Application-driven Intersections between
Information Theory and Machine Learning

by

Litian Liu

B.Eng., The Chinese University of Hong Kong (2016)
M.Eng., Princeton University (2017)

Submitted to the Department of Electrical Engineering and Computer
Science
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2021

© Massachusetts Institute of Technology 2021. All rights reserved.
Application-driven Intersections between Information Theory and Machine Learning

by

Litian Liu

Submitted to the Department of Electrical Engineering and Computer Science on May 18, 2021, in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science and Engineering

Abstract

Machine learning has been tremendously successful in the past decade. In this thesis, we introduce guidance and insights from information theory to practical machine learning algorithms. In particular, we study three application domains and demonstrate the algorithmic gain of integrating machine learning with information theory. In the first part of the thesis, we deploy the principle of network coding to propose a decomposition scheme for distributing a neural network over a physical communication network. We show through experiments that our proposed scheme dramatically reduces the energy used compared to existing communication schemes under various channel statistics and network topologies. In the second part, we design a learning-based coding scheme, developed from the concept of error correction codes, for bio-molecular profiling. We show through simulations that, with a learning-based encoder and a maximize a posterior (MAP) decoder, our scheme significantly outperforms existing schemes in reducing the false negative rate of rare bio-molecular types. In the third part, we exercise guesswork on the machine translation problem. We study machine translation using the seq2seq model and we provide insights into quantifying the uncertainty within. Our results shed light on the design of inference in machine translation for selecting the beam size in beam search.

Thesis Supervisor: Muriel Médard
Title: Cecil H. Green Professor of Electrical Engineering and Computer Science
Acknowledgments

This thesis would not have been possible without the support and kindness of so many.

First and foremost, I would like to thank my thesis advisor, Muriel Médard, for her insightful advice and all-around support. I joined her group vaguely knowing that I would like work on the intersection between information theory and machine learning. I am ever so grateful for Muriel’s unwavering support for me to explore my interest and for all the opportunities she made possible for me that eventually paved the way for the thesis. My gratitude to Muriel goes beyond the academic. I am more than grateful for her support during the inevitable hardships in my graduate journey and for her help and understanding in accommodating those.

I am also extremely grateful to all the members of my thesis committee, Professor Ken Duffy, Professor Yonina Eldar and Professor Vincent Chan. I was fortunate to collaborate with Professor Ken Duffy and Professor Yonina Eldar during my PhD. I benefited greatly from the depth of their experience and the breadth of their knowledge. I worked with Professor Vincent Chan in my undergrad. That very first step in research eventually led me towards pursuing a PhD and pursuing a career of a researcher. I am deeply grateful for his invaluable support and advice.

I would like to thank all the past and present members of the NCRC group, including Amit Solomon, Salman Salamatian, Kathleen Yang, Wei An, Benoit Pit-Claudel, Basak Ozaydin, Diana González, Derya Malak, Ahmad Beirami, Wasim Huleihel, Alejandro Cohen, Rafael D’Oliveira, Homa Esfahanizadeh, Jiange Li, and Vitaly Abdrashitov, among others. I am fortunate to collaborate with many, without whom this thesis would not have been what it is. I also feel blessed to be surrounded by this group of brilliant minds, devoted friends and skilled bakers. I would always remember our good old days in 36-512, all the lunches we had together along with our inexhaustible supply of coffee and chocolate. Of particular, I would like to thank Derya Malak for always being there for me as a mentor and friend and for our shared passion for piano and spicy Chinese food. I would like to thank Salman Salamatian for
our intriguing conversations on guesswork, network coding, the three-body problem, and beyond. I would also like to thank Amit Solomon and Kathleen Yang for feeding me countless times with their amazing baking. And a special thanks goes to Molly Kruko, who bounds our group together with her diligent work and sweet kindness. Thank you Molly for your invaluable friendship and support.

I am also grateful to the many colleagues and co-authors outside the group. Of particular, I would like to thank Mehrtash Babadi. Many of our discussions and pair coding sessions at the Broad Institute directly shaped this thesis. I would also like to thank Ahmad Beirami for mentoring me through an amazing summer in Electronic Arts and for hosting me in his living room in my first week.

I am also grateful for the friendship and support from many who accompanied me along this journey. I am particularly thankful for my family in the US. China is far away and I am fortunate to have a place that I call home in the US. I am also thankful for my fiends in Princeton. My stay in Princeton only lasted for a year but I am blessed that our friendship goes far beyond that. Of particular, I would like to thank Rui Wang, for reminding me to keep hydrated throughout my PhD and for her delightful and compassionate virtual company during the pandemic. I am ever so grateful for my friends at MIT and back in China. Of particular, I would like to thank Qingyun Qian. I am more than grateful that we have the support of each other in this challenging year, just like when we were first-years in college. I would also like to thank Yun Wang. Thank you for your decade-long friendship and support through the ups and downs in my life.

Finally, I would like to thank my parents for encouraging me to take the journey of graduate study and for their unconditional love and unwavering support along the way. Thank you, for being my greatest mentors and closest friends.
### Contents

1 Introduction \hfill 13
   1.1 IT tools incorporated into ML algorithms \hfill 15
      1.1.1 Network Codes for Distributing Neural Networks \hfill 15
      1.1.2 Error Correction Codes for bio-molecular profiling \hfill 16
      1.1.3 Guesswork for Machine Translation \hfill 18
   1.2 A brief overview of literature in the intersection of ML and IT \hfill 19

2 Neural Network Coding \hfill 21
   2.1 Introduction \hfill 21
   2.2 System model \hfill 26
   2.3 Neural Network Coding \hfill 27
   2.4 Performance Evaluation \hfill 31
      2.4.1 Baseline \hfill 31
      2.4.2 Topology \hfill 35
      2.4.3 Channel Statistics \hfill 37
      2.4.4 Non-multicasting \hfill 39
      2.4.5 Functional Reconstruction \hfill 40
   2.5 Experiments \hfill 42
      2.5.1 Tasks of different complexity \hfill 42
      2.5.2 Multi-resolution reconstruction \hfill 43
      2.5.3 Same source for distinct tasks \hfill 44
      2.5.4 Functional Reconstruction of Fashion-MNIST images \hfill 45
   2.6 Discussion \hfill 47
3 Coding in Bio-molecular Profiling

3.1 Introduction

3.1.1 A brief overview of the MERFISH protocol

3.2 Methodology and Results

3.2.1 A generative model for mFISH data

3.2.2 Image processing and model fitting

3.2.3 Quantization, channel model and estimation

3.2.4 Decoding: MAP and MLE decoders

3.2.5 Comparing the performance of different decoders

3.3 Data-driven code construction

3.4 Discussion

4 Guesswork in Machine Translation

4.1 Introduction

4.2 Study on seq2seq model

4.3 Guesswork and its application here

4.3.1 Comparison with the existing bounds

4.3.2 Extension to multiple correct answers

4.4 Discussion

5 Conclusion
List of Figures

2-1 NNC overview ................................................................. 22
2-2 An end-to-end NN example of NNC .................................. 28
2-3 Illustration of NNC transmission quality with different power ........ 33
2-4 Illustration of NNC performance under different topologies .......... 36
2-5 Illustration of NNC performance with channel interference ............ 38
2-6 NNC Performance over fading channels .................................. 39
2-7 Illustration of a non-multicasting task ................................... 40
2-8 Illustration of a functional reconstruction task ......................... 41
2-9 Illustration of a butterfly network where two tasks of different difficulty coexist .................................................. 43
2-10 Illustration of a multi-resolutional reconstruction task ................ 44
2-11 Illustration of using the same source for distinct tasks ................. 46
2-12 Illustration of the functional reconstruction of Fashion-MNIST images 47
3-1 A schematic overview of a typical mFISH experiment ................. 54
3-2 Extraction of isolated spots from MERFISH images .................. 55
3-3 Modeling spot intensities as two-component Gaussian mixture for each round and channel ........................................... 56
3-4 Comparing the performance of different decoding schemes for randomly assigned MHD4 codes .......................................... 60
3-5 Evolutionary optimization of code assignment for MHD4 codes .... 63
4-1 The seq2seq model for machine translation ........................... 68
The approximate upper bound in Theorem 1 for the probability mass covered by beam search with varying beam size.

Numerical comparison of the upper bounds for the probability mass covered in beam search.
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Minimum transmission power per image for achieving the corresponding mean pSNR values under NNC, ANC, and JPEG schemes.</td>
<td>34</td>
</tr>
<tr>
<td>2.2</td>
<td>Minimum transmission power per image for achieving the corresponding mean pSNR values under NNC and ANC with channel interference.</td>
<td>37</td>
</tr>
<tr>
<td>4.1</td>
<td>Symbols and notation in Chapter 4</td>
<td>68</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The past decade has witnessed an unprecedented success of Machine Learning (ML) in a wide range of applications. Powered by the advance of deep neural networks (NN) [49], machine learning goes beyond human-level performance from facial recognition [81] to challenging games such as Go [76] and Starcraft [83]. Despite its huge success, many ML algorithms at current stage are intuition based. Progress in deep NN based algorithms is largely driven by sporadic ingenious intuitions rather than provable and systematic approaches. Much remains to be understood as to answer why deep NNs work and how to improve their performance algorithmically.

In this thesis, we leverage tools from Information Theory (IT) to provide insights and guidance for practical ML algorithms. Information theory studies the general achievability and inachievability in using resources for approximating a function. Traditionally, information theory is oriented towards communication, networks and security. Oftentimes the function of interest is the identity function. For example, in communication and networks, we aim to reconstruct the transmitted signal at the receiver side. In other words, we approximate an identity map of the transmitted signal under the constraints imposed by the system. In security and privacy, on the other hand, we quantify the threat of the system under the attack of an adversary hacking the password. In other words, we quantify the difficulty of reconstructing an identity map of the password. Although information theory is function-driven, same as machine learning, classical information theoretical tools may not be directly appli-
cable to machine learning problems. In contrast to the focus of information theory on identity functions, machine learning is interested in a wide range of functions that vary from applications to applications. For example, in machine translation the goal is to learn a function that maps text from one language to another [6], whereas in image classification, the goal is to map an image to a label corresponding to a category [55]. Because of the differences in the focus and application, information theoretical tools may require adjustments and modifications to be applied in understanding how to best utilize available resource in learning settings.

In this thesis, we explore the application-driven intersections between information theory and machine learning. Inspired by communication and security, we show in this work promising ways of modifying information theoretical tools to provide insights and guidance for practical machine learning algorithms:

1. **Network Coding for Distributing Neural Networks**: The merging scenarios from the Internet of Things (IoT) to autonomous driving feature the model of collaborative inference among devices over a pool of data. One potential scheme for the model is to distribute an NN over a physical network. To this end, we propose a scheme which utilizes the concept of network coding [2] in distributing an NN. Our scheme significantly reduces the energy required compared to existing schemes.

2. **Error Correction Codes for Bio-Molecular Profiling**: One important recent development in the biological sciences is multiplexed fluorescence in situ hybridization (mFISH) [12] microscopy techniques, which enables the concurrent identification and localization of bio-molecules of different types. Core to the technology is a coding problem: how to design a profiling codebook and how to decode the noisy measurements? We adopt the concept of error correction codes from communication and generate codebooks in a data-driven manner with learning algorithms. With the combination of learning-based encoding and maximum a posteriori (MAP) decoding, our scheme provides significant performance gain compared to state-of-art techniques in reducing the false dis-
covery rate of rare molecular types.

3. **Guesswork for Machine Translation**: Inference in machine translation features a difficult problem of searching through a large unconcentrated probability space. Sequence-to-sequence (seq2seq) model is used in practice, together with beam search, to cope with the problem \[80\]. We will exercise guesswork, an information theoretical concept, to propose a framework for quantifying the uncertainty in machine translation with seq2seq model. Our results provide insights for the design of the inference step, in terms of selecting the beam size in beam search in an algorithmic way. We discuss the extension to translation with distortion tolerance.

1.1 **IT tools incorporated into ML algorithms**

In this thesis, we place ourselves in the philosophical realm of actionable techniques. We explore how information theoretical approaches can be incorporated into practical machine learning algorithms. We show promising directions where the insights and guidance from information theory can lead to algorithmic gain in practical machine learning problems.

1.1.1 **Network Codes for Distributing Neural Networks**

Nowadays, the merging scenarios from IoT to autonomous driving introduce new challenges to the communication and computation within a network. Many applications in such scenarios request a group of devices to assess and predict based on their distributed measurements of the environment. For example, considering a smart wearable health device that collects vital signs via body sensors. Based on received signals, the device might choose to store the data or dial 9-1-1 immediately. Such task-driven decision-based applications are reminiscent of neural networks, which has tremendous potential in constructing a function, i.e. making a decision, based on a large pool of data.
One natural scheme for the applications is to transmit the distributed signals using classical communication schemes and then process the data, potentially using an NN, at destinations. However, many features in IoT and autonomous driving put pressure on such a scheme. On the one hand, a centralized scheme is vulnerable to a man-in-the-middle attack, which is crucial in many IoT applications where the data are personal and highly sensitive. On the other hand, the stringent latency requirement and the limited bandwidth and energy in IoT and autonomous driving devices preclude computationally demanding long-block length source and channel coding techniques in classical communication schemes.

Therefore, we are motivated to allocate resources across a physical communication network and to distribute an NN within. Network coding [2], from information theory literature, provides a framework for allocating computation tasks at intermediate nodes either for transmission purposes [37] or for transmission and computation purposes [30]. In Chapter 2 we propose neural network coding (NNC) which deploys the principle of network coding in decomposing an NN. We study a fundamental model of communication networks consisted of links where information flows where information theory applies, which consists of . The network model of switches and routers is beyond the scope of this thesis. In NNC, we propose to encode and decode network codes transmitted on communication links using NNs at each ML-empowered device. Our scheme provides a low-latency low-complexity energy-efficient scheme for collaboratively achieve the communication goal over a network. We show through experiments that NNC leads to huge performance gain in comparison with classical communication schemes. Our work is the first to view an NN as a physical network.

1.1.2 Error Correction Codes for bio-molecular profiling

Error correction codes have long been used for communication over noisy channels. The key idea is to encode data with redundancy so as to control the error. Ever since Shannon established the fundamental rate of error correction codes in 1948 [75], the concept of error correction codes has been applied in many other domains in
combat with the noise in the system. For example, a class of error correction codes, namely Reed–Solomon codes [70], is widely used in data storage techniques such as compact disc and DVD. Another class of error correction codes, Hamming code [34], has widespread applications in correcting errors in computer memory system.

In recent years, error correction codes are brought into attention in the field of single-cell biology. Of particular interest, is the recent advent of mFISH techniques which allows molecular profiling within cells without disturbing the spacial arrangement [12, 62, 61, 74, 28]. The benefit of incorporating coding principles into mFISH techniques is twofold. On the one hand, coding enables the multiplexing of measurements so that a large number of bio-molecular types can be measured in a few rounds of experiments. On the other hand, redundancy can be encoded into so as to correct errors in noisy measurements.

As of current stage, coding schemes in mFISH techniques are immediately borrowed from the communication society. In [12, 62, 61], 16-bit codes with 4 ones, 12 zeros, and minimum Hamming distance 4 are used. The differences between a classical communication model and the mFISH experiments put pressure on the effectiveness of directly applying communication codes in mFISH experiments. First, the false alarm rate and the mis-detection rate are significantly different in mFISH experiments [12] whereas classical Hamming-distance based coding techniques presuppose symmetric error rates. Second, the prior distribution of codewords in mFISH experiments is determined by the distribution of bio-molecules and is therefore highly non-uniform. This is in contrary to the communication model where codeword distribution is uniform as a result of source coding, or compression.

In Chapter 3 we present our application-driven coding scheme for mFISH experiments. We establish a channel model from data and we propose to use a MAP decoder together with codes constructed in a data-driven manner. Our scheme significantly improves the state-of-art performance in terms of reducing the false discovery rate for rare bio-molecular types.
1.1.3 Guesswork for Machine Translation

Guesswork is an information theoretic concept originally studied by Massey in [60]. Guesswork studies the process of guessing the realization of a random variable $X$ by asking the question of “Is $X$ equal to $x$?” until the answer is “Yes”. Guesswork for a sequence of independent identically distributed (henceforth, i.i.d.) random variables is studied in [3]. The result is extended to the Markov case in [58] and other cases in [66], [36]. It is proved in [14] that the guesswork of a stochastic process satisfies a large deviation principle under certain regularity conditions. Furthermore, guessing under mismatch, distortion and source uncertainty is studied in [5, 36, 72, 79].

In literature, guesswork has its applications in security and communication. On the security side, guesswork quantifies the computational effort of a brute force attacker hacking a password. Guessing in multi-user scenario is studied in [15], guessing with limited memory is studied in [38], and guessing with side information is studied in [52]. On the communication side, guesswork is recently introduced into decoding in communication system. Guessing Random Additive Noise Decoding (GRAND) [23] was proposed. The idea of GRAND is to identify the codeword by guessing the noise rather than the codeword itself. A computational efficient variant of GRAND is proposed in [24] and a soft detection variant is introduced in [78].

In Chapter 4 we excise the methods and tools developed in guesswork in a machine translation problem. Machine translation is an intrinsically hard problem. The inference step involves searching through a distributed ginormous probability space [65]. In practice, seq2seq model is used to cope with the uncertainty and beam search is used in the inference step. Quantifying the uncertainty in machine translation is a reminiscence of quantifying the security of a system in facing a brute force attack. We therefore utilize the tool of guesswork and propose a framework for quantitatively measure the uncertainty in machine translation problems. Our results provide insights for designing the inference step, in terms of selecting the beam size in beam search algorithmically.
1.2 A brief overview of literature in the intersection of ML and IT

In this section, we introduce some of the recent literature in the intersection of machine learning and information theory. The selected references here are by no means exclusive. Our goal in this section is to provide the background information so as to better clarify the aim and scope of this thesis.

One line of work in the intersection of information theory and machine learning applies information-theoretical tools analytically in estimating and learning from data. This line of work does not philosophically belong to the school of thought in proposing novel applicable algorithms. Rather, the focus is to analyze existing algorithms or to understand the algorithm-independent limits of learning. Literature along this line of work includes but is not limited to: quantifying the necessary and sufficient number of observations for recovering sparsity pattern [83]; proposing privacy framework for characterizing privacy threat in statistical inference [22]; deriving finite-time guarantees for Stochastic Gradient Langevin Dynamics in non-convex learning problems [68]; and, deriving bounds on the best possible approximation of a distribution from its samples [44].

We do not put ourselves in this school of thought. Instead, we are interested in the practical application of machine learning algorithms, rather than the limit of machine learning.

Another line of previous work in the intersection between information theory and machine learning is to decode or/and encode in communication system using machine learning algorithms. Some of the literature focus on designing ML-based decoders for existing codes. One motivation is that classical decoders are typically designed for the additive white Gaussian noise (AWGN) channel, and are sub-optimal for arbitrary channel statistics [11]. ML-based decoders for short random codes [18] and polar codes [4] are shown to achieve MAP bit error rate performance in [33]. The ML-decoder in [33] is further improved for longer polar codes in [11] and [90]. In addition, ML-based decoders for convolutional codes [43] and turbo codes [8] are shown to
approach optimal performance in minimizing bit error rate and block error rate for AWGN channels and can adapt into arbitrary channel statistics [45]. Other work in this school proposes to design novel codes for communication using ML algorithms and to decode accordingly. A channel coding scheme with the encoder and decoder both powered by neural networks is proposed in [42]. Learning the physical layer representation is studied for the single-input and single-output (SISO) system in [64], the multiple input and multiple output (MIMO) system in [63] and the orthogonal frequency-division multiplexing (OFDM) system in [31]. An NN-based joint source channel coding (JSCC) scheme is proposed for images in [10], and for text in [29].

Part of our work stands along this school of work. In Chapter 2, we present neural network coding to decompose a neural network following the principle of network coding. In our proposed scheme, network codes are constructed and decoded at each ML-empowered device. Our work elevates previous work to a networked perspective.
Chapter 2

Neural Network Coding

2.1 Introduction

The problem of accessing distributed sources over a network has a decades-long history\cite{27}. In various examples such as IoT and autonomous driving, we see a merging of new scenarios of the classical problem with novel features in power budget and latency requirements etc. Applications in such scenarios oftentimes request a group of devices to assess and predict based on their distributed measure. For example, Figure 2-1(a) demonstrates such a scenario, where signals generated by distributed measurements are transmitted over a communication network and reconstructed at both the destination nodes. As shown in the example, the signals in IoT and autonomous driving are often task-specific: they are generated by specific measurements and therefore have a non-trivial underlying distribution. Such signals can be personal and highly sensitive in applications like Smart Home\cite{40}. Moreover, the signals are often of a small size, because of the limited buffer size of edge devices and the stringent latency constraints in IoT and autonomous driving. The energy budget in such scenarios is also low, as edge devices typically have limited energy. Furthermore, the end goal in accessing the data can be versatile. Figure 2-1(a) demonstrates a multi-cast problem, where both destination nodes seek to reconstruct all the input signals. In other cases, a destination node might be interested in a subset of the input signals or a function of the input signals rather than the signals themselves. For example, in smart wearable
health devices [21], one central processor could be solely interested in whether the person needs emergency care. The exact values of the heart rate or blood pressure could matter less. The features in merging scenarios impose challenges on existing schemes, which are to first transmit the signals using classical communication techniques and then process the signals in a centralized manner. This is because the long-block length source and channel coding techniques introduces high computational costs as well as latency overhead into the process. Also, centralized data processing makes the system particularly vulnerable to man-in-the-middle attack, which can be crucial considering the private data collected in IoT applications. On the other hand, the novel features in emerging scenarios also open up new opportunities for designing application-specific schemes for accessing distributed sources over a network.

Communication and computation over a network is an intrinsically hard problem. Network coding, which allows computation in intermediate nodes, is shown to save bandwidth compared to routing for communication over a network in general [2]. However, Ramamoorthy et al. [69] have demonstrated the failure of the separation between source and network coding for multicast networks. Effros et al. [26] have demonstrated the same failure for non-multicast networks. The requirement for joint
coding indicates the insufficiency of using efficient source codes, followed by efficient network codes. A joint source and network coding scheme, namely random linear network coding, has been shown in [37] to be capacity-achieving for multicasting sources over a network, with the error exponent generalizing the linear Slepian-Wolf coding [19]. However, for sources that are neither independent nor linearly correlated, minimum entropy (ME) decoders or MAP decoders are required for random linear network coding. Despite the efforts toward reducing the complexity of ME or MAP decoders [56] [16], it is in general NP-hard to find a ME or MAP solution. As a result, joint source and network coding schemes with low-complexity decoders have been proposed, but those apply only to restricted settings. For example, in [88] sources are restricted to binary ones and in [50] the number of sources is restricted to two. To the best of our knowledge, no practical joint source and network coding schemes exist for multicasting arbitrarily correlated real-life signals over a point-to-point network of arbitrary topology. The general non-multicast communication problem where each destination seeks to reconstruct an arbitrary subset of the input signals is considerably harder than the multicast problem. Sufficient conditions for solving a non-multicast problem using random linear network coding is derived in [46]. A subset of non-multicast problems that can be solved using arbitrary non-linear coding strategies is derived in [91]. To the best of our knowledge, there is no existing solution for non-multicast problems in general. As for the case where destinations are interested in the functions of the input signals, distributed functional computation adds another layer of complexity as computation can happen within the network itself. A modularized coding scheme for distributed functional computing is derived in [30] for tree networks with independent sources. A flow-based analysis for functional compression in general network topologies is conducted in [57] which might not be implemented in practice because it is based on NP-hard compression concepts, such as compression of characteristic graphs. To the best of our knowledge, there is no existing practical solution for functional reconstruction problems in general.

The difficulty of the tasks motivates us to take an application-specific approach. The task-driven decision-based applications in IoT and autonomous driving are rem-
iniscent of NNs, which has tremendous potential in many application domains. And the applications to exact reconstruct the signals are reminiscent of an auto-encoder [32], which is an NN trained to minimize the distortion, e.g., mean square error between its output and input. Therefore, we are motivated to allocate computation resources across a physical communication network and distribute a NN within. We follow the principle of network coding [2] in distributing an NN, since network coding, as we briefly introduced above, provides a framework for distributing computation within a physical network for communication and/or computation purposes [37, 30].

We are also motivated by the increased effort towards data-driven scheme for accessing data, for both signal reconstruction problems [64, 63, 31, 10, 29] and functional reconstruction problems [39]. As opposed to the traditional methods, data-driven approaches make no assumptions on the source statistics, which leads to a significant improvement when looking at complex sources of data such as images. Instead, such methods aim at discovering efficient codes by making use of a (potentially large) pool of data, in conjunction with a learning algorithm. A successful candidate for the latter is an NN. All works that we know have focused on the point-to-point communication problem, under various channels. For example, learning the physical layer representation is studied for the single-input and single-output (SISO) system in [64], the multiple input and multiple output (MIMO) system in [63] and the orthogonal frequency-division multiplexing (OFDM) system in [31]. An NN-based joint source channel coding (JSCC) scheme is proposed for images in [10], and for text in [29]. Though traditional techniques are optimal in the asymptotic regimes, in practical scenarios, it was shown in [64, 63, 31, 10, 29] that NN-based methods were competitive, and even out-performed the state of the art methods in information theory in some signal-to-noise ratio (SNR) regimes. Here, we shift the scope to the network perspective, and make a first step in broadening our understanding of the benefits and challenges of data-driven approaches for a networked data accessing scheme.

To this end, we propose an application-specific data accessing for networked settings, which we refer to as neural network coding [54], or NNC for short. The NNC scheme has the following main benefits: (a) it makes no assumptions on the statis-
tics of the source, but rather makes use of a seed dataset of examples; (b) it is an end-to-end communication scheme, or equivalently, a joint source and network coding scheme; (c) its decoding is of low complexity; (d) it can be applied to any network topology; (e) it can adapt to specific tasks; (g) it acts in a distributed manner and therefore robust to security attacks; and (g) it can also be used with various power constraints at each of the source and intermediate nodes.

Figure 2-1(a) demonstrates an NNC applicable scenario, where arbitrarily correlated signals are multi-cast over a communication network. The network has four source nodes and two destination nodes. From an end-to-end point of view, NNC distributes an end-to-end auto-encoder across the physical network subject to topology constraints. From each device’s point of view, in NNC, the encoders for network codes at the source nodes and intermediate nodes, as well as the decoders at the destination nodes, are NNs as shown in Figure 2-1(b). The resulting network code construction is jointly designed with the encoding phase and decoding phase of the transmission, where real-valued input signals are mapped into channel inputs, and channel outputs are reconstructed into real-valued signals. NNC scheme can be optimized through training and testing offline over a large data set and can be readily implemented. Note that network codes constructed in NNC are non-linear, as oppose to linear codes in the literature [37, 46, 59, 26]. Our experimental results showed the benefit of having a non-linear rather than linear code construction. We also illustrate through experiments on images that NNC achieves a better performance for multicasting problems compared to a separation-based scheme that relies on a compression scheme (JPEG [85]), followed by capacity-achieving network coding. Furthermore, we demonstrate through experiments that NNC can adapt into a variety of network topologies and tasks, including theoretically-hard non-multicasting problems and functional reconstruction problems. We numerically illustrate the benefit of adopting a task-specific scheme like NNC especially in low power regime. While still in its infancy, we believe that NNC and its variants may pave the way to an efficient way to exploit the decomposition of NN, which appears to be a promising approach to for data accessing in networked settings.
The rest of the chapter is organized as follows. Section 2.2 describes our system model where we model the network as a graph: Nodes are devices and each edges are (potentially multiple) connections between devices. Section 2.3 presents our design principle where each node has an NN for network code construction and decoding and we jointly optimize the scheme. Section 2.4 first studies the performance of NNC in a multicasting task over the butterfly network with AWGN links. We then build upon the baseline experiment and study the performance of NNC with one property of the baseline changed. The properties we considered in this section includes network topologies, channel statistics, and the communication task. 2.5 explores various application of NNC. Section 2.6 summarizes the paper and discusses possible extensions.

2.2 System model

Throughout the chapter, we use $x, \vec{x}, X, \vec{X}$ to denote a scalar, a vector, a random variable, and a random vector respectively. We model the communication network as an acyclic directed graph $G = (\mathcal{V}, \mathcal{E})$. Elements $v_i$ of $\mathcal{V}$ are called nodes and elements $(i, j)$ of $\mathcal{E}$ are called links. Each link $(i, j)$ is assigned an energy constraint $p_{i,j} \geq 0$, which specifies the maximum signal energy that can be sent from node $v_i$ to node $v_j$. We consider two disjoint subsets $\mathcal{S}, \mathcal{D}$ of $\mathcal{V}$, where each element in $\mathcal{S}$ is called a source node and each element in $\mathcal{D}$ is called a destination node. Let $N$ and $M$ denote $|\mathcal{S}|$ and $|\mathcal{D}|$ respectively. We consider $n$ virtual sources $\{s_i\}_{i=1}^n$ located at $N$ source nodes. Each $s_i$ generates a random variable $X_i \in \mathbb{R}, i = 1, \ldots, n$, according to the joint probability density function $f_{X_1, \ldots, X_n}$. The resulting random vector is denoted by $\vec{X} \in \mathbb{R}^n$. Observe that $n$ may not be equal to $N$. This setup encompasses the case in which some of the sources are co-located, or some physical sources generate random variables of higher dimension, by grouping some of the sources into a source node in $\mathcal{S}$. Thus, when appropriate we shall refer to a source node $s \in \mathcal{S}$ to represent the collection of virtual sources which are co-located at $s$ (c.f. Experiment sections 2.4 and 2.5).
We model each link in the network as a set of parallel channels. More precisely, two nodes $v_i$ and $v_j$ are connected via $k_{i,j}$ parallel noisy channels. On each link $(i, j)$, $v_i$ may transmit any signal $\tilde{W}_{i,j}$ in $\mathbb{R}^{k_{i,j}}$, subject to the average power constraint on the link, i.e. $\mathbb{E}[\|\tilde{W}_{i,j}\|^2] \leq p_{i,j}$. The signal $\tilde{W}_{i,j}$ on each link is corrupted by noise, fading and/or interference from other signals. The received signal at node $v_j$ is $\tilde{Y}_{i,j} \in \mathbb{R}^{k_{i,j}}$. In the special case of independent zero-mean Additive White Gaussian Noise (AWGN) channels, the node $v_j$ receives $\tilde{Y}_{i,j} = \tilde{W}_{i,j} + \tilde{N}_{i,j}$, where each element of the $k_{i,j}$-dimensional vector $\tilde{N}_{i,j}$ is a zero-mean Gaussian random variable with variance $\sigma_{i,j}^2$. Note that this setup can model either band-limited channels in wireless communication, or the parallel channels in the Ethernet connection.

We study the general communication problem where each destination node $t \in \mathcal{D}$ is interested in a target function $f_t$ of the sources. For example in a multi-cast scenario, the target function $f_t$ for $\forall t \in \mathcal{D}$ is the identity function of all the sources.\footnote{Therefore, the reconstruction of the input signals is a special case of the functional reconstruction. We refer to both as functional reconstruction from now on.} The performance of the scheme can be evaluated by a tuple of distortion measures $\delta_t$, $t \in \mathcal{D}$, where each measure is defined between the target function over the source $f_t(\tilde{X})$ at $t$ and the functional reconstruction $g_t(\tilde{X}^t)$ at $t$. For example for hand-written digits in Figure 2-2, the target function $f_t$ can be digit and $\delta_t$ could be the accuracy.

### 2.3 Neural Network Coding

In NNC, we jointly design the channel inputs at the source nodes and at the intermediate nodes – this makes NNC a joint source and network coding scheme. Existing joint source and network coding schemes, e.g., \cite{37, 88, 50, 46}, assume error-free point-to-point transmission on each link, and focus on the network layer operations. The physical layer then relies on a separate channel coding scheme with potentially a high latency, as it is assumed that each link employs an error correction coding with a large block length. In contrast, in NNC the signal inputs are directly transmitted over the noisy channels, i.e., there are no underlying physical layer codes. As such,
Figure 2-2: An end-to-end NN example of NNC. There are $n = 784$ virtual sources located at $N = 2$ source nodes (at the left), and there are $M = 2$ destination nodes (at the right). Each link consists of 32 noisy channels, as indicated by the number above each arrow. Each node has an inner NN to construct and decode network codes. Each inner NN is represented by a quadrilateral where the left-hand side is the input dimension the right side is the output dimension of all the outgoing links. For example, the inner NN at top source node has input dimension 392 and output dimension 64. Half of the output goes to the upper destination node and the other half goes to the middle node. Each inner NN is implemented to be trainable. Each link is implemented with a non-trainable NN layer of width 32. Technical details of implementation can be found in Section 2.7.
the communication problem described in Section 2.2 can be decomposed into three phases as shown in Figure 2-1(a): the encoding phase, the transmission phase, and the decoding phase. NNC operates in a *one-shot* manner over all three phases. In the encoding phase, real-valued signals at the source nodes are directly mapped into network codes. The length of a network code $\vec{W}_{i,j}$ is designed to match the number of independent channels $k_{i,j}$ contained in link $(i, j)$. Therefore, $\vec{W}_{i,j}$ can concurrently be transmitted through the noisy channels over link $(i, j)$. In the transmission phase, network codes $\vec{W}_{i,j,s}$ are directly constructed at node $v_i$ from the incoming noise-corrupted network codes $\{\vec{Y}_{l,i} : l \in c(i)\}$, where $c(i)$ is the set of direct predecessors of $v_i$. In the decoding phase, each destination node reconstructs the target function of the transmitted signals directly from the noise-corrupted network codes it receives. NNC does not involve any classical long block-length source or channel coding techniques, and therefore is free of their associated latency penalty and computational overhead.

Note that by picking a non-linear activation, the resulting joint source and network code is non-linear by design. As mentioned in Section 2.1, the non-linearity in codes may be crucial in constructing efficient codes for the problem at hand. We design the network code from node $v_i$ to node $v_j$ by constructing a NN with input dimension $d_{i,n}^i$ and output dimension $k_{i,j}$. The last layer of the NN is a normalization layer to enforce the power constraints. When $v_i \not\in \mathcal{S}$, $d_{i,n}^i$ is the number of incoming channels. When $v_i \in \mathcal{S}$, $d_{i,n}^i$ is the dimension of signal generated at $v_i$. During a transmission, noise-distorted network codes received at $v_i$, $\{\vec{Y}_{l,i} : l \in c(i)\}$, are fed into the NN if $v_i \not\in \mathcal{S}$. Or the generated signal is fed into the NN if $v_i \in \mathcal{S}$. The NN output is the network code $\vec{W}_{i,j}$ to be transmitted over link $(i, j)$.

Similarly, for functional reconstruction of the inputs signal, we decode the received noise-distorted network codes with a NN at each destination node $t$. Note that NNs at destination nodes are low-complexity decoders, since each layer of a NN is an affine transformation followed by an element-wise non-linear function. We say that the set of functions for constructing and decoding network codes at each node specifies a NNC *policy* for the communication system, if each of them can be represented as a
NN. Under a NNC policy, the resulting end-to-end encoding-decoding can be seen as a series of NNs connected via communication links, as given by the physical network topology. We simulated the channel statistics of communication links by NN-layers as well, with the main difference that those layers have fixed (non-trainable) parameters which correspond to the channel statistics. Thus, under a NNC policy, we construct an end-to-end NN, where some of the layers have non-trainable parameters. The end-to-end NN has physical topology of the communication system embedded, and has NNs which are used for constructing and decoding network code as its sub-graphs. We refer to the NNs for constructing and decoding network code as inner NNs henceforth. Overall, there are $N$ input layers and $M$ output layers in the end-to-end NN. Each input layer has width equal to the dimension of the source generated at the node. Each output layer has width equal to the output dimension of the target function at the node. An illustration of an end-to-end NN is given in Figure 2-2.

With $\tilde{X}$ partitioned and fed into the input layers, the outputs of the end-to-end NN simulate the functional reconstruction at the destination nodes under the current NNC policy. Recall that $\delta_t$, as defined in Section 2.2, is the task-specific distortion measure between the target function over the source $f_t(\tilde{X})$ and the task-specific functional reconstruction $g_t(\tilde{X})$ at a destination node $t$. Parameters $\{\theta_l\}_t$ of the NNC policy are initialized randomly and are trained to minimize

$$\sum_{t \in \mathcal{D}} \delta_t$$

over a large data set sampled from $f_{X_1,\ldots,X_n}$. The parameters of the NN policy can be trained and tested offline\(^2\) using a variety of available tools, e.g., [13], [1]. Note that for a point-to-point link, NNC reduces to deep JSCC in [10].

\(^2\)For the best performance, efforts are in general required to optimize over the choice of hyper-parameters as well, as is the case in other applications of NNs. Hyper-parameters, such as the number of layers and the activation functions in every inner NN, can also be tuned and tested offline before implementation.
2.4 Performance Evaluation

2.4.1 Baseline

We start with experiment of multi-casting an MNIST image over a butterfly network, as shown in Figure 2-2. Technical details of implementation can be found in 2.7.

In this setup, there are two source nodes \( N = 2 \) and two destination nodes \( M = 2 \). A normalized MNIST image, with pixel values between 0 and 1, is split between the two source nodes, such that each source node observes only one half of the image. In other words, 392 out of \( n = 28 \times 28 \) virtual sources (pixels) are co-located at each source node, where the top 392 pixels are located at the first source node, and the rest at the second. Note that in this setup, sources are correlated. Each link in the butterfly network consists of 32 independent parallel AWGN channels \( (k_{i,j} = 32, \forall i, j) \), with zero-mean noise of variance \( 10^{-4} \). Performance is evaluated by the peak signal to noise ratio (pSNR) at each destination node, defined as

\[
pSNR_t = 10 \log_{10} \frac{\max \{X_i\}^2}{\mathbb{E}[(\bar{X}^t - \bar{X})^2]},
\]

(2.2)

where \( \max \{X_i\} \) is the largest value of the input pixels. The pSNR is essentially a normalized version of the MSE and can be used for performance comparison between inputs with different pixel ranges. The choice of pSNR as a distortion measure is natural for images.

A NNC policy is trained to minimize

\[
\min_{\{\theta_l\}_l} \sum_{t=1}^{2} H_b(\bar{X}, \bar{X}^t) \quad s.t. \quad \mathbb{E}[\bar{W}_{i,j}^2] \leq P \quad \forall i, j,
\]

(2.3)

where \( H_b(\bar{X}, \bar{X}^t) \) is the binary cross entropy between the original image and the reconstructed image at destination node \( t \). Note the dependence of \( H_b \) and \( \bar{W}_{i,j} \) on \( \{\theta_l\}_l \) is omitted in the expression for simplicity. We use binary cross entropy
$H_b(\bar{X}, \bar{X}^t)$, defined as

$$H_b(\bar{X}, \bar{X}^t) = -\frac{1}{n} \sum_i X_i \log(X_i^t) + (1 - X_i) \log(1 - X_i^t), \quad (2.4)$$

in the objective function instead of pSNR as an engineering tweak to speed up the training process. Note that the power constraints on all links are the same and this setup demonstrates a homogeneous butterfly network.

We studied the performance of NNC in a homogeneous butterfly network under different power budgets by varying $P$ in (2.3). Note that since the noise statistic on each link is fixed, the power-distortion trade-off we demonstrated is equivalent to the SNR-distortion trade-off. As expected, the quality of the reconstruction improves as the transmission power increases, from Figure 2-3(a) (low power at $1.75 \times 10^{-2}$ per image transmission) to Figure 2-3(b) (medium power at 1.75 per image transmission), and finally to Figure 2-3(c) (high power at $1.75 \times 10^2$ per image transmission). Note that when the transmission power is forced to be nearly zero, as shown in Figure 2-3(a), the reconstruction at both destination nodes is close to the average of all the training data, which is essentially the best possible reconstruction when no information flows from the sources to destinations.

We compared the performance of NNC with two baseline methods. The first competitor is a linear analog of NNC, the Analog Network Coding scheme (ANC) [59]. In ANC, transmission power ANC is controlled by the amplification factor at the source nodes. Each intermediate node forwards the sum of its inputs. At both destination nodes, the amplification factor as well as the network topology is known and used to reconstruct the source signals. Note that a $28 \times 28$ MNIST image can be sent by NNC in one-shot, but has to be sent over the network in 13 transmissions by ANC, as there is no compression scheme in the ANC baseline and thus at most 64 pixels can be sent in a single transmission. All distortion in the reconstruction comes from the noise in the channel rather than compression under the ANC baseline.

We compared the minimum required transmission powers for achieving certain pSNR values under NNC and ANC scheme in Table 2.1. For NNC, we set the power
Figure 2-3: Illustration of NNC transmission quality with different power. The first row of each subfigure is the original image, and the following row is reconstruction at one of the two destination nodes of a butterfly network. The two reconstructed images at both destination nodes were similar. (a) low power, (b) medium power, (c) high power. Technical details of implementation can be found in Section 2.7.
constraints $P$ on all links to be the same and recorded the performance of both ends with 10 different values of $P$ evenly spaced in log-scale from $10^{-6}$ to $10^{-2}$. For ANC, transmission power is controlled by amplification factor at the source nodes. Note that performances of both schemes are “symmetric” at destination nodes, reconstructing images with similar pSNR. Such “symmetric” performance can be expected since the network is isotropic. Overall, NNC outperforms the ANC baseline when transmission power is low, and the ANC baseline outperforms NNC when transmission power is high. This is consistent with [59] which shows that ANC is in fact capacity-achieving in the high-SNR regime. We see the benefit of allowing non-linearity in the construction of application-specific short codes.

The second competitor, the JPEG baseline, is a scheme that separates source coding from network coding: images are compressed through the JPEG compression algorithm [85] at source nodes, and are then transmitted by capacity-achieving network codes through error-free channel codes over the network. Distortion in the reconstruction under the JPEG baseline only comes from compression in source coding, as the transmission is assumed to be error-free. Notice that the JPEG baseline has potentially a high latency due to error-free channel coding, which operates on a large block length.

In our experiments, the JPEG baseline reconstructs high quality images with impractically high power. With the JPEG baseline, average pSNR between a reconstructed image and the original image ranges from 17 to 46. In Table 2.1, some representative pSNR values along with the required transmission powers for the JPEG baseline are given. However, the minimum power threshold for using the JPEG base-

<table>
<thead>
<tr>
<th>pSNR</th>
<th>NNC</th>
<th>ANC</th>
<th>JPEG</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.7</td>
<td>$7.0 \times 10^{-6}$</td>
<td>2.10</td>
<td>N/A</td>
</tr>
<tr>
<td>12.0</td>
<td>$7.0 \times 10^{-3}$</td>
<td>2.23</td>
<td>N/A</td>
</tr>
<tr>
<td>13.8</td>
<td>$7.0 \times 10^{-2}$</td>
<td>3.31</td>
<td>N/A</td>
</tr>
<tr>
<td>17.4</td>
<td>$7.0 \times 10^{-1}$</td>
<td>7.12</td>
<td>$4.6 \times 10^{51}$</td>
</tr>
<tr>
<td>20.6</td>
<td>$7.0 \times 10^{0}$</td>
<td>14.6</td>
<td>$2.6 \times 10^{53}$</td>
</tr>
<tr>
<td>22.0</td>
<td>$7.0 \times 10^{1}$</td>
<td>20.9</td>
<td>$4.7 \times 10^{54}$</td>
</tr>
</tbody>
</table>

Table 2.1: Minimum transmission power per image for achieving the corresponding mean pSNR values under NNC, ANC, and JPEG schemes.
line is tremendously high as $10^{51}$ per image. The need of such high transmission power is explained by the JPEG algorithm hardly compressing MNIST images. Before compression, each half MNIST image is 392 bytes. After compression, the average file size of half images ranges from 350 bytes to 580 bytes for different qualities. For small images like MNIST, the overhead of the JPEG algorithm is significant. The same problem exists for other compression schemes like JPEG2000 [77]. Ineffective compression by JPEG is a representative example of how traditional schemes may lack the ability to adapt their rates to different communication scenarios.

In each of the following subsections, we exhibit a different experiment building on the baseline experiment we just described. In each experiment, we change a single property of the baseline simulation. Unless specified, the source distribution, the network topology and the number of channels per link, channels statistics are the same as the baseline experiment.

### 2.4.2 Topology

Our second set of experiments studies the performance of NNC under different network topologies. Figure 2-4(a) shows an experiment where both destination nodes aim to infer the whole image while only the lower half of the image can be sent through the network. In the instance visualized in Figure 2-4(a), the power constraints $P$ on all links are set to be 25. Both destination nodes reconstruct a mixture of “5” and “3”, from the lower half of digit “5”. The experiment illustrates that NNC can partially infer the missing signals from the available signals which are correlated to the missing part: This is an example of NNC learning and utilizing the correlation in data. Since no existing transmission scheme that we know of could recover the missing part of the image from the rest, any baseline competitor can only reconstruct the lower half of the image in our setting. As the baseline competitors are worse than NNC at low SNR in the multicasting setting presented in Section 2.4.1, we expect that the NNC would do even better in this case.

We demonstrate in Figure 2-4(b) the multicast of MNIST images in a modified butterfly network where there are three destinations. The link connecting to the
middle consists of 32 parallel AWGN channels with zero-mean noise of variance $10^{-4}$, which is the same as all other links in the network. In the instance visualized in Figure 2-4(b), with power constraints set to be 25 on each link, source image can be reconstructed at the middle destination node with a slight worse reconstruction quality compared to the top and bottom destination nodes. We have the same observation on the pSNRs of the reconstructed images as well. Over the testing set, the average pSNR is 21.4, 18.9, 21.5 at the top, middle and bottom destination node respectively. This is because, as shown in Figure 2-4(b), the middle destination node has less information available compared to the top and bottom destination nodes: the middle node only has access to the information on its one incoming link whereas the top and bottom destination nodes each has an additional incoming link and therefore has additional information available. This experiment further demonstrates the benefit of NNC over linear network coding schemes: Under a linear scheme, a linear combination of the top and bottom source is sent on the middle link, from which to reconstruct at the middle destination node is impossible. Therefore, we expect ANC to perform poorly in this scenario. As for the comparison with the JPEG competitor, we expect NNC to outperform the baseline competitor when SNR is low, as shown in Section 2.4.1.
Table 2.2: Minimum transmission power per image for achieving the corresponding mean pSNR values under NNC and ANC with channel interference.

<table>
<thead>
<tr>
<th>pSNR</th>
<th>NNC</th>
<th>ANC</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.71</td>
<td>$7.0 \times 10^{-3}$</td>
<td>2.10</td>
</tr>
<tr>
<td>11.73</td>
<td>$7.0 \times 10^{-2}$</td>
<td>2.12</td>
</tr>
<tr>
<td>13.73</td>
<td>$7.0 \times 10^{-1}$</td>
<td>3.31</td>
</tr>
<tr>
<td>19.62</td>
<td>$7.0 \times 10^{0}$</td>
<td>11.76</td>
</tr>
</tbody>
</table>

2.4.3 Channel Statistics

In wireless communication, channel impairment is often not limited to noise corruption \[82\]. In this section, we consider a wireless setup where links are subject to fading and simultaneous transmissions interfere. We evaluate the performance of NNC under this setup.

Interference

Consider the multicasting of MNIST images over a butterfly network with interfering channels. As shown in Figure 2-5(a), for each node with multiple preceding nodes, signals on each link interferes with each other. The received signal for such a node is $\vec{W}_u + \vec{W}_l + \vec{N}$, where $\vec{W}_u, \vec{W}_l, \vec{N} \in \mathbb{R}^{32}$ are the signal sent by the upper preceding nodes, the signal sent by the lower preceding nodes, and the AWGN noise with zero mean and variance $10^{-4}$ on the channels, respectively. Our setup models the scenario where preceding nodes use the same set of sub-channels in a Frequency-division Multiple Access (FDMA) scheme simultaneously. With transmission power $P = 25$ on all links, images are reconstructed well with average pSNR of 19.9 and 19.7 over the test set for the top and bottom destination node respectively. Some of the reconstructed images are visualized in Figure 2-5(b). We compared the NNC performance with ANC in table . As shown in the table, NNC outperforms ANC when the power budget is low. As for the JPEG competitor, which separates the source code from network code, we expect to encounter the same problem as in the baseline experiment in Section 2.4.1. The JPEG scheme cannot sufficiently compress the MNIST image and requires tremendous transmission power.
Fading Channel

Consider the multicasting of MNIST images over a butterfly network with Rayleigh fading channels [82]. Instead of having parallel channels transmitting real numbers in each link as in Section 2.4.1, we instead consider parallel channels transmitting complex numbers. Channel gains $h$ are drawn in an i.i.d. fashion from a complex Gaussian distribution for each channel, namely $h \sim \mathcal{CN}(0, H_c)$. The channel coefficients remain unchanged for the entire duration of the transmission, and only change between different images. It is worth mentioning that the capacity of the channel is zero, as the state of the channel is unknown to the communicating parties, hence information cannot be sent reliably [10].

We simulated the fading channel with non-trainable layers. Figure 2-6 shows the reconstruction of MNIST image over a butterfly network with $H_c = 1$, power constraints on each link $P = 25$ and 150 channels per link. We trained a NNC policy with batch size 256. Observe that although the capacity of the channel is zero, NNC can still reconstruct some of the images. As the channel capacity in Shannon’s sense is zero in this scenario, our JPEG competitor is ill-defined here. As the channel gains are unknown to the destination nodes, we expect ANC to perform poorly in this scenario. This is because the channel gains are i.i.d sampled and therefore the signals are unproportionally scaled. As a result, a plain addition and subtraction cannot
2.4.4 Non-multicasting

Our next experiment shows the versatility of NNC by demonstrating its adaption into a non-multicasting task. In this experiment, MNIST images are transmitted through a modified butterfly network with three destinations, and each destination node aims to reconstruct a different pixel range, as shown in Figure 2-7. This is an example of a non-multicasting task as a different subset of the source signals is reconstructed at each destination node. The link connecting to the middle consists of 32 parallel AWGN channels with zero-mean noise of variance $10^{-4}$. Note that this setup is not solvable by linear network coding as it does not satisfy the generalized Min-Cut Max-Flow condition derived in [46]. To the best of our knowledge, there is no existing solution to the non-multicasting task. However, we can learn a NNC policy for the given task by setting the target function $g_t$ at each destination node $t$ to be the identity function of the corresponding subset of the source. With transmission power on each link constrained to be 25, the mean pSNR over the test set is 21.2, 16.4, and 21.7 for the top, middle, and bottom destination node respectively. An instance of the test image reconstruction is sketched in Figure 2-7. Observe that non-linear network codes constructed by the NNC policy can solve the non-multicasting communication problem with tolerable distortion. Note that there is no existing solution to the non-multicasting task and any baseline competitor can only multicast to all the destinations in our setting. As the baseline competitors are worse than
2.4.5 Functional Reconstruction

Our next set of experiments re-emphasized the versatility of NNC and further shows how NNC adapts to tasks other than reconstruction. Consider the network shown in Figure 2-8(a). In this network, both destination nodes are interested in determining which digit is sent rather than reconstructing the original MNIST image. This is an example of functional reconstruction of the source signals at the destination nodes. NNC can adapt into functional reconstruction by selecting a corresponding target function $f_t$ and distortion measure $\delta_t$ introduced in Session 2.2. In the example of digit recovery, the target function $f_t$ is a digit recognition function and distortion measure $\delta_t$ is the digit accuracy. Thus we learn a functional NNC policy that recovers the digit in a distributed manner while forwarding the information from the source nodes to the destination nodes. We compared the performance of functional NNC with a centralized computational scheme in Figure 2-8(b). In the centralized scheme, MNIST images are first transmitted by an NNC policy through the butterfly network.
Figure 2-8: Illustration of a functional reconstruction task. (a) A butterfly network where both destination nodes aim to recover the digit in the source image. (b) Performance of functional NNC and NNC followed by a classification function under different transmission power per image. The pSNR at the two destination nodes are equal under both schemes.

as in Section 2.4.1. Each destination then learns a digit recognition function from the image it reconstructs. We set the per-link power constraints $P$ to be same for all links and recorded the performance of both ends with 10 different values of $P$ evenly spaced in log-scale from $10^{-5}$ to $10^{-4}$. As shown in Figure 2-8, in the high SNR regime destinations under both schemes successfully recover the sent digit with functional NNC slightly better than NNC; in low SNR regime, functional NNC significantly outperforms the centralized scheme. This is because functional NNC optimizes the network for the tasks of digit recovering, whereas NNC optimizes for the pPSNR of the reconstructed images, which is not specialized for the task. In high SNR regime, pSNR is very high, so digit recognition should be an easy task. However, in low SNR regime, lowering image quality might focus energy on irrelevant parts of the image. The result demonstrates the benefit of adopting a task-specific scheme when transmission power is limited. Since no existing transmission scheme that we know of could decompose the digit recognition function within the network, any baseline competitor could only multicast the source and determine the digit at the destinations under our setting. As the baseline competitors are worse than NNC at low SNR in the multicasting setting presented in Section 2.4.1, we postulate that functional the performance of NNC would improve in this scenario.
So far each of our experiments modifies a single property of the baseline experiment. What we present next in Section 2.5 are experiments with multiple properties different from the baseline experiment.

2.5 Experiments

In this section, we illustrate the various uses of NNC. Overall, NNC adapts to different tasks by selecting the corresponding target function \( f_t \) and distortion measure \( \delta_t \) introduced in Session 2.2. Unless specified, the source distribution, the network topology, the number of channels per link and the channels statistics are the same as the baseline experiment in Section 2.4.1. Technical details of implementation can be found in 2.7.

2.5.1 Tasks of different complexity

As shown in Section 2.4.4, NNC can adapt to the different reconstruction goals at different destinations. In the following experiment, we demonstrate the analogy with functional reconstruction. As illustrated in Figure 2-9(a), in this experiment, one destination node of the butterfly network aims to exactly recover the digit in the source image while the other destination is only interested in if the digit is smaller than 5. We learnt a NNC policy for the network by selecting a different target function \( f_t \) for different destination node \( t \): \( f_t \) at the top destination node is the digit recognition function as in Section 2.4.5 whereas \( f_t \) at the bottom is a binary classification function of the range. The distortion measure \( \delta_t \) at both destination is the accuracy. We set the per-link power constraints \( P \) to be same for all links and recorded the performance of both ends with 10 different values of \( P \) evenly spaced in log-scale from \( 10^{-5} \) to \( 10^{-4} \) in Figure 2-9(b). As shown in the figure, the digit recovery task always has a lower accuracy compared to the digit range task. We therefore quantified that the digit range task is easier. When the power budget is high, the tasks at both the destination nodes succeed with an accuracy higher than 97%. As power decreases, the performance at both the destinations drops simultaneously.
Figure 2-9: Illustration of a butterfly network where two tasks of different difficulty coexist. (a) A butterfly network where the top destination node aims to recover the digit in source image and the bottom to recover the range. (b) Performance at two destination nodes over different transmission power per image, where the solid line shows the accuracy of the digit recovery task at the top destination node and the dashed line shows the accuracy of the digit range task.

Comparing to Figure 2-8, we observe that the performance of digit recovery task at the top destination is not affected as the lower destination node changes to an easier task. This gives an example of NNC not sacrificing performance under heterogeneous demands. Since no existing transmission scheme that we know of could decompose the task within the network, any baseline competitor could only multicast the source and compute the functions at the destinations under our setting. As the baseline competitors are worse than NNC at low SNR in the multicasting setting presented in Section 2.4.1, we expect that functional NNC would do even better here.

2.5.2 Multi-resolution reconstruction

In this experiment, we consider the case where two ends of the butterfly network are interested in reconstructing the same image with different resolutions. As shown in Figure 2-10, the upper destination is interested in reconstructing the original image with the same resolution and the lower destination is interested in a lower-resolution version. With the power constraint $P = 25$ on all links, the pSNR at the lower destination node is 24.7 which is higher than pSNR at the upper destination node 21.8. This is a result of the lower destination node requiring less information to complete
its task. Since no existing transmission scheme that we know of could decompose the multi-resolution reconstruction task within the network, any baseline competitor could only multicast the source and adjust the resolution at the destinations under our setting. As the baseline competitors are worse than NNC at low SNR in the multicasting setting presented in Section 2.4.1, we expect that functional NNC’s performance would improve in this case. The example of multi-resolution reconstruction models the scenarios where different destination nodes have different reconstruction requirements and resource constraints. A practical scenario closely linked to our experiment is video streaming at multiple rates, where video rate is customized into user requirement, device resolution, available caching size and bandwidth, etc. Our example shows the potential of NNC in solving such problems.

2.5.3 Same source for distinct tasks

In the experiment in Figure 2-11, we consider a butterfly network where two uncorrelated MNIST images are located at the top and bottom source nodes respectively. The top destination node is interested in the addition of the two digits in the source images and the bottom is interested in the subtraction. This experiment presents a
difficult communication task, as both destination nodes have to recognize which digits are sent as well as to perform the required operations. We learnt a NNC policy for the network by selecting a different target function $f_t$ for different destination nodes $t$: $f_t$ at the top destination node has input dimension $784 \times 2$, output dimension 1, mapping two MNIST images into the addition of numerical digits; $f_t$ at the bottom has the same dimension but maps to the subtraction. The distortion measure $\delta_t$ at both destinations is the accuracy. Despite the difficulty of the task, with the power constraints on all links set to be $P = 25$, our NNC policy succeeds over the test set with an average probability of 89.7% and 90.0% at the top and bottom destination nodes, respectively. This is an example of how the same source data can be used to complete different tasks in NNC. This approach can model the practical example of wearable health devices, where blood pressure and body temperature collected from sensors can be used to diagnose multiple diseases, say COVID-19 and the flu, simultaneously. Since no existing transmission scheme that we know of could decompose the task within the network, any baseline competitor could only multicast the source and compute the functions at the destinations under our setting. As the baseline competitors are worse than NNC at low SNR in the multicasting setting presented in Section 2.4.1, we expect that functional NNC would do even better here.

### 2.5.4 Functional Reconstruction of Fashion-MNIST images

In this experiment, we extend the experiments in Section 2.4.5 to Fashion-MNIST data set [89]. Fashion-MNIST data set consists of images of fashion products from 10 classes and the task of this experiments is to identify the class label of the source image, as illustrated in Figure 2-12(a). Our setup is essentially the same as the experiments in 2.4.5 so details are omitted here. Similarly, we compared the performance of functional NNC with a centralized computational scheme in Figure 2-12(b), where Fashion-MNIST images are first transmitted by an NNC policy through the butterfly network and then classified at each destination node. We observed the same pattern as in Figure 2-8. As shown in Figure 2-12(b), in high SNR regime destinations under both schemes successfully recover the label with functional NNC slightly better.
Figure 2-11: Illustration of using the same source for distinct tasks. The butterfly network has two uncorrelated MNIST images located at two source nodes. The top destination node is interested in the addition of two digit and the bottom is interested in the subtraction.

than NNC; in low SNR regime, functional NNC significantly outperforms the centralized scheme. Note that overall the accuracy of classifying Fashion-MNIST images are lower than classifying MNIST images, which is as expected since Fashion-MNIST images are more complicated and the task of classifying Fashion-MNIST images is harder. Also, the gap at high SNR regime between the performance of functional NNC and NNC for Fashion-MNIST images is larger than that for MNIST images. The reason is that the compression in transmitting the Fashion-MNIST images becomes the bottleneck of the centralized scheme, which is limited by the bandwidth (the number of parallel channels) of the network. As we move to more complicated data sets, such as CIFAR [47] and Imagenet [20], we expect the phenomenon to worsen: the exact reconstruction performance of the source will be limited by the bandwidth of the network. Furthermore, we expect the training to be harder and require more engineering effort in hyper-parameter tuning. When the end goal is to reconstruct functions of the original source, we expect the performance gap between NNC and functional NNC to increase with the complexity of the dataset. The gap might be massive such that functional NNC becomes the only practical approach out of the two.
Figure 2-12: Illustration of the functional reconstruction of Fashion-MNIST images. (a) A butterfly network where both destination nodes aim to identify the class label of the source image. (b) Performance of functional NNC and NNC followed by a classification function under different transmission power per image. The pSNR at the two destination nodes are equal under both schemes.

2.6 Discussion

In this chapter, we proposed a novel way of constructing network codes using NN. Our scheme, NNC, provides a practical solution for communicating sources over networks of arbitrary topologies in an application-specific manner. NNC massively lowers the energy used compared to existing schemes. Furthermore, NNC can be easily learnt and tested offline and implemented in a distributed fashion. We examined the performance of NNC under a variety of network topologies, channel statistics, communication tasks through experiments.

A natural extension of NNC arises where multiple tasks exist asynchronously in a system. A separate NNC policy can be learnt for each task. Thanks to the simplicity of NNC implementation, multiple functions can be implemented at one node. Signals can be designed with a flag piggybacked to trigger task-specific functions for constructing and decoding network codes at each node.

As we only present the initial study of NNC in this chapter, many open questions remain unanswered. One such question is how NNC would perform when data are distributed not only across the source nodes but also over the network. In our experiments, all the data are located at the source node. In other communication systems,
however, data can be partially available at the intermediate nodes or even the destination nodes. Such a system is in general more robust as a single point of failure is avoided. Another question is how NNC would perform with constraints imposed on computational power. In our experiments, the communication power per link is fixed and the computation power depends on the NN structure. In our experiments, we did not put an upper bound on the computational power. This is in general a reasonable assumption, as computation with NN is of low complexity. However, in cases where edge devices are extremely constrained in power, things might be different if we link the problem to computation constraints: the size of the NN at the node is limited and the performance might be undermined as a result. In addition, in all our experiments, the physical topology of the communication system is fixed. Further study are needed to understand if and how NNC can adapt into a communication system with an uncertainty of the topology. Furthermore, we only studied the trade-off between the energy available and NNC performance in this chapter. It remains to study if a similar trade-off exists between the bandwidth available, i.e., the number of parallel channels, and NNC performance. Last but not the least, we took an analytical point of view in this chapter and studied NNC performance. It would be worthwhile to explore the design aspect: how we can design a wireless system such that it would naturally exploits the findings provided by the NNC scheme. Some of our initial observations indicate promising directions in the design aspect. For example, we observed that signal magnitude of NNC network codes is Gaussian distributed when the channel is AWGN, which is theoretically optimal. Another example is that for channels with interference, we observed that NNC tends to overlap the signals instead of allocating signals to distinct sub-channels when SNR is low.

2.7 Technical Details

Unless specified otherwise, each NNC policy in Section 2.4 and Section 2.5 is learnt through training over 60000 MNIST training images for 50 epochs, and is tested on 10000 MNIST testing images. Corresponding to each MNIST image, there is a label
ranging from from 0 to 9, representing the digit in the image. The target functions $f_i$'s in functional reconstructions are built from the labels. Note that the training set and the test set are disjoint.

We implemented the NN architecture in Keras [13] with TensorFlow [1] backend. We used Adadelta optimization framework [92] with learning rate of 1.0 and a mini-batch size of 32 samples. In each of our experiment, we learnt a NNC policy with every inner NN set to be two-layer fully-connected with activation function ReLU: $f(x) = x^+ = \max(0, x)$. In the centralized computational scheme of Section 2.4.5, each of the digit recognition functions is also learnt with a two-layer fully-connected with activation function ReLU. Note that the hyper-parameters here may not be optimal, but the results still serve as a proof of concept.
Chapter 3

Coding in Bio-molecular Profiling

This chapter is based on joint work with Mertash Babadi.

3.1 Introduction

In recent years, the field of single-cell biology has witnessed transformative advances in experimental and computational methods. Of particular interest is the recent advent of multiplexed fluorescence in situ (in-place) hybridization (mFISH) microscopy techniques that allow molecular profiling of hundreds of thousands of cells without disturbing their complex arrangement in space. This highly-informative data modality paves the way to transformative progress in many areas of biology, including understanding morphogenesis, tissue regeneration, and disease at molecular resolution.

One of the major challenges in designing such experiments is the vastness of functional bio-molecules. For example, the human genome codes nearly 30k non-redundant types of RNA molecules, many of which translate to proteins with specific functions. Modern data-driven biology heavily relies on our ability to measure as many different types of functional molecules as possible. Clearly, a sequential imaging approach is impractical. Fortunately, a typical cell produces a rather sparse set of all molecules, and some of the most promising mFISH techniques exploit molecular sparsity in space together with coding ideas in order to multiplex the measurements into fewer imaging rounds [12, 62, 61, 74, 28].
In brief, the mFISH technique involves assigning binary codes to RNA molecules of interest, chemically synthesizing and “hybridizing” these codes to the molecules, and measuring them in space one bit at a time via sequential fluorescence microscopy. A more detailed account of one such pioneering technique known as MERFISH (“multiplexed error-robust fluorescence in situ hybridization”) [12] is given in Section 3.1.1 (also, cf. Figure 3-1). An important part of the MERFISH protocol is the utilization of sparse codes with large minimum distance to allow error correction. Referred to as MHD4 codes [12, 62, 61], these 16-bit codes have minimum Hamming distance 4 and contain 4 ones and 12 zeros each. The bit imbalance is motivated by the empirically observed $\sim 2 \times$ higher signal fallout rate compared to false alarm. There are only 140 such codes and therefore one is limited to measuring at most 140 distinct molecules. These codes are randomly assigned to the RNA molecules of interest. The decoding method in current use relies on quantization, Hamming error correction, and rejection of ambiguous sequences.

We point out in [25] that the assumptions motivating the codebook construction and decoding, tacitly yet heavily, rely on source uniformity and to a certain extent on the binary symmetric channel paradigm, both of which are violated in the context molecular profiling. For channel coding in communication, source can be readily assumed as uniformly distributed thanks to compression in source coding and the separation theorem [75]. In molecular profiling, however, source compression is not applicable and the distribution of RNA molecules is extremely non-uniform. Moreover, fluorescence microscopy is established to be highly asymmetric in terms of fallout and false alarm. These violated assumptions become a source of potential problems when directly applying communication encoding and decoding paradigms. For example, the false discovery rate of rare molecules is found to be unacceptably high in replicate experiments [12, 62], which we later show to be a consequence of the assumed source uniformity. Accurate quantification of rare RNA molecules (e.g. transcription factors) is particularly important for data-driven biological discovery since rare molecules often signal rare events, transient cells states, etc. This motivates our primary goal in this chapter: to incorporate the prior non-uniformity in the decoding process in a
principled way in order to control false discovery rate of rare molecules. In practice, either accurate priors are known, can be estimated from the data, or can be measured cheaply and effortlessly (e.g. using bulk RNA sequencing [12]).

The chapter is organized as follows: we review the MERFISH protocol in Section 3.1.1 and propose a generative model for the data in Section 3.2.1, along with a model fitting algorithm and a procedure to derive a more tractable binary asymmetric channel (BAC) formulation from the fitted model. The BAC framework allows us to evaluate the performance of different encoding and decoding schemes. We incorporate the prior non-uniformity into the decoding algorithm by developing a maximum a posteriori (MAP) decoder with a tunable rejection threshold in Section 3.2.4.

We show that the false discovery rate of rare RNAs, which is the key experimental metric, is vastly improved compared to the presently used MLE-based decoding method [12, 62, 61], even when employed with the existing sub-optimal MHD4 codebook. Finally, we take a first step in data-driven code construction in Section 3.3. Using an evolutionary optimization methodology, we show that by permuting the codebook to better align with the prior, which is an experimentally straightforward procedure, significant further improvements are possible. We conclude the chapter in Section 3.4 with follow up research directions.

3.1.1 A brief overview of the MERFISH protocol

In this section, we briefly review the MERFISH protocol [12], recount different sources of noise and nuisance, and motivate a generative process for MERFISH data. Figure 3-1 shows a schematic overview of the MERFISH technique. This protocol consists of four main steps: 

1. **Step 1.** A unique binary codeword of length $L = 16$ is assigned to the RNA molecules of interest;
2. **Step 2.** The specimen is stained with carefully designed short RNA sequences called *encoding probes*. The middle part of the encoding probes bind with high specificity to a single RNA type while their flanking heads and tails contain a subset of $L$ artificial sequences, $\{R_1, \ldots, R_L\}$, called *readout sequences*. The choice of readout sequences reflects the intended binary codeword. For instance, if the code for a certain RNA type contains “1” at positions 1, 3, 5, and 15,
the encoding probes are designed to have $R_1, R_3, R_5,$ and $R_{15}$ flanking sequences (see Figure 3-1a); **Step 3.** The prepared tissue undergoes $L$ rounds of imaging. Imaging round $l$ begins with attaching fluorescent readout probes for round $l$ to the prepared tissue. These probes bind to the flanking readout sequences and contain a fluorescent dye that emits light upon excitation. The round ends with bleaching the dye. In effect, imaging round $l$ reveals the position of all RNA molecules having “1” in their binary codeword at position $l$. **Step 4.** Finally, the position of RNA molecules, which appear as bright spots, are identified using conventional image processing operations (see Figure 3-2). The data is summarized as an $N \times L$ intensity matrix ($N$ being the number of identified spots) and is ultimately decoded according to the codebook.

MERFISH measurements are affected by several independent sources of noise. These include factors that are intrinsic to individual molecules, such as (1) stochasticity in the hybridization of encoding and readout probes, (2) random walk of molecules between imaging rounds, and (3) CCD camera shot noise. These factors module the intensity measurements independently in each round and are largely uncorrelated.
Figure 3-2: Extraction of isolated spots from MERFISH images. Data in this figure is from [62]). (a) local peak finding; (b) identification of isolated spots; (c) intensity series from 10 random spots (rows); the leftmost 16 columns show the intensity measurements; the last two column show the summed intensity and nearest-neighbor cross-correlations and are used for filtering of poorly localized spots.

across rounds. Extreme multiplexing (e.g. as in the seqFISH+ protocol [28]) further leads to interference noise due to signal mixing between nearby molecules. This nuisance, however, is rather negligible in the MERFISH protocol.

3.2 Methodology and Results

3.2.1 A generative model for mFISH data

In this section, we present a simple generative model for MERFISH spot intensity data, fit the model to real data, and evaluate the goodness of fit. This model will serve as a foundation for developing a MAP decoder. Figure 3-2 shows a typical example of MERFISH data from [62]. We formalize the data generating process as follows:

let $\mathcal{C} \subset \{0, 1\}^L$ be a set of codewords with cardinality $|\mathcal{C}| = K$ which are assigned to $G \leq K$ molecules, let $a : \tilde{\mathcal{C}} \to \{1, \ldots, G\}$ be the bijective code assignment map where $\tilde{\mathcal{C}} \subset \mathcal{C}$, $|\tilde{\mathcal{C}}| = G$ is the set of used codes, and let $\pi_{1:G}$ be the prior distribution of molecules. Setting aside interference effects, we model the fluorescence intensity
Figure 3-3: Modeling spot intensities as two-component Gaussian mixture for each round and channel. (a) model fitting (black and red lines) and empirical histograms (gray); the green lines indicate the quantization thresholds for the ensuing BAC approximation; (b) QQ-plots for each data dimension; the labels shown in the sub-panels indicate hybridization rounds \(\{1, \ldots, 8\}\) and color channels \(\{1, 2\}\).

series \(I_{1:L} \in [0, \infty)^L\) measured for an arbitrary molecule as follows:

\[
g \sim \text{Categorical}(\pi),
\]

\[
c = a^{-1}(g),
\]

\[
\log I_l | c_l \sim \mathcal{N}(\mu_l[c_l], \sigma_l^2[c_l]).
\]

As discussed earlier, the intrinsic spot intensity noise (1) is multiplicative, (2) results from a multitude of independent sources, and (3) is uncorrelated across imaging rounds, motivating factorizing \(I_{1:L} | c_{1:L}\) in \(l\) and modeling each conditional as a Gaussian in the logarithmic space. The well-known heteroscedasticity of fluorescence noise is reflected in having two different \(\sigma^2[c]\) for \(c \in \{0, 1\}\) for the two binary states.

### 3.2.2 Image processing and model fitting

The most straightforward way to fit the generative model to empirical data is by observing that marginalizing the (discrete) molecule identity variable \(g\) yields a two-component Gaussian mixture model (GMM) for \(\log I_l\), with weights determined by the
prior $\pi$, codebook $C$, and the assignment $a$. The model parameters \{$\mu_{1,l}[0], \sigma^2_{1,l}[0], \\
\mu_{1,l}[1], \sigma^2_{1,l}[1]$\} can be readily estimated by ML GMM fitting to each column of the spot intensity table (cf. Figure 3-1c), which can be performed efficiently using the conventional EM algorithm. In order to decouple the intrinsic and extrinsic spot noise in the raw data, we censor the dataset to only spatially isolated molecules. In brief, we process the images as described in [12], subtract the background, identify the position of molecules by local peak-finding, censor dense regions (e.g. cell nuclei), and retain local peaks that are separated from one another at least by $\sim 5$ px, which is a few multiples of the diffraction limit. We perform additional filtering based on the spot intensity pattern and nearest-neighbor Pearson correlation (cf. Figure 3-2) and only retain peaks with a symmetric appearance. This procedure yields $\sim 250k$ spots in the dataset published in [62]. The obtained fits are shown in Figure 3-3 along with QQ-plots that confirm a remarkably good fit to the empirical marginal histograms.

### 3.2.3 Quantization, channel model and estimation

The generative model specified by (3.1) readily yields the posterior distribution $\Pr(g | I; \pi, C, a)$ and can form the basis of an intensity-based MAP decoder. To make the formulation more amenable for computational and theoretical investigation, as well as making a connection to the currently used decoding method, we derive an approximate binary asymmetric channel (BAC) model from (3.1) through quantization. The optimal quantization thresholds $\theta_{1,l}$ are determined for each $l$ to be the point of equal responsibility between the two Gaussian components, i.e. $\sum_{g=1}^{G} \pi_g a^{-1}(g)[l] N(\theta_l | \mu_{l}[1], \sigma^2_{l}[1]) = \sum_{g=1}^{G} \pi_g [1 - a^{-1}(g)[l] N(\theta_l | \mu_{l}[0], \sigma^2_{l}[0])]$, which admits a closed-form solution. Here, $a$ and $\pi$ correspond to the known code assignment and prior distribution of the data used for fitting. The fallout $p^{1\rightarrow 0}$ and false alarm $p^{0\rightarrow 1}$ rates are given by the integrated probability weights of the two Gaussian components below and above the threshold (cf. Figure 3-3a), i.e. $p^{0\rightarrow 1} = \Phi[(\mu_{l}[0] - \theta_l)/\sigma_{l}[0]]$ and $p^{1\rightarrow 0} = \Phi[(\theta_l - \mu_{l}[1])/\sigma_{l}[1]]$, where $\Phi(\cdot)$ is the CDF of the standard normal distribution. We find $p^{0\rightarrow 1}$ and $p^{1\rightarrow 0}$ to be 0.046 and 0.102 (mean in $l$), respectively, for the data given in [62], which is in agreement with the estimates.
reported therein. We, however, observed significant round-to-round variation in the channel parameters and as such, refrained from further simplifying the channel model to a single BAC for all imaging rounds $l$. We refer to the bundle of estimated BAC parameters as $\theta_{\text{BAC}}$.

### 3.2.4 Decoding: MAP and MLE decoders

A gratifying property of the BAC approximation of (3.1) is allowing us to evaluate the performance of various decoding strategies without resorting to time-consuming simulations or further analytical approximations. In the BAC model, the likelihood of a binary sequence $x_{1:L} \in \{0, 1\}^L$ conditioned on the codeword $c \in C$ is given as:

$$
\log \Pr(x \mid c, \theta_{\text{BAC}}) = \sum_{l=1}^{L} \sum_{i,j \in \{0,1\}} \delta_{c_l,i} \delta_{x_l,j} \log p_{l}^{i \rightarrow j},
$$

where $\delta_{\cdot,\cdot}$ is the Kronecker’s delta function. We define the posterior Voronoi set for each codeword $c \in C$ as:

$$
V(c \mid a, \omega, C, \theta_{\text{BAC}}) = \{x \in \{0, 1\}^L \mid \forall c' \in C, c \neq c' : \omega_{a(c)} \Pr(x \mid c, \theta_{\text{BAC}}) > \omega_{a(c')} \Pr(x \mid c', \theta_{\text{BAC}})\},
$$

where $\omega_{1:G}$ is the prior distribution assumed by the decoder. The Voronoi sets are mutually exclusive by construction, can be obtained quickly for short codes by exhaustive enumeration, and determine the optimal codeword for an observed binary sequence. The MLE decoder corresponds to using a uniform prior, i.e. $\omega \leftarrow 1/G$ whereas the MAP decoder corresponds to using the actual (non-uniform) prior governing the data $\omega \leftarrow \pi$. We additionally introduce a MAP$_q$ decoder, which is a MAP decoder obtained from depleting the Voronoi sets from binary sequences for which the posterior probability of the best candidate code is below a set threshold $q$. Intuitively, the MAP$_q$ decoder is a Bayesian decoder with reject option that trades precision gain for sensitivity loss by filtering dubious sequences from the Voronoi sets. The decoding algorithm introduced by [12, 62, 61] can be thought of as the MLE decoder with a rejection
subspace given by $S_{\text{rej}} = \{ x | \exists c, c', c \neq c' \in C : d_H(c, x) = d_H(c', x) = d^*(x, C) \}$
where $d_H(\cdot, \cdot)$ is the Hamming distance and $d^*(x, C) = \inf_{c \in C} d_H(c, x)$. We refer to this decoder as Moffitt (2016). We remark that the acceptance criterion of Moffitt (2016) is extremely stringent: for MHD4 codes, $|S_{\text{acc}}| = 9100$, which is only $\sim 13\%$ of all possible sequences (here, $S_{\text{acc}}$ is the complement of $S_{\text{rej}}$). In all cases, the confusion matrix $\mathcal{T}(c | c')$, i.e. the probability that a molecule coded with $c'$ is decoded to $c$, can be immediately calculated:

$$
\mathcal{T}(c | c'; \pi, \omega, \theta_{\text{BAC}}) = \sum_{x \in V(\omega,...)} \pi_a(c') \Pr(x | c', \theta_{\text{BAC}})
$$

from which the marginal true positive rates $\text{TPR}_{1:\text{G}}$ and false discovery rates $\text{FDR}_{1:\text{G}}$ can be readily calculated.

### 3.2.5 Comparing the performance of different decoders

Developments in previous sections allow us to compare the performance of MLE, MAP, $\text{MAP}_q$, and Moffitt (2016). We use the BAC parameters obtained from the data in [62], 16-bit MHD4 codes with random assignment, and two different previously estimated and published source priors with different degree of non-uniformity. As a first step, we compare the performance of our proposed MAP and MLE decoders separately inside $S_{\text{acc}}$ and $S_{\text{rej}}$, the acceptance and rejection subspaces of Moffitt (2016), in Figure 3-4a, b (middle, bottom). The priors are shown on the top, including the estimated Dirichlet concentration $\alpha$. Marginal performance metrics for different molecules are color-coded according to their prior rank from red (most abundant) to blue (least abundant). The MLE decoder inside $S_{\text{acc}}$ is equivalent to Moffitt (2016). Both decoders perform well in this subspace. While the MLE decoder is performing poorly inside $S_{\text{rej}}$, providing a sound basis for rejection as in Moffitt (2016), the MAP decoder yields acceptable FDR, hinting that the $S_{\text{acc}}$ is too stringent for the MAP decoder and better performance can be expected from $\text{MAP}_q$. It also noticed that MAP decoder controls FDR much better than MLE inside $S_{\text{acc}}$ for the more non-uniform prior. We explore this observation more systematically in panel c. We
Figure 3-4: Comparing the performance of different decoding schemes for randomly assigned MHD4 codes. (a) and (b) correspond to prior distribution for RNA molecules selected in [62] and [61], respectively. The top panels show the rank-ordered prior distribution and the estimated Dirichlet concentration parameter $\alpha$; the middle and bottom panels show the marginal TPR and FDR for each molecule type conditioned on $S_{\text{acc}}$ and $S_{\text{rej}}$ subspaces (cf. Section 3.2.4); markers are color-coded according to prior rank of their corresponding molecules. Shaded regions indicate 5-95 percentile range as a matter of random code assignment; (c) the effect of prior non-uniformity on the performance of MLE and MAP decoders for randomly assigned MHD4 codes. The top panels show the uniform mismatch rate. The bottom panels show the histogram of marginal FDRs vs. Dirichlet prior concentration $\alpha$ in grayscale. The orange lines and regions indicate the median and 5-95 percentile ranges; (d) performance of MAP decoders with reject at different acceptance thresholds compared to method in [62].
sample $\pi$ from a symmetric Dirichlet distribution with concentration $\alpha$ and calculate the distribution of the marginal FDRs (bottom) as well as the uniform mismatch rate (top). We notice that as the prior gets more concentrated $\log \alpha \to -\infty$, the MAP decoder behaves progressively better whereas the MLE decoder degrades and exhibits a bi-modal behavior: extremely low (high) FDR on abundant (rare) codes. As the prior gets more uniform $\log \alpha \to +\infty$, MLE and MAP become indistinguishable. The green and red symbols show the biological priors used in panels a and b, respectively, together with their estimated $\alpha$, in agreement with the trend of the Dirichlet prior model. Finally, panel d compares the performance of the MAP$_q$ decoder at different rejection thresholds $q$ with Moffitt (2016). The prior used here is the same as in panel b. It is noticed that the MAP$_q$ decoder is remarkably effective at controlling FDR for all codes whereas Moffitt (2016) degrades in FDR for rare codes, as expected from the source uniformity assumption. This finding explains the reportedly lower correlation between rare molecules in replicate experiments [12, 62]. The smaller panels at the top of panel c show mean TPR, FDR, and rejection rate across all molecules. The MAP$_{0.5}$ decoder has similar sensitivity to Moffitt (2016) while yielding $\sim 20\%$ lower FDR on average and remarkably $\sim 60\%$ lower 5-95 FDR percentile range, implying significant improvement in reducing the mean and variance of false positives for both abundant and rare molecules.

### 3.3 Data-driven code construction

The results presented so far were obtained randomly assigning a fixed set of MHD4 codes. Constructing codes to better reflect channel asymmetry and prior non-uniformity is another attractive opportunity for improving the performance of mFISH protocols. Constructing application-specific codes for mFISH is outside the scope of the chapter and is a topic for future research. Here, we continue to thread on the theme of utilizing prior non-uniformity and show that optimizing the assignment of the even sub-optimal codes to molecules with respect to prior abundance can significantly reduce FDR. This is to be expected given the rather wide performance outcomes shown
in Figure 3-4 that result from random code assignment. Explicitly, we seek to optimize the scalarized metric $FDR(a, \pi) = G^{-1} \sum_{g=1}^{G} FDR_g(a, \pi)$ over the assignment operator $a$ for a given prior $\pi$ through an evolutionary optimization process. We start with a population of $N = 5000$ random code assignments, mutate the population via pairwise permutations with a small probability of 0.05 per molecule per assignment, and select the fittest $N$ offsprings using $FDR$ as the measure of fitness. We do not use a crossover operation here. We hypothesize that a relevant surrogate for the optimality of $FDR$ is the concordance between the Hamming distance $d_H$ and the prior distance $d_\pi(c, c') \equiv |\pi_{a(c)} - \pi_{a(c')}|$. We investigate the emergence of this order by monitoring the following order parameter during the evolution:

$$\chi(a, \pi) \equiv \frac{1}{G} \sum_{g=1}^{G} \rho_s\left[d_H(a^{-1}(g), C_a), d_\pi(a^{-1}(g), C_a)\right], \quad (3.5)$$

where $\rho_s[\cdot, \cdot]$ denotes the Spearman correlation and $C_a$ is the ordered list of all codes used by $a$ over which the correlation is calculated. We refer to the population average of $\chi(a, \pi)$ as $\overline{\chi}$. We implement the evolutionary algorithm using the PyMOO package [9] and vectorize the calculation of Voronoi sets with GPU acceleration. Figure 3-5 shows the results obtained by running the evolutionary optimization for three days (NVIDIA Tesla P100 GPU, MHD4 codes, prior from [61]). Panel a shows the monotonic decline of $\overline{FDR}$ to $\sim 75\%$ of its initial value (random assignment). This trend proceeds concurrently with a monotonic upturn in $\overline{\chi}$, providing evidence for the hypothesized matching order between $d_H$ and $d_\pi$. Panel b compares the performance metrics of the MAP decoder between the first and last population of code assignments. It is noticed that the optimized code assignment predominantly reduces $FDR$ of rare molecules, the mean $FDR$ of which reduce to $\sim 50\%$ of randomly assigned codes. The possibility to reduce the $FDR$ of rare molecules is a particularly favorable outcome in practice.
Figure 3-5: Evolutionary optimization of code assignment for MHD4 codes. Channel model is described in Figure 3-3 and prior distribution is from [61]). (a) bottom: mean FDR vs. generation; top: $d_H - d_\pi$ matching order parameter vs. generation; (b) the performance of MAP decoder for randomly assigned codes (squares) vs. optimized assignment (circles).
3.4 Discussion

In this chapter, we reviewed multiplexed molecular profiling experiments from the perspective of coding theory, proposed a motivated generative model for the data, based on which we derived an approximate parallel BAC model for the system. We show that the exact MAP decoder of the BAC model vastly outperforms the decoding algorithm in current use in terms of controlling FDR of rare molecules, the key experimental metric. This is achieved by taking into account the non-uniformity of source prior, a “non-classical” aspect of multiplexed molecular profiling viewed as a noisy channel. We also took the first step in data-driven code construction and show that optimizing the assignment of existing sub-optimal codes is another effective method for reducing false positives.

Attractive directions for follow up research include constructing application-specific codes to increase the throughput of the mFISH experiments, theoretical progress in understanding the optimal assignment of existing codes (e.g. by investigating the geometry of Voronoi sets), extending the generative model and the ensuing channel description to $q$-ary codes (e.g. as in seqFISH and seqFISH+ experimental protocols [74, 28]), and taking into account spatial interference and color channel cross-talk in the data generating process.
Chapter 4

Guesswork in Machine Translation

4.1 Introduction

Machine translation has been a popular problem because of its practical importance and the intrinsic challenges it imposes. The learning problem involves searching through an intractably large space of candidate translations. The problem has a one-to-many nature, as multiple valid translations may exist for one sentence [65]. The sequence-to-sequence model [80] is widely used in machine translation today. The model is built on the Recurrent Neural Network (RNN) [71, 87] and consists of an encoding phase and a decoding phase. Fig 4-1 illustrates a seq2seq model with RNN unfolded in time. In the encoding phase, the sentence to be translated is fed to the seq2seq model one word each step. In the decoding phase, one word is selected at each step and fed back into the RNN to generate a distribution of the next word over the vocabulary of the output language. The decoding phase of a seq2seq model breaking down the inference into steps: Instead of inferring a sentence as a whole out of an intractably large number of possible sentences, seq2seq model infers a word at a step from the output vocabulary. This allows inference algorithms to search through the huge output space in a guided manner.

Of existing inference mechanisms, beam search is commonly used. At each step in the decoding part of the seq2seq model, beam search examines all possible one-step extension from current branches, among them only the top $k$ branches ($k \geq 1$) with
the largest probability are kept into next step and the rest are pruned. Beam search is greedy. With beam size $k = 1$, beam search reduces to maximum likelihood inference at each step. A total number of $k$ candidate translations are generated in the process, among which the translation with the largest probability is selected. If the selected translation fails to be a valid process, the model fails that specific experiment.

To predetermine the beam size $k$ is crucial in evaluating the performance of seq2seq model in machine translation. If $k$ is too large, inference in decoding can be prohibitively expensive and practically infeasible. If $k$ is too small, inference in the decoding part is not likely to cover sufficient amount of probability space. As a result, the model can easily fail the experiments even if the model is well trained. This is particularly critical since the output probability space of a well-trained seq2seq model is not concentrated as demonstrated in [65]. There are both intrinsic and extrinsic reasons. The intrinsic uncertainty of machine translation problem roots from its multi-model nature, as multiple correct translations may exist for a single sentence. The extrinsic uncertainty, which [65] suggests a fix on, is because of a mixed source of translation due to lack of data in practice. Note that incrementally testing the appropriate beam size is computationally inefficient. As a greedy algorithm, little information can be carried over from an experiment with a smaller beam size to one with a larger beam size: the computation in previous experiments will be largely wasted. A natural question to ask is: how do we predetermine the beam size $k$? To the best of our knowledge, no theoretical guidance exists except our initial study in [53].

In this chapter, we use guesswork [60], an information theoretical concept, in analyzing machine translation with seq2seq model. We provide a framework to quantify the uncertainty in machine learning and to provide insights in the selection of beam size $k$. Given a pre-trained seq2seq model, the decoding part is a realization of a sequence of random variables. This is in line with the guesswork problem, which has been extensively studied in information theory. Previous results on guesswork, such as the large deviation principle, can be applied to derive a theoretical upper bound for the accuracy of the model with a certain beam size $k$. Literature on guesswork
started with [60] on the problem of guessing a realization of a random variable $X$ by asking the question of “Is $X$ equal to $x$?” until the answer is “Yes”. Guesswork for a sequence of independent identically distributed (henceforth, i.i.d.) random variables is studied in [3]. The result is extended to the Markov case in [58] and other cases in [60], [36]. It is proved in [14] that the guesswork of a stochastic process satisfies a large deviation principle under certain regularity conditions. Guessing under distortion and source uncertainty is studied in [5, 35, 79]. Other extensions include guessing in multi-user scenario [15], and guessing with limited memory [38].

The rest of this chapter is organized as follows: Section 4.2 introduces the seq2seq model, and presents our observations of the model; Section 4.3 derives a theoretical upper bound for probability space covered by beam search with beam size $k$, compares the bound with existing bounds analytically and numerically, and extends the bound to the case when multiple correct answers are available; Section 4.4 discusses a possible extension to accepting distorted translation, and presents the technical difficulty of practically applying the results to a seq2seq model. Symbols and notations used in this chapter is listed in Table 4.1.

### 4.2 Study on seq2seq model

An example of machine translation using the seq2seq model is shown in Fig 4-1, where a French sentence $x_0, x_1, x_2$ is translated into an English sentence $y_0, y_1, y_2, y_3$. Each $z_n$ is a vector representing the hidden state of the model which records all previous actions of the system before time $n$. To be more precise, for the encoding part, each $z_n$ ($n < 0$) records input words up to time $n$. For the decoding part, each $z_n$ ($n \geq 0$) records the whole input sentence as well as the output words up to time $n$.

We are interested in the decoding part of the model, with all information of the source sentence already encoded in state $z_0$. In each step $n$, a distribution over English vocabulary is generated from the hidden state $z_n$ capturing $p(y_n \mid z_0, y_0^{n-1})$. Here, we denote by $y_0^{n-1}$ the sequence $y_0, \ldots, y_{n-1}$. The mapping can be represented as a
Table 4.1: Symbols and notation in Chapter 4

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k$</td>
<td>Number of queries</td>
</tr>
<tr>
<td>$N$</td>
<td>Length of output sequence</td>
</tr>
<tr>
<td>$n$</td>
<td>Index of sequence</td>
</tr>
<tr>
<td>$z_n$</td>
<td>$n_{th}$ hidden state</td>
</tr>
<tr>
<td>$x_n$, $y_n$</td>
<td>$n_{th}$ input; output word</td>
</tr>
<tr>
<td>$x_0^{N-1}$</td>
<td>Sequence to be translated</td>
</tr>
<tr>
<td>$y_0^{N-1}$</td>
<td>Output sequence of machine translation</td>
</tr>
<tr>
<td>$P^N$</td>
<td>Joint distribution of sequence $y_0^{N-1}$</td>
</tr>
<tr>
<td>$</td>
<td>Z</td>
</tr>
<tr>
<td>$</td>
<td>X</td>
</tr>
<tr>
<td>$M$</td>
<td>Transition matrix for Markov pair ${z_n, y_{n-1}}$</td>
</tr>
<tr>
<td>$H_\alpha$</td>
<td>$\alpha$-order Rényi entropy rate of a sequence</td>
</tr>
<tr>
<td>$H_\alpha(y_0^{N-1})$</td>
<td>$\alpha$-order Rényi entropy of the sequence $y_0^{N-1}$</td>
</tr>
<tr>
<td>$H$</td>
<td>Shannon entropy rate of a sequence</td>
</tr>
<tr>
<td>$H(y_0^{N-1})$</td>
<td>Shannon entropy of the sequence $y_0^{N-1}$</td>
</tr>
<tr>
<td>$G(X)$</td>
<td>Guesswork of random variable $X$</td>
</tr>
<tr>
<td>$G_{opt}(X)$</td>
<td>Optimal Guesswork of random variable $X$</td>
</tr>
<tr>
<td>$V$</td>
<td>Number of correct answers</td>
</tr>
</tbody>
</table>

Figure 4-1: The seq2seq model for machine translation.

deterministic function $f_1$ as

$$f_1(y_n, z_n) = p(y_n | z_n) = p(y_n | z_0, y_0^{n-1}).$$  \hspace{1cm} (4.1)

A word $y_n$ is sampled from the distribution, and the values of $z_n$ and $y_n$ jointly determine the value of the next hidden state $z_{n+1}$. In other words, there exists a
deterministic function \( f_2 \), which maps the pair \((z_n, y_n)\) to \(z_{n+1}\):

\[
z_{n+1} = f_2(z_n, y_n). \tag{4.2}
\]

Overall, the probability of translating \( x_0^N \) into \( y_0^N \) can be inferred from \( z_0 \), and de-composed to each decoding step as follows:

\[
p(y_0^N \mid x_0^N) = p(y_0^N \mid z_0) = \prod_{n=0}^{N} p(y_n \mid z_0, y_{0}^{n-1}). \tag{4.3}
\]

We have the following important observations:

\( \{z_n\} \) **is Markov.** The value of \( z_n \) is determined by the value of \( z_{n-1} \) and the value of \( y_{n-1} \), whereas the distribution of \( y_{n-1} \) is determined by the value of \( z_{n-1} \). Therefore, the distribution of \( z_n \) solely depends on \( z_{n-1} \) and is independent of the values of all previous hidden states.

\( \{y_n\} \) **is not Markov.** Given \( z_0 \), the sequence \( \{y_n\} \) is not Markov. This can be shown by analyzing \( y_2 \). The distribution of \( y_2 \) is determined by the value of \( z_2 \), which is jointly decided by the values of \( y_1 \) and \( z_1 \). The value of \( z_1 \) is jointly decided by the values of \( y_0 \) and \( z_0 \). Therefore, the distribution of \( y_2 \) depends on the value of \( y_0 \), and \( \{y_n\} \) is not Markov.

\( \{z_n, y_{n-1}\} \) **is Markov.** Given \( z_n \), the distribution of \( y_n \) is fixed and the value of \( z_{n+1} \) is a function of the value of \( y_n \). In other words, for any \( z, y \)

\[
p\left(\{z_{n+1}, y_n\} = \{ z, y \} \mid z_0, \{ z_1, y_0 \} \ldots, \{ z_n, y_{n-1} \} \right)
\]

\[
= p\left(\{z_{n+1}, y_n\} = \{ z, y \} \mid z_n \right)
\]

\[
= \begin{cases} 
p(y_n = y \mid z_n) = f_1(z_n, y), & \text{if } z = f_2(z_n, y), \\
0, & \text{if } z \neq f_2(z_n, y). \end{cases} \tag{4.4}
\]

Moreover, \( \{z_n, y_{n-1}\} \) is a homogeneous Markov chain since both \( f_1 \) and \( f_2 \) are independent of the time index \( n \). Most likely, the Markov chain is not irreducible: If our input sentence is about chemical engineering, it is very unlikely that the Markov
chain will take us to classical music! In that case, we can always reduce the Markov chain to irreducible parts when required. Henceforth, we restrict our discussion to irreducible Markov chains. The transition matrix $M = (M_{ab})$ for the Markov chain is of size $|Z||Y| \times |Z||Y|$ and the initial state is decided by $z_0$.

4.3 Guesswork and its application here

Let $G(x)$ denote the number of guesses required by the guessing strategy $G$ when $X = x$. Massey observed in [60] that $\mathbb{E}[G(X)]$, the average number of guesses, is minimized by a guessing strategy $G_{opt}$ which guesses the possible values of $X$ in decreasing order of probability. If we model the translation of an input sentence as a realization of a sequence of random variables $y^N_0$ with joint distribution as in (4.3), then decoding using an iterative inquiry algorithm with multi-shot inference criterion is the same model as guesswork.

Recall that the $\alpha$-order Rényi entropy rate for a random sequence $y^N_0$ with joint distribution $P^N(y^N_0)$ is defined as

$$H_\alpha = \lim_{N \to \infty} \frac{1}{N} H_\alpha(y^N_0) = \lim_{N \to \infty} \frac{1}{1 - \alpha} \log \sum_{y^N_0} P^N(y^N_0)^\alpha.$$ (4.5)

If $y^N_0$ is an irreducible Markovian process with transition matrix $M$, it is shown in [67] that $H_\alpha$ has the following explicit formula:

$$H_\alpha = \frac{1}{1 - \alpha} \log \lambda_\alpha,$$ (4.6)

where $\lambda_\alpha$ is the largest eigenvalue of the matrix $M^\alpha$ with entries $M^\alpha_{ab}$.

Based on the Markovian property of the seq2seq model and previous results on guesswork, we have:

**Theorem 1.** Given a seq2seq model with transition matrix $M$ for the sequence of pairs $\{z_n, y_{n-1}\}$, the probability space covered by beam search with beam size $k$ is at
most
\[ \sum_{i=1}^{k} p(G_{opt}(y_0^{N-1} \mid z_0) = i), \]
\[ \approx \sum_{i=1}^{k} \frac{1}{i} \exp(-N \sup_{\alpha > -1} \{ \alpha N^{-1} \log i - (1 + \alpha) \log \lambda \frac{1}{1+\alpha} \}) \] (4.7)

where \( \lambda \frac{1}{1+\alpha} \) is the largest eigenvalue of the matrix \( M^{\frac{1}{1+\alpha}} \) with entries \( M_{ab}^{\frac{1}{1+\alpha}} \).

**Proof.** Note that from (4.4)
\[ p(z_1^N, y_0^{N-1} \mid z_0) = \begin{cases} p(y_0^{N-1} \mid z_0) & \text{if } \forall n, z_{n+1} = f_2(z_n, y_n) \\ 0 & \text{otherwise.} \end{cases} \] (4.8)

The non-zero terms of the joint distribution of \( z_1^N, y_0^{N-1} \) match the marginal distribution of \( y_0^{N-1} \). Thus, under the optimal strategy, the guesswork for guessing the sequence of pairs \( \{z_n, y_{n-1}\} \) is the same as guessing the sequence of words \( \{y_n\} \), i.e.,
\[ G_{opt}^{\alpha}(z_1^N, y_0^{N-1} \mid z_0) = G_{opt}^{\alpha}(y_0^{N-1} \mid z_0). \] (4.9)

For all irreducible Markov chains \([58]\), a large deviation principle-based tool is provided in \([13]\) to estimate the distribution of the optimal guesswork of a random sequence \( u_0^N \):
\[ p(G_{opt}(u_0^N) = i) \approx \frac{1}{i} \exp(-N \Lambda^*(N^{-1} \log i)), \] (4.10)
where \( \Lambda^* \) is the Legendre-Fenchel transform of the scaled cumulant generating function \( \Lambda \) of the sequence \( u_0^N \):
\[ \Lambda(\alpha) = \lim_{N \to \infty} \frac{1}{N} \log \mathbb{E}(e^{\alpha \log G_{opt}(u_0^N)}) \]
\[ = \begin{cases} \alpha H \frac{1}{1+\alpha} & \text{if } \alpha > -1 \\ \lim_{\beta \downarrow -1} \Lambda(\beta) & \text{otherwise} \end{cases} \] (4.11)
\[ \Lambda^*(x) = \sup_{\alpha \in \mathbb{R}} \{ x \alpha - \Lambda(\alpha) \} \]
where $H_{\frac{1}{1+\alpha}}$ is the $\frac{1}{1+\alpha}$-order Rényi entropy rate of the sequence.

Since $N^{-1}\log i$ is positive and $\Lambda(\alpha)$ is constant on $\alpha \in (-\infty, -1]$, $x\alpha - \Lambda(\alpha)$ is monotonically increasing on $\alpha \in (-\infty, -1]$. Therefore,

$$
\Lambda^*(x) = \sup_{\alpha > -1} \{x\alpha - \Lambda(\alpha)\} = \sup_{\alpha > -1} \{x\alpha - \alpha H_{\frac{1}{1+\alpha}}\}.
$$

(4.12)

Theorem 1 follows from (4.6), (4.12), and (4.10).

Note that (4.7) gives an estimate of the total mass on the top-$k$ values of the guesswork. While the approximation is tight when $N \to \infty$, authors in [14] have shown that it is still a good approximation for $N \geq 15$, which is a reasonable sentence length for machine translation. Theorem 1 is an upper bound on the probability of hitting the correct answer within beam size $k$, since the top-$k$ sequences might be trimmed in previous steps and (4.10) is decreasing with $i$.

Theorem 1 quantitatively estimates the distribution by deriving an upper bound for the probability mass covered by the translation of seq2seq model using beam search with beam size $k$. For example, consider a binary Markov sequence of length 13 with symmetric transition probability $p(1 | 0) = p(0 | 1) = 0.31$. Fig 4-2 shows the upper bound of probability mass covered by beam search with beam size ranging from 1 to 100. Note that the example is chosen to match the order of magnitude of the output distribution in [65] of a well-trained English to French machine translation model using seq2seq model. In [65], beam search with $k = 5$ and $k = 200$ covers a probability mass of 14.6% and 22.4%, respectively. As beam search may not end up with the optimal sequences, the probability mass on the top 5 sequences and top 200 sequences are lower bounded by the corresponding values. In the proof of concept example we choose, the top 5 and top 200 sequences have a total probability mass of 1.9% and 28.1%, respectively. For further statistics of a practical seq2seq model, we refer the reader to Fig 1 (left) in [65]. However, we cannot extract sufficient statistics from [65] or other experiments due to the technical difficulty of computing the Rényi entropy for a practical seq2seq model, which will be discussed later in Section 4.4.

We are primarily interested in decoding given the real distribution. The training
Figure 4-2: The approximate upper bound in Theorem 1 for the probability mass covered by beam search with varying beam size. The sequence of interest is a binary Markov sequence of length 13 with transition probability $p(1 \mid 0) = p(0 \mid 1) = 0.31$. The dashed and point lines are the experimental results from [65] indicating the probability mass spanned by beam search with $k = 5$ and $k = 200$ for a practical machine translation model using seq2seq model.

phase is a separate task which is not discussed in the current paper. Since our approach is based on guesswork, it yields better performance when the distribution is skewed away from uniform. However, this is against the generalization theory of machine learning [7] which aims to guide the training towards the real underlying distribution and makes as few assumptions as possible on the unknown structure. In other words, the more uniform the distribution is, the more it is preferred in the training process.

4.3.1 Comparison with the existing bounds

We may compare the bounds given by (4.7) with Theorem 9 and Theorem 10 in [73], which bound the error rate of list decoding by the Rényi entropy of negative order, and the Rényi entropy of order greater than 1. We rewrite Theorem 9 and Theorem 10 as (4.13) and (4.14) using our notation here.

73
For all $\beta \in (-\infty, 0)$, Theorem 9 in [73] gives:

$$
\sum_{i=1}^{k} P(G(y_0^{N-1} \mid z_0) = i) \leq 1 - \exp\left(\frac{1 - \beta}{\beta} [H_{\beta}(y_0^{N-1} \mid z_0) - \log(|Y|^N - k)]\right).$$

(4.13)

For all $\beta \in (1, \infty)$, Theorem 10 in [73] gives:

$$
\sum_{i=1}^{k} P(G(y_0^{N-1} \mid z_0) = i) \leq \exp\left(\frac{1 - \beta}{\beta} [H_{\beta}(y_0^{N-1} \mid z_0) - \log k]\right).
$$

(4.14)

We plot in Fig 4-3 the bounds for the probability of hitting the correct answer within 5 queries when the sequence is binary Markov with length $N = 30$, and has a symmetric transition probability with $p(1 \mid 0) = p(0 \mid 1)$, where $p(1 \mid 0)$ is ranging from 0.05 to 0.5 and the y-axis is of log scale. Note that each of (4.13) and (4.14) gives a set of bounds for different values of $\beta$ for a specific symmetric transition probability. Here, we plot the inf of each set over the possible range of $\beta$. From the plot, we can tell that the bounds are the same when the sequence is completely random, i.e., $p(1 \mid 0) = p(0 \mid 1) = 0.5$. In other cases, the inf of the bounds given by (4.13) is significantly worse than the other two; our bound given by (4.7) and the inf of the bounds given by (4.14) are numerically comparable, but the latter is slightly better.

We will next extend our bound to general non-Markov case and analytically prove that the inf of the bounds in (4.14) is, in general, a lower bound of our bound (See Theorem 2). However, our bound can be easily extended to the scenarios when we have multiple correct translations and when a slightly distorted version of a reference translation can be accepted, which makes our bound more versatile.
Numerical comparison of the upper bounds for the probability mass covered in beam search Beam size \( k = 5 \) and the sequence of interest is a binary Markov sequence of length 30 with symmetric transition probability ranging from 0 to 0.5.

Analytical comparison with existing bounds

The analog of Theorem 1 holds in general non-Markov case. Here we compare the analog extension of Theorem 1 with the inf of the bounds given in (4.14). The result also holds in Markov case. From (4.10), (4.11), (4.12), the general extension of Theorem 1 can be written as

\[
F_k = \sum_{i=1}^{k} \frac{1}{i} \exp(-N \sup_{\alpha > -1} \{\alpha N^{-1} \log i - \alpha H_{1+\alpha}\}).
\]

Substituting \( \beta \) with \( \frac{1}{1+\alpha} \), the inf of the family of the bounds given by (4.14) can be written as

\[
G_k = \exp(-N \sup_{0>\alpha>-1} \{\alpha N^{-1} \log k - \alpha H_{1+\alpha}\}).
\]

Let \( h_i = \frac{1}{i} \exp(-N \sup_{\alpha > -1} \{\alpha N^{-1} \log i - \alpha H_{1+\alpha}\}) \) and \( h_i' = \frac{1}{i} \exp(-N \sup_{0>\alpha>-1} \{\alpha N^{-1} \log i - \alpha H_{1+\alpha}\}) \), we have \( F_k = \sum_{i=1}^{k} h_i \) and \( G_k = kh_k' \).

The next result follows from Lemma 1 and shows that our result is an upper bound of the inf of the bounds given by (4.14) in [73].

**Theorem 2.** For fixed \( N \), we have \( F_k > G_k \) for \( \forall k \).

**Lemma 1.** Define \( z_i(\alpha) = \alpha N^{-1} \log i - \alpha H_{1+\alpha}, \alpha > -1 \). Let \( \alpha_i^* = \arg \sup z_i(\alpha) \). For given \( N \), \( \alpha_i^* \) is increasing with \( i \).
Proof. Since \( \Lambda(\alpha) \) is a scaled cumulant generating function, its second derivative is a variance. It follows that \( \Lambda(\alpha) \) is monotonically increasing and convex. Consider

\[
\frac{dz_i(\alpha)}{d\alpha} = \frac{\log i}{N} - \frac{d\Lambda(\alpha)}{d\alpha}.
\]

(4.15)

Note that both \( \frac{\log i}{N} \) and \( \frac{d\Lambda(\alpha)}{d\alpha} \) are positive. Also \( \frac{d\Lambda(\alpha)}{d\alpha} \) is increasing with \( \alpha \). \( \arg \sup z_i(\alpha) \) is the zero point of (4.15) if there is a zero point on \((-1, \infty)\). Otherwise, \( \arg \sup z_i(\alpha) = -1 \). For any given \( N \), \( \alpha_i^* \) is increasing with \( i \).

Lemma 2. If \( \alpha_n^* \in (-1,0) \), then \( \forall m < n \), \( F_{m+1} - F_m \geq G_{m+1} - G_m \).

Proof. By Lemma 1, we have \( \alpha^*_m < \alpha_n^* < 0 \) for \( m < n \). Therefore, \( \forall m < n \), we have \( h'_m = h_m \) and thus \( G_m = mh_m \). \( h_i \) is the estimate of the probability of \( i \)th largest value based on the large deviation principle \([14]\), therefore \( h_i \) is monotonically decreasing with \( i \). Therefore, we have \( F_{m+1} - F_m = h_{m+1} \geq (m+1)h_{m+1} - mh_m = G_{m+1} - G_m \).

Lemma 3. If \( \alpha_n^* > 0 \), then \( \forall m \geq n \), \( F_{m+1} - F_m \geq G_{m+1} - G_m \).

Proof. By Lemma 1, \( \alpha^*_m \geq \alpha^*_n > 0 \) for \( m \geq n \). Therefore, \( \forall m \geq n \), we have \( h'_m = \frac{1}{m} \exp\{-Nz_m(0)\} = \frac{1}{m} \). It follows that \( G_{m+1} - G_m = (m+1)h'_{m+1} - mh'_m = 0 \). But \( F_{m+1} - F_m = h_{m+1} > 0 \). Hence, \( F_{m+1} - F_m > G_{m+1} - G_m \).

When \( k = 1 \), we have \( F_1 = G_1 \). Induction with Lemma 2 and Lemma 3 completes the proof of Theorem 2.

4.3.2 Extension to multiple correct answers

When evaluating machine translation, we usually have a list of correct translations. Assume there are \( V \) correct translations in total. We may model all \( V \) answers as being independently sampled from an underlying distribution in a human expert’s mind \( p_h(y^N_1 \mid x^N_1) \). The modelling is reasonable in the following sense: We can assume that each one of the \( V \) correct answers is the translation for the given sentence by a human expert, and we may model the translation process in the experts’ mind as a sampling process from a list of candidate sentences. If all \( V \) experts have similar training
backgrounds and are not allowed to communicate while translating the sentence, then $V$ answers are selected in an i.i.d fashion. Recall that the goal of machine translation is to imitate human experts. Then we have $p(y_1^N \mid x_1^N) \approx p_h(y_1^N \mid x_1^N)$, where $p(y_1^N \mid x_1^N)$ is the distribution generated from seq2seq model.

To quantify the uncertainty of a machine translation system when multiple valid translations exist, we defined the probability mass covered in beam search with beam size $k$ as the probability of hitting a valid answer within $k$ translations. Note that this is aligned with our analysis in the case of a single valid translation. There we quantify the uncertainty of the system by looking at the probability space covered with beam size $k$, which is essentially the probability of hitting the valid answer within $k$ translations.

Analysis of guesswork with multiple correct answers can be built on the study of parallel guesswork in [15]. In parallel guesswork, an inquisitor is allowed to check the $n_{th}$ most likely string for all $V$ independent users in the $n_{th}$ inquiry, and the parallel guesswork is defined to be the number of inquiries made until the strings of $U$ users are correctly found. Based on our analysis above, we can observe that guesswork for machine translation with $V$ correct answers is the special case of parallel guesswork in [15] with $U = 1$ and the sequence distribution of $V$ users being identical.

Based on Corollary 1 in [15] and our analysis for single answer case, we have:

$$p(G_{opt}^V(y_0^{N-1} \mid z_0) = i) \approx \begin{cases} \frac{1}{i} \exp(-N\Lambda^*(N^{-1} \log i)), & \text{if } N^{-1} \log i \leq H \\ \frac{1}{i} \exp(-NVA^*(N^{-1} \log i)), & \text{if } N^{-1} \log i \geq H, \end{cases} \quad (4.16)$$

where $\Lambda^*$ is defined as in (4.11), and $H$ is the Shannon entropy rate of the sequence.
of words \( y_{0}^{N-1} \) which is also of the sequence of pairs \( z_{1}^{N}, y_{0}^{N-1} \) as shown by (4.4):

\[
H = \lim_{N \to \infty} \frac{1}{N} H(y_{0}^{N-1}) \\
= -\lim_{N \to \infty} \frac{1}{N} \sum_{y_{0}^{-1}} p^{N}(y_{0}^{N-1}) \log p^{N}(y_{0}^{N-1}). (4.17)
\]

We next present the upper bound of the probability mass covered by beam search with beam size \( k \) when \( V \) valid translations exist.

**Theorem 3.** Given a seq2seq model with transition matrix \( M \) for the sequence of pairs \( \{z_{n}, y_{n-1}\} \), the probability mass covered by beam search with beam size \( k \) when \( V \) valid translations exist is at most

\[
\sum_{i=1}^{k} p(G_{opt}(y_{0}^{N-1} | z_{0}) = i) \\
\approx \sum_{i=1}^{k} \frac{1}{i} \exp(-C(i)N \sup_{\alpha \geq 1} \{\alpha N^{-1} \log i - (1 + \alpha) \log \lambda_{1+\alpha}\}), (4.18)
\]

where \( \lambda_{1+\alpha} \) is the largest eigenvalue of the matrix \( M^{\frac{1}{1+\alpha}} \) with entries \( M_{ab}^{\frac{1}{1+\alpha}} \). \( C(i) = 1 \) when \( i < e^{NH} \), \( C(i) = V \) otherwise.

With \( V = 1 \), Theorem 3 reduces to Theorem 1. Note that the upper bound of the probability covered when \( V \) valid answers exist would be the same as when a single valid answer exists unless \( k > e^{NH} \).

### 4.4 Discussion

Guesswork for inference in seq2seq model can be further extended to the scenario when a slightly distorted version of a reference sentence can be accepted. In reality, a reference list cannot exhaust all possible translations for a given sentence. It would be reasonable to accept outputs within the threshold of a distortion level. Previous results on guessing under distortion in \( [5] \) can be directly applied here, with an element-wise distortion metric properly defined for machine translation.
Our main results in this chapter are based on the approximation in (4.10) as suggested by [14], which is not tight for general $k$ (beam size). The bound based on (4.13) and (4.14) by [73] is tighter for translation with single answer as we mentioned in Sect. 4.3 and proved in Appendix 4.3.1. We aim to extend it to the multi-answer case in the future. Both bounds require the measurement of the Rényi entropy of a practical seq2seq model. There exist technical difficulties for designing a practical measuring algorithm, mainly due to the huge state size. Given $z_0$, if the length of the output sequence is at most $N$, the state size of the neural network can be as large as $|Y|^N$ with $|Y|$ being the vocabulary size of the output language. If the input sequence length is at most $N'$, then the number of values $z_0$ can take can be as large as $|X|^{N'}$ with $|X|$ being the vocabulary size of the input language. In total, the space size of the pair $\{z_n, y_{n-1}\}$ can be as large as $S = |X|^{N'}|Y|^N|Y|$. However, we observe that special structures exist. On one hand, $f_1$ in (4.1) and $f_2$ in (4.2) are implicitly known and they define a very sparse transition matrix $M$. On the other hand, the Markov chain is most likely reducible and breaking it into its irreducible parts will reduce the space size. These observations may help our further study.
Chapter 5

Conclusion

Information theory and machine learning are two sets of tools each with different primary focus. Information theory, originated from communication and security, studies the theoretical limit of utilizing resources in an information-related process. Machine learning, on the other hand, has its primary focus on solving practical problems over a large range. As of current stage, many machine learning algorithms, in particular deep learning based algorithms, are developed from intuition rather than built from provable and systematic approaches. In this thesis, we introduced information theoretical tools to practical machine learning methods and explored the application-driven intersections between information theory and machine learning. We provided actionable learning algorithms that leverage the insights and guidance from information theory. And we showcased the algorithmic gain by combining the two.

The application domains studied in this thesis range from modern communication systems such as IoT to bio-molecular imaging to the problem of machine translation. We first looked at the problem of allocating source within a physical network so as to distribute a neural network. To this end, we deployed the framework of network coding from information theory literature and we proposed NNC as a practical scheme for decomposing NNs. We examined the performance of NNC in a variety of applications under a variety of network topologies and channel statistics through experiments. There, we showed that NNC massively lowers the energy used compared to existing schemes. The second problem we studied is the coding scheme in bio-molecular pro-
filing. We adopted the concept of error correction codes in communication literature and redesigned the encoding scheme in a learning-based data-driven manner. We showed that the false discovery rate for rare RNAs, which is the key experimental metric, is vastly improved with a MAP decoder and a learning-based codebook. In the third problem, we exercised the tool of guesswork in machine translation using seq2seq model. There, we provide a theoretical guideline for quantifying the uncertainty in the system. Our results give insights for selecting the beam size in inference with beam search, in the case of a single valid translation existing or when multiple valid translations existing. We also discuss the extension of the results to translation with distortion tolerance.

What we presented in this thesis is an initial study of incorporating information theoretical tools into machine learning algorithms. As discussed in previous sessions, our proposed IT-integrated ML algorithms are still in their infancy and many exciting open questions remain to be studied. Nevertheless, our preliminary examination here points towards a promising direction of integrating IT into ML for practical algorithmic gains, which could go beyond the application domains studied in this thesis.
Bibliography


[13] François Chollet et al. Keras. [https://keras.io](https://keras.io) 2015.


