2024 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 22–25, 2024, LONDON, UK

DECENTRALIZED FUSION OF EXPERTS OVER NETWORKS

Marco Carpentiero[⋆] *Vincenzo Matta*[⋆] *Stefan Vlaski*[◇] *Ali H. Sayed*[†]

[⋆] University of Salerno, Italy
[◇] Imperial College London, UK
[†] EPFL, Lausanne, Switzerland

ABSTRACT

This article considers a network of agents interested in solving a classification task. The datasets available to accomplish the task are heterogeneous and dispersed across the agents. Each agent is interested in discriminating among the "inner" hypotheses reflected in its individual dataset. Moreover, the datasets at the different agents can be further labeled in terms of additional characteristics, giving rise to an enlarged space of hypotheses. The agents have no local information to distinguish their own dataset from the datasets of other agents. To overcome this issue, they want to share their individual knowledge to build an overall model that is able to address the classification task comprising all possible hypotheses. Starting from the optimal Bayesian fusion rule, we develop a strategy nicknamed decentralized fusion of experts (DeFoE), which is able to build a global classifier starting from the classifiers locally available to the agents, at a reduced complexity, without re-training them from scratch. The effectiveness of the proposed strategy is shown over a benchmark dataset containing real images.

Index Terms— Distributed optimization, decentralized learning and inference, hypothesis testing, supervised classification.

1. INTRODUCTION

The steady progress in the theory of machine learning (ML) and artificial intelligence (AI) has led to the development of several useful strategies that achieve extraordinary results in numerous applications. Many tools are made freely available by the hyperscalers (i.e., big cloud providers) to academic researchers, companies, or practitioners. These tools allow to train sophisticated learning architectures and solve challenging tasks. For example, one can construct an "expert" that is able to label several images to classify them in terms of certain attributes.

However, in face of the complexity of challenging datasets, there still exists a sensible gap between the AI systems developed by the hyperscalers and the systems that an end user (e.g., an academic researcher or a company of medium size) is able to develop. This gap mainly arises from two limitations: the dataset and the computation/memory power. In most cases, the magnificent performance that we observe in the most popular AI applications results from having trained extremely sophisticated deep learning architectures over extremely complex datasets. In contrast, the end user is not able to construct learning models with performance comparable with that of the best available systems. One typical solution in this case for the end user is to borrow some pre-trained model made available by a hyperscaler, and possibly refining this model (e.g., by a fine tuning) over its own dataset.

The work of V. Matta was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART").

1.1. Main Question and Motivation

In this article we focus on the following problem. A group of agents are *individually* able to solve a certain classification task where they want to choose a particular hypothesis θ from a discrete finite set $\Theta = \{\theta_1, \theta_2, \dots, \theta_H\}$. Each agent has learned, from its own dataset, a certain decision rule to classify the hypotheses. For example, the agents might possess datasets of images representing cars or trucks, and they have learned to distinguish the type of vehicle.

The datasets of the agents might be different, in the sense that additional qualifications (i.e., labels) can be added to the original classes. Continuing with the car/truck example, assume that two agents correspond to two companies that produce cars or trucks. Then, one can append to the original hypothesis set

$$\Theta = \{\text{car}, \text{truck}\} \tag{1}$$

a label to identify the particular company, namely, we can define the hypothesis sets

$$\begin{aligned} \Theta_1 &= \{\text{car brand } 1, \text{truck brand } 1\}, \\ \Theta_2 &= \{\text{car brand } 2, \text{truck brand } 2\}. \end{aligned} \tag{2}$$

Assume that both companies have developed or borrowed an AI system that, upon observing an image of a car or truck, is able to correctly label it with high accuracy. Now the companies want to exploit the heterogeneity existing between their datasets to learn more attributes of the images; they want to strengthen their individual classifiers to distinguish also the brand the particular car or truck belongs to, which amounts to classifying a hypothesis belonging to the *network* set

$$\begin{aligned} \Theta_{\mathsf{net}} &= \Theta_1 \cup \Theta_2 \\ &= \{\text{car brand } 1, \text{truck brand } 1, \text{car brand } 2, \text{truck brand } 2\}. \end{aligned} \tag{3}$$

However, to build a decision model over Θ_{net}, the agents face two fundamental limitations. First, they cannot be re-trained from scratch. In the case where they have borrowed some expert from a hyperscaler, this limitation is motivated by the fact that the expert structure is too complex to be learned from scratch given the dataset/computation/memory limitations of the agents. In the case where the agent built the expert on its own, this construction is assumed to be too costly to be implemented again.

The second limitation is related to the fact that the agents are spatially dispersed, and they have to cooperate in a distributed manner. For this reason, the agents want to minimize the training complexity, in terms of communication rounds, learning architecture, and so on.

1.2. Related Work

The problem of fusing the abilities of multiple classifiers is addressed in the context of *ensemble* learning. In this problem, the focus is on combining different classifiers operating on the same hypothesis set to attain improved performance. Two known strategies to achieve this goal are bagging (bootstrap aggregating) [1, 2] and boosting [1,3]. In bagging, some learning machines are trained over a collection of datasets generated through a bootstrap procedure, and then the resulting classifiers are aggregated by means of some voting procedure. In boosting, classifiers are trained sequentially in a row; some scores are assigned to the performance of a classifier to highlight its strength and weakness. Then, the subsequent classifier in the pipeline is trained by accounting for the strength and weakness of the previous classifiers. Finally, the decisions of the classifiers are weighted according to the obtained scores. The problem of enhancing the classification performance through cooperation has been more recently addressed in a *decentralized* setting, in the context of social learning [4, 5].

However, the aforementioned ensemble learning strategies assume that the hypothesis set is one and the same for the different classifiers and the goal is to boost up the performance of the individual classifiers. In the problem addressed in the present work, we have instead different hypothesis sets for different agents, and the goal is to build a decision model over the hypothesis set obtained by aggregating the hypothesis sets of the individual agents. This is a different problem that bears some similarities to the frameworks of *incremental learning* [6, 7] and *accretionary learning* [8]. In these frameworks there is a single learning entity that observes additional hypotheses arising over time. In comparison, in our case the different hypotheses stem *over space* (i.e., across the network agents), rather than over time. The *decentralized* nature of our problem determines the following distinguishing features.

One fundamental peculiarity is that the information necessary to pass from Θ to Θ_{net} can only be obtained by fusing the complementary views of the agents. In other words, each individual agent has some private knowledge that, when shared with the other agents, would permit to differentiate the hypotheses belonging to different clusters. The second fundamental property is that the datasets corresponding to different hypotheses are spatially dispersed and the agents must cooperate locally (i.e., with their neighbors according to a certain network topology) to build the global decision model. Another critical peculiarity is that to combine their own complementary views, the agents are not allowed to share their heterogeneous data. They are only allowed to share model parameters. The mix of these conditions gives rise to the problem that we call *decentralized fusion of experts (DeFoE)*.

To address the problem, we rely on the theory of distributed learning and optimization [1]. By simply exploiting Bayes' rule, we are able to formulate the fusion of experts as a learning problem that involves the construction of an additional decision model aimed at classifying only the additional attributes that distinguish the individual agent datasets. This problem is cast in the form of a distributed optimization problem with a global cost function that is able to account for the complementary views brought by the individual agents.

Notation. We denote random variables with bold font. For a nonnegative function $f(\mu)$ with positive argument μ, the notation $f(\mu) = O(\mu)$ means that $f(\mu) \leq c\,\mu$ for all $\mu \leq \mu_0$, for some positive values c and μ_0. The symbols $\mathbb{E}$ and $\mathbb{P}$ denote expectation and probability, respectively. The symbol $[K]$ denotes the set $\{1, 2, \dots, K\}$. For two matrices X and Y, the notation $X \geq Y$ means that $X - Y$ is positive semidefinite. The function $\mathbb{I}[\mathcal{C}]$ is the indicator function, which is equal to 1 if condition $\mathcal{C}$ is true, and is 0 otherwise.

2. PROBLEM FORMULATION

Consider a network of K agents. Each agent $k \in [K]$ owns a dataset $\boldsymbol{D}_k$ comprised of N_k independent and identically distributed (iid) training examples, namely,

$$\boldsymbol{D}_k = \{\boldsymbol{x}_{k,n}, \boldsymbol{\theta}_{k,n}\}_{n=1}^{N_k}. \tag{4}$$

Each training example is constituted by a (feature, label) pair, with

$$\boldsymbol{x}_{k,n} \in \mathcal{X}, \qquad \boldsymbol{\theta}_{k,n} \in \Theta^{(k)}. \tag{5}$$

The feature space $\mathcal{X}$ is common to all agents. For example, the agents observe the same type of images. Note that the hypothesis set $\Theta^{(k)}$ is allowed to be different across the agents. We do not impose the constraint that all agents have different hypothesis sets. In particular, we assume that there exist C disjoint clusters of hypotheses, denoted by Θ_c, for $c \in [C]$. Then, the hypothesis set of agent k is one of these clusters, namely, we have

$$\Theta^{(k)} = \Theta_{c_k}, \qquad \text{for } c_k \in [C]. \tag{6}$$

The hypothesis set at the network level is

$$\Theta_{\text{net}} \triangleq \bigcup_{c=1}^{C} \Theta_c = \bigcup_{k=1}^{K} \Theta^{(k)}. \tag{7}$$

Note that the sets $\Theta^{(k)}$ are not necessarily disjoint since two agents are permitted to share the same Θ_c.

The agents are interested in the following problem. Assume that they observe a new feature $\boldsymbol{x}$, that is generated from some hypothesis $\boldsymbol{\theta} \in \Theta_{\text{net}}$. There is no additional label specifying the cluster, and the agents are interested in classifying the feature into one of the labels $\theta \in \Theta_{\text{net}}$. To this end, they must focus on the joint distribution of a (feature, label) pair $(\boldsymbol{x}, \boldsymbol{\theta})$ *across the agents*, i.e., for $\theta \in \Theta_{\text{net}}$. The labels $\boldsymbol{\theta} \in \Theta_{\text{net}}$ are distributed according to some prior pmf:

$$\pi(\theta) \triangleq \mathbb{P}[\boldsymbol{\theta} = \theta], \qquad \theta \in \Theta_{\text{net}}. \tag{8}$$

The generative mechanism that rules how $\boldsymbol{x}$ is drawn, given the underlying label $\boldsymbol{\theta}$, is described by a *likelihood* $\ell(x|\theta)$. For simplicity of presentation, we perform the analysis by assuming that $\boldsymbol{x}$ is a discrete random variable. Thus, as a function of x, the likelihood will be a probability mass function (pmf). Note that the final result we are interested in (which is the evaluation of the posterior $\mathbb{P}[\boldsymbol{\theta} = \theta|\boldsymbol{x} = x]$) would still hold for the continuous case. Using Bayes' rule, we can build the joint model

$$p_{\text{net}}(x, \theta) \triangleq \mathbb{P}[\boldsymbol{x} = x, \boldsymbol{\theta} = \theta] = \pi(\theta)\ell(x|\theta). \tag{9}$$

Observe that, since agent k observes only data corresponding to labels $\theta \in \Theta_{c_k}$, the distributions within the individual clusters Θ_c are related to the distribution $p_{\text{net}}(x, \theta)$ as follows. Let

$$\pi(\theta|\Theta_c) = \frac{\pi(\theta)}{\sum\limits_{\theta' \in \Theta_c} \pi(\theta')} \mathbb{I}[\theta \in \Theta_c] \tag{10}$$

be the *conditional* prior pmf given that $\boldsymbol{\theta} \in \Theta_c$. From the total probability law, the prior distribution in (8) can be written as

$$\pi(\theta) = \sum_{c=1}^{C} \pi(\theta|\Theta_c)\mathbb{P}[\boldsymbol{\theta} \in \Theta_c]. \tag{11}$$

The probability that $\boldsymbol{\theta}$ belongs to Θ_c can be evaluated by considering the relative frequency of observing each cluster across the agents, yielding:

$$\mathbb{P}[\boldsymbol{\theta} \in \Theta_c] = \frac{1}{K}\sum_{k=1}^{K} \mathbb{I}[c_k = c] = \frac{K(c)}{K}, \tag{12}$$

where $K(c)$ is the number of agents that are observing a dataset corresponding to cluster Θ_c. Substituting (12) into (11), we obtain

$$\pi(\theta) = \sum_{c=1}^{C} \pi(\theta|\Theta_c)\frac{K(c)}{K}. \tag{13}$$

The *marginal* distribution of the features $\boldsymbol{x}_{k,n}$ is obtained as

$$\mathbb{P}[\boldsymbol{x}_{k,n} = x] = \sum_{\theta \in \Theta_{c_k}} \pi(\theta|\Theta_{c_k})\ell(x|\theta). \tag{14}$$

Moreover, the posterior corresponding to the joint model $p_{\mathsf{net}}(x, \theta)$ from (9) can be obtained by applying the law of total probability to get

$$p_{\mathsf{net}}(\theta|x) \triangleq \sum_{c=1}^{C} \mathbb{P}[\boldsymbol{\theta} = \theta|\boldsymbol{x} = x, \boldsymbol{\theta} \in \Theta_c]\,\mathbb{P}[\boldsymbol{\theta} \in \Theta_c|\boldsymbol{x} = x]. \tag{15}$$

Let us denote the first probability in the summation by

$$p_c(\theta|x) \triangleq \mathbb{P}[\boldsymbol{\theta} = \theta|\boldsymbol{x} = x, \boldsymbol{\theta} \in \Theta_c] \tag{16}$$

and the second by

$$Q(\Theta_c|x) \triangleq \mathbb{P}[\boldsymbol{\theta} \in \Theta_c|\boldsymbol{x} = x]. \tag{17}$$

With these two definitions, we rewrite (15) as

$$p_{\mathsf{net}}(\theta|x) = \sum_{c=1}^{C} p_c(\theta|x)Q(\Theta_c|x). \tag{18}$$

Observe further that

$$\sum_{\theta \in \Theta_c} p_c(\theta|x) = 1, \qquad \sum_{c=1}^{C} Q(\Theta_c|x) = 1. \tag{19}$$

That is, $p_c(\theta|x)$ is a pmf defined over set Θ_c, while $Q(\Theta_c|x)$ is a pmf over the set $[C]$. From the first relation in (19), we conclude that $p_c(\theta|x)$ is equal to 0 for all $\theta \notin \Theta_c$. This implies that, in the summation appearing in (18), only the term corresponding to the cluster c to which θ belongs is actually present.

Bayes' classifier, which minimizes the probability of error, works as follows [1]. First, it computes the posterior probability $p_{\mathsf{net}}(\theta|x)$ and then, given an observation x, it chooses the estimated class $\widehat{\theta}(x) \in \Theta_{\mathsf{net}}$ by maximizing the posterior probability over the possible hypotheses:

$$\widehat{\theta}(x) = \arg\max_{\theta \in \Theta_{\mathsf{net}}} p_{\mathsf{net}}(\theta|x). \tag{20}$$

This rule is known as the MAP (maximum a posteriori) rule.

Let us examine how the MAP rule (20) works in our case. Using (18), we see that, to choose between two hypotheses θ, θ', both belonging to Θ_c, we have to compare (recall that, since $\theta, \theta' \in \Theta_c$, in (18) the terms corresponding to clusters different from c are zero)

$$p_c(\theta|x)Q(\Theta_c|x) \underset{\theta'}{\overset{\theta}{\gtrless}} p_c(\theta'|x)Q(\Theta_c|x) \iff p_c(\theta|x) \underset{\theta'}{\overset{\theta}{\gtrless}} p_c(\theta'|x). \tag{21}$$

This means that, to choose between two hypotheses *within the same cluster c*, the conditional posterior $p_c(\theta|x)$ is sufficient. This is no longer true when we have to compare hypotheses from different clusters, where the role of $Q(\Theta_c|x)$ becomes important. Note that $Q(\Theta_c|x)$ represents a pmf over the clusters, i.e., it would correspond to the relevant posterior if we want to solve a classification problem to decide to which cluster θ belongs to, without paying attention to the inner structure of the cluster.

2.1. Agents' Decision Models

As explained in the previous section, each agent k is assigned a dataset relative to a particular cluster of hypotheses Θ_{c_k}. Accordingly, the posterior corresponding to agent k is the conditional posterior *given that θ belongs to Θ_{c_k}*. In other words, the posterior distribution "seen" by agent k, for its labels $\boldsymbol{\theta}_{k,n}$ given its features $\boldsymbol{x}_{k,n}$, is the quantity $p_{c_k}(\theta|x)$ from (16). However, each agent k is not able to compute exactly this posterior, since this would imply an exact knowledge of the underlying mechanism ruling the data. What the agent can do is to learn a decision model from its dataset $\boldsymbol{D}_k$. In some applications (e.g., the cars/trucks example mentioned in the introduction), we can assume that the same discriminative model can be used for all clusters of hypotheses. That is, agent k could use its local decision model as a proxy for $p_c(\theta|x)$ for all $c \in [C]$. In other applications, the decision models might vary across the clusters. In the latter case, the agents can easily share the parameters of the learned models, such that each agent k possesses discriminative models for all clusters $c \in [C]$. We denote the posterior available to agent k and relative to cluster c as follows:

$$\widehat{p}_c^{(k)}(\theta|x), \quad k \in [K],\ c \in [C]. \tag{22}$$

Each model $\widehat{p}_c^{(k)}(\theta|x)$ is intended to be an approximation of the true conditional posterior $p_c(\theta|x)$. However, we have seen in the previous section that the knowledge of the conditional posteriors pertaining to the individual clusters is not sufficient to solve the classification problem over Θ_{net}. Moreover, in our problem the individual agents have no information regarding the differences *across the clusters*, since each agent has only data pertaining to a single cluster. For this reason, the agents need to cooperate to learn $Q(\Theta_c|x)$, as we explain next.

3. DEFOE STRATEGY

To learn the pmf $Q(\Theta_c|x)$, we focus on a discriminative paradigm [1], where we aim at building an estimated posterior $\widehat{Q}(\Theta_c|x; w)$ that can be cast in the following softmax form:

$$\widehat{Q}(\Theta_c|x; w) \triangleq \frac{e^{\delta(x,\Theta_c;w)}}{\sum_{c=1}^{C} e^{\delta(x,\Theta_c;w)}}, \qquad c \in [C], \tag{23}$$

where $\delta(x, \Theta_c; w)$ represents a decision function that is chosen from some admissible family of functions parametrized by a vector $w \in$

$\mathbb{R}^M$. For example, the family of functions can result from a logistic regression model or a multilayer perceptron [1]. We now illustrate how each agent k can learn the posterior by means of a *decentralized* optimization strategy.

First, we must select a cost function that quantifies the discrepancy between the true posterior and the estimated one. A classical choice in classification problems is the *regularized conditional cross-entropy*. Consider first the conditional cross-entropy between the true posterior $Q(\Theta_c|x)$ and a candidate posterior $\widehat{Q}(\Theta_c|x;w)$ [14]:

$$\begin{aligned} H(w) &\triangleq \sum_{x\in\mathcal{X}} \mathbb{P}[\boldsymbol{x}=x] \sum_{c=1}^{C} Q(\Theta_c|x) \log \frac{1}{\widehat{Q}(\Theta_c|x;w)} \\ &= \sum_{x\in\mathcal{X}} \sum_{c=1}^{C} \mathbb{P}[\boldsymbol{x}=x, \boldsymbol{\theta}\in\Theta_c] \log \frac{1}{\widehat{Q}(\Theta_c|x;w)}. \end{aligned} \tag{24}$$

Likewise, consider the conditional cross-entropy at agent k, which, since agent k observes only labels from set Θ_{c_k}, is given by

$$H_k(w) \triangleq \sum_{x\in\mathcal{X}} \mathbb{P}[\boldsymbol{x}_{k,n}=x] \log \frac{1}{\widehat{Q}(\Theta_{c_k}|x;w)}, \tag{25}$$

where $\mathbb{P}[\boldsymbol{x}_{k,n}=x]$ is defined by (14). Now, each agent k uses as *local* cost function the regularized conditional cross-entropy

$$J_k(w) \triangleq H_k(w) + \rho_k \|w\|^2, \tag{26}$$

where $\rho_k > 0$ is a regularization parameter. It is also useful to introduce the corresponding *global* cost function

$$J(w) \triangleq \frac{1}{K} \sum_{k=1}^{K} J_k(w). \tag{27}$$

Unfortunately, in supervised classification problems, the cost functions $J_k(w)$ cannot be computed since the probability $\mathbb{P}[\boldsymbol{x}_{k,n}=x]$ is unknown. For this reason, for each n, agent k replaces the exact cost function with a *stochastic instantaneous approximation*

$$\widehat{J}_k(w;\boldsymbol{x}_{k,n}) \triangleq \log \frac{1}{\widehat{Q}(\Theta_c|\boldsymbol{x}_{k,n};w)} + \rho_k\|w\|^2. \tag{28}$$

We make the following assumption on the family of decision functions $\delta(x,\Theta_c;w)$ employed by the agents.

Assumption 1 (Regularity of cost functions). *For all $w\in\mathbb{R}^M$, the decision functions $\delta(x,\Theta_c;w)$ are such that: i) each cost function $J_k(w)$ is twice-differentiable and its Hessian matrix satisfies the Lipschitz condition $\nabla^2 J_k(w) \leq \eta I_M$, for some constant $\eta>0$; and ii) the aggregate cost function $J(w)$ in* (27) *is ν-strongly convex, namely, a positive constant ν exists such that $\nabla^2 J(w) \geq \nu I_M$.* ☐

3.1. Adapt-then-Combine Diffusion Strategy

In order to learn the posterior $Q(\Theta_c|x)$ in a distributed manner, the agents resort to the following diffusion strategy, which is known as Adapt-Then-Combine (ATC) [1,9,10]:

$$\begin{cases} \boldsymbol{\psi}_{k,i} = \boldsymbol{w}_{k,i-1} - \mu\nabla\widehat{J}_k(\boldsymbol{w}_{k,i-1};\boldsymbol{x}_{k,i}) & \text{[Adapt]} \\ \boldsymbol{w}_{k,i} = \displaystyle\sum_{j=1}^{K} a_{jk}\boldsymbol{\psi}_{j,i} & \text{[Combine]} \end{cases} \tag{29}$$

According to this strategy, each agent $k\in[K]$, computes a sequence of iterates $\boldsymbol{w}_{k,i}\in\mathbb{R}^M$ over time $i\in\mathbb{N}$. In the adaptation step, agent k at time i computes the stochastic instantaneous approximation $\nabla\widehat{J}_k(\boldsymbol{w}_{k,i-1};\boldsymbol{x}_{k,i})$ of the true gradient, based on its fresh observation $\boldsymbol{x}_{k,i}$. The resulting stochastic gradient is scaled by a small step-size $\mu>0$ to compute an updated parameter vector $\boldsymbol{\psi}_{k,i}$ starting from the previous parameter vector $\boldsymbol{w}_{k,i-1}$ and following the approximate gradient direction. In the combination step, agent k combines the updated states from its neighboring agents. More specifically, the agents interact according to a given *weighted* graph. The graph edges are characterized by weights collected into a combination matrix A, whose (j,k) entry is denoted by a_{jk}. We say that a directed edge exists from j to k if, and only, if, $a_{jk}>0$. The *neighborhood* of agent k is accordingly defined as $\mathcal{N}_k \triangleq \{j\in[K]: a_{jk}>0\}$. As a result, agent k effectively incorporates into $\boldsymbol{w}_{k,i}$ only the updated states $\boldsymbol{\psi}_{j,i}$ received from its neighbors $j\in\mathcal{N}_k$. We make the following standard assumption on the combination matrix [9].

Assumption 2 (Combination matrix). *The combination matrix A is doubly stochastic, i.e., its entries are nonnegative and all rows and columns add up to* 1. *Moreover, matrix A is primitive, which means that there exist paths of common length between any pair of nodes (j,k) in both directions (i.e., from j to k and vice versa) [13].* ☐

The next theorem establishes that the proposed decentralized strategy allows each agent to approach a certain optimal parameter $w^\star$ that turns out to minimize the *overall* regularized conditional cross-entropy across *all agents*.

Theorem 1 (Training performance). *Let Assumptions 1 and 2 be satisfied and consider the distributed strategy in* (29) *with stochastic instantaneous approximation* (28). *Then, as the number of iterations i grows, and for sufficiently small step-sizes μ, each agent k converges within a small neighborhood of the vector parameter $w^\star$ that minimizes the overall regularized conditional cross-entropy computed with respect to the true model, namely,*

$$H(w) + \frac{\|w\|^2}{K}\sum_{k=1}^{K}\rho_k, \tag{30}$$

where $H(w)$ is defined by (24). *More specifically, we have that*

$$\limsup_{i\to\infty} \mathbb{E}\|\boldsymbol{w}_{k,i} - w^\star\|^2 = O(\mu). \tag{31}$$

Proof. For space constraints, we only give a sketch of the proof here. First, one has to appeal to the analytical tools presented, e.g., in [1,9,11,12], to show that, under Assumptions 1 and 2, Eq. (31) holds true when $w^\star$ is the minimizer of an aggregate cost function $\sum_{k=1}^{K} v_k J_k(w)$, where v_k is the kth entry of the Perron vector associated with the combination matrix A. Since by Assumption 2 A is doubly stochastic, it is known that $v_k = 1/K$ for all $k\in[K]$. As a result, we see that $w^\star$ is the minimizer (which can be shown to exist and be unique under Assumption 1) of $J(w)$ from (27). In view of (26), to complete the proof, it remains to show that $H(w) = (1/K)\sum H_k(w)$. Observe from (9) that

$$\begin{aligned} \mathbb{P}[\boldsymbol{x}=x,\boldsymbol{\theta}\in\Theta_c] &= \sum_{\theta\in\Theta_c} \pi(\theta)\ell(x|\theta) \\ &\overset{\text{Eq. (13)}}{=} \sum_{\theta\in\Theta_c} \ell(x|\theta) \sum_{c'=1}^{C} \pi(\theta|\Theta_{c'}) \frac{K(c')}{K} \mathbb{I}[\theta\in\Theta_{c'}] \\ &= \sum_{\theta\in\Theta_c} \ell(x|\theta)\pi(\theta|\Theta_c)\frac{K(c)}{K}. \end{aligned} \tag{32}$$

On the other hand, using (25) we get

$$\begin{aligned}\frac{1}{K}\sum_{k=1}^{K} H_k(w) &= \frac{1}{K}\sum_{k=1}^{K}\sum_{x\in\mathcal{X}} \mathbb{P}[\boldsymbol{x}_{k,n}=x]\log\frac{1}{\widehat{Q}(\Theta_{c_k}|x;w)}\\ &\overset{\text{Eq. (14)}}{=} \frac{1}{K}\sum_{k=1}^{K}\sum_{x\in\mathcal{X}}\sum_{\theta\in\Theta_{c_k}} \pi(\theta|\Theta_{c_k})\ell(x|\theta)\log\frac{1}{\widehat{Q}(\Theta_{c_k}|x;w)}\\ &\overset{\text{(a)}}{=} \frac{1}{K}\sum_{x\in\mathcal{X}}\sum_{c=1}^{C} K(c)\log\frac{1}{\widehat{Q}(\Theta_c|x;w)}\sum_{\theta\in\Theta_c}\pi(\theta|\Theta_c)\ell(x|\theta)\\ &\overset{\text{Eq. (32)}}{=} \sum_{x\in\mathcal{X}}\sum_{c=1}^{C}\mathbb{P}[\boldsymbol{x}=x,\boldsymbol{\theta}\in\Theta_c]\log\frac{1}{\widehat{Q}(\Theta_c|x;w)} = H(w),\end{aligned} \tag{33}$$

where in (a) the summation over k is transformed into a summation over c by counting how many agents k correspond to a cluster c. ■

We can now summarize our decentralized fusion of expert (DeFoE) strategy. Assume that training has been performed until time $i_{\max}$, and denote by $w_k^\star$ the realization of the parameter vector $\boldsymbol{w}_{k,i}$ estimated by agent k at time $i=i_{\max}$. At the end of the training phase, each agent k has learned the posterior pmf $\widehat{Q}_k(\Theta_c|x;w_k^\star)$. Then agent k can use the learned pmf, along with the decision models for the individual clusters from (22), to build its own approximation of the global posterior (18), which results in

$$\widehat{p}^{(k)}(\theta|x)=\sum_{c=1}^{C}\widehat{p}_c^{(k)}(\theta|x)\widehat{Q}_k(\Theta_c|x;w_k^\star), \qquad \theta\in\Theta_{\mathsf{net}}. \tag{34}$$

We will illustrate in the next section how this strategy performs over a benchmark dataset of real images.

4. ILLUSTRATIVE EXAMPLES

We present the results of some experiments conducted over the CIFAR-10 dataset [15]. We extracted from this dataset the images corresponding to cars and trucks. Specifically, we have 5000 images for each class. Each image has 32×32 pixels, with intensity values between 0 and 1, for the three RGB channels. Then we split the resulting dataset uniformly into two datasets obtained by changing the saturation level. Specifically, in one dataset the saturation is set to 1.75, in the other to 0.25. In terms of our notation, this corresponds to two hypothesis sets

$$\begin{aligned}\Theta_1 &= \{\text{car sat. } 1.75\text{ , truck sat. } 1.75\},\\ \Theta_2 &= \{\text{car sat. } 0.25\text{ , truck sat. } 0.25\},\end{aligned} \tag{35}$$

with $\Theta_{\mathsf{net}}=\Theta_1\cup\Theta_2$. Finally, the two datasets are distributed across $K=10$ agents. Specifically, the dataset pertaining to Θ_1 is partitioned into 5 disjoint datasets assigned to agents $1,2,3,4$, and 5. Likewise, the dataset pertaining to Θ_2 is distributed to agents $6,7,8,9$, and 10. As a result, each agent owns 1000 labeled images.

Each agent has learned a decision model to distinguish cars from trucks within its own dataset. These models have been constructed as follows. All agents consider a sophisticated deep learning model for image classification, namely, the Google vision transformer available from [16], further trained on the CIFAR-10 dataset [1]. To allow for some variability across the clusters, we perform a fine tuning of the aforementioned deep learning model over the datasets corresponding to Θ_1 and Θ_2. The classification performance achieved by each agent to distinguish cars from trucks, relative to the individual classification problems pertaining to Θ_1 and Θ_2, is shown in the first two rows (in yellow) of Table 1.

[1]The final model obtained by performing additional training on the CIFAR-10 dataset are available at `https://huggingface.co/edadaltocg/vit_base_patch16_224_in21k_ft_cifar10`

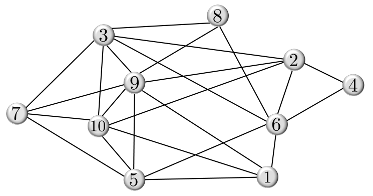

Fig. 1. Network topology used in the examples. Each node has a self-loop, not shown in the picture.

The goal of the agents is now to distinguish also the image saturation. Note that this classification task is significantly simpler than the task of distinguishing cars from trucks. For this reason, it would be wasteful to re-train the classifiers over the entire hypothesis set Θ_{net}. This is why we now exploit the DeFoE strategy to learn the posterior $Q(\Theta_c|x)$ only. To this end, we implement a regularized logistic model [1]. The decision functions $\delta(x,\Theta_c;w)$ are chosen as follows. Given an image x, represented as a collection of 32×32 matrices, X_1, X_2 and X_3 (corresponding to channels R, G, and B, respectively), we compute first the empirical mean-square deviations between channels i and j, for $i,j=1,2,3$:

$$y_{ij}(x)=\frac{1}{32\times 32}\sum_{p_1=1}^{32}\sum_{p_2=1}^{32}\Big(X_i(p_1,p_2)-X_j(p_1,p_2)\Big)^2, \tag{36}$$

where $X_i(p_1,p_2)$ denotes entry (p_1,p_2) of matrix X_i. Then, we aggregate these values and compute a scalar feature

$$\gamma\times\Big(y_{12}(x)+y_{13}(x)+y_{23}(x)\Big), \tag{37}$$

where $\gamma>0$ is a hyperparameter that can be tuned to speed-up the convergence of the logistic regression model. Finally, we add a dummy feature equal to 1 and obtain the 2×1 vector

$$z(x)=[z_1(x),z_2(x)]^\top=\Big[\gamma\times\Big(y_{12}(x)+y_{13}(x)+y_{23}(x)\Big),1\Big]^\top. \tag{38}$$

The dummy feature is useful to incorporate a threshold into the logistic regression classifier. The decision function for the logistic regression model is finally constructed as

$$\delta(x,\Theta_1;w)=w^\top z(x), \quad \delta(x,\Theta_2;w)=0, \tag{39}$$

where w is a 2×1 vector.

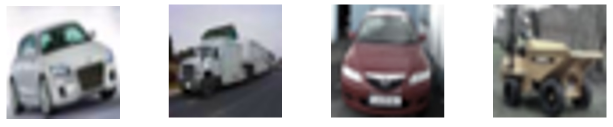

Fig. 2. Sample images used in the experiments.

The network graph is shown in Fig. 1. The graph is equipped with a combination matrix obtained with the Metropolis rule [9].

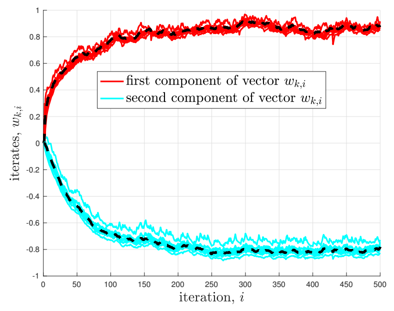

Fig. 3. Convergence of the iterates for the DeFoE strategy. In this plot, the hyperparameter γ is equal to 50.

Strategy	Accuracy
Classify $\theta \in \Theta_1$	99.1%
Classify $\theta \in \Theta_2$	97.5%
Classify Θ_1 vs. Θ_2 (γ=10)	87.17%
Classify Θ_1 vs. Θ_2 (γ=50)	93.02%
Classify Θ_1 vs. Θ_2 (γ=100)	93.76%
Classify $\theta \in \Theta_{\text{net}}$, uniform merging	49.6%
Classify $\theta \in \Theta_{\text{net}}$ (γ=10), DeFoE	85.69%
Classify $\theta \in \Theta_{\text{net}}$ (γ=50), DeFoE	91.49%
Classify $\theta \in \Theta_{\text{net}}$ (γ=100), DeFoE	92.29%

Table 1. Summary of performance. For the DeFoE strategy, the performance is averaged over all agents. Different colors highlight the following scenarios: yellow=classification within the individual (separate) clusters; cyan=classification between clusters; red=overall classification with uniform merging of the classifiers; purple=DeFoE strategy.

The step-size is set to $\mu = 0.1$, and the regularization parameters are set to $\rho_k = 0.1$ for $k \in [K]$. Figure 3 shows the evolution over subsequent iterations of the parameter estimated by all agents. We see that the estimates obtained by the agents in a distributed manner converge in a close neighborhood of the estimate obtained by the centralized system (dashed line). The classification performance to distinguish Θ_1 from Θ_2 is reported in the rows of Table 1 displayed in cyan, which correspond to different values for the parameter γ.

The classification performance over the entire set Θ_{net} achieved by the DeFoE strategy is reported in the rows of Table 1 displayed in purple. This performance is compared against the naïve strategy that simply maximizes the $p_c(\theta|x)$ without accounting for the weighting functions $Q(\Theta_c|x)$ (red row in the table). The latter strategy is almost equivalent to a classifier that, while guessing the type of vehicle, chooses randomly between the two clusters, as its accuracy is on the order of 50%. In comparison, with the DeFoE strategy, the average accuracy across the agents goes from 85.69% to 92.29%, depending on the value of γ, i.e., depending on the performance achieved by the decentralized strategy in estimating $Q(\Theta_c|x)$.

5. CONCLUSION AND FUTURE WORK

Starting from the optimal Bayesian classification rule, we have shown how to devise a decentralized strategy (the DeFoE strategy) that completes the knowledge of a set of distributed experts, infusing them with the incremental knowledge necessary to solve the overall classification task that aggregates their own individual hypotheses. There are several useful extensions and aspects that might be worth of investigation. For example, in this article we tested the method on a problem where the experts were initially trained to solve the same task, e.g., distinguishing cars from trucks. Then, the hypothesis sets of distinct agents were further labeled, giving rise to new classes. However, the DeFoE strategy can be applied to more general settings, where the initial experts solve distinct classification problems. Another aspect concerns the generalization to overlapping clusters.

6. REFERENCES

[1] A. H. Sayed, *Inference and Learning from Data*, 3 vols., Cambridge University Press, 2022.

[2] L Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.

[3] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *J. Comput. Syst. Sci.*, vol. 55, pp.119–139, Aug. 1995.

[4] V. Bordignon, V. Matta, and A. H. Sayed, "Adaptive social learning," *IEEE Trans. Inf. Theory*, vol. 67, no. 9, pp. 6053–6081, 2021.

[5] V. Bordignon, S. Vlaski, V. Matta and A. H. Sayed, "Learning from heterogeneous data based on social interactions over graphs," *IEEE Trans. on Inf. Theory*, vol. 69, no. 5, pp. 3347–3371, 2023.

[6] L. Bruzzone and D. F. Prieto, "An incremental-learning neural network for the classification of remote-sensing images," *Pattern Recognit. Lett.*, vol. 20, nos. 11–13, pp. 1241–1248, 1999.

[7] C. P. Diehl and G. Cauwenberghs, "SVM incremental learning, adaptation and optimization," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2003, pp. 2685–2690.

[8] X. Wei, B. -H. Juang, O. Wang, S. Zhou and G. Y. Li, "Accretionary learning with deep neural networks with applications," *IEEE Transactions on Cognitive Communications and Networking*, vol. 10, no. 2, pp. 660-673, April 2024.

[9] A. H. Sayed, "Adaptation, Learning, and Optimization over Networks," *Found. Trends Mach. Learn.*, vol. 7, no. 4-5, pp. 311–801, 2014.

[10] A. H. Sayed, "Adaptive networks," *Proceedings of the IEEE*, vol. 102, no. 4, pp. 460–497, Apr. 2014.

[11] J. Chen and A. H. Sayed, "On the learning behavior of adaptive networks — part I: Transient analysis," *IEEE Trans. Inf. Theory*, vol. 61, no. 6, pp. 3487–3517, Jun. 2015.

[12] J. Chen and A. H. Sayed, "On the learning behavior of adaptive networks — part II: Performance analysis," *IEEE Trans. Inf. Theory*, vol. 61, no. 6, pp. 3518–3548, Jun. 2015.

[13] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge University Press, 2012.

[14] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley, 1991.

[15] A. Krizhevsky, V. Nair and G. Hinton, "Learning Multiple Layers of Features from Tiny Images," 2009 [Online]. Available: https://www.cs.toronto.edu/ kriz/cifar.html.

[16] A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," available online as arXiv:2010.11929 [cs.CV].