Splitted Neural Networks for Equipment Inspection: Experiments and Resources

Luan Gonçalves[1], Tiago Guerreiro[1], Samya Pinheiro[1], Flávio de Brito[1], Ingrid Nascimento[1], Neiva Linder[2] and Aldebaro Klautau[1]

***Abstract*— This paper presents a feasibility study of performing equipment inspection on networked environments by exploring networks specially designed for semantic segmentation task. It is assumed that the neural network needs to be split and the resulting two pieces need to be allocated in two devices. The provided results suggest that the current state of the art on semantic segmentation is not well-suited for the splitted networks applications even in the context of the 5G networks.**

***Keywords*— splitted neural network, networked environment, equipment inspection, semantic segmentation**

I. Introduction

A typical process inside the industries, to ensure the quality of the production, is the inspection of products, parts, equipment and components. When performed manually, it can be a cumbersome and an expensive process by several reasons, like 1) it requires the creation of specialized labour to perform such inspections; 2) these inspections can cause bottlenecks in the production/time to market timeline; and 3) as manual inspections do not scale as products do, further training is required to have enough specialists for performing timely inspections. Many efforts have been directed for researches like Computer Vision (CV), Internet of Things (IoT), Edge Computing (EC) and Deep Neural Networks (DNNs) in order to automate the inspection process.

In this context, devices with low memory usage, computation cost, power consumption and optimized bandwidth usage are indispensable. For this propose, the adaptation of the applications that run through a specific telecommunication medium plays an important role. The nature of these adaptations are mainly represented by DNNs compression and edge-computing, like network pruning [1], [2], [3] and partitioning of Neural Networkss (NNs) [4], [5] between the User Equipment (UE) and edge or cloud.

Compressing a DNN can reduce the number of parameters, computational cost and memory access to the point that a UE could run it without much accuracy loss while optimizing energy consumption. The method introduced by Han *et al.* [1] achieves a high compression rate by a three-stage pipeline: pruning, quantization and Huffman coding, being able to run large networks on sensing devices. Despite working very well on the majority DNNs, more complex networks can perform poorly depending on the compression pipeline, so it should be used wisely. Another compression algorithm proposes a lossy compression scheme to achieve a parameter and computational reduction while maintaining a good accuracy [3]. Although the above methods achieve good results, for some applications like drone live target detection [6], running the DNN in the UE is not practical and can lead to performance loss. Live drone detection mostly requires image capture, processing and transmission in real-time, which is too much to handle in one device, considering the power, bandwidth and computational costs. In that case, a distributed approach would be well suited.

Surat *et al.* [7] provided an example of distributed computing that uses a smaller model on the UE for feature extraction and a larger model on the cloud or edge for post-processing and classification. Jong *et al.* [4] proposed a split at an intermediate layer of the network, focusing on the transmission accuracy to the end-user machine with minimum energy consumption. Also, encodes the feature map before transmission, to reduce bandwidth usage. Although both seem very similar, distributed computing works by sharing resources between UEs and cloud or edge (multiple DNNs). In contrast, network partitioning splits a DNN in a specific layer and segments the task between the EU and cloud or edge. These approaches decrease memory access and processing on the UE devices by implementing collaborative inference between different entities and increase battery time, allowing better performance in automated tasks.

Industries of several kinds can be benefited by these strategies. As an example, the steel industry struggles with difficulties related to the manual inspection of the production, ranging from the time that this kind of process requires, to dangers related to the trade. As a result of this concern, on the part of the steel industries, Severstal proposed a challenge on Kaggle for steel defect detection [8] in late 2019.

As image segmentation is particularly interesting to perform equipment inspection, it is essential to explore the structure of well-established DNNs for semantic segmentation assignment, like Fully Convolutional Networks (FCN) [9], U-Net [10] and PSPNet [11], in split neural networks scenarios, in order to make possible the automatic object inspection with good resource usage (memory usage, computational cost and bandwidth usage).

This paper presents a feasibility study of using splitted DNNs between UE and cloud for semantic segmentation of defects in steel sheets. In this study, the Severstal dataset [8] was used as a benchmark and some networks specially designed for semantic segmentation (FCN [9], U-Net [10] and PSPNet [11]) were analyzed, exploring its resource usage especially in the

[1]LASSE - 5G & IoT research Group, Federal University of Pará (UFPA), Belém-PA, Brazil, [2]Ericsson Research, Kista, Sweden, E-mails: {luan.goncalves, tiago.guerreiro, samya.pinheiro, flavio.brito, ingrid.nascimento}@itec.ufpa.br, aldebaro@ufpa.br, neiva.linder@ericsson.com.

UE and in the transmission medium.

The major contributions of this paper are 1) to summarize some of the difficulties related to performing complex tasks on splitted neural networks scenarios and 2) to show a feasibility test of using splitted neural networks for automatic detection of defects on steel sheets, using some well-established DNNs.

This paper is organized as follows: the adopted evaluation methodology is described in Sec. II. In Sec. III we present the results and discuss them and, finally, Sec. IV concludes the paper.

II. Adopted evaluation methodology

The methodology used in this paper explores well-consolidated neural networks for image semantic segmentation task, which consists of classifying each pixel of an image into an instance, considering each of it as an object in the scene. In this study, we consider FCN [9], U-Net [10] and PSPNet [11], which are DNNs widely used in the detection of minute details. The FCN structure is an adaptation of DNNs designed and trained for image classification (like AlexNet [12], VGG nets [13] and GoogLeNet [14]). The U-Net has an encoder-decoder structure. The encoder is responsible for collecting precise information of the image, by contracting the feature maps, while the decoder expands it, recovering higher characteristics, i.e. spatial information [15]. Finally, the PSPNet [11] is composed of a ResNet [16] model with dilated network strategy [17], [18] followed by a Pyramid Pooling Module (PPM) that provides the fusion of local and global context information.

A robust dataset was obtained in the Kaggle platform, available at Severstal: Steel Defect Detection competition [8]. Severstal is a Russian company operating mainly in the steel and mining sector. It provided a dataset in December 2019 composed of images of steel sheets and its corresponding defect masks, containing 12568 samples on the training set and 5506 samples on the testing set. The objective of this competition is to detect defects while classifying them, considering four labels of defect masks. Fig. 1 depicts an example of the mentioned dataset of some steel sheets with different defect masks. However, it is not conspicuous to perceive what is the real defect (e.g. bubbles), which makes this a complicated task.

The adopted method was developed aiming a feasibility study of using some well-referenced DNNs designed for semantic segmentation in a networked environment via splitting them in different layers, analyzing computation cost, memory usage and operating speed in Frames Per Second (FPS) for the UE. In this scenario, a Visual Processing Unit (VPU) from Intel Corporation, named Movidius Myriad X [19] (USB stick) acts as the UE. Movidius supports frameworks like Tensorflow and PyTorch besides operating in systems like Windows and Linux. The end-user machine is an ordinary computer, with the following specifications: i7 7th Gen, Nvidia RTX 2070 video card (8GB) and 16GB of RAM. In the aforementioned scenario, an optimal recovery system of the information sent by the UE is assumed on the cloud.

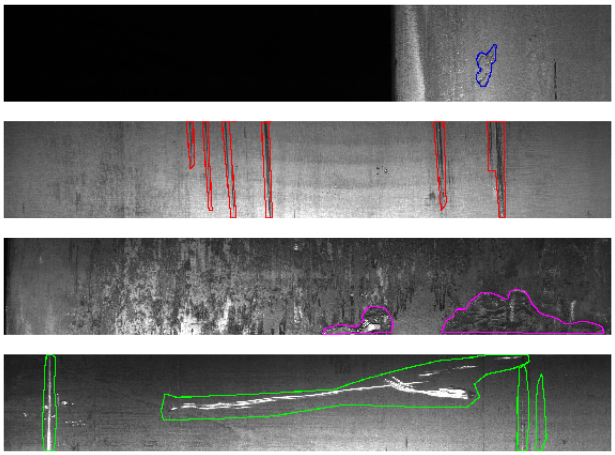

Fig. 1. Four samples from Severstal dataset [8]. The areas surrounded by different colors represent different types of defects.

A. Data measure

This subsection exposes how the data presented in this paper was obtained. The FPS data was acquired by executing inference on the USB stick, using the benchmark software from OpenVINO [20]. The inference time was 30 minutes for all models in which the achieved frame rate was collected. Finally, all the models were converted to Intermediate Representation (IR) format using OpenVINO model optimizer with 16-bit precision.

For memory usage (MB) and computation (GFLOPS), another OpenVINO software, the model analyzer, dumps data by analyzing the model .xml (describes the network topology) and .bin (contains the weights and biases binary data) files.

B. Training process

As the Severstal's dataset [8] is compounded by a training set and a test set, 20% of the training set were used in the validation phase.

Both networks used in this paper were trained in an end-to-end manner for 100 epochs with batch size equals to 4 samples. To prevent overfitting, the model which achieved the best evaluation metric was used in the performed experiments. The initial learning rate for both models is 1×10^{-3} and the default Adam algorithm [21] is used for stochastic optimization.

During the training process, the function that should be minimized is

$$Loss(X, \hat{X}) = BCE(X, \hat{X}) + Dice_{loss}(X, \hat{X}) \quad (1)$$

where $BCE(X, \hat{X})$ is the binary cross-entropy, $Dice_{loss}(X, \hat{X})$ is the dice loss, given by

$$Dice_{loss}(X, \hat{X}) = 1 - Dice_{coeff}(X, \hat{X}) \quad (2)$$

and $Dice_{coeff}(X, \hat{X})$ is the dice coefficient between the ground truth and its predicted set of pixels, respectively.

In order to evaluate the behaviour of the DNNs for this task, we used the evaluation metric required by the Severstal: Steel

Defect Detection competition [8], which is the dice coefficient (Eq. 3).

$$Dice_{coeff}(X, \hat{X}) = \frac{2 \times |X \cap \hat{X}|}{|X| + |\hat{X}|} \quad (3)$$

III. Results and Discussions

A. Neural network analysis

For all DNNs five possible splitting points were considered: i) all the five max-pooling layers of the U-Net encoder structure; ii) all the five maximum pooling layers of the FCN-X (i.e. FCN-32s, FCN-16s, FCN-8s) backbone (VGG-16 [13]); iii) the first maximum pooling and the last convolutional layer of all residual block of the PSPNet backbone (dilated ResNet-50).

The results in Fig. 2 presents the bitrate and the number of layers allocated in the UE for a given DNN and splitting point. According to the picture on top, the achieved bitrate can be controlled by choosing a specific splitting point inside the DNN structures, and the FCN-X provides more variations in the bitrate when switching the splitting point. On the other hand, from the picture on the bottom, the PSPNet allocates more layers in the UE when its structure is split from the second splitting point (which means high computational cost). As the U-Net has almost the same number of layers in the UE per splitting point and provides low values of bitrate, this network is considered in the following results of this paper.

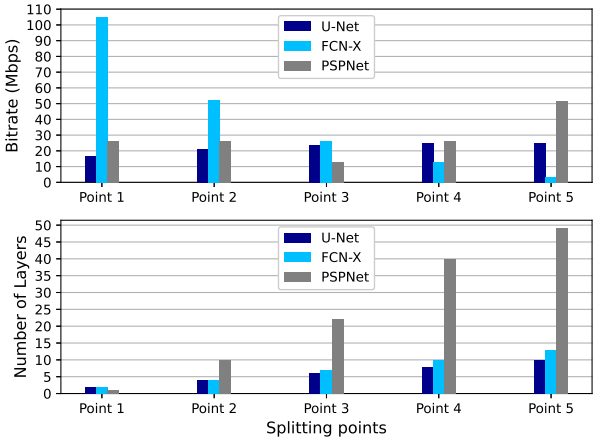

Fig. 2. Comparison of resource usage of the UE for different DNNs and different splitting points. The picture on top presents the bitrate required to transmit the layers' output allocated in the UE, through communications channel, to the cloud and the corresponding number of layers allocated in the UE is presented in the picture on the bottom.

Concerning the quality of the response generated by the DNN, as can be seen in Fig. 3, the U-Net structure presented a good behaviour of the loss function (Eq. 1) on both stages (training and validation).

In practice, the learning strategy presented on Sec. II-B also provided satisfactory results. As can be seen in Fig. 4 there is not a large difference between the dice coefficients of the training and validation stages, indicating a good power of generalization.

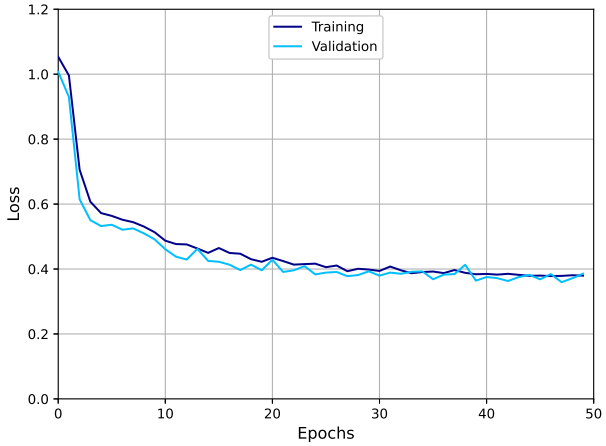

Fig. 3. Similar results for loss function achieved by U-Net in training and validation stages.

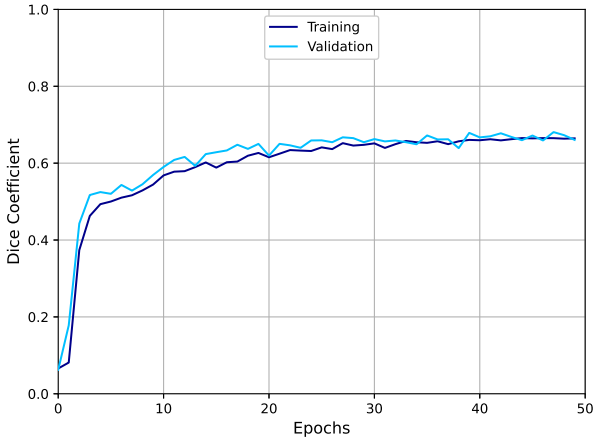

Fig. 4. Similar results for dice coefficient achieved by U-Net in training and validation stages.

B. Networked scenario analysis

This section presents the most critical analysis of this paper. It explores the U-Net structure in order to assess the feasibility of using splitted neural networks for equipment inspection tasks in networked environments.

The first considered aspect is the tradeoff analysis of the partitioning approach based on the inference/transmission demand of the networks when they are split at different layers (Fig. 5). As can be seen on Fig. 5(a), in both networks the computational cost is strongly affected by the firsts and lasts layers, due to the size of their inputs, but when the shortcuts are used to propagate the outputs of the max-pooling layers, there is a reduction of the computational cost. Fig. 5(b) presents the size of the outputs of the UE for each possible splitting point, which clearly shows the enormous contribution of firsts and lasts layers on the memory consumption.

Fig. 5(b) also presents the processing speed of the UE, in frames per second (FPS), for every possible splitting point. At the input point, the whole network is allocated in the cloud, so there is no frame rate related to DNNs models to be registered.

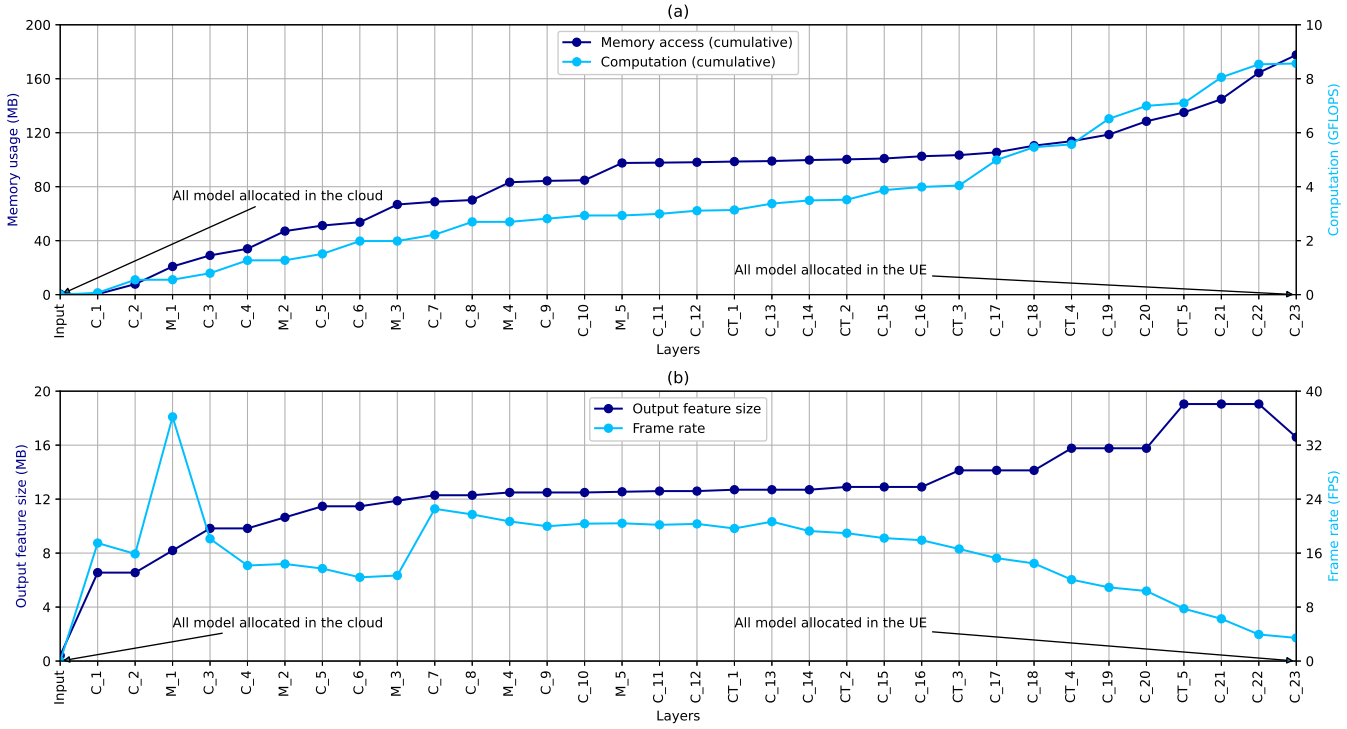

Fig. 5. Behaviour of the sensing device for every possible splitting point of the U-Net. Picture (a) presents the cumulative computation and memory access demand of inference and the picture (b) presents the output feature size and frame rate for each split of U-Net.

In general, the frame rate decreases as the network is split deeper. Although the maximum pooling layers performs a kind of downsampling in the information, the volume to be sent by the UE in these points does not decrease because of the skipped connections along with the U-Net structure. So it is clear that sending information from any intermediate state of the U-Net structure is more costly than sending its input.

The same problem should probably be observed in the others networks (FCN-X and PSPNet), as suggested by Fig. 2, because networks specially designed for semantic segmentation suffer from losing accuracy when downsampling strategies (e.g. pooling, stride) are performed along its structure [17], [18]. For this reason, these networks were built on strategies specially designed to avoid losses of global and local information (e.g. skipped connections [16] and dilated convolutions [17], [18]).

TABLE I

BIT RATE ON SPECIFIED FRAME RATES (MBPS) FOR UE WHEN U-NET IS SPLITTED ON MAX POOLING LAYERS WITH 16-BIT PRECISION.

Splitting point	12 FPS	10 FPS
MaxPool_1	786.43	655.36
MaxPool_2	1022.36	851.97
MaxPool_3	1140.33	950.27
MaxPool_4	1199.31	999.42
MaxPool_5	1204.22	1003.52

Tab. I summarizes the bitrate achieved by the UE when the U-Net is split in one of its five maximum pooling layers given a specific FPS with 16-bit precision and no compression scheme. Clearly, the bitrate increases as the U-Net is split deeper and all values provided by the two option of FPS in which only some of them are achievable only by the requirements for the indoor hotspot scenario of the specification TS 22.261 [22] (1Gbps). On the other hand, if a more aggressive quantization scheme (4-bit precision) is performed, as can be seen in Tab. II, an impressive bitrate reduction is achieved.

TABLE II

BIT RATE ON SPECIFIED FRAME RATES (MBPS) FOR UE WHEN U-NET IS SPLITTED ON MAX POOLING LAYERS WITH 4-BIT PRECISION.

Splitting point	12 FPS	10 FPS
MaxPool_1	196.61	163.84
MaxPool_2	255.59	212.99
MaxPool_3	285.08	237.57
MaxPool_4	299.83	249.86
MaxPool_5	301.06	250.88

IV. CONCLUSION

This paper brings an overview of the problems related to the usage of neural network structures well designed for semantic segmentation (e.g. U-Net, FCN and PSPNet) for the equipment inspection task in networked environments. As the UE commonly has scarce resources, the network split needs to be guided by three aspects: computational and transmission costs, operation speed and the energy efficiency of the edge device. As the networks specially designed for semantic segmentation suffer from losing accuracy when downsampling strategies (e.g. pooling, stride) are performed along with their structures, it is not easy to achieve a good tradeoff between computational and transmission costs.

The presented results confirm that the strategies specially designed to avoid losses of global and local information (e.g. skipped connections and dilated convolutions) can provide a side effect when the network needs to be split in networked environments, by increasing the memory usage and, consequently, resulting in a bad usage of the bandwidth, which represents obstacles for performing real-time inspection (semantic segmentation) in networked environments, through splitted networks environments.

REFERENCES

[1] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, pp. 1–14, 2016.

[2] C. Li, X. Ma, Z. An, and Y. Xu, "A deep neural network compression algorithm based on knowledge transfer for edge device," *Proceedings - 2018 3rd ACM/IEEE Symposium on Edge Computing, SEC 2018*, pp. 334–335, 2018.

[3] S. Jin, S. Di, X. Liang, J. Tian, D. Tao, and F. Cappello, "DeepSZ: A novel framework to compress deep neural networks by using error-bounded lossy compression," *HPDC 2019- Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, pp. 159–170, 2019.

[4] J. H. Ko, T. Na, M. F. Amir, and S. Mukhopadhyay, "Edge-host partitioning of deep neural networks with feature space encoding for resource-constrained internet-of-things platforms," in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2018, pp. 1–6.

[5] W. Shi, Y. Hou, S. Zhou, Z. Niu, Y. Zhang, and L. Geng, "Improving device-edge cooperative inference of deep learning via 2-step pruning," *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 1–6, 2019.

[6] A. C. Anar, E. Bostanci, and M. S. Guzel, "Live target detection with deep learning neural network and unmanned aerial vehicle on android mobile device," *International Conference on Advanced Technologies, Computer Engineering and Science (ICATCES'18)*, pp. 731–736, 2018.

[7] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 328–339.

[8] "Severstal: Steel Defect Detection | Kaggle." [Online]. Available: https://www.kaggle.com/c/severstal-steel-defect-detection/overview

[9] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[10] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[11] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[15] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[17] C. Liang-Chieh, G. Papandreou, I. Kokkinos, k. murphy, and A. Yuille, "Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs," in *International Conference on Learning Representations*, San Diego, United States, May 2015. [Online]. Available: https://hal.inria.fr/hal-01263610

[18] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *International Conference on Learning Representations (ICLR)*, 2016.

[19] "Intel® Movidius™ Neural Compute Stick | Intel® Software." [Online]. Available: https://software.intel.com/en-us/articles/intel-movidius-neural-compute-stick

[20] "Intel® Distribution of OpenVINO™ Toolkit for Linux," 2020. [Online]. Available: https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit.html

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2014.

[22] *Service requirements for the 5G system*, 3GPP, 7 2020, release 15.