Distributed Learning for Autonomous Vehicles in Communication-Constrained Environments

Nicole Cruz[1], Jose Fuentes[1] and Leonardo Bobadilla[1]

***Abstract*— Intelligent Autonomous Systems (IAS), which encompass Unmanned Aerial Systems (UAS), Unmanned Surface Vehicles (USV), and Unmanned Underwater Vehicles (UUV), are set to play a pivotal role across multiple domains. However, due to the demanding nature of their missions (e.g., disaster response and underwater exploration), common communication methods are often limited. This makes it essential to use efficient communication strategies for learning and decision-making. This paper introduces a novel approach, leveraging distributed machine learning algorithms along with data projection to achieve sublinear communication costs. Particularly, we incorporate Convolutional Neural Networks for feature extraction prior to projection, significantly enhancing efficiency and accuracy in autonomy tasks within bandwidth-constrained environments. This method uses the Johnson-Lindenstrauss transform for dimensionality reduction of both data and model parameters, facilitating efficient inter-agent communication by transmitting only projected model parameters. We further outline theoretical underpinnings showcasing the Johnson-Lindenstrauss transform's effectiveness in maintaining convergence properties for Online Gradient Descent algorithms. Through experimental validation using image recognition tasks relevant to maritime surveillance, our approach demonstrates substantial reductions in communication cost while preserving high model performance.**

***Index Terms*— Distributed algorithms, Convolutional neural networks, Machine learning, Maritime communication**

I. INTRODUCTION

Communication challenges for networks of autonomous vehicles often arise in the maritime domain. As the use of underwater autonomous vehicles continues to rise, they become vital for operations such as infrastructure monitoring, mine countermeasures, and environmental conservation. Due to the limitations of conventional communication methods like radio frequency and acoustics in underwater settings, these missions face communication barriers that negatively affect their performance and efficiency. Additionally, communication bandwidth can be affected in adversarial scenarios such as jamming and blocking for ground and aerial groups of autonomous vehicles.

Many maritime surveillance tasks, such as traffic monitoring, illegal fishing, border smuggling detection, and pollution management, rely heavily on computer vision techniques. Studies [1] and [2] show how Deep Learning techniques–specifically, Convolutional Neural Networks (CNNs)–have proved invaluable for image and satellite-based ship detection and classification. Utilizing CNNs combined with the k-Nearest Neighbor (kNN) method trained with aerial images of ships, [1] achieves a classification success rate of over 90%, surpassing traditional methods and other CNN-based strategies. Findings reported in [2] offer similar insight, instead using YOLOv3, YOLOv4, and YOLOv5 architectures to detect ships within two datasets.

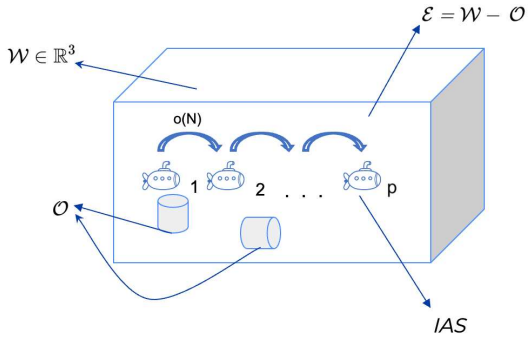


Fig. 1: Visual representation of IAS in a distributed network; each IAS can communicate small amounts of data and they navigate within a region of interest $\mathcal{W}$.

Using a distributed architecture allows for improved scalability and robustness. If an individual machine fails, the rest of the network can continue to operate and process data. It is known that, in communication-constrained environments, the complexity of communication typically outweighs the complexity of learning. Therefore, it is still favorable to distribute the learning amongst various agents, regardless of potentially higher learning costs.

This study introduces a novel approach centered around a distributed machine learning (ML) algorithm leveraging the Johnson-Lindenstrauss (JL) transform. The main idea is to utilize this transform for both data and model parameters, effectively projecting them into a lower dimension. This ensures the learning process operates within a reduced dimensional space, facilitating efficient communication between agents by transmitting only the projected model parameters. Our focus lies particularly on CNN architectures due to their relevance in computer vision tasks and their adaptability through transfer learning to other domains. Building upon the work presented in [3], we employ CNNs as feature extractors. Initially, raw data is preprocessed via a pre-trained CNN to extract essential features, reducing dimensionality and minimizing communication overhead. This step acts as a preliminary projection, significantly trimming down the number of features that need to be processed by the system.

[1]N. Cruz and J. Fuentes, and L. Bobadilla are with the School of Computing and Information Sciences, Florida International University, Miami, FL 33199, USA {ncruz071@,jfuent099@,bobadilla@cs.}fiu.edu

This work is supported in part by NSF grants IIS-2034123, IIS-2024733, IIS-2331908, the Office of Naval Research grant N00014-23-1-2789, U.S. Dept. of Homeland Security grant 23STSLA00016-01-00.

However, the resulting feature vectors are still fairly large, keeping the same order as the original data, so we must perform a projection on the feature data to ensure sublinear communication costs.

Furthermore, we provide theoretical insights into how the JL transform guarantees certain convergence properties for Online Gradient Descent (OGD). Specifically, we examine the convergence behavior of models trained using raw data versus those trained with projections from the JL transform. These theoretical underpinnings underscore the efficacy of this approach.

In addition to theoretical analysis, we implement a distributed OGD algorithm tailored for training ML models, particularly focusing on image recognition tasks pertinent to oceanic surveillance. Our approach extends previous methodologies applied to both neural networks (NNs) and CNNs. Notably, while our results are demonstrated within the context of OGD, they are expected to generalize to other optimization algorithms, such as Stochastic Gradient Descent (SGD) and Gradient Descent (GD) directly.

II. RELATED WORK

A. Communication Complexity

The work presented in this paper is based on Communication Complexity [4], [5], [6], a subarea of Theoretical Computer Science that tries to understand the amount of communication required to solve a problem when the input to the problem is distributed among two or more parties. In a multi-robot system, especially where a large number of robots are present and the amount of bandwidth is limited, reducing the size of the communicated messages can tremendously benefit the coordination process. The study presented in [3] provides a framework for sequential data analysis that guarantees sublinear communication costs. Utilizing the JL lemma, we embed the data to be communicated into a lower dimensional space, which dramatically diminishes the amount of data to be communicated. A key component of our contribution is the employment of a CNN for the initial feature extraction. Employing CNNs enables us to concentrate on processed features instead of the high-dimensional raw data, significantly diminishing communication complexity and improving the system's overall efficiency in bandwidth-constrained scenarios. Our ideas are also connected to the use of communication complexity in multi-robot systems [7], [8].

B. The Johnson-Lindestrauss Lemma and its Construction

The JL lemma asserts that for a set of points $X \subseteq \mathbb{R}^N$ and $0 < \epsilon < 1$, there is a dimension $n = O(\epsilon^{-2}\log(|X|))$ and a linear transformation $T : \mathbb{R}^N \longrightarrow \mathbb{R}^n$ such that

$$(1-\epsilon)||x-y|| \leq ||T(x)-T(y)|| \leq (1+\epsilon)||x-y||. \quad (1)$$

Various bounds exist to calculate n. For instance, [9] shows that it is enough to satisfy

$$n \geq \frac{4\ln(N)}{\epsilon^2/2 - \epsilon^3/3} \quad (2)$$

to ensure the existence of T.

Complementing the lemma, there are numerous efforts to give random, sparse, and deterministic constructions of the transformation T.

1) Random JL Transforms: The first efforts to do transformations for dimensionality reduction relied on building random matrices where each matrix entrance is drawn i.i.d. from a normal distribution $\mathcal{N}(0, 1/n)$; [9], [10], [11] prove its properties as a JL transform and show its benefits for ML applications.

2) Sparse JL Transforms: These are variations that construct the embedding using sparse matrices, significantly reducing the computational complexity of applying the transform, especially useful for sparse high-dimensional data. These transformations operate by setting most of the T's matrix representation values to zero; in [12], the distribution for each entry of the matrix $[M_T]_{ij}$ is considered as

$$[M_T]_{ij} = \begin{cases} \frac{1}{\sqrt{n}} \text{ with probability } 1/6 \\ 0 \text{ with probability } 2/3 \\ -\frac{1}{\sqrt{n}} \text{ with probability } 1/6. \end{cases} \quad (3)$$

3) Deterministic Versions: While the original lemma and many of its extensions rely on random constructions, there have been efforts to develop deterministic versions of the lemma, though these often come with trade-offs in bounds or applicability. Work presented in [13] replaces the random variables used to generate the matrix entrances with hash functions that can perform the same task; moreover, these matrices can be built so that they can keep a certain degree of sparsity.

C. Efficient Federated Learning

Our work aligns with [14], which presents structured and sketched updates to improve communication efficiency in federated learning. Federated learning is a specific type of distributed learning focused on training algorithms across decentralized devices or servers holding local data samples without exchanging them. Structured updates limit the changes to the model by restricting them to a predefined space, thus reducing the number of variables that need to be communicated. This method can involve selecting a subset of the model's parameters to update or using a low-rank approximation. Sketched updates, on the other hand, involve compressing the entire model update (for example, through sparsification) before it is sent to the server. This approach reduces the size of the data transmitted by only sending the most crucial information for updating the model. It is also important to note that many federated learning applications, detailed in [15], focus on improving and maintaining security. Even when bandwidth is not limited, it is crucial to ensure secure communication between agents, especially when handling sensitive data.

D. Convolutional Neural Networks

Traditional NNs are not ideally suited for processing high-resolution images, as they often require a large number of parameters. This not only increases the computational

complexity and the time required to train the model but also heightens the risk of over-fitting, especially if the training data is limited or not diverse enough, as may be the case in real-world scenarios. CNNs are preferred for image classification tasks to reduce communication complexity because they efficiently process and extract features directly from image pixels with minimal pre-processing [16]. Their architecture leverages spatial hierarchies, significantly reducing the amount of data that needs to be communicated for learning. By automatically learning feature representations, CNNs eliminate the need for manual feature extraction and, thus, reduce the dimensionality and complexity of data, leading to more efficient communication in distributed or federated learning environments.

III. PROBLEM FORMULATION

Given the aforementioned limitations inherent in underwater environments, we strive to solve tasks for autonomous systems using a restrictive bandwidth. Particularly in learning processes, when considering datasets or model parameters with $O(N)$ bits, we would like to send $o(N)$ bits among the agents so the autonomy task can be learned and executed in sublinear time. In this scenario, we allow the group's agents to agree on a strategy before starting the learning process.

We consider a group of p agents navigating within a d-dimensional space ($d = 2, 3$), denoted as $\mathcal{W} \subseteq \mathbb{R}^d$. This workspace is partitioned into the obstacle space $\mathcal{O}$ and the free space $\mathcal{E} = \mathcal{W}/\mathcal{O}$. The agents naturally traverse through $\mathcal{E}$ while avoiding $\mathcal{O}$, collecting and processing samples, each of size $O(N)$ bits. Given limited communication capabilities, which are expected to be small compared to the sample size, each agent independently constructs their part of the joint dataset D_i for $i = 1, \ldots, p$ to be used in a distributed learning algorithm. Consequently, the joint dataset is represented as the disjoint union $\mathcal{D} = \cup_{i=1}^{p} D_i$.

Problem: Solving autonomy tasks in sublinear time Given a set of p agents navigating in $\mathcal{W}$, learn and execute distributed ML-based autonomy tasks using $o(N)$ communication complexity.

IV. METHODS

Our scenario of interest encompasses the p agents moving throughout an environment $\mathcal{W}$. They take in data via sensors (e.g. camera, sonar, acoustic); the data is treated as samples of size $O(N)$ bits and acquired independently by each machine a_i to build their part of the dataset $D_i = (X_i, y_i)$. Each machine can send or broadcast data to the rest of the machines in the network. We aim to extend the results presented in [5] since they provide a theoretical framework for distributed ML using sublinear communication to train classic ML models using GD or its variants like SGD and OGD.

A. Johnson-Lindenstrauss Transform and Online Gradient Descent

This transformation is useful when working with long feature vectors where the number of samples is in the same order or bigger than the number of features, i.e. $|X| = O(N)$. This enables the projection of data while nearly preserving the distances between the points in X, allowing us to examine two connected optimization problems simultaneously.

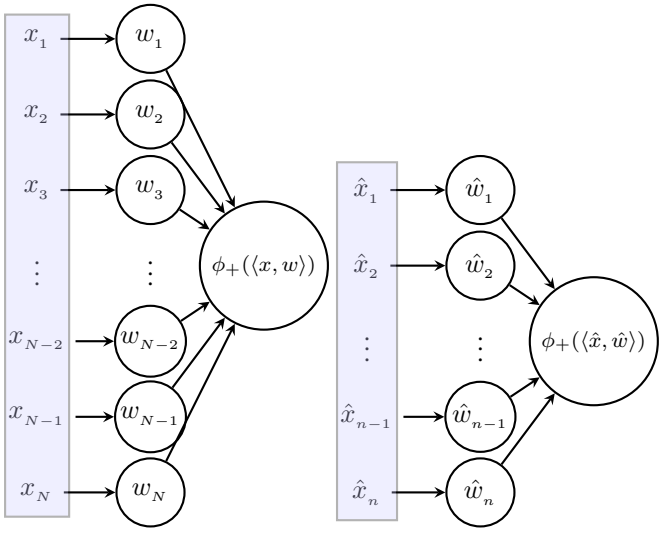
Consider a scenario where we have a set of features $X \subseteq \mathbb{R}^N$ along with their corresponding labels $Y \subseteq \mathbb{R}^P$, which we will use to train an ML model M. We denote the loss function for this model as $\ell(w; (X, y))$. Let T be a JL transform that maps X onto $T(X)$, allowing us to consider the same training problem by replacing X with $T(X)$. This involves adjusting the ML architecture and the loss function to handle features of size n instead of size N. Given an ML model M originally trained on features of size N, we define a *surrogate model* of M represented by $\hat{\mathrm{M}}$; this surrogate architecture is built by adapting M's architecture (often, by shrinking it) to handle features of size n while keeping its structure. The surrogate model concept is illustrated in Fig 2. To maintain the same training structure, we need to define an adaptation of the loss function ℓ to be used on $\hat{\mathrm{M}}$. We define this *surrogate loss function* $\hat{\ell}(\hat{w}; (T(X), y))$ as the result of adapting ℓ to $\hat{\mathrm{M}}$. Despite operating on the projected dataset $T(X)$ instead of the raw dataset X, $\hat{\ell}$ maintains analogous behavior to ℓ. Certainly, the labels remain unaltered in both instances. Here, $\hat{w}$ corresponds to the parameters to be found when optimizing the loss function $\hat{\ell}$ defined on the dataset $T(X)$.

Furthermore, to establish a clear relationship between w and $\hat{w}$, we demonstrate that a JL transformation defined on the feature space $T_F : \mathbb{R}^N \longrightarrow \mathbb{R}^n$ implies the existence of a JL transformation in the parameter (weight) space, $T_W : \mathbb{R}^M \longrightarrow \mathbb{R}^m$ that maps w into $\hat{w}$. This transformation can potentially be applied in the case of more complex models such as NNs or CNNs. In this domain, when the feature vector dimension is N, a typical NN exhibits a weight vector w of size $M = O(N^{n_l})$, where n_l is the number of layers in the NN. Analogously, a surrogate NN trained using the projected feature vectors in $T_F(X) \subseteq \mathbb{R}^n$ has a weight vector of size $m = O(n^{n_l}) = O(\log(N)^{n_l})$, making it larger than the smallest dimension w can be projected down to, which is $O(n_l \log(N))$. This observation underscores the feasibility of projecting weight vectors via a JL transform T_W.

In this section, our primary goal is to provide theoretical reasoning that demonstrates the regret convergence rate of optimizing ℓ using OGD is approximately equivalent to that of $\hat{\ell}$. To do so, let us define the *regret function* for a loss function $l(w; (X, y))$ as

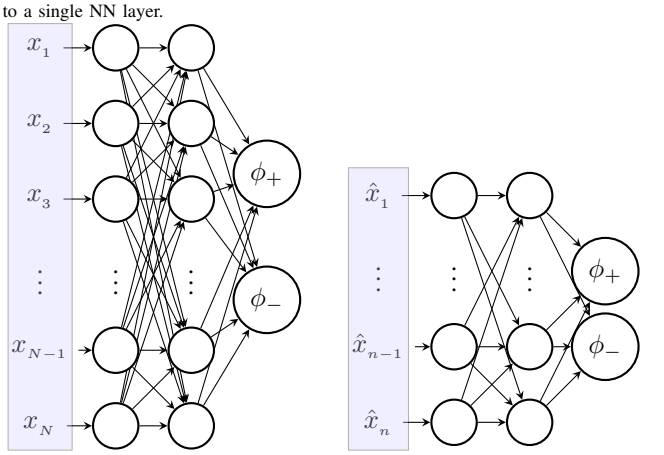
$$R_l(w) = l(w; (X, y)) - l(w^\star; (X, y)) \tag{4}$$

where $w^\star = \arg\min_{w \in \mathbb{R}^N} l(w; (X, y))$. To avoid excessive notation we will drop the dataset $D = (X, y)$ from the notation on the loss functions i.e. $l(w) := l(w; (X, y))$ for any loss function l. Furthermore, we leverage the notation for a loss function and its surrogate loss function to $\ell(w) := \ell(w; (X, y))$ and $\hat{\ell}(\hat{w}) := \hat{\ell}(\hat{w}; (T_F(X), y))$, respectively. From now on, we will assume that the relationship between a variable z and $\hat{z}$ will be $\hat{z} = T_W(z)$ or $\hat{z} = T_W(z)$ depending



(a) Logistic regression model trained using raw data of size N. This is equivalent to a single NN layer.

(b) Surrogate logistic regression model trained using projected data of size n.



(c) Deep NN model, trained with projected data of size N.

(d) NN surrogate model trained with projected data of size n.

Fig. 2: ML models for binary classification and their surrogate models.

on whether z is an element of the parameter space or the feature space.

Using the regret definition, we study the relationship between $R_\ell(w)$ and $R_{\hat{\ell}}(w)$, which is presented in Theorem 1. Before that, we need to determine a bound for the angles formed between the gradient of a loss function at a certain value w_t and the vector $d_t^\star = w_t - w^\star$, which is the vector pointing to the optimal value when executing OGD at time t.

Lemma 1: Let $f(w) = \frac{1}{2}w^\top Aw + h(w)$ be a scalar function such that A is a positive-definite matrix and there is an open neighborhood V containing the origin such that $||\nabla h(w)|| \leq \epsilon||w||$. Then, we have that

$$\frac{|\langle \nabla f(w), w\rangle|}{||\nabla f(w)||\ ||w||} \geq \frac{\lambda_{min} - \epsilon}{\lambda_{max} + \epsilon} = c > 0 \tag{5}$$

for $w \in V$, λ_{min} and λ_{max} are the minimum and maximum eigenvalues of A and $\epsilon < \lambda_{min}$.

Proof: Computing the inner product and using triangle inequalities gives

$$\begin{aligned}\frac{|\langle \nabla f(w), w\rangle|}{||\nabla f(w)||\ ||w||} &= \frac{|w^\top Aw + \nabla h(w)w|}{||Aw + \nabla h(w)||\ ||w||} \\ &\geq \frac{||w^\top Aw| - |\nabla h(w)w||}{||Aw + \nabla h(w)||\ ||w||} \\ &\geq \frac{||w^\top Aw| - |\nabla h(w)w||}{(||Aw|| + ||\nabla h(w)||)||w||},\end{aligned} \tag{6}$$

and using the bounds for positive-definite matrices, h, and the spectral norm we obtain

$$\begin{aligned}\frac{||w^\top Aw| - |\nabla h(w)w||}{(||Aw|| + ||\nabla h(w)||)||w||} &\geq \frac{\lambda_{min}||w||^2 - \epsilon||w||^2}{(||Aw|| + ||\nabla h(w)||)||w||} \\ &\geq \frac{\lambda_{min}||w||^2 - \epsilon||w||^2}{(\lambda_{max}||w|| + \epsilon||w||)||w||} \\ &= \frac{\lambda_{min} - \epsilon}{\lambda_{max} + \epsilon} = c\end{aligned} \tag{7}$$

as desired. ■

Lemma 1 tells us that the angle between the gradient and the vector w of the canonical form $\frac{1}{2}w^\top Aw$ must be acute.

In our main theorem, we seemingly use the Cauchy-Schwartz inequality in reverse order to bound $||\nabla f(w)||\ ||w||$ using $\frac{1}{c}|\langle \nabla f(w), w\rangle|$ for a certain constant c (which turns out to be the angle between $\nabla f(w)$ and w). In general, it is not possible to do so, since c might be too close to zero or zero directly, indicating that both vectors are orthogonal. Lemma 1 prevents this from happening, allowing us to prove and use Lemma 2.

Lemma 2: Let $l(w)$ be a loss function that is minimized using OGD, and let $w^\star$ and $d_t^\star$ be defined as before. Suppose that OGD generates a convergent sequence of iterations $\{w_t\}_{t\in\mathbb{N}}$ with $\lim_{t\to\infty} d_t^\star = 0$. Then, there is a constant $c > 0$ and a natural number T such that

$$||\nabla l(w_t)||\ ||d_t^\star|| \leq \frac{1}{c}|\langle \nabla l(w_t), d_t^\star\rangle| \tag{8}$$

for $t \geq T$.

Proof: Expand $l(w)$ towards $w^\star$ using its Taylor's series representation and the fact that $\nabla l(w^\star) = 0$

$$l(w_t) = l(w^\star) + {d_t^\star}^\top \frac{H_l(w^\star)}{2} d_t^\star + h_l(d_t^\star)||d_t^\star||. \tag{9}$$

Where $H_l(w^\star)$ is the l's Hessian matrix evaluated at $w^\star$. Finally, since $\lim_{d_t^\star \to 0} h_l(d_t^\star) = 0$, $\{w_t\}_{t\in\mathbb{N}}$ is convergent and $H_l(w^\star)$ is positive definite, choose T such that $|h_l(d_t^\star)| \leq \epsilon$ and apply Lemma 1 on the function $l(w_t) - l(w^\star)$ using $d_t^\star$ as its variable. ■

Now, we present our main result, supported by the application of Lemma 2.

Theorem 1: Let $X \subseteq \mathbb{R}^N$ with $|X| \leq N$, consider T_F and T_W: two JL transforms satisfying (1) for the feature and parameter spaces, respectively. ℓ is a loss function and $\hat{\ell}$ is a surrogate loss function. Suppose that ℓ and $\hat{\ell}$ are L-bi-Lipschitz and $\hat{L}$-bi-Lipschitz respectively, and they are

minimized using OGD. Then, there are two constants k and K and two sequences s_t and S_t such that

$$kR_{\hat{\ell}}(\hat{w}_t) + s_t \leq R_\ell(w_t) \leq KR_{\hat{\ell}}(\hat{w}_t) + S_t, \tag{10}$$

with $\lim_{t\to\infty} s_t = \lim_{t\to\infty} S_t = 0$.

Proof: Let $\Delta w_t = w_t - w_{t-1}$ and $d_t^\star$ as in Lemma 2. We expand $\ell(w)$ around w_t to estimate $\ell(w^\star)$

$$\ell(w^\star) = \ell(w_t) - \nabla\ell(w_t)\cdot d_t^\star - r_\ell(d_t^\star), \tag{11}$$

r_ℓ is the error in the Taylor's expansion. We bring the regret definition for ℓ and consider a finite difference scheme to approximate $\nabla\ell(w_t)$

$$\begin{aligned} R_\ell(w_t) &= \nabla\ell(w_t)\cdot d_t^\star + r_\ell(d_t^\star) \\ &= \frac{\ell(w_t) - \ell(w_{t-1})}{||\Delta w_t||}\Delta w_t \cdot d_t^\star + d_\ell(\Delta w_t)\cdot d_t^\star + r_\ell(d_t^\star). \end{aligned} \tag{12}$$

Here, r_ℓ and d_ℓ are the remainders after doing Taylor's approximation and the finite scheme, respectively. We bound $R_\ell(w_t)$ using the triangle inequality, the Cauchy-Schwartz inequality, and the Lipschitz condition on ℓ

$$\begin{aligned} R_\ell(w_t) \leq& \frac{||\ell(w_t) - \ell(w_{t-1})||}{||\Delta w_t||}||\Delta w_t||\ ||d_t^\star|| + u_\ell(\Delta w_t, d_t^\star) \\ \leq& L\frac{||\Delta w_t||^2}{||\Delta w_t||}||d_t^\star|| + u_\ell(\Delta w_t, d_t^\star). \end{aligned} \tag{13}$$

where $u_\ell(\Delta w_t, d_t^\star) = ||d_\ell(\Delta w_t)||\ ||d_t^\star|| + |r_\ell(d_t^\star)|$. We make usage of the JL transform property stated in (1) to relate R_ℓ and $R_{\hat{\ell}}$, transforming Δw_t into $\Delta\hat{w}_t$ and $d_t^\star$ into $\hat{d}_t^\star$. As a result, (13) becomes

$$R_\ell(w_t) \leq \frac{(1+\epsilon)^3 L}{(1-\epsilon)}\frac{||\Delta\hat{w}_t||^2}{||\Delta\hat{w}_t||}||\hat{d}_t^\star|| + u_\ell(\Delta w_t, d_t^\star). \tag{14}$$

Now, we apply the same logic for $\hat{\ell}$ to obtain

$$\begin{aligned} R_\ell(w_t) \leq& \frac{(1+\epsilon)^3 L}{(1-\epsilon)\hat{L}}\frac{||\hat{\ell}(\hat{w}_t) - \hat{\ell}(\hat{w}_{t-1})||}{||\Delta\hat{w}_t||}||\Delta\hat{w}_t||\ ||\hat{d}_t^\star|| \\ &+ u_\ell(\Delta w_t, d_t^\star) \\ \leq& \frac{(1+\epsilon)^3 L}{(1-\epsilon)\hat{L}}||\nabla\hat{\ell}(\hat{w}_t)||\ ||\hat{d}_t^\star|| \\ &+ \frac{(1+\epsilon)^3 L}{(1-\epsilon)\hat{L}}||d_{\hat{\ell}}(\Delta\hat{w}_t)||\ ||\hat{d}_t^\star|| + u_\ell(\Delta w_t, d_t^\star). \end{aligned} \tag{15}$$

At this point, we apply Lemma 2, as it allows us to "reverse" the Cauchy-Schwartz inequality, providing a constant $c > 0$ such that

$$||\nabla\hat{\ell}(\hat{w}_t)||\ ||\hat{d}_t^\star|| \leq \frac{1}{c}|\langle\nabla\hat{\ell}(\hat{w}_t), \hat{d}_t^\star\rangle|. \tag{16}$$

We plug (16) into the last inequality to obtain

$$\begin{aligned} R_\ell(w_t) \leq& \frac{(1+\epsilon)^3 L}{c(1-\epsilon)\hat{L}}|\langle\nabla\hat{\ell}(\hat{w}_t), \hat{d}_t^\star\rangle| \\ &+ \frac{(1+\epsilon)^3 L}{(1-\epsilon)\hat{L}}||d_{\hat{\ell}}(\Delta\hat{w}_t)||\ ||\hat{d}_t^\star|| + u_\ell(\Delta w_t, d_t^\star) \\ =& \frac{(1+\epsilon)^3 L}{c(1-\epsilon)\hat{L}}|R_{\hat{\ell}}(\hat{w}_t) - r_{\hat{\ell}}(\hat{d}_t^\star)| \\ &+ \frac{(1+\epsilon)^3 L}{(1-\epsilon)\hat{L}}||d_{\hat{\ell}}(\Delta\hat{w}_t)||\ ||\hat{d}_t^\star|| + u_\ell(\Delta w_t, d_t^\star). \end{aligned} \tag{17}$$

$d_{\hat{\ell}}$ and $r_{\hat{\ell}}$ are the analogous counterparts for $\hat{\ell}$. Then, we split the terms we are interested in to obtain

$$\begin{aligned} R_\ell(w_t) \leq& \frac{(1+\epsilon)^3 L}{c(1-\epsilon)\hat{L}}R_{\hat{\ell}}(\hat{w}_t) + \frac{(1+\epsilon)^3 L}{(1-\epsilon)\hat{L}}||d_{\hat{\ell}}(\Delta\hat{w}_t)||\ ||\hat{d}_t^\star|| \\ &+ \frac{(1+\epsilon)^3 L}{c(1-\epsilon)\hat{L}}|r_{\hat{\ell}}(\hat{d}_t^\star)| + u_\ell(\Delta w_t, d_t^\star). \end{aligned} \tag{18}$$

Finally, we define $K = \frac{(1+\epsilon)^3 L}{c(1-\epsilon)\hat{L}}$ and S_t as the remainder part of the function that does not contain $R_{\hat{\ell}}(\hat{w}_t)$. Immediately, we conclude that $\lim_{t\to\infty} M_t = 0$ from the properties of the rest functions r_ℓ and $r_{\hat{\ell}}$ and the fact that Δw_t, $\Delta\hat{w}_t$, $d_t^\star$ and $\hat{d}_t^\star$ converge to zero. It is important to note that the difference scheme error functions d_ℓ and $d_{\hat{\ell}}$ do not necessarily converge to zero when their arguments do; however, since they are bounded functions, we may consider the aforementioned analysis. This proves the first inequality; the second can be proved similarly by swapping ℓ and $\hat{\ell}$ and applying the same logic. ■

Theorem 1 states that performing OGD on the raw dataset and on the projected dataset leads to roughly the same convergence rate, so we should see similar performance for both approaches. This result extends [5] to consider a more complex class of functions. For instance, classical loss functions defined for neural networks fall in this category since it is possible to make them fit into the Theorem 1 hypothesis without constricting either $M = N$ or $m = n$, i.e. making the size of w be the same as each feature vector x or their projections. In doing so, we can consider architectures where the vector sizes of x and w do not necessarily match. In exchange, we require that one JL transformation T_F be applied to the features x and another JL transformation T_W be applied to w. Again, note that [5] presents a special case of this by setting $T_F = T_W$.

B. Distributed Learning

Following [14], we apply the concepts of structured and sketched updates in federated learning to a distributed learning scenario. The CNN model acts as a feature extractor for classification, mimicking a structured update. The projection then acts as a sketched update, compressing the data to a lower dimension before it is used to train the final model.

Extending from [3], we construct a distributed network of agents in which each trains a portion of the model. Before projection, the raw data is processed using a predefined

CNN, which allows us to represent each image as a feature vector. To ensure low communication costs, each agent only broadcasts the updated model parameters to the next one in the sequence, and only the final agent acquires the robust model used for classification.

Algorithm 1: DISTRIBUTED LEARNING(M, $\mathcal{D}$)

Input: Set of agents $\mathcal{A} = \{a_1, \ldots, a_p\}$, CNN Headless Model List $\mathcal{M}$, Data Set $\mathcal{D} = \cup_{i=1}^{p}\{(X_i, y_i)\}$

Output: Trained optimal weights $\hat{w}^\star$, $w^\star$

1 a_1 determines which CNN model to use from $\mathcal{M}$
2 a_1 determines the minimum projection dimension n for the features using JL Lemma
3 a_1 determines the matching dimension m for the weights using JL Lemma and the surrogate model
4 a_1 initializes projected weights $\hat{w}_0$
5 a_1 chooses two seeds δ_F and δ_W to generate the JL transforms T_F and T_W
6 **for** $i = 2, \ldots, p$ **do**
7 a_1 broadcasts CNN model identity and both seeds to a_i
8 **for** $i = 1, 2, \ldots, p$ **do**
9 a_i sets its data set $D_i = (X_i, y_i)$
10 a_i passes X_i through the CNN, storing the resulting feature vectors $M(X_i)$
11 a_i projects data as $\hat{X}_i = T_F(M(X_i))$
12 a_i preforms OGD on $\hat{D}_i = (\hat{X}_i, y_i)$ and obtains updated weights $\hat{w}_i$ starting by $\hat{w}_{i-1}$
13 **if** $i < p$ **then**
14 a_i broadcasts $\hat{w}_i$ to a_{i+1}
15 a_p preforms reverse transform on $\hat{w}^\star = \hat{w}_p$ to obtain $w^\star = T_W^{-1}(\hat{w}^\star)$
16 **return** $\hat{w}^\star$, $w^\star$

Algorithm 1 sets up a distributed network in which each agent trains a classification model using the features extracted from the dataset via a CNN. It takes as inputs a set of agents $\mathcal{A}$, a list of headless CNN model types $\mathcal{M}$, and an image dataset $\mathcal{D}$. Theoretically, the dataset is provided and set for each machine; practically, each agent would collect data in real-time. In steps 1-3, the first machine in the network chooses a headless CNN from the given list and determines the minimum projection dimensions for the feature vectors and the weights. In step 5, the two seeds that will allow each agent to generate the appropriate JL transforms for both the data and the model weights are chosen. In step 7, the identity of the chosen CNN is broadcast to the rest of the agents along with the two seeds. Extrapolating [3], [13], the complexity of broadcasting the data projection seed is $O(\log(n/\epsilon)\log(N))$. The complexity of broadcasting the model weights projection seed is $O(\log(m/\epsilon)\log(M)) = O(\log(n/\epsilon)\log(N))$ if $n_l = O(1)$. This process can also be done simultaneously, in which case all agents agree on the CNN identity and the transformation seeds.

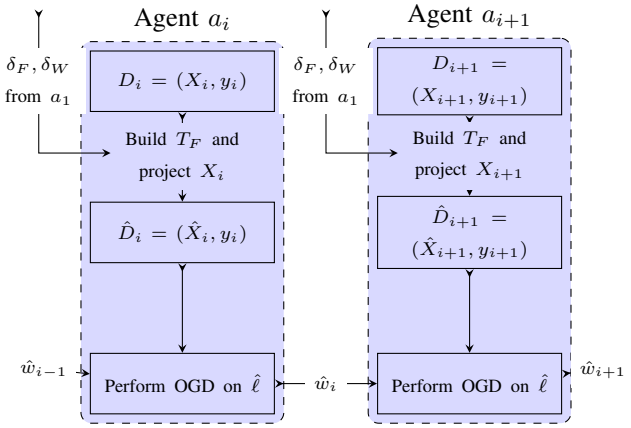


Fig. 3: Illustration of algorithm 1 between two consecutive agents; each agent receives the seeds and the projected weights and broadcasts the updated weights.

To determine the minimum dimension that the data can be projected down to whilst still preserving the distances between the data points, we utilize the JL Lemma proven in [9] and stated in (2). We project the data using the JL transform stated in [17]. Here, the seed controls which entries in the matrix are non-zero and their values. This is critical since the sparsity pattern can significantly affect the performance and quality of the dimensionality reduction.

Once all machines have agreed on the seed, step 10 shows each agent passing its assigned or collected data through the CNN and storing the resulting feature vectors. In step 11, each agent projects its assigned data down to the specified minimum projection dimension. In steps 12-14, OGD is performed on the data and the updated weights $\hat{w}_i$ are broadcast to the next agent in the sequence. This broadcasting step takes $O(\log(N)^{n_l})$ for each weight vector w_i. Finally, in steps 15-16, the final agent in the system performs the inverse transform on the model parameters and uses this to classify the original high-dimensional data. The communication complexity is composed of the cost of broadcasting the seeds δ_F and δ_W plus the cost of broadcasting the projected weights $\hat{w}_i$, leading to a total complexity of $O(\log(N)^{n_l} + \log(m/\epsilon)\log(M) + \log(n/\epsilon)\log(N))$; Figure 3 illustrates how the data is sent among two agents.

V. EXPERIMENTAL RESULTS

A. Overview

For our purposes, the datasets had been previously split into test and train sets. We use three CNN models: ResNet-50 [18], VGG16 [19] and EfficientNetB0 [20]. These state-of-the-art models were chosen for their efficiency, scalability, and predictability. Efficient execution ensures that the model can perform inferences quickly and with minimal computational resources. Scalable architectures can handle larger, more complex tasks without requiring disproportionate increases in communication. Each model is pre-trained and imported without the fully connected layers or "head". Pre-trained models save substantial training time and reduce

the amount of data transfer needed during deployment. Because we are primarily interested in reducing communication complexity, we only need to pass the data through the convolutional layers to flatten the data and can substitute the head for logistic regression, a simpler classification method. In practice, logistic regression has the same behavior as an NN with one classifying layer. After passing the data through the final convolutional layer the CNN architectures may still return fairly large feature vectors, which would be difficult to process without an additional projection. For example, VGG-16 returns a feature vector of size $512 \times 7 \times 7$.

Once the features are extracted from the raw data, they are shuffled and split among the 10 agents, which represent autonomous vehicles configured in a distributed fashion. Each agent does a standard preprocessing step to the data before using any CNN to extract the features. The calculation to determine the minimum dimension that the data can be projected down to is done using (2), which takes the number of samples and an error value $\epsilon = 0.2$ as inputs and returns the target dimension $n = 2125$.

When the final agent in the system is reached, it performs the inverse transform on the resulting model parameters and uses this model to classify the original high-dimension data. The last agent has access to both parameters, the projected $\hat{w}^\star$ and the inverse transformed $w^\star = T_W(\hat{w}^\star)$. Hence, it can make predictions either on the projected data by using $\hat{w}^\star$ or on the high-dimensional data by using $w^\star$; these two possibilities are reflected by the red and green boxplots in Figures 4 and 6, respectively. Moreover, we performed baseline experiments with no projection at all corresponding to the blue boxplots in Figures 4 and 6. For both datasets, the model trained with projected data outperforms or is on par with the baseline model. This is due to the projection itself, which makes the data more manageable for processing and helps stabilize the training process by reducing the variability of the model's performance across different runs. Each experiment is repeated 50 times, returning the mean and standard deviation of the prediction accuracies both on the projected and raw data.

B. CIFAR-10

To test the robustness of the model, we utilize the benchmark CIFAR-10 dataset [21] for testing. It consists of 60,000 color images, each with a resolution of 32×32 pixels, divided between 50,000 training images and 10,000 test images. The images are evenly distributed across 10 classes, which represent different objects and animals: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. For computational efficiency, we select a random subset of 10,000 training samples and 2,000 test samples from the dataset.

Our results, shown in figure 4, are comparable to other tests conducted on the CIFAR-10 dataset using the same CNN models [18] [19] [20]. Marginally lower accuracies can be explained by our use of logistic regression for classification over the original model head. Despite this, we show that the accuracy is preserved and even improved for the models trained on projected data for every architecture.

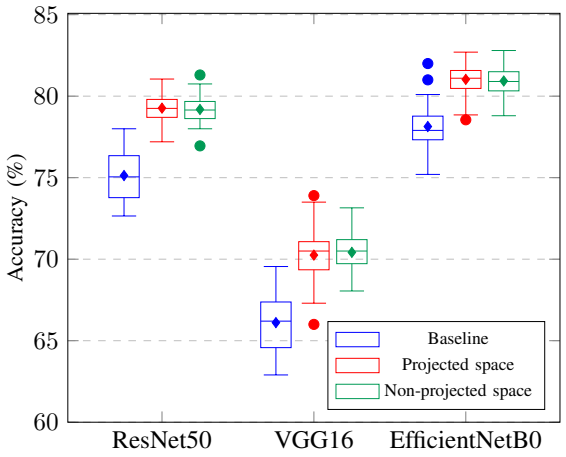


Fig. 4: Boxplot of CIFAR-10 dataset accuracies using ResNet-50, VGG-16 and EfficientNetB0 as feature extractors.

Furthermore, the standard deviation was reduced, especially for ResNet50 and EfficientNetB0, suggesting less variability.

C. Case-Study: Marine Perception

To perform a case study that relates to our goal of achieving sublinear communication costs for agents carrying out naval missions, we utilize a subset of the Open Image dataset [22]. It is composed of 4551 train images and 1325 test images divided between 2 classes: buildings and boats. Practically, the idea is to have a set of p autonomous agents continuously collecting data from their environment through various sensors, classifying the data, and using it for real-time motion planning. To simulate this, we split the dataset among the p agents and have them execute Algorithm 1.

Fig. 5: Boat and building images from the dataset [22].

Figure 6 displays the results of the distributed network test. In each case, the model trained using projected data was comparable to that trained with the non-projected data. The classification accuracy remained consistent after projection and the standard deviation was reduced, signifying a decrease in variation. Note that this is observed for all 3 CNNs, showcasing that the effectiveness of the projection method is robust across different architectures. We demonstrate that sublinear communication costs can be achieved for classification problems without sacrificing accuracy.

VI. CONCLUSIONS

This study explores the application of dimensionality reduction techniques, specifically the Johnson-Lindenstrauss transform, combined with CNNs and distributed networks to address learning challenges in communication-constrained environments. Our proposed system enhances communication efficiency among autonomous systems such as UASs,

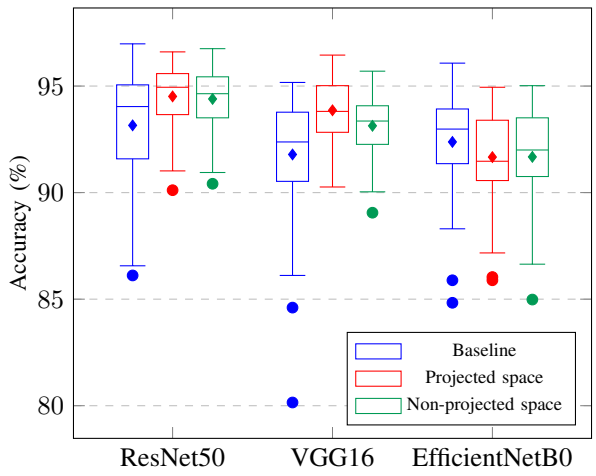


Fig. 6: Boxplot of Open Images dataset accuracies using ResNet-50, VGG-16, and EfficientNetB0 as feature extractors.

UUSVs, and UUVs. The CNN architectures chosen are designed to manage large-scale data efficiently, enabling our system to maintain robust performance even as data volumes grow. This scalability is crucial for naval missions, which require efficient handling of large and increasing data volumes, real-time decision-making, and resource usage optimization.

The experimental results, derived from both the CIFAR-10 dataset and a tailored case study on marine perception, confirm the viability of our approach. Across all 3 CNN architectures, the models trained with projected data achieve similar accuracy to the baseline model trained with the original high-dimensional data. Furthermore, the observed reduction in standard deviation across the tests suggests an improvement in consistency and reliability of model performance.

Future work involves expanding testing to real-world scenarios, such as highly efficient obstacle detection, marine life classification, and environmental health monitoring. To achieve similar accuracy in real time without prior training data, we plan to explore the use of deep reinforcement learning (DRL). The adaptability, efficiency, and robustness of DRL make it a powerful approach for tackling the challenges of real-time classification across various domains.

Furthermore, the distributed learning framework introduced offers a promising avenue for future research, especially in federated learning environments where data privacy and security are paramount. Our findings suggest that structured and sketched updates can be effectively utilized to reduce communication demands.

References

[1] A.-J. Gallego, A. Pertusa, and P. Gil, "Automatic ship classification from optical aerial images with convolutional neural networks," *Remote Sensing*, vol. 10, no. 4, 2018.

[2] K. Patel, C. Bhatt, and P. L. Mazzeo, "Deep learning-based automatic detection of ships: An experimental study using satellite images," *Journal of Imaging*, vol. 8, no. 7, 2022.

[3] J. Acharya, C. De Sa, D. Foster, and K. Sridharan, "Distributed learning with sublinear communication," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 40–50.

[4] A. C.-C. Yao, "Some complexity questions related to distributive computing (preliminary report)," in *Proceedings of the eleventh annual ACM symposium on Theory of computing*, 1979, pp. 209–213.

[5] A. Rao and A. Yehudayoff, *Communication Complexity: and Applications*. Cambridge University Press, 2020.

[6] E. Kushilevitz, "Communication complexity," ser. Advances in Computers, M. V. Zelkowitz, Ed. Elsevier, 1997, vol. 44, pp. 331–360.

[7] M. Alsayegh, A. Dutta, P. Vanegas, and L. Bobadilla, "Lightweight multi-robot communication protocols for information synchronization," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 11 831–11 837.

[8] E. Klavins, "Communication complexity of multi-robot systems," in *Algorithmic Foundations of Robotics V*. Springer, 2004, pp. 275–291.

[9] S. Dasgupta and A. Gupta, "An elementary proof of a theorem of johnson and lindenstrauss," *Random Structures & Algorithms*, vol. 22, no. 1, pp. 60–65, 2003.

[10] S. Dasgupta, "Experiments with random projection," in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, ser. UAI '00. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, p. 143–151.

[11] D. Fradkin and D. Madigan, "Experiments with random projections for machine learning," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 517–522.

[12] D. Achlioptas, "Database-friendly random projections," in *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS '01. New York, NY, USA: Association for Computing Machinery, 2001, p. 274–281.

[13] D. M. Kane and J. Nelson, "A derandomized sparse Johnson-Lindenstrauss transform," *arXiv preprint arXiv:1006.3585*, 2010.

[14] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *CoRR*, vol. abs/1610.05492, 2016.

[15] L. Li, Y. Fan, M. Tse, and K.-Y. Lin, "A review of applications in federated learning," *Computers & Industrial Engineering*, vol. 149, p. 106854, 2020.

[16] T. Guo, J. Dong, H. Li, and Y. Gao, "Simple convolutional neural network on image classification," in *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*, 2017, pp. 721–724.

[17] P. Li, T. J. Hastie, and K. W. Church, "Very sparse random projections," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 287–296.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[20] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 6105–6114.

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

[22] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *IJCV*, 2020.