# Context-Aware Orchestration of Energy-Efficient Gossip Learning Schemes

**Conference Paper** · May 2024

DOI: 10.48550/arXiv.2404.12023



**5 authors**, including:

Mina Aghaei Dinani
Université de Neuchâtel
**10** PUBLICATIONS **39** CITATIONS
SEE PROFILE

Adrian Holzer
Université de Neuchâtel
**123** PUBLICATIONS **1,932** CITATIONS
SEE PROFILE

Hung Nguyen
Ho Chi Minh City University of Technology (HCMUT)
**31** PUBLICATIONS **205** CITATIONS
SEE PROFILE

Gianluca Rizzo
HES-SO Valais-Wallis - University of Foggia
**92** PUBLICATIONS **2,248** CITATIONS
SEE PROFILE

# Context-Aware Orchestration of Energy-Efficient Gossip Learning Schemes

Mina Aghaei Dinani, Adrian Holzer
University of Neuchatel, Switzerland
name.surname@unine.ch

Hung Nguyen
The University of Adelaide, Australia
hung.nguyen@adelaide.edu.au

Marco Ajmone Marsan
Institute IMDEA Networks, Spain
ajmone@polito.it

Gianluca Rizzo
HES-SO Valais, Switzerland, and University of Foggia, Italy
gianluca.rizzo@hevs.ch

***Abstract*—Fully distributed learning schemes such as Gossip Learning (GL) are gaining momentum due to their scalability and effectiveness even in dynamic settings. However, they often imply a high utilization of communication and computing resources, whose energy footprint may jeopardize the learning process, particularly on battery-operated IoT devices. To address this issue, we present Optimized Gossip Learning (OGL), a distributed training approach based on the combination of GL with adaptive optimization of the learning process, which allows for achieving a target accuracy while minimizing the energy consumption of the learning process. We propose a data-driven approach to OGL management that relies on optimizing in real-time for each node the number of training epochs and the choice of which model to exchange with neighbors based on patterns of node contacts, models' quality, and available resources at each node. Our approach employs a DNN model for dynamic tuning of the aforementioned parameters, trained by an infrastructure-based orchestrator function. We performed our assessments on two different datasets, leveraging time-varying random graphs and a measurement-based dynamic urban scenario. Results suggest that our approach is highly efficient and effective in a broad spectrum of network scenarios.**

**keywords — Distributed Learning, Energy Efficiency, Gossip Learning, Opportunistic Communication**

## I. Introduction

Distributed learning schemes are poised to become one of the key enablers of future 6G networks, as they allow fast and efficient training of complex and large-scale models while delivering better reliability and fault-tolerance than traditional, centralized approaches. Among these, Gossip Learning (GL) schemes are of special interest, as they do not require uploading models to a parameter server, thus offering better robustness and scalability.

Originally introduced in [1], GL schemes train ML models over decentralized data via direct model gossiping among nodes. Several versions of GL have been proposed for dynamic settings [2], [3]. In these works, the changing topology is a result of varying patterns of network connectivity, changes in node availability (e.g. due to node duty cycling or battery depletion), and churn, among others, as it is often the case in realistic mobile edge and vehicular scenarios and use cases. GL is based on a combination of iterative local training and model exchange over wireless channels. Both tasks on battery-operated, resource-constrained IoT and edge devices might imply rapid energy budget depletion, potentially slowing down and jeopardizing the whole learning process.

Recently, several works have focused on decreasing the energy footprint of distributed learning, albeit in server-based architectures such as Federated Learning (FL). [4] reduces the amount of exchanged models in FL by decreasing the number of communication rounds. Other approaches focus instead on reducing the number of exchanged models at each round. These methods consider factors such as the size and quality of a node's local dataset [5], [6], [7], [8], the rate of network evolution [4], [9], and the node's trustworthiness [10]. All these works, however, consider a static network. [11] proposes a Gossip Learning scheme, evaluating the effect of the number of models merged at each round in a vehicular network. It shows that the resource-optimal value for these parameters is highly context-specific. However, it considers measurement-based mobility patterns, making it hard to untangle dependencies between mobility features and the performance of the training scheme. All these works leave unanswered the critical question of when and which nodes should exchange models in a fully distributed learning scheme to achieve a target performance (in terms of accuracy and convergence speed) in a dynamic network in an energy-optimal manner.

In this paper, we consider a scenario of reference in which cellular connectivity is pervasive, and it allows taking advantage of an orchestration function that monitors the gossip-based learning process without requiring infrastructure-based exchanges of training data or ML models. The primary contributions of this paper are:

- We propose OGL, a gossip-based training strategy for dynamic networks capable of adapting to a wide range of network topologies and dynamic settings.
- We present a data-driven approach for the dynamic management of OGL, which achieves a target performance in a resource-efficient manner by proactively adapting the models' distribution and training parameters to local conditions. The approach employs a Deep Neural Network (DNN) model, which is trained offline and distributed to all nodes at the start of the learning process, enabling each node to tune the main parameters of the OGL scheme adaptively.
- We assess the effectiveness and efficiency of our approach by leveraging time-varying random graphs and a measurement-based mobility trace. These results suggest that it substantially outperforms a set of baseline approaches while achieving the target minimum accuracy in all of the considered scenarios.

## II. System model

We consider a set $V$ of nodes, with cardinality $|V|$ (modelling, e.g., mobile devices, UAVs, or connected vehicles) moving within a specific region according to an arbitrary mobility model and during a predefined time interval $T$ (the *observation interval*). Let $v \in \mathbb{N}$ denote the unique identifier of a node. We assume nodes can communicate directly with each other through wireless peer-to-peer (P2P) communications (e.g., using DSRC or Bluetooth Low Energy [12]). Furthermore, each node is equipped with a cellular network interface. Two nodes can exchange information whenever they are in *contact*, that is, within each other's transmission range. We assume these exchanges are always unicast (one-to-one). However, the proposed scheme can be easily extended to incorporate the effects of multicasting and broadcasting. We assume there is a coordination function (possibly implemented by a Software Defined Network Controller(SDNC) [13]) in the region, and it resides within the cellular access network. The coordination function comprises an auxiliary ML model $M_{tune}$. Through its cellular network interface, the coordination function transmits $M_{tune}$ to the nodes entering the region. Hence, the architecture of this model is equal for all nodes. Note that all communications are P2P, except for the initial dissemination of the $M_{tune}$ model from the coordination function to nodes. Every node independently employs $M_{tune}$ to fine-tune and adjust its learning parameters. Moreover, we assume each node entering the region possesses a local model $w_v$ and a *local dataset*, partitioned in the training set $\mathcal{D}_v$, and validation set $\mathcal{S}_v$, which generally differ in size and composition for each node. The choice of the validation and training set size is context-specific. We also assume that the observation interval is segmented into $I$ *slots* of equal size, short enough that node mobility patterns can be considered not to vary substantially within each slot. Let $t \in 1, .., I$ be the label of slots.

## III. The OGL Approach to Energy-Efficient Gossip Learning

### A. A Gossip-Based Collaborative Training Algorithm

We assume all nodes in the region train their local model through a gossip-based cooperative learning algorithm, denoted as OGL, and based on P2P model exchanges among nodes. Such an approach is orchestrated by a cellular-based coordination function, without requiring the exchange of the trained models or training data (which would potentially expose it to privacy breaches) between each node and the coordinator. At the beginning of the scheme, all nodes present in the region randomly generate an initial local model $w_0$. We assume the generation procedure to produce the same random initial model for all nodes. Similarly, after the beginning of the scheme, whenever a node enters the region, it generates $w_0$ following the same procedure. Starting from $w_0$, every node elaborates an ML model (which we assume will be used by the node itself. e.g. to carry out the same inference task, e.g. trajectory prediction or image recognition) by alternating local training on each node's local training set, with model aggregation with models received from neighbours. Moreover, to all nodes joining the scheme, the coordination function delivers an *auxiliary ML model* $M_{tune}$. Each node employs $M_{tune}$ for the adaptive tuning of some key parameters of the learning process. Such dynamic management of the learning process at each node is based on each node's available hardware resources and power budget. In addition, it also accounts for each node's context in terms of the number of neighbours, the speed at which they vary over time, and the quality and quantity of their local model (i.e., in terms of mean accuracy or loss), among others. Each node then uses the $M_{tune}$ to modulate the number of local training epochs and to choose, among its neighbours, those whose local model should be requested and used for improving the node's local model, as we will explain later.

Then, at every time slot, the OGL algorithm proceeds through three *phases*. The duration of each phase can be tuned and adapted to the specific training task and setup, and it does not need to be synchronized across nodes.

In the *training* phase, each node in the region applies $M_{tune}$ to get the number of epochs $Z_{v,t}$ that it has to train its local model over its local dataset and train its model accordingly. Subsequently, it assesses its local model over its validation set $\mathcal{S}_v$ to derive the loss value $l_v$ used in the next phase. The choice of the loss function is context-specific.

In the *communication* phase, nodes exchange the loss value of their local model with their neighbours. Each node then employs $M_{tune}$ to identify the neighbouring nodes from which it should request the transfer of their local models. The node then initiates a request directed towards the selected nodes, soliciting their respective models. In response, these requested nodes transmit their models if they are still within range of each other. During this phase, a node may not request any model, e.g. because it has no neighbours or when the $M_{tune}$ indicates that no model is worth requesting among the available neighbours' models. We assume that the connectivity between nodes is relatively stable while exchanging models.

Finally, in the *merging* phase, each node combines the models received from the chosen neighbours and its local model to produce a new version of its local model. The merging procedure consists of a weighted averaging method. The weights associated with each merged model are computed via the DFed Pow strategy [3]. In DFed Pow, the weight of each model to be merged is a function of the inverse of loss calculated on the node's validation set. Note, however, that our approach is more general and does not rely on a specific algorithm for calculating the weights for merging.

The three phases are repeated at each time slot until a stopping condition is met (e.g., after a maximum number of iterations, when the average local models' accuracy surpasses a certain threshold or when there is no significant improvement in the model's accuracy over several rounds). Regardless of the reason, the final round at which the algorithm stops is called the cut-off round.

### B. Formulation of the energy optimization problem

The main goal of our OGL approach is to enable the energy-efficient training of an ML model in a distributed manner. As mentioned, this is enabled by an orchestrator function that elaborates and distributes the $M_{tune}$ model among all the nodes entering the region. Given the node context, as well as some key parameters of the system and the training task, such a model enables each node to tune the number of training epochs and the set of ML models to merge, which allows for achieving a given target accuracy while minimizing a cost function which models the overall energy cost of the training process.

In what follows, we formalize the energy optimization problem that the orchestration function tries to solve. The cost function we consider is the sum of two components. The first one accounts for the computing costs. Generally, the energy consumption associated with a computation task is determined by CPU(or GPU) usage and memory resources [14]. Those, in turn, depend upon the architecture implemented, the quantity of data it processes, and the node's characteristics. In this work, we assume all nodes have the same computing power. Then, the energy required to run the (local) training process is:

$$S(Z) = \sum_{t \in T} \sum_{v \in V} Z_{v,t} d_v (e_g + e_s) \tag{1}$$

- $Z_{v,t}$ depicts the number of epochs required by node $v$ to train its local model using the local dataset at time slot $t$;
- $e_g$ is the energy consumed by the CPU or GPU to perform one training epoch on one sample;
- $e_s$ is the energy required to provide storage and memory resources for the training of one epoch on a sample;
- $d_v$ denotes the number of samples of the local training set of node $v$;
- $T$ is the label of the slot at which convergence happens.

The energy consumed for computing the loss of the local model on the validation set is modelled as follows:

$$\Gamma = \sum_{t \in T} \sum_{v \in V} s_v (e_e + e_{es}) \tag{2}$$

$e_e$ and $e_{es}$ are the energy consumed by the CPU (or GPU) and energy required to provide storage and memory to evaluate the local model on one dataset sample. $s_v$ indicates the size of the validation set at node $v$.

The second component of the cost function accounts for the communication costs. The communication costs consider the exchanges between nodes:

$$C(k) = C^{d2d} \sum_{t \in T} \sum_{v \in V} h_{v,t} L + k_{v,t} (M + R) \tag{3}$$

- $C^{d2d}$ is the cost per byte of a d2d (peer to peer) transfer
- $\mathcal{H}_{v,t}$ is the set of neighbors of node $v$ at time slot $t$, of cardinality $h_{v,t}$;
- $\mathcal{K}_{v,t} \subseteq H_{v,t}$ is the set of chosen neighbours of node $v$ at time slot $t$ from which models to be merged are retrieved, of cardinality $k_{v,t}$.
- $L$, $R$ and $M$ are the message size containing loss value, request and a local model, respectively.

Note that such a cost function neglects the cost of model merging, as it is usually negligible [3], [11]. Let $\mathcal{Z} = \{Z_{v,t}\}$, and $\mathcal{K} = \{\mathcal{K}_{v,t}\}$. Thus, an optimal OGL orchestration scheme is a solution to the following optimization problem:

*Problem 1:*

$$\underset{\mathcal{Z}, \mathcal{K}}{\text{minimize}} \; C(k) + \beta (S(Z) + \Gamma) \tag{4}$$

Subject to:

$$r \geq r^0 \tag{5}$$

Where $r$ denotes the mean accuracy of the trained model across all nodes achieved at convergence, and $r^0$ is its target minimum value. By varying $\beta$, it is possible to adapt the cost function to settings with different resource availability on user devices and at the cellular network and to different incentive schemes for resource sharing and cooperation.

### C. OGL architecture, components and functions

The OGL approach aims at solving Problem 1 by training a DNN-based auxiliary ML model, which enables nodes to adapt in real-time the learning process to available contributions by neighbours and, more generally, to each node's context. The auxiliary model is trained by the orchestrator on a training dataset whose data points are labelled by simulating the system.

We assume the orchestrator regularly collects data from every node, which is used as the feature set of the auxiliary model that it has to train. The data collected are those that are well known from the state of the art to be relevant to the training process and its efficiency. These include computing and communication costs, the number of neighbours for each node at each time slot, the size of the local dataset, the available computing power, and the initial power budget of each node. Such a choice of features as input parameters for the auxiliary model is, however, one of many possible, and our approach is independent of it. From each set of input parameters, the orchestrator derives a set of full system configurations by associating to the input parameters a random value for each of the parameters in $\mathcal{Z}$ and a random subset $\mathcal{K}$ of each node's neighbours. These inputs are fed to a simulator, which labels them with the outputs and performance metrics of the distributed training scheme. Specifically, for every node and every time slot in a given time interval during which the orchestrator has collected data, the simulation derives the local model accuracy, the loss value, and the energy budget of each node. In such a way, a training set is produced, which is then used to train a DNN model.

This model is trained and evaluated using a k-fold cross-validation approach [15], with 10 folds. The architecture of the model is a multi-layer perceptron composed of four layers. The multiple layers allow models to be more efficient at learning complex features [16]. The initial layer is a dense layer with 64 neurons and a rectified linear unit (ReLU) activation function. The second layer is a flattening layer, which reshapes the input to a one-dimensional array. The third and fourth layers are dense layers with 32 and 16 neurons, respectively, and ReLU activation functions. The final layer is a dense layer with two neurons. The model is compiled with a mean squared error loss function and the Adam optimizer. Early stopping and model checkpoint callbacks are used to prevent overfitting and save the best model. This approach ensures a robust evaluation of the model performance, as it assesses the model's ability to generalize to unseen data. Note that the selection of parameters in the model is either empirical or based on extensive usage in the state-of-the-art models. After the training and evaluation process, the best-performing model, referred to as $M_{tune}$, is saved for future use. This model encapsulates the optimal parameters learned during the training process. At runtime, the orchestrator function disseminates the $M_{tune}$ model to all nodes entering the region. Then, at each time slot, each node feeds the $M_{tune}$ model with its own data to determine the optimal number of local training iterations (number of epochs) and the optimal set of models to merge to achieve the given target accuracy while minimizing the energy cost of the whole process.

| CNN parametrs | MNIST | CIFAR-10 |
|---|---|---|
| Input shape | (28,28,1) | (32,32,3) |
| Batch size | 32 | 64 |
| Learning rate | 0.0001 | 0.001 |
| Number of neurons | 100 | 100 |
| Momentum | 0.9 | 0.60 |
| Kernel dimension | 3 | 3 |
| Number of filters | 32 | 32 |
| Number of outputs | 10 | 10 |

TABLE I: Parameter values used to train the CNN model on the CIFAR-10 and MNIST datasets.

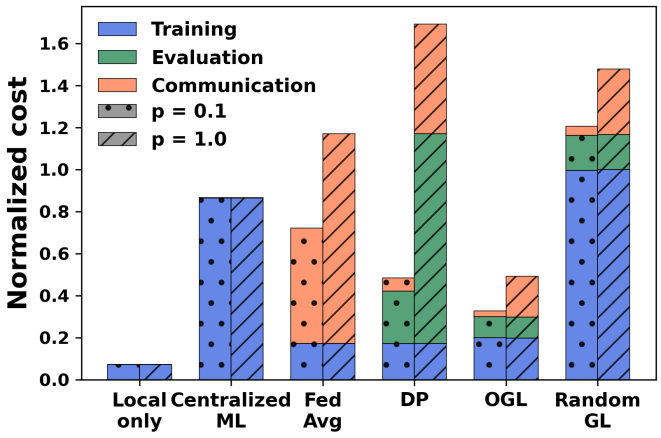



Fig. 1: Comparative analysis of cost function on the MNIST dataset with $|V| = 6$ and various values for $p$. Results are presented with a confidence interval of 95% and an error margin of 2%.

| Algorithm | Acc | F1 | Loss | Precision | Recall | Cut-off round |
|---|---|---|---|---|---|---|
| Centralized ML | 0.87 | 0.87 | 0.45 | 0.88 | 0.87 | 300 |
| Fed Avg | 0.85 | 0.84 | 0.57 | 0.87 | 0.85 | 500 |
| Local only | 0.36 | 0.42 | 3.2 | 0.42 | 0.56 | 200 |
| DP | 0.58 | 0.60 | 1.42 | 0.68 | 0.67 | 500 |
| OGL | **0.88** | **0.88** | **0.44** | **0.88** | **0.89** | 320 |
| Random GL | 0.51 | 0.56 | 1.73 | 0.7 | 0.6 | 500 |

TABLE II: Performance metrics of OGL at the cut-off round, compared to baselines on the MNIST dataset, with $|V| = 6$ and $p = 1$. Results are presented with a 98% confidence interval and a maximum error margin of 1%.

## IV. Numerical Assessment

To assess the effectiveness of our OGL approach in dynamic settings, we consider a set of $V$ nodes which need to perform a handwritten digit recognition task (MNIST dataset [17]) or object recognition (CIFAR-10 dataset [18]). We assume that each node in the system is endowed with a *local dataset* of different sizes and randomly selected without replacement from the original MNIST and CIFAR-10 datasets. The resulting dataset size, denoted by $d_v$, falls within the range of 50-350 samples for each node. This implied a training set size ranging from 600 kB to 3.2 MB when utilizing the CIFAR-10 dataset and between 224 KB and 645 KB when employing

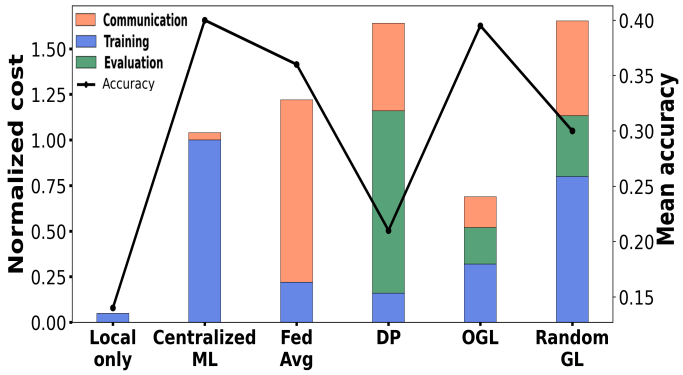



Fig. 2: Comparative analysis of cost function and mean accuracy at convergence on the CIFAR-10 dataset, with $|V| = 6$ and $p = 1$. Results are presented with a confidence interval of 95% and an error margin of 2%.

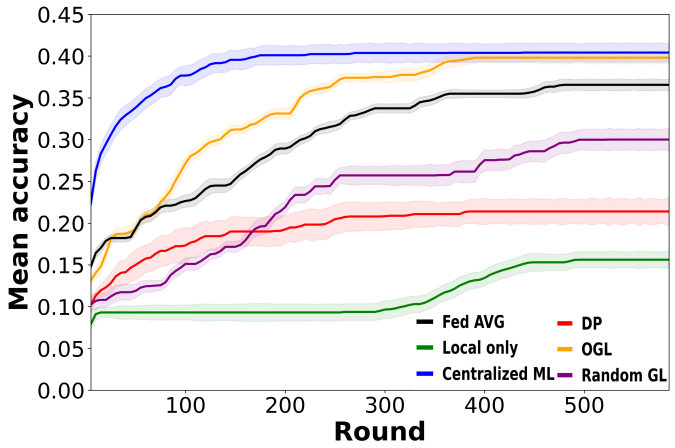
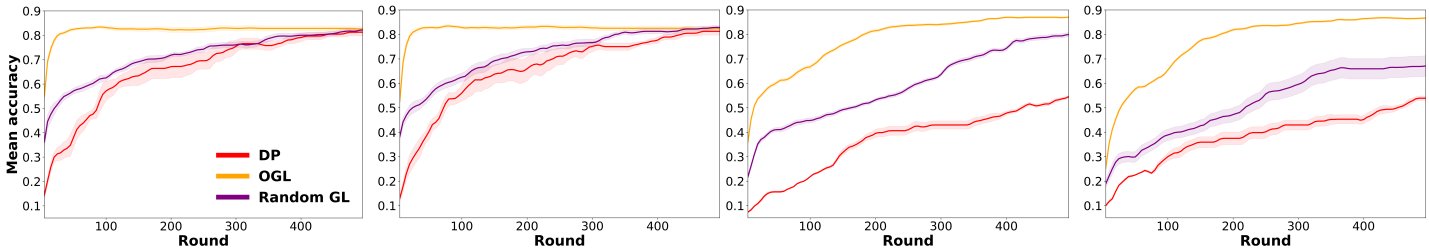



Fig. 3: Mean accuracy versus round of our OGL algorithm compared to baselines using the CIFAR-10 dataset, with $|V| = 6$ and $p = 1$. Each curve is enveloped by a highlighted band, which signifies the 95% confidence interval.

the MNIST dataset. Let us denote the aggregate local dataset across all nodes in the system as the *global dataset*. The size of the global dataset is 700 samples in all scenarios unless stated differently. We aim to ensure that the results remain consistent and are not subject to significant variation due to differences in global information in different scenarios.

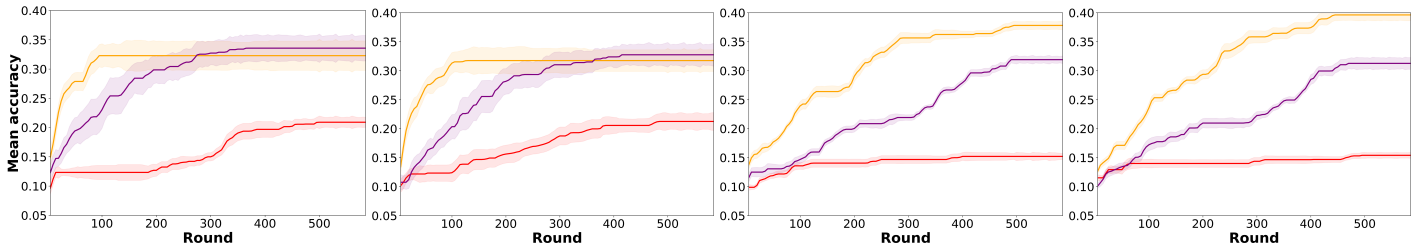
In Problem 1, the coefficient $\beta$ has been set to 1, to ensure both computing and communication costs are equally weighted. The target accuracies have been set to 0.8 and 0.4 for the MNIST and CIFAR-10 datasets, respectively. These targets were set based on the convergence accuracy of a centralized ML model trained on the global dataset. Furthermore, we assume the cost per byte of d2d transfer to be four times less costly than device-to-server transfers. This is because d2d transfers bypass the need for data routing or server maintenance, making them a more cost-effective solution. We associate a *global test set* obtained by random sampling 20% of the source datasets and ensuring that the local datasets and test set are disjoint. We assume that nodes use a CNN model to perform both inference tasks.

The first layer of the CNN model is a Conv2D layer,




(a) $|V| = 3, p = 0.1$ (b) $|V| = 3, p = 0.7$ (c) $|V| = 12, p = 0.1$ (d) $|V| = 12, p = 0.7$

Fig. 4: Mean accuracy versus round for OGL, random GL, and DP algorithm using the MNIST dataset, for different values of number of nodes in the system and edge probability. Each curve is enveloped by a highlighted band, which signifies the 95% confidence interval.




(a) $|V| = 3, p = 0.1$ (b) $|V| = 3, p = 0.7$ (c) $|V| = 12, p = 0.1$ (d) $|V| = 12, p = 0.7$

Fig. 5: Mean accuracy versus round for the OGL, Random GL, and DP algorithms, for the CIFAR-10 dataset, for different values of number of nodes in the system and edge probability. Each curve is enveloped by a highlighted band, which signifies the 95% confidence interval.

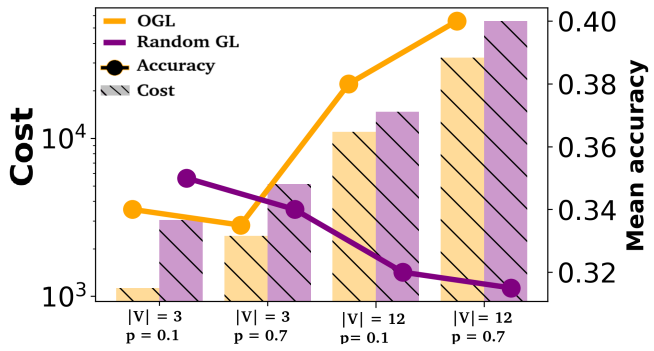



Fig. 6: Comparison of mean accuracy at convergence time versus cost for the OGL and random GL algorithms over different network configurations using the CIFAR-10 dataset. Results are presented with a 98% confidence interval and a maximum error margin of 2%.

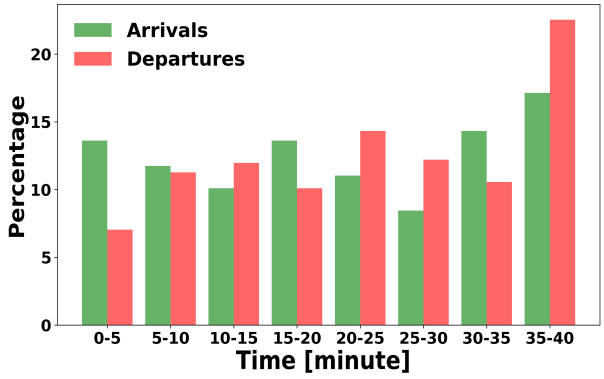



Fig. 7: Histogram representing the percentage of vehicle arrivals and departures across different time slots in the Luxembourg City off-peak scenario (12:00-12:40 PM). Time on the x-axis shows the time from the beginning of the scheme.

which applies a number of convolutional operations to the input image and uses the activation function ReLU. This layer is followed by a MaxPooling2D layer with a 2x2 pool size, which reduces the spatial dimensions of the input. The Flatten layer then transforms the 2D matrix data into a 1D vector. Subsequently, a Dense layer using ReLU activation and He-uniform weight initialization is added. The final layer is another Dense layer, with the number of neurons equal to the number of output classes, and the softmax activation function is used for multi-class classification. The model is compiled with the Stochastic Gradient Descent (SGD) optimizer and categorical cross-entropy loss function. The values of the parameters of each layer are mentioned in Table I. We choose this architecture and parameter values based on two key considerations. Firstly, some choices are widely recognized as effective in extracting shape features from images for the considered datasets [19]. Secondly, we tune some other parameters empirically by conducting a series of experiments.

With the specified parameters, the resulting model size is approximately 320 KB when trained on the MNIST dataset and 1.2 MB when trained on the CIFAR-10 dataset. Note that better energy performance might be achieved by tuning all of the CNN hyperparameters, as they impact the size of the model to be exchanged. However, this is out of the scope of the present work and is left for future developments. We end our simulations after 600 rounds or when the average accuracy across all nodes does not improve by more than 0.5% for 20

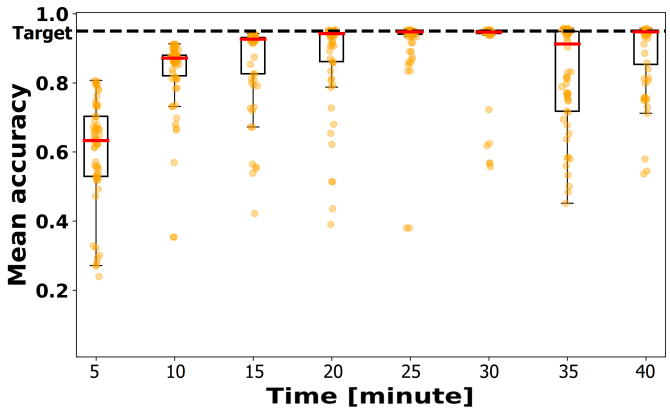



Fig. 8: Distribution of mean accuracy at every five-minute interval for the OGL algorithms in the Luxembourg City off-peak scenario (12:00-12:40 PM) using the MNIST dataset. Each point is the mean accuracy of a single vehicle, averaged in the time interval. The target line shows the target accuracy (95%) obtained using the centralized ML training on the union of the local dataset of all vehicles. Time on the x-axis corresponds to the time from the beginning of the scheme.

consecutive rounds.

In addition to our scheme, we have considered the following baseline approaches:

- *Centralized ML.* In this approach, a central server possesses a dataset identical to the scenario's global dataset, over which it trains the CNN model.
- *Federated Averaging (Fed AVG)* [20]. In this training scheme, a parameter server collects the CNN models trained locally by each node at every round, merges them, and sends the resulting CNN to each node for a new round of local training. For fairness of comparison, we assumed random client subsampling, with an average number of selected clients coinciding with the average number of nodes each node comes in contact with during a round.
- *Decentralized Powerloss (DP)* [11] is a decentralized learning approach. In this approach, all the nodes set the number of local training epochs to 1 and merge the models from all neighbours at a given time slot without exception. In this approach, the weights associated with each model to be merged are derived from a measure of the received models' performance over the node's validation set.
- *Random GL* is derived from OGL algorithm by setting uniformly at random ( and independently for each node and time slot) parameters $\mathcal{K}_{v,t}$ and $\mathcal{Z}_{v,t}$, i.e. the number of training epochs and the set of neighbour nodes whose models have to be merged.
- *Local only*, in which each node trains the local model only on its local dataset, with no data or models exchanged with neighbouring nodes or a server.

The size of local datasets varies across nodes, leading to an uneven distribution of classes. It requires using various performance metrics to compare the effectiveness of our algorithm with baseline methods.

In the first set of experiments, we considered scenarios with different numbers of nodes in the network, specifically $|V| = [3, 6, 12]$. In addition, we model the connectivity graph resulting from node mobility via an Erdős-Rényi dynamic random graph [21]. It is thus a sequence of graphs, each associated with a time slot. In this type of graph, an edge is established between two nodes with probability $p$, independent from other edges. Then, at each time slot, the connectivity graph stays constant, but possibly the set of edges in the graph (connection among nodes) varies. The degree of connectivity in the network is determined by the parameter $p$. A mesh network is formed when $p = 1$, while $p = 0.1$ leads to a sparse network. This type of graph enables a controlled and systematic modification of node numbers, connection patterns, and node interaction frequency and duration [21].

Figure 1 shows the mean total computing and communication costs at convergence for OGL and the baselines in different network configurations utilizing the MNIST dataset. Figure 2 illustrates the mean total amount of computing and communication costs at convergence for OGL as well as for the baselines using the CIFAR-10 dataset. These results suggest that our OGL scheme is by far the most energy-efficient among the gossip learning schemes, particularly concerning communication costs, achieving a level of efficiency comparable to that of Federated Learning. This confirms that context-aware tuning of the local training and merging phases of GL schemes may have a high impact on the efficiency and effectiveness of the training process. Another key aspect resulting from our experiments is the relative mean training performance of our OGL scheme in terms of model accuracy at convergence. As Table II shows, using MNIST dataset OGL outperforms all baseline distributed approaches in all performance metrics, achieving performance comparable to centralized training. Critically, though being significantly more energy efficient, at convergence, OGL improves by more than 40% both mean accuracy and mean loss with respect to DP, i.e. to the best performing gossip-learning approach in the state-of-the-art. Indeed, the two other distributed learning models, DP and random GL, as well as the local-only approach, fail to achieve the target mean accuracy. Figure 3 depicts the mean accuracy versus round (learning process) of the OGL and other baseline algorithms using the CIFAR-10 dataset. The outcomes derived from the CIFAR-10 dataset largely mirror those obtained from the MNIST dataset. It reinforces the effectiveness and consistency of the OGL algorithm across different datasets.

Figure 4, and 5 show the impact of network connectivity and the number of nodes in the system on the evolution of mean accuracy over the learning round for MNIST and CIFAR-10 datasets. In a system with very few nodes, the impact of optimally choosing the neighbours' contributions is relatively modest, with the mean accuracy of Random GL and DP eventually matching that of OGL. In larger systems, our OGL tuning approach is key to achieving faster convergence and higher accuracy in sparse and dense networks. Figure 6 illustrates that OGL maintains superior energy efficiency, notwithstanding an accuracy comparable to Random GL. Note that all schemes perform sensibly worse in the CIFAR-10 dataset, as for the same average local dataset size, its samples are more complex (i.e. larger pictures with more pixels).

To evaluate the effectiveness of our OGL approach in a realistic scenario, we consider a scenario where moving nodes are vehicles traversing a region of interest. We focused on a specific area in the city centre of Luxembourg City. This area, a square with sides measuring 1 km, was observed during a low-traffic period (off-peak) from 12:00 PM to 12:40

PM. During this time interval, there are 492 vehicles in the region, with an average sojourn time of 2.9 minutes. On average, there are about 27.3 vehicles in the region at any given time. In this scenario, vehicles are in contact if they are within each other transmission radius. The transmission radius has been set to 150 m (e.g. typical of DSRC in urban environments [22]). In this case, on average, each vehicle is in contact with 6.7 vehicles at any time. Note that, unlike previous scenarios, the set of nodes in the region may change at different time intervals. Figure 7 depicts the percentage of arrivals and departures at every 5-minute interval. To ensure a dynamic neighbourhood pool and give each vehicle enough time to train its local model, vehicles interchange both the loss values and the trained models at regular intervals of twenty seconds. Figure 8 shows the mean accuracy of the vehicles approach the target accuracy, obtained by training a centralized ML model on the union of all vehicles' datasets, after ten minutes despite having churn in the network. In addition, we observe the adaptive learning capability of new arrivals. New arrivals are characterized by their initial impact on reducing the mean accuracy, as they are identified as outliers within the given time shown in Figure 8. Despite the initial disruption, these outliers demonstrate a capacity to learn from the existing vehicles, thereby gradually aligning with the overall trend. This is evidenced by the subsequent decrease in the number of outliers from time interval 15-20 to 20-25 minutes. This adaptive learning capability of new arrivals contributes to the robustness and resilience of the system, enabling it to maintain overall accuracy over time. It indicates our OGL model is able to maintain high accuracy even in the face of network instability.

## V. Conclusions

This work presents a novel approach to an energy-efficient gossip learning scheme for dynamic settings. We employ an auxiliary DNN model trained by an orchestrator to adaptively tune some of the key parameters of the learning process in a decentralized manner. Results indicate that our approach efficiently achieves accuracy comparable with a centralized ML method across various network conditions, utilizing time-varying random graphs and a measurement-based dynamic urban scenario across two distinct datasets.

For future work, we plan to enhance the scalability and adaptability of our optimization system by developing a fully distributed optimization system, eliminating the need for an orchestrator, where nodes can self-optimize in response to drastic environmental changes. Furthermore, we plan to investigate the impact of different types of DNN models on the optimization and learning process, as DNN models vary in their computational requirements. This could be particularly important in a distributed learning context where computational resources are limited.

## References


[1] R. Ormándi *et al.*, "Gossip learning with linear models on fully distributed data," vol. 25, no. 4, pp. 556–571, 2013.

[2] M. A. Dinani, A. Holzer, H. Nguyen, M. A. Marsan, and G. Rizzo, "A gossip learning approach to urban trajectory nowcasting for anticipatory ran management," *IEEE TMC*, pp. 1–17, 2023.

[3] Dinani, Mina Aghaei and Holzer, Adrian and Nguyen, Hung and Marsan, Marco Ajmone and Rizzo, Gianluca, "Gossip learning of personalized models for vehicle trajectory prediction," in *2021 IEEE WCNC*, 2021, pp. 1–7.

[4] Y. Zhou, Y. Qing, and J. Lv, "Communication-efficient federated learning with compensated overlap-fedavg," 2021.

[5] A. M. Abdelmoniem, A. N. Sahu, M. Canini, and S. A. Fahmy, "REFL: Resource-efficient federated learning," in *Proceedings of the Eighteenth European Conference on Computer Systems*. ACM, may 2023.

[6] O. Marfoq, G. Neglia, L. Kameni, and R. Vidal, "Personalized federated learning through local memorization," 2022.

[7] F. Malandrino and C. F. Chiasserini, "Federated learning at the network edge: When not all nodes are created equal," 2021.

[8] H. Wu and P. Wang, "Fast-convergent federated learning with adaptive weighting," 2021.

[9] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," 2019.

[10] A. Imteaj and M. H. Amini, "Fedar: Activity and resource-aware federated learning model for distributed mobile robots," 2021.

[11] M. A. Dinani, A. Holzer, H. Nguyen, M. A. Marsan, and G. Rizzo, "Vehicle position nowcasting with gossip learning," in *2022 IEEE WCNC*, 2022, pp. 728–733.

[12] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *sensors*, vol. 12, no. 9, pp. 11 734–11 753, 2012.

[13] B. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: Past, present, and future of programmable networks," *Communications Surveys & Tutorials, IEEE*, vol. PP, no. 99, pp. 1–18, 2014.

[14] X. Tang, J. Li, K. Li, and A. Y. Zomaya, "Cpu–gpu utilization aware energy-efficient scheduling algorithm on heterogeneous computing systems," *IEEE Access*, vol. 8, pp. 58 948–58 958, 2020.

[15] P. Refaeilzadeh, L. Tang, and H. Liu, *Cross-Validation*. Boston, MA: Springer US, 2009, pp. 532–538. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_565

[16] R. Kruse, C. Borgelt, F. Klawonn, C. Moewes, M. Steinbrecher, and P. Held, *Multi-Layer Perceptrons*. London: Springer London, 2013, pp. 47–81. [Online]. Available: https://doi.org/10.1007/978-1-4471-5013-8_5

[17] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, conference Name: Proceedings of the IEEE.

[18] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)." [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[19] L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan, "Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, pp. 1–74, 2021.

[20] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[21] A. Di Maio, M. A. Dinani, and G. Rizzo, "The upsides of turbulence: Baselining gossip learning in dynamic settings," in *Proceedings of the Twenty-Fourth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, ser. MobiHoc '23. New York, NY, USA: ACM, 2023, p. 376–381.

[22] K. Abboud, H. A. Omar, and W. Zhuang, "Interworking of dsrc and cellular network technologies for v2x communications: A survey," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 12, pp. 9457–9470, 2016.