In-Network Learning : Distributed Training and Inference in Communication Networks

Matei Moldoveanu

Université Gustave Eiffel

In-Network Learning: Distributed Training and Inference in Communication Networks

Thèse de doctorat de l'Université Gustave Eiffel

École Doctoral MATHÉMATIQUES ET STIC (MSTIC)
Spécialité de doctorat: mathématiques et des sciences et technologies de l'information et de la communication.
Unité de recherche : Laboratoire d'Informatique Gaspard-Monge

Thèse présentée et soutenue à l'Université Gustave Eiffel,

le 30/01/2023, par :

Matei Catalin Moldoveanu

Composition du Jury

Michel Kieffer Professor, CentraleSupélec, Université Paris-Saclay	Rapporteur
Samir Perlaza Chargé de Recherche, INRIA	Rapporteur
Mireille Sarkiss Associate Professor, Telecom SudParis, Institut Polytechnique de Paris	Examinatrice
Mari Kobayashi Professor, CentraleSupelec, Université Paris-Saclay and Apple	Examinatrice
Sheng Yang Professor, CentraleSupélec, Université Paris-Saclay	Examinateur

Encadrement de la thèse

Abderrezak Rachedi Professor, University Gustave Eiffel	Directeur de thèse
Abdellatif Zaidi Associate Professor, University Gustave Eiffel and Huawei	Co-Directeur de thèse

Abstract

Consider a setup in which multiple sensors or nodes work together to track, classify, or otherwise infer a quantity or label of interest. This is a configuration commonly found in many applications, e.g., environmental monitoring, self-driving cars, medical diagnosis. In practical applications, developing analytical models for such problems is challenging due to the complex relationships between the data observed and the target variable that needs to be inferred. Machine learning based models, such as neural networks (NN), have proven to be adept at modelling such complex relationships from data, through a so-called training phase. However, these sensors are usually restricted in terms of the data they are allowed to transmit. This restriction could be due to constraints on the communication network, found in environmental monitoring, or privacy constraints, found in medical problems. To overcome these issues, models that can be trained and used in a distributed manner need to be developed.

In this thesis, we provide a framework, named in-network learning, that can be used in such a distributed setting during both the training and the inference phase. The framework is made up of an architecture that describes how the multiple NNs should interact with each other during training and inference, as well as a loss function that can be used to jointly optimise the parameters of the NNs in a distributed manner. Inspired by the information bottleneck method, the loss function looks to train the NN to achieve the best trade-off between the accuracy of the predictions, under logarithmic loss, and the complexity of the information passed between the nodes of the communication network, measured by their minimum description length (MDL). The loss function and architecture take into account the network topology and can be applied to any communication network that can be modelled by a directed acyclyc graph.

Finally, we consider the problem of scheduling and resource allocation in such distributed inference networks. Inference models are commonly used in constrained communication networks. In this setting, the resources of the communication network, such as power or bandwidth, can be redistributed to improve network performance. Due to the lack of prior knowledge, deciding how to properly allocate available resources, based on only distributed available data, can be a challenging task. The issue appears due to the fact that no one node can provide a definitive picture of the system.

Each data observing node can assess the usefulness of its own data; however, it does not have knowledge of the other nodes data. The decision node has an understanding of the usefulness of each node relative to the other nodes, but only based on compressed representations. As such, in this thesis, we provide a scheduling and power allocation algorithm that overcomes this issue by making use of both local assessments, in which each node assesses the quality of its own data, and global assessments, in which the decision node assesses the usefulness of each node based on compresses representations. Our scheduling and power allocation algorithms then combine these two assessments in a suitable manner to improve the performance of the network.

Keywords - *distributed learning, information theory, classification, Information Bottleneck, logarithmic loss, resource allocation.*

Contents

List of Figures

List of Tables

List of Acronyms

CEO	chief executive officer
CMI	conditional mutual information
DPI	Data Processing Inequality
EU	expected usefulness
FC	fusion center
FL	federated learning
GRI	global relevance indicator
IB	information bottleneck
INL	in-network learning
JSCC	joint source and channel coding
KL	Kullback-Leibler
MI	mutual information
ML	machine learning
MDL	minimum description length
NN	neural network
OFDM	orthogonal frequency division multiplexing
PS	parameter server
RD	rate distortion
RI	relevance indicator
SGD	stochastic gradient descent
SL	split learning
SNR	signal to noise ratio
VIB	variational information bottelneck
WSN	wireless sensor network

Publications

Journals

1. Matei Moldoveanu and Abdellatif Zaidi, "In-Network Learning: Distributed Training and Inference in Networks", Submitted for publication to IEEE Journal on Selected Areas in Communications, 2022.

2. Abdellatif Zaidi, Matei Moldoveanu and Piotr Krasnowski, "In-network Learning: data importance, scheduling and power allocation", *in preparation for submission for IEEE Trans. on Pattern Analysis and Machine Intelligence.*

International Conferences

1. Matei Moldoveanu and Abdellatif Zaidi, "On In-network learning. A Comparative Study with Federated and Split Learning", 2021 IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Virtual (Lucca), Italy, 2021, 221-225. Invited paper in the special session Machine Learning at the Edge.

2. Matei Moldoveanu and Abdellatif Zaidi, "In-network Learning for Distributed Training and Inference in Networks", 2021 IEEE Globecom Workshops (GC Wkshps), Madrid, Spain, 2021, 1-6.

Patents

1. Abdellatif Zaidi and Matei Moldoveanu , "Robust in-network learning: system and method for learning and inference in presence of missing data", International Patent filled 2022. *(This patent does not form part of this thesis.)*

Chapter 1

Introduction

1.1 Motivation

Consider the example problem of identifying or tracking a vehicle in a large urban area. To this end, sensors are placed throughout the area to collect different types of information, such as sound, image/video, proximity, etc. The sensors need to work together to jointly predict where the vehicle is or where it is going. An example of such a problem is illustrated in Figure 1.1.

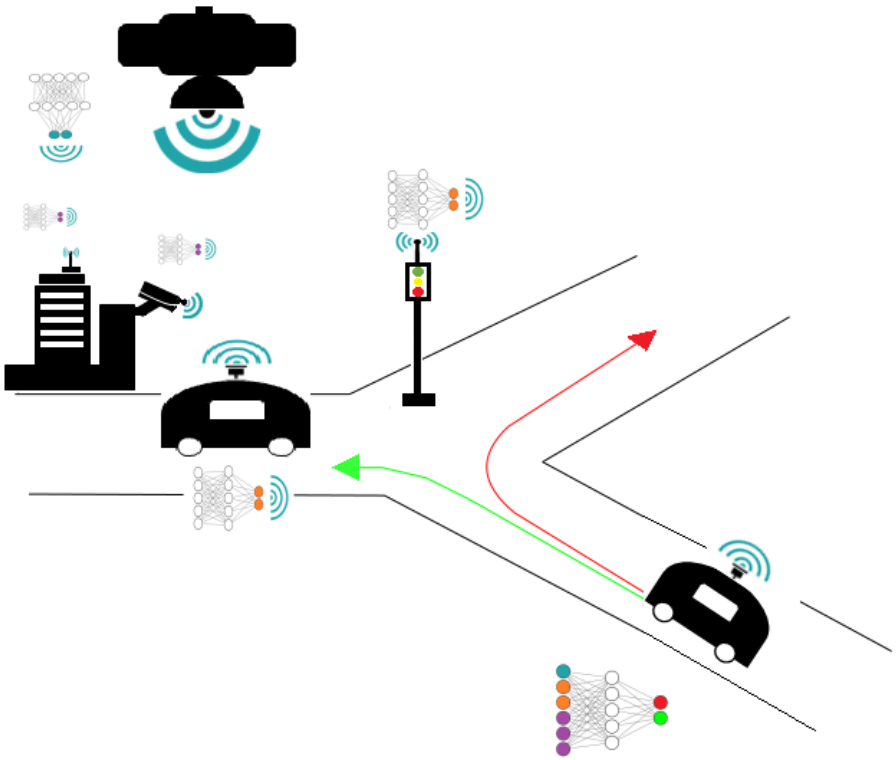


Figure 1.1: Example problem of tracking a vehicle.

Since the sensors are tracking the same vehicle, the data collected by the sensors are

correlated; however, the exact relationship between the data we observe and the data we want to predict is difficult to know in advance. As such, we assume that initially the sensors collect data from which we can train a model to learn the relationship between the observed data and the information we want to predict, that is, tracking the vehicle. We refer to this period of time as the training phase. The most natural approach to learning such a model would be to collect the data samples from the sensors to a central location, where a model can be learnt based on all the data. Unfortunately, sensors can be deployed in an area where the communication infrastructure is limited and intervention is not possible after deployment to improve the infrastructure or collect the data [4]. As such, transferring large amounts of data to a central location might not be possible due to communication constraints. Another issue with centralising data is that sensitive information can be contained in the collected data and its transfer could violate privacy restrictions [5]. Due to these reasons, both the training phase and the deployment of the model, also termed the inference phase, need to take place in a distributed manner, i.e., without the exchange of raw data. This problem is sometimes referred to as the distributed training and inference problem.

More generally, in this thesis, we assume that there are some devices, which we denote as edge nodes, that observe data related to a variable of interest that needs to be inferred at a central decision node or fusion centre. The edge nodes are connected to the central node either directly or through some intermediary nodes. We assume that the topology of the network can be modelled by an acyclic directed graph. The distribution of the data is not known and only a distributed data set is available; that is, different parts of the data are available at different nodes. We equip each node with a neural network (NN). This is done due to the success of machine learning (ML) techniques in learning, from samples, complex relationships between observed data and a target variable in areas such as computer vision [6], image processing [7], robotics [8], and natural language processing [9].

The question is, given some measures of the performance of the model, what is the best way to train such a model? Clearly, removing any of the relevant data can only lead to a reduced prediction performance. However, on the other hand, extracting and transmitting data that are not relevant to the task at hand can put a strain on

the communication network or even degrade the performance of the network. Thus, intuitively, nodes need to extract the smallest number of features that collectively give the best prediction performance. Deciding which features to extract needs to be done without explicit coordination between the nodes. How should that be performed optimally without sharing raw data? In particular, how should each node process information and what should it transmit to other nodes?

1.2 Contributions

In this thesis, we focus on the distributed training and inference problem. We discuss how multiple nodes, that observe data related to the same event, can work together to jointly train a model that can be used in a distributed manner during the inference phase through a novel framework named in-network learning (INL).

INL for a star topology, in which the nodes that observe data are directly connected to the decision node, was first introduced and analysed in [10, 11]. We build on these results by providing a novel experimental comparison of INL with other distributed learning algorithms, namely those of [12, 13]. These results were published in [14].

The extension of the INL to network topologies that can be modelled with a directed, acyclic graph is proposed and studied in the patent document [11] by A. Zaidi, K. Scaman and P. Escamilla. In this thesis, we derive a suitable loss function for such a model. The approach to the derivation of the loss function is borrowed from [10], i.e., first we assume that joint distributions are known and proceed with modelling the problem as a network source coding one under logarithmic loss measure and finding suitable (achievable) rate-distortion trade-offs, then we derive a variational inner bound, and finally we parametrize that lower/inner bound using neural networks. This work was published as part of [15].

Finally, we consider the practical problem of deploying INL for a star topology in a constrained wireless setting; an example would be the deployment in wireless sensor networks (WSNs) [16]. The performance of WSNs is significantly restricted by its available radio resources and the way in which the resources are allocated between nodes. Allocating available resources, when we only have access to a dataset, is

impeded by the distributed nature of the data and the unknown relationships between the data observed by each sensor. The patent [17] already suggested the idea of having: (i) each device measuring its own local assessment of how relevant is the data it holds for the inference task at hand and transmitting it to the decision node, (ii) that device also simultaneously sends the output activation values to the decision node, (iii) the decision node combining all received information (activation values from the various devices as well as their individual local assessments) and then forming its own (global) assessment of how relevant each device's observation is, (iv) application of the idea to device scheduling and power allocation (two algorithms for scheduling were already given in [17]). All these steps were later reported on and detailed in the intermediate version [18]. We differentiate ourselves from these works by describing how the the local nodes can compute their local assessments, as well as providing an appropriate way for the decision node to combine the received local assessments with the activation values to obtain its own global assessment. We also propose an improved scheduling algorithm, as well as a power allocation algorithm, which was not presented in either [17, 18]. The algorithms are validated experimentally.

1.3 Outline

The thesis is organised as follows.

Chapter 2

In Chapter 2 we review some of the results of point-to-point communication, learning, and the connection between the two. We start by reviewing some information-theoretic measures related to both communication and learning, which are used frequently throughout the thesis, including mutual information and entropy. A more in-depth discussion of these measures can be found in [19]. We then discuss the point-to-point communication problem [20], highlighting how mutual information, introduced previously, can be used to measure the number of bits (rate) required to transmit information between two devices. Afterward, we discuss the learning problem, specifically from

the perspective of the information bottleneck (IB) method [21]. Finally, we review some of the results of [3] that connect the IB problem and the point-to-point communication problem.

Chapter 3

In Chapter 3 we study the distributed learning and inference problem in which multiple edge nodes that observe data are connected directly to a decision node that needs to predict some target variable based on the information it receives from the edge nodes. This is a particular case of the learning problem discussed at the beginning of this chapter. We discuss the algorithm proposed by [10] which seeks to solve this problem, including an in-depth analysis of the training and inference procedure provided in [11]. We then provide a novel experimental comparison between the algorithm of [10] and other distributed training algorithms, namely the famous federated learning algorithm of [12] and the split learning of [22].

Chapter 4

In Chapter 4 we study a generalisation of the problem studied in Chapter 3. More specifically, we consider the case in which the edge nodes that observe data do not communicate directly with the decision node, but through some intermediary nodes. We analyse the case in which the topology of the network can be modelled by a directed acyclyc graph. The extension of the INL to network topologies that can be modelled with a directed, acyclic graph is proposed and studied in the patent document [11] by A. Zaidi, K. Scaman and P. Escamilla. In this chapter, we first derive an attainable trade-off between the complexity of the features extracted by each edge node and the prediction performance at the decision node, which depends on the network topology. We then discuss how, given a network topology, one can derive a loss function for training the model in a distributed setting using the derived trade-off. We then exemplify the steps by deriving the loss function for an example network topology. The approach to the derivation of the loss function follows similarly to the ones of [10], i.e., using the

obtained (achievable) rate-distortion trade-offs we derive a variational inner bound, and finally we parameterize that lower/inner bound using neural networks. This work was published as part of [15].

Chapter 5

In Chapter 5 we consider the implementation of the algorithm discussed in Chapter 3, in a resource-constrained wireless sensor network. We study the problem of scheduling and resource allocation in the case where the distribution of the data is not known. In [17, 18] it was observed that in such a situation it is difficult to schedule and allocate resources, since no one node has the complete picture of the setting. While the decision node receives the data from all the nodes and thus can assess their relative usefulness, the information received by the central node is only a compressed representation of the full data. The edge nodes have access to the full data; however, they cannot assess what other nodes observe. In [17], the author patented the idea of having: (i) each device measuring its own local assessment of how relevant is the data it holds for the inference task at hand and transmitting it to the decision node, (ii) that device also simultaneously sends the output activation values to the decision node, (iii) the decision node combining all received information (activation values from the various devices as well as their individual local assessments) and then forming its own (global) assessment of how relevant each device's observation is, (iv) application of the idea to device scheduling and power allocation (two algorithms for scheduling were already given in [17]). In this chapter, we present in detail the ideea of the patent [17], which was also described in [18]. We then build on this ideea by describing how the the local nodes can compute their local assessments, as well as providing an appropriate way for the decision node to combine the received local assessments with the activation values to obtain its own global assessment. We also propose an improved scheduling algorithm, as well as a power allocation algorithm. Finally, we present an experimental comparison to showcase the advantages of the presented scheduling and power allocation algorithms. Part of these results are presented in [23].

In Chapter 6 we discuss potential areas of future investigation.

1.4 Notation

The following notation will be used throughout the paper. Upper case letters denote random variables, e.g. X; lower case letters denote realisations of random variables, e.g x, and calligraphic letters denote sets, e.g., $\mathcal{X}$. The cardinality of a set is denoted by $|\mathcal{X}|$. For a set of natural numbers $\mathcal{S} \subseteq \mathcal{K}$, the complementary set of $\mathcal{S}$ is denoted by $\mathcal{S}^c$, that is, $\mathcal{S}^c = \{k \in \mathbb{N} : k \in \mathcal{K} \setminus \mathcal{S}\}$. For a random variable X, that takes values in $\mathcal{X}$, with probability distribution P_X, the shorthand $p(x) = P_X(x), x \in X$ is used. Boldface letters denote matrices or vectors, e.g., $\mathbf{X}$ or $\mathbf{x}$. A sequence of random variables $(X_k, X_{k+1}, ..., X_j)$ is abbreviated by X_k^j. The sequence X_1^n is sometimes denoted simply as X^n, i.e., $X^n \triangleq (X_1, X_2, ..., X_n)$. For random variables $(X_1, X_2, ...)$ and a set of integers $\mathcal{K} \subseteq \mathrm{N}$, the notation $X_\mathcal{K}$ designates the vector of random variables with indices in the set $\mathcal{K}$, that is, $X_\mathcal{K} \triangleq \{X_k : k \in \mathcal{K}\}$. If $\mathcal{K} = \emptyset$ then $X_\mathcal{K} = \emptyset$. Furthermore, for random vectors with zero mean $\mathbf{x}$ and $\mathbf{y}$, the quantities $\sum_\mathbf{x}$, $\sum_{\mathbf{x},\mathbf{y}}$ and $\sum_{\mathbf{x}|\mathbf{y}}$ denote, respectively, the covariance matrix of the vector $\mathbf{x}$, the covariance matrix of vector $(\mathbf{x}, \mathbf{y})$ and the conditional covariance of $\mathbf{x}$ given $\mathbf{y}$. We will use the notation $\mathrm{diag}([x_1, \ldots, x_n])$ to denote an $n \times n$ matrix whose diagonal elements are given by $[x_1, \ldots, x_n]$, while the off-diagonal elements are zero. We say that the random variables U, X and Y form a Markov Chain in that order, which we denote as $U - X - Y$, if $p(u, x, y) = p(x, u)p(y|x)$. Finally, for two probability measures P_X and Q_X over the same alphabet $\mathcal{X}$, the relative entropy or Kullback-Leibler divergence is denoted as $D_{KL}(P_X||Q_X)$. That is, if P_X is absolutely continuous with respect to Q_X, then $D_{KL}(P_X||Q_X) = \mathbb{E}_{P_X}[\log(P_X(X)/Q_X(X))]$, otherwise $D_{KL}(P_X||Q_X) = \infty$.

Chapter 2

Connection between Communication and Learning

In this chapter, we discuss a simplified version of the problem discussed in the introduction. We consider that there are only two nodes, one node that observes data, and one node that wants to infer the target variable. This setting greatly simplifies the problem, as now the node observing the data does not need to take into account what other nodes might observe. This chapter is a review of relevant results from the literature. The aim of this chapter is to give some intuitions that will be used to derive results for the more general case discussed in Chapter 1. For the case where the distribution of the data is known, and the target variable is the observed data, the problem reduces to that of point-to-point communication. This is a famous problem that was tackled by Shannon [20], and in this chapter we formally introduce the problem of point-to-point communication and restate some of the results presented in [20]. We then discuss the problem of point-to-point communication when the target variable is not the observed data. Afterwards we consider the same setup, however, we assume that the distribution of the data is not known and only a dataset is available. We discuss this learning problem from the perspective of the famous information bottleneck (IB) method [21]. We first discuss the intuition behind the IB problem and its solution for the case in which the distribution of the data is known and discrete. We then present the results of [24] in which the authors extended the IB for the case in which the distribution of the data does not need to be known and only a set of training samples need to be available. Next,

we discuss the connection between IB and the point-to-point communication problem. We revisit the results of [3] that discuss how IB is essentially a compression problem, and then discuss how the rate of compression from point-to-point communication is linked to a key learning metric, namely the generalisation gap. At the beginning of this chapter, we briefly review some information-theoretic measures, which are fundamental to many results in this thesis, before discussing the aforementioned results. The introduction is very brief, with many details left out; for a comprehensive review see [19].

2.1 Preliminaries

Consider the case in which a person draws a word from a hat and we need to predict what word they drew, with as little additional information from the person who drew the word. We know in advance the finite set of words $\mathcal{X} = \{x_1, \dots x_{|\mathcal{X}|}\}$ from which a person can draw a word. In the case in which we cannot receive any information from the person, if all the words are equally like to be used, then it would be hard to actually predict correctly what the word they drew is. However, if the hat contains only one word, then we can easily predict correctly what the chosen word is, without any additional information. Intuitively, the prediction is affected by the amount of uncertainty that we have about the word that will be drawn, and as a consequence the more uncertain we are, the more additional information we need to receive. Let the probability that any word is chosen be modelled by the random variable X with the probability mass function P_X. We would like to be able to quantify the amount of uncertainty contained in X, and we will define this measure of uncertainty as $H(X)$. In his seminal work [20], Shannon suggested the following three conditions that such a measure of uncertainty should satisfy.

Condition 2.1.1. *$H(X)$ should be continuous in P_X.*

Condition 2.1.2. *If $p(x) = \frac{1}{|\mathcal{X}|}$ $\forall x \in \mathcal{X}$ then $H(X)$ should be a monotonically increasing function of $|\mathcal{X}|$.*

Condition 2.1.3. *For all possible groupings of $\mathcal{X}$ into groups $\mathcal{T}$, $H(X)$ satisfies the following*

$$H(X) = H(T) + \sum_{t \in \mathcal{T}} p(t) H(X|T = t). \tag{2.1}$$

The Boltzamn-Shannon entropy (which we will refer to as entropy) was shown to be the only function[1] that satisfies the three conditions mentioned above.

Definition 2.1.1. *Let X be a discrete random variable distributed according to P_X. The entropy of X is defined as*

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x). \tag{2.2}$$

The base of the logarithm sets the scale of the measure. Unless otherwise stated, throughout this thesis, we will assume that the logarithm is in base 2 and we will measure the entropy in bits. Intuitively, the entropy of base 2 can be seen as the minimum number of yes or no questions that we need to ask to figure out what the drawn word is, on average. The definition of entropy can easily be extended to the case of joint entropy, which is defined on a set of random variables $X_1, \ldots, X_N$ and measures the joint uncertainty.

Definition 2.1.2. *For a set of N discrete random variables distributed according to $P_{X_1,\ldots,X_N}$, the joint entropy of the set is defined as*

$$H(X_1, \ldots, X_N) = -\sum_{x_1,\ldots,x_N} p(x_1, \ldots, x_N) \log p(x_1, \ldots, x_N). \tag{2.3}$$

Closely related to entropy is the concept of conditional entropy. Consider again the case in which a person draws a word from the hat that we need to guess; however, we now obtain some additional information from the person, for example, the number of letters in the word, which we will denote by Y. If all words in the space $\mathcal{X}$ have the same number of letters, then this additional information does not reduce the uncertainty; that is, Y does not provide information on X. If all the words in the space $\mathcal{X}$ have, each,

[1] The function is defined up to a multiplicative constant.

a different number of letters, then given Y we can know X. The conditional entropy $H(X|Y)$ of a random variable X given another Y, measures the uncertainty of X that remains once we determine the value of Y.

Definition 2.1.3. *For (X, Y) distributed according to $P_{X,Y}$, the conditional entropy of X given Y is defined as*

$$H(X|Y) = -\sum_{y}\sum_{x} p(x,y)\log p(x|y). \tag{2.4}$$

Closely related to entropy and conditional entropy is the concept of mutual information between two random variables. Mutual information measures the dependency between two random variables. Informally, mutual information $I(X;Y)$ can be thought of as the amount of information that is common to both variables X and Y.

Definition 2.1.4. *Let $(X, Y) \sim P_{X,Y}$ be two discrete variables with marginal distributions $p(x) = \sum_y p(x,y)$ and $p(y) = \sum_x p(x,y)$. The mutual information between X and Y is defined as*

$$I(X;Y) = \sum_{x,y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}. \tag{2.5}$$

From the above definition, it can be obtained that

$$I(X;Y) = I(Y;X) = H(X) + H(Y) - H(X,Y), \tag{2.6}$$

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X). \tag{2.7}$$

A diagram showing the relationship between mutual information, entropy, and conditional entropy can be seen in Figure 2.1.

Finally, we introduce the concept of Kullback-Leibler(KL) divergence. This measure, also known as relative entropy, quantifies the distance between two distributions P_X and Q_X defined over the same space.

Definition 2.1.5. *The KL divergence between two discrete probability distributions P_X and Q_X*

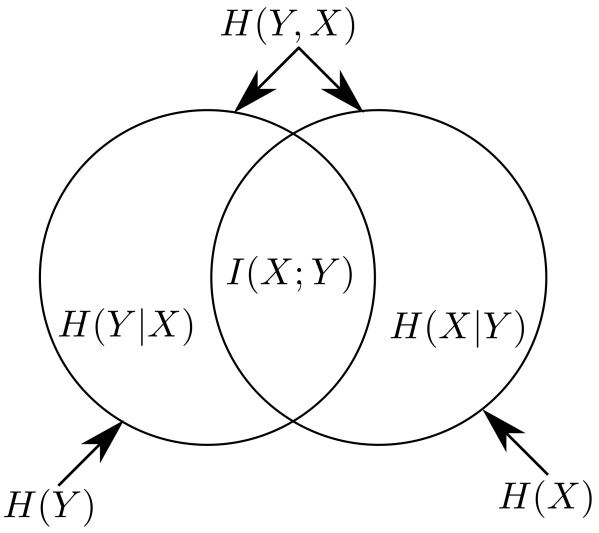


Figure 2.1: Relationship between mutual information and entropy. Figure redrawn from [1].

is defined as

$$D_{KL}(P_X||Q_X) = \sum_x p(x) \log \frac{p(x)}{q(x)}. \tag{2.8}$$

Additionally, KL divergence is related to mutual information through the following well-known equality

$$I(X;Y) = D_{KL}(P(X,Y)||P(X)P(Y)). \tag{2.9}$$

An interesting property of the KL divergence is that it is a positive value and equals zero if and only if $P_X(X) = Q_X(X)$.

Theorem 2.1.1. *[19, Theorem 2.6.3] For P_X and Q_X, two probability distributions on the same space $\mathcal{X}$ then*

$$D_{KL}(P_X||Q_X) \geq 0, \tag{2.10}$$

with equality if and only if $p(x) = q(x)$ for any $x \in \mathcal{X}$.

From (2.9) and Proposition 2.1.1 we can state that

Corollary 2.1.1. *Let $(X, Y) \sim P_{X,Y}$ be two discrete variables, then*

$$I(X;Y) \geq 0, \tag{2.11}$$

with equality if and only if $p(x, y) = p(x)p(y)$, where $p(x) = \sum_y p(x, y)$ and $p(y) = \sum_x p(x, y)$.

Another important property of the mutual information is the Data Processing Inequality (DPI). Intuitively, DPI implies that the information that one random variable X contains about another variable Y, cannot be increased by processing the variable X.

Theorem 2.1.2. *[19, Theorem 2.8.1] If (Y, X, U) are random variables for which $p(x, y, u) = p(x, y)p(u|x)$, then $I(X, Y) \geq I(U; Y)$.*

We denote random variables (Y, X, U) for which $p(x, y, u) = p(x, y)p(u|x)$ as $Y - X - U$, and we say that they form a Markov chain.

So far, we have only talked about discrete random variables; however, the KL divergence can also be defined for continuous random variables.

Definition 2.1.6. *The KL divergence between two continuous probability distributions P_X and Q_X is defined as*

$$D_{KL}(P_X||Q_X) = \int_{\mathcal{X}} p(x) \log \frac{p(x)}{q(x)} dx. \tag{2.12}$$

Similarly, using (2.9), the concept of MI can be extended to continuous random variables.

Definition 2.1.7. *Let $(X, Y) \sim P_{X,Y}$ be two continuous variables with marginal distributions $p(x) = \int_{\mathcal{Y}} p(x, y)$ and $p(y) = \int_{\mathcal{X}} p(x, y)$. Mutual information between X and Y is defined as*

$$I(X;Y) = \int_{\mathcal{X}} \int_{\mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)} dxdy. \tag{2.13}$$

We have introduced the fundamental measures required throughout this paper.

2.2 Point-to-Point Communication

In point-to-point communication, we want to send information from one node, the sender, to another node, the receiver, over some imperfect medium of communication, the channel. Shannon [20], formulated this problem mathematically using the communication model shown in Figure 2.2. The model consists of a discreet source of messages X, a sender, a receiver, and a discreet communication channel that connects the sender to the receiver. The source X has a probability mass function P_X, which takes values in the alphabet $\mathcal{X}$. A channel is defined by its input alphabet $\mathcal{W}$, the output alphabet $\hat{\mathcal{W}}$, and the transition probability $p(\hat{w}|w)$. We will consider the case in which the channel is memoryless, that is, the probability of the output of the channel depends only on the input at the time. We consider that the sender observes a sequence of n independent messages, $X^n := \{X_i\}_{i=1}^n \sim \prod_{i=1}^n P_X$, which must be communicated to the receiver. The sender observes these messages and then encodes them so that they can be transmitted over the channel. For simplicity, in Figure 2.2 and the rest of this thesis, we assume that the sender can use the channel n times to transmit information; this is not always the case, and we can find more general results for the case where the number of channel uses is not equal to the number of messages observed in [19]. As such, the sender needs to find an encoding function that, given X^n, outputs a representation W^n that can be transmitted over the channel. The representation W^n is distorted by the channel transition probability $p(\hat{w}|w)$ and the receiver obtains $\hat{W}^n$. The receiver then, using a decoding function, obtains an estimate of X^n, denoted by $\hat{X}^n$, based on the received $\hat{W}^n$. Given a fixed source P_X and a channel $p(\hat{w}|w)$, the aim is to find the encoder and decoder functions for which X can be reconstructed by the receiver, with the smallest error. It

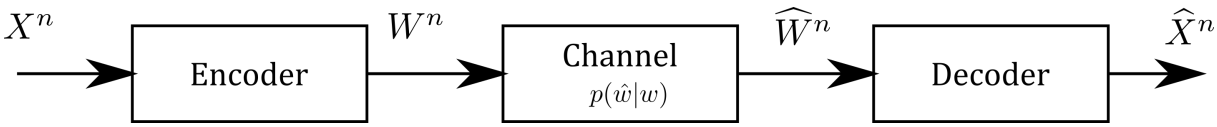


Figure 2.2: Point-to-point communication model.

is interesting to observe that the encoder and decoder depend on both the source and the channel, which means that changes in either could require a redesign of the system. Shannon proposed separating the encoding into two distinct parts, the first which depends only on the source of the data, and the second which depends only on the

channel. The first part is the source coding part, in which redundant information from the source data is removed and a minimal representation is obtained; this is sometimes also referred to as the compression phase. The second part is the channel coding part, in which the compressed representation is enhanced by adding redundancies to offset the error added by the channel during transmission. The proposed model diagram is shown in Figure 2.3. A key result of Shannon [20] was to show that this separation and disjoint design between source coding and channel coding is optimal for the case where messages are encoded and decoded in large sequences, that is, n is large. A formal statement of the theorem will be discussed in Section 2.2.2.

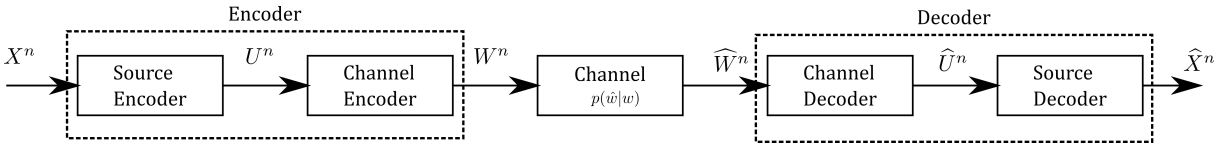


Figure 2.3: Separation of source and channel coding for point-to-point communication model.

We will discuss the problem of source coding and channel coding, relating the amount of bits that need to be transmitted, with the accuracy of the reconstruction of the original messages at the receiver. In future sections, we relate the problem of source coding with that of learning and discuss how choices in the learning problem can have an impact on the communication constraints that the inference scheme would place on the communication network.

2.2.1 Source Coding

Source coding, or source compression, aims to obtain a compressed representation of the original messages X^n, such that the original messages can be recovered from the compressed representation, within some distortion measured by a chosen distortion function $d(x, \hat{x})$.

Loosely speaking, rate-distortion (RD) theory aims to characterise the minimum number of bits required to be transmitted from the sender to the receiver such that the receiver can recover the original messages within some distortion level. More

specifically, consider the following encoding function

$$f : \mathcal{X}^n \longrightarrow \mathcal{U}^n \quad \text{with } \mathcal{U}^n = \{1, \ldots, 2^{nR}\}. \tag{2.14}$$

Under the assumption that the encoding is transmitted over the channel without error, the receiver can then use the following decoding function to estimate the original messages

$$h : \mathcal{U}^n \longrightarrow \hat{\mathcal{X}}^n. \tag{2.15}$$

The pair of (f, h) is usually denoted as a $(2^{nR}, n)$ rate distortion code. The rate of the code, denoted R, represents the average number of bits, per message, that the sender needs to send to the receiver. It is clear that a smaller R implies less communication; as such, we are interested in finding the minimum R for which there exists a pair (f, h) such that the expected distortion is bounded by some value D, with the expected distortion given by

$$D_{f,h}(X) = \sum_{x^n \in \mathcal{X}^n} p(x^n) \frac{1}{n} \sum_{i=1}^{n} d(x_i, \hat{x}_i) \quad \text{with } \hat{x}^n = h(f(x^n)). \tag{2.16}$$

RD theory characterises the achievable region of (R, D).

Definition 2.2.1. *A rate-distortion pair* (R, D) *is said to be achievable if there exists a pair* (f, h)*, with rate* R *and with* $\lim_{n \longrightarrow \infty} D_{f,h}(X) \leq D$

Let the rate distortion function $R(D)$ be defined as the infimum of the rates R such that (R, D) is achievable. Shannon [20] has shown that $R(D)$ can be computed by solving the following constrained optimisation problem

$$\begin{aligned} R(D) := \min_{f,h} & \; I(X, \hat{X}) \\ \text{s.t.} \quad & D_{f,h}(X) \leq D. \end{aligned} \tag{2.17}$$

Although RD theory does not explicitly describe how to find (f, h) that solves the above optimisation problem, this bound can be used to inform the design of the pair (f, h).

2.2.2 Channel Coding

Once the source is compressed, the obtained representation must be transmitted through the channel. The amount of information that can be transmitted over a channel during one use is limited and is usually quantified by the channel capacity. Specifically, the capacity of a channel is defined as the maximum rate in bits per channel use at which information can be transmitted with an arbitrarily low probability of error. The capacity of a channel was shown to be equal to the maximum mutual information between its input and output, which is denoted as the information channel capacity.

Definition 2.2.2. *[19] The capacity of the channel* $(\mathcal{W}, p(\hat{w}, w), \hat{\mathcal{W}})$ *is given by*

$$C = \max_{p(w)} I(W; \hat{W}). \tag{2.18}$$

One of the key results of Shannon's work was the source-channel separation theorem. The theorem suggests that we can design the source and channel encoder-decoder pairs separately, as long as the rate of the source encoder is smaller than the capacity of the channel.

Theorem 2.2.1. *[25, Theorem 3.7] Given a discrete source* X *and a distortion measure* $d(\hat{x}, x)$ *with a rate-distortion function* $R(D)$ *and a channel* $p(\hat{w}|w)$ *with capacity* C*, the following statements hold*

- *if* $R(D) < C$*, then* (r, D) *is achievable for* $r \leq R(D)$*,*
- *if* (r, D) *is achievable, then* $r \leq C$*.*

Throughout this thesis, we will invoke the principle of capacity-achieving codes. This means that we will assume that we can find channel codes that will allow us to transmit information at rates equal to the channel capacity.

2.2.3 Remote Source Coding

Now, we extend the source coding problem to the case in which the data the receiver wants to recover are not the observed data X^n. Instead, the receiver is interested in recovering another variable, Y^n, which is correlated with the observed data. More

specifically, the sender wants to compress the sequence of messages X^n, so that the receiver can recover another variable Y^n, which is related to X^n through some distribution $P_{X|Y}$. We are still interested in characterising the $R(D)$ function; however, now the distortion is measured between Y^n and the estimate $\hat{Y}^n$.

Let the decoder at the receiver be defined as

$$g : \mathcal{U}^n \longrightarrow \hat{\mathcal{Y}}^n. \tag{2.19}$$

The diagram of the communication model is shown in Figure 2.4.

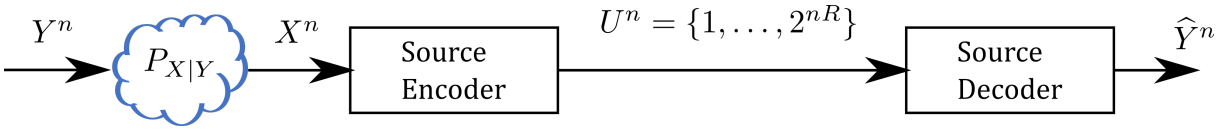


Figure 2.4: Remote source coding model.

In [26, 27] it was shown that the $R(D)$ function in this case is given by

$$\begin{aligned} R(D) &= \min_{f,g} I(X, \hat{Y}) \\ &\text{s.t.} D_{f,g}(Y) \leq D. \end{aligned} \tag{2.20}$$

The proof follows from the fact that the remote source coding problem can be reduced to a direct source coding problem for observable data X^n and a different distortion measure. In RD theory, the design of the distortion measure $d(\hat{x}, x)$ is subject to the choice of the designer. In the next section, we analyse the RD region under a particular distortion function.

2.2.3.1 Remote Source Coding under Logarithmic Loss

In this thesis, we will focus on logarithmic loss (which was introduced in the context of RD theory in [28]). Logarithmic loss, or log-loss for short, is a natural distortion measure in the settings in which the reconstructions do not have to be deterministic. That is, the decoder provides an assessment of the probability of each possible estimate, as opposed to choosing one estimate. Let the random variable Y denote the source with finite alphabet $\mathcal{Y}$ to be recovered by the receiver. Furthermore, let $\mathcal{P}(\mathcal{Y})$ denote the set of probability measures on $\mathcal{Y}$. The log-loss between $y \in \mathcal{Y}$ and its reconstruction

$\hat{y} \in \mathcal{P}(\mathcal{Y})$ is given by

$$d_{\log}(y, \hat{y}) = \log \frac{1}{\hat{y}(y)}. \tag{2.21}$$

Outside RD theory, the measure is widely used in various contexts, including clustering [29] and classification [24], pattern recognition [30], learning and prediction [31], image processing [32], privacy [33], and others. In [34], the authors justify the use of logarithmic loss by showing that it is the only loss function that satisfies a natural data processing property that appears in the presence of side information. In another work it was shown that minimising the logarithmic loss for the binary classification problem minimises the risk associated with any other smooth, proper, and convex functions, see [35]. Finally, in [36], the authors show the universality of log-loss over a finite alphabet in fixed-length lossy compression. Essentially, the authors show that, given any discrete source, fixed-length, lossy-compression problem, with some arbitrary distortion measure, there exists an equivalent fixed-length lossy-compression problem with the same alphabet source where the distortion is measured by the log-loss. By equivalence the authors implies that the optimal scheme for one problem is optimal for the other problem, additionally, a good scheme for one problem is also a good scheme for the other problem. These findings, along with the wide range of applications of log-loss, justify its use in our problem setting.

Let X denote the observed data, Y the target variable, and $P_{X,Y}$ their joint distribution. Let us denote the encoder-decoder model by $\phi(x) = Q(\cdot|x) \in \mathcal{P}(\mathcal{Y})$ for any $x \in \mathcal{X}$. The expected log-loss of such a model is lower bounded by the conditional entropy of Y given X

$$\begin{aligned}
\mathbb{E}_{P_{X,Y}}[d_{\log}(Y, \hat{Y})] &= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log \frac{1}{q(y|x)} \\
&= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log \frac{1}{p(x, y)} + \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{q(y|x)} \\
&= H(Y|X) + D_{KL}(P_{Y|X} || Q_{Y|X}) \\
&\geq H(Y|X)
\end{aligned} \tag{2.22}$$

with equality if an only if $Q_{Y|X}(\cdot|x) = P_{Y|X}(\cdot|x)$.

Now, let the possible stochastic encoding mapping $f : \mathcal{X}^n \longrightarrow \mathcal{U}^n$ with $|U^n| \leq 2^{nR}$. Additionally, the possibly stochastic decoder mapping is $g : \mathcal{U}^n \longrightarrow \mathcal{P}(\mathcal{X}^n)$. Using (2.22) we can lower bound the expected log loss give by (2.16) as

$$D_{f,g} \geq H(Y|U). \tag{2.23}$$

From the above inequality it can be shown that the rate-distortion region of the remote source coding problem under logarithmic loss is given by the union of all pairs (R, D) that satisfy

$$R \geq I(U; X) \tag{2.24a}$$

$$D \geq H(Y|U) \tag{2.24b}$$

where the union is over all the auxiliary random variables U that satisfy the Markov chain $Y - X - U$ and, without loss of generality, for which $|\mathcal{U}| \leq |\mathcal{X}| + 1$, see [3]. As noted in [37] the region presented in (2.24) is related to the IB problem; this connection will be discussed in more depth in Section 2.4.

2.3 Remote Inference

In this chapter so far we have discussed the case in which the edge node observes samples from the source X and needs to transmit some information to the central node such that it can infer the target variable of interest Y, given that the joint distribution $P_{X,Y}$ is known. We now consider the case in which $P_{X,Y}$ is not known and the encoding and decoding functions need to be learned from a dataset. This problem is sometimes referred to as the remote inference problem, and is shown in Figure 2.5.

For the remote inference problem described, the objective is to find a *good* mapping from an observed signal X to a target variable Y, based on training samples $\{(x_i, y_i)\}_{i=1}^n$. Although how to define a *good* mapping is subject to debate [2], the most common approach is to find the mapping that minimises the empirical estimate of some chosen

risk. The mapping is made up of two parts, as shown in Figure 2.5. First, we have an

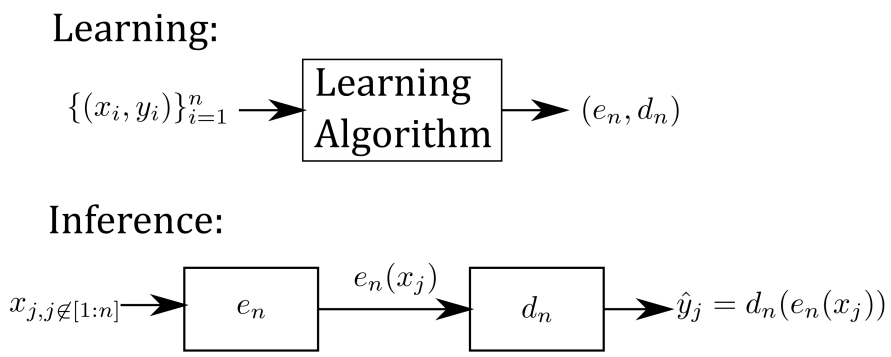


Figure 2.5: The remote inference problem. Figure adapted from [2].

encoder function that extracts the relevant features, also termed latent representation, from the observed data. Second, we have a decoding function that uses the latent representation obtained to predict the target variable.

In [38], the authors discussed the criteria that a good latent representation U of X should meet, for the problem of inferring Y, which are listed bellow

- U is a function of X, that is, the Markov chain $Y - X - U$ holds,
- *Sufficiency*: U is sufficient for task Y, i.e. $I(U;Y) = I(Y;X)$,
- *Minimality*: U discards all variability in X that is not relevant for predicting Y, that is, minimal $I(U;X)$.

Although these criteria give us a method to check if a representation U is a good representation, the problem remains as to how to find such a good representation. For this, we will now introduce the IB method of [21]. When first introduced, the method was designed for use on discrete variables whose distribution is known. It was in [24] that the IB problem was adapted for the learning problem. For completeness, we will first present the IB method as first presented in [21], and then discuss its adaptation to the learning problem.

2.3.1 Information Bottleneck Method

The IB method looks at finding the representation U, satisfying the Markov chain $Y-X-U$, which is maximally informative of Y (represented by high mutual information

$I(U;Y)$), while minimally informative of X (represented by small mutual information $I(U;Y)$). The term $I(U;Y)$ is referred to as relevance, and $I(U;X)$ is referred to as complexity. It is important to note that this formulation of the IB problem can be seen as a relaxation of the problem of finding U that meets the criteria listed in Section 2.3. More specifically, the *sufficiency* condition $I(U;Y) = I(Y;X)$ is relaxed to simply maximising $I(U;Y)$, which due to the Markov chain is always smaller than $I(Y;X)$.

Finding the representation U that maximises $I(U;Y)$ while ensuring that $I(U;X)$ is kept small can be formulated as the following maximisation problem.

$$\Delta(R) := \max_{P_{U|X}:I(U;X)\leq R} I(U;Y). \tag{2.25}$$

This problem (2.25) can be solved by reformulating the problem as the following Lagrangian problem

$$\mathcal{L}_s^{IB} : \max_{P_{U|X}} I(U;Y) - sI(U;X) \tag{2.26}$$

where $\mathcal{L}_s^{IB}$ can be called the IB objective and s designates the Lagrange multiplier. The trade-off between the relevance and complexity of the IB problem can be seen in Figure 2.6.

For a known joint distribution $P_{X,Y}$ and a given value of s, finding the $P_{U|X}$ that maximises $\mathcal{L}_s^{IB}$ can be done in an iterative manner, similar to the Blahut-Arimoto algorithm of [39, 40], using the following self-consistent equations

$$p(u|x) = p(u)\frac{\exp(-sD_{KL}(P_{Y|x}||P_{Y|u})}{\sum_u p(u)s\exp(-sD_{KL}(P_{Y|x}||P_{Y|u}))}, \tag{2.27a}$$

$$p(u) = \sum_x p(u|x)p(x), \tag{2.27b}$$

$$p(y|u) = \sum_x p(y|x)p(x|u) = \sum_x p(y,x)\frac{p(u|x)}{p(u)}. \tag{2.27c}$$

The IB method starts with some initial condition $p_0(u|x)$, and iteratively updates the probabilities $p(u|x)$, $p(u)$, and $p(y|u)$ using equations (2.27) until convergence. One disadvantage of this technique is that the IB problem is non-convex, and as such the algorithm is guaranteed to converge only locally. While this analytical solution can

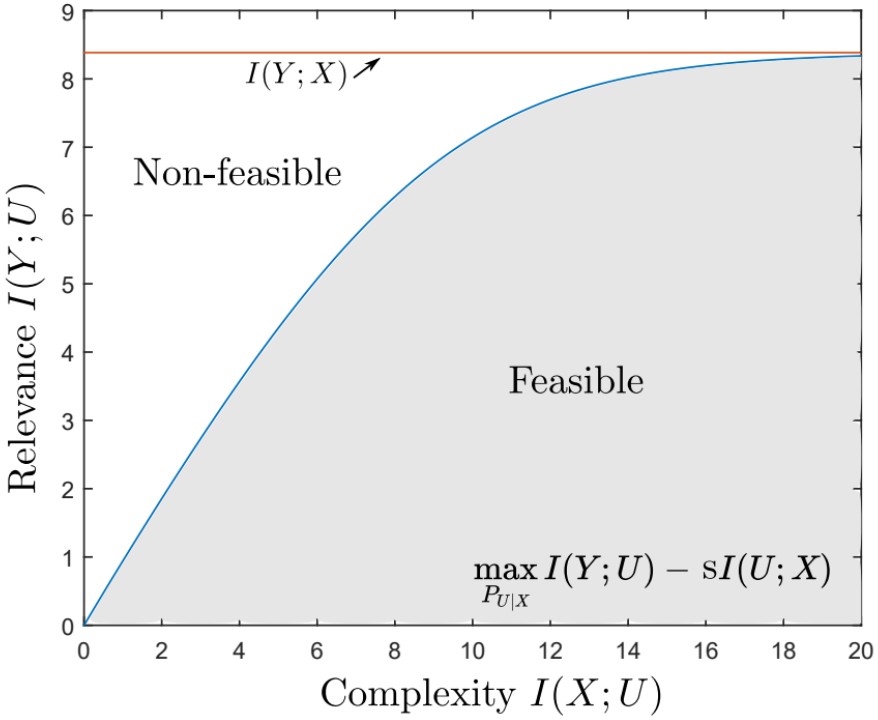


Figure 2.6: The trade-off between the relevance and complexity in the IB problem. The solution to the IB problem of (2.26) results in a relevance-complexity pair on the bound of the feasible region (grey area). Figure taken from [3].

only be used when $P_{X,Y}$ is discrete, there are works that provide analytical solutions to the IB problem when (X, Y) are multivariate Gaussian, see [41–44].

2.3.2 Variational Information Bottleneck

In a learning problem we do not know the joint distribution $P_{X,Y}$, we only have access to a set of training samples $\{(x_i, y_i)\}_{i=1}^n$, which makes computing (2.27) intractable. As a solution, the authors of [24], present a variational lower bound on the IB objective (2.26), which can then be optimised by parameterizing the distributions using NNs, as we will explain in Section 2.3.2.1.

Let us introduce the variational distribution on U, Q_U, and the variational decoder $Q_{Y|U}$. For simplicity, we denote $\mathbf{P} := \{P_{U|X}\}$ and $\mathbf{Q} := \{Q_{Y|U}, Q_U\}$. We define the variational IB cost function as

$$\mathcal{L}_s^{VIB}(\mathbf{P}, \mathbf{Q}) := \mathbb{E}_{P_{X,Y}}\Big[\mathbb{E}_{P_{U|X}}[\log Q_{X|U}] - sD_{KL}(P_{U|X}\|Q_U)\Big]. \tag{2.28}$$

We observe that $I(Y;U) = H(Y) - H(Y|U)$ and that $H(Y)$ is fixed by the problem setting,

as such maximising the cost function (2.26) over $\mathbf{P}$ is equivalent to maximising

$$\tilde{\mathcal{L}}_s^{IB}(\mathbf{P}) := -H(Y|U) - sI(U;X). \tag{2.29}$$

In [24] the authors provide the following relationship between $\tilde{\mathcal{L}}_s^{IB}$ and $\mathcal{L}_s^{VIB}$.

Lemma 2.3.1.

$$\mathcal{L}_s^{VIB}(\mathbf{P},\mathbf{Q}) \leq \tilde{\mathcal{L}}_s^{IB}(\mathbf{P}) \quad \textit{for all pmfs } \mathbf{Q}. \tag{2.30}$$

Furthermore, $\mathbf{Q}$ *that achieves* $\mathcal{L}_s^{VIB}(\mathbf{P},\mathbf{Q}) = \tilde{\mathcal{L}}_s^{IB}(\mathbf{P})$ *is unique and is given by*

$$Q^*_{Y|U} = P_{Y|U}, \;\; Q^*_U = P_U. \tag{2.31}$$

Using Lemma 2.3.1 the optimisation of (2.26) can be written in terms of the variational IB as

$$\max_{\mathbf{P}} \mathcal{L}_s^{IB}(\mathbf{P}) = \max_{\mathbf{P}} \max_{\mathbf{Q}} \mathcal{L}_s^{VIB}(\mathbf{P},\mathbf{Q}). \tag{2.32}$$

2.3.2.1 Learning with the Variational Information Bottleneck

We now look at how $\mathcal{L}_s^{VIB}(\mathbf{P},\mathbf{Q})$ can be used as an objective for a learning algorithm that is capable of training given a dataset $\{(x_i, y_i)\}_{i=1}^n$. Let $P_{\boldsymbol{\theta}}(U|x)$ denote the mapping of observation $x \sim X$ to the latent representation U, which is parameterized by a neural network with parameters (weights and biases) $\boldsymbol{\theta}$. Similarly, we can parameterize the decoding distribution with a NN with parameters $\boldsymbol{\phi}$. We denote the distribution induce by the decoding NN by $Q_{\boldsymbol{\phi}}(Y|U)$. We denote the prior over the latent space as $Q_{\boldsymbol{\varphi}}(U)$ which does not depend on a neural network. Restricting $\mathbf{P}$, $\mathbf{Q}$ to the distributions parameterized by NN, the optimisation of (2.28) becomes

$$\max_{\boldsymbol{\theta},\boldsymbol{\phi},\boldsymbol{\varphi}} \mathbb{E}_{P_{X,Y}}\Big[\mathbb{E}_{P_{\boldsymbol{\theta}}(U|X)}[\log Q_{\boldsymbol{\phi}}(Y|U)] - sD_{KL}(P_{\boldsymbol{\theta}}(U|X)\|Q_{\boldsymbol{\varphi}}(U))\Big] \tag{2.33}$$

The parameterised loss function (2.33) can then be approximated, given a dataset $(x_i, y_i)_{i=1}^n$, by its empirical cost given by

$$\max_{\theta,\phi,\varphi} \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{P_\theta(U_i|x_i)}[\log q_\phi(y_i|U_i)] - sD_{KL}(P_\theta(U_i|x_i)||Q_\varphi(U_i)). \tag{2.34}$$

We now illustrate the training procedure for a possible choice of parametric distributions. Consider the case in which the encoder can be chosen as a multivariate Gaussian, that is, $P_\theta(U|x) = \mathcal{N}(\boldsymbol{\mu}_\theta, \boldsymbol{\Sigma}_\theta)$. In such a case, the NN with parameter $\boldsymbol{\theta}$ would take as input a sample x_i and output the parameters of the Gaussian distribution, that is, the mean $\boldsymbol{\mu}_{\boldsymbol{\theta},i}$ and the covariance $\boldsymbol{\Sigma}_{\boldsymbol{\theta},i}$. For simplicity, we assume that the covariance matrix is diagonal. Let n_u be the dimension of the latent space, that is, $\boldsymbol{\mu}_{\boldsymbol{\theta},i} = [\mu_{\boldsymbol{\theta},i,1}, \ldots, \mu_{\boldsymbol{\theta},i,n_u}]$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta},i} = \text{diag}(\{\sigma^2_{\boldsymbol{\theta},i,k}\}_{k=1}^{n_u})$. For the classification problem, we chose the decoder to be a categorical distribution, that is, the neural network with parameters $\boldsymbol{\phi}$ has a softmax operation in its output layer, which has dimension $|\mathcal{Y}|$. The last layer outputs the probability of each different category of Y. For simplicity, the prior Q_φ is chosen as a multivariate Gaussian $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Calculating the empirical estimate of $\mathbb{E}_{P_\theta(U_i|x_i)}[\log q_\phi(y_i|U_i)]$ can now be done using the Monte Carlo sampling technique and the reparameterization trick of [45] as

$$\mathbb{E}_{P_\theta(U_i|x_i)}[\log q_\phi(y_i|U_i)] \cong \frac{1}{m} \sum_{j=1}^{m} \log q_\phi(y_i|u_{i,j}), \tag{2.35}$$

$$\text{with } u_{i,j} = \mu_{\boldsymbol{\theta},i} + \sqrt{\boldsymbol{\Sigma}_{\boldsymbol{\theta},i}}\epsilon_j, \quad \epsilon_j \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}) \tag{2.36}$$

where m is the number of samples from the Monte Carlo sampling technique and $\mu_{\boldsymbol{\theta},i}$ and $\boldsymbol{\Sigma}_{\boldsymbol{\theta},i}$ are the parameters outputted by the encoding neural network that has observed the sample x_i.

The KL divergence term $D_{KL}(P_\theta(U|X)||Q_\varphi(U))$ can be calculated analytically, since we have chosen both $P_\theta(u|x)$ and Q_φ as multivariate Gaussian distributions. The term $D_{KL}(P_\theta(U|X)||Q_\varphi(U))$ can now be calculated in closed form as follows:

$$D_{KL}(P_\theta(U_i|x_i)||Q_\varphi(U_i)) = \frac{1}{2} \sum_{k=1}^{n_u} \left[\mu_{\boldsymbol{\theta},i,k} - \log \sigma^2_{\boldsymbol{\theta},i,k} - 1 + \sigma^2_{\boldsymbol{\theta},i,k} \right]. \tag{2.37}$$

Together, we have the following cost function for each sample i in the data set $\{(x_i, y_i)\}_{i=1}^n$

$$\mathcal{L}_s^{VIB-NN}(\phi, \theta) = \frac{1}{m}\sum_{j=1}^{m} \log Q_\phi(y_i|u_{i,j}) - \frac{s}{2}\sum_{k=1}^{n_u}\left[\mu_{\theta,i,k} - \log \sigma^2_{\theta,i,k} - 1 + \sigma^2_{\theta,i,k}\right]. \tag{2.38}$$

Finding the parameters that maximise (2.38) can be done using stochastic gradient descent methods (SGD or ADAM [46]). The parameterisation of the distributions using NNs and the sampling procedure described are shown in Figure 2.7. The encoder and

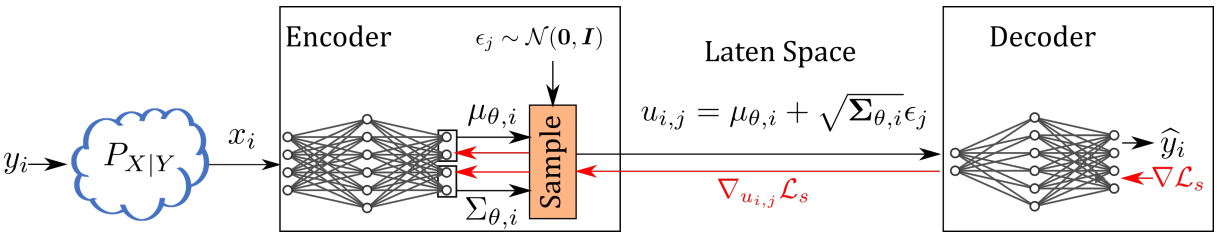


Figure 2.7: The diagram shows the variation information bottleneck when we parameterized the distributions using NNs. Given a sample x_i the encoding NN outputs the parameters of a distribution, in this case a Gaussian. Samples are then drawn from this distribution and an estimate of y_i is obtained by the NN decoding for each sample.

decoder of the VIB solution presented in Figure 2.7 can be deployed at different nodes, allowing the model to be used as a solution to a remote inference problem. Additionally, it can be observed that the training can happen in a distributed manner. During training, the encoding node only needs to transmit the latent representations to the decoding node, while the decoding node only needs to transmit back the error gradients as seen in Figure 2.7. As such, raw data $\{(x_i)\}_{i=1}^n$ does not need to be communicated between the nodes.

2.4 Relationship between Communication and Learning

In this section we look at discussing the connection between communication and learning. That is, we discuss how the search for encoding functions that compress the observed data X, by minimising $I(X, U)$, has a double interpretation. On the one hand, $I(X, U)$ is related to the communication rate; that is, by minimising $I(U, X)$, one would need to transmit fewer bits when communicating U. This connection is shown by connecting the IB problem to the remote source coding problem. On the other hand, $I(X, U)$ can act as a regularizer. Intuitively, by minimising $I(U, X)$ one can reduce the

difference between the performance of the network on the data it is trained and on new data it has not seen before [3, 47].

2.4.1 Information Bottleneck and Remote Source Coding

The rate-distortion region is characterised by the union of all pairs (R, D) that satisfy (2.24), where U is an auxiliary random variable that obeys the Markov chain $Y - X - U$. Using the substitution $\Delta := H(Y) - D$, the region can be written equivalently as the union of all pairs $(R, H(Y) - \Delta)$ that satisfy

$$R \leq I(U; X) \tag{2.39}$$

$$\Delta \geq I(U; Y) \tag{2.40}$$

The boundary of this region is equivalent to the boundary described by (2.26) if solved for all s. As a consequence, the IB problem can essentially be seen as a remote source coding problem with the distortion measured using log-loss. For a more detailed analysis of the connections between the IB problem and the rate distortion problems, see [3, 21, 37, 48]. We can observe that s controls the trade-off between the bit rate constraint R and the desired distortion constraint. As such, by controlling s one can effectively control the overhead that the inference scheme applies on the communication network. This connection has led to multiple lines of work that use the IB method for the problem of inference over a communication channel. For example, in [49] the authors propose an adaptive IB method for designing a joint source and channel encoder-decoder pair. In this method, the parameter s is dynamically adjusted during training to reduce communication requirements while searching for the best balance between communication requirements and network performance. In [50] the authors also use the IB method for designing a variable-length feature encoding scheme based on dynamic neural networks that adaptively adjust to different channel conditions.

2.4.2 Connection between Generalisation Gap and Rate

As discussed in Section 2.3 one of the most common ways of choosing the learning model is to find the model that minimises the empirical estimate of the chosen loss. One

key performance indicator for such a model is how well the chosen model performs on samples outside the observed dataset, referred to as the generalisation gap. As discussed previously, for an encoder f and a decoder g, under log-loss, the expected loss is given by

$$C(f,g) := \mathbb{E}_{P_{X,Y}}\Big[d_{log}(Y, g(f(X)))\Big] \tag{2.41}$$

while, for a given dataset $\{(x_i, y_i)\}_{i=1}^n$, the empirical loss is given by

$$\hat{C}(f,g) := \sum_{i=1}^{n} d_{log}(y_i, g(f(x_i))). \tag{2.42}$$

The generalisation gap for the model (f,g) is given by the difference between the expected loss and the empirical loss

$$gen(f,g) = C(f,g) - \hat{C}(f,g). \tag{2.43}$$

It is commonly believed that one can control the generalisation gap by restricting the complexity of the model. One method is the so-called MDL complexity measure of the model parameters, which is intended to limit the length of the description of the model parameters [51]. An alternative approach,inspired by RD theory, which is used in this thesis and the IB problem [2], considers minimising the description length of the latent representation U, instead of the model paramters. The two problems have already been shown to be connected in [52], where the authors have shown that, for some models, by decreasing the MDL of the parameters, we automatically improve *minimality* (see 2.3) and the disentanglement of the latent representation. These two connections, between the MDL of the weights and the generalisation gap, and between the MDL of the weights and the MDL of the latent representations, intuitively show a relationship between the generalisation gap and the MDL of the latent representation. This connection was formalised in [47], where the authors have shown that the generalisation gap can be bounded in terms of the complexity of the latent representation, as measured by the

[2] It is interesting to note that the original formulation of the IB problem did not consider the connection between the generalisation gap and the description length of the latent representation. This connection was found later.

empirical estimate of mutual information between X and U obtained from the available data set. Let $P(U|X)$ be the distribution of U given X imposed by the chosen encoder f and let $\hat{P}_X^{(n)}$ be the empirical distribution of X given the data set $\{(x_i, y_i)\}_{i=1}^n$. Furthermore, define $\hat{P}_{\hat{Y}|U}^{(n)}$ as the distribution of $\hat{Y}$ given U imposed by the chosen decoder g, and let $P_{\hat{Y}|U}$ be the distribution imposed by the optimal decoder g^* for the true distribution of the data. The generalisation gap is upper-bounded, as

$$gen(f, g) \leq A \sqrt{\hat{I}(\hat{P}_X^{(n)}, P_{U|X})} \frac{\log n}{n} + \frac{B \sqrt{\Lambda\left(P_{U|X}, \hat{P}_{\hat{Y}|U}^{(n)}, P_{\hat{Y}|U}\right)}}{\sqrt{n}} + O(\frac{\log n}{n}) \tag{2.44}$$

where $\hat{I}(\hat{P}_X^{(n)}, P_{U|X})$ is the mutual information between X and U under the distribution $\hat{P}_X^{(n)} P_{U|X}$ and $\Lambda\left(P_{U|X}, \hat{P}_{\hat{Y}|U}^{(n)}, P_{\hat{Y}|U}\right)$ is a function that measures the mismatch between the optimal decoder and the empirical one. A and B are constants.

This bound explicitly suggests that the complexity of the latent representations controls the generalisation gap. As such, the IB problem controls the generalisation gap through the variation of the s parameter. This connection between the generalisation gap and the parameter s was experimentally shown in [24]. In what follows, in this thesis, we will show how the the IB problem can be extended to the distributed case by considering the connection between generalisation and the MDL of the latent representations in an appropriate manner.

Chapter 3

In-network Learning: Star Network Topology

In this chapter, we build on the results of Aguerri-Zaidi [10] and Zaidi et al. [11] by providing a novel comparison of the algorithm of [10] with other distributed learning algorithms. More specifically, in this chapter, we study the distributed inference and training problem shown in Figure 3.1. In this setting, multiple nodes observe data related to the same target variable of interest; data that the nodes need to compress and send directly to a central node so that it can predict the target variable. We restrict the problem to the setting in which each node is equipped with an NN to compress the observed data or predict the target variable; the parameters of the NNs need to be learnt from a set of distributed available dataset.

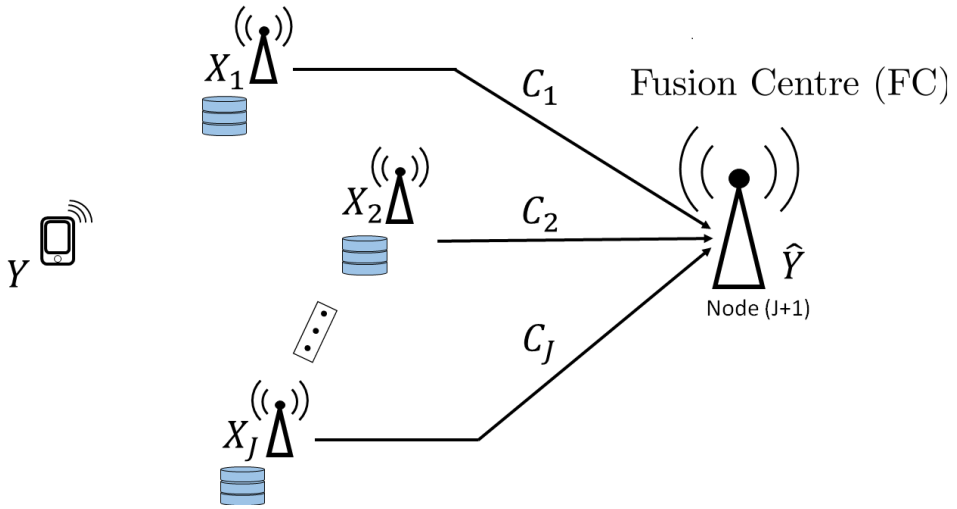


Figure 3.1: Studied distributed inference problem.

This problem was studied in [10], where the authors proposed an algorithm for

training the NN in a distributed manner, the algorithm was named in-network learning (INL) in [14]. In this chapter, we first formally define the problem of Figure 3.1. We also discuss the training procedure for the algorithm of [10], which was first reported in [11]. The contribution of this chapter is that we provide a novel comparison of in-network learning with other distributed leaning algorithms, namely the federated learning (FL) algorithm of [12] and split learning (SL) of [22]. At the end of the chapter, for completeness, we present how [10] derived the loss function for the INL algorithm presented in this chapter.

3.1 Problem Setup

Let us formally introduce the inference problem shown in Figure 3.1. There are nodes $J \geq 1$ which possess or can acquire data relevant for inference on a random variable of interest Y, taking values in $\mathcal{Y}$. Let $\mathcal{J} = \{1, \ldots, J\}$ denote the set of such nodes, and X_j denote the data observed by node j, taking values in $\mathcal{X}_j$. The relationship between the random variable of interest Y and the observed ones $X_1, \ldots, X_J$ is given by the joint probability mass function $P_{X_{\mathcal{J}},Y} := P_{X_1,\ldots,X_J,Y}(x_1, \ldots x_J, y)$, with $(x_1, \ldots, x_j) \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_J$ and $y \in \mathcal{Y}$. The inference on Y needs to be done at some distant node (say, node $(J+1)$) which is connected to the nodes that possess raw data through error-free links of given finite capacities; and has to be performed without any sharing of raw data. The network may represent, for example, a wired network or a wireless mesh network operated in time or frequency division. The processing at node $j \in \mathcal{J}$ is a mapping

$$\phi_j : \mathcal{X}_j \longrightarrow \mathcal{U}_j; \tag{3.1}$$

and that at node $(J+1)$ is a mapping

$$\psi : \mathcal{U}_1 \times \ldots \times \mathcal{U}_J \longrightarrow \hat{\mathcal{Y}}. \tag{3.2}$$

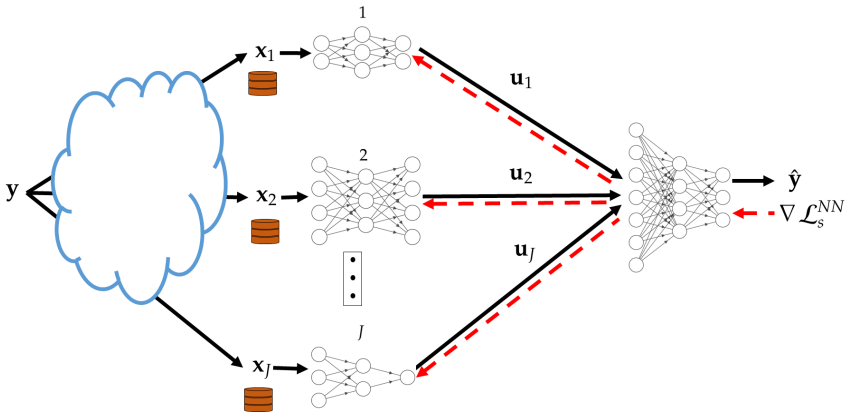
with $|\mathcal{U}_j| \leq 2^{R_j}$. As mentioned in Section 2.2.3.1, we choose the reconstruction set $\hat{\mathcal{Y}}$ to be the set of distributions on $\mathcal{Y}$, i.e., $\hat{\mathcal{Y}} = \mathcal{P}(\mathcal{Y})$; and we measure the discrepancies between the true values of $Y \in \mathcal{Y}$ and their estimated fits in terms of average logarithmic

loss, i.e., for $(y, \hat{P}) \in \mathcal{Y} \times \mathcal{P}(\mathcal{Y})$

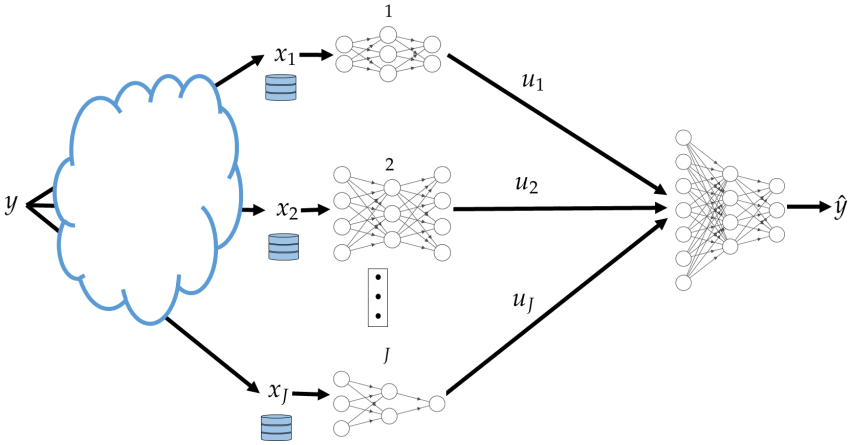
$$d(y, \hat{P}) = \log \frac{1}{\hat{P}(y)}. \tag{3.3}$$

We study the case in which the mappings given by (3.1) and (3.2) need to be learnt from a set of training data samples $\{(x_{1,i}, \ldots, x_{J,i}, y_i)\}_{i=1}^{n}$. The data are distributed so that the samples $\mathbf{x}_j := (x_{j,1} \ldots, x_{j,n})$ are available at node j for $j \in \mathcal{J}$ and the desired predictions $\mathbf{y} := (y_1 \ldots y_n)$ are available at the node $(J+1)$.

3.2 In-network Learning Solution



(a) Training phase



(b) Inference phase

Figure 3.2: In-network learning for the network model of Figure 3.1.

To overcome the issue of unknown distribution, we parameterize the possibly stochastic mappings (3.1) and (3.2) using neural networks. This is shown in Figure 3.2. The NNs at the various nodes are arbitrary and can be chosen independently. It is only required that the following mild condition be met, which, as will become clearer in

what follows, facilitates the backpropagation.

$$\sum_{j=1}^{J}(\text{Size of last layer of NN } j) = \text{Size of first layer of NN (J+1)}. \tag{3.4}$$

3.2.1 Loss Function

A possible suitable loss function was shown to be given by [10]

$$\begin{aligned}\mathcal{L}_s^{\text{NN}}(n) := &\frac{1}{n}\sum_{i=1}^{n}\log Q_{\phi_{\mathcal{J}}}(y_i|u_{1,i},\dots,u_{J,i}) \\ &+\frac{s}{n}\sum_{i=1}^{n}\sum_{j=1}^{J}\left(\log Q_{\phi_j}(y_i|u_{j,i}) - \log\left(\frac{P_{\theta_j}(u_{j,i}|x_{j,i})}{Q_{\varphi_j}(u_{j,i})}\right)\right),\end{aligned} \tag{3.5}$$

where s is a Lagrange parameter and for $j \in \mathcal{J}$ the distributions $P_{\theta_j}(u_j|x_j)$, $Q_{\phi_j}(y|u_j)$, $Q_{\phi_{\mathcal{J}}}(y|u_{\mathcal{J}})$ are variational ones whose parameters are determined by the chosen NNs using the re-parametrization trick of [45]; and $Q_{\varphi_j}(u_j)$ are priors known to the encoders. As described in Section 2.3.2 the encoding distibution can be choosen as multivartiate gaussian denoting by f_{θ_j} the NN used at node $j \in \mathcal{J}$ whose (weight and bias) parameters are given by $\boldsymbol{\theta}_j$, for regression problems the conditional distribution $P_{\theta_j}(u_j|x_j)$ can be chosen to be multivariate Gaussian, i.e., $P_{\theta_j}(u_j|x_j) = \mathcal{N}(u_j; \boldsymbol{\mu}_j^{\theta}, \boldsymbol{\Sigma}_j^{\theta})$. For discrete data, concrete variables (i.e., Gumbel-Softmax) can be used instead.

The rationale behind the choice of loss function (3.5) is presented in detail in Section 3.4. However, intuitively, the authors of [10] have shown that in the regime of large n, if the encoders and decoder are not restricted to use NNs under some conditions [1] the optimal stochastic mappings $P_{U_j|X_j}$, P_U, $P_{Y|U_j}$ and $P_{Y|U_{\mathcal{J}}}$ are found by marginalising the joint distribution that maximises the following Lagrange cost function [10, Proposition 2]

$$\mathcal{L}_s^{DIB} = -H(Y|U_{\mathcal{J}}) - s\sum_{j=1}^{J}\Big[H(Y|U_j) + I(U_j;X_j)\Big]. \tag{3.6}$$

where the maximisation is over all joint distributions of the form $P_Y\prod_{j=1}^{J}P_{X_j|Y}\prod_{j=1}^{J}P_{U_j|X_j}$.

[1] The optimality is proved therein under the assumption that for every subset $\mathcal{S} \subseteq \mathcal{J}$ it holds that $X_{\mathcal{S}} - Y - X_{\mathcal{S}^c}$. The RHS of (3.6) is achievable for arbitrary distributions, however, regardless of such an assumption.

3.2.2 Training Algorithm

We now describe the training algorithm for INL which was first reported in the patent [11] by Zaidi, Scaman and Escamilla. During the forward pass, every node $j \in \mathcal{J}$ processes mini-batches of size, say, b_j of its training data-set $\mathbf{x}_j$. Node $j \in \mathcal{J}$ then sends a vector whose elements are the activation values of the last layer of (NN j). Due to (3.4) the activation vectors are vertically concatenated at the NN input layer (J + 1). The forward pass continues on the NN (J+1) until the last layer of the latter. The parameters of NN (J+1) are updated using standard backpropgation. Specifically, let L_{J+1} denote the index of the last layer of NN $(J+1)$. Also, let, for $l \in [2 : L_{J+1}]$, $\mathbf{w}_{J+1}^{[l]}$, $\mathbf{b}_{J+1}^{[l]}$ and $\mathbf{a}_{J+1}^{[l]}$ denote, respectively, the weights, biases and activation values at layer l for the NN $(J+1)$; and σ is the activation function. Node $(J+1)$ computes the error vectors

$$\boldsymbol{\delta}_{J+1}^{[L_{J+1}]} = \nabla_{\mathbf{a}_{J+1}^{[L_{J+1}]}} \mathcal{L}_s^{NN}(b) \odot \sigma'(\mathbf{w}_{J+1}^{[L_{J+1}]} \mathbf{a}_f^{[L_{(J+1)}-1]} + \mathbf{b}_{J+1}^{[L_{J+1}]}) \tag{3.7a}$$

$$\boldsymbol{\delta}_{J+1}^{[l]} = [(\mathbf{w}_{J+1}^{[l+1]})^T \boldsymbol{\delta}_{J+1}^{[l+1]}] \odot \sigma'(\mathbf{w}_{J+1}^{[l]} \mathbf{a}_{J+1}^{[l-1]} + \mathbf{b}_{J+1}^{[l]}) \ \forall \ l \in [2, L_{J+1} - 1] \tag{3.7b}$$

$$\boldsymbol{\delta}_{J+1}^{[1]} = [(\mathbf{w}_{J+1}^{[2]})^T \boldsymbol{\delta}_{J+1}^{[2]}], \tag{3.7c}$$

and then updates its weight- and bias parameters as

$$\mathbf{w}_{J+1}^{[l]} \rightarrow \mathbf{w}_{J+1}^{[l]} - \eta \boldsymbol{\delta}_{J+1}^{[l]} (\mathbf{a}_{J+1}^{[l-1]})^T, \tag{3.8a}$$

$$\mathbf{b}_{J+1}^{[l]} \rightarrow \mathbf{b}_{J+1}^{[l]} - \eta \boldsymbol{\delta}_{J+1}^{[l]}, \tag{3.8b}$$

where η designates the learning parameter [2].

Remark 3.2.1. *It is important to note that for the computation of the RHS of* (3.7a) *node* $(J+1)$*, which knows* $Q_{\phi_{\mathcal{J}}}(y_i|u_{1i}, \ldots, u_{Ji})$ *and* $Q_{\phi_j}(y_i|u_{ji})$ *for all* $i \in [1 : n]$ *and all* $j \in \mathcal{J}$*, only the derivative of* $\mathcal{L}_s^{NN}(n)$ *w.r.t. the activation vector* $\mathbf{a}_{J+1}^{L_{J+1}}$ *is required. For instance, node* $(J+1)$ *does not need to know any of the conditional variationals* $P_{\theta_j}(u_j|x_j)$ *or the priors* $Q_{\varphi_j}(u_j)$*.*

The backward propagation of the error vector from node $(J+1)$ to the nodes j, $j = 1, \ldots, J$, is as follows. Node $(J+1)$ splits horizontally the error vector of its input layer into J sub-vectors with sub-error vector j having size L_j, the dimension of the last layer of NN j [recall (3.4) and that the activation vectors are concatenated vertically

[2] For simplicity η and σ are assumed here to be identical for all NNs.

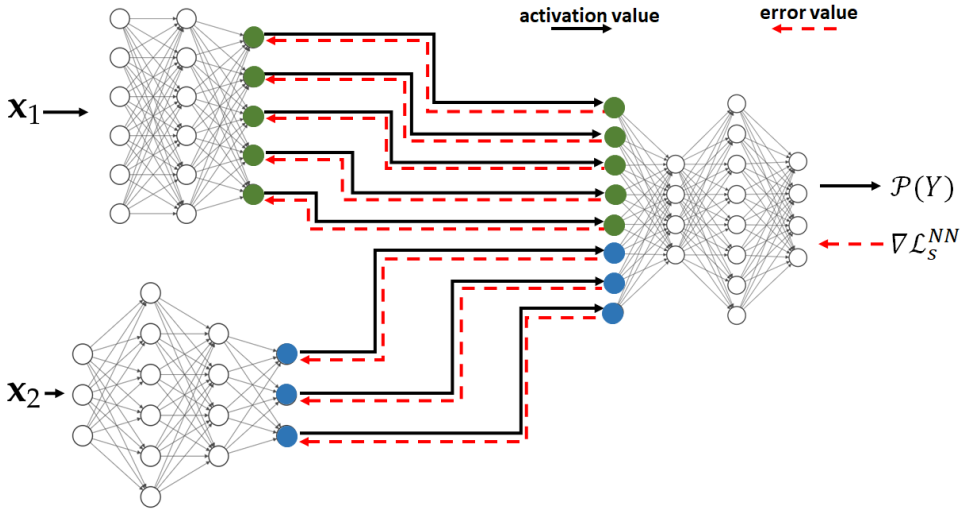


Figure 3.3: Illustration of the forward and backward passes for an example in-network learning with $J = 2$.

during the forward pass]. See Figure 3.3. The backward propagation then continues on each of the J input NNs simultaneously, each of them essentially applying operations similar to (3.7) and (3.8).

Remark 3.2.2. *Let $\boldsymbol{\delta}_{J+1}^{[1]}(j)$ denote the sub-error vector sent back from node $(J+1)$ to node $j \in \mathcal{J}$. It is easy to see that, for every $j \in \mathcal{J}$,*

$$\nabla_{\mathbf{a}_j^{L_j}} \mathcal{L}_s^{NN}(b_j) = \boldsymbol{\delta}_{J+1}^{[1]}(j) - s\nabla_{\mathbf{a}_j^{L_j}} \left(\sum_{i=1}^{b} \log \left(\frac{P_{\theta_j}(u_{j,i}|x_{j,i})}{Q_{\varphi_j}(u_{j,i})} \right) \right); \tag{3.9}$$

and this explains why node $j \in \mathcal{J}$ needs only the part $\boldsymbol{\delta}_{J+1}^{[1]}(j)$, not the entire error vector at node $(J+1)$.

3.2.3 Inference Algorithm

We now describe the inference phase for INL which was first reported in the patent [11] by Zaidi, Scaman and Escamilla. During this phase, node j observes a new sample x_j. It uses its NN to output an encoded value u_j which it sends to the decoder. After collecting $(u_1, \cdots, u_J)$ from all input NNs, node $(J+1)$ uses its NN to output an estimate of Y in the form of soft output $Q_{\phi_{\mathcal{J}}}(Y|u_1, \ldots, u_J)$. The procedure is depicted in Figure 3.2b.

Remark 3.2.3. *A suitable practical implementation in wireless settings can be obtained using Orthogonal Frequency Division Multiplexing (OFDM). That is, the J input nodes are allocated non-overlapping bandwidth segments and the output layers of the corresponding NNs are*

chosen accordingly. The encoding of the activation values can be done, e.g., using entropy type coding [53].

3.3 Comparison to Federated Learning and Split Learning

In this section we compare the proposed INL algorithm with other distributed learning frameworks. We first start by introducing the federated learning (FL) algorithm of [12] and the split learning (SL) algorithm of [22]. We then discuss the amount of data the three algorithms use for communication during training. Finally we compare the three algorithm experimentally.

3.3.1 Review of Distributed Learning Algorithms

3.3.1.1 Federated Learning

FL is an alternative approach to the problem of distributed machine learning. The aim of the FL framework is for the nodes to train one model that can then be used, during inference, by each node locally. Each edge device has a set of local data and a copy of the NN model to be trained, common across all devices. Compared to the in-network learning setting presented before, in FL each node has access to the same feature space as the other nodes, however, observes different samples. More specifically, given the same dataset $\{(x_{1,i}, \ldots, x_{J,i}, y_i)\}_{i=1}^{n}$, in FL each node $j \in \mathcal{J}$ observes a subset $\{(x_{1,i}, \ldots, x_{J,i}, y_i)\}_{i \in \mathcal{N}_j}$ of the samples, with $\mathcal{N}_j \in [1 : n]$. During the training phase, all edge devices simultaneously train their local copy of the NN on their available dataset. Then, each edge device sends the learned NN parameters to the central device, called a cloud- or parameter server (PS), which aggregates them, by simply computing their average. The process repeats, every time reinitialising using the obtained aggregated model, until convergence. The rationale is that this way, the model is progressively adjusted to account for all variations in the data, not only those of the local dataset.

3.3.1.2 Split Learning

SL was presented in [22] as an alternative to FL. In SL a two-part NN model is split into an encoder part and a decoder part is learnt sequentially. The central device does

not have its own data and hosts the decoder part of the NN. The edge devices have access to a different part of the dataset, split in a manner similar to FL, and a copy of the encoding NN. The central device keeps track of the last trained edge device during training and passes the information to the next device to be trained. In every round, the edge device which will be trained has its parameters initialised using those learnt by the previous device. The parameters are exchanged between the devices directly or through the central node. The edge device then trains the two-part NN, together with the central device, on its distinct dataset using classical gradient descent methods. The edge device passes the data through its NN and sends forward to the central device the activation values of the last layer. The central device uses the received data as input into its NN and outputs an estimate. The central device then backpropagates the error through its NN and sends the error vector back from its input layer to the edge device, which continues the procedure on its own NN. During the inference phase, the edge device observing data connects to the central node and the two jointly perform the inference. SL reduces some of the computation load placed on each edge node by the FL algorithm, while also reducing some of the communication overhead. This was shown in [22]. One disadvantage, however, is that the model on each node cannot train simultaneously and as such the model would take more time to train.

3.3.2 Bandwidth Requirements

In this section, we study the requirements in bandwidth of our in-network learning, which was first reported in the patent [11] by Zaidi, Scaman and Escamilla. Let q denote the size of the entire data set (each input node has a local dataset of size $\frac{q}{J}$), $p = L_{J+1}$ the size of the input layer of NN $(J+1)$ and s the size in bits of a parameter. Since as per (3.4), the output of the last layers of the input NNs are concatenated at the input of NN $(J+1)$ whose size is p, and each activation value is s bits, one then needs $\frac{2sp}{J}$ bits for each data point – the factor 2 accounts for both the forward and backward passes; and, so, for an epoch our in-network learning requires $\frac{2pqs}{J}$ bits. Note that the bandwidth requirement of in-network learning does not depend on the sizes of the NNs used at the various nodes, but does depend on the size of the dataset. For comparison, notice that with FL one would require $2NJs$, where N designates the number of (weight and

bias) parameters of an NN at one node. For SL, assuming for simplicity that the NNs $j = 1, \ldots, J$ all have the same size ηN, where $\eta \in [0, 1]$, SL requires $(2pq + \eta NJ)s$ bits for an entire epoch. The bandwidth requirements of the three schemes are summarised and compared in Table 3.1 for two popular neural networks, VGG16 ($N = 138,344,128$ parameters) and ResNet50 ($N = 25,636,712$ parameters) and two example datsets, $q = 50,000$ data points and $q = 500,000$ data points. The numerical values are set as $J = 500$, $p = 25088$ and $\eta = 0.88$ for ResNet50 and 0.11 for VGG16.

	Federated learning	Split learning	In-network learning
Bandwidth requirement	$2NJs$	$(2pq + \eta NJ)\, s$	$\frac{2pqs}{J}$
VGG 16 50,000 data points	4427 Gbits	324 Gbits	0.16 Gbits
ResNet 50 50,000 data points	820 Gbits	441 Gbits	0.16 Gbits
VGG 16 500,000 data points	4427 Gbits	1046 Gbits	1.6 Gbits
ResNet 50 500,000 data points	820 Gbits	1164 Gbits	1.6 Gbits

Table 3.1: Bandwidth requirements of INL, FL and SL.

3.3.3 Experiments

We perform two series of experiments. In both cases, the used dataset is the CIFAR-10 and there are five client nodes. In this setup, we create five sets of noisy versions of the images of CIFAR-10. To this end, the CIFAR images are first normalised and then corrupted by additive Gaussian noise with standard deviation set respectively to $0.4, 1, 2, 3, 4$. For our INL each of the five input NNs is trained on a different noisy version of the same image. Each NN uses a variation of the VGG network of [54], with the categorical cross-entropy as the loss function, L2 regularisation, and Dropout and BatchNormalization layers. Node $(J + 1)$ uses two dense layers. The architecture is shown in Figure 3.4. In the experiments, all five (noisy) versions of every CIFAR-10 image are processed simultaneously, each by a different NN at a distinct node, through a series of convolutional layers. The outputs are then concatenated and then passed through a series of dense layers at node $(J + 1)$.

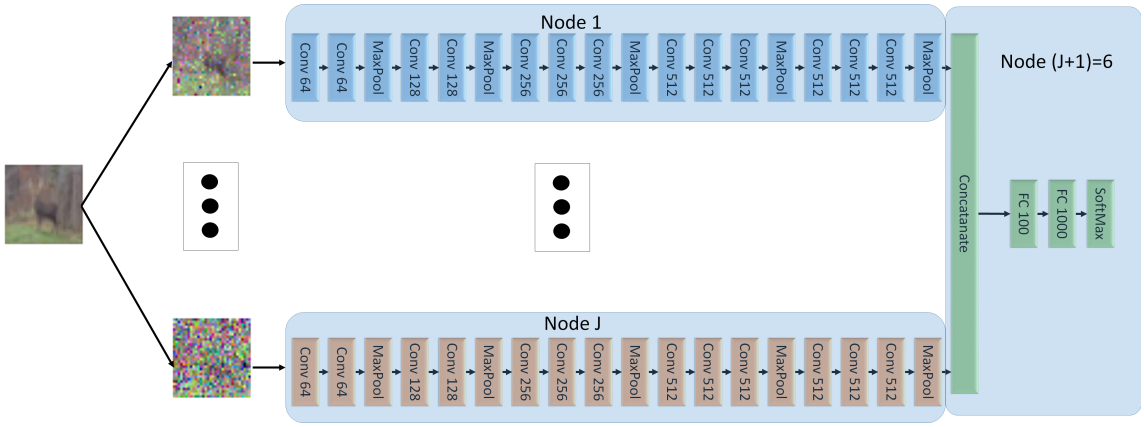
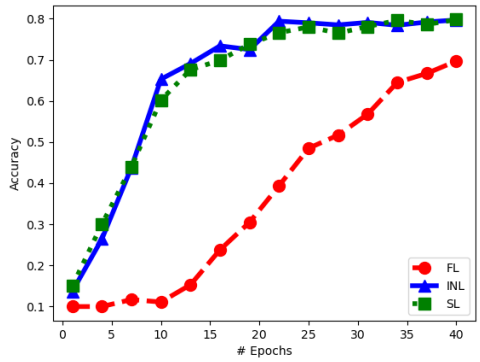


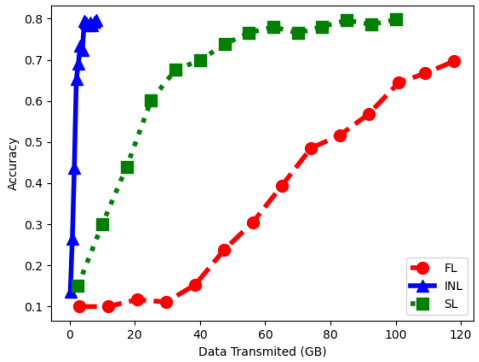
Figure 3.4: Network architecture. *Conv* stands for a convolutional layer, *Fc* stands for a fully connected layer.

3.3.3.1 Experiment 1

For FL, each of the five client nodes is equipped with the *entire* network of Figure 3.4. The dataset is split into five sets of equal sizes; and the split is now performed such that all five noisy versions of a same CIFAR-10 image are presented to the same client NN (distinct clients observe different images, however). For SL of [22], each input node is equipped with an NN formed by *all* fives branches with convolution networks (i.e., all the network of Fig. 3.4, except the part at Node $(J+1)$); and node $(J+1)$ is equipped with fully connected layers at Node $(J+1)$ in Figure 3.4. Here, the processing during training is such that each input NN concatenates vertically the outputs of all convolution layers and then passes that to node $(J+1)$, which then propagates back the error vector. After one epoch at one NN, the learnt weights are passed to the next client, which performs the same operations on its part of the dataset. Figure 3.5a depicts the evolution of the



(a) Accuracy vs. # of epochs.



(b) Accuracy vs. bandwidth cost.

Figure 3.5: Comparison of INL, FL and SL - Experiment 1.

classification accuracy on CIFAR-10 as a function of the number of training epochs,

for the three schemes. As visible from the figure, the convergence of FL is relatively slower comparatively. Also, the final result is less accurate. Figure 3.5b shows the amount of data needed to be exchanged among the nodes (i.e., bandwidth resources) in order to get a prescribed value of classification accuracy. Observe that both our INL and SL require significantly less data exchange than FL; and our INL is better than SL especially for small values of bandwidth.

3.3.3.2 Experiment 2

In Experiment 1, the entire training dataset was partitioned differently for INL, FL and SL (in order to account for the particularities of the three). In this second experiment, they are trained on the same data. Specifically, each client NN sees all CIFAR-10 images during training; and its local dataset differs from those seen by other NNs only by the amount of added Gaussian noise (standard deviation chosen, respectively, as $0.4, 1, 2, 3, 4$). Also, for the sake of a fair comparison between INL, FL and SL the nodes are set to utilize fairly the same NNs for the three of them (see, Fig. 3.6). Figure 3.7 shows the performance of the three schemes during the inference phase in this case (for FL and SL the inference is performed on an image which has average quality of the five noisy input images for INL). Again, observe the benefits of INL over FL and SL in terms of both achieved accuracy and bandwidth requirements.

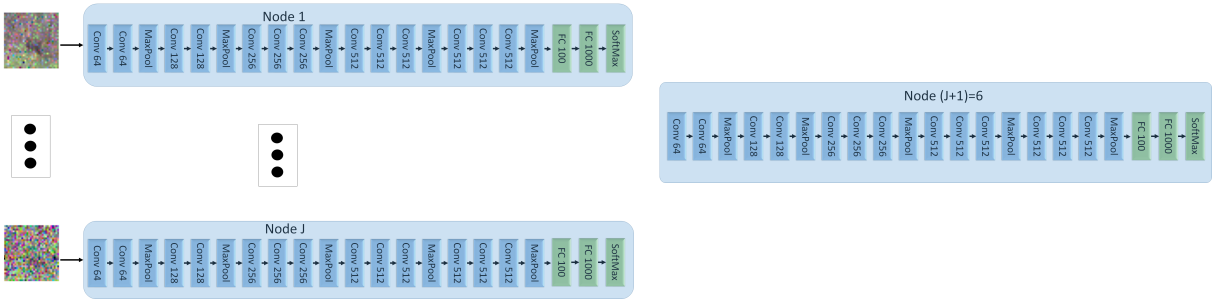
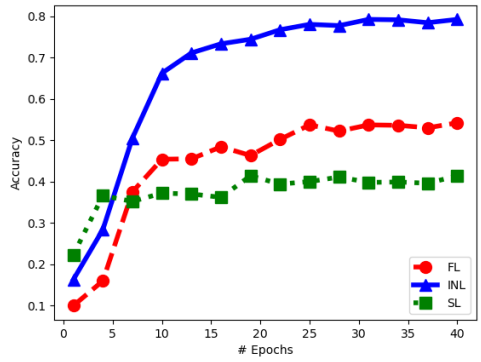


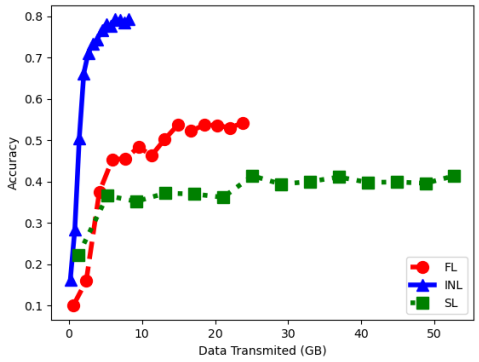
Figure 3.6: Used NN architecture for FL in Experiment 2.

3.4 Derivation of In-network Learning Loss Function

As mention in the introduction in this section, we restate the derivation of the loss function (3.5) presented in Aguerri-Zaidi [10] for completeness. The review is brief and for more details please see [10]. The distributed inference problem presented



(a) Accuracy vs. # of epochs.



(b) Accuracy vs. bandwidth cost.

Figure 3.7: Comparison of INL, FL and SL - Experiment 2.

is related to the J-encoder CEO problem under log-loss presented in [28]. One of the main differences is that in the CEO problem setting n samples are compressed and transmitted jointly, as opposed to just one sample as is the case in an inference problem. We first introduce the CEO problem, and then we present how known results for the CEO problem were used by [10] to derive the loss function for the distributed training and inference problem given by (3.5).

3.4.1 CEO Problem Setup

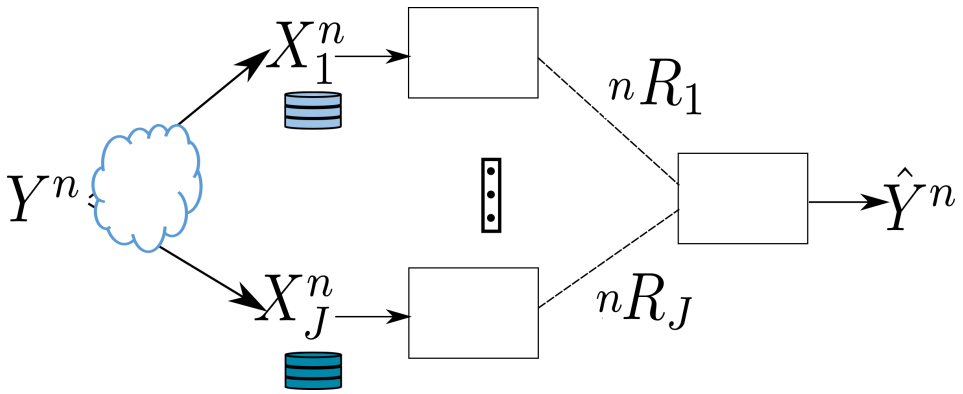


Figure 3.8: CEO problem.

The J-encoder CEO problem is shown in Figure 3.8. Compared to the problem presented at the beginning of the section, in the CEO problem each node observes a sequence of n independent samples of the random variables, i.e. $(Y^n, X_1^n, \ldots, X_J^n) \sim \prod_{i=1}^n P_{X_\mathcal{J},Y}$. The central node is interested in recovering the random variable of interest Y^n, taking values in $\mathcal{Y}^n$, which is related to the observed random variables X_j^n, taking values in $\mathcal{X}_j^n$,through the joint probability mass function $P_{X_\mathcal{J}^n,Y^n} := \prod_{i=1}^n P_{X_\mathcal{J},Y}$. We

additionally assume that for all $\mathcal{S} \subseteq \mathcal{J}$ we have $X_{\mathcal{S}} - Y - X_{\mathcal{S}^c}$, i.e. the data observed by each encoder is independent of the rest of the observation, given Y. Each encoder $j \in \mathcal{J}$ can use nR_j bits to encode the observed data X_j^n. We choose the reconstruction set $\hat{\mathcal{Y}}^n$ to be the set of distributions on $\mathcal{Y}^n$, i.e., $\hat{\mathcal{Y}}^n = \mathcal{P}(\mathcal{Y}^n)$; and we measure the discrepancies between the true values of $Y^n \in \mathcal{Y}^n$ and their estimated fits in terms of their per sample average logarithmic loss, i.e., for $(y^n, \hat{y}^n) \in \mathcal{Y}^n \times \mathcal{P}(\mathcal{Y}^n)$

$$d^{(n)}(y^n, \hat{y}^n) = \frac{1}{n} \sum_{i=1}^{n} \log \frac{1}{\hat{y}_i(y_i)}. \tag{3.10}$$

Definition 3.4.1. *A rate distortion code (of block length n) for the J-encoder model described consists of the encoding functions*

$$\phi_j^n : \mathcal{X}_j^n \longrightarrow [1 : 2^{nR_j}]; \tag{3.11}$$

and the decoding function at node $(J + 1)$

$$\psi : [1 : 2^{nR_1}] \times \ldots \times [1 : 2^{nR_J}] \longrightarrow \hat{\mathcal{Y}}^n. \tag{3.12}$$

Definition 3.4.2. *A rate-distortion tuple* $(R_1, \ldots, R_J, D)$ *is achievable if there exist a block length n, and encoding and decoding functions such that*

$$R_j \geq \frac{1}{n} \log M_j^{(n)}, \quad \textit{for } j \in \mathcal{J}, \tag{3.13}$$

$$D \leq \mathbb{E}\Big[d^{(n)}(Y^n, \psi^n(\phi_1^n(X_1^n), \ldots \phi_J^n(X_J^n)))\Big] \tag{3.14}$$

The rate-distortion region of the defined CEO problem, denoted as $\mathcal{RI}_{CEO}$*, is defined as the closure of all non-negative* $(R_1, \ldots, R_J, D)$ *rate-distortion tuples that are achievable.*

The following theorem characterised the rate-distortion region of the introduced CEO problem, the region was first characterised in [28].

Theorem 3.4.1. *[28, Theorem 10] The rate-distortion region of the described CEO problem is given by the union of all tuples* $(R_1, \ldots, R_J, D)$ *satisfying for all* $\mathcal{S} \subseteq \mathcal{J}$

$$\sum_{j \in \mathcal{S}} R_j + D \leq \sum_{j \in \mathcal{S}} I(X_j; U_j | Y, Q) + H(Y | U_{\mathcal{S}^c}, Q) \tag{3.15}$$

for some set of pmfs $\mathbf{P} := \{P_{U_1|X_1,Q}, \ldots, P_{U_J|X_J,Q}, P_Q\}$ *with the joint distribution of the form* $p(y)p(q)\prod_{j=1}^{J} p(x_j|y)p(u_j|x_J,q)$.

3.4.2 Distributed Information Bottleneck

The authors of [10] consider the problem in which the logarithmic distortion of the presented CEO problem is replaced by the mutual information constraint

$$\Delta \leq \frac{1}{n} I\Big(Y^n, \psi^n(\phi_1^n(X_1^n), \ldots \phi_J^n(X_J^n))\Big) \tag{3.16}$$

where Δ denotes the relevance of the model. Similar to the CEO problem, one now wants to find the set of achivable relevance-complexity tuples $(R_1, \ldots, R_J, \Delta)$, this can be seen as a generalisation of the IB problem [21] to the distributed case.

Definition 3.4.3. *A relevance complexity tuple* $(R_1, \ldots, R_J, \Delta)$ *is achievable if there exist a block length* n*, and encoding and decoding functions such that*

$$R_j \geq \frac{1}{n} \log M_j^{(n)}, \quad \textit{for } j \in \mathcal{J}, \tag{3.17}$$

$$\Delta \leq \frac{1}{n} I(Y^n, \psi^n(\phi_1^n(X_1^n), \ldots \phi_J^n(X_J^n)))\Big] \tag{3.18}$$

The relevance complexity region, denoted as $\mathcal{RI}_{DIB}$ *is defined as the closure of all non-negative* $(R_1, \ldots, R_J, \Delta)$ *relavance-complexity tuples that are achievable.*

By showing the equivalence between the two regions, $\mathcal{RI}_{DIB}$ and $\mathcal{RI}_{CEO}$, the authors of [10] are able to characterise the $\mathcal{RI}_{DIB}$ region as follows

Theorem 3.4.2. *[10, Theorem 1] The relevance-complexity region, denoted as* $\mathcal{RI}_{DIB}$*, is given by the union of all tuples* $(R_1, \ldots, R_J, \Delta)$ *satisfying for all* $\mathcal{S} \subseteq \mathcal{J}$

$$\sum_{j \in \mathcal{S}} R_j \leq \Delta + \sum_{j \in \mathcal{S}} I(X_j; U_j|Y, Q) + I(Y; U_{\mathcal{S}^c}, Q) \tag{3.19}$$

for some set of pmfs $\mathbf{P} := \{P_{U_1|X_1,Q}, \ldots, P_{U_J|X_J,Q}, P_Q\}$ *with the joint distribution of the form* $p(y)p(q)\prod_{j=1}^{J} p(x_j|y)p(u_j|x_J,q)$.

For simplicity, they consider the region of achievable relevance-complexity tuples

under sum-complexity constrained, defined by

$$\mathcal{RI}_{DIB}^{sum} := \{(\Delta, R_{sum}) \exists (R_1, \dots, R_J) \text{s.t.} (\Delta, R_1, R_J) \in \mathcal{RI}s_{DIB} \text{and } R_{sum} = \sum_{j=1}^{J} R_j\} \tag{3.20}$$

In practice, we are interested in finding the pmfs $\mathbf{P} := \{P_{U_1|X_1,Q}, \dots, P_{U_J|X_J,Q}, P_Q\}$ for which the resulting (Δ, R_{sum}) is on the boundary of the $\mathcal{RI}_{DIB}^{sum}$. The following proposition provides such a characterisation.

Proposition 3.4.1. *[10, Proposition 2] For each tuple (Δ, R_{sum}) on the boundary of the relevance complexity region $\mathcal{RI}_{DIB}^{sum}$ there exists $s \geq 0$ such that $(\Delta, R_{sum}) = (\Delta_s, R_s)$ where*

$$\Delta_s := \frac{1}{1+s}\Big[(1+sK)H(Y) + sR_s + \max_{\mathbf{P}} \mathcal{L}_s^{DIB}(\mathbf{P})] \tag{3.21}$$

$$R_s := I(Y; U_{\mathcal{J}}^*) + \sum_{j=1}^{\mathcal{J}} \Big[I(X_j; U_j^*) - I(Y; U_j^*)] \tag{3.22}$$

and $\mathbf{P}^*$ is the set of pmfs that maximize the cost function

$$\mathcal{L}_s^{DIB} = -H(Y|U_{\mathcal{J}}) - s\sum_{j=1}^{J} \Big[H(Y|U_j) + I(U_j; X_j)\Big]. \tag{3.23}$$

It can be seen from (3.23) that, similar to the IB problem, by controlling s one can control the trade-off between the sum complexity of the latent representation $U_{\mathcal{J}}$ and the relevance of the model. Unfortunately, optimising (3.23) requires computing the marginal distributions $P_{Y|U_1,\dots,U_{\mathcal{J}}}$,$P_{Y|U_1}$,$\dots$,$P_{Y|U_{\mathcal{J}}}$, which is not always possible. To overcome this issue, one can introduce the variational distributions

$$\mathbf{Q} := \Big\{Q_{Y|U_1}, Q \dots, QY|U_J, Q_{Y|U_1,\dots,U_J}, Q_{U_1}, \dots, Q_{U_J}\Big\} \tag{3.24}$$

and the variation cost function given by

$$\mathcal{L}_s^{VDIB}(\mathbf{P}, \mathbf{Q}) := \mathbb{E}[\log Q_{Y|U_{\mathcal{J}}}(Y|U_{\mathcal{J}})] + s\sum_{j=1}^{J} \Big(\mathbb{E}[\log Q_{Y|U_j}(Y|U_j)] - D_{KL}(P_{U_j|X_j} \| Q_{U_j})\Big) \tag{3.25}$$

It was show in [10] that the optimisation of (3.23) can be written in terms of the

variational cost function as

$$\max_{\mathbf{P}} \mathcal{L}_s^{DIB} = \max_{\mathbf{P}} \max_{\mathbf{Q}} \mathcal{L}_s^{VDIB}(\mathbf{P}, \mathbf{Q}) \tag{3.26}$$

Finally we restrict the maximisation problem

$$\max_{\mathbf{P}} \max_{\mathbf{Q}} \mathcal{L}_s^{VDIB}(\mathbf{P}, \mathbf{Q}) \tag{3.27}$$

to the encoding distribution $\mathbf{P}$ and variations distribution $\mathbf{Q}$ which can be parameterized by neural networks as described in Section 2.3.2. More specifically, let $P_{\theta_j}(U_j|X_j)$ be the encoding distribution at node $j \in \mathcal{J}$ parameterized by an NN with parameters $\boldsymbol{\theta}_j$. Additionally, let $\log Q_{\phi_j}(Y|U_j), j \in \mathcal{J}$, be the decoding mapping from U_j to Y given by an NN with parameters $\boldsymbol{\phi}_j$ and let $\log Q_{\phi_{\mathcal{J}}}(Y|U_1, \ldots, U_J)$ be the main decoding distribution given by an NN with parameters $\boldsymbol{\phi}_{\mathcal{J}}$. Finally, let $Q_{\varphi_j}(U_j), j \in \mathcal{J}$, be some fixed prior over the latent space. By imposing such a restriction, the maximisation problem of (3.27) becomes

$$\max_{\boldsymbol{\theta}, \boldsymbol{\varphi}, \boldsymbol{\phi}} \mathcal{L}_s^{\mathrm{NNe}}(\boldsymbol{\theta}, \boldsymbol{\varphi}, \boldsymbol{\phi}) \tag{3.28}$$

where $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_J], \boldsymbol{\varphi} = [\boldsymbol{\varphi}_1, \ldots, \boldsymbol{\varphi}_J], \boldsymbol{\phi} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_J, \boldsymbol{\phi}_{\mathcal{J}}]$ and

$$\begin{aligned} \mathcal{L}_s^{\mathrm{NNe}} := {} & \mathbb{E}[\log Q_{\phi_{\mathcal{J}}}(Y|U_1, \ldots, U_J)] \\ & + s \sum_{j=1}^{J} \mathbb{E}[\log Q_{\phi_j}(y_i|u_{j,i})] - D_{KL}\left(P_{\theta_j}(U_j|X_j) \| Q_{\varphi_j}(U_j)\right). \end{aligned} \tag{3.29}$$

The loss function (3.5) can be obtained by taking the empirical estimate of (3.29).

Chapter 4

In-network Learning: Graph Network Topology

In this chapter we discuss the extension of the INL to network topologies that can be modelled with a directed, acyclic graph. This problem was proposed and studied in the patent document [11]. In this chapter, we derive a suitable loss function for such a model, by extending the techniques from [10] to this more general case. More specifically, we consider the case in which the J nodes that observe the data do not communicate directly with the central node. Instead, the nodes are connected to the central node through intermediary nodes, which do not have data of their own; however, they do have computational capabilities to process the data they observe. Moreover, we assume that some of the nodes that observe the data can be connected to other nodes that observe the data. Figure 4.1 shows an example of such a network.

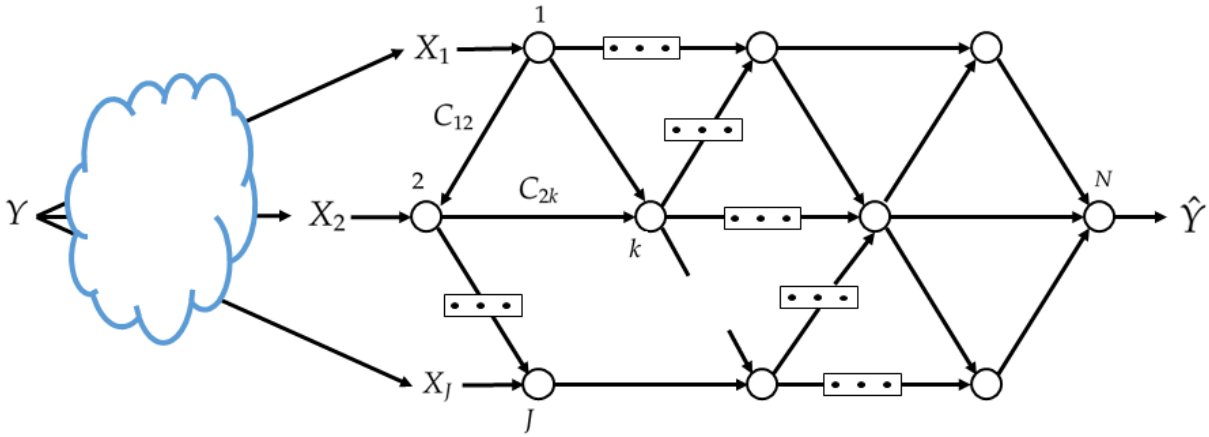


Figure 4.1: Studied network inference model.

We first derive an achievable trade-off between the performance of the system under

log-loss and the complexities of the encoded values that the nodes need to exchange, as measured by their minimum description length (MDL). Based on this theoretical analysis, one can derive a loss function that can train the NNs in a distributed manner, by first deriving a variational inner bound, and then parameterizing that lower/inner bound using neural networks. We showcase these steps for an example five-node network.

4.1 Problem Setup

We model an N-node network by a directed acyclic graph $\mathcal{G} = (\mathcal{N}, \mathcal{E}, \mathcal{C})$, where $\mathcal{N} = [1 : N]$ is the set of nodes, $\mathcal{E} \subset \mathcal{N} \times \mathcal{N}$ is the set of edges and $\mathcal{C} = \{C_{jk} : (j,k) \in \mathcal{E}\}$ is the set of edge weights. Each node represents a device and each edge represents a noiseless communication link with capacity C_{jk}. The processing at the nodes of the set $\mathcal{J}$ is such that each of them assigns an index $m_{jl} \in [1, M_{jl}]$ to each $x_j \in \mathcal{X}_j$ and each received index tuple $(m_{ij} : (i,j) \in \mathcal{E})$ for each edge $(j,l) \in \mathcal{E}$. Specifically, let for $j \in \mathcal{J}$ and l such that $(j,l) \in \mathcal{E}$, the set $\mathcal{M}_{jl} = [1 : M_{jl}]$. The encoding function at node j is

$$\phi_j : \mathcal{X}_j \times \left\{ \bigtimes_{i\,:\,(i,j)\in\mathcal{E}} \mathcal{M}_{ij} \right\} \longrightarrow \bigtimes_{l\,:\,(j,l)\in\mathcal{E}} \mathcal{M}_{jl}, \tag{4.1}$$

where $\bigtimes$ designates the Cartesian product of sets. Similarly, for $k \in [1 : N-1]/\mathcal{J}$, node k assigns an index $m_{kl} \in [1, M_{kl}]$ to each index tuple $(m_{ik} : (i,k) \in \mathcal{E})$ for each edge $(k,l) \in \mathcal{E}$. That is,

$$\phi_k : \bigtimes_{i\,:\,(i,k)\in\mathcal{E}} \mathcal{M}_{ik} \longrightarrow \bigtimes_{l\,:\,(k,l)\in\mathcal{E}} \mathcal{M}_{kl}. \tag{4.2}$$

The range of the encoding functions $\{\phi_i\}$ are restricted in size, as

$$\log|\mathcal{M}_{ij}| \le C_{ij} \quad \forall i \in [1, N-1] \quad \text{and} \quad \forall j : (i,j) \in \mathcal{E}. \tag{4.3}$$

Node N needs to infer on the random variable $Y \in \mathcal{Y}$ using all incoming messages, i.e.,

$$\psi : \bigtimes_{i\,:\,(i,N)\in\mathcal{E}} \mathcal{M}_{iN} \longrightarrow \hat{\mathcal{Y}}. \tag{4.4}$$

Again, we choose the reconstruction set $\hat{\mathcal{Y}}$ to be the set of distributions on $\mathcal{Y}$, i.e., $\hat{\mathcal{Y}} = \mathcal{P}(\mathcal{Y})$; and we measure the discrepancies between the true values of $Y \in \mathcal{Y}$ and their estimated fits in terms of average logarithmic loss, i.e., for $(y, \hat{P}) \in \mathcal{Y} \times \mathcal{P}(\mathcal{Y})$

$$d(y, \hat{P}) = \log \frac{1}{\hat{P}(y)}. \tag{4.5}$$

As such, the performance of a distributed inference scheme $\left((\phi_j)_{j\in\mathcal{J}}, (\phi_k)_{k\in[1,N-1]/\mathcal{J}}, \psi\right)$ for which (4.3) is fulfilled is given by its achievable *relevance*, given by

$$\Delta = H(Y) - \mathbb{E}\left[d(Y, \hat{Y})\right], \tag{4.6}$$

which, for a discrete set $\mathcal{Y}$, is directly related to the error of misclassifying the variable $Y \in \mathcal{Y}$.

4.2 In-network Learning Generalisation

In practice, in a supervised setting, the mappings given by (4.1), (4.2) and (4.4) need to be learnt from a set of training data samples $\{(x_{1,i}, \ldots, x_{J,i}, y_i)\}_{i=1}^{n}$. The data are distributed such that the samples $\mathbf{x}_j := (x_{j,1}, \ldots, x_{j,n})$ are available at node j for $j \in \mathcal{J}$ and the desired predictions $\mathbf{y} := (y_1, \ldots, y_n)$ are available at the end decision node N. We parameterize the possibly stochastic mappings (4.1), (4.2) and (4.4) using NNs. This is depicted in Figure 4.2. We denote the parameters of the NNs that parameterize the encoding function at each node $i \in [1 : (N-1)]$ with $\boldsymbol{\theta}_i$ and the parameters of the NN that parameterize the decoding function at node N with $\boldsymbol{\phi}$. Let $\boldsymbol{\theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{N-1}]$, we aim to find the parameters $\boldsymbol{\theta}, \boldsymbol{\phi}$ that maximize the relevance of the network, given the network constraints of (4.3). Given that the actual distribution is unknown and we only have access to a dataset, the loss function must strike a balance between its performance on the data set, given by an empirical estimate of the relevance, and the network's ability to perform well on samples outside the dataset.

The NNs at the various nodes are arbitrary and can be chosen independently – for instance, they need *not* be identical as in FL. It is only required that the following mild condition which, as will become clearer from what follows, facilitates the back-

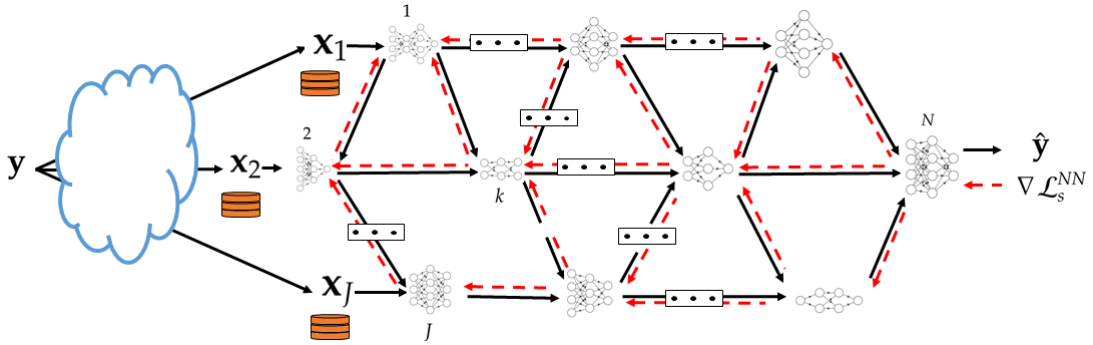
propagation be met. Specifically, for every $j \in \mathcal{J}$ and $x_j \in \mathcal{X}_j$[1] it holds that

$$\begin{aligned} &\text{Size of first layer of NN (j) =} \\ &\text{Dimension } (x_j) + \sum_{i\,:\,(i,j)\in\mathcal{E}} (\text{Size of last layer of NN } (i)). \end{aligned} \tag{4.7}$$

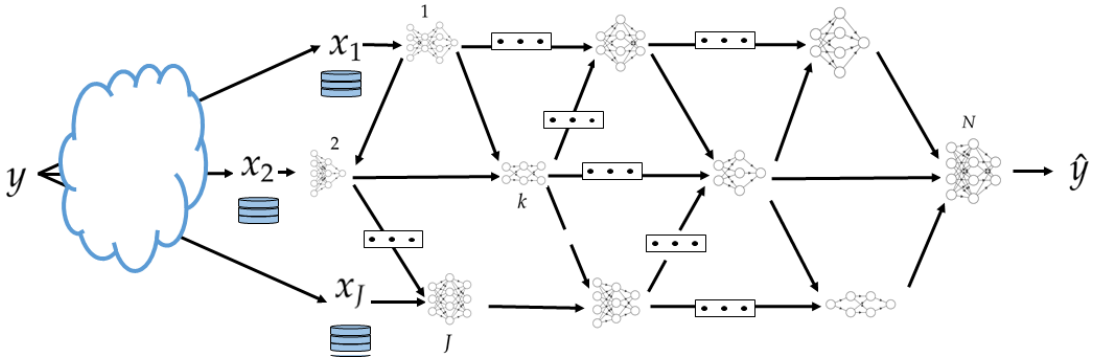
Similarly, for $k \in [1:N]/\mathcal{J}$ we have

$$\begin{aligned} &\text{Size of first layer of NN (k) =} \\ &\sum_{i\,:\,(i,k)\in\mathcal{E}} (\text{Size of last layer of NN } (i)). \end{aligned} \tag{4.8}$$

Remark 4.2.1. *Conditions* (4.7) *and* (4.8) *were imposed only for the sake of ease of implementation of the training algorithm; the techniques present in this paper, including optimal trade-offs between relevance and complexity for the given topology, the associated loss function, the variational lower bound, how to parameterize it using NNs and so on, do not require* (4.7) *and* (4.8) *to hold.*



(a) Training phase.



(b) Inference phase.

Figure 4.2: In-network learning and inference using neural networks.

Consider the general network inference model of Figure 4.1. Part of the difficulty of

[1]We assume all the elements of $\mathcal{X}_j$ have the same dimension.

this problem is in finding a suitable loss function that can be optimised distributively via neural networks that only have access to local data-sets each. The next theorem provides a bound on the relevance achievable (under some assumptions [2]) for an arbitrary network topology $(\mathcal{E}, \mathcal{N})$. For convenience, we define for $\mathcal{S} \subseteq [1, \ldots, N-1]$ and non-negative $(C_{ij} \ : \ (i,j) \in \mathcal{E})$ the quantity

$$C(\mathcal{S}) = \sum_{(i,j) \, : \, i \in \mathcal{S}, j \in \mathcal{S}^c} C_{ij}. \tag{4.9}$$

Theorem 4.2.1. *For the network inference model of Figure 4.1, in the regime of large data-sets the following relevance is achievable,*

$$\Delta = \max I(U_1, \ldots, U_J; Y) \tag{4.10}$$

where the maximisation is over joint measures of the form $P_Q P_{X_1,\ldots,X_J,Y} \prod_{j=1}^{J} P_{U_j|X_j,Q}$*, for which there exist non-negative* $R_1, \ldots, R_J$ *that satisfy*

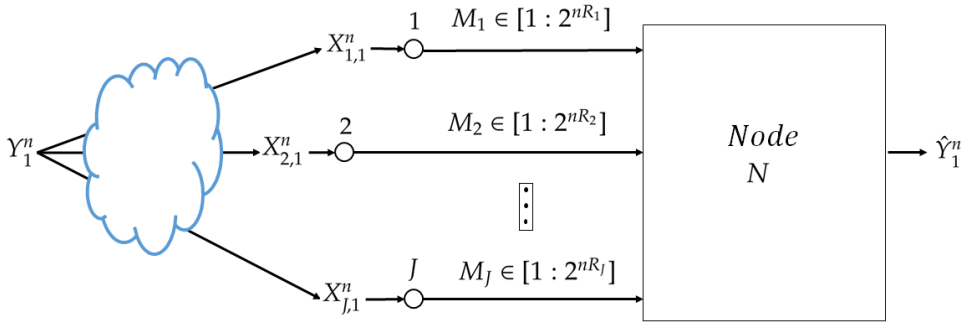
$$\sum_{j \in \mathcal{S}} R_j \geq I(U_{\mathcal{S}}; X_{\mathcal{S}} | U_{\mathcal{S}^c}, Q), \quad \textit{for all} \quad \mathcal{S} \subseteq \mathcal{J} \tag{4.11}$$

$$\sum_{j \in \mathcal{S} \cap \mathcal{J}} R_j \leq C(\mathcal{S}) \quad \textit{for all} \quad \mathcal{S} \subseteq [1 : N-1]$$

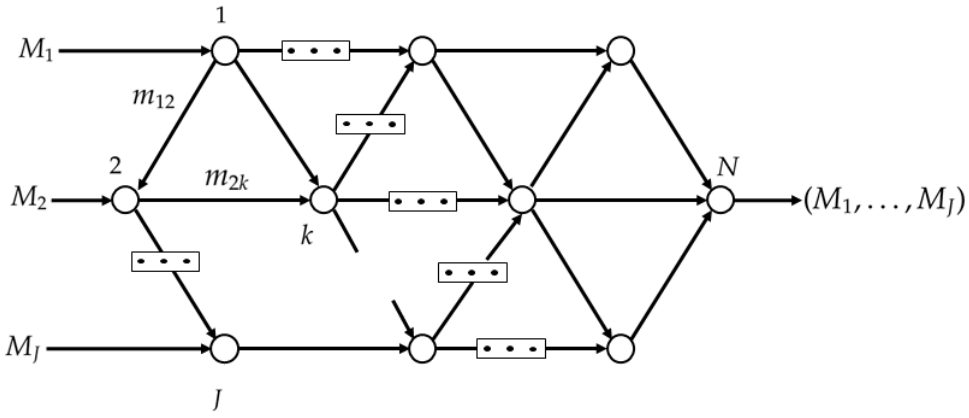
$$\textit{with} \quad \mathcal{S} \cap \mathcal{J} \neq \emptyset. \tag{4.12}$$

Proof. The proof of Theorem 4.2.1 appears in Section 4.4.1. An outline is as follows. The result is achieved using a separate compression-transmission-estimation scheme in which the observations $(\mathbf{x}_1, \ldots, \mathbf{x}_J)$ are first compressed distributively using Berger-Tung coding [56] into representations $(\mathbf{u}_1, \ldots, \mathbf{u}_J)$; and, then, the bin indices are transmitted as independent messages over the network $\mathcal{G}$ using linear-network coding [25, Section 15.5]. The decision node N first recovers the representation codewords $(\mathbf{u}_1, \ldots, \mathbf{u}_J)$; and, then, produces an estimate of the label $\mathbf{y}$. The scheme is illustrated in Figure 4.3. □

[2] The inference problem is a one-shot problem. The result of Theorem 4.2.1 is asymptotic in the size of the training data-sets. One-shot results for this problem can be obtained, e.g., along [55].



(a) Compression using Berger-Tung coding.



(b) Transmission of the bin indices using linear coding.

Figure 4.3: Block diagram of the separate compression-transmission-estimation scheme of Theorem 4.2.1.

4.3 Example: Five Node Network

Part of the utility of the loss function of Theorem 4.2.1 is in that it accounts explicitly for the topology of the network for inference fusion and propagation. Also, although as seen from its proof the setting of Theorem 4.2.1 assumes knowledge of the joint distribution of the tuple $(X_1, \ldots, X_J, Y)$, the result can be used to train, distributively, NNs from a set of available datesets. To do so, we first derive a Lagrangian function, from Theorem 4.2.1, which can be used as an objective function to find the desired set of encoders and decoder. Afterwards, we use a variational approximation to avoid the computation of marginal distributions, which can be costly in practice. Finally, we parameterize the distributions using NNs. For a given network topology in essence, the approach generalises that of the previous chapter to more general networks that involve hops. For simplicity, in what follows, this is illustrated for the example architecture of Figure 4.4. While the example is simple, it showcases the important aspect of any such topology, the fusion of the data at an intermediary nodes, i.e., a hop.

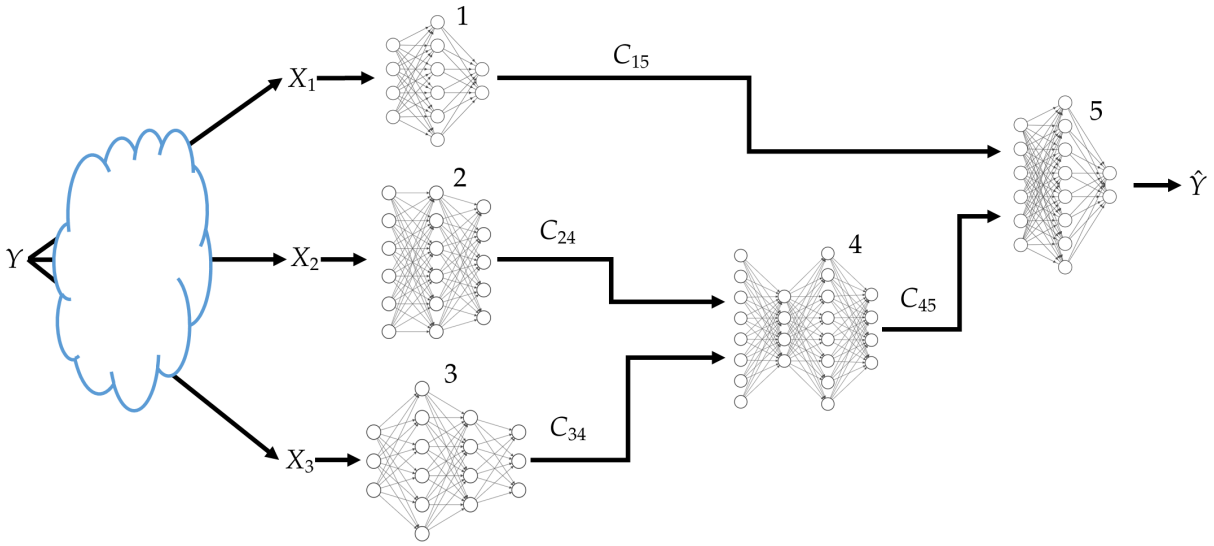


Figure 4.4: An example in-network learning with inference fusion and propagation

4.3.1 Loss function

Using Theorem 4.2.1 we will now derive a loss function for the example architecture of Figure 4.4, where $\mathcal{N} = \{1, 2, 3, 4, 5\}$ and $\mathcal{E} = \{(3, 4), (2, 4), (4, 5), (1, 5)\}$. Let $C_{\text{sum}} = C_{15} + C_{24} + C_{34} + C_{45}$; consider the region of all pairs $(\Delta, C_{\text{sum}}) \in \mathbb{R}_+^2$ for which the relevance level Δ, as given by Theorem 4.2.1, is achievable for some $C_{15} \geq 0$, $C_{24} \geq 0$, $C_{34} \geq 0$ and $C_{45} \geq 0$ such that $C_{\text{sum}} = C_{15} + C_{24} + C_{34} + C_{45}$. Hereafter, we denote such region as $\mathcal{RI}_{\text{sum}}$. Applying Fourier-Motzkin elimination on the region defined by Theorem 4.2.1 for the network of Figure 4.4, we get that the region $\mathcal{RI}_{\text{sum}}$ is given by the union of pairs $(\Delta, C_{\text{sum}}) \in \mathbb{R}_+^2$ for which [3]

$$\Delta \leq I(Y; U_1, U_2, U_3) \tag{4.13a}$$

$$C_{\text{sum}} \geq I(X_1, X_2, X_3; U_1, U_2, U_3) + I(X_2, X_3; U_2, U_3 | U_1) \tag{4.13b}$$

for some measure of the form

$$P_Y P_{X_1, X_2, X_3 | Y} P_{U_1 | X_1} P_{U_2 | X_2} P_{U_3 | X_3}. \tag{4.14}$$

The next proposition gives a useful parametrization of the region $\mathcal{RI}_{\text{sum}}$ as described by (4.13) and (4.14).

Proposition 4.3.1. *For every pair (Δ, C_{sum}) that lies on the boundary of the region described*

[3] The time sharing random variable is set to a constant for simplicity.

by (4.13) *and* (4.14) *there exists* $s \geq 0$ *such that* $(\Delta, C_{sum}) = (\Delta_s, C_s)$*, with*

$$\Delta_s = H(Y) + \max_{\mathbf{P}} \mathcal{L}_s(\mathbf{P}) + sC_s \tag{4.15a}$$

$$C_s = I(X_1, X_2, X_3; U_1^*, U_2^*, U_3^*) + I(X_2, X_3; U_2^*, U_3^* | U_1^*), \tag{4.15b}$$

and $\mathbf{P}^*$ *is the set of pmfs* $\mathbf{P} := \{P_{U_1|X_1}, P_{U_2|X_2}, P_{U_3|X_3}\}$ *that maximise the cost function*

$$\begin{aligned}\mathcal{L}_s(\mathbf{P}) := &-H(Y|U_1, U_2, U_3) - sI(X_1, X_2, X_3; U_1, U_2, U_3) \\ &- sI(X_2, X_3; U_2, U_3|U_1).\end{aligned} \tag{4.16}$$

Proof. See Section 4.4.2 □

In accordance with the studied example network inference problem of Figure 4.4, let a random variable U_4 be such that $U_4 - (U_2, U_3) - (X_1, X_2, X_3, Y, U_1)$. That is, the joint distribution factorises as

$$P_{X_1,X_2,X_3,Y,U_1,U_2,U_3,U_4} = P_{X_1,X_2,X_3,Y} P_{U_1|X_1} P_{U_2|X_2} P_{U_3|X_3} P_{U_4|U_2,U_3}. \tag{4.17}$$

Let for a given $s \geq 0$ and conditional $P_{U_4|U_2,U_3}$ the Lagrange term

$$\begin{aligned}\mathcal{L}_s^{\text{low}}(\mathbf{P}, P_{U_4|U_2,U_3}) = &- H(Y|U_1, U_4) - sI(X_1; U_1) \\ &- 2s\Big[I(X_2; U_2) + I(X_3; U_3)\Big] \\ &+ 2s\Big[I(U_2; U_1) + I(U_3; U_1, U_2)\Big].\end{aligned} \tag{4.18}$$

The following lemma shows that $\mathcal{L}_s^{\text{low}}(\mathbf{P}, P_{U_4|U_2,U_3})$ lower bounds $\mathcal{L}_s(\mathbf{P})$ as given by (4.16).

Lemma 4.3.1. *For every* $s \geq 0$ *and joint measure that factorises as* (4.17)*, we have*

$$\mathcal{L}_s(\mathbf{P}) \geq \mathcal{L}_s^{low}(\mathbf{P}, P_{U_4|U_2,U_3}), \tag{4.19}$$

Proof. See Section 4.4.3. □

For convenience let $\mathbf{P}_+ := \{P_{U_1|X_1}, P_{U_2|X_2}, P_{U_3|X_3}, P_{U_4|U_2,U_3}\}$. The optimisation of (4.18) generally requires the computation of marginal distributions, which can be costly in practice. Hereafter we derive a variational lower bound on $\mathcal{L}_s^{\text{low}}$ with respect to some arbitrary (variational) distributions. Specifically, let

$$\mathbf{Q} := \{Q_{Y|U_1,U_4}, Q_{U_3}, Q_{U_2}, Q_{U_1}\}, \tag{4.20}$$

where $Q_{Y|U_1,U_4}$ represents variational (possibly stochastic) decoders and Q_{U_3}, Q_{U_2} and Q_{U_1} represent priors. Also, let

$$\begin{aligned}\mathcal{L}_s^{\text{v-low}}(\mathbf{P}_+, \mathbf{Q}) := \mathbb{E}[\log Q_{Y|U_1,U_4}(Y|U_1, U_4)] - sD_{\text{KL}}(P_{U_1|X_1}\|Q_{U_1}) \\ - 2sD_{\text{KL}}(P_{U_2|X_2}\|Q_{U_2}) - 2sD_{\text{KL}}(P_{U_3|X_3}\|Q_{U_3}).\end{aligned} \tag{4.21}$$

The following lemma, the proof of which is essentially similar to that of [10, Lemma 1], provides an alternative maximisation problem to that of $\max_{\mathbf{P}_+} \mathcal{L}_s^{\text{low}}(\mathbf{P}_+)$, for every $s \geq 0$.

Lemma 4.3.2. *For $\mathcal{L}_s^{v\text{-}low}$ defined as in* (4.21), *$\mathcal{L}_s^{low}$ as* (4.18) *and $s \geq 0$*

$$\max_{\mathbf{P}_+} \mathcal{L}_s^{low}(\mathbf{P}_+) = \max_{\mathbf{P}_+} \max_{\mathbf{Q}} \mathcal{L}_s^{v\text{-}low}(\mathbf{P}_+, \mathbf{Q}). \tag{4.22}$$

Proof. See Section 4.4.4. □

Since the distribution of the data is not known, but only a set of samples is available $\{(x_{1,i}, \ldots, x_{J,i}, y_i)\}_{i=1}^n$, we restrict the optimisation of (4.21) to the family of distributions that can be parametrised by NNs. Thus, we obtain the following loss function which can be optimised empirically, in a distributed manner, using gradient-based techniques,

$$\begin{aligned}\mathcal{L}_s^{\text{NN}}(n) := \frac{1}{n}\sum_{i=1}^n \left[\log Q_{\phi_5}(y_i|u_{1,i}, u_{4,i}) - s\log\left(\frac{P_{\theta_1}(u_{1,i}|x_{1,i})}{Q_{\varphi_1}(u_{1,i})}\right)\right] \\ - \frac{2s}{n}\sum_{i=1}^n \left[\log\left(\frac{P_{\theta_2}(u_{2,i}|x_{2,i})}{Q_{\varphi_2}(u_{2,i})}\right) + \log\left(\frac{P_{\theta_3}(u_{3,i}|x_{3,i})}{Q_{\varphi_3}(u_{3,i})}\right)\right],\end{aligned} \tag{4.23}$$

where s stands for a Lagrange multiplier and the distributions $Q_{\phi_5}, P_{\theta_4}, P_{\theta_3}, P_{\theta_2}, P_{\theta_1}$ are variational ones whose parameters are determined by the chosen NNs using the

re-parametrisation trick of [45]; and, $\{Q_{\varphi_i} : i \in \{1,2,3\}\}$ are priors known to the encoders. For example, denoting by f_{θ_j} the NN used at node $j \in \mathcal{J}$ whose (weight and bias) parameters are given by $\boldsymbol{\theta}_j$. For regression problems the conditional distribution $P_{\boldsymbol{\theta}_j}(u_{i,j}|x_{i,j})$ can be chosen to be multivariate Gaussian, i.e., $P_{\boldsymbol{\theta}_j}(u_{i,j}|x_{i,j}) = \mathcal{N}(u_{i,j}; \boldsymbol{\mu}_i^{\theta_j}, \boldsymbol{\Sigma}_i^{\theta_j})$. For discrete data, concrete variables (i.e., Gumbel-Softmax) can be used instead.

4.3.2 Training Algorithm

We now describe the training algorithm for INL for the five-node example, this section particularises on the training procedure first reported in the patent [11] by Zaidi, Scaman and Escamilla to the five-node network presented. During the forward pass, every node $j \in \{1,2,3\}$ processes mini-batches of size b_j of its training data set $\mathbf{x}_j$. Nodes 2 and 3 send their vectors formed of the activation values of the last layer of their NNs to node 4, where due to (4.7) the vectors are concatenated vertically at the input layer of NN 4. The forward pass continues on the NN at node 4 until its last layer. Next, nodes 1 and 4 send the activation values of their last layers to node 5. Again, as the sizes of the last layers of the NNs of nodes 1 and 4 satisfy (4.8) the sent activation vectors are concatenated vertically at the input layer of NN 5; and the forward pass continues until the last layer of NN 5. All nodes update their parameters using standard back-propagation technique which is as follows. For node $t \in \mathcal{N}$ let L_t denote the index of the last layer of the NN at node t. Also, let, for $\mathbf{w}_t^{[l]}$, $\mathbf{b}_t^{[l]}$ and $\mathbf{a}_t^{[l]}$ denote, respectively, the weights, biases and activation values at layer $l \in [2 : L_t]$ for the NN at node t; σ is the activation function and the input to the NN is denoted $\mathbf{a}_t^{[1]}$. Node t computes the error vectors

$$\boldsymbol{\delta}_t^{[L_t]} = \nabla_{\mathbf{a}_t^{[L_t]}} \mathcal{L}_s^{NN}(b) \odot \sigma'(\mathbf{w}_t^{[L_t]} \mathbf{a}_t^{[L_t-1]} + \mathbf{b}_t^{[L_t]}) \tag{4.24a}$$

$$\boldsymbol{\delta}_t^{[l]} = [(\mathbf{w}_t^{[l+1]})^T \boldsymbol{\delta}_t^{[l+1]}] \odot \sigma'(\mathbf{w}_t^{[l]} \mathbf{a}_t^{[l-1]} + \mathbf{b}_t^{[l]}) \ \forall \ l \in [2, L_t - 1], \tag{4.24b}$$

$$\boldsymbol{\delta}_t^{[1]} = [(\mathbf{w}_t^{[2]})^T \boldsymbol{\delta}_t^{[2]}] \tag{4.24c}$$

and then updates its weight- and bias parameters as

$$\mathbf{w}_t^{[l]} \rightarrow \mathbf{w}_t^{[l]} - \eta \boldsymbol{\delta}_t^{[l]} (\mathbf{a}_t^{[l-1]})^T, \tag{4.25a}$$

$$\mathbf{b}_t^{[l]} \rightarrow \mathbf{b}_t^{[l]} - \eta \boldsymbol{\delta}_t^{[l]}, \tag{4.25b}$$

where η designates the learning parameter [4].

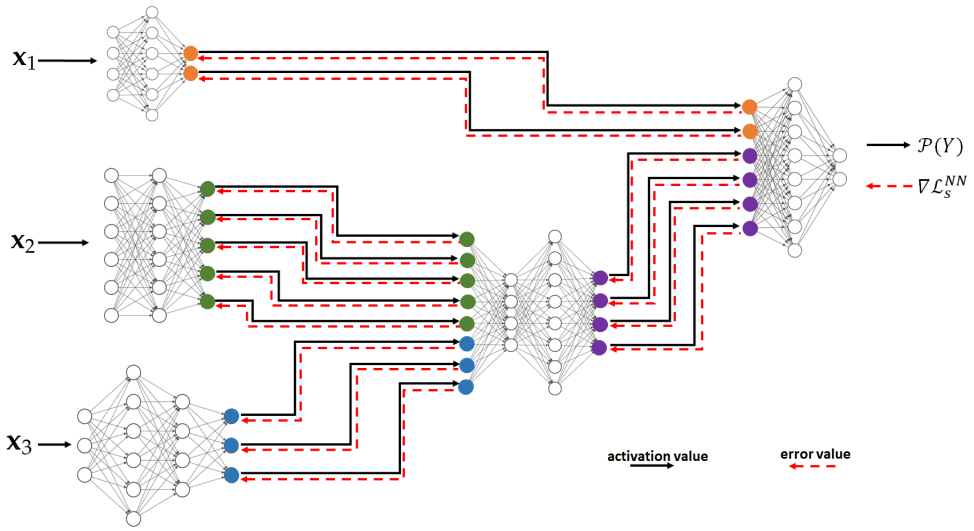


Figure 4.5: Forward and backward passes for the inference problem of Figure 4.4.

During the backward pass, each of the NNs updates its parameters according to (4.24) and (4.25). Node 5 is the first to apply the back-propagation procedure in order to update the parameters of its NN. It applies (4.24) and (4.25) sequentially, starting from its last layer.

The error propagates back until it reaches the first layer of the NN of node 5. Node 5 then splits horizontally the error vector of its input layer (4.24c) into 2 sub-vectors with the top sub-error vector having as size that of the last layer of the NN of node 1 and the bottom sub-error vector having as size that of the last layer of the NN of node 4 – see Figure 4.5. Similarly, the two nodes 1 and 4 continue the backward propagation at their turns simultaneously. Node 4 then splits horizontally the error vector of its input layer (4.24c) into 2 sub-vectors with the top sub-error vector having as size that of the last layer of the NN of node 2 and the bottom sub-error vector having as size that of the last layer of the NN of node 3. Finally, the backward propagation continues on the NNs of nodes 2 and 3. The whole process is repeated until convergence.

[4] For simplicity η and σ are assumed here to be identical for all NNs.

4.3.3 Inference Algorithm

We now describe the inference algorithm for INL for the five-node example, this section similarly particularises on the training procedure first reported in the patent [11] by Zaidi, Scaman and Escamilla to the five-node network presented. During inference, nodes 1, 2 and 3 observe (or measure) each a new sample. Let $\mathbf{x}_1$ be the sample observed by node 1; and $\mathbf{x}_2$ and $\mathbf{x}_3$ those observed by node 2 and node 3, respectively. Node 1 processes $\mathbf{x}_1$ using its NN and sends an encoded value $\mathbf{u}_1$ to node 5; and so do nodes 2 and 3 towards node 4. Upon receiving $\mathbf{u}_2$ and $\mathbf{u}_3$ from nodes 2 and 3, node 4 concatenates them vertically and processes the obtained vector using its NN. The output $\mathbf{u}_4$ is then sent to node 5. The latter performs similar operations on the activation values $\mathbf{u}_1$ and $\mathbf{u}_4$; and outputs an estimate of the label $\mathbf{y}$ in the form of a soft output $Q_{\phi_5}(\mathbf{y}|\mathbf{u}_1, \mathbf{u}_4)$.

4.4 Proofs

4.4.1 Proof of Theorem 4.2.1

The proof of Theorem 1 is based on a scheme in which the observations $\{\mathbf{x}_j\}_{j\in\mathcal{J}}$ are compressed distributively using Berger-Tung coding [56]; and, then, the compression bin indices are transmitted as independent messages over the network $\mathcal{G}$ using linear-network coding [25, Section 15.4]. The decision node N first decompresses the compression codewords and then uses them to produce an estimate $\hat{Y}$ of Y. In what follows, for simplicity, we set the time-sharing random variable to be a constant, i.e., $Q = \emptyset$. Let $0 < \epsilon^{''} < \epsilon^{'} < \epsilon$.

4.4.1.1 Codebook Generation

Fix a joint distribution $P_{X_1,\ldots,X_J,Y,U_1,\ldots,U_J}$ that is factorised as $P_Q P_{X_1,\ldots,X_J,Y} \prod_{j=1}^{J} P_{U_j|X_j,Q}$. Also, let $D = H(Y|U_1,\ldots,U_J)$; and, for $(u_1,\ldots,u_J) \in \mathcal{U}_1 \times \ldots \times \mathcal{U}_J$, the reconstruction function $\hat{y}(\cdot|u_1,\ldots,u_J) \in \mathcal{P}(\mathcal{Y})$ such that $\mathbb{E}\left[d(Y,\hat{Y})\right] \leq \frac{D}{1+\epsilon}$, where $d : \mathcal{Y} \times \mathcal{P}(\mathcal{Y}) \longrightarrow \mathbb{R}_+$ is the distortion measure given by (4.5). For every $j \in \mathcal{J}$, let $\tilde{R}_j \geq R_j$. Also, randomly and independently generate $2^{n\tilde{R}_j}$ sequences $u_j^n(l_j)$, $l_j \in [1 : 2^{n\tilde{R}_j}]$, each ac-

cording to $\prod_{i=1}^{n} p_{U_j}(u_{ji})$. Partition the set of indices $l_j \in 2^{n\tilde{R}_j}$ into equal size bins $B_j(m_j) = \left[(m_j - 1)2^{n\tilde{R}_j - R_j} : m_j 2^{n\tilde{R}_j - R_j}\right]$, $m_j \in [1 : 2^{nR_j}]$. The codebook is revealed to all source nodes $j \in \mathcal{J}$ as well as to the decision node N, but not to the intermediary nodes.

4.4.1.2 Compression of the observations

Node $j \in \mathcal{J}$ observes x_j^n and finds an index $l_j \in [1 : 2^{n\tilde{R}_j}]$ such that $(x_j^n, u_j^n(l_j)) \in \mathcal{T}_{\epsilon''}^{(n)}$. If there is more than one index the node selects one at random. If there is no such index, it selects one at random from $[1 : 2^{n\tilde{R}_j}]$. Let m_j be the index of the bin that contains the selected l_j, i.e., $l_j \in \mathcal{B}_j(m_j)$.

4.4.1.3 Transmission of the compression indices over the graph network

In order to transmit the bins indices $(M_1, \ldots, M_J) \in [1 : 2^{nR_1}] \times \ldots \times [1 : 2^{nR_J}]$ to the decision node N over the graph network $\mathcal{G} = (\mathcal{E}, \mathcal{N}, C)$, they are encoded as if they were independent-messages using the linear network coding scheme of [25, Theorem 15.5]; and then transmitted over the network. The transmission of the multimessage $(M_1, \ldots, M_J) \in [1 : 2^{nR_1}] \times \ldots \times [1 : 2^{nR_J}]$ to the decision node N is without error as long as for all $\mathcal{S} \subseteq [1 : N - 1]$ we have

$$\sum_{j \in \mathcal{S} \cap \mathcal{J}} R_j \leq C(\mathcal{S}) \tag{4.26}$$

where $C(\mathcal{S})$ is defined by (4.9).

4.4.1.4 Decompression and estimation

The decision node N first looks for the unique tuple $(\hat{l}_1, \ldots, \hat{l}_J) \in \mathcal{B}_1(m_1) \times \ldots \times \mathcal{B}_J(m_J)$ such that $(u_1^n(\hat{l}_1), \ldots, u_J^n(\hat{l}_J)) \in \mathcal{T}_{\epsilon}^{(n)}$. With high probability, Node N finds such a unique tuple as long as n is large and for all $\mathcal{S} \subseteq \mathcal{J}$ it holds that [56] (see also [25, Theorem 12.1])

$$\sum_{j \in \mathcal{S}} R_j \geq I(U_{\mathcal{S}}; X_{\mathcal{S}} | U_{\mathcal{S}^c}). \tag{4.27}$$

The decision node N then produces an estimate $\hat{y}^n$ of y^n as $\hat{y}(u_1^n(\hat{l}_1), \ldots, u_J^n(\hat{l}_J))$.

It can be shown easily that the per-sample relevance level achieved using the described scheme is $\Delta = I(U_1, \ldots, U_J; Y)$; and this completes the proof of Theorem 4.2.1.

4.4.2 Proof of Proposition 4.3.1

For $C_{sum} \geq 0$ fix $s \geq 0$ such that $C_s = C_{sum}$; and let $\mathbf{P}^* = \{P_{U_1^*|X_1}, P_{U_2^*|X_2}, P_{U_3^*|X_3}\}$ be the solution to (4.16) for the given s. By making the substitution in (4.15):

$$\begin{aligned} \Delta_s =& I(Y; U_1^*, U_2^*, U_3^*) && (4.28)\\ \leq& \Delta && (4.29) \end{aligned}$$

where (4.29) holds since Δ is the maximum $I(Y; U_1, U_2, U_3)$ over all distribution for which (4.13b) holds, which includes $\mathbf{P}^*$.
Conversely let $\mathbf{P}^*$ be such that (Δ, C_{sum}) is on the bound of the $\mathcal{RI}_{\text{sum}}$ then:

$$\begin{aligned} \Delta =& H(Y) - H(Y|U_1^*, U_2^*, U_3^*)\\ \leq& H(Y) - H(Y|U_1^*, U_2^*, U_3^*) + sC_{\text{sum}}\\ & - s\Big[I(X_2, X_3; U_2^*, U_3^*|U_1^*) + I(X_1, X_2, X_3; U_1^*, U_2^*, U_3^*)\Big] && (4.30)\\ \leq& H(Y) + \max_{\mathbf{P}} \mathcal{L}_s(\mathbf{P}) + sC_{\text{sum}} && (4.31)\\ =& \Delta_s - sC_s + sC_{\text{sum}}\\ =& \Delta_s + s(C_{\text{sum}} - C_s). && (4.32) \end{aligned}$$

Where (4.30) follows from (4.13b). Inequality (4.31) holds due to the fact that $\max_{\mathbf{P}} \mathcal{L}(\mathbf{P})$ takes place over all $\mathbf{P}$, including $\mathbf{P}^*$. Since (4.32) is true for any $s \geq 0$ we take s such that $C_{\text{sum}} = C_s$, which implies $\Delta \leq \Delta_s$. Together with (4.29) this completes the proof.

4.4.3 Proof of Lemma 4.3.1

We have

$$\begin{aligned} \mathcal{L}_s(\mathbf{P}) =& - H(Y|U_1, U_2, U_3) - sI(X_1, X_2, X_3; U_1, U_2, U_3)\\ & - sI(X_2, X_3; U_2, U_3|U_1) && (4.33) \end{aligned}$$

$$
\begin{aligned}
&= -H(Y|U_1,U_2,U_3) \\
&\quad - s\Big[I(X_1;U_1) + 2I(X_2,X_3;U_2,U_3|U_1)\Big] && (4.34)\\
&= -H(Y|U_1,U_2,U_3) - sI(X_1;U_1) - 2sI(X_2;U_2) \\
&\quad - 2s\Big[I(X_3;U_3) - I(U_3;U_1,U_2) - I(U_2;U_1)\Big] && (4.35)\\
&= -H(Y|U_1,U_2,U_3) - sI(X_1;U_1) - 2sI(X_2;U_2) \\
&\quad + 2s\Big[I(U_2;U_1) + I(U_3;U_1,U_2) - I(X_3;U_3)\Big] && (4.36)\\
&\geq -H(Y|U_1,U_4) - sI(X_1;U_1) - 2s\Big[I(X_2;U_2) + I(X_3;U_3)\Big] \\
&\quad + 2s\Big[I(U_2;U_1) + I(U_3;U_1,U_2)\Big] && (4.37)
\end{aligned}
$$

where (4.34) holds since $U_1 - X_1 - (X_2, X_3, U_2, U_3)$ and $(U_2, U_3) - (X_2, X_3) - (U_1, X_1)$; (4.35) holds since $U_2 - X_2 - (U_1, X_3)$ and $U_3 - X_3 - (U_1, U_2, X_2)$; (4.37) hold since $U_4 - (U_2, U_3) - (Y, U_1)$.

4.4.4 Proof of Lemma 4.3.2

From [10, eq. (55)] it can be shown that for any pmf $Q_{Y|Z}(y|z)$, $y \in \mathcal{Y}$ and $z \in \mathcal{Z}$ the conditional entropy $H(Y|Z)$ is :

$$
H(Y|Z) = \mathbb{E}[-\log Q_{Y|Z}(Y|Z)] - D_{\mathrm{KL}}(P_{Y|Z}||Q_{Y|Z}). \tag{4.38}
$$

And from [10, eq. (81)]:

$$
\begin{aligned}
I(X;Z) &= H(Z) - H(Z|X) \\
&= D_{\mathrm{KL}}(P_{Z|X}||Q_Z) - D_{\mathrm{KL}}(P_Z||Q_Z). && (4.39)
\end{aligned}
$$

Now, substituting equations (4.38) and (4.39) in (4.21) the following result is obtained:

$$
\begin{aligned}
\mathcal{L}_s^{\mathrm{low}}(\mathbf{P}_+) = &-H(Y|U_1,U_4) + 2s\Big[I(U_2;U_1) + I(U_3;U_1,U_2) - I(X_2;U_2) - I(X_3;U_3)\Big] \\
=&\mathbb{E}[\log Q_{Y|U_1,U_4}] + D_{\mathrm{KL}}(P_{Y|U_1,U_4}||Q_{Y|U_1,U_4}) - sD_{\mathrm{KL}}(P_{U_1|X_1}||Q_{U_1}) + sD_{\mathrm{KL}}(P_{U_1}||Q_{U_1}) \\
&- 2s\Big(D_{\mathrm{KL}}(P_{U_2|X_2}||Q_{U_2}) + D_{\mathrm{KL}}(P_{U_3|X_3}||Q_{U_3}) + D_{\mathrm{KL}}(P_{U_2}||Q_{U_2}) + D_{\mathrm{KL}}(P_{U_3}||Q_{U_3})\Big) \\
&+ 2s\Big(D_{\mathrm{KL}}(P_{U_2}||Q_{U_2}) + D_{\mathrm{KL}}(P_{U_3|U_1,U_2}||Q_{U_3}) + D_{\mathrm{KL}}(P_{U_3}||Q_{U_3}) + D_{\mathrm{KL}}(P_{U_2|U_1}||Q_{U_2})\Big)
\end{aligned}
$$

$$
\begin{aligned}
&= \mathbb{E}[\log Q_{Y|U_1,U_4}] - sD_{\mathrm{KL}}(P_{U_1|X_1}\|Q_{U_1}) - 2sD_{\mathrm{KL}}(P_{U_2|X_2}\|Q_{U_2}) - 2sD_{\mathrm{KL}}(P_{U_3|X_3}\|Q_{U_3}) \\
&\quad + sD_{\mathrm{KL}}(P_{U_1}\|Q_{U_1}) + 2sD_{\mathrm{KL}}(P_{U_2|U_1}\|Q_{U_2}) + 2sD_{\mathrm{KL}}(P_{U_3|U_1,U_2}\|Q_{U_3}) \\
&\quad + D_{\mathrm{KL}}(P_{Y|U_1,U_4}\|Q_{Y|U_1,U_4}) \\
&= \mathcal{L}_s^{\text{v-low}} + sD_{\mathrm{KL}}(P_{U_1}\|Q_{U_1}) + 2sD_{\mathrm{KL}}(P_{U_2|U_1}\|Q_{U_2}) \\
&\quad + 2sD_{\mathrm{KL}}(P_{U_3|U_1,U_2}\|Q_{U_3}) + D_{\mathrm{KL}}(P_{Y|U_1,U_4}\|Q_{Y|U_1,U_4}) \\
&\geq \mathcal{L}_s^{\text{v-low}} \qquad (4.40)
\end{aligned}
$$

The last inequality (4.40) holds due to the fact that KL divergence is always positive and $s \geq 0$, thus proving the lemma.

Chapter 5

Scheduling and Resource Allocation

In this chapter we consider the problem of resource allocation for the case in which the INL algorithm is deployed in a wireless sensor network (WSN). A solution of such a problem was first presented in the patent document [17] by Zaidi, and then described in more detail in Zaidi-Krasnowski [18]. Before introducing the results of [17, 18] we quickly describe the problem and highlight the challenges in solving it.

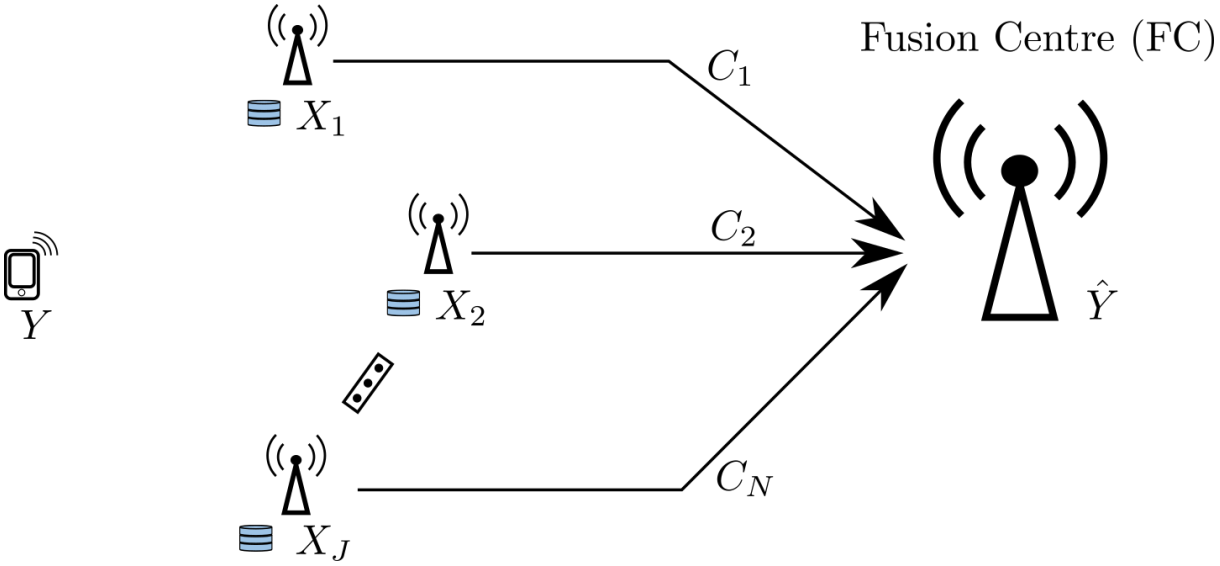


Figure 5.1: Wireless sensor networks problem setting.

We again consider the problem shown in Figure 5.1. The fusion centre (FC) must infer an estimate of a random variable of interest, Y, from the observations $X_1, \ldots, X_J$ acquired/measured each at a distinct device/node. In addition, it is assumed that the transmission between the devices and the FC is resource-constrained. That is, the devices are not allowed to (e.g., due to communication or privacy constraints) share their raw data, X_j, with the FC. As a consequence, the FC needs to output an estimate $\hat{Y}$

of Y based only on compressed versions of the devices' observations. We are interested in the setting in which all devices, including the FC, are equipped with neural networks (NN) and use them to perform some desired operations. We consider the case in which the available communication resources can be arbitrarily allocated between nodes with the aim of improving the quality of the estimate $\hat{Y}$ of Y, on average. To accurately assign the resources, one needs to be able to correctly assess the benefits of allocating resources to each node, which cannot be computed directly due to the distributed nature of the network. It is interesting to observe that while for the inference task at hand, the FC has a clear advantage over each of the devices taken individually, as it observes the compressed representations, which we denote by U_j for $j \in \mathcal{J} = \{1, \ldots, J\}$, from *all* the nodes. The latter have, in their turn, an advantage over the FC. Indeed, rather than a compressed version of the input data, each device has access to the data that has been used to generate that compression. According to the Markov chain $Y - X_j - U_j$, it is clear that U_j holds less information about Y than X_j itself, for any $j \in \mathcal{J}$. Consequently, relevant network resource allocation procedures for such inference tasks look to suitably strike the right balance between the aforementioned advantage. In [17], the authors patented an architecture in which (i) each device measuring its own local assessment of how relevant is the data it holds for the inference task at hand and transmitting it to the decision node, (ii) that device also simultaneously sends the output activation values to the decision node, (iii) the decision node combining all received information (activation values from the various devices as well as their individual local assessments) and then forming its own (global) assessment of how relevant each device's observation is, (iv) application of the idea to device scheduling and power allocation (two algorithms for scheduling were already given in [17]). This architecture was described in detail in [18]. In this chapter, we build upon these works by discussing possible local assessments as well as introducing a novel global assessment measure. Additionally, we provide a new scheduling algorithm that does not require the channel knowledge, as well as, a new power allocation algorithm. A similar work on the development of resource allocation algorithms based on the patent of [17] had been started in [18]; however, in [18] the authors considered the problem of digital transmission and as such proposed algorithms for time slots allocation, as opposed to scheduling and power allocation.

Another key difference between our work and that of [18] is that the global assessment is calculated differently.

5.1 Related Work

To the best of the authors' knowledge, the current resource allocation algorithms are not suitable for use for the inference problem studied in this chapter. This setting introduces the additional challenge of assessing the relative importance of each node in a distributed manner since each node is equipped with a NN, and the relationship between the observed data and the target variable must be learnt from a set of training data.

The works of [57–60] consider the problem of resource allocation or scheduling in a distributed network when there is no known model of the relation between the data and the target variable, and the authors employ NNs as part of their solutions. However, these works consider device scheduling and resource allocation during distributed training, not distributed inference as in our case. Additionally, the model obtained from the training procedure is designed for use in a centralised manner. In [57], the authors looked at the problem of scheduling multiple edge devices that have access to local data that they need to send to the central node, over a noisy channel, such that it can train a central model. In [58–60], the authors investigate the problem of power allocation and scheduling during the training procedure of the federated learning (FL) algorithm, introduced in [12] and presented in Section 3.3.1.

Our work is also related to the problem of distributed detection and distributed estimation. The problem of resource allocation for distributed detection has been extensively investigated and, as such, we refer the reader to [61] and references within for a more in depth survey of the field. Two papers that we consider most related to our work in this field are those of [62, 63]. In [63], the authors derived the optimal power allocation, with respect to a so-called J-divergence performance index, in a WSN for the case where the observations are independent, but do not need to be sampled from identical distributions. Furthermore, the authors have shown that the optimal power allocation depends on both the quality of the observed data and on the quality

of the communication channel. In [62], the authors used a particle swarm optimisation algorithm to find the power allocation for the distributed detection problem for the case of correlated observations. The authors observed that the performance of the system can be significantly improved by taking into account the correlation of the observed variables compared to assuming that they are independent. In both [62] and [63], it is noted that, for optimal performance, nodes with poor observations and/or channel quality need to be turned off, i.e., not transmit data.

For the problem of distributed estimation, in [64–66], the authors investigated decentralised estimation schemes, with perfect channels, in which local nodes send to the FC messages of length determined by the signal-to-noise ratio (SNR) of the observed signal. In [67], the authors extended the results of [64] to the case of an imperfect communication channel and observed that, similarly to distributed detection, nodes with bad data quality or poor channels should be turned off.

As already described in the introduction, this chapter builds upon [17], where the authors patented an architecture in which (i) each device measuring its own local assessment of how relevant is the data it holds for the inference task at hand and transmitting it to the decision node, (ii) that device also simultaneously sends the output activation values to the decision node, (iii) the decision node combining all received information (activation values from the various devices as well as their individual local assessments) and then forming its own (global) assessment of how relevant each device's observation is, (iv) application of the idea to device scheduling and power allocation (two algorithms for scheduling were already given in [17]). This architecture was described in detail in [18]; however in [18] the authors consider the problem of digital transmission and as such they propose algorithms for time slots allocation, as opposed to scheduling and power allocation which is considered in this chapter. In this chapter, we build upon these works by discussing possible local assessments as well as introducing a novel global assessment measure. Additionally, we provide a new scheduling algorithm that does not require the channel knowledge, as well as, a new power allocation algorithm.

5.2 Resource Allocation and Scheduling Problem

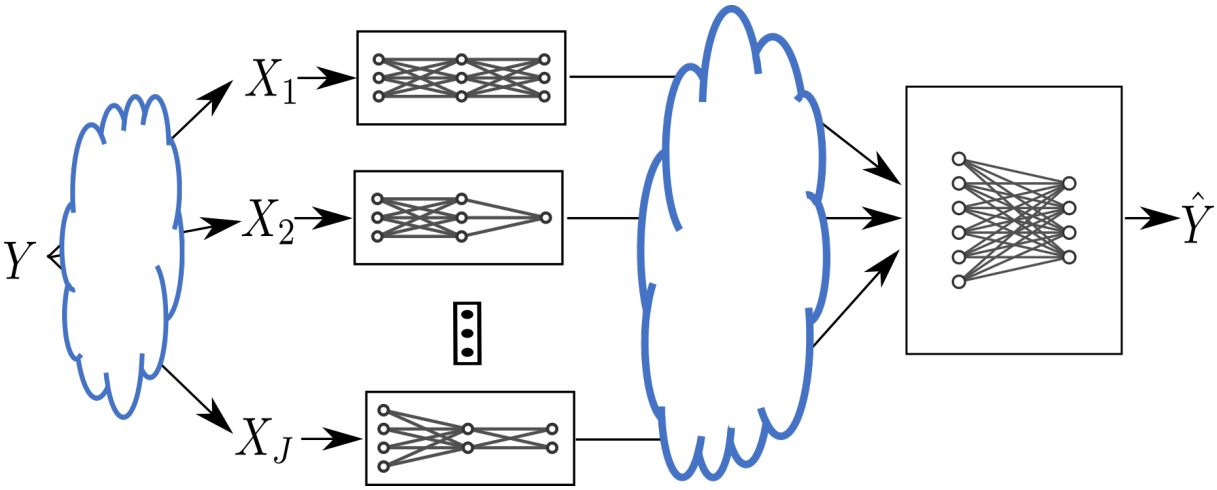


Figure 5.2: In-network learning problem with varying communication channels and data distributions.

Consider the network inference problem of Figure 5.2. The problem setting is similar to that considered in Section 3, with the addition that the quality of the channels between Y and the local (edge) devices, and between the edge devices and the FC vary in a block-stationary manner. This can happen in WSNs, for example, due to faulty sensors or changes in the environment in which the WSN is deployed. We consider a block to be the period in which both the communication channels and the channels between Y and the local (edge) devices are stationary. For a given block, the goal is to schedule only a subset K of devices (those whose data and channels are comparably the best, in a sense that will become clearer in the rest of this chapter) and allocate power to the scheduled nodes such that the performance of the system is improved on average. The reason for selecting only a subset of devices comes from the fact that in doing so, we can allocate more power to the more relevant nodes. In our setting, the assessment of the usefulness of the data observed by a node depends not only on the inference task at hand but also on what other nodes observe.

Due to the fact that the relationship between the data observed by each node is not known by the FC in advance, we introduce a scheduling phase during which the importance of each node can be assessed. As a consequence, the transmission during any given block is divided into two periods. The first period, during which all devices transmit and at the end of which the FC selects the K desired devices by allocating resources to them. In what follows, this is called the scheduling phase. The second

period, during which only the selected K devices transmit. The FC then performs inference based only on the data it receives from the K devices. In what follows, this is called the inference phase. The two distinct periods are shown in Figure 5.3.

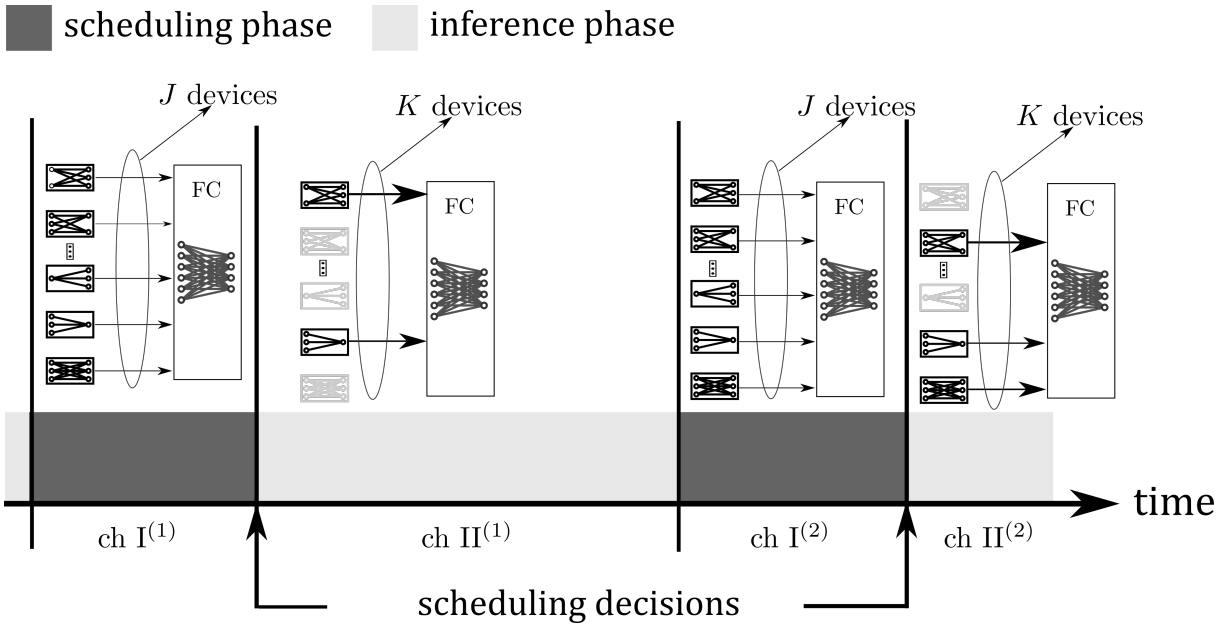


Figure 5.3: Diagram describing the separation between the scheduling and inference phases.

In the rest of this chapter, we concentrate on the design of our solution for a given block. During both the scheduling phase and the inference phase, we assume digital transmission, and that the devices transmit over orthogonal channels, i.e., they do not interfere with each other. The channel gain from a sensing node $j \in \mathcal{J}$ to the FC is denoted by $h_j \in \mathbb{R}_+$. The signals received by the FC are distorted by additive independent Gaussian noise, which we assume to have zero mean and unit variance. The channel capacity between node j and the FC is given by

$$C_j := 0.5 \log_2(1 + h_j^2 P_j), \tag{5.1}$$

where P_j is the signal power at node j. For simplicity of analysis, we assume that the sensing nodes use capacity-achieving channel codes in order to reliably convey information to the FC. As a consequence, the jth sensing node can transmit up to C_j bits of information to the FC. We assume that the channel gain is constant between the scheduling phase and the inference phase.

In this chapter, we focus on the scheduling and resource allocation problem, and

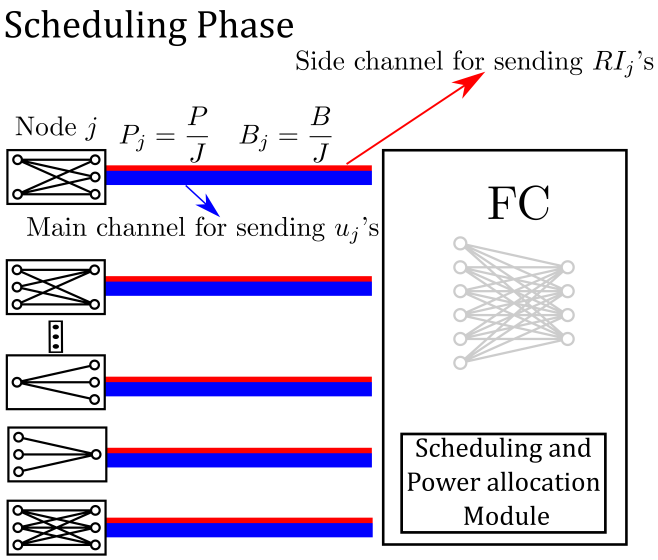
do not discuss the training procedure. We assume that the network can be re-trained during the scheduling phase. We consider power to be the resource that needs to be allocated. We do not treat the problem of intelligent bandwidth allocation and assume that the bandwidth is allocated equally among the nodes during both the scheduling and inference phases. Additionally, we assume that the effect of the bandwidth on the quality of the channel is negligible. As such, the channel capacity is known and fixed for both the scheduling and inference phases. In the following section, we describe the scheduling and power allocation algorithms.

5.3 Proposed Architecture

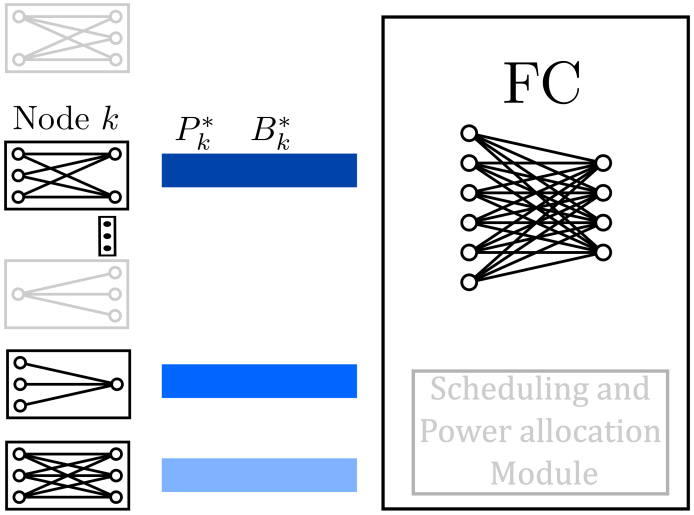
Let B represent the total available frequency bandwidth of the communication channels between the nodes and the FC, and P represent the total available power to be allocated between the nodes. During the scheduling phase, J available subcarriers are used, with the bandwidth B allocated equally among them. Additionally, the total available power is also equally allocated among the nodes. The choice of equal power and bandwidth allocation is important, as it gives each node an equal chance to be selected during the scheduling phase. We divide the scheduling phase in two parts. During the first part, every device obtains its own measure of the relevancy of its observed data. We refer to this measure as the *local assessment*, and denote the value obtained with RI_j for each node $j \in \mathcal{J}$. See Section 5.3.1 for more information on how each node computes this local assessment. The purpose of these local assessments is to help the FC compensate for not observing $X_1, \ldots, X_J$ directly.

Each node $j \in \mathcal{J}$ transmits the obtained RI_j to the FC. The transmission occurs over a negligible portion, $\alpha B/J$, of the available bandwidth of the communication channel, with $\alpha \in [0:1]$ and close to 0. The rest of the bandwidth is allocated for the transmission of the encodings of $\mathbf{x}_j$, $\mathbf{u}_j = [u_{j,1}, \ldots, u_{j,n}]$. This is shown in Figure 5.4a.

The second part of the scheduling phase begins once the FC receives all local assessments $RI_1, \ldots, RI_J$ as well as all standard features $\mathbf{u}_1, \ldots, \mathbf{u}_J$. Using all the received data, the FC performs its own, global assessment of how relevant each node is. The procedure for computing this global assessment is detailed in Section 5.3.2. Then, the



(a) Power and bandwidth allocation during scheduling phase.



(b) Power and bandwidth allocation during inference phase. P_k^* and B_k^* are the power and bandwidth allocated to node k during the scheduling phase.

Figure 5.4: The scheduling and inference phases of our proposed resource allocation solution. Grayed out components represent inactive components.

FC selects a subset of $K \leq J$ devices[1] to be scheduled for use during inference. The scheduling algorithm is detailed in Section 5.3.3.

Subsequently, the FC allocates the total available power to the K selected nodes. This is shown in Figure 5.4b. Each node is allocated power proportionally to the quality of its communication channel and its assigned global assessment. The power allocation procedure is discussed in Section 5.3.4.

5.3.1 Local Assessment

In this section, we describe how each node can compute its local assessment of the data it observes, the RI_j. Other works have shown the usefulness of local assessments for the problem of scheduling or resource allocation. In [68, Example 2], the authors give an example where a distributed quantizor performs best when it is designed based on the knowledge of both the channel and the local distribution. In [63], the authors present experiments where power allocation performs best when both the channel quality and the local assessment of the data are taken into account.

We argue that one can always find a suitable relevance indicator for a given task. In [64], the local assessment is calculated using the SNR. For the purpose of scheduling devices for distributed training, in [57], the authors introduce two *uncertainty* measures, as viewed by the model under training, while the authors of [58] use the significance of the model updates to assess the importance of each node.

In this chapter, we use the mutual information (MI) between the data and the target variable, $I(Y; X_j)$, to obtain the local assessment at each node. As shown in [10], there is a strong connection between the average logarithmic loss, which we use in our framework, and the MI. While in general MI cannot be computed analytically, it can be estimated using samples using NN-based estimators such as MINE [69].

These examples are not exhaustive. Other functions/techniques can be used together with the algorithms we provide in Section 5.3.3 for scheduling devices. The chosen local assessment function must meet three main characteristics. Firstly, the resulting output should be a value that can be sent over a side channel with a much smaller rate than the primary channel. Secondly, data that contain more useful information

[1] Ideally the number K itself should be subject to design, but we do not delve into that in this chapter.

regarding the target variable should receive a higher value from the local assessment. Finally, the local assessment should always output a positive value.

5.3.2 Global Assessment

The global assessment is a task-dependent value, assigned to each node, which indicates the FC's believed usefulness of that node, for the desired task. We denote such a value as a global relevance indicator. When the FC computes its believed usefulness of each node, it needs to take into account the information that the other nodes possess. It was already shown in [62] that power allocation leads to better performance when the correlation between the observations is taken into account, compared to assuming that the observations are independent.

Unfortunately, the FC only observes compressed versions of the data sensed by each node, for a given resource allocation, and not the data itself. As a consequence, the correlation between the data observed by each node cannot be computed. To overcome this issue, our proposed measure, the global relevance indicator, takes into account the correlation between the compressed representations that the FC receives, while also taking into account local assessments of the observed data. More specifically, the global relevance indicator is formed by combining

- the local assessments, described in Section 5.3.1, obtained by each node,
- assessments of the relative usefulness of the data received from each of the nodes.

We measure the relative usefulness of the data received from each node using a scaled conditional mutual information (CMI), i.e., to compute the usefulness of the data from node j, relative to node i, we compute $I(Y;U_j|U_i)$, and then scale it by $I(Y;U_1,\ldots,U_J)$.

More specifically, for the case where there are J nodes, the global relevance indicator of a node $j \in \mathcal{J}$, in relation to a subset of nodes $\mathcal{K}_p \subseteq \mathcal{J}/\{j\}$, is given by

$$GRI_j^{\mathcal{K}_p} = \left(\frac{RI_j}{\sum_{i=1}^{J} RI_i}\right)\left(\frac{I(Y;U_j|U_{\mathcal{K}_p})}{I(Y;U_1,\ldots,U_J)}\right). \tag{5.2}$$

In most practical applications, calculating the MI $I(Y;U_1,\ldots,U_J)$ and CMI $I(Y;U_j|U_{\mathcal{K}_p})$ is not possible; in such cases, the values must be estimated using the available data. Let

$\hat{I}(Y;U_j|U_{\mathcal{K}_p})$ denote such an estimate of $I(Y;U_j|U_{\mathcal{K}_p})$ and $\hat{I}(Y;U_1,\ldots,U_J)$ denote such an estimate of $I(Y;U_1,\ldots,U_J)$. We now define the approximation of the global relevance indicator using

$$\widehat{GRI}_j^{\mathcal{K}_p} = \left(\frac{RI_j}{\sum_{i=1}^{J} RI_i}\right)\left(\frac{\hat{I}(Y;U_j|U_{\mathcal{K}_p})}{\hat{I}(Y;U_1,\ldots,U_J)}\right) \tag{5.3}$$

Any data-based technique can be used to estimate the MI and CMI terms, including those presented in [69–71] and references within. In what follows, we detail a method for estimating $I(Y;U_j|U_{\mathcal{K}_p})$, using the trained NN at the FC. We first split the term into the difference between two conditional entropies

$$I(Y;U_j|U_{\mathcal{K}_p}) = H(Y|U_{\mathcal{K}_p}) - H(Y|U_{\mathcal{K}_p\cup\{j\}}). \tag{5.4}$$

Each of the conditional entropies can be estimated using the decoding NN at the FC

$$H(Y|U_{\mathcal{K}_p}) \approx \sum_{i=1}^{n} \log Q_{\boldsymbol{\phi}}(y_i|\mathbf{u}_{\mathcal{K}_p,i}), \tag{5.5}$$

where n is the number of samples available and $\boldsymbol{\phi}$ are the parameters of the NN at the FC.

The NN at the decoder can be trained to estimate Y based on data from a variable number of nodes, e.g., using techniques to deal with missing data; see [72] for more details, or it can be retrained for different combinations of available encoders. It is important to note that the estimator presented in (5.5) can be biased due to the variational approximation. Despite this, by using the same decoder as the one that will be used during inference, we will obtain a good measure of the usefulness of the nodes during the inference phase, as the bias will also be present during the inference phase.

A similar approach can be used to estimate the MI $I(Y;U_1,\ldots,U_J)$. One can similarly split the term into a difference between two entropy terms

$$I(Y;U_1,\ldots,U_J) = H(Y) - H(Y|U_1,\ldots,U_J), \tag{5.6}$$

and then estimate $H(Y|U_1,\ldots,U_J)$ using (5.5) and estimate $H(Y)$ using an empirical

estimator such as those of [73, 74].

5.3.3 Scheduling

The scheduling policy looks at selecting K nodes, of the total J nodes, to transmit during inference. As such, we use the previously defined global relevance indicator; and the FC selects the K nodes that are ranked the highest using the global relevance indicator technique.

More specifically, the FC ranks the nodes using the global relevance indicator as follows

$$r_i = \underset{j \in \mathcal{J}/\{r_1,\dots,r_{i-1}\}}{\operatorname{argmax}} \; GRI_j^{\{r_1,\dots,r_{i-1}\}}, \tag{5.7}$$

where $GRI_j^{\{r_1,\dots,r_{i-1}\}}$ is defined as in (5.3) and r_i is the index of the node of rank i according to the global relevance indicator. The FC then selects the top K ranked nodes, we denote the set of such nodes as $\mathcal{K} = \{r_1, \dots, r_K\}$. Given that in practice $GRI_j^{\{r_1,\dots,r_{i-1}\}}$ cannot be computed, we instead use the estimate presented in (5.3), and denote it $\widehat{GRI}_j^{\{r_1,\dots,r_{i-1}\}}$. For simplicity, in the remainder of the chapter we will use the notation $\widehat{GRI}_{r_i}$ to refer to $\widehat{GRI}_{r_i}^{\{r_1,\dots,r_{i-1}\}}$. This process is described in Algorithm 1.

Algorithm 1: Algorithm 1 - estimate the relative importance coefficients

Require: descriptions $\mathbf{u}_j$ and local assessments RI_j for $j \in \mathcal{J}$, and the desired number of nodes K.

Ensure: Rankings $r_1, \dots, r_K$ and scores $\widehat{GRI}_{r_1}, \dots, \widehat{GRI}_{r_K}$.

1: **initialize**
2: $r_0 = \emptyset$
3: **for** $i = 1$ to K **do**
4: Compute $\widehat{GRI}_j^{\{r_0,\dots,r_{i-1}\}}$ for every $j \in \mathcal{J}/\{r_0, \dots, r_{i-1}\}$ using (5.3).
5: Select node with highest $\widehat{GRI}_j^{\{r_0,\dots,r_{i-1}\}}$ using

$$r_i = \arg \max_{j \in \mathcal{J}/\{r_0,\dots,r_{i-1}\}} \widehat{GRI}_j^{\{r_1,\dots,r_{i-1}\}}$$

6: Save the computed global relevance indicator value $\widehat{GRI}_{r_i} = \widehat{GRI}_{r_i}^{\{r_0,\dots,r_{i-1}\}}$
7: **end for**
8: **return** $r_1, \dots, r_K, \widehat{GRI}_{r_1}, \dots, \widehat{GRI}_{r_K}$.

5.3.4 Power Allocation

Once the subset $\mathcal{K} = \{r_1, \ldots, r_K\}$ of nodes is selected, one needs to allocate the total available power to them. The simplest way would be to allocate the power equally among the nodes; however, this would not be an efficient way of power allocation. The waterfilling [75, 76] approach has shown that to maximise the amount of information that can be transmitted, it is better to allocate more power to the nodes with better channels. At the same time, each node is limited by the amount of useful information it observes regarding the variable of interest Y; any power allowing the node to send more information than what it observes about Y will not be able to improve the performance of the network. Finally, the power allocation also needs to take into account the usefulness of the node for inferring Y at the FC. As such, the power allocation algorithm needs to strike a balance between maximizing the total information the nodes transmit to the FC, the usefulness of the transmitted information towards inferring Y, and the total amount of useful information regarding Y that each node can transmit to the FC during inference.

To achieve this desired trade-off, we introduce a measure, that for each node $r_k \in \mathcal{K}$, combines the usefulness of the node towards inferring Y under equal power allocation, and the amount of information regarding Y that is contained in X_{r_k} but could not be transmitted, under equal power allocation. We denote this term as the expected usefulness (EU), and define it, for the node ranked in r_k as

$$EU_{r_k} = I(Y; U_{r_k} | U_{r_1}, \ldots, U_{r_{k-1}}) \frac{I(Y; X_{r_k} | U_{r_k})}{I(Y; X_{r_k})} \tag{5.8}$$

The first term, $I(Y; U_{r_k} | U_{r_1}, \ldots, U_{r_{k-1}})$, is estimated during the scheduling phase and quantifies the usefulness of the node under equal power allocation. Although not directly related to the channel gain, it can be observed that, under equal power allocation, a worse channel would be able to transmit less information compared to a node with better channel, leading to a smaller $I(Y; U_{r_k} | U_{r_1}, \ldots, U_{r_{k-1}})$, everything else being the same. As such, nodes with better channels will tend to have higher EU which will lead to higher power allocation, as will be seen later. This ensures that the intuition behind the waterfilling approach is met. On the other hand, we also need to ensure

that the node observes enough information about Y such that it can make use of the allocated power. To resolve this, the second term, $\frac{I(Y;X_{r_k}|U_{r_k})}{I(Y;X_{r_k})}$, is introduced, to estimate the amount of information the node was not able to transmit, regarding Y, due to the channel constraints. The term can be rewritten as

$$\frac{I(Y;X_{r_k}|U_{r_k})}{I(Y;X_{r_k})} = \frac{I(Y;X_{r_k}) - I(Y;U_{r_k})}{I(Y;X_{r_k})} \tag{5.9}$$

where the term $I(Y;X_{r_k}) = RI_{r_k}$ as described in Section 5.3.1 and is estimated locally by the node, while $I(Y;U_{r_k})$ is estimated during the calculation of the global relevance indicator, more specifically during the ranking of the first node.

Afterwards, the total power, denoted by P, is distributed proportionally to the assigned EU. More specifically, the power allocated to node $r_k \in \mathcal{K}$, which we denote by P_{r_k}, is computed as

$$P_{r_k} = \left(\frac{EU_{r_k}}{\sum_{r_j \in \mathcal{K}} EU_{r_j}}\right) P \quad \forall r_k \in \mathcal{K}. \tag{5.10}$$

Remark 5.3.1. *The power allocation strategy described by* (5.10) *allocates the total power to the selected nodes. The same technique can also be used to allocate only the power released by removing or shutting off some of the nodes. Effectively* (5.10) *can be changed to*

$$P_{r_k} = \frac{P}{J} + \left(\frac{EU_{r_k}}{\sum_{r_j \in \mathcal{K}} EU_{r_j}}\right) \frac{P(J-K)}{J} \quad \forall r_k \in \mathcal{K}. \tag{5.11}$$

Using (5.11) *can have an advantage, compared to* (5.10)*, as it will prevent extreme cases, in which most of the power goes to only one node. The disadvantage is that the improvement over equal power allocation will be reduced.*

5.4 Experiments

We perform a series of experiments to highlight the advantages of our technique. We showcase how the scheduling algorithm of Section 5.3.3 takes into account the channel gain, local assessment and the relationship between the data observed by the nodes when making the selection. Additionally, we showcase the improvements our power

allocation strategy offers.

5.4.1 Experimental Setup

Device	DNN Layers	Device	DNN Layers
Encoder $j \in \{1, \ldots, 9\}$	Input layer 14x7x1 Conv 2x2@32 BatchNormalization Dropout 0.5 Conv 2x2@64 Flatten Dense 128 Dropout 0.5 Dense 50 Normalize	Decoder	Input layer Jx50 Dense 128 Dropout 0.3 Dense 10

Table 5.1: NN layers for the encoders and decoder at the local nodes and FC, respectively.

We consider a variation of the problem of classifying MNIST [77] digits. In our setup, there are nine local nodes connected to one FC; each local node observes a noisy segment of the MNIST image. Eight of the nodes observe segments that are non-overlapping and of dimension 14x7 pixels. One node, node 3, observes a noisy version of the same image observed by node 2. Each segment is corrupted with zero-mean additive Gaussian noise of different variance; Table 5.2 shows the noise distribution for each node. The setup is shown in Figure 5.5. Each node $j \in [1:9]$ is equipped with a deterministic NN-based encoder, the architecture of which is shown in Table 5.1, which, for a given input $x_{j,i}$ outputs a latent vector $\mathbf{u}_{j,i}^{50} = [u_{j,i}^1, \ldots, u_{j,i}^{50}]$. Since the following theory applies to all samples $i \in [1:n]$ we will drop the sample subscript for simplicity. For reasons that will become clearer, in what follows, we normalise the vector $\mathbf{u}_j^{50}$ during both inference and training. The FC is similarly equipped with an NN-based decoder whose architecture is shown in Table 5.1. We train the models using categorical cross-entropy with a batch size of 512 for 100 epochs.

Each node j needs to transmit each element of the latent vector $\mathbf{u}_j^{50}$ over a constrained communication channel of capacity C_j, given by equation (5.1). We consider separate source-channel coding; Figure 5.6 shows a diagram of the communication channel. Due to the communication constraints, the source encoder needs to compress the data

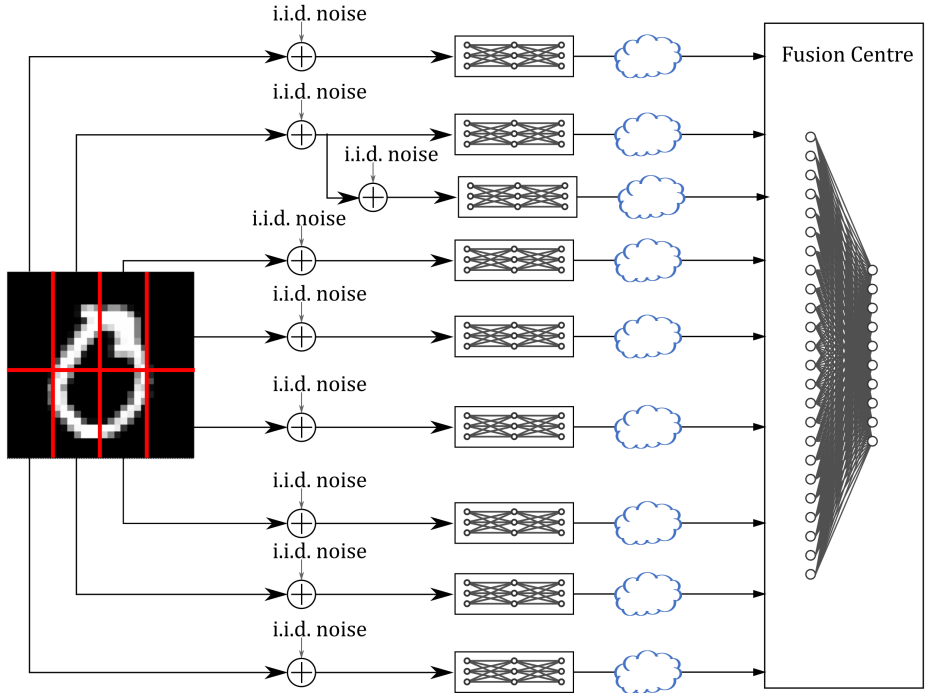


Figure 5.5: Diagram showcasing experiment setup.

into R_j bits, which, under the assumption of capacity-achieving channel codes, satisfies

$$R_j = C_j. \tag{5.12}$$

For ease of calculation, we model the latent variable as i.i.d samples from a Gaussian distribution of zero mean and unit variance. To ensure that the latent vector has zero mean and unit variance, we normalise $\mathbf{u}_j^{50}$ during both the training and inference procedure. To bypass the issue of developing and fixing the codebooks prior to training, we approximate the codewords by adding noise to each instance of the latent variable. We make this approximation, as in this setup we are interested only in finding a relationship between the distortion caused by the compression of the latent variable and the channel capacity. More specifically, we compute the vector received by the FC from node $j \in \mathcal{J}$, which we denote by $\hat{\mathbf{u}}_j^{50}$, as

$$\hat{u}_j^t = u_j^t - z_j^t \; \forall \; t \in [0:50] \quad \text{with} \tag{5.13}$$

$$z_j^t \sim \mathcal{N}\left(\frac{u_j^t}{1+h_j^2 P_j}; \frac{1}{1+h_j^2 P_j} - \frac{1}{(1+h_j^2 P_j)^2}\right). \tag{5.14}$$

This approximation is derived based on the lossy source coding of Gaussian sources [25]. More specifically, we use the fact that to compress a source $U \sim \mathcal{N}(0, \sigma_U^2)$ into $\hat{U}$,

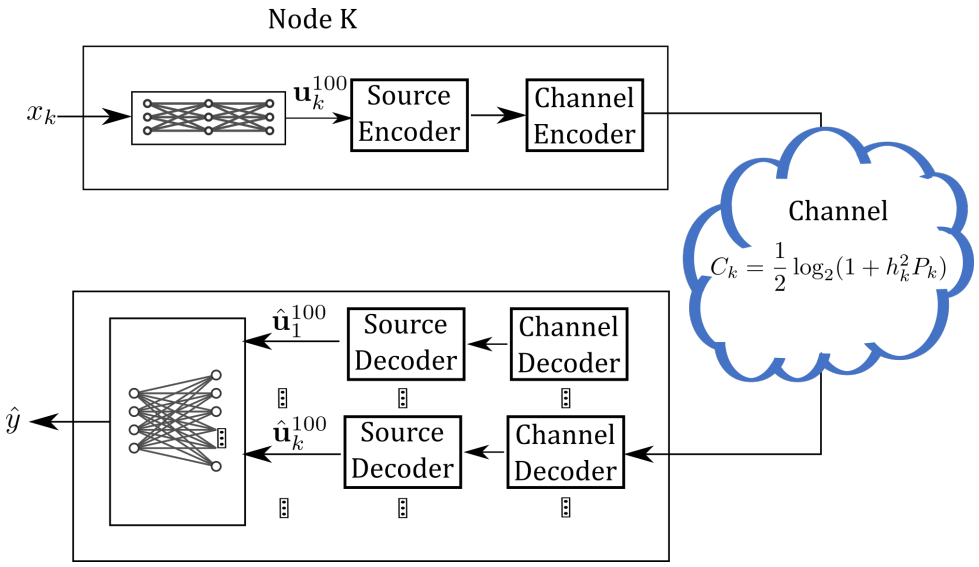


Figure 5.6: Diagram showcasing channel model.

such that $\hat{U}$ can be transmitted over a channel using R bits, with minimal mean square error distortion and in the large data regime, the optimal test channel $p(U|\hat{U})$ is given by

$$U = \hat{U} + Z \quad \text{with} \tag{5.15}$$

$$Z \sim \mathcal{N}(0, \sigma_Z^2); \quad \hat{U} \sim \mathcal{N}(0, \sigma_U^2 - \sigma_Z^2); \tag{5.16}$$

$$R = \frac{1}{2}\log_2 \frac{\sigma_U^2}{\sigma_Z^2}. \tag{5.17}$$

To derive the noise variance for each node, we use equations (5.17), (5.12) and (5.1) from which we obtain

$$\sigma_{Z_j}^2 = \frac{\sigma_{U_j}^2}{1 + h_j^2 P_j}. \tag{5.18}$$

While equation (5.18) gives us the variance of the noise, it is important to note, from equation (5.15), that the noise is not independent of latent variable, i.e., $Z_j \not\perp U_j$. As such we derive $p(Z_j|U_j)$ using equation (5.15) and the fact that U_j, $\hat{U}_j$, and Z_j are all Gaussian

$$p(Z_j|U_j = u_j) \sim \mathcal{N}\left(\frac{\sigma_{Z_j}^2}{\sigma_{U_j}^2} u_j; \sigma_{Z_j}^2 - \frac{\sigma_{Z_j}^4}{\sigma_{U_j}^2}\right). \tag{5.19}$$

We then obtain equation (5.14) from the assumption that $U_j \sim \mathcal{N}(0,1)\ \forall j \in \mathcal{J}$, and

equations (5.19) and (5.18).

In our experiments, the local assessments, *RI*s, are calculated using the estimated mutual information between the local data and the label, which is estimated using a locally trained NN. Table 5.2 shows the channel gains and *RI*s for each node.

Node ID	Data Noise	h_j^2	RI_j
Node 1	No Noise	1.e-01	0.27
Node 2	$\mathcal{N}(0; 0.7)$	7.e-02	1.13
Node 3	$\mathcal{N}(0; 0.1)$	2.e-02	1.11
Node 4	$\mathcal{N}(0; 0.6)$	6.e-04	1.4
Node 5	$\mathcal{N}(0; 0.3)$	1.e+00	0.43
Node 6	$\mathcal{N}(0; 0.1)$	1.e-05	0.44
Node 7	$\mathcal{N}(0; 1.5)$	3.e-03	0.64
Node 8	$\mathcal{N}(0; 0.2)$	1.e-06	1.65
Node 9	$\mathcal{N}(0; 0.2)$	1.e-05	0.43

Table 5.2: The noise of the view, the channel gain, and the local assessment for each of the nodes.

5.4.2 Scheduling Experiment

Our aim is to schedule $K = 3$ devices of the nine total nodes, given a total power of $P = 100$. We use the scheduling algorithm presented in Algorithm 1, the results of which are shown in Table 5.3. The table contains the $\widehat{GRI}_j$ value obtained for each node, given the previously selected nodes, with the values in green representing the highest values, i.e., the chosen nodes. The networks are re-trained for each configuration of the selected encoders for an accurate estimation of the entropy terms. The entropy of Y is estimated using the naive plug-in estimator [73]. We remark that while the GRI_j measure given by (5.2) is always positive, the empirical estimate $\widehat{GRI}_j$ computed using (5.3) can be negative for the case where (5.4) is very small due to estimation errors.

As observed in Table 5.3 the three chosen nodes are nodes 2, 4 and 7. The first node to be selected is node 2. The choice intuitively makes sense, node 2 observes a central part of the image, thus containing a lot of information about the target ariable, i.e., the class. This can be observed from the *RI*; compared to nodes that observe parts of the image at the periphery, node 2 has a better local assessment, i.e., higher *RI*, see Table 5.2. Compared to other nodes that observe central parts of the image, node 2 has

Node ID	$\widehat{GRI}_j^{\emptyset}$	$\widehat{GRI}_j^{\{r_1\}}$	$\widehat{GRI}_j^{\{r_1,r_2\}}$
Node 1	5.54e-03	3.5e-03	-8.12e-04
Node 2	*9.33e-02*	AS	AS
Node 3	7.47e-02	1.21e-02	-6.12e-03
Node 4	6.9e-03	*4.25e-02*	AS
Node 5	1.47e-03	1.36e-02	4.02e-03
Node 6	9.48e-04	5.66e-03	2.1e-03
Node 7	8.22e-03	1.81e-02	*8.38e-03*
Node 8	-5.71e-07	8.66e-04	2.12e-03
Node 9	1.07e-03	5.3e-03	1.3e-03

Table 5.3: The global relevance indicator values obtained for each of the nodes during the scheduling phase. The notation AS signifies nodes that have already been selected, and for which the global relevance indicator values were not computed.

a better channel.

The second node to be chosen is node 4. Compared to the other nodes, the global relevance indicator of node 4, given that node 2 was selected, is the highest. Node 4 has the highest local assessment compared to the other nodes, all except node 8 which has a very bad channel, as such intuitively it makes sense to be selected. It is interesting to observe that node 3 is not selected. Since node 3 observes the same information as node 2, once node 2 is selected, the usefulness of node 3 drops significantly.

The final node to be selected is node 7. It can be observed that the node has a balance of observing good quality data, compared to most of the remaining nodes, and a good channel. Additionally, node 7 observes data related to the lower part of the image, which neither node 2 or 4 do, and as such could provide complementary information. This balance makes node 7 the most useful of the remaining nodes.

We compare our selection strategy with three other possible strategies. In the first case, we select the three nodes with the best channels. In the second case, we select the three nodes with the best local assessments. In the final case, we rank the nodes according to the CMI. More specifically, the node in rank i is selected using the following

$$r_i = \arg \max_{j \in \mathcal{J}/\{r_1,\ldots,r_{i-1}\}} \hat{I}(Y; \hat{U}_j | \hat{U}_{r_1}, \ldots, \hat{U}_{r_{i-1}}). \tag{5.20}$$

We then select the three nodes with the highest rank. This technique is similar to our approach, however, it does not incorporate the use of local assessments.

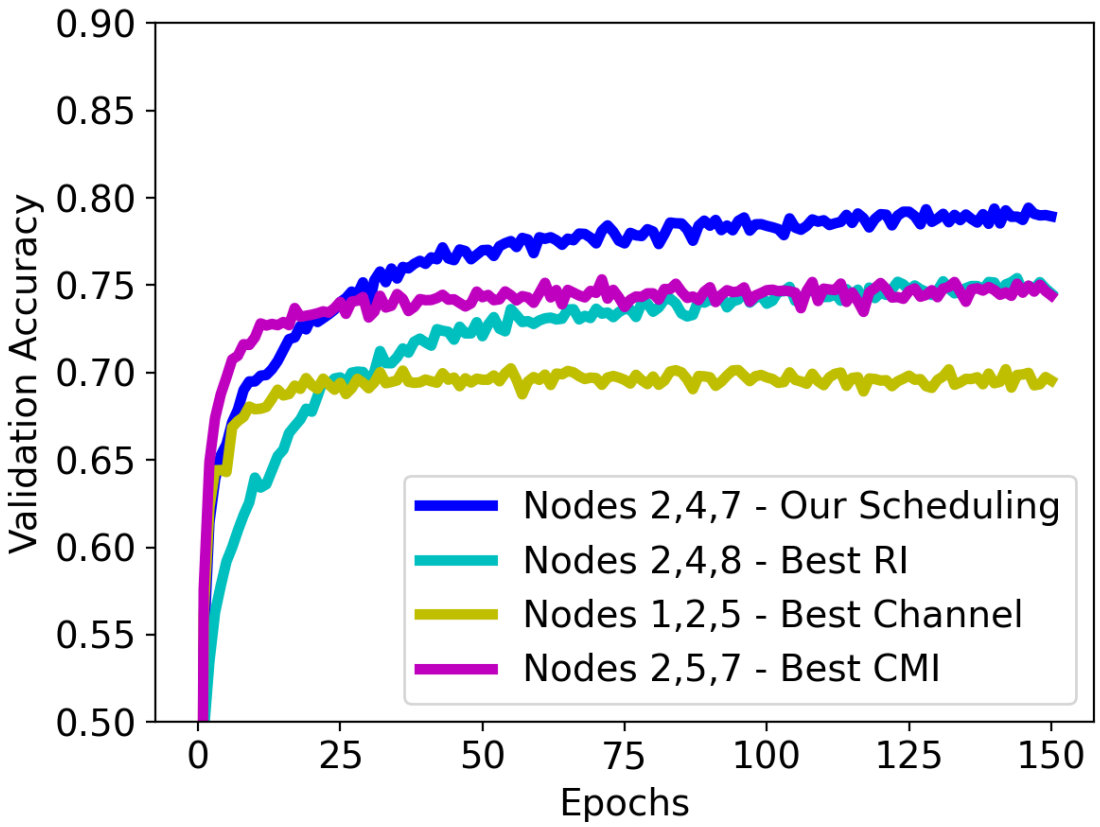


Figure 5.7: The accuracy of the system under different scheduling strategies with equal power allocation.

From Figure 5.7 it can be seen that our scheduling algorithm outperforms the three other selection strategies. This shows that simply using channel information or the local assessment does not provide enough information to make a useful selection strategy. Furthermore, comparing our selection algorithm to that based solely on the CMI, given by (5.20), we highlight the need to transmit the local assessment to the FC and incorporate them into the selection criteria. Intuitively, (5.20) only gives information of the usefulness of each node under the given equal power allocation to all nodes. The scheduling, however, changes the power allocation by allocating all the power to a subset of the nodes, the effect of which the FC cannot take into account, since it only observes compressed representations. By having each node compute a local assessment, which is then incorporated into the computation of the global relevance indicator, we mitigate the effect of the global assessment being computed only on compressed data.

5.4.3 Power Allocation Experiment

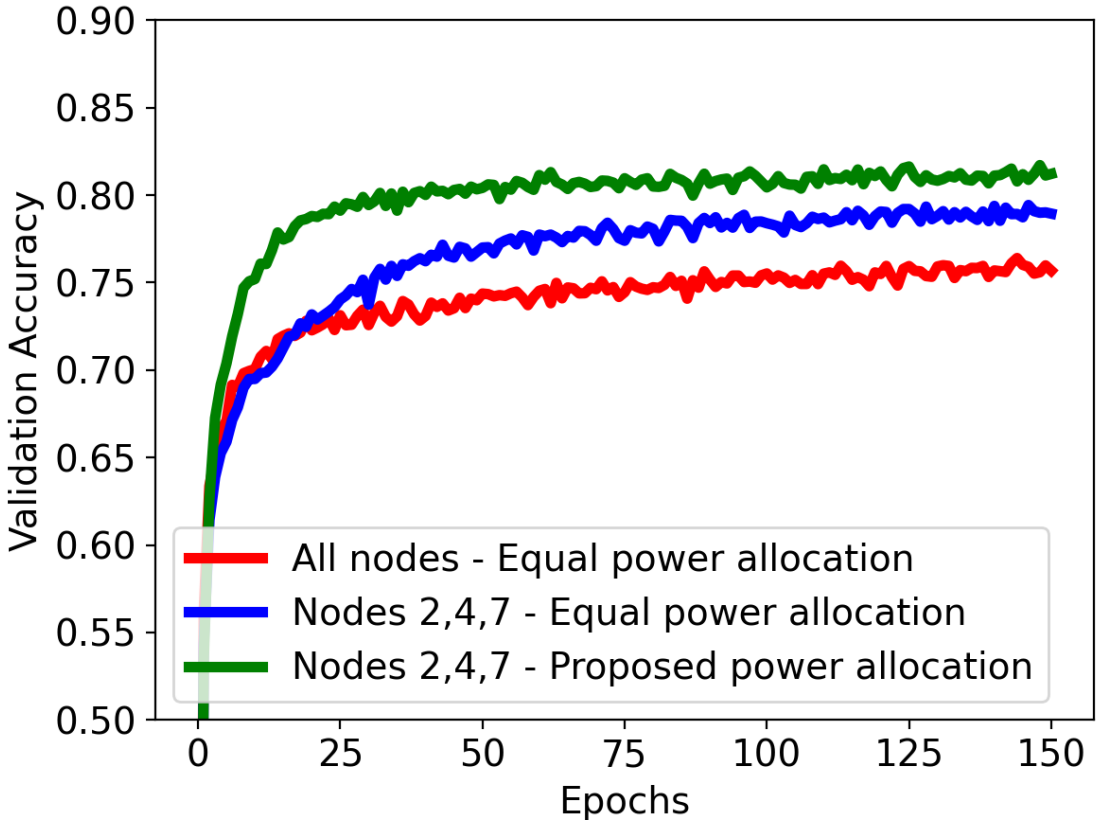


Figure 5.8: The accuracy of the system under different power allocation strategies.

Using our proposed power allocation strategy, described in Section 5.3.4, node 2 is allocated 16.2% of the power, node 4 is allocated 62.6% of the power, while node 7 receives 21.2% of the remaining power. As can be seen from the allocation strategy, the algorithm looks to allocate power to the nodes that are both useful to the prediction of Y and also have additional information regarding Y that was not sent during the equal power allocation phase. Node 4, which both has a great impact on predicting Y and is unable to transmit all the useful data it observes, receives the highest amount of additional power. Node 2 and node 7 receive similar amounts of power. While node 2 is more useful towards predicting Y, node 7 has more additional information that was not transmitted during the equal power allocation phase. As can be seen in Figure 5.8, this power allocation strategy outperforms equal power allocation as well as equal power allocation when all the nodes are scheduled.

Chapter 6

Conclusions and Perspectives

6.1 Conclusions

In this thesis, we have focused on the distributed training and inference problem. We discussed how multiple nodes can work together to jointly train a model that can be used in a distributed manner during the inference phase through a novel algorithm named in-network learning (INL). Compared to other works on distributed training [12,22], we assume that the nodes observe data related to the same event. The proposed INL algorithm provides a loss function derived through the theoretical analysis of the best trade-off the model can achieve between the accuracy of the predictions, under logarithmic loss, and the amount of the information passed between the nodes of the communication network.

We also presented a novel scheduling and power allocation algorithms for INL for the star-like topology. The algorithms combine the central nodes' knowledge of the relationships between the data it receives from the edge nodes with the edge nodes' knowledge of the usefulness of the observed data. We showcase, experimentally, the improvement our proposed scheduling and power allocation algorithms provide by combining these two types of knowledge.

6.2 Perspectives

Through this thesis, we considered the case in which the compression of the sources is designed separately from the transmission of the data over the channel, the so-called separation of source and channel coding. For the distributed case or the remote inference case, one can improve upon these results by designing the source and channel coding jointly, refereed to as joint source-channel coding (JSCC). Several works are already developing JSCC techniques using NNs for the remote inference case [78–80], however, extending this to the distributed cases provides additional challenges. One possible issue is that one might not be able to use gradient descent methods in this case as the channel might not be differentiable with respect to the latent representation transmitted by each node.

Another possible research avenue is the development of an extension of INL in which nodes exchange quantised values during training and inference. This would allow the deployment of INL in digital communication channels and would significantly reduce communication constraints. The challenge of implementing such a technique is the fact that a discreet latent representation would again prevent the use of gradient descent methods. Although there are techniques that allow neural network training even when the latent representation is discreet, see [80–82], these techniques are not appropriate for the problem setting considered in INL. The authors of [80] consider only the problem of communication, and not inference, and in [81] the authors have shown that the quantisation of the latent space should take into account the task that the model seeks to solve. The techniques of both [81,82] can be used for developing an INL model with a discreet latent representation; however, they require that the model be trained locally, which is not the case in INL.

Bibliography

[1] J. Väyrynen, "Learning linguistic features from natural text data by independent component analysis," master's thesis, Helsinki University of Technology, 2005.

[2] C. T. Li, X. Wu, A. Ozgur, and A. El Gamal, "Minimax learning for remote prediction," in *2018 IEEE International Symposium on Information Theory (ISIT)*, p. 541–545, IEEE Press, 2018.

[3] A. Zaidi, I.-E. Aguerri, and S. Shamai, "On the information bottleneck problems: Models, connections, applications and information theoretic views," *Entropy*, vol. 22, no. 2, p. 151, 2020.

[4] M. Majid, S. Habib, A. R. Javed, M. Rizwan, G. Srivastava, T. R. Gadekallu, and J. C.-W. Lin, "Applications of wireless sensor networks and internet of things frameworks in the industry revolution 4.0: A systematic literature review," *Sensors*, vol. 22, no. 6, 2022.

[5] E. Horvitz and D. Mulligan, "Data, privacy, and the greater good," *Science*, vol. 349, no. 6245, pp. 253–255, 2015.

[6] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 years: A survey," *CoRR*, vol. abs/1905.05055, 2019.

[7] J. P. W. Pluim, J. B. A. Maintz, and M. A. Viergever, "Mutual-information-based registration of medical images: a survey," *IEEE Transactions on Medical Imaging*, vol. 22, no. 8, pp. 986–1004, 2003.

[8] J. Kober, J. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, pp. 1238–1274, 09 2013.

[9] O. Vinyals and Q. V. Le, "A neural conversational model," *CoRR*, vol. abs/1506.05869, 2015.

[10] I.-E. Aguerri and A. Zaidi, "Distributed variational representation learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 120–138, 2021.

[11] A. Zaidi, K. Scaman, and P. Escamilla, "In-network learning: Method and system for learning and inference over a telecommunication network," International Patent, Number 87180792 (filled), 2019.

[12] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017*, vol. 54, pp. 1273–1282, 2017.

[13] O. Gupta and R. Raskar, "Secure training of multi-party deep neural network." US Patent Number 20170372201A1, published 28 Dec 2017.

[14] M. Moldoveanu and A. Zaidi, "On in-network learning. a comparative study with federated and split learning," in *2021 IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 221–225, IEEE, 2021. Invited paper in the special session Machine Learning at the Edge.

[15] M. Moldoveanu and A. Zaidi, "In-network learning for distributed training and inference in networks," in *IEEE Globecom 2021 Workshops, Madrid, Spain, December 7-11, 2021*, pp. 1–6, IEEE, 2021.

[16] W. Zhengzhong, L. Zilin, L. Jun, and H. Xiaowei, "Wireless sensor networks for living environment monitoring," in *2009 WRI World Congress on Software Engineering*, vol. 3, pp. 22–25, 2009.

[17] A. Zaidi, "Joint relevance-channel aware device scheduling for in-network learning."

[18] A. Zaidi and P. Krasnowski, "Channel-aware device scheduling and power allocation for distributed inference," *Intermediate version*, June 2022.

[19] T. Cover and J. Thomas, *Elements of Information Theory*. Wiley, 2012.

[20] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.

[21] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," in *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pp. 368–377, 1999.

[22] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *J. Netw. Comput. Appl.*, vol. 116, pp. 1–8, 2018.

[23] A. Zaidi, M. Moldoveanu, and P. Krasnowski, "In-network learning: Distributed training and inference in networks," *In preparation for submission to IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2022.

[24] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," in *International Conference on Learning Representations*, 2017.

[25] A. El Gamal and Y.-H. Kim, *Network Information Theory*. Cambridge University Press, 2011.

[26] H. Witsenhausen, "Indirect rate distortion problems," *IEEE Transactions on Information Theory*, vol. 26, no. 5, pp. 518–521, 1980.

[27] R. Dobrushin and B. Tsybakov, "Information transmission with additional noise," *IRE Transactions on Information Theory*, vol. 8, no. 5, pp. 293–304, 1962.

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

[29] Y. Uğur, G. Arvanitakis, and A. Zaidi, "Variational information bottleneck for unsupervised clustering: Deep gaussian mixture embedding," *Entropy*, vol. 22, no. 2, 2020.

[30] Y. Shkel, M. Raginsky, and S. Verdú, "Universal lossy compression under logarithmic loss," in *2017 IEEE International Symposium on Information Theory (ISIT)*, pp. 1157–1161, 2017.

[31] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge university press, 2006.

[32] T. Andre, M. Antonini, M. Barlaud, and R. M. Gray, "Entropy-based distortion measure for image coding," in *2006 International Conference on Image Processing*, pp. 1157–1160, 2006.

[33] A. Tripathy, Y. Wang, and P. Ishwar, "Privacy-preserving adversarial networks," in *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 495–505, 2019.

[34] J. Jiao, T. Courtade, K. Venkat, and T. Weissman, "Justification of logarithmic loss via the benefit of side information," in *2014 IEEE International Symposium on Information Theory*, pp. 946–950, 2014.

[35] A. Painsky and G. Wornell, "On the universality of the logistic loss function," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 936–940, IEEE, 2018.

[36] A. No and T. Weissman, "Universality of logarithmic loss in lossy compression," in *2015 IEEE International Symposium on Information Theory (ISIT)*, pp. 2166–2170, 2015.

[37] P. Harremoes and N. Tishby, "The information bottleneck revisited or how to choose a good distortion measure," in *2007 IEEE International Symposium on Information Theory*, pp. 566–570, 2007.

[38] A. Achille and S. Soatto, "Information dropout: Learning optimal representations through noisy computation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2897–2905, 2018.

[39] R. Blahut, "Computation of channel capacity and rate-distortion functions," *IEEE Transactions on Information Theory*, vol. 18, no. 4, pp. 460–473, 1972.

[40] S. Arimoto, "An algorithm for computing the capacity of arbitrary discrete memoryless channels," *IEEE Transactions on Information Theory*, vol. 18, no. 1, pp. 14–20, 1972.

[41] G. Chechik, A. Globerson, N. Tishby, and Y. Weiss, "Information bottleneck for gaussian variables," *Advances in Neural Information Processing Systems*, vol. 16, 2003.

[42] A. Winkelbauer and G. Matz, "Rate-information-optimal gaussian channel output compression," in *2014 48th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–5, 2014.

[43] A. Winkelbauer, S. Farthofer, and G. Matz, "The rate-information trade-off for gaussian vector channels," in *2014 IEEE International Symposium on Information Theory*, pp. 2849–2853, 2014.

[44] M. Meidlinger, A. Winkelbauer, and G. Matz, "On the relation between the gaussian information bottleneck and mse-optimal rate-distortion quantization," in *2014 IEEE Workshop on Statistical Signal Processing (SSP)*, pp. 89–92, 2014.

[45] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2014.

[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[47] M. Vera, P. Piantanida, and L. R. Vega, "The role of information complexity and randomization in representation learning," *arXiv preprint arXiv:1802.05355*, 2018.

[48] R. Gilad-Bachrach, A. Navot, and N. Tishby, "An information theoretic tradeoff between complexity and accuracy," in *Learning Theory and Kernel Machines* (B. Schölkopf and M. K. Warmuth, eds.), pp. 595–609, Springer Berlin Heidelberg, 2003.

[49] L. Sun, C. Guo, and Y. Yang, "Adaptive information bottleneck guided joint source-channel coding," *arXiv preprint arXiv:2203.06492*, 2022.

[50] J. Shao, Y. Mao, and J. Zhang, "Learning task-oriented communication for edge inference: An information bottleneck approach," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 1, pp. 197–211, 2022.

[51] G. E. Hinton and D. van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," in *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, COLT '93, (New York, NY, USA), p. 5–13, Association for Computing Machinery, 1993.

[52] A. Achille and S. Soatto, "Emergence of invariance and disentanglement in deep representations," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 1947–1980, 2018.

[53] G. Flamich, M. Havasi, and J.-M. Hernández-Lobato, "Compressing images by encoding their latent representations with relative entropy coding," *CoRR*, vol. abs/2010.01185, 2021.

[54] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pp. 730–734, Nov 2015.

[55] C. T. Li and A. E. Gamal, "Strong functional representation lemma and applications to coding theorems," *IEEE Transactions on Information Theory*, vol. 64, no. 11, pp. 6967–6978, 2018.

[56] T. Berger and R. Yeung, "Multiterminal source encoding with one distortion criterion," *IEEE Transactions on Information Theory*, vol. 35, no. 2, pp. 228–236, 1989.

[57] D. Liu, G. Zhu, J. Zhang, and K. Huang, "Data-importance aware user scheduling for communication-efficient edge machine learning," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 265–278, 2020.

[58] M. M. Amiri, D. Gündüz, S. R. Kulkarni, and H. Vincent Poor, "Update aware device scheduling for federated learning at the wireless edge," in *2020 IEEE International Symposium on Information Theory (ISIT)*, pp. 2598–2603, 2020.

[59] W. Shi, S. Zhou, Z. Niu, M. Jiang, and L. Geng, "Joint device scheduling and resource allocation for latency constrained wireless federated learning," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 453–467, 2021.

[60] J. Perazzone, S. Wang, M. Ji, and K. Chan, "Communication-efficient device scheduling for federated learning using stochastic optimization," *CoRR*, vol. abs/2201.07912, 2022.

[61] J.-F. Chamberland and V. V. Veeravalli, "Wireless sensors in distributed detection applications," *IEEE Signal Processing Magazine*, vol. 24, no. 3, pp. 16–25, 2007.

[62] T. Wimalajeewa and S. K. Jayaweera, "Optimal power scheduling for correlated data fusion in wireless sensor networks via constrained pso," *IEEE Transactions on Wireless Communications*, vol. 7, no. 9, pp. 3608–3618, 2008.

[63] X. Zhang, H. V. Poor, and M. Chiang, "Optimal power allocation for distributed detection over mimo channels in wireless sensor networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 9, pp. 4124–4140, 2008.

[64] J.-J. Xiao and Z.-Q. Luo, "Decentralized estimation in an inhomogeneous sensing environment," *IEEE Transactions on Information Theory*, vol. 51, no. 10, pp. 3564–3575, 2005.

[65] Z.-Q. Luo, "An isotropic universal decentralized estimation scheme for a bandwidth constrained ad hoc sensor network," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 735–744, 2005.

[66] Z.-Q. Luo, "Universal decentralized estimation in a bandwidth constrained sensor network," *IEEE Transactions on Information Theory*, vol. 51, no. 6, pp. 2210–2219, 2005.

[67] J.-J. Xiao, S. Cui, Z.-Q. Luo, and A. Goldsmith, "Power scheduling of universal decentralized estimation in sensor networks," *IEEE Transactions on Signal Processing*, vol. 54, no. 2, pp. 413–422, 2006.

[68] B. Chen, L. Tong, and P. Varshney, "Channel-aware distributed detection in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 23, no. 4, pp. 16–26, 2006.

[69] M. Ishmael Belghazi, A. Baratin, S. Rajeswar, S. Ozair, Y. Bengio, A. Courville, and R. Devon Hjelm, "MINE: Mutual Information Neural Estimation," *arXiv e-prints*, p. arXiv:1801.04062, Jan. 2018.

[70] S. Molavipour, G. Bassi, and M. Skoglund, "Conditional mutual information neural estimator," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5025–5029, 2020.

[71] S. Mukherjee, H. Asnani, and S. Kannan, "Ccmi : Classifier based conditional mutual information estimation," in *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, vol. 115 of *Proceedings of Machine Learning Research*, pp. 1083–1093, PMLR, 22–25 Jul 2020.

[72] R. J. Little and D. B. Rubin, *Statistical analysis with missing data*, vol. 793. John Wiley & Sons, 2019.

[73] L. Paninski, "Estimation of entropy and mutual information," *Neural Computation*, vol. 15, no. 6, pp. 1191–1253, 2003.

[74] L. F. Kozachenko and N. N. Leonenko, "Sample estimate of the entropy of a random vector," *Problems Inform. Transmission*, vol. 23, pp. 95–101, 1987.

[75] E. Telatar, "Capacity of multi-antenna gaussian channels," *European transactions on telecommunications*, vol. 10, no. 6, pp. 585–595, 1999.

[76] M. Thomas and A. T. Joy, *Elements of information theory*. Wiley-Interscience, 2006.

[77] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[78] E. Bourtsoulatze, D. Burth Kurka, and D. Gündüz, "Deep joint source-channel coding for wireless image transmission," *IEEE Transactions on Cognitive Communications and Networking*, vol. 5, no. 3, pp. 567–579, 2019.

[79] N. Farsad, M. Rao, and A. Goldsmith, "Deep learning for joint source-channel coding of text," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 2326–2330, IEEE, 2018.

[80] K. Choi, K. Tatwawadi, A. Grover, T. Weissman, and S. Ermon, "Neural joint source-channel coding," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 1182–1192, PMLR, 09–15 Jun 2019.

[81] O. A. Hanna, Y. H. Ezzeldin, T. Sadjadpour, C. Fragouli, and S. Diggavi, "On distributed quantization for classification," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, no. 1, pp. 237–249, 2020.

[82] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6309–6318, Curran Associates Inc., 12 2017.

French Summary(Résumé Français)

Prenons l'exemple du problème d'identification ou de suivi d'un véhicule dans une grande zone urbaine. Des capteurs sont placés dans toute la zone pour collecter différents types d'informations, telles que le son, l'image/vidéo, la proximité, etc. Les capteurs doivent fonctionner ensemble pour prédire conjointement où se trouve le véhicule ou où il va. Les données recueillies par les capteurs sont corrélées. La relation exacte entre les données que nous observons et les données que nous voulons prédire n'est pas connue. Nous supposons que les capteurs collectent des données à partir desquelles nous pouvons former un modèle pour apprendre la relation entre les données observées et les informations que nous voulons prédire. Nous appelons cette période de temps la phase de formation. La collecte des échantillons de données des capteurs vers un emplacement central n'est pas toujours possible. Les capteurs peuvent être déployés dans une zone où l'infrastructure de communication est limitée [4]. Un autre problème possible est que des informations sensibles peuvent être contenues dans les données collectées. Le transfert des données pourrait violer les restrictions de confidentialité [5]. Pour ces raisons, tant la phase d'apprentissage que le déploiement du modèle, également appelée phase d'inférence, doivent se dérouler de manière distribuée, c'est-à-dire sans échange de données brutes. Ce problème est parfois appelé problème d'apprentissage distribué et d'inférence. La question est, compte tenu de certaines mesures de la performance du modèle, quelles données chaque nœud doit-il extraire de ses données observées? La suppression de l'une des données pertinentes ne peut que réduire les performances de prédiction. D'autre part, l'extraction et la transmission de données qui ne sont pas pertinentes pour la tâche à accomplir peuvent mettre à rude épreuve le réseau de communication ou même dégrader les performances du réseau. Ainsi, intuitivement, les nœuds doivent extraire le plus petit nombre de caractéristiques qui donnent collectivement les meilleures performances de prédiction. Décider quelles fonctionnalités extraire doit être fait sans coordination explicite entre les nœuds. Comment cela devrait-il être effectué de manière optimale sans partager de données brutes? En particulier, comment chaque nœud doit-il apprendre à traiter les informations et que doit-il transmettre aux autres nœuds de manière distribuée? Dans cette thèse, nous supposons qu'il existe des dispositifs, que nous désignons comme

nœuds périphériques, qui observent des données liées à une variable d'intérêt. Cette variable d'intérêt doit être déduite à un nœud de décision central ou à un centre de fusion. Les nœuds périphériques sont connectés au nœud central soit directement, soit via des nœuds intermédiaires. Nous supposons que la topologie du réseau peut être modélisée par un graphe orienté acyclique. La distribution des données n'est pas connue et seul un jeu de données distribué est disponible; c'est-à-dire que différentes parties des données sont disponibles à différents nœuds. Nous équipons chaque nœud d'un réseau de neurones (NN). Cela est dû au succès des techniques d'apprentissage automatique (ML) dans l'apprentissage, à partir d'échantillons, de relations complexes entre des données observées et une variable cible dans des domaines tels que la vision par ordinateur [6], le traitement d'image [7], la robotique [8], et le traitement du langage naturel [9]. Nous considérons également le cas où la performance du modèle est mesurée en termes de perte logarithmique. La perte logarithmique, ou perte logarithmique en abrégé, est une mesure de distorsion naturelle dans les contextes dans lesquels les reconstructions n'ont pas à être déterministes. C'est-à-dire que le décodeur fournit une évaluation de la probabilité de chaque estimation possible, au lieu de choisir une estimation. La mesure est largement utilisée dans divers contextes, y compris le clustering [29] et la classification [24], la reconnaissance de formes [30], l'apprentissage et la prédiction [31], le traitement d'image [32], la confidentialité [33], et autres. Nous concentrons sur la formation distribuée et la résolution de problèmes d'inférence. Nous discutons de la façon dont plusieurs nœuds, qui observent des données liées au même événement, peuvent travailler ensemble pour former conjointement un modèle qui peut être utilisé de manière distribuée pendant la phase d'inférence grâce à un nouveau cadre appelé apprentissage en réseau (INL). INL pour une topologie en étoile, dans laquelle les nœuds qui observent les données sont directement connectés au nœud de décision, a été introduit et analysé pour la première fois dans [10,11]. Nous présentons ces résultats et discutons en détail la procédure de formation pour un tel réseau. Plus précisément, nous discutons comment, dans le cas où chaque nœud est équipé d'un NN, un tel réseau peut être formé en utilisant la rétropropagation. Nous expliquons comment, en imposant la condition selon laquelle le NN au centre de fusion utilise comme entrée un vecteur formé en concaténant les informations reçues des nœuds périphériques, lors de

la rétropropagation, le vecteur d'erreur peut être divisé et transmis du centre de fusion à chaque nœud périphérique correspondant. Cette observation a également été faite dans le document de brevet [11] par A. Zaidi, K. Scaman et P. Escamilla. Nous nous appuyons sur ces résultats en fournissant une nouvelle comparaison expérimentale d'INL avec d'autres algorithmes d'apprentissage distribué, à savoir ceux de [12, 13]. Nous considérons ensuite l'extension de l'INL aux topologies de réseau qui peuvent être modélisées avec un graphe dirigé et acyclique. Cela a été proposé et étudié pour la première fois dans le document de brevet [11] par A. Zaidi, K. Scaman et P. Escamilla. Dans cette thèse, nous dérivons une fonction de perte appropriée pour un tel modèle. L'approche de la dérivation de la fonction de perte est empruntée à [10], c'est-à-dire que nous supposons d'abord que les distributions conjointes sont connues et procédons à la modélisation du problème en tant que source de réseau en codant une sous mesure de perte logarithmique et en trouvant approprié (réalisable) compromis taux-distorsion, puis nous dérivons une limite interne variationnelle, et enfin nous paramétrons cette limite inférieure/interne à l'aide de réseaux de neurones.

Nous dérivons d'abord un compromis débit-distorsion réalisable pour toute topologie de réseau qui peut être modélisée avec un graphe acyclique dirigé. La preuve suit en utilisant un schéma de compression-transmission-estimation séparé dans lequel les observations $(\mathbf{x}_1, \ldots, \mathbf{x}_J)$ sont d'abord compressées de manière distributive en utilisant le codage Berger-Tung [56] en représentations $(\mathbf{u}_1, \ldots, \mathbf{u}_J)$; et, ensuite, les indices bin sont transmis sous forme de messages indépendants sur le réseau $\mathcal{G}$ en utilisant le codage de réseau linéaire [25, Section 15.5]. Le nœud de décision N récupère d'abord les mots de code de représentation $(\mathbf{u}_1, \ldots, \mathbf{u}_J)$; et, ensuite, produit une estimation de l'étiquette $\mathbf{y}$.

Nous discutons ensuite de la manière dont le compromis réalisable obtenu peut être utilisé, nous dérivons ensuite une limite interne variationnelle pour un exemple de réseau à cinq nœuds. Enfin, nous paramétrons la borne obtenue à l'aide d'un réseau de neurones. Nous discutons ensuite de la manière dont le réseau peut être formé et utilisé pour l'inférence de manière distribuée. Encore une fois, nous discutons comment en imposant la condition que le NN à chaque nœud utilise comme entrée un vecteur formé en concaténant les informations reçues des nœuds d'ordre inférieur auxquels

il est connecté, pendant la rétropropagation, le vecteur d'erreur peut être divisé et transmis de chaque nœud aux nœuds d'ordre inférieur correspondants.

Enfin, nous considérons le problème pratique du déploiement d'INL pour une topologie en étoile dans un environnement sans fil contraint; un exemple serait le déploiement dans les réseaux de capteurs sans fil (WSN) [16]. Les performances des WSN sont considérablement limitées par ses ressources radio disponibles et la manière dont les ressources sont allouées entre les nœuds. L'utilisation de techniques d'allocation de ressources développées pour le problème de transmission de données, c'est-à-dire la maximisation de la quantité de données transmises entre les nœuds [20,75], pour les WSN est sous-optimale pour les tâches d'inférence. Intuitivement, cela est dû au fait que les techniques de transmission de données ne prennent pas en considération la relation entre les données transmises et la tâche d'inférence du WSN. Des chercheurs ont proposé des techniques d'allocation de ressources prenant en compte la relation entre les données transmises et la tâche d'inférence pour les problèmes d'estimation distribuée [64–66], et de détection distribuée [61,62,68]. Ces deux problèmes supposent un modèle prédéfini de la distribution entre les données observées et la variable cible à déduire. Pour pallier l'absence de modèle prédéfini de la répartition entre les données observées et la variable cible, le brevet [17] suggérait déjà l'idée d'avoir:

1. chaque appareil mesurant sa propre évaluation locale de la pertinence des données qu'il détient pour la tâche d'inférence en cours et la transmet au nœud de décision,

2. cet appareil envoie également simultanément les valeurs d'activation de sortie au nœud de décision,

3. le nœud de décision combinant toutes les informations reçues (valeurs d'activation des différents appareils ainsi que leurs évaluations locales individuelles) et formant ensuite sa propre évaluation (globale) de la pertinence de l'observation de chaque appareil,

4. application de l'idée à l'ordonnancement des appareils et à l'allocation de puissance (deux algorithmes d'ordonnancement ont déjà été donnés dans [17])

Toutes ces étapes ont ensuite été rapportées et détaillées dans la version intermédiaire [18]. Nous nous différencions de ces travaux en décrivant comment les nœuds locaux peuvent calculer leurs évaluations locales, ainsi qu'en fournissant un moyen approprié pour le nœud de décision de combiner les évaluations locales reçues avec les valeurs d'activation pour obtenir sa propre évaluation globale. Nous proposons également un algorithme d'ordonnancement amélioré, ainsi qu'un algorithme d'allocation de puissance, qui n'a été présenté ni dans [17, 18].

Dans cette thèse, nous proposons de calculer les évaluations locales en utilisant l'information mutuelle (IM) entre les données et la variable cible. Comme le montre [10], il existe un lien étroit entre la perte logarithmique moyenne, que nous utilisons dans notre cadre, et le MI. Alors qu'en général MI ne peut pas être calculé analytiquement, on peut l'estimer à l'aide d'échantillons via des estimateurs basés sur NN tels que MINE [69]. Notre évaluation globale proposée, l'indicateur de pertinence globale (GRI), prend en compte la corrélation entre les représentations compressées que le centre de fusion reçoit, tout en tenant compte des évaluations locales des données observées. Plus précisément, l'indicateur de pertinence globale est formé en combinant

- les bilans locaux, les MI entre les données et la variable cible, obtenus par chaque nœud,
- évaluations de l'utilité relative des données reçues de chacun des nœuds.

Le centre de fusion mesure l'utilité relative des données reçues de chaque nœud à l'aide des informations mutuelles conditionnelles (CMI). Pour calculer l'utilité des données du nœud j, par rapport au nœud i, le centre de fusion a calculé le MI des représentations latentes obtenues du nœud j et de la variable cible, compte tenu des représentations latentes du nœud i. Le CMI est ensuite mis à l'échelle par le MI entre la variable cible et la représentation latente obtenue à partir de tous les nœuds périphériques. L'algorithme de planification cherche à sélectionner les nœuds avec le GRI le plus élevé. En tant que tel, le FC calcule le GRI de chaque nœud, compte tenu des nœuds déjà sélectionnés, puis sélectionne le nœud avec le GRI le plus élevé. Le processus est répété jusqu'à ce que les nœuds soient planifiés. L'algorithme d'allocation de puissance cherche ensuite à trouver un équilibre entre la maximisation des informations totales

que les nœuds sélectionnés transmettent au FC, l'utilité des informations transmises pour déduire la variable cible et la quantité totale d'informations utiles concernant la variable cible que chaque nœud peut transmettre. au FC pendant l'inférence. Pour atteindre ce compromis souhaité, nous introduisons une mesure, notée utilité attendue (EU), qui pour chaque nœud k, combine l'utilité du nœud pour inférer la variable cible sous une allocation de puissance égale, et la quantité d'informations concernant la cible variable qui est contenue dans les données observées mais qui n'a pas pu être transmise, sous allocation de puissance égale. L'utilité du nœud pour déduire la variable cible sous allocation de puissance égale est mesurée par l'évaluation de l'utilité relative du nœud qui a été calculée pour le GRI pendant la phase de programmation. La quantité d'informations concernant la variable cible qui est contenue dans les données observées mais qui n'a pas pu être transmise est mesurée par le CMI entre la variable cible et les données observées compte tenu de la représentation latente transmise, qui est ensuite mise à l'échelle par le MI entre la variable cible et les données observées. La mesure de l'EU est obtenue en prenant le produit entre l'utilité du nœud pour déduire la variable cible sous une allocation de puissance égale, et la quantité d'informations concernant la variable cible qui est contenue dans les données observées mais n'a pas pu être transmise. La puissance est ensuite distribuée proportionnellement à l'EU de chaque nœud.

Mots-clés - *apprentissage distribué, théorie de l'information, classification, Information Bottelneck, perte logarithmique, affectation des ressources.*