Abstract—Fog platform brings the computing power from the remote cloud-side closer to the edge devices to reduce latency, as the unprecedented generation of data causes ineligible latency to process the data in a centralized fashion at the Cloud. In this new setting, edge devices with distributed computing capability, such as sensors, surveillance camera, can communicate with fog nodes with less latency. Furthermore, local computing (at edge side) may improve privacy and trust. In this paper, we present a new method, in which, we decompose the data processing, by dividing them between edge devices and fog nodes, intelligently. We apply active learning on edge devices; and federated learning on the fog node which significantly reduces the data samples to train the model as well as the communication cost. To show the effectiveness of the proposed method, we implemented and evaluated its performance on a benchmark images data set.

Index Terms—Fog Computing, Edge Computing, Active Learning, Federated Learning, Bayesian Neural Network

I. INTRODUCTION

Internet of Thing (IoT) is growing in adoption to become an essential element for the evolution of connected products and related services. To enlarge IoT adoption and its applicability in many different contexts, there is a need for the shift from the cloud-based paradigm towards a fog computing paradigm. In cloud-based paradigm, the devices send all the information to a centralized authority, which processes the data. However, in Fog Computing (FC, [1]) or Edge Computing (EC, [37]) the data processing computation will be distributed among edge devices, fog devices, and the cloud server. FC and EC are interchangeable. This emergence of EC is mainly in response to the heavy workload at the cloud side and the significant latency at the user side. To reduce the delay in fog computing, the concept of fog node is introduced. The Fog Node (FN) [2] is essentially a platform placed between Cloud and Edge Devices (ED) as middleware, and it will further facilitate the ‘things’ to realize their potentials [3]. This change of paradigm will help application domains, such as industrial automation, robotics or autonomous vehicles, where real-time decision making by using machine learning approaches is crucial.

The research leading to these results has received funding from the European Unions Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 764785, FORA-Fog Computing for Robotics and Industrial Automation. This work was further supported by the Danish Innovation Foundation through the DanishCenter for Big Data Analytics and Innovation (DABAI).

Convolutional Neural Network (CNN), which is one sub-type of the artificial neural network and models the system by observing all the data during training at a single place. Notably, the emergence of CNN [27] introduces the common usage of the neural network as it is more efficient concerning computation cost comparing with (fully connected) neural network. However, to train the model, we need to access all the data, which may create communication bottlenecks and privacy issues [28]. Generally, to train a CNN model, users send all the data to a single machine or a cluster of machines in a data center. However, this sharing operation with the central authority in data centers costs both network usages and breaching of user privacy, since the user doesn’t want to share personal, sensitive information, e.g., legally protected medical data.

FL [6] allows the centralized server training models without moving the data to a central location. In particular, FL is used in a distributed way, in which the model is built without direct access to training data, and indeed the data remains in its original location, which provides both data locality and privacy. In the beginning, a server coordinates a set of nodes, each with training data that cannot be accessed by server directly. These nodes train a local model and share individual models with the server. The server uses these individual models to create a federated model and sends the model back to the nodes. Then, another round of local training takes place, and the process continues. Nevertheless, this extra work on edge devices has to be minimized by selecting the most important data samples needed to build the local model. In this context, we want to use Active Learning (AL) as a more effective learning framework. AL chooses training data efficiently when labeling data becomes drastically expensive.

Motivated by the above mentioned, the research issues and possible direction, we propose a new scheme. In literature, there exist some papers that discuss the application of machine-learning algorithm directly on the Fog node platform [30] [31]. As we have already discussed, to efficiently use cloud and fog infrastructure, we need to delegate the work among them. Hence, in this paper, we propose a new efficient privacy-preserving data analytic scheme in Fog environment. We offer using federated learning at the centralized fog devices to create the initial training model. To improve the perfor-
mance further, we recommend using Active Learning at the edge devices, by selecting the sample points effectively. All in all, we propose a possible solution in Edge Computing setting, where the user privacy, training cost, and upload bottleneck are the main issues to address. Our strategy may reduce the training cost by applying AL and preserve the user privacy and reduce the communication by using FL. Moreover, the proof of concept is demonstrated by applying the method on a benchmark dataset.

The remainder of this paper is organized as follows. Section II explains preliminary concepts, in particular, CNN, FL, AL framework, model uncertainty measurement, and previous studies related to our work. In Section III, we introduce the proposed scheme. Section IV covers the details of our experimental design and data collection strategy followed by a discussion on our results. Section V concludes the paper.

II. PRELIMINARY CONCEPTS AND RELATED WORK

In this section, we discuss the different techniques that are essential for our scheme. We also present a brief overview of the major research studies related to our work.

A. Convolutional Neural Network

Convolutional Neural Network (CNN) is proposed in [9] for the first time, which addressed the computational problems imposed by the fully-connected neural network, in particular, deep neural network. It’s composed by the following layers:

- **Convolutional Layer:** This layer helps in largely reducing the dimension by applying the kernel to the input. For example, if we have images with size m*n as input, with a stride equal to one and no padding, after applying the kernel with size k*k, it ends up a matrix with size \([m - (k - 1)] \times [n - (k - 1)]\).

- **Pooling Layer:** This layer performs non-linear downsampling. Maximal and average pooling is the most commonly used in this layer.

- **Activation Layer:** The active layer is the non-linear transformation layer, where both sigmoid and hyperbolic can be used in this layer. We use ReLu in our work, which is abbreviated as rectified linear unit, expressed as:

\[
R(x) = \max(0, x) \quad (1)
\]

- **Fully-connected Layer:** Here, all the neurons in a fully connected layer connect to all activations in the previous layer. We compute the output for node i at layer L, denoted as \(y^{l}_i\), using its weight as \(w_j^{l}\) and input from the previous layer as \(o^{l-1}_j\), i.e.,

\[
y^{l}_i = \sum_j o^{l-1}_j \times w^{l}_{ji} \quad (2)
\]

CNNs commonly have a huge number of parameters, and it will lead to huge communication costs by sending updates for these many values to a server leads. Thus, a simple approach to sharing weight updates is not feasible for larger models. Since uploads are typically much slower than downloads, it is acceptable that users have to download the current model, while compression methods should be applied to the uploaded data.

B. Federated Machine Learning

Federated learning (FL) is a collaborative form of machine learning where the training process is distributed among many users; this enables to build machine learning systems without complete access to training data [6]. In FL, the data remains in its original location, which helps to ensure privacy and reduces communication costs. The server or the central entity does not do most of the work but only coordinates everything by a federation of users. In principle, this idea can be applied to any model for which the criterion of updates can be defined, which naturally includes the methods based on gradient descent, which nowadays most of the popular models do. For instance, linear regression, logistic regression, neural networks, and linear support vector machines can all be used for FL by letting users compute gradients [33] [34].

Concerning data, FL is especially useful in situations where users generate and label data by themselves implicitly. In such a situation, Federated Learning is very powerful since models can be trained with the massive data that actually is not stored and not directly shared with the server at all. We can thus make use of the massive data that we could otherwise not have used without violating the users’ privacy. FL aims to improve communication efficiency and train a high-quality centralized model. The centralized model is trained over the distributed client nodes, which we refer to edge devices in the FC setting. The model is locally trained on every device, and the devices update the refined models to the Fog Node (server node) by sending the parameters of the models. The FC might aggregate the parameters in different ways, for instance, average parameters, choose the best-performed model or sum the weighted parameters.

We define the goal of federate learning to learn a model with parameters embodied in matrix from data stored acrossting numbe of clients (Edge Devices). Suppose the server (FN) distributes the model (at round t) \(W_t\) to N clients for further updating, and the updated models are denoted as \(W^i_t\), \(W^2_t\), \(W^3_t\), ... \(W^N_t\). Then, the clients send the updated models back to the server, and the server updates the model W according to the aggregated information.

\[
W_{t+1} := \sum_i^{N} \alpha_i \cdot W^i_t \quad (3)
\]

Where \(\alpha\) can be uniformly distributed or according to the t – 1 round performance, we use the former one in our work, namely, average the parameters. The learning process can be iteratively carried out.

C. Active Machine Learning

Active learning (AL) is a particular case of machine learning in which a learning algorithm interactively queries the user to obtain the desired outputs at new data points. Typically, AL achieves higher performance with the same number of data samples, using the same method, for example, support
Vector machine, neural network, while with a sophisticated data acquisition [4]. In other words, AL achieves a given performance using less data. Active learning can be divided into two categories: pool-based and stream-based [29]. Pool-based active learning queries the most informative instance from a large pool (See Fig 1), whereas the stream-based one typically draws one at a time from the input source, and the learner must decide whether to query or discard it. In this paper we consider pool-based active learning. The critical point is the way of choosing training data, which is carried out by an interaction between data pool and model: the model strategically picks the new training data according to some specific criteria (refer to Acquisition Function in subsection II-E). The authors in [12] showed that a pool-based support vector machine classifier significantly reduces the needed data to reach some particular level of accuracy in text classification, analogously, [13] for image retrieval application.

Active learning is an appropriate choice when i) labeling data is expensive, or ii) limited data collection. Initially, the researchers fit the machine learning algorithms that mostly work for tabular data to the active learning framework. Recently, it starts registering with the deep neural network, which can work appropriately on image processing by using a neural network.

D. Bayesian Neural Network approximating by Dropout

In this subsection, we will briefly introduce the Bayesian neural network, from which the dropout is applied to approximate the variational inference, proposed by [32]. Bayesian network is defined as placing a prior distribution over the weights of the neural network. Let’s define the weights of neurons as $W = \{w_i\}_{i=1}^{T}$, we are interested in the posterior $p(W|X,Y)$, given all the observations $(X,Y)$. As we know, the posterior is intractable integral from [35]. In [32], it is solved by approximation of the real distribution and Monte Carlo. Thus, we define the approximating variational distribution $q(W_i)$ at layer i as:

$$W_i = M_i \ast \text{diag}([Z_{i,j}]_{j=1}^{K_i}), \quad \text{(4)}$$

where

$$Z_{i,j} \sim \text{Bernoulli}(p_i, \text{dropout}), \quad \text{for} \quad i = 1, ..., L, j = 1, ..., K_i$$

and $M_i$ are variational parameters to be optimized. The diag(.) maps vector to diagonal matrices, whose diagonal are the elements of the vector. $K_i$ indicates the number of neurons in layer i. By given input x and the weights w, which can be sampled from $q(w)$, the predictive distribution of our interest is defined as:

$$p(y^*|x^*, w)q(w)dw \approx \frac{1}{T} \sum_{t=1}^{T} p(y^*|x^*, \tilde{w}_t)$$

with $\tilde{w}_t \sim q(w)$, and it is sampled by applying dropout on the corresponding layer. This is referred to as Monte Carlo dropout (MC-dropout).

E. Acquisition Function

The acquisition function is a measure of how desirable the given data point is for minimizing the loss or maximizing the likelihood. In this paper, we are going to use MC-dropout to sample weights and a particular type of uncertainty-based method called Maximal Entropy to measure the uncertainty, as it outperforms the others reported in [16].

Uncertainty-based methods aim to use uncertain information to enhance the model during the re-training process. It plays the role of the exploitation while acts as the exploration part. We will introduce three different ways to estimate uncertainty.

– **Maximal Entropy**: $H[y|x, D_{train}]$ is the predictive entropy expectation as defined in [10].

$$H[y|x, D_{train}] := -\sum_c \frac{1}{T} \sum_{t=1}^{T} p(y = c|x, w_t)) \cdot \log(\frac{1}{T} \sum_{t=1}^{T} p(y = c|x, w_t)) \quad \text{(7)}$$

– **BALD**: (Bayesian Active Learning by Disagreement [14]) measures the mutual information between data and weights, and it can be interpreted as seeking data points for which the parameters (weights) under the posterior disagrees the most.

$$I[y; w|x, D_{train}] := H[y|x, D_{train}] - \frac{1}{T} \sum_t \sum_c p(y = c|x, w_t) \log p(y = c|x, w_t) \quad \text{(8)}$$

– **Variational ratios**: it maximises the variational ratio by considering the followings [15]:

$$V[x] := 1 - \max_y p(y|x, D_{train}) \quad \text{(9)}$$

It is similar to Maximal Entropy, but less effective as reported in [16].
III. PROPOSED SCHEME

In this section, we discuss the functioning of the different components of the proposed scheme. The primary objective of the proposed scheme is to generate the model by an iterative and distributed way using the fog nodes and edge devices. The overall scheme is shown in Fig. 2, where between edge devices and the cloud server, the fog nodes work as a middleware. Here, a fog node is connected to the edge devices which have a similar task to implement the specific application. For instance, a fog node might be linked to all the surveillance cameras to detect the particular object. Every camera possesses a trained model dispatched by the FN, and it keeps training by the images generated locally under Active Learning framework. It is followed by Federated Learning, namely, the models individually are trained by every device uploading the weights of the (refined) models to FN. The whole process is sketched as follows:

- Firstly, FN trains an initial model “M” using “m” data samples, where “m” is very small which barely helps the model to learn. To generalize, we denote model as $M^t$, where $t$ is the round.
- FN dispatches the model $M^t$ to the edge devices. For example, let’s say, we have four devices, called $E_1$, $E_2$, $E_3$, $E_4$, and each receives $M_t$ from FN.
- All edge devices implement Active Learning locally with Maximal Entropy acquisition function. More specifically, during every acquisition (totally $R$ acquisitions), edge devices train $M^t$ by another “$N$” ($N > m$) data samples.
- Next, Edge devices label the new models. In the example, $E_1$, $E_2$, $E_3$, $E_4$ label their local model as $M_1^t$, $M_2^t$, $M_3^t$, $M_4^t$ respectively.
- Then, the edge devices upload the weights of models $M_1^t$, $M_2^t$, $M_3^t$, $M_4^t$ to the centralized FN.
- FN aggregates the weights either by averaging or choosing the best-trained model, and pass it to next round $t + 1$ if necessary.

In our experiments, we set $m$ equal to 20, and we only consider one round. Moreover, we can update the model on the fog node side during every acquisition.

IV. EXPERIMENTS AND RESULTS

In this section, we discuss the experimental setup along with the data set used for our evaluation, and the results of these experiments.

Experiment Setup:
Initially, we trained the CNN model by 20 images at the centralized node (FN), and then send the model to the edge devices. On the devices side, we further trained the model by additional data points. They are acquired by choosing top 10 data samples that have the highest entropy, and this operation will iterate several times. Notably, the model is independently trained by the edge devices. Then it is followed by updating the refined models from all the devices to the centralized FN. The FN will average the parameters of the models, for the future data analysis.

All the experiments are implemented by Python language, more specifically, the Pytorch package [36]. The codes are run in Mac Os system (High Sierra) with version 10.13.6, with RAM 16 GB.

Data Set:
We implemented the methods on MNIST dataset [8], which is a real data set of handwritten images, with 60000 for training data and 10000 for the test, totally ten classes. All the images have already been pre-processed, built with the size as $28 \times 28$. It is the basic benchmark to test the performance of approaches in machine learning.

A. Experiment I: random sample vs active learning on edge device

To demonstrate the effectiveness of Active Learning, We compare its performance with randomly chosen data, the experiments are carried out on edge devices, as shown in Fig. 3 and 4 for 10 acquisitions and 20 acquisitions. The outcome of every device with active learning outperforms randomly chosen data points. Notably, here we only want to show that AL has better performance than randomly choosing training samples, not for the sake of the state-of-the-art result.

B. Experiment II: AL acquisition number

In this series of experiment, we study how does the number of training data influence the performance by plotting the
learning curve. Recall that during every data acquisition, we include ten additional images for further training. Fig. 5, 6, 7 illustrate the learning curve of edge devices for 10, 20, 40 acquisitions accordingly. Here, we run the experiments for five times and we also plot the standard deviations. Again, the training on edge devices are independent; namely, with the different dataset, which conform to the practical situation, the data is generated locally and independently.

Assume that all the data generated from edge devices are from the same distribution. The first observation is every device has its own learning curve as the data at every device is different though they are from the equal distribution. In addition, we build the data pool by randomly choosing 200 images at every iteration of acquisition, from the whole dataset (10000), in order to reduce the computing cost as all the data in the pool are being measured the uncertainty. It is the main reason we run the experiments several rounds to test the real performance of our method.

Moreover, We compared the accuracy on FN by applying our approach, with the result obtained by training a dataset with the size triple bigger (4*N) than on every edge device (N) as we have four edge devices in our experiment. For instance, if we train the model by 100 data points on the edge device, then we compare it with the result directly training 400 data samples on FN. The details are shown in Table I, and the columns indicate the different number of acquisitions from data pool, during every acquisition we pick ten images, Acq 10 means the model is further trained by another 100 images on every edge device. For the sake of comparison, we directly trained the initial model with 400 images since we have four devices, every device is trained by 100 images. And then we compared the accuracy with two aggregation strategies: average and optimal model. Note that it is arguable that how many images we train directly on FN to compare since the model is not directly trained by 400 images when we apply FL, we train the model by 100 images on every device. Nevertheless, here we train the model by training data with the size equal to the number locally trained on every device time the number of devices, considering the worst case. Notably, when the number of edge devices is large, the advantage of AL is not as obvious as the case (4 edge devices) we demonstrated before. Assuming with 1000 data points, if we have 4 edge devices, every one is trained by 250 images, while if we have 20 devices, then every device 'sees' 50 images. As we can guess, in the second case, the centralized device that uses the averaged weights works worse than one machine trained directly by 1000 images (50 \ll 1000). It can be solved by the communication between devices, cascading the
training process, namely, after one device completes training, shares the weights with the close device.

V. CONCLUSION

In this paper, we for the first time discussed Active Learning in a distributed setting tailored to the so-called Fog platform consisting of distributed edge devices and a centralized fog node. We implemented active learning in edge devices to down scale the necessary training set and reduce the label cost. We presented evidence that it performs similarly to centralized computing with a reduced communication overhead, latency and harvesting the potential privacy benefits. In the future, we will do more experiments on different setting (large number of edge devices) and offering the corresponding solution. In addition, we will study the additional acquisition functions and also address privacy issues in more details.

| TABLE I
| FOG NODE PERFORMANCE WITH/without FEDERATED LEARNING |
|------------------|------------------|------------------|------------------|------------------|
| FN without FL    |      |      |      |      |
| (No. of train data) | 400 | 800 | 1200 | 1600 |
| FN with FL       |      |      |      |      |
| (No. of train data) | 100 | 200 | 300 | 400 |
| Accuracy         |      |      |      |      |
| (FN without FL)  | 0.73 | 0.82 | 0.81 | 0.90 |
| Accuracy         |      |      |      |      |
| (FN with FL (ave)) | 0.8  | 0.90 | 0.88 | 0.91 |
| Accuracy         |      |      |      |      |
| (FN with FL (opt)) | 0.854 | 0.915 | 0.944 | 0.963 |

REFERENCES

[1] Bonomi, Flavio, et al. "Fog computing and its role in the internet of things." Proceedings of the first edition of the MCC workshop on Mobile cloud computing. ACM, 2012.
[2] Bonomi, Flavio, et al. "Fog computing: A platform for internet of things and analytics." Big data and internet of things: A roadmap for smart environments. Springer, Cham, 2014. 169-186.
[3] Dastjerdi, Amir Vahid, and Rajkumar Buyya. "Fog computing: Helping the Internet of Things realize its potential." Computer 49.8 (2016): 112-116.
[4] Settles, Bur. "Active learning literature survey. 2010." Computer Science Technical Report 1648 (2014).
[5] Gal, Yarin, and Zoubin Ghahramani. "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning." International conference on machine learning. 2016.
[6] Koen, Jakub, et al. "Federated learning: Strategies for improving communication efficiency." arXiv preprint arXiv:1610.05492 (2016).
[7] Rasmussen, Carl Edward. "Gaussian processes in machine learning." Advanced lectures on machine learning. Springer, Berlin, Heidelberg, 2004. 63-71.
[8] LeCun, Yann, Corinna Cortes, and C. J. Burges. "MNIST handwritten digit database." AT&T Labs [Online]. Available: http://yann.lecun.com/exdb/mnist 2 (2010).
[9] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.
[10] Shannon, Claude Elwood. "A mathematical theory of communication." Bell system technical journal 27.3 (1948): 379-423.
[11] Wang, Keze, et al. "Cost-effective active learning for deep image classification." IEEE Transactions on Circuits and Systems for Video Technology 27.12 (2017): 2591-2600.
[12] Tong, Simon, and Daphne Koller. "Support vector machine active learning with applications to text classification." Journal of machine learning research 2.Nov (2001): 45-66.
[13] Tong, Simon, and Edward Chang. "Support vector machine active learning for image retrieval." Proceedings of the ninth ACM international conference on Multimedia. ACM, 2001.
[14] Houlsby, Neil, et al. "Bayesian active learning for classification and preference learning." arXiv preprint arXiv:1112.5745 (2011).
[15] Freeman, Linton C. Elementary applied statistics: for students in behavioral science. John Wiley and Sons, 1965.
[16] Gal, Yarin, Riashat Islam, and Zoubin Ghahramani. "Deep bayesian active learning with image data." arXiv preprint arXiv:1703.02910 (2017).
[17] McMahan, H. Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." arXiv preprint arXiv:1602.05629 (2016).
[18] Koen, Jakub, Brendan McMahan, and Daniel Ramage. "Federated optimization: Distributed optimization beyond the datacenter." arXiv preprint arXiv:1511.03575 (2015).
[19] LeCun, Y., Haffner, P., Bottou, L. and Bengio, Y., 1999. Object recognition with gradient-based learning: A survey of results. In Proceedings of the same conference.
[20] Hoi, Steven CH, et al. "Batch mode active learning and its application to medical image classification." Proceedings of the 3rd international conference on Machine learning.
[21] Liu, Ying. "Active learning with support vector machine applied to gene expression data for cancer classification." Journal of chemical information and computer sciences 44.6 (2004): 1936-1941.
[22] Thompson, Cynthia A., Mary Elaine Califf, and Raymond J. Mooney. "Active learning for natural language parsing and information extraction." ICML, 1999.
[23] Osugi, Thomas, Deng Kim, and Stephen Scott. "Balancing exploration and exploitation: A new algorithm for active machine learning." Data Mining, Fifth IEEE International Conference on. IEEE, 2005.
[24] Lakshminarayanan, B., Pritzel, A. and Blundell, C., 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In Advances in Neural Information Processing Systems (pp. 6402-6413).
[25] Osband, I., Blundell, C., Pritzel, A. and Van Roy, B., 2016. Deep exploration via bootstrapped DQN. In Advances in neural information processing systems (pp. 4026-4034).
[26] Gal, Y. and Ghahramani, Z., 2016, June. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In International conference on machine learning (pp. 1050-1059).
[27] Gal, Y., Haffner, P., Bottou, L. and Bengio, Y., 1999. Object recognition with gradient-based learning: A review of results. In Proceedings of the same conference.
[28] Dwork, C., 2008, April. Differential privacy: A survey of results. In International Conference on Theory and Applications of Models of Computation (pp. 1-19). Springer, Berlin, Heidelberg.
[29] Settles, B., 2012. Active learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 6(1), pp.1-114.
[30] Tang, B., Chen, Z., Hefferman, G., Wei, T., He, H. and Yang, Q., 2015, October: A hierarchical distributed fog computing architecture for big data analysis in smart cities. In In Proceedings of the ASE BigData and SocialInformatics 2015 (p. 28). ACM.
[31] Diro, A.A. and Chilamkurti, N., 2018. Distributed attack detection scheme using deep learning approach for Internet of Things. Future Generation Computer Systems, 82, pp.761-768.
[32] Gal, Y. and Ghahramani, Z. 2015. Bayesian convolutional neural networks with Bernoulli approximate variational inference. arXiv preprint arXiv:1506.02158.
[33] Konec, J., McMahan, H.B., Ramage, D. and Richtrik, P., 2016. Federated optimization: Distributed machine learning for on-device inference. arXiv preprint arXiv:1610.00527.
[34] Hong, D. and Si, L., 2012, August. Mixture model with multiple centralized retrieval algorithms for result merging in federated search. In Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval (pp. 821-830). ACM.
[35] Blei, D.M., Kucukelbir, A. and McAuliffe, J.D., 2017. Variational inference: A review for statisticians. Journal of the American Statistical Association, 112(518), pp.859-877.
[36] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. and Lerer, A., 2017. Automatic differentiation in pytorch.
[37] Shi, W., Cao, J., Zhang, Q., Li. Y and Xu, L., 2016. Edge computing: Vision and challenges. IEEE Internet of Things Journal, 3(5), pp.637-646.