UTILIZING NONNEGATIVE TENSOR FACTORIZATION METHODS FOR INFERENCE, MODEL SELECTION, AND ANALYSIS IN SUPERVISED LEARNING

by

Melih Barsbey

B.A., Psychology, Boğaziçi University, 2011

M.A., Psychology, Boğaziçi University, 2014

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2023

ACKNOWLEDGEMENTS

I would first like to express my deep gratitude to Ali Taylan Cemgil, who I feel very lucky to have had as an advisor. His expertise, creativity, and his rigorous yet pragmatic approach to research has been a constant source of learning and inspiration. Joining his vibrant group has been one of my best decisions, and I look forward to enjoying his company as both a friend and a colleague in the years ahead. I also thank my brilliant co-advisor Arzucan Özgür, who has been an exceptional role model in combining professional competence and personal warmth; without her support I could not have completed my studies. I would also like to acknowledge the wonderful members of my thesis committee, Fikret Gürgen, Tunga Güngör, Sinan Yıldırım, and Tolga Birdal; their feedback helped improve this text considerably.

I thank my brilliant colleagues and mentors for generously sharing their time and experience with me: Mehmet Burak Kurutmaz, Umut Şimşekli, Haluk O. Bingöl, A. Caner Türkmen, Gökhan Çapan, S. Adil Sarıbay, Mine Öğretir, Yaman Kındap, Merve Ünlü, Onur Poyraz, David Stutz, Mürsel Taşgın, and Uzay Çetin, among many others. I am also grateful for the invaluable support I received from dear friends: Mehmet Emin Boyacıoğlu, Aylin Deniz Ülkümen, Mustafa Yavaş, Çağlar Çömez, Mine Melodi Çalışkan, Burcu Pehlivan, Lâle Alpar, Serhat Çelikel, Dilek Yücel, Levent Dinçkal, and Göktürk Kuru. I thank my family for inspiring me to learn, and for their love, support, and patience throughout my education. I extend my deepest gratitude to Feriha Kafa, Fikriye Barsbey, and Semiha Turhan; their loving memories still brighten our lives. I finally thank my little Felix, whose abi misses him very much. His love for life will always be inspiring us, long beyond his recent passing.

I am grateful to TÜBİTAK for supporting my studies in the form of scholarship (BİDEB 2211) and research project funding (ARDEB Project No. 116E580). I thank DeepMind London for hosting me during the last stretch of my studies. I also thank my department, institute, and university for their support throughout the program.

ABSTRACT

UTILIZING NONNEGATIVE TENSOR FACTORIZATION METHODS FOR INFERENCE, MODEL SELECTION, AND ANALYSIS IN SUPERVISED LEARNING

This thesis focuses on utilizing nonnegative tensor factorization (NTF) methods in various areas of supervised learning. We start with the introduction of a probabilistic NTF framework that can accommodate a wide range of modeling assumptions while maintaining algorithmic efficiency during inference. The flexibility provided by this framework is then utilized for inference, model selection, and analysis in various supervised learning problems. In the first of these scenarios, we use this approach to effectively model time series with nested, complex seasonalities, ensuring accuracy and interpretability. We then propose a novel method for learning to defer to an expert based on the output of a machine learning model in classification problems, and show that NTF can be utilized to extend this method to arbitrarily complex settings. Afterwards, we investigate when and why deep neural networks' parameters become compressible, and use the aforementioned NTF framework to help analyze how these dynamics are reflected in the representation space. In addition to making independent contributions to various areas of supervised learning, our work shows that, coupled with a convenient modeling approach, NTF can be beneficial for a wide range of supervised learning problems.

.

ÖZET

NEGATİF OLMAYAN TENSÖR AYRIŞTIRMA YÖNTEMLERİNİN GÖZETİMLİ ÖĞRENMEDE ÇIKARIM, MODEL SEÇİMİ VE ANALİZ İÇİN KULLANILMASI

Bu tez, negatif olmayan tensör ayrıştırma (NOTA) yöntemlerinin gözetimli öğrenmenin değişik alanlarında kullanılmasına odaklanmaktadır. Teze, birçok modelleme varsayımını, çıkarımda algoritmik verimlilikle birleştirerek hayata geçirebilen olasılıksal bir NOTA çerçevesini tanıtarak başlamaktayız. Ardından bu çerçevenin sağladığı esneklik, farklı gözetimli öğrenme problemlerinde çıkarım, model seçimi ve analiz için kullanılmaktadır. Bu uygulamaların ilkinde bu yaklaşımdan çok seviyeli, karmaşık mevsimsellik gösteren zaman serilerini yorumlanabilir ve doğru bir şekilde modellemek için faydalanılmaktadır. Ardından, bir sınıflandırma probleminde, bir yapay öğrenme modelinin çıktısına bağlı olarak son kararın ne zaman bir uzmana bırakılması gerektiğinin belirlenmesi için yeni bir yöntem önerilmekte ve NOTA modellerinin bu yöntemi daha karmaşık senaryolara genellemekte kullanılabileceği gösterilmektedir. Sonrasında, derin öğrenme yöntemlerinin parametrelerinin ne zaman ve neden sıkıştırılabilir olduğu incelenmekte ve yukarıda bahsedilen NOTA çerçevesinden bu eğilimlerin gösterim uzayındaki karışılıklarının incelenmesinde yararlanılmaktadır. Tezdeki çalışmalar, gözetimli öğrenmenin değişik alanlarına bağımsız katkılar yapmanın yanı sıra, esnek bir modelleme yaklaşımıyla birleştirildiğinde, NOTA yöntemlerinin birçok gözetimli öğrenme problemini kolaylaştıracak şekilde kullanılabileceğini göstermektedir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES xi
LIST OF TABLES xv
LIST OF SYMBOLS xvii
LIST OF ACRONYMS/ABBREVIATIONS xviii
1. INTRODUCTION 1
1.1. Contributions 2
1.2. Technical Preliminaries 4
1.2.1. Notation 4
1.2.2. Nonnegative Matrix/Tensor Factorization 4
1.2.3. Probabilistic Approaches to NMF and NTF 7
1.3. Bayesian Allocation Model 8
1.3.1. The Model 9
1.3.2. Inference and Model Selection 11
1.3.2.1. Variational Inference 12
1.3.2.2. Sequential Monte Carlo 13
1.3.2.3. Inference with BAM in Applications 14
1.3.3. Hyperparameters 15
1.4. Reuse of Previously Published Material 16
2. LOW-RANK HIERARCHICAL SEASONALITY 17
2.1. Motivation and Background 18
2.1.1. Related Work 19
2.2. Low-Rank Hierarchical Seasonality 22
2.2.1. Utilizing Learned LRHS Components 25
2.3. Bayesian Low-Rank Hierarchical Seasonality 25
2.3.1. Interpretable Seasonality with BLRHS 26

2.4. Experiments . . . 28
2.4.1. Analyzing New York City Yellow Taxi Data . . . 29
2.4.1.1. Extracted Seasonal Patterns . . . 29
2.4.2. Analyzing BART Ridership Data . . . 30
2.4.2.1. Extracted Seasonal Patterns . . . 31
2.4.3. Missing Data Imputation in Multivariate Time Series . . . 33
2.4.3.1. Dataset . . . 34
2.4.3.2. Experimental Settings, Model Selection, and Metrics . 35
2.4.3.3. Results . . . 36
3. LEARNING TO DEFER IN BINARY CLASSIFICATION . . . 39
3.1. Learning to Defer . . . 40
3.1.1. Notation . . . 41
3.1.2. Problem Statement . . . 41
3.1.3. Related Work . . . 42
3.1.4. Learning to Defer under Constraints . . . 45
3.2. Complementarity-Driven Deferral to Clinicians (CoDoC) . . . 46
3.2.1. Density Estimation . . . 48
3.2.1.1. Threshold Search . . . 49
3.3. Experiments . . . 50
3.3.1. Baselines . . . 51
3.3.2. Hyperparameters, Model Selection, and Metrics . . . 51
3.3.3. Datasets and Diagnostic AI Systems . . . 52
3.3.4. Results . . . 53
3.3.4.1. Limits of Confidence-Based Deferral . . . 55
3.3.4.2. Comparison of CoDoC with Threshold Search . . . 56
3.4. Discussion . . . 57
4. LEARNING TO DEFER IN ADVANCED SCENARIOS . . . 59
4.1. Generative Model for Learning to Defer . . . 59
4.2. Modeling Various Deferral and Annotation Scenarios . . . 61
4.2.1. Multiple Machine Learning Models and Annotators . . . 62
4.2.2. Single Annotator Assumption . . . 62

4.2.3. Ensembling Machine Learning Models . . . 63
4.2.4. Dawid-Skene Aggregation Model . . . 63
4.3. Learning to Defer under Arbitrary Modeling Scenarios . . . 64
4.3.1. Optimal Decision Making . . . 65
4.3.2. Discretization of Model Outputs . . . 66
4.4. Experiments . . . 67
4.4.1. Dataset . . . 67
4.4.2. Experimental Settings . . . 68
4.4.3. Results . . . 69
5. COMPRESSIBILITY AND GENERALIZATION IN DEEP LEARNING . . 72
5.1. Motivation and Background . . . 72
5.2. Technical Preliminaries . . . 74
5.2.1. Notation . . . 74
5.2.2. Fully Connected Neural Networks . . . 75
5.2.3. Risk Minimization and SGD . . . 75
5.2.4. Heavy-Tailed Distributions and the α-stable Family . . . 76
5.3. Compressibility and the Heavy-Tailed Mean-Field Regime . . . 77
5.3.1. The Heavy-Tailed Mean-Field Regime . . . 77
5.3.1.1. The Heavy Tail Phenomenon . . . 77
5.3.1.2. Propagation of Chaos and Mean-Field Limits . . . 77
5.3.2. Compression Algorithms . . . 79
5.3.2.1. Magnitude Pruning . . . 80
5.3.2.2. Singular Value Pruning . . . 81
5.3.2.3. Node Pruning . . . 81
5.4. Generalization Bounds . . . 82
5.5. Experiments . . . 82
5.5.1. Experimental Settings . . . 83
5.5.1.1. Tail Index Estimation . . . 83
5.5.2. Results . . . 84
5.5.2.1. Training Hyperparameters and Layer Statistics . . . 84
5.5.2.2. Tail Index and Prunability . . . 85

5.5.2.3. Tail Index and Generalization 86
5.6. Compressibility and Adversarial Robustness 86
5.6.1. Robust Risk and Out-of-Distribution Generalization 87
5.6.2. Experiments on Compressibility and Adversarial Robustness . . 90
6. LATENT REPRESENTATIONS AND ROBUST GENERALIZATION . . . 92
6.1. Motivation and Background 92
6.2. Technical Preliminaries 94
6.2.1. Compressibility Metrics 94
6.2.2. Determining Approximate Rank 94
6.3. Experiments 96
6.3.1. Results 98
6.3.1.1. Activation Compressibility and Generalization 98
6.3.1.2. Activation Rank and Generalization 98
6.3.1.3. Out-of-Distribution Generalization 101
6.3.2. Discussion 102
6.4. Analyzing Representations in Drug-Target Affinity Prediction 104
6.4.1. DTA Prediction 105
6.4.2. Causal Invariance in DTA Prediction 106
6.4.3. Robust Generalization in DTA Prediction 108
7. CONCLUSION 109
REFERENCES 112
APPENDIX A: BAYESIAN ALLOCATION MODEL 142
A.1. Basic Probability Distributions 142
A.2. Details of MFVI for BAM 142
A.3. Analyzing ICEWS Dataset with BAM 144
A.3.1. Results 145
APPENDIX B: LOW-RANK HIERARCHICAL SEASONALITY 148
B.1. Additional Details on NYC YT Experiments 148
B.2. Additional Details on BART Experiments 148
B.3. Additional Details on Missing Data Imputation Experiments 149
APPENDIX C: LEARNING TO DEFER 151

C.1. Additional Details on Density Estimation 151
C.1.1. Discretization 151
C.1.2. Smoothing 152
C.1.3. Normalization 152
C.1.4. Overall Density Smoothing Procedure 152
C.2. Hyperparameter Optimization and Model Selection 153
C.3. Details of the Hypothesis Tests 155
C.4. Results for Experiments with SMC 156
APPENDIX D: COMPRESSIBILITY AND GENERALIZATION 157
D.1. Details on Experiment Settings 157
D.1.1. Datasets 157
D.1.2. Models 157
D.1.3. Training and Hyperparameters 158
D.1.4. Tail Index Estimation 159
D.1.5. Pruning Details 159
D.2. Investigation of Tail Index and Prunability with Synthetic Data 160
D.3. Investigating the HML Condition in Synthetic Experiments 162
D.4. Global Magnitude Pruning and Node Pruning Results 163
D.5. Experiments with Ergodic Averaged Parameters 164
D.6. Hardware and Other Resources 165
APPENDIX E: LATENT REPRESENTATIONS AND GENERALIZATION . 167
E.1. Out-of-Distribution Generalization Results 167

LIST OF FIGURES

Figure 1.1. Visualization of the contents of the thesis. 3

Figure 1.2. NMF/NTF and topic models expressed as instances of BAM. . . . 9

Figure 2.1. Folding time series into a tensor. 20

Figure 2.2. BLRHS utilizing CP and Tucker decompositions. 27

Figure 2.3. Synthetic example for BLRHS. 28

Figure 2.4. Changing dynamics of New York City Yellow Taxi rides. 31

Figure 2.5. Breaking down seasonality patterns with BLRHS. 32

Figure 2.6. Breaking down patterns that contribute to BART ridership. 33

Figure 2.7. Total rides and different travel patterns in BART ridership. 34

Figure 2.8. Relating BLRHS temporal patterns to spatial dimensions. 36

Figure 3.1. Mammography results with UKMD1 and MDAI1. 55

Figure 3.2. Mammography results with USMD2 and MDAI2. 56

Figure 3.3. Tuberculosis results with TBCXR and TBAI1-TBAI5. 57

Figure 3.4. Limits of confidence-based deferral. 58

Figure 4.1. Graphical models for observed vs. unobserved input. 60

Figure 4.2. Low-rank assumption on the observed variables. 61

Figure 4.3. M models and E experts. 62

Figure 4.4. M models and E conditionally independent annotators. 63

Figure 4.5. Ensembling among models. 63

Figure 4.6. Dawid-Skene aggregation model. 64

Figure 4.7. Example image from ImageNet-16H. 68

Figure 5.1. $\hat{\alpha}$ and η/b for each trained model. 84

Figure 5.2. Empirical distribution of a CNN layer trained on MNIST. 85

Figure 5.3. $\hat{\alpha}$ and $1 - \kappa$. 85

Figure 5.4. Accuracy and $1 - \kappa$ for magnitude pruning. 86

Figure 5.5. Accuracy and $1 - \kappa$ for singular value pruning. 87

Figure 5.6. Test accuracy and $\hat{\alpha}$. 88

Figure 5.7. Robust accuracy against adversarial examples and $\hat{\alpha}$. 89

Figure 5.8. Robust accuracy against common image corruptions and $\hat{\alpha}$. 90

Figure 6.1. Activation compressibility and accuracy. 99

Figure 6.2. Class-conditional compressibility and accuracy. 100

Figure 6.3. Activation ranks per layers. 101

Figure 6.4. Maximum effective rank and accuracy. 102

Figure 6.5. Class-conditional effective rank and accuracy. 103

Figure 6.6. Maximum average class-conditional rank and accuracy. 104

Figure A.1. Comparing MFVI and SMC on ICEWS data. 147

Figure D.1. Accuracy and $1 - \kappa$ for synthetic networks and data. 162

Figure D.2. α and $1 - \kappa$ for synthetic weight matrices. 163

Figure D.3. Accuracy and $1 - \kappa$ for global magnitude pruning. 164

Figure D.4. Accuracy and $1 - \kappa$ for node pruning in CNNs. 164

Figure D.5. Accuracy and $1 - \kappa$ for node pruning in FCNs. 164

Figure D.6. Accuracy and $1 - \kappa$ for magnitude pruning using $\tilde{\mathbf{x}}$. 165

Figure D.7. Accuracy and $1 - \kappa$ for singular value pruning using $\tilde{\mathbf{x}}$. 166

Figure D.8. Accuracy and $1 - \kappa$ for node pruning using $\tilde{\mathbf{x}}$. 166

Figure E.1. Activation sparsity and accuracy. 168

Figure E.2. Class-conditional sparsity and accuracy. 168

Figure E.3. Class-conditional compressibility and accuracy with min. η. 169

Figure E.4. Class-conditional compressibility and accuracy with max. η. . . . 169

Figure E.5. Class-conditional rank and accuracy with min. η. 170

Figure E.6. Class-conditional rank and accuracy with max. η. 170

Figure E.7. Activation compressibility and OOD accuracy. 171

Figure E.8. Class-conditional compressibility and OOD accuracy. 171

Figure E.9. Maximum effective rank and OOD accuracy. 172

Figure E.10. Class-conditional rank and OOD accuracy. 172

LIST OF TABLES

Table 2.1. Comparing performances in missing data imputation. 37

Table 3.1. Dataset observation counts. 52

Table 3.2. Mammography results with UKMD1 and MDAI1. 53

Table 3.3. Mammography results with USMD1 and MDAI1. 54

Table 3.4. Mammography results with USMD2 and MDAI2. 55

Table 3.5. Tuberculosis experiment results. 58

Table 4.1. Baseline model accuracies in ImageNet-16H dataset. 70

Table 4.2. Multiclass classification with one model and one expert. 70

Table 4.3. Multiclass classification with two models and one expert. 70

Table 6.1. MSE and CI scores for different learning rates in DeepDTA. 108

Table B.1. BART station mappings. 149

Table B.2. Selected hyperparameters for the imputation experiments. 150

Table C.1. Multiclass classification with SMC, one model, one expert. 155

Table C.2. Multiclass classification with SMC, two models, one expert. 155

Table D.1. Number of parameters for model-dataset combinations. 158

Table D.2. The ranges of η/b for all experiments. 159

Table E.1. Learning rates for the experimental settings used in Figure 6.3. . . 167

LIST OF SYMBOLS

$\mathbf{A}$	An N-dimensional tensor with $N \geq 2$
$\mathbf{a}$	A vector in $\mathbb{R}^d$
$\mathcal{D}$	Dirichlet distribution
$\mathbb{E}_{\mathbf{x},\mathrm{y}}$	Expectation according to the respective distributions of $\mathbf{x}$, y
$\mathcal{GA}$	Gamma distribution
$\mathbb{I}[\cdot]$	A function where $\mathbb{I}[S] = 1$ if S is true, 0 otherwise
ℓ	Loss function
$\mathcal{M}$	Multinomial distribution
$p_\alpha(\cdot)$	Probability distribution, parametrized by α
$\mathcal{PO}$	Poisson distribution
$\mathbb{R}^d$	Real coordinate space of dimension d
$\mathcal{R}$	Risk
$\widehat{\mathcal{R}}$	Empirical risk
$\mathcal{R}_{\mathrm{rob}}$	Robust risk
$S\alpha S$	Symmetric α-stable distribution
α	Tail index of a probability distribution
Γ	Gamma function
$\|\mathbf{A}\|$	Frobenius norm of a matrix $\mathbf{A}$
$\|\mathbf{a}\|_p$	A vector $\mathbf{a}$'s ℓ_p (semi-)norm
$\overset{\text{i.i.d.}}{\sim}$	Independent and identical sampling from a distribution
$[N]$	Set of integers $\{1, \ldots, N\}$
∇f	Gradient of function f

LIST OF ACRONYMS/ABBREVIATIONS

BAM	Bayesian Allocation Model
BLRHS	Bayesian Low-Rank Hierarchical Seasonality
CNN	Convolutional Neural Network
CoDoC	Complementarity-Driven Deferral to Clinicians
CP	Canonical Polyadic, CANDECOMP, PARAFAC
DAG	Directed Acyclic Graph
DPMM	Dirichlet Process Mixture Model
DTA	Drug-Target Affinity
ELBO	Evidence Lower Bound
FCN	Fully Connected Network
HML	Heavy-Tailed Mean-Field Condition
ICEWS	Integrated Crisis Early Warning System Dataset
KL	Kullback–Leibler (Divergence)
LRHS	Low-Rank Hierarchical Seasonality
LTD	Learning to Defer
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MCF	Multiple Cyclical Folding
MCMC	Markov Chain Monte Carlo
MFVI	Mean-Field Variational Inference
NMF	Nonnegative Matrix Factorization
NTF	Nonnegative Tensor Factorization
OOD	Out-of-Distribution
RMSE	Root Mean Squared Error
SGD	Stochastic Gradient Descent
SMC	Sequential Monte Carlo

1. INTRODUCTION

Tensor factorization plays a crucial role in data analysis under various fields, including but not limited to statistics, signal processing, and machine learning [1]. The origins of tensor factorization are usually traced back to 1920's [2], and the beginning of its widespread adoption to the research conducted in psychometrics literature in the following decades [3, 4]. In machine learning, it has since become a core modeling technology, with its importance showing no signs of diminishing [5]. However, since tensor factorization is ideally suited to modeling relational data, most research focus on its analysis in an unsupervised learning setting. In this thesis, we not only make independent contributions to various areas of supervised learning, but also show that a particular variant of tensor factorization, namely *nonnegative tensor factorization (NTF)*, can help model selection, inference, and analysis in these areas, especially when enhanced by an innovative probabilistic modeling framework.

This chapter provides an outline of the thesis, presents our notation, and defines a number of central technical notions. Then, it introduces a probabilistic framework for NTF called Bayesian allocation model (BAM), which is the approach we employ throughout the text for formulating NTF models. BAM not only allows expedient inference and model selection through variational and sampling-based approximate inference methods, but also provides a flexible modeling grammar for multidimensional relational data that unifies various NTF methods and topic models. In Chapter 2, we present a novel approach for efficiently modeling complex, hierarchical seasonality in time series, and utilize BAM within this approach to model large-scale time series in an accurate and interpretable manner. This constitutes our first foray into using NTF in a supervised learning task.

Afterwards, Chapters 3 and 4 present a markedly different supervised learning setting, in which NTF turns out to be also effective. The task in question involves creating a system for *learning to defer* to expert opinion, based on the output of one

or more machine learning models. In Chapter 3 we present a robust and reliable probabilistic modeling solution for this task, given one machine learning model and one human expert. The composite performance produced by our proposed method surpasses individual performances of the model *and* the expert, as demonstrated in our clinical binary classification experiments. However, in Chapter 4 we find that it is statistically and computationally challenging to generalize this promising method to problems with multiple classes, experts, and models. Despite the difficulty of this challenge, we show that BAM can greatly facilitate addressing this complex issue. Our experimental results demonstrate the effectiveness of NTF when applied to this supervised learning task.

Having established the utility of NTF for two different supervised learning tasks, in Chapters 5 and 6, we present an additional application, further underscoring the versatility and usefulness of NTF in the realm of supervised learning, as well as making an independent contribution to research on generalization in deep learning models. In Chapter 5, we present our findings on the intricate relationship between learning rate, compressibility, and generalization in overparametrized deep neural networks. We reveal that neural networks trained with larger learning rates are likely to be more compressible, to generalize better, and to be more robust against adversarial attacks. Then, in Chapter 6 we analyze how these dynamics are reflected in the *representation space*, as we investigate how the rank structure and compressibility of hidden layer activations vary in relation to the aforementioned variables. In this pursuit, having access to a convenient probabilistic method for nonnegative factorization allows us to extend our analyses in previously unexplored directions.

1.1. Contributions

Research undertaken in this thesis enriches the literature on probabilistic NTF and its applications, makes distinct contributions to three separate areas of supervised learning, and shows that having access to a flexible framework for the former can lead to novel and creative solutions in the latter. Although the utilization of tensor fac-

torization methods for expediting supervised learning has a rich history [5, 6], NTF methods remain much less explored from this perspective [7, 8]. However, promising recent research has started to utilize NMF/NTF for improving contemporary supervised learning pipelines in innovative ways, hinting at the potential of this research avenue [9–12]. By demonstrating the utility of NTF for supervised learning in novel contexts, we contribute to the body of work in this underexplored research direction as well. In Figure 1.1 we visually summarize how the topics presented in this thesis interrelate. Due to the diversity of the research areas this thesis incorporates, detailed surveys of relevant literature are presented within their respective chapters. We now delineate our notation and key foundational concepts for this thesis, followed by an introduction to BAM.

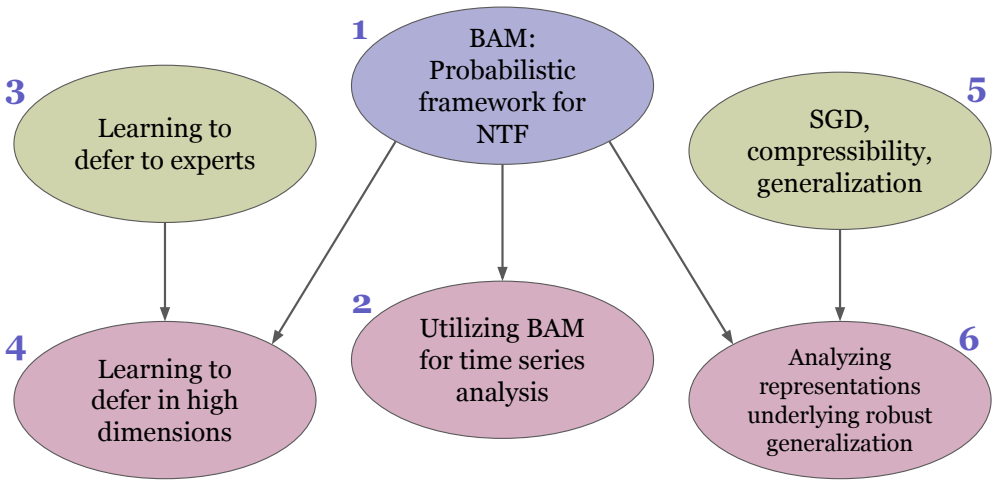


Figure 1.1. Visualization of the contents of the thesis and how they interrelate. The number above each topic corresponds to the chapter in which the topic in question is presented.

1.2. Technical Preliminaries

1.2.1. Notation

The diversity of the topics included in this thesis presents a challenge for notation, as different research areas have their distinct notational conventions. Therefore, while we strive for a consistent notation throughout the thesis, some chapters include their own notation subsections to highlight particularly relevant notation, describe exceptions to those already established, or introduce new ones. Any results from previously published work, either the author's own or otherwise, are expressed in the thesis's established notation convention.

We use lowercase italics for scalars (a, γ), uppercase italics for integers (I, J), lowercase bold letters for vectors $(\mathbf{a}, \mathbf{b})$, and uppercase bold letters for tensors of order $N \geq 2$ $(\mathbf{A}, \mathbf{B})$. a_i refers to the i'th element of a vector $\mathbf{a}$, and $B_{i,j}$ refers to the (i, j)'th element of a matrix $\mathbf{B}$. We denote the elements of tensors using parantheses notation, such that $\mathbf{X}(i, j, k)$ refers to the (i, j, k)'th element of 3-dimensional tensor $\mathbf{X}$. For any positive integer L, we let $[L] := \{1, \ldots, L\}$. We use $i_{[L]} := (i_1, \ldots, i_L)$ to denote an L-tuple of indices. We use this index notation for referring to tensor elements as well, as in $\mathbf{Y}(i_{[L]})$ for the L-dimensional tensor $\mathbf{Y}$.

When the distinction between a *random variable* and its realized value is necessary, the random variable is denoted with regular font and the values it takes are italicized. This is true regardless of the variable's dimensionality, e.g. $p(\mathbf{z} = \boldsymbol{z}, \mathrm{y} = y | \mathbf{X} = \boldsymbol{X})$. We use p to denote a probability or a probability distribution, the intended use being evident from the context. Exceptions to this are explicitly introduced.

1.2.2. Nonnegative Matrix/Tensor Factorization

Matrix factorization aims to approximate a matrix $\mathbf{X} \in \mathbb{R}^{I \times J}$ as product of two or more *factor matrices*, e.g. $\mathbf{W} \in \mathbb{R}^{I \times R}$ and $\mathbf{H} \in \mathbb{R}^{R \times J}$. This approximation can be

expressed as

$$\mathbf{X} \approx \widehat{\mathbf{X}} = \mathbf{WH}. \tag{1.1}$$

Learning the factor matrices involves choosing a suitable *divergence function* D, e.g. Euclidean distance $D_{ED}(\mathbf{X}\|\widehat{\mathbf{X}}) := \|\mathbf{X} - \mathbf{WH}\|$, and a corresponding optimization scheme to find the factors $\mathbf{W}, \mathbf{H}$ that minimize it, e.g. alternating least squares method. The choice of D as well as constraints and/or regularization terms defined on $\mathbf{W}$ and $\mathbf{H}$ (if any) lead to different variants of matrix factorization, each carrying its own inductive bias. The selected optimization method determine the procedure's efficiency, and might contribute its own inductive bias, especially when multiple solutions minimize the divergence function. Matrix factorization is widely used to accomplish a variety of tasks including dimensionality reduction, compression, denoising, anomaly detection, clustering among others. It is utilized for such tasks in fields as diverse as social sciences and signal processing [13–15].

A variant of interest is *nonnegative matrix factorization (NMF)*, which targets nonnegative data, and in which $\mathbf{W}, \mathbf{H}$ are also constrained to have nonnegative entries [16–18]. NMF has been especially popular for the statistical properties of the solutions it produces. For example, let $\mathbf{X} \in \mathbb{R}_{\geq 0}^{M \times N}$ be a matrix that contains M-dimensional (flattened) pixel intensity vectors of N images in its columns. When $\mathbf{X}$ is decomposed this way, one can interpret the columns of the resulting $\mathbf{W}$ as basis or feature vectors and $\mathbf{H}$ as composition coefficients. The nonnegativity constraint would (i) lead the extracted features to be interpretable in the image space, and (ii) lead them to be sparse as they can only be combined additively to approximate the original images [19]. The emergent result of these constraints in the extracted features is at times called "by-parts representation", where the bases produced by NMF resemble a conceptual decomposition [20]. Although a computationally intractable problem, various practical algorithms have been proposed and used since its inception, such as multiplicative updates, alternative least squares, and projected gradient descent [18], [21].

Tensor factorization (TF) and *nonnegative tensor factorization (NTF)* are the respective generalizations of matrix factorization and NMF to N-dimensional arrays (tensors), where $N \geq 2$, and the tensor $\mathbf{X}$ is decomposed into T matrices or tensors [1]. The terms tensor factorization and *tensor decomposition* are often used interchangeably; for the purpose of this thesis we treat them as synonymous. Generalization to N dimensions poses additional statistical and computational challenges, even without the nonnegativity constraint [22]. CP (CANDECOMP/PARAFAC) and Tucker decompositions are two of the most popular of such methods, which can be utilized with or without the nonnegativity constraint [1], [4], [23,24]. We focus on their NTF variants as they are more relevant for our purposes. For a comprehensive review of tensor factorization methods, their variants, frequently used optimization procedures, and other practical and theoretical concerns see [1], [6], [22], [25].

CP decomposition in NTF is the approximation of an N-way nonnegative tensor as the sum of a finite number of nonnegative rank-one N-way tensors. For example, $\mathbf{X} \in \mathbb{R}_{\geq 0}^{I \times J \times K}$ is approximated by the composition of the factor matrices $\mathbf{A} \in \mathbb{R}_{\geq 0}^{I \times R}, \mathbf{B} \in \mathbb{R}_{\geq 0}^{J \times R}, \mathbf{C} \in \mathbb{R}_{\geq 0}^{K \times R}$ such that

$$\mathbf{X} \approx \widehat{\mathbf{X}} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \tag{1.2}$$

where $\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r, r \in \{1, \ldots, R\}$ populate the columns of the factor matrices, and $\circ$ corresponds to an outer product operation. Sometimes factor matrices are assumed to have unit columns and an additional weight vector $\mathbf{w} \in \mathbb{R}_{\geq 0}^{R}$ is included in the model, such that $\mathbf{X} \approx \widehat{\mathbf{X}} = \sum_{r=1}^{R} w_r(\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r)$. Alternatively, Tucker decomposition in NTF approximates the target N-way tensor with an N-way *core tensor* $\mathbf{G}$ and N factor matrices. For example, for $\mathbf{X} \in \mathbb{R}_{\geq 0}^{I \times J \times K}$ we would have

$$\mathbf{X} \approx \widehat{\mathbf{X}} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr}(\mathbf{a}_p \circ \mathbf{b}_q \circ \mathbf{c}_r), \tag{1.3}$$

where g_{pqr} are the entries of the core tensor $\mathbf{G} \in \mathbb{R}_{\geq 0}^{P \times Q \times R}$ and $\mathbf{A} \in \mathbb{R}_{\geq 0}^{I \times P}, \mathbf{B} \in \mathbb{R}_{\geq 0}^{J \times Q}$, and $\mathbf{C} \in \mathbb{R}_{\geq 0}^{K \times R}$ correspond to the factor matrices constructed as above.

Popular selections of divergence for NMF and NTF include Euclidean distance and generalized Kullback–Leibler (KL) divergence (a.k.a. I-divergence) [8]. The former corresponds to

$$D_{\mathrm{ED}}(\mathbf{X}\|\widehat{\mathbf{X}}) := \|\mathbf{X}-\widehat{\mathbf{X}}\| = \left[\sum_{i_{[N]}}\left(\mathbf{X}(i_{[N]})-\widehat{\mathbf{X}}(i_{[N]})\right)^2\right]^{\frac{1}{2}}, \tag{1.4}$$

given the Nth order tensor $\mathbf{X}$ and its approximation $\widehat{\mathbf{X}}$. On the other hand, generalized KL divergence can be expressed as

$$D_{\mathrm{KL}}(\mathbf{X}\|\widehat{\mathbf{X}}) = \sum_{i_{[N]}}\mathbf{X}(i_{[N]})\log\frac{\mathbf{X}(i_{[N]})}{\widehat{\mathbf{X}}(i_{[N]})} - \mathbf{X}(i_{[N]}) + \widehat{\mathbf{X}}(i_{[N]}). \tag{1.5}$$

Adaptations of multiplicative updates and alternating least squares are among commonly used optimization methods in NTF. See [8], [26, 27] for informative reviews on NMF and NTF.

1.2.3. Probabilistic Approaches to NMF and NTF

Probabilistic formulations of NMF and NTF propose a generative model for the observed array $\mathbf{X}$, allowing posterior inference regarding the model parameters, the factors. This can take the form of maximum likelihood estimate (MLE) of the parameters, as well as maximum a posteriori (MAP) estimate or full posterior inference with suitably chosen priors. The formulation of NTF as a probabilistic modeling task allows the rich methodology of Bayesian statistics to be used in this context [28].

Non-probabilistic and probabilistic formulations of NMF/NTF imply equivalent solutions under certain conditions. For example in NMF, minimizing $\|\mathbf{X}-\widehat{\mathbf{X}}\|$ is equivalent to a maximum likelihood estimate under a Gaussian likelihood, $X_{i,j}|\mathbf{WH} \overset{\text{i.i.d.}}{\sim} \mathcal{N}([WH]_{i,j}, 1)$. This correspondence also extends to regularization choices. For example, it is a common practice to include an L2 regularization term for factor matrices in the objective function, which, from a probabilistic perspective is equivalent to assuming an i.i.d. Gaussian prior on the parameters.

Similarly, the solutions implied by assuming a Poisson likelihood in the generative model $X_{i,j}|\mathbf{WH} \overset{\text{i.i.d.}}{\sim} \mathcal{PO}([WH]_{i,j})$ and setting $D = D_{\text{KL}}$ are equivalent, in the case of nonnegative integer data. It has been shown by [29] that the multiplicative update optimization for NMF with KL divergence [18] corresponds to an expectation-maximization procedure for MLE on such a probabilistic model. The formulations that assume Poisson likelihood are also referred to as *Poisson factorization* [30–32].

1.3. Bayesian Allocation Model

We dedicate the rest of the chapter to introducing *Bayesian allocation model (BAM)*, a probabilistic modeling framework for relational count data, which we will utilize to formulate and implement NTF models throughout the thesis. When doing so, in this chapter as well as in the associated Appendix A, we make extensive use of material from the published article [33], with additional content from [34–36], all of which are co-authored by the author of this thesis. See Section 1.4 for a detailed description of how this thesis uses content from the author's previously published research in accordance with publishers' reuse policies. The author of this thesis claims no credit for the original conception of and early research regarding BAM [37, 38]; he is a significant contributor to its later development and applications [33–36].

BAM treats observed data as the marginal of a higher-dimensional latent tensor, and it assumes a low-rank structure on how this tensor is populated, where this structure is parametrized by a discrete directed graphical model. This formulation allows BAM to express not only various NTF methods, but also various topic models [39] within the same framework (Figure 1.2), extending prior research that observed the commonalities among these two model families [40]. In BAM, the prior assumptions on the observations and the discrete probability tables allow a space-efficient mean-field variational inference method. BAM also enables an alternative, sequential interpretation, which in turn allows a Monte Carlo algorithm for unbiased marginal likelihood estimation. We start with the description of the generative model, and move on to detailing inference and model selection.

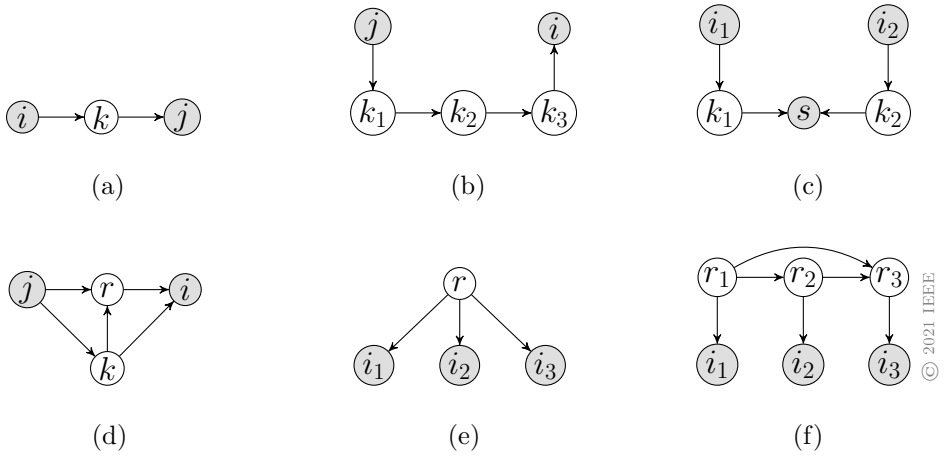


Figure 1.2. Some nonnegative matrix/tensor factorization models and topic models expressed as instances of BAM: (a) KL-NMF - LDA, (b) Pachinko allocation, (c) mixed membership stochastic blockmodel, (d) sum conditioned Poisson factorization, (e) CP decomposition, (f) Tucker decomposition. Figure from [33].

1.3.1. The Model

Suppose we have a random tensor, $\mathbf{S} \in \mathbb{Z}_{\geq 0}^{I_1 \times \cdots \times I_N}$, which is a collection of cells, where the cell indexed by $i_{[N]} := (i_1, \ldots, i_N)$ is said to contain $\mathbf{S}(i_{[N]})$ *tokens*. For $n \in [N] := \{1, \ldots, N\}$, each component i_n of the index $i_{[N]}$ takes values from $[I_n] := \{1, \ldots, I_n\}$, where $I_n \geq 0$ is the cardinality of the n'th component. We suppose that tokens are allocated to cells randomly, and present the probabilistic model that generates $\mathbf{S}$, which we call an *allocation tensor*.

In BAM, we assume the total number of tokens to be random as well, having a Poisson distribution with parameter λ, which itself is also a random variable, with a gamma distribution with shape and scale parameters a and b, respectively,

$$\lambda \sim \mathcal{GA}(a, b). \tag{1.6}$$

Next, we define a probability tensor θ, with the same dimensional structure as $\mathbf{S}$, such that each token is independently allocated to cell indexed by $i_{[N]}$ with probability $\theta(i_{[N]})$. Therefore, $S(i_{[N]})$'s are conditionally independent given θ, each with a Poisson

distribution with rate parameter $\lambda\theta(i_{[N]})$,

$$\mathbf{S}(i_{[N]}) \sim \mathcal{PO}\left(\lambda\theta(i_{[N]})\right), \quad \forall i_{[N]}. \tag{1.7}$$

For large N, explicitly storing the table θ is not feasible due to the fact that its size grows exponentially with N, so alternative representations are of great practical interest. One such representation, which we choose here, is via a Bayesian network [41]. In this representation, the probability model is specified with a reference to a graph $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}})$ with the set of vertices $V_{\mathcal{G}} = [N]$ and directed edges $E_{\mathcal{G}}$. This graph is a directed acyclic graph, meaning that the set of directed edges $E_{\mathcal{G}} \subset V_{\mathcal{G}} \times V_{\mathcal{G}}$ are chosen such that there are no directed cycles in $\mathcal{G}$. Each vertex n of the graph corresponds to a random variable and missing edges represent conditional independence relations. Given a directed acyclic graph $\mathcal{G}$, for each vertex n, we define the set of *parents* $\mathrm{pa}(n)$ as the set of vertices v from which there is a directed edge incident to node n, defined by $\mathrm{pa}(n) := \{v : (v, n) \in E_{\mathcal{G}}\}$. We also define the *family* of the node n as $\mathrm{fa}(n) := \{n\} \cup \mathrm{pa}(n)$. Given $V \subseteq [N]$, we define $i_V := (i_v : v \in V)$. With these definitions, the Bayesian network encodes a factored representation of the original probability table as

$$\theta(i_{[N]}) = \prod_{n=1}^{N} \theta_{n|\mathrm{pa}(n)}(i_n, i_{\mathrm{pa}(n)}). \tag{1.8}$$

Finally, we assign a prior distribution for θ. We start with an order-N tensor α with nonnegative entries $\alpha(i_{[N]}) \geq 0$ for all $i_{[N]}$. In practice, we do not need to explicitly store or construct α; we use it to consistently define the contractions for all $n \in [N]$

$$\alpha_{\mathrm{fa}(n)}(i_n, i_{\mathrm{pa}(n)}) := \sum_{i_{\overline{\mathrm{fa}(n)}}} \alpha(i_{[N]}), \tag{1.9}$$

where by $i_{\overline{\mathrm{fa}(n)}}$ we mean summation over all indices $i'_{[N]}$ such that $i'_{\mathrm{fa}(n)} = i_{\mathrm{fa}(n)}$. We model each factor $\theta_{n|\mathrm{pa}(n)}$ in (1.8) as an independent random conditional probability table: For each $i_{\mathrm{pa}(n)}$, the array $\theta_{n|\mathrm{pa}(n)}(: \,|i_{\mathrm{pa}(n)})$ is independent and has a Dirichlet distribution with prior parameters $\alpha_{\mathrm{fa}(n)}(:, i_{\mathrm{pa}(n)})$, that is,

$$\theta_{n|\mathrm{pa}(n)}(:, i_{\mathrm{pa}(n)}) \sim \mathcal{D}\left(\alpha_{\mathrm{fa}(n)}(:, i_{\mathrm{pa}(n)})\right), \quad \forall i_{\mathrm{pa}(n)}, \tag{1.10}$$

for all $n \in [N]$.

As described above, in most applications the allocation tensor $\mathbf{S}$ is not directly observed. Instead, we observe particular contractions of $\mathbf{S}$, of form

$$\mathbf{X}(i_V) := \sum_{i_{\bar{V}}} \mathbf{S}(i_{[N]}), \quad \forall i_V, \tag{1.11}$$

or, shortly, $\mathbf{X} = \mathbf{S}_V$. Here $V \subset [N]$ is the set of "visible" indices, and $\bar{V} = [N] \setminus V$ are the latent indices. Many latent variable models such as topic models and tensor factorization models have observations with this form. Various structured NTF and topic models can be expressed as instances of BAM by defining an appropriate directed graph $\mathcal{G}$. See Figure 1.2 for a number of examples such as latent Dirichlet allocation (LDA) [42], Pachinko allocation [43], mixed membership stochastic blockmodel [44], sum conditioned Poisson factorization [45], CP decomposition [23, 24], and Tucker decomposition [4]. The technical report [35] includes a more detailed discussion of constructing these models as instances of BAM.

1.3.2. Inference and Model Selection

Given the hyperparameters (α, a, b), the hierarchical generative model for variables $\mathbf{S}$, $\Theta := \{\theta_{n|\mathrm{pa}(n)} : n \in [N]\}$, and λ, the joint distribution $p(\mathbf{S}, \Theta, \lambda)$ is expressed as

$$p(\mathbf{S}, \Theta, \lambda) = p_{a,b}(\lambda) p_\alpha(\Theta) p(\mathbf{S}|\Theta, \lambda), \tag{1.12}$$

where the distributions $p_{a,b}(\lambda)$, $p_\alpha(\Theta)$ and $p(\mathbf{S}|\Theta, \lambda)$ are indicated by (1.6)-(1.10). A particularly interesting distribution is the *marginal allocation probability* of $\boldsymbol{S}$, denoted by $p(\mathbf{S} = \boldsymbol{S})$, which plays a central role in formulating the sequential interpretation of BAM [33].

Given the observed tensor $\mathbf{X} = \boldsymbol{X}$, one important inferential goal is to estimate the *marginal likelihood* of $\boldsymbol{X}$, defined by

$$\mathcal{L}_X := p(\mathbf{X} = \boldsymbol{X}) = \sum_{\boldsymbol{S}} p(\mathbf{S} = \boldsymbol{S}) \mathbb{I}(\boldsymbol{S}_V = \boldsymbol{X}). \tag{1.13}$$

Many likelihood-based inferential goals, such as model scoring, model selection (e.g. rank estimation or selection among structurally different models), or hyperparameter

estimation, rely on $\mathcal{L}_X$.

Another important inferential target is the full posterior of BAM, which can be expressed as

$$p(\mathbf{S}, \Theta, \lambda | \mathbf{X} = \boldsymbol{X}) = \frac{p(\mathbf{S}, \Theta, \lambda, \mathbf{X} = \boldsymbol{X})}{\sum_{\mathbf{S}} p(\mathbf{S} = \boldsymbol{S}) \mathbb{I}(\boldsymbol{S}_V = \boldsymbol{X})}, \tag{1.14}$$

given (1.13) and the joint distribution $p(\mathbf{S}, \Theta, \lambda, \mathbf{X} = \boldsymbol{X}) = p(\mathbf{S}, \Theta, \lambda)\mathbb{I}(\mathbf{S}_V = \boldsymbol{X})$.

1.3.2.1. Variational Inference. Variational Bayes (VB) [46] is a technique where a target posterior distribution $\mathcal{P} := p(\mathbf{S}, \Theta, \lambda | \mathbf{X} = \boldsymbol{X})$ is approximated by a variational distribution $\mathcal{Q}$ most commonly via minimizing Kullback-Leibler divergence $D_{\mathrm{KL}}(\mathcal{Q}||\mathcal{P})$, though other notions of divergence can also be used [47]. In the context of Bayesian model selection, minimization of the $D_{\mathrm{KL}}(\mathcal{Q}||\mathcal{P})$ corresponds to establishing a tight lower bound for the marginal log-likelihood $\log \mathcal{L}_X$, which we refer as *evidence lower bound* (ELBO) and denote by $\mathcal{B}_{\mathcal{P}}[\mathcal{Q}]$. This correspondence is due to the following decomposition of marginal log-likelihood

$$\log \mathcal{L}_X = \mathcal{B}_{\mathcal{P}}[\mathcal{Q}] + D_{\mathrm{KL}}(\mathcal{Q}\|\mathcal{P}) \geq \mathcal{B}_{\mathcal{P}}[\mathcal{Q}], \tag{1.15}$$

due to the nonnegativity of KL divergence. Then, the ELBO is explicitly defined as

$$\mathcal{B}_{\mathcal{P}}[\mathcal{Q}] \equiv \mathbb{E}_{\mathcal{Q}} \left[\log \frac{p(\mathbf{S}, \lambda, \Theta)\ \mathbb{I}(\mathbf{S}_V = \boldsymbol{X})}{\mathcal{Q}(\mathbf{S}, \lambda, \Theta)} \right]. \tag{1.16}$$

Typically in VB, variational distribution $\mathcal{Q}$ is assumed to be a member of a restricted family of distributions. In its most common form, also known as *mean-field* approximation, $\mathcal{Q}$ is assumed to factorize over some partition of the latent variables. Variational inference under this approximation is frequently called *mean-field variational inference (MFVI)* [48, Chapter 28]. For BAM, a natural choice of mean-field approximation is the family of factorized distributions of the form

$$\mathcal{Q}(\mathbf{S}, \lambda, \Theta) = q(\mathbf{S})\ q(\lambda, \Theta). \tag{1.17}$$

A common algorithm for minimizing $D_{\mathrm{KL}}(\mathcal{Q}||\mathcal{P})$, i.e., maximizing ELBO is updating the variational distributions in an alternating fashion, sometimes called *coordinate ascent variational inference (CAVI)* [49]. In the context of BAM, one can iteratively

update $q(\mathbf{S})$ and $q(\lambda, \Theta)$ according to

$$q(\mathbf{S}) \propto \exp\big(\mathbb{E}_{q(\lambda,\Theta)}\left[\log p(\mathbf{S}, \lambda, \Theta) + \log \mathbb{I}(\mathbf{S}_V = \boldsymbol{X})\right]\big), \tag{1.18}$$

$$q(\lambda, \Theta) \propto \exp\big(\mathbb{E}_{q(\mathbf{S})}\left[\log p(\mathbf{S}, \lambda, \Theta) + \log \mathbb{I}(\mathbf{S}_V = \boldsymbol{X})\right]\big). \tag{1.19}$$

For brevity, we defer the specific form of the updates, and the resulting expression for the ELBO to Appendix A.2, and move on to examining the sequential interpretation of BAM.

1.3.2.2. Sequential Monte Carlo. For completeness, we briefly review the alternative, sequential interpretation of BAM that enables a sequential Monte Carlo (SMC) algorithm for an unbiased estimate of the marginal likelihood. However, since we will not be using this estimator within this thesis for reasons provided below, we keep this review at a high level, and direct the reader to [33] and [35] for further details.

With the help of conjugacy, the Θ and λ in (1.12) can be analytically integrated out to obtain an explicit expression for $p(\mathbf{S} = \boldsymbol{S})$. Remembering our definition of the total number of observations as a random variable $\mathbf{S}_+ = \sum_{i_{[N]}} \mathbf{S}(i_{[N]})$, and letting $p_t(\boldsymbol{S}) := p(\mathbf{S} = \boldsymbol{S} | \mathbf{S}_+ = t), t \geq 1$, Proposition 1 in [33] shows that

$$p(\mathbf{S} = \boldsymbol{S}) = p(\mathbf{S}_+ = \boldsymbol{S}_+) p_{\boldsymbol{S}_+}(\boldsymbol{S}), \tag{1.20}$$

with both terms on the right hand side having analytical expressions.

The form of marginal allocation probability provided in (1.20) is crucial for showing the equivalence between the static and sequential interpretations of BAM. Accordingly, [33] define a process $\{\mathbf{S}_t\}_{t>0}$, where the tokens arrive sequentially and populate the allocation tensor, where $\mathbf{S}_t$ denotes is the tensor after t tokens have arrived. In Proposition 2, [33] define the transition probability $f_{t+1}(\boldsymbol{S}_{t+1} | \boldsymbol{S}_t)$ that renders the said process a non-homogeneous Markov chain. Importantly, one-step sampling or evaluation according to $f_{t+1}(\boldsymbol{S}_{t+1} | \boldsymbol{S}_t)$ does not require computations that involve the prohibitively large $\boldsymbol{S}_t$, but rather its much smaller marginals obeying the factorization structure implied by the graph $\mathcal{G}$, due to $f_{t+1}(\boldsymbol{S}_{t+1} | \boldsymbol{S}_t)$ being determined by a collection of dependent Pólya urn processes that obey this structure.

Letting $p^s(\mathbf{S}_{1:t} = \boldsymbol{S}_{1:t}) = \prod_{\tau=0}^{t-1} f_{\tau+1}(\boldsymbol{S}_{\tau+1}|\boldsymbol{S}_\tau)$, [33] show that $p^s(\mathbf{S}_{1:t} = \boldsymbol{S}_{1:t})$ depends only on $\boldsymbol{S}_t$, implying that the arrival process of the tokens is exchangeable. Given the tokens can arrive in $\binom{t}{\boldsymbol{S}_t}$ different orders, we have $\binom{t}{\boldsymbol{S}_t} p^s(\mathbf{S}_{1:t} = \boldsymbol{S}_{1:t}) = p^s(\mathbf{S}_t = \boldsymbol{S}_t)$. It can then be shown that $p^s(\mathbf{S}_{\mathbf{S}_+} = \boldsymbol{S}_{\mathbf{S}_+}) = p_{\mathbf{S}_+}(\boldsymbol{S})$, which implies that the static and sequential interpretations of BAM are equivalent.

Letting $T = \boldsymbol{S}_+$, the importance of this is better understood by observing the fact that with a similar notation to above, $\mathcal{L}_X$ can be factorized as

$$\mathcal{L}_X = p(\mathbf{X}_+ = T)p(\mathbf{X}_T = \boldsymbol{X}), \tag{1.21}$$

where $p(\mathbf{X}_+ = T) = p(\mathbf{S}_+ = T)$ corresponds to the leftmost term within the factorization in (1.20), which has a known analytical form. Then, focusing on the estimation of the probability

$$p(\mathbf{X}_T = \boldsymbol{X}) = \sum_{\boldsymbol{S}_T} p_T(\boldsymbol{S}_T)\mathbb{I}[\boldsymbol{X} = \boldsymbol{S}_{T,V}], \tag{1.22}$$

we can see that being able to cheaply simulate $\mathbf{S}_T$ allows an importance sampler for $p(\mathbf{X}_T = \boldsymbol{X})$. This would involve generating M independent realizations of the generative process until time T and returning the proportion of the number of processes that satisfy $\mathbb{I}[\boldsymbol{X} = \boldsymbol{S}_{T,V}]$, constituting an unbiased estimator of the marginal likelihood. This estimator, however, turns out to perform very poorly due to high variance. Nevertheless, the sequential construction allows a more efficient SMC estimator based on sequential importance sampling - resampling, a.k.a. particle filtering [33].

1.3.2.3. Inference with BAM in Applications. Although the SMC estimator in [33] provides an unbiased marginal likelihood estimate, it has important disadvantages for the purposes of this thesis. As observed in [33], SMC estimator's variance becomes prohibitively large even for moderate T, in addition to its computational complexity scaling with T, instead of e.g. number of non-zero elements. This decreases the usefulness of the SMC estimator for various large-scale datasets considered in the following chapters. Thus, in this thesis we rely on the static interpretation of BAM and the associated variational procedure it implies that we reviewed above. For completeness, in Appendix A.3 we provide a set of experiment results that compare the two algorithms

in a political science dataset, and refer the reader to [33] and [35] for more details on the SMC estimator.

A potential limitation of BAM in applications is its assumption of Poisson likelihood, limiting the model to discrete nonnegative data. Although extending BAM to natively handle such data is an important future direction (see Chapter 7), in practice this is not necessarily an important hindrance for datasets with large number of observations as the data can be interpreted as the sum of discrete values and element-wise independent uniform noise, as noted by [29]. So, given $\mathbf{X} \in \mathbb{R}_{\geq 0}^{I_1 \times \cdots \times I_N}$, we can let $\mathbf{X} = \widetilde{\mathbf{X}} + \mathbf{E}$, where $\widetilde{\mathbf{X}} \in \mathbb{Z}_{\geq 0}^{I_1 \times \cdots \times I_N}$, and $\mathbf{E}(i_{[N]}) \sim \mathcal{U}(0,1), \forall i_{[N]}$. This allows recovering $\widetilde{\mathbf{X}}$ with an element-wise flooring operation. We explicitly describe our method of choice to handle non-discrete observations whenever we encounter such a dataset.

1.3.3. Hyperparameters

We discuss possible choices of hyperparameters of BAM. The (prior) expectation of λ, denoted as $\mathbb{E}[\lambda] = a/b$, is also the expected number of tokens. Given an observed tensor X, $\boldsymbol{S}_+$ is fully observable and we can choose the scale parameter as $b = a/\boldsymbol{S}_+$, following an empirical Bayesian approach.

The key parameter that defines the behavior of the model is α, that describes our *a-priori* belief, how the tokens will be distributed on $\mathbf{S}$. Without *a-priori* preference to a particular group of cells, a natural choice is a flat prior

$$\alpha(i_{[N]}) = \frac{a}{\prod_{n=1}^{N} I_n}, \tag{1.23}$$

also known as a Bayesian Dirichlet (likelihood) Equivalent Uniform prior [50, 51]. The construction of prior parameters in (1.10) from a common α is required if the distribution of $\theta(i_{[N]})$ is desired to be identical for all Markov equivalent models. Defining the parameters of each measure as in (1.9) is necessary and sufficient to ensure such consistency by the equivalent sample size principle [52].

An interesting alternative to the prior selection methodology above is one that *deliberately* disrupts the equivalence of marginal likelihood across Markov equivalent directed acyclic graphs. In causal discovery literature, directed acyclic graphs with causal semantics, i.e. *causal graphs*, are frequently used to infer the *intervention distributions* of the variables in question [53]. When scoring different causal models to select a causal graph that characterizes the data best, equal scoring of Markov equivalent graphs can be undesirable. For example, given two observed random variables x and y, while the graphical models $x \to y$ vs. $x \leftarrow y$ are Markov equivalent, their causal interpretations are starkly different. Although outside of the scope of this thesis, in [34] we propose a Bayesian model selection algorithm for causal discovery, based on the deliberate disruption of the aforementioned likelihood equivalence.

1.4. Reuse of Previously Published Material

In accordance with publishers' policies regarding the reuse of published material, this thesis incorporates verbatim text excerpts and images directly from author's own publications, with or without modifications. Chapter 1 and Appendix A include material from [33–36]; Chapter 2 and Appendix B include material from [36]. Chapter 3 and Appendix C include material from [54], part of which the author contributed during his employment at DeepMind, London, United Kingdom. Chapter 5 and Appendix D include material from [55, 56], and Chapter 6 includes material from [57].

Among the references cited above as author's prior publications, the manuscripts [35], [57] are preprints, and [34], [55,56] are conference proceedings in which the authors did not transfer the copyright to another entity. The material from [33], [36], [54] are reused in agreement with the author reuse policies of these publications. See [58] for the policy regarding [33] and [36], and see [59] for [54]. Original publications are cited within each reused figure or table at the end of the corresponding caption, and relevant copyright notices are provided whenever required. References to public source codes for published articles are provided inside the respective chapters, and source code for the remaining material can be found in [60].

2. LOW-RANK HIERARCHICAL SEASONALITY

In this chapter, we propose a flexible and parameter-efficient approach for modeling complex seasonality in univariate and multivariate time series, and then demonstrate how NTF, through the probabilistic framework we described, can be utilized within this approach for accurate and interpretable analysis of such large-scale time series. Dealing with seasonality, or periodic behavior, is a common and challenging task in time series analysis [61–63]. This becomes even more challenging in contexts where multiple, nested seasonality patterns are present. Given the rhythms that determine human behavior, this is all too common in datasets that record aspects of collective human activity. For example, number of commuters in a subway or total foot traffic in a university campus would involve multiple seasonal patterns, ranging from hourly to yearly cycles. In this chapter, we first introduce our modeling approach for such nested seasonal patterns: *low-rank hierarchical seasonality (LRHS)*, which prescribes casting the problem as tensor factorization. Then, we propose a Bayesian variant of this approach, *Bayesian low-rank hierarchical seasonality (BLRHS)*, where Bayesian allocation model (BAM) can be used for inference and model selection. We show that BLRHS not only helps make sense of large-scale data sets by distilling useful multi-seasonal patterns, but also demonstrates high accuracy in missing data imputation tasks. Our work complements and improves upon previous work that utilizes matrix/tensor factorizations in time series research, and shows the usefulness of BAM in this supervised learning task.

This chapter and the associated Appendix B contains excerpts, figures, and tables from the published article by the author [36], which have been used according to the reuse policies of the publisher with appropriate acknowledgments and copyright notices. See [58] for more details.

2.1. Motivation and Background

Let $y_t \in \mathbb{R}$ represent a uniformly sampled univariate time series indexed by $t \in \mathbb{N}$, such that $\mathbf{y} = (y_t)_{t=1}^{T}$. Borrowing the notation of the classical time series decomposition [64], one can model y_t via the additive decomposition

$$y_t = \ell_t + s_t + \varepsilon_t, \tag{2.1}$$

or the multiplicative decomposition $y_t = \ell_t \cdot s_t \cdot \varepsilon_t$; where ℓ_t is the *trend-cycle* component, s_t is the seasonal component, and ε_t are residual terms. The seasonal component s_t is modeled via a repeating fixed-length pattern, $\mathbf{m} \in \mathbb{R}^P$, where $P \in \mathbb{N}$ is the period of the data known a priori. One then sets $s_t = m_{t \bmod P}$, where mod denotes the modulo operator. Estimation of the classical decomposition is simple. In the additive case, this would involve first detrending the data via ℓ_t, the suitably chosen moving average, and then taking seasonal deviations m_i, $i \in \{1, \cdots, P\}$ as simple averages of deviations for season i. Within this general framework, STL and Holt-Winters methods can be thought of as variations where the trend-cycle and seasonal component estimates take different parametric forms and are updated through time. We refer the reader to [64] for a review of these methods.

Our focus here is on the dimensionality of the seasonal component $\mathbf{m}$. In many common modeling tasks, the time series or signal contains multiple nested periodic patterns. For example, in data sampled at minutely frequency, there could be hourly, daily, and weekly periodic behavior all at once. Representing such seasonal patterns with methods above requires dimensionality of $\mathbf{m}$ to be $60 \times 24 \times 7 = 10080$. To tackle this problem of very long seasonal patterns with nested structure, so-called *multiple-seasonal* heuristics have been proposed [65]. For example, given a short period of P_1 and a longer period $P_1 \times P_2$, one could write $s_t = s_t^{(1)} s_t^{(2)}$ defining $s_t^{(1)} = m^{(1)}_{t \bmod P_1}$ and $s_t^{(2)} = m^{(2)}_{\lfloor t/P_1 \rfloor \bmod P_2}$ where $\mathbf{m}^{(1)} \in \mathbb{R}^{P_1}$ and $\mathbf{m}^{(2)} \in \mathbb{R}^{P_2}$. Concretely, if we are interested in hourly sampled data with both daily and weekly patterns, our notation suggests setting $P_1 = 24$ and $P_2 = 7$. By viewing seasonality as the multiplication of an hour-of-day and day-of-week patterns, this model assumption conveniently reduces the dimensionality of seasonality parameters $\mathbf{m}$ from 24×7 to $24 + 7$.

In this chapter, we extend such multiple seasonal patterns in two directions. First, we introduce N-many nested periodicities $P_1, P_2, \ldots, P_N$. Second, we assume that an array of individual multiseasonal patterns can be combined, e.g.,

$$s_t = \sum_r s_t^{(1,r)} s_t^{(2,r)}, \tag{2.2}$$

where $s^{(1,r)}, s^{(2,r)}$ are defined analogously to above and r indexes a set of distinct seasonal patterns.

We demonstrate that such multilinear combinations of seasonal patterns yield flexible and accurate representations of many cyclical patterns that occur in real-world data. Moreover, we demonstrate that components $s^{(n,r)}$ can be recovered easily by casting estimation as an instance of low-rank tensor factorization.

2.1.1. Related Work

Matrix/tensor based analysis of time series has been attracting considerable interest from various fields. The scope of such research ranges from theoretical and algorithmic developments [6], [66] to numerous applications, the latter including energy management and load monitoring [67, 68], intelligent traffic systems [69, 70], mobility and urban planning [71], logistics [72], and structural monitoring [73]. These work often represent time as one of the modes of a multiway data representation. For example, Dunlavy et al. [74] factorize a three-way tensor where one of the modes is time, and use Holt-Winters method on the temporal factor matrix to extrapolate to new time points. Similarly, de Araujo et al. [75] apply exponential smoothing to the temporal factor for forecasting, within a coupled factorization framework. A Bayesian approach to the same problem is studied by Xiong et al. [76]. Matsubara et al. [77] apply a multi-scale type autoregressive model for extrapolating the temporal dimension.

In a line of work that pertains to epidemiology applications, Matsubara et al. [78] introduce FUNNEL, a model for coevolving epidemics inspired by the well-known SIR model. FUNNEL incorporates diseases, different locations, and time into a data tensor

while accounting for seasonal patterns by including a sinusoidal seasonal component. Rogers et al. [79] propose a multilinear extension of the well known linear dynamical system. Yu et al. [80] propose a matrix factorization approach where the autoregressive time series model likelihood appears as a regularization term. An extension of this idea into the spatio-temporal domain is proposed by Takeuchi et al. [81], and Sun and Chen [82] provide a Bayesian version of the same approach. More recently, Kawabata et al. [83] propose an online inference scheme for multivariate data where the model can detect the appearance of a new seasonality regime.

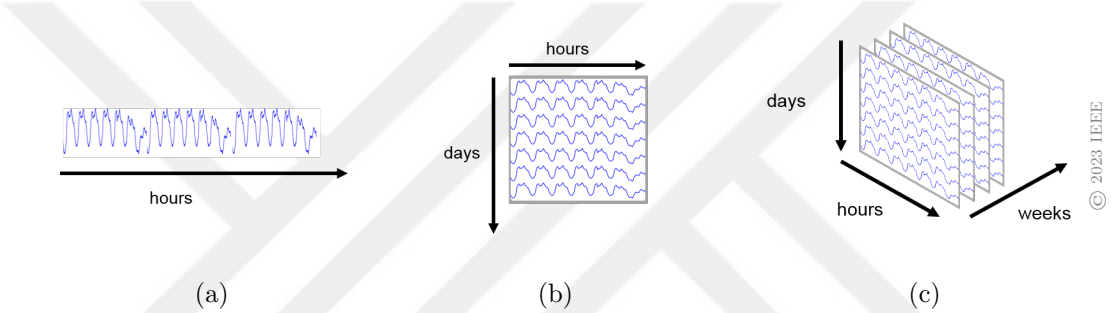


Figure 2.1. Folding time series into a tensor. In (a), the time series is represented as a single vector. In (b), each hour of day is presented in a new mode. In (c), each week is a slice as days of the week are folded into the third mode. Extension to lower or higher seasonality components is straightforward, and explored in our numerical experiments. Figure from [36].

Methods surveyed above use the tensor factorization formalism with only one mode of the tensor of interest representing time, reserving other modes for different entities (e.g., users, items, locations, etc.) An idea that follows immediately is to let one of the modes represent seasonal behavior, i.e., to "fold" the time series into a 2-way array, as in Figure 2.1(b). In a more general tensor factorization framework, this means that each latent dimension has a representation in the entity domains, as well as a certain seasonal pattern. This idea appears early in the beginning of time series analysis with factorizations, e.g. in an atmospheric science application by Xie et al. [84]. Takahashi et al. [85] apply this idea to various time series, with a specific focus on folding periods—or *cyclic(al) folding* in their terms which we reuse

here. Cyclical folding also appears in Matsubara et al. [86, 87], which are models of competing entities inspired by population modeling. Chen and Sun [88] augment the work of [80] by adding another temporal dimension of seasonality for forecasting, and Chen and Lei [89] extend this approach to missing data imputation.

Though less common, some studies extend the cyclical folding idea to higher temporal dimensions, modeling two seasonalities simultaneously. Notably, Tan et al. [90] uses a sliding temporal tensor construction to impute future values for multivariate time series, and Chen and He [91] compare the performance of having cyclical folding to obtain zero, one, or two seasonality dimensions while utilizing a CP (CANDECOMP/PARAFAC) decomposition for missing data imputation. Given [91] provides the only quasi-systematic exploration of multiple cyclical folding, we compare our method to theirs, as well as some other baselines in Section 2.4.3 to provide an informative test of our approach.

Also less common is the use of nonnegative matrix/tensor factorization (NMF/NTF) to model time series data, and especially with consideration of hierarchical seasonality. Espin-Noboa et al. [71] apply NTF to decompose the NYC taxi dataset into coherent patterns to test hypotheses about urban mobility. Wang et al. [92] apply nonnegative matrix factorization techniques for the analysis of traffic patterns, and Figueiredo et al. [67] provide an analysis of power consumption through NTF by including two temporal dimensions, however neither studies exploit hierarchical patterns in seasonality. The relative neglect of nonnegative methods is especially important from a knowledge discovery point of view. As discussed in Chapter 1, this is because nonnegative factorizations are known to produce interpretable, "by-parts" representations [8], [93], which could help extract the simpler multi-seasonal components that produce the emergent complex behavior of such time series. To our knowledge, the Bayesian variant of our approach is the first application of probabilistic NTF to the analysis of complex seasonality.

Our method is based on the hypothesis that low-rank multilinear representations can accurately and efficiently capture nested, complex seasonalities. It directly targets to exploit the low-rank decomposability of such intricate patterns, improving on the sporadic use of cyclical folding as a data representation heuristic for specific multivariate time series problems in the literature. As models and data get larger in modern machine learning, the need for such parsimonious representations increase. As such, our results not only expand applications of probabilistic NTF methods and contribute to tensor-based time series analysis, but also offers valuable insight for classical and deep learning based approaches, given the transferability of this inductive bias.

We now present our proposed approach for modeling hierarchical seasonality, which we call *low-rank hierarchical seasonality* (LRHS). Afterwards, we present its Bayesian variant.

2.2. Low-Rank Hierarchical Seasonality

Recall the definition of a *multiple seasonal* s_t, and how it was defined as the *multiplication* of *factors* $s_t = s_t^{(1)} s_t^{(2)}$ to account for two "nested" periodic patterns, such as days-of-week and hours-of-day. We now formally define our modeling approach by generalizing this idea to N seasonalities and multiple such nested patterns, and we rely on tensor formalism when doing so. This implies a particular representation of the data, where we "fold" an observed seasonality vector $\mathbf{s}$ based on its seasonalities to create a tensor. We define this representation below.

Definition 2.1 (Multiple cyclical folding; MCF)**.** *Let* $\mathbf{s} \in \mathbb{R}^T$ *denote a vector that contains observed values of the seasonal component of a time series, and let* $P_1, \ldots, P_N \in \mathbb{Z}^+$ *be the periods for the* N *nested seasonalities observed in the data. Assume the data includes* $K \in \mathbb{Z}^+$ *full cycles of the longest seasonality, such that* $T = K \cdot \prod_{i=1}^{N} P_i$*. Multiple cyclical folding creates the tensor* $\mathbf{M} \in \mathbb{R}^{P_1 \times \cdots \times P_N \times K}$ *such that*

$$\begin{aligned} \mathbf{M} = \ & \mathit{MCF}(\mathbf{s}), \ \textit{such that } \mathbf{M}(t \bmod P_1, \\ & \lfloor t/\bar{P}_1 \rfloor \bmod P_2, \ldots, \lfloor t/\bar{P}_{N-1} \rfloor \bmod P_N, \lfloor t/\bar{P}_N \rfloor) := s_t, \end{aligned} \tag{2.3}$$

where $\bar{P}_n = \prod_{i=1}^{n} P_i$.

Note that Definition 2.1 uses 0-indexing when referring to tensor elements for convenience, as opposed to the use of 1-indexing convention in the rest of the thesis for a more natural exposition. A visualization of this data representation is presented in Figure 2.1. For example, if we were to model 10 weeks of hourly data with daily and weekly seasonalities, this would imply that $T = 24 \times 7 \times 10 = 1680$ and $\mathbf{s} \in \mathbb{R}^{1680}$, with $P_1 = 24, P_2 = 7$, and $K = 10$. After MCF we would have $\mathbf{M} \in \mathbb{R}^{24\times7\times10} = \text{MCF}(\mathbf{s})$. As defined above, the last dimension of $\mathbf{M}$ indexes the *full cycle*s of the data, such that $k = 1$ for week 1, $k = 2$ for week 2 and so forth. For brevity we call this dimension *cycle index*.

As we will see shortly, our modeling approach naturally extends to multivariate time series. With multivariate data, time series index becomes another dimension of the representation (or dimension*s*, should there exist multiple non-temporal indices). That is, if the example dataset above had measurements from $I = 100$ stations, MCF would produce $\mathbf{M} \in \mathbb{R}^{100\times24\times7\times10}$. Having described our data representation, we now introduce our modeling approach.

Definition 2.2 (Low-rank hierarchical seasonality; LRHS)**.** *Let* $\mathbf{M} \in \mathbb{R}^{P_1\times P_2\times\cdots\times P_N\times K}$ *be the multiple cyclical folding of an observed seasonality* $\mathbf{s} \in \mathbb{R}^T$*, such that* $\mathbf{M} = MCF(\mathbf{s})$*. Low-rank hierarchical seasonality (LRHS) proposes the approximation*

$$\mathbf{M} \approx \widehat{\mathbf{M}} = \mathbf{G} \times_1 \mathbf{E}^{(1)} \dots \mathbf{E}^{(N)} \times_{N+1} \mathbf{K}, \tag{2.4}$$

where $\mathbf{G} \in \mathbb{R}^{R_1\times R_2\times\cdots\times R_N\times R_K}$ *is the latent core tensor,* $\mathbf{E}^{(n)} \in \mathbb{R}^{P_n\times R_n}, \forall n \in [N]$ *and* $\mathbf{K} \in \mathbb{K}^{K\times R_K}$ *are the factor matrices for the seasonal components and the cycle index respectively.* $\times_n$ *is an* n*-mode product [1], thus rendering* (2.4) *an expression of Tucker decomposition. Factors can be found by minimizing* $D(\mathbf{M}\|\widehat{\mathbf{M}})$ *with a suitably chosen divergence function* D*. A multivariate extension is straightforward with an additional mode of cardinality* I *and the corresponding factor matrix* $\mathbf{I} \in \mathbb{R}^{I\times R_I}$.

We follow our definition with some remarks. First, note that the presence of the cycle index, and the corresponding factor $\mathbf{K}$ enables LRHS to detect and model *changing* multi-seasonal patterns through K full cycles. Among various examples of this phenomenon we will see in Section 2.4, Figure 2.4 demonstrates our approach capturing the fact that although weekday daytime taxi trips existed in the weeks before Covid, their relative importance tangibly increased with Covid as other seasonal patterns withered.

Another point to note is that for full generality, we introduced LRHS with a Tucker decomposition in Definition 2.2, where all factor matrices have their own latent factor with different cardinalities $R_1, \ldots, R_N, R_K$, and core tensor $\mathbf{G}$ determines how these latent factors interact. However, if we set $R = R_1 = \cdots = R_N = R_K$ and let $\mathbf{G}$ be superdiagonal, this formulation would correspond to a CP decomposition. Also importantly, other constraints on factors can also be easily integrated to this framework: e.g. a *nonnegative* variant would assume nonnegative observations and would constrain all factors to be nonnegative, which we will do shortly.

The *main hypothesis* of this chapter is that LRHS is an accurate and efficient approach for modeling hierarchical seasonality. LRHS is designed for contexts where the periods of these seasonalities are known a priori: this is most importantly the case for datasets that characterize various aspects of collective human behavior. For example, the number of customers arriving at a restaurant can be driven by two distinct segments of customers, those who prefer lunches on weekdays, and others who come to Friday and Saturday dinners. Our results in Section 2.4 indeed confirm the abundance of such compositions, in complex and meaningful interactions with non-temporal (i.e. spatial) dimensions. LRHS allows the rich arsenal of tensor factorization methodology to be used in the accurate and efficient recovery of such patterns.

2.2.1. Utilizing Learned LRHS Components

LRHS allows the specification of a factorization model and an optimization scheme. Once we recover the component factors based on our modeling assumptions, these learned factors can be utilized for various inferential tasks. First, the learned factors can directly be investigated to discover the underlying dynamics that produce the temporal patterns in question (as in [71]). Second, in the case of *missing data*, given a projector $\mathit{\Pi} : \mathbb{R}^{P_1 \times \cdots \times P_N \times K} \to \mathbb{R}^{P_1 \times \cdots \times P_N \times K}$ that sets cells with missing observations to 0, learning could take place as

$$\underset{\mathbf{G},\mathbf{E}^{(1)},\ldots\mathbf{E}^{(N)},\mathbf{K}}{\arg\min} D\left(\mathit{\Pi}(\mathbf{M}) \| \mathit{\Pi}\left(\mathbf{G} \times_1 \cdots \times_{N+1} \mathbf{K}\right)\right), \tag{2.5}$$

after which the reconstruction $\widehat{\mathbf{M}} = \mathbf{G} \times_1 \mathbf{E}^{(1)} \ldots \mathbf{E}^{(N)} \times_{N+1} \mathbf{K}$ can be used to fill in the missing data (as in [89]).

Lastly, for predicting future observations, imagine without loss of generality that we have a *forecast horizon* of one full cycle, i.e. $1 \cdot \prod_{i=1}^{N} P_i$. Then, an appropriately chosen cycle index row vector $\mathbf{k}'_{K+1} \in \mathbb{R}^{1 \times R_K}$ can be used to compute the prediction for the new cycle:

$$\widehat{\mathbf{M}}_{K+1} = \mathbf{G} \times_1 \mathbf{E}^{(1)} \ldots \mathbf{E}^{(N)} \times_{N+1} \mathbf{k}'_{K+1}, \tag{2.6}$$

where $\widehat{\mathbf{M}}_{K+1}$ hold the predictions for the forecast horizon. $\mathbf{k}'_{K+1}$ can be chosen in various ways, such as setting it equal to the last cycle index of the model $\mathbf{k}'_{K+1} := \mathbf{k}'_K$ (as in [83]), or computing it through the exponential smoothing of past cycle indices: $\mathbf{k}'_{K+1} := \widehat{\mathbf{k}}'_K$, where $\widehat{\mathbf{k}}'_\kappa = \alpha \mathbf{k}'_\kappa + (1-\alpha)\widehat{\mathbf{k}}'_{\kappa-1}$ and $\widehat{\mathbf{k}}'_0 = \mathbf{k}'_1$ by convention (as in [75]). If detrending was conducted beforehand, both imputation and prediction would require re-adding the trend component (or its extrapolation) before finalizing inference.

2.3. Bayesian Low-Rank Hierarchical Seasonality

We now introduce the Bayesian variant of LRHS, which allows us to use BAM for posterior inference and model scoring. Here, we exploit the fact that the time series data of interest usually are nonnegative (e.g. counts of trips, number of purchases,

occupancy of routes, amount of energy consumption), and exploit the additive, sparse, by-parts representations NTF methods are known to produce [8] and combine it with convenient inference and model selection BAM enables. Note that although we utilize BAM as our probabilistic generative modeling approach here, probabilistic NTF is a vibrant research area and another modeling approach can also be utilized to create another variant of BLRHS, with different choices for model structure and conditional distributions [94, 95].

Definition 2.3 (Bayesian low-rank hierarchical seasonality; BLRHS)**.** *Let the multiple cyclical folding of an observed seasonality be* $\mathbf{M} \in \mathbb{Z}_{\geq 0}^{P_1 \times \cdots \times P_N \times K}$*. BLRHS proposes to model this seasonality through a generative model prescribed by BAM, where* $\mathbf{M}$ *is the observed tensor,* $\mathbf{S}$ *is the latent allocation tensor,* λ *is the latent variable that characterizes the total number of observations, and* Θ *include the conditional probability tables that encode the Bayesian network assumed to underlie the factorization.* α, a, b *are the hyperparameters of the model. A multivariate extension of this construction is straightforward where the original tensor is augmented with an additional dimension of cardinality* I *equal to the number of time series.*

See Figure 2.2 for graphical models implied by BLRHS, with CP and Tucker factorizations, structurally identical to the graphical models presented in Figures 1.2(e) and 1.2(f) in our introduction of BAM. We conduct inference through mean-field variational inference as described in Section 1.3.2.

2.3.1. Interpretable Seasonality with BLRHS

To demonstrate how BLRHS helps extract interpretable representations from hierarchically seasonal data, let us start with a toy univariate dataset, pertaining to the ridership counts for a subway station. We model daily and weekly seasonalities ($P_1 = 24$, $P_2 = 7$), and observe this station for K full cycles (weeks). Let us assume that we are analyzing this dataset with BLRHS using a CP decomposition. Notice that the parameters of this factorization according to BLRHS would correspond to

conditional probability tables such that:

$$p(\text{hour} = h|\text{r} = r) := \theta_{\text{hour}|\text{r}}(\text{hour} = h|\text{r} = r). \tag{2.7}$$

Although by definition we do not have direct access to these latent variables, we can use our estimated posterior distribution we obtained through MFVI for this analysis. For example, we can let $\hat{\theta}_{\text{hour}|\text{r}} = \mathbb{E}_Q[\theta_{\text{hour}|\text{r}}]$ and use

$$\widehat{p}(\text{hour} = h|\text{r} = r) := \hat{\theta}_{\text{hour}|\text{r}}(\text{hour} = h|\text{r} = r). \tag{2.8}$$

to examine and utilize these latent factors.

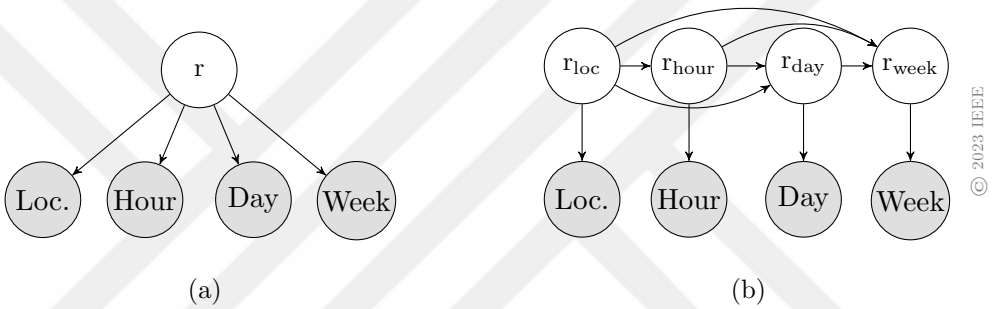


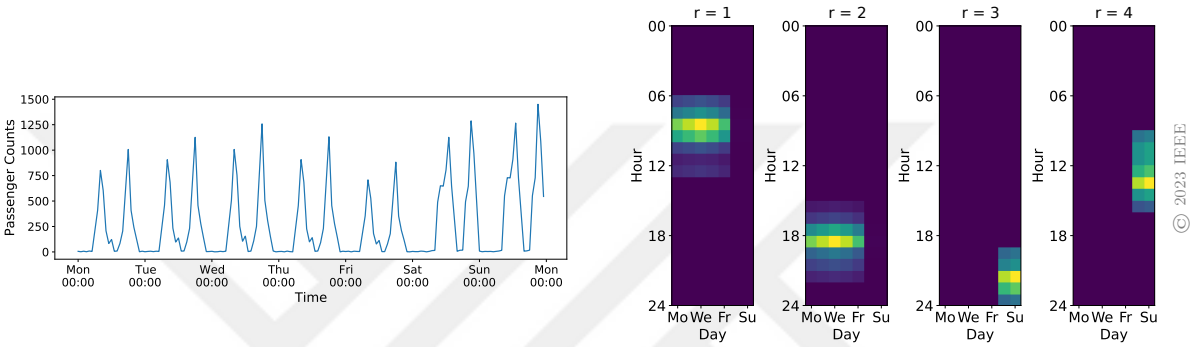
Figure 2.2. BLRHS utilizing (a) CP and (b) Tucker decompositions, modeling multivariate time series with temporal observations at I different locations, and with $P_1 = 24, P_2 = 7$, for K weeks of data. Higher or lower seasonality terms can be added if desired. Hyperparameters are omitted for brevity. Figure from [36].

This probabilistic construction enables various interesting queries to be made through a trained BLRHS model. For example, we could examine the joint daily-weekly seasonality patterns that are captured by our latent factor:

$$\widehat{p}(\text{hour}, \text{day}|\text{r} = r) := \hat{\theta}_{\text{hour}|\text{r}}(\text{hour}|\text{r} = r) \circ \hat{\theta}_{\text{day}|\text{r}}(\text{day}|\text{r} = r). \tag{2.9}$$

We present idealized examples of such queries in Figure 2.3. Conducting inference with BLRHS on the synthetic time series seen in Figure 2.3(a) with $R = 4$ results in the factors seen in Figure 2.3(b). By examining $\widehat{p}(\text{hour}, \text{day}|\text{r} = 1)$ we can see that $\text{r} = 1$ has captured morning commute in this time series. We could also infer that $\text{r} = 2, 3, 4$ correspond to evening commute, weekend night entertainment, and day trips, respectively. We could also look at $\widehat{p}(\text{r} = r)$ to see the strength of these patterns in the overall dataset. We could track their strengths *throughout* weeks, $\widehat{p}(\text{r}|\text{week})$, to

explore the dynamics of these patterns through time, as in weekend night taxi rides disappearing with Covid in Figure 2.4. In Section 2.4, we investigate these complex patterns and examine how they interact with other variables (e.g. location, yearly seasonality) on large real life datasets. The public repository for this chapter includes trained models of BLRHS for the reader to query and explore these spatiotemporal patterns as desired [96].



(a) (b)

Figure 2.3. Synthetic example demonstrating that once BLRHS is trained on a time series (shown in (a)), its estimated factors can be utilized to explore the latent patterns captured by it (shown in (b), lighter implies busier). Heatmaps for $\mathrm{r} = \{1, 2, 3, 4\}$ correspond to four disparate weekly patterns that are captured by BLRHS, in the form of $\widehat{p}(\text{hour}, \text{day}|\mathrm{r} = r)$. Figure from [36].

2.4. Experiments

We now present various experiments that test whether BLRHS accurately captures seasonal patterns in different datasets and problem contexts. For brevity, we direct the reader to [36] for results on univariate time series forecasting with LRHS, where we compare LRHS with CP and Tucker decompositions to canonical baselines for representing seasonality in time series analysis and signal processing, namely discrete Cosine transform, discrete Fourier regression, Fourier basis regression, and Holt-Winters method. Our univariate results show the efficacy of LRHS in parsimoniously capturing and forecasting complex seasonal trends, and demonstrates the usefulness of our approach even without the utilization of BAM [36]. Source code for reproducing

our results and experimenting with the proposed modeling approach is provided in a public repository [96]. All experiments presented in this section have been implemented with the JAX framework in Python programming language [97].

2.4.1. Analyzing New York City Yellow Taxi Data

We now use BLRHS to analyze New York City Yellow Taxi dataset [98], which is a record of the yellow taxi rides within New York City, including start and end locations (265 each), as well as time of the trip. We use the first six months (or 25 weeks) of data from 2020 and create a 5-mode count tensor of dimensions $265 \times 265 \times 24 \times 7 \times 25$ where the dimensions correspond to start and end locations, hour of the day, day of the week, and week of the year, respectively.

We model this dataset using BLRHS with a Tucker decomposition, where we conduct model selection among different ranks and hyperparameters using the evidence lower bound (ELBO). Our procedure selects the model with latent ranks $R = (4, 4, 5, 2, 2)$, with each rank corresponding to the dimensions in the order enumerated above. In this section and the next, we do not explicitly detrend the data and allow the cycle index dimension to account for the long term changes in the dataset. Further details on our model selection procedure can be found in Appendix B.1.

2.4.1.1. Extracted Seasonal Patterns. Our results reveal a good reason for our model selection to have determined $R_{\text{week}} = 2$: this latent factor is used to characterize pre- and post-Covid patterns as shown in Figure 2.4(a). More specifically we see that

$$\widehat{p}(\mathrm{r}_{\text{week}} = 1|\text{week}) \gg \widehat{p}(\mathrm{r}_{\text{week}} = 2|\text{week}), \forall \text{week} \in \{1, \ldots, 10\}, \tag{2.10}$$

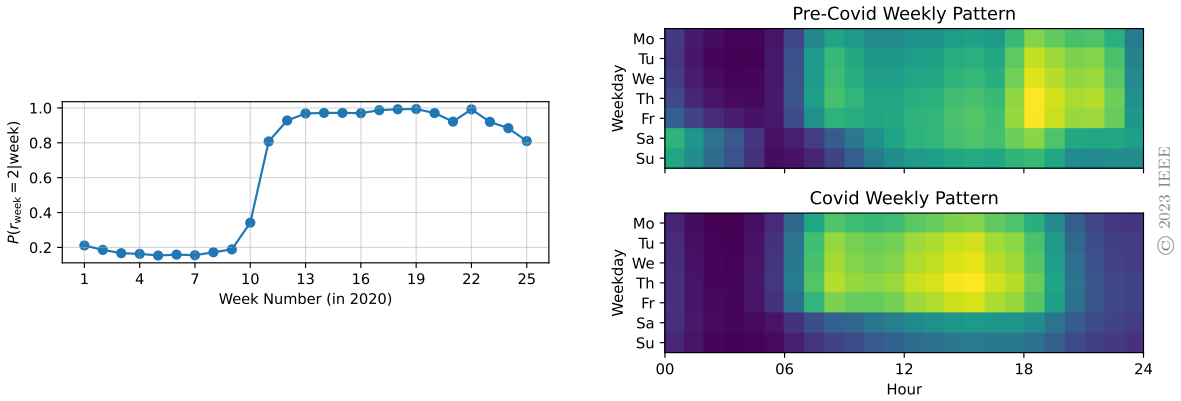
while this trend dramatically reverses after week 10. Note that the model also picks up on the slight, relative return to normal in Summer 2020, as $\widehat{p}(\mathrm{r}_{\text{week}} = 2|\text{week})$ starts to fall back down towards the end of our time window, week 25. Having established that $\mathrm{r}_{\text{week}} = 1$ and 2 roughly characterizes pre- and post-Covid patterns, we further investigate $\widehat{p}(\text{hour}, \text{day}|\mathrm{r}_{\text{week}})$. This reveals the pre- and post-pandemic travel patterns as seen in Figure 2.4(b), where with the pandemic measures, travel patterns

are strongly restricted to daytime activities, most dramatically observed through the disappearance of late night, entertainment-related travel on weekends. Our results are in perfect alignment with those obtained by [83], who also infer Spring 2020 to be a landmark for seasonality regime change in NYC-YT data. That we can corroborate their results with a likelihood-based model selection and posterior inference further confirms the usefulness of our approach. Note that BLRHS can ideally also utilize an online procedure like [83], which we leave as an exciting future direction.

We now further challenge BLRHS to break down the overall observed temporal dynamics to simpler multi-seasonal interaction patterns. For this, we first investigate the latent factor space, and find out the most dominant patterns by examining $\widehat{p}(\mathrm{r}_{\mathrm{hour}}, \mathrm{r}_{\mathrm{day}})$. We then extract the daily-weekly patterns these latent dynamics correspond to by observing $\widehat{p}(\mathrm{hour}, \mathrm{day}|\mathrm{r}_{\mathrm{hour}}, \mathrm{r}_{\mathrm{day}})$. In Figure 2.5, we plot the four most dominant patterns extracted by BLRHS against total taxi ridership pre-Covid (week < 10). BLRHS detects that overall ridership patterns are most strongly contributed by weekday afternoon rides (Pattern 1), weekday night rides (Pattern 2), morning commute rides (Pattern 3), and weekend day trips (Pattern 4) in order. Our results show the various ways BLRHS can be used to make sense of large-scale temporal observations.

2.4.2. Analyzing BART Ridership Data

We now move on to an experiment that shows BLRHS can extract useful information from even larger datasets. For this experiment we use the full BART ridership dataset, which records the hourly number of passengers in the San Francisco Bay Area rail transportation system since 2011, in 2500 different pairwise routes among 50 stations. We model the data between 2011-2022 corresponding to 12 years, and we ignore the data from 2023 as it is incomplete at the time of writing of this text. Given the length of data, we add another seasonality component (yearly), thanks to the flexibility of BLRHS. The resulting tensor is of size $50 \times 50 \times 24 \times 7 \times 53 \times 12$, and contains a total ~ 1.18 billion rides. Even more challenging from a computational perspective, it is not sparse, with only $\sim 60\%$ of its cells empty (vs. NYC YT data's $\sim 98.6\%$).



(a) (b)

Figure 2.4. Changing dynamics of New York City Yellow Taxi rides through the start of Covid-19. Figure (a): Analysis of the latent factors clearly show the transition around 11th week of 2020, just as pandemic measures came into place. Figure (b): Pre-Covid and Covid travel patterns as implied by our model's latent factors (lighter color implies busier). Figure from [36].

2.4.2.1. Extracted Seasonal Patterns. We again start with likelihood-based model selection among models with different ranks, and select a model with $R = (4, 4, 4, 2, 4, 2)$. Examining the results of inference with this model exposes a number of interesting phenomena. We repeat our analysis about dominant daily-weekly patterns and present them in Figure 2.6, visualizing $\widehat{p}(\text{hour}, \text{day}|\text{r}_{\text{hour}}, \text{r}_{\text{day}})$ along with each pattern's strengths $\widehat{p}(\text{r}_{\text{hour}}, \text{r}_{\text{day}})$. The patterns are similar to those in NYC-YT, however take place slightly earlier in the day, given that these are records of mass transportation rides as opposed to private commercial trips. See Appendix B.2 for more details on model selection.

More interestingly, now that we have extracted them, BLRHS allows us to examine these latent patterns' evolution through time. In Figure 2.7, we do exactly this, and see that with Covid-19 the relative prevalence of these different travel tendencies drastically change (while total ridership falls overall). Weekend night travel almost goes extinct by Spring and Summer 2020, not only in absolute terms but also in rela-

tion to other types of travel. In relative terms, the reduction of weekend night travel is not counteracted by the increase of weekday commute, but by increased weekend daytime travel. This somewhat counterintuitive pattern is possibly due to essential workers and travels by residents to parks within the city, while work-from-home measures keep weekday commute at bay. Notice that this is an examination of the *latent* multi-seasonal interactions and is thus not easily accessible through marginal statistics of the data.

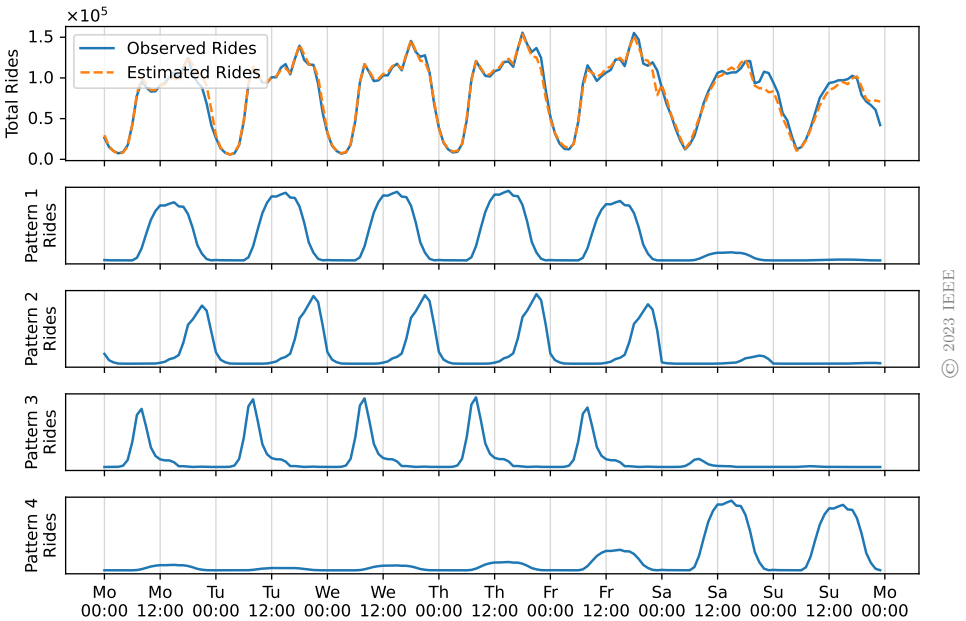


Figure 2.5. BLRHS breaks down the seasonalities that underlie observed patterns. Top figure includes total hourly pre-Covid rides in the NYC YT system and BLRHS's approximation. The four bottom figures plot the most prevalent weekly patterns that contribute to the overall number of rides. y-axes are not in scale for improved visualization. Figure from [36].

Being able to map out these high level interactions further demonstrates the usefulness of BLRHS for making sense of very large temporal data, making full use of the inductive bias that lies at the heart of our approach. One last example of this is presented in Figure 2.8, which relates these latent patterns to multiple spatial components. It demonstrates that a number of urban and commercial hubs (e.g. 6: Civic

Center, 23: Powell Street) witness a large influx from more residential and suburban areas (e.g. 29: Balboa Park, 33: Walnut Creek), and the trend reverses at evening commute time.

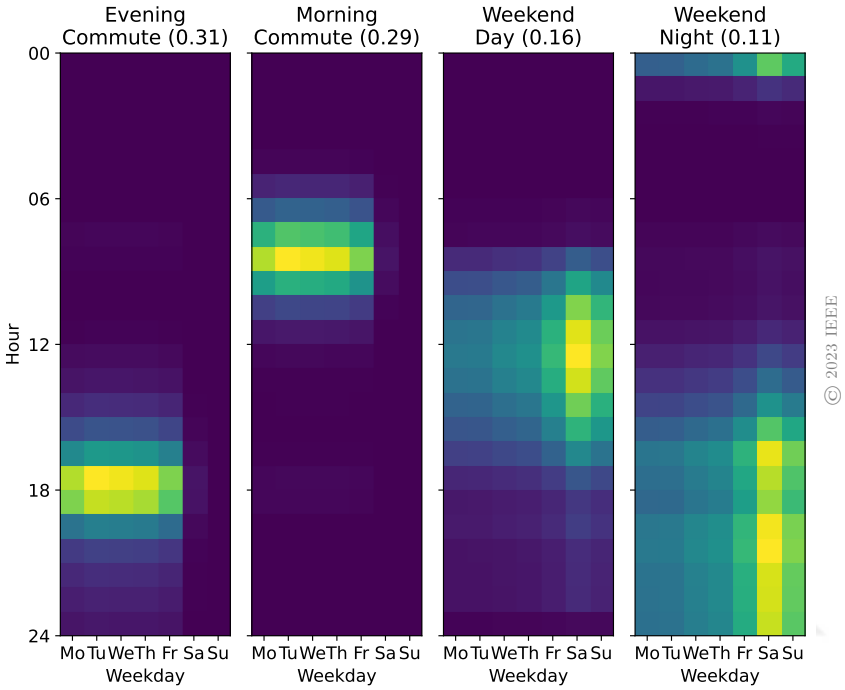


Figure 2.6. BLRHS extracts the daily-weekly latent patterns that contribute most to overall BART ridership. The four strongest are evening commute, morning commute, weekend day travel, and weekend night/entertainment travel. These constitute 0.31, 0.29, 0.16, and 0.11 of all travel respectively in expectation. For all remaining patterns this ratio is < 0.1, lighter implies busier. Figure from [36].

2.4.3. Missing Data Imputation in Multivariate Time Series

We have seen that BLRHS allows scalable and interpretable inference as well as convenient model selection. In this section, we investigate whether it eschews accuracy for interpretability. Our results show that it does not: BLRHS either surpasses or performs on par with other models that target multiple seasonality. As described in Section 2.1.1, one of the most relevant preceding work is by [91], who experiment with one seasonality vs. two nested seasonalities in a given spatiotemporal model,

using Bayesian CP decomposition. Using their setting, we compare BLRHS with their results, as well as a more recent method by [99] who utilize a similar approach yet add a sparse tensor term for modeling outliers.

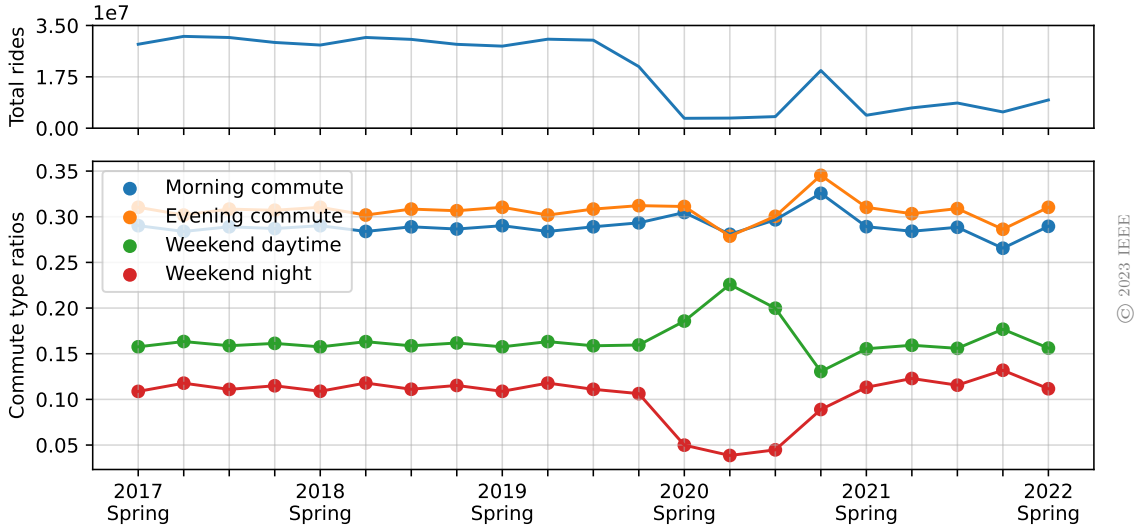


Figure 2.7. Total BART ridership dramatically decreases overall due to the Covid-19 pandemic (top). The relative prevalence of major leading patterns extracted in Figure 2.6 can be tracked through the years (bottom). In relative terms, our results show that during Covid the disappearing weekend night travel is replaced by weekend daytime trips, and not weekday commute. Figure from [36].

2.4.3.1. Dataset. The dataset used is the Guangzhou Traffic Data [100]: a multivariate time series dataset that includes two months of observations of traffic speeds at 214 different road segments in Guangzhou, China, measured with a frequency of 10 minutes, with data dimensions $214 \times 144 \times 61$. We follow the authors' methodology by testing the imputation performance of our method when 10%, 20%, 30%, 40%, and 50% of the data are missing. The entries can be missing either randomly or in a time-correlated fashion (i.e. missing for a day at a time). The dataset has ~ 1.87 million entries, and the 1.29% of the entries are missing in the original dataset.

2.4.3.2. Experimental Settings, Model Selection, and Metrics. We compare our results with Bayesian Gaussian CP (BGCP) from [91] and with Bayesian Robust CP (BRCP) from [99]. Moreover, since we use the exact experimental setting of [91] including randomization [101], we compare our results with baseline model performances included therein as well, using numerical results reported by the authors. We do this because these baselines can surpass BGCP and BRCP, albeit rarely. These alternative baselines include high accuracy low-rank tensor completion (HaLRTC) [102], SVD-combined tensor decomposition (STD) [103], and two simpler baselines daily average (DA; filling in missing data with daily averages over all observations), and a kNN algorithm (using the average of k nearest roads for the missing values). As performance metrics, we use MAPE and RMSE to be able to compare our approach with previous work. MAPE and RMSE are defined such that

$$\text{MAPE}(\mathbf{y}, \widehat{\mathbf{y}}) = \frac{1}{T}\sum_{t=1}^{T}\frac{|y_t - \widehat{y}_t|}{y_t}, \tag{2.11}$$

$$\text{RMSE}(\mathbf{y}, \widehat{\mathbf{y}}) = \left(\frac{1}{T}\sum_{t=1}^{T}(y_t - \widehat{y}_t)^2\right)^{\frac{1}{2}}. \tag{2.12}$$

We use a BLRHS model with CP decomposition. Both [91] and [99] provide their models' results across different hyperparameter settings and/or different data representations. To provide a more rigorous challenge to our method, we use 1% of the data (selected from the uncensored cells) as validation set for model selection, and we compare our selected model with the *best-in-hindsight* versions of the results by [91] and [99].

The change in our model selection procedure is due to two reasons. The first is that Poisson likelihood-based model selection is not an ideal fit for the present task as the ultimate performance comparison involves RMSE and MAPE metrics. Instead, out-of-sample assessment with the same metrics constitutes a more helpful model selection strategy from a generalization perspective. Secondly, the dataset includes traffic speeds with real nonnegative values. In this case, we avoid a discretization scheme like described in Section 1.3.2.3 as it would prevent comparing the performance of BLRHS with previous related work directly. Therefore, in this case we proceed without any

preprocessing, and apply the MFVI updates as described in Appendix A.2. However, this renders marginal likelihood and ELBO undefined, preventing a likelihood-based model comparison.

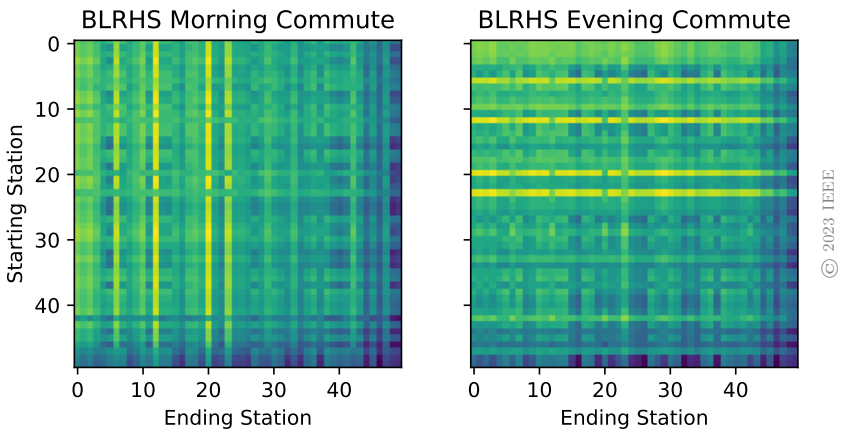


Figure 2.8. Examining how latent temporal patterns extracted by BLRHS can be related to spatial dimensions. Station names for all indices can be found in Appendix B.2, lighter implies busier. Figure from [36].

During model selection we search among $R = \{5, 25, 50, 150, 300, 450\}$, corresponding to a range of 1000-fold to 10-fold parameter decrease in the representation of the data. Since this is not a dataset of counts or frequencies but traffic speeds, the observed entries are almost always larger than 0. Thus, we apply a simple detrending scheme where we subtract the minimum of each time series from all observations before conducting inference with BLRHS, and add this to BLRHS's reconstructed output after inference. We present our results using a 3-order tensor data representation, discussion of BLRHS experiments up to 4 temporal dimensions (5-order tensor), as well as further details on model selection can be found in Appendix B.

2.4.3.3. Results. We present our results in Table 2.1. For randomly missing observations the results are almost unanimous: save for a few exceptions our approach outperforms the best-in-hindsight alternative methods. This is very encouraging, and implies that BLRHS not only provides an interpretable analysis of seasonal patterns, but also an accurate one.

Table 2.1. The comparison of BLRHS with other methods for missing data imputation. Best results in each column are in bold. BLRHS is ours; BRCP is from [99], the rest is from [91]. Table from [36] © 2023 IEEE.

Random Missing										
	10%		**20%**		**30%**		**40%**		**50%**	
Model	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE
DA	0.1213	5.1778	0.1218	5.1905	0.1217	5.1977	0.1217	5.1993	0.1221	5.2113
kNN(10)	0.1303	5.1101	0.1314	5.1486	0.1322	5.1966	0.1333	5.2565	0.1356	5.3573
HaLRTC	0.0776	**3.1716**	0.0817	3.3231	0.0850	3.4748	0.0887	3.6143	0.0931	3.7730
BGCP	0.0795	3.4521	0.0798	3.4531	0.0799	3.4655	0.0801	3.4756	0.807	**3.5042**
BRCP	0.0832	3.5918	0.0836	3.6001	0.0838	3.6160	0.0837	3.6165	0.0844	3.6429
BLRHS (Ours)	**0.0731**	3.2320	**0.0742**	**3.2716**	**0.0765**	**3.3525**	**0.0780**	**3.4625**	**0.0801**	3.5373
Correlated Missing										
	10%		**20%**		**30%**		**40%**		**50%**	
Model	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE	MAPE	RMSE
DA	0.1208	5.1128	0.1207	5.1983	0.1346	5.2591	0.1388	5.4405	0.1445	5.6516
kNN(13)	0.1342	5.1714	0.1340	5.1486	0.1322	5.1966	0.1333	5.2565	0.1356	5.3573
HaLRTC	0.1015	4.1322	0.1022	4.1716	0.1035	4.2372	0.1057	4.3232	0.1090	4.4472
BGCP	0.0980	4.1413	0.0984	4.1477	**0.0980**	**4.1857**	**0.1001**	**4.2881**	0.1030	4.4199
BRCP	0.0982	4.1352	**0.0980**	**4.1400**	0.0989	4.2197	0.1003	4.2909	**0.1022**	4.4497
BLRHS (Ours)	**0.0976**	**4.1314**	0.0984	4.2090	0.0998	4.2508	0.1021	4.3086	0.1045	**4.4191**

For correlated missing data, the results are more equivocal, in that the three methods, BLRHS, BGCP, and BRCP almost equally share the first place in different tasks. An important point to note is that since the metrics in question are RMSE and MAPE, the Gaussian likelihood used by [91] and [99] is arguably advantageous compared to BLRHS's Poisson likelihood. Given this point, and given that BLRHS is losing a portion of the data for model selection instead of providing results for all hyperparameters, we also take these results to be an encouraging sign, and consider improving accuracy of BLRHS in MAPE and RMSE further as an exciting future direction.

In this chapter we first presented a general framework for representing multiple periodic patterns in time series as an additive combination of underlying patterns. We connected our approach to the powerful formalism of low-rank tensor factorizations,

which allowed us to propose generally applicable tensor factorization algorithms for estimating parsimonious representations of cyclical patterns in time series data. Then, we utilized BAM to facilitate knowledge discovery not only through scaleable and accurate posterior inference, but also through likelihood-based model scoring. We showed that having access to a convenient probabilistic NTF framework allows interpretable and accurate modeling of complex seasonal patterns. In the following chapters, we will see other ways NMF/NTF expedites supervised learning.

3. LEARNING TO DEFER IN BINARY CLASSIFICATION

In Chapter 1 we described a probabilistic framework for nonnegative tensor factorization (NTF) called Bayesian allocation model (BAM), and in Chapter 2 demonstrated its efficacy in modeling complex seasonality in time series. This chapter and the next showcase the usefulness of NTF and BAM in a very different supervised learning setting: namely, *learning to defer*. Learning to defer is a research area that aims to exploit human-AI complementarity by repurposing the output of a machine learning model, such that the output is used both as a prediction *and* an indicator as to whether the decision should be deferred to an expert instead of using the model's prediction.

These two chapters are complementary in presenting our work in this area: Present chapter introduces the problem, reviews existing approaches, and proposes a new approach that excels in producing reliable results in binary classification, especially under sparsity. We then examine the performance of our approach in safety-critical applications, i.e. clinical binary classification involving breast cancer and tuberculosis diagnosis. In the next chapter, we show that generalizing our approach to *multiclass, multimodel, multiexpert* settings requires a methodology that can robustly model high-dimensional sparse data, for which BAM will be of great utility.

This chapter and the associated Appendix C contain excerpts, figures, and tables from the published article [54], in compliance with the publisher's reuse policies [59]. Source code for the experiments presented in this chapter can be found in the associated public repository [104]. The author's contributions to [54] were made during and after his employment as a researcher at DeepMind, London, United Kingdom. The datasets used in this chapter were acquired, processed, and provided for research by the said institution, including expert annotations and machine learning model predictions. See [54] for further details.

3.1. Learning to Defer

As machine learning (ML) models increase in predictive power, the areas of their application continue to expand. However, once these areas include safety-critical applications such as healthcare or predictive maintenance, specific shortcomings of such models become apparent. Therefore, in the last decade the emphasis on the development of hybrid decision making systems increased dramatically. The research in this domain can be found under various nomenclature, including but not limited to hybrid intelligence and human-AI collaboration [105, 106].

When a system combines the predictions of machine learning models and human experts, and performs better than any individual model or expert, it is said to exploit *complementarity* among these entities [107]. This is one of the primary aims of hybrid decision making systems. By the intelligent combination of models and experts, such systems not only can surpass the performance of any individual component, but also can reduce the overall cost of making decisions.

As mentioned above, a prominent approach to exploiting human-AI complementarity is called *learning to defer (LTD)*, which we now describe in detail. LTD refers to the development of systems in which given an input, the output of the machine learning model is used not only as a prediction, but also as an indication whether to *defer* this specific case to a downstream expert. The following questions highlight the unique benefits of LTD:

- Why combine ML and humans in decision making? ML models and humans make distinct errors, thus their combination can lead to decision making performance that surpasses both [107].
- Why deferral-based complementarity? Expert resources are limited, and their selective allocation improves overall system performance by removing expert effort from easier or more rote tasks [108].

3.1.1. Notation

In a classification task with L classes, we aim to predict a target label $\mathrm{y} \in \mathcal{Y} = \{1, \ldots, L\}$ based on the input $\mathbf{x} \in \mathcal{X}$, where $\mathbf{x}, \mathrm{y} \sim p(\mathbf{x}, \mathrm{y})$. More specifically, we seek a predictor function $f : \mathcal{X} \to \mathcal{Y}$ that predicts the associated label with satisfactory performance $\hat{\mathrm{y}} = f(\mathbf{x})$. This search is usually conducted with reference to a *loss function* $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, where we search within a space of predictor functions $\mathcal{F}$ to minimize *expected loss*, i.e., $\arg\min_{f \in \mathcal{F}} \mathbb{E}_{\mathbf{x},\mathrm{y}}\, \ell(\mathrm{y}, f(\mathbf{x}))$. In this chapter and the next, we assume $\mathcal{X} = \mathbb{R}^D$, unless explicitly specified otherwise. In a clinical binary classification setting with $\mathcal{Y} = \{0, 1\}$, we let 0 and 1 denote the absence and presence of a condition, respectively. As in previous chapters, we denote random variables with regular font, e.g. $\mathbf{x}, \mathrm{y}$, while their realized values are presented in italics, e.g. $\boldsymbol{x}, y$.

3.1.2. Problem Statement

We start with a simple problem setting, where we have access to a single human expert e and a single ML model m. The human expert $e : \mathcal{X} \to \mathcal{Y}$ produces a prediction given the input image, $\mathrm{h} = e(\mathbf{x})$, whereas the machine learning model $m : \mathcal{X} \to \mathbb{R}^L$ produces a vector of *confidence scores* $\mathbf{z} = m(\mathbf{x})$. To convert machine learning model's output to a prediction, a decision function $\delta : \mathbb{R}^L \to \mathcal{Y}$ is needed in practice. Without loss of generality, in the rest of the text we assume this function to be the often used maximum confidence rule $\delta(\mathbf{z}) = \arg\max_{i \in [L]} \mathrm{z}_i$, unless explicitly stated otherwise. Whenever required for brevity, we denote the composition of m and σ with μ, such that $\mu := \sigma \circ m$.

Given access to the machine learning model m and *query-based* access to the human expert e, the primary aim of LTD can be formulated as creating a *composite* decision making system c that produces predictions, $\hat{\mathrm{y}}^{(c)} = c(\mathbf{x})$. Central to this system

is the *deferral function* $\text{Defer} : \mathcal{X} \to \{0, 1\}$ such that

$$\hat{y}^{(c)} = c(\boldsymbol{x}) = \begin{cases} \mu(\boldsymbol{x}) & \text{if } \text{Defer}(\boldsymbol{x}) = 0, \\ e(\boldsymbol{x}) & \text{if } \text{Defer}(\boldsymbol{x}) = 1. \end{cases} \tag{3.1}$$

We use c for the overall system for brevity, yet a more appropriate, if somewhat cluttered notation would be $c_{\text{Defer},\delta,m,e}$ as c's behavior is determined by the functions $\text{Defer}, \delta, m, e$. As discussed above, the query-based access to the human expert annotator, that is, only accessing human response *if* deemed necessary, is central to LTD, as in many applications human expertise is in short supply and should be selectively allocated.

Importantly, instead of the original observations $\mathbf{x}$, the deferral function can also be formulated to map the confidence scores $\mathbf{z}$ to a deferral decision, such that $\hat{\mathrm{y}}^{(c)} = \text{Defer}(\mathbf{z})$. This is also related to another important distinction within LTD systems: an LTD system can assume the machine learning model to be fixed (pretrained) and only optimize the function Defer, or it can optimize m and Defer jointly. There are respective advantages to either approach that make them attractive and/or feasible in different problem contexts. Although we develop a methodology for the former based on reasons detailed in Section 3.1.4, we start by providing a general review of LTD research in order to provide a more complete picture.

3.1.3. Related Work

Most of the existing research focuses on *accuracy* as the primary metric for evaluating classification performance, so we start by defining this metric. We let $\mathbf{x}, \mathrm{y} \sim p(\mathbf{x}, \mathrm{y})$. We can express the accuracy of the composite system c as

$$\begin{aligned} \text{Accuracy}(c; p(\mathbf{x}, \mathrm{y})) &= \mathbb{E}_{\mathbf{x},\mathrm{y}} \mathbb{I}[\mathrm{y} = c(\mathbf{x})] && (3.2) \\ &= \mathbb{E}_{\mathbf{x},\mathrm{y}} (1 - \text{Defer}(\mathbf{x})) \mathbb{I}[\mathrm{y} = \mu(\mathbf{x})] + \text{Defer}(\mathbf{x}) \mathbb{I}[\mathrm{y} = e(\mathbf{x})], && (3.3) \end{aligned}$$

where (3.3) expands (3.2) according to how $\text{Defer}, \delta, m, e$ determine c. Notice that maximizing accuracy is equivalent to minimizing expected $0-1$ loss $\mathbb{E}_{\mathbf{x},\mathrm{y}} \ell_{0-1}(\mathrm{y}, c(\mathbf{x})) = \mathbb{E}_{\mathbf{x},\mathrm{y}} \mathbb{I}[\mathrm{y} \neq c(\mathbf{x})]$, given (3.2).

One way to produce a decision making system that optimizes accuracy is to parametrize the machine learning model m and the deferral function Defer, propose a *differentiable* surrogate to (3.3), and then either train Defer by itself or train it along with m (sequentially or jointly) using gradient-based training. This arguably is the most prominent approach in LTD, examples of which include [109–115].

Results by [109] are important for this line of research, in which they formulate LTD as a *cost-sensitive learning* problem. Specifically, the authors augment the output space to also include the deferral decision $\perp$ in addition to the L classes: $\mathcal{Y}^{\perp} = \mathcal{Y} \cup \{\perp\}$. They define $L+1$ functions such that $g_k : \mathcal{X} \to \mathbb{R}$ for $k \in [K]$ and $g_{\perp} : \mathcal{X} \to \mathbb{R}$. They optimize these according to the loss function defined as

$$\phi_{SM}(g_1, \ldots, g_K, g_{\perp}; \boldsymbol{x}, y, h) = -\log\left(\frac{\exp(g_y(\boldsymbol{x}))}{\sum_{y' \in \mathcal{Y}^{\perp}} \exp(g_{y'}(\boldsymbol{x}))}\right) - \mathbb{I}[h = y] \log\left(\frac{\exp(g_{\perp}(\boldsymbol{x}))}{\sum_{y' \in \mathcal{Y}^{\perp}} \exp(g_{y'}(\boldsymbol{x}))}\right). \tag{3.4}$$

Using the functions trained on the expected loss, they define the predictor μ and the deferral function Defer as

$$\mu(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} g_y(\boldsymbol{x}), \tag{3.5}$$

$$\text{Defer}(\boldsymbol{x}) = \mathbb{I}[g_{\perp}(\boldsymbol{x}) > \max_k g_k(\boldsymbol{x})], \tag{3.6}$$

where we remember that $\mu := \sigma \circ m$. In this setting, ϕ_{SM} is a convex (in $g_j, j \in [K] \cup \{\perp\}$) surrogate loss that is *Fisher consistent*, in that the minimizers of ϕ_{SM}, $g_1^*, \ldots, g_{\perp}^*$ also minimize the $0-1$ loss. More specifically, the optimal classifier and rejector satisfy the equations

$$\mu^*(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} p(\mathrm{y} = y|\boldsymbol{x}), \tag{3.7}$$

$$\text{Defer}^*(\boldsymbol{x}) = \mathbb{I}[p(\mathrm{h} = \mathrm{y}|\boldsymbol{x}) \geq \max_{y \in \mathcal{Y}} p(\mathrm{y} = y|\boldsymbol{x})]. \tag{3.8}$$

In addition to consistency, the fact that this loss is *differentiable* is attractive given the prevalence of gradient-based learning systems. Therefore, this result and the corresponding algorithm by the authors has a seminal role in LTD research. However, later research points out two potentially disadvantageous aspects of this approach.

As pointed out by [114], the first challenge is that (3.4) is not a *proper* surrogate loss, in that this function provides bad (and at times invalid) estimates for the accuracy of the expert i.e. $p(\mathrm{h} = \mathrm{y}|\mathbf{x})$. Accordingly, [114] partially mitigate this by an alternative formulation of [109]'s loss. The second problem with this approach is that it is not clear how this optimization procedure behaves under less than ideal circumstances, such as being optimized with finite samples, as well as the optimization taking place within a particular class of functions. Under ideal conditions, the composite algorithm provided in (3.5) and (3.6) can be expected to perform better than the Bayes-optimal ML classifier only if the annotators have access to input beyond $\mathbf{x}$, such as metadata. However, the fact that LTD approaches commonly improve performance even when the experts and the humans are exposed to the same information [108] implies that in practice these ideal performance conditions are not met. This in turn implies that understanding how $p(\mathbf{z}) = p(m(\mathbf{x}))$ behaves under these constraints is a critical issue.

Indeed, [115] address the losses by [109] and [114], and remark that although these losses are consistent, with modest changes to the problem setting they devolve into a strong *label smoothing* for the base model m, which disrupts estimation in *finite samples*. To resolve this, the authors suggest post hoc estimators to improve calibration of the resulting methods and show improved performance with respect to composite performance. These results imply that although consistency and differentiability are desirable properties for a loss function in LTD, the actual problem setting might be too disparate from the idealized conditions targeted by these seminal works, potentially requiring alternative approaches.

Also related are a set of methods proposed by [116], where they use a large neural network that co-calibrates its output to amortize the distributions $p(\mathrm{y}|\mathbf{x})$, $p(\mathrm{h}|\mathbf{x})$, and $p(\mathrm{y}|\mathbf{x}, \mathrm{h})$. Compared to the aforementioned work, this is a more direct approach in that it targets to learn distributions critical to assess the superiority of the ML model or the expert. However, as this learning procedure relies on approximating multiple probability distributions using gradient-based learning, it is potentially open to the same weaknesses with the approaches previously described, and without the statistical

guarantees provided by them.

3.1.4. Learning to Defer under Constraints

One consistent shortcoming of methods in the field of LTD is the fact that they require data-rich settings to perform well. Ironically, the exact opposite is likely to be the case in situations where LTD would be most desirable. For example, although reducing the burden of decision makers is very attractive in safety-critical applications, expert annotation is costly and is therefore scarce to begin with. Therefore, any LTD system that is supposed to function well in such an application has to be able to do so under (at times extreme) sparsity. The realistic settings where LTD might be useful can also present other unique challenges. For example, sometimes the machine learning predictor m in question can be proprietary and/or fixed under regulation, allowing only access to the confidence scores of the algorithm, $\mathbf{z} = m(\mathbf{x})$. Moreover, in certain contexts access to the inputs $\mathbf{x}$ themselves might be prohibited, e.g. due to privacy-related concerns. These limitations necessitate alternative approaches to LTD that can work under realistic constraints.

Motivated by the constraints of crucial healthcare applications, we develop an LTD system for binary image classification that optimizes the function Defer, assuming m as fixed, and requiring access only to model confidence scores $\mathbf{z}$ instead of the original images $\mathbf{x}$. The actual constraints and requirements of such applications call for methods which can *robustly* and *flexibly* improve performance, with crucial downstream benefits in terms of cost savings and improved diagnostic results. The potential advantages of such methods are also highlighted by the fact that the usefulness of joint training of m and Defer seems to decrease as the classifier capacity increases [112], and that classifiers trained to collaborate with a single, individual expert might be unreasonably brittle for future applications [106]. In our approach, we focus on the joint distribution of the classifier output, human annotation, and ground truth labels to determine deferral, and formulate learning the deferral function as a low-dimensional density estimation problem. The advantages of this approach are manifold:

(i) It *does not require access* to machine learning models or the input data (which might be confidential and/or proprietary),

(ii) It does not require the baseline classifier to be *calibrated*,

(iii) It does not rely on *absolute* performance estimates of the model vs. expert in the *input space*, but only on the *relative* estimates in the *confidence score space*,

(iv) It is generalizable to cases with *multiple models* and *experts*,

(v) It is generalizable to cases where the data is only *partially annotated* by any or all experts.

In this chapter and the next, we demonstrate the practical benefits conferred by these advantages.

3.2. Complementarity-Driven Deferral to Clinicians (CoDoC)

We now introduce our approach, called CoDoC (Complementarity-Driven Deferral to Clinicians), for binary classification in a safety-critical scenario, with a single ML model and a clinical expert. Given that we deal with binary labels $\mathrm{y} \in \{0, 1\}$, we assume that machine learning model produces a scalar *confidence score* $\mathrm{z} \in [0, 1]$ instead of the vector $\mathbf{z} \in \mathbb{R}^L$. As described above, in the problem setting we assume, at training time we do not have access to $\mathbf{x}$. Instead, we have access to model output $\mathrm{z} = m(\mathbf{x})$, as well as the expert opinion $\mathrm{h} = e(\mathbf{x})$, and lastly the ground truth label y. In this context, we assume the data distribution $\mathrm{z}, \mathrm{h}, \mathrm{y} \sim p(\mathrm{z}, \mathrm{h}, \mathrm{y})$. As above, we denote the output of the composite system CoDoC with $\hat{\mathrm{y}}^{(c)} = c_{\mathrm{Defer},\delta,e}(\mathrm{z})$, where c is no longer determined by m since c uses confidence scores z as input, and we similarly refer to the CoDoC system with c for brevity, unless specified otherwise.

In (3.3) we have expressed Accuracy for a composite system that takes $\mathbf{x}$ as input. We now modify this expression to accommodate our new system with the confidence score-based deferral function $\mathrm{Defer} : [0, 1] \to \{0, 1\}$,

$$\mathrm{Accuracy}(c; p(\mathrm{z}, \mathrm{h}, \mathrm{y})) = \mathbb{E}_{\mathrm{z},\mathrm{h},\mathrm{y}}(1 - \mathrm{Defer}(\mathrm{z}))\mathbb{I}[\mathrm{y} = \delta(\mathrm{z})] + \mathrm{Defer}(\mathrm{z})\mathbb{I}[\mathrm{y} = \mathrm{h}], \qquad (3.9)$$

where $\delta : [0, 1] \to \{0, 1\}$ is the decision function that converts the confidence scores of

ML model to predictions, as defined above. For a concise exposition, we will abbreviate (3.9) as Accuracy.

Given the application context, defining *sensitivity* and *specificity* for CoDoC at this point is also useful, since these metrics can have special importance in evaluating clinical procedures. We define sensitivity of the composite system such that

$$\begin{aligned}\text{Sensitivity}(c; p(\mathrm{z}, \mathrm{h}, \mathrm{y})) &= p(c(\mathrm{z}) = 1 | \mathrm{y} = 1) && (3.10)\\ &= \int_0^1 p(\mathrm{z} = z, \mathrm{h} = 1 | \mathrm{y} = 1)\text{Defer}(z) \\ &\quad + p(\mathrm{z} = z, \delta(z) = 1 | \mathrm{y} = 1)(1 - \text{Defer}(z))dz. && (3.11)\end{aligned}$$

Notice that first term in (3.11) contributes to the sum where the case is deferred to the expert, i.e. $\text{Defer}(z) = 1$, and the second term contributes otherwise $\text{Defer}(z) = 0$. We define specificity similarly, such that

$$\begin{aligned}\text{Specificity}(c; p(\mathrm{z}, \mathrm{h}, \mathrm{y})) &= p(c(\mathrm{z}) = 0 | \mathrm{y} = 0) && (3.12)\\ &= \int_0^1 p(\mathrm{z} = z, \mathrm{h} = 0 | \mathrm{y} = 0)\text{Defer}(z) \\ &\quad + p(\mathrm{z} = z, \delta(z) = 0 | \mathrm{y} = 0)(1 - \text{Defer}(z))dz. && (3.13)\end{aligned}$$

We abbreviate the two metrics as Sensitivity and Specificity. Note that it is possible to reduce them to a single metric G using a trade-off parameter λ, where our metric becomes a convex combination of the two,

$$G = \lambda \cdot \text{Sensitivity} + (1 - \lambda) \cdot \text{Specificity}. \tag{3.14}$$

Based on these, Proposition 1 describes the optimal decision making procedure.

Proposition 1 (Proposition 1, [54])**.** *Let* $\mathrm{z}, \mathrm{h}, \mathrm{y}$ *stand for the prediction of the ML model, human expert annotation, and ground truth label respectively. Given the conditional distribution* $p(\mathrm{z}, \mathrm{h}|\mathrm{y})$*, the decision function* δ *for the ML model, and sensitivity-specificity trade-off parameter* λ*, deferral function* Defer^* *that optimizes the following composite objective*

$$G = \lambda \cdot \text{Sensitivity} + (1 - \lambda) \cdot \text{Specificity} \tag{3.15}$$

can be computed by setting the advantage function Adv *to be*

$$\mathrm{Adv}(z) = \begin{cases} -(1-\lambda)p(\mathrm{z}=z,\mathrm{h}=1|\mathrm{y}=0) + \lambda p(\mathrm{z}=z,\mathrm{h}=1|\mathrm{y}=1), & \textit{if } \delta(z)=0, \\ (1-\lambda)p(\mathrm{z}=z,\mathrm{h}=0|\mathrm{y}=0) - \lambda p(\mathrm{z}=z,\mathrm{h}=0|\mathrm{y}=1), & \textit{if } \delta(z)=1, \end{cases} \tag{3.16}$$

and letting

$$\mathrm{Defer}^*(z) = \begin{cases} 0, & \textit{if } \mathrm{Adv}(z) \leq 0, \\ 1, & \textit{if } \mathrm{Adv}(z) > 0. \end{cases} \tag{3.17}$$

See [54] for the proof of this proposition. It is worth commenting on the implications of (3.16). $\mathrm{Adv}(z)$ is increased in cases where the human expert is more likely to be correct than the model, and decreased in the opposite case. How much the advantage function is increased by the said "correctness" is also weighted by λ according to this predetermined trade-off parameter.

This approach shifts the challenge of extracting complementarity to accurate density estimation, that is, estimating $p(\mathrm{z},\mathrm{h}|\mathrm{y})$. We now present our methodology for this task. The next chapter demonstrates that extending this methodology to multiple classes, models, and/or experts requires a qualitatively different approach, where BAM will become exceedingly useful.

3.2.1. Density Estimation

We now consider the problem of estimating the density $p(\mathrm{z},\mathrm{h}|\mathrm{y})$, which can also be written as $p(\mathrm{z},\mathrm{h}=h|\mathrm{y}=y); h, y \in \{0,1\}$. This expression can be factored as $p(\mathrm{z},\mathrm{h}=h|\mathrm{y}=y) = p(\mathrm{z}|\mathrm{h}=h,\mathrm{y}=y)p(\mathrm{h}=h|\mathrm{y}=y)$. Thus, our task reduces to estimating four 1-dimensional continuous conditional densities $p(\mathrm{z}|\mathrm{h}=h,\mathrm{y}=y)$ in addition to the discrete conditional distribution $p(\mathrm{h}|\mathrm{y})$. The conditional $p(\mathrm{h}|\mathrm{y})$ corresponds to the sensitivity and specificity of the clinical workflow, which are likely to be available from historical data. If not, these can be easily estimated from the confusion matrix based on the existing dataset. Therefore, the main challenge at this

stage is estimating $p(\mathrm{z}|\mathrm{h} = h, \mathrm{y} = y)$, which we now focus on.

Our approach for estimating $p(\mathrm{z}|\mathrm{h} = h, \mathrm{y} = y)$ is designed to address two main concerns in this task: *robustness* and *computational efficiency*. Need for robustness stems from the fact that our method should be applicable in cases where the number of observations are limited, especially considering all four combinations of h and y. Need for efficiency is due to the fact that the density estimation procedure needs to be repeated for different hyperparameter settings.

Guided by these concerns, our choice for this problem is a discretized kernel density estimation (KDE) method [117, 118]. KDE is an ubiquitous choice in 1-D density estimation problems, given its statistical robustness: KDE allows controlling the variance of the estimate, which is crucial for obtaining low errors at the sparse data regime [119]. While KDE is an attractive choice in and of itself, discretized KDE methods vastly improve the efficiency of KDE with little cost to accuracy, especially in 1-D density estimation tasks as demonstrated by previous work [120]. Thus, using discretized KDE allows addressing both concerns simultaneously.

A detailed description of the estimation procedure can be found in Appendix C.1. Overall, it involves creating a histogram of z for each $h, y \in \{0, 1\}$, smoothing the histogram counts using a kernel (in our case a Gaussian kernel), normalizing the histogram counts, and using the normalized count values to approximate $p(\mathrm{z}|\mathrm{h} = h, \mathrm{y} = y)$. See [118–120] for the generalization performance of discretized KDE methods. The procedure involves the following hyperparameters: number of bins (T), amount of prior (pseudo-)observations in these bins (κ), and the smoothing bandwidth (σ). These are determined through model selection as described later in Section 3.3.2 and Appendix C.2.

3.2.1.1. Threshold Search. Using an appropriate density estimation method leads to smooth and robust estimates of the conditional densities and therefore the the advantage function, which is critical for sparse data settings. A potential alternative is what

can be called a *threshold search*, which corresponds to applying the same strategy without any smoothing on the conditional distributions [121], approximating these with the empirical distribution. The fundamental problem with this approach is the fact that in most situations data cannot sufficiently constrain the advantage function. That is, the choice of threshold points are underdetermined: There are multiple choices for when to defer that achieve the same performance in terms of sensitivity and specificity on the available dataset. Our approach relies on a robust density estimation method to compute a smooth advantage function, which has unique zero crossings that define the deferral regions. Results presented in Section 3.3 will demonstrate the advantages of this robustness in terms of performance.

3.3. Experiments

We test our approach for two sets of clinical tasks: breast cancer detection from X-ray mammograms and recommending patients for diagnostic sputum testing for tuberculosis (TB) based on TB-suspicious findings in their chest X-ray. We consider multiple international clinical settings. For Mammogram X-rays we use a UK mammography dataset and two US mammography datasets, which differ not only in terms of the patients and technical practice involved but also expert diagnostic process. For TB detection we use a TB screening dataset from Bangladesh. See [54] for further details on these datasets.

Given the nature of the task, it is important that CoDoC can be integrated into decision making processes where there are established ML models which are proprietary and cannot be trained or fine-tuned. So, our datasets include different commercial AI model predictions as well as human expert opinions. As such, our results reflect directly on the present logistic/regulatory/algorithmic situation in application, in that any positive results directly translate to a potential benefit in current clinical practice. To emphasize their function in existing clinical workflows, at times we will refer to ML models involved in our experiments as *diagnostic AI* models.

3.3.1. Baselines

For each task, our aim is to produce a diagnostic composite system that surpasses *both* clinician and diagnostic AI performance. Therefore, the individual performances of the experts and the diagnostic AI will be our primary baselines. Although for reasons described above most existing methods in LTD are not directly applicable to our problem setting, an important alternative baseline is the threshold search method described in Section 3.2.1.1, which we compare with our method in Section 3.3.4.2.

As each diagnostic AI m produces confidence scores z $\in [0, 1]$, an operating point θ, i.e. threshold, needs to be selected to convert this score to a binary prediction. This corresponds to the decision function δ described above being defined as $\delta(z) = \mathbb{I}[z > \theta]$. The operating point for each diagnostic AI model is chosen to reflect the trade-off made between sensitivity and specificity by the clinical workflow. More specifically: for the purposes of comparing reader, diagnostic AI, and CoDoC performance, we obtain an operating point that matched the clinical workflow on specificity, another that matched the clinical workflow on sensitivity, and pick the mean of those two as the operating point for the diagnostic AI we compared to, following the published precedent in [122].

3.3.2. Hyperparameters, Model Selection, and Metrics

For each clinical task, the set of all available data is partitioned into a training set and a held-out test set (D_{test}), that is only used once for evaluating the final CoDoC system. The training set is further partitioned into a tuning set (D_{tune}) for the diagnostic AI and a validation set (D_{val}) that is used for model selection i.e., to determine the hyperparameters to be used. Different selections of hyperparameters, such as number of bins, smoothing bandwidth, and the sensitivity-specificity trade-off parameter introduced before (λ) lead to CoDoC models with different sensitivity and specificity performances. To evaluate our approach, in each experiment setting we create two versions of CoDoC: one in which model selection is targeted towards superior sensitivity with noninferior specificity, and another that seeks superior speci-

ficity with noninferior sensitivity (vis-a-vis expert(s) and diagnostic AI). In the results presented, these CoDoC models are referred to as "CoDoC for sens." and "CoDoC for spec." respectively. Further details on our model selection procedure can be found in Appendix C.2.

3.3.3. Datasets and Diagnostic AI Systems

As stated before, we test our approach in multiple settings within a mammography breast cancer detection task and TB detection task. For mammography, we include *three datasets* and *two diagnostic AI systems* from previous research [123, 124]. The datasets UK Mammography Dataset 1 (UKMD1) and US Mammography Dataset 1 (USMD1) are from [123], whereas US Mammography Dataset 2 (USMD2) is from [124]. For tuberculosis, the chest X-ray dataset (TBCXR) was obtained from [125]. Table 3.1 summarizes the number of observations (number of positive observations in parantheses) in each of these datasets and splits.

Table 3.1. Number of cases in each dataset and their respective splits. Number of positive cases are specified in parantheses. UKMD1 has 25497 (405) test observations when using the first reader scores due to missing data for some cases [54].

Dataset	Tune	Validation	Test
UKMD1	30377 (590)	30378 (608)	25811 (417)
USMD1	4392 (69)	4391 (70)	8981 (174)
USMD2	4739 (21)	2335 (10)	7074 (30)
TBCXR	5988 (918)	5989 (919)	11977 (1838)

Our baseline models for mammography, Mammography Diagnostic AI 1 (MDAI1) and Mammography Diagnostic AI 2 (MDAI2) are from [123] and [124] respectively. The tune and validation splits of UKMD1 and USMD1 are used in hyperparameter selection for MDAI1. Tune and validation splits of USMD2 are used in the training of MDAI2. Test splits used to assess the performance of CoDoC, expert, or diagnostic AI systems were not seen by either of the diagnostic AI systems. For tuberculosis, we have five

commercial diagnostic AI systems (TBAI1-TBAI5) that were examined in [125]. None of the AI systems used TBCXR in training or model selection.

With respect to expert annotations in mammography datasets, the US datasets (USMD1 and USMD2) include opinions from a single reader. On the other hand, the UK dataset (UKMD1), in addition to opinions from a single reader, also include the consensus opinions from 2 or 3 radiologists, reflecting different clinical processes within different countries. This also gives us opportunity to examine whether CoDoC still confers advantage given the presence of an "expert" of higher performance (as produced by the panel consensus). In tuberculosis detection, TBCXR expert annotations correspond to the consensus opinion of 3 medical experts. [54] contains further information regarding the datasets and their respective clinical data collection processes.

3.3.4. Results

Results with different datasets and diagnostic models for mammography classification tasks can be seen in Tables 3.2, 3.3, and 3.4, and in Figures 3.1 and 3.2. Similarly, results for classification task in tuberculosis diagnosis can be found in Table 3.5 and Figure 3.3. In all tables, boldface entries refer to statistically significant superiority (with significance threshold of 0.05), non-boldface entries imply noninferiority, to both individual expert and AI performances. Superiority and non-inferiority were defined by standardized hypothesis tests (McNemar and Wald; see Appendix C.3). In all tables, entries with that include "-" are same as above.

Table 3.2. Mammography results with dataset UKMD1 and model MDAI1 [54].

Clinical Workflow	CoDoC		Reader(s)		AI Only	
Defer to single reader, CoDoC for sens.	**72.6**	91.8	62.7	92.8	64.9	93.9
Defer to single reader, CoDoC for spec.	64.4	**96.5**	-	-	-	-
Defer to reader consensus, CoDoC for sens.	**71.0**	95.1	67.6	96.1	-	-
Defer to reader consensus, CoDoC for spec.	68.3	**96.9**	-	-	-	-

Our results show that CoDoC reliably and significantly improves upon both expert *and* diagnostic AI's performance in almost all settings that we described. In Figure 3.1, we can see that in experiments with the UK dataset UKMD1 and baseline model MDAI1, CoDoC can combine the reader and the diagnostic ML model to surpass the performance of either, regardless of the reader quality. More specifically, note that when the diagnostic AI and single reader (green circle) is combined, CoDoC produces results (dashed line with circles) that push the Pareto frontier beyond expert and AI performance in terms of sensitivity and specificity. When the diagnostic AI is the same but we take a better reader into the system (pink diamond), CoDoC can *further* push the performance of the composite system, as is seen in the dashed line with diamonds. Results in Table 3.2 show that in addition to pushing the Pareto front beyond the baseline AI and expert(s), models selected by CoDoC perform significantly better than both the baselines when model selection is conducted to optimize sensitivity (resp. specificity), with statistically non-inferior performance in specificity (resp. sensitivity). Table 3.3 presents a similar set of results in the US sample with dataset USMD1 and model MDAI1.

Table 3.3. Mammography results with dataset USMD1 and model MDAI1 [54].

Clinical Workflow	CoDoC		Reader(s)		AI Only	
Defer to single reader, CoDoC for sens.	**56.9**	87.5	50.0	88.0	48.3	86.3
Defer to single reader, CoDoC for spec.	50.0	**91.2**	-	-	-	-

An even more interesting case is presented in Figure 3.2, where the dataset is characterized by extreme sparsity, where the tune and validation sets have 21 and 10 positive examples each. Even in this case, CoDoC can reliably improve upon specificity of both the diagnostic model and the clinician, as can also be seen in Table 3.4. Note that in this case CoDoC does not provide improvement to sensitivity as the clinician has near perfect performance in terms of sensitivity already. Note that unlike CoDoC, this sparsity leads the threshold search method described above to fail dramatically in terms of generalization, as seen in Figure 3.2 and further discussed in Section 3.3.4.2. It is also important to note that the advantage conferred by CoDoC is achieved despite

constraining the percentage of deferrals to below 50% of cases in all problem settings in tune and validation splits during model selection.

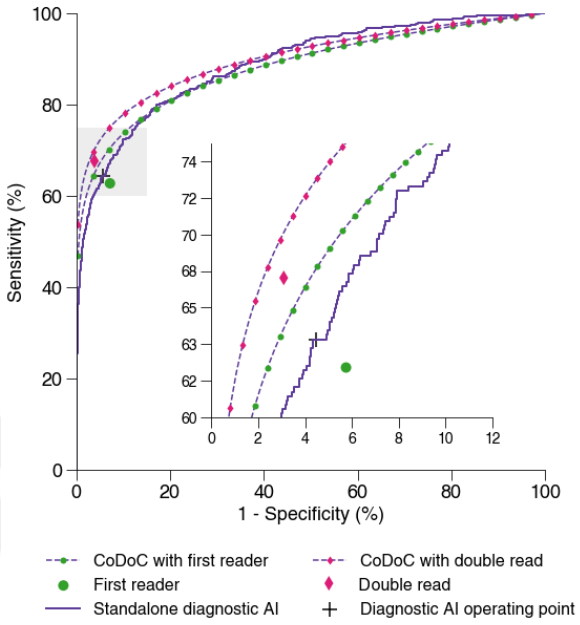


Figure 3.1. Mammography results with dataset UKMD1 and model MDAI1 [54].

Table 3.4. Mammography results with dataset USMD2 and model MDAI2 [54].

Clinical Workflow	CoDoC		Reader(s)		AI Only	
Defer to single reader, CoDoC for sens.	96.7	**90.9**	96.7	84.5	86.7	70.0

In TB experiments, as can be seen in Figure 3.3 and Table 3.5, CoDoC benefits specificity significantly for 3 out of 5 models. The statistical power is low in this setting for sensitivity, so improvements in sensitivity are not significant. However, this is not the case for specificity, and the next section analyzes this phenomenon.

3.3.4.1. Limits of Confidence-Based Deferral. The only setting where CoDoC does not achieve performance improvement in specificity is tuberculosis detection with the last two diagnostic AI models, as shown in Table 3.5. We follow this point up to assess whether, even under idealized conditions, any confidence based deferral strategy could have succeeded in improving performance in these two settings. To do so, we make two

changes to the CoDoC system: First, we allow as many as ten deferral regions, allowing the system to learn a very flexible function of the confidence score to determine when to defer. Moreover, we train the system directly *on the test set* used for evaluation, in order to test the limits of this approach more thoroughly, free from the statistical limitations on generalization in this limited sample regime. The performance of CoDoC in this setting can therefore be seen as a theoretical upper bound on the performance of any deferral strategy that is based only on the confidence score of the AI model.

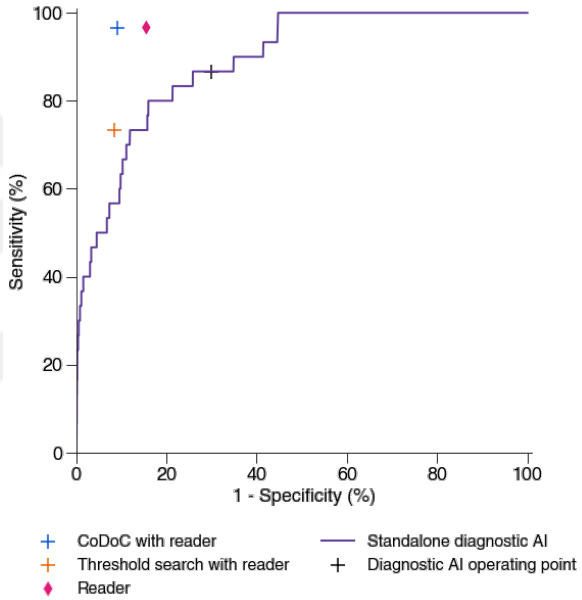


Figure 3.2. Mammography results with dataset USMD2 and model MDAI2 [54].

The results in Figure 3.4 demonstrate that, for the two strongest commercially available TB models we study, this theoretical performance limit (as plotted in the red dashed line) is very close to the performance of the diagnostic AI model, thereby showing that confidence based deferral strategies had very little potential for improving performance in this setting to begin with.

3.3.4.2. Comparison of CoDoC with Threshold Search. We present results comparing our approach with the previously mentioned threshold search approach proposed in prior work [121]. We find that on datasets where we have a large tuning dataset (like

UKMD1), the two approaches gave nearly identical results. This can be attributed to the fact that when discretized, the empirical distribution might be sufficient to provide good density estimation in data rich regimes. However, when the tuning dataset is small (like USMD2), we find that the probabilistic approach drastically outperforms the threshold search approach [121] on unseen test data, as seen in Figure 3.2.

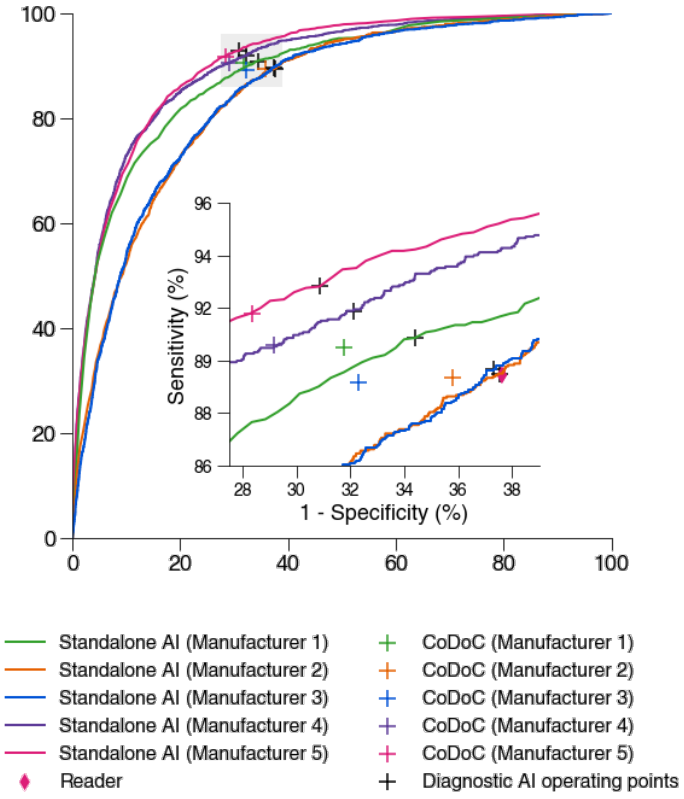


Figure 3.3. Tuberculosis results with dataset TBCXR and models TBAI1-TBAI5 [54].

3.4. Discussion

Our results show that our approach to LTD confers robust improvements to the existing clinical workflow in mammography and tuberculosis detection, improving upon the individual performances of the human expert and the machine learning model. We also demonstrate that the cases where CoDoC did not outperform the existing systems might reflect limits of confidence-based deferral [107]. The next chapter generalizes our approach to multiple classes, models, and readers.

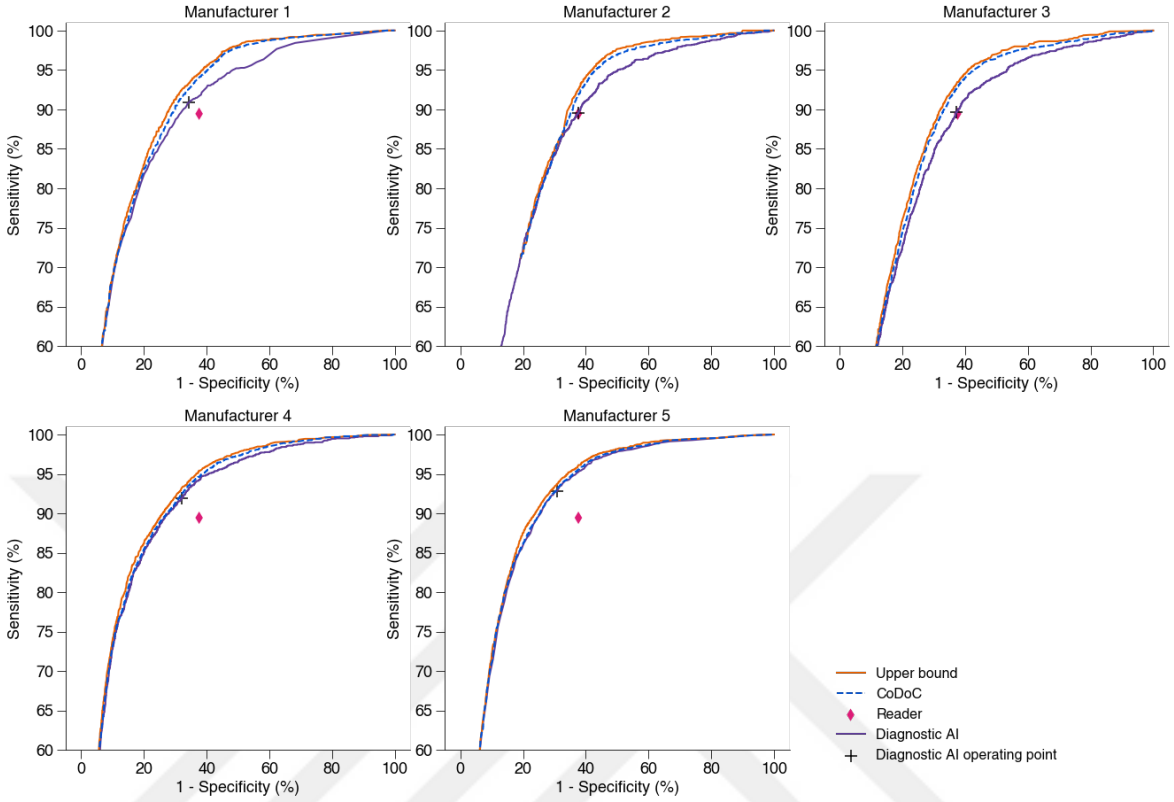


Figure 3.4. Examining the limits of confidence based deferral in tuberculosis experiments, for TBAI1-5 [54].

Table 3.5. Tuberculosis results with dataset TBCXR and models TBAI1-TBAI5. The clinical workflow involves deferral to reader consensus, and CoDoC for specificity [54].

AI model	CoDoC		Reader(s)		AI Only	
TBAI1	90.5	**68.3**	89.4	62.4	90.9	65.6
TBAI2	89.3	**64.2**	-	-	89.5	62.5
TBAI3	89.5	**62.5**	-	-	89.2	67.7
TBAI4	89.7	62.7	-	-	90.6	70.8
TBAI5	91.9	67.9	-	-	91.8	71.7

4. LEARNING TO DEFER IN ADVANCED SCENARIOS

In the previous chapter we introduced a robust framework for learning to defer (LTD), where we formulated deferral based on the outputs of a pretrained machine learning model, and tested this approach under a useful yet simple scenario: binary classification with a single machine learning (ML) model and a single human expert. In this chapter, we demonstrate that generalizing this useful framework to multiple classes, models, and experts requires an improved methodology for handling high dimensional density estimation, for which Bayesian allocation model (BAM) is uniquely well-suited.

4.1. Generative Model for Learning to Defer

We now examine the potential challenges of extending the approach described in the previous chapter, starting by examining the generative model for the LTD problem. The LTD variants that propose the joint training of the machine learning model m and the deferral function Defer such as [109], [112] implicitly assume the generative model presented in Figure 4.1(a). At training time, these methods assume access to $\mathbf{x}, \mathrm{h}, \mathbf{z}$, and y. As stated in (3.5) and (3.6), in this approach the intention is to train the machine learning model and the deferral function such that $\mu(\mathbf{x}) = \arg\max_y \; p(\mathrm{y} = y|\mathbf{x})$ and $\mathrm{Defer}(\mathbf{x}) = \mathbb{I}[p(\mathrm{y} = \mathrm{h}|\mathbf{x}) > \max_y p(\mathrm{y} = y|\mathbf{x})]$.

As explained in Chapter 3, CoDoC is designed for situations where we have no access to the data or the parameters of the machine learning model for privacy, regulatory, and/or commercial reasons. This corresponds to the model depicted in Figure 4.1(b), where $\mathbf{x}$ is now a latent variable. Our observations in this setting only include $\mathbf{z}, \mathrm{h}, \mathrm{y}$, and $p(\mathbf{z}, \mathrm{h}, \mathrm{y})$ conform to the graph that results from marginalizing over the latent $\mathbf{x}$, as depicted in Figure 4.1(c). Accordingly, in the previous chapter's Section 3.2.1, we proposed a method to estimate the joint conditional distribution $p(\mathbf{z}, \mathrm{h}, \mathrm{y})$: we used the trade-off parameter λ to stand in for $p(\mathrm{y} = y)$ and estimated $p(\mathbf{z}, \mathrm{h}|\mathrm{y})$ from the dataset. Let us now examine the complexity of this probabilistic

estimation task and how it grows with number of classes, models, and experts.

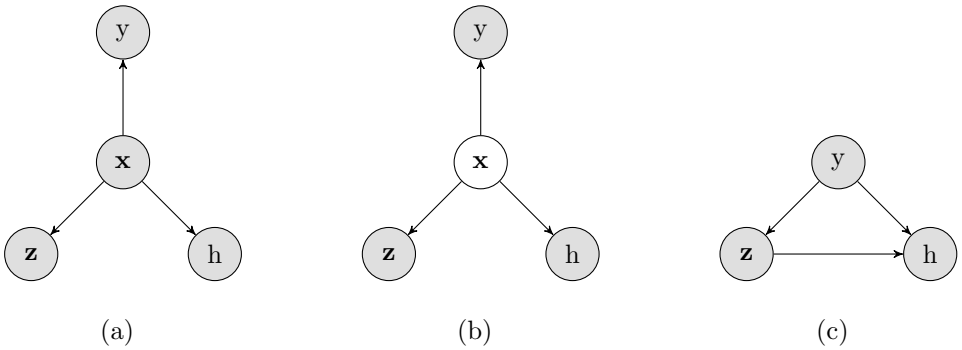


Figure 4.1. (a) The graphical model for when input $\mathbf{x}$ is observed. (b) The generative model when $\mathbf{x}$ is not observed. (c) Marginalizing over $\mathbf{x}$, we end up with a complete graph, which CoDoC relied on learning in binary classification.

To make the analysis easier, assume $\mathbf{z}$ have been discretized with an appropriate method with the number of bins $T = 100$, and let the output space be binary as in the previous chapter, $\mathcal{Y} = \{0, 1\}$. Then, the probability distribution $p(\mathbf{z}, \mathrm{h}, \mathrm{y})$ has $100 \times 2 \times 2 = 400$ parameters to be estimated (ignoring the -1 term that should follow from the available degrees of freedom for brevity).

Let us now make the problem more challenging by assuming a multiclass classification problem with 16 classes, and that we obtain confidence scores $\mathbf{z}^{(1)}$ and $\mathbf{z}^{(2)}$ from two different machine learning models, and annotations $\mathrm{h}^{(1)}$ and $\mathrm{h}^{(2)}$ from two separate experts. In this case the number of parameters that describe $p(\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \mathrm{h}^{(1)}, \mathrm{h}^{(2)}, \mathrm{y})$ would equal $100 \times 100 \times 16 \times 16 \times 16 = 4,096,000$, which would be much more challenging to estimate with finite data under most realistic circumstances.

One possible way of sidestepping this complexity however, would be to assume that this joint distribution has certain conditional independences that reduce dimensionality based on a latent structure. For example, if we assume the existence of a latent variable that renders the observed variables conditionally independent, as in the CP decomposition we utilized with BAM, this would expedite the estimation task

immensely. Calling this latent variable $\hat{\mathbf{x}}$, our proposed model would look as depicted in Figure 4.2. Comparing this model with Figure 4.1(b) shows that this assumption is not only convenient but also is natural given the structure of the problem: In our case, this latent variable would correspond to a (lower-dimensional) reconstruction of $\mathbf{x}$. In terms of the complexity of estimation, given a latent variable $\hat{\mathbf{x}}$ with 50 dimensions, for example, this would reduce the number of parameters of the joint probability distribution to $11,600$, roughly corresponding to a 400-fold decrease in parameters. As mentioned above, this problem can be formulated as probabilistic CP decomposition, for which BAM is well-suited for inference and model selection.

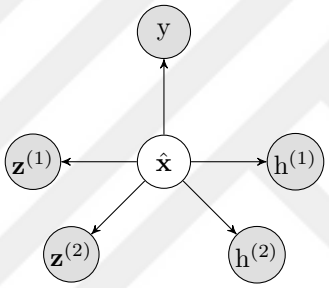


Figure 4.2. Graphical model that results from a low-rank assumption on the observed variables in a scenario with multiple classes, two experts, and two experts.

4.2. Modeling Various Deferral and Annotation Scenarios

Now that we have formulated our problem through the lens of probabilistic latent variable modeling, and that we have a convenient inference and model selection methodology in BAM, we can explore the different modeling opportunities that this approach affords. For all scenarios below, we assume a multiclass classification task and that models' confidence scores $\mathbf{z}^{(i)}$ are discretized in a task-appropriate fashion. The flexibility of BAM allows expediently implementing the various modeling assumptions we describe below.

4.2.1. Multiple Machine Learning Models and Annotators

A straightforward extension of the scenario above is the inclusion of arbitrary number of predictors and human experts. This also highlights the unique strengths of our approach. Many methods in the literature are limited by specific deferral scenarios [109], [112], whereas as we demonstrate below, this is not a limitation that our approach faces. Let us have M machine learning models and E human experts, with their respective confidence scores $(\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(M)})$ and annotations $(\mathrm{h}^{(1)}, \ldots, \mathrm{h}^{(E)})$. The graphical model for this scenario is demonstrated in Figure 4.3. In Section 4.3.1 we provide ideal decision making rule for this scenario.

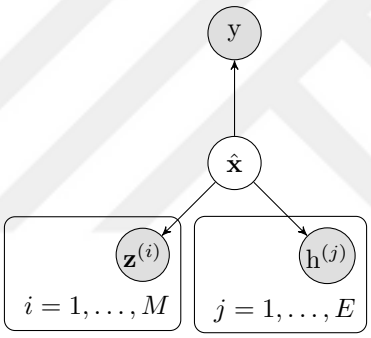


Figure 4.3. The graphical model for a scheme with M models and E experts.

4.2.2. Single Annotator Assumption

We observe that other ways of structuring the latent space corresponds to different well-known modeling assumptions on the supervision signal. For example, we might assume that all observer annotations are conditionally independent samples from a latent prototypical human annotator. Although this approach has its obvious downsides, it sometimes can be attractive to make this assumption especially in cases where annotators are anonymized and/or annotations per annotator are too small in number to make any inference with respect to specific annotators' characteristics. This approach for annotation results in the graphical model depicted in Figure 4.4.

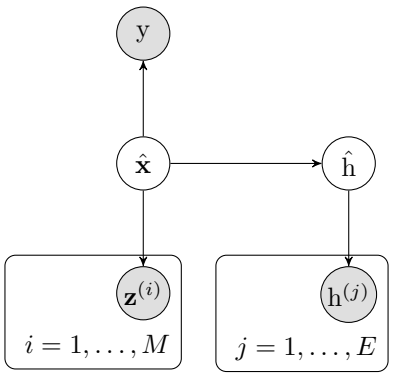


Figure 4.4. The graphical model for a scheme with M models and E annotators, with annotators assumed to be conditionally independent given the annotation of the prototypical annotator, $\hat{\mathrm{h}}$.

4.2.3. Ensembling Machine Learning Models

Note that we can use this methodology for conducting ensembling among the machine learning models, from which we have the confidence scores. In our default methodology this would lead to a discrete decision in terms of which algorithm's prediction to use; however this can easily be modified to conduct (weighted) voting on the outputs of the models. The graphical model would look like in Figure 4.5 in this scenario.

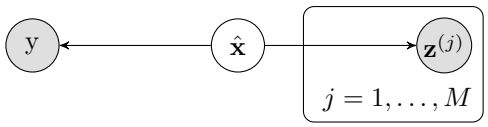


Figure 4.5. Ensembling among models using our methodology.

4.2.4. Dawid-Skene Aggregation Model

To show the flexibility of our framework, we briefly present an example from the *annotation modeling* literature. A usually encountered scenario in this research area is having access to $\mathbf{x}$, but instead of y, only having access to multiple annotations for each

input $(\mathrm{h}^{(1)}, \ldots, \mathrm{h}^{(E)})$. A commonly used model in this literature is the Dawid-Skene model of noisy annotations [126], which assumes reader-specific class-conditional error characteristics in the form of

$$p(\mathrm{h}^{(1)} = h^{(1)}, \ldots, \mathrm{h}^{(E)} = h^{(E)} | \mathrm{y} = y) = \prod_{i=1}^{E} p(\mathrm{h}^{(i)} = h^{(i)} | \mathrm{y} = y). \tag{4.1}$$

This model can be expressed with the graphical model depicted in Figure 4.6. A straightforward utilization of our framework for this case would be to infer $p(\mathrm{y}|\mathrm{h}^{(1)}, \ldots, \mathrm{h}^{(E)})$ based on the latent variable structure depicted, and use the posterior probability distribution for supervision. This model can be extended in multiple ways to allow directly training on noisy supervision labels, as well as correctly evaluating under them. See the paper [127], co-authored by the author of this thesis for a discussion of evaluation under such an aggregation model.

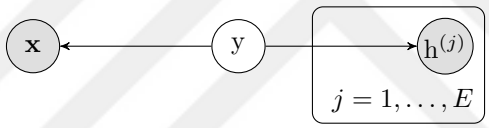


Figure 4.6. The graphical model for Dawid-Skene model of annotation noise.

4.3. Learning to Defer under Arbitrary Modeling Scenarios

Having described why and how to extend our approach to multiple classes, models, and experts, we now redefine our framework to accommodate this improved problem setting. We denote the output of the M ML models as

$$\mathbf{Z} = [\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(M)}] = [m^{(1)}(\mathbf{x}), \ldots, m^{(M)}(\mathbf{x})] \tag{4.2}$$

and the query-based output of E human annotators as

$$\mathbf{h} = (\mathrm{h}^{(1)}, \ldots, \mathrm{h}^{(E)}) = (e^{(1)}(\mathbf{x}), \ldots, e^{(E)}(\mathbf{x})). \tag{4.3}$$

Letting $K = M + E$, we concatenate the predictions of all ML models and human experts in the vector $\hat{\mathbf{y}} \in \mathcal{Y}^K$ such that

$$\hat{\mathbf{y}} = (\delta(\mathbf{z}^{(1)}), \ldots, \delta(\mathbf{z}^{(M)}), \mathrm{h}^{(1)}, \ldots, \mathrm{h}^{(E)}) \tag{4.4}$$

$$= (\hat{\mathrm{y}}^{(1)}, \ldots, \hat{\mathrm{y}}^{(K)}). \tag{4.5}$$

We still denote the ultimate composite decision making system and its prediction as $\hat{\mathrm{y}}^{(c)} = c_{\mathrm{Defer},\delta,\mathbf{e}}(\mathbf{Z})$, again abbreviated as c, which includes the generalization of the deferral function to the current setting, $\mathrm{Defer} : \mathbb{R}^{L\times M} \to [K]$. We still assume that by default we have access to the confidence scores of M models, and access and adopt human annotator i's prediction $\hat{\mathrm{y}}^{(c)} = \hat{\mathrm{y}}^{(i)}$ only if $i = \mathrm{Defer}(\mathbf{Z})$.

4.3.1. Optimal Decision Making

We now determine the optimal decision making rule with respect to accuracy in the multi-class context, with multiple models and experts, as in Figure 4.3.

Proposition 2. *Define the vector valued advantage function* $\mathbf{Adv}(\boldsymbol{Z}) : \mathbb{R}^{L\times M} \to \mathbb{R}^{K}_{\geq 0}$, *such that* $\mathbf{Adv}(\boldsymbol{Z}) = (\mathrm{Adv}_1(\boldsymbol{Z}), \ldots, \mathrm{Adv}_K(\boldsymbol{Z}))$, *where given* $p(\mathbf{Z}, \hat{\mathbf{y}}, \mathrm{y})$, *we let*

$$\mathrm{Adv}_i(\boldsymbol{Z}) = \sum_{y\in\mathcal{Y}} p(\mathbf{Z} = \boldsymbol{Z}, \hat{\mathrm{y}}^{(i)} = y, \mathrm{y} = y). \tag{4.6}$$

Then, the function Defer^* *that maximizes the accuracy of* c *is*

$$\mathrm{Defer}^*(\boldsymbol{Z}) = \underset{i\in[K]}{\arg\max}\ \mathrm{Adv}_i(\boldsymbol{Z}). \tag{4.7}$$

Proof. Absent any global limits on deferral, we seek

$$\underset{c}{\arg\max}\, p(\mathrm{y} = c(\mathbf{Z})) = \underset{c}{\arg\max} \int p(\mathrm{y} = c(\boldsymbol{Z})|\mathbf{Z} = \boldsymbol{Z}) p(\mathbf{Z} = \boldsymbol{Z}) d\boldsymbol{Z}. \tag{4.8}$$

Then, given any $\boldsymbol{Z}$, we need to choose a predictor such that

$$\underset{i\in[K]}{\arg\max}\ p(\mathrm{y} = \hat{\mathrm{y}}^{(i)}|\mathbf{Z} = \boldsymbol{Z}). \tag{4.9}$$

Notice that (4.6) can alternatively be written as

$$\begin{aligned}
\mathrm{Adv}_i(\boldsymbol{Z}) &= p(\mathrm{y} = \hat{\mathrm{y}}^{(i)}, \mathbf{Z} = \boldsymbol{Z}) && (4.10)\\
&= p(\mathrm{y} = \hat{\mathrm{y}}^{(i)}|\mathbf{Z} = \boldsymbol{Z}) p(\mathbf{Z} = \boldsymbol{Z}) && (4.11)\\
&\propto p(\mathrm{y} = \hat{\mathrm{y}}^{(i)}|\mathbf{Z} = \boldsymbol{Z}), && (4.12)
\end{aligned}$$

which implies that

$$\mathrm{Defer}^*(\boldsymbol{Z}) = \underset{i\in[K]}{\arg\max}\ \mathrm{Adv}_i(\boldsymbol{Z}) = \underset{i\in[K]}{\arg\max}\ p(\mathrm{y} = \hat{\mathrm{y}}^{(i)}|\mathbf{Z} = \boldsymbol{Z}). \tag{4.13}$$

□

Notice that if we assume no *class imbalance*, we set $p(\mathrm{y}) = \frac{1}{L}$, which converts (4.6) to

$$\mathrm{Adv}_i(\boldsymbol{Z}) = \frac{1}{L}\sum_{y\in\mathcal{Y}} p(\mathbf{Z} = \boldsymbol{Z}, \hat{\mathrm{y}}^{(i)} = y|\mathrm{y} = y), \tag{4.14}$$

further simplifying the estimation of the advantage function.

Similar to the binary classification case, the success of our method depends on reliably estimating the joint distribution $p(\mathbf{Z}, \hat{\mathbf{y}}, \mathrm{y})$, or the joint conditional $p(\mathbf{Z}, \hat{\mathbf{y}}|\mathrm{y})$ if we assume no class imbalance. We showed above that invoking BAM to handle this high-dimensional estimation problem makes this a realistic proposition. However, as mentioned above, BAM requires $\mathbf{Z}$ to be discretized, and the grid-based methodology used in Chapter 3 is not feasible due to the high dimensionality of the product space. Below we propose another method for this discretization task.

4.3.2. Discretization of Model Outputs

Given the high dimensionality of the task and the potential scarcity of the data, we adopt a nonparametric approach and use a *Dirichlet process mixture model (DPMM)* to discretize our model output $\mathbf{Z}$. DPMM is an example of what are called *infinite mixture models* [49]. Intuitively, DPMM assumes that the data belong to clusters of which there are infinitely many. Cluster-conditional densities can be assumed parametric $F(\mathbf{x}|\psi)$, where each cluster has its own parameters ψ. If we assume $F(\mathbf{x}|\psi)$ to be a Gaussian distribution, for example, each cluster k would have a mean η_k and covariance Σ_k. As mentioned above, in DPMM these clusters are such that $k \in 1, \ldots, \infty$. Given N observations, the generative model for DPMM can be written as below

$$G \sim \mathrm{DP}(\alpha, H), \tag{4.15}$$

$$\bar{\psi}_i \sim G, \qquad i = 1, \ldots, N, \tag{4.16}$$

$$\mathbf{x}_i \sim F(\bar{\psi}_i), \qquad i = 1, \ldots, N, \tag{4.17}$$

where DP stands for a *Dirichlet process*. Given these, the infinite mixture model is defined as $G_F(\mathbf{x}) \sim \sum_{k=1}^{\infty} \pi_k F(\mathbf{x}|\psi_k)$, where π_k are the weight of each mixture component. As mentioned above, a common choice for F is a Gaussian distribution with

$\psi = (\eta, \Sigma)$, and a complementing choice for H is normal-inverse-Wishart distribution. In practice, posterior inference with DPMM is used to estimate a number of clusters that explain the existing data, and (soft or hard) assignment of the existing data to the said clusters. The flexibility offered by DPMM makes it an appropriate choice for our purposes in this context. There are various method for fitting a DPMM [49]. In our experiments, we use an existing library in `bnpy` [128], through which we use a variational Bayes algorithm for inference, assuming Gaussian distributions with diagonal covariance for F. Note that in this approach we discretize the confidence scores of different machine learning models $\{\mathbf{Z}^{(n)}\}_{n=1}^{N}$ jointly, potentially exploiting the mutual information of different models to obtain better estimates of performances of all predictors.

4.4. Experiments

To test our extended framework for exploiting complementarity, we use the ImageNet-16H dataset published by [107]. Below we first describe the dataset, introduce our experimental setting, and finally present our results.

4.4.1. Dataset

ImageNet-16H is a dataset created by [107] to investigate human-AI complementarity. The dataset includes 1200 unique images belonging to 16 classes from the ImageNet Large Scale Visual Recognition Challenge (ILSRVR) dataset [129]. To make classification for human annotators and ImageNet trained classifiers more difficult, each image has four noisy variants, with each progressively more distorted by phase noise, as demonstrated in Figure 4.7. The dataset includes predictions (confidence scores) from five public machine learning models: AlexNet [130], DenseNet161 [131], ResNet152 [132], GoogleNet [133], and VGG-19 [134]. It also includes predictions from six human annotators per image, though the annotations are crowdsourced from potentially different human annotators. Maximum entries that a single annotator has provided predictions for are 200. The annotators also provide how confident they

are in their predictions, which the authors use in the original paper when estimating human-AI complementarity in a given task [107].

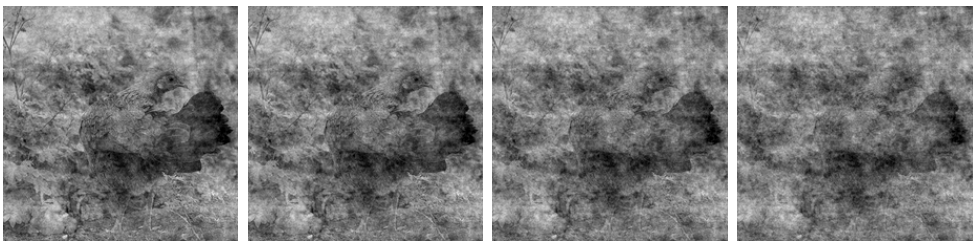


Figure 4.7. An example image from ImageNet-16H, showing an image with progressively more noise [107].

4.4.2. Experimental Settings

Given the fact that the identity of the human annotators vary throughout the dataset, we use machine learning models to stand in for human annotators as in [109], [114]. When a model is used as an ML model proper, we use its confidence scores; when it stands in for a human expert we only use its predicted labels, determined through the maximum confidence rule. In contrast to model selection with CoDoC in the binary classification case, we now employ a Bayesian approach and conduct model selection through the marginal likelihood estimate, thus do not use a validation set. Therefore we use 3600 observations as our tuning set (D_{tune}) and 1200 observations as our test set (D_{test}), where the samples are stratified in the splits proportionally according to 16 classes and 4 noise levels.

For all experiment settings we describe below, the overall procedure is similar: we first discretize the machine learning models' output with DPMM, which allocates each model prediction to a single cluster in the prediction space. We then create a tensor that includes DPMM allocations, true labels, and predictions from experts. As a concrete example, if we have two machine learning models, one expert, and if DPMM produced 40 clusters, then our tuning set can be represented by a $40 \times 16 \times 16$ tensor $\mathbf{T}$, with a total of 3600 observations. Note that the tensor dimensions ignore the prediction

space of the machine learning models. This is intentional, as we exploit the fact that we can use the clustering results of models as an implicit ensembling, and directly conduct inference on the resulting cluster indices. This greatly expedites computation with almost no cost to performance. However, if there is any particular reason to do so, one can keep the machine learning models' predictions in $\mathbf{T}$.

When conducting experiments, after clustering the continuous output by the model(s) and creating the tensor $\mathbf{T}$, we use BAM with the CP latent structure portrayed in Figure 4.3. We use mean-field variational inference (MFVI) for obtaining parameter estimates, and we use the associated evidence lower bound (ELBO) to conduct model selection for the nonnegative rank R as well as the hyperparameter a, as described in Section 1.3.2 and Appendix A.2.

4.4.3. Results

In Table 4.1 we first present the individual baseline tune and test accuracies of the algorithms in question, so that the later experiment results can be compared against them. Then, we aim to see if we can exploit complementarity in a multiclass classification setting given one model and one expert. In order to make sure that complementarity can be exploited both when the model or the expert is performing better overall, for each model-expert combination we include both directions in the experiments. In order to be able to constrain the number of experiments, we use the three best performing models, namely ResNet152, GoogleNet, and DenseNet161. The results for our experiment for all six model-expert combinations can be seen in Table 4.2. The results are straightforward to interpret: our approach can exploit complementarity reliably with a small set of observations, and notably does so in a way that performance increases can happen when the deferred-to expert is overall performing lower than the original model or vice versa. This confirms a crucial insight for complementarity research: even an overall worse performing predictor might be performing better in a subset of inputs; the ideal of learning to defer is to learn when this is the case and exploit it accordingly.

We now move on to the scenario where we use two of the models from the previous experiments, and defer to the remaining, third model to see if this leads to an additional improvement in performance. We present our findings in Table 4.3, where we also include the *best* of the combined performance of models 1 and 2 from the previous results as a strong baseline. The results confirm our previous findings: the deferral to a third model improves the performance of the composite system above and beyond the previously achieved performance.

Table 4.1. Individual model accuracies in tune and test sets in ImageNet-16H dataset.

	AlexNet	DenseNet161	ResNet152	GoogleNet	VGG-19
Tune set	0.276	0.480	0.438	0.448	0.346
Test set	0.279	0.474	0.428	0.429	0.351

Table 4.2. Deferral from one machine learning model to an expert using ImageNet-16H dataset. Presenting model accuracy, expert accuracy, and composite accuracy on test set respectively.

Model	Expert	Mod. acc.	Exp. acc.	Comp. acc.
ResNet152	GoogleNet	0.428	0.429	0.448
ResNet152	DenseNet161	0.428	0.474	0.483
GoogleNet	ResNet152	0.429	0.428	0.476
GoogleNet	DenseNet161	0.429	0.474	0.495
DenseNet161	GoogleNet	0.474	0.429	0.493
DenseNet161	ResNet152	0.474	0.428	0.487

Table 4.3. Deferral from two models to an expert using ImageNet-16H dataset.

Model 1	Model 2	Expert	Baseline	Comp. acc.
ResNet152	GoogleNet	DenseNet161	0.476	0.495
DenseNet161	ResNet152	GoogleNet	0.487	0.507
GoogleNet	DenseNet161	ResNet152	0.495	0.517

Lastly, to test our method with all available models, we take the final scenario we mentioned and use GoogleNet and DenseNet161 as ML models, and as experts include not only ResNet152 but also the worst performing models, VGG19 and AlexNet. We find the composite performance to be 0.504. Although this is better than the performances of all individual algorithms, it is lower than the scenario where GoogleNet and DenseNet161 deferred only to ResNet152, meaning that addition of low quality experts can lower composite performance, both through the incorrect responses of the experts and through making the density estimation task more difficult.

For completeness, we repeat the experiments above by using sequential Monte Carlo (SMC) estimator for model selection and inference [33] instead of MFVI. We use the marginal likelihood estimate produced by SMC for model selection, and a sample from among the resulting particles according to particle weight is used for inference. The results are similar to those obtained with MFVI, which we present in Appendix C.4.

Overall, we find that the utilization of nonnegative tensor factorization through BAM is uniquely advantageous in extending our approach to arbitrary LTD scenarios. In Chapter 6, we will see yet another application for nonnegative factorization and BAM in supervised learning, as we use it to analyze the latent representations of deep learning models.

5. COMPRESSIBILITY AND GENERALIZATION IN DEEP LEARNING

Having examined how nonnegative tensor factorization (NTF) and Bayesian allocation model (BAM) can be used in modeling time series data with multiple seasonalities and in learning to defer under complex scenarios, we now move on to yet another research setting in supervised machine learning, where we find BAM to be again useful. This chapter and the following Chapter 6 are complementary, similar to Chapters 3 and 4. The current chapter presents our results regarding how and when compressibility arises in overparametrized SGD-trained neural networks, and how it relates to generalization. In the next chapter, we analyze how the structure of the latent activations relate to these results, where the expedient model selection afforded by BAM allow us to extend this investigation in previously unexplored directions.

We start with our results on the relationship between training hyperparameters, parameter statistics, compressibility, and generalization. The contents of this chapter and the associated Appendix D are mostly from the article [55], with some additional content from [56]. For results in which the author of this thesis was not a primary contributor, we explicitly refer the reader to [55] for full details, after a brief overview thereof for completeness.

5.1. Motivation and Background

With the increasing model sizes in deep learning and with its increasing use in low-resource environments, network compression is becoming ever more important. Among many network compression techniques, network pruning has been arguably the most commonly used method [135], and it is rising in popularity and success [136]. Though various pruning methods are successfully used in practice and their theoretical implications in terms of generalization are increasingly apparent [137], a thorough understanding of *why* and *when* neural networks become compressible is lacking.

A common conclusion in pruning research is that overparametrized networks can be greatly compressed by pruning with little cost to generalization, including with simple schemes such as *magnitude pruning* [135,136]. For example, research on *iterative magnitude pruning* [138] demonstrated the possibility of compressing trained deep learning models by iteratively eliciting ever sparser substructures. While it is known that the choices of training hyperparameters such as *learning rate* (also called *step size*) affect the performance of such pruning strategies [139–141], such observations are usually low in granularity and are almost never theoretically motivated. Overall, the field lacks a framework to understand why or when a pruning method should be useful [135].

Another strand of research that highlights the importance of understanding network compressibility includes various studies [137], [142–145] that present generalization bounds and/or empirical evidence implying that the more compressible a network is, the more likely it is to generalize well. These results are particularly interesting since classical generalization bounds increase with parameter dimensionality and hence become irrelevant in high dimensional deep learning settings, and fall short of explaining the generalization behavior of overparametrized neural networks. These results again illustrate the importance of understanding the conditions that give rise to compressibility, given their implications regarding generalization.

In this chapter, we investigate why and when modern neural networks can be amenable to very simple pruning strategies, and how this relates to generalization. Our results are based on two recent distinct discoveries regarding deep neural networks trained with the *stochastic gradient descent (SGD)* algorithm. The first one is the emergence of *heavy-tailed* stationary distributions, which appear when the networks are trained with large learning rates and/or small batch sizes [146, 147]. The second one is the *propagation of chaos* phenomenon, which indicates that, as the network size goes to infinity, the network weights behave independently [148–150].

We show that, assuming both of the aforementioned phenomena occur *simultaneously*, fully connected neural networks will be provably compressible in a precise sense, and the compression errors of (i) unstructured global or layer-wise magnitude pruning, (ii) pruning based on singular values of the weight matrices, (iii) and node pruning can be made arbitrarily small for any compression ratio as the dimension increases [55]. Our formulation of network compressibility in terms of "ℓ_p-compressibility" enables us to access results from *compressed sensing* literature [151, 152] to be used in neural network analysis. Moreover, we prove generalization bounds adapted to our framework that agree with existing compression-based generalization bounds [137], [142, 143] and confirm that compressibility implies better generalization. We conduct experiments on fully connected and convolutional networks and show that the results are in strong accordance with our theory.

Our findings reveal an interesting phenomenon: depending on the training hyperparameters such as learning rate and batch size, the resulting neural networks might possess different compressibility properties. Under the decoupling effect of propagation of chaos that emerges with overparametrization [150], the networks become compressible in a way that they are amenable to simple pruning strategies as the tails get heavier, which is shown to depend on the learning rate/batch size ratio [147]. Finally, when compressible, the networks become likely to generalize better. In this sense, our results also provide an alternative perspective to the recent theoretical studies that suggest that heavy tails can be beneficial in terms of generalization [153, 154].

5.2. Technical Preliminaries

5.2.1. Notation

For convenience, in this chapter and the next, our notation slightly diverges from those of the previous chapters. More specifically, we allow lower case letters to denote integers, use the alternative indexing notation $\mathbf{x}[i]$ to denote the i'th element of the vector $\mathbf{x}$, and let a sequence of n scalars $x_1, \ldots, x_n$ be shown as $\{x_i\}_{i=1}^n$. Similar

notations are used for sequences of matrices $\{\mathbf{X}_i\}_{i=1}^{l}$ and vectors $\{\mathbf{x}_i\}_{i=1}^{l}$, whose entries are indexed with convention $\mathbf{X}_i = [\mathbf{X}_i]_{m,n}$ and $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,n})$, respectively. We denote the ℓ_p (semi-)norm of a vector $\mathbf{x} \in \mathbb{R}^d$ as $\|\mathbf{x}\|_p = (\sum_{i=1}^{d} |x_i|^p)^{1/p}$ for all $p \in (0, \infty)$ and $\|\mathbf{x}\|$ implies $\|\mathbf{x}\|_2$. For a matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$, $\|\mathbf{A}\|$ denotes its Frobenius norm.

5.2.2. Fully Connected Neural Networks

In the entirety of the chapter, we consider a multi-class classification setting. We denote the space of data points by $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$, where $\mathcal{X} \subset \mathbb{R}^{d_X}$ is the space of features and $\mathcal{Y} = [d_Y]$ is the space of the labels. Similar to prior art (e.g., [155]), we focus our attention on the bounded feature domain $\mathcal{X} = \mathcal{X}_B := \left\{\mathbf{x} \in \mathbb{R}^{d_X}; \|\mathbf{x}\| \leq B\right\}$.

We denote a *fully connected neural network* with L layers by a collection of *weight matrices* $\{\mathbf{W}_l\}_{l=1}^{L}$, such that $\mathbf{W}_l \in \mathbb{R}^{h_l \times h_{l-1}}$, where h_l denotes the number of hidden units at layer l with $h_0 = d_X$ and $h_L = d_Y$. Accordingly, the *prediction function* $f_{\mathbf{w}}(\mathbf{x}) : \mathcal{X}_B \mapsto \mathbb{R}^{d_Y}$, with elements $f_{\mathbf{w}}(\mathbf{x}) \coloneqq (f_{\mathbf{w}}(\mathbf{x})[1], \ldots, f_{\mathbf{w}}(\mathbf{x})[d_Y])$, corresponding to the neural network is defined as

$$f_{\mathbf{w}}(\mathbf{x}) = \mathbf{W}_L \phi\left(\mathbf{W}_{L-1} \phi\left(\cdots \phi\left(\mathbf{W}_1 \mathbf{x}\right)\right)\right), \tag{5.1}$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is the *rectified linear unit (ReLU)* activation function, i.e., $\phi(x) = \max(0, x)$, and it is applied element-wise when its input is a vector. For notational convenience, we define $d_l \coloneqq h_l \times h_{l-1}$ and $d \coloneqq \sum_{l=1}^{L} d_l$. Furthermore, let $\mathbf{w}_l$ denote the *vectorized* weight matrix of layer l, i.e., $\mathbf{w}_l \coloneqq \mathbf{vec}\left(\mathbf{W}_l\right) \in \mathbb{R}^{d_l}$, where $\mathbf{vec}$ denotes vectorization. Finally, let $\mathbf{w}$ be the concatenation of all the vectorized weight matrices, i.e., $\mathbf{w} \coloneqq (\mathbf{w}_1, \ldots, \mathbf{w}_L) \in \mathbb{R}^d$. We assume that $L \geq 2$.

5.2.3. Risk Minimization and SGD

In order to assess the quality of a neural network represented by its weights $\mathbf{w}$, we consider a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}_{>0}$, such that $\ell(y, f_{\mathbf{w}}(\mathbf{x}))$ measures the loss incurred by predicting the label of $\mathbf{x}$ as $\arg\max_j f_{\mathbf{w}}(\mathbf{x})[j]$, when the true label is y. By following a standard statistical learning theoretical setup, we consider an unknown *data*

distribution μ_Z over $\mathcal{Z}$, and a *training dataset* with n elements, i.e., $S = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$, where each $\mathbf{z}_i := (\mathbf{x}_i, y_i) \overset{\text{i.i.d.}}{\sim} \mu_Z$. We then denote the *population* and *empirical risks* as $\mathcal{R}(\mathbf{w}) \coloneqq \mathbb{E}_{\mathbf{x},y}\left[\ell(y, f_{\mathbf{w}}(\mathbf{x}))\right]$ and $\widehat{\mathcal{R}}(\mathbf{w}) \coloneqq \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f_{\mathbf{w}}(\mathbf{x}_i))$.

Since μ_Z is unknown, we cannot directly attempt to minimize $\mathcal{R}$ in practice. One popular approach to address this problem is the *empirical risk minimization* strategy, where the goal is to solve the following optimization problem: $\min_{\mathbf{w}\in\mathbb{R}^{\mathrm{d}}} \widehat{\mathcal{R}}(\mathbf{w})$. To tackle this problem, SGD has been one of the most popular optimization algorithms, which is based on the recursion

$$\mathbf{w}^{\{k+1\}} = \mathbf{w}^{\{k\}} - \eta \nabla \widetilde{\mathcal{R}}_{k+1}(\mathbf{w}^{\{k\}}), \quad \text{where}$$
$$\nabla \widetilde{\mathcal{R}}_k(\mathbf{w}) \coloneqq (1/b) \sum\nolimits_{i\in\Omega_k} \nabla \ell\left(y_i, f_{\mathbf{w}}(\mathbf{x}_i)\right). \tag{5.2}$$

Here, $\mathbf{w}^{\{k\}}$ denotes the weight vector at iteration $k \in \mathbb{N}^+$, $\eta > 0$ is the learning rate (or step size), $\nabla\widetilde{\mathcal{R}}_k(\cdot)$ is the stochastic gradient, and Ω_k is the mini-batch with size b, drawn with or without replacement from $[n]$.

5.2.4. Heavy-Tailed Distributions and the α-stable Family

In this study, we mainly deal with heavy-tailed random variables. While there exist different definitions of heavy tails in the literature, here, we call a random variable heavy-tailed if its distribution function has a *power-law* decay, i.e., $p(\mathrm{x} \geq x) \sim cx^{-\alpha}$ as $x \to \infty$, for some $c > 0$ and $\alpha \in (0, 2)$. Here, the *tail index* α determines the tail thickness of the distribution, i.e., as α gets smaller, the distribution becomes heavier-tailed.

An important subclass of heavy-tailed distributions is the family of *stable distributions*. A random variable x has symmetric α-stable distribution, denoted by $\mathcal{S}\alpha\mathcal{S}(\sigma)$, if its characteristic function is equal to $\mathbb{E}\left[\exp(iw\mathrm{x})\right] = \exp\left(-|\sigma w|^{\alpha}\right)$, where $\alpha \in (0, 2]$ is again the tail index and $\sigma \in (0, \infty)$ is the scale parameter. An important property of $\mathcal{S}\alpha\mathcal{S}$ is that whenever $\alpha < 2$, $\mathbb{E}|\mathrm{x}|^p$ is finite if and only if $p < \alpha$. This implies that the distribution has infinite variance as soon as $\alpha < 2$. In addition to their wide use

in applied fields [156], recently, $\mathcal{S}\alpha\mathcal{S}$ distributions have also been considered in deep learning theory [153, 154], [157, 158] and optimization [159].

5.3. Compressibility and the Heavy-Tailed Mean-Field Regime

In this section, we present our compression schemes. We first identify a sufficient condition, under which we show that the compression errors of different pruning techniques become arbitrarily small as network size increases [55].

5.3.1. The Heavy-Tailed Mean-Field Regime

Due to the peculiar generalization behavior of neural networks trained with SGD, recent years have witnessed an extensive investigation of the theoretical properties of SGD in deep learning [153], [160–167]. We now mention two recently established such properties, and we study the compressibility of networks under the assumption that these two seemingly unrelated phenomena occur simultaneously.

5.3.1.1. The Heavy Tail Phenomenon. Several recent studies [154], [157], [168–172] empirically illustrate that neural networks can exhibit heavy-tailed behavior when optimized with SGD. Theoretically investigating the origins of this heavy-tailed behavior, [146] and [147] later prove several theoretical results on online SGD, with rather surprising implications: due to the "multiplicative" structure of the gradient noise (i.e., $\nabla\widetilde{\mathcal{R}}_k(\mathbf{w}) - \nabla\widehat{\mathcal{R}}(\mathbf{w})$), as $k \to \infty$, the distribution of the SGD iterates $\mathbf{w}^{\{k\}}$ can converge to a heavy-tailed distribution with infinite second-order moment, and perhaps more surprisingly, this behavior can even emerge in simple linear regression with Gaussian data. The authors of [147] further show that, in the linear regression setting, the tail index α is *monotonic* with respect to the step size η and batch size b: larger η and/or smaller b result in smaller α (i.e., heavier tails).

5.3.1.2. Propagation of Chaos and Mean-Field Limits. Another interesting property of SGD appears when network size goes to infinity. Recent research shows that under

an appropriate scaling of step sizes, the empirical distribution of the network weights converges to a fixed mean-field distribution, and the SGD dynamics can be represented as a *gradient flow* in the space of probability measures [148–150], [173]. Moreover, [150] show that a *propagation of chaos* phenomenon occurs in this setting, which indicates that when the network weights are initialized independently (*i.e.,* a-priori chaotic behavior), the weights stay independent as the algorithm evolves over time (i.e., the chaos propagates) [174].

Our focus in this chapter is the setting where both of these phenomena occur simultaneously: we assume that as the size of the network goes to infinity, the SGD iterates become *independent* and their distribution converges to a *heavy-tailed* distribution. To formalize this setting, we introduce some additional notation. For an integer $m_l < d_l$, we let $\mathbf{w}_{l,(1:m_l)}$ denote the first m_l coordinates of $\mathbf{w}_l$, and for $m_l \geq d_l$ set $\mathbf{w}_{l,(1:m_l)} = \mathbf{w}_l$. Furthermore, we parametrize the dimension of each layer d_l with a parameter $\rho \in \mathbb{R}$, i.e., $d_l = d_l(\rho)$, such that for all $l = 1, \ldots, L$, we have

$$d_l(\rho) \in \mathbb{Z}_{>0}, \quad \text{and} \quad \lim_{\rho \to \infty} d_l(\rho) = \infty. \tag{5.3}$$

This construction enables us to take the dimensions of each layer to infinity simultaneously [175].

Condition 1 (Heavy-tailed mean-field limit condition)**.** *The SGD recursion* (5.2) *satisfies the heavy-tailed mean-field limit (HML) condition, if the dimensions obey* (5.3) *and if there exist heavy-tailed probability measures* $\{\mu_l^\star\}_{l=1}^L$ *on* $\mathbb{R}$ *with tail indices* $\{\alpha_l\}_{l=1}^L$*, such that* $\mu_l^\star(\{0\}) = 0$ *for all* l *and for any* $m_1, \ldots, m_L \in \mathbb{Z}_{>0}$*, the joint distribution of*

$$\left(\mathbf{w}_{1,(1:m_1)}^{\{k\}}, \ldots, \mathbf{w}_{L,(1:m_L)}^{\{k\}}\right) \quad \text{weakly converges to} \quad (\mu_1^\star)^{\otimes m_1} \otimes \cdots \otimes (\mu_L^\star)^{\otimes m_L}, \tag{5.4}$$

as $\rho, k \to \infty$*, where* $\mu \otimes \nu$ *denotes the product measure and* $\mu^{\otimes n}$ *denotes the* n*-fold product measure.*

Informally, the HML condition states that in the infinite size and infinite iteration limit, the entries of the weight matrices will become independent, and the distribution of the elements within the same layer will be identical and heavy-tailed. We acknowledge that the above condition may not be always satisfied, and theoretical

evidence for this condition is provided in rather seemingly simpler settings such as linear regression [147] and/or two-layer neural networks [150]. However, our results in Section 5.5.2.1 and Appendix D.3 imply that this might not necessarily be an unrealistic assumption. Note that, in (5.4), the particular form of independence in the limit is not crucial and weaker alternatives are discussed in [55].

We further note that [150] proved that a similar form of (5.4) indeed holds for SGD applied on single hidden-layered neural networks, where the limiting distributions possess second-order moments (i.e., not heavy-tailed) and the independence is column-wise. Recently, [158] investigated the infinite width limits of fully connected networks initialized from a $\mathcal{S}\alpha\mathcal{S}$ distribution and proved heavy-tailed limiting distributions. On the other hand, heavy-tailed propagation of chaos results have been proven in theoretical probability [176,177]; however, their connection to SGD has not been yet established. We believe that (5.4) can be shown to hold under appropriate conditions, which we leave as future work.

5.3.2. Compression Algorithms

In this section, we analyze the three different compression schemes utilized in our study. All three methods are based on *pruning*, which we formally define as follows. Let $\mathbf{x}$ be a vector of length d, and consider its sorted version in descending order with respect to the magnitude of its entries, i.e., $|x_{i_1}| \geq |x_{i_2}| \geq \cdots \geq |x_{i_d}|$, where $\{i_1, \ldots, i_d\} = \{1, \ldots, d\}$. For any $k \leq d$, the *k-best term approximation* of $\mathbf{x}$, denoted as $\mathbf{x}^{(k)} = (x_1^{(k)}, \ldots, x_d^{(k)})$, is defined as follows: for $l \in [\lceil k \rceil]$, $x_{i_l}^{(k)} := x_{i_l}$ and for $l \notin [\lceil k \rceil]$, $x_{i_l}^{(k)} := 0$. Informally, we keep the k-largest entries of $\mathbf{x}$ with the largest magnitudes, and "prune" the remaining ones. Our current results pertain to one-shot pruning with no fine-tuning; other settings are left for future work (cf. [178]).

In this section, we consider that we have access to a sample from the stationary distribution of the SGD, i.e. $\mathbf{w}^{\{\infty\}}$, and for conciseness we simply denote it by $\mathbf{w}$. We then consider a *compressed network* $\widehat{\mathbf{w}}$ (that can be obtained by different compression

schemes) and measure the performance of the compression scheme by its *relative* ℓ_p-*compression error* (cf. [151, 152]), defined as: $\|\widehat{\mathbf{w}} - \mathbf{w}\|_p / \|\mathbf{w}\|_p$.

5.3.2.1. Magnitude Pruning. Magnitude pruning has been one of the most common and efficient algorithms among all the network pruning strategies [136], [145], [179]. In this section, we consider the global and layer-wise magnitude pruning strategies under the HML condition.

More precisely, given a network weight vector $\mathbf{w} \in \mathbb{R}^d$ and a *remaining parameter ratio* $\kappa \in (0, 1)$, the global pruning strategy compresses $\mathbf{w}$ by using $\mathbf{w}^{(\kappa d)}$, i.e., it prunes the smallest (in magnitude) $(1-\kappa)d$ entries of $\mathbf{w}$. We refer to $(1-\kappa)$ as *pruning ratio*. Also, note that $1/\kappa$ corresponds to the frequently used metric *compression rate* [136]. On the other hand, the layer-wise pruning strategy applies the same approach to each layer separately, i.e., given layer-wise remaining parameter ratios $\kappa_l \in (0, 1)$, we compress each layer weight $\mathbf{w}_l \in \mathbb{R}^{d_l}$ by using $\mathbf{w}_l^{(\kappa_l d_l)}$. Theorem 1 in [55] shows that any pruning ratio of the weights is achievable as long as the network size is sufficiently large and the network weights are close enough to an i.i.d. heavy-tailed distribution. Empirical studies report that global magnitude pruning often works better than layer-wise magnitude pruning [136], except when it leads to over-aggressive pruning of particular layers [180].

The success of this strategy under the HML condition is due to a result from compressed sensing theory, concurrently proven in [151, 152], which informally states that for a large vector of i.i.d. heavy-tailed random variables, the norm of the vector is mainly determined by a small fraction of its entries. An important question here is that, to achieve a fixed relative error ε_l, how would the smallest κ_l differ with varying tail-indices α_l, i.e., whether "heavier tails imply more compression". In our experiments, we illustrate this behavior positively: heavier-tailed weights are indeed more compressible. We leave full theoretical justification for this behavior as future work.

5.3.2.2. Singular Value Pruning. In recent studies, it has been illustrated that the magnitudes of the eigenvalues of the sample covariance matrices (for different layers) can decay quickly, hence pruning the singular values of the weight matrices, i.e., only keeping the largest singular values and corresponding singular vectors, is a sensible pruning method. Exploiting the *low-rank* nature of fully connected and convolutional layer weights in network compression has been investigated theoretically and empirically [137], [181]. Here we present a simple scheme to demonstrate our results.

More precisely, for the weight matrix at layer l, $\mathbf{W}_l$, consider its singular value decomposition, $\mathbf{W}_l = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^\top$, and then, with a slight abuse of notation, define $\mathbf{W}_l^{[\kappa_l h_{l-1}]} := \mathbf{U}\boldsymbol{\Sigma}^{(\kappa_l h_{l-1})}\mathbf{V}^\top$, where $\boldsymbol{\Sigma}^{(\kappa_l h_{l-1})}$ is the diagonal matrix whose diagonal entries contain the $\lceil \kappa_l h_{l-1} \rceil$ largest singular values (i.e., prune the diagonal of $\boldsymbol{\Sigma}$). Accordingly, denote $\mathbf{w}_l^{[\kappa_l h_{l-1}]} := \mathbf{vec}(\mathbf{W}_l^{[\kappa_l h_{l-1}]})$. Theorem 2 in [55] shows that under the HML condition with an additional requirement that the limiting distributions are $\mathcal{S}\alpha\mathcal{S}$, the eigenvalues of the (properly normalized) sample covariance matrices will be indeed compressible and the pruning strategy achieves negligible errors as the network size grows.

5.3.2.3. Node Pruning. The last pruning strategy that we consider is a structured pruning strategy, that corresponds to the removal of the whole columns of a fully connected layer weight matrix. Even though below we consider pruning based on the norms of the weight layer columns, the same arguments apply for pruning rows; see [55] for further discussion.

The idea in column pruning is that, for a given layer l, we first sort the columns of the weight matrix $\mathbf{W}_l \in \mathbb{R}^{h_l \times h_{l-1}}$ with respect to their ℓ_p-norms for a given $p \geq \max_l \alpha_l$. Then, we remove the *entire columns* that have the smallest ℓ_p-norms. More precisely, let $\mathbf{W}_l(i) \in \mathbb{R}^{h_l}$ be the i-th column of $\mathbf{W}_l$, for $i \in [h_{l-1}]$, and suppose that $\|\mathbf{W}_l(i_1)\|_p \geq \|\mathbf{W}_l(i_2)\|_p \geq \cdots \geq \|\mathbf{W}_l(i_{h_{l-1}})\|_p$, where $\{i_1, \ldots, i_{h_{l-1}}\} = \{1, \ldots, h_{l-1}\}$. Then, we define the *k-best column approximation* of $\mathbf{W}_l$, denoted as $\mathbf{W}_l^{\{k\},p} \in \mathbb{R}^{h_l \times h_{l-1}}$, as follows: for $j \in [\lceil k \rceil]$, $\mathbf{W}_l^{\{k\},p}(i_j) := \mathbf{W}_l(i_j)$ and for $j \notin [\lceil k \rceil]$, $\mathbf{W}_l^{\{k\},p}(i_j) := 0$. Denote

also $\mathbf{w}_l^{\{k\},p} \coloneqq \mathbf{vec}\left(\mathbf{W}_l^{\{k\},p}\right)$.

Theorem 3 in [55] indicates that we can remove entire columns in each layer, without considerably affecting the network weights, as long as the network is large enough. In other words, effectively the widths of the layers can be reduced. Structured pruning schemes are commonly used in CNNs where filters, channels, or kernels can be pruned by norm-based or other criteria [179], [182, 183]. In [55] additional results demonstrate that a small compression error also implies a small perturbation on the network output.

5.4. Generalization Bounds

So far, we have shown that the heavy-tailed behavior in the weights of a network together with their independence result in compressibility. In [55], we further show that these phenomena bring forth a better generalization performance bound in the network. More precisely, we establish a generalization bound such that if a network is more compressible, then the corresponding compressed network has a smaller generalization error bound. The said results constitute a counterpart to the results by [137], [142–144]; yet, they cannot be directly used in our specific setting, hence, we prove results that are customized to our setup. More precisely, although these works, similar to our result, are based on the assumption of the "compressibility" of the network and moreover the generalization gap of the population risk with respect to the empirical margin risk is considered in [137], [144], they cannot be applied directly to the ℓ_p-compressibility based strategies discussed in the previous section. We refer the reader to [55] for this aspect of our work, and proceed to present experimental evidence for these and previous theoretical findings we presented.

5.5. Experiments

In this section, we present experiments conducted with neural networks to investigate our theory. We first describe our experimental methodology, followed by our

results. The source code for the experiments can be seen in [184].

5.5.1. Experimental Settings

We use three different model architectures: a fully connected network with 4 hidden layers (FCN4), a fully connected network with 6 hidden layers (FCN6), and a convolutional neural network with 8 convolutional layers (CNN). Hidden layer widths were 2048 for both FCN models. All networks include ReLU activation functions and none include batch normalization, dropout, residual blocks, or any explicit regularization term in the loss function. Each model is trained on MNIST [185] and CIFAR-10 [186] datasets under various hyperparameter settings, using the default splits for training and evaluation. The total number of parameters were approximately $14M$ for FCN4-MNIST, $19M$ for FCN4-CIFAR-10, $23M$ for FCN6-MNIST, $27M$ for FCN6-CIFAR-10, and $9M$ for both CNN models. All models were trained with SGD until convergence with constant learning rates and no momentum. The convergence criteria comprised 100% training accuracy and a training negative log-likelihood less than 5×10^{-5}. The training hyperparameter settings include two batch sizes ($b = 50, 100$) and various learning rates (η) to generate a large range of η/b values. We provide further details in the sections below.

5.5.1.1. Tail Index Estimation. By invoking [147, Corollary 11] which shows the ergodic average of heavy-tailed SGD iterates converges to a multivariate stable distribution, after convergence, we kept running SGD for 1000 additional iterations to obtain the average of the parameters to be used in this estimation. The tail index estimations were made by the estimator proposed in [187, Corollary 2.4], which has been used by other recent studies that estimate tail index in neural network parameters [147], [153,154]. The ergodic averaging does not change the tail index of the parameters, and was employed to facilitate tail index estimation by enabling the utilization of the aforementioned estimator. The remaining experiments have been conducted without ergodic averaging. We also observe that the results with/without ergodic averaging are virtually identical in both tail index estimation tasks and pruning experiments. In

all pruning methods, the parameters were centered before pruning is conducted with the median value to conform with the tail index estimation. Further details on experimental settings can be found in Appendix D.1.

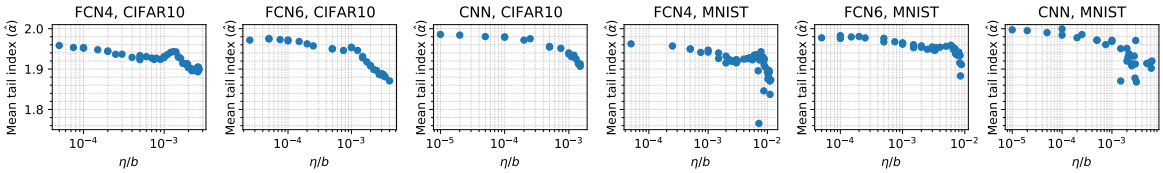


Figure 5.1. Mean estimated tail index ($\hat{\alpha}$) and η/b for each trained model. x-axes are log-scaled [55].

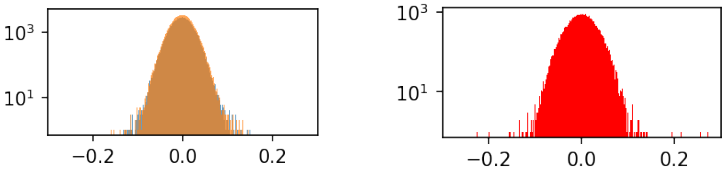
5.5.2. Results

We now present our results, first starting with examining layer statistics resulting from different choices of hyperparameters.

5.5.2.1. Training Hyperparameters and Layer Statistics. We first verify that models trained with higher learning rate to batch size ratio (η/b) lead to heavier-tailed parameters, replicating the results presented in [147]. For each trained model, we compute the mean of separately estimated tail indices for all layers, so that each model is represented by a single value ($\hat{\alpha}$). This averaging is a heuristic without a clear theoretical meaning, and has been also used by recent works in the literature [147], [171]. Results presented in Figure 5.1 demonstrates that higher η/b leads to heavier tails (lower mean estimated tail index, $\hat{\alpha}$).

Also of interest are the distribution of the resulting parameters from training. Figure 5.2(a) demonstrates a representative example of the parameters from a CNN layer trained on MNIST, where two overlaid histograms representing empirical distributions of a random partition of the parameters are almost identical. In Figure 5.2(b), this distribution is compared to a simulated symmetric α-stable distribution that has the same tail index (≈ 1.95). The figure demonstrates that the two distributions have

similar qualitative properties.



(a) (b)

Figure 5.2. Empirical distribution of a CNN layer trained on MNIST. (a) Overlaid histograms of a random partition of the weights, showing an identical distribution. (b) Comparing the network weights to samples simulated i.i.d. from a symmetric α-stable distribution with the same tail index $\alpha \approx 1.95$. y-axes are log-scaled [55].

5.5.2.2. Tail Index and Prunability. In this section we examine whether networks with heavier-tailed layers, trained with higher η/b ratios, are more prunable. As a baseline test, we first examine whether neural network layers which are heavier-tailed can be pruned more given a fixed maximum relative compression error. Figure 5.3 demonstrates for an error of 0.1 that this is indeed the case. We next test our hypothesis that posit models with heavy-tailed parameters to be more prunable. Both the results pertaining to layer-wise magnitude pruning and singular value pruning, demonstrated in Figures 5.4 and 5.5 show that this is indeed the case. Here, relative test accuracy stands for (test accuracy after pruning) / (original test accuracy). The results show that models with heavier-tailed parameters (shown with darker colors) are starkly more robust against pruning. Similar results with global magnitude pruning are in Appendix D.4.

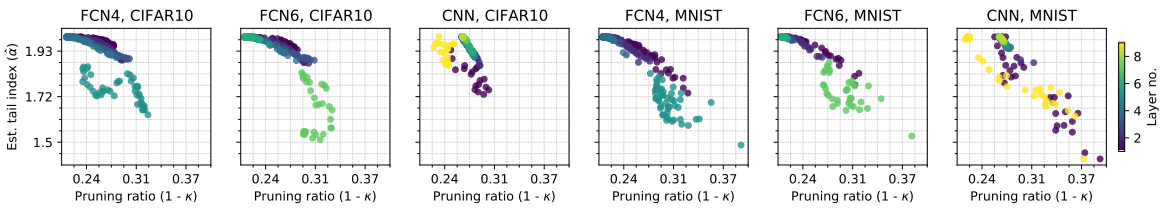


Figure 5.3. Mean estimated tail index ($\hat{\alpha}$) and pruning ratio ($1 - \kappa$), relative compression error $= 0.1$, $p = 2$ [55].

For node pruning/structured pruning, with results presented in Appendix D.4, we prune 3×3 kernel parameters in CNN models, and the results show a similar, hypothesis conforming pattern. Results for FCNs were underwhelming; perhaps unsurprisingly as structured pruning is not as commonly used in FCNs [135]. More successful attempts could be due to alternative scoring methods [142]; our approach might require wider layers to conform with theoretical conditions.

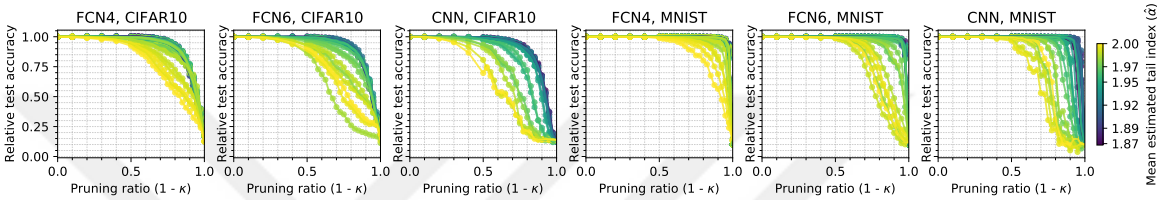


Figure 5.4. Relative test accuracy and pruning ratio $(1-\kappa)$ for layer-wise magnitude pruning. Colors denote mean estimated tail index $(\hat{\alpha})$ [55].

5.5.2.3. Tail Index and Generalization. Following our theoretical results, we examine whether the heavier-tailed networks lead to better generalization performance. For these results, we also include the additional CIFAR-100 [186] experiments conducted in [56]. Consistent with our hypothesis, Figure 5.6 shows that models with the highest tail index have consistently the worst test performances. We also find that the relationship between tail index and generalization also holds for a more difficult dataset in CIFAR-100. The conclusion regarding generalization gap also applies to generalization performance as training accuracy is 100% for all models trained on MNIST and CIFAR-10, and 99% for CIFAR-100. See Appendix D for further details on experiment settings, and additional experiment results and discussion.

5.6. Compressibility and Adversarial Robustness

As important as generalization is, another crucial demand from modern machine learning systems is robustness against unexpected input. Performance under such changes (i.e. *distribution shifts*) is usually referred to as *out-of-distribution (OOD)*

generalization performance. While distribution shifts can occur naturally for deployed machine learning systems [188, 189], an especially interesting distribution shift is presented by the so-called *adversarial examples* [190]. Such examples show that although neural networks might perform well in out-of-sample instances in the distribution they were trained, their performance might be drastically reduced by specifically developed inputs that deviate from the original input, even if these deviations are imperceptible to human eye. This initiated numerous research around the concepts of *adversarial robustness* and *robustness-accuracy trade-off* [191, 192], and determining when and where this trade-off occurs is still a vibrant research topic. For these reasons, we find it important to at least provide a preliminary analysis of the adversarial robustness of SGD-trained overparametrized neural networks, in relation to learning rate and parameter statistics.

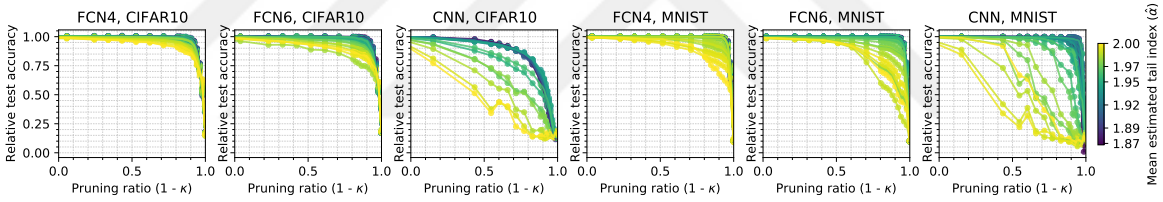


Figure 5.5. Relative test accuracy and pruning ratio $(1 - \kappa)$ for singular value pruning. Colors denote mean estimated tail index $(\hat{\alpha})$ [55].

5.6.1. Robust Risk and Out-of-Distribution Generalization

Remembering the respective definitions of (population) risk and empirical risk; $\mathcal{R}(\mathbf{w}) \coloneqq \mathbb{E}_{\mathbf{x},y}\left[\ell(y, f_{\mathbf{w}}(\mathbf{x}))\right]$ and $\widehat{\mathcal{R}}(\mathbf{w}) \coloneqq \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, f_{\mathbf{w}}(\mathbf{x}_i))$, let us now define adversarial perturbation, adversarial robustness, and adversarial risk.

Definition 5.1 (Adversarial perturbation)**.** *Let* $\mathbb{B}_p(\mathbf{x}, \epsilon)$ *denote the* ϵ*-ball around* $\mathbf{x} \in \mathcal{X}$ *in the* ℓ_p*-sense. Let* $\mathbb{B}_\infty$ *be the* ℓ_∞*-ball. Any* $\mathbf{x}' \in \mathbb{B}_p(\mathbf{x}, \epsilon) \coloneqq \mathbf{x} + \delta$ *is known to be an adversarial input p-perturbed by* δ*.*

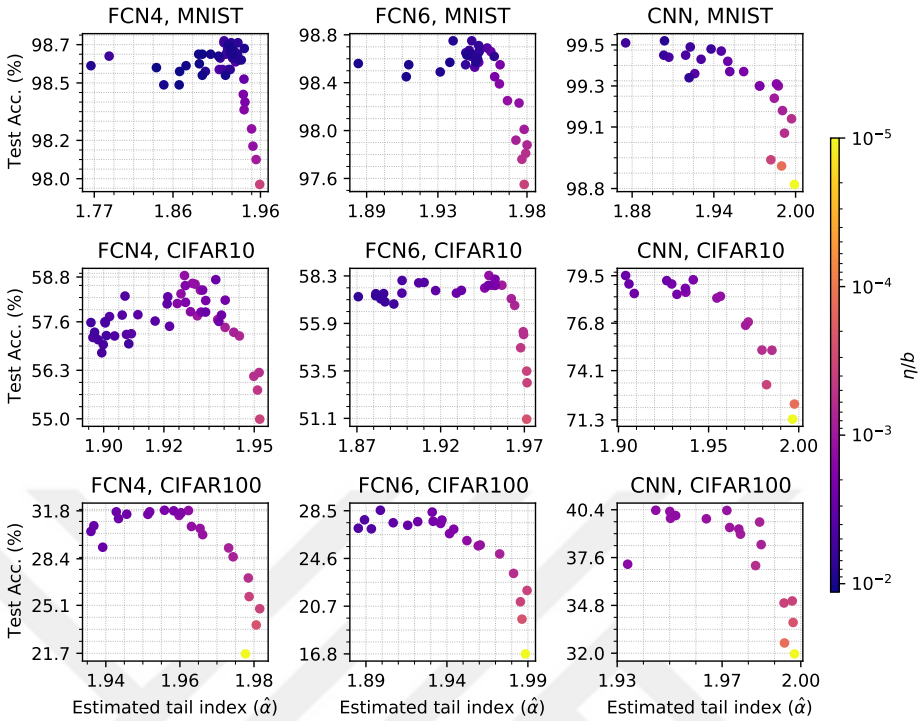


Figure 5.6. Test accuracy and mean estimated tail index ($\hat{\alpha}$) [56].

Using this definition, we can define adversarial robustness.

Definition 5.2 (Adversarial robustness)**.** *A classifier* $f_{\mathbf{w}}(\cdot)$ *with parameters* $\mathbf{w}$ *is said to be robust at* $\mathbf{x}$ *with radius* $\epsilon > 0$*, if* $f_{\mathbf{w}}(\mathbf{x}') = f_{\mathbf{w}}(\mathbf{x})\,, \forall \mathbf{x}' \in \mathbb{B}(\mathbf{x}, \epsilon)$.

The most commonly used adversarial perturbations set $p = 2$ or $p = \infty$. In the adversarial robustness literature, these settings are called ℓ_2 and ℓ_∞ *threat models* [192]. As opposed to standard accuracy, accuracy of a model under such threat models is called *robust accuracy* [191]. Accordingly, we can define robust risk as

$$\mathcal{R}_{\text{rob}}(w, S) := \mathbb{E}_{\mathbf{x},y}[\max_{\mathbf{x}' \in \mathbb{B}(\mathbf{x},\epsilon)} \ell(y, f_{\mathbf{w}}(\mathbf{x}'))]. \tag{5.5}$$

After having been introduced in [190], there have been numerous findings and involved discussion on why adversarial examples exist, as well as how their effects can be mitigated with changes to the training procedure [193]. Developing adversarial examples, i.e. *adversarial attacks*, requires some form of optimization in the input space. When this optimization is made with access to model parameters, the resulting adversarial attack is called a *white box attack*, and when it is made without access to model internals it is called a *black box attack*, with the former being the more rigorous test of adversarial robustness [192]. A review that summarizes these discussions from a geometric perspective can be found in [194].

Another test of robustness of a model in an image classification context is investigating the effects of perturbations to input images such as blurring and cropping on a classifier. A robust image classifier would be expected to perform reasonably well under such perturbations that could conceivably occur "in the wild" [188]. Such image corruptions can be used to both test the robustness of a model to such perturbations as well as being used in the training procedure as a form of data augmentation [189].

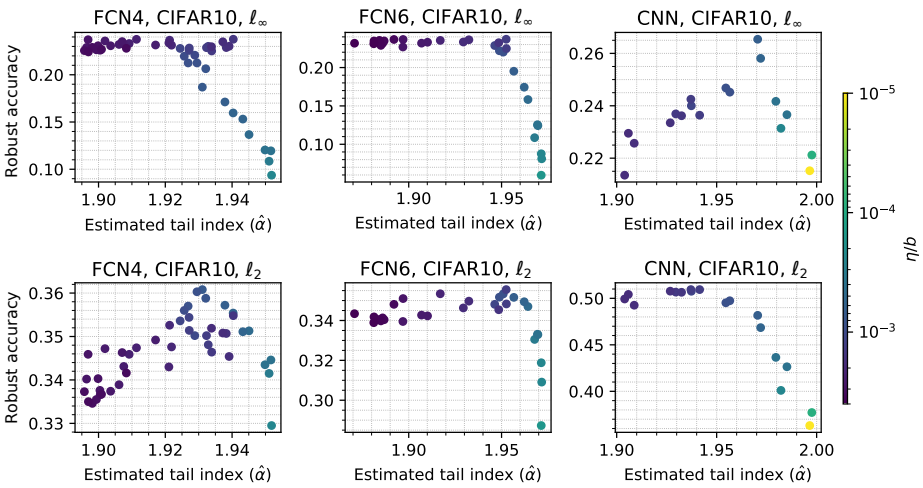


Figure 5.7. Robust accuracy against adversarial examples and mean estimated tail index ($\hat{\alpha}$) results with adversarial attacks on 10000 samples of CIFAR-10-C dataset.

5.6.2. Experiments on Compressibility and Adversarial Robustness

To test adversarial robustness, we use the framework provided by [192], RobustBench: a standardized benchmark for adversarial attacks; which includes three white box attacks and one black box attack. Moreover, it also provides interface for investigating a model's robustness against common image corruptions using the image corruption benchmark CIFAR-10-C [189].

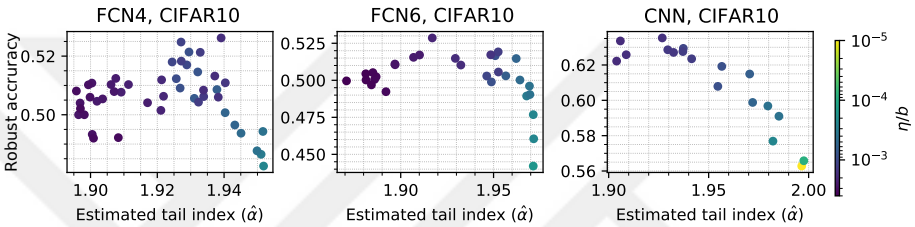


Figure 5.8. Robust accuracy against common image corruptions and mean estimated tail index ($\hat{\alpha}$) results with 10000 samples of CIFAR-10-C dataset.

The question we attempt to answer in this section is: Given the experimental setting described in Chapter 5, are more compressible and better generalizing models also more robust? We test the models we trained for the experiments in this chapter, and test robustness against adversarial attacks and image corruptions using RobustBench. Our results in Figures 5.7 and 5.8 show that networks trained with larger η/b are indeed also more robust against both ℓ_2 and ℓ_∞ adversarial perturbations, and common image corruptions. The levels of adversarial robustness demonstrated by our models are not necessarily on par with state of the art, which is to be expected since state of the art in this case would be models selected and trained explicitly against adversarial examples [193]. However, it is interesting to see that the downstream effects of training with large learning rates also extend to adversarial robustness. This is corroborated by some other results in the literature. For example, [195] theoretically and empirically show that sparse weights and sparse middle layer activations both separately lead to increased adversarial robustness in deep neural networks (unless they are *too* sparse). Further investigation of how parameter statistics, compressibility, generalization, and

robustness interrelate is a promising direction for future research. In the next chapter, we investigate this relationship from a different perspective, through the analysis of the latent representations.

6. LATENT REPRESENTATIONS AND ROBUST GENERALIZATION

In the previous chapter, we investigated how training hyperparameters, parameter statistics, compressibility, and generalization interrelate in overparametrized neural networks trained with stochastic gradient descent (SGD). Previous research focusing on this problem from a generalization perspective usually utilize a notion of *intrinsic dimensionality* of the neural network *parameters* [55], [137], [196–200]. In contrast, statistical properties of *latent activations* have been of less interest in this context. More specifically, although latent representations have been investigated extensively in previous research [201, 202], those that focus on how activation statistics relate to robust generalization have been less common. In this chapter, we combine the model selection capabilities of Bayesian allocation model (BAM) with other notions of rank, as well as the notion of compressibility from the previous chapter to provide an examination of representation space corollaries of robust generalization. Although this is a preliminary investigation, our results provide interesting and sometimes counterintuitive insights, and show the importance of understanding how training hyperparameters, parameter statistics, latent activations, and generalization interact for developing insights on modern deep learning systems.

6.1. Motivation and Background

In addition to being less common in general, research that examine the relationship between activation statistics and generalization tend to produce results that are more ambivalent compared to work previously discussed. For example, while [203] and later [204] find low-rank activations in deep neural networks to be advantageous, [205] find no consistent effect of representation-related regularization on either sparsity *or* rank. While [195] find that increased sparsity improves adversarial robustness, [206] show that neural network representations show a discriminability-transferability trade-off based on *over-compression*, where compressing the representations starts to hurt the

usefulness for those representations in related tasks, while still increasing in distribution generalization performance.

In this chapter, we examine how compressibility and rank structure of latent space activations relate to both in distribution and out-of-distribution (OOD) generalization. For our investigation, we utilize two different notions of compressibility and three different notions of rank. Importantly, we take advantage of the fact that in neural networks with ReLU activation functions, post-activation representations are nonnegative, and include *nonnegative rank* among our different approaches for characterizing activation rank. This is made possible by the expedient model selection allowed by BAM, and to our knowledge constitutes the first utilization of nonnegative rank in analyzing neural network activations. Before moving on to detailing our experimental settings, we summarize the findings of this chapter below.

- Activation compressibility is positively correlated with generalization: This is true for both the overall performance of the model (i.e. models with more compressible representations generalizing better), and for class-conditional performance within models (i.e. the classes that are more compressible generalizing better in a given model).
- Nonnegative rank and other frequently used notions of approximate rank provide qualitatively similar results: Ascertaining this allows us to opt for computational expedience whenever required.
- Activation rank is positively correlated with generalization: Models with lower activation ranks perform worse in generalization.
- Class-conditional activation rank is negatively correlated with generalization: Within a model, in classes with lower activation ranks the model performs better in generalization.
- Robustness-accuracy trade-off is not observed in the presence of compressibility: The results above replicate for OOD generalization, albeit with some caveats.

6.2. Technical Preliminaries

6.2.1. Compressibility Metrics

We start by defining different ways of characterizing activation compressibility. An obvious metric for this task is *sparsity*, which, given a d-dimensional vector $\mathbf{z} \in \mathbb{R}^d$ can be defined as

$$\text{Sparsity}(\mathbf{z}) = 1 - \frac{\|\mathbf{z}\|_0}{d}. \tag{6.1}$$

Although Sparsity as defined above is an intuitive and useful notion, more nuanced notions of compressibility are also desirable. This is because a (deterministic or probabilistic) vector might be "summarizable" with a small subset of its elements regardless of the number of 0 entries in the vector. Therefore, in addition to Sparsity, we utilize (q, κ)-Compressibility inspired by [151, Definition 1], which is closely related to the notion of relative ℓ_p compression error we used in Chapter 5. Formally, we can state that

$$(q, \kappa)\text{-Compressibility}(\mathbf{z}) = 1 - \frac{\inf_{\|\mathbf{y}\|_0 \leq \lceil \kappa d \rceil} \|\mathbf{z} - \mathbf{y}\|_q}{\|\mathbf{z}\|_q}. \tag{6.2}$$

(q, κ)-Compressibility intuitively quantifies how well the original vector $\mathbf{z} \in \mathbb{R}^d$ can be approximated if we restrict ourselves to using only a small ratio (κ) of its entries, setting the rest to 0. For our experiments, we set $q = 2$, corresponding to ℓ_2 norm, and set κ to an arbitrarily small value, 0.05. If a vector's (q, κ)-Compressibility is high with $\kappa = 0.05$, this means that we can approximate this vector with little error (in an ℓ_2 sense) by using 0.05 of its elements. To our knowledge, this is the first analysis of latent activations in deep learning with a notion of compressibility that reflects the nuanced view from compressed sensing.

6.2.2. Determining Approximate Rank

We now review the notions of rank we use in our investigation of activations. As previous research noted, using the actual notion of *matrix rank* is not suitable in the context we plan to utilize it [204], given that activation matrices we will be using

can be "full rank" due to various uninformative reasons, including issues relating to numerical precision. Therefore, our investigation requires *approximate* notions of rank, which would allow the quantification of activation matrices' ranks while also tolerating a degree of "observation noise". Given there are different approaches to determining approximate rank, we utilize three such variants to make sure the conclusions we make are not the artifacts of a specific choice for computing approximate rank.

As described above, we uniquely include nonnegative rank among these notions of approximate rank. Given that nonnegative matrix factorization is known for producing sparse, semantically meaningful decomposition of the data [19], (approximate) nonnegative rank is a notion of rank that could constitute and important, independent source of information for the analyses that we target in this chapter. That previous research did not focus on the notion of nonnegative rank might potentially be due to the difficulty of estimating this quantity [8], which is a problem that BAM substantially mitigates. Given a matrix $\mathbf{Z} \in \mathbb{Z}_{\geq 0}^{n \times d}$, and letting $\tau := \min(n, d)$, we define approximate nonnegative rank as

$$\text{NNRank}(\mathbf{Z}) = \arg\max_{R \in [\tau]} \hat{\mathcal{L}}_{\text{BAM}}(\mathbf{Z}; R), \tag{6.3}$$

where $\hat{\mathcal{L}}_{\text{BAM}}(\mathbf{Z}; R)$ denotes the log-marginal likelihood estimate by BAM for a rank R, and where we use BAM to conduct nonnegative matrix factorization (NMF), i.e. a CP decomposition on the 2-dimensional $\mathbf{Z}$.

In addition to nonnegative rank, we utilize two other, more commonly used notions of approximate rank that are deterministic. The first of these is *stable rank*, which is often used in generalization-related investigation of deep neural networks [203–205]. With $\mathbf{Z}$ defined identically to above, stable rank is defined as

$$\text{StableRank}(\mathbf{Z}) = \frac{\|\mathbf{Z}\|_F}{\|\mathbf{Z}\|_2}. \tag{6.4}$$

Another notion of rank that we use is *effective rank*. Proposed by [207], effective rank is based on the entropy of the normalized singular values of matrix's SVD. Let $\sigma_1, \ldots, \sigma_\tau$ denote the singular values of $\mathbf{Z}$. Defining $\bar{\sigma}_i = \sigma_i / \sum_j \sigma_j$, effective rank can

be computed as

$$\text{EffectiveRank}(\mathbf{Z}) = -\sum_{i=1}^{\tau} \bar{\sigma}_i \log(\bar{\sigma}_i), \tag{6.5}$$

where without loss of generality we ignore the exponentiation in the original definition [207], as in [204].

6.3. Experiments

We conduct our analyses on the models used in the previous chapter, due to the controlled manipulation of these models' training hyperparameters and lack of explicit regularization that could potentially complicate analyses. Given that our investigation in this chapter extends to OOD generalization as well, we ignore the experiment setting with MNIST and focus on models trained on CIFAR-10 and CIFAR-100. Given that no immediate effect of batch size was observed in the results presented in Chapter 5, we limit ourselves to models trained with a batch size of 100.

As before, we let the classifier $f_{\mathbf{w}}$ be a fully connected neural network with L hidden layers, and *weight matrices* $\{\mathbf{W}_l\}_{l=1}^{L}$, such that $\mathbf{W}_l \in \mathbb{R}^{h_l \times h_{l-1}}$. Similarly, h_l denotes the number of units at hidden layer l with $h_0 = d_X$ and $h_L = d_Y$, where d_X and d_Y are the dimensionalities of the input space $\mathcal{X}$ and output space $\mathcal{Y}$, respectively. Given an input $\mathbf{x}$, the neural network $f_{\mathbf{w}}(\mathbf{x}) : \mathcal{X} \mapsto \mathcal{Y}$ computes the output as

$$f_{\mathbf{w}}(\mathbf{x}) = \mathbf{W}_L \phi\left(\mathbf{W}_{L-1} \phi\left(\cdots \phi\left(\mathbf{W}_1 \mathbf{x}\right)\right)\right), \tag{6.6}$$

where again $\phi : \mathbb{R} \to \mathbb{R}$ is the element-wise applied ReLU activation function, i.e., $\phi(x) = \max(0, x)$. For brevity, from this point onwards we will abbreviate $f_{\mathbf{w}}$ as f unless otherwise specified. Accordingly, for $l \leq L$, we compute the *latent activation* of an input $\mathbf{x}$ for an *activation depth* of l as

$$f_l(\mathbf{x}) = \mathbf{W}_l \phi\left(\mathbf{W}_{l-1} \phi\left(\cdots \phi\left(\mathbf{W}_1 \mathbf{x}\right)\right)\right). \tag{6.7}$$

A frequently used latent activation is what is at times called a *learned representation* [203], which corresponds to $f_{L-1}(\mathbf{x})$. Given a dataset of n observations, let $\mathbf{X} \in \mathbb{R}^{n \times d_X} := [\mathbf{x}_1, \ldots, \mathbf{x}_n]^T$ denote the matrix representation of the inputs, where

each row constitutes another input image, flattened based on color representation. Then, we obtain the activation matrix for such a dataset at activation depth l through $\mathbf{Z}_l \in \mathbb{R}_{\geq 0}^{n \times d_l} := [\mathbf{z}_1, \ldots, \mathbf{z}_n]^T = [f_l(\mathbf{x}_1), \ldots, f_l(\mathbf{x}_n)]^T$.

Note that this construction does not translate to latent activations within CNN's in a straightforward manner. Ideally, we can utilize the flexibility offered by BAM to conduct NTF on the activations of the CNN models. For example, we can determine approximate nonnegative rank of the activation tensor through a CP decomposition on the 4-dimensional activation tensors. In order to keep comparability between different models, here we conduct a vectorization of CNN activations followed by NMF, and leave the aforementioned alternative as an intriguing future research direction. Thus, given the CNN classifier g and given the post-ReLU activation of an input $\mathbf{x}$ at activation depth l, $g_l(\mathbf{x})$, we let $\mathbf{Z}_l \in \mathbb{R}_{\geq 0}^{n \times d_l} := [\mathbf{z}_1, \ldots, \mathbf{z}_n]^T = [\mathbf{vec}(g_l(\mathbf{x}_1)), \ldots, \mathbf{vec}(g_l(\mathbf{x}_n))]^T$, where **vec** is again the vectorization operation. Given the fact that activations are not discrete, we conduct the discretization described in Section 1.3.2.3 before conducting our analyses with BAM for determining nonnegative rank.

In all these settings, given a trained network, the said representations can be extracted for the training data, $\mathbf{X}_{\text{train}}$, test data, $\mathbf{X}_{\text{test}}$, or some out-of-distribution test data, such as test images exposed to corruption such as blurring or cropping, $\mathbf{X}_{\text{OOD}}$. In our experiments, we conduct our investigations on all three splits. We conduct analyses on all post-ReLU activations. These correspond to 3, 5, and 5 different set of representations for FCN4, FCN6, and CNN respectively. We utilize all 50000 train and 10000 test samples in our analyses. Exceptions include limiting training split to 10000 (class-stratified) samples in CNN, due to the large dimensions of early activation matrices. We limit ourselves to 1000 samples when our examinations include NNRank, given that this requires scoring all candidate ranks through the computation of log-marginal likelihood for each trained network's all activation depths. To expedite computation, we fix the hyperparameter for BAM $a = 1e5$, given our preliminary observations that this hyperparameter leads to no qualitative changes in model orderings.

6.3.1. Results

6.3.1.1. Activation Compressibility and Generalization. Our first set of results pertain to compressibility of the representations and how they relate to generalization. As we show in Figures E.1 and E.2 in Appendix E, sparsity and (q, κ)-Compressibility provide qualitatively equivalent results, so we proceed with the latter in this chapter, calling this metric "compressibility" from this point onwards for brevity. For these experiments, we extract the learned representations, and compute average compressibility of samples from the train set for each model in question, and compare this with test accuracy of the models. Our results in Figure 6.1 show that models with more compressible representations are likely to have more test accuracy, closely tracking the results in the previous chapter. More interestingly, we also see that this relationship holds *within* a single model, compared across classes. In Figure 6.2 we take a single model from each experimental setting, i.e. model with learning rate = 0.01. We see that in each of the six settings, classes whose representations are on average more compressible are more likely to be accurately predicted. To our knowledge, this observation has not been made in previous literature that examined representation statistics and generalization. In Figures E.3 and E.4 in Appendix E we present the same figure with minimum or maximum available learning rate models in each setting, demonstrating that the results are not an artifact of the specific training learning rates we chose from within experiment settings. With these results, we move on to analyze how activation rank and generalization interrelate.

6.3.1.2. Activation Rank and Generalization. We first start by a comparison of the three different notions of rank. For this experiment, we select a subset of the models, with learning rates from all over the available range for each experimental setting (See Table E.1 in Appendix E for the list of learning rates selected per experiment). We then determine rank for each post-ReLU activation in all settings, for 1000 random samples from the test set. The computation of stable rank and effective rank rely on the SVD of the activation matrices, and for NNRank we use BAM with variational inference over a range of ranks, and select the highest scoring rank as the NNRank. We plot the

ranks of the activations against test set accuracy for all settings in Figure 6.3.

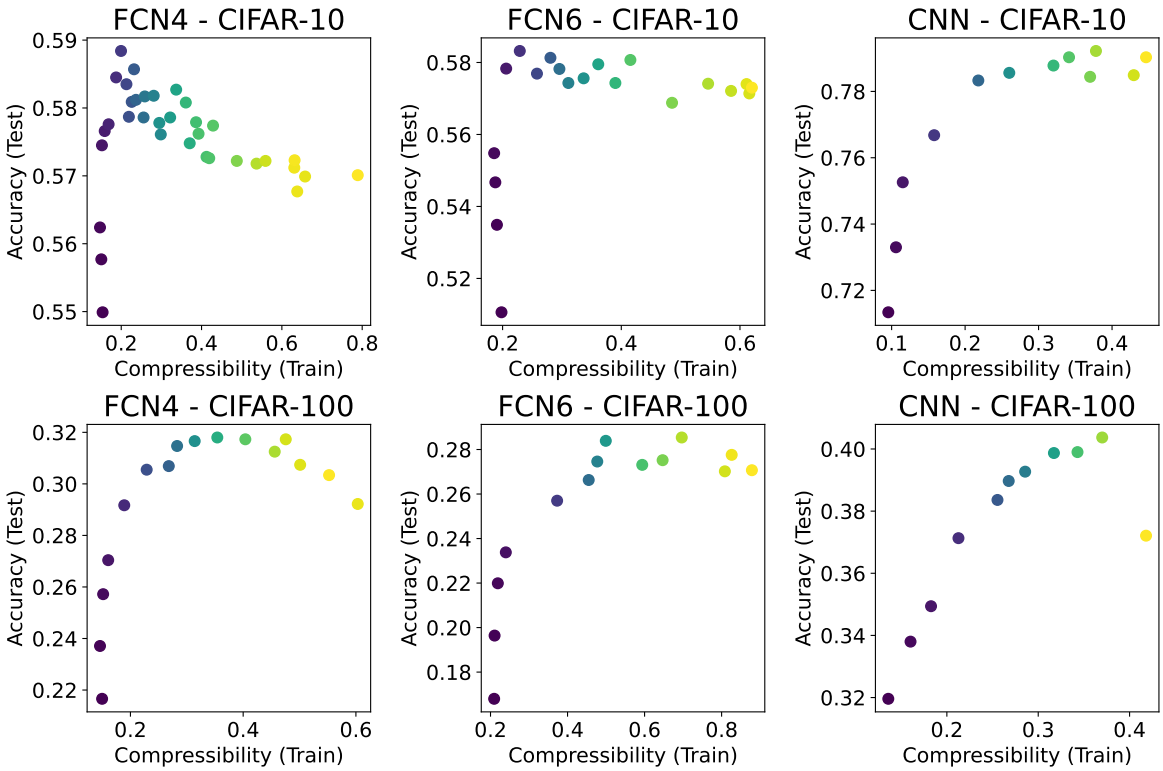


Figure 6.1. Activation compressibility and accuracy. Here compressibility refers to (q, κ)-Compressibility as defined in (6.2), with $q = 2, \kappa = 0.05$. Brighter colors imply higher learning rates (comparable only within subgraphs).

Our results show that although all three notions of rank demonstrate some variation among themselves, the overall pattern of results are qualitatively similar, showing that models whose activations have *higher* ranks are more likely to be more accurate. It is likely that some aspect of the SGD training regime is simultaneously leading the with higher learning rates to more compressible representations *and* higher activation ranks. Our results also establish that estimated nonnegative rank does not greatly diverge from the two other notions of approximate rank for the purposes of our experiments. As our findings call for further, more exhaustive experimentation, we leave the independent information provided by these different notions of rank as a future research direction, and continue only with effective rank for the remaining experiments for computational convenience.

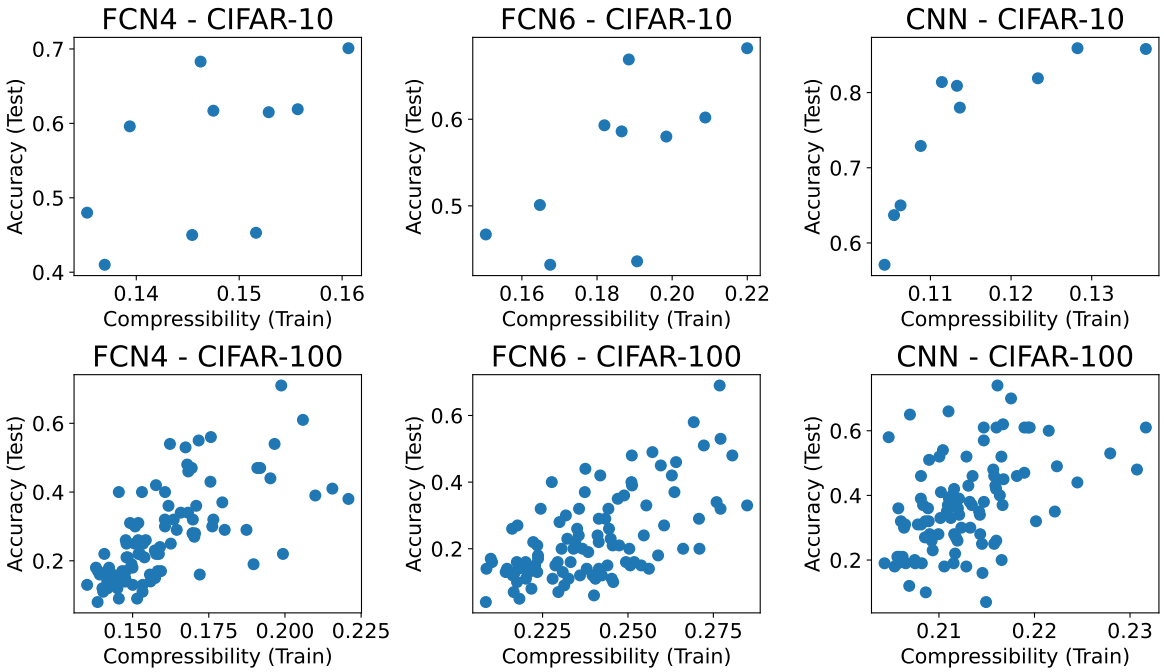


Figure 6.2. Class-conditional compressibility and accuracy. Compressibility of the activations are examined within a single model (learning rate = 0.01) for each setting, separately for each class. Here compressibility refers to (q, κ)-Compressibility as defined in (6.2), with $q = 2, \kappa = 0.05$.

We further investigate these results by extending the previous analysis to all trained models from the previous chapter. In Figure 6.4, we examine the relationship between test accuracy and effective rank in the training set. To produce a single rank for each model, we take the maximum rank produced by the network's activations throughout the layers. Here, results corroborate those presented in Figure 6.3. Higher test accuracy is correlated with higher effective rank, albeit imperfectly.

While overall compressibility and class-conditional compressibility show similar patterns in relation to accuracy, in the analysis of activation ranks we encounter a different pattern. Similar to above, we focus on a single model from each setting (learning rate = 0.01), and examine the relationship between class-conditional latent activation rank and test accuracy. Here we use the activations at the activation depth, $l = 1$, since they present the most variation among models, as seen in Figure 6.3,

though other choices present qualitatively similar patterns. We present our results in Figure 6.5, which show that class-conditional ranks are *inversely* correlated with accuracy. See Figures E.5 and E.6 in Appendix E as above for qualitatively identical results with minimum and maximum learning rates. Interestingly, as shown in Figure 6.6, maximum class-conditional rank averages for models still follow the original positive relationship. This leads to the intriguing conclusion that while representations having higher rank is associated with higher accuracy *overall*, within a single model, classes that have lower ranks are likely to be more accurately predicted. We discuss the implication of our results after presenting our results on OOD generalization below.

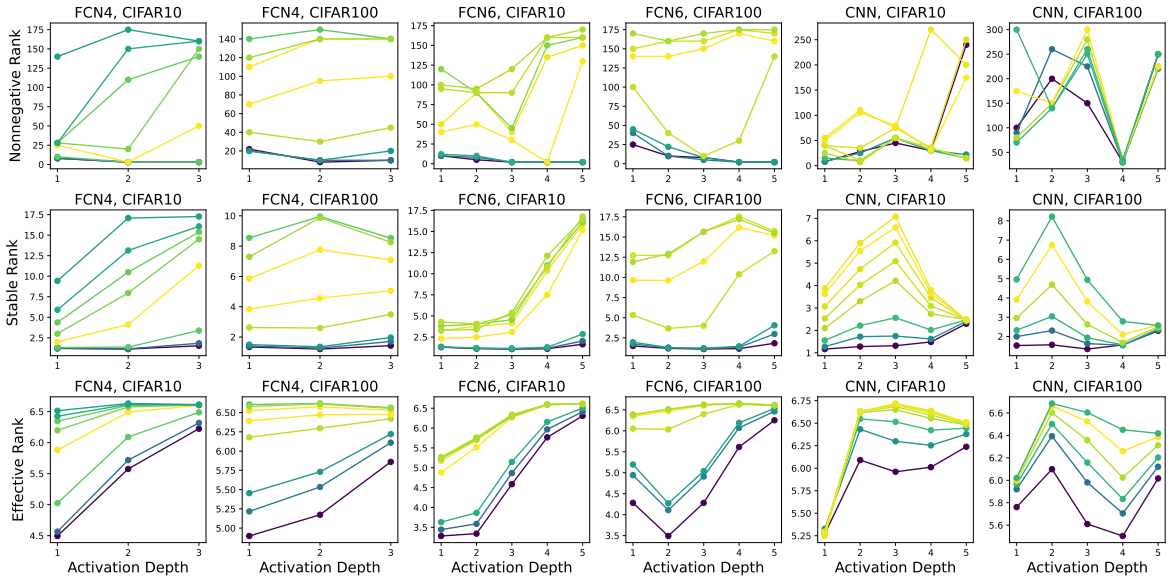


Figure 6.3. Activation ranks per layers on test set, using different methods for computing rank. Brighter colors imply higher overall test accuracies (comparable only within subgraphs).

6.3.1.3. Out-of-Distribution Generalization. Lastly, we examine how activation compressibility and rank relate to OOD generalization. For this we use the CIFAR-10-C [208] and CIFAR-100-C [209], as provided by [192], the former of which was also used in previous chapter. These two datasets add 19 different naturalistic noise types (e.g. blur, changing brightness, cropping) to the test sets of the CIFAR-10 and CIFAR-100 datasets respectively. We utilize 10000 test set examples with maximum amount

of noise applied for each of 19 noise types. We present our results in Appendix E, which show that although overall OOD performances are lower in general, the patterns described above almost completely hold for OOD generalization. One exception to this conformity is that some models are more robust than others to distribution shifts in CIFAR-10-C experiments, independently of learning rate. In addition to our results, Appendix E.1 provide a preliminary discussion regarding interesting performance differences between various noise types.

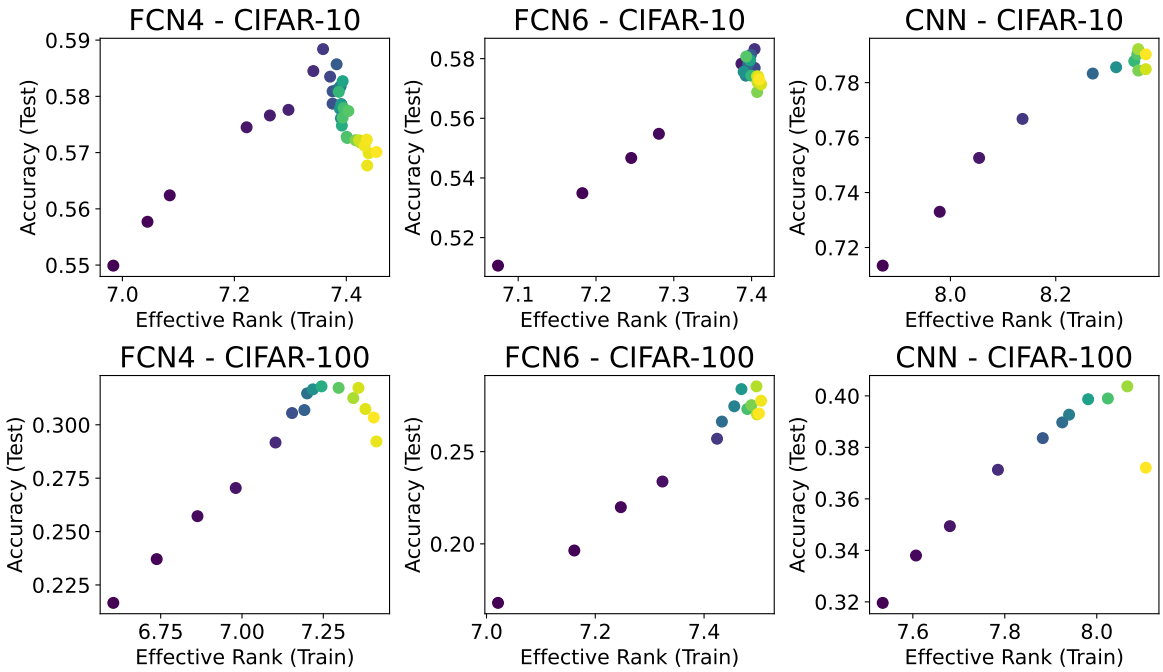


Figure 6.4. Maximum effective rank and accuracy. Each model's activations' effective rank has been summarized by the maximum effective rank observed in the trained model. Brighter colors imply higher learning rates (comparable only within subgraphs).

6.3.2. Discussion

We have shown that investigating how latent representations' compressibility and rank structure relate to robust generalization produce interesting and novel insights, over and above afforded by the analysis of model parameters. We consider this to be a

preliminary investigation, since as interesting as they are, their limitations are obvious. Various results in this chapter seem to be correlated with parameter statistics results presented in the previous chapter, possibly spuriously through learning dynamics induced by changing learning rates. Future research would benefit from focusing on 1- the mechanisms through which these characteristics emerge, 2- the reasons why overall and class-conditional rank-related results diverge while compressibility results do not, and 3- whether these trends persevere under different training schemes and model architectures. We come back to the last point in the next section.

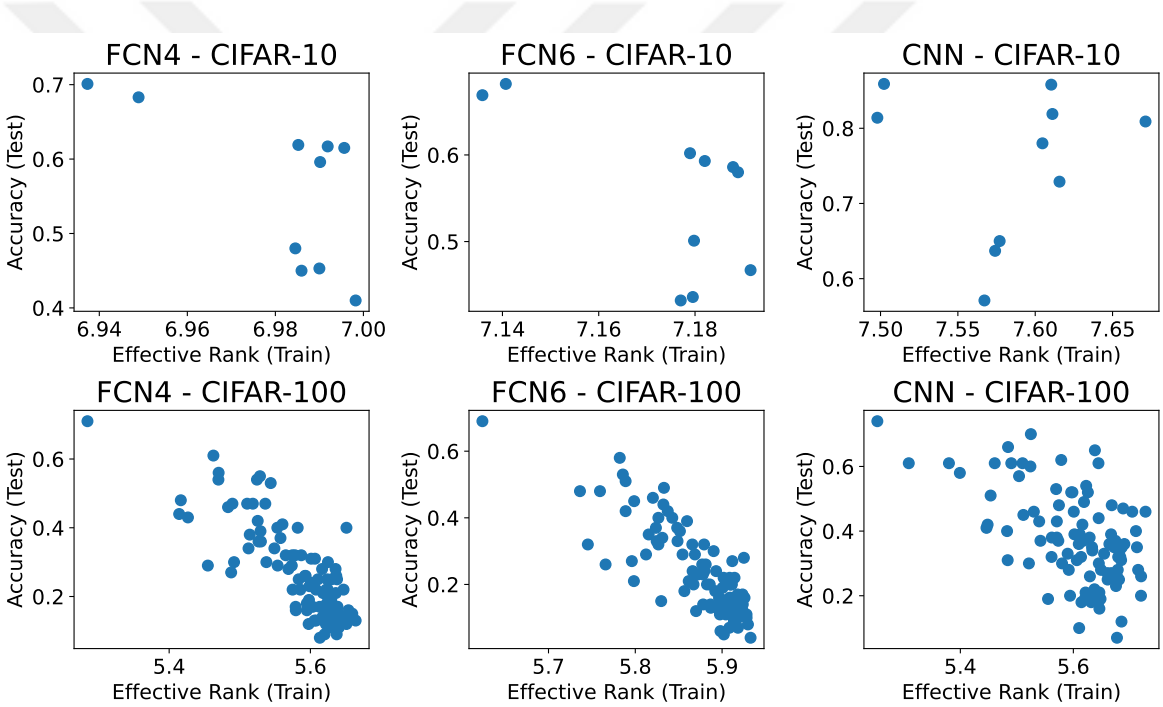


Figure 6.5. Class-conditional effective rank and accuracy. Effective rank of the initial activations are examined within a single model (learning rate = 0.01) for each setting, separately for each class. Brighter colors imply higher learning rates (comparable only within subgraphs).

Another limitation of the investigation in this chapter is that since the activations are produced by potentially heavy-tailed layers as presented in the previous chapter, their rank statistics might not be straightforward to compute. For example, Poisson likelihood BAM assumes is unlikely to be an accurate choice for the observations in

question. Future research on this topic should take into consideration the statistical properties of such activation patterns.

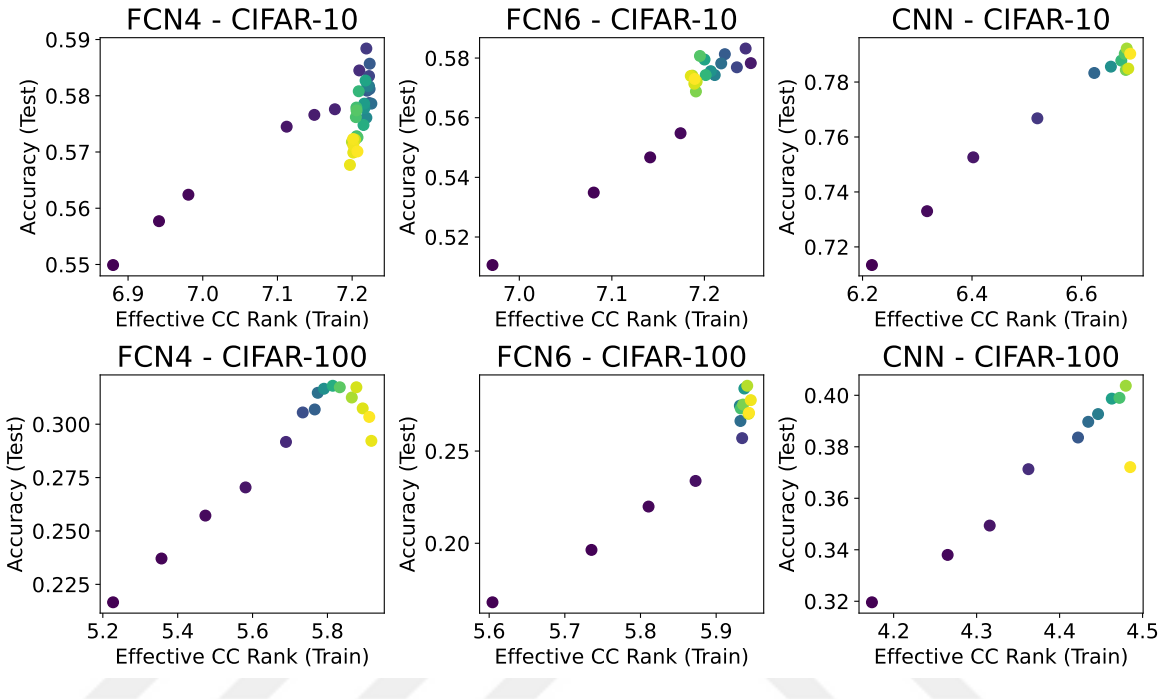


Figure 6.6. Maximum average class-conditional rank and accuracy. For each model, average class-conditional effective ranks are computed and maximum of these ranks across activation depths have been chosen to represent the model. Brighter colors imply higher learning rates (comparable only within subgraphs).

6.4. Analyzing Representations in Drug-Target Affinity Prediction

The results presented in the previous and current chapters demonstrate particular relationships between parameter and activation statistics, and robust generalization. An important future direction for such research is to investigate these relationships under different problem contexts, data types, model architectures, and different methodologies for achieving robust generalization. Therefore, a great candidate to provide comparative insights involve a framework that aims to achieve robust *drug-target affinity (DTA)* predictions by importance weighting of the input drug-target pairs in a way that mitigates the effect of spurious correlations. We now review this framework

and discuss how it can enrich the previous discussion and produce alternative insights. This section includes content from the article [57], co-authored by the author of this thesis.

6.4.1. DTA Prediction

Proteins and chemicals interact with each other by following fundamental physicochemical principles, and a complete understanding of these principles would allow the accurate prediction of protein-chemical interactions. Without such an understanding, machine learning approaches that rely on the available knowledge space of large interaction datasets can help to rapidly identify high-affinity protein-chemical pairs in the vast combination space by learning affinity patterns from measurements for millions of protein-chemical pairs [210]. However, a representative sampling of the entire combination space is impossible or infeasible, and thus the available datasets explore only portions of this space [211]. Therefore, machine learning algorithms that are trained on these available datasets are prone to learn some spurious relationships that are specific to those datasets and that are not generalizable to whole combination space [212–215]. Previous work suggests that these potentially misleading spurious patterns may be relatively easier for prediction models to pick up and can reduce their generalizability to novel proteins and chemicals, for which the learned patterns are non-applicable [216–220].

The problem of overcoming dataset biases in generalization performance of course is closely related to OOD generalization in machine learning at large [188]. The fact that models can pick up on spurious, simple relationships that would lead their predictions to suffer in terms of generalizability has been observed in the wider machine learning field as well, with this tendency sometimes being called *simplicity bias* or *shortcut learning* [221–223]. DebiasedDTA [57] is a framework that addresses this issue by changing the distribution of the input data through reweighting of the specific examples. Let us now describe the rationale behind this approach.

6.4.2. Causal Invariance in DTA Prediction

DebiasedDTA consists of two components, which is referred to as "guide" and "predictor". The guide aims to weight the training set instances in a way that facilitates learning the invariant relationships between biomolecules and their affinity, instead of dataset-specific spurious relationships. The predictor, on the other hand, can be any DTA prediction model that can use instance weighting in its loss function. These DTA prediction models include, but are not limited to, deep learning models, kernel machines, and tree-based ensemble models.

Given the high-dimensional nature of biomolecule inputs under commonly used representations, and the difficulty of collecting experimental data on affinity for all possible protein-chemical pairs, training datasets contain narrow regions of the potential interaction space and they are highly likely to contain spurious associations that can be picked up by supervised models. Consequently, models trained on them might fail to generalize to unseen data when the test distribution does not contain these spurious associations. The current method aims to address this problem by increasing the weights of observations that are likely to contain unique information that is not captured by such spurious associations and training the DTA prediction model with these new weights.

Let $\mathbf{x}_l, \mathbf{x}_p$ denote a ligand-protein pair (with arbitrary representations) and let y denote their affinity, such that $\mathbf{x}_l, \mathbf{x}_p, \mathrm{y} \sim p(\mathbf{x}_l, \mathbf{x}_p, \mathrm{y})$. The aim of any supervised DTA prediction task is to estimate $p(\mathrm{y}|\mathbf{x}_l, \mathbf{x}_p)$. Importantly, as discussed above, the conditional distribution $p(\mathrm{y}|\mathbf{x}_l, \mathbf{x}_p)$ can be assumed to be *invariant* between any training or test distributions since this is driven by mechanical/causal relationships. More formally, given the training and test set distributions p_S and p_T, we can say that

$$p(\mathrm{y}|\mathbf{x}_l, \mathbf{x}_p) = p_S(\mathrm{y}|\mathbf{x}_l, \mathbf{x}_p) = p_T(\mathrm{y}|\mathbf{x}_l, \mathbf{x}_p), \tag{6.8}$$

whereas $p_S(\mathbf{x}_l, \mathbf{x}_p)$ and $p_T(\mathbf{x}_l, \mathbf{x}_p)$ might be different, e.g. different ligands and proteins might be represented to varying extents in different datasets. This assertion regarding the invariance of $p(\mathrm{y}|\mathbf{x}_l, \mathbf{x}_p)$ is related to the well-known *causal invariance* and *inde-*

pendent causal mechanisms assumptions from statistical causality literature [53], [224].

Given the pair $\mathbf{x}_l, \mathbf{x}_p$ we have introduced above, let $\mathbf{z}_l, \mathbf{z}_p$ be simple, low dimensional statistics of $\mathbf{x}_l, \mathbf{x}_p$, such as the non/existence of a single atom or amino acid in the biomolecules. If the training data distribution p_S conforms to the graphical model

$$\mathrm{y} \leftarrow \mathbf{x}_l, \mathbf{x}_p \rightarrow \mathbf{z}_l, \mathbf{z}_p, \tag{6.9}$$

this would induce a spurious relationship between y and $\mathbf{z}_l, \mathbf{z}_p$. Then, a statistical learner might end up estimating a posterior distribution $\hat{p}_S$ such that $\hat{p}_S(\mathrm{y}|\mathbf{x}_l, \mathbf{x}_p) = \hat{p}_S(\mathrm{y}|\mathbf{z}_l, \mathbf{z}_p)$. This implies that this learner is deciding on affinity based on simple statistics of the input biomolecules. This estimated relationship would not be *robust* to distribution shifts, in the sense that it would not lead to correct decisions in a test distribution p_T where this spurious relationship is either nonexistent, i.e. $\mathrm{y} \leftarrow \mathbf{x}_l, \mathbf{x}_p \not\rightarrow \mathbf{z}_l, \mathbf{z}_p$, or changed in any other way. For example, relying on only the existence of sulfur in the ligand to decide on affinity might superficially work in a specific, limited training distribution, but this would not generalize to a large number of other ligand-protein combinations.

To combat this potential tendency in statistical learners, in addition to the causal invariance of affinity mechanisms, we assume that the spurious $y \leftrightarrow \mathbf{z}_l, \mathbf{z}_p$ relationship *can* be learned with a low-complexity model. Then, if a training instance's affinity cannot be predicted well by such a simple learner (which we can test through cross-validated prediction errors, for example), the implication would be that this instance includes some *unique information* that does not conform to the simple, spurious relationships learned by the low-complexity model. Then, increasing the weight of such instances modifies the training distribution so as to increase the influence of samples that do *not* conform to these spurious patterns. Overall, a predictor trained on this modified data distribution would then provide a different, better estimate of the p_S, such that

$$\tilde{p}_S(\mathrm{y}|\mathbf{x}_l, \mathbf{x}_p) \approx p_S(\mathrm{y}|\mathbf{x}_l, \mathbf{x}_p) \neq p_S(\mathrm{y}|\mathbf{z}_l, \mathbf{z}_p). \tag{6.10}$$

Note that $\tilde{p}_S$ is a much more desirable estimator compared to $\hat{p}_S$, since it ends up ap-

proximating the invariant, generalizable affinity relationship $\tilde{p}_S(\mathrm{y}|\mathbf{x}_l, \mathbf{x}_p) \approx p_S(\mathrm{y}|\mathbf{x}_l, \mathbf{x}_p) = p_T(\mathrm{y}|\mathbf{x}_l, \mathbf{x}_p)$, justifying this weighting strategy.

Table 6.1. Mean squared error (MSE) and concordance index (CI) scores for models trained with different learning rates in DeepDTA.

	Learning Rate					
	0.0005	0.001	0.005	0.01	0.05	0.075
MSE	2.3049	2.2889	1.9433	1.6796	1.9517	2.5024
CI	0.5486	0.5808	0.6737	0.7295	0.6915	0.5978

6.4.3. Robust Generalization in DTA Prediction

Since DebiasedDTA targets robust performance through a completely different *mechanism*, a systematic comparison between the previous training setting and DebiasedDTA-trained models is a promising direction for future research to further elucidate how model characteristics and performance relate. What makes this research even more interesting is the fact that when deep learning based DTA predictor model DeepDTA [225] is trained with various learning rates, the results that emerge show a striking parallelism to those found with FCNs and CNN in Chapter 5, as demonstrated in Table 6.1. Here, too, models trained with larger learning rates tend to perform better overall, except for near-maximum learning rate which similar to previous experiments becomes detrimental to performance. Here the evaluation is made in terms of *mean squared error (MSE)*, which relies on the magnitude of the prediction errors, and *concordance index (CI)*, which is an order-based metric that assesses the agreement between the orderings of biomolecule pairs in terms of predicted and actual affinity values. A systematic comparison of parameter statistics and representations both across different approaches to robustness within the DTA prediction problem, and between DTA prediction and other machine learning problems such as image classification stand out as promising future research directions.

7. CONCLUSION

In this thesis, we first examined the probabilistic modeling framework Bayesian allocation model (BAM), which combines flexibility and efficiency in formulating non-negative tensor factorization (NTF) models. We found that when combined with a simple yet effective approach in modeling seasonality, BAM allows the interpretable and accurate modeling of large-scale time series with complex, hierarchical seasonality. Then, we introduced a robust method for learning to defer to experts based on model output, and showed that NTF and BAM can play a crucial role in generalizing our approach to multiple classes, experts, and/or models. Further, we investigated the relationship between training hyperparameters, compressibility, parameter statistics, generalization, and robustness in deep learning models trained with SGD. We then used BAM to examine the corollaries of robust generalization in the representation space in previously unexplored directions. In addition to making standalone contributions to various supervised learning problems, our results showcase the exciting potential of more closely integrating NTF methods in supervised learning systems, with the conceptual and computational advantages of frameworks such as BAM.

The research included in this thesis covers a wide variety of topics, and future work can expand on what is presented here in a multitude of directions. For BAM itself, extending the framework to accommodate non-discrete data and different model assumptions would be of great use, some examples include extending it to coupled tensor factorization (data fusion) [226], different conditional distributions [227], or variants of NMF such as semi-NMF [1]. On the inference side, although mean-field variational inference (MFVI) proves to be a practical inference method, developing alternatives that do not make simplifying assumptions on the shape of the posterior is desirable. Although the SMC method introduced in [33] allows unbiased marginal likelihood estimates, its variance becomes prohibitive with a large number of observations. In addition to its variance, that its computational complexity scales with the number of observations makes it an unattractive choice for analyzing large-scale data. Mitigating

these concerns is an important future direction, as the convenience of MFVI comes at the cost of simplifying assumptions on the posterior distribution of the parameters.

For time series analysis using probabilistic NTF, exciting future directions include utilizing other observation likelihoods like Gaussian [91], using other models and/or inference methods for BLRHS [94, 95], more closely integrating the modeling of trend to our framework [228], using an additive sparse tensor term for outliers [99], involving ordinal dimensions [229], and capacity for handling seasonalities of statistically different nature such as those with heavy-tailed seasonality [227]. As expressed before, our results have implications not only for matrix/tensor based methods, but also for methods based on classical time series decompositions, which are still widely used [230, 231], as well as deep learning based methods [232, 233]. Therefore, the integration of LRHS/BLRHS within such work constitutes another important future research direction.

For learning to defer, given the flexibility afforded by BAM to model different scenarios for human-AI complementarity, more extensive experimentation that would test different scenarios of prediction and annotation would be helpful to determine opportunities and limitations of the existing method [106], [115], [114]. Methodologically, improving the current approach to accommodate missing data would be directly beneficial, given a lot of datasets that include expert annotation involve what is sometimes called *low-redundancy* scenarios, where any two experts co-annotate in only a number of cases. Ultimately, implications of our approach for training or fine-tuning baseline predictor models would also be of great interest.

For compressibility and generalization, future directions for research include formally identifying settings in which heavy-tailed mean-field limit (HML) condition holds and further examination of structured pruning in FCNs. Extending our work to other pruning strategies and network structures is also another important future direction. For example, it has recently been observed that gradients also exhibit heavy-tailed behavior [154], [170], [172]; we suspect that our results might be applicable in the case

of gradient-based pruning as well. The extension of our analyses to other network structures such as recurrent layers or ResNet blocks [234] would also be valuable.

The dearth of findings relating compressibility and robust generalization to the structure of the representation space, in the form of rank information and otherwise presents a promising future direction for this strand of research [235]. Examining how different approaches to achieving robustness affect the relationship between network behavior, parameter statistics, and representation space structure, and understanding how these in turn relate to training hyperparameters such as learning rate and training loss can make valuable contributions to our understanding of modern machine learning systems. In addition to potential contributions of NMF/NTF to the analysis of such systems, recent findings show that NMF can be highly effective as a module in end-to-end deep learning systems [9], [12], which constitutes another promising future research direction.

Throughout all of the topics we investigated, we were able to contribute to various supervised learning problems, and was able to use NTF through the BAM framework to enhance these contributions in novel ways, demonstrating the usefulness of having access to a flexible, convenient NTF framework for diverse tasks in supervised learning. As learning systems and architectures rise and fall in popularity, and as models and data get larger, improvements in fundamental statistical learning methods will likely continue to help develop more robust, reliable, and accurate systems in novel and ever-adapting ways.

REFERENCES

1. Kolda, T. G., and B. W. Bader, "Tensor Decompositions and Applications", *SIAM Review*, Vol. 51, No. 3, pp. 455–500, 2009.

2. Hitchcock, F. L., "The Expression of a Tensor or a Polyadic as a Sum of Products", *Journal of Mathematics and Physics*, Vol. 6, No. 1-4, pp. 164–189, 1927.

3. Cattell, R. B., "Parallel Proportional Profiles and Other Principles for Determining the Choice of Factors by Rotation", *Psychometrika*, Vol. 9, No. 4, pp. 267–283, 1944.

4. Tucker, L. R., "Some Mathematical Notes on Three-Mode Factor Analysis", *Psychometrika*, Vol. 31, No. 3, pp. 279–311, 1966.

5. Panagakis, Y., J. Kossaifi, G. G. Chrysos, J. Oldfield, M. A. Nicolaou, A. Anandkumar, and S. Zafeiriou, "Tensor Methods in Computer Vision and Deep Learning", *Proceedings of the IEEE*, Vol. 109, No. 5, pp. 863–890, 2021.

6. Bi, X., X. Tang, Y. Yuan, Y. Zhang, and A. Qu, "Tensors in Statistics", *Annual Review of Statistics and Its Application*, Vol. 8, No. 1, pp. 345–368, 2021.

7. Shashua, A., and T. Hazan, "Non-Negative Tensor Factorization with Applications to Statistics and Computer Vision", *International Conference on Machine Learning*, New York City, New York, pp. 792–799, 2005.

8. Cichocki, A., R. Zdunek, A. H. Phan, and S.-i. Amari, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*, Wiley, Chichester, UK, 2009.

9. Geng, Z., M.-H. Guo, H. Chen, X. Li, K. Wei, and Z. Lin, "Is Attention Better Than Matrix Decomposition?", *International Conference on Learning Represen-*

tations, Vienna, Austria, pp. 1–16, 2021.

10. Jamali, A. A., Y. Tan, A. Kusalik, and F.-X. Wu, "NTD-DR: Nonnegative Tensor Decomposition for Drug Repositioning", *PLOS One*, Vol. 17, No. 7, p. e0270852, 2022.

11. Ouyang, D., Y. Liang, J. Wang, X. Liu, S. Xie, R. Miao, N. Ai, L. Li, and Q. Dang, "Predicting Multiple Types of miRNA–Disease Associations Using Adaptive Weighted Nonnegative Tensor Factorization with Self-Paced Learning and Hypergraph Regularization", *Briefings in Bioinformatics*, Vol. 23, No. 6, pp. 1–14, 2022.

12. Ashtari, P., D. M. Sima, L. De Lathauwer, D. Sappey-Marinier, F. Maes, and S. Van Huffel, "Factorizer: A Scalable Interpretable Approach to Context Modeling for Medical Image Segmentation", *Medical Image Analysis*, Vol. 84, p. 102706, 2023.

13. Hubert, L., J. Meulman, and W. Heiser, "Two Purposes for Matrix Factorization: A Historical Appraisal", *SIAM Review*, Vol. 42, No. 1, pp. 68–82, 2000.

14. Koren, Y., R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems", *Computer*, Vol. 42, No. 8, pp. 30–37, 2009.

15. Lu, J., "Numerical Matrix Decomposition and Its Modern Applications: A Rigorous First Course", *arXiv:1807.10399*, 2023.

16. Cohen, J. E., and U. G. Rothblum, "Nonnegative Ranks, Decompositions, and Factorizations of Nonnegative Matrices", *Linear Algebra and Its Applications*, Vol. 190, pp. 149–168, 1993.

17. Paatero, P., and U. Tapper, "Positive Matrix Factorization: A Non-Negative Factor Model with Optimal Utilization of Error Estimates of Data Values", *Environmetrics*, Vol. 5, No. 2, pp. 111–126, 1994.

18. Lee, D. D., and H. S. Seung, "Algorithms for Non-Negative Matrix Factorization", *Advances in Neural Information Processing Systems*, Vol. 14, Vancouver, Canada, pp. 556–562, 2001.

19. Gillis, N., "The Why and How of Nonnegative Matrix Factorization", *arXiv:1401.5226*, 2014.

20. Lee, D. D., and H. S. Seung, "Learning the Parts of Objects by Non-Negative Matrix Factorization", *Nature*, Vol. 401, No. 6755, pp. 788–791, 1999.

21. Gillis, N., "Introduction to Nonnegative Matrix Factorization", *arXiv:1703.00663*, 2017.

22. Sidiropoulos, N. D., L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor Decomposition for Signal Processing and Machine Learning", *IEEE Transactions on Signal Processing*, Vol. 65, No. 13, pp. 3551–3582, 2017.

23. Harshman, R. A., "Foundations of the Parafac Procedure : Models and Conditions for an Explanatory Multimodal Factor Analysis", *UCLA Working Papers in Phonetics No. 16*, 1970.

24. Carroll, J. D., and J.-J. Chang, "Analysis of Individual Differences in Multidimensional Scaling via an N-Way Generalization of Eckart-Young Decomposition", *Psychometrika*, Vol. 35, No. 3, pp. 283–319, 1970.

25. Cichocki, A., D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor Decompositions for Signal Processing Applications: From Two-Way to Multiway Component Analysis", *IEEE Signal Processing Magazine*, Vol. 32, No. 2, pp. 145–163, 2015.

26. Wang, Y.-X., and Y.-J. Zhang, "Nonnegative Matrix Factorization: A Comprehensive Review", *IEEE Transactions on Knowledge and Data Engineering*,

Vol. 25, No. 6, pp. 1336–1353, 2013.

27. Chen, W.-S., Q. Zeng, and B. Pan, "A Survey of Deep Nonnegative Matrix Factorization", *Neurocomputing*, Vol. 491, pp. 305–320, 2022.

28. Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian Data Analysis*, Chapman and Hall/CRC, Boca Raton, Florida, 2013.

29. Cemgil, A. T., "Bayesian Inference for Nonnegative Matrix Factorisation Models", *Computational Intelligence and Neuroscience*, Vol. 2009, p. e785152, 2009.

30. Titsias, M., "The Infinite Gamma-Poisson Feature Model", *Advances in Neural Information Processing Systems*, Vol. 20, Vancouver, Canada, pp. 1513–1520, 2007.

31. Ermis, B., and A. T. Cemgil, "A Bayesian Tensor Factorization Model via Variational Inference for Link Prediction", *arXiv:1409.8276*, 2014.

32. Schein, A., J. Paisley, D. M. Blei, and H. Wallach, "Inferring Polyadic Events With Poisson Tensor Factorization", *NIPS 2014 Workshop on Networks: From Graphs to Rich Data*, Montreal, Canada, pp. 1–9, 2014.

33. Yıldırım, S., M. B. Kurutmaz, M. Barsbey, U. Şimşekli, and A. T. Cemgil, "Bayesian Allocation Model: Marginal Likelihood-Based Model Selection for Count Tensors", *IEEE Journal of Selected Topics in Signal Processing*, Vol. 15, No. 3, pp. 560–573, 2021.

34. Kurutmaz, M. B., M. Barsbey, A. T. Cemgil, S. Yıldırım, and U. Şimşekli, "Bayesian Model Selection for Identifying Markov Equivalent Causal Graphs", *Symposium on Advances in Approximate Bayesian Inference*, Vancouver, Canada, pp. 1–21, 2019.

35. Cemgil, A. T., M. B. Kurutmaz, S. Yıldırım, M. Barsbey, and U. Şimşekli,

"Bayesian Allocation Model: Inference by Sequential Monte Carlo for Nonnegative Tensor Factorizations and Topic Models using Polya Urns", *arXiv:1903.04478*, 2019.

36. Barsbey, M., and A. T. Cemgil, "Modeling Hierarchical Seasonality Through Low-Rank Tensor Decompositions in Time Series Analysis", *IEEE Access*, Vol. 11, pp. 85770–85784, 2023.

37. Kurutmaz, M. B., A. T. Cemgil, U. Şimşekli, and S. Yıldırım, "Bayesian Non-negative Matrix Factorization as an Allocation Model", *NIPS 2017 Workshop on Advances in Approximate Bayesian Inference*, Long Beach, California, pp. 1–5, 2017.

38. Kurutmaz, M. B., A. T. Cemgil, U. Şimşekli, M. Barsbey, and S. Yıldırım, "Bayesian Learning of Non-Negative Matrix/Tensor Factorizations by Simulating Pólya Urns", *Symposium on Advances in Approximate Bayesian Inference*, Montreal, Canada, pp. 1–6, 2018.

39. Blei, D. M., "Probabilistic Topic Models", *Communications of the ACM*, Vol. 55, No. 4, pp. 77–84, 2012.

40. Buntine, W., and A. Jakulin, "Discrete Component Analysis", C. Saunders, M. Grobelnik, S. Gunn, and J. Shawe-Taylor (Editors), *Subspace, Latent Structure and Feature Selection*, pp. 1–33, Springer, Berlin, Heidelberg, 2006.

41. Pearl, J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann Publishers, San Francisco, CA, 1988.

42. Blei, D., A. Ng, and M. Jordan, "Latent Dirichlet Allocation", *Advances in Neural Information Processing Systems*, Vol. 14, Vancouver, Canada, pp. 601–608, 2001.

43. Li, W., and A. McCallum, "Pachinko Allocation: DAG-structured Mixture Models of Topic Correlations", *International Conference on Machine Learning*, Pitts-

burgh, Pennsylvania, pp. 577–584, 2006.

44. Airoldi, E. M., D. Blei, S. Fienberg, and E. Xing, "Mixed Membership Stochastic Blockmodels", *Advances in Neural Information Processing Systems*, Vol. 21, Vancouver, Canada, pp. 33–40, 2008.

45. Çapan, G., S. Akbayrak, T. Y. Ceritli, and A. T. Cemgil, "Sum Conditioned Poisson Factorization", *Latent Variable Analysis and Signal Separation*, Guildford, UK, pp. 24–35, 2018.

46. Jordan, M. I., Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An Introduction to Variational Methods for Graphical Models", *Machine Learning*, Vol. 37, No. 2, pp. 183–233, 1999.

47. Wainwright, M. J., and M. I. Jordan, "Graphical Models, Exponential Families, and Variational Inference", *Foundations and Trends in Machine Learning*, Vol. 1, No. 1–2, pp. 1–305, 2008.

48. Barber, D., *Bayesian Reasoning and Machine Learning*, Cambridge University Press, Cambridge, UK, 2011.

49. Murphy, K. P., *Probabilistic Machine Learning: Advanced Topics*, MIT Press, Cambridge, Massachusetts, 2023.

50. Buntine, W., "Theory Refinement on Bayesian Networks", *Conference on Uncertainty in Artificial Intelligence*, San Francisco, California, pp. 52–60, 1991.

51. Heckerman, D., D. Geiger, and D. M. Chickering, "Learning Bayesian Networks: The Combination of Knowledge and Statistical Data", *Machine Learning*, Vol. 20, No. 3, pp. 197–243, 1995.

52. Geiger, D., and D. Heckerman, "A Characterization of the Dirichlet Distribution through Global and Local Parameter Independence", *The Annals of Statistics*,

Vol. 25, No. 3, pp. 1344–1369, 1997.

53. Peters, J., D. Janzing, and B. Schoelkopf, *Elements of Causal Inference: Foundations and Learning Algorithms*, MIT Press, Cambridge, MA, 2017.

54. Dvijotham, K. D., J. Winkens, M. Barsbey, S. Ghaisas, R. Stanforth, N. Pawlowski, P. Strachan, Z. Ahmed, S. Azizi, Y. Bachrach, L. Culp, M. Daswani, J. Freyberg, C. Kelly, A. Kiraly, T. Kohlberger, S. McKinney, B. Mustafa, V. Natarajan, K. Geras, J. Witowski, Z. Z. Qin, J. Creswell, S. Shetty, M. Sieniek, T. Spitz, G. Corrado, P. Kohli, T. Cemgil, and A. Karthikesalingam, "Enhancing the Reliability and Accuracy of AI-enabled Diagnosis via Complementarity-Driven Deferral to Clinicians", *Nature Medicine*, Vol. 29, No. 7, pp. 1814–1820, 2023.

55. Barsbey, M., M. Sefidgaran, M. A. Erdogdu, G. Richard, and U. Şimşekli, "Heavy Tails in SGD and Compressibility of Overparametrized Neural Networks", *Advances in Neural Information Processing Systems*, Vol. 34, Vancouver, Canada, pp. 29364–29378, 2021.

56. Raj, A., M. Barsbey, M. Gurbuzbalaban, L. Zhu, and U. Şimşekli, "Algorithmic Stability of Heavy-Tailed Stochastic Gradient Descent on Least Squares", *International Conference on Algorithmic Learning Theory*, Singapore, pp. 1292–1342, 2023.

57. Özçelik, R., A. Bağ, B. Atıl, M. Barsbey, A. Özgür, and E. Özkırımlı, "Debiased-DTA: A Framework for Improving the Generalizability of Drug-Target Affinity Prediction Models", *arXiv:2107.05556*, 2023.

58. IEEE, "Avoid Infringement upon IEEE Copyright", 2023, `https://journals.ieeeauthorcenter.ieee.org/choose-a-publishing-agreement/avoid-infringement-upon-ieee-copyright/`, accessed on July 23, 2023.

59. Nature Medicine, "Permissions Requests", 2023, `https://www.nature.com/nature-portfolio/reprints-and-permissions/permissions-requests`, accessed on July 23, 2023.

60. Barsbey, M., "Source Code for PhD Dissertation: Utilizing Nonnegative Tensor Factorization Methods for Inference, Model Selection, and Analysis in Supervised Learning", 2023, `https://github.com/mbarsbey/phd-dissertation`, accessed on July 25, 2023.

61. Ghysels, E., D. R. Osborn, and T. J. Sargent, *The Econometric Analysis of Seasonal Time Series*, Cambridge University Press, Cambridge, UK, 2001.

62. Franses, P. H., and R. Paap, *Periodic Time Series Models*, Oxford University Press, Oxford, UK, 2004.

63. Hamilton, J. D., *Time Series Analysis*, Princeton University Press, Princeton, New Jersey, 1994.

64. Hyndman, R. J., and G. Athanasopoulos, *Forecasting: Principles and Practice*, OTexts, Melbourne, Australia, 2018.

65. Gould, P. G., A. B. Koehler, J. K. Ord, R. D. Snyder, R. J. Hyndman, and F. Vahid-Araghi, "Forecasting Time Series with Multiple Seasonal Patterns", *European Journal of Operational Research*, Vol. 191, No. 1, pp. 207–222, 2008.

66. Chang, J., J. He, L. Yang, and Q. Yao, "Modelling Matrix Time Series via a Tensor CP-decomposition", *Journal of the Royal Statistical Society Series B: Statistical Methodology*, Vol. 85, No. 1, pp. 127–148, 2023.

67. Figueiredo, M., B. Ribeiro, and A. de Almeida, "Analysis of Trends in Seasonal Electrical Energy Consumption via Non-negative Tensor Factorization", *Neurocomputing*, Vol. 170, pp. 318–327, 2015.

68. Ji, T., Y. Jiang, M. Li, and Q. Wu, "Ultra-Short-Term Wind Speed and Wind Power Forecast via Selective Hankelization and Low-Rank Tensor Learning-Based Predictor", *International Journal of Electrical Power & Energy Systems*, Vol. 140, p. 107994, 2022.

69. Chen, X., C. Zhang, X.-L. Zhao, N. Saunier, and L. Sun, "Nonstationary Temporal Matrix Factorization for Multivariate Time Series Forecasting", *arXiv:2203.10651*, 2022.

70. Chan, R. K. C., J. M.-Y. Lim, and R. Parthiban, "Missing Traffic Data Imputation for Artificial Intelligence in Intelligent Transportation Systems: Review of Methods, Limitations, and Challenges", *IEEE Access*, Vol. 11, pp. 34080–34093, 2023.

71. Espín Noboa, L., F. Lemmerich, P. Singer, and M. Strohmaier, "Discovering and Characterizing Mobility Patterns in Urban Spaces: A Study of Manhattan Taxi Data", *International Conference on World Wide Web*, Montreal, Canada, pp. 537–542, 2016.

72. Gao, Y., L. T. Yang, J. Yang, D. Zheng, and Y. Zhao, "Jointly Low-Rank Tensor Completion for Estimating Missing Spatiotemporal Values in Logistics Systems", *IEEE Transactions on Industrial Informatics*, Vol. 19, No. 2, pp. 1814–1822, 2023.

73. Zhang, P., P. Ren, Y. Liu, and H. Sun, "Autoregressive Matrix Factorization for Imputation and Forecasting of Spatiotemporal Structural Monitoring Time Series", *Mechanical Systems and Signal Processing*, Vol. 169, p. 108718, 2022.

74. Dunlavy, D. M., T. G. Kolda, and E. Acar, "Temporal Link Prediction Using Matrix and Tensor Factorizations", *ACM Transactions on Knowledge Discovery from Data*, Vol. 5, No. 2, pp. 1–27, 2011.

75. de Araujo, M. R., P. M. P. Ribeiro, and C. Faloutsos, "TensorCast: Forecasting

with Context using Coupled Tensors", *IEEE International Conference on Data Mining*, New Orleans, Louisiana, pp. 71–80, 2017.

76. Xiong, L., X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell, "Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization", *SIAM International Conference on Data Mining*, Columbus, Ohio, pp. 211–222, 2010.

77. Matsubara, Y., Y. Sakurai, C. Faloutsos, T. Iwata, and M. Yoshikawa, "Fast Mining and Forecasting of Complex Time-Stamped Events", *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Beijing, China, pp. 271–279, 2012.

78. Matsubara, Y., Y. Sakurai, W. G. Van Panhuis, and C. Faloutsos, "FUNNEL: Automatic Mining of Spatially Coevolving Epidemics", *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York City, New York, pp. 105–114, 2014.

79. Rogers, M., L. Li, and S. J. Russell, "Multilinear Dynamical Systems for Tensor Time Series", *Advances in Neural Information Processing Systems*, Vol. 26, Stateline, Nevada, pp. 2634–2642, 2013.

80. Yu, H.-F., N. Rao, and I. S. Dhillon, "Temporal Regularized Matrix Factorization for High-Dimensional Time Series Prediction", *Advances in Neural Information Processing Systems*, Vol. 29, Barcelona, Spain, pp. 847–855, 2016.

81. Takeuchi, K., H. Kashima, and N. Ueda, "Autoregressive Tensor Factorization for Spatio-Temporal Predictions", *IEEE International Conference on Data Mining*, New Orleans, Louisiana, pp. 1105–1110, 2017.

82. Sun, L., and X. Chen, "Bayesian Temporal Factorization for Multidimensional Time Series Prediction", *arXiv:1910.06366*, 2019.

83. Kawabata, K., S. Bhatia, R. Liu, M. Wadhwa, and B. Hooi, "SSMF: Shifting Sea-

sonal Matrix Factorization", *Advances in Neural Information Processing Systems*, Vol. 34, Vancouver, Canada, pp. 3863–3873, 2021.

84. Xie, Y.-L., P. K. Hopke, P. Paatero, L. A. Barrie, and S.-M. Li, "Identification of Source Nature and Seasonal Variations of Arctic Aerosol by Positive Matrix Factorization", *Journal of the Atmospheric Sciences*, Vol. 56, No. 2, pp. 249–260, 1999.

85. Takahashi, T., B. Hooi, and C. Faloutsos, "AutoCyclone: Automatic Mining of Cyclic Online Activities with Robust Tensor Factorization", *International Conference on World Wide Web*, Perth, Australia, pp. 213–221, 2017.

86. Matsubara, Y., Y. Sakurai, and C. Faloutsos, "The Web as a Jungle: Non-linear Dynamical Systems for Co-evolving Online Activities", *International Conference on World Wide Web*, Montreal, Canada, pp. 721–731, 2015.

87. Matsubara, Y., Y. Sakurai, and C. Faloutsos, "Non-linear Mining of Competing Local Activities", *International Conference on World Wide Web*, Montreal, Canada, pp. 737–747, 2016.

88. Chen, X., and L. Sun, "Low-Rank Autoregressive Tensor Completion for Multivariate Time Series Forecasting", *arXiv:2006.10436*, 2021.

89. Chen, X., M. Lei, N. Saunier, and L. Sun, "Low-Rank Autoregressive Tensor Completion for Spatiotemporal Traffic Data Imputation", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 23, No. 8, pp. 12301–12310, 2022.

90. Tan, H., Y. Wu, B. Shen, P. J. Jin, and B. Ran, "Short-Term Traffic Prediction Based on Dynamic Tensor Completion", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 17, No. 8, pp. 2123–2133, 2016.

91. Chen, X., Z. He, and L. Sun, "A Bayesian Tensor Decomposition Approach for Spatiotemporal Traffic Data Imputation", *Transportation Research Part C:*

Emerging Technologies, Vol. 98, pp. 73–84, 2019.

92. Wang, Y., Y. Zhang, L. Wang, Y. Hu, and B. Yin, "Urban Traffic Pattern Analysis and Applications Based on Spatio-Temporal Non-Negative Matrix Factorization", *IEEE Transactions on Intelligent Transportation Systems*, Vol. 23, No. 8, pp. 12752–12765, 2022.

93. Mnih, A., and R. R. Salakhutdinov, "Probabilistic Matrix Factorization", *Advances in Neural Information Processing Systems*, Vol. 20, Vancouver, Canada, pp. 1257–1264, 2007.

94. Hu, C., P. Rai, C. Chen, M. Harding, and L. Carin, "Scalable Bayesian Non-negative Tensor Factorization for Massive Count Data", *Machine Learning and Knowledge Discovery in Databases*, Porto, Portugal, pp. 53–70, 2015.

95. Schein, A., M. Zhou, D. M. Blei, and H. Wallach, "Bayesian Poisson Tucker Decomposition for Learning the Structure of International Relations", *International Conference on Machine Learning*, New York City, New York, pp. 2810–2819, 2016.

96. Barsbey, M., and A. T. Cemgil, "Source Code for the Paper: Modeling Hierarchical Seasonality through Low-Rank Tensor Decompositions in Time Series Analysis", 2023, `https://github.com/mbarsbey/lrhs`, accessed on July 25, 2023.

97. Bradbury, J., R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, "JAX: Composable Transformations of Python+NumPy Programs", 2018, `http://github.com/google/jax`, accessed on July 25, 2018.

98. City of New York, "TLC Trip Record Data", 2023, `https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page`, accessed on July 25, 2023.

99. Zhu, Y., W. Wang, G. Yu, J. Wang, and L. Tang, "A Bayesian Robust CP Decomposition Approach for Missing Traffic Data Imputation", *Multimedia Tools and Applications*, Vol. 81, No. 23, pp. 33171–33184, 2022.

100. Chen, X., Y. Chen, and Z. He, "Urban Traffic Speed Dataset of Guangzhou, China", 2018, `https://zenodo.org/record/1205229`, accessed on July 25, 2023.

101. Chen, X., "BGCP", 2019, `https://github.com/mcgill-smart-transport/bgcp_imputation`, accessed on July 25, 2023.

102. Liu, J., P. Musialski, P. Wonka, and J. Ye, "Tensor Completion for Estimating Missing Values in Visual Data", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 35, No. 1, pp. 208–220, 2013.

103. Chen, X., Z. He, and J. Wang, "Spatial-Temporal Traffic Speed Patterns Discovery and Incomplete Data Recovery via SVD-combined Tensor Decomposition", *Transportation Research Part C: Emerging Technologies*, Vol. 86, pp. 59–77, 2018.

104. DeepMind, "Source Code for the Paper: Complementarity-Driven Deferral to Clinicians (CoDoC)", 2023, `https://github.com/deepmind/codoc`, accessed on July 25, 2023.

105. Kamar, E., "Directions in Hybrid Intelligence: Complementing AI Systems with Human Intelligence", *International Joint Conference on Artificial Intelligence*, New York City, New York, pp. 4070–4073, 2016.

106. Leitão, D., P. Saleiro, M. A. T. Figueiredo, and P. Bizarro, "Human-AI Collaboration in Decision-Making: Beyond Learning to Defer", *arXiv:2206.13202*, 2022.

107. Steyvers, M., H. Tejeda, G. Kerrigan, and P. Smyth, "Bayesian Modeling of Human–AI Complementarity", *Proceedings of the National Academy of Sciences*, Vol. 119, No. 11, p. e2111547119, 2022.

108. Raghu, M., K. Blumer, G. Corrado, J. Kleinberg, Z. Obermeyer, and S. Mullainathan, "The Algorithmic Automation Problem: Prediction, Triage, and Human Effort", *arXiv:1903.12220*, 2019.

109. Mozannar, H., and D. Sontag, "Consistent Estimators for Learning to Defer to an Expert", *arXiv:2006.01862*, 2021.

110. Madras, D., T. Pitassi, and R. Zemel, "Predict Responsibly: Improving Fairness and Accuracy by Learning to Defer", *Advances in Neural Information Processing Systems*, Vol. 31, Montreal Canada, pp. 6147–6157, 2018.

111. Okati, N., A. De, and M. Rodriguez, "Differentiable Learning Under Triage", *Advances in Neural Information Processing Systems*, Vol. 34, Vancouver, Canada, pp. 9140–9151, 2021.

112. Charusaie, M.-A., H. Mozannar, D. Sontag, and S. Samadi, "Sample Efficient Learning of Predictors That Complement Humans", *International Conference on Machine Learning*, Baltimore, Maryland, pp. 2972–3005, 2022.

113. Keswani, V., M. Lease, and K. Kenthapadi, "Towards Unbiased and Accurate Deferral to Multiple Experts", *AAAI/ACM Conference on AI, Ethics, and Society*, New York City, New York, pp. 154–165, 2021.

114. Verma, R., and E. Nalisnick, "Calibrated Learning to Defer with One-vs-All Classifiers", *International Conference on Machine Learning*, Baltimore, Maryland, pp. 22184–22202, 2022.

115. Narasimhan, H., W. Jitkrittum, A. K. Menon, A. Rawat, and S. Kumar, "Post-Hoc Estimators for Learning to Defer to an Expert", *Advances in Neural Information Processing Systems*, Vol. 35, New Orleans, Louisiana, pp. 29292–29304, 2022.

116. Wilder, B., E. Horvitz, and E. Kamar, "Learning to Complement Humans", *In-*

ternational Joint Conference on Artificial Intelligence, Yokohama, Japan, pp. 1526–1533, 2020.

117. Silverman, B. W., "Algorithm AS 176: Kernel Density Estimation Using the Fast Fourier Transform", *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, Vol. 31, No. 1, pp. 93–99, 1982.

118. Hall, P., and M. P. Wand, "On the Accuracy of Binned Kernel Density Estimators", *Journal of Multivariate Analysis*, Vol. 56, No. 2, pp. 165–184, 1996.

119. Silverman, B. W., *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, Boca Raton, Florida, 1986.

120. Fan, J., and J. S. Marron, "Fast Implementations of Nonparametric Curve Estimators", *Journal of Computational and Graphical Statistics*, Vol. 3, No. 1, pp. 35–56, 1994.

121. Leibig, C., M. Brehmer, S. Bunk, D. Byng, K. Pinker, and L. Umutlu, "Combining the Strengths of Radiologists and AI for Breast Cancer Screening: A Retrospective Analysis", *The Lancet Digital Health*, Vol. 4, No. 7, pp. e507–e519, 2022.

122. Ardila, D., A. P. Kiraly, S. Bharadwaj, B. Choi, J. J. Reicher, L. Peng, D. Tse, M. Etemadi, W. Ye, G. Corrado, D. P. Naidich, and S. Shetty, "End-to-End Lung Cancer Screening with Three-Dimensional Deep Learning on Low-Dose Chest Computed Tomography", *Nature Medicine*, Vol. 25, No. 6, pp. 954–961, 2019.

123. McKinney, S. M., M. Sieniek, V. Godbole, J. Godwin, N. Antropova, H. Ashrafian, T. Back, M. Chesus, G. S. Corrado, A. Darzi, M. Etemadi, F. Garcia-Vicente, F. J. Gilbert, M. Halling-Brown, D. Hassabis, S. Jansen, A. Karthikesalingam, C. J. Kelly, D. King, J. R. Ledsam, D. Melnick, H. Mostofi, L. Peng, J. J. Reicher, B. Romera-Paredes, R. Sidebottom, M. Suleyman, D. Tse, K. C. Young, J. De Fauw, and S. Shetty, "International Evaluation of an AI System for Breast

Cancer Screening", *Nature*, Vol. 577, No. 7788, pp. 89–94, 2020.

124. Stadnick, B., J. Witowski, V. Rajiv, J. Chledowski, F. E. Shamout, K. Cho, and K. J. Geras, "Meta-Repository of Screening Mammography Classifiers", *arXiv:2108.04800*, 2022.

125. Qin, Z. Z., S. Ahmed, M. S. Sarker, K. Paul, A. S. S. Adel, T. Naheyan, R. Barrett, S. Banu, and J. Creswell, "Tuberculosis Detection from Chest X-Rays for Triaging in a High Tuberculosis-Burden Setting: An Evaluation of Five Artificial Intelligence Algorithms", *The Lancet Digital Health*, Vol. 3, No. 9, pp. e543–e554, 2021.

126. Dawid, A. P., and A. M. Skene, "Maximum Likelihood Estimation of Observer Error-Rates Using the EM Algorithm", *Applied Statistics*, Vol. 28, No. 1, p. 20, 1979.

127. Stutz, D., A. T. Cemgil, A. G. Roy, T. Matejovicova, M. Barsbey, P. Strachan, M. Schaekermann, J. Freyberg, R. Rikhye, B. Freeman, J. P. Matos, U. Telang, D. R. Webster, Y. Liu, G. S. Corrado, Y. Matias, P. Kohli, Y. Liu, A. Doucet, and A. Karthikesalingam, "Evaluating AI Systems under Uncertain Ground Truth: A Case Study in Dermatology", *arXiv:2202.08821*, 2023.

128. Hughes, M. C., and E. B. Sudderth, "BNPy : Reliable and Scalable Variational Inference for Bayesian Nonparametric Models", *Probabilistic Programming Workshop at NIPS*, Montreal, Canada, pp. 1–4, 2014.

129. Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge", *arXiv:1409.0575*, 2015.

130. Krizhevsky, A., I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", *Communications of the ACM*, Vol. 60,

No. 6, pp. 84–90, 2017.

131. Huang, G., Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks", *IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, Hawaii, pp. 4700–4708, 2017.

132. He, K., X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition", *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, Nevada, pp. 770–778, 2016.

133. Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going Deeper With Convolutions", *IEEE Conference on Computer Vision and Pattern Recognition*, Boston, Massachusetts, pp. 1–9, 2015.

134. Simonyan, K., and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", *arXiv:1409.1556*, 2015.

135. O'Neill, J., "An Overview of Neural Network Compression", *arXiv:2006.03669*, 2020.

136. Blalock, D., J. J. G. Ortiz, J. Frankle, and J. Guttag, "What Is the State of Neural Network Pruning?", *arXiv:2003.03033*, 2020.

137. Arora, S., R. Ge, B. Neyshabur, and Y. Zhang, "Stronger Generalization Bounds for Deep Nets via a Compression Approach", *International Conference on Machine Learning*, Stockholm, Sweden, pp. 254–263, 2018.

138. Frankle, J., and M. Carbin, "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks", *International Conference on Learning Representations*, New Orleans, Louisiana, pp. 1–13, 2019.

139. Frankle, J., G. K. Dziugaite, D. Roy, and M. Carbin, "Linear Mode Connectiv-

ity and the Lottery Ticket Hypothesis", *International Conference on Machine Learning*, Vienna, Austria, pp. 3259–3269, 2020.

140. Hattie Zhou, Janice Lan, Rosanne Liu, and J. Yosinski, "Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask", *arXiv:1905.01067*, 2020.

141. Renda, A., J. Frankle, and M. Carbin, "Comparing Rewinding and Fine-tuning in Neural Network Pruning", *arXiv:2003.02389*, 2020.

142. Suzuki, T., H. Abe, T. Murata, S. Horiuchi, K. Ito, T. Wachi, S. Hirai, M. Yukishima, and T. Nishimura, "Spectral Pruning: Compressing Deep Neural Networks via Spectral Analysis and its Generalization Error", *International Joint Conference on Artificial Intelligence*, Yokohama, Japan, pp. 2839–2846, 2020.

143. Suzuki, T., H. Abe, and T. Nishimura, "Compression Based Bound for Non-Compressed Network: Unified Generalization Error Analysis of Large Compressible Deep Neural Network", *International Conference on Learning Representations*, Addis Ababa, Ethiopia, pp. 1–14, 2020.

144. Hsu, D., Z. Ji, M. Telgarsky, and L. Wang, "Generalization Bounds via Distillation", *International Conference on Learning Representations*, Vienna, Austria, pp. 1–10, 2021.

145. Kuhn, L., C. Lyle, A. N. Gomez, J. Rothfuss, and Y. Gal, "Robustness to Pruning Predicts Generalization in Deep Neural Networks", *arXiv:2103.06002*, 2021.

146. Hodgkinson, L., and M. W. Mahoney, "Multiplicative Noise and Heavy Tails in Stochastic Optimization", *arXiv:2006.06293*, 2020.

147. Gurbuzbalaban, M., U. Şimşekli, and L. Zhu, "The Heavy-Tail Phenomenon in SGD", *International Conference on Machine Learning*, Vienna, Austria, pp. 3964–3975, 2021.

148. Mei, S., A. Montanari, and P.-M. Nguyen, "A Mean Field View of the Landscape of Two-Layer Neural Networks", *Proceedings of the National Academy of Sciences*, Vol. 115, No. 33, pp. E7665–E7671, 2018.

149. Sirignano, J., and K. Spiliopoulos, "Mean Field Analysis of Neural Networks: A Central Limit Theorem", *Stochastic Processes and Their Applications*, Vol. 130, No. 3, pp. 1820–1852, 2020.

150. De Bortoli, V., A. Durmus, X. Fontaine, and U. Şimşekli, "Quantitative Propagation of Chaos for SGD in Wide Neural Networks", *Advances in Neural Information Processing Systems*, Vol. 33, Vancouver, Canada, pp. 278–288, 2020.

151. Gribonval, R., V. Cevher, and M. E. Davies, "Compressible Distributions for High-Dimensional Statistics", *IEEE Transactions on Information Theory*, Vol. 58, No. 8, pp. 5016–5034, 2012.

152. Amini, A., M. Unser, and F. Marvasti, "Compressibility of Deterministic and Random Infinite Sequences", *IEEE Transactions on Signal Processing*, Vol. 59, No. 11, pp. 5193–5201, 2011.

153. Şimşekli, U., O. Sener, G. Deligiannidis, and M. A. Erdogdu, "Hausdorff Dimension, Heavy Tails, and Generalization in Neural Networks", *Advances in Neural Information Processing Systems*, Vol. 33, Vancouver, Canada, pp. 5138–5151, 2020.

154. Zhou, P., J. Feng, C. Ma, C. Xiong, S. C. H. Hoi, and W. E, "Towards Theoretically Understanding Why SGD Generalizes Better Than Adam in Deep Learning", *Advances in Neural Information Processing Systems*, Vol. 33, Vancouver, Canada, pp. 21285–21296, 2020.

155. Neyshabur, B., S. Bhojanapalli, and N. Srebro, "A PAC-Bayesian Approach to Spectrally-Normalized Margin Bounds for Neural Networks", *arXiv:1707.09564*,

2018.

156. Nolan, J. P., *Univariate Stable Distributions: Models for Heavy Tailed Data*, Springer, Cham, Switzerland, 2020.

157. Şimşekli, U., L. Sagun, and M. Gurbuzbalaban, "A Tail-Index Analysis of Stochastic Gradient Noise in Deep Neural Networks", *International Conference on Machine Learning*, New Orleans, Louisiana, pp. 5827–5837, 2019.

158. Peluchetti, S., S. Favaro, and S. Fortini, "Stable Behaviour of Infinitely Wide Deep Neural Networks", *International Conference on Artificial Intelligence and Statistics*, Online, pp. 1137–1146, 2020.

159. Wang, H., M. Gurbuzbalaban, L. Zhu, U. Şimşekli, and M. A. Erdogdu, "Convergence Rates of Stochastic Gradient Descent under Infinite Noise Variance", *Advances in Neural Information Processing Systems*, Vol. 34, Vancouver, Canada, pp. 18866–18877, 2021.

160. London, B., "A PAC-Bayesian Analysis of Randomized Learning with Application to Stochastic Gradient Descent", *Advances in Neural Information Processing Systems*, Vol. 30, Long Beach, California, pp. 2931–2940, 2017.

161. Dziugaite, G. K., and D. M. Roy, "Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data", *arXiv:1703.11008*, 2017.

162. Kuzborskij, I., and C. H. Lampert, "Data-Dependent Stability of Stochastic Gradient Descent", *arXiv:1703.01678*, 2018.

163. Zhou, Y., Y. Liang, and H. Zhang, "Understanding Generalization Error of SGD in Nonconvex Optimization", *Machine Learning*, Vol. 111, No. 1, pp. 345–375, 2022.

164. Allen-Zhu, Z., Y. Li, and Z. Song, "On the Convergence Rate of Training Recurrent Neural Networks", *Advances in Neural Information Processing Systems*, Vol. 32, Vancouver, Canada, pp. 6676–6688, 2019.

165. Neu, G., "Information-Theoretic Generalization Bounds for Stochastic Gradient Descent", *arXiv:2102.00931*, 2021.

166. Camuto, A., G. Deligiannidis, M. A. Erdogdu, M. Gurbuzbalaban, U. Şimşekli, and L. Zhu, "Fractal Structure and Generalization Properties of Stochastic Optimization Algorithms", *Advances in Neural Information Processing Systems*, Vol. 34, Vancouver, Canada, pp. 18774–18788, 2021.

167. Birdal, T., A. Lou, L. J. Guibas, and U. Şimşekli, "Intrinsic Dimension, Persistent Homology and Generalization in Neural Networks", *Advances in Neural Information Processing Systems*, Vol. 34, Vancouver, Canada, pp. 6776–6789, 2021.

168. Şimşekli, U., L. Zhu, Y. W. Teh, and M. Gürbüzbalaban, "Fractional Underdamped Langevin Dynamics: Retargeting SGD with Momentum under Heavy-Tailed Gradient Noise", *International Conference on Machine Learning*, Vienna, Austria, pp. 8970–8980, 2020.

169. Camuto, A., X. Wang, L. Zhu, C. Holmes, M. Gurbuzbalaban, and U. Şimşekli, "Asymmetric Heavy Tails and Implicit Bias in Gaussian Noise Injections", *International Conference on Machine Learning*, Vienna, Austria, pp. 1249–1260, 2021.

170. Şimşekli, U., M. Gürbüzbalaban, T. H. Nguyen, G. Richard, and L. Sagun, "On the Heavy-Tailed Theory of Stochastic Gradient Descent for Deep Neural Networks", *arXiv:1912.00018*, 2019.

171. Martin, C. H., and M. W. Mahoney, "Traditional and Heavy-Tailed Self Regularization in Neural Network Models", *arXiv:1901.08276*, 2019.

172. Zhang, J., S. P. Karimireddy, A. Veit, S. Kim, S. Reddi, S. Kumar, and S. Sra, "Why are Adaptive Methods Good for Attention Models?", *Advances in Neural Information Processing Systems*, Vol. 33, Vancouver, Canada, pp. 21285–21296, 2020.

173. Chizat, L., and F. Bach, "On the Global Convergence of Gradient Descent for Over-parameterized Models Using Optimal Transport", *Advances in Neural Information Processing Systems*, Vol. 31, Montreal, Canada, pp. 3036–3046, 2018.

174. Sznitman, A.-S., "Topics in Propagation of Chaos", *Ecole d'été de Probabilités de Saint-Flour*, Saint-Flour, France, pp. 165–251, 1989.

175. Nguyen, P.-M., and H. T. Pham, "A Rigorous Framework for the Mean Field Limit of Multilayer Neural Networks", *arXiv:2001.11443*, 2021.

176. Jourdain, B., S. Méléard, and W. A. Woyczynski, "Nonlinear SDEs Driven by Lévy Processes and Related PDEs", *Alea*, Vol. 4, pp. 1–29, 2008.

177. Liang, M., M. B. Majka, and J. Wang, "Exponential Ergodicity for SDEs and McKean-Vlasov Processes with Levy Noise", *arXiv:1901.11125*, 2020.

178. Elesedy, B., V. Kanade, and Y. W. Teh, "Lottery Tickets in Linear Models: An Analysis of Iterative Magnitude Pruning", *arXiv:2007.08243*, 2020.

179. Han, S., J. Pool, J. Tran, and W. Dally, "Learning Both Weights and Connections for Efficient Neural Network", *Advances in Neural Information Processing Systems*, Vol. 28, Montreal, Canada, pp. 1135–1143, 2015.

180. Wang, C., G. Zhang, and R. Grosse, "Picking Winning Tickets Before Training by Preserving Gradient Flow", *arXiv:2002.07376*, 2020.

181. Yu, X., T. Liu, X. Wang, and D. Tao, "On Compressing Deep Models by Low Rank and Sparse Decomposition", *IEEE Conference on Computer Vision and*

Pattern Recognition, Honolulu, Hawaii, pp. 67–76, 2017.

182. Li, H., A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning Filters for Efficient ConvNets", *arXiv:1608.08710*, 2017.

183. He, Y., X. Zhang, and J. Sun, "Channel Pruning for Accelerating Very Deep Neural Networks", *arXiv:1707.06168*, 2017.

184. Barsbey, M., M. Sefidgaran, M. A. Erdogdu, G. Richard, and U. Şimşekli, "Source Code for the Paper: Heavy Tails in SGD and Compressibility of Overparametrized Neural Networks", 2021, `https://github.com/mbarsbey/sgd_comp_gen`, accessed on July 25, 2023.

185. LeCun, Y., C. Cortes, and C. Burges, "MNIST Handwritten Digit Database", 2010, `http://yann.lecun.com/exdb/mnist`, accessed on October 10, 2022.

186. Krizhevsky, A., "Learning Multiple Layers of Features from Tiny Images", 2009, `https://www.cs.toronto.edu/ kriz/learning-features-2009-TR.pdf`, accessed on October 10, 2022.

187. Mohammadi, M., A. Mohammadpour, and H. Ogata, "On Estimating the Tail Index and the Spectral Measure of Multivariate α-stable Distributions", *Metrika*, Vol. 78, No. 5, pp. 549–561, 2015.

188. Koh, P. W., S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao, T. Lee, E. David, I. Stavness, W. Guo, B. A. Earnshaw, I. S. Haque, S. Beery, J. Leskovec, A. Kundaje, E. Pierson, S. Levine, C. Finn, and P. Liang, "WILDS: A Benchmark of in-the-Wild Distribution Shifts", *arXiv:2012.07421*, 2021.

189. Hendrycks, D., and T. Dietterich, "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations", *arXiv:1903.12261*, 2019.

190. Szegedy, C., W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing Properties of Neural Networks", *arXiv:1312.6199*, 2014.

191. Raghunathan, A., S. M. Xie, F. Yang, J. Duchi, and P. Liang, "Understanding and Mitigating the Tradeoff Between Robustness and Accuracy", *arXiv:2002.10716*, 2020.

192. Croce, F., M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, and M. Hein, "RobustBench: A Standardized Adversarial Robustness Benchmark", *arXiv:2010.09670*, 2021.

193. Madry, A., A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards Deep Learning Models Resistant to Adversarial Attacks", *arXiv:1706.06083*, 2019.

194. Ortiz-Jimenez, G., A. Modas, S.-M. Moosavi-Dezfooli, and P. Frossard, "Optimism in the Face of Adversity: Understanding and Improving Deep Learning Through Adversarial Robustness", *Proceedings of the IEEE*, Vol. 109, No. 5, pp. 635–659, 2021.

195. Guo, Y., C. Zhang, C. Zhang, and Y. Chen, "Sparse DNNs with Improved Adversarial Robustness", *Advances in Neural Information Processing Systems*, Vol. 31, Montreal, Canada, pp. 242–251, 2018.

196. Suzuki, T., H. Abe, T. Murata, S. Horiuchi, K. Ito, T. Wachi, S. Hirai, M. Yukishima, and T. Nishimura, "Spectral Pruning: Compressing Deep Neural Networks via Spectral Analysis and Its Generalization Error", *International Joint Conference on Artificial Intelligence*, Yokohama, Japan, pp. 2839–2846, 2021.

197. Langenberg, P., E. R. Balda, A. Behboodi, and R. Mathar, "On the Effect of Low-Rank Weights on Adversarial Robustness of Neural Networks", *arXiv:1901.10371*, 2021.

198. Lee, K.-H., A. Arnab, S. Guadarrama, J. Canny, and I. Fischer, "Compressive

Visual Representations", *Advances in Neural Information Processing Systems*, Vol. 34, Vancouver, Canada, pp. 19538–19552, 2021.

199. Galanti, T., Z. S. Siegel, A. Gupte, and T. Poggio, "SGD and Weight Decay Provably Induce a Low-Rank Bias in Neural Networks", *arXiv:2206.05794*, 2023.

200. Timor, N., G. Vardi, and O. Shamir, "Implicit Regularization Towards Rank Minimization in ReLU Networks", *International Conference on Algorithmic Learning Theory*, Singapore, pp. 1429–1459, 2023.

201. Liao, R., A. Schwing, R. Zemel, and R. Urtasun, "Learning Deep Parsimonious Representations", *Advances in Neural Information Processing Systems*, Vol. 29, Barcelona, Spain, pp. 5076–5084, 2016.

202. Raghu, M., T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy, "Do Vision Transformers See Like Convolutional Neural Networks?", *Advances in Neural Information Processing Systems*, Vol. 34, Vancouver, Canada, pp. 12116–12128, 2021.

203. Sanyal, A., V. Kanade, P. H. S. Torr, and P. K. Dokania, "Robustness via Deep Low-Rank Representations", *arXiv:1804.07090*, 2020.

204. Huh, M., H. Mobahi, R. Zhang, B. Cheung, P. Agrawal, and P. Isola, "The Low-Rank Simplicity Bias in Deep Networks", *arXiv:2103.10427*, 2023.

205. Choi, D., K. Lee, D. Hwang, and W. Rhee, "Statistical Characteristics of Deep Representations: An Empirical Investigation", *Artificial Neural Networks and Machine Learning*, Cham, Switzerland, pp. 43–55, 2021.

206. Cui, Q., B. Zhao, Z.-M. Chen, B. Zhao, R. Song, B. Zhou, J. Liang, and O. Yoshie, "Discriminability-Transferability Trade-Off: An Information-Theoretic Perspective", *European Conference on Computer Vision*, Cham, Switzerland, pp. 20–37, 2022.

207. Roy, O., and M. Vetterli, "The Effective Rank: A Measure of Effective Dimensionality", *European Signal Processing Conference*, Poznan, Poland, pp. 606–610, 2007.

208. Hendrycks, D., "CIFAR-10-C and CIFAR-10-P", 2019, `https://zenodo.org/record/2535967`, accessed on July 25, 2023.

209. Hendrycks, D., "CIFAR-100-C", 2019, `https://zenodo.org/record/3555552`, accessed on July 25, 2023.

210. McInnes, C., "Virtual Screening Strategies in Drug Discovery", *Current Opinion in Chemical Biology*, Vol. 11, No. 5, pp. 494–502, 2007.

211. Mestres, J., E. Gregori-Puigjané, S. Valverde, and R. V. Solé, "Data Completeness–the Achilles Heel of Drug-Target Networks", *Nature Biotechnology*, Vol. 26, No. 9, pp. 983–984, 2008.

212. Bietz, S., K. T. Schomburg, M. Hilbig, and M. Rarey, "Discriminative Chemical Patterns: Automatic and Interactive Design", *Journal of Chemical Information and Modeling*, Vol. 55, No. 8, pp. 1535–1546, 2015.

213. Chaput, L., J. Martinez-Sanz, N. Saettel, and L. Mouawad, "Benchmark of Four Popular Virtual Screening Programs: Construction of the Active/Decoy Dataset Remains a Major Determinant of Measured Performance", *Journal of Cheminformatics*, Vol. 8, No. 1, p. 56, 2016.

214. Wallach, I., and A. Heifets, "Most Ligand-Based Classification Benchmarks Reward Memorization Rather than Generalization", *Journal of Chemical Information and Modeling*, Vol. 58, No. 5, pp. 916–932, 2018.

215. Sieg, J., F. Flachsenberg, and M. Rarey, "In Need of Bias Control: Evaluating Chemical Data for Machine Learning in Structure-Based Virtual Screening", *Journal of Chemical Information and Modeling*, Vol. 59, No. 3, pp. 947–961, 2019.

216. Chen, L., A. Cruz, S. Ramsey, C. J. Dickson, J. S. Duca, V. Hornak, D. R. Koes, and T. Kurtzman, "Hidden Bias in the DUD-E Dataset Leads to Misleading Performance of Deep Learning in Structure-Based Virtual Screening", *PLOS One*, Vol. 14, No. 8, p. e0220113, 2019.

217. Tran-Nguyen, V.-K., C. Jacquemard, and D. Rognan, "LIT-PCBA: An Unbiased Data Set for Machine Learning and Virtual Screening", *Journal of Chemical Information and Modeling*, Vol. 60, No. 9, pp. 4263–4273, 2020.

218. Yang, J., C. Shen, and N. Huang, "Predicting or Pretending: Artificial Intelligence for Protein-Ligand Interactions Lack of Sufficiently Large and Unbiased Datasets", *Frontiers in Pharmacology*, Vol. 11, pp. 1–9, 2020.

219. Boyles, F., C. M. Deane, and G. M. Morris, "Learning from the Ligand: Using Ligand-based Features to Improve Binding Affinity Prediction", *Bioinformatics*, Vol. 36, No. 3, pp. 758–764, 2020.

220. Özçelik, R., H. Öztürk, A. Özgür, and E. Ozkirimli, "ChemBoost: A Chemical Language Based Approach for Protein – Ligand Binding Affinity Prediction", *Molecular Informatics*, Vol. 40, No. 5, pp. 200–212, 2021.

221. Xiao, K., L. Engstrom, A. Ilyas, and A. Madry, "Noise or Signal: The Role of Image Backgrounds in Object Recognition", *arXiv:2006.09994*, 2020.

222. Shah, H., K. Tamuly, A. Raghunathan, P. Jain, and P. Netrapalli, "The Pitfalls of Simplicity Bias in Neural Networks", *Advances in Neural Information Processing Systems*, Vol. 33, Vancouver, Canada, pp. 9573–9585, 2020.

223. Geirhos, R., J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, "Shortcut Learning in Deep Neural Networks", *Nature Machine Intelligence*, Vol. 2, No. 11, pp. 665–673, 2020.

224. Arjovsky, M., L. Bottou, I. Gulrajani, and D. Lopez-Paz, "Invariant Risk Mini-

mization", *arXiv:1907.02893*, 2020.

225. Öztürk, H., E. Ozkirimli, and A. Özgür, "DeepDTA: Deep Drug-Target Binding Affinity Prediction", *Bioinformatics*, Vol. 34, No. 17, pp. i821–i829, 2018.

226. Yı lmaz, K., A. Cemgil, and U. Şimşekli, "Generalised Coupled Tensor Factorisation", *Advances in Neural Information Processing Systems*, Vol. 24, Granada, Spain, pp. 2051–2059, 2011.

227. Şimşekli, U., A. Liutkus, and A. T. Cemgil, "Alpha-Stable Matrix Factorization", *IEEE Signal Processing Letters*, Vol. 22, No. 12, pp. 2289–2293, 2015.

228. Gong, C., and Y. Zhang, "Urban Traffic Data Imputation With Detrending and Tensor Decomposition", *IEEE Access*, Vol. 8, pp. 11124–11137, 2020.

229. Stoehr, N., B. J. Radford, R. Cotterell, and A. Schein, "The Ordered Matrix Dirichlet for Modeling Ordinal Dynamics", *arXiv:2212.04130*, 2022.

230. Taylor, S. J., and B. Letham, "Forecasting at Scale", *The American Statistician*, Vol. 72, No. 1, pp. 37–45, 2018.

231. Bandara, K., R. J. Hyndman, and C. Bergmeir, "MSTL: A Seasonal-Trend Decomposition Algorithm for Time Series with Multiple Seasonal Patterns", *arXiv:2107.13462*, 2021.

232. Sen, R., H.-F. Yu, and I. S. Dhillon, "Think Globally, Act Locally: A Deep Neural Network Approach to High-Dimensional Time Series Forecasting", *Advances in Neural Information Processing Systems*, Vol. 32, Vancouver, Canada, pp. 4837–4846, 2019.

233. Lim, B., S. Ö. Arık, N. Loeff, and T. Pfister, "Temporal Fusion Transformers for Interpretable Multi-Horizon Time Series Forecasting", *International Journal of Forecasting*, Vol. 37, No. 4, pp. 1748–1764, 2021.

234. He, K., X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition", *arXiv:1512.03385*, 2015.

235. Awasthi, P., H. Jain, A. S. Rawat, and A. Vijayaraghavan, "Adversarial Robustness via Robust Low Rank Representations", *Advances in Neural Information Processing Systems*, Vol. 33, Vancouver, Canada, pp. 11391–11403, 2020.

236. Boschee, E., J. Lautenschlager, S. O'Brien, S. Shellman, J. Starz, and M. Ward, "ICEWS Coded Event Data, v28", 2015, `https://dataverse.harvard.edu/dataverse/icews`, accessed on December 16, 2020.

237. Schein, A., *Allocative Poisson Factorization for Computational Social Science*, Ph.D. Thesis, University of Massachusetts Amherst, 2019.

238. GDELT, "The GDELT Project", 2013, `https://www.gdeltproject.org/data.html`, accessed on July 25, 2023.

239. Liu, J.-p., H.-m. Hsueh, E. Hsieh, and J. J. Chen, "Tests for Equivalence or Non-Inferiority for Paired Binary Data", *Statistics in Medicine*, Vol. 21, No. 2, pp. 231–245, 2002.

240. McNemar, Q., "Note on the Sampling Error of the Difference between Correlated Proportions or Percentages", *Psychometrika*, Vol. 12, No. 2, pp. 153–157, 1947.

241. Fagerland, M. W., S. Lydersen, and P. Laake, "Recommended Tests and Confidence Intervals for Paired Binomial Proportions", *Statistics in Medicine*, Vol. 33, No. 16, pp. 2850–2875, 2014.

242. Aickin, M., and H. Gensler, "Adjusting for Multiple Testing When Reporting Research Results: The Bonferroni vs Holm Methods", *American Journal of Public Health*, Vol. 86, No. 5, pp. 726–728, 1996.

243. Tzagkarakis, G., J. P. Nolan, and P. Tsakalides, "Compressive Sensing of Temporally Correlated Sources Using Isotropic Multivariate Stable Laws", *European Signal Processing Conference*, Rome, Italy, pp. 1710–1714, 2018.

244. Sathe, A. M., and N. S. Upadhye, "Estimation of the Parameters of Multivariate Stable Distributions", *Communications in Statistics - Simulation and Computation*, Vol. 51, No. 10, pp. 5897–5914, 2022.

245. Paszke, A., S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library", *Advances in Neural Information Processing Systems*, Vol. 32, Vancouver, Canada, pp. 8024–8035, 2019.

APPENDIX A: BAYESIAN ALLOCATION MODEL

A.1. Basic Probability Distributions

Given the gamma function $\Gamma(z) = (z-1)!$ for nonnegative integer z, *Poisson distribution* can be expressed as

$$\mathcal{PO}(\mathrm{x};\lambda) = \frac{\lambda^{\mathrm{x}} e^{-\lambda}}{\Gamma(\mathrm{x}+1)}, \tag{A.1}$$

where λ is the rate parameter. *Gamma distribution* can be expressed as

$$\mathcal{GA}(\mathrm{x};a,b) = \frac{1}{\Gamma(a)b^a}\mathrm{x}^{a-1}e^{-\frac{\mathrm{x}}{b}}, \tag{A.2}$$

with shape and scale parameters a and b respectively. For $\mathbf{x} \in [0,1]^K$, with $\sum_{k=1}^{K} x_k = 1$, *Dirichlet distribution* can be expressed as

$$\mathcal{D}(\mathbf{x};\boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})}\prod_{n=1}^{N}\mathrm{x}_n^{\alpha_n-1}, \tag{A.3}$$

where $\boldsymbol{\alpha} = (\alpha_1,\ldots,\alpha_N)$ contain the concentration parameters, and $B(\boldsymbol{\alpha})$ is the *multivariate beta function*,

$$B(\boldsymbol{\alpha}) = \frac{\prod_n \Gamma(\alpha_n)}{\Gamma(\sum_n \alpha_n)}. \tag{A.4}$$

Given its parameters $\boldsymbol{\theta} \in [0,1]^K, \sum_k \theta_k = 1$, called event probabilities, and N, called number of trials, *multinomial distribution* can be written as

$$\mathcal{M}(\mathbf{x};N,\boldsymbol{\theta}) = \binom{N}{\mathrm{x}_1,\ldots,\mathrm{x}_K}\prod_{k=1}^{K}\theta_k^{\mathrm{x}_k}, \tag{A.5}$$

subject to $\mathrm{x}_1+\cdots+\mathrm{x}_K = N$. When $N=1$, multinomial distribution is called the *categorical distribution*.

A.2. Details of MFVI for BAM

The explicit evaluation of (1.18) and (1.19) implies the variational distributions

$$q_{\Phi_V}(\mathbf{S}) = \prod_{i_V}\mathcal{M}\big(\mathbf{S}(:,i_V);\mathbf{X}(i_V),\Phi_V(:,i_V)\big), \tag{A.6}$$

$$q(\lambda, \Theta) = q(\lambda)q(\Theta) = q(\lambda) \prod_{n=1}^{N} q(\theta_{n|\text{pa}(n)}), \tag{A.7}$$

where we write the parameters explicitly as subscripts when relevant. For (A.7) we have

$$q_{\hat{a}}(\lambda) = \mathcal{GA}(\lambda; \hat{a}, b+1), \tag{A.8}$$

$$q_{\hat{\alpha}}(\theta_{n|\text{pa}(n)}) = \prod_{i_{\text{pa}(n)}} \mathcal{D}(\theta_{n|\text{pa}(n)}(: |\text{pa}(n)); \hat{\alpha}_{\text{fa}(n)}(:, \text{pa}(n))), \ \forall n \in [N]. \tag{A.9}$$

Φ_V is a tensor that has same dimensionality with $\mathbf{S}$ and $\hat{\alpha}$ is a set of vectors / matrices / tensors with the dimensions identical to those in Θ. During inference, the variational parameter updates are made in an alternating fashion, such that

$$\hat{a} \leftarrow a + \mathbb{E}_{\mathcal{Q}}\left[\mathbf{S}_{+},\right] \tag{A.10}$$

$$\hat{\alpha}_{\text{fa}(n)}(:, i_{\text{pa}(n)})) \leftarrow \alpha_{\text{fa}(n)}(:, i_{\text{pa}(n)}) + \mathbb{E}_{\mathcal{Q}}\left[\mathbf{S}_{\text{fa}(n)}(:, i_{\text{pa}(n)})\right], \quad \forall n \in [N], \forall i_{\text{pa}(n)}, \tag{A.11}$$

$$\Phi_V(i_{1:N}) \leftarrow \frac{\exp\left(\sum_{n=1}^{N} \mathbb{E}_{\mathcal{Q}}\left[\log \theta_{n|\text{pa}(n)}(i_n|i_{\text{pa}(n)})\right]\right)}{\sum_{i_{\tilde{V}}} \exp\left(\sum_{n=1}^{N} \mathbb{E}_{\mathcal{Q}}\left[\log \theta_{n|\text{pa}(n)}(i_n|i_{\text{pa}(n)})\right]\right)}, \quad \forall i_{1:N}, \tag{A.12}$$

where $\mathbf{S}_{+} = \sum_{i_{[N]}} \mathbf{S}(i_{[N]})$, $\mathbf{S}_{\text{fa}(n)}$ corresponds to the sum of $\mathbf{S}$ over $[N] \backslash \text{fa}(n)$, and $\mathbf{S}_{\text{fa}(n)}(:, i_{\text{pa}(n)})$ corresponds to the slice/fiber/subtensor of this sum selected by $i_{\text{pa}(n)}$.

The expectations included above can be written as

$$\mathbb{E}_{\mathcal{Q}}\left[\lambda\right] = \hat{a}/(b+1), \tag{A.13}$$

$$\mathbb{E}_{\mathcal{Q}}\left[\log \lambda\right] = \psi(a + \mathbb{E}_{\mathcal{Q}}\left[\mathbf{S}_{+}\right]) - \log(b+1), \tag{A.14}$$

and

$$\mathbb{E}_{\mathcal{Q}}\left[\theta_{n|\text{pa}(n)}(i_n \mid i_{\text{pa}(n)})\right] = \psi(\hat{\alpha}_{\text{fa}(n)}(i_n, i_{\text{pa}(n)})) - \psi(\hat{\alpha}_{\text{pa}(n)}(i_{\text{pa}(n)})), \tag{A.15}$$

$$\mathbb{E}_{\mathcal{Q}}\left[\mathbf{S}_{\text{fa}(n)}\right] = \left(\mathbb{E}_{\mathcal{Q}}\left[\mathbf{S}\right]\right)_{\text{fa}(n)}, \tag{A.16}$$

$$\mathbb{E}_{\mathcal{Q}}\left[\mathbf{S}(i_{[N]})\right] = \mathbf{X}(i_V)\Phi_V(i_{[N]}), \tag{A.17}$$

with ψ corresponding to the *digamma* function. The update procedure described above requires $\mathbb{E}_{\mathcal{Q}}\left[\mathbf{S}_{\text{fa}(n)}\right]$, i.e. *marginal* statistics of $\mathbb{E}_{\mathcal{Q}}\left[\mathbf{S}\right]$. This in turn requires access to $\Phi_V(i_{[N]}), \forall i_{[N]}$. While explicitly storing Φ_V might be potentially prohibitive, since $\Phi_V(i_{[N]})$ respects the factorization implied by the graphical model in question, we can compute the marginal statistics $\mathbb{E}_{\mathcal{Q}}\left[\mathbf{S}_{\text{fa}(n)}\right]$ with the junction tree algorithm, which

drastically reduces the memory requirements for the overall procedure [33].

The updates can be stopped as the ELBO converges to a local optimum. Given that it has no guarantees to converge to a global minimum, it is a good practice to start this procedure with different random initializations and pick the solution with highest ELBO. The resulting ELBO can be and is frequently used for model selection as described above. The full expression for ELBO can be written as

$$\begin{aligned}\mathcal{B}_{\mathcal{P}}[\mathcal{Q}] =& a\log b - (a + \mathbb{E}_{\mathcal{Q}}[\mathbf{S}_+])\log(b+1) + \log\Gamma(a + \mathbb{E}_{\mathcal{Q}}[\mathbf{S}_+]) - \log\Gamma(a) \\ &+ \sum_{n=1}^{N}\sum_{i_{\mathrm{pa}(n)}} \log\Gamma(\alpha_{\mathrm{pa}(n)}(i_{\mathrm{pa}(n)})) - \sum_{n=1}^{N}\sum_{i_{\mathrm{fa}(n)}} \log\Gamma(\alpha_{\mathrm{fa}(n)}(i_{\mathrm{fa}(n)})) \\ &- \sum_{n=1}^{N}\sum_{i_{\mathrm{pa}(n)}} \log\Gamma(\alpha_{\mathrm{pa}(n)}(i_{\mathrm{pa}(n)}) + \mathbb{E}_{\mathcal{Q}}\left[\mathbf{S}_{\mathrm{pa}(n)}\right](i_{\mathrm{pa}(n)})) \\ &+ \sum_{n=1}^{N}\sum_{i_{\mathrm{fa}(n)}} \log\Gamma(\alpha_{\mathrm{fa}(n)}(i_{\mathrm{fa}(n)}) + \mathbb{E}_{\mathcal{Q}}\left[\mathbf{S}_{\mathrm{fa}(n)}\right](i_{\mathrm{fa}(n)})) \\ &- \sum_{i_V} \log\Gamma(\boldsymbol{X}(i_V)+1) - \sum_{i_{[N]}} \mathbb{E}_{\mathcal{Q}}\left[\mathbf{S}(i_{[N]})\right]\log\Phi_V(i_{[N]}),\end{aligned} \tag{A.18}$$

as shown by [33]. The reader is encouraged to consult [33] and [35] for a more in-depth treatment of variational inference under these modeling assumptions.

A.3. Analyzing ICEWS Dataset with BAM

We now present a set of experiments where we analyze a well known international relations dataset using BAM [33]. The experiments compare the marginal likelihood estimates produced by the mean-field variational inference (MFVI) and sequential Monte Carlo (SMC) estimators on different representations of this dataset. See Section 1.3.2 for a review of these two methods, and see [33] and [35] for more detailed accounts of these inference methods and their respective advantages.

ICEWS [236] is a popular international relations dataset that comprises of discrete events which are coded according to the CAMEO event coding scheme (see [237] for a recent example of its analysis with NTF approaches). At its core, each event includes

a source actor, a target actor, an action type, and a timestamp. Sources and targets might have various affiliations, and action types are represented at various levels of abstraction.

In our experiments, we consider only the national affiliations of actors: therefore we construe actions to be taking place between source and target *nations*. Further, we consider the action types with respect to the highest level of abstraction, namely the *quad class* [238], which classifies all actions according to four groups: *Verbal Cooperation, Material Cooperation, Verbal Conflict*, and *Material Conflict*. Accordingly, an example (simplified) event would be: *(Russia, India, Verbal Cooperation, 2017/06/12)*. Finally, we limit our attention to the year 2017 and focus on the relationships among the 20 most active nations.

A.3.1. Results

We first compare MFVI and SMC for model scoring under different sparsity levels. To create different levels of sparsity using the same data, we rely on a weekly aggregation of the event data. Let $A(i, j, k, w)$ be the number of actions of type k taken by country i on country j in week w. We create the $20 \times 20 \times 4$ (country $\times$ country $\times$ action type) data tensor $\mathbf{X}$ as

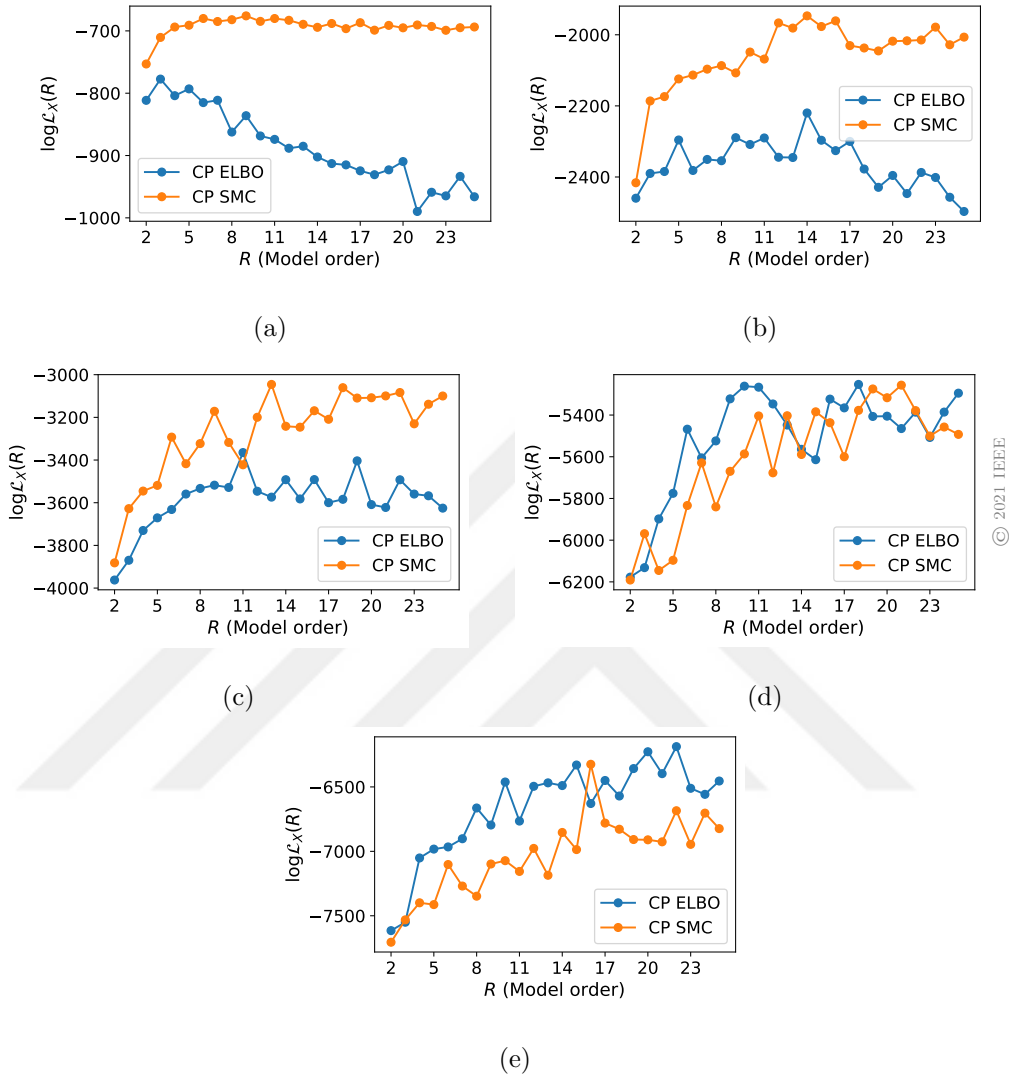
$$\mathbf{X}(i, j, k) = \sum_{w=1}^{52} \mathbb{I}(A(i, j, k, w) > c_k), \tag{A.19}$$

that is, $\mathbf{X}(i, j, k)$ is the number of weeks the country i conducted action k on country j more than c_k times.

To obtain data tensor of various sparsity levels, we varied c_k to be equal to 50'th, 75'th, 90'th, 95'th, or 99'th percentile of the empirical distribution of $\{A(i, j, w, k) : i, j, w\}$, i.e., all weekly occurrences of the action k. As a result, we obtained five tensors, $\mathbf{X}_{50}$, $\mathbf{X}_{75}$, $\mathbf{X}_{90}$, $\mathbf{X}_{95}$, $\mathbf{X}_{99}$, which have the same dimensions but are progressively sparser.

We use the CP model to analyze the data. For SMC, for each setting, the number of particles was linearly scaled with number of observations, and the average of 10 runs was taken as the final estimate. For MFVI, for each setting, the largest ELBO out of 10 runs was taken the final estimate of the marginal likelihood.

Figure A.1 demonstrates the comparison of marginal likelihood approximation using MFVI and SMC with the latent dimension ranging from 2 to 25, with the hyperparameter a fixed as $a = 1$ and α is chosen as in (1.23) yielding a flat prior. The results show that SMC becomes more likely to dominate MFVI as the sparsity of the data increases. Note the fact that the true marginal likelihood is always greater than the MFVI estimate and that the SMC estimator is unbiased. Hence, a lower estimate by SMC than the MFVI estimate, observed for denser tensors, can be attributed to the increased variance of the SMC estimator in the number observations and that the distribution of the SMC estimator is skewed to the left. Thus, the results imply that in datasets with a larger number of observations, MFVI is the more preferable model scoring approach due to the high variance exhibited by the SMC estimator in its current incarnation.



© 2021 IEEE

Figure A.1. Comparing MFVI and SMC in terms of log-marginal likelihood estimates vs R at different tensor sparsities: (a) $\mathbf{X}_{99}$ (Density = 0.07), (b) $\mathbf{X}_{95}$ (Density = 0.20), (c) $\mathbf{X}_{90}$ (Density = 0.32), (d) $\mathbf{X}_{75}$ (Density = 0.53), (e) $\mathbf{X}_{50}$ (Density = 0.67) where *density* stands for $\sum_{i,j,k} \mathbb{I}(\mathbf{X}(i,j,k) > 0)/IJK$. Figure from [33].

APPENDIX B: LOW-RANK HIERARCHICAL SEASONALITY

B.1. Additional Details on NYC YT Experiments

For NYC Yellow Taxi data, which has the dimensions $265 \times 265 \times 24 \times 7 \times 25$, we select among latent ranks $R = (R_{\text{loc}}, R_{\text{loc}}, R_{\text{time}}, R_{\text{time}}, R_{\text{week}})$ for Tucker decomposition models, where $R_{\text{loc}} \in \{2, 3, 4\}$, $R_{\text{time}} \in \{2, 3, 4, 5\}$, and $R_{\text{week}} \in \{2, 3\}$. The model with the highest ELBO had $R = (4, 4, 5, 2, 2)$ and $a = 1e3$. Our experiments showed that further increasing the latent dimensions increased ELBO *only* for spatial dimensions, which are not of interest for this experiment. Therefore we upper bounded R_{loc} to 4 for more expedient inference.

B.2. Additional Details on BART Experiments

For BART ridership data the observed tensor dimensions were $50 \times 50 \times 24 \times 7 \times 53 \times 12$. We select among Tucker decomposition models with latent ranks $R = (R_{\text{loc}}, R_{\text{loc}}, R_{\text{time}}, R_{\text{time}}, R_{\text{time}}, R_{\text{year}})$, where $R_{\text{loc}} \in \{2, 3, 4\}, R_{\text{time}} \in \{2, 3, 4, 5\}$, and $R_{\text{year}} \in \{2, 3\}$. The chosen model had $R = (4, 4, 4, 2, 4, 2)$ and $a = 1e3$.

Note that we use a cardinality of 53 for the week of the year variable in our experiments. This is due to the irregularity of weeks according to years. ISO 8601's week date system prescribes some years to include an additional "leap week", making the total 53 for these select years. Although assuming 52 weeks and conducting cyclical folding is acceptable for the purposes of the univariate experiments in [36], for the experiments presented here we opted to observe consistency with calendar years, so that our Covid-related results are easier to interpret.

Lastly, the mapping we used between station names and station indices while presenting BART readership analyses in Figure 9 can be found in Table B.1.

Table B.1. BART station mappings. Table from [36] © 2023 IEEE.

Code	Station	Code	Station	Code	Station	Code	Station	Code	Station
12TH	0	16TH	1	19TH	2	24TH	3	ASHB	4
BAYF	5	CIVC	6	COLS	7	CONC	8	DALY	9
DBRK	10	DELN	11	EMBR	12	FRMT	13	FTVL	14
GLEN	15	LAFY	16	LAKE	17	MCAR	18	MLBR	19
MONT	20	PHIL	21	PITT	22	POWL	23	ROCK	24
SBRN	25	SSAN	26	UCTY	27	WOAK	28	BALB	29
DUBL	30	NBRK	31	ORIN	32	WCRK	33	NCON	34
RICH	35	SANL	36	COLM	37	HAYW	38	SHAY	39
CAST	40	PLZA	41	SFIA	42	WDUB	43	ANTC	44
WARM	45	PCTR	46	OAKL	47	BERY	48	MLPT	49

B.3. Additional Details on Missing Data Imputation Experiments

For the Guangzhou traffic data in the imputation experiments, which had the dimensions $214 \times 61 \times 144$, we use a CP decomposition and search among $R = \{5, 25, 50, 150, 300, 450\}$, corresponding to a range of 1000-fold to 10-fold parameter decrease. The model selection is conducted based on a held-out dataset that corresponds to 0.01 of the total cells, from among the uncensored data. The held-out dataset is used for early stopping the variational inference procedure, and the RMSE of the best epoch is used to compare models between latent ranks and hyperparameters. The selected hyperparameters can be seen in Table B.2.

Similar to [91] we experiment with different data representations, and go up to 4 temporal indices creating a 5-order tensor with dimensions $214 \times 6 \times 24 \times 7 \times 9$. The cardinality 6 in the second mode is due to the observations having been made with 10 minute intervals. For this specific task, like [91] we find that a 3-order tensor with dimensions $214 \times 61 \times 144$ produces the best results, according to which we present our findings. This is not necessarily surprising given the nature of this data: there is no inherent reason for traffic speed to have an hourly seasonality. The opposite can be expected in other contexts. For example, in a dataset that records elevator use in

a company, half and full hours might see increased travel between meeting rooms and offices due to meetings starting and ending, leading to a strong hourly seasonality.

Table B.2. Selected hyperparameters for the missing data imputation experiments. Table from [36] © 2023 IEEE.

	Random Missing				
Missing Ratio	0.1	0.2	0.3	0.4	0.5
R	450	450	450	450	450
a	6×10^4	6×10^4	7.5×10^4	6×10^4	6×10^4
	Correlated Missing				
Missing Ratio	0.1	0.2	0.3	0.4	0.5
R	35	45	40	40	40
a	1.1×10^4	1×10^4	5×10^4	12.5×10^4	12.5×10^4

APPENDIX C: LEARNING TO DEFER

C.1. Additional Details on Density Estimation

As noted in Section 3.2, the optimal decision making procedure requires $p(\mathrm{z}, \mathrm{h}|\mathrm{y})$ or an estimate thereof, and we use a discretized KDE method for this task [119]. We now describe our estimation procedure in further detail.

C.1.1. Discretization

As mentioned above, our method relies on a discretization of the output space to produce a histogram, in order to conduct efficient density estimation. To do this, based on D_{tune} we compute a histogram of model outputs z for $h = 0$ and $h = 1$ and $y = 0$ and $y = 1$, with T number of bins. We represent these histograms with the count vector $\mathbf{b}^{h,y} \in \mathbb{Z}^T_{\geq 0}$, such that e.g. the histogram of z's for when $\mathrm{h} = 0, \mathrm{y} = 1$ is represented with the vector $\mathbf{b}^{0,1}$.

For computing the histogram, we divide the interval $[0, 1]$ into T bins of equal width $I_t = [(t-1)/T, t/T]$ with $t = 1, ..., T$. We add the prior pseudocount hyperparameter $\kappa \geq 0$ to the counts, so as to avoid assigning 0 counts to certain bins even when data is highly sparse. We then compute the elements of these histogram vectors by

$$b_t^{h,y} = \kappa + \sum_{i=1}^{N_{\text{tune}}} \mathbb{I}[\mathrm{z}_{(i)} \in I_t, \mathrm{h}_{(i)} = h, \mathrm{y}_{(i)} = y], \tag{C.1}$$

where the function $\mathbb{I}[\cdot]$ tests for the truth of the expression inside, N_{tune} is the number of observations in D_{tune}, and the subscript (i) is used to denote distinct observations instead of indexing a vector.

C.1.2. Smoothing

We use a Gaussian kernel to smooth the histograms produced above, where we apply a discrete convolution to the vectors $\mathbf{b}^{h,y}$, with σ as the smoothing bandwidth of the Gaussian kernel. We denote the smoothed $\mathbf{b}^{h,y}$ with the vector $\widehat{\mathbf{b}}^{h,y}$.

C.1.3. Normalization

After smoothing, $\widehat{\mathbf{b}}^{h,y}$ can be treated as unnormalized discrete probability density functions. Lastly, we normalize these vectors so that they can be used as density estimates. We conduct the normalization by computing

$$\bar{\mathbf{b}}^{h,y} = \bar{\mathbf{b}}_t^{h,y} / \sum_{t,h} \bar{\mathbf{b}}_t^{h,y}. \tag{C.2}$$

After this normalization, we can use the normalized vectors as the density estimates such that

$$p(\mathrm{z} = z, \mathrm{h} = h | \mathrm{y} = y) \approx \bar{\mathbf{b}}_t^{h,y}, \tag{C.3}$$

where $z \in I_t$. Note that the normalization over h in (C.2) also integrates the estimation of $p(\mathrm{h}|\mathrm{y})$ to this procedure, resulting in the joint conditional at (C.3). In cases where prior information regarding $p(\mathrm{h}|\mathrm{y})$ exists, this information can be explicitly used, and normalization in (C.2) can be done without including h.

C.1.4. Overall Density Smoothing Procedure

To summarize, below we provide our density estimation algorithm in full:

(i) Compute the histogram of z for $h, y \in \{0, 1\}$ producing $\mathbf{b}^{h,y}$.
(ii) Smooth the histograms to produce $\widehat{\mathbf{b}}^{h,y}$.
(iii) Normalize $\widehat{\mathbf{b}}^{h,y}$ to produce $\bar{\mathbf{b}}^{h,y}$.
(iv) Use $\bar{\mathbf{b}}^{h,y}$ an estimate of $p(\mathrm{z} = z, \mathrm{h} = h | \mathrm{y} = y)$.

We provide an implementation of our method in the Python programming language. The discretization we apply allows convenient vectorization/parallelization, which we achieve through the use of the JAX computational framework [104]. Note that the step 1 above involves the hyperparameters T and κ (number of bins and pseudocounts), and step 2 involves the hyperparameter σ (smoothing bandwidth). The model selection procedure to determine these hyperparameters is described next.

C.2. Hyperparameter Optimization and Model Selection

In Section 3.2 we describe how the deferral function is obtained given an objective function trade-off parameter λ, operating point θ (representing the decision function δ), and algorithmic hyperparameters: pseudocounts κ, number of bins T, and the smoothing bandwidth σ. We view $\lambda, \theta, \kappa, T, \sigma$ as the hyperparameters of CoDoC and perform a grid-based hyperparameter search to determine these. We first do a parameter sweep over the hyperparameters above, and produce deferral models for each combination (see the source code provided for the hyperparameter ranges we used in our results [104]). The parameter sweep produces a pool of deferral algorithms trained on D_{tune}, we call this pool `DeferralModels`. To make sure the models we choose make selective use of clinical expertise, we remove any model that defers $> 50\%$ of cases either in D_{tune} or D_{val} from `DeferralModels`.

Then, model selection among `DeferralModels` is conducted on D_{val} according to the final criterion of interest. In particular, we consider:

(i) Obtaining sensitivity superiority and specificity non-inferiority over the baselines (AI-only and human-only)
(ii) Obtaining specificity superiority and sensitivity non-inferiority over the baselines (AI-only and human-only)

The algorithm for model selection for either of these objectives conform to the following procedure for choosing a CoDoC model, e.g. a model that is superior on

specificity and non-inferior on sensitivity to the diagnostic AI and clinical workflow:

(i) Filter out models from among `DeferralModels` that are inferior on sensitivity (by a margin of 1% or more) to either the diagnostic AI or clinical workflow according to D_{tune} or D_{val} to find a set of viable models.

(ii) Set $k \geq 2$ to the smallest value such that there are enough viable models with at most k zero-crossings in the advantage function $\text{Adv}(z)$). Note that $k = 2$ corresponds to a model with a single deferral region, so we prefer the simplest (in terms of number of deferral regions) viable model. If there are not enough viable models to choose from (e.g. < 10), increase k iteratively until there are.

(iii) Select the model that performs the best according to specificity on D_{val}, ignoring the top 1% of models to avoid overfitting the validation set.

A similar approach (interchanging sensitivity and specificity above) is used to choose a model that is superior on sensitivity and non-inferior on specificity. The margin of 1% is chosen as a conservative value since we ultimately test for non-inferiority with a margin of 5%, following prior work (see Appendix C.3). This margin is increased up to 5% when choosing models for sensitivity, given the loss of statistical power due to the scarcity of positive cases. In data-rich regimes (e.g. UK Mammography Dataset 1), the estimates from the empirical distribution are much more reliable, thus these margins are ignored, except for a 2.5% specificity inferiority allowance when selecting for sensitivity superiority, again for increased statistical power. At the end of this procedure, we obtain a single CoDoC model (along with its operating point and deferral regions) that is chosen to be evaluated on the test set, as outlined in Appendix C.3. Note that this model selection procedure is flexible: it can be conducted for any desired combination of sensitivity and specificity, and any newly introduced hyperparameters can be included in this procedure, such as different nonlinear transformations of the confidence scores.

C.3. Details of the Hypothesis Tests

In our experiments, the performance of CoDoC is compared to that of a standalone diagnostic AI system (machine learning model) and clinical experts. To establish non-inferiority we use the Wald test [239], and the McNemar test [240] to establish superiority, where we use a statistical significance threshold of, i.e. an α value of 0.05. For non-inferiority comparisons, we specify a 5% absolute margin before the inspection of the test set. We utilize Wald intervals to characterize the confidence intervals on the differences [241]. Our primary comparisons number 28 in total. For each setting, we compare the sensitivity and specificity of CoDoC to both the sensitivities and specificities of the standalone diagnostic AI system and the clinical experts. All p values survived Holm–Bonferroni correction for multiple comparisons within a setting [242].

Table C.1. Deferral from one machine learning model to an expert using ImageNet-16H dataset, using SMC algorithm.

Model	Expert	Mod. acc.	Exp. acc.	Comp. acc.
ResNet152	GoogleNet	0.428	0.429	0.45
ResNet152	DenseNet161	0.428	0.474	0.476
GoogleNet	ResNet152	0.429	0.428	0.476
GoogleNet	DenseNet161	0.429	0.474	0.495
DenseNet161	GoogleNet	0.474	0.429	0.489
DenseNet161	ResNet152	0.474	0.428	0.489

Table C.2. Deferral from two models to an expert using ImageNet-16H dataset, using SMC algorithm.

Model 1	Model 2	Expert	Baseline	Comp. acc.
ResNet152	GoogleNet	DenseNet161	0.476	0.496
DenseNet161	ResNet152	GoogleNet	0.489	0.469
GoogleNet	DenseNet161	ResNet152	0.495	0.509

C.4. Results for Experiments with SMC

As noted in Section 4.4.3, in Tables C.1 and C.2 we present learning to defer experiment results with the SMC estimator [33], instead of MFVI. While overall the results demonstrate a similar pattern, SMC's performance is surpassed by MFVI in two model, one expert settings.

APPENDIX D: COMPRESSIBILITY AND GENERALIZATION

D.1. Details on Experiment Settings

Here we provide additional details on the experiments conducted in Chapter 5.

D.1.1. Datasets

The experiments in Chapter 5 involved a supervised learning setting, where images are classified into a number of predefined class labels. Each model architecture with given hyperparameters were trained on MNIST [185], CIFAR-10, and CIFAR-100 [186] datasets. The MNIST dataset includes 28×28 black and white handwritten digits, with digits ranging from 0 to 9. The dataset in its original form includes 60000 training and 10000 test samples. CIFAR-10 and CIFAR-100 are also image classification datasets comprising 32×32 color images of objects or animals, making up 10 and 100 classes respectively. There are 50000 training and 10000 test images in either of these datasets, and the instances are divided among classes equally. We used the standard train-test splits in all datasets.

D.1.2. Models

We used three different architectures in our experiments: a fully connected network with 4 hidden layers (FCN4), another fully connected network with 6 hidden layers (FCN6), and a convolutional neural network (CNN). In both FCN architectures, all hidden layer widths were 2048. All architectures featured ReLU activation functions. Batch normalization, dropout, residual layers, or any explicit regularization term in the loss function were not used in any part of the experiments. None of the architectures in question included bias nodes. The architecture we chose for our CNN model closely follows that of VGG11 model [134], with the significant difference that only

a single linear layer with a softmax output follows the convolutional layers presented below:

$$64, M, 128, M, 256, 256, M, 512, 512, M, 512, 512, M. \tag{D.1}$$

Here, integers describe the number of filters for 2-dimensional convolutional layers - for which the kernel sizes are 3×3. M stands for 2×2 max-pooling operations with a stride value of 2. This architecture was slightly modified for the MNIST experiments by removing the first max-pooling layer due to the smaller dimensions of the MNIST images. The Table D.1 describes the number of different parameters used for each model-dataset combination.

Table D.1. Number of parameters for model-dataset combinations [55].

	FCN4	FCN6	CNN
MNIST	14,209,024	22,597,632	9,221,696
CIFAR-10	18,894,848	27,283,456	9,222,848
CIFAR-100	18,899,456	27,288,064	9,227,456

D.1.3. Training and Hyperparameters

As described in the main text, the models were trained with SGD until convergence on the training set. The convergence criteria for MNIST and CIFAR-10 is a training negative log-likelihood (NLL) of $< 5 \times 10^{-5}$ and a training accuracy of 100%, and for CIFAR-100 these are a NLL of $< 1 \times 10^{-2}$ and a training accuracy of $> 99\%$. We use two different batch sizes ($b = 50, 100$) and a diversity of learning rates (η) to generate a large range of η/b values. Table D.2 presents the η/b ranges created for each experiment setting. The varying nature of these ranges are due to the fact that different η/b values might lead to heavy-tailed behavior or divergence under different points in this hyperparameter space. The public source code for [55], shared in [184], includes the enumerations of specific combinations of these hyperparameters for MNIST and CIFAR-10 experiments, and [60] includes those for CIFAR-100.

Table D.2. The ranges of η/b for all experiments [55].

	FCN4	FCN6	CNN
MNIST	5×10^{-5} to 1.14×10^{-2}	5×10^{-5} to 8.8×10^{-3}	1×10^{-5} to 6.35×10^{-3}
CIFAR-10	5×10^{-5} to 2.7×10^{-3}	2.5×10^{-5} to 4×10^{-3}	1×10^{-5} to 1.5×10^{-3}
CIFAR-100	1×10^{-5} to 1.6×10^{-3}	1×10^{-5} to 2.25×10^{-3}	1×10^{-5} to 7×10^{-4}

D.1.4. Tail Index Estimation

We use the following multivariate tail index estimator proposed by [187].

Theorem 1 ([187, Corollary 2.4]). *Let $\{\mathbf{x}_i\}_{i=1}^{K}$ be a collection of i.i.d. random vectors where each $\mathbf{x}_i$ is multivariate strictly stable with tail index α, and $K = K_1 \times K_2$. Define $\mathbf{y}_i \triangleq \sum_{j=1}^{K_1} \mathbf{x}_{j+(i-1)K_1}$ for $i \in [K_2]$. Then, the estimator*

$$\widehat{\frac{1}{\alpha}} \triangleq \frac{1}{\log K_1}\Big(\frac{1}{K_2}\sum_{i=1}^{K_2}\log\|\mathbf{y}_i\| - \frac{1}{K}\sum_{i=1}^{K}\log\|\mathbf{x}_i\|\Big). \tag{D.2}$$

converges to $1/\alpha$ almost surely, as $K_2 \to \infty$.

This estimator has been used in previous research such as [170] and [243]. We center the observations using the median values before the estimation. Using the alternative univariate tail index estimator [187, Corollary 2.2] in the same paper has no qualitative effects on our results, an additional benefit of our choice is additional analyses it makes possible as presented in Section D.3. Comparisons with alternative tail index estimators with symmetric α-stable assumption revealed no dramatic differences between various estimators [244].

D.1.5. Pruning Details

We first provide a review of the pruning methods we use. All three notions of pruning in our experiments correspond to the magnitude-wise ordering of certain parameters and the "pruning" of a certain ratio of parameters that correspond to smallest magnitudes (Note that a more relaxed definition of pruning would be "systematic removal of model parameters" to allow for different scoring methods in pruning [136].

However, we proceed with our specific definition since this allows us to communicate our results more concisely). When the parameters that are pruned are the weight parameters themselves, this corresponds to magnitude-based pruning or *magnitude pruning* as known in the literature, which can be conducted layer-wise or globally [136]. *Singular value pruning*, as described here, corresponds to pruning of the smallest singular values (and, by implication, the related singular vectors) in the SVD of specific layers. To apply the SVD to CNN layers, we reshape the parameter tensors into matrices of shape (# channels) $\times$ (# filters $\times 3 \times 3$). Lastly *node pruning* corresponds to the pruning of the whole columns in the weight matrices. Similarly, the CNN counterpart of node pruning is open to interpretation; we choose to prune specific kernels according to the their norms.

Before any pruning is done, the parameters to be pruned are centered with the estimated median of the observations, in order to conform with our tail index estimation methodology. We chose median due to its robustness against extreme observations especially with a small sample - however our results were qualitatively unchanged when the mean was used in the centering. After the pruning (in all three methods), the median was added to the pruned parameter vectors before testing the performance of the resulting model. Note that the median (or mean) was usually very small in norm and omitting centering made no qualitative effect on the results.

Lastly, while *remaining parameter ratio* (κ) or *pruning ratio* ($1 - \kappa$) are easy to interpret in the case of magnitude pruning or node pruning, in SVD κ would equal (number of singular value and vector parameters left after pruning) / (number of weight parameters in the layer), and pruning ratio would be determined accordingly.

D.2. Investigation of Tail Index and Prunability with Synthetic Data

Experimental manipulation of tail indices directly in neural networks trained on real data is hard to formulate and realize. However, we conduct a number of experiments with synthetically generated networks and data to further test our hypotheses.

In this setting, we created neural networks that were structural analogues of FCN6 networks presented in the original experiments, that is, feed-forward neural networks of 6 hidden layers and a width of 2048 units for each hidden layer, with ReLU activation functions after each hidden layer. For the experiments, for each $\alpha_i \in \{1.50, 1.55, 1.60, \ldots, 2.00\}$, we randomly sampled the parameters of these networks independently from a $\mathcal{S}\alpha\mathcal{S}$ distribution with an α_i tail index parameter. After the sampling of the layer parameters, each layer was converted to a unit length vector, in order to avoid the possibly confounding effects of scale between different tail indices. This procedure excluded the much smaller final layer, which was sampled as in the initialization of the original experiments.

We also created random data for these experiments, in the shape of the MNIST training data, sampled independently from a standard normal distribution. For each network, the labels for these data were created by passing the synthetic data through the synthetic network and choosing the label with the maximum final value. Then, for each network created, we conducted layer-wise magnitude pruning for different values of κ, and evaluated the performance of the pruned versions of the networks with their original performances. This was repeated 10 times, and Figure D.1 presents the mean of the accuracy values for each of the existing tail index (α) and pruning ratio ($1 - \kappa$) combinations.

The results again support our hypothesis: networks with lower tail indices are more prunable, that is, more robust to pruning in terms of performance decay. This parallels the conclusion of the original experiments, where a similar conclusion was reached with networks trained on real data. We leave the extension of these experiments for different data generation schemes, layer structures, and pruning types to future work.

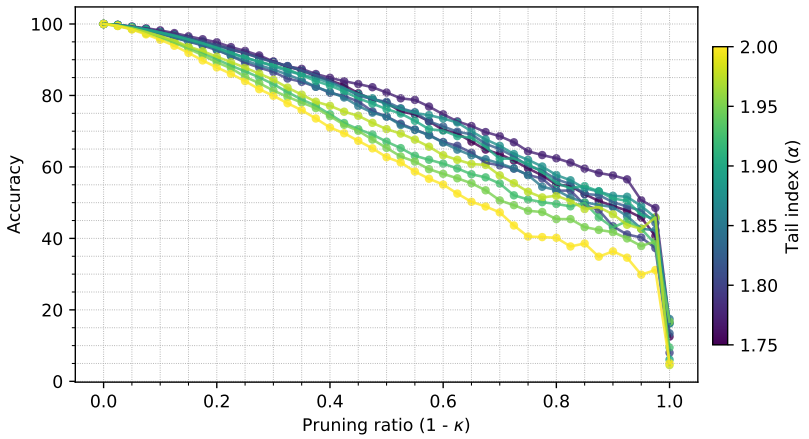


Figure D.1. Accuracy and pruning ratio $(1-\kappa)$ for synthetically generated networks and data [55].

D.3. Investigating the HML Condition in Synthetic Experiments

Recall that in Figure 5.3 we examined, for a given relative compression error, whether lower tail index is associated with higher prunability. The results demonstrated that this was indeed the case. Here we compare our empirical results with some synthetic results to get additional insights regarding whether HML condition is actually observed in our networks.

For this experiment, we randomly sample tail indices $\alpha_i \sim \mathcal{U}(1.75, 2)$, where $i \in \{1, \dots, 250\}$ and $\mathcal{U}$ stands for the uniform distribution. Then for each α_i, we sample three different "weight matrices": $\mathbf{W}_{\text{ind},i}, \mathbf{W}_{\text{col},i}, \mathbf{W}_{\text{lay},i} \in \mathbb{R}^{500\times500}$. The elements of $\mathbf{W}_{\text{ind},i}$ are sampled independently from a $\mathcal{S}\alpha\mathcal{S}$ with tail index α_i; this corresponds to the case where weight parameters are statistically independent as prescribed by the HML condition. On the other hand, columns of the $\mathbf{W}_{\text{col},i}$ are independently sampled from a 500-dimensional multivariate *elliptically contoured* $\mathcal{S}\alpha\mathcal{S}$ with tail index α_i. A d-dimensional elliptically contoured multivariate $\mathcal{S}\alpha\mathcal{S}$ has the characteristic function

$$\mathbb{E}\left[\exp(i\langle\omega, \mathrm{x}\rangle)\right] = \exp(-\|\omega\|^{\alpha}), \tag{D.3}$$

where $\mathrm{x}, \omega \in \mathbb{R}^d$ and $\langle\cdot,\cdot\rangle$ stands for inner product. This means that while the columns of the matrix are independent, column elements can be correlated. Lastly, all elements

of $\mathbf{W}_{\text{lay},i}$ are sampled from a (500×500)-dimensional elliptically contoured multivariate $\mathcal{S}\alpha\mathcal{S}$, creating a case where all elements of the matrix can be correlated. We repeat the analysis presented in Figure 5.3 for all three sets of sampled synthetic layer weights, and present the results in Figure D.2.

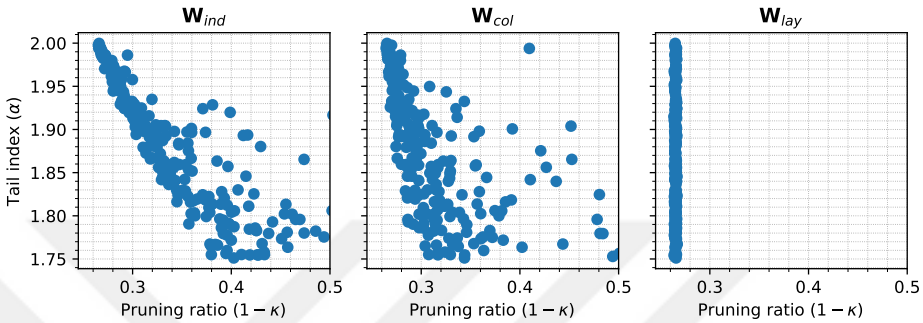


Figure D.2. Tail index and pruning ratio $(1-\kappa)$, with relative compression error $= 0.1$, for synthetically generated weight matrices [55].

The results demonstrate two interesting phenomena. First, a comparison of these results with Figure 5.3 shows that our empirical results show the most similarity with results obtained with $\mathbf{W}_{\text{ind},i}$, showing support for the existence of the HML condition. Another observation is that as the layer parameters become correlated, the prunability advantage conferred by heavier tails disappears. This observation both supports the existence of HML condition in overparametrized neural networks and invites further discussion on the importance of propagation of chaos phenomenon in such architectures for compressibility and generalization [150].

D.4. Global Magnitude Pruning and Node Pruning Results

In Figure D.3 present the global magnitude pruning results for magnitude pruning. The results are qualitatively very similar to those of the layer-wise magnitude pruning. Figures D.4 and D.5 present the results of node pruning on CNNs and FCNs respectively. As mentioned in Chapter 5, the less impressive results with FCNs might have to do with the layer widths not being sufficient for our theoretical conditions. A more favorable approach to structured pruning in FCNs would factor in the fact that

removal of columns from a layer is also equivalent to the removal of corresponding rows from the previous layer. When computing the pruning ratio, factoring in these corresponding rows would produce more benevolent results. However, we have not done this in our experiments since this is not necessarily implied by our theory.

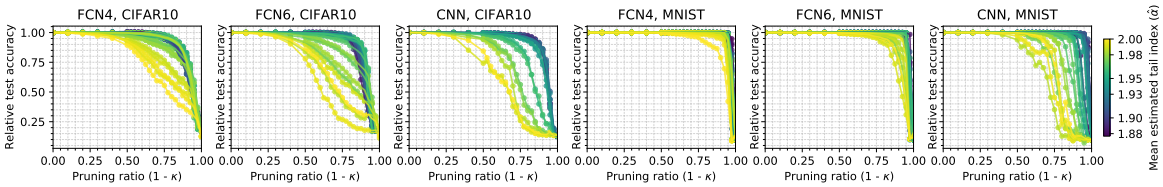


Figure D.3. Relative test accuracy and pruning ratio $(1 - \kappa)$ for global magnitude pruning. Color: $\hat{\alpha}$ [55].

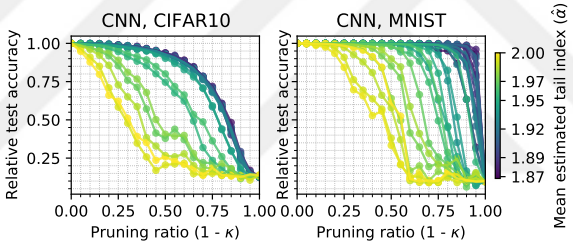


Figure D.4. Relative test accuracy vs. pruning ratio $(1 - \kappa)$ for structured pruning in CNNs. Color: $\hat{\alpha}$ [55].

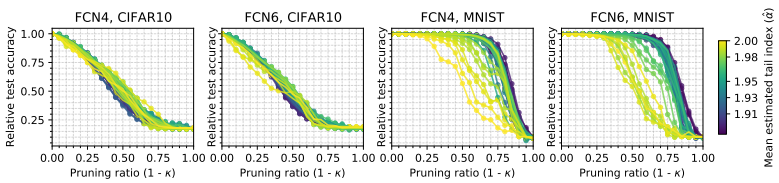


Figure D.5. Relative test accuracy and pruning ratio $(1 - \kappa)$ for node pruning in FCNs. Color: $\hat{\alpha}$ [55].

D.5. Experiments with Ergodic Averaged Parameters

In the main article we note that whether the pruning experiments are conducted on the final parameters or the ergodic averaging thereof does not lead to any concrete

differences in the results. The following Figures D.6, D.7, and D.8, which include results obtained with parameters that were ergodically averaged ($\tilde{\mathbf{x}}$), present results that were virtually identical with the originals presented in the paper.

D.6. Hardware and Other Resources

The computational resources for the experiments were provided by a research institute. The bulk of the resources were expended on the real data experiments, where a roughly equal division of labor between NVIDIA Titan X, 1080 Ti, and 1080 model GPU's. Our results rely on 273 models, training of which brings about a GPU-heavy computational workload. The training of a single model took approximately 4.5 hours, with an approximate estimated total GPU time for the ultimate results 1270 hours. This total also includes the training time for the 40 models which diverged during training, with the training stopping around 1 hour mark on average. The computational time expended for tail index estimation in real data experiments and the totality of synthetic experiments amounted to approximately 20 hours of computation with similar hardware as described above. We also conducted tail index estimation and pruning experiments on these networks, however the computational load of these experiments are negligible compared to those of the training of the algorithms, with an estimated CPU time of 48 hours for all estimation and pruning tasks that were published.

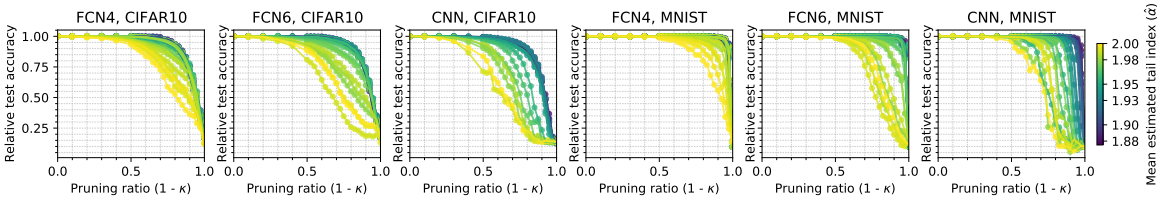


Figure D.6. Relative test accuracy and pruning ratio $(1 - \kappa)$ for layer-wise magnitude pruning using ergodic averages ($\tilde{\mathbf{x}}$). Color: $\hat{\alpha}$ [55].

As can be seen in the source code for the experiments in [184], the experiments were conducted using Python programming language. The deep learning framework PyTorch [245], including its code and tutorials were utilized in implementing the experiments.

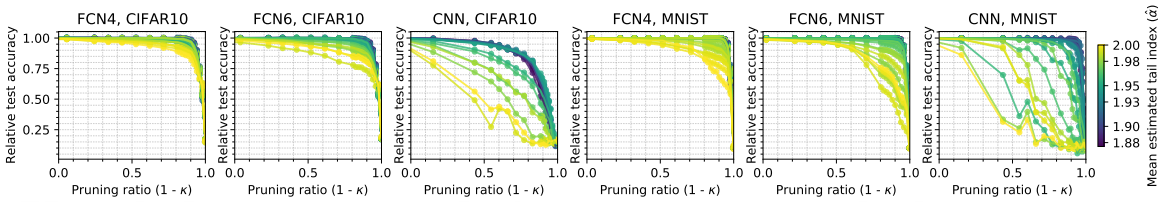


Figure D.7. Relative test accuracy and pruning ratio $(1 - \kappa)$ for singular value pruning using ergodic averages $(\tilde{\mathbf{x}})$. Color: $\hat{\alpha}$ [55].

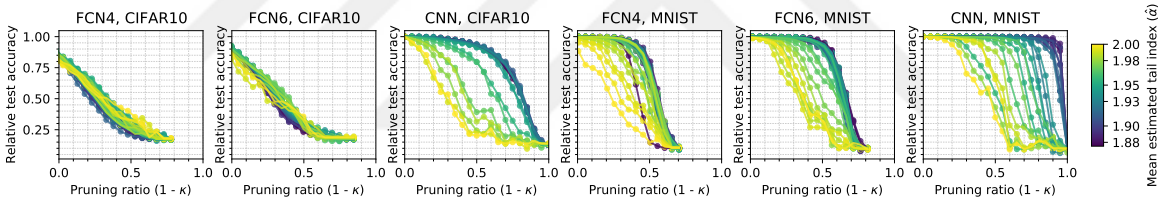


Figure D.8. Relative test accuracy and pruning ratio $(1 - \kappa)$ for node pruning using ergodic averages $(\tilde{\mathbf{x}})$. Color: $\hat{\alpha}$ [55].

APPENDIX E: LATENT REPRESENTATIONS AND GENERALIZATION

This appendix includes tables and figures that have been referred to in Chapter 6. We do not repeat the descriptions of the said items here for brevity, please see the said chapter for relevant context.

E.1. Out-of-Distribution Generalization Results

We provided an overall analysis of our results on OOD generalization presented in Figures E.7, E.8, E.9, and E.10 in Section 6.3.1.3. An important additional observation regarding these results is the unexpectedly high performances of some models in CIFAR-10-C results. While CIFAR-100-C results almost completely mirror the results for the in distribution test set (albeit with a lower accuracy overall, as expected), in CIFAR-10-C experiments, some models seem to almost preserve their test performance levels. As these are average performances across different corruptions, an analysis of the 19 different corruptions show that this happens in only some of the corruptions such as brightness and contrast, but not others, such as shot noise. We believe that corruption type-specific investigation of the said results is a promising future direction for the research presented here.

Table E.1. Learning rates for the experimental settings used in Figure 6.3.

FCN4, CIFAR-10	FCN4, CIFAR-100	FCN6, CIFAR-10	FCN6, CIFAR-100	CNN, CIFAR-10	CNN, CIFAR-100
0.005	0.001	0.0025	0.001	0.001	0.001
0.01	0.005	0.005	0.005	0.01	0.005
0.025	0.01	0.01	0.01	0.025	0.01
0.08	0.05	0.1	0.1	0.05	0.025
0.15	0.1	0.2	0.2	0.075	0.05
0.2	0.15	0.275	0.22	0.1	0.07
0.24	0.155	0.3	0.225	0.13	
0.27	0.16	0.34		0.15	

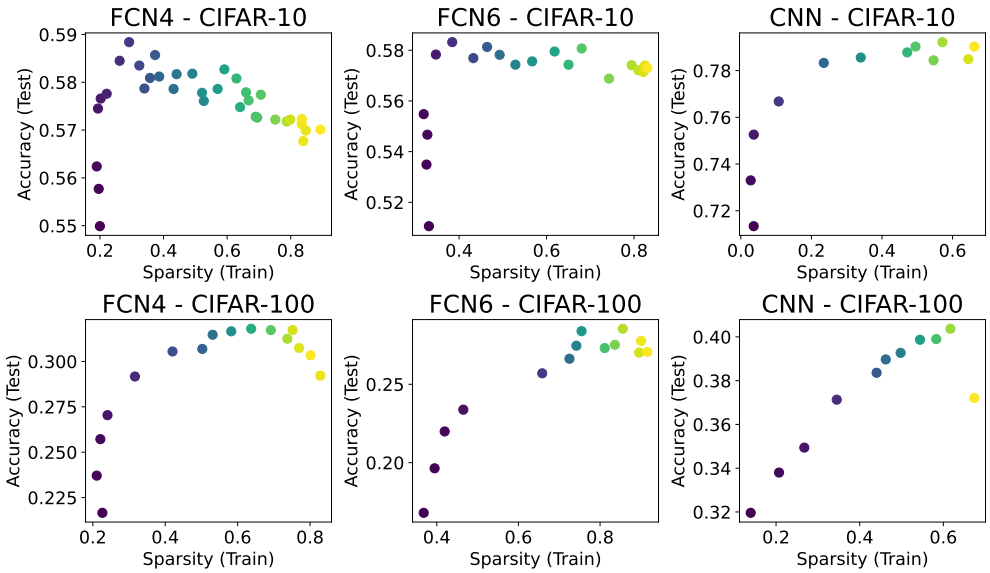


Figure E.1. Activation sparsity and accuracy. Brighter colors imply higher learning rates (comparable only within subgraphs).

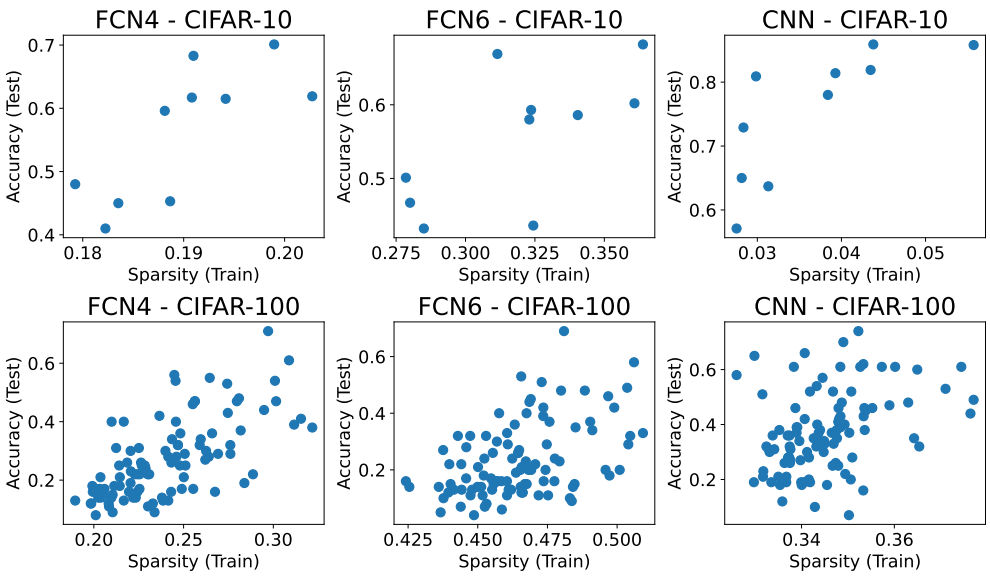


Figure E.2. Class-conditional sparsity and accuracy. Sparsity of the activations are examined within a single model (learning rate $\eta = 0.01$) for each setting, separately for each class.

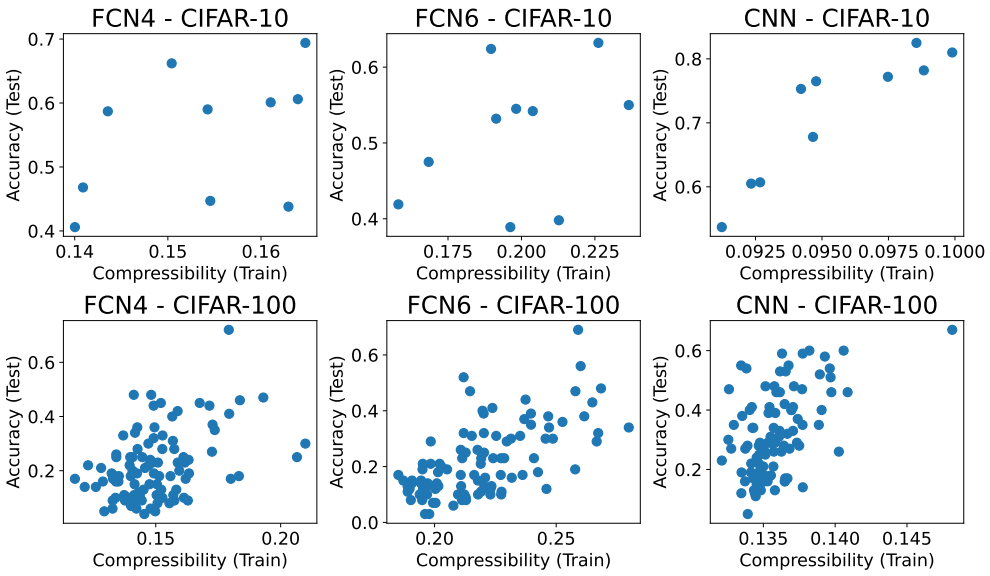


Figure E.3. Class-conditional compressibility and accuracy. Compressibility of the activations are examined within the model with minimum learning rate (η) for each setting, separately for each class. Here compressibility refers to (q, κ)-Compressibility as defined in (6.2), with $q = 2, \kappa = 0.05$.

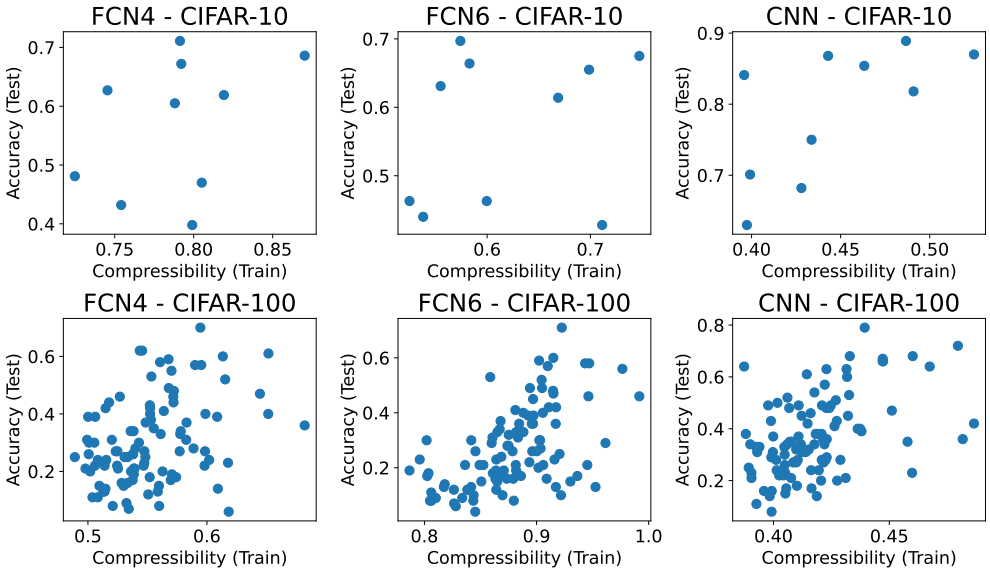


Figure E.4. Class-conditional compressibility and accuracy. Compressibility of the activations are examined within the model with maximum learning rate (η) for each setting, separately for each class. Here compressibility refers to (q, κ)-Compressibility as defined in (6.2), with $q = 2, \kappa = 0.05$.

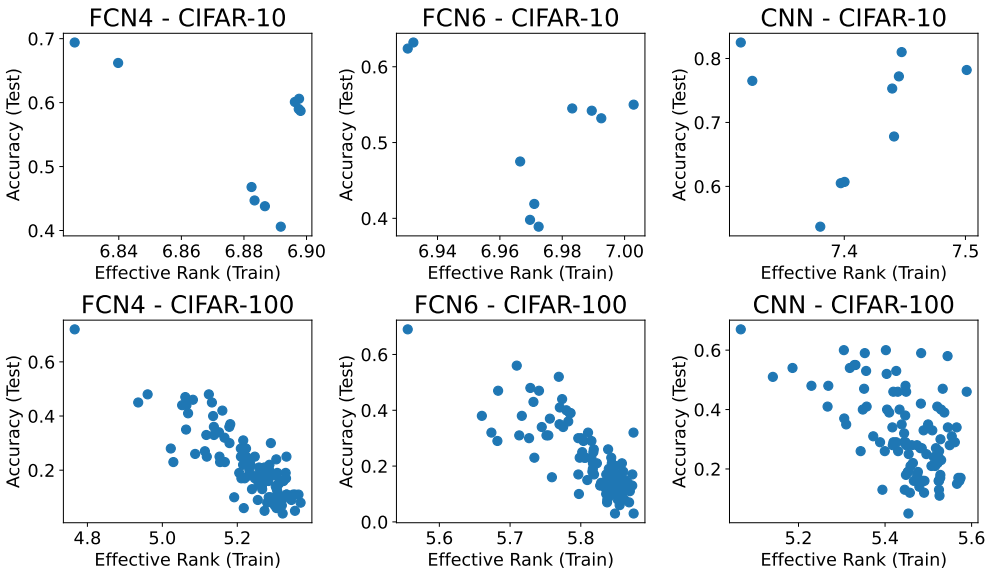


Figure E.5. Class-conditional effective rank and accuracy. Effective rank of the initial activations are examined within the model with minimum learning rate (η) for each setting, separately for each class.

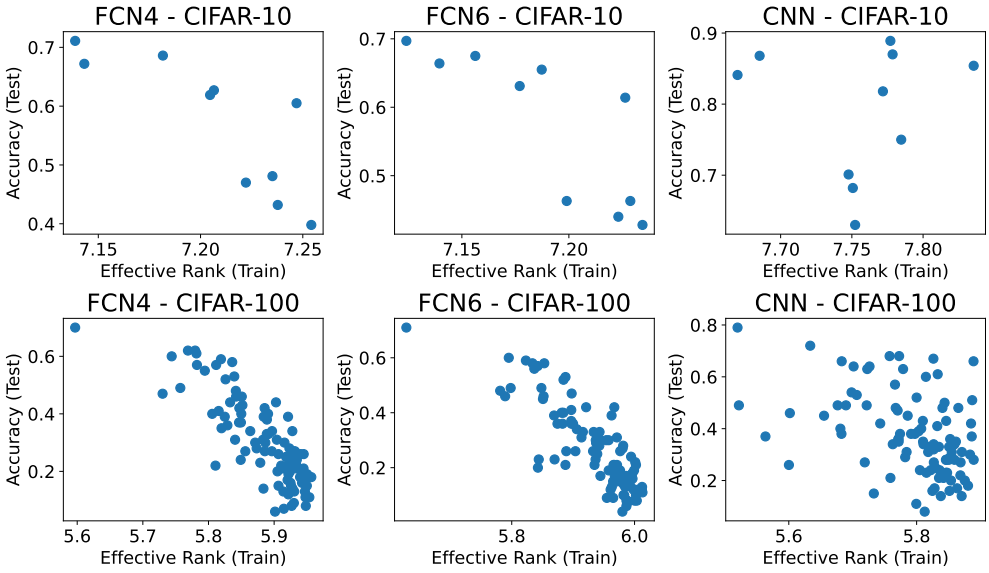


Figure E.6. Class-conditional effective rank and accuracy. Effective rank of the initial activations are examined within the model with maximum learning rate (η) for each setting, separately for each class.

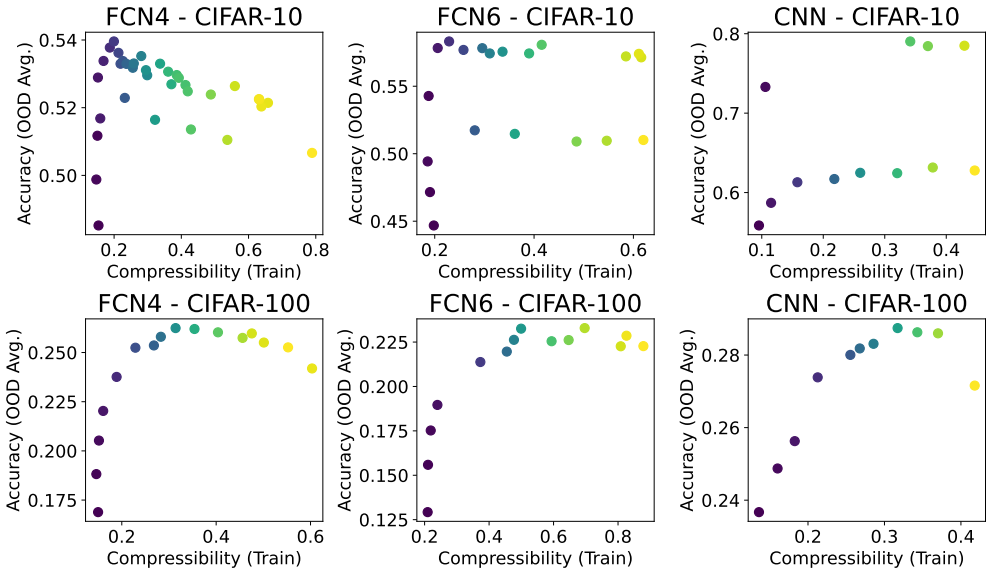


Figure E.7. Activation compressibility and OOD accuracy. Here compressibility refers to (q, κ)-Compressibility as defined in (6.2), with $q = 2, \kappa = 0.05$. Brighter colors imply higher learning rates (comparable only within subgraphs).

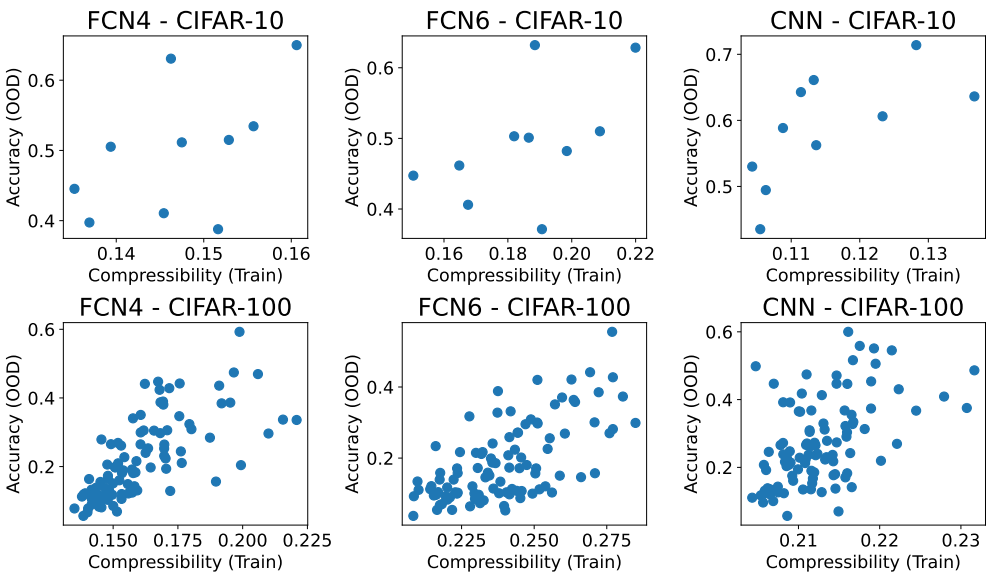


Figure E.8. Class-conditional compressibility and OOD accuracy. Compressibility of the activations are examined within a single model (learning rate = 0.01) for each setting, separately for each class. Here compressibility refers to (q, κ)-Compressibility as defined in (6.2), with $q = 2, \kappa = 0.05$.

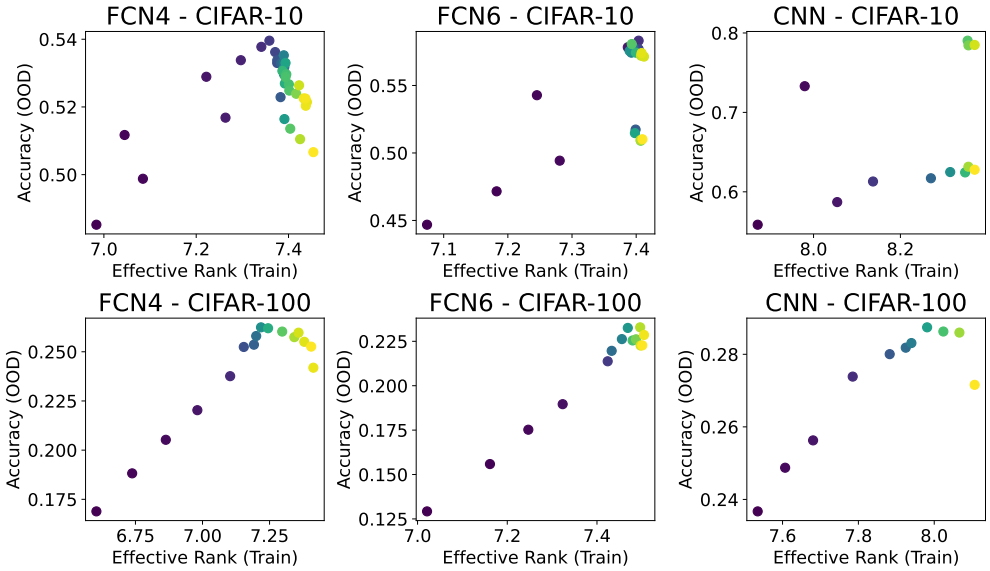


Figure E.9. Maximum effective rank and OOD accuracy. Each model's activations' effective rank has been summarized by the maximum effective rank observed in the trained model. Brighter colors imply higher learning rates (comparable only within subgraphs).

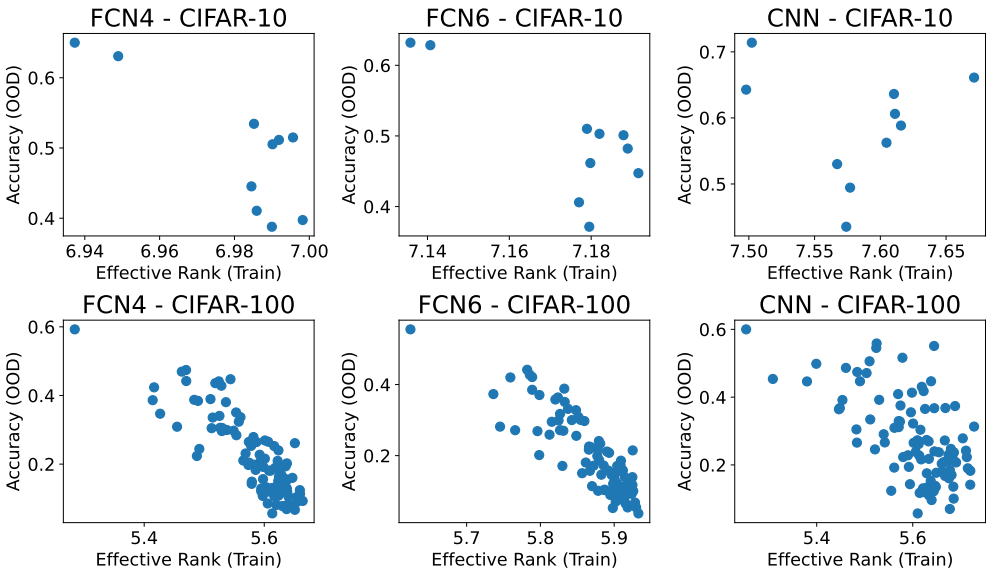


Figure E.10. Class-conditional effective rank and OOD accuracy. Effective rank of the initial activations are examined within a single model (learning rate = 0.01) for each setting, separately for each class.