DNN inference splitting and offloading in the Internet of Things : A survey

Chakir Bouarouguene, Moufida Maimour, Ouahab Kadri, Eric Rondeau, Abderrazak Benyahia

DNN Inference Splitting and Offloading in the Internet of Things : A Survey *

Chakir Bouarouguene[1], Moufida Maimour[2], Ouahab Kadri[1],
Eric Rondeau[2], Abderrazak Benyahia[1]

[1] LASTIC laboratory, University of Batna 2, Batna, Algeria
c.bouarouguene@univ-batna2.dz o.kadri@univ-batna2.dz a.benyahia@univ-batna2.dz
[2] CRAN, Université de Lorraine, CNRS, UMR 7039, BP 70239, F-54000 Nancy,France
moufida.maimour@univ-lorraine.fr eric.rondeau@univ-lorraine.fr

Keywords : *Internet of Things, Deep Neural Network, Industry 4.0.*

1 Introduction

Nowadays, the Internet of Things (IoT) crosses Deep Neural Network (DNN) inference in various intelligent applications such as those related to Industry 4.0. Constructing DNNs to run in real-time on resource-constrained devices is one challenging problem that can be addressed using two main strategies. The first is offloading DNN layers from resource-constrained devices to a powerful server to alleviate inference time [8]. The second is splitting DNNs, which aims to select a set of layers to run locally, yielding the lowest possible inference time [3]. The complete survey answers 5 questions and considers 46 papers. In this version, we limit the content to the following questions :

- **Q1 :** Are small DNNs optimal enough to run in The Edge without DNN offloading being required ?
- **Q2 :** What are the used algorithms for DNN splitting and Offloading ?
- **Q3 :** What are DNN splitting and offloading open issues ?

2 Trade-off Between DNN Inference Speed and Precision

To answer the first question (Q1), we viewed the performances of state-of-the-art Neural Networks in the field of object detection. We also selected *MS COCO* data set to train DNNs running over Raspberry devices because this data set contains many objects linked to industry 4.0, especially electronic devices. We found an accurate DNN model *Faster RCNN Inception v2* proposed in [8]. The mean Average Precision (mAP) has reached up to 97%, however, it is rather slow with only 0.08 frames per second (FPS). Another work [5] proposes a DNN known as *MobileNet-Tiny*, which is relatively faster (4.5 FPS) but less accurate (19.4 % mAP). Therefore, it can be noted that the more accurate the models, the slower they are and vice versa. Therefore, offloading DNNs to a powerful server is very beneficial.

*This work was supported in part by the PHC TASSILI 21MDU323

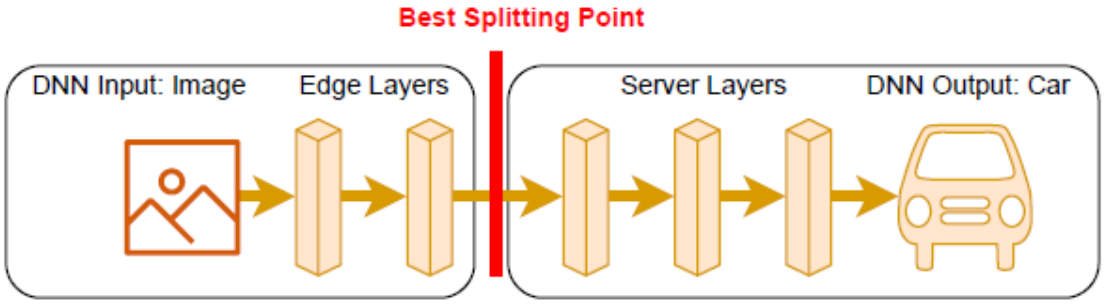

FIG. 1 – The first two DNN layers run on the Edge, and the last three layers run on the server.

3 DNN Splitting and Offloading Algorithms

This section aims to answer the second question (Q2). After a thorough study of different DNN distribution algorithms in IoT environment, we classified them as follows :

3.1 Algorithms that determine DNN splitting points

In these algorithms, DNNs are split in two parts : one that runs locally and the other is offloaded to the Cloud server as shown in Figure 1. Among these binary distribution algorithms, there is [1]. The authors represented DNNs in form of a direct acyclic graph, and used the Minimum-cut technique to find a fast splitting point. They were able to enhance latency up by 8.08 times. Another related paper is [3], in which the authors proposed the allocation of DNN over Edge devices and Cloud in form of a latency-minimum allocation problem. An algorithm was designed to solve the problem in a polynomial time based on testes over collected data about the latency between nodes and the computation time of DNN segments on each device. Concerning interpretability, the chosen splitting point could affect the outputs of DNN layers because edge and server DNN parts are generally linked with an auto-encoder. The latter should be implemented right after the layer which contains the maximum density of neurons with a high gradient respecting the right class decision to keep the information flowing up to that point [2].

3.2 Algorithms that compress Edge layers.

After dividing the DNN into different parts, there are several works that attempt to compress layers running on resource-constricted devices, like using early-exit [6]. After conducting numerous experiments with different Edge devices and network bandwidths, they were able to improve processing time by 70 % demonstrating that the more powerful the Edge device is, the higher the chances are to accelerate the inference time.

Another important work [4] compresses the Edge layers using knowledge distillation, as well as compression of intermediate-data sent to the server using Auto-Encoder demonstrating that the optimal cutting layer varies depending on memory and bandwidth constraints. The intermediate data compressed by the Auto-Encoder are supported by rather small bandwidths. Also, Edge model's memory size could be divided by 2 by implementing knowledge distillation for VGG16 and by 7 for MobileNetV1 models.

3.3 Algorithms concerned with loss tolerance.

It's commonplace that the nature of IoT makes it vulnerable to hardware failure and network packets loss. In most algorithms, if there had been any kind of loss, we have to resend or reoffload the weights from clients to servers, which make such algorithms rather slow in terms of inference time. However, in the case of DNN distribution, there are other methods that permit

facing losses without the need for data re-transmission, such as the Dropout and Residual Connections. The Dropout technique is defined as temporarily disabling some neurons in the network, as well as all its incoming and outgoing connections. This paper [7] used Dropout technique in neural networks so as to simulate packets loss in the physical network by disabling certain neurons and train the DNN to maintain accuracy despite some neurons being disabled.

4 Open Issues

To provide an answer to the third question (Q3), this section summarizes the future works mentioned in the previously discussed algorithms, as well as the challenges encountered. The open issues can be summarized in the following points :

— The need for dynamic solutions in which the distribution varies depending on the available processing and transmission resources [4][3].
— The lack of model parallelism to accelerate inference time for the algorithms that represent DNNs in the form of direct acyclic graphs [1].
— Balancing the accuracy and latency for compression algorithms [6].
— The lack of algorithms for DNN distribution over large- scale IoT environment and mobile devices [4].

5 Conclusion

In this survey, we proposed an answer to various questions in the field of DNN splitting and offloading. We demonstrated that the accurate model barely runs in non-GPU Edge devices. Then, we classified and discussed the algorithms used to distribute DNN. We concluded this work by mentioning open issues and solutions to reinforce splitting algorithms. The complete version will consider other questions and hence will include more articles with additional information.

References

[1] C.Hu et al. Dynamic adaptive dnn surgery for inference acceleration on the edge. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 1423–1431. IEEE, 2019.

[2] F.Cunico et al. I-split : Deep network interpretability for split computing. In *2022 26th International Conference on Pattern Recognition (ICPR)*, pages 2575–2581. IEEE, 2022.

[3] L.Hu et al. Coedge : Exploiting the edge-cloud collaboration for faster deep learning. *IEEE Access*, 8 :100533–100541, 2020.

[4] M.Sbai et al. Cut, distil and encode (cde) : Split cloud-edge deep inference. In *2021 18th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE, 2021.

[5] NS.Sanjay et al. Mobilenet-tiny : A deep neural network-based real-time object detection for rasberry pi. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 647–652. IEEE, 2019.

[6] RG.Pacheco et al. Inference time optimization using branchynet partitioning. In *2020 IEEE Symposium on Computers and Communications (ISCC)*, pages 1–6. IEEE, 2020.

[7] S.Itahara et al. Packet-loss-tolerant split inference for delay-sensitive deep learning in lossy wireless networks. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2021.

[8] XK.Dang et al. Applying convolutional neural networks for limited-memory application. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 19(1) :244–251, 2020.