Neural network scoring for efficient computing

Hugo Waltsburger[†‡], Erwan Libessart[‡], Chengfang Ren[†], Anthony Kolar[‡], Régis Guinvarc'h[†]

[†]SONDRA, CentraleSupélec, Université Paris Saclay, Gif-sur-Yvette
{Hugo.Waltsburger, Chengfang.Ren, Regis.Guinvarch}@centralesupelec.fr
[‡]Université Paris-Saclay, CentraleSupélec, CNRS, Laboratoire de Génie Electrique et Electronique de Paris, 91192, Gif-sur-Yvette, France.
Sorbonne Université, CNRS, Laboratoire de Génie Electrique et Electronique de Paris, 75252, Paris, France
{Hugo.Waltsburger, Erwan.Libessart, Anthony.Kolar}@geeps.centralesupelec.fr

***Abstract*—Much work has been dedicated to estimating and optimizing workloads in high-performance computing (HPC) and deep learning. However, researchers have typically relied on few metrics to assess the efficiency of those techniques. Most notably, the accuracy, the loss of the prediction, and the computational time with regard to GPUs or/and CPUs characteristics. It is rare to see figures for power consumption, partly due to the difficulty of obtaining accurate power readings. In this paper, we introduce a composite score that aims to characterize the trade-off between accuracy and power consumption measured during the inference of neural networks. For this purpose, we present a new open-source tool allowing researchers to consider more metrics: granular power consumption, but also RAM/CPU/GPU utilization, as well as storage, and network input/output (I/O). To our best knowledge, it is the first fit test for neural architectures on hardware architectures. This is made possible thanks to reproducible power efficiency measurements. We applied this procedure to state-of-the-art neural network architectures on miscellaneous hardware. One of the main applications and novelties is the measurement of algorithmic power efficiency. The objective is to allow researchers to grasp their algorithms' efficiencies better. This methodology was developed to explore trade-offs between energy usage and accuracy in neural networks. It is also useful when fitting hardware for a specific task or to compare two architectures more accurately, with architecture exploration in mind.**

I. Introduction

Comparing the performance of different software and hardware architecture in the field of deep neural networks is a challenging endeavor. To establish a benchmark for comparison between various architectures, one must identify relevant metrics comparable across a wide variety of architectures and systems. Those metrics must have an adequate level of precision. However, the literature shows that benchmarks seldom comprise metrics other than the time required for the execution, the accuracy of algorithms, and the number of parameters and necessary multiply-adds [1]–[4]. This approach is relevant in a paradigm where algorithms need to run faster and more accurately, with little regard to the marginal cost of increased performance. However, this approach is not optimal if the objective is to optimize the power consumption of the algorithm, be it out of ecological concern or due to constraints on the hardware available. Such purposes require specific metrics. We advocate that comparing systems should be done using more criteria. The aim is to balance out accuracy and efficiency, specifically in power efficiency. This goal can be achieved by using scores. One of the first scores in the literature, characterizing the trade-off between accuracy and complexity, was introduced in [5] as:

$$Score = \frac{\text{Accuracy}}{\text{Number of parameters}} \tag{1}$$

With the idea of representing the amount of accuracy captured by a single parameter. However, the number of parameters does not always correlate well with the complexity of a network [6]. Especially, convolutional neural networks comprise few parameters but are computationally expensive. Another iteration in neural network scoring, NetScore, taking multiply-accumulates (MACs) into account, was thus introduced in [7]. They propose Netscore, which they define as

$$Netscore = 20\log_{10}\left(\frac{\text{Accuracy}^2}{\sqrt{\text{MACs} \times \text{Parameters}}}\right) \tag{2}$$

Netscore offers a more comprehensive view of the accuracy-efficiency trade-off of a neural network and seems especially relevant for convolutional neural network scoring. To elaborate upon this work, we want to introduce something that better reflects efficiency - especially in terms of power efficiency and adequation between hardware and software.

This paper presents a new score that uses measurements rather than technical information on the network. We believe introducing measurements in neural network scoring is necessary to characterize a neural network's behavior accurately. To obtain the necessary metrics, we introduce Tub.ai [8], a new open-source tool that provides researchers with more diverse and granular metrics on their systems. We believe this methodology can be used in many fields - autonomous vehicles/drones, spaceborne applications, high-performance computing - and not only in AI. Its use is measuring both software and hardware performance. This paper also presents a benchmark of our score computed on various NN architectures during inference. Scoring was realized on various hardware platforms.

This paper will first provide an overview of the motivation behind our new score and the tools we use in section II. We

The final version of this paper is present in the proceedings of the 56th edition of IEEE's International Symposium on Circuits and Systems (ISCAS), 2023.

will then take measurements during inference using Tub.ai in section III, before presenting the scores obtained by several state-of-the-art NN architectures in section IV.

II. Proposed score and tooling

As seen in section I, some methodologies for NN-scoring already exist. However, they are solely computed using the accuracy and parameters from the neural networks' technical sheets (MACs and parameters). When considering power efficiency, the literature shows [6] that power consumption does not scale linearly with either multiply-accumulates or parameters, which are the more common estimators for complexity in deep learning. Moreover, while these scores provide an estimation of the trade-off between accuracy and complexity, they are not measures.

Since our logic is not to maximize accuracy, the best neural network for a specific application will depend on the hardware. We aim to strike an optimum between accuracy and power consumption. Only the inference phase is considered here. What seemed the most important to us was (i) accuracy, (ii) speed of inference, and (iii) power consumption per inference (in mWh). After several tests, we chose the formula in equation (3):

$$Composite\ Score = \frac{\text{Accuracy}^2}{\text{Power consumption per inference}} \tag{3}$$

This formulation, when developed as

$$Score = \frac{\left(\frac{\text{Number of correct inferences}}{\text{Number of inferences}}\right)^2}{\left(\frac{\text{Total Power consumption}}{\text{Number of inferences}}\right)}$$

Simplifies as

$$Score = \frac{\text{Accuracy}}{\text{Average power consumed to obtain one correct inference}} \tag{4}$$

We found that the power consumption per inference was sufficiently dependent on the speed of inference (the slower the inference, the higher the power consumption per inference) not to include the speed of inference as is. Using the rate of inference seemed to favor fast neural networks too much. This version of the score is easily comparable and has meaning from a physics standpoint. As shown in equation (4), it represents the accuracy captured by the average amount of energy consumed to obtain a single correct inference.

As stated above, this score requires measurements made during inference. Therefore, a way of obtaining reliable measurements of energy consumption was needed.

One of the main problems encountered when attempting to measure power consumption is how to measure metrics on our systems precisely. To do this, we created a new tool, Tub.ai. Using various software and protocols, Tub.ai allows us to gather utilization metrics from our machines. Here, these metrics were gathered directly from the CPU[9] and GPU[8] drivers, via RAPL and Nvidia-DCGM measurements.

Once the data sources are set up, we need to aggregate them to exploit said data efficiently. To do this, we use a time series database that will store our data and allow us to query it at will. Over this database, we add a display engine that runs in the browser for easy visualization.

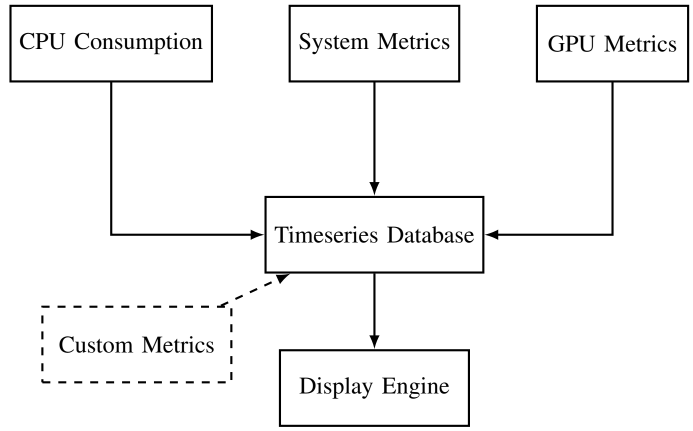

Figure 1: Flowchart of Tub.ai's architecture

Once all the individual bricks have been aggregated, we obtain the architecture displayed in figure 1. Tub.ai only uses open-source applications and will be made open-source for the community to use after publication. Tub.ai is highly modular and can be used to gather an extensive range of data pertaining to the usage of a computer, including custom metrics.

III. Curating data on different NN architectures

In this section, we extract data from our system while several computer vision NN architectures infer on the ILSVRC2012 dataset [14]. This dataset was chosen due to its size and the amount of research available on it. It comprises 1000 classes, with a training dataset containing 1.28 million images, a test dataset containing 100,000 images, and a validation dataset containing 50,000 images. We chose to test several versions of the Inception-ResnetV2 [10], NasNet [11], MobileNetV3 [12] and EfficientNetv2 [13] architectures. These architectures, released between 2016 and 2021, have either achieved then state of the art performance or approached it closely.

The experiment consists of inferences on ILSVRC2012. All models were implemented using Tensorflow 2.10 with Keras, Cuda 11.4, CudNN 8.3, and Python 3.10. We used several different testing machines representing several grades of computers.

- For HPC servers: A single A100 GPU with 40GB of V-RAM, coupled with an AMD EPYC 7742 Processor (128 cores) and 504GB of RAM
- For upper-grade workstations: A Bi-Xeon 5222 workstation with 64GB of RAM and a Quadro RTX 5000 GPU with 16GB of V-RAM.
- For lower grade machines: A laptop comprising an i7-8565 CPU and 8GB of RAM

For the sake of readability, all runs will be given different tags: "A100" for the A100 runs, "Quadro" for the RTX 5000 runs, "Bi-Xeon" for the dual Xeon 5222 runs, and "i7" for the laptop runs. Each neural network's implementation was downloaded with pre-trained weights via the Tensorflow API.

The final version of this paper is present in the proceedings of the 56th edition of IEEE's International Symposium on Circuits and Systems (ISCAS), 2023.

Table I: Comparison of NN architectures based on Tub.ai's metrics, inference on ILSVRC2012 validation dataset, Nvidia A100

Model Name	Year	Validation accuracy	Inference time (s)	Average GPU power consumption (W)	Total power used (Wh)	Average GPU usage (%)	Average GPU memory usage (%)	Average CPU usage (%)
Inception ResNet V2 [10]	2016	80.3%	76	211	4.5	92	99	4.3
NasNet Mobile [11]	2017	74.4%	39	144	1.6	57	99	5.3
NasNet Large [11]	2017	82.5%	204	264	15	95	99	2.7
MobileNetV3 Small [12]	2019	68.1%	15	107	0.5	40	99	10.7
MobileNetV3 Large [12]	2019	75.6%	16	162	0.7	62	99	9.9
EfficientNetV2 S[13]	2021	83.9%	89	253	6.3	88	99	4.7
EfficientNetV2 M[13]	2021	85.1%	212	259	15.3	93	99	3.4
EfficientNetV2 L[13]	2021	85.7%	365	262	26.5	96	99	3.2

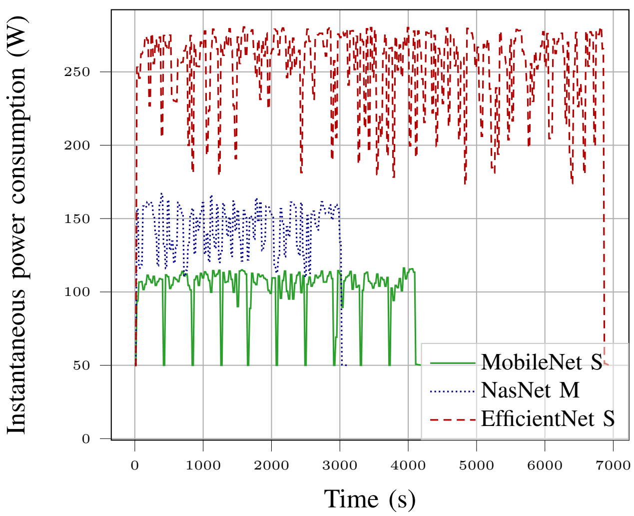

Figure 2: Nvidia A100 power consumption for 3 architectures

Figure 2 presents the instantaneous power consumption measured by Tub.ai for the A100 GPU during several inferences runs on the training dataset using three different architectures. EfficientNet S and NasNet Mobile were run three times in succession with no timeout. MobileNet Small was run ten times with a 30 seconds timeout between each run to provide a reference. It can be seen that the instantaneous power consumption may exhibit significant variations over a single run. It is interesting to observe that NasNet Mobile and MobileNet Small yield a much lower power consumption than EfficientNet. This suggests that the former architectures are bandwidth-bound on this setup, while the computing capabilities of the GPU bind the latter architecture.

In table I, we provide the data obtained on the Nvidia A100 tests. All key indicators were measured during three inferences on the complete ImageNet validation dataset. Using three inferences was deemed robust enough as the results proved very consistent: even when chaining ten successive inferences over 1 million images each, the average deviation to the mean for each run settled below 2%. The most significant observed outlier remained within 5% of the average across all runs. No occurrences of thermal throttling or startup lag were observed.

CPU usage can be considered a good indicator of the model's throughput for GPU runs since the CPU handles the I/O.The CPU performance overhead introduced by Tub.ai is negligible compared to the CPU resources used by the inference during the benchmark: we measured an average load of 0.05%. The GPU's constant 99% memory usage is due to TensorFlow's inner workings, systematically reserving as much V-RAM as possible. When comparing the least and most consuming architectures, we can remark that there is a more than two-fold factor between instantaneously available metrics on this setup. We also observe a fifty-fold total power consumption factor.

IV. Scoring neural networks

This metric aims to be a tool to evaluate the efficiency of different neural network architectures. It can be helpful in constrained systems, where there are limitations on both power consumption and computing power. It can also be used to evaluate the main leverage of any advancement in neural network architectures. The edge of an architecture may be its capacity to leverage Moore's Law and the algorithmic advances made on the various frameworks [1], or it can be a less quantifiable change made in the structure of the architecture that makes it sparser, faster or lighter.

Finally, other researchers may want to employ different coefficients or consider more factors for specific use cases. We hope that our approach and the developed tooling will be valuable for the community to introduce new scores. Using our score, we obtain the ranking presented in table II-and table III.

Table II: Ranking of NN architectures using different scores

Model	Score[5]	Netscore	A100 score	Xeon score
ResNetV2	5	5	4	4
NasNet Mobile	3	3	3	2
NasNet Large	7	7	7	6
MobileNetV3 S	1	1	1	1
MobileNetV3 L	2	2	2	3
EfficientNetV2 S	4	4	5	5
EfficientNetV2 M	6	6	6	7
EfficientNetV2 L	8	8	8	8

The final version of this paper is present in the proceedings of the 56th edition of IEEE's International Symposium on Circuits and Systems (ISCAS), 2023.

Table III: Power efficiency of neural network architectures based on our new metric

Model	Validation accuracy	Power consumption/inference (µWh) A100	Quadro	Bi-Xeon	i7	Score [5]	NetScore	Our Score A100$_{/A100}$	Quadro$_{/A100}$	Bi-Xeon$_{/A100}$	i7$_{/A100}$
ResNetV2 [10]	80.3%	89	290	3,017	3,961	1.44	47	7.2$_{/1}$	2.2$_{/3.3}$	0.21$_{/34}$	0.16$_{/45}$
NasNet Mobile [11]	74.4%	31	63	302	580	14	70.1	17.9$_{/1}$	8.8$_{/2}$	1.84$_{/8.9}$	0.95$_{/19}$
NasNet Large [11]	82.5%	299	792	4,792	9,766	0.9	42.4	2.3$_{/1}$	0.86$_{/2.6}$	0.14$_{/16}$	0.07$_{/33}$
MobileNetV3 S [12]	68.1%	9	16	143	120	**28.4**	**83.1**	**51.5**$_{/1}$	**29.0**$_{/1.8}$	**3.24**$_{/16}$	**3.87**$_{/13}$
MobileNetV3 L [12]	75.6%	14	29	334	293	14	74.5	40.8$_{/1}$	19.5$_{/2.1}$	1.70$_{/24}$	1.95$_{/21}$
EfficientNetV2 S [13]	83.9%	125	286	3,340	4,195	3.9	54.2	5.6$_{/1}$	2.46$_{/2.3}$	0.21$_{/27}$	0.17$_{/34}$
EfficientNetV2 M [13]	85.1%	305	877	8,794	12,236	1.6	46.0	2.4$_{/1}$	0.83$_{/2.9}$	0.08$_{/29}$	0.06$_{/40}$
EfficientNetV2 L [13]	85.7%	530	1,539	16,698	24,346	0.7	39	1.4$_{/1}$	0.48$_{/2.9}$	0.04$_{/32}$	0.03$_{/46}$

First, our ranking seems generally consistent with the scores obtained using [5] and [7]. Second, we observe a difference in the hierarchy depending on the setup: on the Bi-Xeon platform, NasNet Mobile outperforms MobileNetV3 Large, and NasNet Large outperforms EfficientNetV2 Medium. Third, by observing the factor of proportionality between our scores and the A100 score (in small fonts next to each of our scores), we can see that some architectures (Inception ResNetV2, EfficientNetv2 Large) are much more penalized than other architectures (MobileNet, NasNet) when changing setup. This result confirms the value of taking measurements on neural network inferences rather than relying on platform-independent metrics. We hypothesize that this difference is due to a balance between computing and bandwidth requirements making some architectures less hardware constrained.

Despite having the highest average power consumption of all platforms, the A100 has the highest score across all architectures. While higher-grade hardware has higher idle and in-charge energy usage, it usually exhibits a lower power consumption per inference due to increased inference speed. Notwithstanding, while the bi-Xeon setup outperforms the i7 on most architectures, the i7 fares better on MobileNets. This result emphasizes the need to study the envisioned application of a given neural network to extract its maximum performance. It also calls for more work on the architecture of neural networks to identify the limiting factor across different hardware settings and obtain the best hardware-software fit.

MobileNetv3 Small has a net lead over all other networks and ranks first on all platforms, which seems reasonable since the MobileNet architecture was created to be *"a class of efficient models [...] for mobile and embedded vision applications"* [15]. Some of the more recent models perform worse than older models in terms of score. It seems likely that, in the future, the discrepancy between networks made for optimum efficiency and those created solely for performance will drift apart further. This observation leads to new approaches, with some recent architectures [16] [17] seeking to optimize training/inference speed rather than accuracy or parameter efficiency and advocating for better architecture-algorithm adequation.

We hope that it will be more commonplace in the future that some benchmarks rank neural network architectures using not only the validation accuracy but also more metrics, especially power consumption measured on both training and inference.

V. Conclusion and future works

In this paper, we attempted to create a score based not on a technical datasheet but on measurements made on a platform. The idea is to measure the algorithm-architecture adequation between specific NNs and hardware. To obtain the required metrics to compute our score, we created Tub.ai.

Tub.ai provides researchers with various metrics that can be leveraged to create meaningful comparisons between different software implementations and architectures. Tub.ai is open source, induces very little overhead in terms of performance, and is easy to use. This approach significantly benefits system developers who have to develop or optimize algorithms for specific hardware.

Our deep neural network architecture benchmarks estimate how well they are fitted to be run on different platforms. The score values we have obtained across several architectures show that there can be significant discrepancies between platforms for the same neural network - which proves the interest of scoring using measurements. To our best knowledge, this is the first time power efficiency measurements have been included in the performance evaluation of neural network architectures. In a context of growing concern for the ecological impact of machine learning and high-performance computing in general, it is a step forward in the field of HPC where the main focus is not solely the algorithm's or architecture's performance but also their power efficiency.

In the future, we plan on furthering the work we have done on scoring by exploring other hardware, architectures, and frameworks. Using identical neural network architectures implemented on different frameworks and running them on various test hardware, we would like to evaluate how well available deep learning frameworks can exploit the hardware's capabilities. An examination of the power efficiency of specialized neural processing units with regard to our score is planned. Similar undertakings on other datasets are considered. We also plan on updating Tub.ai and making it more precise and easier to use. We would be especially interested in seeing the results of neural network architecture search using this new score as the optimizing criterion.

The final version of this paper is present in the proceedings of the 56th edition of IEEE's International Symposium on Circuits and Systems (ISCAS), 2023.

REFERENCES

[1] D. Hernandez and T. B. Brown, "Measuring the algorithmic efficiency of neural networks," *arXiv preprint arXiv:2005.04305*, 2020.

[2] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[5] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv preprint arXiv:1605.07678*, 2016.

[6] H. Cai, J. Lin, Y. Lin, *et al.*, "Enable deep learning on mobile devices: Methods, systems, and applications," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 27, no. 3, pp. 1–50, 2022.

[7] A. Wong, "Netscore: Towards universal metrics for large-scale performance analysis of deep neural networks for practical on-device edge usage," in *Image Analysis and Recognition: 16th International Conference, ICIAR 2019, Waterloo, ON, Canada, August 27–29, 2019, Proceedings, Part II*, Berlin, Heidelberg: Springer-Verlag, 2019, pp. 15–26.

[8] H. Waltsburger, E. Libessart, C. Ren, A. Kolar, and R. Guinvarc'h. "Tub.ai - public repository." (2023), [Online]. Available: https://gitlab-research.centralesupelec.fr/hugo.waltsburger/tub.ai.

[9] H. David, E. Gorbatov, U. R. Hanebutte, R. Khanna, and C. Le, "Rapl: Memory power estimation and capping," in *2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*, 2010, pp. 189–194. DOI: 10.1145/1840845.1840883.

[10] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-V4, Inception-ResNet and the impact of residual connections on learning," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[11] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[12] A. Howard, M. Sandler, G. Chu, *et al.*, "Searching for MobileNetV3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019.

[13] M. Tan and Q. Le, "EfficientNetV2: Smaller models and faster training," in *International Conference on Machine Learning*, PMLR, 2021, pp. 10 096–10 106.

[14] O. Russakovsky, J. Deng, H. Su, *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. DOI: 10.1007/s11263-015-0816-y.

[15] A. G. Howard, M. Zhu, B. Chen, *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[16] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 428–10 436.

[17] S. Li, M. Tan, R. Pang, *et al.*, "Searching for fast model families on datacenter accelerators," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 8085–8095.

The final version of this paper is present in the proceedings of the 56th edition of IEEE's International Symposium on Circuits and Systems (ISCAS), 2023.