss which focuses on the final logits, contrastive loss is defined on the learnt embeddings and thus becomes more expressive. Contrastive loss also includes more samples in estimating the NCE and therefore outperforms the triplet loss. In Appendix A.1.1, we show triplet loss is actually a special case of our contrastive loss.

C-GMVAE

C-GMVAE inherits the general VAE framework, but with a learnable Gaussian mixture (GM) prior. During training, each sample's label set activates and mixes the positive Gaussian subspaces to derive the prior. Contrastive learning is applied to boost the embedding learning, using a contrastive loss between the feature and label embeddings. Figure 2.18 provides a full illustration, and the following subsections will elaborate on the details.

Gaussian Mixture Latent Space Given a sample (x, y) where feature $x \in \mathbb{R}^D$ and label $y \in \{0, 1\}^L$, many previous works take y as the input and transform it to a dense representation through a fully-connected layer [221, 222]. This layer essentially maps each label category to an embedding and sums up all the

embeddings using label y as weights (0 or 1). The summed embedding is fed into the label encoder to produce a probabilistic latent space.

In C-GMVAE, however, we directly map each label embedding $w_i^l \in \mathbb{R}^E$ of label class i to an individual latent Gaussian distribution $\mathcal{N}(\mu_i, diag(\sigma_i^2))$, where $\mu_i \in \mathbb{R}^d, \sigma_i^2 \in \mathbb{R}^d$, and μ_i, σ_i^2 are derived from w_i^l through the NN-based label encoder. The randomly initialized embeddings w_i^l are learnable during the training process, similar to [58], and they share the same label encoder. In Figure 2.18, the label categories *car, sea,..., bird* are transformed to embeddings first. Embeddings are then passed directly to label encoder rather than summed up. Each label category (e.g., *car*) corresponds to a unimodal Gaussian in the latent space. y activates "positive" Gaussians ($y_i = 1$) and forms a Gaussian mixture subspace. Given a random variable $z \in \mathbb{R}^d$, the probability density function (PDF) in the subspace is defined as

$$p_\psi(z|y) = \frac{1}{\sum_i y_i} \sum_{i=1}^{L} \mathbb{1}\{y_i = 1\} \mathcal{N}(z|\mu_i, diag(\sigma_i^2)) \tag{2.18}$$

where $\mathbb{1}(\cdot)$ is the indicator function and the label encoder is parameterized by ψ (NN). In Figure 2.18, y activates *sea* and *bird*, then we have

$$\begin{aligned} p_\psi(z|y) = \frac{1}{2}(&\mathcal{N}(z|\mu_{sea}, diag(\sigma_{sea}^2)) + \\ &\mathcal{N}(z|\mu_{bird}, diag(\sigma_{bird}^2))) \end{aligned} \tag{2.19}$$

Most VAE-based frameworks optimize over an evidence lower bound (ELBO) [200]:

$$\begin{aligned} \text{ELBO} = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \\ D_{KL}[q_\phi(z|x) \| p(z)] \end{aligned} \tag{2.20}$$

The feature encoder is parameterized by ϕ (NN). One pitfall of this objective is owing to the minimization of the KL-divergence. If the divergence between

the posterior $q_\phi(z|x)$ and the prior $p_\psi(z)$ vanishes, the learnt latent codes would become non-informative. This is called posterior collapse. Many recent works suggest learnable priors [238] or more sophisticated priors [239] to avoid this issue, and we adopt these ideas in our design of the prior. Compared to a standard VAE, our prior is informative, learnable and multimodal.

We form a standard posterior in our model and match it with the prior. However, unlike vanilla VAE, we cannot analytically compute the KL term. Instead, we use the following estimation:

$$\begin{aligned}\mathcal{L}_{KL} &\approx \log q_\phi(z_0|x) - \log p_\psi(z_0|y) \\ &= \log \mathcal{N}(z_0|\mu_\phi(x), diag(\sigma^2_\phi(x))) - \\ &\quad \log \frac{1}{\sum_i y_i} \sum_{i=1}^{L} \mathbb{1}\{y_i = 1\} \mathcal{N}(z_0|\mu_i, diag(\sigma_i^2))\end{aligned} \tag{2.21}$$

where $z_0 \sim q_\phi(z|x)$ denotes a single latent sample. Our formulation follows the design of [232], which has been shown to outperform the formulation in [224].

The reconstruction loss is a standard negative log-likelihood with decoder parameters θ,

$$\mathcal{L}_{recon} = - E_{q_\phi(z|x)}[\log p_\theta(x|z)] \tag{2.22}$$

Contrastive Learning Module

The decoder function $f^d_\theta(\cdot)$ decodes the sample from the latent space to a feature embedding $w^f_x \in \mathbb{R}^E$. We learn w^f_x together with label embeddings $\{w^l_i\}^L_{i=1}$. The objective function includes both contrastive loss and cross-entropy loss terms.

Prior works explicitly capture the label-label interactions with GNNs or covariance matrices, which impose the structure *a priori* and might not be the best

modeling approach. Our contrastive module instead captures the correlations in a data-driven manner. For example, if in most of the samples, "beach" and "sunshine" appear together, the contrastive learning will implicitly pull their embeddings together (see the derivation in Appendix A.1.2). In other words, if two labels do co-appear often, their label embeddings would become similar (Figure 2.19). On the other hand, if they never co-occur or only co-appear occasionally, their connections are not significant and our model will not optimize for their similarity.

Original contrastive learning [123] augments inputs and learns the instance-level invariance, but it may not generalize to the category-level invariance. In the supervised setting, however, the learning can benefit from the labels and discover the category-level invariance [236]. Let $A \equiv \{1...L\}$. We define $P(y) \equiv \{i \in A : y_i = 1\}$ for sample (x, y). Suppose we have a batch of samples, $\mathcal{B}$, the contrastive loss can be written as

$$\mathcal{L}_{CL} = \frac{1}{|\mathcal{B}|} \sum_{(x,y)\in\mathcal{B}} \frac{1}{|P(y)|} \sum_{p\in P(y)} -\log \frac{\text{sim}(w_x^f, w_p^l)}{\sum_{t\in A} \text{sim}(w_x^f, w_t^l)} \tag{2.23}$$

Here, $\text{sim}(\cdot)$ is a function measuring the similarity between two embeddings, and w_x^f, w_i^l denote the feature and label embeddings respectively. Eq. 2.23 is built on top of NCE [133], and the equation is equivalent to a categorical cross-entropy of correctly predicting positive labels. The choice of $\text{sim}(\cdot)$ can be a log-bilinear function [123], or a more complicated neural metric function [235]. In our experiments, we find it is simple and effective to take $\text{sim}(w_1, w_2) = \exp(w_1 \cdot w_2/\tau)$ where $\cdot$ means inner product and τ is a temperature parameter controlling the scale of the inner product.

In the single-label scenarios like SupCon [236], if one class is positive, all other classes are contrastive to it. However, in MLC, if "beach" is positive in the

label set while "sea" is not for one particular sample, we cannot say these two classes are contrastive. Their correlations should be captured implicitly by all the samples. Therefore, we do not enforce contrastive relations between labels and thus preserve the label correlations. Instead, we choose the feature embedding to be the anchor and label embeddings to be the positive and negative samples. If two label embeddings co-appear often as positive samples, they would implicitly become similar (see Figure 2.19). Eq. 2.23 saves the effort of manually configuring the positive and negative samples, and is totally data-driven. The number of positive or negative samples could be greater than one, depending on the label set. Though L limits the max samples we can have, this formulation has already used many more samples compared to triplet loss, and we will show in experiments that this formulation is very effective.

The triplet loss often used in multi-label learning [240] can be seen as a special case of Eq. 2.23 with only one positive and one negative. Furthermore, one desired property of embedding learning is that when a good positive embedding is already close enough to our anchor embedding, it contributes less to the gradients, while poorly learnt embeddings contribute more to improve the model performance. In Appendix A.1, we also show that the contrastive loss can implicitly achieve this goal and a full derivation of the gradients is provided.

Our objective function also includes a supervised cross-entropy loss term to further facilitate the training. With the label embeddings w_i^l and the feature embedding w_x^f, the cross entropy loss for each (x, y) is given by

$$\mathcal{L}_{CE} = \sum_{i=1}^{L} y_i \log s(w_x^f w_i^l) + (1 - y_i) \log(1 - s(w_x^f w_i^l)) \tag{2.24}$$

where $s(\cdot)$ is the sigmoid function. In self-supervised learning, the contrastive loss typically helps the pretraining stage and the learnt representations are applied

to downstream tasks. In the supervised setting, though some models [236] stick to the two-stage training process where the model is trained with contrastive loss in the first stage and with cross-entropy loss in the second stage, we did not observe its superiority over the one-stage scheme in our MLC scenario. This is partly because we also learn a latent space that is closely connected to label embeddings. We train the model with an objective function incorporating all the losses. A joint training strategy reconciles different modules. We show in the experiments that the learnt embeddings are semantically meaningful and can reveal the label correlations.

Objective Function The final objective function to minimize is simply the summation of different losses,

$$\mathcal{L} = \mathcal{L}_{KL} + \mathcal{L}_{recon} + \alpha\mathcal{L}_{CL} - \beta\mathcal{L}_{CE} \tag{2.25}$$

where α, β are trade-off weights. The model is trained with Adam [186]. Our model is optimized with $\mathcal{L}$ and will be tested on five different metrics. This is different from the methods that only optimize and test for specific metrics [241, 242].

Prediction

During the testing phase, the input sample x will be passed to the feature encoder and decoder to obtain its embedding w_x^f. Label embeddings w_i^l are fixed in testing. The inner products between w_x^f and w_i^l will be passed through a sigmoid function to obtain the prediction probability for each class i.

Insights behind C-GMVAE

C2AE and MPVAE have shown the importance of learning a shared latent space for both features and labels. These methods share the same high-level insight similar to a teacher-student regime: we map labels (teacher) to a latent space with some certain structure, which preserves the label information and is easier to decode back to labels. Then the features (student) are expected to be mapped to this latent space to facilitate the label prediction. Two general concerns exist for these methods: 1) the unimodal Gaussian space previously used is too restrictive to impose sophisticated structures on prior, and 2) they do not properly capture label correlations with embeddings. To address the first, we learn a modality for each label class to form a mixture latent space. For the second, we replace the commonly used ranking and triplet losses with contrastive loss since contrastive loss involves more samples than triplet loss and has a larger capacity than ranking loss.

2.3.3 Existing MLC Works

Learning a shared latent space for features and labels is a common and useful idea. For single-label prediction tasks, CADA-VAE [243] learns and aligns latent label and feature spaces through distribution alignment losses. Similar ideas can be seen in out-of-distribution detection as well [223]. In multi-label scenarios, methods adopting this idea typically have a similar module that directly maps the multi-hot labels to embeddings [221, 244, 222]. This is a rather difficult learning task. Suppose we have 30 label categories. There could be up to 2^{30} label sets. For probabilistic models like MPVAE, that means one latent label

space has to represent up to 2^{30} label combinations. In contrast, C-GMVAE learns per-category subspaces and forms a mixture prior distribution based on the observed samples' label sets.

Contrastive learning has become one of the most popular self-supervised learning techniques. It has also been applied to supervised learning tasks. SupCon [236] first demonstrated the effectiveness of supervised contrastive loss in image classification tasks. It was soon generalized to other domains like visual reasoning [245, 246]. Nevertheless, these methods depend on vision-specific augmentation techniques and attention mechanisms. Another related work is multi-label contrastive learning [247]. But the work does not deal with MLC. Instead, it extends contrastive learning to the identification of more than one positive sample, which resembles a multi-label scenario.

Some earlier works also attempted metric learning or triplet loss in MLC [248]. Triplet loss typically only takes one pair of positive and negative samples for one anchor, while contrastive loss uses many more negative and positive samples. Recent papers found that more samples can greatly boost performance [235, 237]. Though our contrastive module is constrained by the maximum number of label classes, the number of used samples has already surpassed other losses (e.g., triplet loss), and our observations reinforce that more samples help with the performance.

2.3.4 Experiments on Text and Ecology Data

We have various setups to validate the performance of C-GMVAE. First, we compare the example-F1, micro-F1 and macro-F1 scores, Hamming accuracies and

Metric	example-F1					
Dataset	*eBird*	*yeast*	*sider*	*reuters*	*bookmarks*	*delicious*
BR	0.365	0.630	0.766	0.733	0.171	0.174
MLKNN	0.510	0.618	0.738	0.703	0.213	0.259
HARAM	0.510	0.629	0.722	0.711	0.216	0.267
SLEEC	0.258	0.643	0.581	0.885	0.363	0.308
C2AE	0.501	0.614	0.768	0.818	0.309	0.326
LaMP	0.477	0.624	0.766	0.906	0.389	0.372
MPVAE	0.551	0.648	0.769	0.893	0.382	0.373
ASL	0.528	0.613	0.752	0.880	0.373	0.359
RBCC	0.503	0.605	0.733	0.857	-	-
C-GMVAE	**0.576**	**0.656**	**0.771**	**0.917**	**0.392**	**0.381**
std (±)	0.001	0.001	0.001	0.001	0.001	0.002

Metric	micro-F1					
Dataset	*eBird*	*yeast*	*sider*	*reuters*	*bookmarks*	*delicious*
BR	0.384	0.655	0.796	0.767	0.125	0.197
MLKNN	0.557	0.625	0.772	0.680	0.181	0.264
HARAM	0.573	0.635	0.754	0.695	0.230	0.273
SLEEC	0.412	0.653	0.697	0.845	0.300	0.333
C2AE	0.546	0.626	0.798	0.799	0.316	0.348
LaMP	0.517	0.641	0.797	0.886	0.373	0.386
MPVAE	0.593	0.655	0.800	0.881	0.375	0.393
ASL	0.580	0.637	0.795	0.869	0.354	0.387
RBCC	0.558	0.623	0.784	0.825	-	-
C-GMVAE	**0.633**	**0.665**	**0.803**	**0.890**	**0.377**	**0.403**
std (±)	0.001	0.002	0.000	0.001	0.001	0.002

Table 2.8: The example-F1 (ex-F1) and micro-F1 (mi-F1) scores of different methods on all datasets. C-GMVAE's numbers are averaged over 3 seeds. The standard deviation (std) is also shown. 0.000 means an std< 0.0005.

precision@1 of different methods. Second, we compare their performance when fewer training data are available. Third, an ablation study shows the importance of the introduced modules. Finally, we demonstrate the interpretability of the label embeddings on the *eBird* dataset.

Metric	macro-F1					
Dataset	*eBird*	*yeast*	*sider*	*reuters*	*bookmarks*	*delicious*
BR	0.116	0.373	0.588	0.137	0.038	0.066
MLKNN	0.338	0.472	0.667	0.066	0.041	0.053
HARAM	0.474	0.448	0.649	0.100	0.140	0.074
SLEEC	0.363	0.425	0.592	0.403	0.195	0.142
C2AE	0.426	0.427	0.667	0.363	0.232	0.102
LaMP	0.381	0.480	0.668	0.520	0.286	0.196
MPVAE	0.494	0.482	0.690	0.545	0.285	0.181
ASL	0.467	0.484	0.668	0.563	0.264	0.183
RBCC	0.443	0.480	0.654	0.503	-	-
C-GMVAE	**0.538**	**0.487**	**0.691**	**0.582**	**0.291**	**0.197**
std (±)	0.000	0.002	0.002	0.001	0.001	0.001

Metric	Hamming Accuracy					
Dataset	*eBird*	*yeast*	*sider*	*reuters*	*bookmarks*	*delicious*
BR	0.816	0.782	0.747	0.994	0.990	0.982
MLKNN	0.827	0.784	0.715	0.992	0.991	0.981
HARAM	0.819	0.744	0.650	0.905	0.990	0.981
SLEEC	0.816	0.782	0.675	0.996	0.989	0.982
C2AE	0.771	0.764	0.749	0.995	0.991	0.981
LaMP	0.811	0.786	0.751	0.997	0.992	0.982
MPVAE	0.829	0.792	0.755	0.997	0.991	0.982
ASL	0.831	0.796	0.759	0.997	0.991	0.982
RBCC	0.815	0.793	0.753	0.997	-	-
C-GMVAE	**0.847**	**0.796**	**0.767**	**0.997**	**0.992**	**0.983**
std (±)	0.001	0.002	0.003	0.000	0.000	0.000

Table 2.9: The macro-F1 (ma-F1) and Hamming accuracy (HA) scores of different methods on all datasets. C-GMVAE's numbers are averaged over 3 seeds.

Setup

For the main evaluation experiments, we use six datasets, including text datasets *reuters, bookmarks, delicious* [249, 250, 251], biology datasets *sider*, *yeast* [252, 253], and ecology dataset *eBird* [254] (see Tab. 2.11 for dataset statistics). Most of these features are sequence data and have be processed through sequence models (GRU) or vectorization. The feature pre-processing is standard following previ-

Dataset	*eBird*	*yeast*	*sider*	*reuters*	*bookmarks*	*delicious.*
BR	0.598	0.745	0.573	0.752	0.301	0.485
MLKNN	0.772	0.730	0.916	0.753	0.310	0.460
MLARAM	0.768	0.682	0.930	0.679	0.312	0.419
SLEEC	0.656	0.745	0.882	0.908	0.415	0.676
C2AE	0.753	0.749	0.923	0.845	0.407	0.609
LaMP	0.737	0.740	0.937	0.927	0.420	0.663
MPVAE	0.820	0.743	0.958	0.930	0.437	0.696
ASL	0.818	0.752	0.954	0.929	0.418	0.692
RBCC	0.805	0.745	0.942	0.913	-	-
C-GMVAE	**0.825**	**0.751**	**0.962**	**0.939**	**0.465**	**0.707**

Table 2.10: The precision@1 scores of different methods on all datasets.

Dataset	# Samples	# Labels	Mean Labels /Sample	Median Labels /Sample	Max Labels /Sample	Mean Samples /Label
eBird	41778	100	20.69	18	96	8322.95
bookmarks	87856	208	2.03	1	44	584.67
reuters	10789	90	1.23	1	15	106.50
sider	1427	27	15.3	16	26	731.07
yeast	2417	14	4.24	4	11	363.14
delicious	16105	983	19.06	20	25	250.15

Table 2.11: Dataset Statistics.

ous works [111, 222] and the datasets are public[2]. Each dataset is separated into training (80%), validation (10%) and testing (10%) splits. The datasets are also preprocessed to fit the input formats of different methods. We use mini-batch training with batch size 128. Each batch is randomly sampled from the dataset.

The evaluation metrics are three F1 scores, Hamming accuracy and precision@1. The evaluation process, model selection and preprocessing strictly follow previous works [255, 111, 222]. Most numbers are also directly quoted from the corresponding papers for comparison. Our method is compared against ASL [256], RBCC [257], MPVAE [222], LaMP [111], C2AE [221], SLEEC [220], HARAM

[2]http://mulan.sourceforge.net/datasets-mlc.html

	variations	eb-F1	mi-F1	ma-F1
ebird	uni-Gaussian	0.545	0.583	0.490
	GM only	0.561	0.603	0.511
	contrastive only	0.558	0.594	0.515
	GM+contrastive	0.576	0.633	0.538

Table 2.12: Ablation study on the contrastive learning module and the Gaussian mixture module. Note that both modules are contributions of this work. As shown in the table, GM consistently improves performance. The contrastive module can also further boost the performance.

	method (data %)	HA	ex-F1	mi-F1	ma-F1
ebird	MPVAE (100%)	0.829	0.551	0.593	0.494
	C-GMVAE (50%)	0.842	0.557	0.615	0.521

Table 2.13: Comparisons between MPVAE and C-GMVAE using 100% and 50% respectively.

[258], MLKNN [219], and BR [259].

ASL introduces asymmetric loss, a variant of BCE and focal loss for MLC, and requires tuning of focusing parameters. RBCC is based on a Bayesian network, which requires structure learning to derive a directed acyclic graph (DAG) first. MPVAE is a novel method which learns and aligns the probabilistic feature and label subspaces. Label correlations are captured by a multivariate probit module. LaMP adopts attention-based neural message passing to handle the label correlations, which is a neural extension of previous CRF-based methods. C2AE was one of the first papers to use NNs to learn and align latent spaces. C2AE imposes a canonical correlation analysis (CCA) constraint on the latent space. SLEEC explores the low-rank assumption in MLC, to reduce the effective number of labels. Other deep methods make similar low-rank assumptions. HARAM was one of the first methods to introduced NNs to MLC. MLKNN is a classic MLC method using k-nearest neighbors (KNN). It finds nearest examples to a test sample and adopts Bayesian inference to select assigned labels. Lastly,

module in C-GMVAE	*eBird*	*yeast*	*sider*
Covariance	0.601	0.650	0.787
GNN	0.599	0.655	0.801
contrastive	**0.633**	**0.665**	**0.803**

Table 2.14: mi-F1 performance after replacing our contrastive module with a GNN or a covariance matrix.

binary relevance (BR) is one of the most intuitive solutions for MLC, which decomposes the multi-label scenario into independent binary prediction tasks.

Metrics

We evaluate our method trained with objective Eq. 2.25 on several commonly used multi-label metrics. Suppose the ground-truth label is y and the predicted label is $\hat{y}$. We denote true positives, false positives, false negatives by tp_j, fp_j, fn_j respectively for the j-th of L label categories. (i) HA: $\frac{1}{L}\sum_{j=1}^{L}\mathbb{1}[y_j = \hat{y}_j]$ (ii) example-F1: $\frac{2\sum_{j=1}^{L} y_i\hat{y}_i}{\sum_{j=1}^{L} y_i + \sum_{j=1}^{L}\hat{y}_i}$ (iii) micro-F1: $\frac{\sum_{j=1}^{L} tp_j}{\sum_{j=1}^{L} 2tp_j + fp_j + fn_j}$ (iv) macro-F1: $\frac{1}{L}\sum_{j=1}^{L}\frac{2tp_j}{2tp_j + fp_j + fn_j}$.

Furthermore, precision@1 is the proportion of correctly predicted labels in the top-1 predictions.

Architecture and Hyperparameters

As we state in the introduction, we do not require very sophisticated neural architectures in C-GMVAE. All the neural layers are based on MLP or GRU. We set $\alpha = 1, \beta = 0.5, E = 2048$ by default. Grid search is applied to find the best learning rate, dropout ratio, and weight decay ratio for each dataset. We use one Nvidia V100 GPU for all experiments.

Evaluations

Full supervision In the full supervision scenario, which is commonly adopted by the methods we compare against, we evaluate five metrics: example-F1 (ex-F1), micro-F1 (mi-F1), macro-F1 (ma-F1), Hamming accuracy (HA) and precision@1. The ex-F1 score is the averaged F1-score over all the samples. The mi-F1 score measures the aggregated contributions of all classes. The ma-F1 treats each class equally and takes the class-wise average. HA counts the correctly predicted labels regardless of samples or classes.

Tab. 2.8, 2.9 and 2.10 present the performance of all the methods w.r.t. the metrics. C-GMVAE outperforms the existing state-of-the-art methods on all the datasets. The best numbers are marked in bold. All the numbers for C-GMVAE are averaged over 3 seeds for stability and the standard deviations are included in the table. C-GMVAE outperforms other methods consistently.

Ablation study To demonstrate the strength of C-GMVAE, we compare it with a unimodal Gaussian latent model, a Gaussian mixture only latent model (without contrastive module), and a contrastive learning only model (without the KL divergence term) in Tab. 2.12. Our C-GMVAE (GM+contrastive) consistently outperforms other models by a large margin. For instance, on ma-F1, C-GMVAE improves over the unimodal Gaussian model by 7%. In Tab. 2.14, we show that the contrastive module outperforms both the GNN and covariance matrix modules.

Training on fewer data Contrastive learning learns contrastive views and thus requires less information compared to generative learning, which demands a

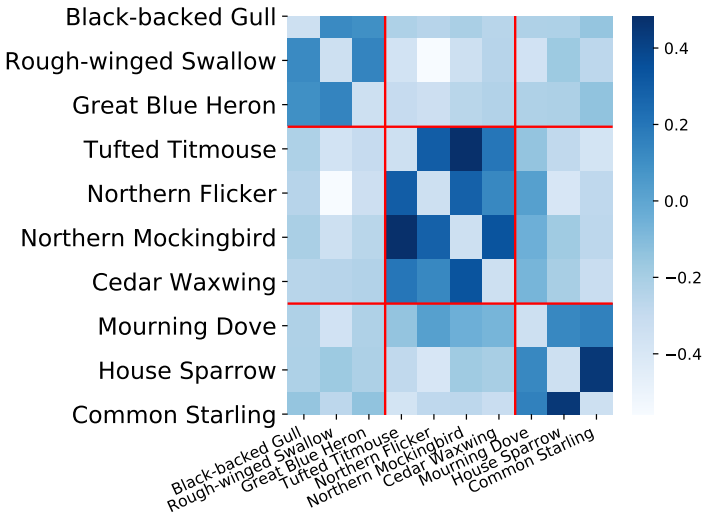


Figure 2.19: Label-label inner-products from C-GMVAE. C-GMVAE demonstrates sharp and meaningful inner-products.

more complete representation for reconstruction. Contrastive learning has the potential to discover the intrinsic structure present in the data, and therefore is widely used in self-supervised learning because it generalizes well. We observe this with C-GMVAE as well. To demonstrate this, we shrink the size of training data by 50% or 90% and train methods on them. Surprisingly, we find C-GMVAE can often match the performance of other methods with only 50% of the training data. Tab. 2.13 compares MPVAE trained on all data and C-GMVAE trained on 50% of the data. Their performance is approximately the same. We further compare several major state-of-the-art methods including ours, all trained on the same **randomly** selected 10%, 50% and 100% of the data, and show their performance over C2AE.

We provide relative performances of several major state-of-the-art methods including ours to C2AE, on HA, ex-F1, mi-F1, ma-F1 scores. All methods are trained on 10% or 50% of the data, including C2AE. The compared results have the same amount of data for training and thus the comparison is fair. Figure 2.20

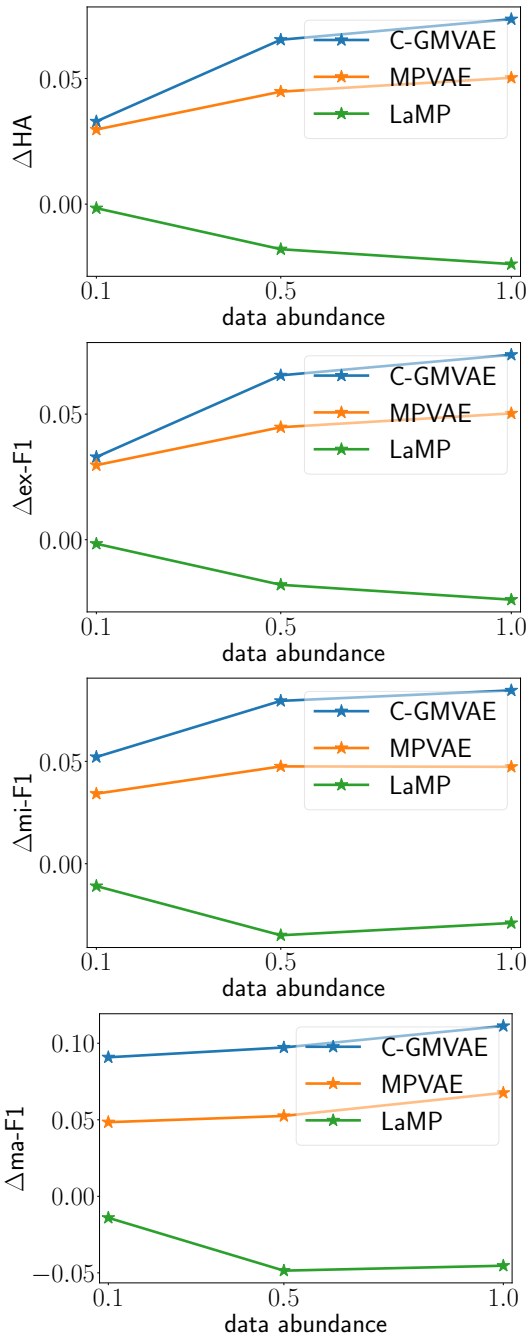


Figure 2.20: Relative performances w.r.t. HA, ex-F1, mi-F1 and ma-F1 on *ebird* dataset.

show the relative performance of various state-of-the-art methods over C2AE, on *eBird* dataset.

Interpretability Our work is also motivated by ecological applications [114], where it is important to understand species interactions. Figure 2.19 shows a map of inner-product weights of label embeddings on the eBird dataset. The bird species on the x-axis and the y-axis are the same. The first 3 bird species are water birds. The following 4 bird species are forest birds. The last 3 bird species are residential birds. Darker colors indicate more similar birds. We subtract the diagonal to exclude the self correlation. The heatmap matrix clearly forms three blocks on the diagonal. The first block contains Black-backed Gull, Rough-winged Swallow and Great Blue Heron. These three birds are water birds living near sea or lake. The second block has Tufted Titmouse, Northern Flicker, Northern Mockingbird, and Cedar Waxwing. These birds typically live in the forest with a lot of trees. The remaining birds are commonly seen residential birds, Mourning Dove, House Sparrow and Common Starling. They live inside or near human residences. Since human activities are wide-spread, the distribution of these birds is therefore quite broad. For example, the Mourning Dove is also correlated with forest birds in Figure 2.19. But one can observe that for each group of birds, their intra-group correlations are always stronger than inter-group correlations. Therefore, the learnt embeddings do encompass semantic meanings. The derived correlations could also help the study of wildlife protection [260].

2.3.5 Discussion

In this work, we introduce the contrastive learning boosted Gaussian mixture variational autoencoder (C-GMVAE), a novel method for multi-label prediction tasks. C-GMVAE combines the learning of Gaussian mixture latent spaces and the

contrastive learning of feature and label embeddings. Not only does C-GMVAE achieve the state-of-the-art performance, it also provides insights into semi-supervised learning and model interpretability. Interesting future directions include the exploration of various contrastive learning mechanisms, model architecture improvements, and other latent space structures.

CHAPTER 3

VERSATILE SELF-SUPVERSION IN SEQUENCE LEARNING

In this chapter, I will show the versatility of self-supervision in sequence learning. More specifically, I will demonstrate self-supervision for pretraining, disentanglement and even supervised training. First, the mutual information based self-supervision can boost the pretraining of feature encoder. The learnt encoder can be finetuned for various downstream tasks. Second, self-supervision can help disentangle the static and dynamic factors of sequence data without any supervision. Third, I will demonstrate self-supervision can be jointly trained with supervision to further promote the model performance. This chapter is based on the following publications: Bai et al. (2021) [261], Bai et al. (2021) [262], Bai et al. (2022) [263].

3.1 Self-supervision for Pretraining: Deep Autoencoding Predictive Components for Sequence Representation Learning

3.1.1 Introduction

Self-supervised representation learning methods aim at learning useful and general representations from large amounts of unlabeled data, which can reduce sample complexity for downstream supervised learning. These methods have been widely applied to various domains such as computer vision [123, 264, 235, 265], natural language processing [266, 267, 31], and speech processing [57, 268, 269, 237, 37]. In the case of sequence data, representation learning may force the model to recover the underlying dynamics from the raw data, so that the

learnt representations remove irrelevant variability in the inputs, embed rich context information and become predictive of future states. The effectiveness of the representations depends on the self-supervised task which injects inductive bias into learning. The design of self-supervision has become an active research area.

One notable approach for self-supervised learning is based on maximizing mutual information between the learnt representations and inputs. The most commonly used estimate of mutual information is based on contrastive learning. A prominant example of this approach is CPC [123], where the representation of each time step is trained to distinguish between positive samples which are inputs from the near future, and negative samples which are inputs from distant future or other sequences. The performance of contrastive learning heavily relies on the nontrivial selection of positive and negative samples, which lacks a universal principle across different scenarios [270, 235, 271]. Recent works suspected that the mutual information lower bound estimate used by contrastive learning might be loose and may not be the sole reason for its success [272, 273].

In this work, we leverage an estimate of information specific to sequence data, known as *predictive information* (PI, 134), which measures the mutual information between the past and future windows in the latent space. The estimate is exact if the past and future windows have a joint Gaussian distribution, and is shown by prior work to be a good proxy for the true predictive information in practice [274]. We can thus compute the estimate with sample windows of the latent sequence (without sampling negative examples), and obtain a well-defined objective for learning the encoder for latent representations. However, simply using the mutual information as the learning objective may lead to degenerate

representations, as PI emphasizes simple structures in the latent space and a powerful encoder could achieve this at the cost of ignoring information between latent representations and input features. To this end, we adopt a masked reconstruction task to enforce the latent representations to be informative of the observations as well. Similar to [237], we mask input dimensions as well as time segments of the inputs, and use a decoder to reconstruct the masked portion from the learnt representations; we also introduce variants of this approach to achieve superior performance.

Our method, Deep Autoencoding Predictive Components (DAPC), is designed to capture the above intuitions. From a variational inference perspective, DAPC also has a natural probabilistic interpretation. We demonstrate DAPC on both synthetic and real datasets of different sizes from various domains. Experimental results show that DAPC can recover meaningful low dimensional dynamics from high dimensional noisy and nonlinear systems, extract predictive features for forecasting tasks, and obtain state-of-the-art accuracies for Automatic Speech Recognition (ASR) with a much lower cost, by pretraining encoders that are later finetuned with a limited amount of labeled data.

3.1.2 Deep Autoencoding Predictive Components

The main intuition behind Deep Autoencoding Predictive Components is to maximize the predictive information of latent representation sequence. To ensure the learning process is tractable and non-degenerate, we make a Gaussian assumption and regularize the learning with masked reconstruction. In the following subsections, we elaborate on how we estimate the predictive information and

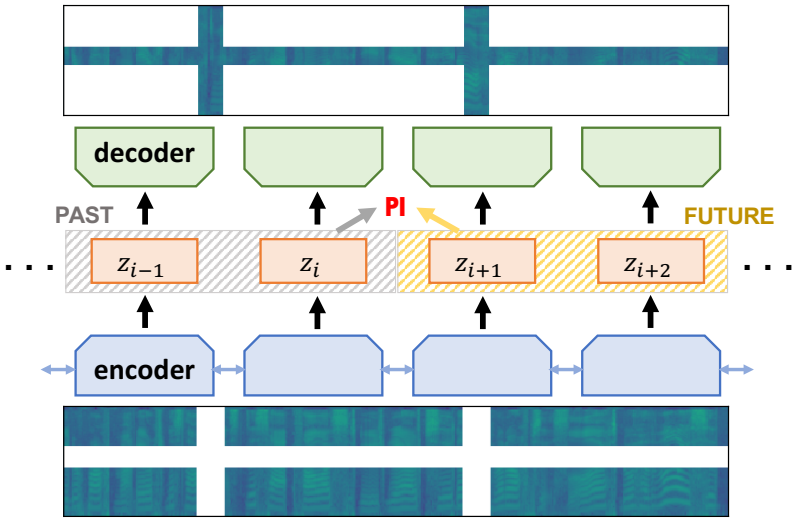


Figure 3.1: The overall framework of DAPC.

how we design the masked reconstruction task. A probabilistic interpretation of DAPC is also provided to show the connection to deep generative models.

Predictive Information

Given a sequence of observations $X = \{x_1, x_2, ...\}$ where $x_i \in \mathbb{R}^n$, we extract the corresponding latent sequence $Z = \{z_1, z_2, ...\}$ where $z_i \in \mathbb{R}^d$ with an encoder function $e(X)$, e.g., recurrent neural nets or transformers [28].[1] Let $T > 0$ be a fixed window size, and denote $Z_t^{past} = \{z_{t-T+1}, ..., z_t\}$, $Z_t^{future} = \{z_{t+1}, ..., z_{t+T}\}$ for any time step t. The predictive information (PI) is defined as the mutual information (MI) between Z_t^{past} and Z_t^{future}:

$$MI(Z_t^{past}, Z_t^{future}) = H(Z_t^{past}) + H(Z_t^{future}) - H(Z_t^{past}, Z_t^{future}) \tag{3.1}$$

where H is the entropy function. Intuitively, PI measures how much knowing Z_t^{past} reduces the uncertainty about Z_t^{future} (and vice versa). PI reaches its mini-

[1] In this work, the latent sequence has the same length as the input sequence, but this is not an restriction; one can have a different time resolution for the latent sequence, using sub-sampling strategies such as that of [275].

mum value 0 if Z_t^{past} and Z_t^{future} are independent, and it is maximized if Z_t^{future} is a deterministic function of Z_t^{past}. Different from the MI estimate used by contrastive learning, which measures the MI between representation at each *single* time step and its future inputs, predictive information measures the MI between two windows of T time steps collectively. The window size T used in PI estimation reflects the time resolution for which the time series is more or less stationary.

PI was designed as a general measure of the complexity of underlying dynamics which persists for a relatively long period of time [276]. Furthermore, PI is aware of temporal structures: different dynamics could lead PI to converge or diverge even if they look similar. These virtues of PI contribute to the versatility of this measure. The use of PI beyond a static complexity measure (and as a learning objective) is done only recently in machine learning by [274], which proposes to learn a linear dimensionality reduction method named Dynamical Component Analysis (DCA) to maximize the PI of the projected latent sequence.

One approach for estimating PI is through estimating the joint density $P(Z_t^{past}, Z_t^{future})$, which can be done by density estimation methods such as k-NN and binning [277, 278]. However, such estimates heavily rely on hyperparameters, and it is more challenging to come up with differentiable objectives based on them that are compatible with deep learning frameworks. Our approach for estimating PI is the same as that of DCA. Assume that every $2T$ consecutive time steps $\{z_{t-T+1}, ..., z_t, ..., z_{t+T}\}$ in the latent space form a stationary, multivariate Gaussian distribution. $\Sigma_{2T}(Z)$ is used to denote the covariance of the distribution, and similarly $\Sigma_T(Z)$ the covariance of T consecutive latent steps. Under the stationarity assumption, $H(Z_t^{past})$ remains the same for any t so we can omit the subscript t, and $H(Z^{past})$ is equal to $H(Z^{future})$ as well. Using the fact that

$H(Z^{past}) = \frac{1}{2}\ln(2\pi e)^{dT}|\Sigma_T(Z)|$, PI for the time series z reduces to

$$I_T = MI(Z^{past}, Z^{future}) = \ln|\Sigma_T(Z)| - \frac{1}{2}\ln|\Sigma_{2T}(Z)|. \tag{3.2}$$

Detailed derivations can be found in Appendix B.1.1. It is then straightforward to collect samples of the consecutive $2T$-length windows and compute the sample covariance matrix for estimating $\Sigma_{2T}(Z)$. An empirical estimate of $\Sigma_T(Z)$ corresponds to the upper left sub-matrix of $\Sigma_{2T}(Z)$. Recall that, under the Gaussian assumption, the conditional distribution $P(Z^{future}|Z^{past})$ is again Gaussian, whose mean is a linear transformation of Z^{past}. Maximizing I_T has the effect of minimizing the entropy of this conditional Gaussian, and thus reducing the uncertainty of future given past.

Though our estimation formula for PI is exact only under the Gaussian assumption, it was observed by [274] that the Gaussian-based estimate is positively correlated with a computationally intensive estimate based on non-parametric density estimate, and thus a good proxy for the full estimate. We make the same weak assumption, so that optimizing the Gaussian-based estimate improves the true PI. Our empirical results show that representations learnt with the Gaussian PI have strong predictive power of future (see Sec 3.1.4). Furthermore, we find that a probabilistic version of DAPC (described in Sec 3.1.2) which models (Z^{past}, Z^{future}) with a Gaussian distribution achieves similar performance as this deterministic version (with the Gaussian assumption).

We now describe two additional useful techniques that we develop for PI-based learning.

Multi-scale PI One convenient byproduct of this formulation and estimation for I_T is to reuse $\Sigma_{2T}(Z)$ for estimating $I_{T/2}$, $I_{T/4}$ and so on, as long as the window

size is greater than 1. Since the upper left sub-matrix of $\Sigma_{2T}(Z)$ approximates $\Sigma_T(Z)$, we can extract $\Sigma_T(Z)$ from $\Sigma_{2T}(Z)$ without any extra computation, and similarly for $\Sigma_{T/2}(Z)$. We will show that multi-scale PI, which linearly combines PI at different time scales, boosts the representation quality in ASR pretraining.

Orthogonality penalty Observe that the PI estimate in (3.2) is invariant to invertible linear transformations in the latent space. To remove this degree of freedom, we add the penalty to encourage latent representations to have identity covariance, so that each of the d latent dimensions will have unit scale and different dimensions are linearly independent and thus individually useful. This penalty is similar to the constraint enforced by deep canonical correlation analysis [279], which was found to be useful in representation learning [280].

Masked Reconstruction and Its Shifted Variation

The PI objective alone can potentially lead to a degenerate latent space, when the mapping from input sequence to latent sequence is very powerful, as the latent representations can be organized in a way that increases our PI estimate at the cost of losing useful structure from the input. This is also observed empirically in our experiments (see Sec 3.1.4). To regularize PI-based learning, one simple idea is to force the learnt latent representations to be informative of the corresponding input observations. For this purpose, we augment PI-based learning with a masked reconstruction task.

Masked reconstruction was first proposed in BERT [267], where the input text is fed to a model with a portion of tokens masked, and the task is to reconstruct the masked portion. [237] extended the idea to continuous vector sequence data

(spectrograms). The authors found that randomly masking input dimensions throughout the sequence yields further performance gain, compared to masking only consecutive time steps. We adopt their formulation in DAPC to handle continuous time series data.

Given an input sequence X of length L and dimensionality n, we randomly generate a binary mask $M \in R^{n \times L}$, where $M_{i,j} = 0$ indicates $X_{i,j}$ is masked with value 0 and $M_{i,j} = 1$ indicates $X_{i,j}$ is kept the same. We feed the masked inputs to the encoder $e(\cdot)$ to extract representations (in R^d) for each time step, and use a feed-forward network $g(\cdot)$ to reconstruct the masked input observations. $e(\cdot)$ and $g(\cdot)$ are trained jointly. The masked reconstruction objective can be defined as

$$R = \|(1 - M) \odot (X - g(e(X \odot M)))\|_{fro}^2. \tag{3.3}$$

Figure 3.1 gives an illustration for the masked spectrogram data. We randomly generate n_T time masks each with width up to w_T, and similarly n_F frequency masks each with width up to w_F. In our experiments, we observe that input dimension masking makes the reconstruction task more challenging and yields higher representation quality. Therefore, this strategy is useful for general time series data beyond audio.

We introduce one more improvement to masked reconstruction. Standard masked reconstruction recovers the masked inputs for the same time step. Inspired by the success of Autoregressive Predictive Coding [269], we introduce a shifted variation of masked reconstruction, in which the latent state z_i is decoded to reconstruct a future frame x_{i+s} (than x_i). Formally, the shifted masked reconstruction loss R_s is defined as

$$R_s = \|(1 - M^{\rightarrow s}) \odot (X^{\rightarrow s} - g(e(X \odot M)))\|_{fro}^2 \tag{3.4}$$

where $\rightarrow s$ indicates right-shifting s time frames while the input dimensions remain unchanged. When $s = 0$, R_s reduces to the standard masked reconstruction objective, and in the ASR experiments we find that a nonzero s value helps. We ensure no information leakage by enforcing that the portion to be reconstructed is never presented in the inputs. As indicated by [281], predicting a future frame encourages more global structure and avoids the simple inference from local smoothness in domains like speech, and therefore helps the representation learning.

To sum up, our overall loss function is defined as the combination of the losses described above:

$$\min_{e,g} \; L_{s,T}(X) = -\left(I_T + \alpha I_{T/2}\right) + \beta R_s + \gamma R_{ortho} \tag{3.5}$$

where α, β, γ are tradeoff weights and $R_{ortho} = ||\Sigma_1 - I_d||^2_{fro}$ is the orthonormality penalty discussed in Sec. 3.1.2, with $\Sigma_1 \in R^{d \times d}$ corresponding to the top left sub-matrix of Σ_{2T} estimated from the latent sequence $Z = e(X \odot M)$. The whole framework of DAPC is illustrated in Figure 3.1.

A Probabilistic Interpretation of DAPC

We now discuss a probabilistic interpretation of DAPC in the Variational AutoEncoder (VAE) framework [282]. Let $X = (X_p, X_f)$ and $Z = (Z_p, Z_f)$, where the subscripts p and f denote *past* and *future* respectively. Consider a generative model, where the prior distribution is $p(Z_p, Z_f) \sim \mathcal{N}(\mathbf{0}, \Sigma)$. One can write down explicitly $p(Z_f|Z_p)$, which is a Gaussian with

$$\mu_{f|p} = \Sigma_{f,p}\Sigma^{-1}_{p,p}Z_p, \qquad \Sigma_{f|p} = \Sigma_{ff} - \Sigma_{f,p}\Sigma^{-1}_{p,p}\Sigma_{p,f}.$$

The linear dynamics in latent space are completely defined by the covariance matrix Σ. Large predictive information implies low conditional entropy $H(Z_f|Z_p)$.

Let (Z_p, Z_f) generate (X_p, X_f) with a stochastic decoder $g(X|Z)$. We only observe X and would like to infer the latent Z by maximizing the marginal likelihood of X. Taking a VAE approach, we parameterize a stochastic encoder $e(Z|X)$ for the approximate posterior, and derive a lower bound for the maximum likelihood objective. Different from standard VAE, here we would not want to parameterize the prior to be a simple Gaussian, in which case the Z_p and Z_f are independent and have zero mutual information. Instead we encourage the additional structure of high predictive information for the prior. This gives us an overall objective as follows:

$$\min_{\Sigma,e,g} \int \hat{p}(X) \left\{ \int -e(Z|X) \log g(X|Z) dz + KL(e(Z|X)||p(Z)) \right\} dx - \eta I_T(\Sigma)$$

where $\hat{p}(X)$ is the empirical distribution over training data, the first term corresponds to the reconstruction loss, the second term measures the KL divergence between approximate posterior and the prior, and the last term is the PI defined in (3.2).

The challenge is how to parameterize the covariance Σ. We find that simply parameterizing it as a positive definite matrix, e.g., $\Sigma = AA^T$, does not work well in our experiments, presumably because there is too much flexibility with such a formulation. What we find to work better is the pseudo-input technique discussed in VampPrior [238]: given a set of pseudo-sequences X^* which are learnable parameters (initialized with real training sequences), we compute the sample covariance from $e(Z|X^*)$ as Σ.

This approach yields an overall objective very similar to (3.5), with the benefit of a well-defined generative model (and the Gaussian assumption being perfectly

satisfied), which allows us to borrow learning/inference techniques developed in the VAE framework. For example, masking the input for the encoder can be seen as amortized inference regularization [283]. We show experimental results on this probabilistic DAPC in section 3.1.4 and 3.1.4. In general, probabilistic DAPC performs similarly to the deterministic counterpart, though the training process is more time and memory intensive. On the other hand, these empirical results show that deviating from the Gaussian assumption, as is the case for deterministic DAPC, does not cause significant issues for representation learning in practice if proper regularization is applied.

Related to this interpretations are VAE-base sequential models [230, 284, 285] that also use reconstruction and enforce different structures/dynamics in the latent space. Most of them are designed for the purpose of generating high quality sequence data, while the qualities of their latent representations are mostly not shown for downstream tasks.

3.1.3 Existing Pretraining Strategies

Mutual information (MI) maximization is a principal approach for representation learning [286], where the objective is to maximize the MI estimate between learnt representations and inputs. The currently dominant approach for estimating MI is based on contrastive learning. For sequence data, CPC [123] uses representations at current time as a classifier to discriminate inputs of nearby frames (positive samples) from inputs of far-away steps or inputs from other sequences (negative samples) with a cross-entropy loss; this leads to the noise-contrastive estimation (NCE, 133). Deep InfoMax (DIM, [264]) generalizes the NCE estimator

with a few variants, and proposes to maximize MI between global summary features and local features from intermediate layers (rather than the inputs as in CPC). SimCLR [235] extends the contrastive loss to use a nonlinear transformation of the representation (than the representation itself) as a classifier for measuring MI. Contrastive Multiview Coding [287] generalizes the contrastive learning frame to multiple views. Momentum Contrast [270] saves memory with a dynamic dictionary and momentum encoder.

Meanwhile, there have been concerns about the contrastive learning framework. One concern is that postive and negative sample selection is sometimes time and memory consuming. To address this issue, BYOL [265] proposes to get rid of negative samples by learning a target network in an online fashion and gradually bootstrapping the latent space. Another concern is regarding the MI estimation. Though contrastive learning has an MI backbone, [273] suggests that the inductive bias of the feature extractor and parametrization of estimators might contribute more than the MI estimate itself. [272, 288] raise the concern that the MI lower bound used by contrastive learning might be too loose, and propose to use an estimate based on Wasserstein distance.

Unlike prior work, our principle for sequence representation learning is to maximize the MI between past and future latent representations, rather than the MI between representations and inputs (or shallow features of inputs). Partially motivated by the above concerns, our mutual information estimate requires no sampling and is exact for Gaussian random variables. To keep useful information from input, we use a masked reconstruction loss which has been effective for sequence data (text and speech), with an intuition resembling that of denoising autoencoders [289].

Note that by the data processing inequality, methods that maximize mutual information between current representation and future inputs also *implicitly* maximizes an upper bound of mutual information between high level representations, since $MI(Z^{past}, Z^{future}) \leq MI(Z^{past}, X^{future})$. Our method *explicitly* maximizes the mutual information between high level representations itself, while having another regularization term (masked reconstruction) that maximizes information between current input and current representations. Our results indicate that explicitly modeling the trade-off between the two can be advantageous.

In the audio domain where we will demonstrate the applicability of our method, there has been significant interest in representation learning for reducing the need for supervised data. Both contrastive learning based [57, 290, 291] and reconstruction-based [292, 293, 294, 237, 269, 295, 296] methods have been studied, as well as methods that incorporate multiple tasks [297, 298]. Our work promotes the use of a different MI estimate and combines different intuitions synergistically.

3.1.4 Experiments on Physics System, Forecasting and ASR

Noisy Lorenz Attractor

The Lorenz attractor system (also called "butterfly effect") is generated by the following differential equations: [299, 274]:

$$\begin{aligned} dx &= \sigma(y - x) \\ dy &= x(\rho - z) - y \\ dz &= xy - \beta z \end{aligned} \tag{3.6}$$

SNR	DCA	CPC	PI	MR	DAPC-det	DAPC-prob
0.3	0.084	0.676	0.585	0.574	0.865	0.816
1.0	0.153	0.738	0.597	0.885	0.937	0.943
5.0	0.252	0.815	0.692	0.929	0.949	0.949

Table 3.1: The R^2 scores of recovered 3D trajectories of noisy Lorenz attractor by different methods.

where (x, y, z) are the 3D coordinates, and we use $\sigma = 10, \beta = 8/3, \rho = 28$. The integration step for solving the system of equations is 5×10^{-3}.

We lift the 3D trajectory into 30D using a nonlinear neural network with 2 hidden layers, each with 128 neurons and the Exponential Linear Unit [300]. We further perturb the 30D trajectory with additive Gaussian noise to obtain datasets of three different Signal-to-Noise Ratios (SNRs, ratio between the power of signal and the power of noise): 0.3, 1.0, and 5.0. The smaller the SNR is, the more challenging it is to recover the clean 3D trajectory (see the left panel of Figure 3.2 for the comparison between noisy 30D trajectory and the clean 30D trajectory).

We compare both deterministic and probabilistic DAPC against representative unsupervised methods including DCA [274], CPC [123], pure PI learning which corresponds to DAPC with $\beta = 0$, and masked reconstruction (MR) [237]. Except for DCA which corresponds to maximizing PI with a linear orthogonal feedforward net, the other methods use bidirectional GRU [75] for mapping the inputs into feature space (although uni-GRU performs similarly well). A feedforward DNN is used for reconstruction in MR and DAPC.

More specifically, CPC, MR, DAPC all use the bidirectional GRU where the learning rate is 0.001, and dropout rate is 0.7. Our GRU has 4 encoding layers with hidden size 256. The batch size is (20, 500, 30). For CPC, the temporal lag

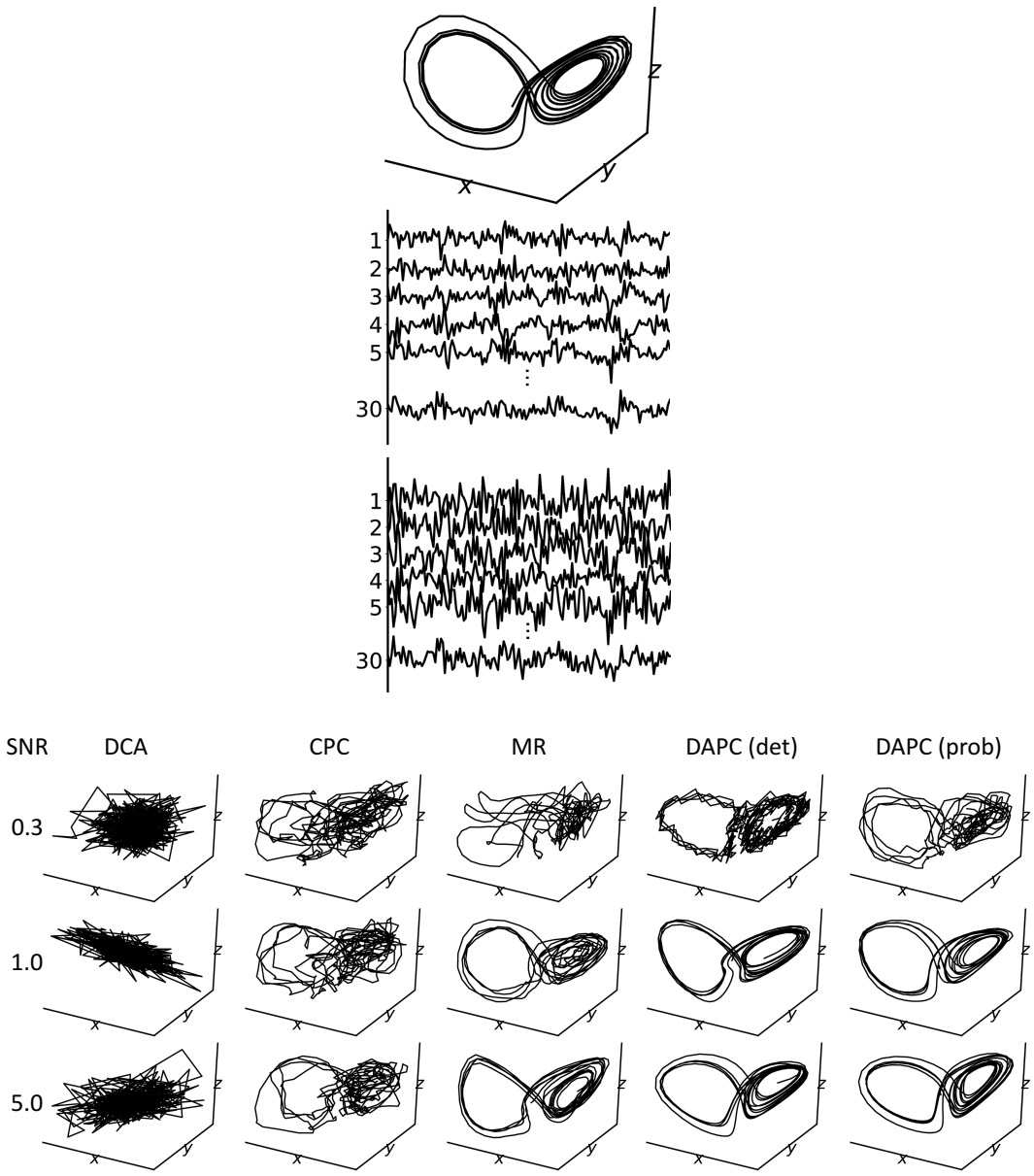


Figure 3.2: **Upper panel**. Top: the ground-truth 3D Lorenz attractor. Middle: the 30D non-linearly lifted trajectory. Bottom: corrupted 30D trajectory by white noise with SNR=0.3. **Lower Panel**. 3D trajectories extracted by different methods for three SNR levels: 0.3, 1.0, 5.0.

k=4. For DAPC, $\beta = 0.1$, $T = 4$, $s = 0$, $\alpha = 0$, $\gamma = 0.1$. For masked reconstruction, we use at most 2 masks on the frequency axis with width up to 5, and at most 2 masks on the time axis with width up to 40. The DNN decoder has 3 hidden layers, each with size 512. DCA's setup is completely adopted from [274]. The

SNR	Full Recon	uni-GRU	Regular
0.3	0.803	0.857	0.865
1.0	0.812	0.905	0.937
5.0	0.852	0.903	0.949

Table 3.2: The R^2 scores for the ablation study of (deterministic) DAPC for Lorenz attractor.

SNR	Full Recon only	Full Recon with PI
0.3	0.441	0.803
1.0	0.737	0.812
5.0	0.802	0.852

Table 3.3: The R^2 scores for full reconstruction only and full reconstruction with PI.

same architectures are used in the forecasting tasks.

Figure 3.2 provides qualitatively results for the recovered 3D trajectories by different methods. Observe that DCA fails in this scenario since its feature extraction network has limited capacity to invert the nonlinear lifting process. CPC is able to recover the 2 lobes, but the recovered signals are chaotic. Masked reconstruction is able to produce smoother dynamics for high SNRs, but its performance degrades quickly in the more noisy scenarios. Both deterministic and probabilistic DAPC recover a latent representation which has overall similar shapes to the ground truth 3D Lorenz attractor, and exhibits smooth dynamics enforced by the PI term.

We quantitatively measure the recovery performance with the R^2 score, which is defined as coefficient of determination. R^2 score normally ranges from 0 to 1 where 1 means the perfect fit. Negative scores indicate that the model fits the data worse than a horizontal hyperplane. The R^2 results are given in Table 3.1. Our results quantitatively demonstrate the clear advantage of DAPC across different noise levels.

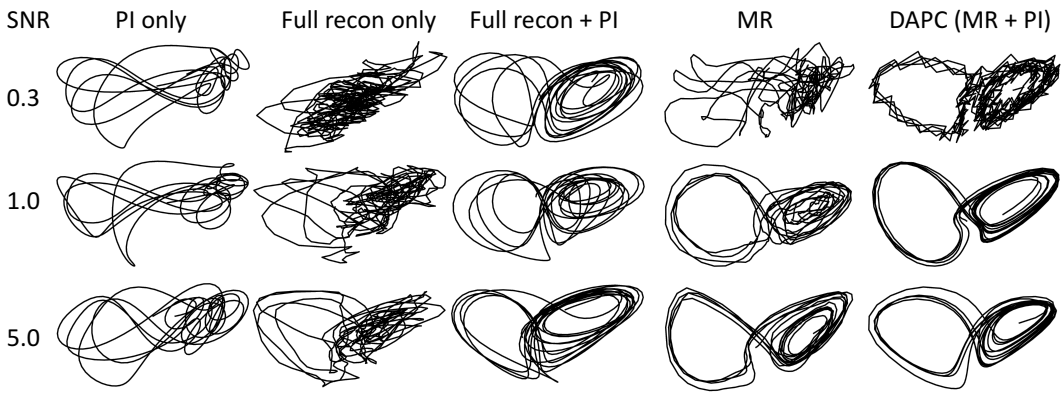


Figure 3.3: Illustration of how PI can improve both full reconstruction and masked reconstruction (MR). We can observe that PI can greatly improve the recovery quality, especially when SNR is low (very noisy).

SNR	k=2	k=4	k=6	k=8	k=10
0.3	0.477	0.676	0.663	0.658	0.608
1.0	0.556	0.738	0.771	0.721	0.642
5.0	0.652	0.815	0.795	0.775	0.717

Table 3.4: The R^2 scores for CPC with different temporal lags (k).

We also give an ablation study on several components of DAPC in Table 3.2, where we attempt full reconstruction without masking, masked reconstruction with unidirectional encoder uni-GRU, and the regular setup (masked reconstruction + bi-GRU). Using full reconstruction yields worse results than using masked reconstruction at all noise levels, while uni-GRU degrades the performance less.

We show in Table 3.3 and Figure 3.3 how PI can improve both full reconstruction and masked reconstruction. In Table 3.3, when SNR=0.3, PI can greatly boost the performance of full reconstruction. We also tuned the temporal lag parameter k w.r.t. both quantitative and qualitative results (Table 3.4 and Figure 3.4). CPC performance starts to deteriorate after k=8, while k=4, 6, 8 have similar results. Based on the R^2 scores, we select k=4 as our final temporal lag. Similar tuning is also performed for CPC on the downstream forecasting experiments.

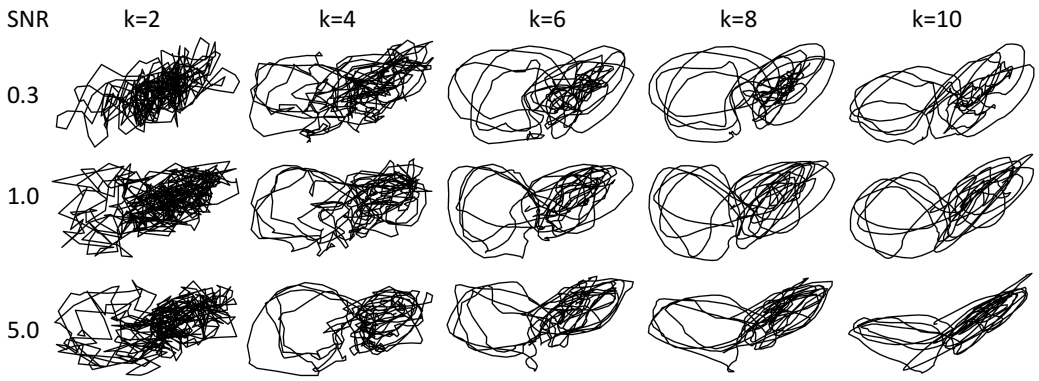


Figure 3.4: Qualitative recovery results by CPC w.r.t. different temporal lags (k).

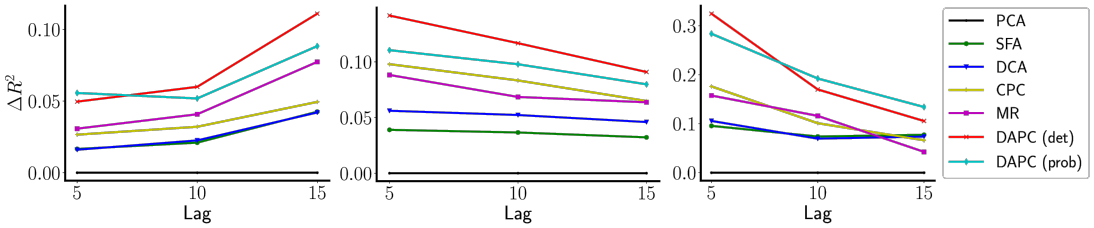
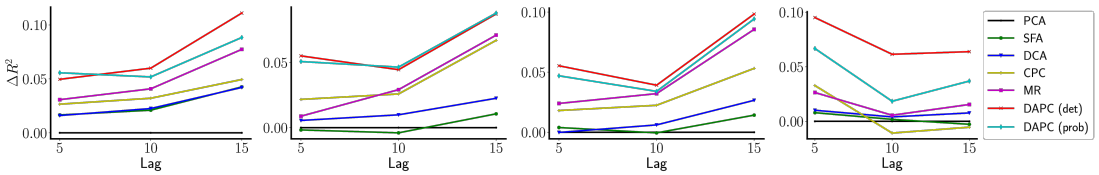


Figure 3.5: Different models' R^2 score improvements over PCA for three forecasting tasks, each with three different *lag* values. Left: temperature. Middle: dorsal hippocampus. Right: motor cortex.

Forecasting with Linear Regression

We then demonstrate the predictive power of learnt representations in downstream forecasting tasks on 3 real-world datasets used by [274], involving multi-city temperature time series data (Temp, [301]), dorsal hippocampus study (HC, [302]), and motor cortex (M1, [303]). For each model, unsupervised representation learning is performed on the training set with a uni-directional GRU, which prevents information leakage from the future. After that, we freeze the model and use it as a feature extractor. The representations at each time step are used as inputs for predicting the target at a future time step. As an example, we can extract a representation for today's weather based on past weather only (as the encoder is uni-directional), and use it to predict future temperature which is *lag* days away (a larger *lag* generally leads to a more difficult forecasting task).



$d = 5$ $d = 10$ $d = 15$ $d = 20$

Figure 3.6: On temperature dataset, we analyze the performances of different latent dimensions (dimension of z_t): 5, 10, 15, 20. ΔR^2 corresponds to R^2 score improvement over PCA.

dataset	lag	FR	FR+PI	Improvement	MR	MR+PI	Improvement	PI only
Temp	5	0.721	0.725	0.58%	0.705	0.724	2.68%	0.698
	10	0.672	0.685	1.99%	0.672	0.691	2.85%	0.645
	15	0.675	0.686	1.64%	0.673	0.707	4.99%	0.632
HC	5	0.252	0.260	3.24%	0.251	0.304	21.25%	0.169
	10	0.222	0.231	4.46%	0.222	0.271	21.72%	0.137
	15	0.183	0.194	5.97%	0.205	0.232	13.28%	0.085
M1	5	0.390	0.405	4.03%	0.352	0.519	47.45%	0.239
	10	0.372	0.394	5.83%	0.369	0.422	14.64%	0.156
	15	0.268	0.293	9.10%	0.241	0.304	26.10%	0.076

Table 3.5: The R^2 scores for full reconstruction (FR), full reconstruction with PI (FR+PI), masked reconstruction (MR) and DAPC (MR+PI). We also demonstrate the improvements in percentage brought by adding PI. On average, PI improves full reconstruction by 4.09% and masked reconstruction by 17.22%. Additionally, in the last column, we show PI objective alone is not powerful enough to learn predictive components (R^2 scores are low).

Following [274], the predictor from the extracted feature space to the target is a linear mapping, trained on samples of paired current feature and future target, using a least squares loss. We use the same feature dimensionality as in their work, for each dataset. These forecasting tasks are evaluated by the R^2 regression score, which measures the linear predictability.

Besides DCA, CPC and MR, we further include PCA and SFA [304] (similar to DCA with $T = 1$ for PI estimation), which are commonly used linear dimension reduction methods in these fields. PCA serves as the baseline and we report R^2 score improvements from other methods. Figure 3.5, 3.6 give the performances of different methods for the three datasets, with three different *lag*s (the number

d	k=2	k=4	k=6	k=8	k=10
5	0.685	0.701	0.690	0.687	0.681
10	0.661	0.663	0.675	0.631	0.622
15	0.633	0.645	0.637	0.634	0.617

Table 3.6: The R^2 scores for CPC with different temporal lags (k) on the temperature dataset.

of time steps between current and future for the forecasting task): 5, 10, and 15. DAPC consistently outperforms the other methods. In Table 3.5, we show how PI helps DAPC improve over either full reconstruction (e.g., classical auto-encoder) or masked reconstruction, and how reconstruction losses help DAPC improve over PI alone on this task. These results demonstrate that the two types of losses, or the two types of mutual information (MI between input and latent, and MI between past and future) can be complementary to each other.

Table 3.6 shows the temporal lag tuning for CPC on the temperature dataset.

Pretraining for Automatic Speech Recognition (ASR)

A prominent usage of representation learning in speech processing is to pretrain the acoustic model with an unsupervised objective, so that the resulting network parameters serve as a good initialization for the supervised training phase using labeled data [305]. As supervised ASR techniques have been improved significantly over the years, recent works start to focus on pre-training with large amounts of unlabeled audio data, followed by finetuning on much smaller amounts of supervised data, so as to reduce the cost of human annotation.

We demonstrate different methods on two commonly used speech corpora for this setup: Wall Street Journal [306] and LibriSpeech [120]. For WSJ, we pretrain on *si284* partition (81 hours), and finetune on *si84* partition (15 hours) or the *si284*

partition itself. For Librispeech, we pretrain on the *train_960* partition (960 hours) and finetune on the *train_clean_100* partition (100 hours). Standard dev and test splits for each corpus are used for validation and testing.

In the experiments, we largely adopt the transformers-based recipe from ESPnet [307], as detailed in [308], for supervised finetuning. Note that we have spent effort in building strong ASR systems, so that our baseline (without pretraining) already achieves low WERs and improving over it is non-trivial. This can be seen from the result table where our baseline is often stronger than the best performance from other works. In the pretraining stage, we pretrain an encoder of 14 transformer layers, which will be used to initialize the first 14 layers of ASR model. For masked reconstruction, we use 2 frequency masks as in finetuning, but found more time masks can improve pretraining performance. We set the number of time masks to 4 for WSJ, and 8 for LibriSpeech which has longer utterances on average.

The hyperparameters we tune include T, s, α, β and γ from our learning objective. We select hyperparameters which give the best dev set WER, and report the corresponding test set WER. In the end, we use $T = 4$ for estimating the PI term, $\gamma = 0.05, \beta = 0.005$ and set $s = 2$ for WSJ and $s = 1$ for LibriSpeech if we use shifted reconstruction. Since the pretraining objective is a proxy for extracting the structure of data and not fully aligned with supervised learning, we also tune the number of pretraining epochs, which is set to 5 for WSJ and 1 for LibriSpeech.

We perform an ablation study for the effect of different variants of DAPC (MR+PI) on the WSJ dataset, and give dev/test WERs in Table 3.7. We tune the hyperparameters for multi-scale PI and shifted reconstruction for the 15-hour finetuning setup, and observe that each technique can lead to further improve-

Methods	*dev93*	*eval92*
Finetune on 15 hours		
w.o. pretrain	12.91±0.36	8.98±0.44
PI only	12.54±0.32	9.02±0.43
MR	12.27±0.23	8.15±0.34
DAPC	12.31±0.36	7.74±0.20
DAPC + multi-scale PI	12.15±0.35	7.64±0.15
DAPC + shifted recon	11.93±0.16	7.68±0.05
DAPC + both	**11.57**±0.22	**7.34**±0.13
Finetune on 81 hours		
w.o. pretrain	6.34±0.13	3.94±0.33
MR	6.24±0.14	3.84±0.09
DAPC	5.90±0.16	3.58±0.08
DAPC + both	**5.84**±0.04	**3.48**±0.08

Table 3.7: Ablation study on different variants of DAPC. We give WERs (%) of ASR models pretrained with different variants on WSJ. Models are pretrained on 81 hours, and finetuned on either 15 hours or 81 hours. All results are averaged over 3 random seeds.

ment over the basic DAPC, while combining them delivers the best performance. The same hyperparameters are used for the 81-hour finetuning setup, and we find that with more supervised training data, the baseline without pretraining obtains much lower WERs, and pure masked reconstruction only slightly improves over the baseline, while the strongest DAPC variant still achieves 7.9% and 11.7% relative improvements on *dev93* and *eval92* respectively.

In Table 3.8, we provide a more thorough comparison with other representation learning methods on the LibriSpeech dataset. We compare with CPC-type mutual information learning methods including wav2vec [57], vq-wav2vec which performs CPC-type learning with discrete tokens followed by BERT-style learning [290], and wav2vec 2.0 which incorporates masking into contrastive learning [37]. We also compare with two reconstruction-type learning approaches DeCoAR [295] and TERA [296]. Observe that DAPC and its variant achieve lower WER than MR: though our baseline is strong, DAPC still reduces WER by 8%,

Methods	*dev_clean*	*test_clean*
wav2vec [57]	-	6.92
discrete BERT+vq-wav2vec [290]	4.0	4.5
wav2vec 2.0 [37]	**2.1**	**2.3**
DeCoAR [295]	-	6.10
TERA-large [296]	-	5.80
MPE [309]	8.12	9.68
Bidir CPC [310]	8.86	8.70
MR [237]	4.66	5.02
MR (sub-word)	5.57	6.18
w.o. pretrain	4.84	5.11
DAPC	4.52	4.86
DAPC+multi-scale PI	4.46	4.77
DAPC+shifted recon	4.50	4.80
DAPC+multi-scale PI+shifted recon	**4.42**	**4.70**
w.o. pretrain (sub-word)	6.16	6.81
DAPC (sub-word)	5.50	5.79

Table 3.8: WER results of different methods on LibriSpeech. All representation methods are pretrained on the full corpus (960h) and finetuned on *train_clean_100* (100h). For MR and DAPC, the default ASR recipe uses characters as token set and decodes with word RNNLM. For results denoted with sub-word, the ASR recipe uses 5000 unigrams as token set and decodes with token-level RNNLM. All the results are averaged over 3 seeds.

while MR only improves by 1.76%. This shows the benefit of PI-based learning in addition to masked reconstruction. DAPC achieves 15% relative improvement over the baseline (without pretraining), showing that our method is generally effective for different types of ASR systems. Nevertheless, our method does not outperform vq-wav2vec and wav2vec 2.0; we suspect it is partly because our models have much smaller sizes (around 30M weight parameters) than theirs (vq-wav2vec has 150M weight parameters, and wav2vec has 300M weight parameters for the acoustic model, along with a very large neural language model) and it is future work to scale up our method.

3.1.5 Discussion

In this work, we introduce a novel representation learning method, DAPC, for sequence data. Our learnt latent features capture the essential dynamics of the underlying data, contain rich information of both input observations and context states, and are shown to be useful in a variety of tasks. As future work, we may investigate other predictive information estimators that further alleviate the Gaussian assumption. On the other hand, more advanced variational inference techniques may be applied to the probabilistic version of DAPC to boost the performance. DAPC provides a general alternative for mutual information-based learning of sequence data and we may investigate its potential usage in other domains such as NLP, biology, physics, etc.

3.2 Self-supervision for Disentanglement: Contrastively Disentangled Sequential Variational Audoencoder

3.2.1 Introduction

The goal of self-supervised learning methods is to extract useful and general representations without any supervision, and to further facilitate downstream tasks such as generation and prediction [124]. Despite the difficulty of this task, many existing works have shed light on this field across different domains such as computer vision [123, 264, 235, 311], natural language processing [266, 91, 31] and speech processing [290, 269, 237, 37, 261] (also see a huge number of references in these papers). While the quality of the learnt representations

improves gradually, recent research starts to put more emphasis on learning disentangled representations. This is because disentangled latent variables may capture separate variations of the data generation process, which could contain semantic meanings, provide the opportunity to remove unwanted variations for a lower sample complexity of downstream learning [312, 313], and allow more controllable generations [314, 315, 316]. These advantages lead to a rapidly growing research area, studying various principles and algorithmic techniques for disentangled representation learning [317, 318, 319, 320, 321, 322, 323, 324]. One concern raised in [322] is that without any inductive bias, it would be extremely hard to learn meaningful disentangled representations. On the other hand, this concern could be much alleviated in the scenarios where the known structure of the data can be exploited.

In this work, we are concerned with the representation learning for sequence data, which has a unique structure to utilize for disentanglement learning. More specifically, for many sequence data, the variations can be explained by a dichotomy of a static (time-invariant) factor and dynamic (time-variant) factors, each varies independently from the other. For example, representations of a video recording the movements of a cartoon character could be disentangled into the character identity (static) and the actions (dynamic). For audio data, the representations shall be able to separate the speaker information (static) from the linguistic information (dynamic).

We introduce Contrastively Disentangled Sequential Variational Autoencoder (C-DSVAE), a method seeking for a clean separation of the static and dynamic factors for the sequence data. Our method extends the previously proposed sequential variational autoencoder (VAE) framework, and performs learning

with a different evidence lower bound (ELBO) which naturally contains mutual information (MI) terms to encourage disentanglement. Due to the difficulty in estimating high dimensional complex distributions (e.g., for the dynamic factors), we further incorporate the contrastive estimation for the MI terms with systematic data augmentation techniques which modify either the static or dynamic factors of the input sequence. The new estimation method turns out to be more effective than the minibatch sampling based estimate, and introduces additional inductive biases towards the invariance. To our knowledge, we are the first to synergistically combine the contrastive estimation and sequential generative models in a principled manner for learning disentangled representations. We validate C-DSVAE on four datasets from the video and audio domains. The experimental results show that our method consistently outperforms the previous state-of-the-art (SOTA) methods, both quantitatively and qualitatively.

3.2.2 C-DSVAE for Sequence Disentanglement

We denote the observed input sequence as $x_{1:T} = \{x_1, x_2, ..., x_T\}$ where x_i represents the input feature at time step i (e.g., x_i could be a video frame or the spectrogram feature of a short audio segment), and T is the sequence length. The latent representations are divided into the static factor s, and the dynamic factors $z_{1:T}$ where z_i is the learnt dynamic representation at time step i.

Probabilistic Model

We assume that in the ground-truth generation process, z_i depends on $z_{<i} = \{z_0, z_1, ..., z_{i-1}\}$ where $z_0 = \mathbf{0}$, and the observation x_i is independent of other frames

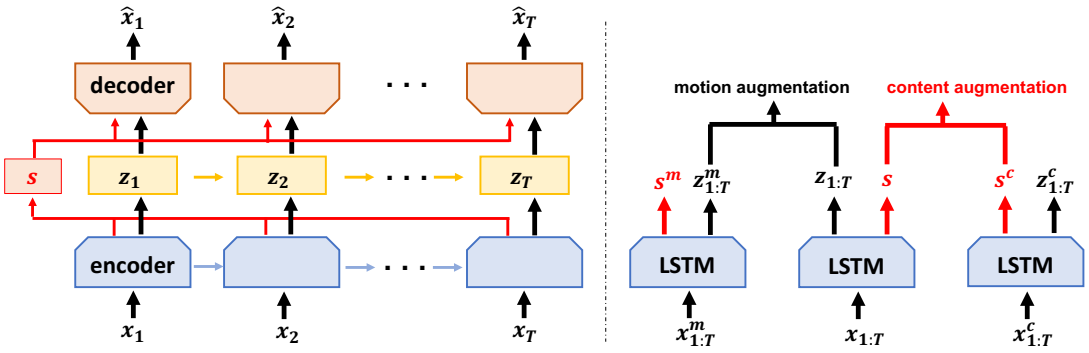


Figure 3.7: The illustration of our C-DSVAE model. **Left panel** is the general structure of the sequence-to-sequence auto-encoding process: each frame is passed to the LSTM cell; dynamic factors $z_{1:T}$ are extracted for each time step; the static factor s is extracted by summarizing the full sequence; the generation/reconstruction of frame i depends on s and z_i. **Right panel** depicts the contrastive learning module of C-DSVAE: $x^m_{1:T}$ is the motion augmentation of $x_{1:T}$ and $x^c_{1:T}$ is the content augmentation of $x_{1:T}$; dynamic factors $z^m_{1:T}$ of $x^m_{1:T}$ can be seen as the positive sample for the anchor $z_{1:T}$ in the contrastive estimation w.r.t. the motion, and similarly the static factor s^c of $x^c_{1:T}$ can be viewed as the positive sample for s w.r.t. the content.

conditioned on z_i and s. Furthermore, we assume the static variable s and the dynamic variables $z_{1:T}$ are independent from each other, i.e., $p(s, z_{1:T}) = p(s)p(z_{1:T})$. Formally, let $z = (s, z_{1:T})$ and we have the following complete likelihood

$$p(x_{1:T}, z) = p(z)p(x_{1:T}|z) = \left[p(s)\prod_{i=1}^{T} p(z_i|z_{<i})\right] \cdot \prod_{i=1}^{T} p(x_i|z_i, s) \tag{3.7}$$

where $p(z)$ is the prior for sampling z. Our formulation captures the general intuition that we can separate the variations of the sequence into the time-dependent dynamic component (described by z_i's), and a static component (described by s) which remains the same for different time steps in the same sequence but differs between sequences. For example, in the cartoon character video (see Figure 3.9), the hair, shirt and pants should be kept the same when the character walks around, but different videos can have different characters. Similarly, for a speech utterance, the phonetic transcription controls the vocal tract motion and the sound being produced over time, but the speaker identity remains the same for the whole utterance. In this work, we refer to the dynamic component as

"motion" and the static component as "content".

For the prior distributions, we choose $p(s)$ to be the standard Gaussian $\mathcal{N}(0, I)$, and $p(z_i|z_{<i})$ to be $\mathcal{N}(\mu(z_{<i}), \sigma^2(z_{<i}))$ where $\mu(\cdot)$ and $\sigma(\cdot)$ are modeled by LSTMs [19]. In operations, the drawn sample s is shared throughout the sequence. To draw a sample of z_t, we first take a sample of z_{t-1} as the input of the current LSTM cell. Then forwarding the LSTM for one step gives us the distribution of z_t, from which we draw a sample with the reparameterization trick [282].

To extract the latent representations given only the observed data $x_{1:T}$ where the motion and content are mixed together, we hope to learn a posterior distribution $q(z|x_{1:T})$ where the two components are disentangled. That is, similar to the prior, our posterior should have a factorized form:

$$q(z|x_{1:T}) = q(z_{1:T}, s|x_{1:T}) = q(z_{1:T}|x_{1:T})q(s|x_{1:T}) = q(s|x_{1:T}) \prod_{i=1}^{T} q(z_i|z_{<i}, x_{\leq i}). \tag{3.8}$$

The posterior distributions are also modeled by LSTMs. A nested sampling procedure, resembling that of the prior, is applied to the posterior of dynamic variables. Similar parameterizations of dynamic variables by recurrent networks have been proposed in prior works [230, 325, 326]. The standard loss function for learning the latent representations is an ELBO [285, 327] (also see a derivation in Appendix B.2.1):

$$\max_{p,q} \ \mathbb{E}_{x_{1:T} \sim p_D} \mathbb{E}_{q(z|x_{1:T})} \left[\log p(x_{1:T}|z) - KL[q(z|x_{1:T})||p(z)] \right] \tag{3.9}$$

where p_D is the empirical data distribution. Under the parameterization of the posterior where s and $z_{1:T}$ are mutually independent, the KL-divergence term reduces to

$$KL[q(z|x_{1:T})||p(z)] = KL[q(s|x_{1:T})||p(s)] + KL[q(z_{1:T}|x_{1:T})||p(z_{1:T})] \tag{3.10}$$

where the second term is approximated with the sampled trajectories of the dynamic variables $z_{1:T}$.

Our approach: Mutual information-based disentanglement

Several existing sequence representation learning methods [285, 327, 316] are built on top of the loss function (3.9). However, this formulation also brings several issues. The KL-divergence regularizes the posterior of the static or dynamic factors to be close to the corresponding priors. When modeling them with powerful neural architectures like LSTMs, it is possible for the KL-divergence to be close to zero, yet at the same time, the posteriors become non-informative of the inputs; this is a common issue for deep generative models like VAE [328]. Techniques have been proposed to alleviate this issue, including adjusting the relative weights of the loss terms to regularize the capacity of posteriors [318], replacing the individual posteriors in the KL terms with aggregated posteriors [238, 329, 330] and enforcing latent structures (such as disentangled representations) in the posteriors [321, 320, 331].

Our principled approach for learning useful representations from sequences is inspired by these prior works, and at the same time incorporates the unique sequential structure of the data. Without inductive biases, the goal of disentanglement can hardly be achieved since it is possible to find entangled s and $z_{1:T}$ that explain the data equally well (in fact, by Theorem 1 of [322], there could exist infinitely many such entangled factors). The problem may seem less severe in our setup, since s is shared across time and it is hard for such s to capture all dynamics. Nevertheless, since both s and z_i are used in generating x_i, it is still possible for z_i to carry some static information. In the extreme case where each

z_i encompasses s, s would no longer be indispensable for the generation. This issue motivated prior works to optimize the (estimate of) mutual information among latent variables and inputs [327, 316, 332].

Our method seeks to achieve clean disentanglement of s and $z_{1:T}$, by optimizing the following objective function, which introduces additional MI terms to the vanilla ELBO in (3.9):

$$\begin{aligned}
&\max_{p,q} \mathbb{E}_{x_{1:T}\sim p_D}\mathbb{E}_{q(z|x_{1:T})}[\log p(x_{1:T}|z)] - KL[q(z)\|p(z)] \\
&\quad = \mathbb{E}_{x_{1:T}\sim p_D}\mathbb{E}_{q(z|x_{1:T})}[\log p(x_{1:T}|z)] - (KL[q(s|x_{1:T})\|p(s)] + KL[q(z_{1:T}|x_{1:T})\|p(z_{1:T})]) \\
&\qquad\qquad + I_q(s; x_{1:T}) + I_q(z_{1:T}; x_{1:T}) - I_q(z_{1:T}; s)
\end{aligned} \tag{3.11}$$

where the aggregated posteriors are defined as $q(z) = q(s, z_{1:T}) = \mathbb{E}_{p_D}[q(s|x_{1:T})q(z_{1:T}|x_{1:T})]$, $q(s) = \mathbb{E}_{p_D}[q(s|x_{1:T})]$, $q(z_{1:T}) = \mathbb{E}_{p_D}[q(z_{1:T}|x_{1:T})]$, and the MI terms are defined as $I_q(s; x_{1:T}) = \mathbb{E}_{q(s,x_{1:T})}\left[\log \frac{q(s|x_{1:T})}{q(s)}\right]$, $I_q(z_{1:T}; x_{1:T}) = \mathbb{E}_{q(z_{1:T},x_{1:T})}\left[\log \frac{q(z_{1:T}|x_{1:T})}{q(z_{1:T})}\right]$ (and $I_q(z_{1:T}; s)$ is defined similarly). The intuition behind (3.11) is simple: besides explaining the data and matching the posteriors with priors, $z_{1:T}$ and s shall contain useful information from $x_{1:T}$ while excluding the redundant information from each other. We further justify (3.11) by showing that it still forms a valid ELBO.

Theorem 1 *With our parameterization of* $(s, z_{1:T})$, (3.11) *is a valid lower bound of the data log-likelihood* $\mathbb{E}_{x_{1:T}\sim p_D} \log(x_{1:T})$.

The full proof can be found in Appendix B.2.2. With this guarantee, we can follow the spirit of [318, 321] to add and adjust the additional weight coefficients α to the KL terms, β to the MI terms $I_q(s; x_{1:T})$, $I_q(z_{1:T}; x_{1:T})$, and γ to $I_q(z_{1:T}; s)$,

$$\begin{aligned}
&\mathbb{E}_{x_{1:T}\sim p_D}\mathbb{E}_{q(z|x_{1:T})}[\log p(x_{1:T}|z)] - \alpha(KL[q(s|x_{1:T})\|p(s)] + KL[q(z_{1:T}|x_{1:T})\|p(z_{1:T})]) \\
&\qquad + \beta(I_q(s; x_{1:T}) + I_q(z_{1:T}; x_{1:T})) - \gamma I_q(z_{1:T}; s).
\end{aligned} \tag{3.12}$$

It remains to estimate the objective (3.12) for optimization. The KL term for $z_{1:T}$ is estimated with the standard Monte-Carlo sampling, using trajectories of $z_{1:T}$ [285]. The KL term for s can be estimated analytically. For the MI terms, we attempt two estimations. The first estimation uses a standard mini-batch weighted sampling (MWS) following [321, 327]. Due to the high dimensionality of $z_{1:T}$ and the complex dependency among time steps, it may be hard for MWS to estimate the distributions accurately, so we also explore non-parametric contrastive estimations for $I_q(s; x_{1:T})$ and $I_q(z_{1:T}; x_{1:T})$ with additional data augmentations, which we will detail below.

C-DSVAE: Contrastive estimation with augmentation

A contrastive estimation of $I(z_{1:T}; x_{1:T})$ can be defined as follows

$$C(z_{1:T}) = \mathbb{E}_{p_D} \log \frac{\phi(z_{1:T}, x^+_{1:T})}{\phi(z_{1:T}, x^+_{1:T}) + \sum_{j=1}^{n} \phi(z_{1:T}, x^j_{1:T})} + \log(n+1) \tag{3.13}$$

where x^+ is a "positive" sequence, while x^j, $j = 1, \ldots, n$ is a collection of n "negative" sequences. When $\phi(z_{1:T}, x_{1:T}) = \frac{q(x_{1:T}|z_{1:T})}{q(x_{1:T})}$, one can show that (3.13) approximates $I_q(z_{1:T}, x_{1:T})$; see Appendix B.2.4 for a proof. To implement (3.13), $z_{1:T}$ is the trajectory obtained by using the mean at each time step from $q(z_{1:T}|x_{1:T})$ for a input sequence $x_{1:T}$, while $x^j_{1:T}$ is a randomly sampled sequence from the minibatch. A similar contrastive estimation is defined for s as well. To provide meaningful positive sequences that encourage the invariance of the learnt representations, we obtain $x^+_{1:T}$ by systematically perturbing $x_{1:T}$. In Sec B.2.4, we discuss how the augmented data give good estimate of (3.13) under additional assumptions.

Figure 3.8: Data augmentations on SM-MNIST and Sprites. **Left panel**: SM-MNIST. The first row is the raw input sequence of moving digits. The second row reverses the sequence order of the raw sequence, serving as the content augmentation. The third row stretches the frames and enhances the color, which changes the digit styles but does not change the digit movements and thus forms a motion augmentation. **Right panel**: Sprites. The first row is the raw input character video. The second row is the content augmentation with a random order. The third row is a motion augmentation produced by distorting colors and adding Gaussian noise.

Content augmentation The static factor (e.g., the character identity in videos or the speaker in audios) is shared across all the time steps, and should not be affected by the exact order of the frames. We therefore randomly shuffle or simply reverse the order of time steps to generate the content augmentation of $x_{1:T}$ and denote it by $x^c_{1:T}$. The static and dynamic latent factors of $x^c_{1:T}$, modeled by q, are denoted by s^c and $z^c_{1:T}$. This is an inexpensive yet useful strategy, applicable to both audios and videos. Similar ideas were introduced in [327] albeit not used for contrastive estimations.

Motion augmentation In motion augmentation, we would like to maintain the dynamic factors (e.g., actions or movements) while replacing the content with a meaningful alternative. Thanks to the recent efforts in contrastive learning, multiple effective strategies have been proposed. For video datasets, we adopt the combination of cropping, color distortion, Gaussian blur and reshaping [235, 333]. For audio datasets, we use classical unsupervised voice conversion algorithms [334, 335]. The motion augmentation of $x_{1:T}$ is denoted by $x^m_{1:T}$, with the static factor s^m and dynamic factors $z^m_{1:T}$ estimated by q.

Note that $x^m_{1:T}$ is the motion augmentation of $x_{1:T}$, and $x_{1:T}$ in turn is also the motion augmentation of $x^m_{1:T}$. Similarly, s and s^c are mutually the "positive" sample to each other w.r.t. the static factor. During the training, it is efficient to generate the augmentations for each sequence in the minibatch, and apply contrastive estimation on both the original data and the augmented data. This leads to the following final estimates:

$$\begin{aligned} I_q(z_{1:T}; x_{1:T}) &\approx \frac{1}{2}(C(z_{1:T}) + C(z^m_{1:T})), \\ I_q(s; x_{1:T}) &\approx \frac{1}{2}(C(s) + C(s^c)) \end{aligned} \tag{3.14}$$

where we use $\phi(z_{1:T}(x_{1:T}), x^*_{1:T}) = \exp(\text{sim}(z_{1:T}, z^*_{1:T})/\tau)$ with * indicating the latent variables for the positive/augmented or negative sequences (extracted from q). $\text{sim}(\cdot, \cdot)$ is the cosine similarity function and $\tau = 0.5$ is a temperature parameter. This form of ϕ using cosine similarity is widely adopted in contrastive learning [235, 236] as it removes one degree of freedom (length) for high dimensional feature space. Plugging (3.14) into (3.12) gives our final learning objective with contrastive estimation. We name our method Contrastively Disentangled Sequential Variational Encoder (C-DSVAE), and a full model illustration is shown in Figure 3.7.

In practice, we find the contrastive estimation is more effective than the MWS based on Gaussian probabilities (see an ablation study in section 3.2.4). Interestingly, we observe that with only the contrastive estimation and augmentations, $I_q(z_{1:T}; s)$ (last term in (3.11)) decreases even if we do not include its estimation in our optimization process, indicating that a good disentanglement between $z_{1:T}$ and s could be attained through contrastive estimations solely.

As it will become evident in the experiments, our method achieves a cleaner separation of the dynamic and static factors than previous methods. We believe

this is partly due to the inductive bias introduced by our approach: in order to consistently map two sequences with the same motion but different contents (which vary independently from the motion according to the generation process (3.7)) to the same latent representation $z_{1:T}$, we must discard the information of the inputs regarding s; similar arguments hold for the static variables. At the same time, we enforce that the extracted $z_{1:T}$ and s together should reconstruct $x_{1:T}$ so that no information is lost in the auto-encoding process. To our knowledge, we are the first to use the contrastive estimation with augmentations in the sequential VAE scenario, for pushing the disentanglement of variations.

3.2.3 Prior Efforts for Disentanglement

In terms of the general frameworks, disentangled representation learning methods can be categorized into GAN-based [336] and VAE-based [282] approaches. GAN-based methods like MoCoGAN [337] aim at generating videos from content noise and motion noise. These models typically do not explicitly learn the encoder from inputs to the latent variables, rendering them less applicable for representation learning.

The VAE framework parameterizes both the encoder (variational posterior) and the decoder (reconstruction), and optimize them jointly with a well-defined ELBO objective. The encoder allows us to directly control and reason about the properties of the extracted features, which can be straightforwardly used for downstream tasks. Many variants of VAE were proposed to encourage the disentanglement/interpretability of *individual latent dimensions*. β-VAE [318] adds a coefficient to the per-sample KL-divergence term to better constrain the

information bottleneck. FactorVAE [320] and β-TCVAE [321] explicitly separate out the total correlation term, and adjust its weight coefficient in the objective to encourage the per-dimensional disentanglement of the *aggregated* posterior. From a theoretical standpoint, [322] points out that without further inductive biases, it is impossible to learn disentangled representations with the standard VAE.

To handle the sequence data, vanilla VAE is extended to the recurrent version, and prior works have explored different approaches for separating the content and the motion. FHVAE [284] designs a hierarchical VAE model which uses "sequence-dependent variables" (corresponding to the speaker factor) and "sequence-independent variables" (corresponding to the linguistic factors) for modeling speech sequences, and performs learning with the per-sequence ELBO while respecting the hierarchy in designing the posterior. FHVAE does not use additional loss terms for encouraging the disentanglement. Different from FHVAE, DSVAE [285] explicitly models the static and dynamic factors in its graphical model and its posterior has a factorized form. A few recent works (including ours) inherit the clean formulation of DSVAE in terms of the graphical model and prior/posterior parameterization. S3VAE [327] introduces additional loss terms to the objective of DSVAE in an ad-hoc fashion, such as a triplet loss that encourages the invariance of the learnt static factors by permuting the frame order (spiritually similar to our contrastive estimation for $I(s, x_{1:T})$), an additional prediction loss on $z_{1:T}$ leveraging external supervision for the motion labels, and an MI term of $I(s, z_{1:T})$ to be minimized. Our C-DSVAE differs from S3VAE in that we naturally introduce the MI terms from the fundamental principle of VAE, and use augmentations instead of external supervision for learning the static and dynamic factors. Also based on the DSVAE formulation, R-WAE [316]

proposes to replace the distance measure between the aggregated posterior and the prior with the Wasserstein distance, in the belief that the KL divergence is too restrictive. In R-WAE, Maximum Mean Discrepancy (MMD) based estimation or GAN-based estimation of Wasserstein distance is used, either of which requires heavy tuning of hyperparameters. Another similar method, IDEL [338], optimizes the additional set of MI terms similar to ours. However, in our work, we show that these MI terms can be naturally derived from the fundamental principle of VAE, and we additionally present the contrastive estimation and data augmentations to strengthen them, which was not done before.

Another relevant research area is nonlinear ICA, which tries to understand the conditions under which the disentanglement can be achieved, possibly with contrastive learning or VAE [339, 340, 341, 323, 324]. Our general approach is well-aligned with the current understanding in this direction. That is, while general disentanglement (per-dimensional disentanglement) is hard to achieve without stringent assumptions on the data generation process, we can seek some certain level of disentanglement (in our case, the group-wise disentanglement between the static and dynamic factors) given auxiliary information (which we provide through augmentations). Our intuitive arguments for the reason of separation (see the last paragraph of Sec 3.2.2) show that VAE and contrastive learning may be complementary to each other, and combining them could potentially gain better disentanglement.

Methods	Acc↑	IS↑	H($y|x$)↓	H(y)↑
MoCoGAN	92.89%	8.461	0.090	2.192
DSVAE	90.73%	8.384	0.072	2.192
R-WAE	98.98%	8.516	0.055	2.197
S3VAE	99.49%	8.637	0.041	2.197
C-DSVAE	**99.99%**	**8.871**	**0.014**	**2.197**

Table 3.9: Disentanglement metrics on Sprites.

Methods	Acc↑	IS↑	H($y|x$)↓	H(y)↑
MoCoGAN	63.12%	4.332	0.183	1.721
DSVAE	54.29%	3.608	0.374	1.657
R-WAE	71.25%	5.149	0.131	1.771
S3VAE	70.51%	5.136	0.135	1.760
C-DSVAE	**81.16%**	**5.341**	**0.092**	**1.775**

Table 3.10: Disentanglement metrics on MUG.

3.2.4 Experiments on Videos and Audios

We compare C-DSVAE with the state-of-the-art sequence disentanglement learning methods: FHVAE [284], MoCoGAN [337], DSVAE [285], S3VAE [327] and R-WAE [316], with the same experimental setups used by them.

Datasets

Sprites [342] is a cartoon character video dataset. Each character's (dynamic) motion can be categorized into three actions (walking, spellcasting, slashing) and three directions (left, front, right). The (static) content comprises of each character's skin color, tops color, pants color and hair color. Each color has six variants. Every sequence is composed of 8 frames of RGB images with size 64×64.

MUG [343] is a facial expression video dataset. The static factor corresponds to a single person's identity. Each individual performs six expressions (motion): anger, fear, disgust, happiness, sadness and surprise. Following [337], each sequence contains 15 frames of RGB images with size 64×64 (after resizing).

SM-MNIST ([344], Stochastic Moving MNIST) is a dataset that records the random movements of two digits. Each sequence contains 15 gray scale images of size 64×64. Individual digits are collected from MNIST.

TIMIT [345] is a corpus of read speech for acoustic-phonetic studies and speech recognition. The utterances are produced by American speakers of eight major dialects reading phonetically rich sentences. Following [284, 316], we extract per-frame spectrogram features (with a shift size of 10ms) from audio, and segments of 200ms duration (20 frames) are chunked from the original utterances and then treated as independent sequences for learning. All datasets are separated into training, validation and testing splits following [284, 327, 316].

Disentanglement Metrics

The quantitative performance measures all the models from two aspects: first, by fixing either the static or dynamic factors and randomly sample the other, how well the fixed factor can be recognized; and second, with the randomly sampled factor, how different the generated sequence is from the original one. To this end, we pretrain a separate classifier C to identify the static or dynamic factors (if available). The classifier C is carefully trained with the full supervision and thus qualifies as a judge.

We use five metrics to evaluate the disentanglement performance: accuracy,

Methods	Acc↑	IS↑	H($y|x$)↓	H(y)↑
MoCoGAN	74.55%	4.078	0.194	0.191
DSVAE	88.19%	6.210	0.185	2.011
R-WAE	94.65%	6.940	0.163	2.147
S3VAE	95.09%	7.072	0.150	2.106
C-DSVAE	**97.84%**	**7.163**	**0.145**	**2.176**

Table 3.11: Disentanglement metrics on SM-MNIST.

Methods	content EER↓	motion EER↑
FHVAE	5.06%	22.77%
DSVAE	5.64%	19.20%
R-WAE	4.73%	23.41%
S3VAE	5.02%	25.51%
C-DSVAE	**4.03%**	**31.81%**

Table 3.12: Disentanglement metrics on TIMIT.

Inception Score, inter-entropy, intra-entropy and equal error rate. Accuracy (Acc) measures how well the fixed factor can be identified by the classifier C.

Accuracy (Acc) measures how well the fixed factor can be identified by the classifier C. Given an input sequence $x_{1:T}$, the encoder would produce $z_{1:T}$ and s. If we randomly sample from the prior instead of the posterior of s for decoding/generation, the classifier C should still recognize the same motion captured by $z_{1:T}$ while identifying different contents induced by the random s. For example, in the Sprites dataset, if we randomly sample s from the prior and fix $z_{1:T}$, we should see the generated characters with different colors performing the same action in the same direction.

Inception Score (IS) computes the KL-divergence between the conditional predicted label distribution $p(y|x_{1:T})$ and marginal predicted label distribution $p(y)$ from C, $IS = \exp(\mathbb{E}_{p(x)}[KL[p(y|x)||p(y)]])$. y is the predicted attribute such as color,

action, expression, etc. $p(y|x_{1:T})$ is usually taken from the logits after the softmax layer. $p(y)$ is the marginalization across all the test samples, $p(y) = \frac{1}{N}\sum_{i=1}^{N} p(y|x)$. Since we hope the generated samples to be as diverse as possible, IS is expected to be high.

Inter-Entropy is similar to IS but measures the diversity solely on the marginal predicted label distribution $p(y)$, $H(y) = -\sum_y p(y)\log p(y)$. The higher $H(y)$ is, the more diverse generated sequences would be.

Intra-Entropy measures the entropy over $p(y|x)$, $H(y|x) = -\sum_y p(y|x)\log p(y|x)$. Lower intra-entropy means more confident predictions.

Equal Error Rate (EER) is only used in the TIMIT experiments for audio evaluation. It means the common value when the false rejection rate is equal to the false acceptance rate.

Hyperparameters and Architectures

As is commonly done in VAE learning [318], in (3.12) we add coefficients α to the KL terms, β to the contrastive terms, while the coefficient γ of $I(s; z_{1:T})$ is fixed to be 1. In our experiments, α is tuned over $\{0.6, 0.9, 1.0, 2.0\}$ and β is tuned over $\{0.1, 0.2, 0.5, 0.7, 1.0, 2.0, 5.0\}$. We use the Adam optimizer [186] with the learning rate chosen from $\{0.0005, 0.001, 0.0015, 0.002\}$ through grid search. Models are trained until convergence and the one with the best validation accuracy would be chosen for testing. Our network architecture follows [327]: motion priors and posteriors are both parameterized by uni-directional LSTMs, which output μ and σ at each time step for the dynamic factors, while the content posterior produces μ and σ for the static factor of the whole sequence with a bi-directional LSTM.

Quantitative Results

Tables 3.9, 3.10, 3.11, 3.12 demonstrate the disentanglement evaluations on the datasets; ↑ means the higher the value the better the performance, and ↓ means the reverse. In Table 3.9, we fix $z_{1:T}$ and randomly sample s from $p(s)$ for the evaluation. We observe that R-WAE, S3VAE, C-DSVAE can all deliver high accuracies in predicting a total of 9 classes (3 actions $\times$ 3 directions), but C-DSVAE outperforms all the others. On the other hand, when we fix s and randomly sample $z_{1:T}$, all the methods achieve near-perfect accuracies for predicting character identities (and thus we do not show this result here). Similarly, in Table 3.10, we evaluate the ability to maintain the facial expression by sampling s and fixing $z_{1:T}$. C-DSVAE outperforms R-WAE and S3VAE by over 10% relatively in the prediction accuracy, and its intra-entropy is lower by 30% relatively. Table 3.11 measures how well the static factor s, which corresponds to one of the total 100 digit combinations, can be recognized when the dynamic factors are randomly sampled (the dynamic factors are continuous and thus hard to be categorized). While R-WAE, S3VAE, and C-DSVAE can all generate high-quality moving digits (see Sec 3.2.4), C-DSVAE non-trivially outperforms the others on all metrics.

For TIMIT, we extract both the content factor (s, corresponds to the speaker identity) and the dynamic factors ($z_{1:T}$, correspond to the linguistic content) for the sequences. And for each of them, we compute the cosine similarity of representations between pairs of sequences, based on which we perform thresholding to classify if the two sequences are produced by the same speaker. Through varying the threshold, we compute the EER for the speaker verification task [284, 285]. For a well disentangled latent space, we expect the EER on s (content EER) to be low, while the EER on $z_{1:T}$ (motion EER) to be high. The EER

Figure 3.9: Manipulate the static and dynamic factors on Sprites. Row 1, 2, 4 are the raw test input sequences, while row 3, 5 are manipulated generations. **Left panel**: Row 3 uses s from row 1 and $z_{1:T}$ from row 2. Row 5 uses s from row 1 and $z_{1:T}$ from row 4. **Right panel**: Row 3 uses $z_{1:T}$ from row 1 and s from row 2. Row 5 uses $z_{1:T}$ from row 1 and s from row 4.

Figure 3.10: Row 1, 2, 4, 6, 8 are the raw test input sequences. **Left:** For row i (i is odd and $i > 1$), s is set to be the same as the s from row 1, while $z_{1:T}$ of each row is retained. **Right:** For row i (i is odd and $i > 1$), $z_{1:T}$ is set to be the same as the $z_{1:T}$ from row 1, while s of each row is retained.

results are given in Table 3.12. Compared with the previous SOTA by R-WAE, C-DSVAE reduces the content EER by over 10% relatively.

Figure 3.11: The random generations of contents and motions. **Left:** $z_{1:T}$ from row 1 is fixed and s is sampled from $p(s)$ for other rows. **Right:** s from row 1 is fixed and $z_{1:T}$ is sampled from $p(z_{1:T})$ for other rows.

(a) C-DSVAE (b) S3VAE

Figure 3.12: Fix the facial expression on MUG. Row 1, 2, 4 are the raw test sequences with different facial expressions. We replace $z_{1:T}$ of row 2, 4 with $z_{1:T}$ of row 1 and compare the performance of C-DSVAE and S3VAE. Hence row 3, 5 are expected to display happiness as well. Our C-DSVAE's generations are clean and sharp, while S3VAE produces blurred sequences (zoom in to see the details).

Qualitative Results

We now compare different methods qualitatively with the following tasks: given the extracted static factor s and dynamic factors $z_{1:T}$, we fix one of them and manipulate the other to see if we can observe the desired variations in the generated sequences.

Sprites Figure 3.9 gives example generations for Sprites. We observe that the character identities are well preserved when we fix s (left) and the motion is well

(a) Fix the content. Row 1, 3, 5, 7 use the same s. Meanwhile, they also take $z_{1:T}$ from the previous row for generation. The person identity is well-preserved with different expressions.

(b) Fix the motion. Row 1, 3, 5, 7 use the same $z_{1:T}$. Meanwhile, they also take s from the previous row for generation. All the faces have the same expression and the identity is consistent with the previous row.

Figure 3.13: Fix either the static or dynamic factors and replace the other.

maintained when we fix $z_{1:T}$ (right), demonstrating the clean disentanglement by C-DSVAE. We show more experiments of fixing one factor and replacing the other on Sprites in Figure 3.10. In addition, Figure 3.11 shows the generated sequences when the factors are sampled from the priors (either static or dynamic factors).

(a) C-DSVAE (b) R-WAE

Figure 3.14: Fix the movement and replace the digits. Row 1, 2, 4 are input test sequences with different digits and movements. Row 3, 5 use $z_{1:T}$ from row 1 while retaining their own s. The sequences generated by C-DSVAE are clean and consistent, while R-WAE sometimes produces blurred or incorrect digits.

MUG Figure 3.12 compares the generated sequences when the motion (facial expression) is fixed for C-DSVAE and S3VAE on MUG. We replace $z_{1:T}$ for each input sequence with the one from the first row, and therefore the generated sequences are expected to have the smiling expression. C-DSVAE and S3VAE both generate recognizable smiling faces and preserve the individual's identity, but the sequences generated by S3VAE have blurs and are less sharp. Figure 3.13 shows the generated sequences when we fix either the motion or content factors. In Figure 3.13a, the person identity is consistent with row 1 but has the same expression as the previous row. In Figure 3.13b, the person identity is consistent with the previous row but all with the same expression as row 1.

SM-MNIST In Figure 3.14, we fix the digit movements and take the content (digit identities) from various other sequences. Entangled factors might cause the extracted $z_{1:T}$ to carry information of s. Row 3 of Figure 3.14b shows such an example produced by R-WAE where the motion factor of one digit carries the content information of "9" from row 1.

Figure 3.15: Generate different contents. For every two rows separated by the red lines, the first row is from the test set and the second row inherits the dynamic factors $z_{1:T}$ from the first row, while samples s from prior $p(s)$. As a result, the same motion is preserved with different digits.

Note that S3VAE and R-WAE are both capable of generating recognizable videos, which is validated by our classifier C with good accuracies. Nevertheless, the results here show that C-DSVAE can learn better disentangled factors for the high-quality generation.

Figure 3.15 generates random digits from $p(s)$ to replace the content from the raw test input sequences. The motions are well-preserved while the contents are totally different.

On the other hand, Figure 3.16 preserves the content digits and randomly sample $p(z_{1:T})$. The motions of the digits then become different.

Another interesting generation task is swapping, as shown in Figure 3.17. Given two input sequences, we exchange their dynamic and static factors.

Figure 3.16: Generate different motions. For every two rows separated by the red lines, the first row is from the test set and the second row inherits the static factor s. $z_{1:T}$ is sampled from the prior $p(z_{1:T})$. As a result, the same digits are preserved with different movements.

Ablation Study on Estimation Methods

Methods	Acc↑	IS↑	H(y|x)↓	H(y)↑
DSVAE	54.29%	3.608	0.374	1.657
DSVAE+all MWS est	66.25%	4.796	0.175	1.743
C-DSVAE	81.16%	5.341	0.092	1.775

Table 3.13: Compare MWS and contrastive estimation of $I(s; x_{1:T})$ and $I(z_{1:T}; x_{1:T})$ on MUG. "all MWS est" means that C-DSVAE optimizes MI terms all estimated by MWS rather than the contrastive estimation.

Our C-DSVAE estimates $I(s; x_{1:T})$, $I(z_{1:T}; x_{1:T})$ with contrastive learning and $I(s; z_{1:T})$ with MWS. To further demonstrate the advantage of the contrastive estimation, we compare it with the model with MWS estimations on all the MI terms including $I(s; x_{1:T})$, $I(z_{1:T}; x_{1:T})$. Table 3.13 and 3.14 give the observations. With all MWS estimations, the performance would drop heavily.

These results help support the claim that the inductive biases brought by

Figure 3.17: Swap the static and dynamic factors. For every 4 rows separated by the red line, row 1 and row 3 are the raw test sequences with different contents and motions. Row 2 takes s from row 3 and $z_{1:T}$ from row 1. Row 4 takes s from row 1 and $z_{1:T}$ from row 3. We present 2 such swapping sets: row 1~4, row 5~8.

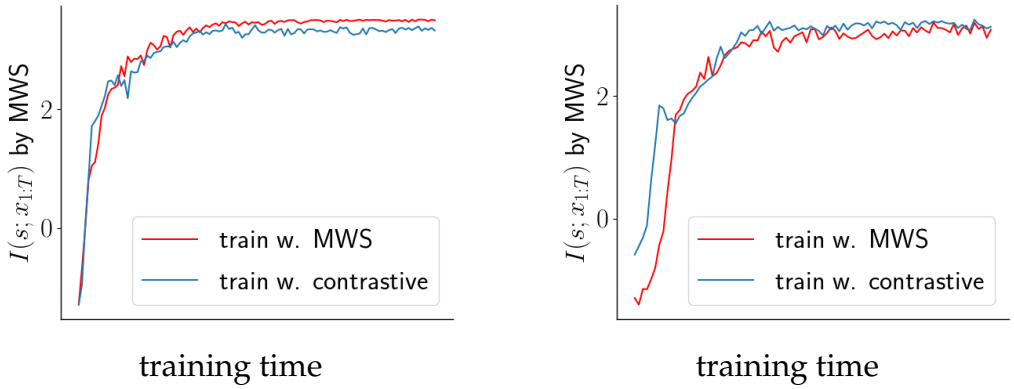


Figure 3.18: Learning curves of the MI terms (estimated by MWS) between latent factors and the input sequences, when the training uses either MWS or the contrastive estimation for the MI terms. Though learning with different estimation methods, the MI curves for the two estimation methods are close to each other. **Left**: Sprites. **Right**: TIMIT.

contrastive learning might contribute more to the good performance than the MI estimation. Also note that contrastive learning works well when the goal of learning is to maintain some invariance between different views. For other MI estimation tasks, contrastive learning might not be the best option.

Figure 3.19: Interpolation in the latent space. For every 5 rows separated by the red line, the first and last rows have different contents s, but share the same motion. The 3 rows in between keep the same motion but their s's linearly interpolate between row 1 and row 5. One can observe that the content transition is smooth, while the motion is intact.

Methods	Acc↑	IS↑	H($y|x$)↓	H(y)↑
DSVAE	88.19%	6.210	0.185	2.011
DSVAE+all MWS est	91.81%	6.312	0.205	2.107
C-DSVAE	97.84%	7.163	0.145	2.176

Table 3.14: Compare MWS and contrastive estimations of $I(s; x_{1:T})$ and $I(z_{1:T}; x_{1:T})$ on SM-MNIST.

Interpolation in Latent Space

To further show our learnt latent space is smooth and meaningful, we linearly interpolate between 2 content factors corresponding to different digit pairs and generate the sequences. In Figure 3.19, one can see that the transition from one content to another is smooth. In the first example, ("5", "4") gradually changes to ("8", "8"). In the second example, ("0", "2") gradually changes to ("9", "1").

batch size	content EER↓	motion EER↑
64	4.21%	30.23%
128	4.07%	31.42%
256	4.03%	31.81%

Table 3.15: Performance with different batch sizes on TIMIT.

While "2" transforms to "1", it first becomes "7" and its font gets slimmer.

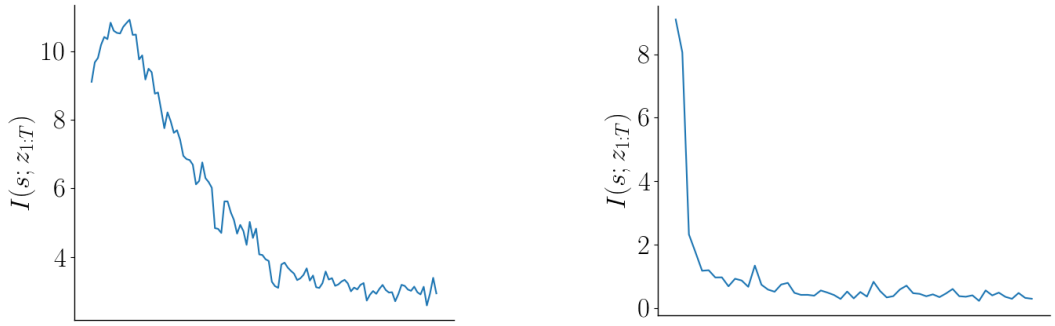
Sensitivity Analysis on Batch Size

In Table 3.15, we show how different batch sizes would affect the evaluation performance on TIMIT. Batch size 256 gives the best numbers.

Other analysis

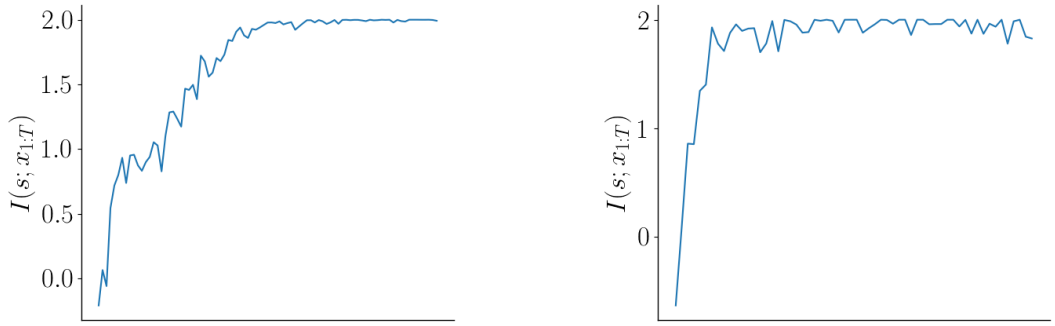
As mentioned in Sec 3.2.2, we attempt to estimate the MI terms using either MWS or the contrastive estimation with augmentations. In Figure 3.18, we show the MWS estimation of $I(s; x_{1:T})$ improves when the C-DSVAE training objective optimizes the contrastive estimation of $I(s; x_{1:T})$. This means the contrastive estimation could be a surrogate for MI in generative models, and its additional inductive bias can be a main contributor to the superior performance (see section 3.2.4 for quantitative comparisons).

Regarding the augmentations, it is intuitive to adopt both the content augmentation and the motion augmentation. Having both of them can not only boost the learning of the dynamic or static factors, but also improves the disentanglement. Adding only one of them could lead to unbalanced or entangled representation learning. As shown in Table 3.16, content augmentation alone can improve the



(a) Sprites (b) TIMIT

Figure 3.20: The mutual information $I(s; z_{1:T})$ estimated by MWS decreases during the training even if we don't include $I(s; z_{1:T})$ in the objective.



(a) SM-MNIST (b) MUG

Figure 3.21: The mutual information $I(s, x_{1:T})$ estimated by MWS increases during the training, demonstrating the effectiveness of the contrastive estimation.

accuracy by 3.8%, and motion augmentation alone can improve the accuracy by 7.1%. Two augmentations together can bring a gain of 9.1%. Similarly, in Table 3.17, though adding only content augmentation or motion augmentation can outperform DSVAE, combining them leads to the largest gain.

Figure 3.20a demonstrates the curve of MI $I(s; z_{1:T})$ during training on Sprites when we don't include $I(s; z_{1:T})$ in the objective function. We estimate $I(s; z_{1:T})$ using the Minibatch Weighted Sampling (MWS) estimation (though not opti-

Table 3.16: Ablation study of the augmentations on SMMNIST.

Methods	Acc↑	IS↑	H($y|x$)↓	H(y)↑
DSVAE	88.19%	6.210	0.185	2.011
C-DSVAE w/o content aug	92.02%	6.360	0.184	2.083
C-DSVAE w/o motion aug	95.25%	6.471	0.175	2.093
C-DSVAE	97.84%	7.163	0.145	2.176

Table 3.17: Ablation study of the augmentations on TIMIT.

Methods	content EER↓	motion EER↑
DSVAE	5.64%	19.20%
C-DSVAE w/o content aug	5.09%	24.30%
C-DSVAE w/o motion aug	4.31%	31.09%
C-DSVAE	4.03%	31.81%

mized directly). The curve increases in the early stage of training, implying that while learning useful representations for reconstruction, disentanglement is compromised in the early stage. But as the training continues, the model picks up the disentanglement terms besides the reconstruction. Figure 3.20b demonstrates the same experiment on another dataset TIMIT.

We show in Figure 3.21 that, the contrastive estimation could boost $I(s; x_{1:T})$ during training. Our training process optimizes the contrastive estimation, but as we can see in the figure, the MI term $I(s; x_{1:T})$ estimated by MWS also increases accordingly.

3.2.5 Discussion

In this work, we introduce Contrastively Disentangled Sequential Variational Autoencoder (C-DSVAE) to learn disentangled static and dynamic latent factors for sequence data without external supervision. Our learning objective is a

novel ELBO derived differently from prior works, and naturally encourages disentanglement. C-DSVAE uses contrastive estimations of the MI terms to further inject the inductive biases. Our method achieves the state-of-the-art performance on multiple datasets, in terms of the disentanglment metrics and the generation quality. In the future, we plan to extend C-DSVAE to other domains, such as text, biology, agriculture and weather prediction. We will also improve our model's ability to capture long-range dependencies.

3.3 Self-supervision with Supervision: Joint Unsupervised and Supervised Training for Multilingual ASR

3.3.1 Introduction

Self-supervised learning is an effective method in unveiling the useful and general latent representations from large-scale unlabeled data. It is often adopted to pretrain a sequence-to-sequence model and facilitate downstream tasks [91, 123, 346]. In speech recognition, recent works have shown successes of the two-stage pretrain-finetune schemes [237, 347, 348, 349, 350]. Pretrained models can greatly reduce the sample complexity for downstream finetuning. For instance, finetuning wav2vec 2.0 (w2v2) pretrained on 60k hours with only 1h labeled data can outperform most fully supervised models [37].

While self-supervised learning has been successful for sequence modeling, some concerns have also been raised. For example, finetuning a pretrained model is prone to catastrophic forgetting [140, 351]. The model might *forget* the previ-

ously learnt knowledge when trained with supervision, particularly when the supervised set is large. Another concern is the pretrained checkpoint selection. The downstream performance varies from one checkpoint to another, and the one pretrained longer may not be the best one. These issues are even more severe in multilingual ASR, since different languages are often heterogeneous and the corpus is often imbalanced. In *Multilingual LibriSpeech* (MLS) [121], English has up to 44k hours while Polish only has 100 hours. Most existing methods tackle multilingual ASR from 2 directions. The first direction is transfer learning from a source multilingual corpus to a target low-resource multilingual dataset. In [7], the work first trains the model on Google's 15-language VoiceSearch (VS) traffic and then uses it to seed the transfer learning on MLS. Even though some languages in MLS are not included in VS, the model can deliver satisfactory WERs on those low-resource languages, demonstrating its generalization capability. However, such transfer learning requires massive supervised source corpora which may not be easily accessible. Another direction is to learn useful representations through pretraining and perform finetuning with supervision, similar to monolingual ASR. [352] explores the unsupervised pretraining using cross-lingual language modeling and [353] investigates the cross-lingual transfer of phoneme features. XLSR [354] builds on w2v2 and pretrains the model on 53 languages using the self-supervised losses. XLSR also stands for the state-of-the-art (SOTA) on MLS dataset.

In this work, we introduce a novel Joint Unsupervised and Supervised Training (JUST) method for multilingual ASR, to reconcile the unsupervised and supervised losses synergistically. JUST includes two self-supervised losses, contrastive loss [37] and MLM loss [91], together with a supervised RNN-T loss [355]. Our model architecture inherits from w2v-bert [349], a novel variation

of w2v2. The outputs from w2v-bert are passed to the decoder and produce the RNN-T loss. We explore two types of learning with JUST: 1) JUST trained from scratch, and 2) JUST finetuned from a pretrained checkpoint. We compare these 2 settings with XLSR and other standard baselines. Experiments show that JUST can consistently outperform other SOTA and baselines. For instance, on 8 languages from MLS, JUST improves over XLSR by 30% on average.

3.3.2 Related Work

Early works adopted joint training to learn robust and transferable representations. In NLP, [356] proposes joint training for machine translation. [357] suggests multiple pretraining objectives for domain-adaptive applications. In speech, PASE [268] jointly solves multiple self-supervised tasks to learn general representations. More recent research found the joint training with both supervised and unsupervised losses can directly optimize the ASR performance. [358] alternatively minimizes an unsupervised masked CPC loss and a supervised CTC loss [32]. This single-stage method is shown to match the performance of the two-stage w2v2 on the Librispeech 100-hours dataset. Similarly, UniSpeech [359] optimizes a combination of phonetic CTC loss and contrastive loss. To further increase the quantizer codebook usage, UniSpeech randomly replaces contextual representations with quantized latent codes. [360] also designs a similar hybrid multitask learning to train acoustic models under low-resource settings, comprising of supervised CTC, attention and self-supervised reconstruction losses. Similarly, [361] combines self- and semi-supervised learning methods for online ASR model. These methods only contain one self-supervised loss in their optimization and often tackle with speech recognition in the phoneme level

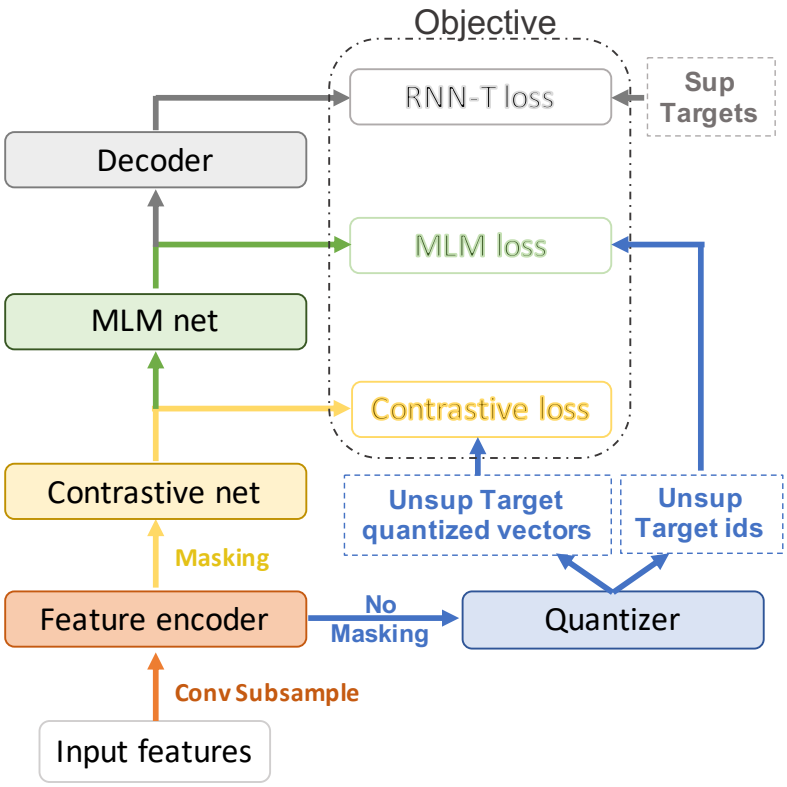


Figure 3.22: An overview of our JUST framework. Feature encoder, contrastive net, MLM net and decoder are stacked sequentially. The output of each module constitutes a loss in the objective function. Target vectors and ids in the blue boxes are for unsupervised losses. Supervised targets in the grey box are for RNN-T loss.

[359, 360]. JUST incorporates two self-supervised losses (contrastive and MLM losses), and replaces the CTC loss with an RNN-T loss. RNN-T extends CTC with a prediction network to simulate the effect of LM and has been widely adopted in prior multilingual ASR systems [7]. Furthermore, unlike [268, 360] where each of the multiple tasks has its own branch, JUST computes different losses simply using the intermediate outputs from different layers (Fig. 3.22).

3.3.3 JUST Method for Multilingual ASR

Our JUST framework is comprised of multiple modules for unsupervised and supervised losses. All modules (except for the quantizer) are stacked sequentially and each reads the output from the previous module. We will elaborate on them, along with the losses, in the following sections. Fig. 3.22 presents an overview.

Feature encoder

The feature encoder converts the original log-mel filter bank features $\{x_i\}_{i=1}^{L}$ to the latent speech representations $\{z_i\}_{i=1}^{T}$ for T time steps. T is smaller than the original length L due to time reduction. Unlike [37] where seven blocks of CNN are used, JUST only has two CNN blocks both with filter size 3×3 and strides (2, 2), the same as [347]. One can also view the feature encoder as a convolutional subsampling, with 4x reduction in the feature dimensionality and sequence length.

Quantizer

JUST adopts a complex quantization mechanism [37]. After the abstraction of the original inputs through feature encoder, the latent representations $\{z_i\}_{i=1}^{T}$ are passed to a quantizer (without any masking). The goal of the quantizer is to "summarize" all the latent speech representations to a finite set (referred to as a codebook) of representative discriminative speech tokens $\{e_j\}_{j=1}^{V}$ where V is the size of the codebook. The codebook in the quantizer stores all these tokens and each latent representation from the feature encoder is mapped to a token index corresponding to a token in the codebook, through Gumbel softmax [362]

which enables differentiation of discrete codebook selection. JUST uses a single codebook rather than multiple ones [37]. All the tokens in the codebook are learnable during training. Quantizer module generates target quantized vector (token) q_i and target id (token index) y_i for each z_i, where $q_i \in \{e_j\}_{j=1}^{V}$, $y_i \in [1...V]$. To encourage the use of the codebook, [37] introduces the entropy-based diversity loss $\mathcal{L}_d$. We include it in JUST as well.

Contrastive net

The outputs of feature encoder $\{z_i\}_{i=1}^{T}$ are not only used for quantization, but also fed into the contrastive net after masking. For masking, some z_i's are randomly chosen and replaced with random vectors. Contrastive net reads z_i of all time steps (either masked or unmasked) and outputs contrastive context vectors $\{c_i\}_{i=1}^{T}$ for deriving contrastive self-supervised loss. Contrastive net is a stack of Conformer blocks [87], each with multi-headed self-attention, depth-wise convolution and feed-forward layers. To derive the contrastive loss $\mathcal{L}_c$, for anchor c_i, we take q_i as the positive sample and K negative samples/distractors $\{\tilde{q}_i\}_{i=1}^{K}$ uniformly sampled from q_j of other masked z_j's in the same utterance:

$$\mathcal{L}_c = -\log \frac{sim(c_i, q_i)}{sim(c_i, q_i) + \sum_{j=1}^{K} sim(c_i, \tilde{q}_j)} \tag{3.15}$$

where $sim(a, b)$ is the exponential of the cosine similarity between a and b.

MLM net

We further boost the contextualized representation learning through a masked prediction task with the quantizer. The inputs of MLM net are $\{c_i\}_{i=1}^{T}$ from contrastive net. Similar to contrastive net, MLM net is also a stack of Conformer

Method	External data	**en**	**de**	**nl**	**fr**	**es**	**it**	**pt**	**pl**	Avg	Avg (w/o **en**)
Monolingual [121]	-	6.76	7.10	13.09	6.58	6.68	11.78	20.52	21.66	11.8	12.5
+ 5-gram LM [121]	-	5.88	6.49	12.02	5.58	6.07	10.54	19.49	20.39	10.8	11.5
XLSR-53 [354]	Y	-	7.0	10.8	7.6	6.3	10.4	14.7	17.2	10.6	10.6
B0 (random init.) [7]	Y	6.1	5.5	11.9	6.9	5.8	11.9	16.2	15.4	10.0	10.5
B0 (15-language model init.) [7]	Y	6.6	5.0	11.1	6.1	4.7	10.1	15.5	10.9	8.8	9.1
E3 (15-language model init.) [7]	Y	**5.8**	4.3	9.9	**4.9**	4.2	8.8	15.2	10.4	7.9	8.2
JUST ($\beta = 0$)	N	6.9	5.5	10.3	6.0	4.1	9.3	9.4	11.3	7.8	8.0
w2v2 Pretrain ($\mathcal{L}_c$+$\alpha\mathcal{L}_d$) + pure Finetune ($\mathcal{L}_s$)	N	6.8	4.7	10.3	5.8	4.1	9.9	12.6	12.1	8.3	8.5
w2v-bert Pretrain ($\mathcal{L}_u$) + pure Finetune ($\mathcal{L}_s$)	N	6.6	4.3	9.9	5.0	3.8	9.1	14.6	8.1	7.7	7.8
w2v-bert Pretrain ($\mathcal{L}_u$) + JUST Finetune ($\mathcal{L}$)	N	6.6	4.2	9.5	5.0	4.0	9.0	15.1	7.6	7.6	7.8
w2v2 Joint Training ($\mathcal{L}_s$+β($\mathcal{L}_c$+$\alpha\mathcal{L}_d$))	N	6.7	4.6	9.9	5.7	4.1	8.9	9.3	9.8	7.4	7.5
JUST ($\mathcal{L}$)	N	6.8	4.6	9.9	5.7	3.9	9.1	8.6	9.1	7.2	7.3
JUST ($\mathcal{L}$) + pure Finetune ($\mathcal{L}_s$)	N	6.5	**4.1**	**9.5**	5.2	**3.7**	**8.2**	**8.0**	**6.6**	**6.5**	**6.5**

Table 3.18: WER(%) results on MLS for different methods. JUST-based methods greatly outperforms the compared baselines. XLSR-53 used external unsupervised data for pretraining. B0 and E3 used external supervised data. Our JUST did not use any external data.

blocks. We denote the outputs of MLM net as $\{m_i\}_{i=1}^{T}$, which are high-level context vectors. Each m_i is used for token id prediction through a linear layer. The predicted id $\hat{y}_i \in [1..V]$ is compared with the target token id y_i from the quantizer, by the standard cross-entropy loss $\mathcal{L}_m$.

Together with $\mathcal{L}_d$ and $\mathcal{L}_c$, the unsupervised loss is computed as:

$$\mathcal{L}_u = \mathcal{L}_c + \mathcal{L}_m + \alpha\mathcal{L}_d \tag{3.16}$$

α is set to 0.1, following [349].

Decoder

The decoder of JUST is a 2-layer RNN Transducer. $\{m_i\}_{i=1}^{T}$ are passed through Swish activation, batch normalization, and finally fed into the decoder. The

Method	Ext.	**en**	**de**	**nl**	**fr**	**es**	**it**	**pt**	**pl**	Avg	Avg (w/o **en**)
JUST ($\beta = 0$)	N	6.9	5.5	10.3	6.0	4.1	9.3	9.4	11.3	7.8	8.0
JUST ($\beta = 0.03$)	N	7.4	5.0	10.3	6.3	4.3	9.3	9.1	8.7	7.5	7.6
JUST ($\beta = 0.05$)	N	6.8	5.2	9.9	5.7	4.4	9.4	8.8	9.3	7.4	7.5
JUST ($\beta = 0.07$)	N	6.8	4.6	9.9	5.7	3.9	9.1	8.6	9.1	7.2	7.3
JUST ($\beta = 0.1$)	N	6.8	5.8	10.0	5.8	4.1	10.3	8.6	9.7	7.6	7.8

Table 3.19: Weight sensitivity study on β. When $\beta = 0.07$, the unsupervised loss is roughly the same as the supervised loss, which means balancing the unsupervised and supervised losses can be critical in joint training.

output vocabulary of the decoder is a unified grapheme set pooled from all the 8 languages in MLS. RNN-T loss is used in this work as the supervised loss, denoted by $\mathcal{L}_s$. Our final objective function is simply the combination of $\mathcal{L}_u$ and $\mathcal{L}_s$:

$$\mathcal{L} = \mathcal{L}_s + \beta \mathcal{L}_u \tag{3.17}$$

β is a trade-off weight. $\mathcal{L}$ is optimized via Adam [186].

3.3.4 Experiments

Dataset

MLS dataset [121] is used as the benchmark in our experiments. It is derived from read audiobooks of LibriVox. There are 8 languages (namely English (**en**), German (**de**), Dutch (**nl**), French (**fr**), Spanish (**es**), Italian (**it**), Portuguese (**pt**) and Polish (**pl**)), with 44.5k hours of English and 6k hours for other languages combined. Some low-resource language like Polish only has 100 hours. Each utterance is 10-20 seconds long.

Training details

Architecture The inputs are 80-d filter bank features. Feature encoder has 2 convolutional layers with filter size (3,3), strides (2,2). The two layers have 128 and 32 channels respectively. Contrastive net consists of 8 Conformer blocks, each with hidden dimensionality 1024, 8 attention heads and convolution kernel size 5. MLM net consists of 16 Conformer blocks with the same configuration. Our decoder uses a 2-layer 768-d LSTM-based RNN-T with 3072 hidden units. $\{c_i\}_{i=1}^{T}$ from contrastive net and $\{m_i\}_{i=1}^{T}$ from MLM net are used in computing self-supervised losses after layer normalization. Our codebook has size V=1024, with each token of length 1024.

Masking To mask $\{z_i\}_{i=1}^{T}$, we randomly sample 6.5% of all time steps and replace each of the selected time steps and its subsequent 10 time steps with random normal vectors (from $\mathcal{N}(0, 0.1)$). Some spans might overlap.

Hyperparameters We train JUST with batch size 1024 on 64 TPUs. Adam optimizer is employed with $\beta_1 = 0.9, \beta_2 = 0.98$ for training. Our global learning rate schedule is the same as [349] but with warm-up steps 5000 and peak learning rate 4e-4. The decoder uses a separate schedule rather than the global one, with 1500 warm-up steps and peak learning rate 7e-4. We set $\alpha = 0.1$ following prior works [347, 349] and $\beta = 0.07$ via tuning with grid search.

Evaluation We show the WER for each language, as well as the average WER with or without **en** included.

Compared methods

We compare JUST with several baselines. MLS paper [121] provides competitive monolingual baselines without any LM and with a 5-gram LM. Using LM improves the monolingual performance. XLSR [354] pretrains a w2v2 on 53 languages from MLS, CommonVoice and BABEL, and finetunes the model on MLS. XLSR finetuned on the full set of MLS can outperform some low-resource monolingual baselines like **it**, **pt**, **pl**, but not all (Table 3.18). We also include transfer learning models, B0 and E3, from [7], which used heavy supervision from external *VoiceSearch* (VS) dataset containing 15 languages. Both B0 and E3 are first trained on VS with supervision and then finetuned on MLS. B0 is a smaller model with 370M parameters and E3 is a larger model with 1B parameters. We also include a B0 model trained from scratch for comparison. Besides these existing baselines from literature, we further train a w2v-bert model from scratch on MLS (JUST with $\beta = 0$), and a two-stage pretrain-finetune w2v-bert model on MLS without any external data.

Results

For JUST, we either train it from scratch on MLS, or jointly finetune it from a pretrained checkpoint on MLS where the pretraining phase would only optimize the unsupervised loss. Note that compared to XLSR, our pretraining would incorporate more self-supervised losses. Our JUST has 600M parameters, which is roughly the same scale as B0, XLSR but much smaller than E3.

Average WER On the average WER of all 8 languages, all JUST-based methods outperform previous works. In particular, JUST (with $\beta = 0.07$) outperforms the

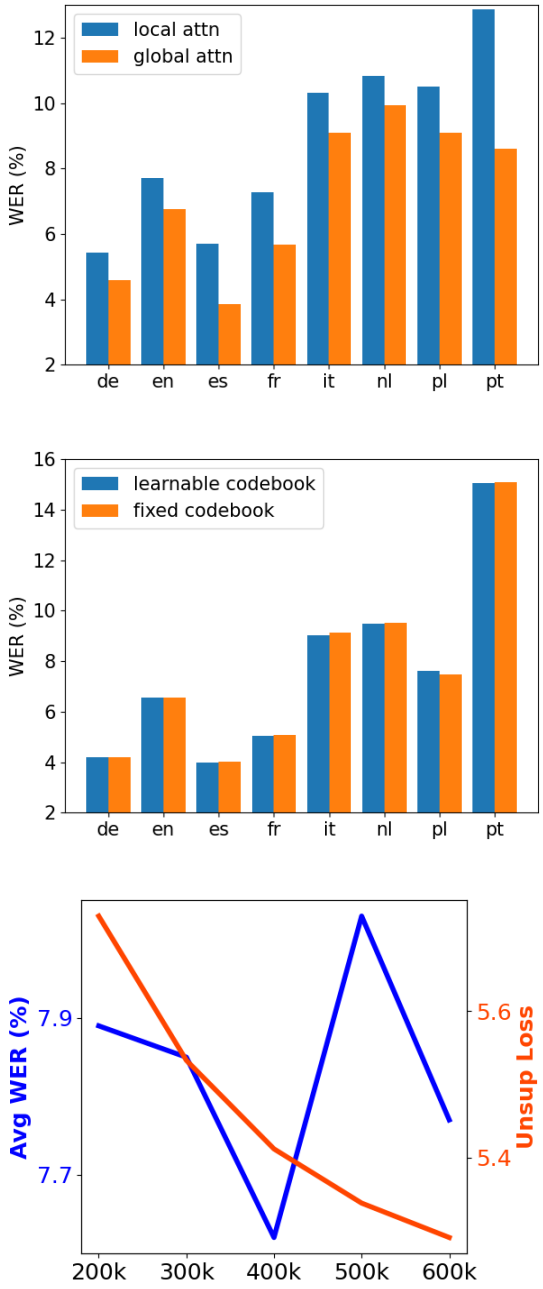


Figure 3.23: **Top:** Comparison between using local attention with left/right context size 128, and using global attention. Global attention clearly boosts the performance. **Middle:** For JUST finetuning, we either allow the codebook to be updated, or to remain fixed during the whole finetuning. They deliver similar results. **Bottom:** The average WERs of JUST finetuning ($\beta = 0.01$) from different checkpoints. The checkpoint with smaller unsupervised loss may not lead to the best finetuning results.

monolingual baseline with 5-gram LM by 33.3%, XLSR-53 by 32.0%, B0 by 18.2%, E3 by 8.8%. Note that E3 is a transfer learning method with much larger size and

heavy external supervision. JUST's improvement over E3 validates the effectiveness of our architecture and joint training scheme. If we exclude English WER and compare other languages as in XLSR [354], JUST outperforms monolingual, XLSR-53, B0, E3 by 36.5%, 31.1%, 19.8%, 11.0% respectively. Compared to JUST with $\beta = 0$, JUST with joint training improves the average WER (w/o **en**) by 7.7% (8.8%). To show the necessity of MLM loss in joint training, we further include the results of w2v2 joint training with the objective $\mathcal{L}_s+\beta(\mathcal{L}_c+\alpha\mathcal{L}_d)$. JUST still performs better on the average WERs and low-resource **es**, **pl**, **pt**.

Low-resource languages JUST improves WERs for low-resource languages such as **pl**, **pt**. On **pl**, JUST's WER is less than half of the monolingual WER baseline and roughly half of XLSR's WER. On **pt**, JUST's WER is at least 40% lower than any of XLSR, B0 or E3, which is significant.

JUST finetune Two finetuning schemes are attempted. **First**, we take a pretrained checkpoint trained with $\mathcal{L}_u$, and finetune it with JUST objective $\mathcal{L}$. Compared to w2v2 Pretrain+pure Finetune (no MLM loss), it improves on all languages except **pt**. Compared to w2v-bert Pretrain+pure Finetune (with MLM loss), it also improves on **de**, **en**, **fr**, **it**, **nl**, **pl**. It is interesting to compare JUST and JUST Finetune on **pt**, **pl**. Different training schemes lead to different quantized tokens and cause the discrepancy. Empirically, JUST from scratch can better facilitate the low-resource languages and reduce WER of each language to below 10. For JUST finetuning, we set $\beta = 0.01$ to de-weight the unsupervised loss instead of matching with the supervised loss. **Second**, we take a checkpoint from JUST trained from scratch, and finetune it with only supervised loss $\mathcal{L}_s$. It achieves the best average WER, further improving the average WER (w/o **en**) of JUST by 10% (11%). On **de**, **es**, **it**, **nl**, **pl**, **pt**, this scheme outperforms all compared methods

and remains competitive on other languages.

β sensitivity Table 3.19 also includes the sensitivity study on β. When $\beta = 0.07$, the unsupervised and the supervised losses are balanced, resulting in the best performance.

Attention We compare two attention mechanisms for JUST from scratch: a local attention mechanism with both left and right context 128, and a global attention mechanism with full context. The results are shown in Fig. 3.23. Global attention clearly outperforms local attention on all languages.

Codebook Original w2v2 doesn't update codebook in the finetuning phase. JUST finetuning, however, keeps the unsupervised loss and could further update the codebook. We compare the performance with learnable or fixed codebook during JUST finetuning ($\beta = 0.01$), and find their results are close (Fig. 3.19). This implies that fixing codebook in JUST finetuning would not degrade the performance. In practice, we also find larger β would bias the updates to the codebook, leading to worse results.

Pretrained checkpoints Different checkpoints can lead to different downstream performance. The later checkpoints do not necessarily lead to better downstream WERs. To verify this, we finetune multiple pretrained checkpoints and evaluate their finetuning quality. The rightmost subfigure from Fig. 3.23 shows the constantly descending unsupervised loss $\mathcal{L}_u$, while the downstream average WERs don't follow the same trend.

3.3.5 Discussion

This work introduce a novel uniform multilingual ASR system for the end-to-end speech recognition on multiple languages. Our method, JUST, is composed of a contrastive module for learning discrete speech representations and an MLM module that performs a masked language modeling task. JUST jointly optimizes the unsupervised contrastive loss and MLM loss, together with the supervised RNN-T loss. Compared to the prevalent two-stage pretrain-finetune models, JUST-based methods can guide the whole training process with the unsupervised and supervised losses jointly. JUST's performance is validated on a public multilingual ASR dataset, MLS, and outperforms the monolingual baselines, a SOTA two-stage pretrain-finetune model XLSR, and the latest transfer learning methods, proving the effectiveness of joint training. On low-resource languages, our JUST and its variants can consistently bring gains and boost performance. In the future, we will investigate how the objective function affects the codebook learning, and also explore the joint training with more languages and other unsupervised losses, as well as the tradeoff between unsupervised and supervised components.

CHAPTER 4

CONCLUSION

In this thesis, I focus on two main areas of sequence learning, *supervised sequence learning* for structured data, and *self-supervised learning* on sequence data. Both extend the frontiers of sequence learning and address real-world challenges.

First, I demonstrate sequence learning techniques for processing structured data, especially graph data, with regard to three scenarios: sequence-on-graph, graph-to-sequence, and sequence-to-graph. In **sequence-on-graph** learning, each node of the graph has dynamic changes across time, while the graph connections remain fixed. Each node thus can be viewed as a sequence on graph, and the edges connect these sequences. We propose a GNN-RNN framework to first extract the graph information and then sequentially encode the dynamics. The GNN-RNN framework is successfully applied to the county-level crop yield prediction task for most counties for the continental US. In **graph-to-sequence**, a given graph has an associated sequence property. Different graphs could have different properties. We adapt the sequence-to-sequence model for this case and introduce a fully attentional model for graph encoding and sequence decoding. The encoder learns attentions for the input nodes and the decoder is based on transformer. The novel method, Xtal2DoS, achieves state-of-the-art prediction performance on density of sates. In **sequence-to-graph**, we explore the underlying label structure behind input sequences for multi-label classification in data domains such as text and biology. Labels are represented by embeddings, and their inner-products implicitly form a latent closeness graph to capture the label correlations for prediction. These three scenarios illustrate the generalizability of supervised sequence learning for various types of problems.

The second part of my thesis discusses self-supervised sequence learning, which is suitable for settings where labeled data is scarce or unavailable. It can also complement supervised learning when labels are available but insufficient. While most existing works concentrate on pretraining with self-supervision, I show the versatility of self-supervision in latent structure enforcement, sequence disentanglement, and compatibility with supervision. For **pretraining**, we design a mutual information based self-supervised learning model, DAPC, which can recover the latent space of noisy dynamical systems, extract predictive features for forecasting tasks, and improve automatic speech recognition when used to pretrain the encoder on unlabeled data. For **disentanglement**, we illustrate that self-supervision can effectively separate the static and dynamic factors for sequence data when appropriately combined with augmentations. Our proposed disentangled sequence model stems from VAE and contrastive learning, and excels in terms of disentanglement metrics and audio/video generation quality, without the need of any supervision. Moreover, we show that the **joint** training with both **self-supervision** and **supervision** can exceed the pure supervised learning model in multilingual ASR. The resulting model, JUST, optimizes for both unsupervised and supervised loss simultaneously, to overcome the catastrophic forgetting and checkpoint selection challenges from the traditional two-stage training. JUST surpasses all the prior baselines in the latest large-scale multilingual corpus. In general, I explore the diverse uses of self-supervision and showcase its capabilities in a wide range of applications in this research direction.

In the future, I will keep exploring the possibilities of self-supervision, as well as the ways to exploit the hidden structures found in the sequence data. I hope to expand these techniques to more real-world domains including multi-modality

learning, causal inference and reinforcement learning.

My research journey was made possible through the Institute for Computational Sustainability and I want to thank again my amazing collaborators including the teams at the Joint Center for Artificial Photosynthesis at Caltech, the Cornell Lab of Ornithology, the Gulf of Maine Research Institute and the Atkinson Center for Sustainable Future. I'm also glad to see some of my research works have won paper awards, been covered by the press, and made real impacts on academy and society. All of these will motivate me and empower me with courage and confidence in the next chapter of my life.

APPENDIX A

APPENDIX FOR CHAPTER 2

A.1 Contrastive Learning Module

A.1.1 Connection with Triplet Loss

Triplet loss [363] is one of the popular ranking losses used in multi-label learning [240].

Given an anchor embedding v_x^f, a positive embedding v_+ and a negative embedding v_-, they form a triplet (v_x^f, v_+, v_-). A triplet loss is defined as

$$\begin{aligned} &\mathcal{L}_{trip}(v_x^f, v_+, v_-) \\ =&max\{0, g + dist(v_x^f, v_+) - dist(v_x^f, v_-)\} \end{aligned} \tag{A.1}$$

where g is a gap parameter measuring the distance between (v_x^f, v_+) and (v_x^f, v_-), and $dist(\cdot, \cdot)$ is a distance function. This hinge loss $\mathcal{L}_{trip}$ encourages fewer violations to "positive>negative" ranking order. Let $\tau = 1/2$. With the same triplet, we can write down a contrastive loss

$$\begin{aligned} &\mathcal{L}_{CL}(v_x^f, v_+, v_-) \\ =&-\log \frac{\exp(2 \cdot v_x^f \cdot v_+)}{\sum_{t\in\{+,-\}} \exp(2 \cdot v_x^f \cdot v_t)} \\ =&\log(1 + \frac{\exp(2 \cdot v_x^f \cdot v_-)}{\exp(2 \cdot v_x^f \cdot v_+)}) \\ \approx&1 + (2 \cdot v_x^f \cdot v_- - 2 \cdot v_x^f \cdot v_+) \\ =&1 + (-v_x^f \cdot v_x^f + 2v_x^f \cdot v_- - v_- \cdot v_- \\ &+ v_x^f \cdot v_x^f - 2 \cdot v_x^f \cdot v_+ + v_+ \cdot v_+) \\ =&\|v_x^f - v_+\|^2 + \|v_x^f - v_-\|^2 + 1 \end{aligned} \tag{A.2}$$

Note that in the second to the last equation, v_+ and v_- have the same norm due to the normalization in our contrastive learning module.

By setting $dist(\cdot,\cdot)$ to commonly used ℓ_2 distance and $g = 1$, Eq. A.2 is a fair approximation of Eq. A.1. Therefore, triplet loss can be viewed as a special case of contrastive loss. In contrastive loss, embeddings are normalized and more positives/negatives are available. As shown in [235], contrastive loss generally outperforms triplet loss.

A.1.2 Gradients of Contrastive Loss

Recall our contrastive loss:

$$\mathcal{L}_{CL} = \sum_{(x,y)\in\mathcal{B}} \frac{1}{|P(y)|} \sum_{p\in P(y)} -\log \frac{\exp(v_x^f \cdot v_p^l/\tau)}{\sum_{t\in A}\exp(v_x^f \cdot v_t^l/\tau)} \tag{A.3}$$

For the illustration purpose, we only consider one sample (x, y) instead of one batch:

$$\mathcal{L}_{CL} = \frac{1}{|P(y)|} \sum_{p\in P(y)} -\log \frac{\exp(v_x^f \cdot v_p^l/\tau)}{\sum_{t\in A}\exp(v_x^f \cdot v_t^l/\tau)} \tag{A.4}$$

Define $N(y) \equiv A \setminus P(y)$. We now derive the gradients w.r.t. v_x^f.

$$\begin{aligned}
\frac{\partial \mathcal{L}_{CL}}{\partial v_x^f} =& \frac{1}{\tau|P(y)|} \sum_{p\in P(y)} \left(\frac{\sum_{t\in A} v_t^l \exp(v_x^f \cdot v_t^l/\tau)}{\sum_{t\in A}\exp(v_x^f \cdot v_t^l/\tau)} - v_p^l\right) \\
=& \frac{1}{\tau|P(y)|} \sum_{p\in P(y)} \left(\frac{\sum_{t\in P(y)} v_t^l \exp(v_x^f \cdot v_t^l/\tau)}{\sum_{t\in A}\exp(v_x^f \cdot v_t^l/\tau)} + \right. \\
& \left. \frac{\sum_{t\in N(y)} v_t^l \exp(v_x^f \cdot v_t^l/\tau)}{\sum_{t\in A}\exp(v_x^f \cdot v_t^l/\tau)} - v_p^l\right)
\end{aligned}$$

$$
\begin{aligned}
=&\frac{1}{\tau}\frac{\sum_{t\in P(y)} v_t^l \exp(v_x^f \cdot v_t^l/\tau)}{\sum_{t\in A}\exp(v_x^f \cdot v_t^l/\tau)}+\\
&\frac{1}{\tau}\frac{\sum_{t\in N(y)} v_t^l \exp(v_x^f \cdot v_t^l/\tau)}{\sum_{t\in A}\exp(v_x^f \cdot v_t^l/\tau)}-\\
&\frac{1}{\tau|P(y)|}\sum_{p\in P(y)} v_p^l\\
=&\frac{1}{\tau}[\sum_{t\in P(y)} v_t^l(\frac{\exp(v_x^f \cdot v_t^l/\tau)}{\sum_{a\in A}\exp(v_x^f \cdot v_a^l/\tau)}-\frac{1}{|P(y)|})+\\
&\sum_{t\in N(y)} v_t^l\frac{\exp(v_x^f \cdot v_t^l/\tau)}{\sum_{a\in A}\exp(v_x^f \cdot v_a^l/\tau)}]
\end{aligned}
$$

Further, we have the unnormalized feature embedding w_x^f, $v_x^f = \frac{w_x^f}{\|w_x^f\|}$.

$$
\frac{\partial v_x^f}{\partial w_x^f} = \frac{1}{\|w_x^f\|}(I - \frac{w_x^f {w_x^f}^T}{\|w_x^f\|^2}) = \frac{1}{\|w_x^f\|}(I - v_x^f {v_x^f}^T) \tag{A.5}
$$

where I is an $E \times E$ identity matrix. The gradient of $\mathcal{L}_{CL}$ w.r.t. w_x^f can be derived with chain rule,

$$
\begin{aligned}
&\frac{\partial \mathcal{L}_{CL}}{\partial w_x^f} = \frac{\partial v_x^f}{\partial w_x^f}\frac{\partial \mathcal{L}_{CL}}{\partial v_x^f}\\
=&\frac{1}{\|w_x^f\|}(I - v_x^f {v_x^f}^T)\frac{1}{\tau}[\sum_{t\in P(y)} v_t^l(\frac{\exp(v_x^f \cdot v_t^l/\tau)}{\sum_{a\in A}\exp(v_x^f \cdot v_a^l/\tau)}\\
&-\frac{1}{|P(y)|}) + \sum_{t\in N(y)} v_t^l\frac{\exp(v_x^f \cdot v_t^l/\tau)}{\sum_{a\in A}\exp(v_x^f \cdot v_a^l/\tau)}]\\
=&\frac{1}{\tau\|w_x^f\|}[\sum_{t\in P(y)}(v_t^l - (v_x^f v_t^l)v_x^f)(\frac{\exp(v_x^f \cdot v_t^l/\tau)}{\sum_{a\in A}\exp(v_x^f \cdot v_a^l/\tau)}\\
&-\frac{1}{|P(y)|})+\\
&\sum_{t\in N(y)}(v_t^l - (v_x^f v_t^l)v_x^f)\frac{\exp(v_x^f \cdot v_t^l/\tau)}{\sum_{a\in A}\exp(v_x^f \cdot v_a^l/\tau)}]
\end{aligned} \tag{A.6}
$$

We can then observe that if v_x^f and v_t^l are orthogonal ($v_x^f v_t^l \to 0$), $\|v_t^l - (v_x^f v_t^l)v_x^f\|$

will be close to 1 and the gradients would be large. Otherwise, for weak positives or negatives ($|v_x^f v_t^l| \to 1$), the gradients would be small.

APPENDIX B

APPENDIX FOR CHAPTER 3

B.1 Appendix for Chapter 3.1

B.1.1 Derivation of Predictive Information

In this section, we give a self-contained derivation of the predictive information.

A multivariate Gaussian random variable $X \in \mathbb{R}^N$ has the following PDF:

$$p(x) = \frac{1}{\sqrt{2\pi}^N \sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

where μ is the mean and Σ is the covariance matrix with $\Sigma = \mathbb{E}[(X-\mathbb{E}[X])(X-\mathbb{E}[X])]$.

From the definition of entropy

$$H(X) = -\int p(x) \ln p(x) dx$$

we can derive the entropy formula for multivariate Gaussian

$$\begin{aligned} H(X) &= -\int p(x) \ln \frac{1}{\sqrt{2\pi}^N \sqrt{|\Sigma|}} \exp(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)) dx \\ &= -\int p(x) \ln\left(\frac{1}{(\sqrt{2\pi})^N \sqrt{|\Sigma|}}\right) dx - \int p(x)\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu))\right) dx \\ &= \frac{1}{2} \ln((2\pi e)^N |\Sigma|). \end{aligned} \tag{B.1}$$

Consider the joint Gaussian distribution $\begin{pmatrix} X \\ Y \end{pmatrix} \sim \mathcal{N}(\mu, \Sigma) = \mathcal{N}\left(\begin{pmatrix} \mu_X \\ \mu_Y \end{pmatrix}, \begin{pmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{pmatrix}\right)$ where $X \in \mathbb{R}^p$, and $Y \in \mathbb{R}^q$. We can plug in the entropy in (B.1) and obtained

$$MI(X,Y) = H(X) + H(Y) - H(X,Y)$$

$$= \frac{1}{2}\ln((2\pi e)^p|\Sigma_{XX}|) + \frac{1}{2}\ln((2\pi e)^q|\Sigma_{YY}|) - \frac{1}{2}\ln((2\pi e)^{p+q}|\Sigma|)$$
$$= -\frac{1}{2}\ln\frac{|\Sigma|}{|\Sigma_{XX}||\Sigma_{YY}|}$$

For a latent sequence $Z = \{z_1, z_2, ...\}$ where $z_i \in \mathbb{R}^d$, we define $Z_t^{past} = \{z_{t-T+1}, ..., z_t\}$, and $Z_t^{future} = \{z_{t+1}, ..., z_{t+T}\}$. Based on our stationarity assumption, all the length-$2T$ windows of states within the range are drawn from the same Gaussian distribution with covariance $\Sigma_{2T}(Z)$, and similarly for all the length-T windows. As a result and under the stationary assumption, $H(Z_t^{past}) = H(Z_t^{future}) = \frac{1}{2}\ln((2\pi e)^{Td}|\Sigma_T(Z)|)$, $H(Z_t^{past}, Z_t^{future}) = \frac{1}{2}\ln((2\pi e)^{2Td}|\Sigma_{2T}(Z)|)$, and

$$\begin{aligned} I_T &= MI(Z_t^{past}, Z_t^{future}) \\ &= H(Z_t^{past}) + H(Z_t^{future}) - H(Z_t^{past}, Z_t^{future}) \\ &= \frac{1}{2}\ln((2\pi e)^{Td}|\Sigma_T(Z)|) + \frac{1}{2}\ln((2\pi e)^{Td}|\Sigma_T(Z)|) - \frac{1}{2}\ln((2\pi e)^{2Td}|\Sigma_{2T}(Z)|) \\ &= \ln|\Sigma_T(Z)| - \frac{1}{2}\ln|\Sigma_{2T}(Z)| \end{aligned}$$

The predictive information I_T only depend on T but not specific time index t.

B.2 Appendix for Chapter 3.2

B.2.1 Derivation of the per-sequence ELBO

For the input sequence $x_{1:T}$, we show the ELBO derived from the approximate posterior is a lower estimate of its log-likelihood. This proof below is adapted from the standard VAE framework [282, 327], noticing that either the prior or the

approximate posterior factorizes over s and $z_{1:T}$.

$$
\begin{aligned}
&\log p(x_{1:T}) \\
\geq &- KL[q(s, z_{1:T}|x_{1:T})||p(s, z_{1:T}|x_{1:T})] + \log p(x_{1:T}) \\
= &\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}\left[\log p(s, z_{1:T}|x_{1:T}) - \log q(s, z_{1:T}|x_{1:T}) + \log p(x_{1:T})\right] \\
= &\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}[\log p(x_{1:T}|s, z_{1:T}) - \log q(s, z_{1:T}|x_{1:T}) + \log p(s, z_{1:T})] \\
= &\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}[\log p(x_{1:T}|s, z_{1:T}) - \log q(s|x_{1:T}) - \log p(z_{1:T}|x_{1:T}) + \log p(s) + \log p(z_{1:T})] \\
= &\mathbb{E}_{q(z_{1:T},s|x_{1:T})}\left[\log p(x_{1:T}|s, z_{1:T})\right] - KL[q(s|x_{1:T})||p(s)] - KL[q(z_{1:T}|x_{1:T})||p(z_{1:T})]\,.
\end{aligned}
$$

B.2.2 Proof of Theorem 1

Let p_D be the empirical data distribution, assigning probability mass $1/N$ for each of the N training sequences D. Define the aggregated posteriors as follows:

$$q(s) = \mathbb{E}_{x_{1:T}\sim p_D}[q(s|x_{1:T})] = \frac{1}{N}\sum_{x_{1:T}\in D} q(s|x_{1:T}),$$

$$q(z_{1:T}) = \mathbb{E}_{x_{1:T}\sim p_D}[q(z_{1:T}|x_{1:T})] = \frac{1}{N}\sum_{x_{1:T}\in D} q(z_{1:T}|x_{1:T}),$$

$$q(s, z_{1:T}) = \mathbb{E}_{x_{1:T}\sim p_D}[q(s|x_{1:T})q(z_{1:T}|x_{1:T})] = \frac{1}{N}\sum_{x_{1:T}\in D} q(s|x_{1:T})q(z_{1:T}|x_{1:T}).$$

With these definitions, we have

$$
\begin{aligned}
&\mathbb{E}_{x_{1:T}\sim p_D}[KL[q(s|x_{1:T})||p(s)]] \\
= &\mathbb{E}_{x_{1:T}\sim p_D}\mathbb{E}_{q(s|x_{1:T})}[\log q(s|x_{1:T}) - \log q(s) + \log q(s) - \log p(s)] \\
= &\mathbb{E}_{q(x_{1:T},s)}\log\left[\frac{q(s|x_{1:T})}{q(s)}\right] + \mathbb{E}_{q(x_{1:T},s)}[\log q(s) - \log p(s)] \\
= &I_q(x_{1:T}; s) + KL[q(s)||p(s)].
\end{aligned}
\tag{B.2}
$$

In other words,

$$KL[q(s)||p(s)] = \mathbb{E}_{x_{1:T}\sim p_D}[KL[q(s|x_{1:T})||p(s)]] - I_q(x_{1:T}; s). \tag{B.3}$$

Similarly, we have

$$KL[q(z_{1:T})||p(z_{1:T})] = E_{x_{1:T}\sim p_D}[KL[q(z_{1:T}|x_{1:T})||p(z_{1:T})]] - I_q(x_{1:T}; z_{1:T}). \tag{B.4}$$

We are now ready to prove the theorem. We derive a dataset ELBO by subtracting a different KL-divergence from the data log-likelihood:

$$\begin{aligned}
&\frac{1}{N}\sum_{x_{1:T}\in D}\log p(x_{1:T}) = \mathbb{E}_{x_{1:T}\sim p_D}[\log p(x_{1:T})] \\
\geq&\mathbb{E}_{x_{1:T}\sim p_D}[\log p(x_{1:T}) - KL[q(s, z_{1:T})||p(s, z_{1:T}|x_{1:T})]] \\
=&\mathbb{E}_{x_{1:T}\sim p_D}[\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}[\log p(x_{1:T}) - (\log q(s, z_{1:T}) - \log p(s, z_{1:T}|x_{1:T}))]] \\
=&\mathbb{E}_{x_{1:T}\sim p_D}[\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}[\log p(x_{1:T}) - \log q(s, z_{1:T}) + \log p(s, z_{1:T}|x_{1:T})]] \\
=&\mathbb{E}_{x_{1:T}\sim p_D}[\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}[\log p(x_{1:T}) - \log q(s, z_{1:T}) \\
&\qquad + \log p(x_{1:T}|s, z_{1:T}) + \log p(s, z_{1:T}) - \log p(x_{1:T})]] \\
=&\mathbb{E}_{x_{1:T}\sim p_D}[\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}[\log p(x_{1:T}|s, z_{1:T}) - \log q(s, z_{1:T}) + \log p(s, z_{1:T})]] \\
=&\mathbb{E}_{x_{1:T}\sim p_D}[\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}[\log p(x_{1:T}|s, z_{1:T})]] - \mathbf{KL}[\mathbf{q(s, z_{1:T})}||\mathbf{p(s, z_{1:T})}] \\
=&\mathbb{E}_{x_{1:T}\sim p_D}[\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}[\log p(x_{1:T}|s, z_{1:T})]] \\
&\qquad - \mathbb{E}_{x_{1:T}\sim p_D}[\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}[\log q(s, z_{1:T}) - \log p(s, z_{1:T})]] \\
=&\mathbb{E}_{x_{1:T}\sim p_D}[\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}[\log p(x_{1:T}|s, z_{1:T})]] \\
&- \mathbb{E}_{x_{1:T}\sim p_D}[\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}[\log q(s, z_{1:T}) - \log q(s)q(z_{1:T}) + \log q(s)q(z_{1:T}) - \log p(s, z_{1:T})]] \\
=&\mathbb{E}_{x_{1:T}\sim p_D}[\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}[\log p(x_{1:T}|s, z_{1:T})]] \\
&\qquad - \mathbb{E}_{x_{1:T}\sim p_D}\left[\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}\left[\log\frac{q(s, z_{1:T})}{q(s)q(z_{1:T})} + \log\frac{q(s)q(z_{1:T})}{p(s, z_{1:T})}\right]\right] \\
=&\mathbb{E}_{x_{1:T}\sim p_D}[\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}[\log p(x_{1:T}|s, z_{1:T})]]
\end{aligned}$$

$$
\begin{aligned}
&- I_q(s; z_{1:T}) - \mathbb{E}_{x_{1:T}\sim p_D}\left[\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}\left[\log \frac{q(s)q(z_{1:T})}{p(s)p(z_{1:T})}\right]\right] \\
=&\mathbb{E}_{x_{1:T}\sim p_D}[\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}[\log p(x_{1:T}|s, z_{1:T})]] - I_q(s; z_{1:T}) \\
&- \mathbb{E}_{x_{1:T}\sim p_D}\left[\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}\left[\log \frac{q(s)}{p(s)}\right]\right] - \mathbb{E}_{x_{1:T}\sim p_D}\left[\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}\left[\log \frac{q(z_{1:T})}{p(z_{1:T})}\right]\right] \\
=&\mathbb{E}_{x_{1:T}\sim p_D}[\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}[\log p(x_{1:T}|s, z_{1:T})]] - I_q(s; z_{1:T}) \\
&- KL[q(s)||p(s)] - KL[q(z_{1:T})||p(z_{1:T})] \\
=&\mathbb{E}_{x_{1:T}\sim p_D}[\mathbb{E}_{q(s,z_{1:T}|x_{1:T})}[\log p(x_{1:T}|s, z_{1:T})]] - I_q(s; z_{1:T}) \\
&- (\mathbb{E}_{x_{1:T}\sim p_D}[KL[q(s|x_{1:T})||p(s)]] - I_q(s; x_{1:T})) \\
&- (\mathbb{E}_{x_{1:T}\sim p_D}[KL[q(z_{1:T}|x_{1:T})||p(z_{1:T})]] - I_q(z_{1:T}; x_{1:T})) \\
=&\mathbb{E}_{x_{1:T}\sim p_D}[\mathbb{E}_{q(z_{1:T},s|x_{1:T})}[\log p(x_{1:T}|s, z_{1:T})] \\
&- \mathbb{E}_{x_{1:T}\sim p_D}[KL[q(s|x_{1:T})||p(s)]] - \mathbb{E}_{x_{1:T}\sim p_D}[KL[q(z_{1:T}|x_{1:T})||p(z_{1:T})]] \\
&+ I_q(s; x_{1:T}) + I_q(z_{1:T}; x_{1:T}) - I_q(s; z_{1:T}).
\end{aligned}
$$

where we have plugged in (B.3) and (B.4) in the third to last step. The last equation is the dataset ELBO objective shown in the main text.

B.2.3 MWS estimation of $I(s; z_{1:T})$

We estimate $I(s; z_{1:T})$ using minibatch weighted sampling (MWS). The estimation is simply adapted from [321, 327]:

$$
\begin{aligned}
&I(s; z_{1:T}) = H(s) + H(z_{1:T}) - H(s, z_{1:T}) \\
&H(s) = -\mathbb{E}_{q(s,z_{1:T})}[\log q(s)] \approx -\frac{1}{M}\sum_{i=1}^{M}\left[\log \sum_{j=1}^{M} q(s(x^i_{1:T})|x^j_{1:T}) - \log(NM)\right]
\end{aligned}
\tag{B.5}
$$

where N is the dataset size and M is the minibatch size. $H(z_{1:T}), H(s, z_{1:T})$ are estimated similarly, with the trajectory of $z_{1:T}$ sampled for each sequence in the minibatch.

B.2.4 Contrastive estimation of MI

Suppose $z_{1:T}$ is derived from the anchor sequence $x_{1:T}$, and we have one "positive" sequence $x^+_{1:T}$ as well as n "negative" sequences $\{x^j_{1:T}\}_{j=1}^n$ in total. We have

$$
\begin{aligned}
&\mathbb{E}_{p_D}\left[\log \frac{\frac{q(x^+_{1:T}|z_{1:T})}{q(x^+_{1:T})}}{\frac{q(x^+_{1:T}|z_{1:T})}{q(x^+_{1:T})} + \sum_{j=1}^n \frac{q(x^j_{1:T}|z_{1:T})}{q(x^j_{1:T})}}\right] \\
=&-\mathbb{E}_{p_D}\left[\log(1 + \frac{q(x^+_{1:T})}{q(x^+_{1:T}|z_{1:T})}\sum_{j=1}^n \frac{q(x^j_{1:T}|z_{1:T})}{q(x^j_{1:T})})\right] \\
\approx&-\mathbb{E}_{p_D}\left[\log\left(1 + \frac{q(x^+_{1:T})}{q(x^+_{1:T}|z_{1:T})}\cdot n\mathbb{E}_{x^j_{1:T}}\left[\frac{q(x^j_{1:T}|z_{1:T})}{q(x^j_{1:T})}\right]\right)\right] \\
=&-\mathbb{E}_{p_D}\left[\log\left(1 + \frac{q(x^+_{1:T})}{q(x^+_{1:T}|z_{1:T})}\cdot n\right)\right] \\
=&-\mathbb{E}_{p_D}\left[\log\left(\frac{1}{1+n} + \frac{q(x^+_{1:T})}{q(x^+_{1:T}|z_{1:T})}\cdot \frac{n}{1+n}\right)\right] - \log(n+1) \\
\leq&-\mathbb{E}_{p_D}\left[\frac{1}{n+1}\log 1 + \frac{n}{n+1}\log\frac{q(x^+_{1:T})}{q(x^+_{1:T}|z_{1:T})}\right] - \log(n+1) \\
=&-\frac{n}{n+1}E_{p_D}\left[\log\frac{q(x^+_{1:T})}{q(x^+_{1:T}|z_{1:T})}\right] - \log(n+1) \\
=&\frac{n}{n+1}\mathbb{E}_{p_D}\left[\log\frac{q(x^+_{1:T}|z_{1:T})}{q(x^+_{1:T})}\right] - \log(n+1) \\
\approx&\frac{n}{n+1}I(x_{1:T}; z_{1:T}) - \log(n+1)
\end{aligned}
\tag{B.6}
$$

where the first $\leq$ step uses Jensen's inequality, and the approximations by sampling follow the development of CPC [123]. Similar derivations can also be obtained for s.

Use augmentation in contrastive estimation Note that in CPC, for each frame, the positive example is a nearby frame. That is, CPC uses the temporal smoothness as the inductive bias for learning per-frame representations. In our setup however, we treat the entire trajectory $z_{1:T}$ as samples in contrastive estimation, and we must resort to other inductive biases for finding positive examples.

Here we provide another motivation for the use of augmented sequences as positive examples. Imagine that the latent factors are discrete, and the mapping from the latent factors $(s, z_{1:T})$ to observations $x_{1:T}$ is done by a deterministic mapping (while the factors remain random variables with prior distributions). Furthermore, the mapping is invertible in the sense that we could identify a unique $(s, z_{1:T})$ that generates $x_{1:T}$. Then for two sequences $x^1_{1:T}$ and $x^2_{1:T}$ generated with common dynamic factors $z^*_{1:T}$ but different static factors s^1 and s^2, we have

$$
\begin{aligned}
p(x^1_{1:T}) = p(z_{1:T} = z^*_{1:T}, s = s^1) = p(z_{1:T} = z^*_{1:T})\, p(s = s^1), \\
p(x^1_{1:T} | z_{1:T}) = 1(z_{1:T} = z^*_{1:T})\, p(s = s^1), \\
\frac{p(x^1_{1:T} | z_{1:T})}{p(x_{1:T})} = \frac{1(z_{1:T} = z^*_{1:T})}{p(z_{1:T} = z^*_{1:T})} = \frac{p(x^2_{1:T} | z_{1:T})}{p(x_{1:T})}.
\end{aligned}
$$

The last equation is obtained by dividing the second line by the first line, where the $p(s = s^1)$ is conveniently canceled out. In other words, under the above simplifying assumptions, the probability ratio used in contrastive estimation is the same for the original sequence and the augmented sequence.

B.2.5 Summary of objective function and estimations

Though the objective function has been proven to be an ELBO, we don't have to stick to the natural coefficients. For example, to control the information bottleneck, one can add coefficients to the KL terms. To improve the disentanglement, one can add coefficients to the MI terms. This leads to the final training objective (3.12).

In practice, $\gamma = 1$ gives good results and we mainly tune α, β. As mentioned in Sec 3.2.2, $I(s; x_{1:T}), I(z_{1:T}; x_{1:T})$ are estimated contrastively. $I(s; z_{1:T})$ is estimated

through mini-batch weighted sampling (MWS) as discussed in B.2.3. Note that we use contrastive learning here to maintain some invariance (static or dynamic factors) across different views, besides just estimating the MI terms. For tasks where no such invariance exists, MWS is still a good option for the estimation (e.g. $I(z_{1:T}; s)$).

BIBLIOGRAPHY

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[2] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, 2014.

[3] Tara N Sainath, Brian Kingsbury, George Saon, Hagen Soltau, Abdelrahman Mohamed, George Dahl, and Bhuvana Ramabhadran. Deep convolutional neural networks for large-scale speech tasks. *Neural networks*, 64:39–48, 2015.

[4] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[5] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius B Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago M Paixao, Filipe Mutz, et al. Self-driving cars: A survey. *Expert Systems with Applications*, 165:113816, 2021.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[7] Bo Li, Ruoming Pang, Tara N Sainath, Anmol Gulati, Yu Zhang, James Qin, Parisa Haghani, W Ronny Huang, Min Ma, and Junwen Bai. Scaling end-to-end models for large-scale multilingual asr. In *2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 1011–1018. IEEE, 2021.

[8] Junwen Bai, Yexiang Xue, Johan Bjorck, Ronan Le Bras, Brendan Rappazzo, Richard Bernstein, Santosh K Suram, Robert Bruce Van Dover, John M Gregoire, and Carla P Gomes. Phase mapper: Accelerating materials discovery with ai. *AI Magazine*, 39(1):15–26, 2018.

[9] Di Chen, Yiwei Bai, Sebastian Ament, Wenting Zhao, Dan Guevarra, Lan Zhou, Bart Selman, R Bruce van Dover, John M Gregoire, and Carla P Gomes. Automating crystal-structure phase mapping by combining deep learning with constraint reasoning. *Nature Machine Intelligence*, 3(9):812–822, 2021.

[10] Shufeng Kong, Francesco Ricci, Dan Guevarra, Jeffrey B Neaton, Carla P Gomes, and John M Gregoire. Density of states prediction for materials discovery via contrastive learning from probabilistic embeddings. *Nature communications*, 13(1):1–12, 2022.

[11] Sebastian Ament, Maximilian Amsler, Duncan R Sutherland, Ming-Chiang Chang, Dan Guevarra, Aine B Connolly, John M Gregoire, Michael O Thompson, Carla P Gomes, and R Bruce van Dover. Autonomous materials synthesis via hierarchical active learning of nonequilibrium phase diagrams. *Science Advances*, 7(51):eabg4930, 2021.

[12] Maithra Raghu and Eric Schmidt. A survey of deep learning for scientific discovery. *arXiv preprint arXiv:2003.11755*, 2020.

[13] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[15] David Kirk et al. Nvidia cuda software and gpu parallel computing architecture. In *ISMM*, volume 7, pages 103–104, 2007.

[16] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th annual international symposium on computer architecture*, pages 1–12, 2017.

[17] Seref Sagiroglu and Duygu Sinanc. Big data: A review. In *2013 international conference on collaboration technologies and systems (CTS)*, pages 42–47. IEEE, 2013.

[18] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[19] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[20] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew

Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[22] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[23] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[24] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[26] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[27] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence

learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[29] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[30] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[31] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[32] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[33] Daniel Povey, Arnab Ghoshal, and Gilles Boulianne. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding*. IEEE Signal Processing Society, 2011.

[34] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[35] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4580–4584. IEEE, 2015.

[36] Yu Zhang, William Chan, and Navdeep Jaitly. Very deep convolutional networks for end-to-end speech recognition. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 4845–4849. IEEE, 2017.

[37] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460, 2020.

[38] Jessica Vamathevan, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran, George Lee, Bin Li, Anant Madabhushi, Parantu Shah, Michaela Spitzer, et al. Applications of machine learning in drug discovery and development. *Nature reviews Drug discovery*, 18(6):463–477, 2019.

[39] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[40] Edward O Pyzer-Knapp, Jed W Pitera, Peter WJ Staar, Seiji Takeda, Teodoro Laino, Daniel P Sanders, James Sexton, John R Smith, and Alessandro Curioni. Accelerating materials discovery using artificial intelligence, high performance computing and robotics. *npj Computational Materials*, 8(1):1–9, 2022.

[41] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017.

[42] Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical Transactions of the Royal Society A*, 379(2194):20200209, 2021.

[43] Ariel Ortiz-Bobea, Toby R Ault, Carlos M Carrillo, Robert G Chambers, and David B Lobell. Anthropogenic climate change has slowed global agricultural productivity growth. *Nature Climate Change*, 11(4):306–312, 2021.

[44] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[45] Mehar Vijh, Deeksha Chandola, Vinay Anand Tikkiwal, and Arun Kumar. Stock closing price prediction using machine learning techniques. *Procedia computer science*, 167:599–606, 2020.

[46] Shruti Kaushik, Abhinav Choudhury, Pankaj Kumar Sheron, Nataraj Dasgupta, Sayee Natarajan, Larry A Pickett, and Varun Dutt. Ai in healthcare: time-series forecasting using statistical, neural, and ensemble architectures. *Frontiers in big data*, 3:4, 2020.

[47] Oleksii Kuchaiev, Jason Li, Huyen Nguyen, Oleksii Hrinchuk, Ryan Leary, Boris Ginsburg, Samuel Kriman, Stanislav Beliaev, Vitaly Lavrukhin, Jack Cook, et al. Nemo: a toolkit for building ai applications using neural modules. *arXiv preprint arXiv:1909.09577*, 2019.

[48] Suman Ravuri, Karel Lenc, Matthew Willson, Dmitry Kangin, Remi Lam, Piotr Mirowski, Megan Fitzsimons, Maria Athanassiadou, Sheleem Kashem, Sam Madge, et al. Skilful precipitation nowcasting using deep generative models of radar. *Nature*, 597(7878):672–677, 2021.

[49] MMAction2 Contributors. Openmmlab's next generation video understanding toolbox and benchmark. `https://github.com/open-mmlab/mmaction2`, 2020.

[50] Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. Videogpt: Video generation using vq-vae and transformers. *arXiv preprint arXiv:2104.10157*, 2021.

[51] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339, 1989.

[52] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. *Advances in neural information processing systems*, 28, 2015.

[53] Jesse Vig, Ali Madani, Lav R Varshney, Caiming Xiong, Nazneen Rajani, et al. Bertology meets biology: Interpreting attention in protein language models. In *International Conference on Learning Representations*, 2020.

[54] Volodymyr Kuleshov, Chao Jiang, Wenyu Zhou, Fereshteh Jahanbani, Serafim Batzoglou, and Michael Snyder. Synthetic long-read sequencing reveals intraspecies diversity in the human microbiome. *Nature biotechnology*, 34(1):64–69, 2016.

[55] Chiheb Ben Mahmoud, Andrea Anelli, Gábor Csányi, and Michele Ceriotti. Learning the electronic density of states in condensed matter. *Physical Review B*, 102(23):235130, 2020.

[56] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[57] Steffen Schneider, Alexei Baevski, Ronan Collobert, and Michael Auli. wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*, 2019.

[58] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[59] Steph-Yves Louis, Yong Zhao, Alireza Nasiri, Xiran Wang, Yuqi Song, Fei Liu, and Jianjun Hu. Graph convolutional neural networks with global attention for improved materials property prediction. *Physical Chemistry Chemical Physics*, 22(32):18141–18148, 2020.

[60] Robert DiPietro and Gregory D Hager. Deep learning: Rnns and lstm. In *Handbook of medical image computing and computer assisted intervention*, pages 503–519. Elsevier, 2020.

[61] Jie Yang and Yue Zhang. Ncrf++: An open-source neural sequence labeling toolkit. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.

[62] Mirco Ravanelli and Yoshua Bengio. Speaker recognition from raw waveform with sincnet. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 1021–1028. IEEE, 2018.

[63] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. Deep learning techniques for music generation–a survey. *arXiv preprint arXiv:1709.01620*, 2017.

[64] John S Garofolo. Timit acoustic phonetic continuous speech corpus. *Linguistic Data Consortium, 1993*, 1993.

[65] Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, et al. Findings of the 2017 conference on machine translation (wmt17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, 2017.

[66] Marina Fomicheva, Shuo Sun, Erick Fonseca, Chrysoula Zerva, Frédéric Blain, Vishrav Chaudhary, Francisco Guzmán, Nina Lopatina, Lucia Specia, and André FT Martins. Mlqe-pe: A multilingual quality estimation and post-editing dataset. *arXiv preprint arXiv:2010.04480*, 2020.

[67] Christopher Olah. Understanding lstm networks, 2015.

[68] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

[69] John J Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10):3088–3092, 1984.

[70] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[71] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proc. Interspeech 2014*, pages 338–342, 2014.

[72] Shuohang Wang and Jing Jiang. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*, 2015.

[73] Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*, 2014.

[74] Xiangyun Qing and Yugang Niu. Hourly day-ahead solar irradiance prediction using weather forecasts by lstm. *Energy*, 148:461–468, 2018.

[75] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[76] Shudong Yang, Xueying Yu, and Ying Zhou. Lstm and gru neural network performance comparison study: Taking yelp review dataset as an example. In *2020 International workshop on electronic communication and artificial intelligence (IWECAI)*, pages 98–101. IEEE, 2020.

[77] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015.

[78] Alexander M Rush. The annotated transformer. In *Proceedings of workshop for NLP open source software (NLP-OSS)*, pages 52–60, 2018.

[79] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *arXiv preprint arXiv:2106.04554*, 2021.

[80] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[81] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[82] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.

[83] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

[84] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

[85] David So, Quoc Le, and Chen Liang. The evolved transformer. In *International Conference on Machine Learning*, pages 5877–5886. PMLR, 2019.

[86] Linhao Dong, Shuang Xu, and Bo Xu. Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5884–5888. IEEE, 2018.

[87] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

[88] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. *arXiv preprint arXiv:1809.04281*, 2018.

[89] Ronghang Hu, Amanpreet Singh, Trevor Darrell, and Marcus Rohrbach. Iterative answer prediction with pointer-augmented multimodal transformers for textvqa. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9992–10002, 2020.

[90] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei

Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019.

[91] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[92] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, 2021.

[93] Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*, 2018.

[94] Weiwei Liu, Haobo Wang, Xiaobo Shen, and Ivor Tsang. The emerging trends of multi-label learning. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[95] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*, 2022.

[96] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*, 2019.

[97] Olivier Henaff. Data-efficient image recognition with contrastive predictive

coding. In *International conference on machine learning*, pages 4182–4192. PMLR, 2020.

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

[99] Charles Kittel, Paul McEuen, and Paul McEuen. *Introduction to solid state physics*, volume 8. Wiley New York, 1996.

[100] Neil W Ashcroft and N David Mermin. Solid state physics (saunders college, philadelphia, 1976). *Appendix N*, 166:87, 2010.

[101] Tele Atlas North America and Inc ESRI. Us counties, 2010.

[102] Darrel L Good and Scott H Irwin. Usda corn and soybean acreage estimates and yield forecasts: dispelling myths and misunderstandings, 2011.

[103] David Rolnick, Priya L Donti, Lynn H Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-Dupont, Natasha Jaques, Anna Waldman-Brown, et al. Tackling climate change with machine learning. *ACM Computing Surveys (CSUR)*, 55(2):1–96, 2022.

[104] Ariel Ortiz-Bobea and Jesse Tack. Is another genetic revolution needed to offset climate change impacts for us maize yields? *Environmental Research Letters*, 13(12):124009, 2018.

[105] Saeed Khaki, Lizhi Wang, and Sotirios V Archontoulis. A cnn-rnn framework for crop yield prediction. *Frontiers in Plant Science*, 10:1750, 2020.

[106] Joshua Fan, Junwen Bai, Zhiyun Li, Ariel Ortiz-Bobea, and Carla P Gomes. A gnn-rnn approach for harnessing geospatial and temporal information: application to crop yield prediction. *arXiv preprint arXiv:2111.08900*, 2021.

[107] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[108] Min-Ling Zhang and Zhi-Hua Zhou. A k-nearest neighbor based algorithm for multi-label classification. In *2005 IEEE international conference on granular computing*, volume 2, pages 718–721. IEEE, 2005.

[109] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2013.

[110] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. A review of multi-label classification methods. In *Proceedings of the 2nd ADBIS workshop on data mining and knowledge discovery (ADMKD 2006)*, pages 99–109. Citeseer, 2006.

[111] Jack Lanchantin, Arshdeep Sekhon, and Yanjun Qi. Neural message passing for multi-label classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 138–163. Springer, 2019.

[112] Wenting Zhao, Shufeng Kong, Junwen Bai, Daniel Fink, and Carla Gomes. Hot-vae: Learning high-order label correlation for multi-label classification via attention-based variational autoencoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 15016–15024, 2021.

[113] Chris Wood, Brian Sullivan, Marshall Iliff, Daniel Fink, and Steve Kelling. ebird: engaging birders in science and conservation. *PLoS biology*, 9(12):e1001220, 2011.

[114] Carla Gomes, Thomas Dietterich, Christopher Barrett, Jon Conrad, Bistra Dilkina, Stefano Ermon, Fei Fang, Andrew Farnsworth, Alan Fern, Xiaoli Fern, et al. Computational sustainability: Computing for a better world and a sustainable future. *Communications of the ACM*, 62(9):56–65, 2019.

[115] Jack Choquette, Wishwesh Gandhi, Olivier Giroux, Nick Stam, and Ronny Krashinsky. Nvidia a100 tensor core gpu: Performance and innovation. *IEEE Micro*, 41(2):29–35, 2021.

[116] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images, 2009.

[117] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[118] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.

[119] Douglas B Paul and Janet Baker. The design for the wall street journal-based csr corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*, 1992.

[120] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 5206–5210. IEEE, 2015.

[121] Vineel Pratap, Qiantong Xu, Anuroop Sriram, Gabriel Synnaeve, and Ronan Collobert. Mls: A large-scale multilingual dataset for speech research. *arXiv preprint arXiv:2012.03411*, 2020.

[122] Yann LeCun and Ishan Misra. Self-supervised learning: The dark matter of intelligence. *Meta AI*, 23, 2021.

[123] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[124] 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.

[125] J MacQueen. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pages 281–297, 1967.

[126] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[127] Rie Kubota Ando, Tong Zhang, and Peter Bartlett. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6(11), 2005.

[128] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[129] Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of the 20th SIGNLL conference on computational natural language learning*, pages 51–61, 2016.

[130] Mike Wu, Chengxu Zhuang, Milan Mosse, Daniel Yamins, and Noah Goodman. On mutual information in contrastive learning for visual representations. *arXiv preprint arXiv:2005.13149*, 2020.

[131] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[132] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(3), 2010.

[133] Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings, 2010.

[134] William Bialek, Ilya Nemenman, and Naftali Tishby. Predictability, complexity, and learning. *Neural computation*, 13(11):2409–2463, 2001.

[135] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9243–9252, 2020.

[136] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[137] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital signal processing*, 73:1–15, 2018.

[138] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[139] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897, 2020.

[140] Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. *arXiv preprint arXiv:2004.12651*, 2020.

[141] Junwen Bai, Shufeng Kong, and Carla P Gomes. Gaussian mixture variational autoencoder with contrastive learning for multi-label classification. In *International Conference on Machine Learning*, pages 1383–1398. PMLR, 2022.

[142] John Theodore Houghton, Geoffrey J Jenkins, and Jim J Ephraums. Climate change: the ipcc scientific assessment. *American Scientist (United States)*, 80(6), 1990.

[143] Kevin E Trenberth, John T Fasullo, and Theodore G Shepherd. Attribution of climate extreme events. *Nature Climate Change*, 5(8):725–730, 2015.

[144] Chuang Zhao, Bing Liu, Shilong Piao, Xuhui Wang, David B Lobell, Yao Huang, Mengtian Huang, Yitong Yao, Simona Bassu, Philippe Ciais, et al. Temperature increase reduces global yields of major crops in four independent estimates. *Proceedings of the National Academy of Sciences*, 114(35):9326–9331, 2017.

[145] Ariel Ortiz-Bobea, Erwin Knippenberg, and Robert G Chambers. Growing climatic sensitivity of us agriculture linked to technological change and regional specialization. *Science advances*, 4(12):eaat4343, 2018.

[146] MP Reynolds, R Ortiz, et al. Adapting crops to climate change: a summary. *Climate change and crop production*, pages 1–8, 2010.

[147] Ali Raza, Ali Razzaq, Sundas Saher Mehmood, Xiling Zou, Xuekun Zhang, Yan Lv, and Jinsong Xu. Impact of climate change on crops adaptation and strategies to tackle its outcome: A review. *Plants*, 8(2):34, 2019.

[148] Thomas Van Klompenburg, Ayalew Kassahun, and Cagatay Catal. Crop yield prediction using machine learning: A systematic literature review. *Computers and Electronics in Agriculture*, 177:105709, 2020.

[149] PR Shukla, J Skeg, E Calvo Buendia, V Masson-Delmotte, H-O Pörtner, DC Roberts, P Zhai, R Slade, S Connors, S van Diemen, et al. Climate change and land: an ipcc special report on climate change, desertification, land degradation, sustainable land management, food security, and greenhouse gas fluxes in terrestrial ecosystems. *Intergovernmental Panel on Climate Change (IPCC)*, 2019.

[150] Rachael D Garrett, Eric F Lambin, and Rosamond L Naylor. Land institu-

tions and supply chain configurations as determinants of soybean planted area and yields in brazil. *Land Use Policy*, 31:385–396, 2013.

[151] Javad Ansarifar, Faezeh Akhavizadegan, and Lizhi Wang. Performance prediction of crosses in plant breeding through genotype by environment interactions. *Scientific Reports*, 10(1):1–11, 2020.

[152] T Horie, M Yajima, and H Nakagawa. Yield forecasting. *Agricultural systems*, 40(1-3):211–236, 1992.

[153] Mohsen Shahhosseini, Guiping Hu, Isaiah Huber, and Sotirios V Archontoulis. Coupling machine learning and crop modeling improves crop yield prediction in the us corn belt. *Scientific reports*, 11(1):1–15, 2021.

[154] Guoyong Leng and Jim W Hall. Predicting spatial and temporal variability in crop yields: an inter-comparison of machine learning, regression and process-based models. *Environmental Research Letters*, 15(4):044027, 2020.

[155] José R Romero, Pablo F Roncallo, Pavan C Akkiraju, Ignacio Ponzoni, Viviana C Echenique, and Jessica A Carballido. Using classification algorithms for predicting durum wheat yield in the province of buenos aires. *Computers and electronics in agriculture*, 96:173–179, 2013.

[156] Snehal S Dahikar and Sandeep V Rode. Agricultural crop yield prediction using artificial neural network approach. *International journal of innovative research in electrical, electronics, instrumentation and control engineering*, 2(1):683–686, 2014.

[157] Oskar Marko, Sanja Brdar, Marko Panic, Predrag Lugonja, and Vladimir Crnojevic. Soybean varieties portfolio optimisation based on yield prediction. *Computers and Electronics in Agriculture*, 127:467–474, 2016.

[158] Jiaxuan You, Xiaocheng Li, Melvin Low, David Lobell, and Stefano Ermon. Deep gaussian process for crop yield prediction based on remote sensing data. In *Thirty-First AAAI conference on artificial intelligence*, 2017.

[159] Saeed Khaki, Zahra Khalilzadeh, and Lizhi Wang. Predicting yield performance of parents in plant breeding: A neural collaborative filtering approach. *Plos one*, 15(5):e0233382, 2020.

[160] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[161] Junteng Jia and Austion R Benson. Residual correlation in graph neural network regression. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 588–598, 2020.

[162] Amol Kapoor, Xue Ben, Luyang Liu, Bryan Perozzi, Matt Barnes, Martin Blais, and Shawn O'Banion. Examining covid-19 forecasting using spatio-temporal graph neural networks. *arXiv preprint arXiv:2007.03113*, 2020.

[163] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[164] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[165] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

[166] Zhiyong Cui, Kristian Henrickson, Ruimin Ke, and Yinhai Wang. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 21(11):4883–4894, 2019.

[167] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. *Advances in Neural Information Processing Systems*, 30:6530–6539, 2017.

[168] Victor Garcia Satorras and Joan Bruna Estrach. Few-shot learning with graph neural networks. In *International Conference on Learning Representations*, 2018.

[169] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[170] Will Hamilton et al. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[171] Jing Liu, CE Goering, and Lei Tian. A neural network for setting target corn yields. *Transactions of the ASAE*, 44(3):705, 2001.

[172] Scott T Drummond, Kenneth A Sudduth, Anupam Joshi, Stuart J Birrell, and Newell R Kitchen. Statistical and neural methods for site–specific yield prediction. *Transactions of the ASAE*, 46(1):5, 2003.

[173] Niketa Gandhi, Owaiz Petkar, and Leisa J Armstrong. Rice crop yield prediction using artificial neural networks. In *2016 IEEE Technological Innovations in ICT for Agriculture and Rural Development (TIAR)*, pages 105–110. IEEE, 2016.

[174] Petteri Nevavuori, Nathaniel Narra, and Tarmo Lipping. Crop yield prediction with deep convolutional neural networks. *Computers and electronics in agriculture*, 163:104859, 2019.

[175] Nari Kim, Kyung-Ja Ha, No-Wook Park, Jaeil Cho, Sungwook Hong, and Yang-Won Lee. A comparison between major artificial intelligence models for crop yield prediction: Case study of the midwestern united states, 2006–2015. *ISPRS International Journal of Geo-Information*, 8(5):240, 2019.

[176] Yuksel Cakir, Murvet Kirci, and Ece Olcay Gunes. Yield prediction of wheat in south-east region of turkey by using artificial neural networks. In *2014 The Third International Conference on Agro-Geoinformatics*, pages 1–4. IEEE, 2014.

[177] Saeed Khaki and Lizhi Wang. Crop yield prediction using deep neural networks. *Frontiers in plant science*, 10:621, 2019.

[178] Alberto Gonzalez-Sanchez, Juan Frausto-Solis, and Waldo Ojeda-Bustamante. Predictive ability of machine learning methods for massive crop yield prediction. *Spanish Journal of Agricultural Research*, 12(2):313–328, 2014.

[179] Anna X Wang, Caelin Tran, Nikhil Desai, David Lobell, and Stefano Ermon. Deep transfer learning for crop yield prediction with remote sensing data. In *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*, pages 1–5, 2018.

[180] USDA. National agricultural statistics service. *United States Department of Agriculture*, 2013.

[181] Christopher Daly and Kirk Bryant. The prism climate and weather system: an introduction. *Northwest Alliance for Computational Science and Engineering. Oregon State University, Corvallis, USA.*, 2017.

[182] Youlong Xia, Kenneth Mitchell, Michael Ek, Justin Sheffield, Brian Cosgrove, Eric Wood, Lifeng Luo, Charles Alonge, Helin Wei, Jesse Meng, et al. Continental-scale water and energy flux analysis and validation for the north american land data assimilation system project phase 2 (nldas-2): 1. intercomparison and application of model products. *Journal of Geophysical Research: Atmospheres*, 117(D3), 2012.

[183] Soil Survey Staff. Gridded soil survey geographic (gssurgo) database for the conterminous united states., 2020.

[184] Soil Survey. Soil texture calculator, 2021.

[185] Collin H Homer, Joyce A Fry, Christopher A Barnes, et al. The national land cover database. *US geological survey fact sheet*, 3020(4):1–4, 2012.

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

[187] Yue Liu, Tianlu Zhao, Wangwei Ju, and Siqi Shi. Materials discovery and design using machine learning. *Journal of Materiomics*, 3(3):159–177, 2017.

[188] Jonathan Schmidt, Mário RG Marques, Silvana Botti, and Miguel AL Marques. Recent advances and applications of machine learning in solid-state materials science. *npj Computational Materials*, 5(1):1–36, 2019.

[189] Tian Xie and Jeffrey C Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical review letters*, 120(14):145301, 2018.

[190] Olexandr Isayev, Corey Oses, Cormac Toher, Eric Gossett, Stefano Curtarolo, and Alexander Tropsha. Universal fragment descriptors for predicting properties of inorganic crystals. *Nature communications*, 8(1):1–12, 2017.

[191] Maarten De Jong, Wei Chen, Randy Notestine, Kristin Persson, Gerbrand Ceder, Anubhav Jain, Mark Asta, and Anthony Gamst. A statistical learning framework for materials science: application to elastic moduli of k-nary inorganic polycrystalline compounds. *Scientific reports*, 6(1):1–11, 2016.

[192] Ya Zhuo, Aria Mansouri Tehrani, and Jakoah Brgoch. Predicting the band gaps of inorganic solids by machine learning. *The journal of physical chemistry letters*, 9(7):1668–1673, 2018.

[193] MT Eismann. Spectral properties of materials'. *Hyperspectral Remote Sensing*, pages 133–198, 2012.

[194] R Osborn, EA Goremychkin, AI Kolesnikov, and DG Hinks. Phonon density of states in mgb 2. *Physical Review Letters*, 87(1):017005, 2001.

[195] Joseph P Heremans, Vladimir Jovovic, Eric S Toberer, Ali Saramat, Ken Kurosaki, Anek Charoenphakdee, Shinsuke Yamanaka, and G Jeffrey Snyder. Enhancement of thermoelectric efficiency in pbte by distortion of the electronic density of states. *Science*, 321(5888):554–557, 2008.

[196] Krzysztof Parlinski. Computing for materials, 2010.

[197] M Ali Omar. *Elementary solid state physics: principles and applications*. Pearson Education India, 1975.

[198] Richard M Martin. *Electronic structure: basic theory and practical methods*. Cambridge university press, 2020.

[199] Chi Chen, Weike Ye, Yunxing Zuo, Chen Zheng, and Shyue Ping Ong. Graph networks as a universal machine learning framework for molecules and crystals. *Chemistry of Materials*, 31(9):3564–3572, 2019.

[200] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[201] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. Masked label prediction: Unified message passing model for semi-supervised classification. *arXiv preprint arXiv:2009.03509*, 2020.

[202] Jingyu Zhao, Feiqing Huang, Jia Lv, Yanjie Duan, Zhen Qin, Guodong Li, and Guangjian Tian. Do rnn and lstm have long memory? In *International Conference on Machine Learning*, pages 11365–11375. PMLR, 2020.

[203] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[204] Lev M Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR computational mathematics and mathematical physics*, 7(3):200–217, 1967.

[205] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[206] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[207] Yongtae Kim, Youngsoo Kim, Charles Yang, Kundo Park, Grace X Gu, and Seunghwa Ryu. Deep learning framework for material design space

exploration using active transfer learning and data augmentation. *npj Computational Materials*, 7(1):1–7, 2021.

[208] Vıctor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021.

[209] Zhantao Chen, Nina Andrejevic, Tess Smidt, Zhiwei Ding, Qian Xu, Yen-Ting Chi, Quynh T Nguyen, Ahmet Alatas, Jing Kong, and Mingda Li. Direct prediction of phonon density of states with euclidean neural networks. *Advanced Science*, 8(12):2004214, 2021.

[210] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning so (3) equivariant representations with spherical cnns. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 52–68, 2018.

[211] Dhairya Kumar. Introduction to data preprocessing in machine learning, 2018.

[212] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[213] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[214] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[215] Jiang Wang, Yi Yang, Junhua Mao, Zhiheng Huang, Chang Huang, and Wei Xu. Cnn-rnn: A unified framework for multi-label image classification. In *CVPR*, 2016.

[216] Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit Dhillon. X-bert: extreme multi-label text classification with using bidirectional encoder representations from transformers. *arXiv preprint arXiv:1905.02331*, 2019.

[217] Guoxian Yu, Huzefa Rangwala, Carlotta Domeniconi, Guoji Zhang, and Zhiwen Yu. Protein function prediction using multilabel ensemble classification. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2013.

[218] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2009.

[219] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7):2038–2048, 2007.

[220] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse local embeddings for extreme multi-label classification. *Advances in neural information processing systems*, 28:730–738, 2015.

[221] Chih-Kuan Yeh, Wei-Chieh Wu, Wei-Jen Ko, and Yu-Chiang Frank Wang. Learning deep latent space for multi-label classification. In *AAAI*, 2017.

[222] Junwen Bai, Shufeng Kong, and Carla Gomes. Disentangled variational autoencoder based multi-label classification with covariance-aware multivariate probit model. *IJCAI*, 2020.

[223] Vijaya Kumar Sundar, Shreyas Ramakrishna, Zahra Rahiminasab, Arvind Easwaran, and Abhishek Dubey. Out-of-distribution detection in multi-label datasets using latent space of β-vae. *arXiv preprint arXiv:2003.08740*, 2020.

[224] Nat Dilokthanakul, Pedro AM Mediano, Marta Garnelo, Matthew CH Lee, Hugh Salimbeni, Kai Arulkumaran, and Murray Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016.

[225] Mike Wu and Noah Goodman. Multimodal generative models for scalable weakly-supervised learning. In *Advances in Neural Information Processing Systems*, 2018.

[226] Wei Bi and James Kwok. Multilabel classification with label correlations and missing labels. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2014.

[227] David Belanger and Andrew McCallum. Structured prediction energy networks. In *International Conference on Machine Learning*, 2016.

[228] Zhao-Min Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. Multi-label image recognition with graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5177–5186, 2019.

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

[230] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for

sequential data. *Advances in neural information processing systems*, 28:2980–2988, 2015.

[231] SM Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Koray Kavukcuoglu, and Geoffrey E Hinton. Attend, infer, repeat: Fast scene understanding with generative models. *arXiv preprint arXiv:1603.08575*, 2016.

[232] Rui Shu. Gaussian mixture vae: Lessons in variational inference, generative models, and deep nets, 2016.

[233] Yuge Shi, Narayanaswamy Siddharth, Brooks Paige, and Philip HS Torr. Variational mixture-of-experts autoencoders for multi-modal deep generative models. *arXiv preprint arXiv:1911.03393*, 2019.

[234] Yuge Shi, Brooks Paige, Philip HS Torr, and N Siddharth. Relating by contrasting: A data-efficient framework for multimodal generative models. *arXiv preprint arXiv:2007.01179*, 2020.

[235] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020.

[236] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2021.

[237] Feng Wang, Huaping Liu, Di Guo, and Sun Fuchun. Unsupervised representation learning by invariance propagation. *Advances in Neural Information Processing Systems*, 33:3510–3520, 2020.

[238] Jakub Tomczak and Max Welling. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223. PMLR, 2018.

[239] Prince Zizhuang Wang and William Yang Wang. Neural gaussian copula for variational autoencoder. *arXiv preprint arXiv:1909.03569*, 2019.

[240] Zachary Seymour and Zhongfei Zhang. Multi-label triplet embeddings for image annotation from user-generated tags. In *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*, pages 249–256, 2018.

[241] Oluwasanmi Koyejo, Nagarajan Natarajan, Pradeep Ravikumar, and Inderjit S Dhillon. Consistent multilabel classification. In *NIPS*, volume 29, pages 3321–3329, 2015.

[242] Stijn Decubber, Thomas Mortier, Krzysztof Dembczyński, and Willem Waegeman. Deep f-measure maximization in multi-label classification: A comparative study. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 290–305. Springer, 2018.

[243] Edgar Schönfeld, Sayna Ebrahimi, Samarth Sinha, Trevor Darrell, and Zeynep Akata. Generalized zero-and few-shot learning via aligned variational autoencoders. In *CVPR*. IEEE, 2019.

[244] Chen Chen, Haobo Wang, Weiwei Liu, Xingyuan Zhao, Tianlei Hu, and Gang Chen. Two-stage label embedding via neural factorization machine for multi-label classification. In *AAAI*, 2019.

[245] Mikołaj Małkiński and Jacek Mańdziuk. Multi-label contrastive learning for abstract visual reasoning. *arXiv preprint arXiv:2012.01944*, 2020.

[246] Son D. Dao, Zhao Ethan, Phung Dinh, and Cai Jianfei. Contrast learning visual attention for multi label classification. *arXiv preprint arXiv:2107.11626*, 2021.

[247] Jiaming Song and Stefano Ermon. Multi-label contrastive predictive coding. *Advances in Neural Information Processing Systems*, 33, 2020.

[248] Mauro Annarumma and Giovanni Montana. Deep metric learning for multi-labelled radiographs. *arXiv preprint arXiv:1712.07682*, 2017.

[249] Franca Debole and Fabrizio Sebastiani. An analysis of the relative hardness of reuters-21578 subsets. *Journal of the American Society for Information Science and technology*, 56(6):584–596, 2005.

[250] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. Multilabel text classification for automated tag suggestion. *ECML PKDD Discovery Challenge*, page 75, 2008.

[251] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Effective and efficient multilabel classification in domains with large number of labels. In *Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08)*, volume 21, pages 53–59, 2008.

[252] Michael Kuhn, Ivica Letunic, Lars Juhl Jensen, and Peer Bork. The sider database of drugs and side effects. *Nucleic acids research*, 44(D1):D1075–D1079, 2016.

[253] Kenta Nakai and Minoru Kanehisa. A knowledge base for predicting protein localization sites in eukaryotic cells. *Genomics*, 14(4):897–911, 1992.

[254] D Fink, T Auer, F Obregon, WM Hochachka, M Iliff, B Sullivan, C Wood,

I Davies, and S Kelling. The ebird reference dataset version 2016 (erd2016), 2017.

[255] Lifu Tu and Kevin Gimpel. Learning approximate inference networks for structured prediction. *arXiv preprint arXiv:1803.03376*, 2018.

[256] Tal Ridnik, Emanuel Ben-Baruch, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. Asymmetric loss for multi-label classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 82–91, 2021.

[257] Walter Gerych, Tom Hartvigsen, Luke Buquicchio, Emmanuel Agu, and Elke A Rundensteiner. Recurrent bayesian classifier chains for exact multi-label classification. *Advances in Neural Information Processing Systems*, 34:15981–15992, 2021.

[258] Fernando Benites and Elena Sapozhnikova. Haram: a hierarchical aram neural network for large-scale text classification. In *2015 IEEE international conference on data mining workshop (ICDMW)*. IEEE, 2015.

[259] Min-Ling Zhang, Yu-Kun Li, Xu-Ying Liu, and Xin Geng. Binary relevance for multi-label learning: an overview. *Frontiers of Computer Science*, 12(2):191–202, 2018.

[260] A Johnston, WM Hochachka, ME Strimas-Mackey, V Ruiz Gutierrez, OJ Robinson, ET Miller, T Auer, ST Kelling, and D Fink. Best practices for making reliable inferences from citizen science data: case study using ebird to estimate species distributions. *BioRxiv*, page 574392, 2019.

[261] Junwen Bai, Weiran Wang, Yingbo Zhou, and Caiming Xiong. Repre-

sentation learning for sequence data with deep autoencoding predictive components. In *International Conference on Learning Representations*, 2021.

[262] Junwen Bai, Weiran Wang, and Carla P Gomes. Contrastively disentangled sequential variational autoencoder. *Advances in Neural Information Processing Systems*, 34:10105–10118, 2021.

[263] Junwen Bai, Bo Li, Yu Zhang, Ankur Bapna, Nikhil Siddhartha, Khe Chai Sim, and Tara N Sainath. Joint unsupervised and supervised training for multilingual asr. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6402–6406. IEEE, 2022.

[264] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *ICLR*, 2018.

[265] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[266] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, 2018.

[267] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert:

Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.

[268] Santiago Pascual, Mirco Ravanelli, Joan Serrà, Antonio Bonafonte, and Yoshua Bengio. Learning problem-agnostic speech representations from multiple self-supervised tasks. *arXiv preprint arXiv:1904.03416*, 2019.

[269] Yu-An Chung and James Glass. Generative pre-training for speech with autoregressive predictive coding. In *ICASSP*, 2020.

[270] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[271] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[272] Sherjil Ozair, Corey Lynch, Yoshua Bengio, Aaron Van den Oord, Sergey Levine, and Pierre Sermanet. Wasserstein dependency measure for representation learning. In *NIPS*, 2019.

[273] Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. In *ICLR*, 2019.

[274] David Clark, Jesse Livezey, and Kristofer Bouchard. Unsupervised discovery of temporal structure in noisy data with dynamical components analysis. In *NIPS*, 2019.

[275] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals. Listen, attend and spell: A

neural network for large vocabulary conversational speech recognition. In *ICASSP*, 2016.

[276] Ming Li and Paul Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer, 2008.

[277] Peter Dayan and Laurence F Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems*. Computational Neuroscience Series, 2001.

[278] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 2004.

[279] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *ICML*, 2013.

[280] Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On deep multi-view representation learning. In *ICML*, 2015.

[281] Yu-An Chung, Wei-Ning Hsu, Hao Tang, and James Glass. An unsupervised autoregressive model for speech representation learning. *Interspeech*, 2019.

[282] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[283] Rui Shu, Hung H. Bui, Shengjia Zhao, Mykel J. Kochenderfer, and Stefano Ermon. Amortized inference regularization. In *NIPS*, 2018.

[284] 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*, pages 1878–1889, 2017.

[285] Yingzhen Li and Stephan Mandt. Disentangled sequential autoencoder. *arXiv preprint arXiv:1803.02991*, 2018.

[286] Anthony J Bell and Terrence J Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 1995.

[287] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

[288] David McAllester and Karl Stratos. Formal limitations on the measurement of mutual information. In *AISTATS*, 2020.

[289] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. A. Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 2010.

[290] Alexei Baevski, Michael Auli, and Abdelrahman Mohamed. Effectiveness of self-supervised pre-training for speech recognition. *arXiv preprint arXiv:1911.03912*, 2019.

[291] Dongwei Jiang, Xiaoning Lei, Wubo Li, Ne Luo, Yuxuan Hu, Wei Zou, and Xiangang Li. Improving transformer-based speech recognition using unsupervised pre-training. arXiv:1910.09932 [cs.CL], 2019.

[292] J. Chorowski, R. J. Weiss, S. Bengio, and A. V. D. Oord. Unsupervised speech representation learning using wavenet autoencoders. *IEEE Trans. Audio, Speech and Language Process.*, 2019.

[293] Y-A. Chung, W-N. Hsu, H. Tang, and J. Glass. An unsupervised autoregressive model for speech representation learning. In *Interspeech*, 2019.

[294] Xingchen Song, Guangsen Wang, Zhiyong Wu, Yiheng Huang, Dan Su, Dong Yu, and Helen Meng. Speech-xlnet: Unsupervised acoustic model pretraining for self-attention networks. *ArXiv*, 2019.

[295] Shaoshi Ling, Yuzong Liu, Julian Salazar, and Katrin Kirchhoff. Deep contextualized acoustic representations for semi-supervised speech recognition. In *ICASSP*, 2020.

[296] Andy T Liu, Shang-Wen Li, and Hung-yi Lee. Tera: Self-supervised learning of transformer encoder representation for speech. *arXiv preprint arXiv:2007.06028*, 2020.

[297] S. Pascual, M. Ravanelli, J. Serrà, A. Bonafonte, and Y. Bengio. Learning problem-agnostic speech representations from multiple self-supervised tasks. In *Interspeech*, 2019.

[298] Mirco Ravanelli, Jianyuan Zhong, Santiago Pascual, Pawel Swietojanski, Joao Monteiro, Jan Trmal, and Yoshua Bengio. Multi-task self-supervised learning for robust speech recognition. In *ICASSP*, 2020.

[299] Steven H Strogatz. *Nonlinear dynamics and chaos with student solutions manual: With applications to physics, biology, chemistry, and engineering*. CRC press, 2018.

[300] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[301] David Beniaguev. Historical hourly weather data 2012-2017, 2017.

[302] Joshua I Glaser, Ari S Benjamin, Raeed H Chowdhury, Matthew G Perich,

Lee E Miller, and Konrad P Kording. Machine learning for neural decoding. *eNeuro*, 2020.

[303] Joseph E. O'Doherty, Mariana M. B. Cardoso, Joseph G. Makin, and Philip N. Sabes. Nonhuman primate reaching with multichannel sensorimotor cortex electrophysiology: Broadband for indy_20160630_01, 2018.

[304] Laurenz Wiskott and Terrence Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural computation*, 2002.

[305] Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 2012.

[306] Douglas B. Paul and Janet M. Baker. The design for the wall street journal-based CSR corpus. In *Proceedings of the workshop on Speech and Natural Language*, 1992.

[307] Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. ESPnet: End-to-end speech processing toolkit. In *Interspeech*, 2018.

[308] Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, Shinji Watanabe, Takenori Yoshimura, and Wangyou Zhang. A comparative study on transformer vs rnn in speech applications. In *ASRU*, 2019.

[309] Lu Liu and Yiheng Huang. Masked pre-trained encoder base on joint ctc-transformer. *arXiv preprint arXiv:2005.11978*, 2020.

[310] Kazuya Kawakami, Luyu Wang, Chris Dyer, Phil Blunsom, and Aaron van den Oord. Learning robust and multilingual speech representations. *arXiv preprint arXiv:2001.11128*, 2020.

[311] Adam Golinski, Reza Pourreza, Yang Yang, Guillaume Sautiere, and Taco S Cohen. Feedback recurrent autoencoder for video compression. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[312] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. In *International Conference on Learning Representations*, 2017.

[313] Emily L Denton and vighnesh Birodkar. Unsupervised learning of disentangled representations from video. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, 2017.

[314] Yizhe Zhu, Mohamed Elhoseiny, Bingchen Liu, Xi Peng, and Ahmed Elgammal. A generative adversarial approach for zero-shot learning from noisy texts. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1004–1013, 2018.

[315] Yu Tian, Jian Ren, Menglei Chai, Kyle Olszewski, Xi Peng, Dimitris N Metaxas, and Sergey Tulyakov. A good image generator is what you need for high-resolution video synthesis. In *International Conference on Learning Representations*, 2020.

[316] Jun Han, Martin Renqiang Min, Ligong Han, Li Erran Li, and Xuan Zhang. Disentangled recurrent wasserstein autoencoder. In *International Conference on Learning Representations*, 2021.

[317] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 29, pages 2172–2180, 2016.

[318] 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. *International Conference on Learning Representations*, 2016.

[319] Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, 2017.

[320] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, pages 2649–2658. PMLR, 2018.

[321] Ricky TQ Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *NIPS*, 2018.

[322] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations.

In *International Conference on Machine Learning*, pages 4114–4124. PMLR, 2019.

[323] Ilyes Khemakhem, Diederik Kingma, Ricardo Monti, and Aapo Hyvarinen. Variational autoencoders and nonlinear ica: A unifying framework. In *International Conference on Artificial Intelligence and Statistics*, pages 2207–2217. PMLR, 2020.

[324] Francesco Locatello, Ben Poole, Gunnar Rätsch, Bernhard Schölkopf, Olivier Bachem, and Michael Tschannen. Weakly-supervised disentanglement without compromises. In *International Conference on Machine Learning*, pages 6348–6359. PMLR, 2020.

[325] Anirudh Goyal, Alessandro Sordoni, Marc-Alexandre Côté, Nan Rosemary Ke, and Yoshua Bengio. Z-forcing: Training stochastic recurrent networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[326] Rahul G Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

[327] Yizhe Zhu, Martin Renqiang Min, Asim Kadav, and Hans Peter Graf. S3vae: Self-supervised sequential vae for representation disentanglement and data generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6538–6547, 2020.

[328] Samuel Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, 2016.

[329] Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. Fixing a broken elbo. In *ICLR*, 2018.

[330] Hiroshi Takahashi, Tomoharu Iwata, Yuki Yamanaka, Masanori Yamada, and Satoshi Yagi. Variational autoencoder with implicit optimal priors. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5066–5073, 2019.

[331] Michael Tschannen, Olivier Bachem, and Mario Lucic. Recent advances in autoencoder-based representation learning. *arXiv preprint arXiv:1812.05069*, 2018.

[332] Kei Akuzawa, Yusuke Iwasawa, and Yutaka Matsuo. Information-theoretic regularization for learning global features by sequential vae. *Machine Learning*, 110(8):2239–2266, 2021.

[333] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. In *NIPS*, 2019.

[334] Werner Verhelst and Marc Roelands. An overlap-add technique based on waveform similarity (wsola) for high quality time-scale modification of speech. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 554–557. IEEE, 1993.

[335] Jonathan Driedger and Meinard Müller. A review of time-scale modification of music signals. *Applied Sciences*, 6(2):57, 2016.

[336] 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*, volume 27, 2014.

[337] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1526–1535. IEEE Computer Society, 2018.

[338] Pengyu Cheng, Martin Renqiang Min, Dinghan Shen, Christopher Malon, Yizhe Zhang, Yitong Li, and Lawrence Carin. Improving disentangled text representation learning with information-theoretic guidance. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7530–7541, 2020.

[339] Aapo Hyvarinen and Hiroshi Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ica. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, 2016.

[340] Aapo Hyvarinen and Hiroshi Morioka. Nonlinear ica of temporally dependent stationary sources. In *Artificial Intelligence and Statistics*, pages 460–469. PMLR, 2017.

[341] Luigi Gresele, Paul K Rubenstein, Arash Mehrjou, Francesco Locatello, and Bernhard Schölkopf. The incomplete rosetta stone problem: Identifiability results for multi-view nonlinear ica. In *Uncertainty in Artificial Intelligence*, pages 217–227. PMLR, 2020.

[342] Scott E Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. Deep visual analogy-making. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and

R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, 2015.

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

[344] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *International Conference on Machine Learning*, pages 1174–1183. PMLR, 2018.

[345] J. Garofolo, Lori Lamel, W. Fisher, Jonathan Fiscus, D. Pallett, N. Dahlgren, and V. Zue. Timit acoustic-phonetic continuous speech corpus. *Linguistic Data Consortium*, 1992.

[346] Zhehuai Chen, Yu Zhang, Andrew Rosenberg, Bhuvana Ramabhadran, Gary Wang, and Pedro Moreno. Injecting text in self-supervised speech pretraining. *arXiv preprint arXiv:2108.12226*, 2021.

[347] Yu Zhang, James Qin, Daniel S Park, Wei Han, Chung-Cheng Chiu, Ruoming Pang, Quoc V Le, and Yonghui Wu. Pushing the limits of semi-supervised learning for automatic speech recognition. *arXiv preprint arXiv:2010.10504*, 2020.

[348] Junwen Bai, Weiran Wang, Yingbo Zhou, and Caiming Xiong. Representation learning for sequence data with deep autoencoding predictive components. In *International Conference on Learning Representations*, 2020.

[349] Yu-An Chung, Yu Zhang, Wei Han, Chung-Cheng Chiu, James Qin, Ruoming Pang, and Yonghui Wu. W2v-bert: Combining contrastive learning

and masked language modeling for self-supervised speech pre-training. *arXiv preprint arXiv:2108.06209*, 2021.

[350] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *arXiv preprint arXiv:2106.07447*, 2021.

[351] Samuel Kessler, Bethan Thomas, and Salah Karout. Continual-wav2vec2: an application of continual learning for self-supervised automatic speech recognition. *arXiv preprint arXiv:2107.13530*, 2021.

[352] Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems*, 32:7059–7069, 2019.

[353] Morgane Riviere, Armand Joulin, Pierre-Emmanuel Mazaré, and Emmanuel Dupoux. Unsupervised pretraining transfers well across languages. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7414–7418. IEEE, 2020.

[354] Alexis Conneau, Alexei Baevski, Ronan Collobert, Abdelrahman Mohamed, and Michael Auli. Unsupervised cross-lingual representation learning for speech recognition. *arXiv preprint arXiv:2006.13979*, 2020.

[355] Alex Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

[356] Yong Cheng, Wei Wang, Lu Jiang, and Wolfgang Macherey. Self-supervised and supervised joint training for resource-rich machine translation. *arXiv preprint arXiv:2106.04060*, 2021.

[357] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. Don't stop pretraining: adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*, 2020.

[358] Chaitanya Talnikar, Tatiana Likhomanenko, Ronan Collobert, and Gabriel Synnaeve. Joint masked CPC and CTC training for ASR. *arXiv preprint arXiv:2011.00093*, 2020.

[359] Chengyi Wang, Yu Wu, Yao Qian, Kenichi Kumatani, Shujie Liu, Furu Wei, Michael Zeng, and Xuedong Huang. Unispeech: Unified speech representation learning with labeled and unlabeled data. In *2021 International Conference on Machine Learning*, July 2021.

[360] Srinivasa Raghavan and Kumar Shubham. Hybrid unsupervised and supervised multitask learning for speech recognition in low resource languages. In *Proc. Workshop on Machine Learning in Speech and Language Processing*, 2021.

[361] Dongseong Hwang, Ananya Misra, Zhouyuan Huo, Nikhil Siddhartha, Shefali Garg, David Qiu, Khe Chai Sim, Trevor Strohman, Françoise Beaufays, and Yanzhang He. Large-scale asr domain adaptation using self- and semi-supervised learning. *arXiv preprint arXiv:2110.00165*, 2021.

[362] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[363] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image simi-

larity with deep ranking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1386–1393, 2014.