Energy Consumption of Neural Networks on NVIDIA Edge Boards: an Empirical Model

Seyyidahmed Lahmer, Aria Khoshsirat, Michele Rossi, Andrea Zanella
Department of Information Engineering
University of Padova
Padova, Italy
{firstname.lastname}@unipd.it

Abstract—**Recently, there has been a trend of shifting the execution of deep learning inference tasks toward the edge of the network, closer to the user, to reduce latency and preserve data privacy. At the same time, growing interest is being devoted to the energetic sustainability of machine learning. At the intersection of these trends, in this paper we focus on the energetic characterization of machine learning at the edge, which is attracting increasing attention. Unfortunately, calculating the energy consumption of a given neural network during inference is complicated by the heterogeneity of the possible underlying hardware implementation. In this work, we aim at profiling the energetic consumption of inference tasks for some modern edge nodes by deriving simple but accurate models. To this end, we performed a large number of experiments to collect the energy consumption of fully connected and convolutional layers on two well-known edge boards by NVIDIA, namely, Jetson TX2 and Xavier. From these experimental measurements, we have then distilled a simple and practical model that can provide an estimate of the energy consumption of a certain inference task on these edge computers. We believe that this model can prove useful in many contexts as, for instance, to guide the search for efficient neural network architectures, as a heuristic in neural network pruning, to find energy-efficient offloading strategies in a split computing context, or to evaluate and compare the energy performance of deep neural network architectures.**

Index Terms—**Energy consumption, Deep Neural Networks, Edge Computing, Inference**

I. Introduction

Machine learning is being used in many applications, exploiting the abundance of data in the modern era and delivering state-of-the-art performance on a huge number of tasks. For new emerging mobile applications, the traditional way of running inference tasks in cloud computing facilities and sending back the predictions to the end users is not always feasible because of the need for preserving privacy and ensuring low latency. On the other hand, shifting the inference towards the end devices presents its challenges, due to the limited resources available on these devices. To tackle these limitations, computation offloading from resource-limited end users to more powerful edge servers is being advocated as a promising method to schedule and execute user-generated tasks [1].

In fact, *Edge Computing* not only can provide faster online computations, closer to the end users, but it can also exploit the smart distribution and scheduling of computations to benefit from renewable energy resources (RERs), so as to reduce the carbon footprint of computing technology [2]. Besides improving throughput and latency, the energy efficiency of edge networks has gained much attention lately. For example, reference [3] studies the whole network's energy consumption, including access points, edge servers, and user equipment for a computation offloading scenario. According to this paper, the more we push the computation from cloud servers to the network's edge, the more crucial it becomes to consider the energy consumption of the models that are being exploited by end user applications.

Although deep learning (DL) [4] has been known for its great success in terms of accurate predictions in a wide variety of tasks, energy and memory requirements of modern DL architectures may make the use of large deep neural networks for edge computing challenging. Split computing techniques have been proposed to tackle this problem. They basically focus on splitting a neural network at different candidate points, and performing early exit at such candidate points to obtain a trade-off between computing effort and quality of the result. This facilitates the deployment of deep networks at the network's edge, see, e.g., [5]. Further, designing energy efficient neural networks that have the same prediction accuracy as their more power hungry versions is receiving much attention from the research community [6], [7].

Overall, current developments are evolving along two main axes: (i) providing online and energy efficient schedulers for edge computing networks that allow end users to offload their tasks, e.g., [1], [2], and (ii) devising new energy efficient DL architectures, also entailing but not limited to the split computing paradigm, e.g., [5]–[7]. We advocate that proper designs along both axes would greatly benefit from accurate energy consumption models of DL, especially tailored to modern edge computing hardware. These models are largely missing in the literature and are the objective of the present work.

In most of the existing literature on edge task scheduling, the energy cost models that were used for predicting the energy consumption mainly used the number of CPU cycles required to perform the tasks [8] or the amount of workload that a task produces [9], using simple equations that proportionally depend on the squared CPU frequency or on the workload. While these models were very valuable to derive initial theories and results on scheduling algorithms, they may not suit

ISBN 978-3-903176-49-2 © 2022 IFIP

well with the parallelizable computations on modern multi-core processing unit architectures. In fact, an accurate energy consumption estimation tool requires one to take into account the architecture of the host device, the different parameters in the neural network model that can exploit the parallel hardware architectures, and the exact number of operations a neural network requires for inference.

In this paper, we propose an experimentally validated and simple energy consumption model for neural networks on recent NVIDIA Jetson edge computers. The model allows one to estimate the energy drained by performing inference tasks on DL models composed of fully connected and convolutional layers, without having to perform online measurements of the energy drained. As we elaborate in the following, the main indicator for the energy consumption is the total number of multiply and accumulate (MAC) operations that are performed, as expected. Based on this number, for a convolutional layer, the energy consumption shows a multi-modal behavior governed by the number of kernels that are exploited. The derived empirical model is fully described by two hardware dependent parameters, which are here provided for Jetson TX2 and Xavier NX boards from NVIDIA. The model fitting for a simpler fully connected layer follows a similar rationale, but only requires a single parameter and shows a single slope in the MAC *vs* energy plot.

The remainder of this paper is organized as follows: The related work is briefly commented in Section II. In Section III we present the experiment setup and configurations. The observations and discussions, in addition to an energy estimation model are provided in Section IV. Finally, conclusions and future research lines are discussed in Section V.

II. Related work

Profiling the power/energy consumption of running *Neural Network* (NN)s on low-power edge devices has gained an increasing attention in recent years. In [10], the authors measured the power consumption of an entire NN as well as single NN layers on an NVIDIA Jetson Nano. A framework that predicts the energy consumption of CNNs on the Jetson TX1 based on real measurements has been proposed in [11]. This work is however still very preliminary, as it just presents the general measurement setup/methodology and some limited results. For the Jetson TX2 device, in [12] the authors have reported the power consumption of GPU and CPU, the memory usage and the time of executing the test phase on a fixed small *Convolutional Neural Network* (CNN) architecture. Although the results in this paper are measured from real hardware, no analytical model is provided to gauge the energy consumption of the edge board from the neural network parameters.

In a research paper more similar to our present work, but based on simulations instead of real measurements [13], the authors have provided an energy estimation tool for different types of neural network layers. They have shown that the energy consumption is not always proportional to the number of computations or parameters involved in a layer. Our results somehow confirm these observations, since the pure number of operations, *per se*, is not sufficient to characterize the energy consumption of the boards. Nonetheless, with a careful and systematic analysis of the collected measurements, we were able to identify the effect of the different computational model parameters on the energy consumption of a single inference stage and, hence, define a model that captures reasonably well the experimental behavior of the computing boards.

To the best of our knowledge, this is the first work to explore the real-world effect of choosing different configurations of a NN layer on the energy consumption of two NVIDIA Jetson edge devices (TX2 and Xavier NX), providing a parameterized analytical energy estimation model based on empirical measurements. Our model allows estimating the energy consumption of any custom set of layer configurations in common feed-forward deep neural networks.

III. Experimental setup

We experimentally characterize the energy consumption of two energy-efficient embedded computing devices from NVIDIA, namely, Jetson TX2, and Jetson Xavier NX. These two edge computers are currently being used in several fields such as manufacturing, agriculture, retail, life sciences, etc. For instance, an image processing algorithm for thermal events has been recently proposed for the Jetson TX2 [14]. The configurations of both devices are shown in Table I (Jetson TX2) and II (Jetson Xavier NX).

TABLE I: NVIDIA **Jetson TX2** configurations

CPU	Quad-Core ARM Cortex-A57 @ 2 GHz + Dual-Core NVIDIA Denver2 @ 2 GHz
GPU	NVIDIA Pascal 256 CUDA cores @ 1300 MHz
Memory	8 GB 128-bit LPDDR4 @ 1866 Mhz, 59.7 GB/s
Performance	1.3 TFLOPS

TABLE II: NVIDIA **Jetson Xavier NX** configurations

CPU	6-core NVIDIA Carmel ARM®v8.2 64-bit CPU 6 MB L2 + 4 MB L3
GPU	384-core NVIDIA Volta™ GPU with 48 Tensor Cores
Memory	8 GB 128-bit LPDDR4x 59.7 GB/s
Performance	21 TFLOPS

To assess the energy profile of these edge computers, we measure the timing and energy figures of neural network architectures, focusing on one single layer of the whole NN architecture. In fact, as demonstrated in [13], and also independently verified by us, the energy consumption of two neural network layers L_1, L_2 that are executed in sequence adds up, i.e., if their energy consumption is respectively $E(L_1) = E_1$ and $E(L_2) = E_2$, then sequentially using these two layers into a single model results in a total energy consumption of $E(L_1, L_2) \simeq E_1 + E_2$, where the approximation accounts for the measurement noise and the intrinsic variability of the energy consumption of each single layer (as it will be seen later on in this paper). We hence focus our analysis on two widely utilized layer types, namely *fully connected* and *convolutional*, as better described in the following.

Fully Connected layer. A fully connected layer consists of a bipartite set of input and output neurons, with each input neuron being connected to all the output neurons through weighted links. The output neurons apply a non-linear transformation to the weighted sum of the input vector, producing the corresponding output value. The following variables are hence used to describe a fully connected layer:

- i_size: Input feature map size, i.e., number of input neurons;
- o_size: Output feature maps size, i.e., number of output neurons.

We refer to the *Computational Load* of a fully connected layer (CLF) L_i as the product of the number of input and output features of the layer, i.e., the value

$$\mathrm{CLF}(L_i) = \text{i\_size} \times \text{o\_size}. \tag{1}$$

Note, that CLF corresponds to the number of elements in the weight matrix and, hence, is proportional to the number of multiplications and additions that are performed as the layer is executed, i.e., to propagate the input to the output section.

Convolutional layer. We consider a generic convolutional layer defined by a multidimensional matrix of input values, with size $w \times h \times d$, and a set of kernel functions, each defined by a square matrix of real values of size $k \times k \times d$. Here, d is referred to as the *depth* and should match the depth of the input feature map. Each kernel shifts along the input matrix with a step defined by another parameter called *stride*. For each position, the dot product between the kernel and the corresponding elements of the input matrix is computed, and the results are then summed together to return one point of the output matrix. Each kernel generates one output map.

The following variables are then used to describe a convolutional layer:

- i_size: Input feature map size (i.e., $w = h$);
- ifm: Number of input feature maps (i.e., d);
- ofm: Number of output feature maps (i.e., the number of kernel functions),
- ksize Kernel size parameter (i.e., k),
- stride: Stride parameter (i.e., the sliding step of the kernel over the input matrix).

We define the *Computational Load* involved for a **single kernel** (KCLC) via the number of multiply-add operations, also referred to as Multiply–accumulate (MAC) operations, required to compute the convolution of the input maps with a single kernel (i.e., to obtain each one of the output maps). For a two dimensional convolutional layer L_i, neglecting the padding, it is obtained as follows

$$\begin{aligned}\mathrm{KCLC}(L_i) &= \left(\frac{\text{i\_size}_i - \text{ksize}_i}{\text{stride}_i} + 1\right)^2 \cdot \text{ifm}_i \cdot \text{ksize}_i^2 \\ &\simeq \left(\frac{\text{i\_size}_i}{\text{stride}_i}\right)^2 \cdot \text{ifm}_i \cdot \text{ksize}_i^2, \end{aligned} \tag{2}$$

where the approximation follows when the input size is much larger than the kernel size, which is typical in most practical cases. We then define the Computational Load for the whole convolutional layer (CLC) as Eq. (3),

$$\mathrm{CLC(L_i)} = \mathrm{KCLC}(L_i) \cdot \mathrm{ofm_i}. \tag{3}$$

The variables mentioned above are varied to generate different layers' configurations for the experiments. For each configuration, 50 inference runs are performed using randomly generated non-zero inputs. Moreover, for each inference operation, the time is split into timeslots of the same duration $\delta = 0.1$ ms and the power of the edge board is obtained from the onboard sensors at the end of each timeslot. The average energy consumed by the board over a time period of T seconds is estimated as,

$$\mathrm{E_{board}(T)} \simeq \mathrm{T} \cdot \mathrm{AverageBoardPower(T)}. \tag{4}$$

IV. Energy Characterization

A. Power consumption

Fig. 1 reports the empirical histograms of the power consumption of the two boards when performing the inference tasks with different configurations of the convolutional or fully connected layers parameters. As shown in these two figures, the power consumption follows a normal distribution for the different inference tasks, with the mean placed close to the amount of power consumption of the device when the CPU is at 100 percent workload. This observation of the frequency of power consumption is especially relevant when the inference devices use renewable power sources, such as solar panels, that cannot provide more than a specific peak or mean power for a long period of time (due to the intermittent nature of renewable energy).

B. Energy consumption when varying the model parameters

Experimentally, we can understand the impact of the different parameters of each layer's configuration on the energy consumption; we came up with the following key observations:

- As it will be further analyzed in the next Section IV-C, the average energy drained for a convolutional layer L_i is accurately approximated by $E_{\mathrm{conv2d}}(L_i) \simeq \mathrm{CLC}(L_i) \times H(\mathrm{ofm}_i)$, where $H(\cdot)$ is a non-linear function (to be specified shortly).
- For a convolutional layer, the average energy consumption grows linearly with respect to CLC when varying the i_size, ksize, ifm, or the stride. Fig. 2 depicts this behavior: from the Eq. (3), the CLC decreases polynomially with respect to an increasing stride, and so does the average energy; the CLC increases polynomially with increasing ksize and i_size, and so does the average energy. The CL increases linearly with an increasing ifm, and so does the average energy.
- Moreover, still for a convolutional layer L_i, the average energy consumption $E_{\mathrm{conv2d}}(L_i)$ does not grow linearly with CLC when changing the ofm parameter. From Fig. 3, and with both edge computing boards, we notice that the relationship between average energy and CLC can be interpolated through a linear function for a fixed

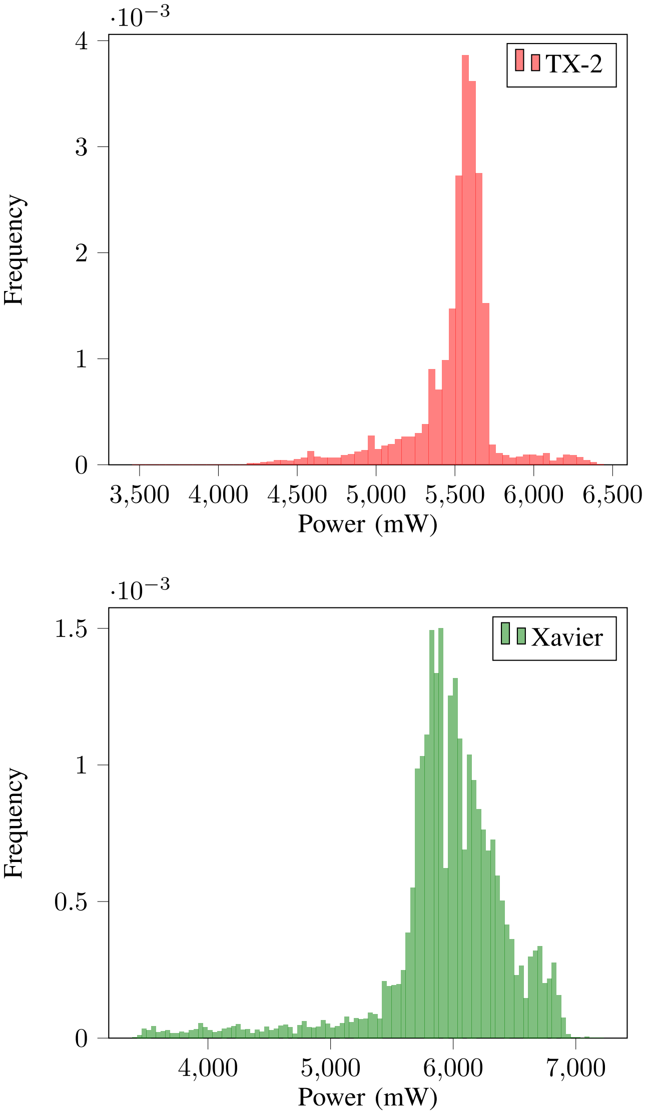


Fig. 1: Distribution of the average board power consumption on Jetson TX 2 and Xavier NX for inference of many different neural network configurations.

ofm value. Also, we observe that the slope of this linear approximation changes (decreases) by changing the ofm (increasing). Function $H(\text{ofm})$ is here introduced to model this change of slope; it is also remarked that there is a noise term in the measurements, whose variance increases with an increasing CLC parameter.

- For a fully connected layer, the average energy consumption grows linearly as a function of CLF (constant slope model), see Fig. 4.

C. Energy modeling

To find a suitable shape for $H(\text{ofm})$, which maps an ofm value onto a slope in the MAC-vs-energy plane, we generated additional data. The set of N generated data pairs is denoted by (x_i, y_i) where $i = 1, \ldots, N$, x_i defines an ofm value, and y_i defines the corresponding slope $H(x_i)$; Fig. 5 shows the function $H(\cdot)$, which approximately takes the form

$$H(\text{ofm}_i) \simeq a_c \times \frac{1}{\text{ofm}_i} + b_c. \tag{5}$$

With this dataset, we use Mean Square Error (MSE) minimization as described in Eq. (6) to estimate the parameters a_c, and b_c, which results in the red fitting curve in Fig. 5.

$$\begin{aligned}
\text{MSE} &= \frac{1}{N}\sum_{i=0}^{N-1}(H(x_i) - y_i)^2 \\
&= \frac{1}{N}\sum_{i=0}^{N-1}\left(a_c \times \frac{1}{x_i} + b_c - y_i\right)^2 \\
\frac{\partial \text{MSE}}{\partial a_c} = 0 &\iff \frac{\partial \sum_{i=0}^{N-1}\left(a_c \times \frac{1}{x_i} + b_c - y_i\right)^2}{\partial a_c} = 0 \\
&\iff a_c \sum_{i=0}^{N-1}\frac{1}{x_i^2} + b_c \sum_{i=0}^{N-1}\frac{1}{x_i} = \sum_{i=0}^{N-1}\frac{y_i}{x_i} \\
\frac{\partial \text{MSE}}{\partial b_c} = 0 &\iff b_c = \frac{1}{N}\left[\sum_{i=0}^{N-1} y_i - a_c \sum_{i=0}^{N-1}\frac{1}{x_i}\right] \\
a_c &= \frac{\sum_{i=0}^{N-1}\frac{y_i}{x_i} - \frac{1}{N}\sum_{i=0}^{N-1}\frac{1}{x_i}\sum_{i=0}^{N-1} y_i}{\sum_{i=0}^{N-1}\frac{1}{x_i^2} - \frac{1}{N}\left(\sum_{i=0}^{N-1}\frac{1}{x_i}\right)^2}.
\end{aligned} \tag{6}$$

With the previous key results and observations, we define the following model describing the average energy consumption for a convolutional layer, via Eq. (7):

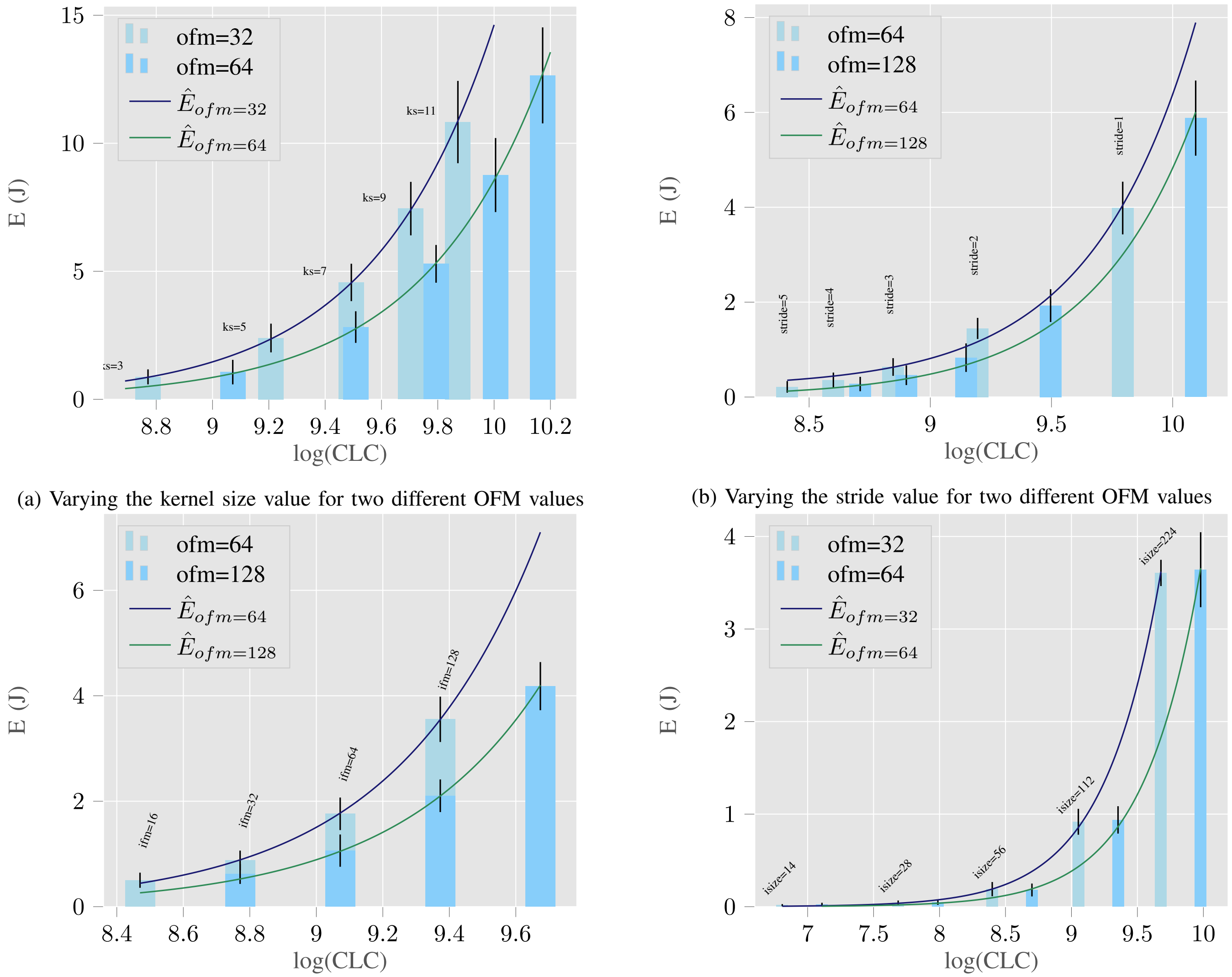
$$\begin{aligned}
E_{\text{conv2d}}(L_i) &= \text{CLC}(L_i) \times H(\text{ofm}_i) \\
&= \text{KCLC}(L_i) \times \text{ofm}_i \times \left(\frac{a_c}{\text{ofm}_i} + b_c\right) \\
&= \text{KCLC}(L_i) \times (a_c + b_c \times \text{ofm}_i).
\end{aligned} \tag{7}$$

For a fully-connected layer, we describe the average energy consumption through Eq. (8) here below. The same procedure is followed to obtain the slope parameter a_f, with x_i denoting the CLF, and y_i denoting the corresponding average energy drained. The b_f coefficient is set to zero, as with a zero CLF there is no energy expenditure.

$$E_{\text{fc}}(L_i) = \text{CLF}(L_i) \times a_f. \tag{8}$$

Given a general CNN architecture description, the average energy expenditure of Conv2D and FC layers is gauged through the CLF and the platform-specific parameters (a_c, b_c) (Conv2D layers) and (a_f) (FC layers), which we obtained empirically (see Table. III).

Furthermore, for a feed-forward NN architecture with L layers, one can estimate the average energy consumption of the whole NN on the considered edge computing boards via Algorithm 1.



(a) Varying the kernel size value for two different OFM values

(b) Varying the stride value for two different OFM values

(c) Varying the ifm value for two different OFM values

(d) Varying the i_size value for two different OFM values

Fig. 2: Convolutional layer: exemplary demonstration of the linear relationship between CLC and the stride, ksize, ifm, and i_size parameters; vertical bars show 99% confidence intervals. Continuous lines are obtained from our approximated model to gauge the average energy as a function of CLC ($\hat{E}$). The x-axis is represented in a $\log 10$ scale.

TABLE III: Empirical Parameters

Parameter	**Value**
a_c **(TX2)**	2.6727e-08
b_c **(TX2)**	1.21334e-10
a_c **(Xavier NX)**	2.8674e-08
b_c **(Xavier NX)**	4.7639e-10
a_f **(Xavier NX)**	6.2454e-09

V. Concluding Remarks

In this work, we have analyzed the energy consumption of NNs on two NVIDIA edge boards, based on readings from the power sensors included in these devices. We have also investigated the effect of different parameters of convolutional and fully-connected layers on energy consumption during inference on CPU. Moreover, we have observed that the boards' peak and average power requirement when using CPU is less than when using GPU. This makes doing inference on CPU more inviting for limited power setups.

The energy estimation model provided in this work, which is backed-up with actual energy measurements on the edge devices, can help understand the effect of parameter choices on energy consumption for efficient development of new neural network architectures. It can also be used as a metric to optimise the scheduling of tasks on the network's edge, when energy efficiency is an important consideration.

In our future work, we plan to extend the experiments to additional edge devices and to other power profile settings of the edge boards, to study how inference can be customized in respect to power, latency and energy. Other interesting research directions include the investigation of effect of NN parameters

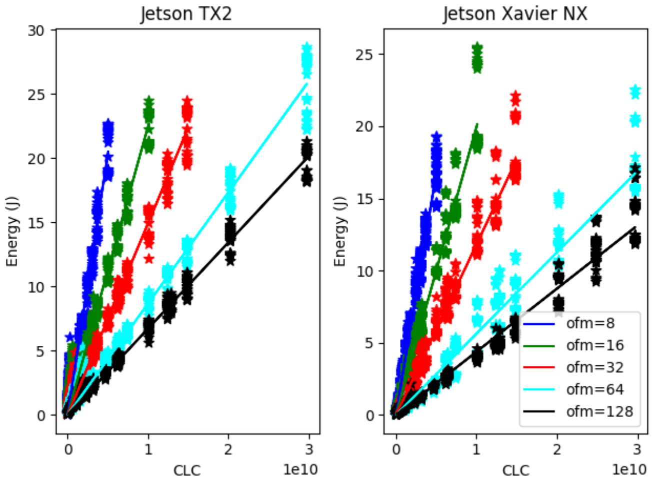


Fig. 3: The relationship between the computational load and the average energy growth for convolutional layer for a different values of ofm.

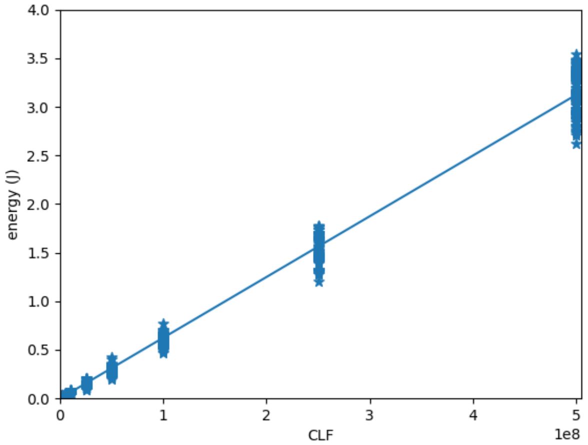


Fig. 4: The relationship between the average energy consumption and the computational load for a fully connected layer.

when inference is performed on GPU and measuring the energy consumption for other layer types, e.g., long-short term memory (LSTM).

Acknowledgment

This work has been supported by the EU H2020 MSCA ITN project Greenedge (grant no. 953775) and by MIUR (Italian Ministry for Education and Research) under the "Departments of Excellence" initiative (Law 232/2016).

References

[1] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet of Things Journal*, vol. 8, no. 5, pp. 3774–3785, 2021.

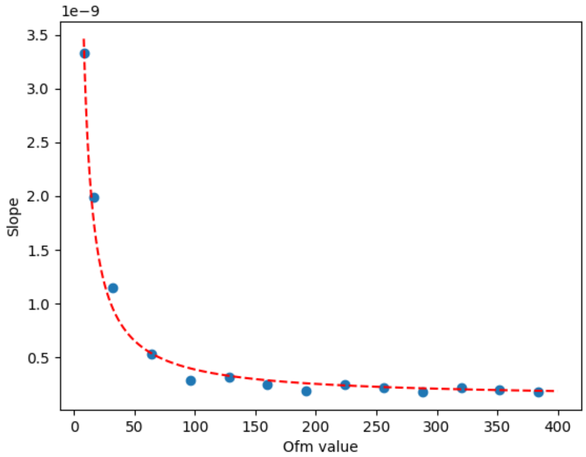


Fig. 5: The behaviour slope changes with respect to the ofm value represented by the blue points, with the corresponding interpolation represented by the red function.

Algorithm 1: Energy Estimator

```
Input:
$\mathrm{nnArch} = [(\mathrm{t}, \mathrm{i/o\_size_i}, \mathrm{ifm_i}, \mathrm{ofm_i}, \mathrm{ksize_i}, \mathrm{stride_i})]_{\mathrm{i=0}}^{\mathrm{L-1}}$
$\mathrm{a_{conv}}, \mathrm{b_{conv}}, \mathrm{a_{fc}}$ : Float/* Platform specific
    parameters                                    */
Output: TotalEnergy
TotalEnergy = 0
for $\mathrm{i} \in [0..\mathrm{L}-1]$ do
    if nnArch[i].t is CONV2D then
        TotalEnergy =
         $\mathrm{TotalEnergy} + \mathrm{E_{conv2d}}(\mathrm{nnArch[i]}, \mathrm{a_c}, \mathrm{b_c})$
    else if nnArch[i].t is FC then
        TotalEnergy =
         $\mathrm{TotalEnergy} + \mathrm{E_{fc}}(\mathrm{nnArch[i]}, \mathrm{a_f})$
```

[2] G. Perin, M. Berno, T. Erseghe, and M. Rossi, "Towards sustainable edge computing through renewable energy resources and online, distributed and predictive scheduling," *IEEE Transactions on Network and Service Management*, vol. 19, no. 1, pp. 306–321, 2022.

[3] M. Merluzzi, N. di Pietro, P. Di Lorenzo, E. C. Strinati, and S. Barbarossa, "Discontinuous computation offloading for energy-efficient mobile edge computing," *IEEE Transactions on Green Communications and Networking*, pp. 1–1, 2021.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[5] Y. Matsubara, M. Levorato, and F. Restuccia, "Split computing and early exiting for deep learning applications: Survey and research challenges," 2021. [Online]. Available: https://arxiv.org/abs/2103.04505

[6] R. Ding, Z. Liu, R. D. S. Blanton, and D. Marculescu, "Quantized deep neural networks for energy efficient hardware-based inference," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2018, pp. 1–8.

[7] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, 2021.

[8] K. De Vogeleer, G. Memmi, P. Jouvelot, and F. Coelho, "The energy/frequency convexity rule: Modeling and experimental validation on mobile devices," in *Parallel Processing and Applied Mathematics*.

Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 793–803.

[9] R. Bertran, M. Gonzalez, X. Martorell, N. Navarro, and E. Ayguade, “A systematic methodology to generate decomposable and responsive power models for cmps,” *IEEE Transactions on Computers*, vol. 62, no. 7, pp. 1289–1302, 2013.

[10] S. Holly, A. Wendt, and M. Lechner, “Profiling energy consumption of deep neural networks on nvidia jetson nano,” in *2020 11th International Green and Sustainable Computing Workshops (IGSC)*, 2020, pp. 1–6.

[11] C. F. Rodrigues, G. Riley, and M. Luján, “Fine-grained energy profiling for deep convolutional neural networks on the jetson tx1,” in *2017 IEEE International Symposium on Workload Characterization (IISWC)*, 2017, pp. 114–115.

[12] A. A. Süzen, B. Duman, and B. Şen, “Benchmark analysis of jetson tx2, jetson nano and raspberry pi using deep-cnn,” in *2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, 2020, pp. 1–5.

[13] T.-J. Yang, Y.-H. Chen, J. Emer, and V. Sze, “A method to estimate the energy consumption of deep neural networks,” in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, 2017, pp. 1916–1920.

[14] B. Jabłoński, D. Makowski, and P. Perek, “Implementation of thermal event image processing algorithms on nvidia tegra jetson tx2 embedded system-on-a-chip,” *Energies*, vol. 14, no. 15, 2021.