Similarity-based data transmission reduction solution for edge-cloud collaborative AI

Aya Elouali
Department of computer science
Technology and Computation,
University of Alicante, Alicante, Spain
ae56@alu.ua.es

Higinio Mora Mora
Department of computer science
Technology and Computation,
University of Alicante, Alicante, Spain
hmora@ua.es

Francisco J. Mora Gimeno
Department of computer science
Technology and Computation,
University of Alicante, Alicante, Spain
fjmora@ua.es

ABSTRACT

Edge-cloud collaborative processing for IoT data is a relatively new approach that tries to solve processing and network issues in IoT systems. It consists of splitting the processing done by a Neural Network model into edge part and cloud part in order to solve network, privacy and load issues. However, it also has it shortcomings such as the big size of the edge part's output that has to be transmitted to the cloud. In this paper, we are proposing a data transmission reduction method for edge-cloud collaborative solutions that is based on data similarities in stationary objects. The performed experiments proved that we were able to reduce 62% of the data sent.

CCS CONCEPTS

• **Computing methodologies**; • **Artificial intelligence**; • **Distributed artificial intelligence**;

KEYWORDS

Neural Network splitting, Edge, Cloud

ACM Reference Format:
Aya Elouali, Higinio Mora Mora, and Francisco J. Mora Gimeno. 2022. Similarity-based data transmission reduction solution for edge-cloud collaborative AI. In *2022 5th Artificial Intelligence and Cloud Computing Conference (AICCC) (AICCC 2022), December 17–19, 2022, Osaka, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3582099.3582107

1 INTRODUCTION

Artificial Intelligence of Things (AIoT) is a research field that has attracted a lot of attention in the last few years and has reached a mature and stable state [9]. However, it still has some unsolved issues such as the location of the processing. Since the AI models are heavy and resource consuming, using them on the connected object itself or other devices at the edge will result in a big load on this device and a long processing time [8] [29]. Even so, offloading the processing task to a more powerful entity like the cloud will also lead to some problems such as long communication delay and response time as well as privacy issues [5].

One of the solutions that has been proposed recently is edge-cloud collaborative data processing, where the AI model responsible for the processing (generally a Neural Network (NN)) is split into two parts, the first part is used at the edge and the second in the cloud. This solution consists of partitioning the NN in the most appropriate position to reduce the data sent to the cloud and therefore network related issues, benefit from the powerful resources of the cloud, not overload the IoT object and preserve data privacy [30] [27] [12]. However, the output of hidden layers of a NN, is generally voluminous or layers of smaller output size are at the end of the network which means burdening the edge device with the biggest part of processing.

In this paper, we propose a solution to reduce the data to be sent to the cloud while benefiting from the advantages of segmenting the processing task. In IoT systems where data is captured by a stationary object such as cameras and sensors, data captured successively is too much the same [11]. Supposing that to a pre-trained NN (has defined weights) input similarity entails similarity of the hidden layers' outputs, instead of sending the output of the first part of the split DNN directly to the cloud, we propose sending only the difference between this output and the previous one.

This paper is organized as follows: in Section 2 we review most relevant work in the data transmission reduction and Neural Network splitting fields, then we determine how our work is situated compared to them. In Section 3 the proposed solution is presented and explained formally. In section 4 several experiments are conducted in order to confirm the suggestions and test the proposed solution on a real world use case. Section 5 concludes the paper and outlines the future work.

2 STATE OF THE ART

Our solution is a combination of two domains: data transmission reduction and NN splitting. For this, in this section we present the most relevant papers in these two domains.

2.1 Data Transmission Reduction

In the literature, several solutions have been proposed to reduce data sendings from connected objects to servers.

Some of the proposed solutions are based on prediction models. Those solutions consist of using the same model trained to predict the next data on both sides and only transmit poorly predicted values. This solution has been used in different situations such in [16–19] to reduce temperature, humidity, light, and voltage data sent from sensors to the sink node. In [20], authors proposed a prediction model to reduce the periodical information uploaded from smart industrial machines by not sending data of the same value or data increasing/decreasing linearly because it can be easily predicted.

AICCC 2022, December 17–19, 2022, Osaka, Japan

ACM ISBN 978-1-4503-9874-9/22/12…$15.00
https://doi.org/10.1145/3582099.3582107

Other solutions are based on classification models. Those models are trained on the importance of data according to the system, then used at the IoT objects level to decide if this data is worth sending or not. Such in [21] for wearable sensor networks or between connected vehicles and edge server in [22]. Data compression is also a widely used solution to reduce the size of data sent from one entity to another. It consists of reducing the number of bits used to represent the message by applying a compression algorithm such as in [6, 7, 17].

Reducing the data sent can also be based on data aggregation such as in [23, 26] where the authors proposed an aggregation model based on eliminating redundant similar data received from sensor nodes in a cluster. For visual sensor networks, authors of [24] proposed to eliminate near-duplicate images using near-duplicate clustering, seed image selection then deleting the rest of images in the cluster. Sajedi et al.proposed in [25] a data aggregation system based on fuzzy logic in order to manage the dynamic and nonlinear nature of healthcare data and reduce the load on the network.

2.2 Neural Network Models Splitting

In order to benefit from the advantages of collaborative processing such as reducing the energy and resource consumption for the IoT object, reducing the load on the cloud, gaining time as well as augmenting the security of the data by not sending the real data via the network. Research work in this area is mostly focused on defining algorithms and platforms that search for the most convenient splitting point. Taking in consideration the characteristics of the edge device, the network, the input data, the output of the layer, and the task to be achieved. The defined algorithms return the layer that gives the best characteristics of the split model.

The proposed solutions are generally iterative algorithms that checks if the conditions are met for all layers such as in [8], where authors presented NNs auto-splitting algorithm that analyzes the data size that needs to be transmitted at each layer, then solves the optimization problem to generate a list of splitting possibilities. In [12] also, authors defined a set of iterative algorithms that based on available bandwidth and the latency requirement decide the best layer to partition the NN. In [29] the choice of the splitting point was based on the waiting delay, the layer that can make the delay smaller is chosen as output of the first part. In [3] the NN was split between edge and cloud with the objective of reducing bandwidth consumption so the splitting can only be made at layers whose output dimensions are lower than input image dimensions. Authors of [27] split a CNN trained on pedestrian detection into two parts in order to process the first part in an edge-heavy sensor and the second on the server. The data sent was twelve times lower than the input data size. However, they reduced the CNN layers' and filters' numbers too much in order to reduce the size of the first part's output size. In [4] the algorithm defines several splitting points then calculated latency for each one compared to cloud only and edge only using different devices with different costs at the edge. For smaller edge devices, edge-only computation was the cheapest and the fastest solution, while for more powerful devices a middle split between edge and cloud gave better results.

Other methods were also used to find the best splitting point such as data-flow computing in [30] based on a data-flow computing model, taking in consideration the architecture of the NN, network's characteristics and computation resources availability define the splitting point. The solution in [10] was based on genetic algorithms for a multi-objective (memory usage, latency, energy consumption) optimization problem.

2.3 Findings

After reviewing work of both NN splitting and data transmission reduction, our findings are as follow:

- Research work in this area is mostly focused on defining the most convenient position to split the NN taking in consideration the characteristics of the edge device, the network, the input data, and the task to be achieved. The solution is generally an algorithm that checks if the conditions are met for all layers, then return the most appropriate splitting position.
- For deep NN, the output of the layers is large, it only gets reduced near the end of the NN. Solutions proposed for NN splitting either split at the last layers of the NN so the entity to be sent is smaller but then the processing done at the IoT object is heavy. Or use more powerful devices at the edge to compute the first part of the processing. Another option is to reduce the number of filters and calculation in the layers so the output is smaller as in [27], however the functioning is less efficient especially for complex tasks such as real time object detection.
- Data transmission reduction solutions are only proposed for the original data and they are generally based on prediction, regression or classification models which make the task achieved by the IoT object heavier.

Our proposed solution reduces data to be sent in case of split NN in order not to remain exclusive for powerful edge device or light weight models. Also, it is a solution based on data similarities that doesn't overburden the IoT objects with predicting or classification models.

3 THE PROPOSED SOLUTION

3.1 General Description

To solve data transmission related problems such as limited bandwidth availability, and low latency tolerance for most IoT applications, we are proposing a data transmission reduction solution for split NN models where the first part is processed at the object´s level, then the result is transferred to the cloud to be used as an input of the second part. The result of the first part is the output matrix of the last layer in the first part. Considering applications where the object is stationary, the collected data do not vary much between one sending and the next one as shown in figure 1. Since the inputs are similar and the NN is the same, we suppose that the outputs of the first part will also be similar somehow. Our proposed solution only sends the non similarities between the actual output of the first part and the previous one.

Figure 2 and Figure 3 explain the functioning of this solution. The pre-trained NN is split into two parts, the first part used at the object side and second at the cloud server side. For the first data captured by this object, the output of the first part is directly sent

Figure 1: Successive images captured by a traffic camera.

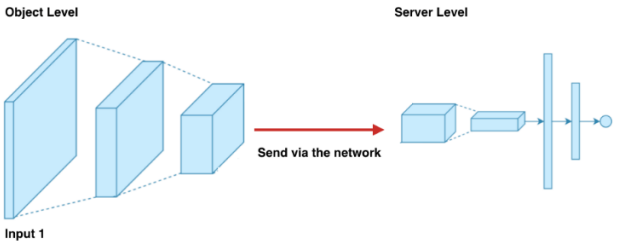

Figure 2: The functioning of the proposed solution for the first data captured.

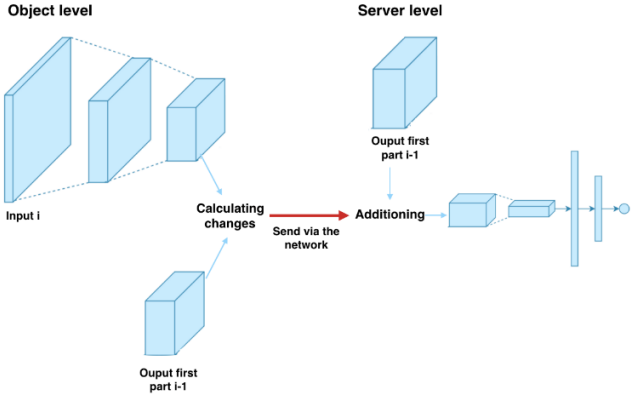

Figure 3: The functioning of the proposed solution for the rest of the data.

to the cloud. At the cloud level, this received entity is used as an input for the second part of the NN. This entity is also stored on both sides for the purpose of using it in the subsequent sendings. For the next data, the object calculates the difference between the output of the actual data and the previous output, eliminates zeros then sends it to the cloud. At the cloud server level, the input of the second part is calculated by adding zeros to the received entity, then additioning it with the previous output (either received entirely in the case of the first data or calculated previously).

3.2 Formalization

A neural network is composed of an input layer, an output layer, and a set of hidden layers in between. One layer's output is a weighted sum of the outputs of previous layers, added to a threshold value called bias, then passed to a nonlinear function called activation function. The output of each layer is defined as [15] [28]:

$$f_i(X) = a(W_i X_i + B_i) \tag{1}$$

Where:

- a is the activation function whose objective is to make the NN learn nonlinear features from the input [14]. It can be the hyperparabolic tangent, sigmoid, softmax, rectifier function or other functions (Fig. 4).
- X_i is the input matrix at layer i.
- W_i is the weight matrix at layer i.
- B_i is the bias matrix at layer i.

Then, the whole network is defined as a composition of functions as [13] [15]:

$$Y = f_k^{\circ} \ldots^{\circ} f_i^{\circ} \ldots^{\circ} f_1(X) \tag{2}$$

$$Y = f_k(\ldots(f_i \ldots(f_1(X)))) \tag{3}$$

$$Y = a(W_k a(\ldots . W_i a(\ldots W_1 X + B_1) \ldots . + B_i) \ldots . + B_k) \tag{4}$$

Where:

- K is the number of layers.
- X is the input data.
- Y is the output of the NN.

Let i be an intermediate layer where $1 < i < k$.

Let (n, l), be the dimensions of the input X_i of layer i and (m, n) the dimensions of the weight matrix W_i, then the dimension of the multiplication $W_i X_i$ is of dimensions (m, l).

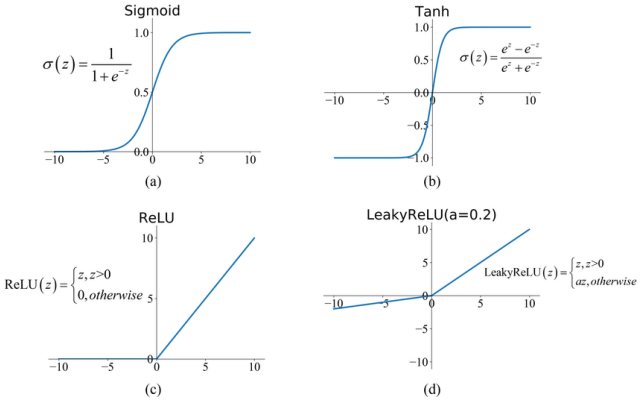

Figure 4: Commonly used activation functions: (a) Sigmoid, (b) Tanh, (c) ReLU, and (d) LreLU [6].

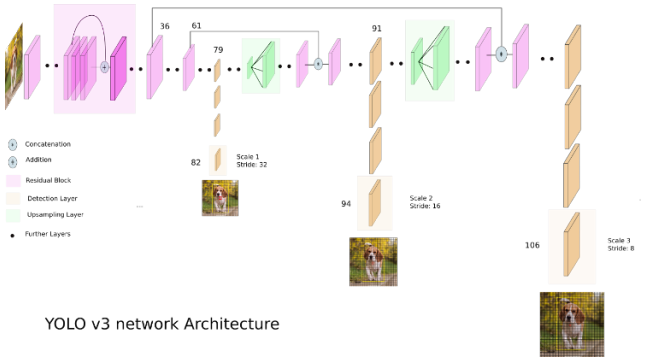

Figure 5: YoloV3 architecture [31].

Thus, the dimensions of the bias matrix B_i is also (m, l) and the layer output is also of dimensions (m, l). $Y_i \in \mathbb{R}^{m\times 1}$

Observing activation functions graphs in figure 4, we notice that for similar inputs the output is similar. And since we have similar NN inputs and same pretrained weights, then the outputs of intermediate layers are similar. Therefore, in this paper, we only send the changes that occur in the new output of the first part.

We define C_i as the difference matrix between two outputs Y_{1i},Y_{2i} $\in \mathbb{R}^{m\times 1}$ of similar inputs X_1,X_2 at layer i.

$$\Delta(Y_{1i}, Y_{2i}) = C; C \in \mathbb{R}^{m\times 1} \tag{5}$$

Instead of sending the matrix C, we only send the values that represent a change: values different from zero. The entity to be sent to the cloud is defined as:

$$D = (c_{r,p}) \in C;\ c_{r,p} \neq 0 \tag{6}$$

$$D \in \mathbb{R}^{n\times t} \text{ where } (n, t) < (m, l) \tag{7}$$

4 EXPERIMENTS

4.1 Experiment's Settings

Use case: Traffic cameras are one of the connected devices that collect a massive amount of voluminous data especially if the camera is of high resolution. The captured images are generally sent to the cloud where it can be used in different types of applications such as accident detection, cars counting, car number detection, low violation detection, etc. Therefore, we chose traffic cameras as a use case to test our proposed solution. Since detecting objects is the first step for most applications on traffic data, we decided to do object detecting where the used AI model is trained to detect objects, frame them and predict their class.

NN model: The first step of our experiment is to split a neural network model into two parts. We used the pre-trained YOLO (You Only Look Once) model, one of the most used NN for object detection [32]. Yolo architecture is composed of convolutional layers followed by connected layers predicting bounding boxes in the image. Several versions of Yolo pre-trained are available for use. We used the version 3 of Yolo (figure 5) contains 106 layers and was trained on COCO dataset (Common Objects in Context) [33]. This dataset contains 80 class among which we have "car", "bus", "truck", "person", "bicycle", "motorbike", "traffic light", "stop sign" that are the objects that can be detect in traffic data.

Data: Data needed to test our solution is traffic data captured successively using the same camera and in order. For this we used Sherbrooke Urban Tracker dataset that provides successive images captured by traffic cameras with a rate of 30 frames per second in Montreal-Canada [2].

4.2 NN splitting

We split the pre-trained YoloV3 into two parts. We chose layer 11 as a splitting point. The objective of our research is not to find the best position to split. Therefore, we chose manually a layer that is not so far from the input of the NN and does not have loops in order not to send two outputs. The output of layer 11 is of size (1, 104, 104, 128). We name the first part sub-model1 and the second sub-model 2.

4.3 Outputs Comparison

We calculated the difference between two successive outputs of sub-model 1. Figure 6 presents two examples of numbers frequency in the difference of two successive outputs. The frequency of numbers can vary depending on the image, however zero is always the most frequent value: 785177 (56,71% of the difference) in first example and 490663 (35,44% of the difference) in the second.

4.4 Transmission Reduction

This experiment consists of eliminating zeros from the difference between the current output and the previous one. We also send the position of data not sent in order to reconstruct the same output at the cloud level. We send to the cloud a list that contains non-zero numbers and a bit matrix that for each position contains 1 if the number is sent and 0 if not. Then, we calculate the difference in size between the original output size and the entity send when reducing zeros.

At the cloud level we have the inverse function of the applied representation method which allows rebuilding the data without causing any data loss.

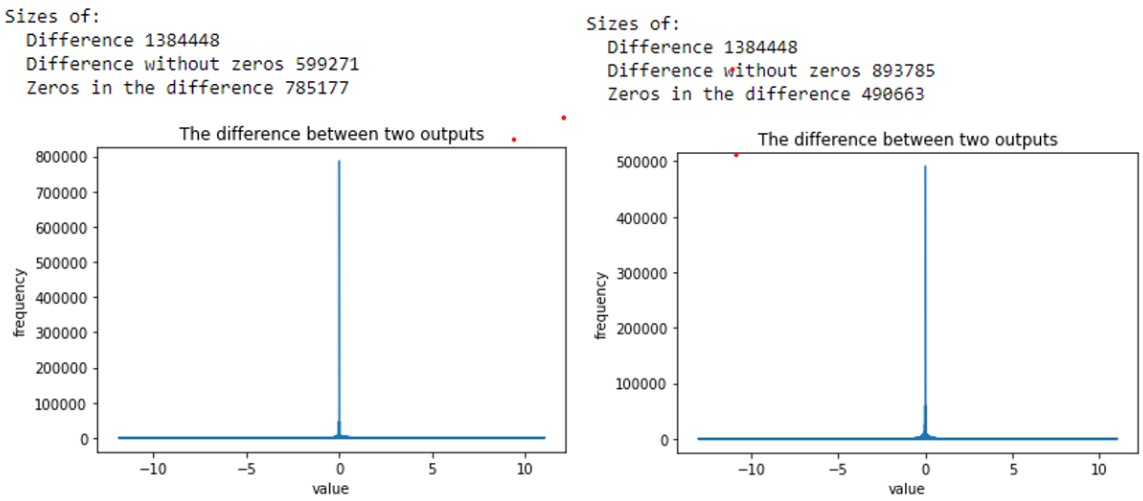

Figure 6: Number's distribution in the difference D.

Table 1: Size of data sent

	2 images (MB)	100 images (MB)
Submodel1's output	21.13	1699.84
Zeros reduced	18.07	1075.2

We calculated the size of data to be sent when we don't send zeros for two images and hundred successive images. Results are presented in Table 1.

Data to be sent was considerably reduced compared to the output of sub-model1. We have 14.48% bytes reduction for two images and 36,74% for hundred images. We notice that reduction is higher in long terms when similarities are accumulated.

4.5 Values Close to Zero

We noticed that values close to zero may not be frequent (because those values are up to nine digits after the decimal point) but they are numerous. Reducing them will reduce furthermore the data transmitted. And since it is a value close to zero in the difference, that means that the output didn't change much. Not sending it will not affect the results much.

This experiment consists of reducing values close to zero. When not sent, these values are replaced by zeros at cloud level. Then, we check if sub-model2 is able to detect and classify data correctly even without receiving those values.

The method calculating if data is correctly detected compares the number of objects detected, the classification as well as the positions of the bounding box (we allowed a small shift of the position of the bounding box estimated by $\frac{image\ length}{10}$).

Figure 7, 8 and 9 presents the results of this experiment where we gradually enlarged the interval of data not sent while verifying that the end result is correct.

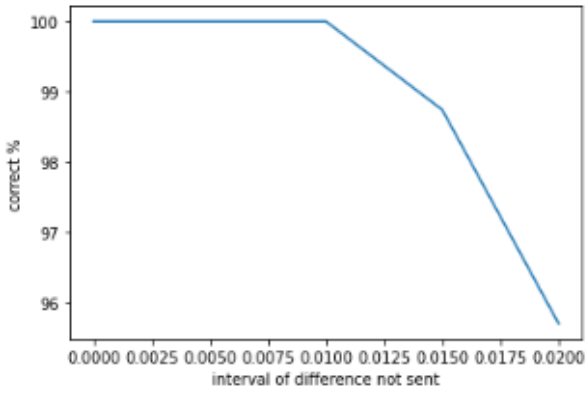

Figure 7: Percentage of correctly detected objects in relation with the interval of data not sent.

The results show that not sending values in interval [-0.01, 0.01] reduces the number of data to be sent to 60% from the size of sub-model1 output (35% more than when reducing only zeros) while maintaining 100% correct detection and classification. In terms of bytes, when not sending data in interval [-0.01, 0.01] we only send 38% (62% not sent) of bytes (including data and positions) compared to when we send all the output.

The reduction percentage may vary from one set of images to another depending on the similarity rate between images. However, our experiments proved that data transmitted in edge-cloud collaborative AI can be considerably reduced using the proposed similarity-based method.

5 CONCLUSION AND FUTURE WORK

In order to benefit from the advantages of collaborative processing (reducing network related issues as well as the load on the IoT object), in this paper we defined a solution to reduce data sending.

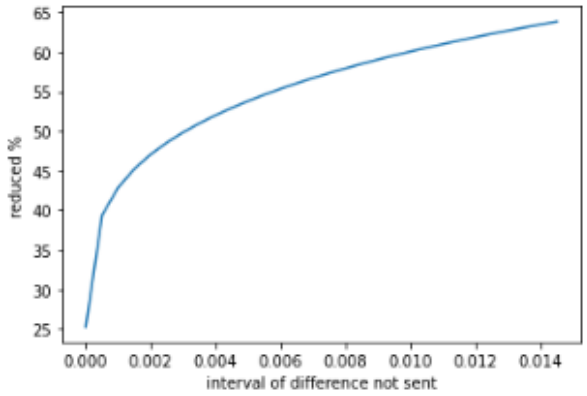

Figure 8: Percentage of numbers not sent in relation with the interval of data not sent.

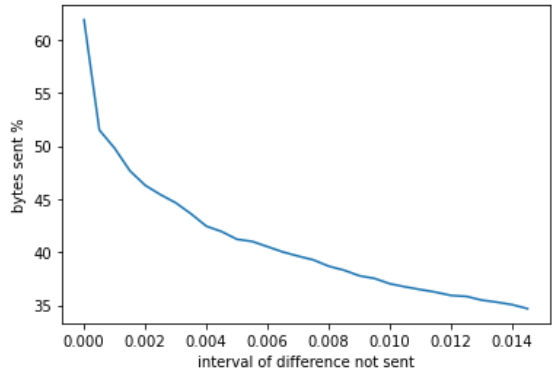

Figure 9: Percentage of bytes sent in relation with the interval of data not sent.

Based on similarities in data captured by stationary objects, the proposed method reduces the values sent from the IoT object to cloud while being able to reconstruct the real data at the cloud level.

Several experiments were performed to prove the efficacy of the proposed solution. We demonstrated that the output of the first part of a split NN model for similar inputs contains a significant percentage of repetitions. Thus, we only send the difference between two outputs when it is different from zero. Sending only non-zero values and their positions reduced 36,74% of bytes sent.

We also experimentally defined the interval ([-0.01, 0.01]) of data that is not worth sending since it can be replaced by zero at cloud level without causing a deterioration in the object detection task. By not sending data in this interval, we reduce bytes sent by 62%.

To our knowledge, this paper is the first to propose a data transmission reduction method for edge-cloud collaborative processing.

For future work we will test this solution on other use cases with different data types in order to provide a better generalization.

REFERENCES

[1] E.N. Sánchez and A.Y. Redes Neuronales: Conceptos Básicos y Aplicaciones. Alanis Automática y Robótica, Pearson Educación (2006)
[2] J. Jodoin, G. Bilodeau and N. Saunier, Urban Tracker: Multiple object tracking in urban mixed traffic, IEEE Winter Conference on Applications of Computer Vision, 2014, Pages 885-892. doi.org/10.1109/WACV.2014.6836010.
[3] R. Mehta, R. Shorey, DeepSplit: Dynamic Splitting of Collaborative Edge-Cloud Convolutional Neural Networks, 2020 International Conference on COMmunication Systems & NETworkS (COMSNETS), 2020, Pages 720-725. doi.org/10.1109/COMSNETS48256.2020.9027432.
[4] P. A. Leroy, T. Goedeme, Optimal distribution of cnn computationson edge and cloud. inICAART (2), 2021, Pages 604–613. doi.org/10.5220/0010203706040613
[5] O.K. Shahryari, H. Pedram, V. Khajehvand and M.D TakhtFooladi, Energy-Efficient and delay-guaranteed computation offloading for fog-based IoT networks, Comput. Networks, 2020, doi.org/10.1016/j.comnet.2020.107511
[6] F.Junxi, H. Xiaohai, T. Qizhi, R. Chao, C. Honggang, L.Yang, Reconstruction of porous media from extremely limited information using conditional generative ad-versarial networks, Physical Review E, 2019. doi.org/10.1103/PhysRevE.100.033308
[7] J. Uthayakumar, M. Elhoseny, K. Shankar, Highly Reliable and Low-Complexity Image Compression Scheme Using Neighborhood Correlation Sequence Algorithm in WSN, IEEE Transactions on Reliability 69, 2020, Pages1398–1423. doi.org/10.1109/TR.2020.2972567
[8] A. Banitalebi-Dehkordi, N. Vedula, J. Pei, F. Xia, L. Wang, and Y. Zhang. Auto-Split: A General Framework of Collaborative Edge-Cloud AI. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (KDD '21), 2021, Pages 2543–2553. doi.org/10.1145/3447548.3467078
[9] H. Nozari, A. Szmelter-Jarosz, J. Ghahremani-Nahr, Analysis of the Challenges of Artificial Intelligence of Things (AIoT) for the Smart Supply Chain (Case Study: FMCG Industries), Sensors, 2022. doi.org/10.3390/s22082931
[10] P. Ishan, B. Aniruddh, V. Rohit, S. Rajeev. SmartSplit: Latency-Energy-Memory Optimisation for CNN Splitting on Smartphone Environment. 2021. doi.org/10.48550/arXiv.2111.01077
[11] M. Ji, J. Yoon, J. Choo, M. Jang and A. Smith, LoRa-based Visual Monitoring Scheme for Agriculture IoT, IEEE Sensors Applications Symposium (SAS), 2019, Pages 1-6. doi.org/10.1109/SAS.2019.8706100
[12] E. Li, L. Zeng, Z. Zhou and X. Chen, Edge AI: On-Demand Accelerating Deep Neural Network Inference via Edge Computing, in IEEE Transactions on Wireless Communications, 2020, Pages 447-457. doi.org/10.1109/TWC.2019.2946140.
[13] Z. Ma, The function representation of artificial neural network, arXiv, 2019. doi.org/10.48550/arXiv.1908.10493
[14] A. D. Rasamoelina, F. Adjailia and P. Sinčák, A Review of Activation Function for Artificial Neural Network, IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI), 2020, Pages 281-286, doi.org/10.1109/SAMI48414.2020.9108717.
[15] A.S. Lapedes, R.M. Farber. How neural nets work. Proceedings of the Neural Information Processing System, 1988, pp. 442-456
[16] G. B. Tayeh, A. Makhoul, J. Demerjian, D. Laiymani, A new autonomous data transmission reduction method for wireless sensors networks, IEEE Middle East and North Africa Communications Conference, 2018. doi.org/10.1109/MENACOMM.2018.8371030.
[17] A. Jarwan, A. Sabbah, M. Ibnkahla, Data Transmission ReductionSchemes in WSNs for Efficient IoT Systems, IEEE Journal on SelectedAreas in Communications, 2019, Pages 1307–1324. doi.org/10.1109/300JSAC.2019.2904357.
[18] T. Shu, J. Chen, V. K. Bhargava, C. W. De Silva, An Energy-Efficient Dual Prediction Scheme Using LMS Filter and LSTM in Wireless Sensor Networks for Envi-ronment Monitoring, IEEE Internet of Things Journal, 2019, Pages 6736-6747 doi.org/10.1109/JIOT.2019.2911295.
[19] Y. Fathy, P. Barnaghi, R. Tafazolli, An adaptive method for data re-duction in the Internet of Things, in: IEEE World Forum on Internetof Things, WF-IoT, 2018. doi.org/10.1109/WF-IoT.2018.8355187.310
[20] M.F. Tsai, Y.C. Chu, M.H. Li, L.W. Chen, Smart Machinery Monitor-ing Sys-tem with Reduced Information Transmission and Fault PredictionMethods Using Industrial Internet of Things, Mathematics 9, 2020. doi.org/10.3390/math9010003.
[21] M. Lewandowski, B. Płaczek, M. Bernas, Classifier-based data transmission reduction in wearable sensor network for human activity monitoring, Sensors (Swit-zerland) 21, 2021 doi.org/10.3390/s21010085.
[22] Y. Inagaki, R. Shinkuma, T. Sato, E. Oki, Prioritization of Mobile IoTData Transmission Based on Data Importance Extracted From Machine Learning Model, IEEE Access 7, 2019, Pages 93611–93620. doi.org/10.1109/ACCESS.2019.2928216.
[23] [A. K. Idrees and A. K. M. Al-Qurabat, Energy-Efficient Data Transmission and Aggregation Protocol in Periodic Sensor Networks Based Fog Computing, J. Netw. Syst. Manag, 2020. doi.org/10.1007/s10922-020-09567-4.
[24] Z. Zhou, Q. J. Wu, F. Huang, X. Sun, Fast and accurate near-duplicateimage elimination for visual sensor networks, International Journalof Distributed Sensor Networks 13, 2017. doi.org/10.1177/1550147717694172.
[25] S. N. Sajedi, M. Maadani, and M. Nesari Moghadam, F-LEACH: a fuzzy-based data aggregation scheme for healthcare IoT systems, J. Supercomput, 2022, , Pages 1030–1047 doi.org/10.1007/s11227-021-03890-6.
[26] K. Jain and A. Kumar. A lightweight data transmission reduction method based on a dual prediction technique for sensor networks, Trans. Emerg. Telecommun. Technol, 2021. doi.org/10.1002/ett.4345

[27] Y. Ikeda *et al.*, Reduction of Communication Cost for Edge-Heavy Sensor using Divided CNN, IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2018, Pages 244-245. doi.org/10.1109/RTCSA.2018.00042.

[28] J. Gao, Y. Guo, and Z. Wang, Matrix neural networks, in Proc. Int. Symp. Neural Netw. Cham, Switzerland: Springer, 2017, Pages 313–320. doi.org/10.1007/978-3-319-59072-1_37

[29] B. Zhang, Y. Li, S. Zhang, Y. Zhang, B. Zhu, An Adaptive Task Migration Scheduling Approach for Edge-Cloud Collaborative Inference, Wireless Communica-tions and Mobile Computing, 2022. doi.org/10.1155/2022/8804530

[30] B. Jani, T. Bo, N. Jari, Edge-PRUNE: Flexible Distributed Deep Learning Infer-ence, ArXiv, 2022. doi.org/10.48550/arxiv.2204.12947

[31] A. Kathuria, What's new in YOLO v3? 2018. https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b . Last ac-cess : June 2022

[32] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, You Only Look Once: Uni-fied, Real-Time Object Detection, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, Pages 779-788, doi.org/10.1109/CVPR.2016.91.

[33] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, Microsoft COCO: Common objects in context. In ECCV, 2014. doi.org/10.1007/978-3-319-10602-1_48