Evaluation and Prediction of Resource Usage for multi-parametric Deep Learning training and inference

Eleni Tsalera *
Department of Informatics and Computer Engineering University of West Attica, Greece
etsalera@uniwa.gr

Dimitrios Stratogiannis
Department of Electrical and Electronic Engineering Educators, School of Pedagogical and Technological Education (ASPETE), Athens, Greece
dstratog@mail.ntua.gr

Andreas Papadakis
Department of Electrical and Electronic Engineering Educators, School of Pedagogical and Technological Education (ASPETE), Athens, Greece
apapadakis@aspete.gr

Ioannis Voyiatzis
Department of Informatics and Computer Engineering University of West Attica, Greece
voyageri@uniwa.gr

Maria Samarakou
Department of Informatics and Computer Engineering University of West Attica, Greece
marsam@uniwa.gr

ABSTRACT

Deep learning is increasingly used in diverse application fields with results typically surpassing those of traditional machine learning techniques. The portfolio of available neural networks is wide, consisting of the full range in terms of complexity, from compact networks to large ones with multiple layers and parameters. This heterogeneity in the model topology is reflected, not necessarily linearly, on the required computational resources for training and inference. Similarly, the environments where the neural networks are trained and executed are transformed from fully-fledged centralized nodes to distributed architectures with constrained resources. In this view, computational resource requirements can be one of the criteria for resource usage management and neural network selection. In this work we measure the training times for a set of five convolutional neural networks of varying complexity and age (GoogleNet, ShuffleNet, VGGish, YAMNet) under different training configurations, considering the batch size, the number of epochs and the learning rate. These measurements are used to create a CPU-time-training dataset of more than 500 values. This dataset is used to train and evaluate models, based on neural networks, for estimating and predicting training times depending on the models employed and the training parameters. Five regression models have been trained and evaluated in terms of correlation coefficient and root mean square error. In addition, we measure the CPU times needed for inference for a subset of the trained models, which prove to be uncorrelated with the corresponding training times.

*Corresponding author.

PCI 2023, November 24–26, 2023, Lamia, Greece

ACM ISBN 979-8-4007-1626-3/23/11.
https://doi.org/10.1145/3635059.3635070

KEYWORDS

Computing methodologies, Machine learning, Machine learning approaches, Classification and regression treesAdditional Keywords and Phrases: Convolutional neural network, computational resources, CPU time, training and inference durations

ACM Reference Format:
Eleni Tsalera, Dimitrios Stratogiannis, Andreas Papadakis, Ioannis Voyiatzis, and Maria Samarakou. 2023. Evaluation and Prediction of Resource Usage for multi-parametric Deep Learning training and inference. In *27th Pan-Hellenic Conference on Progress in Computing and Informatics (PCI 2023), November 24–26, 2023, Lamia, Greece.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3635059.3635070

1 INTRODUCTION

Recent advancements in computing, mostly related to the application of deep learning techniques in a wide variety of domains (e.g., computer vision and language processing) and the integration of IoT technologies and edge computing impose the development of cost-effective hosting of multiple applications in a secure, customizable computing environment. This yields to higher resources utilization with reduced costs at all levels [1]. More in detail, deep learning algorithms are considered as the state-of-the-art techniques for several computing tasks related either to image processing with high-level of understanding of semantics requirements such as image classification, object segmentation and clustering, anomaly detection or either cases where low level image processing tasks are required. In parallel, datasets are being created, to train and validate the learning algorithms and tasks can take place in a centralized manner and / or in a distributed manner involving devices and nodes of varying resources and capabilities. To support its increasingly diverse application fields, the deep learning models evolve into larger ones with more layers and parameters. This results in increasing training time and need for resource utilizations. Despite the advances in learning schemes, users are still facing issues related to the optimal configuration of deep learning execution settings and mechanisms related to computing resources allocation and their relation to the training time and the accuracy.

Resource management techniques play a noteworthy role for all types of computing and it is related to the requirements of high virtualization, scalability, and transparency. Resource monitoring is a part of resource management for a modern computing environment where deep learning is applied, which provides a better understanding for resource allocation serving towards the increased efficiency of services such as: task scheduling, capacity planning, proactive auto-scaling and load balancing, improving computing and network performance. A prerequisite for effective resource management is a priori estimation of the resources needed.

This is a challenging point as there is a lack of an association mechanism between the learning tasks (training and inference), the volume of the networks involved and the resources needed, in terms of CPU and memory. Towards this goal an accurate and simple estimation of resources utilization is required considering the dynamic and time-varying workloads of modern computing environments under different constraints. A simple model based on existing from the self-products may serve this scope in order to capture the workload dynamics. Furthermore, modern computing devices seem as deep learning friendly managing different workloads under a resource constraint environment [2]. Despite, the new design approaches in the area, resource allocation evaluation and performance monitoring for deep learning applications remains stimulating especially when deep learning models should be deployed under a resource constraint environment of e.g. mobile devices or IoT sensors, where the optimal configuration of execution settings is required.

In this paper, a study on the required computational resources is performed presenting different scenarios of deep-learning schemes. The method presented is based on a multi parametric evaluation of resource utilization based on simple tools to identify Convolutional Neural Networks' (CNNs) configurations, training parameters and input characteristics so that we can perform a series of training and inference experiments under different problem settings related to computational resources and achieved accuracies. Our adaptive multi-parameter approach considers different scenarios encountered in image and sound classification and enables selecting the predictive method, based on regression, that learns best and predicts resource requirements.

The rest of the paper is organized as follows. In Section 2, selected related work is presented focusing on performance evaluation and monitoring of computing resources. In Section 3 the applied methodology is described as well as the CNNs and the datasets employed. Section 4 presents the set of parameters' values that affect the training time, and the relevant measures. These results have been used to train selected regression models (based on neural networks), and the performance of these models is described in terms of correlation coefficient and root mean square error. Section 5 includes the conclusions and future work.

2 RELATED WORK

A detailed survey in the area of efficient inference is provided in [3], where the main research challenges are identified and analyzed. In this paper, the significance of resource constraints and monitoring are studied considering edge environment. In addition, a representative survey analysis for deep learning models is presented providing also the appropriate metrics for the performance evaluation. A comprehensive analysis of deep neural network architectures is given in [4]. In this work, various neural network architectures are described and evaluated based on multiple performance indices, such as recognition accuracy, model complexity, computational complexity, memory usage, and inference time. This study provides an understanding over the impact of resource constraints when these architectures are under practical deployments. More in detail, 44 different neural networks models are tested and evaluated in a resource constrained environment, giving experimental results by the neural networks testing.

Moreover, in [5] an analysis of pre-trained neural networks application is performed. The importance and the usefulness of pretrained models is presented, showcasing the transfer learning approach. In this case, the advantages of the pre-trained models are presented in terms of computational requirements and resources, enabling faster inference and reducing complexity.

Estimation of resources utilization and the accurate use of the appropriate model has been investigated in different contexts. In [6], a classifier is trained using previous resource usage to select a resource utilization prediction model within specific time intervals. Towards this goal, various prediction methods are applied in order to improve the accuracy of the forecasted computing resources since the prediction of resource requirements, such as CPU usage, is a complex task amd depends on incoming workloads. In [7], a deep learning model is proposed and evaluated according to its forecasting accuracy and error minimization. Similar studies have been presented aiming to predict the resources and support the most appropriate network based on evaluation techniques showing the importance of resource performance evaluation, such as [8] where a novel scheme is proposed. More in detail, graph neural networks are employed to predict the resource consumption of diverse workloads and transfer learning can be further exploited in order to extend the graph neural networks and adapt to differences in computing and resources environments. Moreover, optimization techniques are applied in [9] where adaptive model selection is applied based on machine learning to develop a low-cost predictive model to quickly select a pretrained neural network by considering the desired accuracy and inference time on a given data input.

Finally, the impact of transfer learning is examined in [10], where the authors have explored the influence of retraining parameters, including the optimizer, the mini-batch size, the learning rate, and the number of epochs, on the classification accuracy and the processing time needed. Another technique employed to increase efficiency and potentially decrease the computational load required is to select a subset of the available features of the principal signal (such as sound or image / video) so that the models are trained upon this subset. A methodology employing feature selection based on the Principal Component Analysis (PCA) weights is presented in [11].

3 METHODOLOGY

In this work, we observe and measure the computational (CPU) time needed for the training of a selected set of convolutional neural networks, under different training parameters and configurations. These networks have been trained mainly for audio classification. Audio files are converted into scalograms and/or spectrograms, i.e.,

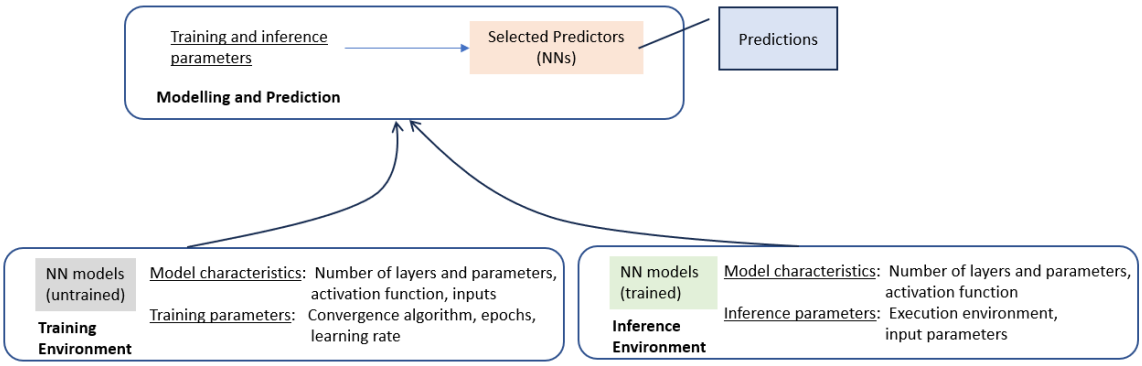

Figure 1: Prediction based on measured load during training and inference

figures, which have been used for training the neural networks. Similarly, we have collected the corresponding times of a subset of the trained models when they are used for inference. As depicted in Figure 1, the training and inference activities (lower layer) depend on the environment, the model characteristics and the training or inference parameters respectively. The configuration and measurements retrieved from the training and inference activities are provided for modelling and prediction (upper layer of Figure 1). In this layer a set of models is retrained upon these parameters and measurements so that they can provide estimations and predictions (output of the overall platform). Such predictions constitute the estimations of the required resources for training and inference activities upon a representative subset of CNNs and can support / guide efficient management of computational resources as well as the selection of CNNs (considering at the same time the achieved classification accuracies).

The neural networks that have been selected for training and inference include GoogleNet, SqueezeNet, ShuffleNet, VGGish, and YAMNet. These networks are well-known, and widely used in a rich portfolio of applications. They are used for image recognition and classification (with these images being the spectrograms and or scalograms of converted audio files). The design of the selected networks starts from 2014 (year when GoogleNet has been designed) until recent years. They are also adequately complex, as the number of layers ranges from 9 to 50, allowing for efficient parameterization of their training procedures. Network characteristics include the architecture and design of the network, the number of layers, number of parameters, the activation function, and the size in memory.

GoogleNet is a relatively complex network consisting of twenty-two layers with 7 million parameters and uses filters of different dimensions. The outputs are concatenated into a single output (inception module). The usage of 1x1 convolutional layer in the Inception module and the pooling result in a significant reduction of the number of parameters [12]. SqueezeNet, with its eighteen layers, includes 1.24 million parameters. It employs the fire module, which reduces the dimension of the feature map and concatenates the feature maps [13]. ShuffleNet consists of 50 layers and 1.4 million parameters, using shuffling and pointwise convolution to increase classification accuracy and reduce computational requirements [14]. VGGish and YAMNet CNNs, with nine and twenty-eight layers respectively are used to classify sounds converted to images. VGGish involves 72.1 million parameters and YAMNet 3.75 million respectively.

Training of a neural network is computationally intense procedure, possibly involving extensive periods of time. A working option is the application of transfer learning, knowledge retrieved from one origin (source) domain is applied to another destination (target) domain. Practically, this means that the network is initially trained in the origin domain and a part (or parts) of it is retrained based on the data of the destination domain. Apart from the reduction of the requirements on training time and resources, this mechanism also alleviates the possibility of a reduced dataset in the destination domain. The extension of the network part depends, typically involving only the classifier part. In some cases, the convolutional part of the network may be re-trained (with the addition, removal, or adaptation of layers) along with the classification part. In rather infrequent cases, the full network is trained from scratch considering the datasets of the destination domain.

The selection depends on each specific problem, as well as the affinity of the origin and destination domains. As the more typical cases involve the adaptation (retraining) of the classification parts of the networks, keeping the overall architecture as well as the number of layers. In this view our measurements refer to the retraining duration (CPU time), when the classification part of the previous networks is affected. The training (retraining) is affected by parameters which influence the computational resources. These involve the optimizer, the size of mini-batches, the number of epochs, and the learning rate.

Of course, the input dataset parameters also play their role. These involve the number of the files used for training, their resolution and format (for example jpg, png or tiff). GoogleNet and ShuffleNet receive as input images of $224 \times 224 \times 3$, while images used by SqueezeNet have resolution of $227 \times 227 \times 3$. VGGish and YAMNet receive input image size of $96 \times 64 \times 1$.

The datasets involved include UrbanSound8K, the ESC-10, and the Air Compressor. Interesting dataset characteristics include the number of files and the number of classes. UrbanSound8K (with its 8732 files) includes 10 classes of outdoor sounds, ESC-10 dataset

Table 1: Network and training parameters and their values

Parameter	Values				
CNN	GogLeNet	SqueezeNet	ShuffleNet	VGGish	YAMNet
# Layers	22	18	50	9	28
Parameters (million)	7	1.24	1.4	72.1	3.75
Memory size (MB)	27	5.2	5.4	289	15.5
Input resolution	224x224x3	227x227x3	224x224x3	96x64x1	96x64x1
Training hyperparameters	Optimizer	Mini-batch size	Epochs	Learning rate	
	Adam, SGDM	8,16, 32 (Image CNNs)	6, 8, 10	0.5, 1, 2 ($x10^{-3}$)	
		64, 128, 256 (Sound CNNs)			
Datasets	UrbanSound8K	ESC-10	Air Compressor		
# Files	8732	400	1800		
# Classes	10	10	8		

also 10 classes, with each one consisting of 40 audio ogg files. The Air Compressor dataset contains eight classes with each class being associated with 225 audio wav files.

4 RESULTS AND DISCUSSION

4.1 Measurements Dataset

The dataset we have created includes the CPU time measurements according to the number of CNNs, the number of training datasets, as well as the number of applicable optimizers, mini-batch sizes, epochs and learning rates:
$[\#\ CNNs] * [\#\ Datasets] * [Optimizers] * [\#Mini-Batch\ Sizes] * [\#Epochs] * [\#LearningRates]$

The available values are presented in Table 1 and have resulted in 504 combinations and measurements.

The CPU time is measured using the environment employed to perform both training and inference, namely Matlab (version 2023b). In addition to the training CPU times, the corresponding times for inference have been measured for a subset of the trained networks (one per family) and the relationship (correlation) with investigation times has been investigated. All training and inference sessions are performed using the same computational infrastructure and normalization has been applied. Specifically, a desktop PC with memory of 32 GB RAM, processor Intel Core i7-10700K, eight cores up to 3.8GHz, with the graphic card NVIDIA GeForce RTX 3060 has been used.

4.2 Prediction based on Regression

Prior to training, a selection process among available regression models has taken place. The procedure involves training candidate models including linear regression, support vector machines, regression trees and ensembles of regression trees, kernel approximation as well as neural networks. The selection has been based on the minimization of the validation error, calculated as the Root Mean Squared Error (RMSE).

All selected models have been based on neural networks and specifically include the narrow, the medium and the wide neural networks with 1-layer and the bi-layered and tri-layered networks with 2 and 3 layers respectively. All networks use the ReLu activation function.

For training and testing, 80% and 20% of the dataset have been used (respectively) and the prediction results for the CPU training times are depicted in Figure 2. This figure consists of 5 sub-figures presenting the results of each of the five predicting neural networks. The evaluation of available models has been performed considering the root mean square error (RMSE), and the coefficient of determination (R^2). The calculation formulas are provided below:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\widehat{load\,(i)} - load\,(i)\right)^2} \tag{1}$$

$$R^2 = \frac{SSR}{SST} = 1 - \frac{SSE}{SST} = 1 - \frac{\sum_{i=1}^{N}\left(\widehat{load\,(i)} - load\,(i)\right)^2}{\sum_{i=1}^{N}\left(load\,(i)\right) - \frac{1}{N}\sum_{i=1}^{N}\left(load\,(i)\right)} \tag{2}$$

Where $\widehat{load(i)}$ is the predicted value and $load(i)$ the actual value. Furthermore, SST is the sum of squares total, SSR the sum of squares regression, and SSE the sum of squares error.

The correlation coefficients R versus the RMSE for each predictor are presented in Figure 3, vertical and horizontal axis respectively.

As presented in Figure 3, the model that provides the most accurate prediction is the bi-layered network, which achieves the largest value of R (slightly below 0.86) with the smaller value of RMSE (slightly larger than 270). It is followed by the narrow neural network and the medium network. The tri-layered and the wide networks present the weakest performance despite their relatively complex structure.

4.3 Inference time

We have measured the inference time for a subset of the trained networks, one per category and with the most typical training configurations. Specifically, the optimizer has been Adam, the epochs 8, the learning rate 2×10^{-3} and for GoogleNet, ShuffleNet and SqueezeNet the mini-batch size has been set to 32, while for the Sound CNNs the mini-batch size 256 (as in the pre-processing phase each file is split by 0.96). The UrbanSound8K data set has been used for inference

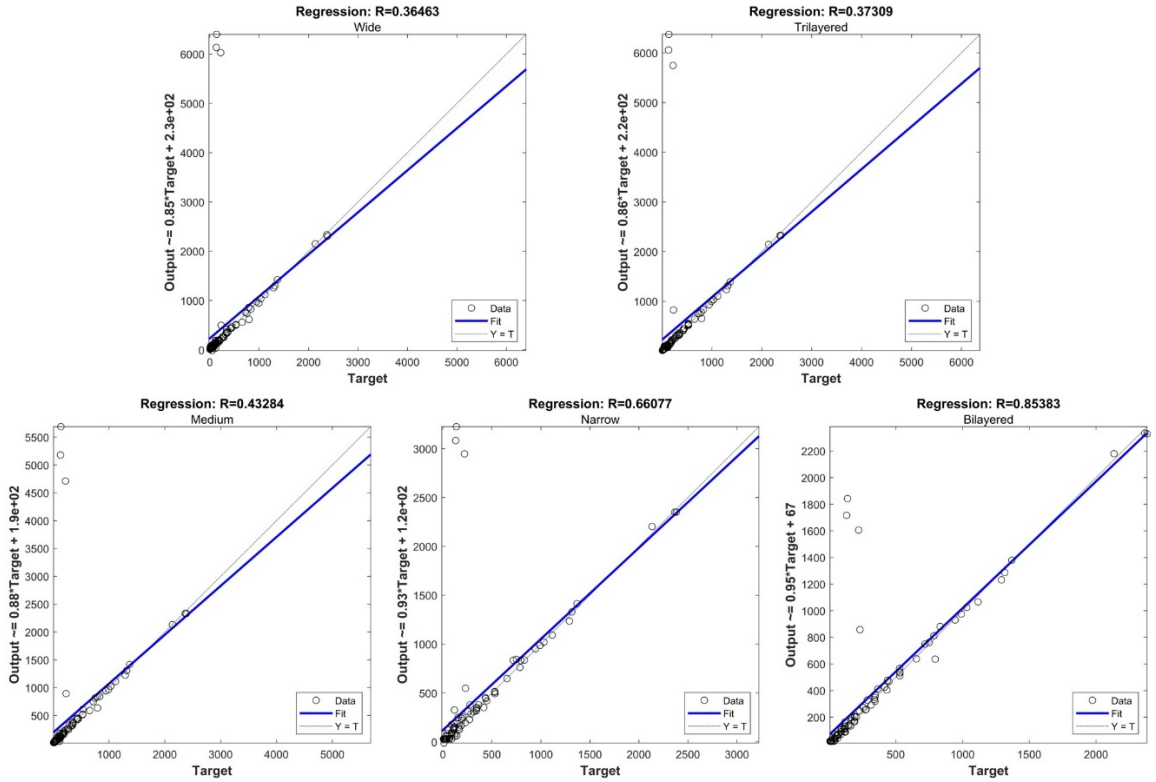

Figure 2: Prediction results

Table 2: The training and inference time for each CNN.

CNN	Training time (s)	Inference time (ms)
GoogleNet	342	2.04
SqueezeNet	154	1.28
ShuffleNet	1063	2.00
VGGish	373	2.94
YAMNet	590	26.67

and specifically the 20% of the files (corresponding to 1747 sound files). The training and inference times are depicted in Table 2.

ShuffleNet has the longest training time, which is partially explained by its architecture, while its inference time is shorter. On the other hand, YAMNet has a relatively large training time and the largest inference time. SqueezeNet has the shortest inference duration, and this makes it more appropriate for environments of more constrained resources. Another interesting observation is that the inference and the corresponding training durations are not correlated (the correlation index is at 0.15).

While in a centralized deployment, inference and training may take place independently even using different environments (training in the central node and inference at peripheral nodes), in more distributed and decentralized settings, training and inference can be intertwined, taking place in the same node and with different frequency. In this view, we can consider a linear combination of the training and inference durations to form an index, as below:

$$I_{train,inf} = a_1 T_{tr} + a_2 T_{inf} \quad (3)$$

The coefficients a_1 and a_2 can be approached through the estimation of relative frequency of training and inference respectively.

5 CONCLUSION AND FUTURE WORK

Training and re-training of neural networks can be challenging in terms of computational resources, while their execution to provide inference also requires a portion of such resources. Given a specific computational environment, the resource requirements, along with the performance and the accuracy achieved, may be a criterion for selecting deep learning models. For this, mechanisms and tools are needed to estimate the required resources.

In this work we have considered a set of five convolutional neural networks (CNN) used for image classification, GoogleNet, SuffleNet,

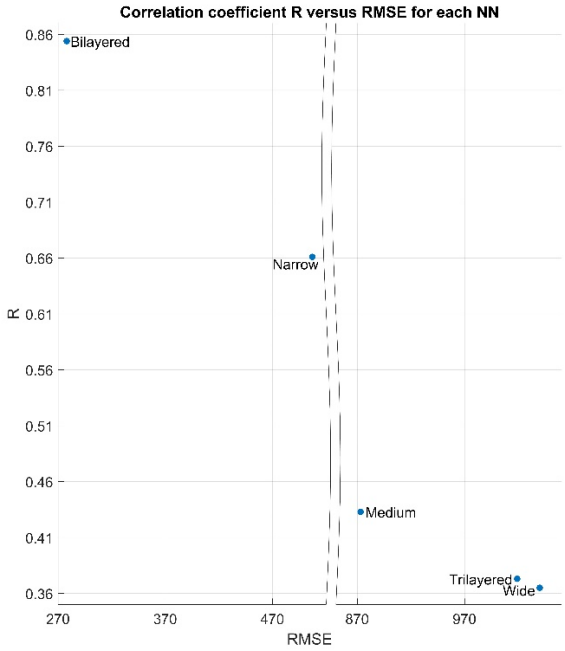

Figure 3: correlation coefficient R versus the RMSE for each predictor

SqueezeNet, VGGish and YAMNet. The selection has been representative in terms of the evolution of the NNs and their complexity, ranging from 9 to 50 layers and reaching up to 72 million parameters. These models have been re-trained according to different training configurations, changing the values of epoch numbers, training rates and batch sizes. For each of the combinations of the NNs and training parameters, the training CPU time has been measured, resulting in a dataset of more than 500 values, offering adequate variation.

The second step of the work has been to train a set of NN-based regression models for the estimation of the training CPU time depending on the selected model and training configuration. Five neural networks have been used and evaluated in terms of root mean squared error and correlation coefficient. From them, the bilayered neural network, followed by the narrow one, has achieved high predictive performance, offering the largest correlation coefficient (R) and the lower root mean square error.

Another observation is related to the measurements of the inference times for each of the trained models. While for the four out of five of the models the inference duration has been relatively homogeneous (from 1.28 to 2.94 milli-seconds), YAMNet presents much higher inference time. It is also interesting that the training and inference durations are not correlated. To combine these two metrics (training and inference durations) we have also briefly discussed a straightforward, linear combination, based on the relative frequencies of training and inference activities.

In terms of future work, we foresee to extend the concept of observing the usage of computational resources, in different execution environments, including other portable devices, such as Raspberry pi and smart phones. In addition, the measurements may consider additional neural network architectures, performing classification, regression and clustering, so that the methodology is application agnostic. In addition, while in this current work, we have employed the CPU time, as an indication of the computational load for training and inference, this can be extended to the other computational resources such as the memory employed by the model. At last, the straightforward, linear combination of training and inference duration can be further extended, considering the characteristics of the resource-constrained infrastructure, as performed in [15].

ACKNOWLEDGMENTS

The paper has been partially or fully funded by the University of West Attica.

REFERENCES

[1] Davy Preuveneers, Ilias Tsingenopoulos and Wouter Joosen. 2020. Resource Usage and Performance Trade-offs for Machine Learning Models in Smart Environments. Sensors, 20(4), 1176. https://doi.org/10.3390/s20041176

[2] Jingoo Han, Luna Xu, Mustafa M. Rafique, Ali R. Butt and Seung-Hwan Lim. 2019. A Quantitative Study of Deep Learning Training on Heterogeneous Supercomputers. 2019 IEEE International Conference on Cluster Computing (CLUSTER), 1-12. https://doi.org/10.1109/CLUSTER.2019.8890993

[3] Md. Maruf H. Shuvo, Syed. K. Islam, Jianlin Cheng and Bashir I. Morshed. 2022. Efficient Acceleration of Deep Learning Inference on Resource-Constrained Edge Devices: A Review. In Proceedings of the IEEE, 111(1), 42-91. https://doi.org/10.1109/JPROC.2022.3226481

[4] Simone Bianco, Remi Cadene, Luigi Celona and Paolo Napoletano. 2018. Benchmark Analysis of Representative Deep Neural Network Architectures. IEEE Access, 6, 64270-64277. https://doi.org/10.1109/ACCESS.2018.2877890

[5] Mimoun Lamrini, Mohamed Y. Chkouri and Abdellah Touhafi. 2023. Evaluating the Performance of Pre-Trained Convolutional Neural Network for Audio Classification on Embedded Systems for Anomaly Detection in Smart Cities. Sensors, 23(13), 6227. https://doi.org/10.3390/s23136227

[6] Shuja-Ur-Rehman Baig, Waheed Iqbal, Josep L. Berral, Abdelkarim Erradi and David Carrera. 2019. Adaptive Prediction Models for Data Center Resources Utilization Estimation. In IEEE Transactions on Network and Service Management, 16(4), 1681-1693. https://doi.org/10.1109/TNSM.2019.2932840

[7] Mahfoudh S. Al-Asaly, Mohamed A. Bencherif, Ahmed Alsanad and Mohammad M. Hassan. 2022. A deep learning-based resource usage prediction model for resource provisioning in an automatic cloud computing environment. Neural Computing and Applications, 34(13), 10211-10228. https://doi.org/10.1007/s00521-021-06665-5

[8] Gyeongsik Yang, Changyong Shin, Jeungwan Lee, Yeonho Yoo and Chuck Yoo. 2022. Prediction of the Resource Consumption of Distributed Deep Learning Systems. Proceeding of the ACM on Measurement and Analysis of Computing Systems, 6(2), 1-25. https://doi.org/10.1145/3530895

[9] Vicent S. Marco, Ben Taylor, Zheng Wang, Yehia Elkhatib. 2019. Optimizing Deep Learning Inference on Embedded Systems Through Adaptive Model Selection. ACM Transactions on Embedded Computing Systems, 19(1), 1-18. https://doi.org/10.1145/3371154

[10] Eleni Tsalera, Andreas Papadakis, Maria Samarakou. 2021. Comparison of Pre-Trained CNNs for Audio Classification Using Transfer Learning. Journal of Sensor and Actuator Networks, 10(4), 72. https://doi.org/10.3390/jsan10040072

[11] Eleni Tsalera, Andreas Papadakis, Maria Samarakou. 2021. Novel principal component analysis-based feature selection mechanism for classroom sound classification. Compuatational Intelligence, 37(4), 1827-1843. https://doi.org/10.1111/coin.12468

[12] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Nincent Vanhoucke and Andrew Rabinovich. 2015. Going deeper with convolutions. In Proceeding of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1-9. https://doi.org/10.1109/CVPR.2015.7298594

[13] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally and Kurt Keutzer. 2016. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size. arXiv:1602.07360

[14] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, Jian Sun. 2017. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 6848-6856. https://doi.org/10.48550/arXiv.1707.01083

[15] Eleni Tsalera, Andreas Papadakis, Ioannis Voyiatzis, Maria Samarakou. 2023. CNN-based, contextualized, real-time fire detection in computational resource-constrained environments. Energy Reports 9(9), 247-257. https://doi.org/10.1016/j.egyr.2023.05.260