entropy

MDPI

Article

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

Matei Moldoveanu [1,2] and Abdellatif Zaidi [1,2,*]

[1] Laboratoire d'Informatique Gaspard-Monge, Université Paris-Est, 77454 Marne-la-Vallée, France
[2] Mathematical and Algorithmic Sciences Lab, Paris Research Center, Huawei Technologies, 92100 Boulogne-Billancourt, France
* Correspondence: abdellatif.zaidi@univ-eiffel.fr
† This paper is an extended version of our paper published in 2021 IEEE Globecom Workshops, Madrid, Spain, 7–11 December 2021.

Abstract: In this paper, we study distributed inference and learning over networks which can be modeled by a directed graph. A subset of the nodes observes different features, which are all relevant/required for the inference task that needs to be performed at some distant end (fusion) node. We develop a learning algorithm and an architecture that can combine the information from the observed distributed features, using the processing units available across the networks. In particular, we employ information-theoretic tools to analyze how inference propagates and fuses across a network. Based on the insights gained from this analysis, we derive a loss function that effectively balances the model's performance with the amount of information transmitted across the network. We study the design criterion of our proposed architecture and its bandwidth requirements. Furthermore, we discuss implementation aspects using neural networks in typical wireless radio access and provide experiments that illustrate benefits over state-of-the-art techniques.

Keywords: distributed learning; AI at the edge; inference over graphs

Citation: Moldoveanu, M.; Zaidi, A. In-Network Learning: Distributed Training and Inference in Networks. *Entropy* **2023**, *25*, 920. https://doi.org/10.3390/e25060920

Academic Editors: Gerhard Bauch and Jan Lewandowsky

Received: 27 April 2023
Revised: 2 June 2023
Accepted: 6 June 2023
Published: 10 June 2023

1. Introduction

The unprecedented success of modern machine learning (ML) techniques in areas such as computer vision [1], neuroscience [2], image processing [3], robotics [4] and natural language processing [5] has led to an increasing interest for their application to wireless communication systems in recent years.

Early efforts along this line of work fall into what is sometimes referred to as the "learning to communicate" paradigm, in which the goal is to automate one or more communication modules such as the modulator-demodulator, the channel coder-decoder, or others, by replacing them with suitable ML algorithms. Although important progress has been made for some particular communication systems, such as the molecular one [6], it is still not yet clear whether ML techniques can offer a reliable alternate solution to model-based approaches, especially as typical wireless environments suffer from time-varying noise and interference.

Wireless networks have other important intrinsic features which may pave the way for more cross-fertilization between ML and communication, as opposed to applying ML algorithms as black boxes in replacement of one or more communication modules. For example, while in areas such as computer vision, neuroscience, and others, relevant data is generally available at one point, it is typically highly distributed across several nodes in wireless networks.

Examples include self-driving cars where multiple sensors, both external and internal to the car can be used to help the car navigate its environment, medical applications to diagnose a patient based on data from different medical institutions or environmental monitoring to detect hazardous events or pollution, and others, see [7,8] for more information. We give

more details of the usefulness of such setups in Examples 1 and 2. A prevalent approach for the implementation of ML solutions in such cases would consist of collecting all relevant data at one point (a cloud server) and then training a suitable ML model using all available data and processing power. Because the volumes of data needed for training are generally large, and with the scarcity of network resources (e.g., power and bandwidth), that approach might not be appropriate in many cases, however. In addition, some applications might have stringent latency requirements which are incompatible with sharing the data, such as in automatic vehicle driving. In other cases, it might be desired not to share the raw data for the sake of enhancing the privacy of the solution, in the sense that infringing the user's privacy is generally more easily accomplished from the raw data itself than from the output of a neural network (NN) that takes the raw data as input.

The above has called for a new paradigm in which intelligence moves from the heart of the network to its edge, which is sometimes referred to as "Edge Learning". In this new paradigm, communication plays a central role in the design of efficient ML algorithms and architectures because both data and computational resources, which are the main ingredients of an efficient ML solution, are highly distributed. A key aspect towards building suitable ML-based solutions is whether the setting assumes only the training phase involves distributed data, sometimes referred to as *distributed learning*, such as the Federated Learning (FL) of [9] or if the inference (or test) phase also involves distributed data.

The considered problem setup is strongly related to the problems of distributed estimation and detection (see, e.g., [10–13] and references therein). We differentiate ourselves from these problems as we assume no prior knowledge of distribution of the data. This is a common setup in many practical applications, such as image or speech processing, or text analysis, where the distribution between the observed data and the target variable is unknown or too complex to model.

In particular, of those most closely related to this paper, a growing line of works focus on developing distributed learning algorithms and architectures. The works of [14,15] address the problem of distributed learning using kernel methods when each node observes independent samples drawn from the same distribution. In our specific setup, however, the nodes observe correlated data, necessitating collaboration among all nodes during inference. On the other hand, works such as [16,17] are focused on the narrower problem of detection and impose certain restrictions on the scope of their investigation. However, perhaps most popular and related to our work is the FL of [9] which, as we already mentioned, is most suitable for scenarios in which the training phase has to be performed distributively, while the inference phase has to be performed centrally at one node. To this end, during the training phase, nodes (e.g., base stations) that possess data are all equipped with copies of a single NN model which they simultaneously train on their locally available data-sets. The learned weight parameters are then sent to a cloud or parameter server (PS) which aggregates them, e.g., by simply computing their average. The process is repeated, every time re-initializing 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 data-set. For recent advances on FL and applications in wireless settings, the reader may refer to [18–20] and references therein. Another relevant work is the Split Learning (SL) of [21] in which, for a multiaccess type network topology, a two-part NN model, split into an encoder part and a decoder part, is learned sequentially. The decoder does not have its own data and in every round the NN encoder part is fed with a distinct data-set and its parameters are initialized using those learned from the previous round. The learned two-part model is then used as follows during the inference: one part of this model is used by an encoder, and the other one by a decoder. Another variation of SL, sometimes called "vertical SL", was proposed recently in [22]. The approach uses vertical partitioning of the data; in the special case of a multi-access topology, it is similar to the in-network learning solution that we propose in this paper.

Compared to both SL and FL, which consider only the training phase to be distributed, in this paper we focus on the problem in which the inference phase also takes place

distributively. More specifically, in this paper, we study a network inference problem in which some of the nodes possess each, or can acquire, part of the data that is relevant for inference on a random variable Y. The node at which the inference needs to be performed is connected to the nodes that possess the relevant data through a number of intermediate other nodes. We assume that the network topology is fixed and known. This may model, e.g., a setting in which a macro BS needs to make inference on the position of a user on the basis of summary information obtained from correlated CSI measurements $X_1, \ldots, X_J$ that are acquired at some proximity edge BSs. Each of the edge nodes is connected with the central node either directly, via an error free link of given finite capacity, or via intermediary nodes. While in some cases it might be enough to process only a subset of the J nodes, we assume that processing only a (any) strict subset of the measurements cannot yield the desired inference accuracy and, as such, the J measurements $X_1, \ldots, X_J$ need to be processed during the inference or test phase.

Example 1. *(Autonomous Driving) One basic requirement of the problem of autonomous driving is the ability to cope with problematic roadway situations, such as those involving construction, road hazards, hand signals, and reckless drivers. Current approaches mainly depend on equipping the vehicle with more on-board sensors. Clearly, while this can only allow a better coverage of the navigation environment, it seems unlikely to successfully cope with the problem of blind spots due, e.g., to obstruction or hidden obstacles. In such contexts, external sensors such as other vehicles' sensors, cameras installed on the roofs of proximity buildings or wireless towers may help perform a more precise inference, by offering a complementary, possibly better, view of the navigation scene. An example scenario is shown in Figure 1. The application requires real-time inference which might be incompatible with current cellular radio standards, thus precluding the option of sharing the sensors' raw data and processing it locally, e.g., at some on-board server. When equipped with suitable intelligence capabilities, each sensor can successfully identify and extract those features of its measurement data that are not captured by other sensors' data. Then, it only needs to communicate those, not its entire data.*

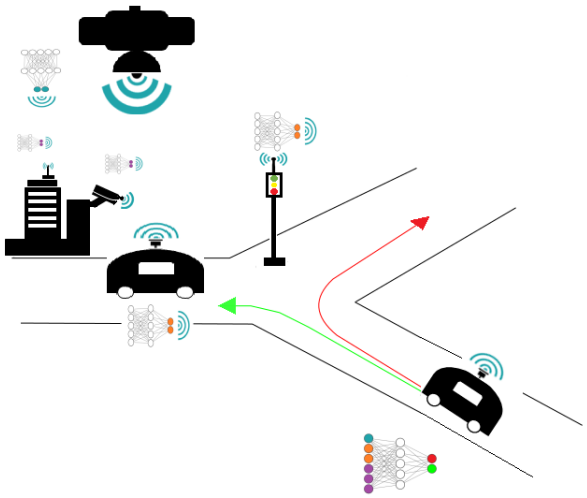


Figure 1. Fusion of inference from on-board and external sensors for automatic vehicle navigation.

Example 2. *(Public Health) One of the early applications of machine learning is in the area of medical imaging and public health. In this context, various institutions can hold different modalities of patient data in the form of electronic health records, pathology test results, radiology, and other sensitive imaging data such as genetic markers for disease. The correct diagnosis may be contingent on being able to using all relevant data from all institutions. However, these institutions may not be authorized to share their raw data. Thus, it is desired to distributively train machine learning models without sharing the patient's raw data in order to prevent illegal, unethical or unauthorized usage of it [23]. Local hospitals or tele-health screening centers seldom acquire enough diagnostic*

images on their own; collaborative distributed learning in this setting would enable each individual center to contribute data to an aggregate model without sharing any raw data.

1.1. Contributions

In this paper, we study the aforementioned network inference problem in which the network is modeled as a weighted acyclic graph and inference about a random variable is performed on the basis of summary information obtained from possibly correlated variables at a subset of the nodes. Following an information-theoretic approach in which we measure discrepancies between true values and their estimated fits using average logarithmic loss, we first develop a bound on the best achievable accuracy given the network communication constraints. Then, considering a supervised setting in which nodes are equipped with NNs and their mappings need to be learned from distributively available training data-sets, we propose a distributed learning and inference architecture and we show that it can be optimized using a distributed version of the well-known stochastic gradient descent (SGD) algorithm that we develop here. The resulting distributed architecture and algorithm, which we herein name "in-network (INL) learning", generalize those introduced in [24] (see also [25,26]) for a specific case, multiaccess type, network topology. We investigate in more detail what the various nodes need to exchange during both the training and inference phases, as well as associated requirements in bandwidth. Finally, we provide a comparative study with (an adaptation of) the FL and the SL algorithms, and experiments that illustrate our results. Part of the results this paper have also been presented in [27,28]. However, in this paper, we go beyond those works by offering a more comprehensive and detailed review of the state-of-the-art. Additionally, we provide proofs for the theorem and lemmas presented in this paper, which were not included in the previous publications. Furthermore, we introduce additional insights and conclusions that further contribute to the overall understanding and significance of the research findings.

1.2. Outline and Notation

In Section 2 we describe the studied network inference problem formally. In Section 3 we present our in-network inference architecture, as well as a distributed algorithm for training it distributively. Section 4 contains a comparative study with FL and SL in terms of bandwidth requirements; as well as some experimental results. Finally, in Section 5 we summarize the insights and results presented in this paper.

Throughout the paper, the following notation will be used. Upper case letters denote random variables, e.g., X; lower case letters denote realizations 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 random variable X with probability mass function 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 $\mathbf{x}$. For random variables $(X_1, X_2, \ldots)$ 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 $\mathcal{K}$, i.e., $X_{\mathcal{K}} \triangleq \{X_k : k \in \mathcal{K}\}$. If $\mathcal{K} = \emptyset$ then $X_{\mathcal{K}} = \emptyset$. In addition, for zero-mean random vectors $\mathbf{x}$ 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 $\mathbf{y}$. 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 $D_{KL}(P_X \| Q_X) = \infty$.

2. Network Inference: Problem Formulation

We consider the distributed supervised learning setup, in which multiple nodes observe different features relating to the same sample, sometimes refered to as distributed learning with vertically partitioned dataset, see [8,29]. We additionally assume the learning takes place over a communication constrained network. Specifically, consider an N node distributed network. Of these N nodes, $J \geq 1$ nodes possess or can acquire data that is relevant for inference on a random variable (r.v.) of interest Y, with alphabet $\mathcal{Y}$. Let

$\mathcal{J} = \{1, \ldots, J\}$ denote the set of such nodes, with node $j \in \mathcal{J}$ observing samples from the random variable X_j, with alphabet $\mathcal{X}_j$. The relationship between the r.v. 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 $y \in \mathcal{Y}$. For simplicity, we assume that random variables are discreet, however our technique can be applied to continuous variables as well. Inference on Y needs to be performed at some node N which is connected to the nodes that possess the relevant data through a number of intermediate other nodes. It has to be performed without any sharing of raw data. The network is modeled as a weighted directed acyclic graph and may represent, for example, a wired network or a wireless mesh network operated in time or frequency division, where the nodes may be servers, handsets, sensors, base stations or routers. We assume that the network graph is fixed and known. The edges in the graph represent point-to-point communication links that use channel coding to achieve close to error-free communication at rates below their respective capacities. For a given loss function $\ell(\cdot, \cdot)$ that measures discrepancies between true values of Y and their estimated fits, what is the best precision for the estimation of Y? Clearly, discarding any of the relevant data X_j can only lead to a reduced precision. Thus, intuitively features that collectively maximize information about Y need to be extracted distributively by the nodes from the set $\mathcal{J}$, without explicit coordination between them and they then need to propagate and combine appropriately at the node N. How should that be performed optimally without sharing raw data? In particular, how should each node process information from the incoming edges (if any) and what should it transmit on every one of its outgoing edges? Furthermore, how should the information be fused optimally at Node N?

More formally, we model an N-node network by a directed acyclic graph $\mathcal{G} = (\mathcal{N}, \mathcal{E}, C)$, 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}. See Figure 2. 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

$$\omega_j : \mathcal{X}_j \times \left\{ \Pi_{i\,:\,(i,j)\,\in\,\mathcal{E}} \mathcal{M}_{ij} \right\} \longrightarrow \Pi_{l\,:\,(j,l)\,\in\,\mathcal{E}} \mathcal{M}_{jl}, \tag{1}$$

where Π 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,

$$\omega_k : \Pi_{i\,:\,(i,k)\,\in\,\mathcal{E}} \mathcal{M}_{ik} \longrightarrow \Pi_{l\,:\,(k,l)\,\in\,\mathcal{E}} \mathcal{M}_{kl}. \tag{2}$$

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

$$\log |\mathcal{M}_{ij}| \leq C_{ij} \quad \forall i \in [1, N-1] \quad \text{and} \quad \forall j : (i,j) \in \mathcal{E}. \tag{3}$$

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

$$\psi : \Pi_{i\,:\,(i,N)\,\in\,\mathcal{E}} \mathcal{M}_{iN} \longrightarrow \hat{\mathcal{Y}}. \tag{4}$$

In this paper, 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 discrepancies between true values of $Y \in \mathcal{Y}$ and their estimated fits in terms of average logarithmic loss, i.e., for $(y, \hat{P}) \in \mathcal{Y} \times \mathcal{P}(\mathcal{Y})$

$$d(y, \hat{P}) = \log \frac{1}{\hat{P}(y)}. \tag{5}$$

As such, the performance of a distributed inference scheme $\left((\omega_j)_{j\in\mathcal{J}}, (\omega_k)_{k\in[1,N-1]/\mathcal{J}}, \psi\right)$ for which (3) is fulfilled is given by its achievable *relevance* given by

$$\Delta = H(Y) - \mathbb{E}\Big[d(Y,\hat{Y})\Big], \tag{6}$$

which, for a discrete set $\mathcal{Y}$, is directly related to the error of misclassifying the variable $Y \in \mathcal{Y}$. It is imporant to note that $H(Y)$ is problem specific constant and as such the relavance given by (6) is simply a another form of the logarithmic loss.

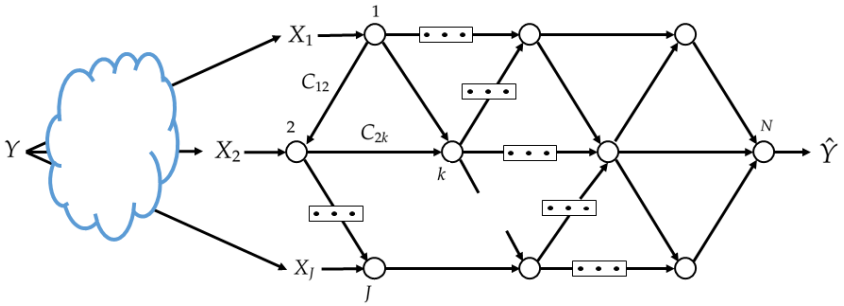


Figure 2. Studied network inference model.

In practice, in a supervised setting, the mappings given by (1), (2) and (4) need to be learned from a set of training data samples $\{(x_{1,i},\ldots,x_{J,i},y_i)\}_{i=1}^{n}$. The data is 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 parametrize the possibly stochastic mappings (1), (2) and (4) using NNs. This is depicted in Figure 3. 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 parameterizes 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 (3). Given that the actual distribution is unknown and we only have access to a dataset, the loss function needs to strike a balance between its performance on the dataset, given by 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 $x_j \in \mathcal{X}_j$, under the assumtion that all elements of $\mathcal{X}_j$ have the same dimension, it holds that

$$\begin{aligned}&\text{Size of first layer of NN } (j) = \\ &\text{Dimension } (x_j) + \sum_{i\,:\,(i,j)\in\mathcal{E}} (\text{Size of last layer of NN } (i)).\end{aligned} \tag{7}$$

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

$$\begin{aligned}&\text{Size of first layer of NN } (k) = \\ &\sum_{i\,:\,(i,k)\in\mathcal{E}} (\text{Size of last layer of NN } (i)).\end{aligned} \tag{8}$$

Remark 1. *Conditions* (7) *and* (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* (7) *and* (8) *to hold. Alternative aggregation techniques, such as element-wise multiplication or element-wise averaging, can be*

employed to combine the information received by each node, in replacement to concatenation. The impact of these aggregation techniques has been analyzed in [22].

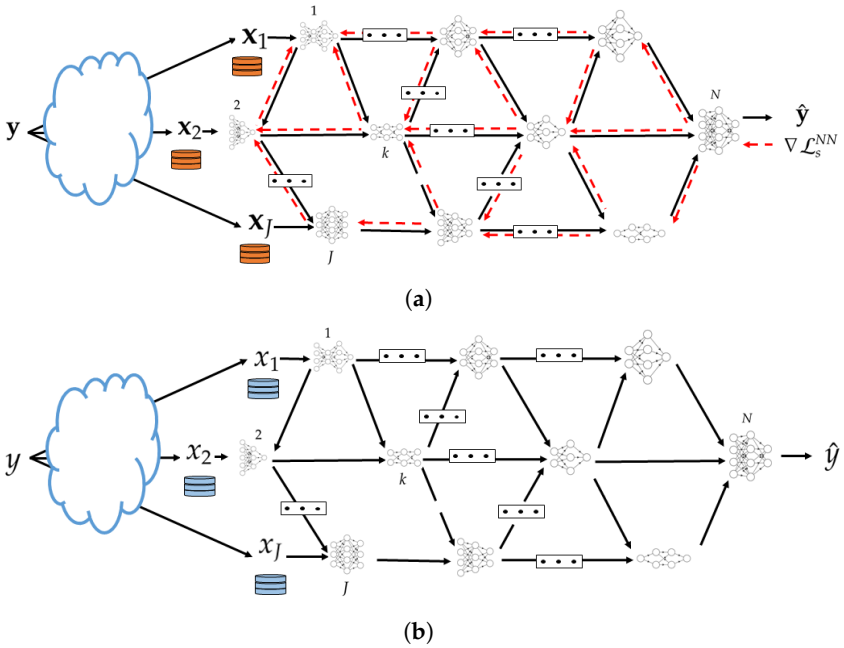


Figure 3. In-network learning and inference using neural networks. (**a**) Training phase. (**b**) Inference phase.

3. Proposed Solution: In-Network Learning and Inference

For convenience, we first consider a specific setting of the model of network inference problem of Figure 3 in which $J = N - 1$ and all the nodes that observe data are only connected to the end decision node, but not among them.

3.1. A Specific Model: Fusing of Inference

In this case, a possible suitable loss function was shown by [25] to be:

$$\mathcal{L}_s^{\mathrm{NN}}(n) = \frac{1}{n}\sum_{i=1}^{n} \log Q_{\phi_{\mathcal{J}}}(y_i|u_{1,i},\ldots,u_{J,i}) + \frac{s}{n}\sum_{i=1}^{n}\sum_{j=1}^{J}\left(\log Q_{\phi_j}(y_i|u_{j,i}) - \log\left(\frac{P_{\theta_j}(u_{j,i}|x_{j,i})}{Q_{\varphi_j}(u_{j,i})}\right)\right), \tag{9}$$

where s is a Lagrange parameter and for $j \in \mathcal{J}$ the distributions $P_{\boldsymbol{\theta}_j}(u_j|x_j)$, $Q_{\boldsymbol{\phi}_j}(y|u_j)$, $Q_{\boldsymbol{\phi}_{\mathcal{J}}}(y|u_{\mathcal{J}})$ are variational ones whose parameters are determined by the chosen NNs using the re-parametrization trick of [30] and $Q_{\boldsymbol{\varphi}_j}(u_j)$ are priors known to the encoders. For example, denoting by $f_{\boldsymbol{\theta}_j}$ 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 $P_{\boldsymbol{\theta}_j}(u_j|x_j)$ can be chosen to be multivariate Gaussian, i.e., $P_{\boldsymbol{\theta}_j}(u_j|x_j) = \mathcal{N}(u_j; \boldsymbol{\mu}_j^{\theta}, \boldsymbol{\Sigma}_j^{\theta})$, where $\boldsymbol{\mu}_j^{\theta}, \boldsymbol{\Sigma}_j^{\theta}$ are outputs of $f_{\boldsymbol{\theta}_j}(x_j)$. For discrete data, concrete variables (i.e., Gumbel-Softmax) can be used instead.

The rationale behind the choice of loss function (9) is that in the regime of large n, if the encoders and decoder are not restricted to use NNs under some conditions. The optimality is proved therein under the assumption that for every subset $\mathcal{S} \subseteq \mathcal{J}$, it holds that $X_{\mathcal{S}} \multimap Y \multimap X_{\mathcal{S}^c}$. The RHS of (10) is achievable for arbitrary distributions, however, regardless of such an assumption; the optimal stochastic mappings $P_{U_j|X_j}$, P_U, $P_{Y|U_j}$ and

$P_{Y|U_{\mathcal{J}}}$ are found by marginalizing the joint distribution that maximizes the following Lagrange cost function [25] (Proposition 2)

$$\mathcal{L}_s^{\text{optimal}} = -H(Y|U_{\mathcal{J}}) - s\sum_{j=1}^{J}\Big[H(Y|U_j) + I(U_j;X_j)\Big]. \tag{10}$$

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

3.1.1. Inference Phase

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 $(u_1, \ldots, u_J)$ 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 4b.

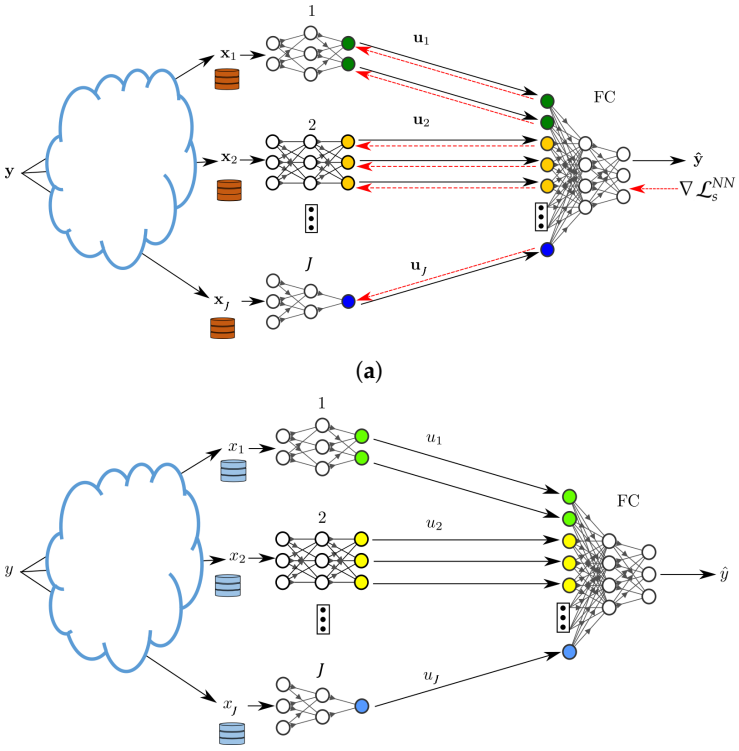


Figure 4. In-network learning for the network model for the case without hops. (**a**) Training phase. (**b**) Inference phase.

Remark 2. *One can combine our proposed technique with an appropriate transmission scheme and channel coding. One possible suitable practical implementation in wireless settings can be obtained using Orthogonal Frequency-Division Multiple Access (OFDMA). 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 performed, e.g., using entropy type coding [31].*

3.1.2. Training Phase

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, $\mathbf{u}_j$, whose elements are the activation values of the last layer of (NN j), see Figure 4a. Due to (8) the activation vectors are concatenated vertically at the input layer of NN $(J+1)$. 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 backpropagation. Specifically, let L_{J+1} denote the index of the last layer of NN $(J+1)$. Additionally, let $\mathbf{w}_{J+1}^{[l]}$, $\mathbf{b}_{J+1}^{[l]}$ and $\mathbf{a}_{J+1}^{[l]}$ denote the weights, biases and activation values at layer $l \in [2 : L_{J+1}]$ for the NN $(J+1)$ and σ is the activation function, respectively. Node $(J+1)$ computes the error vectors

$$\boldsymbol{\delta}_{J+1}^{[L_{J+1}]} = \nabla_{\mathbf{a}_{J+1}^{[L_{J+1}]}} \mathcal{L}_s^{NN}(b) \odot \sigma'(\mathbf{w}_{J+1}^{[L_{J+1}]} \mathbf{a}_{J+1}^{[L_{(J+1)}-1]} + \mathbf{b}_{J+1}^{[L_{J+1}]}) \tag{11a}$$

$$\boldsymbol{\delta}_{J+1}^{[l]} = [(\mathbf{w}_{J+1}^{[l+1]})^T \boldsymbol{\delta}_{J+1}^{[l+1]}] \odot \sigma'(\mathbf{w}_{J+1}^{[l]} \mathbf{a}_{J+1}^{[l-1]} + \mathbf{b}_{J+1}^{[l]}) \ \ \forall \ l \in [2, L_{J+1}-1], \tag{11b}$$

$$\boldsymbol{\delta}_{J+1}^{[1]} = [(\mathbf{w}_{J+1}^{[2]})^T \boldsymbol{\delta}_{J+1}^{[2]}] \tag{11c}$$

and then updates its weight- and bias parameters as

$$\mathbf{w}_{J+1}^{[l]} \to \mathbf{w}_{J+1}^{[l]} - \eta \boldsymbol{\delta}_{J+1}^{[l]} (\mathbf{a}_{J+1}^{[l-1]})^T, \tag{12a}$$

$$\mathbf{b}_{J+1}^{[l]} \to \mathbf{b}_{J+1}^{[l]} - \eta \boldsymbol{\delta}_{J+1}^{[l]}, \tag{12b}$$

where η designates the learning parameter; for simplicity, η and σ are assumed here to be identical for all NNs.

Remark 3. *It is important to note that for the computation of the RHS of* (11a) *node* $(J+1)$*, which knows* $Q_{\phi_{\mathcal{J}}}(y_i|u_{1,i}, \ldots, u_{J,i})$ *and* $Q_{\phi_j}(y_i|u_{j,i})$ *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* $P_{\boldsymbol{\theta}_j}(u_j|x_j)$ *or the priors* $Q_{\boldsymbol{\varphi}_j}(u_j)$*.*

The backward propagation of the error vector from node $(J+1)$ to the nodes j, $j \in \{1, \ldots, J\}$, is as follows. Node $(J+1)$ horizontally splits the error vector of its input layer into J sub-vectors with sub-error vector j having the same size as the dimension of the last layer of NN j [recall (8) and that the activation vectors are concatenated vertically during the forward pass]. See Figure 4a. The backward propagation then continues on each of the J input NNs simultaneously, each of them essentially applying operations similar to (11) and (12).

Remark 4. *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* $j \in \mathcal{J}$*,*

$$\nabla_{\mathbf{a}_j^{L_j}} \mathcal{L}_s^{NN}(b_j) = \boldsymbol{\delta}_{J+1}^{[1]}(j) - s \nabla_{\mathbf{a}_j^{L_j}} \left(\sum_{i=1}^{b} \log \left(\frac{P_{\boldsymbol{\theta}_j}(u_{j,i}|x_{j,i})}{Q_{\boldsymbol{\varphi}_j}(u_{j,i})} \right) \right); \tag{13}$$

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. General Model: Fusion and Propagation of Inference

Consider now the general network inference model of Figure 2. Part of the difficulty of this problem is in finding a suitable loss function which can be optimized distributively via NNs that only have access to local data-sets each. The next theorem provides a bound on the achievable relevance (under some assumptions) for an arbitrary network topology $(\mathcal{E}, \mathcal{N})$. The result of Theorem 1 is asymptotic in the size of the training data-sets, while the inference problem is a one-shot problem. One-shot results for this problem can be

obtained, e.g., along the approach of [32]. For convenience, we define for $\mathcal{S} \subseteq [1, \ldots, N-1]$ and non-negative $(C_{ij} \ : \ (i,j) \in \mathcal{E})$ the quantity

$$C(\mathcal{S}) = \sum_{(i,j) \, : \, i \in \mathcal{S}, j \in \mathcal{S}^c} C_{ij}. \tag{14}$$

Theorem 1. *For the network inference model of Figure 2, in the regime of large data-sets the following relevance is achievable,*

$$\Delta = \max I(U_1, \ldots, U_J; Y) \tag{15}$$

where the maximization is over joint measures of the form

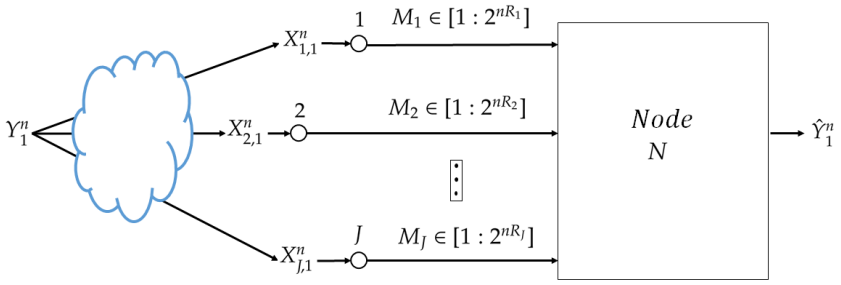
$$P_Q P_{X_1, \ldots, X_J, Y} \prod_{j=1}^{J} P_{U_j | X_j, Q} \tag{16}$$

for which there exist non-negative $R_1, \ldots, R_J$ that satisfy

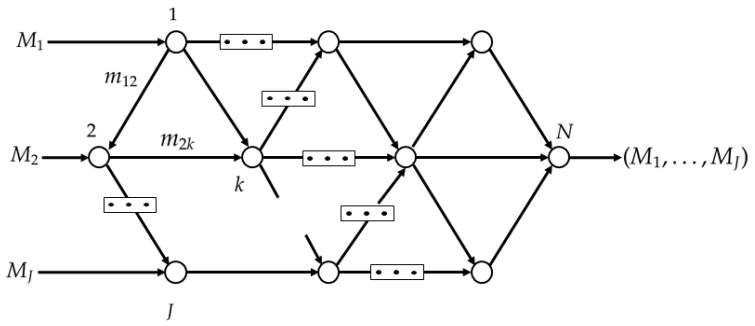
$$\sum_{j \in \mathcal{S}} R_j \geq I(U_{\mathcal{S}}; X_{\mathcal{S}} | U_{\mathcal{S}^c}, Q), \quad \textit{for all} \quad \mathcal{S} \subseteq \mathcal{J}$$

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

Proof. The proof of Theorem 1 appears in Appendix A. 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 [33] 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 [34] (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 5. □



(**a**)



(**b**)

Figure 5. Block diagram of the separate compression-transmission-estimation scheme of Theorem 1. (**a**) Compression using Berger-Tung coding. (**b**) Transmission of the bin indices using linear coding.

Part of the utility of the loss function of Theorem 1 is in that it accounts explicitly for the network topology for inference fusion and propagation. In addition, although as seen from its proof the setting of Theorem 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 date-sets. To do so, we first derive a Lagrangian function, from Theorem 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 suing NNs. For a given network topology in essence, the approach generalizes that of Section 3.1 to more general networks that involve hops. For simplicity, in what follows, this is illustrated for the example architecture of Figure 6. 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. Firstly, we leverage Theorem 1 to establish a feasible trade-off between the performance of the network illustrated in Figure 6, quantified by its *relevance*, and the quantity of information that must be communicated between the nodes. Subsequently, employing the aforementioned approach, we derive a loss function tailored for the scenarios where the nodes are equipped with neural networks, as depicted in Figure 7.

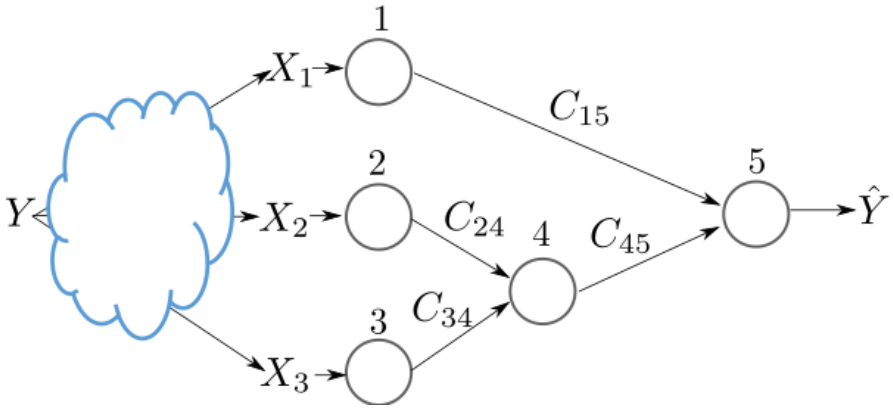


Figure 6. An example in-network learning with inference fusion and propogation.

Setting $\mathcal{N} = \{1, 2, 3, 4, 5\}$ and $\mathcal{E} = \{(3, 4), (2, 4), (4, 5), (1, 5)\}$ in Theorem 1, we obtain that

$$\Delta = \max I(U_1, U_2, U_3; Y) \tag{17}$$

where the maximization is over joint measures of the form

$$P_Q P_{X_1, X_2, X_3, Y} P_{U_1|X_1, Q} P_{U_2|X_2, Q} P_{U_3|X_3, Q} \tag{18}$$

for which the following holds for some $R_1 \geq 0$, $R_2 \geq 2$ and $R_3 \geq 0$:

$$C_{15} \geq R_1,\ C_{24} \geq R_2,\ C_{34} \geq R_3,\ C_{45} \geq R_2 + R_3 \tag{19a}$$
$$R_1 \geq I(U_1; X_1 | U_2, U_3, Q), \tag{19b}$$
$$R_2 \geq I(U_2; X_2 | U_1, U_3, Q), \tag{19c}$$
$$R_3 \geq I(U_3; X_3 | U_1, U_2, Q) \tag{19d}$$
$$R_3 + R_2 \geq I(X_2, X_3; U_2, U_3 | U_1, Q), \tag{19e}$$
$$R_3 + R_1 \geq I(X_1, X_3; U_1, U_3 | U_2, Q) \tag{19f}$$
$$R_2 + R_1 \geq I(X_1, X_2; U_1, U_2 | U_3, Q), \tag{19g}$$
$$R_2 + R_1 + R_3 \geq I(X_1, X_2, X_3; U_1, U_2, U_3 | Q). \tag{19h}$$

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 the RHS of (17) 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 (17) and (19), we obtain that the region $\mathcal{RI}_{\text{sum}}$ is given by the union of pairs $(\Delta, C_{\text{sum}}) \in \mathbb{R}_+^2$ for which (the time sharing random variable is set to a constant for simplicity)

$$\Delta \leq I(Y; U_1, U_2, U_3) \tag{20a}$$

$$C_{\text{sum}} \geq I(X_1, X_2, X_3; U_1, U_2, U_3) + I(X_2, X_3; U_2, U_3 | U_1) \tag{20b}$$

for some measure of the form

$$P_Y P_{X_1, X_2, X_3|Y} P_{U_1|X_1} P_{U_2|X_2} P_{U_3|X_3}. \tag{21}$$

The next proposition gives a useful parameterization of the region $\mathcal{RI}_{\text{sum}}$ as described by (20) and (21).

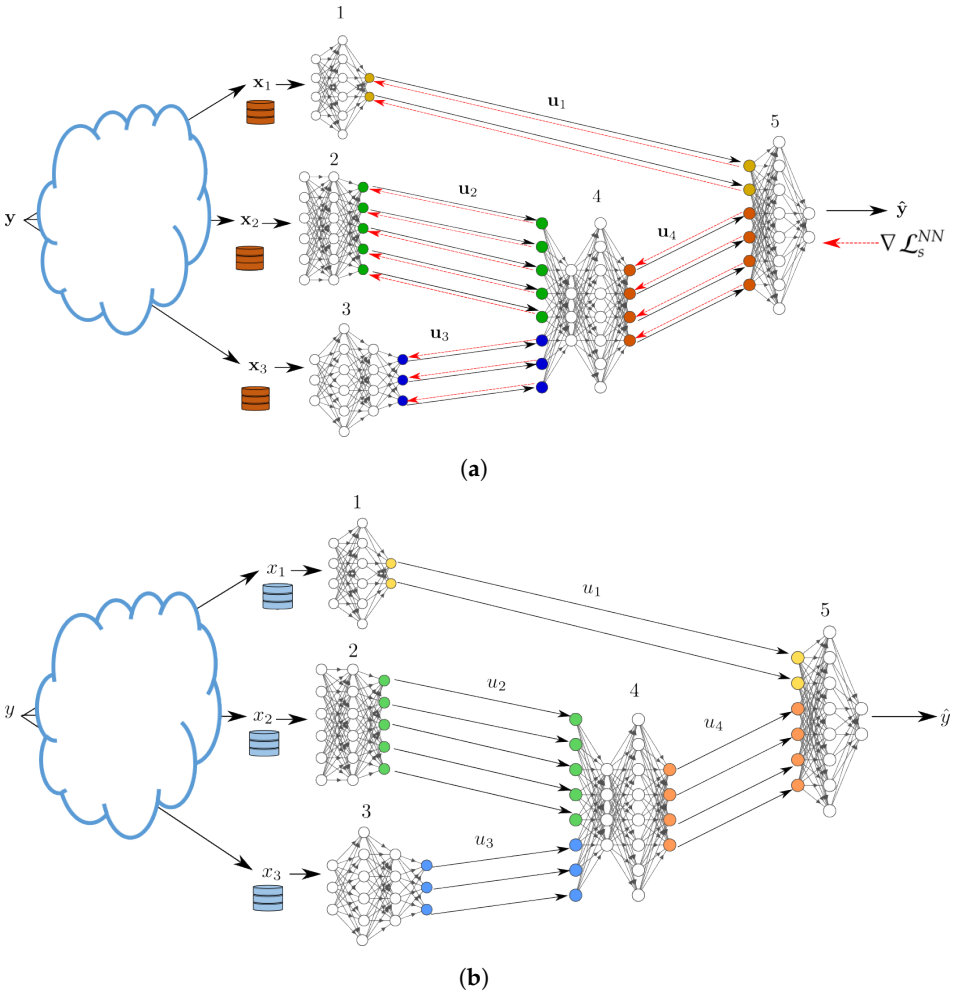


Figure 7. Forward and backward passes for the inference problem of Figure 6. (**a**) Training phase. (**b**) Inference phase.

Proposition 1. *For every pair* (Δ, C_{sum}) *that lies on the boundary of the region described by* (20) *and* (21) *there exists* $s \geq 0$ *such that* $(\Delta, C_{sum}) = (\Delta_s, C_s)$*, with*

$$\Delta_s = H(Y) + \max_{\mathbf{P}} \mathcal{L}_s(\mathbf{P}) + sC_s \tag{22a}$$

$$C_s = I(X_1, X_2, X_3; U_1^*, U_2^*, U_3^*) + I(X_2, X_3; U_2^*, U_3^* | U_1^*), \tag{22b}$$

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 maximize the cost function*

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

$$- sI(X_2, X_3; U_2, U_3|U_1). \tag{23}$$

Proof. See Appendix B. □

In accordance with the studied example network inference problem shown in Figure 6, let a random variable U_4 be such that $U_4 \multimap (U_2, U_3) \multimap (X_1, X_2, X_3, Y, U_1)$. That is, the joint distribution factorizes as

$$P_{X_1,X_2,X_3,Y,U_1,U_2,U_3,U_4} = P_{X_1,X_2,X_3,Y} P_{U_1|X_1} P_{U_2|X_2} P_{U_3|X_3} P_{U_4|U_2,U_3}. \tag{24}$$

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

$$\begin{aligned} \mathcal{L}_s^{\text{low}}(\mathbf{P}, P_{U_4|U_2,U_3}) = &- H(Y|U_1, U_4) - sI(X_1; U_1) - 2sI(X_2; U_2) \\ &- 2s\Big[I(X_3; U_3) - I(U_2; U_1) - I(U_3; U_1, U_2)\Big]. \end{aligned} \tag{25}$$

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 (23).

Lemma 1. *For every $s \geq 0$ and joint measure that factorizes as (24), we have*

$$\mathcal{L}_s(\mathbf{P}) \geq \mathcal{L}_s^{low}(\mathbf{P}, P_{U_4|U_2,U_3}), \tag{26}$$

Proof. See Appendix C. □

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 optimization of (25) 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

$$\mathbf{Q} := \{Q_{Y|U_1,U_4}, Q_{U_3}, Q_{U_2}, Q_{U_1}\}, \tag{27}$$

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. Additionally, let

$$\begin{aligned} \mathcal{L}_s^{\text{v-low}}(\mathbf{P}_+, \mathbf{Q}) := &\mathbb{E}[\log Q_{Y|U_1,U_4}(Y|U_1, U_4)] - sD_{\text{KL}}(P_{U_1|X_1}\|Q_{U_1}) \\ &- 2sD_{\text{KL}}(P_{U_2|X_2}\|Q_{U_2}) - 2sD_{\text{KL}}(P_{U_3|X_3}\|Q_{U_3}). \end{aligned} \tag{28}$$

The following lemma, the proof of which is essentially similar to that of [25] (Lemma 1), shows that for every $s \geq 0$, the cost function $\mathcal{L}_s^{\text{low}}(\mathbf{P}, P_{U_4|U_2,U_3})$ is lower-bounded by $\mathcal{L}_s^{\text{v-low}}(\mathbf{P}_+, \mathbf{Q})$ as given by (28).

Lemma 2. *For fixed $\mathbf{P}_+$, we have*

$$\mathcal{L}_s^{low}(\mathbf{P}_+) \geq \mathcal{L}_s^{v\text{-}low}(\mathbf{P}_+, \mathbf{Q}) \tag{29}$$

for all pmfs $\mathbf{Q}$, with equality when:

$$Q_{Y|U_1,U_4} = P_{Y|U_1,U_4}, \tag{30}$$

$$Q_{U_3} = P_{U_3|U_2,U_1}, \tag{31}$$

$$Q_{U_2} = P_{U_2|U_1}, \tag{32}$$

$$Q_{U_1} = P_{U_1}, \tag{33}$$

where $P_{Y|U_1,U_4}$, $P_{U_3|U_2,U_1}$, $P_{U_2|U_1}$, P_{U_1} are calculated using (24).

Proof. See Appendix D. □

From the above, we get that

$$\max_{\mathbf{P}_+} \mathcal{L}_s^{\text{low}}(\mathbf{P}_+) = \max_{\mathbf{P}_+} \max_{\mathbf{Q}} \mathcal{L}_s^{\text{v-low}}(\mathbf{P}_+, \mathbf{Q}). \tag{34}$$

Since, as described in Section 2, 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 optimization of (28) to the family of distributions that can be parameterized by NNs. Thus, we obtain the following loss function which can be optimized empirically, in a distributed manner, using gradient based techniques,

$$\begin{aligned}\mathcal{L}_s^{\text{NN}}(n) := \frac{1}{n}\sum_{i=1}^{n}\left[\log Q_{\phi_5}(y_i|u_{1,i}, u_{4,i}) - s\log\left(\frac{P_{\theta_1}(u_{1,i}|x_{1,i})}{Q_{\varphi_1}(u_{1,i})}\right)\right] \\ - \frac{2s}{n}\sum_{i=1}^{n}\left[\log\left(\frac{P_{\theta_2}(u_{2,i}|x_{2,i})}{Q_{\varphi_2}(u_{2,i})}\right) + \log\left(\frac{P_{\theta_3}(u_{3,i}|x_{3,i})}{Q_{\varphi_3}(u_{3,i})}\right)\right],\end{aligned} \tag{35}$$

with 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-parametrization trick of [30] and $\{Q_{\varphi_i} : i \in \{1,2,3\}\}$ are priors known to the encoders. The parameterization of the distributions with NNs is performed similarly to that for the setting of Section 3.1.

3.2.1. Inference Phase

During this phase, nodes 1, 2 and 3 each observe (or measure) a new sample. Let x_1 be the sample observed by node 1 and x_2 and x_3 those observed by node 2 and node 3, respectively. Node 1 processes x_1 using its NN and sends an encoded value u_1 to node 5 and so do nodes 2 and 3 towards node 4. Upon receiving u_2 and u_3 from nodes 2 and 3, node 4 concatenates them vertically and processes the obtained vector using its NN. The output u_4 is then sent to node 5. The latter performs similar operations on the activation values u_1 and u_4 and outputs an estimate of the label y in the form of a soft output $Q_{\phi_5}(y|u_1, u_4)$.

3.2.2. Training Phase

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 vector formed of the activation values of the last layer of their NNs to node 4. Because the sizes of the last layers of the NNs of nodes 2 and 3 are chosen according to (8) the sent activation 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 (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.

During the backward pass, each of the NNs updates its parameters according to (11) and (12). Node 5 is the first to apply the back propagation procedure in order update the parameters of its NN. It applies (11) and (12) sequentially, starting from its last layer.

Remark 5. *It is important to note that, similar to the setting of Section III-A, for the computation of the RHS of* (11a) *for node* 5*, only the derivative of* $\mathcal{L}_s^{NN}(n)$ *w.r.t. the activation vector* $\mathbf{a}_5^{L_5}$ *is required, which depends only on* $Q_{\phi_5}(y_i|u_{1,i}, u_{4,i})$*. The distributions are known to node 5 given only* $u_{1,i}$ *and* $u_{4,i}$*.*

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 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 7a. 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 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 entire process continues until convergence.

Remark 6. *Let $\boldsymbol{\delta}_j^{[1]}(j)$ denote the sub-error vector sent back from node J to node j. It is easy to see that, for every $j \in \mathcal{J}$,*

$$\begin{aligned}
\nabla_{\mathbf{a}_4^{[L]}} \mathcal{L}_s^{NN}(b) &= \boldsymbol{\delta}_5^{[1]}(4), \\
\nabla_{\mathbf{a}_3^{[L]}} \mathcal{L}_s^{NN}(b) &= \boldsymbol{\delta}_4^{[1]}(3) - 2s\nabla_{\mathbf{a}_3^{[L]}} \left[\frac{1}{b} \sum_{i=1}^{b} \left[\log\left(\frac{P_{\theta_3}(u_{3,i}|x_{3,i})}{Q_{\varphi_3}(u_{3,i})} \right) \right] \right], \\
\nabla_{\mathbf{a}_2^{[L]}} \mathcal{L}_s^{NN}(b) &= \boldsymbol{\delta}_4^{[1]}(2) - 2s\nabla_{\mathbf{a}_2^{[L]}} \left[\frac{1}{b} \sum_{i=1}^{b} \left[\log\left(\frac{P_{\theta_2}(u_{2,i}|x_{2,i})}{Q_{\varphi_2}(u_{2,i})} \right) \right] \right], \\
\nabla_{\mathbf{a}_1^{[L]}} \mathcal{L}_s^{NN}(b) &= \boldsymbol{\delta}_5^{[1]}(1) - s\nabla_{\mathbf{a}_1^{[L]}} \left[\frac{1}{b} \sum_{i=1}^{b} \left[\log\left(\frac{P_{\theta_1}(u_{1,i}|x_{1,i})}{Q_{\varphi_1}(u_{1,i})} \right) \right] \right].
\end{aligned}$$

and this explains why, for back propagation, nodes $1, 2, 3, 4$ need only part of the error vector at the node they are connected to.

3.3. Bandwidth Requirements

In this section, we study the bandwidth requirements of our in-network learning. 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 (8), 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 a NN at one node. For the SL of [21], 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 summarized and compared in Table 1 for two popular NNs architectures, 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 = 25,088 and $\eta = 0.88$ for ResNet50 and 0.11 for VGG16.

Compared to FL and SL, INL has an advantage in that all nodes work jointly also during inference to make a prediction,not just during the training phase. As a consequence nodes only need to exchange latent representations, not model parameters, during training.

Table 1. Comparison of bandwidth requirements.

	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

4. Experimental Results

We perform two series of experiments for which we compare the performance of our INL with those of FL and SL. The dataset used is the CIFAR-10 and there are five client nodes. In the first experiment, the three techniques are implemented in such a way such that during the inference phase the same NN is used to make the predictions. In the second experiment, the aim is to implement each of the techniques such that the data is spread in the same manner across the five client nodes for each of the techniques.

4.1. Experiment 1

In this setup, we create five sets of noisy versions of the images of CIFAR-10. To this end, the CIFAR images are first normalized, 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 [35], with the categorical cross-entropy as the loss function, L2 regularization, and Dropout and BatchNormalization layers. Node $(J+1)$ uses two dense layers. The architecture is shown in Figure 8. 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 $(J+1)$.

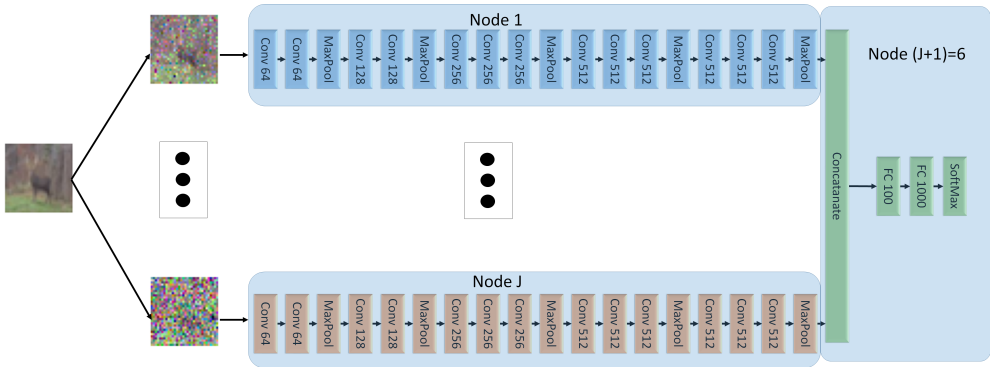


Figure 8. Network architecture. *Conv* stands for a convolutional layer, *Fc* stand for a fully connected layer.

For FL, each of the five client nodes is equipped with the *entire* network of Figure 8. 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 [21], each input node is equipped with an NN formed by *all* fives branches with convolutional networks (i.e., all the network of

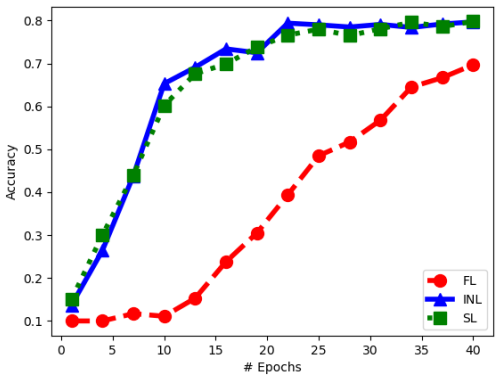
Figure 8, except the part at Node $(J+1)$) and node $(J+1)$ is equipped with fully connected layers at Node $(J+1)$ in Figure 8. Here, the processing during training is such that each input NN concatenates vertically the outputs of all convolutional layers and then passes that to node $(J+1)$, which then propagates back the error vector. After one epoch at one NN, the learned weights are passed to the next client, which performs the same operations on its part of the dataset.

The model depicted in Figure 8, which utilizes convolutional layers with a filter size of 3×3, comprises of approximately seventy-four million parameters, with 99.5% of these parameters constituting the encoding parts of the neural network. Table 2 presents the bandwidth requirements per epoch for the three techniques, considering the variation of the CIFAR-10 dataset used in the experiment, as well as the scenario where a dataset with ten times the amount of data is employed. It is observed that increasing the data size results in higher bandwidth requirements for both SL and INL, whereas the bandwidth requirements for FL remain unaffected.

Table 2. Experiment 1 bandwidth requirements of INL, FL and SL.

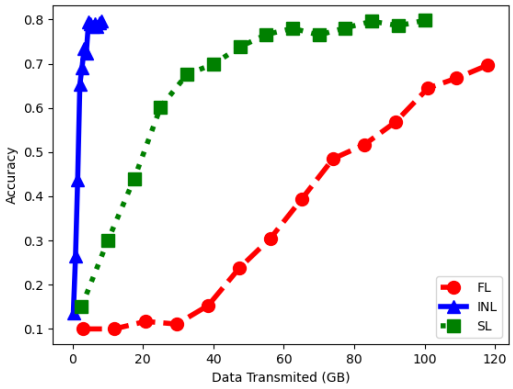
	Federated Learning	**Split Learning**	**In-Network Learning**
Bandwidth requirement	$2NJs$	$(2pq + \eta NJ)s$	$\frac{2pqs}{J}$
250,000 data points	2.96 GB	2.5 GB	0.2 GB
2,500,000 data points	2.96 GB	11.71 GB	2.05 GB

Figure 9a depicts the evolution of the 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. The final result is also less accurate. Figure 9b 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. This experiment showcases that the INL framework can save bandwidth, compared to SL and FL, when training large models by exchanging latent representations as opposed to model parameters. This is particularly relevant as some works argue to overparametrizing models can result in better model performance [36].



(**a**)

Figure 9. *Cont.*



(**b**)

Figure 9. Comparison of INL, FL and SL—Experiment 1. (**a**) Accuracy vs. # of epochs. (**b**) Accuracy vs. bandwidth cost.

4.2. 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 all 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 as $0.4, 1, 2, 3, 4$, respectively). Additionally, 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, Figure 10).

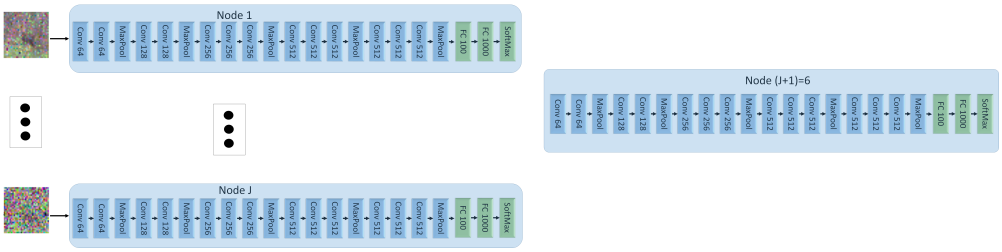


Figure 10. Used NN architecture for FL in Experiment 2.

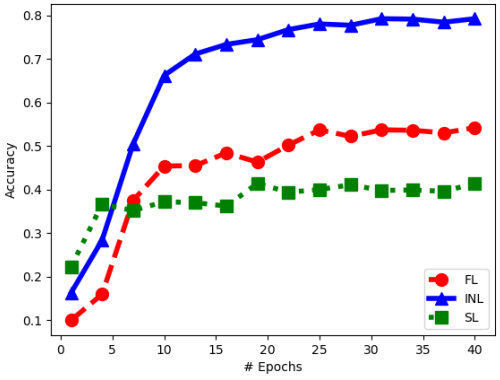
The model shown in Figure 10, for convolutional layers with filter of size 3×3, has approximately fifteen million parameters, with 97.6% of the parameters forming the decoding part of the network. Table 3 shows the bandwidth requierments for the three techniques per epoch for the variation of the CIFAR-10 dataset used in the experiment as well as for the case in which another dataset would be used that had ten times the amount of data. It is observed that increasing the data size results in higher bandwidth requirements for both SL and INL, whereas the bandwidth requirements for FL remain unaffected.

Table 3. Experiment 2 bandwidth requirements of INL, FL and SL.

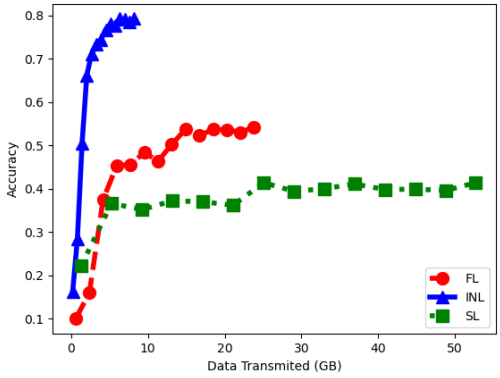
	Federated Learning	**Split Learning**	**In-Network Learning**
Bandwidth requirement	$2NJs$	$(2pq + \eta NJ)s$	$\frac{2pqs}{J}$
250,000 data points	0.6 GB	1.32 GB	0.2 GB
2,500,000 data points	0.6 GB	10.53 GB	2.05 GB

Figure 11b shows the performance of the three schemes during the inference phase in this case (for FL the inference is performed on an image which has average quality of the five noisy input images for INL and SL). Again, observe the benefits of INL over FL

and SL in terms of both achieved accuracy and bandwidth requirements. This experiment showcases INL's ability to make use of the correlations between the data observed by the different nodes, thus resulting in better network performance.



(**a**)



(**b**)

Figure 11. Comparison of INL, FL and SL—Experiment 2. (**a**) Accuracy vs. # of epochs. (**b**) Accuracy vs. bandwidth cost.

5. Conclusions

In this paper, our focus is on addressing the problem of distributed training and inference. We introduce INL, a novel framework which enables multiple nodes to collaboratively train a model that can be utilized in a distributed manner during the inference phase. Unlike existing works on distributed estimation and detection, our framework does not require prior knowledge of the data distribution; instead, it only necessitates access to a set of training samples. Furthermore, while other approaches to distributed training, such as FL and SL, assume local decision-making during the inference phase, we consider a scenario where the nodes observe data associated with the same event, thus enabling a joint decision that can lead to improved accuracy. The proposed INL algorithm offers a loss function derived through theoretical analysis, aiming to achieve the best trade-off between prediction accuracy, measured by logarithmic loss, and the amount of information exchanged among the nodes in the communication network.

Author Contributions: Conceptualization, A.Z.; methodology, A.Z.; software, M.M.; validation, M.M.; formal analysis, A.Z. and M.M.; investigation, M.M.; data curation, M.M.; writing—original draft preparation, A.Z. and M.M.; writing—review and editing, A.Z. and M.M.; visualization, M.M.; supervision, A.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data is contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Proof of Theorem 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 [33], then, the compression bin indices are transmitted as independent messages over the network $\mathcal{G}$ using linear-network coding [34] (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 $0 < \epsilon'' < \epsilon' < \epsilon$.

Appendix A.1. Codebook Generation

Fix a joint distribution $P_{X_1,\ldots,X_J,Y,U_1,\ldots,U_J}$ that factorizes as given by (16). Additionally, let $D = H(Y|U_1,\ldots,U_J)$, 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 $\mathbb{E}\Big[d(Y,\hat{Y})\Big] \leq \dfrac{D}{1+\epsilon}$, where $d : \mathcal{Y} \times \mathcal{P}(\mathcal{Y}) \longrightarrow \mathbb{R}_+$ is the distortion measure given by (5). For every $j \in \mathcal{J}$, let $\tilde{R}_j \geq R_j$. In addition, 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 according 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 $B_j(m_j) = \Big[(m_j-1)2^{n\tilde{R}_j - nR_j} : m_j 2^{n\tilde{R}_j - nR_j}\Big]$, $m_j \in [1 : 2^{nR_j}]$. 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.

Appendix A.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., $l_j \in \mathcal{B}_j(m_j)$.

Appendix A.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 $\mathcal{G} = (\mathcal{E}, \mathcal{N}, \mathcal{C})$, they are encoded as if they were independent-messages using the linear network coding scheme of [34] (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

$$\sum_{j\in\mathcal{S}\cap\mathcal{J}} R_j \leq C(\mathcal{S}) \tag{A1}$$

where $C(\mathcal{S})$ is defined by (14).

Appendix A.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 [33] (see also [34] (Theorem 12.1))

$$\sum_{j\in\mathcal{S}} R_j \geq I(U_{\mathcal{S}}; X_{\mathcal{S}}|U_{\mathcal{S}^c}). \tag{A2}$$

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 1.

Appendix B. Proof of Proposition 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 (23) for the given s. By making the substitution in (22):

$$\begin{aligned} \Delta_s =& I(Y; U_1^*, U_2^*, U_3^*) && \text{(A3)} \\ \leq& \Delta && \text{(A4)} \end{aligned}$$

where (A4) holds since Δ is the maximum $I(Y; U_1, U_2, U_3)$ over all distribution for which (20b) 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:

$$\begin{aligned} \Delta =& H(Y) - H(Y|U_1^*, U_2^*, U_3^*) \\ \leq& H(Y) - H(Y|U_1^*, U_2^*, U_3^*) + sC_{\text{sum}} \\ & - s\Big[I(X_2, X_3; U_2^*, U_3^*|U_1^*) + I(X_1, X_2, X_3; U_1^*, U_2^*, U_3^*)\Big] && \text{(A5)} \\ \leq& H(Y) + \max_{\mathbf{P}} \mathcal{L}_s(\mathbf{P}) + sC_{\text{sum}} && \text{(A6)} \\ =& \Delta_s - sC_s + sC_{\text{sum}} \\ =& \Delta_s + s(C_{\text{sum}} - C_s). && \text{(A7)} \end{aligned}$$

where (A5) follows from (20b). Inequality (A6) holds due to the fact that $\max_{\mathbf{P}} \mathcal{L}(\mathbf{P})$ takes place over all $\mathbf{P}$, including $\mathbf{P}^*$. Since (A7) 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 (A4) this completes the proof.

Appendix C. Proof of Lemma 1

We have

$$\begin{aligned} \mathcal{L}_s(\mathbf{P}) =& -H(Y|U_1, U_2, U_3) - sI(X_1, X_2, X_3; U_1, U_2, U_3) \\ & - sI(X_2, X_3; U_2, U_3|U_1) && \text{(A8)} \\ =& -H(Y|U_1, U_2, U_3) \\ & - s\Big[I(X_1; U_1) + 2I(X_2, X_3; U_2, U_3|U_1)\Big] && \text{(A9)} \\ =& -H(Y|U_1, U_2, U_3) - sI(X_1; U_1) - 2sI(X_2; U_2) \\ & - 2s\Big[I(X_3; U_3) - I(U_3; U_1, U_2) - I(U_2; U_1)\Big] && \text{(A10)} \\ =& -H(Y|U_1, U_2, U_3) - sI(X_1; U_1) - 2sI(X_2; U_2) \\ & + 2s\Big[I(U_2; U_1) + I(U_3; U_1, U_2) - I(X_3; U_3)\Big] && \text{(A11)} \\ \geq& -H(Y|U_1, U_4) - sI(X_1; U_1) - 2s\Big[I(X_2; U_2) + I(X_3; U_3)\Big] \\ & + 2s\Big[I(U_2; U_1) + I(U_3; U_1, U_2)\Big] && \text{(A12)} \end{aligned}$$

where (A9) holds since $U_1 \multimap X_1 \multimap (X_2, X_3, U_2, U_3)$ and $(U_2, U_3) \multimap (X_2, X_3) \multimap (U_1, X_1)$ (A10) holds since $U_2 \multimap X_2 \multimap (U_1, X_3)$ and $U_3 \multimap X_3 \multimap (U_1, U_2, X_2)$; (A12) hold since $U_4 \multimap (U_2, U_3) \multimap (Y, U_1)$.

Appendix D. Proof of Lemma 2

From [25] (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:

$$H(Y|Z) = \mathbb{E}[-\log Q_{Y|Z}(Y|Z)] - D_{\mathrm{KL}}(P_{Y|Z}\|Q_{Y|Z}). \tag{A13}$$

From [25] (eq. (81)):

$$\begin{aligned} I(X;Z) &= H(Z) - H(Z|X) \\ &= D_{\mathrm{KL}}(P_{Z|X}\|Q_Z) - D_{\mathrm{KL}}(P_Z\|Q_Z). \end{aligned} \tag{A14}$$

Now substituting Equations (A13) and (A14) in (28) the following result is obtained:

$$\begin{aligned} \mathcal{L}_s^{\text{low}}(\mathbf{P}_+) =& -H(Y|U_1,U_4) - sI(X_1;U_1) - 2sI(X_2;U_2) \\ & - 2sI(X_3;U_3) + 2s\Big[I(U_2;U_1) + I(U_3;U_1,U_2)\Big] \\ =& \mathbb{E}[\log Q_{Y|U_1,U_4}] + D_{\mathrm{KL}}(P_{Y|U_1,U_4}\|Q_{Y|U_1,U_4}) \\ & - sD_{\mathrm{KL}}(P_{U_1|X_1}\|Q_{U_1}) + sD_{\mathrm{KL}}(P_{U_1}\|Q_{U_1}) \\ & - 2sD_{\mathrm{KL}}(P_{U_2|X_2}\|Q_{U_2}) + 2sD_{\mathrm{KL}}(P_{U_2}\|Q_{U_2}) \\ & - 2sD_{\mathrm{KL}}(P_{U_3|X_3}\|Q_{U_3}) + 2sD_{\mathrm{KL}}(P_{U_3}\|Q_{U_3}) \\ & + 2sD_{\mathrm{KL}}(P_{U_2|U_1}\|Q_{U_2}) - 2sD_{\mathrm{KL}}(P_{U_2}\|Q_{U_2}) \\ & + 2sD_{\mathrm{KL}}(P_{U_3|U_1,U_2}\|Q_{U_3}) - 2sD_{\mathrm{KL}}(P_{U_3}\|Q_{U_3}) \\ =& \mathcal{L}_s^{\text{v-low}} + sD_{\mathrm{KL}}(P_{U_1}\|Q_{U_1}) + 2sD_{\mathrm{KL}}(P_{U_2|U_1}\|Q_{U_2}) \\ & + 2sD_{\mathrm{KL}}(P_{U_3|U_1,U_2}\|Q_{U_3}) + D_{\mathrm{KL}}(P_{Y|U_1,U_4}\|Q_{Y|U_1,U_4}) \\ \geq& \mathcal{L}_s^{\text{v-low}} \end{aligned} \tag{A15}$$

The last inequality (A15) holds due to the fact that KL divergence is always positive and $s \geq 0$, thus proving the lemma.

References

1. Zou, Z.; Shi, Z.; Guo, Y.; Ye, J. Object Detection in 20 Years: A Survey. *arXiv* **2019**, arXiv:1905.05055.
2. Glaser, J.I.; Benjamin, A.S.; Farhoodi, R.; Kording, K.P. The roles of supervised machine learning in systems neuroscience. *Prog. Neurobiol.* **2019**, *175*, 126–137. [CrossRef]
3. Pluim, J.P.W.; Maintz, J.B.A.; Viergever, M.A. Mutual-information-based registration of medical images: A survey. *IEEE Trans. Med. Imaging* **2003**, *22*, 986–1004. [CrossRef]
4. Kober, J.; Bagnell, J.; Peters, J. Reinforcement Learning in Robotics: A Survey. *Int. J. Robot. Res.* **2013**, *32*, 1238–1274. [CrossRef]
5. Vinyals, O.; Le, Q.V. A Neural Conversational Model. *arXiv* **2015**, arXiv:1506.05869.
6. Farsad, N.; Yilmaz, H.B.; Eckford, A.; Chae, C.; Guo, W. A Comprehensive Survey of Recent Advancements in Molecular Communication. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 1887–1919. [CrossRef]
7. Peter Hong, Y.W.; Wang, C.C. In-Network Learning via Over-the-Air Computation in Internet-of-Things. In Proceedings of the 2021 IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Lucca, Italy, 27–30 September 2021; pp. 141–145. [CrossRef]
8. Du, R.; Magnusson, S.; Fischione, C. The Internet of Things as a deep neural network. *IEEE Commun. Mag.* **2020**, *58*, 20–25. [CrossRef]
9. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-Efficient Learning of Deep Networks from Decentralized Data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, Fort Lauderdale, FL, USA, 20–22 April 2017; Volume 54, pp. 1273–1282.
10. Xiao, J.J.; Ribeiro, A.; Luo, Z.Q.; Giannakis, G. Distributed compression-estimation using wireless sensor networks. *IEEE Signal Process. Mag.* **2006**, *23*, 27–41.
11. Kreidl, O.P.; Tsitsiklis, J.N.; Zoumpoulis, S.I. Decentralized detection in sensor network architectures with feedback. In Proceedings of the 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, USA, 29 September–1 October 2010; pp. 1605–1609. [CrossRef]
12. Chamberland, J.f.; Veeravalli, V.V. Wireless Sensors in Distributed Detection Applications. *IEEE Signal Process. Mag.* **2007**, *24*, 16–25.

13. Tsitsiklis, J.N. Decentralized detection. In *Advances in Statistical Signal Processing*; JAI Press: Stamford, CT, USA, 1993; pp. 297–344.
14. Simic, S. A learning-theory approach to sensor networks. *IEEE Pervasive Comput.* **2003**, *2*, 44–49.
15. Predd, J.; Kulkarni, S.; Poor, H. Distributed learning in wireless sensor networks. *IEEE Signal Process. Mag.* **2006**, *23*, 56–69.
16. Nguyen, X.; Wainwright, M.; Jordan, M. Nonparametric decentralized detection using kernel methods. *IEEE Trans. Signal Process.* **2005**, *53*, 4053–4066. [CrossRef]
17. Jagyasi, B.; Raval, J. Data aggregation in multihop wireless mesh sensor Neural Networks. In Proceedings of the 2015 9th International Conference on Sensing Technology (ICST), Auckland, New Zealand, 8–10 December 2015; pp. 65–70. [CrossRef]
18. Tran, N.H.; Bao, W.; Zomaya, A.; Nguyen, M.N.H.; Hong, C.S. Federated Learning over Wireless Networks: Optimization Model Design and Analysis. In Proceedings of the IEEE INFOCOM 2019—IEEE Conference on Computer Communications, Paris, France, 29 April–2 May 2019; pp. 1387–1395. [CrossRef]
19. Amiri, M.M.; Gündüz, D. Federated learning over wireless fading channels. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 3546–3557. [CrossRef]
20. Yang, H.H.; Liu, Z.; Quek, T.Q.S.; Poor, H.V. Scheduling Policies for Federated Learning in Wireless Networks. *IEEE Trans. Commun.* **2020**, *68*, 317–333. [CrossRef]
21. Gupta, O.; Raskar, R. Distributed learning of deep neural network over multiple agents. *J. Netw. Comput. Appl.* **2018**, *116*, 1–8. [CrossRef]
22. Ceballos, I.; Sharma, V.; Mugica, E.; Singh, A.; Roman, A.; Vepakomma, P.; Raskar, R. SplitNN-driven Vertical Partitioning. *arXiv* **2020**, arXiv:2008.04137.
23. National Institutes of Health. *NIH Data Sharing Policy and Implementation Guidance*; National Institutes of Health: Bethesda, MD, USA, 2003; Volume 18, p. 2009.
24. Aguerri, I.E.; Zaidi, A. Distributed Information Bottleneck Method for Discrete and Gaussian Sources. In Proceedings of the IEEE International Zurich Seminar on Information and Communications, Zurich, Switzerland, 21–23 February 2018.
25. Aguerri, I.E.; Zaidi, A. Distributed Variational Representation Learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *43*, 120–138. [CrossRef]
26. Zaidi, A.; Aguerri, I.E.; Shamai (Shitz), S. On the Information Bottleneck Problems: Models, Connections, Applications and Information Theoretic Views. *Entropy* **2020**, *22*, 151. [CrossRef]
27. Moldoveanu, M.; Zaidi, A. On in-network learning. A comparative study with federated and split learning. In Proceedings of the 2021 IEEE 22nd International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Lucca, Italy, 27–30 September 2021; pp. 221–225.
28. Moldoveanu, M.; Zaidi, A. In-network Learning for Distributed Training and Inference in Networks. In Proceedings of the IEEE Globecom 2021 Workshops, Madrid, Spain, 7–11 December 2021; pp. 1–6. [CrossRef]
29. Yang, Q.; Liu, Y.; Chen, T.; Tong, Y. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol. (TIST)* **2019**, *10*, 1–19. [CrossRef]
30. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.
31. Flamich, G.; Havasi, M.; Hernández-Lobato, J.M. Compressing images by encoding their latent representations with relative entropy coding. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 16131–16141.
32. Li, C.T.; Gamal, A.E. Strong Functional Representation Lemma and Applications to Coding Theorems. *IEEE Trans. Inf. Theory* **2018**, *64*, 6967–6978. [CrossRef]
33. Berger, T.; Yeung, R. Multiterminal source encoding with one distortion criterion. *IEEE Trans. Inf. Theory* **1989**, *35*, 228–236. [CrossRef]
34. El Gamal, A.; Kim, Y.H. *Network Information Theory*; Cambridge University Press: Cambridge, UK, 2011. [CrossRef]
35. Liu, S.; Deng, W. Very deep convolutional neural network based image classification using small training sample size. In Proceedings of the 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), Kuala Lumpur, Malaysia, 3–6 November 2015; pp. 730–734. [CrossRef]
36. Liu, H.; Chen, M.; Er, S.; Liao, W.; Zhang, T.; Zhao, T. Benefits of overparameterized convolutional residual networks: Function approximation under smoothness constraint. In Proceedings of the International Conference on Machine Learning. PMLR, Baltimore, MD, USA, 17–23 July 2022; pp. 13669–13703.