Machine Learning Perspectives in Compression, Distributed Computing, and Brain Imaging

by

MohammadReza Ebrahimi

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy

Department of Electrical and Computer Engineering
University of Toronto

Machine Learning Perspectives in Compression, Distributed Computing, and Brain Imaging

MohammadReza Ebrahimi
Doctor of Philosophy

Department of Electrical and Computer Engineering
University of Toronto
2024

Abstract

This thesis explores three critical dimensions in machine learning: *modeling*, *training*, and *theory*. Each dimension, represented by studies in brain imaging, distributed computing, and compression, addresses unique challenges with the goal of advancing machine learning methodologies and applications.

First, within the domain of data modeling, we introduce Shared Gaussian Process Factor Analysis (S-GPFA), a novel probabilistic model for analyzing multi-subject fMRI datasets. S-GPFA addresses the challenge of modeling individual variability while uncovering shared temporal dynamics and spatial organization of brain activity. By incorporating Gaussian Process priors and emphasizing the temporal dimension of data, S-GPFA offers a more accurate and interpretable representation of brain activity compared to traditional static methods. The application of S-GPFA to a large fMRI dataset demonstrates its ability to identify group-specific dynamical characteristics and brain regions with meaningful functional variability, providing valuable insights into socioemotional cognitive capacity and potential avenues for studying psychiatric disorders.

Second, focusing on the training aspect, we address the problem of straggler mitigation in distributed training of machine learning models. We present two innovative coding schemes, Selective Reattempt Sequential Gradient Coding (SR-SGC) and Multiplexed Sequential Gradient Coding (M-SGC), that leverage coding across both the spatial and temporal dimensions to achieve straggler resilience while reducing computational load. These schemes exploit the temporal diversity of straggler behavior, adapting to varying worker speeds and minimizing delays. Experiments on a large-scale AWS Lambda cluster demonstrate the effectiveness of the proposed schemes in reducing runtime and improving training performance under real-world conditions.

Third, from a theoretical perspective, we investigate the foundations of data coupling and compression through the lens of information theory. We introduce the Minimum Entropy Coupling with Bottleneck (MEC-B) framework for lossy compression under logarithmic loss. This framework extends the classical Minimum Entropy Coupling (MEC) by incorporating rate limits, enabling a more controlled and flexible approach to compression. We explore the Entropy-Bounded Information Maximization (EBIM) formulation for compression and propose a novel search algorithm for identifying deterministic mappings with guaranteed performance bounds. Additionally, we characterize the optimal solution in the neighbourhood of deterministic mappings, providing valuable theoretical insights into the problem structure.

Through these studies, this thesis contributes to machine learning methodologies and applications across diverse domains, ranging from brain imaging and distributed computing to information theory and data compression.

To my beloved family,
Parvin, Hamid, Elham, Maryam, and Helia
for their unconditional love and support.

Acknowledgements

I would like to extend my appreciation to my supervisor, Prof. Ashish Khisti, whose exceptional insights and systematic thinking have profoundly shaped this work. I am grateful for his support during challenging times.

I am also thankful to Prof. Jun Chen, Prof. Aristotle Voineskos, and Prof. Colin Hawco for their expert advice and perspectives, which have significantly enriched my research. I extend my heartfelt thanks to my exceptional coauthors, Navona Calarco and Nikhil Krishnan, for their collaborative efforts and contributions.

My special thanks go to Prof. Ben Liang and Prof. Alireza Makhzani, members of my Departmental PhD thesis committee, for their insightful feedback and engagement since the thesis proposal stage. I am also grateful to Prof. Soosan Beheshti for her thoughtful thesis appraisal, and to Prof. Ravi Adve for his participation on the examination committee and for his meticulous editorial feedback.

I am deeply grateful to my family—my parents, Parvin and Hamid, and my sisters, Elham and Maryam. Their unconditional love and support have been my foundation throughout this journey. Though physically distant, their warmth and encouragement have been ever-present in my heart.

Lastly, I thank my best friend and beloved wife, Helia, who has been my steadfast companion on this journey. Her support and love have been a constant source of strength every step of the way.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Machine learning (ML) has profoundly transformed the way we analyze data, make predictions, and solve complex problems across numerous fields. At its core, machine learning involves the development of algorithms that allow computers to learn from and make decisions based on data. The impact of machine learning is profound and far-reaching, affecting industries and sciences by providing innovative solutions to complex, data-driven problems. It has transformed sectors such as healthcare, finance, and telecommunications, where predictive analytics and automated decision-making processes significantly enhance efficiency and effectiveness. In the scientific domain, machine learning techniques have become indispensable tools in fields ranging from genomic sequencing to astrophysics, helping to unravel complex patterns and predict phenomena with unprecedented accuracy.

This thesis explores three critical dimensions in machine learning: *modeling*, *training*, and *theory*. Each dimension is represented by a study in brain imaging, distributed computing, and compression, addressing unique challenges and contributing to the overarching goal of advancing machine learning methodologies and applications.

Modeling refers to the process of providing simplifying assumptions on the data generation process, which leads to algorithms capable of identifying patterns in data and making predictions or decisions based on those patterns. Training involves optimizing these models using data, often requiring substantial computational resources and careful coordinating and aggregating, especially in distributed systems. Lastly, theoretical insights, particularly from information theory, are essential for understanding and improving machine learning systems, by quantifying the limits of data compression, storage, and communication.

1.1 Core Themes of the Thesis

The following themes are examined through targeted studies that tackle distinct challenges within these areas.

1.1.1 Modeling

Classically, statistical modeling involves assuming a simplified generation process for the observed data with the goal of capturing specific properties of data. In modern deep learning, models are much

more flexible, so modeling largely entails making architectural choices that inject specific inductive biases into the learning algorithm.

Functional Magnetic Resonance Imaging (fMRI) offers a window into the dynamic activity of the human brain, through a noisy proxy that poses significant challenges for interpretation. Machine learning, with its array of modeling techniques, provides powerful tools to extract meaningful insights from this complex neural data. Chapter 2 presents a novel probabilistic modeling approach, known as Shared Gaussian Process Factor Analysis (S-GPFA), to model the shared temporal dynamics and spatial organization of brain activity across individuals. The proposed model simultaneously performs functional aggregation, dimensionality reduction, and dynamical modeling of fMRI data in multi-subject datasets.

S-GPFA addresses a critical challenge in fMRI research: accounting for individual variability while uncovering shared neural dynamical patterns. A key innovation of S-GPFA lies in its emphasis on the temporal dimension of data during modeling. Unlike traditional approaches that often necessitate making unjustifiable modeling assumptions, S-GPFA leverages the inherent temporal dimension within fMRI data to achieve a more interpretable representation of brain activity. This emphasis on temporal dynamics allows S-GPFA to better model individual variability while uncovering shared neural patterns across subjects.

1.1.2 Training

The ever-increasing size of datasets and the growing complexity of machine learning models necessitate the use of distributed training, where multiple computing nodes work together to train a model. However, the presence of stragglers, or slow-performing workers, can significantly impede the efficiency of distributed training, creating bottlenecks that delay the overall process.

Chapter 3 focuses on distributed training and introduces novel coding schemes that mitigate the impact of stragglers, thereby enhancing training efficiency and scalability. Traditional approaches to straggler mitigation often rely on replication or simple redundancy mechanisms, which can lead to increased computational overhead and resource utilization. Chapter 3 presents two innovative coding schemes, Selective Reattempt Sequential Gradient Coding (SR-SGC) and Multiplexed Sequential Gradient Coding (M-SGC), that leverage coding across both the spatial (workers) and temporal (rounds) dimensions to achieve straggler resilience with significantly reduced computational load. These schemes exploit the temporal diversity of straggler behavior, where periods of high straggler activity are often followed by periods of low straggler activity, to optimize resource allocation and minimize delays. By incorporating coding across time, SR-SGC and M-SGC effectively adapt to the dynamic nature of straggler patterns, ensuring efficient training even in the presence of varying worker speeds. This adaptability is crucial for real-world distributed training scenarios, where network fluctuations and resource contention can lead to unpredictable straggler behavior.

1.1.3 Theory

Information theory provides a fundamental framework for understanding the limits of data compression, communication, and representation. Its principles have profound implications for machine learning, guiding the design of efficient algorithms and providing insights into the inherent trade-offs between compression and fidelity. Chapter 4 explores the intersection of information theory and ma-

chine learning, introducing a novel framework called Minimum Entropy Coupling with Bottleneck (MEC-B) that addresses the challenge of lossy data compression under logarithmic loss.
Traditional lossy compression methods often focus on minimizing distortion measures such as mean squared error. MEC-B, on the other hand, utilizes log-loss as its distortion metric, providing a more suitable measure of fidelity for tasks involving probabilistic predictions or soft reconstructions. Additionally, MEC-B extends the popular Minimum Entropy Coupling (MEC) framework by incorporating rate limits, allowing for a more controlled and flexible coupling. While MEC focuses on finding the joint distribution with minimal entropy given marginal distributions, MEC-B introduces a constraint on the entropy of the compressed representation, effectively regulating the amount of entropy of the resulting coupling.
The chapter also proposes a relevant formulation for compression known as Entropy-Bounded Information Maximization (EBIM), where the objective is to maximize the mutual information between the input and the compressed code, subject to an entropy constraint on the code. This formulation aligns with the goals of lossy compression, seeking to preserve as much relevant information as possible while adhering to a specified rate limit.
To provide efficient approximate solutions for EBIM, the chapter proposes a novel search algorithm for identifying deterministic mappings with guaranteed performance bounds. Additionally, it characterizes the optimal solution in the vicinity of deterministic mappings, providing insights into the structure of the solution space and paving the way for further algorithmic development.
The MEC-B framework offers a unique perspective on compression and information retrieval, enabling the design of algorithms that achieve high fidelity while adhering to specific entropy constraints. This capability is particularly relevant for machine learning applications where efficient data representation and information extraction are crucial.

1.2 Summary of Contributions

The first study, detailed in Chapter 2, introduces Shared Gaussian Process Factor Analysis (S-GPFA) as a novel probabilistic model for analyzing fMRI data. S-GPFA addresses the challenge of modeling individual variability while uncovering shared temporal dynamics and spatial organization of brain activity in multi-subject datasets. The results presented in Chapter 2 have appeared in [1]. The key contributions of this study include:

- **Modeling Shared Temporal Dynamics:** S-GPFA incorporates Gaussian Process priors to model the temporal correlations within fMRI data, leading to more accurate and interpretable representations of brain activity compared to traditional static methods.
- **Functional Aggregation and Dimensionality Reduction:** The model effectively combines functional aggregation, dimensionality reduction, and dynamical modeling, providing a comprehensive framework for analyzing multi-subject fMRI datasets.
- **Scientific Utility:** The application of S-GPFA to the SPINS dataset demonstrates its ability to identify group-specific dynamical characteristics and brain regions with meaningful functional variability, offering valuable insights into socioemotional cognitive capacity and potential avenues for studying psychiatric disorders.

The second study, presented in Chapter 3, focuses on mitigating the impact of stragglers in distributed training of machine learning models. The chapter introduces two novel coding schemes, Selective Reattempt Sequential Gradient Coding (SR-SGC) and Multiplexed Sequential Gradient Coding (M-SGC), that leverage coding across both the spatial and temporal dimensions to achieve straggler resilience with reduced computational load. The results presented in Chapter 3 have been published in [2], where the author holds joint first authorship. Contributions primarily focused on the real-world evaluation and implementation of the schemes.
The main contributions of this study are:

- **Exploiting Temporal Diversity:** SR-SGC and M-SGC effectively exploit the temporal diversity of straggler behavior, adapting to varying worker speeds and minimizing delays caused by stragglers.
- **Reduced Computational Load:** M-SGC significantly reduces the computational load per worker compared to traditional Gradient Coding schemes, leading to improved training efficiency.
- **Real-world Performance and Insights:** Experiments on a large-scale AWS Lambda cluster demonstrate the effectiveness of the proposed schemes in reducing runtime and improving training performance in real-world conditions with naturally occurring stragglers.

The third study, explored in Chapter 4, investigates the theoretical foundations of data coupling and compression through the lens of information theory. The chapter introduces Minimum Entropy Coupling with Bottleneck (MEC-B) as a framework for lossy data compression under logarithmic loss. As of the writing of this thesis, publications related to this work are currently under review.
The primary contributions of this study include:

- **Extension of Minimum Entropy Coupling:** MEC-B extends the classical MEC framework by incorporating rate limits, enabling a more controlled and flexible approach to coupling.
- **Entropy-Bounded Information Maximization:** The chapter introduces and investigates the EBIM formulation for compression, which seeks to maximize the mutual information between the input and the compressed representation subject to an entropy constraint.
- **Novel Search Algorithm and Theoretical Insights:** A novel search algorithm is proposed for identifying deterministic mappings with guaranteed performance bounds in the context of EBIM. Additionally, the chapter characterizes the optimal solution in the neighbourhood of deterministic mappings, providing valuable theoretical insights into the problem structure.

1.3 Organization of the Thesis

This thesis is structured into five chapters, each addressing a specific aspect of the research conducted.
Chapter 2 focuses on fMRI data analysis and presents Shared Gaussian Process Factor Analysis (S-GPFA) as a novel probabilistic model for capturing shared temporal dynamics and individual variability in brain activity. The chapter provides a comprehensive overview of fMRI, highlighting

the challenges associated with analyzing this complex data, and introduces S-GPFA as a solution that addresses these challenges. The chapter also includes experimental results demonstrating the effectiveness of S-GPFA in identifying group-specific dynamical characteristics and brain regions with meaningful functional variability.

Chapter 3 explores the problem of straggler mitigation in distributed training of machine learning models. The chapter begins by discussing the challenges posed by stragglers in distributed computing environments and introduces two innovative coding schemes, SR-SGC and M-SGC, that leverage coding across both the spatial and temporal dimensions to achieve straggler resilience. The chapter further presents experimental results showcasing the performance improvements achieved by these schemes on a large-scale AWS Lambda cluster.

Chapter 4 investigates the theoretical foundations of data coupling and compression through the lens of information theory. The chapter introduces Minimum Entropy Coupling with Bottleneck (MEC-B) as a framework for lossy data compression under logarithmic loss and explores its connection to the classical Minimum Entropy Coupling (MEC) framework. The chapter also introduces the Entropy-Bounded Information Maximization (EBIM) formulation for compression and proposes a novel search algorithm for identifying deterministic mappings with guaranteed performance bounds.

Chapter 5, Conclusions, summarizes the key findings and contributions of the thesis and outlines potential avenues for future work.

Chapter 2

Group Temporal Dynamics Analysis in fMRI

2.1 Introduction

Functional Magnetic Resonance Imaging (fMRI) is the most widely applied method to image human cognition. In it, neural hemodynamics are imaged in three-dimensional space, over a fourth dimension of time. In task fMRI, participants additionally engage in an in-scanner experiment designed to engage some aspect of cognition. Of interest to neuroscientists are the observed topological and, increasingly, dynamical patterns associated with the given cognitive state. While fMRI has illuminated the 'neural signatures' underlying many aspects of cognition, others, such as social cognition, remain elusive.

In recent years, neuroscientists have attempted to better capture true variability in human cognition. Larger and more representative imaging studies are increasingly the norm, boasting hundreds or even thousands of participants, often conducted across multiple research centers. These investigations have reported large individual differences in anatomical [3] and functional networks, the latter in both the spatial [4] and temporal domains [5]. Moreover, consensus evidence shows that task fMRI captures not only task-related cognition, but also background activity, physiological processes, and motion, as well as a variety of artifacts associated with scanner hardware [6]. In short: the observed fMRI signal represents a noisy amalgam of signals of varying and often unknown provenance.

Several Factor Analysis (FA) methods have been proposed to compensate for functional variability among subjects. For instance, Shared Response Model (SRM) [7] provides a means of aligning participants' activation via a shared low dimensional response and subject-specific bases, and Hierarchical Topographic Factor Analysis (HTFA) [8] learns a global template of brain activity and casts each participant's response as a perturbation of that template. However, these methods are limited in that they treat time as static, though converging evidence suggests that temporal correlation structure – even on the slow timescale of hemodynamics – carries meaningful signal [9]. Gaussian Process Factor Analysis (GPFA) [10] accommodates time by linking together factor analyzers in low-dimensional latent space, using imposed Gaussian process priors. This allows GPFA to model temporal and spatial structure in observation space, and has been applied within subjects to model dynamic functional connectivity [11]: an initial proof of principle of its suitability to fMRI.

We introduce 'Shared Gaussian Process Factor Analysis' (S-GPFA), a novel probabilistic model for finding subject-specific topographies and shared temporal dynamics in multi-subject fMRI datasets. The proposed model simultaneously performs functional aggregation, dimensionality reduction, and dynamical modeling of fMRI data. We demonstrate the scientific utility of our model in identifying group-specific dynamical characteristics in the context of socioemotional cognitive capacity. Furthermore, identifying brain regions with meaningful functional variability across subjects within a heterogeneous sample provides preliminary evidence that S-GPFA can enable exploratory and hypothesis-driven examinations of functional topographies in psychiatric disorders. The results presented in this chapter have appeared in [1].

2.2 Related Works

Shared Response Model (SRM) [7] is a multi-view extension of probabilistic Principal Component Analysis (pPCA) [12], wherein latent factors (named the shared response) are common across all subjects for each time point, while fixed factor loadings are specific to each subject. In addition, SRM explicitly imposes an orthogonality constraint on its loading matrices. Variants of this model such as Robust Shared Response Model (RSRM) [13] have been proposed in which both shared and private components in subjects' responses are explicitly modeled, similar to probabilistic Canonical Correlation Analysis (pCCA) [14]. Similarly, in [15] an independent component analysis (ICA)-based method has been proposed to separate shared and private sources in subjects' fMRI observations based on a similar factor model by exploiting time-delayed correlations.

Similar to static dimensionality reduction methods like Linear Factor Analysis (LFA) and PCA, SRM treats multivariate time series of fMRI recordings as a collection of independent snapshots in time, entirely disregarding temporal information. In other words, SRM remains invariant if training time series are shuffled through the time dimension. As with PCA, SRM cannot discover temporal dynamics unless such dynamics materialize as variance, which stems from both dynamics and noise [16].

Moreover, to address the rotational ambiguity of latent variables and factor loadings, SRM adopts a similar solution as PCA, by enforcing subjects' factor loadings to be orthonormal. Although this restricts the output to be unique, the resulting solution may not be interpretable since the assumption of orthogonal topographies is likely an undue simplification of physiology. Nonetheless, SRM's solution captures the shared structures in subjects' fMRI observations. An alternative solution for resolving rotational ambiguity is to utilize the inherent temporal dynamic of the data. This can address the aforementioned problems together, i.e., finding unique and interpretable solutions while bringing insight into shared dynamic structures in observations. We adopt a similar approach as in Gaussian Process Factor Analysis (GPFA) [10], where latent variables are linked through time by employing Gaussian Process priors, allowing viable modeling of both temporal and spatial covariance in data.

Closely related to S-GPFA, the family of Matrix Normal SRM [17] models temporal dynamics of fMRI data by imposing a Matrix-Normal prior over shared time series. However, S-GPFA does not impose a Kronecker-separable covariance structure over shared time series, and hence, is not in the family of matrix-variate normal models. Instead, S-GPFA models independent latent temporal components that can unfold over different timescales, effectively discovering components of diverse

temporal scales. In MN-SRM however, the same temporal covariance structure is assumed over different components of the shared space.

2.3 The Proposed Model: Shared-GPFA

In the following, we present the mathematical formulation of the probabilistic model for S-GPFA. We denote by $\boldsymbol{A}_{i,:}$, $\boldsymbol{A}_{:,j}$, and $[\boldsymbol{A}]_{m,n}$ the ith row vector, the jth column vector, and element (m,n) of matrix $\boldsymbol{A}$, respectively. Let $\boldsymbol{Y}^{(m)} \in \mathbb{R}^{Q\times T}$ be the observed high-dimensional fMRI time series for subject $m \in \{1,2,\ldots,M\}$ where Q is the number of voxels (or in general, brain regions), T is the number of temporal samples, and M is the total number of subjects in the dataset. We denote by $\boldsymbol{Y}^{(m)}_{q,:} \in \mathbb{R}^{1\times T}$ the q-th row of the observation matrix, or equivalently, the time series of brain activation in region q for subject m. We assume all subjects are exposed to identical and time-synchronized stimuli while brain activities are recorded and activation time series are centered over time. S-GPFA extracts shared low-dimensional latent trajectories $\boldsymbol{X} \in \mathbb{R}^{P\times T}$ describing the common latent state of all subjects, where their linear combinations through loading matrices of each subject describe the observed activation time series. Here, P is a hyperparameter to be set as the dimensionality of the latent space ($P < Q$). This can be done using any standard hyperparameter tuning method, such as likelihood cross-validation [18], as done in this work. Following the standard Factor Analysis model, we define a set of linear (isotropic) Gaussian systems for fMRI observations and latent trajectories:

$$\boldsymbol{Y}^{(m)}_{:,t} | \boldsymbol{X}_{:,t}\,, \boldsymbol{W}^{(m)} \sim \mathcal{N}\left(\boldsymbol{W}^{(m)}\boldsymbol{X}_{:,t}, \boldsymbol{\Psi}^{(m)}\right) \tag{2.1}$$

where $\boldsymbol{W}^{(m)} \in \mathbb{R}^{Q\times P}$ and $\boldsymbol{\Psi}^{(m)} \in \mathbb{R}^{Q\times Q}$ denote the factor loading matrix and observation noise covariance matrix for subject m, respectively. Note that columns of factor loadings can be considered as brain activation bases meant to capture subject-specific topographies (we use the terms 'factor loadings' and 'subject topographies' interchangeably). Constraining $\boldsymbol{\Psi}^{(m)}$ to be diagonal will let $\boldsymbol{W}^{(m)}$ and $\boldsymbol{X}$ completely define the covariance structure of region activation patterns. To explicitly let brain region q of subject m accommodate different noise levels, we model the diagonal elements as $[\boldsymbol{\Psi}^{(m)}]_{q,q} = \rho^2_{(m)}\sigma^2_q$, where ρ and σ are subject-specific and region-specific learnable parameters, respectively. Following standard GPFA, to model the temporal correlation of the data, we impose independent Gaussian Process (GP) priors over each latent trajectory:

$$\boldsymbol{X}_{p,:} \sim \mathcal{GP}\left(0, \kappa_p(\cdot,\cdot)\right), \qquad p \in \{1,\ldots,P\} \tag{2.2}$$

where κ_p denotes a Mercer kernel function. Therefore, the covariance matrix for the pth latent trajectory, $\boldsymbol{K}_p$, will be the Gram matrix of the kernel over the index set $\{1,2,\ldots,T\}$, i.e., $[\boldsymbol{K}_p]_{t_1,t_2} = \kappa_p(t_1,t_2)$. In general, the choice of kernel function imposes important assumptions on the form and smoothness of observed time series. Following prior literature [9, 11], we opt to employ commonly used Squared Exponential (SE) kernel functions:

$$\kappa_p(t_1,t_2) = \alpha_p^2 \exp\left(-\frac{(t_1-t_2)^2}{2\tau_p^2}\right) + \eta_p^2\,\mathbb{1}_{t_1=t_2} \tag{2.3}$$

Hence, the GP prior is fully parameterized through the kernel variance α_p^2, characteristic timescale $\tau_p \in \mathbb{R}_+$, and the kernel independent noise variance η_p^2. Furthermore, we fix the scale of $\boldsymbol{X}$ to have

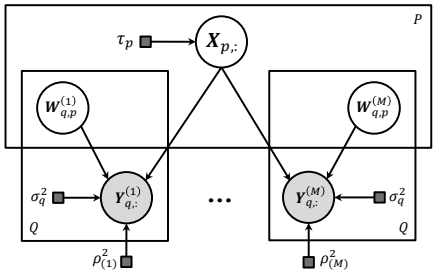


Figure 2.1: Graphical Model for S-GPFA.

$$\boldsymbol{X}_{p,:} \sim \mathcal{GP}\left(0, \kappa_p(.,.)\right)$$
$$\boldsymbol{W}^{(m)}_{q,p} \sim \mathcal{N}(0,1)$$
$$\boldsymbol{Y}^{(m)}_{:,t} | \boldsymbol{X}_{:,t}, \boldsymbol{W}^{(m)} \sim \mathcal{N}\left(\boldsymbol{W}^{(m)}\boldsymbol{X}_{:,t}, \boldsymbol{\Psi}^{(m)}\right)$$
$$\kappa_p(t_1,t_2) = \exp\left(-\frac{(t_1-t_2)^2}{2\tau_p^2}\right)$$

$\boldsymbol{X}_{:,t} \sim \mathcal{N}(0, I)$ by setting $\alpha_p^2 + \eta_p^2 = 1$. This will prevent the identifiability issue in the scale of $\boldsymbol{X}$ and $\boldsymbol{W}^{(m)}$ by allowing unconstrained learning for factor loadings [9–11]. We also set η_p^2 to a small value to act as a diagonal jitter for numerical stability. Therefore, the characteristic timescales τ_p can fully define the priors over latent trajectories. Figure 2.1 shows the graphical model for S-GPFA along with the summary of model specification.

To find the model parameters, we employ gradient ascent (using ADAM optimizer [19] and TensorFlow Probability) to maximize the joint probability distribution of observations and latent variables, conditioned on model parameters. The collective set of model parameters and latent variables to be inferred is denoted by $\boldsymbol{\theta} = \left\{\boldsymbol{X}, \{\boldsymbol{W}^{(m)}, \rho^2_{(m)}\}_{m=1}^M, \{\sigma_q^2\}_{q=1}^Q, \{\tau_p\}_{p=1}^P\right\}$. The resulting maximum a posteriori (MAP) estimate of model parameters is the minimizer of the objective function:

$$\begin{aligned}
&\sum_{m=1}^{M}\sum_{q=1}^{Q}\left[\frac{T}{2}\log(2\pi\rho^2_{(m)}\sigma_q^2) + \frac{\left\|\boldsymbol{Y}^{(m)}_{q,:} - \boldsymbol{W}^{(m)}_{q,:}\boldsymbol{X}\right\|^2}{2\rho^2_{(m)}\sigma_q^2}\right] \\
&+\lambda\sum_{p=1}^{P}\left[\frac{1}{2}\log\det(2\pi\boldsymbol{K}_p) + \frac{1}{2}\boldsymbol{X}_{p,:}\boldsymbol{K}_p^{-1}\boldsymbol{X}^T_{p,:}\right] \\
&+\sum_{m=1}^{M}\frac{1}{2}\left\|\boldsymbol{W}^{(m)}\right\|_F^2
\end{aligned} \tag{2.4}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. For now, let us set $\lambda = 1$ to achieve the standard MAP solution – we will explain the motivation behind including this constant in the objective function later in this section. The first summation term in (2.4) is the reconstruction loss promoting data fit. We refer to the second summation term as the smoothness loss, where $\log\det(\boldsymbol{K}_p)$ constitutes a model complexity penalty (in terms of smoothness) and $\boldsymbol{X}_{p,:}\boldsymbol{K}_p^{-1}\boldsymbol{X}^T_{p,:}$ promotes smooth latent trajectories governed by τ_p. Finally, the last term is a weight decay loss for subjects' factor loadings. While the reconstruction loss favors complex models to fit the observed data, smoothness and loading weight decay losses act as regularizers to promote smooth and less flexible models, respectively.

Adding a new subject $M+1$ to a previously trained model is done by finding the maximizer of $\mathrm{P}(\boldsymbol{Y}^{(M+1)}, \boldsymbol{W}^{(M+1)} | \boldsymbol{X}, \rho^2_{(M+1)}, \{\sigma_q^2\}_{q=1}^Q)$ with respect to $\boldsymbol{W}^{(M+1)}$ and $\rho^2_{(M+1)}$. Note that the shared latent trajectories, existing subject's topographies, and region/subject noise factors will remain unchanged. In addition, using previously learned topographies, shared timescales, and noise factors, one can map new observations from the existing subjects in the training cohort into the shared space. This can be done by maximizing $\mathrm{P}(\{\boldsymbol{Y}^{(m)}_{\text{new}}\}_m, \boldsymbol{X}_{\text{new}} | \{\boldsymbol{W}^{(m)}, \rho^2_{(m)}\}_m, \{\sigma_q^2\}_{q=1}^Q, \{\tau_p\}_{p=1}^P)$ with respect to $\boldsymbol{X}_{\text{new}}$, where $\boldsymbol{Y}_{\text{new}}$ denotes new observations from the subjects in the training cohort and $\boldsymbol{X}_{\text{new}}$

Figure 2.2: $\lambda = 1$: Simulated dataset with parameters $M = 20$, $Q = 50$, $T = 200$, $P = 2$. Latent trajectories sampled from independent GP with SE kernel and fixed timescales $\tau_1 = 1, \tau_2 = 6$. Factor loadings, as well as subject and region noise levels, are sampled from a standard normal distribution.

Figure 2.3: $\lambda = 0.1 \times MQ/P = 50$. The same dataset as in Figure 2.2. Amplified smoothness loss results in finding more accurate parameters.

shows the associated shared latent trajectories. Similarly, note that previously trained subject's topographies, shared latent timescales, and region/subject noise factors will remain unchanged.

Modified Training Objective Increasingly, fMRI studies include sample sizes of hundreds of participants (M) with thousands of voxels recorded (Q). An examination of the objective function in (2.4) reveals that the objective is extensively dominated by the reconstruction loss, since the total number of observed time series, QM, is often orders of magnitudes larger than the latent space dimensionality P. In such a case, the solution of (2.4) cannot afford to model the temporal smoothness properties of the data, and consequently, attempts to fit the observed data with a possibly complex model. To resolve this issue, we amplify the smoothness loss in (2.4) by hyperparameter $\lambda \propto MQ/P$ to balance the weight of the smoothness loss against the reconstruction loss. Note that from a probabilistic perspective, such regularization can be interpreted as weighted likelihood [20].

Although, as with any hyperparameter, standard methods like cross-validation can be employed to tune λ, we use the fixed value of $0.1 \times MQ/P$ for all experiments.

Using simulated data, we can demonstrate how amplifying the smoothness loss helps the model use temporal dynamics of the observations to accurately estimate shared latent responses and subject-specific topographies. But prior to this, we should address two inherent identifiability issues. First, the order of latent variables is naturally arbitrary, and second, the signs of shared latent trajectories $\{\boldsymbol{X}_{p,:}\}_{p=1}^{P}$ and subject topographies $\{\boldsymbol{W}_{:,p}\}_{p=1}^{P}$ are in opposite interplay. Moreover, in SRM, factor loadings are constrained to be orthonormal, hindering direct comparison of the true and estimated parameters. However, neither the order nor the scale and sign of these parameters are of interest. Therefore, to deal with the former issue, for each model we choose the ordering of latent dimensions that maximizes the sum of absolute correlations between matched pairs of the true and estimated latent time series. To address the latter issue, we use the absolute value of correlations between the ground truth and estimations to assess the performance of each model.

To generate simulated data, we sample $P = 2$ latent trajectories from independent Gaussian Processes (with SE kernel and fixed timescales). We also sample factor loading weights from a standard normal distribution, independently for each of $M = 20$ subjects. Linear combination of latent trajectories and factor loadings generate $Q = 50$ dimensional times series of length $T = 200$ for each subject. Noisy versions of these observations are used to train S-GPFA and SRM models. We use the optimized implementation of SRM [21] from the Brain Imaging Analysis Kit. Figures 2.2 and 2.3 show ground truth versus estimated shared latent trajectories and timescales, as well as the correlation between the true and estimated subject topographies for S-GPFA, with and without smoothness amplification. As Figure 2.2 depicts, with $\lambda = 1$, although S-GPFA outperforms SRM in terms of revealing true latent trajectories and subject topographies, it is unable to accurately discover the temporal dynamic of the shared responses, rendering the estimated latent trajectories and subject topographies non-informative. However, as results shown in Figure 2.3 suggest, increasing λ not only allows S-GPFA to estimate timescales and shared latent trajectories accurately, but also increases the accuracy of determining subject topographies, resulting in more meaningful – and in practice, more interpretable – description of observed data.

2.4 fMRI Datasets

We used the Raider [22] and Sherlock [23] datasets in the first two experiments of Section 2.5. The Raider dataset collects recordings of 1000 voxels from the ventral temporal cortex, for 10 healthy adult participants passively watching the full-length movie "Raiders of the Lost Ark". In the Sherlock dataset, 17 subjects watched the same episode of the TV series "Sherlock" while 481 voxels from the posterior medial cortex were recorded. Further details of both datasets, including inclusion criteria and demographic and clinical characterization, have been published elsewhere by the original study authors [22, 23].

The third experiment uses the "Social Processes Initiative in the Neurobiology of the Schizophrenia(s)" (SPINS) dataset [24]. Relevant fMRI acquisition and preprocessing steps are summarized in the following section. Table 2.1 summarizes essential information about each dataset used in the experiments.

Table 2.1: Datasets used in the experiments

Dataset	Subjects	TRs	Voxels/Regions	ROI
SPINS (EA Task) [24]	332	See Table 2.2	400 regions [25]	whole brain
Raider [22]	10	2203	1000	ventral temporal cortex (VT)
Sherlock[23]	17	1976	481	posterior medial cortex (PMC)

2.4.1 SPINS dataset

We used a subset (332 subjects) of the NIMH Data Archive study "Social Processes Initiative in the Neurobiology of the Schizophrenia(s)" (SPINS) [24], which records whole-brain activity, for 332 adults with (187 subjects) and without (145 subjects) schizophrenia, while performing the Emotional Accuracy (EA) task. Participants in this task watch video vignettes of emotional narratives, while providing real-time ratings of emotional valence.
We selected SPINS as an experimental dataset because it includes participants with and without schizophrenia, and we held that the anticipated variability in brain structure, function, and cognitive performance would provide an interesting test of S-GPFA.

Emotional Accuracy (EA) Task The EA task collects fMRI as participants watch videos of an actor ('target') recounting autobiographical events. In total, 9 videos were shown, lasting for between 2-2.5 minutes. Participants provide ratings of the target's valence on a 9-point scale (1=extremely negative, 9=extremely positive) in real time via button press. The tasks' primary dependent measure, the EA score, is the correlation between the participant's ratings of the targets' emotions, and the "gold standard" rating of the targets' ratings of their own emotions, calculated in 2 second time epochs. Table 2.2 presents information regarding each EA video.

Table 2.2: EA videos

Video	Valence	Emotion	Description	Narrator	Timepoints
1	positive	delighted	trip	female	85
2	negative	sad	soccer	male	73
3	positive	amused	comedian	female	55
4	negative	anger	paycheck	male	72
5	positive	delighted	wedding	male	69
6	positive	amused	movie	male	73
7	negative	sad	death	female	59
8	negative	anger	room-mate	female	89
9	negative	anger	truck	female	64

Acquisition and Preprocessing The fMRI acquisition was an echo-planar sequence, with Repetition Time (TR) of 2000 ms, Echo Time (TE) of 30 ms, voxel size of 3 mm isotropic, 50 volumes, and a 64364 matrix; importantly, all parameters were matched as closely as possible across the study's three centers and six scanners, within limitations of hardware. We removed the first 3 TRs of each of the 9 video runs, regressed known confounds, detrended mean values, and applied a low pass and high pass filter. Confound regression adjusted for six rigid-body motion parameters (i.e., transformation and rotation in each 3-dimensional plane), two nuisance confounds estimated by

physiological fluctuations in white matter and cerebral spinal fluid, and the first three components based on anatomical noise correlation methods calculated by fMRIprep [26]. We did not apply spatial smoothing. The anatomical image used in preprocessing was a fast-gradient echo T1 with a 2300ms repetition time, 0.9 mm isotropic voxels, no slice gap, and an interleaved ascending acquisition. We parcellated the brain into 400 functionally defined cortical regions of interest using the Schaefer atlas [25], interpreted in conjunction with the Yeo 17 networks [27], as depicted in Figure 2.4.

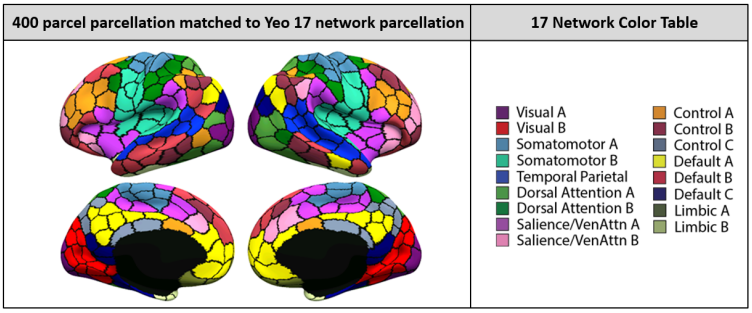


Figure 2.4: Visualization of the Schaefer 2018 Local-Global 400 parcellation in fslr32k space. Parcels were colored to match Yeo 17 network parcellation [27]. Taken from [25].

The SPINS dataset had, at the time of data request, 451 consented subjects that continued to meet eligibility criteria over three study visits. Of these, 403 subjects completed the 3 fMRI acquisitions required by the 9-video EA task, and 372 subjects provided usable EA ratings data (i.e., complied with the task, did not fall asleep). Of these, 332 subjects were found to have usable data (we discarded data gauged to be of insufficient quality on the basis of technician error, irreparable scanner artifacts, and/or high participant motion, defined as a mean frame-wise displacement of 0.5 mm or greater over the course of any EA task video). The final dataset of 332 subjects included 187 individuals with and 145 individuals without schizophrenia.

2.5 Experiments

We conducted three experiments to demonstrate the utility of S-GPFA. First, inter-subject similarity, allows us to test the ability of our model in finding dynamical components shared among all subjects' recordings. Second, time segment matching measures generalization of learned temporal dynamics to unseen subjects and observations. In the third experiment, we apply our model to a multi-subject dataset, allowing us to evaluate and interpret subjects' topographies and group-specific temporal dynamics.

2.5.1 Inter-subject Similarity

As suggested by [7], we may examine the extent to which a shared response exists among subjects in observation space as follows: Leaving one subject out, we find the average response of the remaining subjects; next, we calculate the Pearson correlation over voxels between the left-out and averaged responses for each time point. Averaging over all combinations of left-out subjects (in the Fisher Z domain) provides a measure of inter-subject similarity for every time point.

In order to examine the ability of our model to find shared temporal components of participants' fMRI recordings, we calculated inter-subject similarities in the shared space learned by the model. In this case, data from all subjects were split in time into two equally-sized parts. Using the first part, we trained a model to learn subject-specific topographies. Next, we mapped the second part of the data into the shared space using the learned topographies, separately for each subject (Section 2.3). Finally, to find the similarity of mapped responses at each time point, we performed a similar leave-subject-out procedure in the shared space by finding Pearson correlations of each subject and the average of the other subjects, over latent space components. Figure 2.5 shows histogram plots of inter-subject similarities for the Raider dataset, calculated in voxel space as well as shared spaces learned by S-GPFA and SRM. Observed similarity in the shared embedding space found by S-GPFA confirms that temporal dynamics found by our model are common over all subjects.

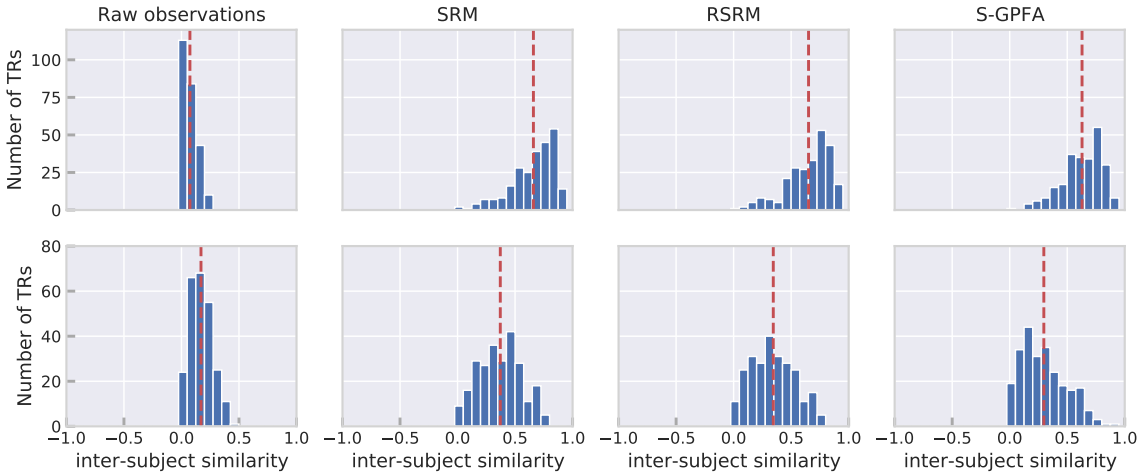


Figure 2.5: Inter-subject similarity. Top: Raider dataset (10 subjects, first 500 TRs). Bottom: Sherlock dataset (17 subjects, first 500 TRs), comparing raw observations, SRM, RSRM, and S-GPFA. We used $P = 10$ shared space dimensions. The dashed line indicates mean similarity over TRs.

2.5.2 Time Segment Matching

A time segment matching experiment, as first introduced by [22], can evaluate how shared dynamics found by S-GPFA generalizes to new subjects and unseen data. We follow the modified version of the experiment presented in [7], where the task is to locate an unseen segment of a test subject's response in time. We partition the fMRI dataset in time into two equal-size parts and leave one subject out to use the remaining subjects as training participants. We shall note the four resulting parts as $\boldsymbol{Y}_1^{\text{train}}$, $\boldsymbol{Y}_1^{\text{test}}$, $\boldsymbol{Y}_2^{\text{train}}$, and $\boldsymbol{Y}_2^{\text{test}}$.

As shown in the top diagram in Figure 2.6, the first half of the data is used to learn subject-specific topographies. First, an S-GPFA model is fit to the first half of training subjects' data $\boldsymbol{Y}_1^{\text{train}}$, to learn shared latent trajectories $\boldsymbol{X}_1^{\text{train}}$ and subject-specific topographies $\boldsymbol{W}^{\text{train}}$. Next, using the learned shared trajectories $\boldsymbol{X}_1^{\text{train}}$ and the first half of the test subject's data $\boldsymbol{Y}_1^{\text{test}}$, topographies for the held-out subjects $\boldsymbol{W}^{\text{test}}$ is calculated, as discussed in Section 2.3. The second half of the data is used for measuring time segment matching accuracy. More specifically, given the second

half of training subjects' data, $\boldsymbol{Y}_2^{\text{train}}$, we wish to locate an unknown segment from the held-out subject's data $\boldsymbol{Y}^{\text{query}}$ in time. By only using the raw observations as a baseline, one can first find the average response of training subjects, then report the time where $\boldsymbol{Y}^{\text{query}}$ and the average response are maximally correlated. Moreover, using learned subject topographies, we can transform $\boldsymbol{Y}_2^{\text{train}}$ and $\boldsymbol{Y}^{\text{query}}$ into the shared space using $\boldsymbol{W}^{\text{train}}$ and $\boldsymbol{W}^{\text{test}}$ as discussed in Section 2.3, to find $\boldsymbol{X}_2^{\text{train}}$ and $\boldsymbol{X}^{\text{query}}$. Using a similar correlation classifier, we can locate the test segment at the point where $\boldsymbol{X}_2^{\text{train}}$ and $\boldsymbol{X}^{\text{query}}$ are maximally correlated. Figure 2.6 compares the time segment matching accuracy for different query segment lengths. S-GPFA demonstrates similar performance as SRM in terms of time segment matching accuracy. This posits that shared temporal dynamics (timescales) found in training subjects can be generalized to new subjects with unseen observations.

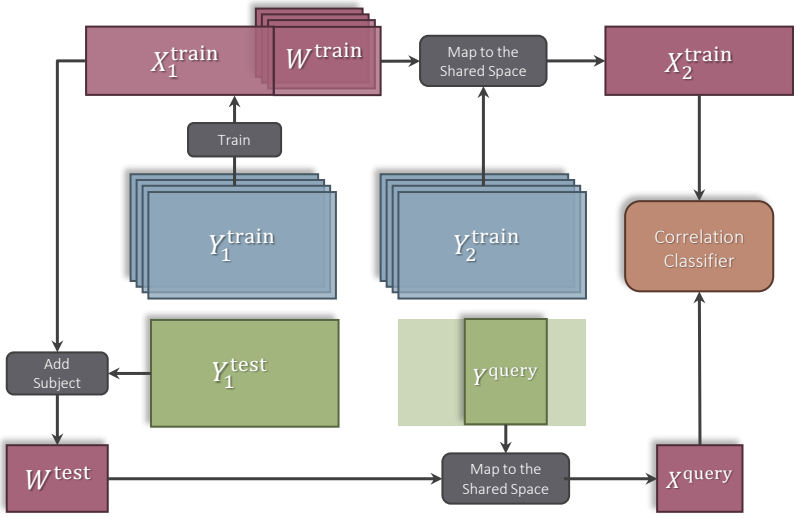
2.5.3 Application: SPINS Dataset

For our final set of experiments, we applied S-GPFA to a subset (332 subjects) from the NIMH Data Archive study "Social Processes Initiative in the Neurobiology of the Schizophrenia(s)" (SPINS) who completed the Emotional Accuracy (EA) task. The EA task collects fMRI as participants watch videos of an actor ('target') recounting autobiographical events. In total, 9 videos were shown, lasting for between 2-2.5 minutes. Participants provide ratings of the target's valence on a 9-point scale (1=extremely negative, 9=extremely positive) in real time via button press. The tasks' primary dependent measure, the EA score, is the correlation between the participant's ratings of the targets' emotions, and the "gold standard" rating of the targets' ratings of their own emotions, calculated in 2 second time epochs. We selected SPINS as an experimental dataset because it includes participants with and without schizophrenia, and we held that the anticipated variability in brain structure, function, and cognitive performance would provide an interesting test of S-GPFA.

Consistency of Functional Topographies

In our first analysis, we examined whole-brain variation in activation during the EA task, across all subjects. Learned factor loading matrices $\boldsymbol{W}$ in S-GPFA are, by model definition, subject-specific basis vectors that generate each subject's observation from a shared set of latent trajectories: $\boldsymbol{W}^{(m)}\boldsymbol{X} \simeq \boldsymbol{Y}^{(m)}$. Therefore, columns of factor loadings $\{\boldsymbol{W}_{:,p} \in \mathbb{R}^Q\}_{p=1}^P$, for different subjects, are perturbed versions of an activation template. This allows the model to capture functional variability across subjects. Hence, we can examine the amount of subject variability in each individual brain region of interest (ROI) for a given task, by looking at the variance of learned topographies over participants. To this end, for every topography p of subject m, we first normalize $\boldsymbol{W}_{:,p}^{(m)}$, then we find the variance of these normalized topographies over subjects. This will result in P variance values for each of Q regions indicating a measure of subject variability, as shown in Figure 2.7. Note that, for the present analysis, we used cortical parcellations from the Schaefer atlas (400 ROIs) [25], grouped in accordance with the Yeo 17 network parcellation [27], as shown in Figure 2.4.

Examination of subject variance in functional topography itself shows variability across ROIs. Perhaps unsurprisingly, the lowest variance (blue) is evident in the motor network, engaged similarly by the task's continuous button-press demands, and in temporal lobe areas related to audition, engaged by the audio aspect of the task. Excitingly, we see lower consistency (red) in the distributed frontal-parietal ('Control A') network, which contains the inferior frontal gyrus (IFG) and



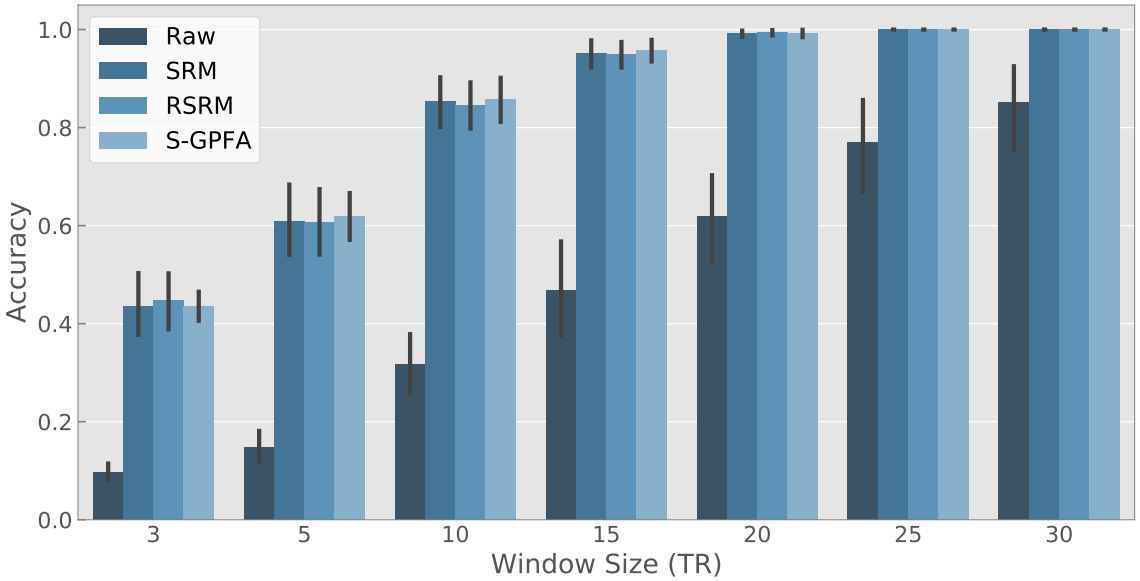


Figure 2.6: Time Segment Matching Experiment. Top: schematic procedure of the experiment. Bottom: Time segment matching accuracy for Raider dataset (10 subjects, first 400 TRs). We used $P = 10$ for SRM, RSRM, and S-GPFA. Error bars show $\pm$ standard deviations.

inferior parietal lobule (IPL), that together constitute the canonical human mirror neuron system [28]. That we were able to identify high variability in a network believed to subserve social cognition provides important proof of principle that S-GPFA can identify meaningful variance within a heterogeneous sample. We expect that S-GPFA will enable important tests of both exploratory and hypothesis-driven examinations of functional topography in psychiatric disorders.

Figure 2.8 presents the result of the analysis in the main text for all nine EA videos of the SPINS dataset.

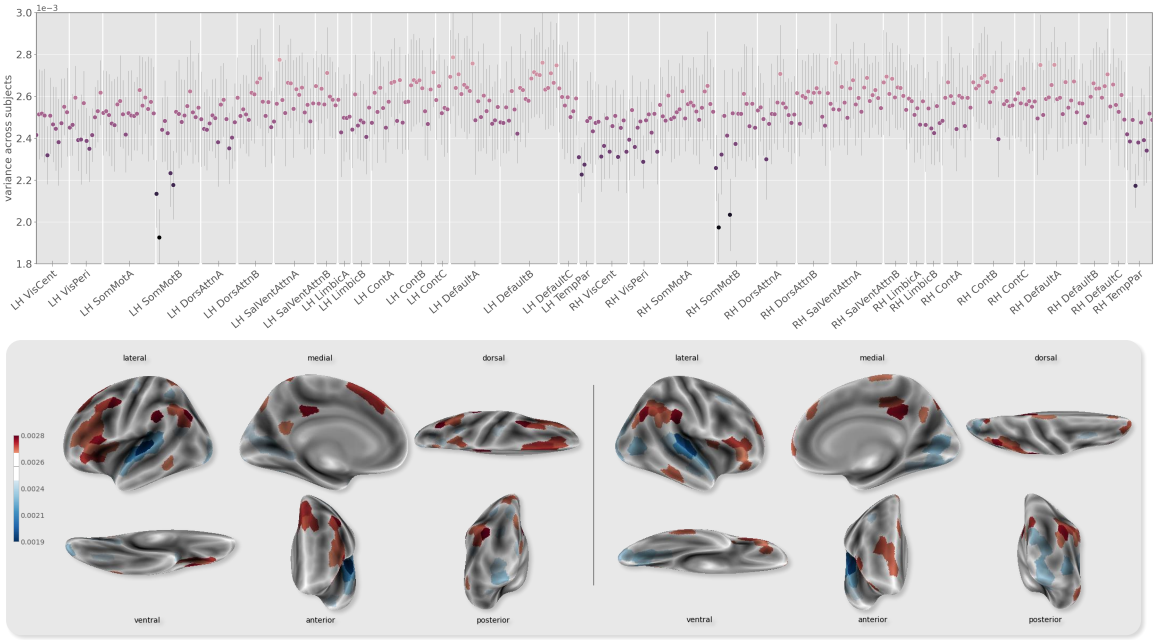


Figure 2.7: Top: Variance of normalized $P = 20$ topographies across subjects (Y-axis) for different regions (X-axis). Colored dots indicate the mean (across different topographies) of variances for each region, and error bars show $\pm$ standard deviation. Bottom: Brain surface plots of average variance of normalized topographies, for left and right hemispheres. Blue (red) indicates higher (lower) consistency across subjects. In both panels, the second of nine EA videos is shown: all videos showed comparable patterns. See Figure 2.8 for other EA videos.

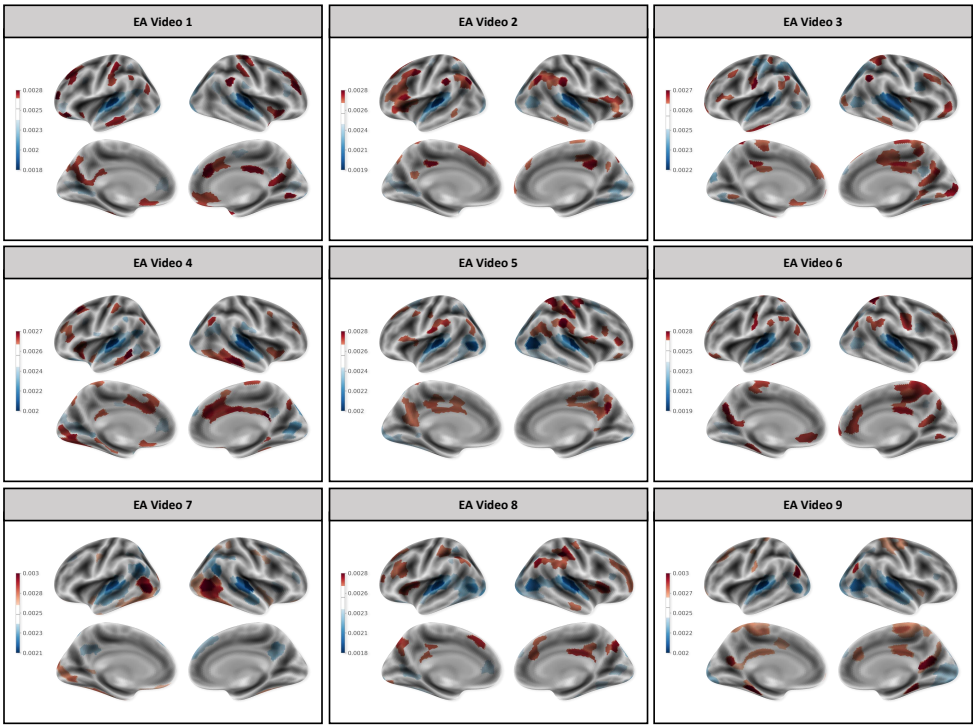


Figure 2.8: Consistency of functional topographies – 9 EA Videos. Brain surface plots of average variance of normalized topographies, for left and right hemispheres. Blue (red) indicates higher (lower) consistency across subjects.

Group-specific Temporal Dynamics

Next, we employed S-GPFA to examine group-specific temporal dynamics. Our method is illustrated on the left of Figure 2.9. First, a *global* model is trained using subjects from all groups. Residuals of the global model, as defined by $\boldsymbol{Y}^{(m)} - \boldsymbol{W}^{(m)}\boldsymbol{X}$, for $m \in \{1, \ldots, M\}$, allow us to remove components of the observations that are shared among all groups [7]. Next, separately in each group, two models are trained over the residuals of the global model. These *local* models capture group-specific dynamical components in observations after the globally shared components of the data are removed. We applied this approach to the SPINS dataset, opting to split the sample into two smaller groups comprised of individuals demonstrating EA scores within the top and bottom 20th percentiles, respectively. This bifurcation allowed us to isolate subjects with distinctly strong and poor socioemotional cognition, irrespective of schizophrenia diagnosis [29].

The right of Figure 2.9 shows group-specific timescales discovered by the local models, after extraction of the residuals from the global model, i.e., after all signals with a global structure have been removed. In the local models, we observe that the group with the strongest EA performance shows less temporal variability, i.e., the dynamical patterns in those individuals with the greatest socioemotional cognitive capacity unfolded over slower timescales, in contrast to those participants with comparatively impoverished capacity. This observation is consistent with broad evidence of

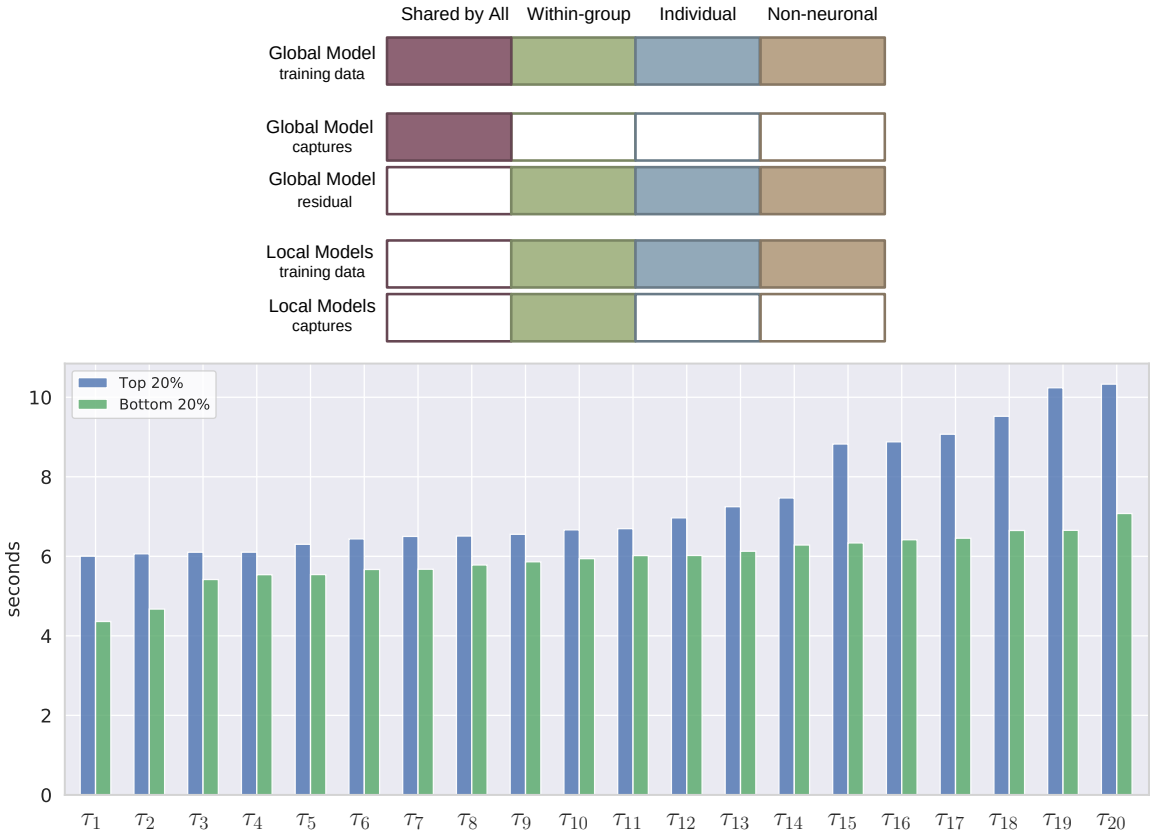


Figure 2.9: Group-specific Temporal Dynamics. Top: Different data components captured using the global and local models. Bottom: Group-specific timescales (sorted by magnitude) discovered using local models, after removing temporal patterns shared across all subjects.

a relationship between domain-specific task performance, and observed modularity versus flexibility of task-relevant brain networks, e.g., [30], and the recent dissociation that strong performance on tasks involving little executive function or cognitive control (consistent with EA task demands) may be subserved by modular network activation [31]. Importantly, the isolation of low-dimensional group-specific timescales allows for the identification of differences of small effects, which may be statistically occluded in global models.

2.6 Summary and Concluding Remarks

We introduced S-GPFA, a novel application of Gaussian Process Factor Analysis for finding subject-specific topographies and shared temporal dynamics in multi-subject fMRI datasets. The proposed model simultaneously allows functional aggregation, dimensionality reduction, and dynamical modeling of fMRI data.

First, we delineated the need for amplifying the smoothness loss in the training objective, which was supported by the evaluation on simulated data. Then, we conducted three sets of experiments on human fMRI datasets, designed in turn to show that the dynamical structures found by our model are (i) shared between all subjects; (ii) generalizable to new subjects; and (iii) may be extended to isolate dynamical elements within groups. We showcased the utility of our model by analyzing its learned model parameters in the large multi-site SPINS dataset, on a social cognition task from participants with and without schizophrenia.

This work is motivated by the growing interest in modeling dynamics in fMRI, which has increased alongside the recognition that functional topographies vary substantially across individuals. The goal of cognitive neuroscience is to understand the brain structures and functions that drive human thought processes. Finding brain structures or functions that can predict mental health issues could help identify the causes of these disorders. Importantly, recognizing these correlates provides specific targets for testing drugs or behavioral treatments that may help slow or reverse the progression of the disease, offering a significant advancement in enhancing the quality of life for patients who do not respond to current treatments.

Chapter 3

Sequential Gradient Coding

3.1 Introduction

Stragglers, or slow processing workers, are common in distributed computing systems that potentially delay the timely completion of computational jobs. One may use the following analogy to understand the importance of erasure codes in the context of distributed computing. Consider an $[n, k]$ erasure code that encodes k bits of data to produce $n > k$ coded bits for transmission over a channel that could erase a few bits. Even if the channel erases part of the n coded bits, the redundant $(n - k)$ bits aid in data recovery. The final job result to be obtained by the master node in a distributed computing system may be compared to the k-bit data that is being encoded.

Computations undertaken by a worker may therefore be thought of as a coded bit of an erasure code. Naturally, delayed responses due to stragglers correspond to bit erasures. The core idea behind coding for distributed computing is that erasure codes offer a way to efficiently add redundancy to computations or data so that jobs can still be completed on schedule, even when some computations are unavailable.

We consider a distributed system consisting of a master node and n computing nodes (will be referred to as workers). We are interested in computing a sequence of gradients $g(1)$, $g(2)$, $\ldots$, $g(J)$. If we naively distribute the computation of each gradient among n workers, delayed arrival of results from any of the workers will become a bottleneck. Such a delay could be due to various reasons such as slower processing at workers, network issues leading to communication delays, etc. Irrespective of the actual reason, we will refer to a worker providing delayed responses to the master node, as a straggler.

In a recent work [32], the authors propose Gradient Coding (GC) to distribute the computation of a single gradient across multiple workers in a straggler-resilient manner. Using (n, s)-GC, the master node is able to compute the gradient as soon as $(n - s)$ workers respond (s is an integer such that $0 \leq s < n$). Adapting GC to our setting, we will have a scheme where $g(t)$ is computed in round-t, and in every round, up to s stragglers are tolerated (the concept of a round will be formally introduced in Section 3.3).

Experimental results in [33], along with findings from the current work, demonstrate that intervals of "bad" rounds (where each round has a large number of stragglers) are followed by "good" rounds (where there are relatively fewer stragglers), leading to a natural temporal diversity. In this scenario,

GC will enforce a large s, as dictated by the number of stragglers expected in bad rounds, leading to a large computational load per worker.

The natural question to ask here is: *do better coding schemes exist that exploit the temporal diversity and achieve better performance?*. In this chapter, we present *sequential gradient coding* schemes that answer this question affirmatively, in settings without strict dependency between consecutive jobs, e.g., concurrently training multiple models, as discussed in Section 3.3.1. In contrast to existing GC approaches where coding is performed only across workers, we propose two coding schemes that explore coding across rounds as well as workers.

Our first scheme, Selective Reattempt Sequential Gradient Coding (SR-SGC) (Section 3.5.2), is a natural extension of the (n, s)-GC scheme, where unfinished tasks are selectively reattempted in future rounds. Despite the simplicity, the SR-SGC scheme tolerates a strict superset of straggler patterns compared to (n, s)-GC, for the same computational load.

In our more involved second scheme, namely, Multiplexed Sequential Gradient Coding (M-SGC) (Section 3.5.3), we divide tasks into two sets: those that are protected against stragglers via reattempts, and those that are protected via GC. We then carefully multiplex these tasks to obtain a scheme where the computational load per worker is significantly reduced. In particular, the load is decreased by a factor of s compared to the (n, s)-GC scheme.

Our experiments (Section 3.6) on the AWS Lambda service involving 256 worker nodes demonstrate that the M-SGC scheme significantly reduces runtime compared to traditional Gradient Coding, in real-world conditions involving naturally occurring, non-simulated stragglers.

The results presented in this chapter have been published in [2], where the author holds joint first authorship. Contributions primarily focused on the real-world evaluation and implementation of the schemes.

3.2 Related Works

3.2.1 Gradient Coding

The application of erasure coding to address stragglers in distributed computing systems was first explored in [34], specifically within the context of distributed matrix multiplication. A thorough overview of recent advancements in this field is provided in the survey [35].

Consider the following toy example to demonstrate the key idea employed in [34]. Assume that the master node wants to distribute the job of computing $A^\top x$ across $n = 3$ workers, where any one of the workers could be a straggler. The master node will partition the columns of A as $[A_0 \mid A_1]$. It will now pass $\{A_0, x\}$, $\{A_1, x\}$ and $\{A_2 \triangleq (A_0 + A_1), x\}$ to workers 0, 1 and 2, respectively. Worker-i will attempt to compute $A_i^\top x$ and return the result. Clearly, the master node can compute $A^\top x$ as soon as it receives results from any two workers and hence the system is resilient against one straggler.

The gradient coding framework proposed in [32] examines the possibility of using erasure codes to perform distributed stochastic gradient descent in a straggler-resilient manner. We borrow the following example from the original paper to highlight the key idea. Let the dataset $\mathbf{D}$ be partitioned into $\mathbf{D}_0$, $\mathbf{D}_1$ and $\mathbf{D}_2$. Workers 0, 1 and 2 store $\{\mathbf{D}_0, \mathbf{D}_1\}$, $\{\mathbf{D}_1, \mathbf{D}_2\}$ and $\{\mathbf{D}_2, \mathbf{D}_0\}$, respectively. Worker-0 is supposed to compute partial gradients $\{g_0, g_1\}$ on data chunks $\{\mathbf{D}_0, \mathbf{D}_1\}$ and then

return $\ell_0 \triangleq \frac{g_0}{2} + g_1$ to the master node. Similarly, workers 1 and 2 attempt to return $\ell_1 \triangleq g_1 - g_2$ and $\ell_2 \triangleq \frac{g_0}{2} + g_2$, respectively. It can be inferred that $g \triangleq g_0 + g_1 + g_2$ can be computed from the results returned by any two workers. For instance, suppose workers 0 and 1 have returned their results and worker-2 is a straggler. In this case, we have $g = 2(\frac{g_0}{2} + g_1) - (g_1 - g_2) = 2\ell_0 - \ell_1$.

3.2.2 Ignoring Stragglers

In [36] the authors use synchronous minibatch SGD, and propose to simply ignore a certain number of stragglers and do a gradient step when enough workers are done. While this approach can reduce the delay caused by slower workers, it may also decrease the effective batch size and require more overall rounds to achieve convergence. [36] explores this trade-off and estimates the optimal number of stragglers to ignore for the fastest convergence.

However, as noted in [32], if certain workers are more likely to be stragglers, the samples on those machines may be underrepresented in the final model. There may also be situations where the gradient calculation for specific samples takes longer, for example in variable-sized inputs, conditional computation, and adaptive or sparse techniques. Consequently, computationally-intensive examples might be frequently ignored.

Another disadvantage of ignoring stragglers is that the resulting gradient updates are noisy, and may not be used with some accelerated gradient update schemes [32], because errors can accumulate rapidly, potentially leading to divergence of the method. [37].

Finally, [32] compares the generalization error between gradient coding and ignoring stragglers, demonstrating that gradient coding achieves significantly better generalization error in a practical distributed training setting with natural delay profiles using EC2 clusters.

Therefore, while ignoring stragglers offers a straightforward solution in scenarios where the associated drawbacks can be managed, it cannot replace gradient coding methods. Even in some cases, it can be used alongside coding methods to enhance straggler resilience in synchronous distributed training settings [38].

3.2.3 Gradient Coding Across Time

The terminology of "sequential gradient coding" initially appears in [39]. Similar to the current work presented in this chapter, the work in [39] extends the classical GC setting [32], by exploiting the temporal dimension. The authors of [39] provide a *non-explicit* coding scheme that is resilient against communication delays and does not extend when stragglers are due to computational slowdowns. In contrast, we propose two *explicit* coding schemes that are oblivious to the reason for straggling behavior. Moreover, under equivalent straggler conditions, the computational load requirements of the newly proposed SR-SGC scheme and the scheme in [39] are identical, whereas the new M-SGC scheme offers a significantly smaller load requirement. The works [40, 41] explore a trade-off between communication and straggler resiliency in GC. Several variations to the classical GC setting appear in papers [42–46].

There is a rich literature on distributing matrix multiplication or more general, polynomial function computation (see [33, 34, 47–53] and references therein). In [54], the authors introduce a sequential matrix multiplication framework, where coding across temporal dimension is exploited to reduce the cumulative runtime for a sequence of matrix multiplication jobs. While we also explore the idea of

introducing temporal dimension to obtain improved coding schemes, extending the approach in [54] to our setting yields an inferior solution requiring a large computational load.

3.2.4 Gilbert-Elliot and Sliding-window-based Models

The Gilbert-Elliott (GE) model has been classically used in the literature to model packet failure in communication networks [55]. GE is a 2-state probabilistic model with state transition probabilities as outlined in Figure 3.1. In the context of modeling stragglers, a worker is a straggler if and only if it is in state $\mathcal{S}$ and a non-straggler otherwise. If a worker is a straggler in a given round, it will continue to be a straggler in the next round with probability $(1-p_{\mathcal{S}})$. Similarly, a non-straggler will continue to remain so in the next round with probability $(1-p_{\mathcal{N}})$. In essence, the GE model suggests that straggling behavior occurs periodically and is often followed by non-straggling behavior.

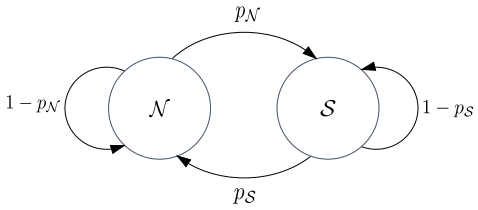


Figure 3.1: The 2-state Gilbert-Elliott model.

In distributed systems, stragglers occur due to reasons such as resource sharing, communication bottlenecks, routine maintenance activities, etc. The periodic spikes in latency caused by several of these factors (see [56, 57]) are naturally captured by the GE model. The authors of [33] make an empirical observation that the GE model can accurately track the state transitions of workers on an Amazon EC2 cluster, particularly in the context of applying erasure codes for straggler mitigation. In the current study, we approximate the GE model using deterministic, sliding-window-based models since they are more tractable in terms of code design (see Section 3.4 for a formal definition of straggler models). As our experiments indicate, such a design strategy finally led to techniques that outperform the baseline GC on an Amazon Lambda cluster.

Note that the sliding-window-based models introduce constraints on the local structure of straggler patterns. For example, in the bursty model, a burst of straggling rounds is followed by a period when the worker is a non-straggler. This provides a simplified approximation of the GE model, capturing the dominant set of straggler patterns associated with the GE channel (any other straggler pattern will be handled through wait-outs). Similarly, in the arbitrary straggler model, we put a constraint on the total number of straggling rounds in each window without requiring that the straggling rounds be consecutive. Finally, in our proposed setting of sequential gradient coding, it turns out that modeling local constraints on straggler patterns using the sliding-window-straggler model is a natural fit – for a job that starts in round-i and is completed in round-$(i+\Delta)$, the straggler patterns around the interval i to $i+\Delta$ are relevant.

3.3 Sequential Gradient Coding Setting

For integers a, b, let $[a:b] \triangleq \{i \mid a \leq i \leq b\}$ and $[a:b]^* \triangleq \{i \mod n \mid a \leq i \leq b\}$, where n is the number of workers. For integer c, we have $c+[a:b]^* \triangleq \{c+i \mid i \in [a:b]^*\}$. Workers are indexed by

$[0:n-1]$. We are interested in computing a sequence of J gradients $\{g(t)\}_{t\in[1:J]}$. The computation of gradient $g(t)$ is referred to as *job-t*. All gradients are computed with respect to a single data set $\mathbf{D}$ (this assumption is just for simplicity in exposition and our schemes can easily be adapted to include multiple data sets).

Data placement Master node partitions $\mathbf{D}$ into η data chunks $\{\mathbf{D}_0, \mathbf{D}_1, \ldots, \mathbf{D}_{\eta-1}\}$, possibly of different sizes. Each worker-i stores data chunks $\{\mathbf{D}_j\}_{j\in D_i}$, where $D_i \subseteq [0:\eta-1]$. Let $g_j(t)$ denote the t-th partial gradient computed with respect to $\mathbf{D}_j$, i.e., $\sum_{j\in[0:\eta-1]} g_j(t) = g(t)$. Naturally, we also say that the partial gradient $g_j(t)$ corresponds to job-t.

Encoding Master node operates based on a certain concept of *rounds* and it takes $J+T$ rounds to finish computing all the J gradients. Here, $T \geq 0$ is a system parameter which takes integer values. Let $t \in [1 : J+T]$. In round-t, worker-i computes $\gamma_i(t)$ partial gradients with respect to a subset of data chunks it stores. These partial gradients may correspond to any of the jobs $[1:t]$. In other words, the partial gradients computed by worker-i in round-t are from within the set $\{g_j(t')\}_{t'\in[1:t],j\in D_i}$. The process of worker-$i$ computing $\gamma_i(t)$ partial gradients will be referred to as a *task*. Once the $\gamma_i(t)$ partial gradients are computed in round-t, worker-i returns a *task result* $\phi_i(t)$, which is a function of the $\gamma_i(t)$ partial gradients, via an *encoding* step.

Identification of stragglers In each round-t, let $\kappa(t)$ denote the time (in seconds) taken for the fastest worker (say, worker-i) to return the task result. Master node then waits for $\mu\kappa(t)$ ($\mu > 0$ is a *tolerance parameter*) more seconds and possibly more workers will return their task results during this time. Any worker who does not return their task result during this time will be marked as a straggler. Generally, any pending tasks of stragglers are canceled, and the system proceeds to the next round. However, depending on the coding scheme, the master node may decide to *wait out* all stragglers if specific conditions for waiting are met. Clearly, if worker-i is a straggler in round-t, the task undertaken by it has "failed" and task result $\phi_i(t)$ will not be returned to the master node. We note that determining stragglers using the tolerance parameter μ is in line with the earlier work [54].

Decoding At the end of round-$(t+T)$, master node attempts to decode $g(t)$ using all available results from the set $\{\phi_i(t')\}_{i\in[0:n-1],t'\in[1:t+T]}$. Essentially, master node aims to complete job-t at the end of round-$(t+T)$. Decoding is assumed to incur negligible computational cost compared to partial gradient computation. Since workers begin computing partial gradients for any job-t starting from round-t onward, the parameter T essentially represents a delay in job completion.

Normalized computational load per worker Let d denote the number of data points in $\mathbf{D}$. Let $d_i(t)$ denote the total number of data points across which worker-i computes $\gamma_i(t)$ partial gradients in round-t. The normalized (computational) load at worker-i in round-t is then given by $L_i(t) \triangleq d_i(t)/d$. Note that the normalized load for an uncoded scheme is $1/n$.

3.3.1 Dependency Between Jobs and Concurrent Training

The flexibility of coding across rounds enables us to design coding schemes that feature lower computational load and tolerate practically motivated straggler models by exploiting the temporal diversity of stragglers. This in turn leads to smaller cumulative runtime (in seconds) for finishing jobs if a delay of T rounds is tolerable.

However, some applications cannot inherently accommodate delays in gradient computation due to the dependencies between subsequent jobs. Particularly, when training a single neural network, the calculation of $g(t)$ must await the model's weight update using $g(t-1)$.

Managing dependencies between jobs requires an appropriate selection of the parameter T. If job-i_2 depends on the computation of $g(i_1)$, with $i_1 < i_2$, then T should be chosen to satisfy $T \leq i_2 - i_1 - 1$. An important scenario is *pipelining* the training of M neural networks. Here, jobs $Mi+1, \ldots, Mi+M$ represent the i-th iterations ($i = 0, 1, \ldots$) of training models 1 through M, respectively. Here, we need $T \leq M-1$. Therefore, when the decoding delay of the sequential coding scheme is T rounds, training at least $M = T+1$ models needs to be pipelined.

Consider concurrently training four neural networks, denoted by $\mathrm{NN}_1, \mathrm{NN}_2, \mathrm{NN}_3$, and NN_4, as performed in Section 3.6. For each neural network NN_j where $j = 1, 2, 3, 4$, we use $\mathbf{w}_j^{(0)}$ to represent the initial weights and $\mathbf{w}_j^{(i)}$ to denote the weights after i rounds of gradient descent updates:

$$\mathbf{w}_j^{(i)} = \mathbf{w}_j^{(i-1)} - \epsilon_j^{(i)} \mathbf{g}_j^{(i-1)}, \tag{3.1}$$

where $\mathbf{g}_j^{(i-1)}$ denotes the gradient associated with neural network NN_j with weights $\mathbf{w}_j^{(i-1)}$ and $\epsilon_j^{(i)}$ denotes the learning rate. We assume that the training of $\mathrm{NN}_1, \mathrm{NN}_2, \mathrm{NN}_3$ and NN_4 is interleaved across rounds – i.e., model updates for the *interleaved training* uses the following schedule:

- Weights of NN_1 are updated in rounds: 1 (Initialization), 5, 9, 13, ...
- Weights of NN_2 are updated in rounds: 2 (Initialization), 6, 10, 14, ...
- Weights of NN_3 are updated in rounds: 3 (Initialization), 7, 11, 15, ...
- Weights of NN_4 are updated in rounds: 4 (Initialization), 8, 12, 16, ...

Thus if we consider the training of say NN_1, then there two steps:

1. At the start of round-$(4i+1)$, the server will generate $\mathbf{w}_1^{(i)}$. It will have decoded the gradient vector $\mathbf{g}_1^{(i-1)}$ by the end of round-$4i$ and performed the SGD update in (3.1).
2. The server will issue a request at the start of round-$(4i+1)$ to the workers to compute the associated partial gradients such that the gradient vector $\mathbf{g}_1^{(i)}$ can be computed by the end of round-$(4i+4)$.

By following a similar approach for each model it is clear that for neural network NN_j, when computing the gradient $\mathbf{g}_j^{(i)}$, a request will be issued by the server to the clients at the start of round-$(4i+j)$ and the computation must be completed by the end of round-$(4i+j+3)$. Thus our interleaved approach is designed so that that each of the gradient vectors can be decoded within a span of $M = 4$ rounds.

Our method can be viewed as a pipelined approach for training multiple neural networks. Note that in our current setting, we assume that the parameters of only one model can be updated in each round, which arises in resource-constrained devices. Our method also naturally complements other approaches for parallel training of multiple models and leverages the temporal structure of straggler patterns to achieve speedups. We also emphasize that we do not require multiple neural network models to be trained on the same dataset. Each neural network model can be trained on a different dataset. We also do not require that the architecture of the neural networks be identical. Nevertheless, we point out that our setting would be most efficient when the compute time for the gradients for each model is approximately the same. We believe this is a rather benign requirement. Finally, we discuss a few applications where multiple-model training arises naturally.

- In the training of deep learning models, we are often required to perform a search over various hyperparameters and this is done through some form of a grid search [58]. Each choice of hyperparameter corresponds to a new model. Ultimately the ideal hyperparameters are selected through a validation set.
- In ensemble learning [59], several models need to be trained simultaneously. Their predictions are then combined through some averaging mechanism.
- Since our approach can use completely different datasets for each of the models, it is also applicable in settings of "multi-model learning" [60, 61], where multiple datasets are used for different models. For instance, sensors deployed in time-varying or periodic environments (e.g., day/night camera images, orbiting satellite data, etc.) or collecting different modalities (speech, images, etc.) would naturally generate multiple datasets where different models could be trained for each one. Multi-model learning has also been applied in real-time video surveillance applications [62].

3.4 Straggler Models

Deterministic straggler models have been employed in many works as a reliable approximation of the GE model; for instance, in the early classical work [63], in [64] in the context of control systems, [65] in the context of low-latency communications and [54] in the context of distributed matrix multiplication. See Section 3.2 for more details.
For the sake of our analysis and code design, we will refer to the following three straggler models. However, as explained in Remark 3.1 at the end of this section, and validated in Section 3.6, our coding schemes apply to any naturally occurring straggler patterns.

(B, W, λ)-bursty straggler model Let $S_i(t)$ be an indicator function which is 1 if worker-i is a straggler in round-t and 0 otherwise. The straggler model is defined by the following two properties:

1. *Spatial correlation:* In every window W_j of the form $[j : j+W-1]$ consisting of W consecutive rounds starting at round-j, there are at most $\lambda \leq n$ distinct stragglers. i.e., size of the set of workers given by $\{i \mid S_i(t) = 1 \text{ for some } t \in W_j\}$ is at most λ.

2. *Temporal correlation:* For any worker-i, first and last straggling slots (if any) are not more than $B-1$ rounds apart in every window W_j. i.e., if $S_i(t) = 1$, for some $t \in W_j$, then $S_i(l) = 0$ for all $l \in [t + B : j + W - 1]$.

Clearly, the parameters λ, B satisfy: $0 \leq \lambda \leq n$, $1 \leq B \leq W$.

(N, W', λ')-arbitrary straggler model This is a natural extension of the (B, W, λ)-bursty straggler model where we consider arbitrary stragglers instead of bursty stragglers. As in the bursty model, in every window W'_j of the form $[j : j + W' - 1]$, there are at most λ' distinct stragglers. Moreover, for every worker-i, $\sum_{t \in W'_j} S_i(t) \leq N$. The parameters λ', N satisfy: $0 \leq \lambda' \leq n$, $0 \leq N \leq W'$.

***s*-stragglers-per-round model** In this model, in each round, at most s workers can be stragglers. Here s is an integer satisfying $0 \leq s < n$.

Remark 3.1. While the actual behavior of stragglers may not always align with the model assumed during code design, we ensure that every job-t is completed by the end of round-$(t + T)$. This is achieved as follows: if the actual straggler pattern diverges from the expected model in any given round, the master node will wait for the stragglers to return their results within that round. This approach allows the actual straggler pattern to "effectively" continue to conform to the assumed straggler model. For instance, we evaluate the performance of our coding schemes on a large AWS cluster in Section 3.6, in the presence of naturally occurring stragglers.

3.5 Proposed Sequential Gradient Coding Schemes

3.5.1 Preliminaries: Gradient Coding

We present here a summary of the (n, s)-GC scheme. The data set $\mathbf{D}$ is partitioned into $\eta = n$ equally sized data chunks $\mathbf{D}_0, \mathbf{D}_1, \ldots, \mathbf{D}_{n-1}$. Worker-$i$ stores $s+1$ data chunks $\{\mathbf{D}_j\}_{j \in [i:i+s]^*}$. With respect to each $\mathbf{D}_j$ stored, worker-i computes the partial gradient g_j. Hence, worker-i computes the following $s + 1$ partial gradients: $\{g_j\}_{j \in [i:i+s]^*}$. Worker-$i$ then transmits the result, which is a linear combination of the $s + 1$ partial gradients, $\ell_i = \sum_{j \in [i:i+s]^*} \alpha_{i,j} g_j$. The coefficients $\{\alpha_{i,j}\}$ are designed so that if master node has access to results returned by any $n - s$ out of n workers, $g \triangleq g_0 + g_1 + \cdots + g_{n-1}$ can be computed. In other words, for any $\mathcal{W} \subseteq [0 : n-1]$ with $|\mathcal{W}| = n - s$, there exist coefficients $\{\beta_{\mathcal{W},w}\}_{w \in \mathcal{W}}$ such that $g = \sum_{w \in \mathcal{W}} \beta_{\mathcal{W},w} \ell_w$. Clearly, the scheme tolerates s stragglers. Applying the (n, s)-GC scheme to our framework, we get a scheme that computes $g(t)$ in round-t (i.e., delay $T = 0$). Each worker-i in round-t works on a task corresponding to job-t. Specifically, worker-i computes $\{g_j(t)\}_{j \in [i:i+s]^*}$ and returns $\ell_i(t) = \sum_{j \in [i:i+s]^*} \alpha_{i,j} g_j(t)$. Normalized load is given by $L_{\text{GC}} = (s + 1)/n$. This scheme tolerates any straggler pattern conforming to the s-stragglers-per-round model. We will now present both our proposed schemes.

3.5.2 Selective Reattempt Sequential Gradient Coding (SR-SGC)

SR-SGC scheme is a natural extension of the (n, s)-GC scheme. We begin with (n, s)-GC as the "base scheme" and whenever there are more stragglers than the base scheme can handle, we will

carefully reattempt certain tasks across time. This simple idea helps the scheme tolerate a strict superset of straggler patterns compared to the classical (n, s)-GC scheme for the same normalized load. We present the formal statement in Proposition 3.1.

Design parameters The parameter set is given by $\{n, B, W, \lambda\}$, where $0 < \lambda \leq n$, $B > 0$, and B divides $W-1$. That is, there exists an integer $x \geq 1$ such that $W = xB+1$. We set $s \triangleq \lceil \frac{B\lambda}{W-1+B} \rceil = \lceil \frac{\lambda}{x+1} \rceil$. The scheme incurs a delay of $T = B$ and the normalized load is $L_{\text{SR-SGC}} = (s+1)/n$, similar to GC.

Scheme outline Recall the notation $\ell_i(t)$ presented in Section 3.5.1. In round-t, worker-i will attempt to compute either $\ell_i(t)$ or else, $\ell_i(t-B)$ as we will see now. Using the property of (n, s)-GC, a job can be finished if master node receives $(n-s)$ task results corresponding to that job. In round-t, if master node finds out that $(n-\nu) < (n-s)$ task results corresponding to job-$(t-B)$ are received in round-$(t-B)$, the minimum additionally required number of $(\nu - s)$ tasks corresponding to job-$(t-B)$ will be attempted in round-t. These tasks will be attempted by workers who did not previously return task results corresponding to job-$(t-B)$ in round-$(t-B)$. Rest of the $(n-\nu+s)$ workers will attempt tasks corresponding to job-t. On the other hand, if job-$(t-B)$ is already finished in round-$(t-B)$, all tasks in round-t correspond to job-t. Let $\mathcal{N}(t)$ denote the number of task results corresponding to job-t returned to master node in round-t. We formally describe the task assignments in Algorithm 3.1. As jobs are indexed in the range $[1 : J]$, for consistency in notation, we assume that all task results corresponding to job-t' are known to master node by default (i.e., $\mathcal{N}(t') = n$), whenever $t' \notin [1 : J]$.

Algorithm 3.1 Algorithm used by master node to assign tasks in round-t

```
Initialize δ ≜ N(t − B).
for i ∈ [0 : n − 1] do
    if δ < n − s and ℓ_i(t − B) is not returned by worker-i in round-(t − B) then
        Worker-i attempts to compute ℓ_i(t − B).
        Set δ = δ + 1
    else
        Worker-i attempts to compute ℓ_i(t).
```

Proposition 3.1. *An SR-SGC scheme designed for parameters $\{n, B, W, \lambda\}$ tolerates any straggler pattern that conforms to either (i) the (B, W, λ)-bursty straggler model or else, (ii) the s-stragglers-per-round model.*

The proof of Proposition 3.1 requires a careful analysis of dependencies between successive rounds due to the repetition of tasks. We refer readers to Appendix D of [2] for complete proof.

Remark 3.2. Let $\lambda < n$. Without selective repetition, classical GC requires $s = \lambda$ to tolerate the (B, W, λ)-bursty straggler model. In SR-SGC, we are able to choose a lower s value of $\lceil \frac{B\lambda}{W-1+B} \rceil$, and as a result, the normalized load is lower as well. In other words, the SR-SGC scheme tolerates a superset of straggler patterns compared to the GC scheme for the same normalized load.

3.5.3 Multiplexed Sequential Gradient Coding (M-SGC)

In contrast to SR-SGC where all tasks are coded using a base (n, s)-GC scheme, a large fraction of computations in the M-SGC scheme are uncoded, thus incurring no computational overheads. This results in the M-SGC scheme achieving a significantly lower normalized load compared to SR-SGC or GC schemes. Such a lower normalized load eventually translates to reduced cumulative runtime, as we see in Section 3.6.

Design parameters The parameter set for M-SGC scheme is given by $\{n, B, W, \lambda\}$, where $0 \leq \lambda < n$, $0 < B < W$. The scheme incurs a delay of $T = W - 2 + B$. A minor modification of the scheme to cover the $\lambda = n$ scenario is discussed in Remark 3.3. As the M-SGC scheme is more involved, it is initially introduced through an example highlighting the key concepts. Subsequently, we will detail the general coding scheme. The scheme tolerates straggler patterns that conform to either (B, W, λ)-bursty straggler model or $(N = B, W' = W + B - 1, \lambda' = \lambda)$-arbitrary straggler model (See Proposition 3.2).

Example

Data placement Consider parameters $\{n = 4, B = 2, W = 3, \lambda = 2\}$. Assume that data set $\mathbf{D}$ contains d data points. $\mathbf{D}$ is partitioned into 16 data chunks of unequal sizes $\{\mathbf{D}_0, \ldots, \mathbf{D}_{15}\}$. Data chunks $\mathcal{D}_1 \triangleq \{\mathbf{D}_0, \ldots, \mathbf{D}_7\}$ are of equal size and contain $\frac{3}{32}d$ data points each. Similarly, data chunks $\mathcal{D}_2 \triangleq \{\mathbf{D}_8, \ldots, \mathbf{D}_{15}\}$ contain $\frac{1}{32}d$ data points each. These 16 data chunks are distributed across workers in the following manner; worker-0: $\{\mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_8, \mathbf{D}_9, \mathbf{D}_{10}, \mathbf{D}_{12}, \mathbf{D}_{13}, \mathbf{D}_{14}\}$, worker-1: $\{\mathbf{D}_2, \mathbf{D}_3, \mathbf{D}_9, \mathbf{D}_{10}, \mathbf{D}_{11}, \mathbf{D}_{13}, \mathbf{D}_{14}, \mathbf{D}_{15}\}$, worker-2: $\{\mathbf{D}_4, \mathbf{D}_5, \mathbf{D}_{10}, \mathbf{D}_{11}, \mathbf{D}_8, \mathbf{D}_{14}, \mathbf{D}_{15}, \mathbf{D}_{12}\}$ and worker-3: $\{\mathbf{D}_6, \mathbf{D}_7, \mathbf{D}_{11}, \mathbf{D}_8, \mathbf{D}_9, \mathbf{D}_{15}, \mathbf{D}_{12}, \mathbf{D}_{13}\}$. Note that data chunks in $\mathcal{D}_1$ are not replicated, whereas each data chunk in $\mathcal{D}_2$ is replicated 3 times. Each job-t is finished when master node computes $g(t) = g_0(t) + \cdots + g_{15}(t)$. A high-level idea of the coding scheme is as follows. Failed partial gradient computations for data chunks $\mathcal{D}_1$ will be reattempted across rounds. Partial gradient computations with respect to data chunks $\mathcal{D}_2$ will be made straggler-resilient by employing an $(4, 2)$-GC scheme.

Diagonally interleaved mini-tasks The task performed by each worker-i in round-t consists of $W - 1 + B = 4$ sequentially performed *mini-tasks* $\mathcal{T}_i(t; 0), \ldots, \mathcal{T}_i(t; 3)$. If worker-$i$ is a straggler in round-t, results of mini-tasks $\mathcal{T}_i(t; 0), \ldots, \mathcal{T}_i(t; 3)$ will not reach the master node and as far as the master node is concerned, all of them "failed". Conversely, if worker-i is a non-straggler in round-t, all the four mini-task results will reach the master node at the end of round-t. Mini-tasks $\mathcal{T}_i(t; 0), \mathcal{T}_i(t + 1; 1), \ldots, \mathcal{T}_i(t + 3; 3)$ involve partial gradient computations corresponding to job-t, as illustrated in Figure 3.2.

Fixed mini-task assignment Computations done as part of first two mini-tasks $\mathcal{T}_i(t; 0), \mathcal{T}_i(t; 1)$ are fixed for all t. Specifically, mini-tasks along the "diagonal" $\mathcal{T}_i(t; 0)$ and $\mathcal{T}_i(t + 1; 1)$ involve computing $g_{2i}(t)$ and $g_{2i+1}(t)$, respectively.

Adaptive mini-task assignment The other two mini-tasks $\mathcal{T}_i(t + 2; 2), \mathcal{T}_i(t + 3; 3)$, which also involve partial gradient computations corresponding to job-t, are assigned adaptively, based on the

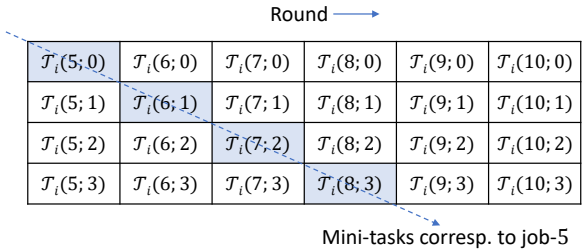


Figure 3.2: For an M-SGC scheme, all mini-tasks across a "diagonal" correspond to the same job.

straggler patterns seen in previous rounds. Specifically, if master node did not receive $g_{2i}(t)$ in round-t, $\mathcal{T}_i(t+2;2)$ involves computing $g_{2i}(t)$. Else if master node did not receive $g_{2i+1}(t)$ in round-$(t+1)$, $\mathcal{T}_i(t+2;2)$ involves computing $g_{2i+1}(t)$. In a similar manner, if master node did not receive $g_{2i+1}(t)$ in rounds $(t+1)$ and $(t+2)$, $\mathcal{T}_i(t+3;3)$ involves computing $g_{2i+1}(t)$. Note that so far we have described only partial gradient computations with respect to data chunks in $\mathcal{D}_1$. In round-$(t+2)$, if $g_{2i}(t)$ and $g_{2i+1}(t)$ are already available to master node, $\mathcal{T}_i(t+2,2)$ will involve computation of three partial gradients $\{g_l(t)\}_{l\in 8+[i:i+2]^*}$ and obtaining the linear combination $\ell_{i,0}(t) = \sum_{j\in[i:i+2]^*} \alpha_{i,j} g_{j+8}(t)$ (applying a $(4,2)$-GC). Using properties of GC-scheme, if master node has access to any two among $\{\ell_{0,0}(t),\ldots,\ell_{3,0}(t)\}$, $g_8(t)+g_9(t)+g_{10}(t)+g_{11}(t)$ can be obtained. Similarly, in round-$(t+3)$, if $g_{2i}(t)$ and $g_{2i+1}(t)$ are already available to master node, $\mathcal{T}_i(t+3,3)$ involves computing $\ell_{i,1}(t) = \sum_{j\in[i:i+2]^*} \alpha_{i,j} g_{j+12}(t)$. master node can recover $g_{12}(t)+g_{13}(t)+g_{14}(t)+g_{15}(t)$ from any two results among $\{\ell_{0,1}(t),\ldots,\ell_{3,1}(t)\}$. This completes the description of the M-SGC scheme. Note that the number of data points involved is the same in both fixed and adaptive mini-tasks and hence, the computational load remains the same in both situations.

Analysis of straggler patterns We will now show that the scheme tolerates any straggler pattern conforming to the $(B=2, W=3, \lambda=2)$-bursty straggler model. In Figure 3.3, we illustrate mini-task assignments with respect to a straggler pattern conforming to this model. As jobs are indexed in the range $[1:J]$, mini-tasks corresponding to job-t, $t \notin [1:J]$ are indicated using $\mathbf{0}$ in the figure. These are trivial mini-tasks that do not incur any computation. Consider the computation of $g(2)$ (i.e., job-2) based on Figure 3.3. Since, worker-0 is a straggler in round-2 and worker-1 is a straggler in rounds 2 and 3, computations of $\{g_0(2), g_2(2), g_3(2)\}$ failed initially, got reattempted and succeeded. From $\ell_{2,0}(2)$ and $\ell_{3,0}(2)$ (both finished in round-4), the master node can recover $g_8(2)+g_9(2)+g_{10}(2)+g_{11}(2)$, owing to the use of $(4,2)$-GC. Similarly, $g_{12}(2)+g_{13}(2)+g_{14}(2)+g_{15}(2)$ can be recovered using $\ell_{0,1}(2)$ and $\ell_{3,1}(2)$ in round-5. Hence, master node computes $g(2) \triangleq g_0(2) + \cdots + g_{15}(2)$ after round-5 (delay $T = W-2+B = 3$).

General scheme

Data placement Assume dataset D contains d data points and is split into two subsets; $\mathcal{D}_1$, with $(W-1)n$ chunks of equal size:

$$\mathcal{D}_1 \triangleq \{\mathbf{D}_0, \mathbf{D}_1, \ldots, \mathbf{D}_{(W-1)n-1}\} \tag{3.2}$$

$$|\mathbf{D}_i| = \frac{\lambda+1}{n(B+(W-1)(\lambda+1))}\, d, \quad \forall \mathbf{D}_i \in \mathcal{D}_1, \tag{3.3}$$

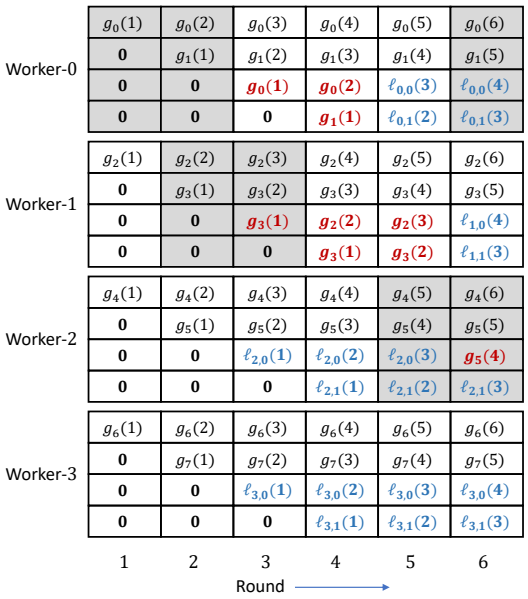


Figure 3.3: Rectangles depict mini-tasks (shaded ones have failed due to stragglers). Reattempted mini-tasks are indicated in red. Mini-task results in blue are linear combinations of 3 partial gradients.

and $\mathcal{D}_2$, with Bn chunks of equal size:

$$\mathcal{D}_2 \triangleq \{\mathbf{D}_{(W-1)n}, \mathbf{D}_{(W-1)n+1}, \ldots, \mathbf{D}_{(W-1+B)n-1}\} \tag{3.4}$$

$$|\mathbf{D}_i| = \frac{1}{n(B+(W-1)(\lambda+1))}\, d, \quad \forall \mathbf{D}_i \in \mathcal{D}_2. \tag{3.5}$$

Data chunks in $\mathcal{D}_1$ are divided into n groups consisting of $(W-1)$ data chunks each. Every worker will store one of these groups of data chunks. In other words, worker-i stores data chunks $\{\mathbf{D}_l\}_{l\in[i(W-1):(i+1)(W-1)-1]}$.
Similarly, $\mathcal{D}_2$ is divided into B groups consisting of n data chunks each. Group-j, $j \in [0:B-1]$, consists of data chunks $\{\mathbf{D}_{(W-1+j)n}, \ldots, \mathbf{D}_{(W+j)n-1}\}$. The n equally sized data chunks in each group will be treated as n partitions of data set in an (n,λ)-GC scheme (see Section 3.5.1) and will be stored in workers accordingly. That is, from each group-j, worker-i stores the $(\lambda+1)$ data chunks $\{\mathbf{D}_l\}_{l\in(W-1+j)n+[i:i+\lambda]^*}$.

Mini-task assignment In round-t, each worker-i sequentially performs $(W-1+B)$ mini-tasks labelled as $\{\mathcal{T}_i(t;0), \ldots, \mathcal{T}_i(t;W-2+B)\}$. A mini-task involves either (i) computing partial gradient with respect to one of the data chunks in $\mathcal{D}_1$ or else (ii) one partial gradient each with respect to $\lambda+1$ data chunks (thus, $\lambda+1$ partial gradients in total) in $\mathcal{D}_2$. As noted in the example, failed mini-tasks belonging to scenario (i) will be reattempted whereas those in scenario (ii) will be compensated via use of (n,λ)-GC scheme. Recall that delay $T = W-2+B$. Algorithm 3.2 describes how mini-tasks are assigned. In addition, the normalized load for this scheme is:

$$L_{\text{M-SGC}} = \begin{cases} \frac{(\lambda+1)(W-1+B)}{n(B+(W-1)(\lambda+1))}, & \text{if } \lambda < n, \\ \frac{W-1+B}{n(W-1)}, & \text{if } \lambda = n. \end{cases} \tag{3.6}$$

Remark 3.3 (Case of $\lambda = n$)**.** For the special case $\lambda = n$, data set $\mathbf{D}$ will be partitioned into $(W-1)n$ equally sized data chunks $\mathcal{D}_1 \triangleq \{\mathbf{D}_0, \mathbf{D}_1, \ldots, \mathbf{D}_{(W-1)n-1}\}$. i.e., effectively we have $\mathcal{D}_2 \triangleq \Phi$ (null set). For notational consistency, we set partial gradients $g_l(t) \triangleq \mathbf{0}$, $l \in [(W-1)n : (W-1+B)n-1]$. These are trivial partial gradients which do not incur any computation in Algorithm 3.2.

Remark 3.4. It is straightforward to note that for given $\{n, W, B\}$, normalized load is the largest when $\lambda = n$. As $B < W$, we thus have $L_{\text{M-SGC}} \leq \frac{2}{n}$ irrespective of the choice of λ. In contrast, normalized load $\frac{s+1}{n}$ of SR-SGC scheme scales with λ ($s \triangleq \lceil \frac{B\lambda}{W-1+B} \rceil \geq 1$).

Algorithm 3.2 Algorithm used by master node to assign mini-tasks in round-t

```
for i ∈ [0 : n − 1] do
    for j ∈ [0 : W − 2] do
        Assign computation of g_{i(W−1)+j}(t − j) as mini-task T_i(t; j).
    for j ∈ [W − 1 : W − 2 + B] do
        if master node received all of {g_{i(W−1)+j'}(t − j)}_{j'∈[0:W−2]} prior to round-t then
            Assign computation of (λ + 1) partial gradients {g_{jn+l}(t − j)}_{l∈[i:i+λ]*} as T_i(t; j).
        else
            for j' ∈ [0 : W − 2] do
                if master node has not received g_{i(W−1)+j'}(t − j) prior to round-t then
                    Assign computation of g_{i(W−1)+j'}(t − j) as mini-task T_i(t; j).
                    break
```

Proposition 3.2. *The M-SGC scheme designed for parameters $\{n, B, W, \lambda\}$ tolerates any straggler pattern that conforms to either (i) the (B, W, λ)-bursty straggler model, or (ii) the $(N = B, W' = W + B - 1, \lambda' = \lambda)$-arbitrary straggler model.*

We refer readers to Appendix E in [2] for complete proof.

3.6 Experiments

In this section, we evaluate the performance of proposed schemes by training $M = 4$ neural network models concurrently. The pipelined training follows the discussion provided in Section 3.3.1. For workers, we use the AWS Lambda, [1] a fully managed and cost-efficient serverless cloud computing service. Workers are invoked from the master node using HTTP requests, and task results are received in the HTTP response payload.

Section 3.6.1 provides insights into the response time of workers, while Section 3.6.2 compares the performance of our coding schemes with GC and non-coded approach in training a small CNN on MNIST. Following this, Section 3.6.3 outlines the network architecture. Section 3.6.4 introduces a methodology for tuning the parameters of each coding scheme, accompanied by best practices for parameter selection and sensitivity analysis. Overhead times for decoding and parameter selection are analyzed in detail in Section 3.6.5. Finally, training results for a Resnet-18 model on Cifar-100, using the same infrastructure are presented in Section 3.6.6.

[1] https://aws.amazon.com/lambda/

3.6.1 Analysis of Response Time

Our experiment setup consists of a master node and $n = 256$ workers. In Figure 3.4, we demonstrate response time statistics across 100 rounds, where each worker calculates gradients for a batch of 16 MNIST images on a CNN involving three convolutional layers, followed by two fully connected layers. Figure 3.4(a) shows the response time of each worker at every round. White cells represent stragglers. As discussed in Section 3.3, a worker is deemed straggler when its response time exceeds $(1 + \mu)$ times the response time of the fastest worker in the round. For the sake of consistency, we choose $\mu = 1$ for all experiments. Nonetheless, such a choice of μ is by no means critical to observe stragglers. This can be seen in Figure 3.4(c), where the empirical CDF of workers' completion time exhibits a relatively long tail. Figure 3.4(b) plots the number of straggler bursts of different lengths over this response profile. It can be observed that our response profile does not include nodes that continue to remain stragglers for a long duration. This motivates the use of coding across the temporal dimension as proposed in the present work.

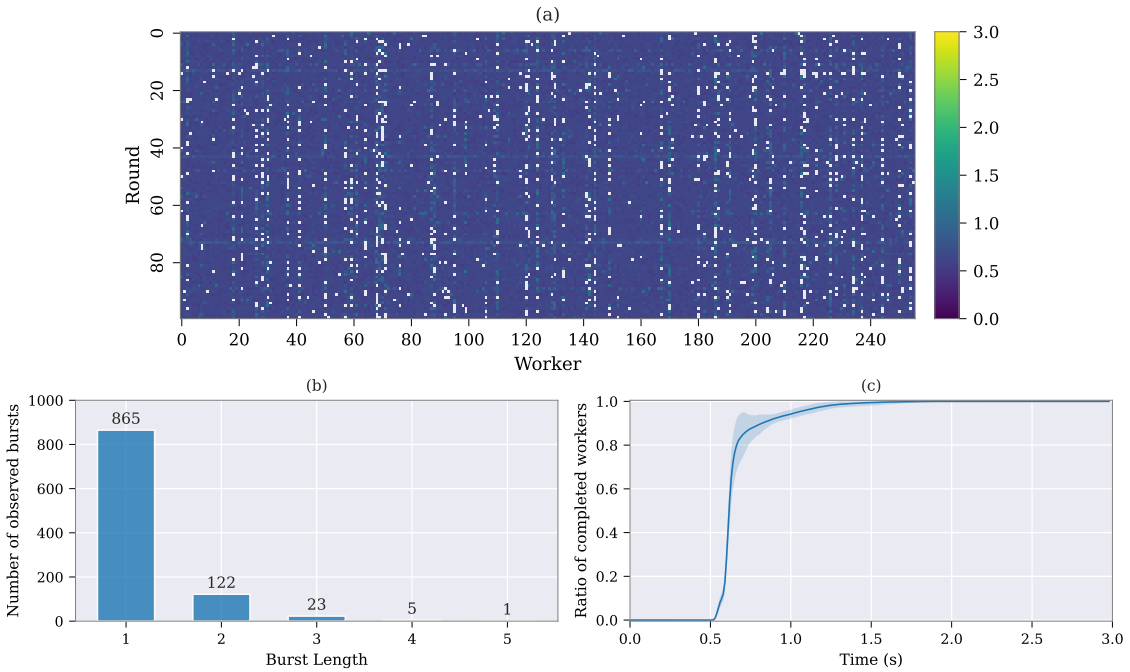


Figure 3.4: Statistics of response time for 256 workers across 100 rounds. Each worker calculates the gradients of the loss for a batch of 16 MNIST images on a convolutional neural network. (a) Each white cell represents a worker (x-axis) who is a straggler at the corresponding round (y-axis). (b) Histogram of stragglers' burst lengths. (c) Empirical CDF of workers' completion time, averaged over 100 rounds (shades represent standard deviation).

3.6.2 Comparison of Coding Schemes

Using the setup described in Section 3.6.1, we train $M = 4$ CNN classifiers for MNIST concurrently, following the approach stated in Section 3.3.1. In every round, master node samples a batch of 4096 data points and distributes them among the $n = 256$ workers. Non-straggling workers compute partial gradients and return task results to the master node at the end of each round. After

completion of one update, master node uploads the updated model parameters to a shared network file system, accessible to the workers. We use cross entropy as the loss function and ADAM [66] as the optimizer. Moreover, the same dataset and architecture are used for all the models.

In each experiment, we run a total of $J = 480$ jobs (120 jobs per classifier) using the three schemes, namely GC, SR-SGC, and M-SGC. As a baseline, we also train the classifiers without any coding wherein the master node should wait for all the workers to return their task results. Finally, each experiment is repeated 10 times to report the first and second-order statistics of total run times. Before training the models, we perform some shorter experiments to choose the best-performing parameters for each of the three coding schemes. Specifically, for GC, we perform a grid search over s and select the value corresponding to the shortest run time. We refer readers to Section 3.6.4 for a detailed discussion on the procedure of selecting the parameters for SR-SGC and M-SGC schemes, as well as an analysis of sensitivity to parameters.

Table 3.1 presents the total run time achieved by each coding scheme, along with the selected parameters and resulting normalized loads. The selection of small values for parameters B and W in our sequential coding schemes matches the empirical evidence in Figure 3.4(b) that isolated short-length-bursts are prevalent. It is interesting to note that the effective value of parameter s in SR-SGC ($s = 12$) turns out to be close to that of GC ($s = 15$). Figure 3.5(a) plots the total number of completed jobs (for all $M = 4$ models) across time, and Figure 3.5(b) shows the course of training loss (of the first model out of the 4 models) as a function of time, for all coding schemes.

Table 3.1: Total run time achieved by different coding schemes

Scheme	**Parameters**	**Normalized Load**	**Run Time (s)**
M-SGC	$B = 1, W = 2, \lambda = 27$	0.008	891.37 ± 43.10
SR-SGC	$B = 2, W = 3, \lambda = 23 \quad (s = 12)$	0.051	994.22 ± 43.66
GC	$s = 15$	0.062	1064.96 ± 46.72
No Coding	–	0.004	1307.79 ± 61.88

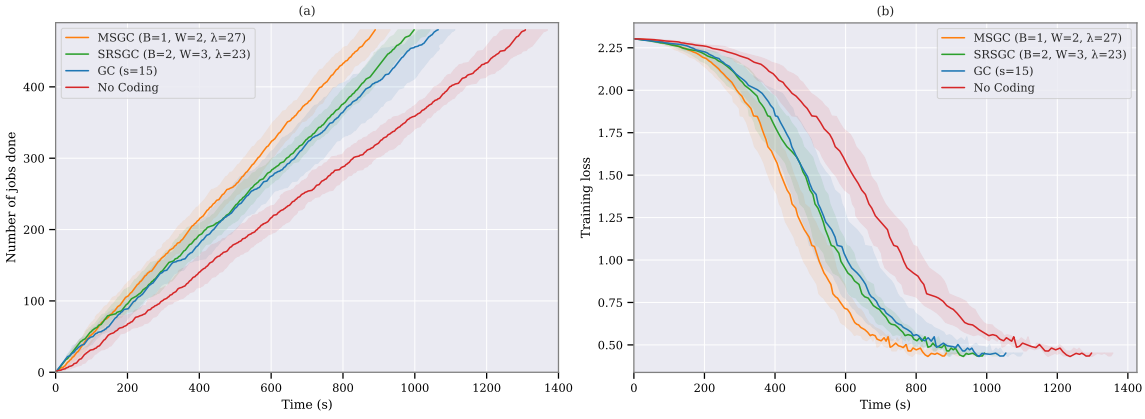


Figure 3.5: (a) Number of completed rounds vs. clock time, averaged over 10 independent experiments. (b) Training loss vs. clock time for the first model (out of four concurrently trained models), averaged over 10 independent runs. Shades here represent standard deviation.

The first clear observation from Table 3.1 is that our proposed M-SGC achieves 16% lower run time

while maintaining a smaller normalized load compared to the classical GC scheme. Furthermore, compared to GC, SR-SGC shows slight improvements in total runtime and normalized load simultaneously, demonstrating the potential of incorporating selective repetition into GC. Next, as shown in Figure 3.5 and Table 3.1, the existence of stragglers is validated by the fact that any of the coding schemes significantly outperforms the case of not using any coding. This is indeed in line with the empirical observation of Figure 3.4(c), where the tail of the cumulative distribution of workers' completion time signals the existence of stragglers. Section 3.6.5 discusses how the overheads of decoding time and parameter selection time can be completely removed in the training process. In Section 3.6.6, we present analogous results for concurrently training four ResNet-18 models on CIFAR-100 dataset.

3.6.3 AWS Lambda Architecture

This section discusses the overall architecture, limitations, and additional details about the setup and usage of the AWS cloud resources used in our experiments.

We use AWS Lambda functions as workers in our distributed training experiments. Each Lambda instance has 2500 MB of RAM, 1 vCPU, and supports the Python 3.8 runtime environment. A Lambda layer is used to inject external dependencies into the runtime environment (e.g., PyTorch, TorchVision, NumPy, etc). Since the size of required external libraries exceeds the 200MB limit of Lambda layers, we zip some libraries in the layer package and unzip them at the time of Lambda instance creation. Note that this will not affect workers' run times as we perform a *warm-up* round before each experiment to ensure that our Lambda instances are initialized and functional.

Another limitation concerning the use of Lambda functions for training ML models is the total payload size quota of 6 MB. i.e., the total sum of payload sizes in the HTTP request and response cannot exceed 6 MB. Note that ideally the master node includes current model weights in the HTTP request payload, and receives the task results via the HTTP response payload. This incurs a serious limitation on any reasonably sized neural network. To overcome this, we need to use a proxy storage service to communicate model weights and task results.

Fortunately, we have two storage options; Amazon S3 (Simple Storage Service) and AWS EFS (Elastic File System). We use the latter, as it will provide higher throughput. EFS is a shared network file system that will be mounted on all Lambda instances at their time of creation. This way, it can be used as a means for communication between the master node and workers. In our experiments, we reserve the payload limit for the task result communication, and use EFS to communicate updated model weights to workers, as depicted in Figure 3.6 (a). The overall architecture of our cloud resources is shown in Figure 3.6 (b). We use the AWS Serverless Application Model (SAM) tool to define, manage, and deploy the cloud resources (included in the code submitted as supplementary material).

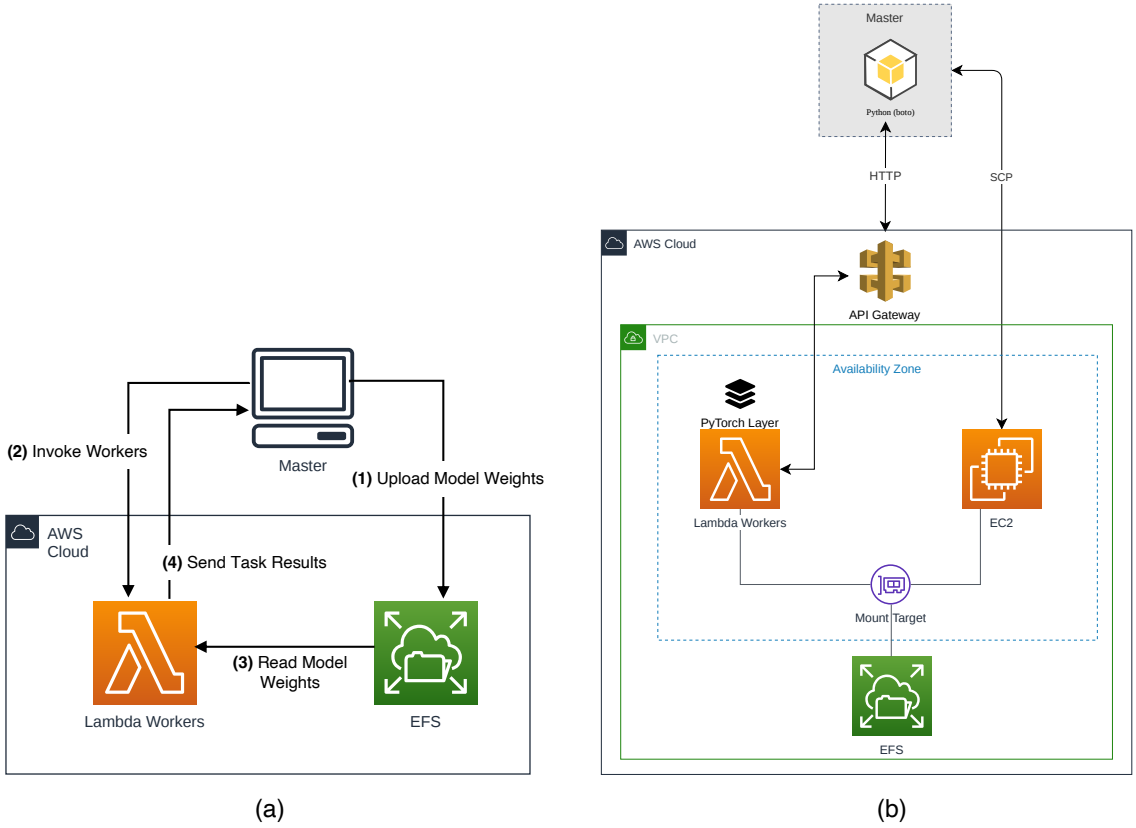


Figure 3.6: (a) Communication between master node and Lambda workers at each round. (b) The overall architecture of AWS cloud resources used.

3.6.4 Selecting coding scheme parameters

This section discusses the parameter selection method used for our proposed sequential gradient coding schemes, SR-SGC and M-SGC. We begin by noting that the total number of valid parameter combinations for each of these schemes is too large for a grid search to be feasible, as evaluation of each parameter combination requires training models for multiple rounds. Instead, we leverage the observation that increasing the normalized load linearly increases the average runtime of the workers. Figure 3.7 shows the average job completion time of 256 workers across 100 rounds for multiple values of load in $[0, 1]$.

We can exploit the observation above to estimate the delay profile corresponding to various coding schemes with variable normalized loads. After estimating the slope of linear fit in Figure 3.7 (let this be α), we run our distributed training experiment for $T_{\text{probe}} = 80$ jobs with no coding, and store the observed runtime of workers across rounds (we call this the *reference delay profile*). Note that the normalized worker load in case of no coding will be $1/n$.

Next, consider a coding scheme with fixed parameters B, W, λ, and a fixed normalized load of L. We can estimate the runtime of our coding scheme by feeding the reference delay profile to the master node to simulate the run time of workers. Based on our previous discussion, we should take into account the increase in the workers' run time due to the increase of the load from $1/n$ to L, by adding $(L - 1/n)\alpha$ seconds to our reference delay profile. Using this *load-adjusted* delay profile and

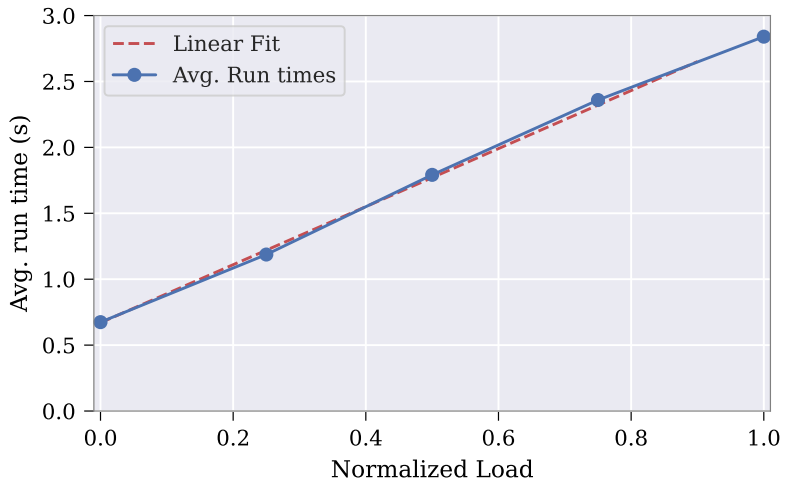


Figure 3.7: Average run time (256 workers, 100 rounds) scales linearly with computational load.

the considered coding scheme, the master node will try to resolve stragglers at each round, and if not, it will wait out all the workers, resulting in a simulated total run time for the coding scheme. Figure 3.8 shows the estimated runtime of 80 rounds for different choices of parameters B, W, λ in SR-SGC and M-SGC. For each of the two schemes, we select the parameters corresponding to the smallest estimated training runtime, denoted by the blue circles on Figure 3.8 and listed in Table 3.1.

Sensitivity to Parameters

Figure 3.8 also demonstrates the sensitivity of the coding schemes to their parameters, where we can observe smooth changes in training time with respect to coding parameters.
For SR-SGC, the normalized load varies significantly with the choice of parameters: $L = \frac{s+1}{n}$ with $s = \lceil \frac{B\lambda}{W-1+B} \rceil$. Specifically, the load is directly proportional to λ. As observed in Figure 3.8 (left), for each choice of B and W, increasing λ above a certain threshold leads to a significant increase in the runtime. Therefore, λ should be chosen carefully as it will affect the load and runtime heavily. On the other hand, the computational load in M-SGC is less dependent on selected parameters, as the load is upper-bounded by $2/n$ (see Remark 3.4). Therefore, the choice of λ does not play a crucial role as long as it is above the typical number of stragglers. Also, as observed in Figure 3.8 (right), the runtime is fairly insensitive to the choice of B as long as W and B are close.

Remark 3.5. Recommendations for selecting parameters of M-SGC and SR-SGC:

- Keeping W close to B seems to be the right rule of thumb for both schemes. Also, the dependence of both schemes on B is less critical, and increasing W is generally not preferred as it reduces the straggler correction capability of the coding schemes.
- For M-SGC, the choice of λ is not critical as long as it is above a certain threshold, but for SR-SGC it is an important consideration as it affects the load significantly.
- Therefore, it is recommended to start with a fixed B, choose W as close as possible to B, and find a large enough λ for M-SGC or a small enough λ for SR-SGC, based on the straggler pattern.

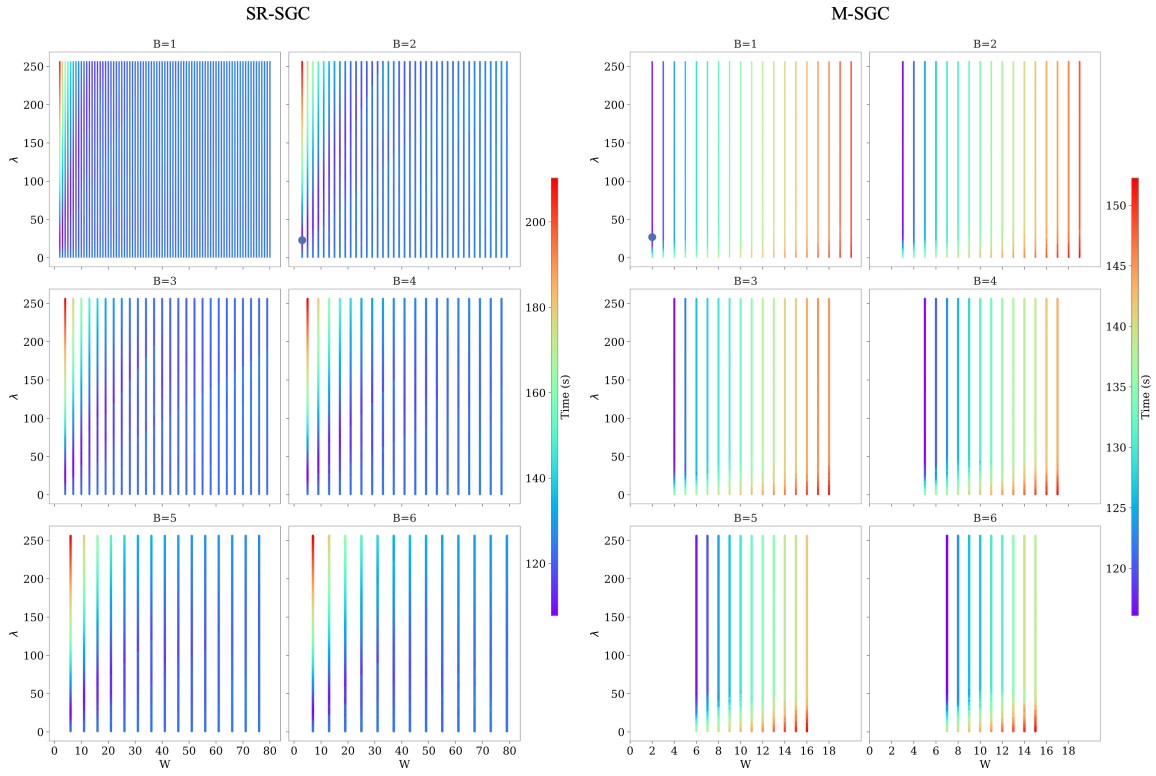


Figure 3.8: Estimated runtime of 80 rounds for different choices of parameters B, W, λ. Left: SR-SGC, Right: M-SGC. Blue dot marks the shortest runtime (selected parameters).

Table 3.2: Selected parameters using different values of T_{probe}.

Scheme	T_{probe}	**Selected Parameter**	**Load**	**Runtime (avg. ± std.)**
M-SGC (B, W, λ)	10	(1, 2, 24)	0.007512	872.78 ± 80.67 (s)
	20	(1, 2, 24)	0.007512	872.78 ± 80.67 (s)
	40	(1, 2, 27)	0.007543	871.99 ± 59.76 (s)
	60	(1, 2, 27)	0.007543	871.99 ± 59.76 (s)
	80	(1, 2, 27)	0.007543	871.99 ± 59.76 (s)
SR-SGC (B, W, λ)	10	(2, 3, 15)	0.035156	1226.90 ± 93.53 (s)
	20	(2, 3, 15)	0.035156	1226.90 ± 93.53 (s)
	40	(2, 3, 20)	0.042969	1060.61 ± 62.72 (s)
	60	(2, 3, 22)	0.046875	964.06 ± 53.48 (s)
	80	(2, 3, 23)	0.050781	984.37 ± 51.02 (s)
GC (s)	10	(9)	0.039062	1288.57 ± 94.15 (s)
	20	(10)	0.042969	1261.61 ± 80.87 (s)
	40	(11)	0.046875	1168.67 ± 77.60 (s)
	60	(14)	0.058594	1142.01 ± 38.41 (s)
	80	(15)	0.062500	1067.56 ± 40.92 (s)

Next, we evaluate the sensitivity of parameter selection to the number of rounds used in the selection process T_{probe}. Table 3.2 lists the selected parameters using different values of T_{probe}. For each unique set of parameters, we performed the experiments discussed in Section 3.6.2 10 times and included the average runtime as well. As expected, we observe a general trend of improvement in total training time by increasing T_{probe}. Moreover, for M-SGC even a few number of rounds are enough to tune the coding scheme, as M-SGC with parameters selected over only 10 rounds outperforms other coding schemes across all other values of T_{probe}.

3.6.5 Analysis of Overheads

In this section, we discuss two overheads in proposed sequential gradient coding schemes, as well as ways to effectively remove those overheads in practice.

Decoding Time

At the end of each round, the master node decodes the task results obtained from a subset of workers to calculate gradients. Both sequential coding methods proposed in this work use GC as their core coding/decoding method: SR-SGC with parameters $\{B, W, \lambda\}$ uses an $(n, \lceil \frac{B\lambda}{W-1+B} \rceil)$-GC, and M-SGC uses B independent (n, λ)-GC schemes. In (n, s)-GC, gradients are simply obtained by a linear combination of task results from $n - s$ workers [32]. Therefore, decoding consists of two steps:

1. Finding the decoding coefficients based on the observed straggler pattern. This can be done by solving a linear matrix equation.
2. Calculating the linear combination of task results from non-straggling workers, using the coefficients from step 1.

Table 3.3 summarizes the decoding time for the main experiment presented in Section 3.6.2.

Table 3.3: Decoding time of different schemes

Scheme	**Parameters**	**Decoding Time (avg. ± std.)**	**Longest Decoding**	**Fastest Round**
M-SGC	$B = 1, W = 2, \lambda = 27$	290 ± 18.7 ms	479 ms	1177 ms
SR-SGC	$B = 2, W = 3, \lambda = 23$	309 ± 20.1 ms	401 ms	1349 ms
GC	$s = 15$	204 ± 20.6 ms	408 ms	1379 ms

It is important to note that if training of more than $T + 1$ models is pipelined, the decoding can be done in the master node's idle time, and the decoding time will not add any overhead to the training time. As stated in Section 3.3.1, when the decoding delay of the sequential coding scheme is T rounds, training at least $M = T + 1$ models needs to be pipelined. Specifically, with M models and decoding delay of T rounds, the gradient of each model is ready to be decoded $D = M - T - 1$ rounds earlier than the next round of the model begins. In the case of $M = T + 1$, the gradients of a model are ready for decoding just before the next round for the model begins ($D = 0$). However, when $M > T + 1$, i.e., $D > 0$, the master node has at least one gap round to decode the gradients, and therefore can run the decoding at its idle time while workers are performing their assigned tasks.

We emphasize that this is indeed applicable to the experiments presented in Section 3.6.2, where $M = 4$ and the delay of all models are smaller than $M - 1 = 3$.
As an example, let us consider the SR-SGC scheme with parameters $\{B = 2, W = 3, \lambda = 23\}$ discussed in Section 3.6.2, where $M = 4$ and $T = 2$. Calculations of gradients for model 1 start at rounds $t_1 = 1, t_2 = 5, t_3 = 9, \cdots$. Calculating the first gradients of model 1 starts at round $t_1 = 1$. By the end of round $t_1 + T = 3$, the gradients are ready to be decoded. Note that the second gradients of model 1 are to be calculated at round $t_2 = 5$. Therefore, the master node can perform the decoding during its idle time over round 4, when it is waiting for the worker nodes (note that as can be seen from Table 3.3, the longest decoding time is shorter than the fastest round time). This way, the gradients are decoded before round $t_2 = 5$ begins and thus no decoding overhead is imposed.

Parameter Selection Time

In Section 3.6.4, we discussed the process of selecting parameters $\{B, W, \lambda\}$ for the coding schemes. This process requires running uncoded training for some number of rounds (T_{probe}) and storing the job completion time of workers. This delay profile is then used to search for the best coding parameters. In the experiment Section 3.6.2, we started coded training from round-1 (delay profile measurement and selection of best parameters are done beforehand). However, in practice, one can opt to start with uncoded training in round-1 and then switch to coded training after T_{probe} uncoded rounds (a few additional rounds will be needed to perform the exhaustive search for the best parameters as well). This way, the time to be spent initially for delay profile measurements can be utilized towards the completion of jobs.

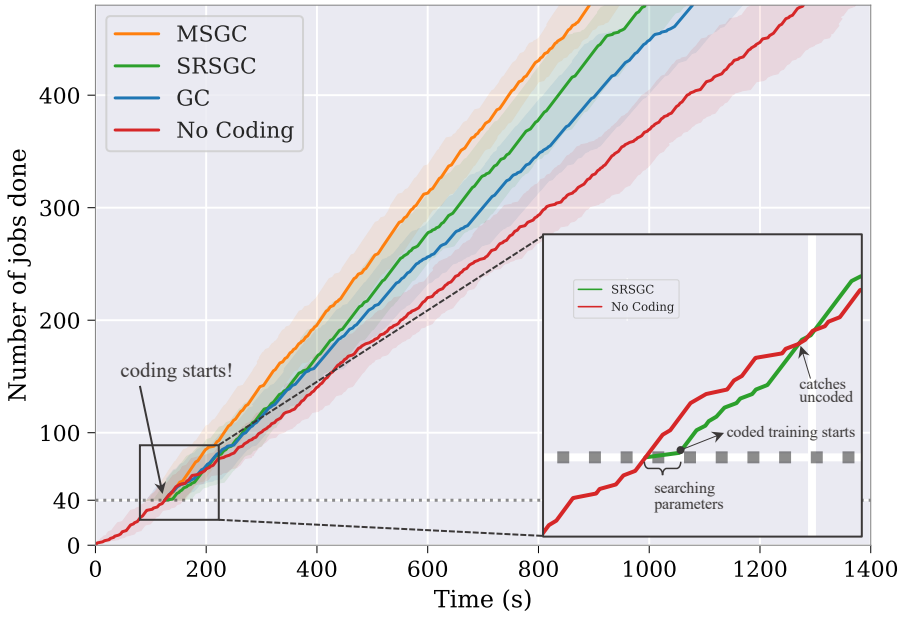


Figure 3.9: The same setup as in Section 3.6.2, but training starts uncoded and switches to coded mode after $T_{\text{probe}} = 40$ rounds. The delay profile measured from the initial 40 rounds is used to select the coding parameters. The plot shows average $\pm$ std. over 10 independent trials. Transition to SR-SGC is zoomed in.

Figure 3.9 demonstrates this method in which training starts uncoded, and after $T_{\text{probe}} = 40$ rounds, the master node uses the observed delay profile from the past rounds to perform an exhaustive search for the best parameters of the coding schemes. In this experiment, it took ~ 8 seconds for SR-SGC, ~ 2 seconds for M-SGC, and less than a second for GC to search over all valid parameters. The training is then switched to coded mode after the search is over. Here, we still observe significant gains from M-SGC compared to uncoded and GC methods.

3.6.6 Training ResNet-18 on CIFAR-100

Here, we present the results of concurrently training $M = 4$ ResNet-18 models [67], on CIFAR-100 [68], over AWS Lambda, using the different coding schemes. ResNet-18 has 11,227,812 parameters and hence, the size of model weights and gradient updates are roughly 22.5 MB in 16-bit floating point format. This is much larger than the 6MB payload size limit of AWS Lambda (see Section 3.6.3). This means the master node and workers have to essentially use a shared storage (Amazon EFS) to communicate model weights and coded gradients at each round, as depicted in Figure 3.10 (a). At each round, the master node first uploads updated model weights to the dedicated shared storage and invokes Lambda workers via an HTTP request. Lambda workers then read the model weights from the storage and proceed to calculate the task results. Finally, each worker uploads the coded gradients to the shared storage and signals the master node via the HTTP response.

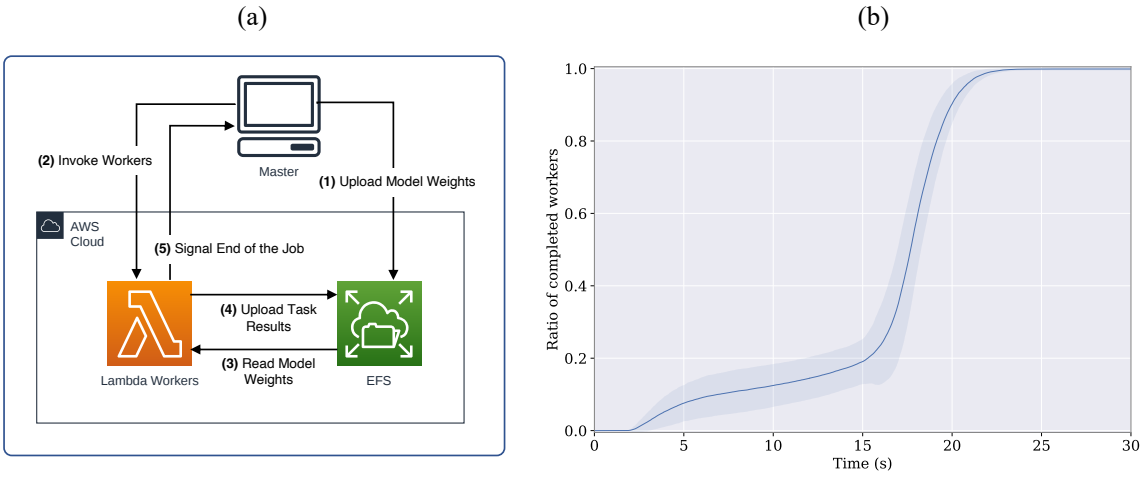


Figure 3.10: (a) Communication between master node and Lambda workers at each round. (b) Empirical CDF of workers' completion time, averaged over 100 rounds and 256 workers (shades represent standard deviation. Each worker calculates gradients for a batch of 2 CIFAR-100 images on ResNet-18, and uploads the gradients to EFS.

We used $n = 256$ Lambda workers to train $M = 4$ models concurrently for 1000 rounds (250 rounds for each classifier). A batch size of 512 samples and ADAM optimizer is used. Figure 3.11 plots the number of completed jobs (for all $M = 4$ models) over time for the three coding schemes, as well as uncoded training. Coding parameters are selected based on the method discussed in Section 3.6.4 using $T_{\text{probe}} = 20$ rounds. The results showed that M-SGC finishes training 11.6% faster than GC (while maintaining a significantly smaller normalized load), and 21.5% faster than uncoded training.

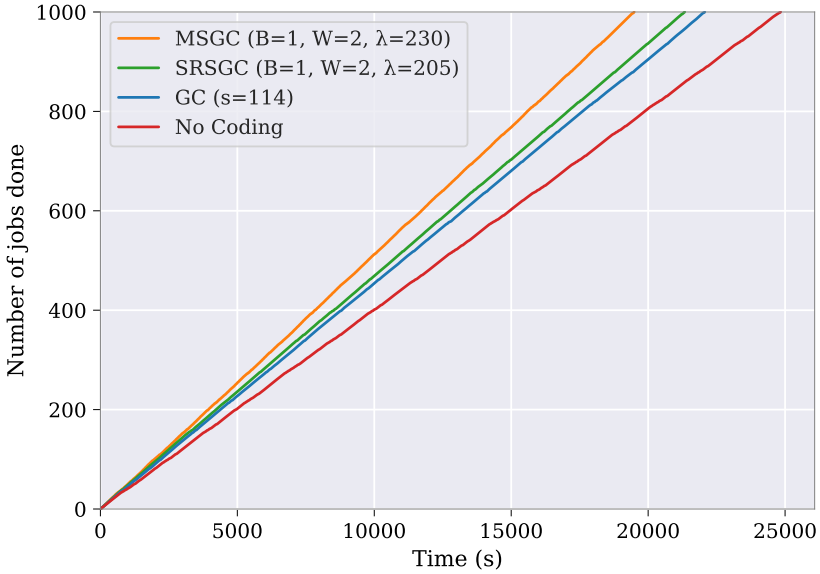


Figure 3.11: Training 4 ResNet-18 models on CIFAR-100. Number of completed rounds (for all $M = 4$ models) vs. time.

3.7 Summary and Concluding Remarks

This chapter addressed the critical challenge of stragglers in the distributed training of machine learning models. We develop a new class of gradient coding schemes that exploit coding across the temporal dimension, in addition to the spatial dimension (i.e., coding across workers) considered in prior works.

To facilitate the analysis and design of coding schemes, we established three straggler models: the bursty straggler model, the arbitrary straggler model, and the s-stragglers-per-round model, emphasizing the ability of the proposed coding schemes to handle specific straggler patterns.

The core of the chapter presented two novel coding schemes: Selective Reattempt Sequential Gradient Coding (SR-SGC) and Multiplexed Sequential Gradient Coding (M-SGC). SR-SGC extends the classical GC scheme by selectively reattempting unfinished tasks in future rounds, enabling it to tolerate a broader range of straggler patterns for the same computational load. M-SGC, on the other hand, divides tasks into two sets: those protected against stragglers via reattempts and those protected via GC. It then carefully multiplexes these tasks to achieve a significantly lower computational load per worker compared to GC and SR-SGC.

To evaluate the performance of the proposed schemes, we presented experimental results on a large-scale AWS Lambda cluster involving 256 worker nodes. The experiments involved concurrently training multiple neural network models and demonstrated significant gains in runtime and reduced computational load with M-SGC compared to other schemes.

The chapter also discussed the architecture and limitations of the AWS Lambda service, presented a method for selecting optimal coding parameters, and analyzed and addressed overheads associated with decoding time and parameter selection time.

Chapter 4

Minimum Entropy Coupling with Bottleneck

4.1 Introduction

Consider the following Markov Chain modeling a general lossy compression framework:

$$X \xrightarrow{p_{T|X}} T \xrightarrow{q_{Y|T}} Y \tag{4.1}$$

Here, the input X, with a marginal distribution p_X, is encoded by the probabilistic encoder pto generate the code T. Subsequently, the probabilistic decoder qconstructs Y from T. The objective is to identify the encoder and decoder that minimize the distortion between X and Y, subject to a constraint on the expected code length $H(T)$.

It is common to measure the sample-wise distortion via direct comparison of (x, y) pairs through a distortion function $d(\cdot, \cdot)$, and consider the expectation $\mathbb{E}\left[d(X, Y)\right]$ as a measure of average distortion. Instead, we propose using the logarithmic loss (log-loss) $H(X|Y)$, or equivalently $I(X;Y)$, as an alternative metric to enforce the distortion constraint.

The log-loss distortion measure, commonly employed in learning theory, was first explored within rate-distortion theory by [69] and [70]. This measure is particularly suitable in scenarios where reconstructions can be soft, meaning that the decoder produces a distribution rather than a distorted sample point [71]. Consequently, the optimization problem is formulated as follows:

$$\begin{aligned} \min_{p_{T|X},\, q_{Y|T}} \quad & H(X|Y) \\ \text{s.t.} \quad & X \leftrightarrow T \leftrightarrow Y, \\ & H(T) \leq R, \\ & P(X) = p_X \end{aligned} \tag{4.2}$$

It is straightforward to check that the optimal solution of (4.2) is achieved when $T = Y$, with the identity decoder. To address this issue of decoder collapse, we introduce a constraint on the output marginal distribution, $P(Y)$:

Minimum Entropy Coupling with Bottleneck (MEC-B)

$$\begin{aligned} \mathcal{I}_{\text{MEC-B}}(p_X, p_Y, R) = \max_{p_{T|X},\, q_{Y|T}} \quad & I(X;Y) \\ \text{s.t.} \quad & X \leftrightarrow T \leftrightarrow Y, \\ & H(T) \leq R, \\ & P(Y) = p_Y, \\ & P(X) = p_X \end{aligned} \tag{4.3}$$

We explore two special cases of (4.3), where either the encoder or decoder is bypassed. This allows us to optimize the encoder and decoder separately using these cases.
First, consider the case where the bottleneck is removed, meaning the constraint $H(T) \leq R$ is relaxed, or $R \geq H(X)$. In this scenario, $X = T$, and the optimization simplifies to:

Minimum Entropy Coupling (MEC)

$$\begin{aligned} \mathcal{I}_{\text{MEC}}(p_X, p_Y) = \max_{p_{Y|X}} \quad & I(X;Y) \\ \text{s.t.} \quad & P(Y) = p_Y, \\ & P(X) = p_X \end{aligned} \tag{4.4}$$

This involves identifying the probabilistic coupling $p_{Y|X}$ between the marginals p_X and p_Y that maximizes the obtained mutual information. This problem, as described in (4.4), has been extensively studied in the literature as minimum entropy coupling (MEC), with early research conducted by [72–75], among others. Thus, we define the original problem presented in (4.3) as minimum entropy coupling with bottleneck (MEC-B).
Next, consider the case where the decoder is removed, resulting from the relaxation of the output distribution constraint in (4.3):

Entropy-Bounded Information Maximization (EBIM)

$$\begin{aligned} \mathcal{I}_{\text{EBIM}}(p_X, R) = \max_{p_{T|X}} \quad & I(X;T) \\ \text{s.t.} \quad & H(T) \leq R, \\ & P(X) = p_X \end{aligned} \tag{4.5}$$

Similar to minimum entropy coupling, this problem identifies the joint distribution between two random variables that maximizes their mutual information. However, rather than imposing a marginal distribution constraint, it enforces a more flexible entropy constraint on one of the variables.
Lemma 4.1 provides a decomposition for the mutual information between input and output $I(X;Y)$, given the Markov chain $X \leftrightarrow T \leftrightarrow Y$.

Lemma 4.1. *Given a Markov chain $X \leftrightarrow T \leftrightarrow Y$:*

$$I(X;Y) = I(X;T) + I(Y;T) - I(T;X,Y) \tag{4.6}$$

Proof. By multiple applications of the chain rule for mutual information, i.e., $I(A;B,C) = I(A;B) + I(A;C|B)$, we have:

$$\begin{aligned} I(X;Y) &= I(X;Y,T) - I(X;T|Y) && (4.7)\\ &= [I(X;T) + I(X;Y|T)] - [-I(T;Y) + I(T;X,Y)] && (4.8)\\ &= I(X;T) + I(Y;T) - I(T;X,Y) && (4.9) \end{aligned}$$

Note that from the Markov chain property, we have $I(X;Y|T) = 0$. □

The following lower bound on the MEC-B objective is attainable based on Lemma 4.1:

$$I(X;Y) \geq I(X;T) + I(Y;T) - R \tag{4.10}$$

In this work, we consider maximizing the lower bound (4.10) as a proxy to the main objective. This allows a decomposition of the encoder and decoder for the MEC-B formulation in (4.3):

1. **Encoder Optimization:** The encoder is first optimized separately, according to Entropy-Bounded Information Maximization in (4.5), $\mathcal{I}_{\text{EBIM}}(p_X, R)$, resulting in the marginal distribution $\hat{p}_T$ on the code T.
2. **Decoder Optimization:** The decoder is then optimized by solving a minimum entropy coupling in (4.4) between the code and output marginals, $\mathcal{I}_{\text{MEC}}(\hat{p}_T, p_Y)$.

Therefore, in terms of problems (4.3), (4.4), and (4.5):

$$\begin{aligned} \mathcal{I}_{\text{MEC-B}}(p_X, p_Y, R) &= \max_{p_{T|X},\ q_{Y|T}} I(X;Y) && (4.11)\\ &\geq \max_{p_{T|X},\ q_{Y|T}} [I(X;T) + I(Y;T) - R] && (4.12)\\ &\geq \mathcal{I}_{\text{EBIM}}(p_X, R) + \mathcal{I}_{\text{MEC}}(\hat{p}_T, p_Y) - R && (4.13) \end{aligned}$$

Section 4.3 provides an overview of the minimum entropy coupling problem. Following that, Section 4.4 focuses on the entropy-bounded information maximization problem as formulated in (4.5), in great detail. Section 4.5 discusses the application of minimum entropy coupling with bottleneck in the context of Markov Coding Games. Lastly, Section 4.6 presents experimental results.

4.2 Related Works

4.2.1 Couplings and Minimum Entropy Coupling

A fundamental problem in probability theory, known as coupling, concerns determining the *optimal* joint distribution of random variables given their marginal distributions. This problem has a long history, with early examples like [76] seeking the joint distribution that maximizes correlation subject to marginal constraints. References [77–80] provide a broader treatment of these problem classes and their applications. Notably, optimal transport (OT) emerges as a significant class within this framework, where optimality is defined as minimizing the expected value of a loss function over the joint distribution. See [81] for an in-depth treatment of the optimal transport problem.

The minimum entropy coupling (MEC) focuses on finding the joint distribution with the smallest entropy given the marginal distribution of some random variables. This problem has been first studied in [72–75], among others. Although couplings are directly relevant to optimal transport, MEC cannot directly be framed as an optimal transport problem because the cost function in OT does not depend on the joint distribution itself. Nonetheless, given that the set of all couplings is called the transport polytope, the minimum entropy coupling is identified as the element with the lowest entropy within this transport polytope.

Entropic Optimal Transport (EOT) introduces an entropy regularization to the classical optimal transportation problem, encouraging a transport plan with high entropy. This regularization transforms the problem into a strictly convex one, eliminating the need for linear programming [82]. It can be efficiently solved using the Sinkhorn-Knopp matrix scaling algorithm [83], which converges linearly and is highly parallelizable, making it exceptionally well-suited for large-scale implementations on GPUs. Note that EOT encourages couplings that exhibit high entropy, which is technically distinct from MEC, which aims for couplings with minimal entropy.

While it is shown in [72, 74] that MEC is NP-Hard, the literature contains many approximation algorithms for this problem. One of the earliest greedy algorithms for MEC was introduced by [84] in the context of causal inference. Specifically, the authors considered the problem of identifying the causal direction between two discrete random variables X and Y. Generally, X causes Y if there exists an exogenous random variable E (independent of X) and a deterministic function f such that $Y = f(X, E)$, with the central assumption that the exogenous variable has low entropy in the true direction and high entropy in the wrong direction. They show that the problem of finding the exogenous variable with minimum entropy is equivalent to the problem of finding minimum joint entropy given marginal distributions. They propose a greedy algorithm for MEC that finds a local minimum having $1 + \log n$ bits gap from the global optimum, with n being the cardinality of the support of random variables. Later in [85], this gap was shown to be tighter; within $\log_2(e)$ bits of the optimal.

Based on tools from the theory of majorization [86], the authors of [87] developed a new greedy algorithm and demonstrated that it produces a joint distribution whose entropy exceeds the minimum possible by no more than 1 bit. They also expanded this algorithm to accommodate k random variables, ensuring an additive gap of no more than $\log k$ bits. Subsequent improvements in [88] enabled the construction of a coupling whose entropy is within 2 bits of the optimal value, regardless of the number of random variables involved.

While [85] showed that improvements beyond the majorization lower bounds are impossible, thus establishing the *majorization barrier*, [89] deviated from the majorization strategy by introducing a new technique for lower bounding the coupling entropy, termed the profile method. This approach yields stronger bounds for coupling two or more random variables.

Minimum entropy coupling finds innovative applications beyond causal inference [84, 90, 91]. For instance, [92] utilized it in Markov coding games to enable reinforcement learning agents to communicate via Markov decision process trajectories. This application showcased MEC's utility in enabling efficient information transmission through constrained environments like video game interactions. Similarly, [93] applied MEC in communication but in the context of steganography, to securely encode secret information in regular text, showing MEC corresponds to the maximally efficient secure procedure.

These applications, along with those in dimension reduction of probability distribution [94] and stochastic processes [72], underline the broad applicability and significance of minimum entropy coupling across different domains of computer science and information theory.

4.2.2 Information Bottlneck

The Information Bottleneck (IB) method [95], addresses the problem of lossy data compression and representation. The IB principle seeks an optimal trade-off between the compression of a random variable X and the fidelity of compressed representation relative to a relevant variable Y. The IB problem is formalized through the following optimization objective:

$$\min_{p(t|x)} I(X;T) - \beta I(T;Y) \tag{4.14}$$

subject to the Markov constraint $T \leftrightarrow X \leftrightarrow Y$. Note that X is the input variable, Y is the variable of interest with which X shares mutual information, and T represents the compressed representation of X. The term $I(X;T)$ measures the mutual information between X and its representation T, serving as a quantification of the compression level, whereas $I(T;Y)$ quantifies the preserved relevant information about Y in the representation T. The parameter $\beta > 0$ balances the trade-off between compression and fidelity.

Solving the optimization in (4.14) is inherently non-trivial, often requiring iterative algorithms for its resolution. The IB framework has found extensive applications across various domains, including clustering [96], deep learning [97, 98], and quantization [99].

Deterministic information bottleneck [100] proposes a new formulation of the information bottleneck focusing on redefining the concept of compression from minimizing mutual information $I(X;T)$ as a communication cost between X and T, towards $H(T)$ from a source coding perspective that emphasizes reducing the representational cost of T.

Inherently, the information bottleneck and our proposed framework impose relevance on the compressed code in distinct ways. The differing modeling assumptions are evident from the unique Markov chain constraints within the two frameworks: $T \leftrightarrow X \leftrightarrow Y$ in information bottleneck, versus $X \leftrightarrow T \leftrightarrow Y$ in minimum entropy coupling with bottleneck. While IB defines relevance in terms of the information provided about a secondary variable Y, MEC with bottleneck directly establishes the relationship of the input X to the output Y through the code. Specifically, the input and output are conditionally independent given the code.

4.2.3 Lossy Source Coding

While log-loss is widely used in prediction and learning, its application as a distortion measure in the context of source coding has been less explored, with the first examples appearing in [69] and [70]. Log-loss is particularly suited as a distortion measure in soft reconstructions, meaning the decoder outputs a distribution.

[71] explores a single-shot lossy source coding setting under logarithmic-loss, using a straightforward encoding scheme. Unlike the EBIM formulation in (4.5) which imposes a direct entropy constraint on the code, this approach constrains the code by the cardinality of its support. In their compression scheme, each input symbol is encoded into a message that has accumulated the smallest total

probability up to that point. We numerically compare our proposed solution to EBIM with this encoding scheme in Section 4.4.1.

Finally, [101] examines a lossy compression scenario where the reconstruction distribution differs from the source distribution, similar to our approach. While our setup uses log-loss, they apply Mean Squared Error (MSE) as their distortion metric. They demonstrate that their setting can be formulated as a generalization of optimal transport with an entropy bottleneck. Additionally, they analyze the trade-off between compression rate and achievable distortion, with and without shared common randomness between the encoder and decoder.

4.3 Minimum Entropy Coupling

Consider two discrete random variables X and Y, over alphabets $\mathcal{X}$ and $\mathcal{Y}$ with probability mass functions p_X and p_Y, respectively. The goal of minimum entropy coupling is to find the joint distribution p_{XY} that minimizes the joint entropy $H(X,Y)$:

$$\begin{aligned} &\min_{p_{XY}} H(X;Y) \\ &\text{s.t.} \sum_{y\in\mathcal{Y}} p_{XY}(x,y) = p_X(x) \quad \forall x \in \mathcal{X}, \\ &\qquad \sum_{x\in\mathcal{X}} p_{XY}(x,y) = p_Y(y) \quad \forall y \in \mathcal{Y} \end{aligned} \tag{4.15}$$

This is a concave minimization problem over a standard polyhedron [102]. Therefore, every vertex of the polyhedron is a local minimum and the global minimum happens at a subset of the vertices. Note that an standard polyhedron is defined as $\mathcal{P} = \{\boldsymbol{x} \in \mathbb{R}^n | \ \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \ \boldsymbol{x} \geq \boldsymbol{0}\}$, where $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ with linearly independent rows. A point $\boldsymbol{x}^* \in \mathcal{P}$ is a vertex if and only if it has $n-m$ zero elements and columns of $\boldsymbol{A}$ corresponding to other m non-zero elements are linearly independent. Hence, to exhaustively iterate all the vertices:

1. Choose m linearly independent columns $\boldsymbol{A}_{\pi(1)}, \cdots, \boldsymbol{A}_{\pi(m)}$.
2. Let $\boldsymbol{x}_i = 0$ for all $i \in \pi(1), ..., \pi(m)$
3. Solve the system of m equations $\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = 0$ for the unknowns $\boldsymbol{x}_\pi(1), \cdots, \boldsymbol{x}_\pi(m)$

Therefore a crude upper-bound on the number of vertices would be $\binom{n}{m}$. This can be enhanced to $\binom{n-m/2}{m/2}$ (see [103]), which is still exponential in m. In fact, [74] shows that the minimum entropy coupling problem as defined in (4.15) is NP-Hard. For the sake of completeness, we include here a proof based on a reduction from another NP-complete problem, k-Subset-Sum.

Remark 4.1. *The minimum entropy coupling problem in (4.15) is NP-Hard [74].*

Proof. To show an optimization problem is NP-Hard, we need to show the corresponding decision problem is NP-Hard. Given an optimization problem, a decision version is whether or not any target value t is achievable. Without the loss of generality, assume $|\mathcal{X}| > |\mathcal{Y}|$. We set $t = H(Y)$, i.e., to decide if there exists a function $f : \mathcal{X} \to \mathcal{Y}$ such that $Y = f(X)$. Let's call this problem *Deterministic Matching*.

Next, we show any instance of the k-Subset-Sum problem can be reduced to an instance of Deterministic Matching, by a polynomial-time procedure (denoted by the notation $<_p$). Consider a general instance of the k-Subset-Sum problem: Given set $\mathcal{S}$ of integers and target values $\{t_i | 1 \leq i \leq k\}$, decide if there exists a partition $\{\mathcal{S}_i | 1 \leq i \leq k\}$ of size k on $\mathcal{S}$ such that $\sum \mathcal{S}_i = t_i$ for all $1 \leq i \leq k$. Now, set $p_X(i) = s_i / \sum(\mathcal{S}), \forall s_i \in \mathcal{S}$ and $p_Y(i) = t_i / \sum(t_j)$. Then, clearly solving Deterministic Matching for p_X, p_Y will solve the original k-Subset-Sum problem. Therefore, k-Subset-Sum $<_p$ Deterministic Matching and hence, Deterministic Matching is NP-Hard. Consequently, Minimum Entropy Coupling is an NP-Hard optimization problem. □

Finally, we introduce two linear-time approximate greedy algorithms for the minimum entropy coupling problem, and numerically compare their achieved minima to a general approximate algorithm.

Algorithm 4.1 Max-Seeking Minimum Entropy Coupling

Input: marginal distributions p_X, p_Y

Output: joint distribution p_{XY}

$p_{XY}(x, y) \leftarrow 0, \quad \forall x, y \in \mathcal{X}, \mathcal{Y}$
while $p_X, p_Y \neq \mathbf{0}$ **do**
  $x^* \leftarrow \text{argmax}_x \, p_X(x)$
  $y^* \leftarrow \text{argmax}_y \, p_Y(y)$
  $p_{XY}(x^*, y^*) \leftarrow \min\{p_X(x^*), p_Y(y^*)\}$
  $p_X(x^*) \leftarrow p_X(x^*) - \min\{p_X(x^*), p_Y(y^*)\}$
  $p_Y(y^*) \leftarrow p_Y(y^*) - \min\{p_X(x^*), p_Y(y^*)\}$
return p_{XY}

Algorithm 4.2 Zero-Seeking Minimum Entropy Coupling

Input: marginal distributions p_X, p_Y

Output: joint distribution p_{XY}

$p_{XY}(x, y) \leftarrow 0, \quad \forall x, y \in \mathcal{X}, \mathcal{Y}$
while $p_X, p_Y \neq \mathbf{0}$ **do**
  $(x^*, y^*) \leftarrow \text{argmin}_{x,y} \, |p_X(x) - p_Y(y)|$
  $p_{XY}(x^*, y^*) \leftarrow \min\{p_X(x^*), p_Y(y^*)\}$
  $p_X(x^*) \leftarrow p_X(x^*) - \min\{p_X(x^*), p_Y(y^*)\}$
  $p_Y(y^*) \leftarrow p_Y(y^*) - \min\{p_X(x^*), p_Y(y^*)\}$
return p_{XY}

At each step, each algorithm selects a symbol from each random variable and connects them in the joint distribution by assigning the higher probability of the two symbols, updating the marginals accordingly. The max-seeking version targets the symbols with the largest remaining probability mass at each step, whereas the zero-seeking version pairs symbols with the most similar probability mass. Furthermore, the greedy algorithm described in [84] resembles the max-seeking version outlined in Algorithm 4.1.

As a simple baseline, we randomly generated 100 pairs of joint distributions and fed them to our greedy solvers. We also used a general concave minimization method, Successive Linearization

Algorithm (SLA) [104], and compared the achieved joint entropy. Table 4.1 summarizes the average joint entropy over 100 trials for each method.

Table 4.1: Minimum Entropy Coupling: achieved joint entropy of various approximations.

Name	**Entropy**
Independent Joint	5.443 ± 0.101
SLA	3.225 ± 0.141
Max-Seeking Greedy	2.946 ± 0.064
Zero-Seeking Greedy	2.937 ± 0.058

4.4 Entropy-Bounded Information Maximization

Consider a discrete random variable X defined over the alphabet $\mathcal{X} = \{1, \ldots, n\}$ with a given marginal probability distribution p_X. The following problem aims to establish a maximal information coupling between X and another random variable T, defined over the alphabet $\mathcal{T} = \{1, \ldots, m\}$, where the entropy of T is constrained to be no more than R bits. Unlike minimum entropy coupling, the marginal distribution of the second random variable T is not predetermined; the only constraint on T is its entropy.

$$\mathcal{I}_{\text{EBIM}}(p_X, R) = \max_{p_{XT} \in \mathcal{M}} I(X;T), \tag{4.16}$$

where set $\mathcal{M}$ consists of all joint distributions p_{XT} that satisfy the following conditions:

1. $\sum_t p_{XT}(x,t) = p_X(x)$, ensuring that the marginal distribution of X is preserved.
2. $H(T) \leq R \leq H(X)$, ensuring the entropy of T is constrained to be no more than R.

We call this problem Entropy-Bounded Information Maximization (EBIM). Note that the objective in (4.16) is upper-bounded by R, since:

$$\begin{aligned} \mathcal{I}_{\text{EBIM}}(p_X, R) &= \max_{p_{XT} \in \mathcal{M}} I(X;T) \\ &\leq \max_{p_{XT} \in \mathcal{M}} H(T) \leq R. \end{aligned} \tag{4.17}$$

Theorem 4.1 shows that only deterministic couplings can achieve this upper-bound.

Theorem 4.1. *$\mathcal{I}_{EBIM}(p_X, R) = R$ if and only if there exists a function $g : \mathcal{X} \to \mathcal{T}$ such that $H(g(X)) = R$.*

Proof. If such g exists, let

$$p^*_{XT}(x,t) = \begin{cases} p_X(x) & t = g(x) \\ 0 & \text{otherwise} \end{cases}$$

This joint distribution effectively sets $T = g(X)$. Note that $p^*_{XT} \in \mathcal{M}$ and we have $I(X;T) = H(T) - H(T|X) = H(g(X)) = R$. Since $\mathcal{I}_{\text{EBIM}}(p_X, R) \leq R$, we conclude that $\mathcal{I}_{\text{EBIM}}(p_X, R) = R$ for $p_{XT} = p^*_{XT}$.

Conversely, if $\mathcal{I}_{\text{EBIM}}(p_X, R) = R$, then there exists $p^*_{XT} \in \mathcal{M}$ such that $I(X;T) = R$. Therefore

$$H(T) = I(X;T) + H(T|X) = R + H(T|X) \leq R$$
$$\Rightarrow H(T|X) \leq 0$$

As a result, $H(T|X) = 0$ and $H(T) = R$, which means p^*_{XT} defines a function g such that $T = g(X)$, and $H(g(X)) = H(T) = R$. □

Remark 4.2. *The mutual information $I(X;T)$ is invariant to permutations on T. Specifically, for any permutation $\pi : \mathcal{T} \to \mathcal{T}$, the mutual information remains unchanged, that is, $I(X;T) = I(X, \pi(T))$. Given that problem (4.16) only constrains the entropy $H(T)$, it is indifferent to column-permutations of the joint distribution p_{XT}.*

Definition 4.1. *The permutation group of the joint distribution p_{XT} is defined to be the set of all its column-permutations:*

$$\{P \mid P(x, \pi(t)) = p_{XT}(x,t), \quad \forall \pi : \mathcal{T} \to \mathcal{T}\} \tag{4.18}$$

Remark 4.3. *Each partition of $\mathcal{X}$ is associated with a permutation group of a deterministic mapping. Consequently, the total number of potential deterministic mapping groups, independent of the entropy constraint on T, will be the total number of feasible partitions of $\mathcal{X}$. The total number of ways to partition a set of size n corresponds to the n-th Bell number [1], symbolized by B_n. The growth rate of the Bell numbers is $O(n^n)$.*

Note that while Remark 4.3 highlights the infeasibility of iterating over all deterministic mappings, this does not necessarily imply that EBIM is NP-hard.

Example. *For $p_X = \begin{bmatrix}0.5 & 0.3 & 0.2\end{bmatrix}$, there exists 5 deterministic mappings corresponding to 5 possible partitions of $\mathcal{X} = \{1,2,3\}$:*

$$\underset{\begin{bmatrix}0.5&0&0\\0.3&0&0\\0.2&0&0\end{bmatrix}}{\{1,2,3\}} \quad \underset{\begin{bmatrix}0.5&0&0\\0&0.3&0\\0&0.2&0\end{bmatrix}}{\{1\},\{2,3\}} \quad \underset{\begin{bmatrix}0.5&0&0\\0.3&0&0\\0&0.2&0\end{bmatrix}}{\{1,2\},\{3\}} \quad \underset{\begin{bmatrix}0.5&0&0\\0&0.3&0\\0.2&0&0\end{bmatrix}}{\{1,3\},\{2\}} \quad \underset{\begin{bmatrix}0.5&0&0\\0&0.3&0\\0&0&0.2\end{bmatrix}}{\{1\},\{2\},\{3\}}$$

In Figure 4.1, we applied a brute force method to solve EBIM (4.16) for $p_X = [0.7, 0.2, 0.1]$. As observed, there are 5 potential partitions on p_X, each corresponding to a point where $\mathcal{I}_{\text{EBIM}}(p_X, R) = R$. In Section 4.4.1, we introduce a greedy search algorithm to identify deterministic mappings with a guaranteed gap from the optimal. Subsequently, in Section 4.4.2, we explore optimal mappings close to the deterministic mappings, providing a strategy to close the gap between the deterministic mappings.

[1] https://en.wikipedia.org/wiki/Bell_number

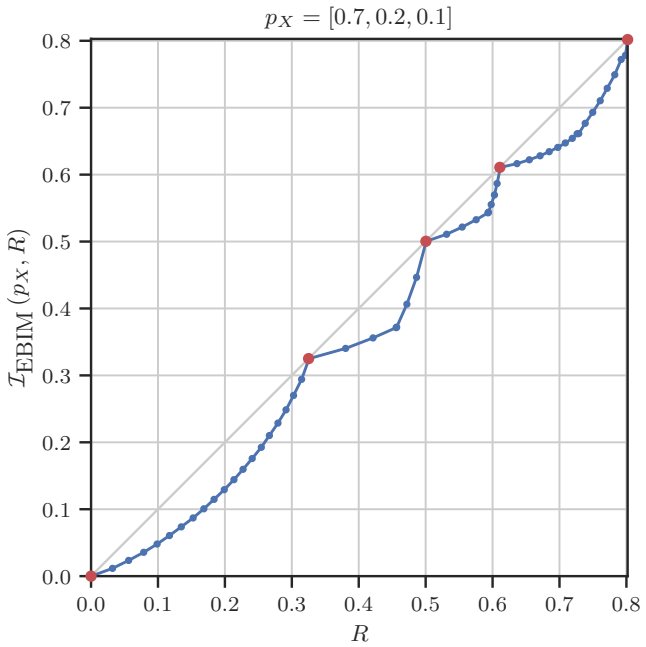


Figure 4.1: Brute force solutions for $\mathcal{I}_{\text{EBIM}}(p_X, R)$ across possible values of R with $p_X = [0.7, 0.2, 0.1]$. Points where $\mathcal{I}_{\text{EBIM}}(p_X, R) = R$ are marked with red circles, indicating the 5 possible deterministic mappings (5 possible partitions of $\mathcal{X}$).

4.4.1 Proposed Search Algorithm for Deterministic Mappings

Since iterating over all deterministic mappings is not feasible, one should look for carefully constructed search algorithms to find such mappings with resulting $H(T)$ as close as possible to R. Without the loss of generality, suppose $p_X = \left[p_1,\ p_2,\ \cdots,\ p_n\right]$ is arranged in a decreasing order. Algorithm 4.3 presents a search approach for discovering a deterministic mapping $T = g(X)$, resulting in $I(X;T)$ that is at most $h(p_2)$ bits away from the optimal $\mathcal{I}_{\text{EBIM}}(p_X, R)$.

Algorithm 4.3 Deterministic EBIM Solver

Input: p_X, R
Output: p_{XT}

$p_{XT} \leftarrow \text{diag}(p_X)$ ▷ define $p^{(0)} := \text{diag}(p_X)$
if $R \geq H(p_X)$ **then**
    return p_{XT}
for $i \leftarrow 1$ to $|p_X| - 1$ **do**
    $p^{(i)} \leftarrow$ Merge two columns with the largest sum in p_{XT}.
    $I^{(i)} \leftarrow I(X;T)$ imposed by $p^{(i)}$.
    if $I^{(i)} \leq R$ **then**
        return $p^{(i)}$
    else
        $p_{XT} \leftarrow p^{(i)}$

Let $n = |p_X|$. The procedure outlined in Algorithm 4.3 establishes a series of deterministic mappings $p^{(0)}, p^{(1)}, p^{(2)}, \cdots, p^{(n-1)}$, corresponding to a decreasing sequence of mutual information values

$I^{(0)}$, $I^{(1)}$, $I^{(2)}$, $\cdots$, $I^{(n-1)}$. This is done by iteratively merging two elements in $\mathcal{T}$ with the largest probabilities. The algorithm then picks the mapping with the maximum mutual information that does not exceed R. Therefore

$$\begin{aligned} R - I(X;T) &\leq \max\left\{I^{(0)} - I^{(1)},\ I^{(1)} - I^{(2)},\ \cdots,\ I^{(n-2)} - I^{(n-1)}\right\} \\ &\leq \max_{i\in\{1,\cdots,n-1\}} I^{(i-1)} - I^{(i)}. \end{aligned} \tag{4.19}$$

Example. *For* $p_X = \begin{bmatrix}0.4 & 0.2 & 0.15 & 0.15 & 0.1\end{bmatrix}$, *Algorithm 4.3 traverses through the following deterministic mappings, from left to right:*

	$p^{(0)}$	$p^{(1)}$	$p^{(2)}$	$p^{(3)}$	$p^{(4)}$
p_{XT}	$\begin{bmatrix}.4 & 0 & 0 & 0 & 0\\ 0 & .2 & 0 & 0 & 0\\ 0 & 0 & .15 & 0 & 0\\ 0 & 0 & 0 & .15 & 0\\ 0 & 0 & 0 & 0 & .1\end{bmatrix}$	$\begin{bmatrix}.4 & 0 & 0 & 0\\ .2 & 0 & 0 & 0\\ 0 & .15 & 0 & 0\\ 0 & 0 & .15 & 0\\ 0 & 0 & 0 & .1\end{bmatrix}$	$\begin{bmatrix}.4 & 0 & 0\\ .2 & 0 & 0\\ .15 & 0 & 0\\ 0 & .15 & 0\\ 0 & 0 & .1\end{bmatrix}$	$\begin{bmatrix}.4 & 0\\ .2 & 0\\ .15 & 0\\ .15 & 0\\ 0 & .1\end{bmatrix}$	$\begin{bmatrix}.4\\ .2\\ .15\\ .15\\ .1\end{bmatrix}$
$I(X;T)$	$H(p_X)$	$H\left(\begin{bmatrix}.6 & .15 & .15 & .1\end{bmatrix}\right)$	$H\left(\begin{bmatrix}.75 & .15 & .1\end{bmatrix}\right)$	$H\left(\begin{bmatrix}.9 & .1\end{bmatrix}\right)$	0

Definition 4.2. *Let* μ' *be a probability distribution resulted from merging two elements* $p > 0$ *and* $q > 0$ *in an original distribution* μ, *i.e.* $\mu = \begin{bmatrix}\cdots & p & \cdots & q & \cdots\end{bmatrix}$ *and* $\mu' = \begin{bmatrix}\cdots & p+q & \cdots\end{bmatrix}$. *Then, the amount of decrease in the entropy from this merge operation is characterized by:*

$$\begin{aligned} \Delta_H(p,\ q) &= H(\mu) - H(\mu') \\ &= p\log\frac{1}{p} + q\log\frac{1}{q} - (p+q)\log\frac{1}{p+q} \\ &= p\log\left(1+\frac{q}{p}\right) + q\log\left(1+\frac{p}{q}\right) \end{aligned}$$

Lemma 4.2. *The following properties hold for the function* Δ_H*:*

1. $\Delta_H(\cdot,\ \cdot)$ *is monotonically increasing in both arguments.*
2. $\Delta_H(\cdot,\ \cdot)$ *is concave.*
3. $\Delta_H(p,\ 1-p) = h(p)$.

Proof. The properties are derived through straightforward derivative calculations:

1. $\frac{\partial}{\partial p}\Delta_H = \log(1+q/p) \geq 0$, and $\frac{\partial}{\partial q}\Delta_H = \log(1+p/q) \geq 0$.
2. The Hessian of Δ_H is negative semidefinite:

$$\mathbf{H}_{\Delta_H} = \frac{1}{p+q}\begin{bmatrix}-q/p & 1\\ 1 & -p/q\end{bmatrix}$$

with eigenvalues $\lambda_1 = 0$ and $\lambda_2 = -(\frac{q}{p} + \frac{p}{q})(\frac{1}{p+q}) < 0$.

3. $\Delta_H\left(p\,,\ 1-p\right) = -p\log p - (1-p)\log(1-p) = h(p)$.

□

Theorem 4.2. *If the output of Algorithm 4.3 yields mutual information $\widehat{I}$, then*

$$\mathcal{I}_{EBIM}(p_X, R) - \widehat{I} \leq h(p_2),$$

where $h(\cdot)$ is the binary entropy function, and p_2 denotes the second largest element of p_X.

Proof. For the gap to the optimal objective, $\mathcal{I}_{\text{EBIM}}(p_X, R) - \widehat{I}$, we have:

$$\begin{aligned}
&\triangleright \text{ Equation (4.17)} && \mathcal{I}_{\text{EBIM}}(p_X, R) - \widehat{I} \leq R - \widehat{I} \\
&\triangleright \text{ Equation (4.19)} && \leq \max_{i\in\{1,\cdots,n-1\}} I^{(i-1)} - I^{(i)} \\
&\triangleright \text{ Algorithm 4.3, Line 6} && = \max_{i\in\{1,\cdots,n-1\}} H\left(\sum_x p^{(i-1)}\right) - H\left(\sum_x p^{(i)}\right) \\
&\triangleright \text{ Definition 4.2} && = \max_{i\in\{1,\cdots,n-1\}} \Delta_H\left(\sum_{k=1}^{i} p_k\,,\ p_{i+1}\right) \\
&\triangleright \text{ Lemma 4.2.1} && \leq \max_{i\in\{1,\cdots,n-1\}} \Delta_H\left(\sum_{k=1}^{i} p_k + \sum_{k=i+2}^{n} p_k\,,\ p_{i+1}\right) \\
& && = \max_{i\in\{1,\cdots,n-1\}} \Delta_H\left(1 - p_{i+1}\,,\ p_{i+1}\right) \\
&\triangleright \text{ Lemma 4.2.3} && = \max_{i\in\{1,\cdots,n-1\}} h\left(p_{i+1}\right) \\
&\triangleright \ p_2, p_3, \cdots, p_n \leq 0.5 && = h(p_2)
\end{aligned}$$

□

As discussed in Section 4.2, our proposed search method in Algorithm 4.3 is compared with the encoder from Shkel et al. (2017) [71]. Our formulation directly imposes an entropy constraint on the code, whereas the encoding scheme by Shkel et al. limits the code by its alphabet size. In their approach, given a code alphabet size, the encoder iterates over all input symbols, assigning each one to a message that has accumulated the smallest total probability up to that point.

Figure 4.2 displays the mutual information obtained for each maximum allowed code rate value, considering two different input distributions. As observed, the two methods yield comparable mutual information in the high-rate regime. However, in the low-rate regime, our proposed algorithm identifies more mappings and thus significantly outperforms the encoder described in [71].

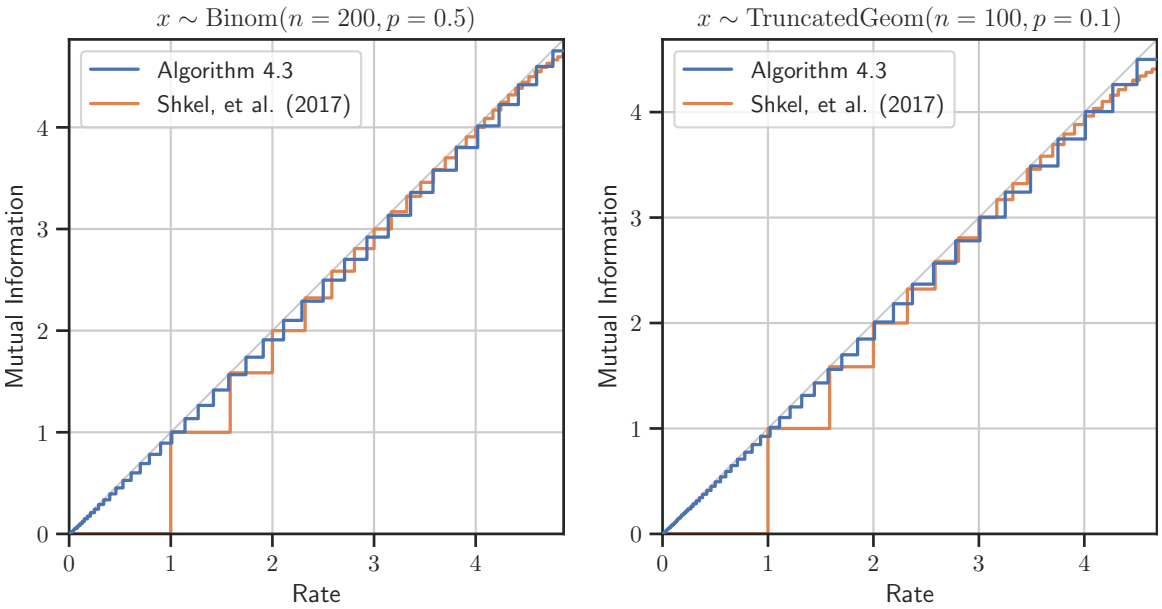


Figure 4.2: Obtained $I(X;T)$ vs. maximum allowed $H(T)$ for Binomial (left) and Truncated Geometric (right) input distributions.

Additionally, note that the couplings generated by Algorithm 4.3 display finer granularity at lower rates. This occurs because the algorithm merges input symbols with the largest probabilities at each iteration. However, as shown in Figure 4.3, more couplings are generated at higher rates if the merger involves the two elements with the smallest probability. Therefore, a *hybrid* merger strategy optimizes granularity across both low and high rate regimes by performing a largest merger for rates less than $H(X)/2$ and a smallest merger for rates greater than $H(X)/2$, as in Figure 4.3 left.

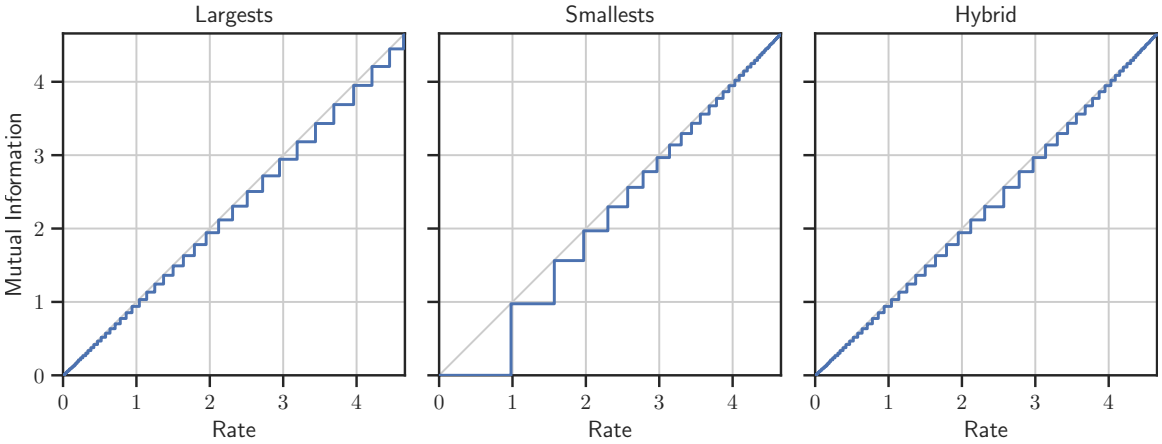


Figure 4.3: Iteratively merging two elements in $\mathcal{T}$ with either the largest (left) or smallest (middle) probabilities. A *hybrid* merger (right) equates to a largest merger if the compression rate exceeds 2 and to a smallest merger otherwise.

Algorithm 4.4 demonstrates this hybrid merger strategy.

Algorithm 4.4 Deterministic EBIM Solver with Hybrid Merger

Input: p_X, R
Output: p_{XT}

$p_{XT} \leftarrow \text{diag}(p_X)$
if $R \geq H(p_X)$ **then**
    return p_{XT}
for $i \leftarrow 1$ to $|p_X| - 1$ **do**
    if $R > H(p_X)/2$ **then**
        $p^{(i)} \leftarrow$ Merge two columns with the smallest sum in p_{XT}.
    else
        $p^{(i)} \leftarrow$ Merge two columns with the largest sum in p_{XT}.
    $I^{(i)} \leftarrow I(X;T)$ imposed by $p^{(i)}$.
    if $I^{(i)} \leq R$ **then**
        return $p^{(i)}$
    else
        $p_{XT} \leftarrow p^{(i)}$

4.4.2 Optimal Coupling Around Deterministic Mappings

Section 4.4.1 introduced a greedy algorithm for identifying deterministic mappings with a guaranteed gap from the optimal. In this section, we identify optimal mappings close to any deterministic mapping. This approach will enable us to bridge the gap between the deterministic mappings.

Theorem 4.3. *Let p_{XT} denoted by a $|\mathcal{X}| \times |\mathcal{T}|$ matrix, defines a deterministic mapping $T = g(X)$, with $I(X;T) = H(T) = R_g$. We have $\mathcal{I}_{EBIM}(p_X, R_g) = R_g$, and for small enough $\epsilon > 0$:*

1. *$\mathcal{I}_{EBIM}(p_X, R_g + \epsilon)$ is achieved as follows:*
   Normalize the columns by dividing each column by its sum. Then, select the cell with the smallest normalized value and move an infinitesimal probability mass from this cell to a new column of p_{XT} in the same row.

2. *$\mathcal{I}_{EBIM}(p_X, R_g - \epsilon)$ is achieved as follows:*
   Identify the columns with the smallest and largest sums in p_{XT}. Select the cell with the smallest value in the column with the lowest sum. Transfer an infinitesimal probability mass from this cell to the column with the highest sum in the same row.

Example. *The following depicts an application of Theorem 4.3:*

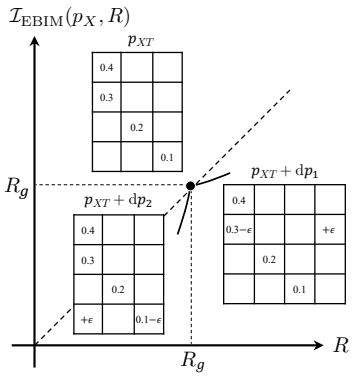


Figure 4.4: Optimal solutions in the neighbourhood of a deterministic mapping.

Proof. Let us view the joint distribution as a $n \times m$ matrix p_{XT}. Note that:

$$p_{XT}(x,t) = \begin{cases} p_X(x) & t = g(x) \\ 0 & \text{otherwise} \end{cases} \tag{4.20}$$

For example

$$p_{XT} = \begin{bmatrix} 0.4 & 0 & 0 & 0 \\ 0.3 & 0 & 0 & 0 \\ 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \end{bmatrix}, \quad g(x) = \begin{cases} 1, & x = 1,2 \\ 2, & x = 3 \\ 3, & x = 4 \end{cases} \tag{4.21}$$

Consider a perturbation $\mathrm{d}P \in \mathbb{R}^{n \times m}$ to p_{XT}. For $p_{XT} + \mathrm{d}P$ to be a valid distribution in $\mathcal{M}$, we need:

1. $\sum_t \mathrm{d}P(x,t) = 0, \quad \forall x \in \mathcal{X}$
2. $\mathrm{d}P(x,t) \geq 0, \quad \forall x,t \text{ s.t. } t \neq g(x)$
3. $\mathrm{d}P(x,t) \leq 0, \quad \forall x,t \text{ s.t. } t = g(x)$

We define the set of all such perturbations as $\Omega \subset \mathbb{R}^{n \times m}$. Next, let us define basis perturbations $\Delta_{x,t}$ for $t \neq g(x)$ as:

$$[\Delta_{x,t}]_{ij} = \begin{cases} -\varepsilon, & \text{if } i = x,\ j = g(x) \\ +\varepsilon, & \text{if } i = x,\ j = t \\ 0, & \text{otherwise} \end{cases} \tag{4.22}$$

Note that $\Delta_{x,t}$ represents moving a probability mass of ε from non-zero cell $(x, g(x))$ to empty cell (x,t). For the example in equation (4.21):

$$\Delta_{2,3} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\varepsilon & 0 & +\varepsilon & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

The significance of these bases is that any perturbation in Ω can be represented as:

$$\mathrm{d}P = \sum_{\substack{x,t \\ t \neq g(x)}} \alpha_{x,t}\ \Delta_{x,t}, \tag{4.23}$$

with coefficients $\alpha_{x,t} \geq 0$. For example:

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ -3\varepsilon & +2\varepsilon & +\varepsilon & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} = 2 \times \Delta_{1,1} + \Delta_{1,2}$$

Realizing I_{XT}, H_{XT}, and H_T as functions of a joint distribution, we are interested in calculating the ratio $\mathrm{d}I_{XT}/\mathrm{d}H_T$ with respect to a perturbation $\mathrm{d}P \in \Omega$ as $\varepsilon \to 0$ at p_{XT}. Note that since for any $\mathrm{d}P \in \Omega$, $\mathrm{d}H_X = 0$, we have:

$$\frac{\mathrm{d}I_{X,T}}{\mathrm{d}H_T} = \frac{\mathrm{d}H_X + \mathrm{d}H_T - \mathrm{d}H_{X,T}}{\mathrm{d}H_T} = 1 - \frac{\mathrm{d}H_{X,T}}{\mathrm{d}H_T}.$$

Therefore

$$\begin{aligned}
\frac{\mathrm{d}I_{X,T}}{\mathrm{d}H_T} &= 1 - \frac{\mathrm{d}H_{X,T}\left(p_{XT}, \mathrm{d}P\right)}{\mathrm{d}H_T\left(p_{XT}, \mathrm{d}P\right)} \\
&= 1 - \frac{\mathrm{d}H_{X,T}\left(p_{XT}, \sum_x \sum_{t \neq g(x)} \alpha_{x,t}\ \Delta_{x,t}\right)}{\mathrm{d}H_T\left(p_{XT}, \sum_x \sum_{t \neq g(x)} \alpha_{x,t}\ \Delta_{x,t}\right)} \\
&= 1 - \frac{\sum_x \sum_{t \neq g(x)} \alpha_{x,t}\ \mathrm{d}H_{X,T}\left(p_{XT}, \Delta_{x,t}\right)}{\sum_x \sum_{t \neq g(x)} \alpha_{x,t}\ \mathrm{d}H_T\left(p_{XT}, \Delta_{x,t}\right)}
\end{aligned} \tag{4.24}$$

$\mathrm{d}H_{X,T}\left(p_{XT}, \Delta_{x,t}\right)$ represents the amount of change in the joint entropy, when an infinitesimal mass of ε is moved from $(x, g(x))$ to (x, t). More precisely, from (4.20) and (4.22):

$$\begin{aligned}
\mathrm{d}H_{X,T}\left(p_{XT}, \Delta_{x,t}\right) &= H_{X,T}\left(p_{XT} + \Delta_{x,t}\right) - H_{X,T}\left(p_{XT}\right) \\
&= \left[-(p_X(x) - \varepsilon)\log(p_X(x) - \varepsilon) - \varepsilon \log \varepsilon\right] - \left[-p_X(x)\log p_X(x)\right] \\
&= p_X(x)\log\frac{p_X(x)}{p_X(x) - \varepsilon} + \varepsilon \log \frac{p_X(x) - \varepsilon}{\varepsilon} \\
&= \varepsilon + \mathcal{O}(\varepsilon^2) + \varepsilon \log \frac{p_X(x)}{\varepsilon}
\end{aligned} \tag{4.25}$$

The last line uses the fact that for small enough x, $f(x) = a \log \frac{a}{a-x} = x + \mathcal{O}(x^2)$. Similarly:

$$\begin{aligned}
\mathrm{d}H_T\left(p_{XT}, \Delta_{x,t}\right) &= H_T\left(p_{XT} + \Delta_{x,t}\right) - H_T\left(p_{XT}\right) \\
&= \Big[-\Big(p_T\big(g(x)\big) - \varepsilon\Big)\log\Big(p_T\big(g(x)\big) - \varepsilon\Big) - \Big(p_T(t) + \varepsilon\Big)\log\Big(p_T(t) + \varepsilon\Big)\Big] \\
&\qquad - \Big[-p_T\big(g(x)\big)\log p_T\big(g(x)\big) - p_T(t)\log p_T(t)\Big] \\
&= p_T\big(g(x)\big)\log\frac{p_T\big(g(x)\big)}{p_T\big(g(x)\big) - \varepsilon} + p_T(t)\log\frac{p_T(t)}{p_T(t) + \varepsilon} + \varepsilon\log\frac{p_T\big(g(x)\big) - \varepsilon}{p_T(t) + \varepsilon} \\
&= \varepsilon + \mathcal{O}(\varepsilon^2) - \varepsilon + \mathcal{O}(\varepsilon^2) + \varepsilon\log\frac{p_T\big(g(x)\big)}{p_T(t) + \varepsilon} \\
&= \log\frac{p_T\big(g(x)\big)}{p_T(t) + \varepsilon} + \mathcal{O}(\varepsilon^2)
\end{aligned} \tag{4.26}$$

Plugging (4.25) and (4.26) back to (4.24), we will get:

$$\frac{\mathrm{d}I_{X,T}}{\mathrm{d}H_T} = 1 - \frac{\sum_x \sum_{t\neq g(x)} \alpha_{x,t}\left[\varepsilon + \varepsilon \log \frac{p_X(x)}{\varepsilon} + \mathcal{O}(\varepsilon^2)\right]}{\sum_x \sum_{t\neq g(x)} \alpha_{x,t}\left[\log \frac{p_T\big(g(x)\big)}{p_T(t)+\varepsilon} + \mathcal{O}(\varepsilon^2)\right]}$$

$$= 1 - \frac{\sum_x \sum_{t\neq g(x)} \overline{\alpha}_{x,t} \log \frac{p_X(x)}{\varepsilon}}{\sum_x \sum_{t\neq g(x)} \overline{\alpha}_{x,t} \log \frac{p_T\big(g(x)\big)}{p_T(t)+\varepsilon}} \tag{4.27}$$

where $\alpha_{x,t}$ is normalized by:

$$\overline{\alpha}_{x,t} = \frac{\alpha_{x,t}}{\sum_x \sum_{t\neq g(x)} \alpha_{x,t}}.$$

Let's focus on the limit of (4.27) when $\varepsilon \to 0$: If there is any $t \in \mathcal{T}$ with $p_T(t) = 0$ and $\overline{\alpha}_{x,t} > 0$, the denominator of the second term will grow without bound, otherwise the denominator remains bounded. Therefore, for the limit of (4.27) we have:

$$\lim_{\varepsilon\to 0} \frac{\mathrm{d}I_{X,T}}{\mathrm{d}H_T} = \begin{cases} -\infty & \text{if} \quad \overline{\alpha}_{x,t} = 0 \quad \forall t \text{ s.t. } p_T(t) = 0 \\ 1 - \Big(\sum_x \sum_{p_T(t)=0} \overline{\alpha}_{x,t}\Big)^{-1} & \text{if} \quad \exists t : \overline{\alpha}_{x,t} > 0 \text{ and } p_T(t) = 0 \end{cases} \tag{4.28}$$

For $\mathrm{d}H_T > 0$, we need to find a perturbation (i.e., coefficients $\alpha_{x,t}$) that maximizes $\mathrm{d}I_{XT}/\mathrm{d}H_T$. From (4.28), this means $\exists t \in \mathcal{T}$ with $\overline{\alpha}_{x,t} > 0$ and $p_T(t) = 0$.

$$\overline{\alpha} = \operatorname*{argmax}_{\overline{\alpha}} \frac{\mathrm{d}I_{X,T}}{\mathrm{d}H_T} = \operatorname*{argmax}_{\overline{\alpha}} \; 1 - \Big(\sum_x \sum_{p_T(t)=0} \overline{\alpha}_{x,t}\Big)^{-1}$$

$$= \operatorname*{argmax}_{\overline{\alpha}} \sum_x \sum_{p_T(t)=0} \overline{\alpha}_{x,t}$$

Therefore, $\sum_x \sum_{p_T(t)=0} \overline{\alpha}_{x,t} = 1$ which means $\overline{\alpha}_{x,t} = 0$ if $p_T(t) > 0$. In other words, we should only consider perturbations where masses are moved to all-zero columns. Continuing (4.27):

$$\overline{\alpha} = \operatorname*{argmax}_{\overline{\alpha}} \; \frac{\mathrm{d}I_{X,T}}{\mathrm{d}H_T}$$

$$= \operatorname*{argmax}_{\overline{\alpha}} \; 1 - \frac{\sum_x \sum_{p_T(t)=0} \overline{\alpha}_{x,t} \log \frac{p_X(x)}{\varepsilon}}{\sum_x \sum_{p_T(t)=0} \overline{\alpha}_{x,t} \log \frac{p_T\big(g(x)\big)}{\varepsilon}}$$

$$= \operatorname*{argmax}_{\overline{\alpha}} \; 1 - \frac{-\log\varepsilon + \sum_x \sum_{p_T(t)=0} \overline{\alpha}_{x,t} \log p_X(x)}{-\log\varepsilon + \sum_x \sum_{p_T(t)=0} \overline{\alpha}_{x,t} \log p_T\big(g(x)\big)}$$

$$= \operatorname*{argmax}_{\overline{\alpha}} \quad \frac{-1}{\log \varepsilon} \left[\sum_x \sum_{p_T(t)=0} \overline{\alpha}_{x,t} \log \frac{p_T\big(g(x)\big)}{p_X(x)} \right]$$

$$= \operatorname*{argmin}_{\overline{\alpha}} \quad \sum_x \sum_{p_T(t)=0} \overline{\alpha}_{x,t} \log \frac{p_X(x)}{p_T\big(g(x)\big)}$$

This is achieved by selecting

$$\Rightarrow \alpha_{x,t} = \begin{cases} 1, & x = \operatorname*{argmin}_{x'} \dfrac{p_X(x')}{p_T\big(g(x')\big)} \quad \text{and} \quad p_T(t) = 0 \\ 0, & \text{otherwise} \end{cases} \tag{4.29}$$

In other words, first, normalize columns in p_{XT} by their sum, then move an infinitesimal probability mass from the cell with the smallest normalized value to an all-zero column. It is easy to confirm that $\mathrm{d}H_T > 0$ for such a perturbation. For the example distribution in (4.21):

$$p_{XT} + \mathrm{d}P = \begin{bmatrix} 0.4 & 0 & 0 & 0 \\ 0.3 - \varepsilon & 0 & 0 & \varepsilon \\ 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \end{bmatrix} \tag{4.30}$$

On the other hand, for $\mathrm{d}H_T < 0$, we need to find a perturbation (i.e., coefficients $\alpha_{x,t}$) that minimizes $\mathrm{d}I_{XT}/\mathrm{d}H_T$. From (4.28), this means $\overline{\alpha}_{x,t} = 0$ for all $t \in \mathcal{T}$ that $p_T(t) = 0$. Therefore, as in (4.27):

$$\overline{\alpha} = \operatorname*{argmin}_{\overline{\alpha}} \quad \frac{\mathrm{d}I_{X,T}}{\mathrm{d}H_T}$$

$$= \operatorname*{argmin}_{\overline{\alpha}} \quad 1 - \frac{\sum_x \sum_{t \neq g(x)} \overline{\alpha}_{x,t} \log \dfrac{p_X(x)}{\varepsilon}}{\sum_x \sum_{t \neq g(x)} \overline{\alpha}_{x,t} \log \dfrac{p_T\big(g(x)\big)}{p_T(t)}}$$

$$= \operatorname*{argmin}_{\overline{\alpha}} \quad 1 - \frac{-\log \varepsilon + \sum_x \sum_{t \neq g(x)} \overline{\alpha}_{x,t} \log p_X(x)}{\sum_x \sum_{t \neq g(x)} \overline{\alpha}_{x,t} \log \dfrac{p_T\big(g(x)\big)}{p_T(t)}}$$

$$= \operatorname*{argmin}_{\overline{\alpha}} \quad \sum_x \sum_{t \neq g(x)} \overline{\alpha}_{x,t} \log \frac{p_T\big(g(x)\big)}{p_T(t)}$$

This is achieved by selecting

$$\Rightarrow \overline{\alpha}_{x,t} = \begin{cases} 1, & x = \operatorname*{argmin}_{x'} p_T\big(g(x')\big) \quad \text{and} \quad t = \operatorname*{argmax}_{t'} p_T(t') \\ 0, & \text{otherwise} \end{cases} \tag{4.31}$$

In other words, moving an infinitesimal probability mass from the smallest column to the largest column of p_{XT}. It is easy to confirm that $\mathrm{d}H_T < 0$ for such a perturbation. For the example distribution of (4.21):

$$p_{XT} + \mathrm{d}P = \begin{bmatrix} 0.4 & 0 & 0 & 0 \\ 0.3 & 0 & 0 & 0 \\ 0 & 0.2 & 0 & 0 \\ \varepsilon & 0 & 0.1 - \varepsilon & 0 \end{bmatrix} \tag{4.32}$$

□

While Algorithm 4.3 effectively identifies deterministic mappings that produce mutual information close to the budget R, Theorem 4.3 can help bridge the remaining gap. More specifically, one can begin with a deterministic mapping and use two probability mass transformations outlined in Theorem 4.3 to navigate across the $I - R$ plane.

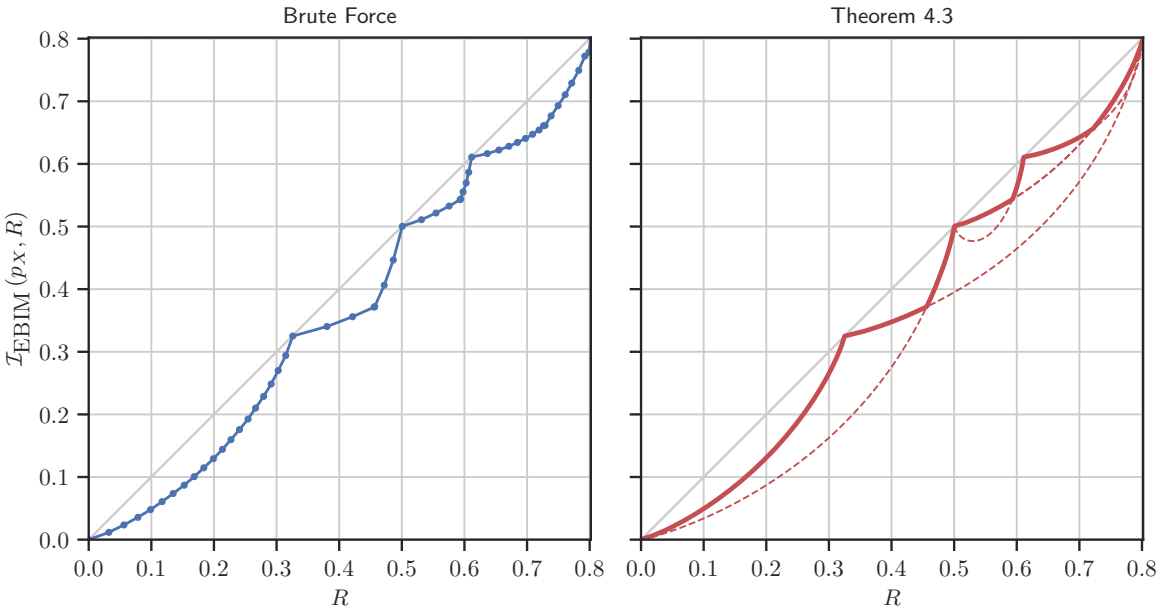


Figure 4.5: Solutions to the EBIM problem for $p_X = [0.7, 0.2, 0.1]$. Left: brute force solution. Right: Application of the transformations from Theorem 4.3 to each deterministic mapping (dashed lines) and selection of solutions with maximal mutual information for each R value (thick solid line). This strategy effectively recovers optimal solutions, aligning with those found by brute force in this case.

Figure 4.5 illustrates this strategy; for $p_X = [0.7,\ 0.2,\ 0.1]$, identifying all 5 possible deterministic mappings is straightforward. Applying the transformations from Theorem 4.3 then yields various solutions across the $I - R$ plane (represented by dashed lines). Subsequently, one can select the solution that maximizes mutual information for any given value of R (highlighted with a thick solid line), thus producing a comprehensive solution for every value of R. As demonstrated in Figure 4.5, this strategy recovers the optimal solutions, as determined by brute force, for the simple case of an input alphabet of three. However, the optimality of this approach, while effective, remains a conjecture.

4.5 Application: Markov Coding Game with Rate Limit

Markov Coding Games (MCGs) [92] represent a type of multi-player decentralized Markov Decision Process (MDP) that include the following components: a source, an agent, a Markov decision process, and a receiver. A Markov Coding Game unfolds over four distinct steps: In the first step, the agent receives a private message from the source, where it is responsible for indirectly conveying the message to the receiver. In the second step, the sender engages in a Markov decision process episode. Next, the receiver observes the sender's MDP trajectory. The final step involves the receiver attempting to decode the original message from the observed trajectory. The combined reward for both the sender and receiver in this game is determined by a combination of the total rewards accumulated during the MDP episode and a factor that indicates the accuracy of the receiver in decoding the message.

MCGs are particularly interesting due to their ability to extend the scope of several critical frameworks. A prime example is referential games, where a sender aims to communicate a message to a receiver through low-cost or inconsequential actions, known as cheap talk, which don't affect the game's transition or reward dynamics. MCGs can be seen as an expansion of referential games, incorporating scenarios where the sender's actions might have associated costs.

We will consider a natural extension to Markov Coding Games, where the link from the source to the agent is rate-limited. This means, contrary to the original setting in [92], the agent does not fully observe the message at each MDP round, but will receive a compressed version of the message iteratively, and in turn, encodes information about the message in the MDP trajectory for the receiver.

4.5.1 Backgrounds and Notations

Markov Decision Process MDPs are represented by the tuple notation $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ denoted the action space, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ defines the reward function, and $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ represents the transition function. The way an agent interacts with an MDP is determined by its policy $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$, which assigns distributions over actions for each state. Our main focus is on episodic MDPs, which terminate after a limited sequence of transitions. The sequence of states and actions, called a trajectory, is recorded as $z = (s_0, a_0, \ldots, s_T)$. The total rewards accrued throughout a sequence is $R(z) = \sum_t \gamma^t R(s_t, a_t)$, where $0 < \gamma < 1$ is the reward discount factor. The primary aim of an MDP is to devise a policy that maximizes the expected cumulative reward $\mathbb{E}[R(Z)|\pi]$.

Maximum Entropy Reinforcement Learning a policy that exhibits a high degree of randomness is preferred in certain situations. Under these circumstances, the maximum-entropy RL objective

$$\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t} R(S_t, A_t) + \beta H(A_t|S_t) \right] \tag{4.33}$$

serves as a compelling substitute to the conventional goal of maximizing expected aggregate rewards [105]. This objective trades off expected returns with conditional entropy of the selected policy, modulated by the temperature hyperparameter β. A generalization of the Q-value iteration

method for maximum-entropy RL objective (also known as soft Bellman equation [106]) is shown in Algorithm 4.5.

Algorithm 4.5 Soft Q-Value Iteration

Input: MDP, β
Initialize: π_0 to any policy
$i \leftarrow 0$
repeat
  $Q^{i+1}_{\text{soft}}(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} \Pr(s'|s,a) \widetilde{\max}_{a'\beta} Q^{i}_{\text{soft}}(s',a')$
  $i \leftarrow i+1$
until $\|Q^{i}_{\text{soft}}(s,a) - Q^{i-1}_{\text{soft}}(s,a)\|_\infty \leq \epsilon$
Extract policy: $\pi_{\text{greedy}}(\cdot|s) = \text{softmax}\left(Q^{i}_{\text{soft}}(s,\cdot)/\beta\right)$

The soft maximum operator is defined as $\widetilde{\max}_{a\beta} Q(s,a) = \beta \log \sum_a \exp\left(\frac{Q(s,a)}{\beta}\right)$.

Markov Coding Games with Rate Limit Following [92], we define a rate-limited MCG as a tuple $\langle(\mathcal{S},\mathcal{A},\mathcal{T},\mathcal{R}),\mathcal{M},\mu,\zeta,R\rangle$, where $(\mathcal{S},\mathcal{A},\mathcal{T},\mathcal{R})$ is a Markov decision process, $\mathcal{M}$ is a set of messages, μ is the prior distribution over messages $\mathcal{M}$, ζ is a non-negative real number we call the message priority, and finally, R is the communication rate limit between the source and the agent. The goal of the agent is to maximize the expected weighted sum of the MDP payoff and the receiver's accuracy. An MCG proceeds in the following steps:

1. Message $M \sim \mu$ is sampled from the prior over messages at the source.
2. Based on the selected message M and the history of the MDP episode, the source generates and transmits signal T to the agent, adhering to the rate limit R.
3. The Agent uses a conditional policy $\pi_{|T}$, which takes current state $s \in \mathcal{S}$ and received signal T as input and outputs distributions over MDP actions $\mathcal{P}(\mathcal{A})$, to generate the next action a.
4. After repeating steps 2 and 3, the agents' terminal MDP trajectory Z is given to the receiver as an observation.
5. The receiver uses the terminal MDP trajectory Z to output a distribution over messages $\mathcal{P}(\mathcal{M})$ estimating the message $\hat{M}$.

The objective of the agents is to maximize the expected weighted sum of the MDP reward and the accuracy of the receiver's estimate $\mathbb{E}[\mathcal{R}(Z) + \zeta\mathbb{I}[M = \hat{M}]]$. Optionally, If a suitable distance function exists, instead, the objective can also be adjusted to minimize the difference between the actual message and the guess. A diagram of the structure MCG with rate limit is shown in Figure 4.6.

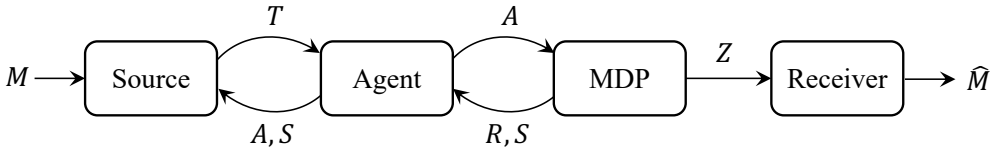


Figure 4.6: The structure of a Markov Coding Game with Rate Limit.

4.5.2 Method Description

Marginal Policy Following [92], before execution, we first derive a marginal policy π for the MDP, based on the Maximum-Entropy objective outlined in Equation (4.33). The value of β in the Maximum-Entropy objective needs to be determined in accordance with the message priority ζ of the MGC. Note that this marginal policy does not depend on the choice of a specific message. By introducing stochasticity into this policy, we can encode information about the message into the selection of actions at each step during runtime

Step 1 - Source: Message Compression At the beginning of each round, given the updated message belief p_M, the source compresses the message to generate the signal T, adhering to the source-agent rate limit R, by solving (4.16). The source then transmits the signal T to the agent. Subsequently, after observing the action taken by the agent, the source updates the message belief for the next round. Algorithm 4.6 outlines the steps taken by the source.

Algorithm 4.6 Source

Input: marginal policy π, message prior μ, Rate R, initial MDP state s^0
Observe message $m \sim \mu$
initialize message belief $p_M \leftarrow \mu$
initialize state $s \leftarrow s^0$
while Source's turn **do**
    $p_{MT} \leftarrow \texttt{compress}(p_M, R)$
    $t \sim p_{T|M}(m)$ ▷ *In our method, $p_{T|M}$ represent a function.*
    Send t to the agent.
    //repeating agent's actions to update message belief:
    $p_T \leftarrow \sum_{m'} p_{MT}(m', \cdot)$
    $p_{TA} \leftarrow \texttt{min\_ent\_coupling}(p_T, \pi(s))$
    $p_{MA} \leftarrow \sum_{t'} p_{MT}(\cdot, t')\ p_{A|T}(t')$
    $a, s \leftarrow$ **Observe** action and next state from MDP.
    $p_M \leftarrow p_{M|A}(a)$

Step 2 - Agent: Minimum Entropy Coupling As illustrated in Algorithm 4.7, at each round, upon receiving the signal T, the agent constructs a conditional policy $\pi_{|T}$ by performing minimum entropy coupling between the action distribution from the marginal policy $\pi(s)$ with the signal distribution p_T. Subsequently, the next action is sampled from the conditional policy, $a \sim \pi_{|T}$. Finally, the agent updates the message belief based on the chosen action.

Receiver: Decoding the Message Given the agent's MDP trajectory, the receiver mirrors the actions of the source and agent to update the message belief at each step. As outlined in Algorithm 4.8, the process begins with the receiver compressing the message based on the current message belief. This is followed by performing minimum entropy coupling between the marginal policy and the distribution of the compressed message.

Algorithm 4.7 Agent

Input: marginal policy π, message prior μ, Rate R, initial MDP state s^0
initialize message belief $p_M \leftarrow \mu$
initialize state $s \leftarrow s^0$
while Agent's turn **do**
    $p_{MT} \leftarrow \texttt{compress}(p_M, R)$
    Receive t from the source.
    $p_T \leftarrow \sum_{m'} p_{MT}(m', \cdot)$
    $p_{TA} \leftarrow \texttt{min\_ent\_coupling}(p_T, \pi(s))$
    $\pi_{|T} \leftarrow p_{A|T}(t)$
    $a \sim \pi_{|T}$
    $s \leftarrow$ commit action a to MDP.
    $p_{MA} \leftarrow \sum_{t'} p_{MT}(\cdot, t')\ p_{A|T}(t')$
    $p_M \leftarrow p_{M|A}(a)$

Algorithm 4.8 Receiver

Input: MDP trajectory z, marginal policy π, message prior μ, Rate R, initial MDP state s^0
initialize message belief $p_M \leftarrow \mu$
initialize state $s \leftarrow s^0$
for $s, a \in z$ **do**
    $p_{MT} \leftarrow \texttt{compress}(p_M, R)$
    $p_T \leftarrow \sum_{m'} p_{MT}(m', \cdot)$
    $p_{TA} \leftarrow \texttt{min\_ent\_coupling}(p_T, \pi(s))$
    $p_{MA} \leftarrow \sum_{t'} p_{MT}(\cdot, t')\ p_{A|T}(t')$
    $p_M \leftarrow p_{M|A}(a)$
estimated message $\hat{m} \leftarrow \arg\max_{m'} p_M(m')$

4.6 Experiments

4.6.1 Minimum Entropy Coupling with Bottleneck

As discussed in Section 4.1, optimizing the encoder and decoder separately for the Minimum Entropy Coupling with Bottleneck (MEC-B) problem, as outlined in (4.3), involves first designing the encoder by solving the Entropy-Bounded Information Maximization (EBIM) in (4.16). This is followed by optimizing the decoder using Minimum Entropy Coupling (MEC) between the code distribution (derived from the previous step) with the output distribution.

To illustrate the couplings generated, we apply the MEC-B framework to inputs and outputs that are uniformly distributed across an alphabet of size 30. For EBIM, we only search for deterministic mappings using Algorithm 4.3, while for MEC, we employ the max-seeking method outlined in Algorithm 4.1.

Figure 4.7 illustrates the generated couplings for varying encoder compression rates, defined by the ratio of the entropy of the input $H(X)$ to the allowed code budget $H(T)$. Greater compression rates are observed to lead to larger entropy couplings; moving from completely deterministic mappings to increasingly stochastic ones.

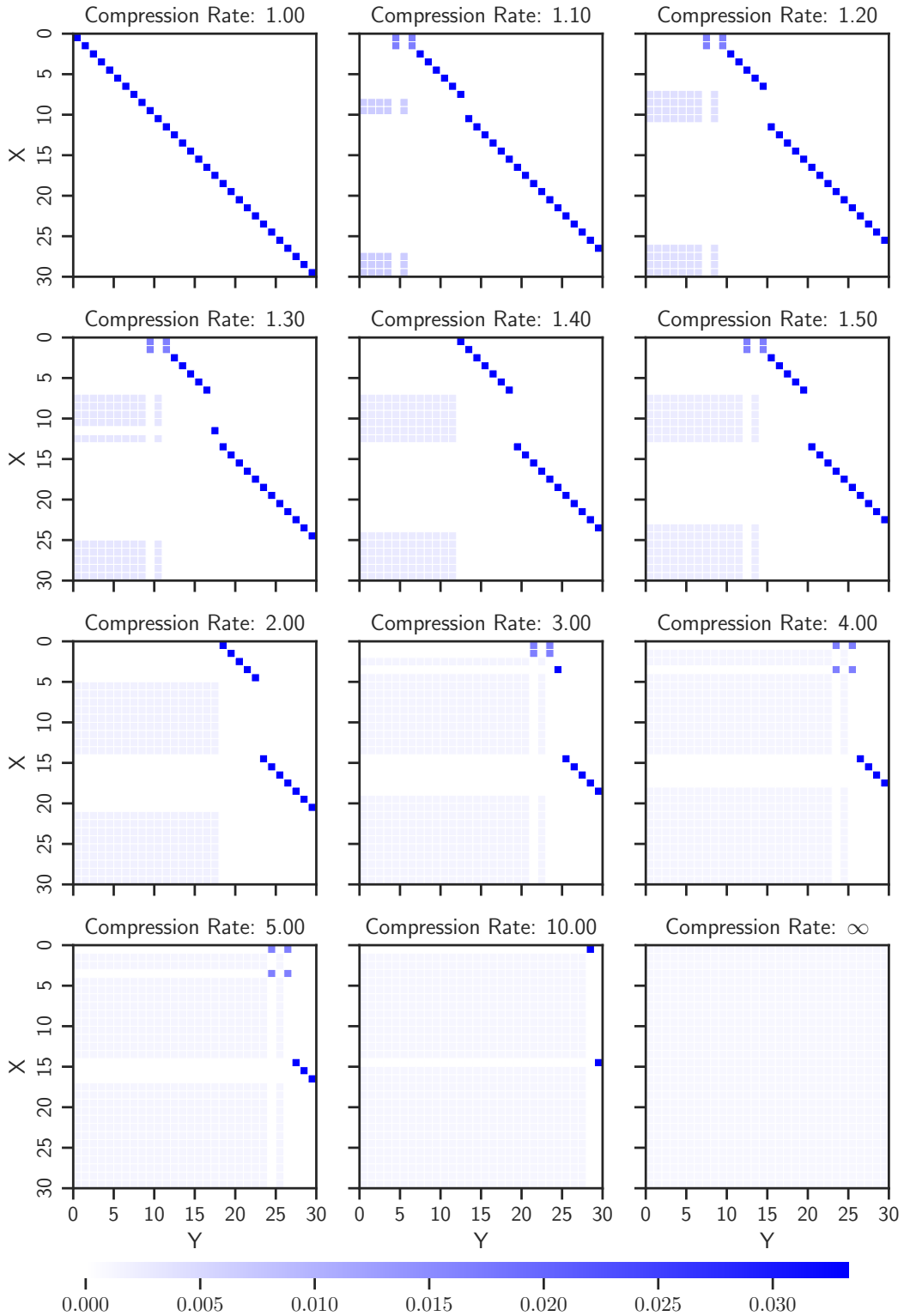


Figure 4.7: Generated couplings in MEC-B formulation (4.3), for uniform input and output distributions. The compression rate is defined as $H(X)/R$. Higher compression rates lead to more stochastic couplings with increased entropy.

4.6.2 Markov Coding Games with Rate Limits

This section presents the experimental results of the method described in Section 4.5.2, applied to Markov Coding Games. For our experiments, we utilize a simple environment known as *Grid World*, for the Markov Decision Process. In this setup, the agent is placed on an 8×8 grid and, at each step, can move left, right, up, and down. The primary objective for the agent is to navigate from the starting cell to the goal cell to receive a reward of 1, while avoiding a trap cell with a reward of -1. Also, the environment is noisy; even if the agent decides to move in a specific direction, the environment might, with a certain noise probability, force a move in a direction 90° off the intended path. The rewards received are discounted by a factor of 0.95. Finally, the receiver has to decode a message, uniformly chosen from an alphabet of size 1024, given the final trajectory of the agent. Figure 4.8 illustrates the Grid World used in this experiment and depicts a trajectory taken by the agent.

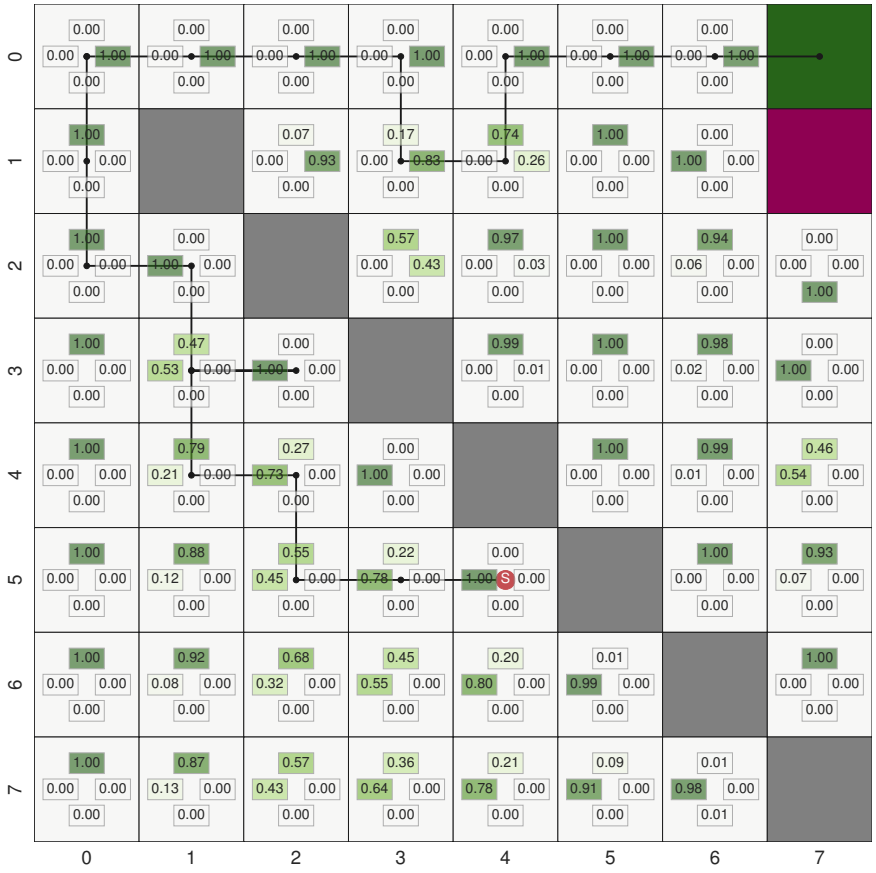


Figure 4.8: The Grid World Setup used in the experiments. The starting cell is depicted by a red circle, while the goal, trap, and obstacle cells are colored green, red, and grey, respectively. Additionally, a non-deterministic policy is demonstrated through the probabilities of actions in each direction within each cell. The path taken by the agent is traced in black. Note that due to the noisy environment, the agent may move in directions not explicitly suggested by the policy.

The marginal policy is learned through Soft Q-Value iteration, as described in Algorithm 4.5. By increasing the value of β in Equation (4.33), we induce more randomness into the marginal policy. Consequently, higher values of β lead to an increase in the total number of steps taken by the agent to

reach the goal, resulting in a more heavily discounted reward. Conversely, as the entropy of actions at each state is increased, there is an increase in the mutual information between the actions and the compressed message during the minimum entropy coupling at each step. This dynamic establishes a fundamental trade-off between the MDP reward and the accuracy of the message decoded by the receiver, through the adjustment of β. Figure 4.9 shows two policies learned by high and low values of β.

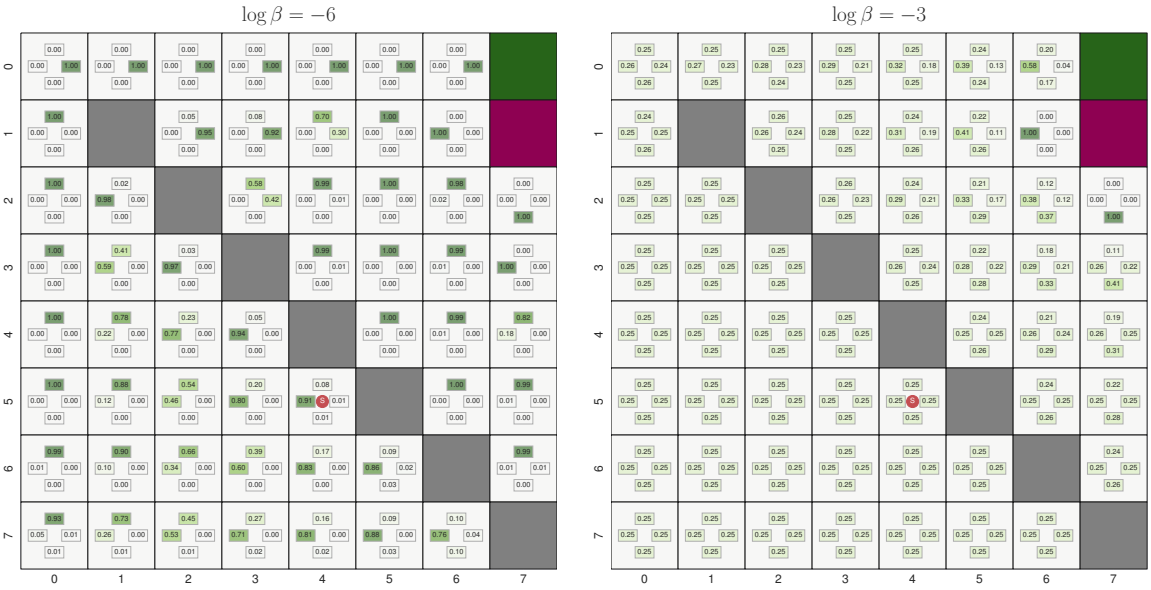


Figure 4.9: The Maximum Entropy policy learned through Soft Q-Value iteration of Algorithm 4.5, for $\log \beta = -6$ (left) and $\log \beta = -3$ (right).

We compare our proposed compression algorithm in Algorithm 4.3 with a baseline of uniform quantization. As detailed in Algorithm 4.9, given an entropy budget R, the input symbols are uniformly partitioned into $\lfloor 2^R \rfloor$ groups, and each group is encoded into the same code.

Algorithm 4.9 Uniform Quantizer Encoder

Input: p_X, R
Output: p_{XT}

$n \leftarrow$ length of p_X
$m \leftarrow \lfloor 2^R \rfloor$
partition_size $\leftarrow \lceil n/m \rceil$
Initialize p_{XT} as an $n \times m$ zero matrix
for $i \leftarrow 0$ **to** $m - 1$ **do**
    $start \leftarrow i \times$ partition_size
    $end \leftarrow \min(start + \text{partition\_size}, n)$
    $p_{XT}[start : end, i] \leftarrow p_X[start : end]$
return p_{XT}

Figure 4.10 illustrates the trade-off between MDP reward and the receiver's accuracy for different values of β, using our deterministic mapping search algorithm for EBIM in Algorithm 4.3 and the uniform quantization encoder in Algorithm 4.9. Here, the compression rate is defined by the ratio of the entropy of the message to the allowed code budget $H(T)$.

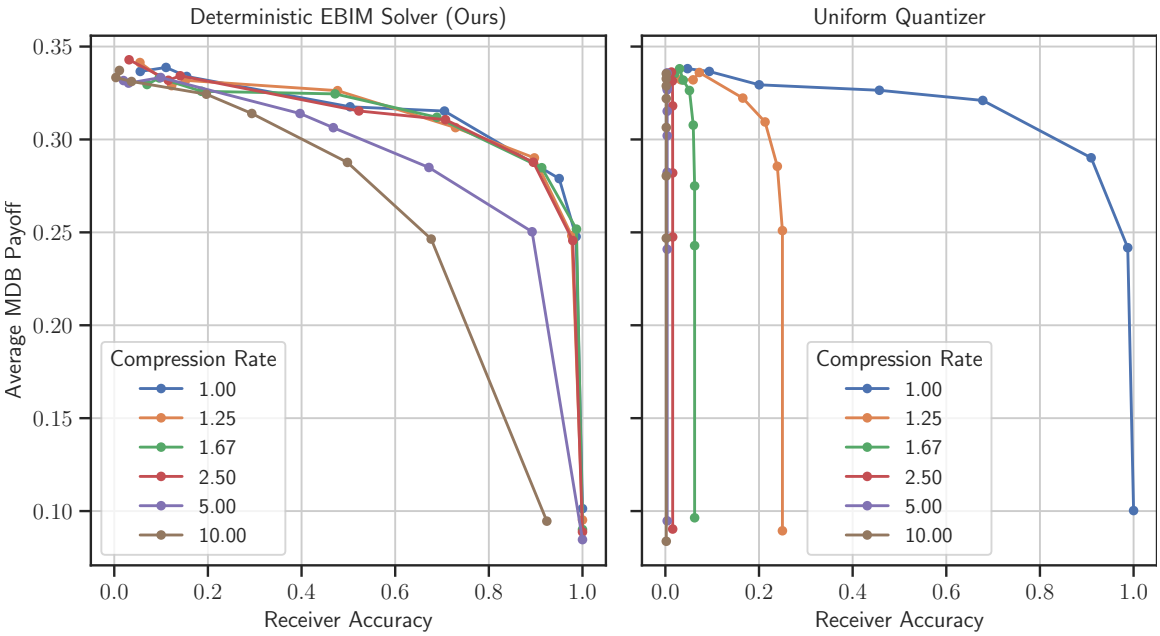


Figure 4.10: The trade-off between the MDP reward vs. receiver's accuracy navigated for different values of β. Left: using our search algorithm for compression (Algorithm 4.3), Right: using uniform quantization in Algorithm 4.9.

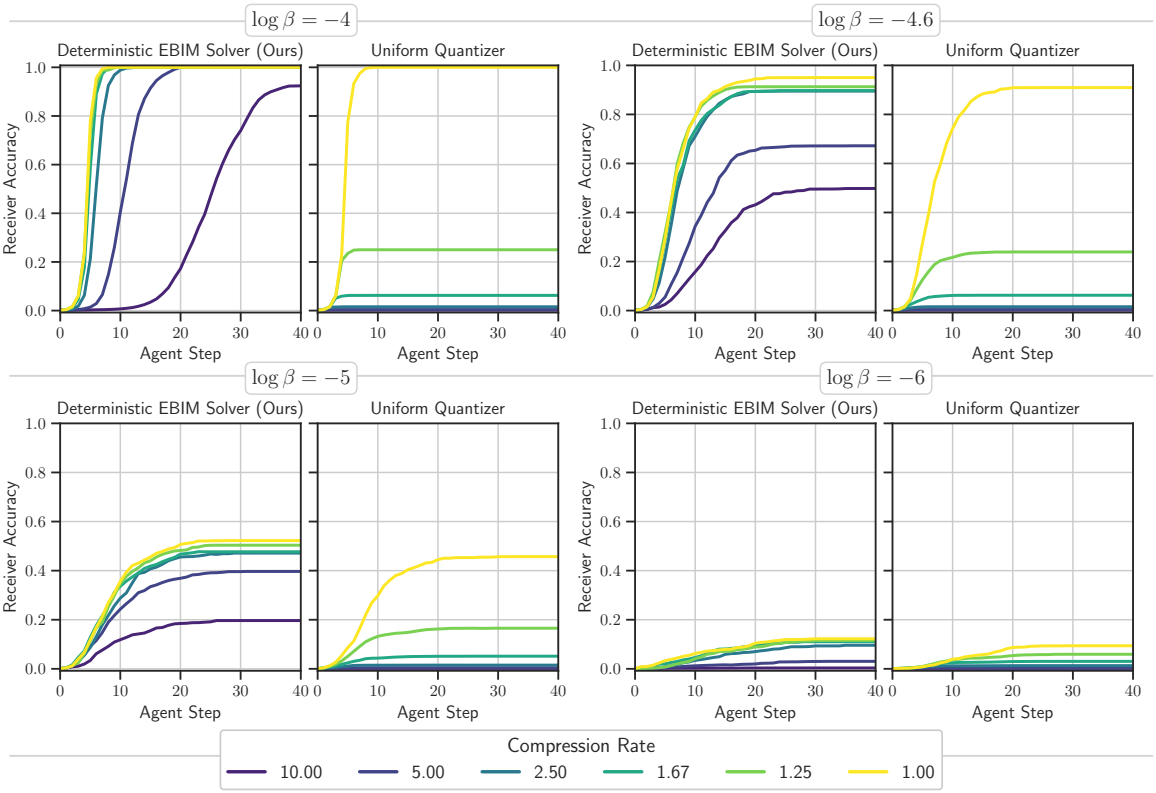


Figure 4.11: Evolution of message belief over time, for various values of β and rate budget, using our search algorithm for compression in Algorithm 4.3 vs. uniform quantization in Algorithm 4.9.

Figure 4.11 illustrates the evolution of message belief over time for various values of β and rate budgets. A marginal policy optimized with a higher β prioritizes message accuracy over MDP payoff, as higher entropy of actions at each state provides more room for the agent to encode information about the message. Consequently, as observed, this leads to improved receiver accuracy in fewer steps. In addition, a lower compression rate permits the agent to retain more information about the message, enabling more effective encoding of information in the selected trajectory.

4.7 Summary and Concluding Remarks

We investigated a lossy compression framework under logarithmic loss, where the output distribution differs from the source distribution. This framework supports joint compression and retrieval applications such as super-resolution from compressed data, or more generally, cases where distributional shifts occur due to processing. We demonstrated that this framework effectively extends the classical minimum entropy coupling by incorporating a bottleneck, which regulates the degree of stochasticity or determinism in the coupling.

We demonstrated that separately optimizing the encoder and decoder decomposes the Minimum Entropy Coupling with Bottleneck (MEC-B) into two distinct problems: Entropy-Bounded Information Maximization (EBIM) for the encoder, followed by Minimum Entropy Coupling (MEC) for the decoder. We conducted an extensive study of the EBIM problem, provided a functional mapping search algorithm with guaranteed performance, and characterized the optimal solution adjacent to functional mappings, offering valuable theoretical insights into the problem structure.

To illustrate an application of MEC-B, the chapter presented experiments on Markov Coding Games (MCGs) with rate limits. MCGs involve a sender, an agent, and a receiver, where the agent aims to convey a message from the sender to the receiver through actions within a Markov Decision Process (MDP). The chapter focused on a rate-limited MCG where the agent receives a compressed version of the message iteratively and encodes information about the message into the MDP trajectory for the receiver. Experiments were conducted using GridWorld environment as the MDP. The results demonstrated the trade-off between MDP reward and receiver accuracy, with varying compression rates, compared to baseline compression schemes.

Chapter 5

Conclusion and Future Directions

This thesis explored three crucial dimensions of machine learning: *modeling, training, and theory*. Each dimension was addressed through dedicated projects, contributing to the advancement of machine learning methodologies and applications across diverse domains.

Chapter 2, focusing on the theme of data modeling, introduced Shared Gaussian Process Factor Analysis (S-GPFA) as a novel probabilistic model for analyzing fMRI data. S-GPFA effectively captures the shared temporal dynamics and spatial organization of brain activity across individuals in multi-subject fMRI datasets, enabling researchers to uncover common neural patterns while accounting for individual differences. By incorporating Gaussian Process priors to model temporal correlations within fMRI data, S-GPFA provides a more interpretable representation of brain activity compared to traditional static methods. The model's ability to simultaneously perform functional aggregation, dimensionality reduction, and dynamical modeling makes it a powerful tool for analyzing multi-subject fMRI datasets.

Chapter 3 addressed the theme of distributed training and the challenge of mitigating the impact of stragglers on the training process. The chapter introduced two novel coding schemes, Selective Reattempt Sequential Gradient Coding (SR-SGC) and Multiplexed Sequential Gradient Coding (M-SGC), that leverage coding across both spatial and temporal dimensions to achieve straggler resilience with reduced computational load. These schemes exploit the temporal diversity of straggler behavior, where periods of high straggler activity are often followed by periods of low straggler activity, to trade off computation and training time. Experiments on a large-scale AWS Lambda cluster demonstrated the effectiveness of M-SGC in significantly reducing runtime and improving training performance compared to other schemes.

Chapter 4 investigated the information theoretical foundations of coupling and compression, aligning with the theme of theory. The chapter introduced Minimum Entropy Coupling with Bottleneck (MEC-B) as a framework for lossy data compression under logarithmic loss. MEC-B extends the classical Minimum Entropy Coupling (MEC) framework by incorporating a bottleneck that regulates the degree of stochasticity or determinism in the coupling. The chapter also introduced the Entropy-Bounded Information Maximization (EBIM) formulation for compression and proposed a novel search algorithm for identifying deterministic mappings with guaranteed performance bounds. Experiments on Markov Coding Games with rate limits illustrated the practical application of MEC-B.

5.1 Future Directions

5.1.1 Group Temporal Dynamics Analysis in fMRI

While S-GPFA offers a promising approach for analyzing fMRI data, there are several avenues for future research to enhance its capabilities and broaden its applications. One limitation of S-GPFA is its scalability with respect to the number of time samples in fMRI recordings. As fMRI datasets continue to grow in size and complexity, it becomes increasingly important to develop methods that can efficiently handle large-scale data. One potential solution is to explore techniques for dividing long recordings into smaller chunks and feeding them as multiple samples to a single S-GPFA model. This approach could improve the model's computational efficiency and enable it to handle longer and more complex fMRI time series.

Another direction for future research is to incorporate Bayesian inference techniques into the S-GPFA framework. By adopting approximate Bayesian inference methods, it would be possible to draw uncertainty estimates for model parameters, such as subject-specific topographies and shared timescales. This would provide valuable insights into the reliability and variability of the model's estimates, enhancing the interpretability of the results and allowing researchers to better understand the confidence intervals associated with the identified neural patterns.

Furthermore, exploring the use of change-point and non-stationary kernels could improve S-GPFA's ability to capture complex temporal dynamics in fMRI data. Change-point kernels could help identify abrupt changes or transitions in brain activity patterns, while non-stationary kernels could account for gradual changes or trends over time. These advancements could provide a more nuanced and accurate representation of the dynamic nature of brain activity, potentially leading to new discoveries about the neural mechanisms underlying cognitive processes and psychiatric disorders.

5.1.2 Sequential Gradient Coding

Future research on sequential gradient coding could address the inherent privacy and security risks associated with data sharing in distributed computing frameworks. As distributed training involves sharing data and model updates across multiple servers, developing methods that protect sensitive information and ensure data security is crucial. One promising direction is to explore the integration of privacy-preserving techniques and private gradient computation into the proposed coding schemes. Furthermore, incorporating gradient compression methods into the experimental setup could address communication constraints and improve the efficiency of distributed training. Gradient compression techniques, such as quantization or sparsification, can significantly reduce the amount of data that needs to be transmitted between servers, thereby alleviating communication bottlenecks and improving training speed. Integrating these techniques with SR-SGC and M-SGC could lead to significant practical improvements in the scalability and efficiency of distributed training for large-scale machine learning models.

5.1.3 Minimum Entropy Coupling with Bottleneck

Several avenues for future research could further enhance the capabilities and applications of the proposed solution for Minimum Entropy Coupling with Bottleneck (MEC-B). One important direction is to quantify the gap between the separate optimization of the encoder and decoder, as

presented in this thesis, and the optimal joint setting. Understanding the gap between separate and joint optimization could guide the development of methods that jointly optimize both components, potentially improving the overall performance of MEC-B and achieving higher fidelity with more efficient compression.

Another area for future exploration is enabling fine-grained control over the entropy spread in the coupling. As evident in Figure 4.10, the current solution for MEC-B controls the entropy of the coupling by mapping parts of the input to the output deterministically, while leaving others completely stochastic. Developing methods that allow for more flexible control over the spread of entropy within the coupling could expand the applicability of MEC-B to a wider range of problems.

Additionally, the application of Entropy-Bounded Information Maximization (EBIM) in watermarking language models, as demonstrated in [107], suggests a valuable intersection with state-of-the-art AI applications.

Moreover, extending this framework to continuous cases could lead to the design of neural network architectures based on the proposed model and provide information-theoretic insights into a broad spectrum of deep learning problems. These include unpaired sample-to-sample translation [108–110], joint compression and upscaling [101, 111], and the InfoMax framework [112, 113], among others.

A significant challenge in the continuous case is the issue of distribution permutations: within the current framework, solutions can be achieved by maximizing the mutual information between bijective functions of input and output. A straightforward example of this is swapping the channels of images in the input and output while still maintaining maximal mutual information. Proposing strategies to mitigate this issue is another potential direction for future research in the continuous domain.

Bibliography

[1] M. Ebrahimi, N. Calarco, C. Hawco, A. Voineskos, and A. Khisti, "Time-resolved fmri shared response model using gaussian process factor analysis," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2023, pp. 1–5.

[2] N. K. M. Krishnan, M. Ebrahimi, and A. J. Khisti, "Sequential gradient coding for straggler mitigation," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=-lGvSmht7a.

[3] R. Kanai and G. Rees, "The structural basis of inter-individual differences in human behaviour and cognition," *Nature Reviews Neuroscience*, vol. 12, no. 4, pp. 231–242, 2011.

[4] S. Mueller *et al.*, "Individual variability in functional connectivity architecture of the human brain," *Neuron*, vol. 77, no. 3, pp. 586–595, 2013.

[5] E. N. Davison *et al.*, "Individual differences in dynamic functional brain connectivity across the human lifespan," *PLoS computational biology*, vol. 12, no. 11, 2016.

[6] T. T. Liu, "Noise contributions to the fmri signal: An overview," *NeuroImage*, vol. 143, pp. 141–151, 2016.

[7] P.-H. C. Chen, J. Chen, Y. Yeshurun, U. Hasson, J. Haxby, and P. J. Ramadge, "A reduced-dimension fmri shared response model," in *Advances in Neural Information Processing Systems*, 2015, pp. 460–468.

[8] J. R. Manning *et al.*, "Hierarchical topographic factor analysis," in *2014 International Workshop on Pattern Recognition in Neuroimaging*, IEEE, 2014, pp. 1–4.

[9] S. A. S. S. Ajmera, S. Rajagopal, R. Rehman, and D. Sridharan, "Infra-slow brain dynamics as a marker for cognitive function and decline," in *Advances in Neural Information Processing Systems*, 2019, pp. 6947–6958.

[10] M. Y. Byron, J. P. Cunningham, G. Santhanam, S. I. Ryu, K. V. Shenoy, and M. Sahani, "Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity," in *Advances in neural information processing systems*, 2009, pp. 1881–1888.

[11] L. Li, D. Pluta, B. Shahbaba, N. Fortin, H. Ombao, and P. Baldi, "Modeling dynamic functional connectivity with latent factor gaussian processes," in *Advances in Neural Information Processing Systems*, 2019, pp. 8261–8271.

[12] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.

[13] J. S. Turek, C. T. Ellis, L. J. Skalaban, N. B. Turk-Browne, and T. L. Willke, "Capturing shared and individual information in fmri data," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2018, pp. 826–830.

[14] F. R. Bach and M. I. Jordan, "A probabilistic interpretation of canonical correlation analysis," 2005.

[15] A. S. Lukic, M. N. Wernick, L. K. Hansen, J. Anderson, and S. C. Strother, "A spatially robust ica algorithm for multiple fmri data sets," in *Proceedings IEEE International Symposium on Biomedical Imaging*, IEEE, 2002, pp. 839–842.

[16] D. Clark, J. Livezey, and K. Bouchard, "Unsupervised discovery of temporal structure in noisy data with dynamical components analysis," in *Advances in Neural Information Processing Systems*, 2019, pp. 14 267–14 278.

[17] M. Shvartsman, N. Sundaram, M. Aoi, A. Charles, T. Willke, and J. Cohen, "Matrix-normal models for fmri analysis," A. Storkey and F. Perez-Cruz, Eds., ser. Proceedings of Machine Learning Research, vol. 84, Playa Blanca, Lanzarote, Canary Islands: PMLR, Sep. 2018, pp. 1914–1923. [Online]. Available: http://proceedings.mlr.press/v84/shvartsman18a.html.

[18] P. Smyth, "Cross-validated likelihood for model selection in unsupervised learning," in *Sixth International Workshop on Artificial Intelligence and Statistics*, PMLR, 1997, pp. 473–480.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization. iclr (2015)," *arXiv preprint arXiv:1412.6980*, vol. 9, 2015.

[20] C. Agostinelli and L. Greco, "Weighted likelihood in bayesian inference," 2012.

[21] M. J. Anderson *et al.*, "Enabling factor analysis on thousand-subject neuroimaging datasets," in *2016 IEEE International Conference on Big Data (Big Data)*, IEEE, 2016, pp. 1151–1160.

[22] J. V. Haxby *et al.*, "A common, high-dimensional model of the representational space in human ventral temporal cortex," *Neuron*, vol. 72, no. 2, pp. 404–416, 2011.

[23] J. Chen, Y. C. Leong, C. J. Honey, C. H. Yong, K. A. Norman, and U. Hasson, "Shared memories reveal shared structure in neural activity across individuals," *Nature neuroscience*, vol. 20, no. 1, pp. 115–125, 2017.

[24] C. Hawco *et al.*, "Separable and replicable neural strategies during social brain function in people with and without severe mental illness," *American Journal of Psychiatry*, vol. 176, no. 7, pp. 521–530, 2019.

[25] A. Schaefer *et al.*, "Local-global parcellation of the human cerebral cortex from intrinsic functional connectivity mri," *Cerebral Cortex*, vol. 28, no. 9, pp. 3095–3114, 2018.

[26] O. Esteban *et al.*, *fMRIPrep: a robust preprocessing pipeline for functional MRI*, version 20.1.1, Jun. 2020. DOI: 10.5281/zenodo.3876458. [Online]. Available: https://doi.org/10.5281/zenodo.3876458.

[27] B. Thomas Yeo *et al.*, "The organization of the human cerebral cortex estimated by intrinsic functional connectivity," *Journal of neurophysiology*, vol. 106, no. 3, pp. 1125–1165, 2011.

[28] M. Iacoboni and M. Dapretto, "The mirror neuron system and the consequences of its dysfunction," *Nature Reviews Neuroscience*, vol. 7, no. 12, p. 942, 2006.

[29] T. R. Insel, "The nimh research domain criteria (rdoc) project: Precision medicine for psychiatry," *American Journal of Psychiatry*, vol. 171, no. 4, pp. 395–397, 2014.

[30] A. Olsen, J. Ferenc Brunner, K. A. I. Evensen, B. Garzon, N. I. Landrø, and A. K. Håberg, "The functional topography and temporal dynamics of overlapping and distinct brain activations for adaptive task control and stable task-set maintenance during performance of an fmri-adapted clinical continuous performance test," *Journal of cognitive neuroscience*, vol. 25, no. 6, pp. 903–919, 2013.

[31] A. I. Ramos-Nuñez, S. Fischer-Baum, R. C. Martin, Q. Yue, F. Ye, and M. W. Deem, "Static and dynamic measures of human brain connectivity predict complementary aspects of human cognitive performance," *Frontiers in human neuroscience*, vol. 11, p. 420, 2017.

[32] R. Tandon, Q. Lei, A. G. Dimakis, and N. Karampatziakis, "Gradient Coding: Avoiding Stragglers in Distributed Learning," in *Proc. International Conference on Machine Learning, ICML*, vol. 70, PMLR, 2017, pp. 3368–3376.

[33] C. Yang, R. Pedarsani, and A. S. Avestimehr, "Timely Coded Computing," in *Proc. IEEE International Symposium on Information Theory, ISIT*, IEEE, 2019, pp. 2798–2802.

[34] K. Lee, M. Lam, R. Pedarsani, D. S. Papailiopoulos, and K. Ramchandran, "Speeding Up Distributed Machine Learning Using Codes," *IEEE Trans. Inf. Theory*, vol. 64, no. 3, pp. 1514–1529, 2018.

[35] S. Li and S. Avestimehr, "Coded Computing," *Found. Trends Commun. Inf. Theory*, vol. 17, no. 1, pp. 1–148, 2020.

[36] J. Chen, X. Pan, R. Monga, S. Bengio, and R. Jozefowicz, "Revisiting distributed synchronous sgd," *arXiv preprint arXiv:1604.00981*, 2016.

[37] O. Devolder, F. Glineur, and Y. Nesterov, "First-order methods of smooth convex optimization with inexact oracle," *Mathematical Programming*, vol. 146, pp. 37–75, 2014.

[38] X. Su, B. Sukhnandan, and J. Li, "On arbitrary ignorance of stragglers with gradient coding," in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, IEEE, 2023, pp. 660–670.

[39] M. N. Krishnan, E. Hosseini, and A. Khisti, "Sequential Gradient Coding for Packet-Loss Networks," *IEEE J. Sel. Areas Inf. Theory*, vol. 2, no. 3, pp. 919–930, 2021.

[40] M. Ye and E. Abbe, "Communication-Computation Efficient Gradient Coding," in *Proc. International Conference on Machine Learning, ICML*, vol. 80, PMLR, 2018, pp. 5606–5615.

[41] S. Kadhe, O. O. Koyluoglu, and K. Ramchandran, "Communication-Efficient Gradient Coding for Straggler Mitigation in Distributed Learning," in *Proc. IEEE International Symposium on Information Theory, ISIT*, IEEE, 2020, pp. 2634–2639.

[42] N. Raviv, I. Tamo, R. Tandon, and A. G. Dimakis, "Gradient Coding From Cyclic MDS Codes and Expander Graphs," *IEEE Trans. Inf. Theory*, vol. 66, no. 12, pp. 7475–7489, 2020.

[43] W. Halbawi, N. A. Ruhi, F. Salehi, and B. Hassibi, "Improving Distributed Gradient Descent Using Reed-Solomon Codes," in *Proc. IEEE International Symposium on Information Theory, ISIT*, IEEE, 2018, pp. 2027–2031.

[44] H. Wang, S. Guo, B. Tang, R. Li, and C. Li, "Heterogeneity-aware Gradient Coding for Straggler Tolerance," in *Proc. IEEE International Conference on Distributed Computing Systems, ICDCS*, IEEE, 2019, pp. 555–564.

[45] H. Wang, Z. B. Charles, and D. S. Papailiopoulos, "ErasureHead: Distributed Gradient Descent without Delays Using Approximate Gradient Coding," *CoRR*, vol. abs/1901.09671, 2019.

[46] R. K. Maity, A. S. Rawat, and A. Mazumdar, "Robust Gradient Descent via Moment Encoding and LDPC Codes," in *IEEE International Symposium on Information Theory, ISIT*, IEEE, 2019, pp. 2734–2738.

[47] Q. Yu, M. A. Maddah-Ali, and S. Avestimehr, "Polynomial Codes: an Optimal Design for High-Dimensional Coded Matrix Multiplication," in *Proc. Annual Conference on Neural Information Processing Systems*, 2017, pp. 4403–4413.

[48] A. Ramamoorthy, L. Tang, and P. O. Vontobel, "Universally Decodable Matrices for Distributed Matrix-Vector Multiplication," in *Proc. IEEE International Symposium on Information Theory, ISIT*, IEEE, 2019, pp. 1777–1781.

[49] A. M. Subramaniam, A. Heidarzadeh, and K. R. Narayanan, "Random Khatri-Rao-Product Codes for NumericallyStable Distributed Matrix Multiplication," in *Proc. Allerton Conf. on Comm., Contr., and Comp.*, 2019, pp. 253–259.

[50] Q. Yu, S. Li, N. Raviv, S. M. M. Kalan, M. Soltanolkotabi, and A. S. Avestimehr, "Lagrange Coded Computing: Optimal Design for Resiliency, Security, and Privacy," in *Proc. International Conference on Artificial Intelligence and Statistics, AISTATS*, vol. 89, PMLR, 2019, pp. 1215–1225.

[51] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. R. Cadambe, and P. Grover, "On the Optimal Recovery Threshold of Coded Matrix Multiplication," *IEEE Trans. Inf. Theory*, vol. 66, no. 1, pp. 278–301, 2020.

[52] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Straggler Mitigation in Distributed Matrix Multiplication: Fundamental Limits and Optimal Coding," *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1920–1933, 2020.

[53] A. Ramamoorthy, A. B. Das, and L. Tang, "Straggler-resistant distributed matrix computation via coding theory," *CoRR*, vol. abs/2002.03515, 2020.

[54] M. N. Krishnan, S. Hosseini, and A. Khisti, "Coded Sequential Matrix Multiplication For Straggler Mitigation," in *Proc. Annual Conference on Neural Information Processing Systems*, 2020.

[55] G. Hasslinger and O. Hohlfeld, "The gilbert-elliott model for packet loss in real time services on the internet," in *14th GI/ITG Conference - Measurement, Modelling and Evalutation of Computer and Communication Systems*, 2008, pp. 1–15.

[56] J. Dean and L. A. Barroso, "The tail at scale," *Commun. ACM*, vol. 56, no. 2, pp. 74–80, 2013.

[57] J. Bolot, "End-to-End Packet Delay and Loss Behavior in the Internet," in *Proc. ACM SIGCOMM Conference on Communications Architectures, Protocols and Applications*, ACM, 1993, pp. 289–298.

[58] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for Hyper-Parameter Optimization," in *Proc. Annual Conference on Neural Information Processing Systems*, 2011.

[59] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[60] N. Bhuyan, S. Moharir, and G. Joshi, "Multi-Model Federated Learning with Provable Guarantees," *CoRR*, vol. abs/2207.04330, 2022.

[61] R. V. da Silva, J. Choi, J. Park, G. Brante, and R. D. Souza, "Multichannel ALOHA Optimization for Federated Learning With Multiple Models," *IEEE Wirel. Commun. Lett.*, vol. 11, no. 10, pp. 2180–2184, 2022.

[62] Y. Wu, L. Liu, and R. Kompella, "Parallel detection for efficient video analytics at the edge," in *Proc. IEEE International Conference on Cognitive Machine Intelligence, CogMI*, IEEE, 2021, pp. 1–10.

[63] G. Forney, "Burst-Correcting Codes for the Classic Bursty Channel," *IEEE Trans. Commun. Tech.*, vol. 19, no. 5, pp. 772–781, 1971.

[64] A. Saberi, F. Farokhi, and G. N. Nair, "State Estimation via Worst-Case Erasure and Symmetric Channels with Memory," in *Proc. IEEE International Symposium on Information Theory, ISIT*, IEEE, 2019, pp. 3072–3076.

[65] E. Martinian and C. W. Sundberg, "Burst erasure correction codes with low decoding delay," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2494–2502, 2004.

[66] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[67] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[68] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[69] T. A. Courtade and R. D. Wesel, "Multiterminal source coding with an entropy-based distortion measure," in *2011 IEEE International Symposium on Information Theory Proceedings*, IEEE, 2011, pp. 2040–2044.

[70] T. A. Courtade and T. Weissman, "Multiterminal source coding under logarithmic loss," *IEEE Transactions on Information Theory*, vol. 60, no. 1, pp. 740–761, 2013.

[71] Y. Y. Shkel and S. Verdú, "A single-shot approach to lossy source coding under logarithmic loss," *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 129–147, 2017.

[72] M. Vidyasagar, "A metric between probability distributions on finite sets of different cardinalities and applications to order reduction," *IEEE Transactions on Automatic Control*, vol. 57, no. 10, pp. 2464–2477, 2012.

[73] A. Painsky, S. Rosset, and M. Feder, "Memoryless representation of markov processes," in *2013 IEEE International Symposium on Information Theory*, IEEE, 2013, pp. 2294–298.

[74] M. Kovačević, I. Stanojević, and V. Šenk, "On the entropy of couplings," *Information and Computation*, vol. 242, pp. 369–382, 2015.

[75] F. Cicalese, L. Gargano, and U. Vaccaro, "How to find a joint probability distribution of minimum entropy (almost) given the marginals," in *2017 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2017, pp. 2173–2177.

[76] M. Fréchet, "Sur les tableaux de corrélation dont les marges sont données," *Ann. Univ. Lyon, 3ˆ e serie, Sciences, Sect. A*, vol. 14, pp. 53–77, 1951.

[77] F. Den Hollander, "Probability theory: The coupling method," *Lecture notes available online (http://websites. math. leidenuniv. nl/probability/lecturenotes/CouplingLectures. pdf)*, 2012.

[78] G. D. Lin, X. Dou, S. Kuriki, and J.-S. Huang, "Recent developments on the construction of bivariate distributions with fixed marginals," *Journal of Statistical Distributions and Applications*, vol. 1, pp. 1–23, 2014.

[79] V. Benes and J. Stepán, *Distributions with given marginals and moment problems*. Springer Science & Business Media, 2012.

[80] L. Yu and V. Y. Tan, "Asymptotic coupling and its applications in information theory," *IEEE Transactions on Information Theory*, vol. 65, no. 3, pp. 1321–1344, 2018.

[81] C. Villani *et al.*, *Optimal transport: old and new*. Springer, 2009, vol. 338.

[82] M. Cuturi, "Sinkhorn distances: Lightspeed computation of optimal transport," *Advances in neural information processing systems*, vol. 26, 2013.

[83] P. A. Knight, "The sinkhorn–knopp algorithm: Convergence and applications," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 1, pp. 261–275, 2008.

[84] M. Kocaoglu, A. Dimakis, S. Vishwanath, and B. Hassibi, "Entropic causal inference," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

[85] S. Compton, "A tighter approximation guarantee for greedy minimum entropy coupling," in *2022 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2022, pp. 168–173.

[86] A. W. Marshall, I. Olkin, and B. C. Arnold, "Inequalities: Theory of majorization and its applications," 1979.

[87] F. Cicalese, L. Gargano, and U. Vaccaro, "Minimum-entropy couplings and their applications," *IEEE Transactions on Information Theory*, vol. 65, no. 6, pp. 3436–3451, 2019.

[88] C. T. Li, "Efficient approximate minimum entropy coupling of multiple probability distributions," *IEEE Transactions on Information Theory*, vol. 67, no. 8, pp. 5259–5268, 2021.

[89] S. Compton, D. Katz, B. Qi, K. Greenewald, and M. Kocaoglu, "Minimum-entropy coupling approximation guarantees beyond the majorization barrier," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2023, pp. 10 445–10 469.

[90] S. Compton, M. Kocaoglu, K. Greenewald, and D. Katz, "Entropic causal inference: Identifiability and finite sample results," *Advances in Neural Information Processing Systems*, vol. 33, pp. 14 772–14 782, 2020.

[91] M. A. Javidian, V. Aggarwal, F. Bao, and Z. Jacob, "Quantum entropic causal inference," in *Quantum Information and Measurement*, Optica Publishing Group, 2021, F2C–3.

[92] S. Sokota *et al.*, "Communicating via markov decision processes," in *International Conference on Machine Learning*, PMLR, 2022, pp. 20 314–20 328.

[93] C. S. de Witt, S. Sokota, J. Z. Kolter, J. Foerster, and M. Strohmeier, "Perfectly secure steganography using minimum entropy coupling," *arXiv preprint arXiv:2210.14889*, 2022.

[94] F. Cicalese, L. Gargano, and U. Vaccaro, "Approximating probability distributions with short vectors, via information theoretic distance measures," in *2016 IEEE International Symposium on Information Theory (ISIT)*, IEEE, 2016, pp. 1138–1142.

[95] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," *arXiv preprint physics/0004057*, 2000.

[96] N. Slonim and N. Tishby, "Document clustering using word clusters via the information bottleneck method," in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, 2000, pp. 208–215.

[97] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *2015 ieee information theory workshop (itw)*, IEEE, 2015, pp. 1–5.

[98] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," *arXiv preprint arXiv:1612.00410*, 2016.

[99] A. Bhatt, B. Nazer, O. Ordentlich, and Y. Polyanskiy, "Information-distilling quantizers," *IEEE Transactions on Information Theory*, vol. 67, no. 4, pp. 2472–2487, 2021.

[100] D. Strouse and D. J. Schwab, "The deterministic information bottleneck," *Neural computation*, vol. 29, no. 6, pp. 1611–1630, 2017.

[101] H. Liu, G. Zhang, J. Chen, and A. J. Khisti, "Lossy compression with distribution shift as entropy constrained optimal transport," in *International Conference on Learning Representations*, 2021.

[102] O. L. Mangasarian, "Machine learning via polyhedral concave minimization," in *Applied Mathematics and Parallel Computing: Festschrift for Klaus Ritter*, H. Fischer, B. Riedmüller, and S. Schäffler, Eds. Heidelberg: Physica-Verlag HD, 1996, pp. 175–188, ISBN: 978-3-642-99789-1. DOI: 10.1007/978-3-642-99789-1_13. [Online]. Available: https://doi.org/10.1007/978-3-642-99789-1_13.

[103] M. X. Goemans, *Spectral graph theory and numerical linear algebra*. [Online]. Available: http://www.cs.cmu.edu/afs/cs/user/glmiller/public/Scientific-Computing/F-11/RelatedWork/Goemans-LP-notes.pdf.

[104] F. Palacios-Gomez, L. Lasdon, and M. Engquist, "Nonlinear optimization by successive linear programming," *Management science*, vol. 28, no. 10, pp. 1106–1120, 1982.

[105] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey, *et al.*, "Maximum entropy inverse reinforcement learning.," in *Aaai*, Chicago, IL, USA, vol. 8, 2008, pp. 1433–1438.

[106] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[107] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein, "A watermark for large language models," in *International Conference on Machine Learning*, PMLR, 2023, pp. 17 061–17 084.

[108] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks. arxiv e-prints," *arXiv preprint arXiv:1611.07004*, vol. 1611, 2016.

[109] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[110] J. Hoffman *et al.*, "Cycada: Cycle-consistent adversarial domain adaptation," in *International conference on machine learning*, Pmlr, 2018, pp. 1989–1998.

[111] B. Kang, S. Tripathi, and T. Q. Nguyen, "Toward joint image generation and compression using generative adversarial networks," *arXiv preprint arXiv:1901.07838*, 2019.

[112] M. Tschannen, J. Djolonga, P. K. Rubenstein, S. Gelly, and M. Lucic, "On mutual information maximization for representation learning," *arXiv preprint arXiv:1907.13625*, 2019.

[113] R. D. Hjelm *et al.*, "Learning deep representations by mutual information estimation and maximization," *arXiv preprint arXiv:1808.06670*, 2018.