Predictive GPU-based ADAS Management in Energy-Conscious Smart Cities

Sergio Pérez*†, Jaime Pérez*†, Patricia Arroba*†, Roberto Blanco*†, José L. Ayala‡†, José M. Moya*†

*Laboratorio de Sistemas Integrados (LSI), Universidad Politécnica de Madrid,
ETSI Telecomunicación, Avenida Complutense 30, Madrid 28040, Spain
†Center for Computational Simulation, Universidad Politécnica de Madrid,
Campus de Montegancedo UPM, Boadilla del Monte 28660-Madrid, Spain
‡DACYA, Complutense University of Madrid, Madrid, Spain.

***Abstract*—The demand of novel IoT and smart city applications is increasing significantly and it is expected that by 2020 the number of connected devices will reach 20.41 billion. Many of these applications and services manage real-time data analytics with high volumes of data, thus requiring an efficient computing infrastructure. Edge computing helps to enable this scenario improving service latency and reducing network saturation. This computing paradigm consists on the deployment of numerous smaller data centers located near the data sources. The energy efficiency is a key challenge to implement this scenario, and the management of federated edge data centers would benefit from the use of microgrid energy sources parameterized by user's demands. In this research we propose an ANN predictive power model for GPU-based federated edge data centers based on data traffic demanded by the application. We validate our approach, using real traffic for a state-of-the-art driving assistance application, obtaining 1 hour ahead power predictions with a normalized root-mean-square deviation below 7.4% when compared with real measurements. Our research would help to optimize both resource management and sizing of edge federations.**

***Index Terms*—Predictive Power Modeling, Edge Computing, Artificial Neural Network, Driving Assistance**

I. Introduction

Nowadays, Smart Cities together with the Internet of Things (IoT) technologies help to increase people's quality of life as well as efficiency in the management of resources in modern cities. The data collected by most of these infrastructures are not processed on the IoT devices, but are sent to the cloud. Some connected systems alone require a great deal of computing and data transmission capacity. Advanced Driver Assistance Systems (ADAS) are a clear example of this.

According to Nvidia, a car equipped with 10 high-resolution cameras produces 2 gigapixels of data per second that will generate a processing need of 250 trillion operations per second [1]. Clusters based on Graphic Processing Units (GPUs) are one of the best options to provide the workload requirements demanded by these applications, as they are capable to run analytic algorithms of Deep Learning (DL), and in particular Deep Neural Networks (DNNs), with a significantly higher performance than CPUs [2].

Traditionally, Cloud Computing has been developed in large data centers that collect and process all information in a centralized manner. Until few years ago, it was a viable strategy, but the increase in demand for IoT and Smart City applications makes it difficult to use only this type of infrastructures in terms of latency and bandwidth. According to Gartner [3], the number of IoT devices will reach 20.41 billion units by 2020. On the other hand, the edge computing paradigm aims to distribute the data processing to the edge of the network instead of centralizing it in a single megastructure. Thus, minimizing the volume of data sent to the Cloud and significantly reducing latencies [4].

This computing paradigm is based on the deployment of numerous and smaller edge data centers, which are located closer to the data sources, in both urban and rural areas. The use of this architecture has great advantages: (i) avoids the saturation of the Cloud, reducing network congestion; (ii) drastically decreases the latency compared to Cloud Computing; (iii) improves security (e.g. by preprocessing private data prior delivering the information to the Cloud) and (iv) provides high reliability (e.g. the edge data center will be able to maintain some services despite the failure of the provider's central servers) [5].

On the other hand, edge data centers have specific restrictions (specially in urban areas) since they need to be deployed efficiently near the data sources. The main requirements are: (i) occupy a small area, due to the high price of urban areas; (ii) be low cost, as this computing paradigm demands a high number of edge infrastructures; (iii) provide an efficient use of power consumption (not exceeding the energy supplied by the power grid because they are located near the data sources) thus improving costs and sustainability; and (iv) be dimensioned to support the traffic required by IoT services at each data source. It is worth noting that the power consumption of the edge data centers will depend on the resource demand of the application generated by the volume of users.

Our research is focused on power consumption prediction, to assist in energy optimization policies and dimensioning of the edge scenario, which will improve their sustainability and scalability. To this end, we propose a methodology based on Artificial Neural Networks (ANNs), to predict the future consumption of an GPU-based edge data center using historical data traffic. We validate our research, using real traffic for a state-of-the art ADAS application. Our model achieves an error below 7.4% interms of power prediction when compared with real measurements.

The remainder of this paper is organized as follows: Section II gives further information on the related work on this topic. The power modeling and the driving assistance case of use are presented in Sections III and IV respectively. Section V describes profusely the experimental results. Finally, in Section VI the main conclusions are drawn.

II. Background

Advanced Driver Assistance Systems (ADAS) arise from the need to obtain a safer driving experience, reducing the number of both accidents and victims. ADAS are based on monitoring the environment and the vehicle, as well as its occupants, to predict and detect emergency situations. There exists a great industrial interest in the development of technologies that allow ADAS, as a precursor of the autonomous vehicle, to become a reality.

Achieving optimized infrastructures that support these kind of technologies, in terms of power consumption and automatic maintenance, are essential requirements to enable their deployment. The application of ML and DNNs techniques to carry out these challenges has been tested with very positive results by some reliable companies. In this research [6], the Google's DeepMind team explains how they have managed to reduce by 40% the energy used in the cooling of their Data Centers, making use of DNNs-based predictive simulations.

Many research works, as in He et al. [7] and Yuan et al. [8], present edge-based frameworks for intelligent road sensing in ADAS applications. Our work help to enhance this kind of architectures to incorporate the prediction of traffic and edge power demand so energy-efficient proactive optimizations may be applied. The present research would be useful to optimize not only the resource management under power constraints (e.g. microgrid usage and power capping among others) but also the dimensioning of edge federations.

On the other hand, many research focus on modeling the power consumption of GPUs. Research by Nagasaka et al. [9] and Lim et al. [10] analyze the correlation of GPU's performance counters on the energy consumed by CUDA applications. Ge et al. [11] and Greathouse et al. [12], also study the effects of frequency management on GPU's energy efficiency. However, their research works only take into account the consumption of entire applications, not considering power variations during runtime.

Regarding the development of predictive or estimation algorithms, classical models imply important limitations with respect to the need for linearity in data (or relationships between features) and missing data. The fast generation of accurate power models for high-end servers is a complex challenge that designers have not yet fulfilled by analytical approaches. Research by Song et al. [13] provides a snapshot assessment of GPU device runtime power using Machine Learning techniques. Our research presented in this paper, based on Deep Learning models of Artificial Neural Networks, follows a similar approach.

Moreover, our work is also adapted to perform in a service-oriented application environment together with a predictive traffic model. This kind of application models, typical from the edge computing paradigm, are not yet considered in the current state-of-the-art. To the best of our knowledge our research is the first to propose GPU-edge power forecasting, allowing predictive model generation from traffic demand during runtime and considering a scenario with a high level of complexity. Using Deep Learning methods gives us great flexibility and automation in model generation, as well as the potential to enhance predictions. Also, our research allows the increase of the prediction windows once the edge data center is deployed, as the modeling process would have access to a large amount of real data to learn from.

III. Predictive Power Modeling

This work focuses on the accurate prediction of the power consumption in GPU-based edge data centers running real-time data analytics. In these systems, the resource requirements of the application depend on the data traffic demanded by users. So, predicting the number of users connected to the system is a key parameter in order to estimate the future power demand of the infrastructure.

A. Model description

Formally, we claim that traffic demand prediction ($\widehat{TD}$) for a certain time instant, Δ samples into the future, is a function of past data measurements (TD) within a window of size k = $\{0 \cdots W_{traffic}\}$ as expressed in Equation 1. For the GPU power expression in Equation 2, our claim is that the GPU power (P_{GPU}) is driven by the allocation of the traffic demand in the specific device $TD_{GPU}(t)$ and the clock and memory frequencies (f_{CLK}, and f_{MEM} respectively). It is important to note that all the data traffic must be allocated in the computing resources as in Equation 3, where $numGPUs$ is the number of GPUs in the system. Finally, the power prediction for the edge federation $\widehat{P_{EDGE}}$ depends on the traffic demand prediction and the power consumption of the GPUs of the data center federation.

$$\widehat{TD}(t+\Delta) = f(TD(t-k)) \quad (1)$$

$$P_{GPU}(t) = f(TD_{GPU}(t), f_{CLK}, f_{MEM}) \quad (2)$$

$$\widehat{TD}(t+\Delta) = \sum_{GPU=1}^{numGPUs} TD_{GPU}(t+\Delta) \quad (3)$$

$$\widehat{P_{EDGE}}(t+\Delta) = f(\widehat{TD}(t+\Delta), P_{GPU}(t+\Delta)) \quad (4)$$

Our predictive model considers specific features of the target GPU architecture that impact on power in a dynamic environment. Hence, we are able to predict in advance the total edge power consumption considering current traffic data and resource management policies based on workload allocation and Dynamic Voltage and Frequency Scaling (DVFS) techniques, thus enabling novel proactive optimizations.

B. Error Metrics

The main challenge of our research is to obtain models that accurately describe the behavior of both traffic demand and power consumption. Thus, our metrics need to evaluate the resulting error in the modeling process. In order to measure the predictions' accuracy we use the following error metrics.

Mean Square Deviation (MSD): This metric measures the average squared difference between the observed value and its prediction. In this metric all the errors provide the same weight. We use MSD as loss function during the modeling process.

Normalized Root Mean Square Deviation (NRMSD): The RMSD error is used to penalize higher errors in the sample. We normalize this value so the metric could be presented as a percentage to facilitate comparisons with other datasets or models.

Coefficient of determination (R^2): This metric provides the percentage of variance that can be predicted from the independent variable. Higher coefficient values provide better predictions.

C. Deep Learning for time series forecasting

Deep Learning recurrent methods, such as those using LSTM (Long short-term memory) or GRU (Gated Recurrent Unit) neurons, have been successfully applied to learn time dependencies automatically in time series prediction problems [14] and also natively support data entries in sequence form. Although classical machine learning methods sometimes provide better results, neural networks can offer the great advantage of not requiring previous feature engineering, and are also able to perform where classical methods fail (missing data, complex nonlinear relationships, multi-step predictions, etc.). So they become a great candidate for deployment in real-time systems with minimal human interaction, such as edge data centers.

An important factor in the development of Deep Learning algorithms is to adjust all network settings (known as hyperparameters) to provide satisfactory results. The most significant hyperparameters are number of layers, number of neurons, activation functions, loss function, optimizer, learning rate, learning decay, batch size and number of epochs among others. The definition of these hyperparameters determines how and how much the neural network learns during the training phase.

There is no optimal method to find the best configuration of the model, however it is considered good practice in most cases to proceed as follows: (i) first develop a model based on state of the art examples that obtain relatively positive results; (ii) optimize the neuronal structure (number of neurons and layers) which will define the learning capacity of the model and it is recommended to start with small structures; (iii) then, optimize the most relevant parameters in the method of training itself (batch size, epochs, loss function and optimizer); (iv) finally modify the fine adjustment parameters that help the model to reach more optimal minimums (learning rate, learning decay and activation functions).

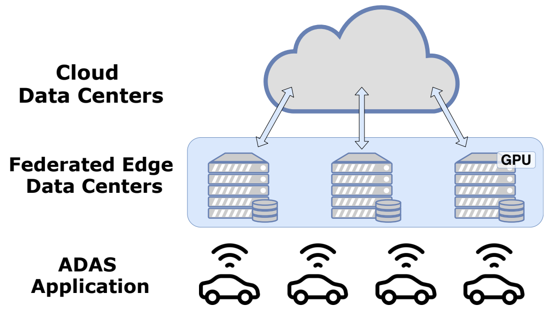

Fig. 1. Proposed ADAS scenario with edge federation

In this paper, we apply Deep Learning recurrent methods to provide models that accurately predict traffic and power demand in our case of use scenario.

IV. Case of Use: Driving Assistance in Edge Computing

In this section we describe a particular case study for the application of the devised predictive model presented in Section III. ADAS scenario presents a dynamic environment of real-time data analytics that manage high volumes of data, so its development will rely on the deployment of edge architectures. Figure 1 presents the proposed architecture of our ADAS scenario.

A. Traffic demand predictive model

To study a real scenario of vehicle traffic volume in a city with the potential to deploy edge data centers, we have selected the dataset *Hourly Traffic on Metropolitan Transportation Authority (MTA) Bridges and Tunnels* provided by *New York City Open Data*[1]. This dataset provides data, in a time series format, showing the number of vehicles (including cars, buses, trucks and motorcycles) that pass through each of the bridges and tunnels operated by the MTA per hour in New York City, USA.

The model obtains as input the traffic in the previous 24h ($W_{traffic}$ = 24) and provides as output the prediction of the traffic in the next hour (Δ = 1). In order to configure our neural networks and to perform the optimization of the hyperparameters in a more automatic way we use Talos[2], a Python library for the optimization of hyperparameters in Keras working over TensorFlow. Talos incorporates search strategies based on grid search, random search and probabilistic optimization. In this work we have used a grid search combined with a random search algorithm based on Mersenne Twister pseudo-random generator to speed up the search process.

We selected a specific measurement region (Queens Midtown Tunnel) from the data set containing 11,063 data traces (461 days, one trace per hour) of which we used 65% for the training phase, 20% for validation and 15% for test. In addition, data are normalized in the range [-1, 1] to enhance model convergence, and a cross-validation process has been

[1]opendata.cityofnewyork.us/
[2]pypi.org/project/talos/

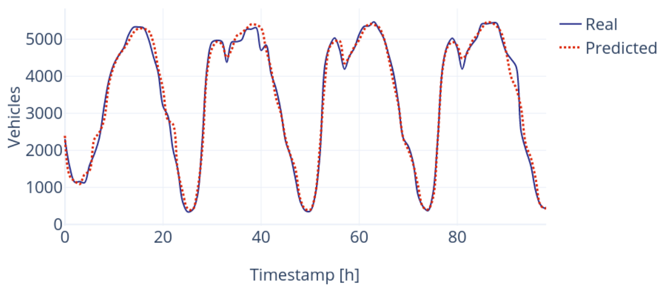

Fig. 2. Predicted number of vehicles per hour

conducted in the training data to avoid overfitting in the model. After the optimization process, the best results found were obtained with the configuration of the hyperparameters shown in Table I.

TABLE I
HYPERPARAMETER CONFIGURATION FOR TRAFFIC PREDICTION

Hidden layers:	3 GRU (64, 32 and 16 neurons) + 2 Feedforward (16 and 12 neurons)
Batch size:	72
Epochs:	100
Loss function:	Mean Square Deviation
Activation function:	ReLU
Optimizer:	Nadam
Learning rate:	0.00192
Learning decay:	0.004

Once the model has been trained and validated, the test is run obtaining a NRMSD of 4.9% and a coefficient R^2 of 97.2%. Figure 2 shows part of the test that compares our predictive model with real traffic data.

B. GPU power modeling and architecture

To model the power consumption in a GPU-based architecture, data have been collected gathering real measurements from a Sapphire Pulse Radeon RX 580. GPU power consumption, clock frequency and memory frequency have been measured using the ROCm platform and the Dynamic Kernel Module Support (DKMS). To build a model that includes power dependence with these variables, we use the ROCm System Management Interface to modify the GPU DVFS modes during runtime.

We use a real workload profile for the ADAS environment. The workload run in the GPU consists of several Convolutional Neural Networks (CNNs) running at the same time. Focusing on the ADAS scenario, vehicles need to include the best possible models at any time, ready to make the most accurate predictions for better user experience and safety. These models are Deep Neural Networks processing thousands of pictures, thus implying a high computational cost. In order to reduce computational demand on-board, edge data centers' GPUs will train the models and return the results to the vehicles for their use. In our system each car will work with a single model at a given time so we have fixed the ratio between cars and neural network models to 1:1.

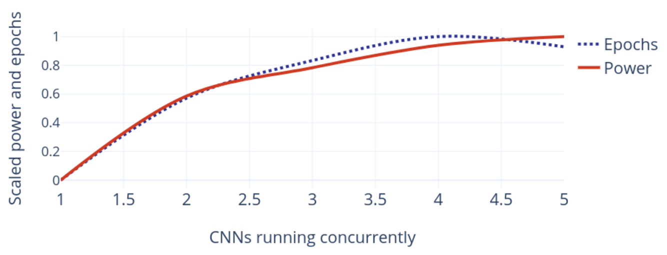

Fig. 3. Power and epochs normalized per number of CNNs

Apart from the measured GPU's variables, the number of completed epochs during a fixed time interval has been also collected as performance metric. This metric helps us to evaluate both the performance of several CNNs running concurrently and the different frequency profiles available in the GPU. As we have mentioned before, the real workload will process a high volume of images from on-board cameras featuring SD resolution (640x480 pixels), therefore the dataset used in the CNN model for the workload needs to fulfill this requirement. Consequently, we have selected the ADAS dataset from *Elektra Autonomous Vehicle*[3]. The CVC11 Driver Face dataset contains images of male and female driver's faces while driving in real scenarios recorded with an SD resolution camera. Their head's position in each image is tagged in order to use the data for designing a classification model. The model used for the CNN is based on a CIFAR-10 model[4], a dataset well-known in the field.

For achieving a complete dataset for our GPU power model, all the GPU variable combinations are exploited during the workload. These variables are: a) the memory frequency, which features three frequencies ranging from 300MHz to 2,000MHz; b) the clock frequency, which features eight frequencies ranging from 300MHz to 1,366MHz; and c) the number of CNNs running in parallel, ranging from one to five scripts. We decided to run just up to five of them because from that point the efficiency (completed epochs compared to power consumption) diminishes, as seen in Figure 3.

Thus, we obtain 120 possible combinations, each of them tested for ten minutes. Data logs are recorded every ten seconds, sent by Kafka Streams and stored in an Apache Cassandra database. The final dataset includes 6,420 records, of which we use 70% for training, 20% for validation and 10% for test. The test consists of data belonging to all possible combinations of model inputs (memory frequency, clock frequency and number of running CNNs) to verify that the model is valid for all possible GPU configurations. Also, before entering the data into the neural network they are normalized in the range [-1, 1]. For the hyperparameters optimization in this power model, we apply the same methodology as in the traffic forecasting model explained in Section III. Best results are shown in Table II.

Once the model has been trained and validated, the test

[3]adas.cvc.uab.es/elektra/
[4]https://keras.io/examples/cifar10_cnn/

TABLE II
HYPERPARAMETER CONFIGURATION FOR POWER CONSUMPTION

Hidden layers:	2 Feedforward (32 and 12 neurons)
Batch size:	5
Epochs:	140
Loss function:	Mean Square Deviation
Activation function:	ReLU
Optimizer:	Nadam
Learning rate:	0.002
Learning decay:	0.004

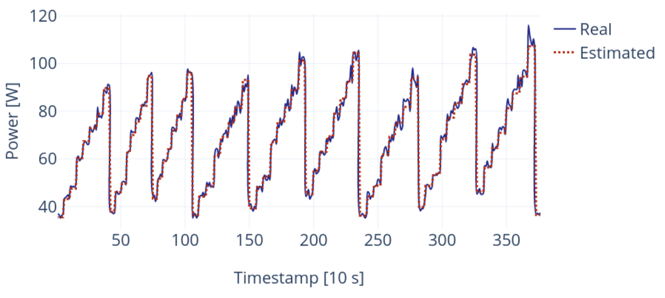

Fig. 4. Power model testing

is executed obtaining an NRMSD of 2.45% and an R^2 of 99.01%. Figure 4 shows a fragment of the test (since the entire test is too large to be visualized with precision) that compares the estimations with the real measures of GPU power consumption.

V. PERFORMANCE EVALUATION

A. Edge dimensioning

The edge data center federation consists of a fixed number of GPUs, enough to cover the maximum anticipated demand. Searching in the dataset, the maximum number of vehicles passing through the road toll within an hour is 6,135. However, as we have stated before, each GPU can feature up to five CNN scripts running at the same time. After analyzing the consumed power and completed epochs per number of CNN scripts running concurrently (our performance metric), which can be seen in Figure 3, we found out that the most efficient number of CNNs per GPU is four, thus the final and total number of GPUs within our edge infrastructure is 1,534. On the other hand, the frequency set in the GPUs, for both memory and clock frequencies, is the maximum allowed in the device. The use of different frequency ranges will be further analyzed in our future work.

B. Resource allocation strategies

Edge data centers' workload may be allocated using different approaches for better performance and higher power savings. In this paper, we have applied three different strategies: Round Robin, Stack OFF and Stack ON. Round Robin consists in allocating the minimum number of CNN scripts per GPU while using the maximum number of devices. Stack OFF and Stack ON policies allocate the maximum number of CNN scripts in a GPU before using the next one. The GPUs that do not feature any CNNs at runtime remain switched off

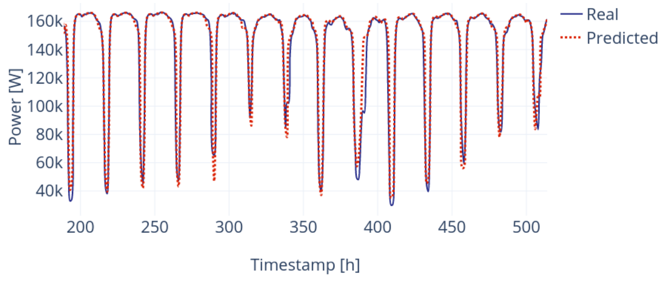

Fig. 5. Power prediction for Round Robin strategy

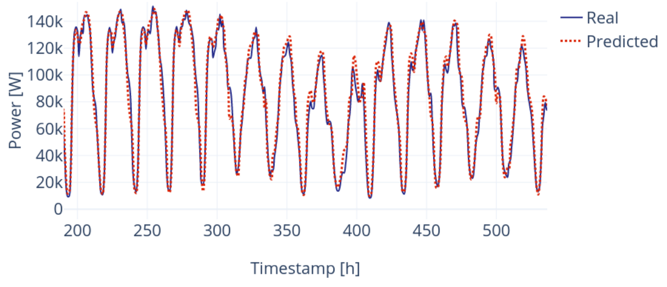

Fig. 6. Power prediction for Stacked Off strategy

or in idle state for Stack OFF and Stack ON respectively. There is a trade-off between these two options, the first one reduces power consumption but makes the system more time-consuming while the second one penalizes power consumption to obtain quicker responses.

C. Final results and error metrics

We calculate for each strategy (for both real and predicted traffic) the number of CNNs and GPUs needed to cover the demand per timestamp, and the power consumed by it using the data from the test dataset. Following, we obtained the error of each strategy by applying our power model and comparing the estimated power using the real and the predicted traffic, thus providing the NRMSD error metric. Our results can be found in Table III. Figures 5, 6 and 7 present the power predictions for the Round Robin, Stack OFF and Stack ON allocation strategies respectively. Figure 8 present the comparison of the power evolution for each allocation policy.

TABLE III
ENERGY AND ERROR PER RESOURCE ALLOCATION STRATEGY

Allocation strategy	P(TD) $\cdot t$ [kW $\cdot h$]	P($\widehat{TD}$) $\cdot t$ [kW $\cdot h$]	NRMSD [%]
Round Robin	144.54	145.87	4.70
Stack OFF	87.41	89.62	4.90
Stack ON	125.60	126.76	4.90

On the other hand, we calculate the NRMSD for the overall test as the summation of two independent errors, the predicted traffic NRMSD (4.95%) and the estimated power NRMSD (2.45%), resulting in an error below 7.4%. Comparing it to the strategy errors in Table III, we find out a higher value than those due to the use of the estimated power for calculating

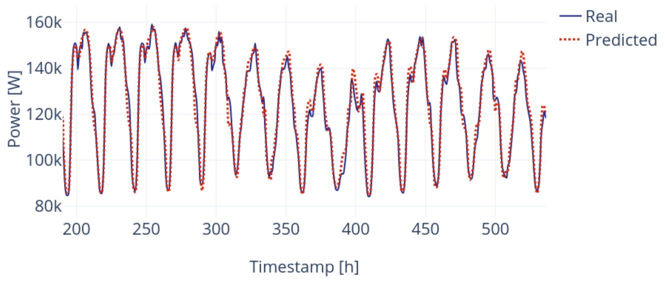

Fig. 7. Power prediction for Stacked On strategy

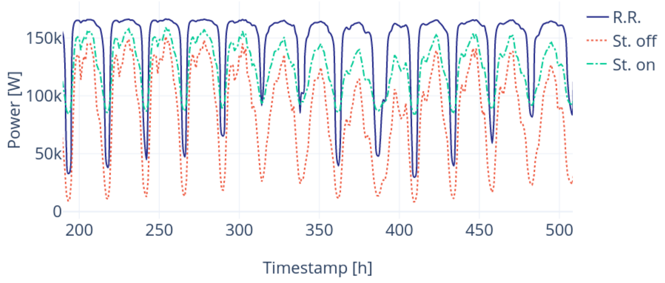

Fig. 8. Power prediction comparison for the allocation strategies

the power consumed per strategy, thus the error obtained in each strategy is akin to the predicted traffic error and not to the overall error. Our proposed scenario for 461 days of real traffic demand presents a edge data center federation of 1,534 GPUs. Comparing the energy consumption prediction using the actual and forecasted traffic, we obtain errors below 2.3 kWh.

From these results, several conclusions are inferred: a) The model's accuracy is sufficient for applying optimization policies one hour in advance; b) our research allows the design of different resource allocation techniques leading to high power savings as demonstrated for the ADAS case of use; c) without these techniques, edge data centers would undergo an overprovisioning problem resulting in underutilized resources, which are limited and highly demanded, as well as in higher economic expenses; d) comparing the energy consumed by the best and worst strategy, our results show a 38.6% reduction in power savings, presenting a high potential for the use of energy-efficient runtime optimizations; e) the results contribute to define a general methodology for edge data centers' dimensioning based on GPU architectures running real-time Deep Learning data processing.

VI. Conclusions

In this paper we have presented a novel research for the automatic generation of models, to predict the power behavior of GPU-based edge data centers during runtime, for real-time data analytics applications. As a proof of concept, the devised method has been applied to the ADAS scenario using real traces of traffic demand, with a real CNN-based application profile, on real GPU devices. Our approach provides the automatic generation of a predictive model based on Deep Learning that forecasts the energy consumption of the edge infrastructure with an NRMSD error of less than 7.4%. Our modeling strategy combines a traffic demand prediction model (based on Recurrent Neural Networks) and a power estimation model (based on Feedforward neurons) as a function of the demand executed by the GPUs. In our scenario, by sizing the edge data center for 1,534 GPUs and comparing the energy consumption estimation with 461 days of the actual traffic and forecasted traffic, we obtain errors below 2.3 kWh. This research would help on the design of new proactive optimizations to improve energy efficiency in edge data centers, thus contributing to the research, development and deployment of service-oriented applications in the edge computing paradigm.

Acknowledgment

This project has been partially supported by the Centre for the Development of Industrial Technology (CDTI) under contracts IDI-20171194, IDI-20171183 and RTC-2017-6090-3 and by the and by the Spanish Ministry of Science, Innovation and Universities under the contract TIN2015-65277-R.

References

[1] NVIDIA, "Self-driving safety report," NVIDIA Corp., Tech. Rep., 2018.
[2] G. Ananthanarayanan, P. Bahl, P. Bodk, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha, "Real-time video analytics: The killer app for edge computing," *Computer*, vol. 50, no. 10, pp. 58–67, 2017.
[3] Gartner. Press Releases, "Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016," 2017.
[4] J. Pan and J. McElhannon, "Future edge cloud and edge computing for internet of things applications," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 439–449, Feb 2018.
[5] C. Chang, S. N. Srirama, and R. Buyya, "Internet of things (iot) and new computing paradigms," *CoRR*, vol. abs/1812.00591, 2018.
[6] R. Evans and J. Gao, "Deepmind ai reduces google data centre cooling bill by 40%," https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/, online; accessed 3 April 2019.
[7] Y. He, X. Fan, F. Wang, F. Wang, and J. Liu, "Edge computing empowered generative adversarial networks for realtime road sensing," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, June 2018, pp. 1–2.
[8] Q. Yuan, H. Zhou, J. Li, Z. Liu, F. Yang, and X. S. Shen, "Toward efficient content delivery for automated driving services: An edge computing solution," *IEEE Network*, vol. 32, no. 1, pp. 80–86, Jan 2018.
[9] H. Nagasaka, N. Maruyama, A. Nukada, T. Endo, and S. Matsuoka, "Statistical power modeling of gpu kernels using performance counters," in *Int. Conference on Green Computing*, Aug 2010, pp. 115–122.
[10] J. Lim, N. B. Lakshminarayana, H. Kim, W. Song, S. Yalamanchili, and W. Sung, "Power modeling for gpu architectures using mcpat," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 19, no. 3, pp. 26:1–26:24, Jun. 2014.
[11] R. Ge, R. Vogt, J. Majumder, A. Alam, M. Burtscher, and Z. Zong, "Effects of dynamic voltage and frequency scaling on a k20 gpu," in *Proceedings of the 2013 42Nd International Conference on Parallel Processing*, ser. ICPP '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 826–833.
[12] J. L. Greathouse and G. H. Loh, "Machine learning for performance and power modeling of heterogeneous systems," in *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, Nov 2018, pp. 1–6.
[13] S. Song, C. Su, B. Rountree, and K. W. Cameron, "A simplified and accurate model of power-performance efficiency on emergent gpu architectures," in *2013 IEEE 27th International Symposium on Parallel and Distributed Processing*, May 2013, pp. 673–686.
[14] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu, "Lstm network: a deep learning approach for short-term traffic forecast," *IET Intelligent Transport Systems*, vol. 11, no. 2, pp. 68–75, 2017.