Deep Learning on FPGAs with Multiple Service Levels for Edge Computing

Cong Gao, Sangeet Saha, Yufan Lu, Rappy Saha, Klaus D McDonald-Maier, and Xiaojun Zhai
University of Essex, Colchester, United Kingdom
{cg21670, sangeet.saha, yl19888, kdm, xzhai}@essex.ac.uk
{rappysaha10@gmail.com}

***Abstract*—In the Internet of Things (IoT) era, deep learning is emerging as a promising approach for extracting information from IoT devices. Deep learning is also employed in the edge computing environment based on the demand for faster processing. In the edge server, various hardware accelerators have been proposed in recent studies to speed up the execution of such DNNs. One such accelerator is Xilinx's Deep Learning Processor Unit (DPU), designed for FPGA-based systems. However, the limited resource capacity of FPGAs in these edge servers imposes an enormous challenge for such implementation. Recent research has shown a clear trade-off between the "resources consumed" vs. the "performance achieved". Taking a cue from these findings, we address the problem of efficient implementation of deep learning into the edge computing environment in this paper. The edge server employs FPGAs for executing the deep learning model. Each deep learning network is equipped with multiple distinct implementations represented by different service levels based on resource usage (where a higher service level implies higher performance with high resource consumption). To this end, we propose an Integer Linear Programming based optimal solution strategy for selecting a service level to maximize the overall performance subject to a given resource bound. Proof-of-concept case study with a deep learning network of multiple service levels of DPUs on a physical FPGA has also been provided.**

I. Introduction

Deep learning has recently emerged as a canonical methodology in a variety of domains, including computer vision, bioinformatics, natural language processing, and robotics, to mention a few [1]. The reason behind its success can be attributed to its ability to learn from huge volume of data. Another field which is known for generating huge amount of data is Internet of Things (IoT). Recently, Tiny machine learning (TinyML) is also emerging as a new Internet of Things (IoT) prospect that calls for putting the ML algorithm within the IoT device, thanks to rapid advancements in the shrinking of low-power embedded devices and improvements in the optimization of machine learning (ML) algorithms [2].

Many recent research works have considered using deep learning networks to process the IoT data [3]. As a promising result, deep learning has successfully predicted home electricity consumption based on the data from smart meters [4]. Deep learning has also been used to provide location aware services in indoor environments and marketing in retailers [5].

This work is supported by the UK Engineering and Physical Sciences Research Council through grants EP/V034111/1, EP/V000462/1 and EP/X015955/1.

Modern IoT systems demand fast processing, where data needs to be processed in smaller scale platforms rather than execution of complex process of data transfer to a cloud server for analysis. Hence, it will be worth mentioning that the use of centralized cloud for computation of deep neural network will generate bottleneck due to transfer of large set of data with limited network performance [6]. In such scenario, edge computing [7]–[9], a prolific technology for IoT services, emerges as a promising solution. Edge computation offloads the computing tasks from centralized cloud to the edge server located near the IoT devices. Moreover, edge computing is well suited for the applications like deep learning as its intermediate data size is smaller than the input data size. Self driving car is another emerging example of employing deep learning on edge environment [10].

Hardware makers are exploiting existing hardware, such as CPUs and GPUs, as well as developing bespoke application-specific integrated circuits (ASICs) for deep learning, such as Google's tensor processing unit (TPU) to speed up deep learning inference [11]. Deep Neural Network accelerators based on field-programmable gate arrays (FPGA) are another promising method, as FPGA can enable fast computing while keeping reconfigurability [11]. The deep learning processing unit (DPU) is developed as a general accelerator on an FPGA to handle multiple CNN layers, such as convolution, pooling, and activation, and to meet various CNN architectures [12].

Recent research works [13], [14] have assumed that the edge server has sufficient hardware resources (FPGAs or cpus) in terms of computation capacity (memory size) to successfully extract intermediary features using deep learning layers. However, this assumptions will not be true in many cases. In case of resource constrained edge computing environment [15], it has also been observed that there exists scenarios where completion of tasks is more critical than achieving the higher performance [3], [15]. Hence, in order to carry out successful execution of deep learning in a resource constrained FPGA-based edge computing framework, we consider each deep learning network, to be equipped with multiple distinct implementations represented by *"service levels"*. Each implementation of the learning network produces the same result of prediction or classification, but with different levels of performance (in terms of GOPs). Higher service level will return higher performance however, it can typically be achieved at the expense of more resource (i.e. DPU size

cosuming more FPGA resources like LUTs, BRAMS, etc.).

The idea of of having different service levels for deep learning network is supported by the research findings reported in [16]. In this research, the authors have found out that out of all, the memory requirement weight parameters contribute most to the memory footprint. The research further proves that a reduced precision in representing 20% weight parameters results in 1% performance loss. Taking cue from these findings, we propose a strategy for deploying deep learning for IoT into the edge computing environment. Depending upon availability of resource budget each deep learning network on an edge server is executed at a particular service level.

In this paper, we develop the algorithmic support for efficient implementation of deep learning into the FPGA-based edge computing environment, where multiple versions of DPUs are used for executing the neural networks. Specifically, we answer the following question: *Given a upper-bound on available resources in an edge computing environment (maximum area/resource capacity of the FPGA), how do we ensure that the DPUs will efficiently execute deep learning network in a specific service level, while maximizing the overall performance (GOPs) of the process.*

The contributions of this work are summarized as follows:

- Introduction of framework for deploying deep learning in FPGA-based edge computing framework of IoT systems with multiple service levels.
- Development of Integer Linear Programming (ILP) based technique to obtain optimal selection of service level for deep learning network.
- Evaluation of the proposed ILP based strategy with simulation experiments.
- Exhibition of the proof-of-concept by a case study which implements image classification applications on DPUs with multiple service levels with various area requirements.

II. System Model and Problem Description

In the proposed system model, Deep Learning Processor Unit (DPU) from AMD-XILINX FPGA have been employed to speed up the execution of Convolutional Neural Networks (CNNs). In [17], [18], the authors have showed that although each CNN can only consume one DPU, many DPUs can be constructed together on an FPGA to enable concurrent CNN operation. Each DPU has its own requirements of computing resources and BRAMs.

Without loss of generality, we assumed an FPGA-based edge computing environment where the edge server is equipped with FPGAs, where each FPGA may contain multiple DPUs. In the given edge computing environment, let us assume that D denotes the set of N DPUs available in an FPGA: $D = \{D_1, D_2, ..., D_N\}$.

It has been assumed that based on the degree of FPGA resources allocated, each DPU will be equipped to execute the deep learning network for testing in different service levels based on the available resources. An DPU, at any instant, will execute the deep learning model in any one service level among the possible q service levels i.e., $l_i = \{l_i^1, l_i^2, \ldots, l_i^q\}$. Hence, j^{th} service level of D_i can be denoted as l_i^j. The service of a level is proportional to its level-ID. Thus, 1 is the lowest and q denotes the highest execution level.

It can be concluded that higher be the service levels, higher will be its resource consumption. This resource consumption could be in terms of energy consumption, memory consumption of the FPGA. On the other hand, execution of the network in high service level will result in more enhancement of the performance level of the testing process. Service levels with varying degrees of *performance Vs. resource trade-offs* can be obtained by controlling the degree of resources incorporated in the DPU [18]. This concept has also been validated in Section V. This work assumes that, higher be the service level of D_i^j, higher is its resource consumption Res_i^j ($l_i^j > l_i^{j'} \implies Res_i^j > Res_i^{j'}$). Res_i^j denotes the resource consumed by D_i while it executes the deep learning network in j^{th} service level. Similarly, we have also assumed that performance per_i^j will be assigned to D_i^j if the i^{th} edge-server (E_i) successfully executes the deep learning network in j^{th} service level by fulfilling the resource demand.

We have assumed an resource constrained edge computing environment as stated in [15]. Thus, we are imposing the following constraints i.e. i. The overall resource budget $\vec{R}_{total}$ is fixed for the FPGA. The detailed calculation of is provided in Section V. ii. Having the given $\vec{R}_{total}$, each DPU has to finish the execution of deep learning network by selecting a service level.

Problem Description: *Given N DPUs equipped to execute deep learning network in q service levels, determine a service level for each DPU such that the overall testing performance is maximised, while satisfying the given constraints. The pictorial description of the problem is given in Figure 1.*

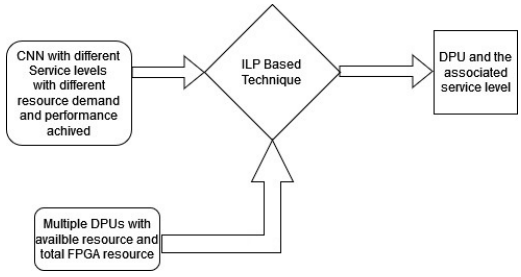

Fig. 1. The proposed framework.

III. ILP Based Level Selection Strategy

In this section, we present an *Integer Linear Programming (ILP)* solution to our proposed problem. For this purpose, we define a binary decision variables: i. $\mathcal{Z} = \{Z_i^j : i = 1, 2, ..., N; j = 1, 2, ..., q$. Here, indices i and j respectively denote DPU ID and corresponding selected service level ID. $Z_i^j = 1$, if DPU D_i executes in j^{th} service level and obtains Per_i^j performance value. $Z_i^j = 0$, otherwise.

We now present the required constraints on the decision variable to model this problem before presenting its overall objective function.

TABLE I
RESOURCE AND PERFORMANCE VALUES FOR EACH DPU

D_1			D_2			D_3		
Service level	Required resource	Obtained performance	Service level	Required resource	Obtained performance	Service level	Required resource	Obtained performance
1	2	12	1	6	2	1	7	4
2	5	13	2	14	9	2	10	6
3	7	16	3	18	16	3	13	8

TABLE II
OUTCOME: ILP

DPUs	Selected level	Obtained performance
D_1	3	16
D_2	3	16
D_3	2	6
Total obtained performance		38

1) **Resource Constraint:** The deep learning network has to be executed on FPGA within the total available resources in FPGAs, $\vec{R}_{total}$. This essentially means that the summation of consumed resource by each DPU should not exceed the given budget for the entire FPGA. This constraint is imposed through the following equation.

$$\sum_{i=1}^{N}\sum_{j=1}^{q} Res_i^j \times Z_i^j \leq \vec{R}_{total} \tag{1}$$

2) **Service level Uniqueness Constraint:** Each DPU will only be allowed to execute the deep learning network in at most one service level. That is,

$$\sum_{j=1}^{q} Z_i^j \leq 1, \forall i \in [1, N], Z_i^j \in \{0, 1\} \tag{2}$$

3) **Objective:** The objective of the formulation is to choose that feasible solution which maximizes the overall performance of the prediction / testing process through appropriate choice of service levels. Hence, the objective can be written as follows:

$$Maximize \quad \sum_{i=1}^{N}\sum_{j=1}^{q} Per_i^j \times Z_i^j \tag{3}$$

A. Example: Proposed strategies at work

In this section, we have illustrated the working mechanism of the proposed strategy through an example for ease of understanding. Let us assume, there exists three DPUs, i,e., D_1, D_2 and D_3 in an FPGA, resource demand (Res_i^j) and corresponding performance Per_i^j value for each service level is provided in Table I. We have also assumed that the available overall resource budget (RES_BGT) is 35. The total obtained result is shown in Table II. We have solved the ILP based technique for the same input values, provided in Table I through CPLEX solver [19] and the obtained outcome is presented in Table II.

IV. SIMULATION & RESULTS

The performance of the proposed strategy has been evaluated using simulation based experiments. In this current experimental scenario, We have considered that the FPGA is capable of executing deep learning network in 3 distinct service levels and FPGA consists of 3 DPUs , as shown in [17]. The area consumption and corresponding performance values have been taken from [20].

A. Results

Experiments have been conducted to evaluate the performance of the proposed strategy i.e., ILP based technique. The performance metrics which have been considered for the evaluation are:

1) Average service level allocated to each DPU
2) **N**ormalized **O**btained **T**hroughput (NOT), NOT is defined as the ratio between the ultimately achieved performance value for the DPU and the maximum possible achievable throughput by executing the network at their highest service level. Mathematically, NOT can be formulated as:

$$NOT = \frac{\sum_{i=1}^{N}(\frac{per_i^j}{per_i^q})}{N} \times 100\% \tag{4}$$

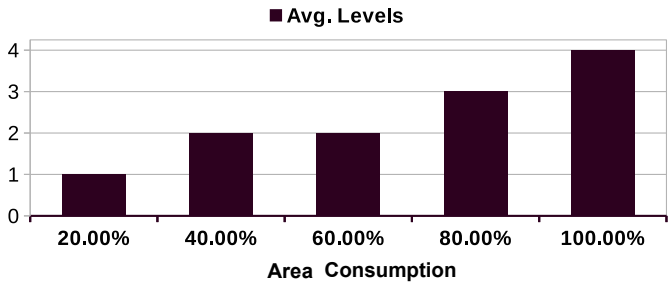

Fig. 2. Average allocated level Vs RES_BGT

Figure 2 shows the plots for the average levels allocated to each DPU by ILP based technique. As the overall resource budget (RES_BGT) varies. Here, we have normalized the total FPGA area and represented it in a scale of (0-100). It may be observed from this figure that the average level allocated to each DPU increases with the increasing available overall resource budget.

Figure 3 depicts the plot for *NOT* achieved by the strategy. It may be observed from the figure that the aggregate NOT obtained by the strategy increase with increasing available

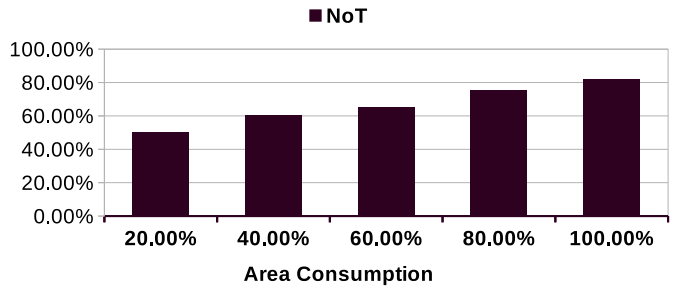

Fig. 3. NOA Vs RES_BGT

RES_BGT. This is because the NOT value obtained by the strategy is proportional to the allocated levels to the DPU and therefore, average levels allocated to all edge servers increase with available resource budget (as shown in Figure 2).

V. Experiment for Proof of Concept

In this section, we provided a proof-of-concept that each DPU in the FPGA can be configured with different service levels by varying the resource utiliation. To this end, we implement a deep learning-based image classification application on an AMD-Xilinx ZCU102 development platform (e.g. Zynq UltraScale+ XCZU9EG-2FFVB1156 MPSoC). This platform has been configured with a Petalinux-based operating system on an MPSoC, and it is configured with 3 separated DPUs within the programmable logic fabric. The Xilinx DPU is a configurable computation engine dedicated to convolutional neural networks. The degree of parallelism utilized in the engine is a design parameter and application. It includes highly optimized instructions and supports most convolutional neural networks, such as VGG, ResNet, GoogleNet, YOLO, SSD, MobileNet, FPN, and others. Each DPU is configured into different architectures with different hardware resource-allocation strategies so that the different architectures represent different "service levels," as stated in Section II.

To set up the onboard system, we have built the image file using an AMD-Xilinx Vitis 2022.1. Since AMD-Xilinx has already provided some standard developed packages, we re-designed the image file by exporting the packages and running the TCL scripts in Vitis. Two packages are used in the image file production, a MPSOC standard image system, and a ZCU102 base platform. The MPSoC standard image system package contains a prebuilt Linux kernel and root file system that can be used with any Zynq, ZynqMP, or Versal board for embedded Vitis platform developers.

In our experiments, a series of DPUs with different configurations are used as different service levels, listed in Table IV. The name of the architecture represents the peak performance of the DPU. For instance, B512 means the DPU can conduct up to 512 operations in one clock. To implement those service levels, we experimented with various parallelism techniques inside a DPU.

There are three dimensions of parallelism in the DPU convolution architecture - pixel parallelism (PP), input channel parallelism (ICP), and output channel parallelism (OCP). Figure.4 explains the meaning of each three dimensions, and pixel parallelism is 2, input channel parallelism and output channel parallelism both equal to 3 respectively in this figure. The input channel parallelism is always equal to the output channel parallelism. The different architectures require different programmable logic resources. The larger architectures can achieve higher performance with more resources. The parallelism for the different architectures is listed in Table III.

TABLE III
DPU Configurations

DPU architecture	PP	ICP	OCP	Peak operations
B512	4	8	8	512
B800	4	10	10	800
B1024	8	8	8	1024
B1152	4	12	12	1152
B1600	8	10	10	1600
B2304	8	12	12	2304
B3136	8	14	14	3136
B4096	8	16	16	4096

The experiment flow is described in Figure 5. An image classification application is running on the board, and the application is executed, split into several separated tasks through the OS on Cortex A53, and sent to different configured DPUs.

We use a Resnet-50 network to classify the image stream file with multi-thread (8). With the API interference provided by AMD-Xilinx, we deployed the neural networks in the application.

The proposed framework can also be extended to other classificication neural networks, and different model parameters would lead to a different stragety for resource allocation. Usually models with more parameters will have better accuracy (like Resnet50/Resnet25).

The program also measured the power consumption, the accuracy of the classification tasks, and the total execution time.

Three types of resources are required i.e., LUTs, BRAMs, and DSPs on the different architectures of DPUs, and the resource consumption has been described in Table IV.

There are 8 different architectures (levels) deployed on the DPUs on board, and each setting means a single DPU structure, which has different sizes of LUTs, BRAMs, and DSPs. We use these three parameters to describe the resource required with a 3-dimension vector (LUT, BRAM, DSP). Each element in the vector means the usage of the LUT or BRAM, or DSP on the DPU.

TABLE IV
Parameters for each DPU

DPU Architecture	LUT	BRAM	DSP
B512	27893	73.5	78
B800	30468	91.5	117
B1024	34471	105.5	154
B1152	33238	123	164
B1600	38716	127.5	232
B2304	42842	167	326
B3136	47667	210	436
B4096	53540	257	562

However, LUTs, BRAM and DSPs are using different units. To bring the notion of "RES"(resource) as stated in earlier

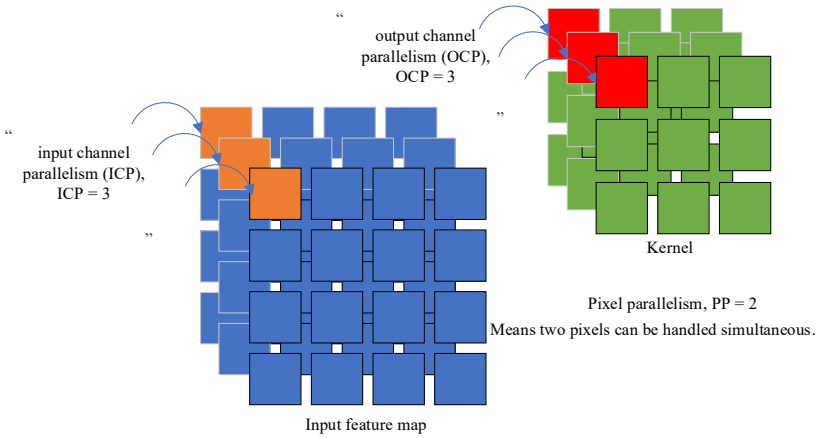

Fig. 4. An example of DPU internal arithmetic operation flow [21].

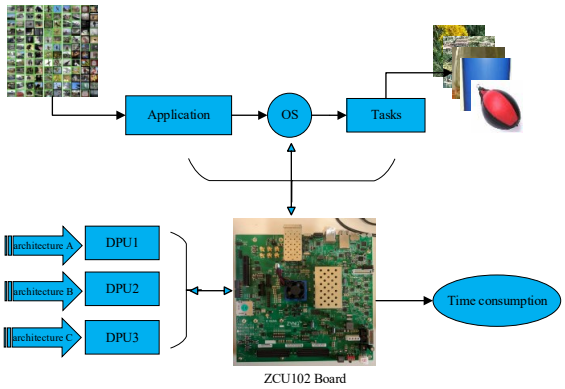

Fig. 5. Experimental illustration.

section, the numbers of LUTs, BRAM and DSPs have been normalized. R_l (No. of LUTs), R_r (No. of BRAMS) and R_d(No. of DSPs) are normalized by the total available resources in FPGAs, $\vec{R}_{total}$. For example, R_l can be represented as follows:

$$R_l = 100 \times \frac{R_l}{MaxR_l}$$

where $MaxR_l$ represents maximum available LUTs. The results have been scaled by a factor of 100 to simplify the calculation. In section II, Res is used as an unified parameter to represent required resources. Here we use the norm of $\vec{R}_{total}$ to calculate Res:

$$Res = ||\vec{R}_{total}|| = ||\{R_l, R_r, R_d\}||$$

The resource consumption for different DPU architectures is shown in Table V. These results align with our idea proposed in Section 2 and depict that the expense of higher resource consumption can achieve higher service levels. Now, we will look at how these different service levels represent the notion of performance variations.

TABLE V
NORMALIZED REQUIRED RESOURCES OF DIFFERENT DPU ARCHITECTURES

DPU Architecture	R_l	R_r	R_d	Res
B512	10.18	8.06	3.10	13.35
B800	11.12	10.03	4.64	15.68
B1024	12.58	11.57	6.11	18.15
B1152	12.13	13.49	6.51	19.27
B1600	14.13	13.98	9.21	21.91
B2304	15.64	18.31	12.94	27.34
B3136	17.39	23.03	17.30	33.65
B4096	19.53	28.18	22.30	40.90

For the performance evaluation, we evaluated it with estimated performance and onboard performance. The estimated performance is defined by the peak operations of different DPU architectures, and a higher peak operation refers that the higher data throughput will be handled per clock cycle. We use the average processing time t in a single task for the onboard performance to evaluate the efficiency of the DPUs. We choose $1/t$ to describe the onboard performance. The higher the value is, the better the performance is. The on-board and estimated performance are shown in Fig. 6.

Table V and Fig 6, validate the proposed concept in Section 2 via a real-life case study in physical FPGA. By using different settings of the DPUs, we can obtain a combination of different configurations for hardware resource allocations and the corresponding performance parameters. We can conclude that choosing the DPU at a high service level will enhance

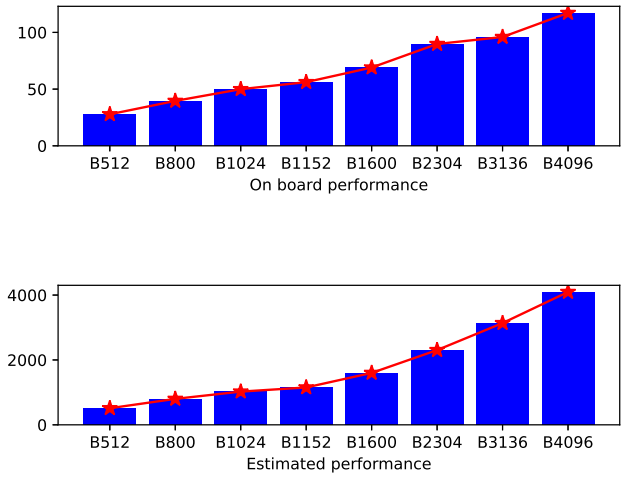

Fig. 6. Estimated performance Vs On-board performance

the performance level. Hence, the Service levels with varying degrees of *performance Vs. Resource trade-offs* are obtained by controlling the degree of resources incorporated in a DPU.

VI. Conclusion

In this work, a new concept of efficient implementation of deep learning with multiple service levels for IoT into the FPGA-based edge computing environment has been introduced. The problem has been formulated as an optimization problem where each DPU can execute the network with different service levels by exhibiting *performance Vs. Resource trade-offs*. An ILP-based strategy to maximize overall performance without violating the resource constraint has been proposed. Experimental analysis reveals the practical efficacy of our scheme. The proposed scheme can achieve 80% of throughput. Finally, a case study that implements deep learning network with multiple service levels on DPUs (on an FPGA) has been presented. The higher the resource consumption of a DPU architecture, the higher the performance will be. Thus, the obtained trend from these experiments proves that the proposed concept is valid in a real scenario on physical FPGA.

When deploying models in practical situation, resource limitation may be dominated the application requirements. Therefore, this strategy can be used as a bridge to minimise the variance by using different sizes of models, and to achieve the best performance in terms of accuracy, speed and power consumption for the application needs at run-time.

In the future, we will propose a heuristic-based strategy, and an end-to-end hardware (FPGA) validation of the software outcomes will be presented.

Acknowledgment

For the purpose of open access, the author(s) has applied a Creative Commons Attribution (CC BY) license to any Accepted Manuscript version arising.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[2] L. Dutta and S. Bharali, "Tinyml meets iot: A comprehensive survey," *Internet of Things*, vol. 16, p. 100461, 2021.

[3] M. Mohammadi, A. Al-Fuqaha, S. Sorour, and M. Guizani, "Deep learning for iot big data and streaming analytics: A survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2923–2960, 2018.

[4] Y. Wang, Q. Chen, T. Hong, and C. Kang, "Review of smart meter data analytics: Applications, methodologies, and challenges," *IEEE Transactions on Smart Grid*, vol. 10, no. 3, pp. 3125–3148, 2018.

[5] J.-F. Toubeau, J. Bottieau, F. Vallée, and Z. De Grève, "Deep learning-based multivariate probabilistic forecasting for short-term scheduling in power markets," *IEEE Transactions on Power Systems*, vol. 34, no. 2, pp. 1203–1215, 2018.

[6] H. Cai, B. Xu, L. Jiang, and A. V. Vasilakos, "Iot-based big data storage systems in cloud computing: perspectives and challenges," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 75–87, 2016.

[7] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through vm migration and transmission power control," *IEEE Transactions on Computers*, vol. 66, no. 5, pp. 810–819, 2016.

[8] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[9] X. Liu, J. Yu, J. Wang, and Y. Gao, "Resource allocation with edge computing in iot networks via machine learning," *IEEE Internet of Things Journal*, 2020.

[10] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1697–1716, 2019.

[11] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1655–1674, 2019.

[12] Y. Lu, X. Zhai, S. Saha, S. Ehsan, and K. D. McDonald-Maier, "Fpga based adaptive hardware acceleration for multiple deep learning tasks," in *2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*. IEEE, 2021, pp. 204–209.

[13] H. Li, K. Ota, and M. Dong, "Learning iot in edge: Deep learning for the internet of things with edge computing," *IEEE network*, vol. 32, no. 1, pp. 96–101, 2018.

[14] C. Liu, Y. Cao, Y. Luo, G. Chen, V. Vokkarane, M. Yunsheng, S. Chen, and P. Hou, "A new deep learning-based food recognition system for dietary assessment on an edge computing service infrastructure," *IEEE Transactions on Services Computing*, vol. 11, no. 2, pp. 249–261, 2017.

[15] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[16] Z. Deng, C. Xu, Q. Cai, and P. Faraboschi, "Reduced-precision memory value approximation for deep learning," *Hewlett Packard Labs, HPL-2015-100*, 2015.

[17] S. Goel, R. Kedia, M. Balakrishnan, and R. Sen, "Infer: Interference-aware estimation of runtime for concurrent cnn execution on dpus," in *2020 International Conference on Field-Programmable Technology (ICFPT)*. IEEE, 2020, pp. 66–71.

[18] R. Kedia, S. Goel, M. Balakrishnan, K. Paul, and R. Sen, "Design space exploration of fpga-based system with multiple dnn accelerators," *IEEE Embedded Systems Letters*, vol. 13, no. 3, pp. 114–117, 2020.

[19] I. I. Cplex, "V12. 1: User's manual for cplex," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[20] G.-Z. Lin, H. M. Nguyen, C.-C. Sun, P.-Y. Kuo, and M.-H. Sheu, "A novel bird detection and identification based on dpu processor on pynq fpga," in *2021 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*. IEEE, 2021, pp. 1–2.

[21] "Dpuczdx8g for zynq ultrascale+ mpsocs product guide (pg338)," https://docs.xilinx.com/r/en-US/pg338-dpu/reg$_{d}pu_{i}sr$, 2022.