Magnum: A Distributed Framework for Enabling Transfer Learning in B5G-Enabled Industrial-IoT

Pallav Kumar Deb, *Graduate Student Member, IEEE*, Sudip Misra, *Senior Member, IEEE*, Tamoghna Sarkar, and Anandarup Mukherjee, *Graduate Student Member, IEEE*

***Abstract*—In this paper, we propose a lightweight blockchain-inspired framework - Magnum - as a magazine of transfer learning models in blocks. We propose the storage of these blocks on proximal fog nodes to simplify access to pre-trained base models by industrial plants to tune them before deployment. We design Magnum for B5G-enabled scenarios to reduce the block transfer time. We formulate a demand-centric distribution scheme to further reduce search and access time by adopting a Nonlinear Program model and solving it using the branch and bound method. Through extensive experiments and comparison with state-of-the-art solutions, we show that Magnum retains the accuracy of the models and present its feasibility with a maximum CPU and memory usage of 80% and 6%, respectively. Additionally, while Magnum requires a maximum of 10 seconds for writing models as large as 17 Mb on the blocks, it requires 16 micro-seconds for fetching the same.**

***Index Terms*—Transfer learning, IIoT, I4.0, Distributed learning, Blockchain, B5G communications, 6G communications, Fog computing**

I. Introduction

Machine Learning (ML) solutions are typically application-specific and industries need to develop *unique models* for each job/task explicitly. The *collection* of training data and its conversion to structured forms and then *training* the models for yielding highly *accurate results* for Industry 4.0 (I4.0) requires *significant time*. Moreover, *sharing* such huge data across the globe is cumbersome, and storing redundant information across all branches of an industry causes *wastage* in memory. Further, each branch may require multiple models for its operations. In such cases, Transfer Learning (TFL) methods pave the way towards one solution for all paradigms and its *interoperability* with reduced *computing power* requirements and *deployment time* [1]. However, the *reliable transfer* of the base models across geographically *distributed* branches is challenging. An easy-to-deploy solution that offers *transparency, immutability, distribution,* and *sequential storage* is beneficial. Although blockchain (BC)-based methods are a promising solution for the *immutable sharing* of such models [2], the steps involved in updating the chain and distribution of the blocks consist of complex operations. Reducing the complexity of these operations in a BC and the *demand-centric* distribution of the blocks is necessary for overcoming the aforementioned issues.

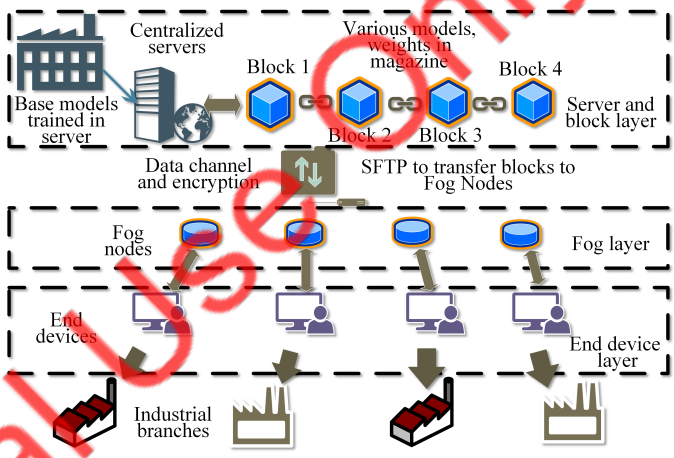


Figure 1: Overview of the proposed Magnum framework.

In this work, we propose and develop a magazine – Magnum – to store pre-trained TFL models and their corresponding weights in the form of interconnected blocks across fog nodes. We propose training the base models at pre-defined locations, following a centralized infrastructure. We design the blocks for providing the pre-trained models to the concerned branches based on their features and requirements. As shown in Fig. 1, we distribute the magazine of models from the servers across the network. We make them available to the geographically separated branches under a particular industry using Fog Nodes (FNs) in the fog computing layer. As ML training routines generate *sizeable* models, we also adopt the upcoming B5G technology to illustrate the delivery of the blocks on the factory floor with minimal delay. Moreover, we propose a demand-based distribution of the blocks to further reduce search and access times. The demand-based placement of the blocks facilitates storage close to the requesting branches. The end devices in these branches acquire the pre-trained models from the FNs and tune them according to their needs before deployment.

Example Scenario: Consider an industry that has its branches spread across the globe. Typically, these distributed branches may be sub-divided into smaller sets based on their similarity with respect to its objectives, infrastructure, and resources. However, the granular parameters differ according to location and raw materials, mandating the need for minor variations of the same base ML model. Magnum allows these branches to

P. K. Deb, S. Misra, and A. Mukherjee are with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India. e-mail: (pallv.deb,sudipm)@iitkgp.ac.in, anandarupmukherjee@ieee.org

T. Sarkar is with the Department of Electronics and Instrumentation Engineering, SRM Institute of Science and Technology, India. e-mail: tamoghnasarkar97@gmail.com

download the models from the corresponding blocks stored in proximal FNs and tune it according to their needs, avoiding the overhead of training the same models from scratch. The demand-centric block distribution and B5G communications also helps in reducing search and access times.

A. Background

In this section, we briefly describe some of the concepts necessary to follow this paper.

1) B5G Networks: The current network infrastructure typically operates at a center frequency of 2.4 GHz with a bandwidth of 20 MHz per channel (4G). Due to the increase in number of end devices and demand for higher data rates, researchers propose the adoption of the 30 GHz frequency spectrum with a bandwidth of 100 MHz per channel (5G). The increased data rate helps in exchanging colossal data in real-time. In this work, we consider an even higher range of center frequencies than 5G, Beyond 5G (B5G) to further reduce transmission delays. In particular, we consider the frequencies analogous to the 6G networks with frequencies in the range of $1-10$ THz and 500 GHz bandwidth. Although such high frequencies help in achieving high data rates, they suffer from low transmission ranges (1 m). Additionally, the 6G networks suffers from environmental factors, particularly absorption and spreading due to water vapour instead of the conventional ones [3], which needs attention before standardization. We discuss the relevant parameters and formulae in Section III-C.

2) Transfer Learning: TFL is an ML technique which enables reusing a model trained for a particular task and use it for another. TFL first trains a base model for a particular task and then repurposes the learned parameters to serve another task. For instance, a model trained to identify cars and identify trucks too. In industries, machines such as Friction Stir Welding (FSW) may use the same components to operate globally. However, the conditions vary according to the region of deployment. In such scenarios, a singular ML model does not suffice for all and need region-based parameter tuning. In this work, we propose a method for seamless sharing and accessing the varying TFL models with minimum delay.

B. Motivation

I4.0 and ML are being used to cater to a plethora of applications in major industries today. However, it is cumbersome for the industries to repeatedly develop unique models for each branch. Training a model takes significant time and effort to collect data and its conversion to a structured form. Repeating the same effort for all the branches associated with an industrial company is daunting. Sharing the data and models across the globe is challenging and also causes wastage in memory due to data redundancy. In such cases, a framework that allows the distribution of blocks containing pre-trained models among these industrial branches is beneficial. Further, enabling the branches to tune the parameters according to their needs reduces computation efforts and deployment time. These shortcomings motivate us to develop the proposed magazine Magnum, that allows the seamless distribution of TFL models across the globe. We also extend this work to minimize the blocks' access and search time by proposing a demand-centric distribution.

C. Why Not Blockchain?

The proposed TFL sharing scheme – Magnum is a derivative of the traditional BC, with variations in terms of its *consensus and computational complexities*. Primarily, Magnum removes the need for a *miner*. Since we consider resource-constrained FNs, excluding miners removes the need for high-performance devices in the fog layer. The security and authenticity of the data inserted into the blocks depend on central authorities responsible for creating the base TFL models, which reduces the complexity of mining huge volumes of data and also removes the need for employing additional third parties for verification. However, each block's security and privacy may be taken care of by incorporating encryption techniques or by adopting Attribute-Based Encryption (ABE) schemes. With the use of ABE, industries may easily store the models into the blocks and restrict their visibility/access based on the attributes/features of the branches.

D. Contribution

In this work, we propose a magazine for storing pre-trained TFL models for industries to acquire and tune according to their needs before deployment. Towards this, the specific set of contributions in this work are:

- ***Magnum:*** We propose a magazine consisting of interconnected blocks that contain pre-trained ML models. With the elimination of miners and their complex consensus mechanisms, the magazine is suitable for resource-constrained devices.
- ***IoT-Based Architecture:*** We propose the use of a fog-cloud architecture to use centralized servers for preparing the pre-trained models and then store them across geographically spread FNs.
- ***Demand-Centric Distribution:*** We distribute the blocks of the proposed magazine in the fog layer based on the demands from each location. This further reduces the search and access times.
- ***Reduced latency:*** We exploit the features of B5G communications technology to further reduce communication delays. In comparison to conventional technologies, B5G network offers real-time communications on the factory floor.

It may be noted that we use the FNs only for storing Magnum (series of TFL models). The end devices fetch the models from the nearby FNs for on-board storage and inferences.

We organize the rest of the paper as follows: In Section II, we present some of the existing work in literature. We define our network architecture and formulate the distribution scheme for Magnum in Section III. We then present and discuss our observations on deploying Magnum and simulating it on a B5G environment in Section IV and then finally conclude in Section V.

II. Related Work

In this section, we briefly discuss some of the existing literature categorized as: 1. Decentralized ML and 2. Model Sharing and Aggregation techniques.

A. Decentralized ML

Industries use ML methods for addressing a myriad of applications, which usually include data management, analysis, and security. Zhao *et al.* [4] proposed an algorithm for data clustering while maintaining user privacy. Additionally, blockchain is a popular technique for sharing information. Consensus and computational complexities involved in such solutions increase the overhead of the devices. Moreover, they are open to threats like majority attack and double-spending. Towards this, Tanwar *et al.* [5] proposed a machine learning-based solution for overcoming this issue. Yu *et al.* [6] highlighted the data-driven challenges involved in industry 4.0 and its applications and proposed a big data solution for fault detection and diagnosis. Predictive maintenance is also an essential attribute in industrial scenarios. Susto *et al.* [7] proposed an ML-based model using quantitative machine health indicators for determining and predicting maintenance needs and its outcomes. Data poisoning is yet another concerning factor in dealing with volumes of sensitive data in industrial scenarios. Towards this, Chen *et al.* [8] proposed a routine for determining the number of loops for training a model for detecting data poison.

B. Model Sharing and Aggregation

Yan *et al.* [9] demonstrated a distributed power allocation mechanism for the edge users in distributed wireless networks for enhancing the spectrum efficiency using a Federated Learning (FL) system. One prime drawback of FL is its aggregation, which is dependent on centralized servers. Towards this, the authors in [10] proposed a distributed learning approach for overcoming this issue. Similarly, Wang *et al.* [11] proposed a theoretical model for determining the tradeoff between local and global aggregations in FL. Lu *et al.* [12] designed a secure data sharing framework for distributed parties using blockchain. They shared FL models instead of the actual data for enhancing security. Li *et al.* [13] exploited the features of TFL to develop a system for recommending places according to interest as users move from one location to the other. Researchers have also been exploiting the applications of TFL in a plethora of applications.

C. Synthesis

Researchers have been developing novel methods for sharing and delivering ML models. However, there exists a lacuna in a precisely distributed framework for sharing trained TFL models among industrial plants/branches. Towards this, we propose Magnum, a lightweight ML model sharing scheme, to bridge the gap between the industrial plants and their accessibility to the TFL models. Further, blockchain-based methods involve complex operations for proof-of-work, which is undesired in privately operating operations. We envision the proposed framework to enable the sharing of models, speed up the process of training and delivery for the requesting branches, and lower the computations necessary for deployment. The proposed framework is independent of the dataset and the TFL model. Magnum may be used by industries for training a base model and sharing it with the geographically distributed branches for applications ranging from defect analysis [14] to security [15]. Other use cases for Magnum includes, but not limited to, detection of ice on windmill blades [16], multistage TFL models for vibration-based fault diagnostics [17], diagnosis in sparse auto-encoders [18], and others.

III. System Model

In this section, we first present the proposed network architecture for realising Magnum and then formulate the proposed scheme for delivering/storing the blocks close to the branches demanding for the models. It may be noted that we refer to branches and plants interchangeably.

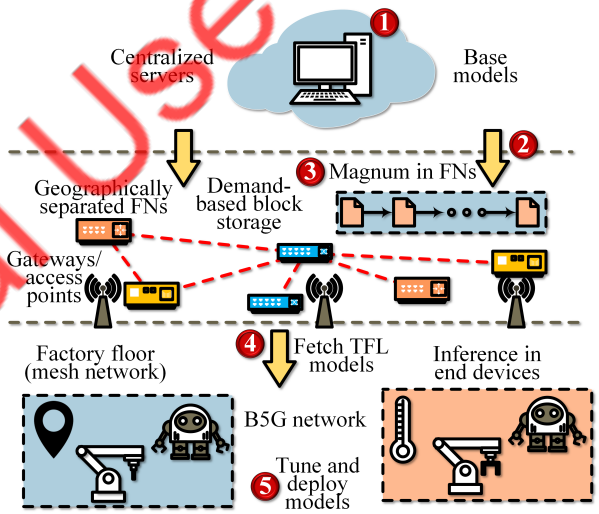


Figure 2: Network architecture and information flow.

A. Network Architecture

We consider the network architecture in Fig. 2 for realising the proposed Magnum framework. We consider a set of centralized servers $S = \{s_1, s_2, ..., s_p\}$ that develop the pre-trained base models (Step 1). The same authorities insert these models into the blocks of the magazine for distributing among the FNs in different networks (Step 2). The distribution is based on the location of the demands from the plants. We consider a set of models $M = \{m_1, m_2, ..., m_n\}$ from the servers and their corresponding set of weights as $W = \{w_1^{m_1}, w_2^{m_2}, ..., w_n^{m_n}\}$. We store these models and weights in the blocks and then distribute among the set of FNs $F = \{f_1, f_2, f_3, ..., f_q\}$ close to the locations where demand for the particular block is high (Step 3). This helps in reducing delays in accessing the models in the proposed distributed framework. We consider a block in the network b_1 among the set $B = \{b_1^{m_1}, b_2^{m_2}, ..., b_n^{m_n}\}$ and send it to an arbitrary f_j for storage. The set of end devices $E = \{e_1, e_2, ..., e_r\}$ in the industrial plants download

the necessary blocks (Step 4) and tune it accordingly before deployment (Step 5). We consider a B5G network on the factory floor, which reduces fetching time from the FNs and transmissions across the devices. We use the FNs only for storing Magnum (series of TFL models). The end devices fetch the models from the nearby FNs for on-board storage and inferences. We now present the selection of the subsets from F for distributing the blocks in B.

B. On-Demand block Distribution

The central servers is where the magazine containing N independent blocks $B_{[1:N]}$ are present. We assume the i^{th} block to be of P_i bits. The distribution of the blocks is a two stage process: 1) Placement and 2) Delivery. During the placement stage, we populate the memory in e_i and we denote the content as denoted by Z_i^c at the end of the placement stage. The requests from the end devices in E via the FNs f_j are revealed after the completion of the placement stage. We use the generic demand evaluation model from the works of [19] to determine the demand (D_i^c) from E. We calculate the demand function as:

$$D_i^c(t) = \frac{\mathcal{N}_c^e}{T_c^{decay}} . \exp\left(\frac{t-\theta_c}{T_c^{decay}}\right) \tag{1}$$

where $\mathcal{N}_c^e$ is the number of end users requesting to access the blocks, T_c^{decay} is the popularity decay time, θ_c is the introductory time or release date of a block, and t is the current time instant. From equation 1, we derive the Popularity $P(t)$ and Rate of requests per user $R(t)$ of the i^{th} block as:

$$P_i(t) = \frac{D_i^c(t)}{R_i(t)} \tag{2}$$

where $R_i(t) = \sum_c D_i^c(t)$. We group the FNs from F as G_k^f based on the equations 1 to find the most requested/demanded block by the end devices and also the most popular block. We distribute these blocks to the identified groups of FNs, which the end users access when necessary.

C. Block Transfer

We consider a B5G scenario for distributing the blocks from the FNs identified in G_k^f. For the signals $y = hx + \sigma$, where h is the channel gain, x is the actual signal and σ is the noise, we consider the spreading (A_{spr}) and molecular absorption (A_{abs}) losses in the network to optimise the time necessary for the blocks to reach the FNs. Both spreading and molecular absorption are functions of the center frequencies $(\mathcal{F})$ and distance (d). We calculate these values in (dB) as:

$$A_{spr} = 20log\left(\frac{4\pi\mathcal{F}d}{c}\right) \tag{3}$$

where c is the speed of light and

$$A_{abs} = k(\mathcal{F})d10\log_{10} e \tag{4}$$

where $k(\mathcal{F})$ is the absorption coefficient. We calculate the overall effect on the signal as $A(\mathcal{F}, d) = (A_{spr} + A_{abs})$. For a signal transmitted with power ρ, the receiver receives the signal with power $A^{-1}(\mathcal{F}, d) \times \rho$. Using these factors, we calculate the data rate (DR) of a link (l) in a B5G network as $DR_l = BW_l \times log(1 + SNR_l)$:

$$DR_l = BW_l \times log\left(1 + \frac{\rho A_l^{-1}(\mathcal{F}, d)}{\sigma_l(\mathcal{T}, BW_l)}\right) \tag{5}$$

where BW_l is the bandwidth of the link ranging from $250 - 500$ GHz, SNR_l is the signal to noise ratio, $\sigma(\mathcal{T}, BW)$ is the Johnson Nyquist noise at $\mathcal{T}$ temperature, represented as $\sigma(\mathcal{T}, BW) = k_B \times \mathcal{T} \times BW$, where k_B is Boltzmann constant. The blocks use these links to travel to the FNs in G_k^f through multiple hops (H_k) in the network. Using the expression for DR in equation 5, we calculate the transmission time (t_i^{trans}) of the i^{th} block of size b_i^{size} as:

$$t_{b_i}^{trans} = \sum_{k=1}^{|G_k^f|} \sum_{l=1}^{H_k} \frac{b_i^{size}}{DR_l} \tag{6}$$

Utility Function: We design the proposed model for minimizing the time for transferring the blocks to the FNs. Towards this, we consider the number of hops in the network to reach each FN in G_k^f by each block b in B. Mathematically,

$$T_{opt} = \text{Min} \sum_{k=1}^{|G_k^f|} \sum_{i=1}^{B} \sum_{l=1}^{H_k} t_{b_i}^{trans} \tag{7}$$

subject to $d \leq d_{max}$, $\mathcal{F} \leq \mathcal{F}_{max}$, $\sigma_c < \rho$, $\sigma_c < \sigma_{max}$, $\mathcal{E}_{loss} < \mathcal{E}_{total}$, and $(A_{spr}, A_{abs}) \geq 0$. These constraints represent the following (left to right): first two constraints imply that the distance of transmission and the frequency in the B5G channels is less than or equal to the maximum allowable thresholds; third constraint implies that the amount of channel noise σ_c present in the link is always less than the total power of the signals ρ; fourth constraint states that the channel noise in the link is less than the maximum noise permitted σ_{max}; fifth constraint specifies that the amount of energy loss $\mathcal{E}_{loss}$ incurred in the link is less than the total energy of the link $\mathcal{E}_{total}$; sixth constraint represents positive values for A_{spr} and A_{abs}.

Theorem 1. *The utility function (T_{opt}) in equation 7 is convex.*

Proof. We prove the convexity of the proposed utility function in equation 7 using a Hessian Matrix. We calculate the first derivative of the expressions as:

$$T_i = \left(\frac{S_1}{DR_1}\right) + \left(\frac{S_2}{DR_2}\right) + ... + \left(\frac{S_n}{DR_n}\right) \tag{8}$$

We observe that they have unit values in all cases except that for those with respect to data rate, which are in the range of $\frac{-1}{DR_1^2}$ to $\frac{-1}{DR_n^2}$. On calculating the second derivative of the equation, we obtain the Hessian Matrix $\mathcal{H}(T_i)$ as:

$$\mathcal{H}(T_i) = \begin{bmatrix} \frac{d^2T}{ds_1^2} \cdots & & \cdots \frac{d^2T}{ds_1 dDR_n} \\ & \ddots & \\ \frac{d^2t}{dDR_n ds_1} & & \frac{d^2T}{dDR_n^2} \end{bmatrix}$$

On solving the derivatives, we observe that $\mathcal{H}(T_i)$ contains 0 in every cell except the diagonals which range from $\frac{2}{DR_1^3}$ to $\frac{2}{DR_n^3}$, implying values greater than 0. We obtain a 2n $\times$ 2n matrix towards the end and since $\mathcal{H}(T_i) \geq 0$, and its transpose is equal to the original resultant matrix, we prove the convexity of the objective function equation. □

D. Solution Approach

We solve the proposed optimization function in equation 7 for transferring the blocks from the FNs. For simplicity in representation, we consider the hops and reduce the expression as:

$$T_i = \sum_{l=1}^{H} \left(\frac{b_i^{size}}{DR_l} \right)$$
$$T_i = \frac{b_i^{size}}{BW \times log(1 + SNR)} \tag{9}$$

In this section, we recursively solve the equation 9 using the branch and bound method, primarily based around $A(\mathcal{F}, d)$. We aim to minimize the value of the objective function by focusing on A_{spr}, A_{abs} and assign them as decision variables of our optimization problem. Mathematically,

$$\text{Min } Z = A_{spr} + A_{abs} \tag{10}$$

subject to the constraints in equation 7. Expanding each constraint for the branch and bound method, we obtain the expressions: $A_{spr}d_1 + A_{abs}d_2 \leq d_{max}$, $A_{spr}\mathcal{F}_1 + A_{abs}\mathcal{F}_2 \leq \mathcal{F}_{max}$, $A_{spr}\sigma_1 + A_{abs}\sigma_2 < \rho$, $A_{spr}\sigma_{c1} + A_{abs}\sigma_{c2} < \sigma_{max}$, and $A_{spr}\mathcal{E}_1 + A_{abs}\mathcal{E}_2 < \mathcal{E}_{total}$. We use Branch and Bound minimization technique to the Nonlinear Programming problem to obtain the smallest possible values of A_{spr} and A_{abs}. Depending upon the number of constraint equations present, we identify the total combination of possible equations. Typically, the number of combinations possible is $\binom{\beta}{\mathcal{D}}$, where β is the number of constraints and $\mathcal{D}$ is the number of decision variables. For each of the expanded equations, we consider the solutions that yield the minimum value for T_i. We explain the same in Algorithm 1 to illustrate one solution for A_{spr} and A_{abs} using 2 equations for FNs from G_k^f for some arbitrary k. The overall process follows a similar optimization procedure for all the possible combinations and determining minimal A_{spr} and A_{abs} values.

As we obtain the minimum values of A_{spr} and A_{abs}, we substitute the values equation 5 to minimize the T_i in 7. For an error tolerance of ϵ, the proposed routine in Algorithm 1 has a time complexity of $\mathcal{O}(\frac{N^2\beta}{\epsilon})$, where β is the corresponding number of constraints for the branch and bound method. Asymptotically, Algorithm 1 has a time complexity of $\mathcal{O}(N^2)$.

E. Delay at End Devices

We define the time necessary for the end devices/users to fetch the blocks using a modified expression of equation 7. In addition to the transmission time, the end devices need to search through the entire magazine for the appropriate block before downloading. We represent the optimization function for determining the delay for end users as:

$$\text{Min } T_i^{user} = \sum_{i=1}^{|B|} \mathcal{S}_i + \sum_{l=1}^{H} \frac{b_j^{size}}{DR_l} \tag{11}$$

where $\mathcal{S}$ is the time necessary for searching through one block and the rest of the variables are the same as discussed in the previous sections. We use the same constraints as in equation 7 with an additional constraint that $|B| \leq |B|_{max}$, which restricts the search space to the number of blocks in the magazine.

Algorithm 1: Optimizing the block transfer from the FNs

Input: d_{max}, $\mathcal{F}_{max}$, ρ, σ_{max}, $\mathcal{E}_{total}, G_k^f$
Result: Z optimized - A_{spr}, A_{abs}
initialization;
for *i = 0 to N, (N = 10)* **do**
  Compute A_{spr} and A_{abs} using equations **??**, **??** ;
  // Considering one possible combination of constraint equations
  Compute Z using A_{spr} and A_{abs} ;
  // Values of A_{spr}, A_{abs} from previous step
  Compute lower bounds of branch 1 using A_{abs} and A_{spr};
  Compute upper bounds of branch 1 using (m, n);
  // (m, n) rounded up values of A_{spr}, A_{abs}
  Consider lower bound higher decimal values in A_{spr} and A_{abs} for next branch;
  Determine new values for A_{spr} and A_{abs};
  if $Z_n > Z_1$ **then**
    Particular node is not feasible ;
  else
    continue;
  end
  if *Upper bound = Lower bound* **then**
    Optimal solution: Min A_{spr}, A_{abs} ;
  else
    Continue branch and bound until Upper bound = Lower bound;
  end
end

IV. Performance Evaluation

We perform our experiments on an Intel i5, 2.5GHz system with 8 GB RAM, running on Google Colaboratory Kernels. We design and implement the proposed blocks and magazine on the mentioned system and simulate the B5G environment on the same. We use Python 3.5 for realising 5 TFL models to establish a proof of concept for the proposed Magnum framework using the MNIST and CIFAR-10 datasets. We consider 6 models in this work: Neural Network (NN), Random Forest (RF), Decision Tree (DT), Support Vector Machine (SVM), Gaussian Naive Bayes (GNB), and ResNet50. We simulate the B5G network on an area of 50×50 m^2. We deploy 200 devices

Table I: Comparison of model sizes before and after inserting them into the blocks.

Models	Actual Size	Magnum	BlockTDM
NN	4 Kb	5.6 Kb	6.9 Kb
RF	7.80 Mb	9.3 Mb	9.7 Mb
DT	89 Kb	92.3 Kb	93.6 Kb
SVM	15 Mb	16.8 Mb	17.0 Mb
GNB	17.6 Kb	20.1 Kb	21.6 Kb
ResNet50	1.2 Mb	1.53 Mb	1.91 Mb

on the factory floor and design our environment for testing the proposed Magnum framework according to the architecture discussed in Section III-A and Fig. 2. We present and discuss our observations in the subsequent sections in comparison with the works of Zhaofeng [2] and Lu *et al.* [12]. While the work in [12] focuses on enhancing security, they proposed including federated learning into the consensus mechanism. On the other hand, the authors in [2] (BlockTDM) adopted conventional consensus method. Since BlockTDM has lower resource demands, we use it as the benchmark solution for comparing the Magnum framework.

A. Block size

We present the size of each block on storing the models of each kind. Table I represents the comparison between the sizes of 6 models (actual) after training and its corresponding size on inserting them into the blocks (Magnum and BlockTDM). We observe that the actual size of the model is less than its corresponding size in the block. The increase in the block size is due to extraneous variables such as addresses to the next block and other control variables necessary for the functioning of the Magnum framework. The size of the blocks depends on the size of the problem, the training data, algorithm, feature size, and others. We observe an increase of 23% in the case of BlockTDM as it adds the necessary mining details such as ID, consensus mechanism, and others. Focusing on Magnum, we observe that the RF model generates sizeable blocks (9.3 Mb) and a 19% increase in model size in the block. Such sizes appear because the RF models store multiple decision trees. On the other hand, we observe a maximum of 40% increase in size on writing into a block in the case of the NN models. Intuitively, we observe such values of block sizes as some amount of slack memory usage is always taken into consideration during storage. We notice that the SVM models generate blocks of the largest size. This is because of the computation of the kernel matrix involved in SVMs, which is relatively large. In case we remove the kernel matrix, the size of the block will reduce. However, on the removal of the matrix, the SVM models need to recalculate the matrix values repeatedly, which increases delay.

B. Accuracy

It is crucial to ensure the reliability of the pre-trained base models contained in them before sharing the blocks with different industrial branches. For the same dataset, we vary the parameters of each model and record their accuracy before and after inserting them into the blocks for a comprehensive view of each block. We present our observations in Table II, where we compare the accuracy of the models before and after inserting them into the blocks. As expected, we observe that the accuracies of each model remain unchanged for both Magnum and BlockTDM. We infer that the blocks are transmitted and stored at the FNs without degrading the quality of the model or distortion during transmission, proving the reliability of the proposed Magnum framework.

Table II: Comparison of accuracies before and after inserting into the blocks.

Models	Actual Model	Magnum	BlockTDM
NN	98.2%	98.2%	98.2%
RF	93.8%	93.8%	93.8%
DT	98.1%	98.1%	98.1%
SVM	93.4%	93.4%	93.4%
GNB	91.5%	91.5%	91.5%
ResNet50	91.3%	91.3%	91.3%

Table III: Memory and CPU usage by the blocks.

Models	Memory Usage		CPU Usage	
	Magnum	BlockTDM	Magnum	BlockTDM
NN	0.3%	0.5%	7.79%	8.01%
RF	4.7%	5.6%	33.88%	34.45%
DT	2.4%	3.1%	28.25%	28.88%
SVM	5.9%	6.4%	61.54%	62.22%
GNB	0.9%	1.3%	19.46%	19.74%
ResNet50	2.9%	3.6%	20.75%	24.12%

C. Memory Usage

We analyze the percentage of memory used by the blocks at different time instances in an FN. We capture the memory usage percentage of a block by distributing the blocks to the FNs at multiple instants and present them in Table III. Across multiple instances, we observe a constant amount of memory usage with respect to each model. We notice that the memory usage corresponds to our observations in Table I and confirms that the block size $\propto$ memory usage for each block. Moreover, due to additional content specific to mining, we observe an increase of 1% in memory consumption in the case of BlockTDM. We notice that the SVM model consumes the maximum memory, followed by the neural network model. We infer that the selection of the devices/FNs for storing the blocks depending on the type of the model is necessary for optimized operations of the Magnum chain.

D. CPU usage

We record the CPU usage while writing and reading the models from the blocks over multiple instances and present them in Fig. 3. We observe that the blocks consume a constant amount of CPU percentage over all the iterations, irrespective of the models. We observe that on average, BlockTDM demonstrates 1% more CPU consumption because of the mining process in Table III. It may be noted that we observe such small increase as we use an i5 processor during implementation and will demonstrate higher difference in case of resource-constrained devices. The neural network model has a minimum requirement (in Magnum) of less than 10%, while all the others demonstrate the need for more than 20% of the CPU.

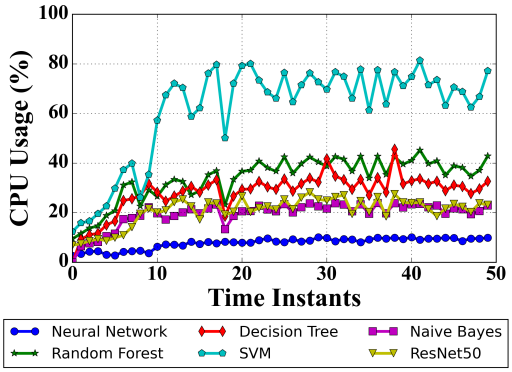


Figure 3: Percentage of CPU usage at various instances for different blocks.

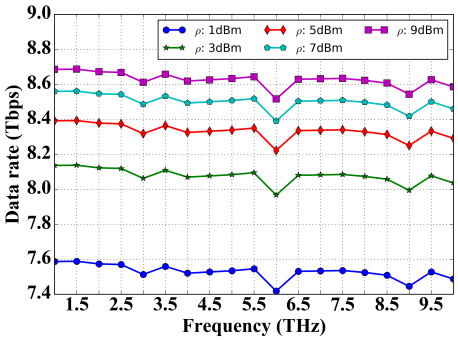


Figure 4: Data rate in the B5G channels.

Although all the models demonstrate a maximum of 40% CPU usage, the SVM model demonstrates the requirement of almost 80% of the CPU. We attribute this behavior to the fetching and computations across the kernel matrix. We comment that the different blocks require a varying percentage of the CPU, depending on the type of the model, which mandates the need for optimized device/FN selection.

E. Data rate in the B5G channel

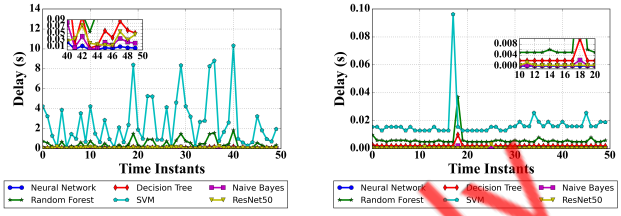
We numerically compute the data rates in the B5G channels by accounting for the effects due to A_{spr} and A_{abs} in equations 3 and 4. We then calculate the data rate using equation 5 for a bandwidth of 500 GHz. Fig. 4 depicts the data rates with varying ρ values over different center frequencies. As expected, we observe that the data rate increases with ρ. From these values, we calculate the delays for transmission over the network with data rates varying in the range of $7.5-8.5$ Tbps.

Table IV: Writing time of the models into the blocks.

Models	Magnum	BlockTDM
NN	0.006 s	0.06 s
RF	0.475 s	17.61 s
DT	0.057 s	0.143 s
SVM	2.664 s	74.73 s
GNB	0.023 s	0.083 s
ResNet50	0.062 s	2.628 s

F. Delays

We categorize the delays necessary for operating under the Magnum framework into two categories: block 1) writing, 2) reading, and 3) transmission time.



(a) Writing time. (b) Reading time.

Figure 5: Writing and reading time for each block.

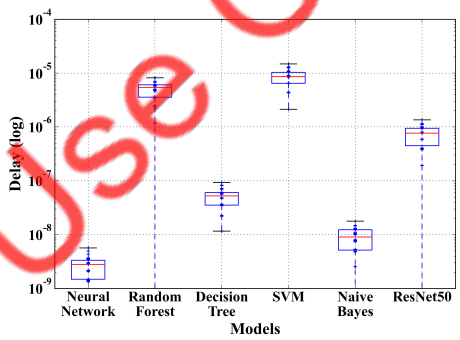


Figure 6: Transmission delay for the blocks to reach the end devices from the FNs.

Block Writing Time: We define the block writing time as the time necessary for the models and their specifics to be written/loaded on the block. We record these delays over multiple instants for each of the models considered in this work and present our observations in Fig. 5(a). We observe that the nature of all the curves is similar across all models (less than 1 second), apart from the SVM model (almost 10 seconds). We attribute this behavior to the size of the blocks discussed in Table I. On the other hand, BlockTDM adds each block after mining the data, which incurs additional delay (74.74 s), as shown in Table IV. The delay is lower in case of smaller models and increases as the model size increases. Such delays are unacceptable in real-time environments, which introduces operational challenges. From these observations, we infer that the writing time is directly proportional to the size of a block and the removal of mining time in Magnum saves significant time, specifically for larger models.

Block Reading Time: We define the block reading time as the time necessary for the devices to fetch the models and their specifics from the blocks. We record these delays from multiple instances and present them in Fig. 5(b). We observe that, on average, the Magnum framework requires an overall time of 0.01 seconds for reading the blocks. We observe the least amount of reading delay for Neural Network models. It takes an infinitesimally small amount of time to read the model details stored in the blocks compared to that of the

writing time in Fig. 5(a). We observe a similar trend that with increasing size of the blocks, the reading time also increases. We comment that the model fetching time is limited to 0.1 seconds for blocks as large as those for SVM, implying minimal delay of the proposed system.

Transmission Delay: We present the transmission time for transferring the blocks from the FNs to the densely deployed devices on the factory floor. As expected, we notice that the delays in Fig. 6 correspond to the block sizes. It may be noted that the transmission delays in Fig. 6 are in log scale for better representation due to the diminutive delay values for NN, DT, and naive bayes models. In the worst case for the SVM models, we observe a maximum of 16 μs delay in transferring the model to the FNs and a minimum of 4.7 ns in case of neural network.

V. Conclusion

In this work, we proposed a lightweight magazine for storing TFL models for industries in the form of blocks. The central servers in an industry train base models and insert them into the blocks. We formulate a model for identifying the locations where demand for a particular block is high and distribute it. The end users (plants operating under an industry) identify the necessary models and fetch them from these blocks. They tune the parameters according to their need and deploy them after training. We further propose the use of a cloud-fog-based architecture to enhance the access to the geographically spread plants and also to minimize transmission and fetching delays. The proposed scheme Magnum helps reduce the time for training and minimize storage and data redundancy. We further propose the use of B5G communication technologies for reducing transmission delays. Through extensive experiments (for Magnum) and simulation results (for B5G networks), we presented the feasibility of the proposed framework with respect to the block size, CPU and memory usage, and delays. In the future, we plan to extend this work by designing access policies and enhancing security for the blocks.

References

[1] H. Liang, W. Fu, and F. Yi, "A Survey of Recent Advances in Transfer Learning," in *Proceedings of IEEE* 19^{th} *International Conference on Communication Technology (ICCT)*, 2019, pp. 1516–1523.

[2] M. Zhaofeng, W. Xiaochang, D. K. Jain, H. Khan, G. Hongmin, and W. Zhen, "A Blockchain-Based Trusted Data Management Scheme in Edge Computing," *IEEE Trans. Ind. Informat*, vol. 16, no. 3, pp. 2013–2021, 2020.

[3] J. M. Jornet and I. F. Akyildiz, "Channel Modeling and Capacity Analysis for Electromagnetic Wireless Nanonetworks in the Terahertz Band," *IEEE Trans. Wireless Commun.*, vol. 10, no. 10, pp. 3211–3221, 2011.

[4] Y. Zhao, L. T. Yang, and J. Sun, "A Secure High-Order CFS Algorithm on Clouds for Industrial Internet of Things," *IEEE Trans. Ind. Informat*, vol. 14, no. 8, pp. 3766–3774, 2018.

[5] S. Tanwar, Q. Bhatia, P. Patel, A. Kumari, P. K. Singh, and W. Hong, "Machine Learning Adoption in Blockchain-Based Smart Applications: The Challenges, and a Way Forward," *IEEE Access*, vol. 8, pp. 474–488, 2020.

[6] W. Yu, T. Dillon, F. Mostafa, W. Rahayu, and Y. Liu, "A Global Manufacturing Big Data Ecosystem for Fault Detection in Predictive Maintenance," *IEEE Trans. Ind. Informat*, vol. 16, no. 1, pp. 183–192, 2020.

[7] G. A. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi, "Machine Learning for Predictive Maintenance: A Multiple Classifier Approach," *IEEE Trans. Ind. Informat*, vol. 11, no. 3, pp. 812–820, 2015.

[8] Y. Chen, Y. Mao, H. Liang, S. Yu, Y. Wei, and S. Leng, "Data Poison Detection Schemes for Distributed Machine Learning," *IEEE Access*, vol. 8, pp. 7442–7454, 2020.

[9] M. Yan, B. Chen, G. Feng, and S. Qin, "Federated Cooperation and Augmentation for Power Allocation in Decentralized Wireless Networks," *IEEE Access*, vol. 8, pp. 48 088–48 100, 2020.

[10] S. Savazzi, M. Nicoli, and V. Rampa, "Federated Learning With Cooperating Devices: A Consensus Approach for Massive IoT Networks," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4641–4654, 2020.

[11] 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," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, 2019.

[12] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and Federated Learning for Privacy-Preserved Data Sharing in Industrial IoT," *IEEE Trans. Ind. Informat*, vol. 16, no. 6, pp. 4177–4186, 2020.

[13] D. Li, Z. Gong, and D. Zhang, "A Common Topic Transfer Learning Model for Crossing City POI Recommendations," *IEEE Trans. Cybern.*, vol. 49, no. 12, pp. 4282–4295, 2019.

[14] K. Imoto, T. Nakai, T. Ike, K. Haruki, and Y. Sato, "A CNN-Based Transfer Learning Method for Defect Classification in Semiconductor Manufacturing," *IEEE Trans. Semicond. Manuf.*, vol. 32, no. 4, pp. 455–459, 2019.

[15] L. Vu, Q. U. Nguyen, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, "Deep Transfer Learning for IoT Attack Detection," *IEEE Access*, vol. 8, pp. 107 335–107 344, 2020.

[16] H. Yun, C. Zhang, C. Hou, and Z. Liu, "An Adaptive Approach for Ice Detection in Wind Turbine With Inductive Transfer Learning," *IEEE Access*, vol. 7, pp. 122 205–122 213, 2019.

[17] J. Zhou, L. Zheng, Y. Wang, and C. Gogu, "A Multistage Deep Transfer Learning Method for Machinery Fault Diagnostics Across Diverse Working Conditions and Devices," *IEEE Access*, vol. 8, pp. 80 879–80 898, 2020.

[18] L. Wen, L. Gao, and X. Li, "A New Deep Transfer Learning Based on Sparse Auto-Encoder for Fault Diagnosis," *IEEE Trans. Syst., Man, Cybern. Syst*, vol. 49, no. 1, pp. 136–144, 2019.

[19] D. De Vleeschauwer and K. Laevens, "Performance of Caching Algorithms for IPTV On-Demand Services," *IEEE Trans. Broadcast.*, vol. 55, no. 2, pp. 491–501, 2009.