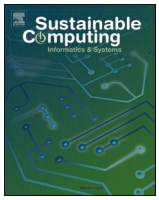
Sustainable Computing: Informatics and Systems 31 (2021) 100580

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom



New universal sustainability metrics to assess edge intelligence

Nicola Lenherr [a], René Pawlitzek [b], Bruno Michel [a,*]

[a] Smart System Integration, IBM Research – Europe, CH-8803, Rueschlikon, Switzerland
[b] Interstaatliche Hochschule für Technik Buchs NTB, CH-9471, Buchs, Switzerland

ARTICLE INFO

Keywords:
Deep learning
Sustainability
Edge computing
Green AI
Accuracy
Accelerator

ABSTRACT

The single recent focus on deep learning accuracy ignores economic, and environmental cost. Progress towards Green AI is hindered by lack of universal metrics that equally reward accuracy and cost and can help to improve all deep learning algorithms and platforms. We define recognition and training efficiency as new universal metrics to assess deep learning sustainability and compare them to similar, less universal metrics. They are based on energy consumption measurements, on deep learning inference, on recognition gradients, and on number of classes and thus universally balance accuracy, complexity and energy consumption. Well-designed edge accelerators improve recognition and training efficiencies compared to cloud CPUs and GPUs due to reduced communication overhead. Cradle to grave sustainability of edge intelligence models and platforms is assessed with novel deep learning lifecycle efficiency and life cycle recognition efficiency metrics that include the number of times models are used. Artificial and natural intelligence efficiencies are compared leading to insights on deep learning scalability.

1. Introduction

Artificial Intelligence (AI) and machine learning (ML) have revolutionized how industries address data deluge, and rapid courses of action. AI involves multiple components that have to work together to provide actionable knowledge. Data is fused, structured, accumulated, and converted to information for neural networks, which extract patterns and predict events [1]. These components run on computers that changed from exponential performance growth to stagnation: Moore's law, clock frequency, core counts, and instructions per clock only show marginal progress. This triggered novel computing architectures with circuit specialization where often-used functions are accelerated with a better balance between performance and flexibility. Understanding their benefits is key for development of embedded AI constrained in size, weight, and power [2,3].

Since 2012, AI reported much better accuracy in compute-intensive deep learning (DL) models but training cost increased by 300,000x which is not sustainable. This trend is driven by leader-boards, which report accuracy but omit cost or efficiency and thus ignore sustainability. Schwartz [4] define Red AI, show that this research was over-dominant, and ask for a re-balance towards Green AI. The key to accelerate the development towards Green AI is to make efficiency an inherent evaluation criterion for research alongside accuracy and related measures. New models also improved natural language processing (NLP) accuracy at the expense of computational resources and energy consumption. Extensive cost of NLP models too urges for improvement [5,6]. The high energy consumption of convolutional neural networks (CNNs) limits the platforms where they can be deployed on. While training is only done rarely, inference is invoked thousands of times on millions of devices. Therefore, testing energy on resource-constrained systems is critical. Since architecture optimization reduces energy demand up to 40-fold, optimizing energy-efficiency with only minimally reducing performance is important for large-scale deployment and Green AI [7].

Commercial low-power accelerators are crucial for embedded ML since inference accounts for ~90 % cost. Edge TPU (TPU) [8] and Movidius Neural Compute stick (NCS2) [9] were compared using an USB multimeter (*Tab. I*) with giga operations per second (GOPS), power (W), and GOPS/W along with average model load time and image inference time. Edge TPU and NCS2 have lower power consumption and higher model load times than the i9. However, image inference times are the same, though the NCS2 is slower. Edge TPU GOPS/W are similar, while the measured GOPS/W is lower than the GOPS/W for the NCS2 [1]. From these measurements it becomes clear that edge accelerators showed a huge recent progress which allowed them to closely approach deep learning accuracies of Cloud systems (Table 1).

* Corresponding author.
E-mail addresses: Nicola.lenherr@gmail.com (N. Lenherr), rene.pawlitzek@ost.ch (R. Pawlitzek), bmi@zuerich.ibm.com (B. Michel).

https://doi.org/10.1016/j.suscom.2021.100580
Received 14 September 2020; Received in revised form 22 February 2021; Accepted 6 June 2021
Available online 9 June 2021

Table 1
Embeddeed Device Descriptions [1].

	EdgeTPU	NCS2	i9-SSE4	i9-AVX2
NN Environment	TF Lite	OpenVINO	TF Lite	TF Lite
MobileNet Model	v1	v2	v2	v2
Reported GOPS	58.5	160		
Measured GOPS	47.4	8.29	38.4	40.9
Reported Power (W)	2.0	2.0	205	205
Reported GOPS/W	29.3	80.0		
Measured GOPS/W	55.8	6.14		
Model Load Time (s)	3.66	5.32	0.36	0.36
Inference time (ms)	27.4	96.4	19.6	20.8

DL progress requires efficient methods and balancing of flexibility, number of classes, and accuracy. DL is compute-intensive by design: The flexibility that lets it outperform expert models renders it expensive, scaling faster than necessary [10]. Computations for training grow as square of data points and fourth power of performance, while inference energy grows with the square of classes. Regularization eliminates small coefficients, but at full costs of estimating all parameters. DL performs well because of over-parametrization and uses regularization to make the complexity tractable. Graphic Processors (GPU)- and ASIC-based DL led to widespread adoption, but computing grew faster, at 10x pa. from 2012 to 2019 [10]. Rearchitecting lowers computational intensity so that scaling becomes less onerous: For regularized flexible models it is O (Performance^4), better than DL scaling. Computational complexity is reduced by pruning weights, quantizing the network, or using low-rank compression but improvements are not sufficient in comparison to the slow overall increases of computation [10].

2. Related work

To identify energy-efficient CNNs, accurate runtime, power, as well as energy models and measurements are important. Common metrics for CNN complexity are too crude to predict energy consumption since it depends on architecture and hardware platform. Traditional profiling is inefficient for large search spaces, fails to capture effects of architectural changes on energy, and if service and training platforms differ [1].

2.1. Edge Intelligence and Federated learning

There is a need for edge AI computations but research on edge intelligence is still in its infancy [11]. Edge computing algorithms and models proliferated due to reduced latency, saved bandwidth, improved availability, and better privacy. The Open Framework for Edge Intelligence (OpenEI) enables lean algorithms, edge packages, and accelerated hardware. With federated learning (FL) clients compute an update based on distributed data and communicate this to a server for model aggregation. Communication cost is reduced by structured and sketched updates [12,13]. McMahan [14] show model averaging robust to non-IID data and 10−100x reduced communication compared to stochastic gradient descent (SGD). Artificial Intelligence on Edge (AIE) better handles user demands, avoiding communication latency [15]. Efficiency is improved by model compression, conditional computation, and algorithm asynchronization.

FL preserves privacy by distributed training deep neural networks (DNNs) on local data to optimize global models by averaging trained gradients. SGD updates local gradients, which is viewed as mini-batch. FL of a CNN, a long-short term memory (LSTM) recursive neural network (RNN), and a multilayer perceptron (MLP) were tested on resource constrained edge devices (Raspberry Pi4s, RPi4) with trainings on the Modified Nat'l Inst. of Standards and Technol. (MNIST) databases of handwritten characters. For training speed, a low number of batches per round ensured enough learning. For independent and identically distributed (IID) random variables, the number of batches per round was increased to minimize data transfer [16].

2.2. Efficiency improvement approaches

Algorithmic efficiency reduces computing to train models but is difficult to measure. Efficiency gains on sorting are simpler to measure than in ML because of clear task difficulty: O (n log n). Increased inference efficiency is important since inference costs dominate. Policymaking will improve by a focus on measurement and assessment of AI systems, in terms of impact and accurate intuitions about the cost of deploying AI capabilities. Hardware and algorithmic efficiency gains are multiplicative and should be both tracked [17].

Ensemble methods and distillation are promising in FL by: (1) identifying device cohorts with similar data distributions, (2) exploring privacy guarantees, (3) improving accuracy with few-shot FL, and (4) exploring non-convex models [18]. DNNs are computationally demanding, time-consuming, and memory intensive. Personalization is provided by updating DNNs on edge devices. Weight pruning reduces storage while re-training of pruned networks improves personalization and fault tolerance. [19]. Hardware constraints hinder DNN extension to small platforms and black box mechanisms hamper adoption. [20]. Network pruning reduces inference cost in low-resource settings. However, fine-tuning a pruned model gives similar performance than ab-initio training [21]. DNNs are heavily used with image data but sequential data such as text and audio and general time series can also be efficiently classified with DNNs [22].

The classification precision for large *Cl*, large number of features, and for little training data can improve due to weaker features contributing to classification [23]. Measuring energy consumption is needed since execution time can be longer and energy consumption lower. [24]. A tradeoff in FL is to select more clients in each round with less communication. MOCHA optimizes communication cost, stragglers, and fault tolerance [25]. Energy efficient transmission and computation resource optimization for FL reduces delay by 25.6 % and energy consumption by 37.6 % [26].

The major FL bottleneck is the communication overhead. Fusion Learning only sends distribution parameters and local model parameters, the server regenerates and fuses all data to build a global model resulting in similar accuracy [27]. FL is heavy on batteries, but a two-layered process provides 20 % energy savings. The first layer improves the initialization while the second provides local energy saving [28]. Network topology and batch size and data-level parallelism reduce energy consumption [29]. Data movement is minimized with algorithm and hardware co-design, reduced bit width, increased sparsity, data reuse, and compression. [30]. Hardware optimizations and lossless compression each improve energy-efficiency twofold.

Mobile inference reduces latency at the expense of billions of operations and parameter-reads. Frequent inferences drain batteries unless compressing models, minimizing data transfer, and offloading reduce consumption [31]. Layer decomposition exploits matrix approximation to decrease parameter redundancy, while reducing accuracy loss. Quantization enforces connections to share the same value by indexing a code book followed by re-training to increase accuracy [31]. Han's energy-efficient engine runs compressed networks, leveraging pruning, activation sparsity, weight sharing and quantization with 3,400× less energy compared to a GPU albeit at lower accuracy [32]. After having introduced efficiency improvements we are focusing now on the platforms, models, and frameworks used for our study.

3. Methods

3.1. Energy-precision ratio

The Energy-Precision Ratio (M) is a guideline towards both accurate and energy efficient CNN architectures [7]:

$$M = Error^{\alpha} \times EPI \quad (1)$$

where it uses classification error (*Error*) with adjustment exponent alpha (α), and energy consumption per data item classified (*EPI*). Higher α enlarge the importance of accuracy compared to energy consumption. A lower *M* shows a better energy-efficiency to accuracy tradeoff: ResNet-50 has the lowest error but higher energy consumption. With $\alpha = 4$, when accuracy is favoured, *M* of VGG-16 is smaller than AlexNet [7]. The disadvantage of *M* is that α is arbitrary and does not allow comparisons between models with different complexity. The error is different for models having 2 or 1′000 classes since model complexity increase with number of classes (*Cl)*. For a fair comparison *Cl* has to be included in an assessment.

3.2. Recognition gradient

Evaluation metrics are important for classifiers. Accuracy (*acc*) or error rate (*err*) are easy to compute for multi-class and multi-label problems and easy-to-use but lack distinctiveness, informativeness, and are biased to majority class data. $acc = tp + tn \;/\; (tp + fp + tn + fn)$ evaluates the quality of solution based on percentage of correct predictions over total number of instances. The complement metric, $err = fp + fn \;/\; (tp + fp + tn + fn)$, evaluates percentage of incorrect predictions [48]. Where *tp* is true positive, *tn* is true negative, *fn* is false negative, and *fp* is false positive. Alternative metrics among others are F-Measure (FM) and geometric-mean (GM).

The recognition gradient (REC_{grad}) is defined as difference between probability of the best and the second-best class. For an anomaly detection REC_{grad} is: Probability of anomaly minus probability of normal. For REC_{grad} we use the gradient between *tp* and next best probability. For a *fp* prediction the gradient *tp - fp* becomes negative and can be multiplied by a factor larger than -1. For search results the gradient is calculated similar to classifications while *fp* gradients are also multiplied with a negative factor. REC_{grad} is determined statistically from the sets of test cases. In classification tasks typically an accuracy of 80–90 % is achieved, which means 10–20 % are wrongly classified. In Fig. 1. this is seen as negative values. To "punish" false classifications stronger, they can in rare cases be subtracted with a two to threefold amplification.

Fig. 1 shows how the recognition gradient is calculated statistically over more than 100 tests. Such a process is beneficial since it closely mimics what is expected from classification processes. This approach makes the recognition gradient more stable and useful compared to classical approaches to determine accuracy. The recognition gradient is used both with recognition and training efficiency (see below). The recognition gradient can be calculated statistically for a series of top 1, 2, 3, 4, and 5 hits or misses. While top 5 accuracy is used to get a more balanced assessment for single events, the statistical analysis over more than 100 samples reaches a better result for top 1 analyses. As shown on Fig. 1, misses are observed in ~20 % of the cases and result in a negative gradient being added to the overall result. As expected, the risk for a Top-1 miss is higher than for a Top-5 miss but this is compensated by higher positive values when there are hits.

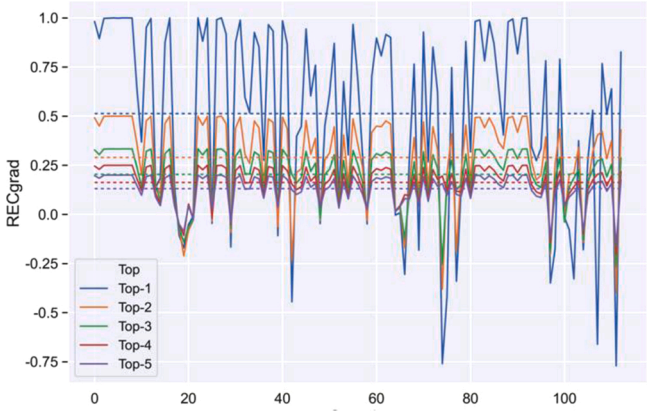


Fig. 1. Process to determine recognition gradient for Top-1 (blue line) up to Top-5 image classification tasks. The values for Top-1 are higher because to determine the gradient, accuracies are divided by N for Top-N predictions (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.).

3.3. Recognition and trainig efficiency

Energy-Precision ratio M compares models and platforms including accuracy [7]. M with $\alpha = 1$ looks very similar like the inference energy comparison and is thus not very helpful to provide a more balanced comparison between accuracy and energy. While *M* with $\alpha = 4$ provides a better viewpoint that well compares models trained on the same dataset. However, it fails on different datasets with different *Cl*: 1000 (image), 30 (audio), and 4 (RR). Deciding among 4 classes is simpler leading to low *M* than among 1000 classes. Discussions based on energy-precision graphs do not lead to a much better understanding on model, library, and platform efficiency. For this reason, we introduce the more universal "Recognition Efficiency" (*RE*) that balances accuracy, complexity, and energy consumption as follows:

$$RE = \frac{RECgrad \times Cl}{\sqrt[2]{Einf}} \tag{2}$$

where REC_{grad} is the recognition gradient as previously introduced, *Cl* the number of classes distinguished and E_{inf} the energy in milli Joules (mJ) to perform the inference computation. *RE* uses the square root of energy because a doubling of *RE* triggers a fourfold higher energy demand. *RE* can be universally applied across all models, libraries, and accelerators to define a new figure of merit for deep learning that leads us towards Green AI and that could also serve as basis for an efficiency standard. In the result section we test our method against a set of models, libraries, and platforms to demonstrate its universality. Complexity can be expressed as number of classes in a classification task but also selectivity in a page ranking task. In a natural language recognition effort complexity results from the number of options from which a word pattern is created on average. In general complexity is defined as: Out of how many options does the model decide.

In a similar fashion like for *RE*, REC_{grad} is applied to training leading to the training efficiency (*TE*):

$$TE = \frac{(RECgrad \times Cl)^2}{\sqrt[2]{Etrain}} \tag{3}$$

where *Cl* is the number of classes the model distinguishes and E_{train} the energy in kilo Joules (kJ) for DL last training and verification round. *TE* scales with the square of energy invested. With the 4th power of accuracy *TE* rewards models that distinguish among many classes with a good REC_{grad}. *TE*, like *RE* universally compares deep learning models. E_{train} does not include the overall energy for the model development with its hundreds of attempts and hyperparameter grid or gradient searches. This is too diverse and difficult to generalize. One simple approach could be to assume an overhead factor between 10 and 100 to approximate model development. This factor might even provide an average between initial model development and repetitive model maintenance when a smaller hyperparameter grid search is performed.

3.4. Life cycle aspects

For a complete evaluation of DL, inference and training need to be combined with a factor that reflects how often inference is executed per training cycle. The combined term is called deep learning lifecycle efficiency (*DLLCE*):

$$\mathrm{DLLCE} = 1 \;/\; (\mathrm{E_{train}}/(\mathrm{E_{inf}} \times \mathrm{F}) + 1) \tag{4}$$

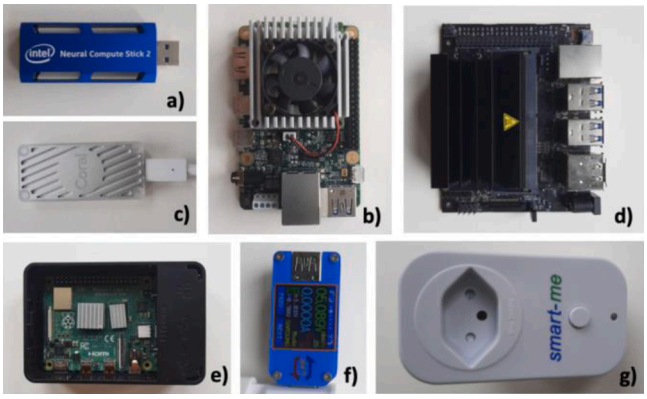


Fig. 2. Platforms used and of the tools for acquisition of electrical power consumed: a) Movidius NCS2, b) Coral Dev board c) Coral Dev Sick, d) Jetson Nano, e) Raspberry Pi4, f) Joy-IT UM25C, g) SmartMe.

where E_{train} is the training energy, E_{inf} the inference energy, and F the factor how more often inference is used. *DLLCE* shows that E_{train} is much larger than E_{inf} but when models are often used, the energy investment is amortized. *DLLCE* approaches 1 for large *F*. *TE* in eq. 3 rewards a steep REC_{grad} and large *Cl*. The reason is to fairly distinguish different models because data volume for training and model size increase more than linearly with *Cl*. REC_{grad} decreases with *Cl*. *RE* varies with included training energy divided by the number of uses of the model:

$$RE_{LC} = \frac{RECgrad * Cl}{\sqrt[2]{Einf + Etrain/F}} \quad (5)$$

The lifecycle recognition efficiency (RE_{LC}) combines *RE* with *TE* while dividing *TE* with the number of uses F. RE_{LC} reaches the same values like *RE* but only for large number of uses (*F*) when *TE* has been deprecated. The situation does not vary considerably between centralized (a) or FL trained models (b) because data for inference remain on the edge and E_{inf} dominates. The reason for the selection of *F* ranging from 100 to 10'000'000 is that for the vast majority of models the transition from full training energy domination to full inference energy domination happens in this range.

4. Results

4.1. Devices

To cover a broad spectrum, we used different accelerators (Fig. 2, Table 2). For all devices detailed documentations are available in the cited literature, and we do not repeat them for the sake of brevity:

First, the Intel MovidiusX processor [9,35] (Fig. 2a), an embedded video processor with a Neural Engine for video processing and object detection. The NCS2 features an Movidius Myriad VPU. To run inference, the models are converted into the OpenVINO toolkit supported Intermediary Representation format. For the experiments we used a RPi4 [33] host system.

Second, the Google Coral Dev Board (Fig. 2b and c) features the Edge TPU running with a Quadcore ARM A53 processor. To run a model on the Edge TPU it must be a quantized TensorFlow Lite model which is compiled by Coral's edge compiler. There is also the USB version of the Edge TPU [34].

Third, the Nvidia Jetson Nano (JNA, Fig. 2d) with a Quadcore ARM A57 processor and a GPU with 128 CUDA Maxwell cores. In 5 W mode only 2 of the four cores are active and CPU and GPU run at a lower clock to reduce power. The MAXN mode uses 10 W with increased clock rate. Nvidia offers compiled Python wheels for TensorFlow and PyTorch. Additionally, models can be optimized with TensorRT [36].

Fourth, a RPi4 (Fig. 2e) uses a 64-bit quadcore ARM72 running at 1.5 GHz. It has 2 USB2 and 2 USB 3 sockets to connect NCS2, and CORAL sticks. Downstream USB is limited to 1.1 A and 5.5 W. It is equipped with 4 GB SDRAM.

Finally, experiments on a Windows workstation serve as reference (*Tab. II*). As NCS 2 and Edge TPU require specific model formats, we used the TensorRT optimized models on the JNA to achieve best comparability.

4.2. Models and frameworks

1) Heart rate variability parameters (HRV pars): The model distinguishes between 4 stress classes: "relaxed", "mental stress", "physical stress", and "combined stress". The data consists of RR intervals [37] of persons undergoing stress tests that are converted to 17 HRV pars [38] used for training of a two-layer MLP model with 128 neurons followed by a dense layer of 4 and a SoftMax activation.
2) Heart rate variability from RR intervals: The model is a 1D CNN applied on the raw RR intervals [39] (Table 3). Both models (1,2) used ReLU activation and were trained with the Adam optimization algorithm [40]. The final layer uses SoftMax activation.
3) Environmental Sound Classification (ESC) contains 2000 5 s clips recorded with 44.1 kHz separated into 50 different classes, (https://github.com/karolpiczak/ESC-50#citing). The model is a standard CNN. To reduce training time the data augmentation was

Table 3
Raw RR Interval Model.

Layer	Kernel Size	Pooling Size	# Kernels / Neurons
Conv1D	3	–	64
Conv1D	3	–	64
MaxPool1D	–	2	–
Conv1D	3	–	64
Conv1D	3	–	64
MaxPool1D	–	2	–
Flatten	–	–	–
Dense	–	–	100
Dense	–	–	4 (# classes)

Table 2
Devices and Specifications.

Device Overview	Specifications		
	CPU	GPU/VPU	Frameworks/Models
(1, a) Neural compute Stick 2	Movidius Myriad™ X Vision Processing Unit @ 700 MHz	–	OpenVINO
(2, b) Coral Dev Board	Quadcore ARM A53 @ 1.5 GHz	–	TF Lite, Quant, Edge compiled
(2, c) Coral USB Accelerator	–	–	TF Lite Quant + Edge compiled
(3, d) Jetson Nano	Quadcore ARM A57 5 W: @ 921 MHz 10 W: @ 1.5 MHz	128-c Maxwell 5 W: 614MHz 10 W: 921 MHz	TF, TF Lite, PyTorch, TensorRT
(4, e) RPi 4 Model B	Quadcore ARM A72 @ 1.5GHz	–	TF Lite Interpreter
(5) Windows Dell Precision 5520	Core i7−7820HQ @ 2.9GHz	Nvidia Quadro M1200, 1093 MHz	TF PyTorch

skipped and only 30 classes were used. The model's inputs are the 40 Mel frequency cepstral coefficients (MFCC) [41].

4) Image Classification: To be comparable to literature three image classification model architectures were included: MobileNetV2 (1.0), ResNet-50 and VGG-16, trained on the ImageNet dataset with image input size 224 × 224 pixels and 1000 distinct classes. [40,42–45].

As mentioned before the devices require a certain model format for the accelerator:

The OpenVINO Toolkit uses IR representation and is converted using the Model Optimizer. OpenVINO Linux distribution, was installed in a Docker container. For image classification, models from the ONNX Model Zoo were used. HRV and ESC30 are trained with PyTorch and converted to ONNX for FP16 or FP32. During conversion, the Model Optimizer fuses operations for a faster throughput, scaling, and centering the data. We did not apply centering to separate inference from the data pipeline to better compare with other platforms. We also did not cut off model parts. The output contains and XML file with network topology, and a binary file with weights. We used the Inference Engine Python API.

The Edge TPU requires quantized TensorFlow Lite (TF Lite) with image classification models from TensorFlow Hub or the tensorflow.keras.applications package. Models for HRV, RR, and ESC30 are saved in the SavedModel format. Post-Training Quantization was used. Compilation of the quantized models with the Edge TPU Compiler allocates the parameters to the SRAM. Models larger than 8 MB do not fit on the Edge TPU and require slow parameter re-loading from external memory. We used the TF Lite Python API with the Edge TPU delegate.

For trained models CUDA-based TensorRT on JNA combines layers, selects the optimal kernels, and applies additional optimizations for lower precision such as FP16 or INT8 when supported by the hardware. As the JNA does not support INT8 only FP32 and FP16 models were used. The tool for model conversion is trtexec which generates serialized engines based on the same ONNX models as for OpenVINO. We used the TensorRT Python API.

4.3. Power and energy measurement

First, measurements of inference energy were implemented. For each device a single inference step is one data item (batch size = 1). The inference speed includes copying the data to and from device. This is necessary because of lack of full control of data transfer. A single inference latency measurement consists of a starting and an ending timestamp. Based on these two the inference latency in milliseconds was calculated.

For USB powered devices the metering device UM25C (Joy-IT, Germany) (Fig. 2f) [46] was plugged between power adapter and USB cable. The UM25C transfers power, voltage and current via a Bluetooth interface. During the inference test power consumption was logged every 400 ms. This is the shortest interval to get results. For other devices (PC and JNA in MAXN mode) power was recorded with the Smart-me (Rotkreuz, Switzerland) (Fig. 2g) [47] and data was accessed via web interface. The power consumption was logged with the same recording frequency.

Inference and power consumption measurements were logged with timestamps. To calculate the inference power, we used timestamps to map the power to the inference and calculated the mean between the start and end timestamp. For most measurements this was one data point. For comparison between devices a baseline for each device was established. The baseline is the average power consumption of 40 s (~100 data points) on the device while idling. For the TPU and the NCS2 USB sticks, the baseline was recorded on the RPi4 while the device was plugged in. The amount was subtracted from the power recorded during the inference and the difference was then used to calculate the energy (Section IV). To separate data loading and resizing, data is loaded and all steps for the data to be ready for inference are applied. Then inferences are run repeatedly with the same data and the power consumption is recorded. This triggers a short power spike that could lead to false results. For most models and devices, the inference time is less than 400 ms, the measuring frequency of the UM25. This could associate the inference timestamp with the power before the inference phase. To prevent the ramp up phase influencing the results, inferences are run for a couple of seconds before and after. During the inference no data is written, so the I/O has no influence. The timestamps and the inference delta are written to disk after the phase out.

A sophisticated baselining process is essential to allow accurate power measurements in small and also in larger Cloud systems. Cloud systems are defined as typical personal computer or server processors that run as part of multi-user, multi-tenant environments where we have no control over all activities in the system. A baseline measurement is carried out every time before a measurement is taken (Fig. 3). The process is executed on all platforms with the assumption that background processes on larger systems do not massively change during execution of the monitored process. This is mostly true except for a few humps in the baseline recording (orange) and in the inference recording (brown) which are triggered by spurious, non-identified processes. The frequency of such events is larger on systems with many background processes or independent tasks. It is assumed that loads do not change on a second level between baselining and measurement. Spurious activities happen on all systems, but they are statistically subtracted out since baselines and measurements are done multiple times and averaged.

The objective is to compare DL models for image, audio, and heart rate variability data on accelerated platforms using different libraries. 2D image data is computationally more demanding than 1D audio data (44 kHz) and heart rate or heart rate variability data (1 Hz). The difference in complexity is more than four orders of magnitude which is also reflected in the power consumed. We first start out with a comparison of energy consumption for inference and learning.

4.4. Energy and accuracy

Energy consumption is compared with measurements (Fig. 3) using USB or AC power meters. To accurately measure the inference or DL power a base line is recorded while the system is put into a controlled waiting state. Additionally, the power of inference or training is also monitored separately from the phase where the data and/or models are loaded. Large models (brown) consume more energy than medium (violet) and small (red) models. The factor is >6′500 between the largest (VGG-16) on Cloud and the smallest (MobileNet) on TPU. This factor becomes 300′000 when models on audio (green) and HRV pars (blue)

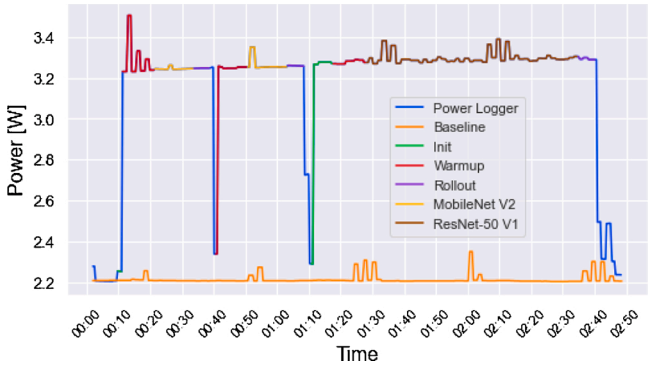


Fig. 3. Power measurement graph showing how the baseline was recorded to isolate the energy associated with inference and/or training. The line colors show the different phases (init, green; warmup, red; rollout, violet; and power logger, blue) that were isolated from the inference process of MobileNet V2 (yellow), or ResNet-50 V1 (brown) to improve the accuracy (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.).

are included. On the same platform (NCS2, JNA) the inference energy E_{inf} varies from 578 mJ (large), over 186 mJ (medium) to 70 mJ (small). This factor is larger (106x) on TPU. Small image recognition models save 10−100x energy compared to large ones, albeit at slightly smaller accuracies. But the accuracy difference is much smaller than the energy consumption difference. Accuracies on all platforms are: 81.2, 82.8, 84.9, 76.2, 83.3, 74.6 % for HRV pars, RR, ESC30, MobileNet, ResNet-50, and VGG-16, respectively.

The comparison is unfair since TPUs use 8-bit precision. The NCS2 and JNA consume 2x more energy for the large model irrespective whether it is 32- or 16-bit, and 10–20x more for the small MobileNet model. The difference between 16- and 32- bit precision should be >2 unless not implemented on all network layers. The power consumption of Cloud is 100x larger for VGG-16 and >1000x larger for MobileNet irrespective whether computed on GPUs or CPUs. The reason is the added data transfer energy to the Cloud (70 kB for Images, 50 kB for sounds, and 0.5 kB for RR). The energy for transfer of a GB from a Mobile/IoT device to the Cloud is 202 J/Mbyte [49,50]. Inference comparisons between platforms are unfair for models with different accuracies. It would be desirable to compare models and platforms balancing energy consumption and accuracy.

4.5. Energy-precision ratio

Energy-Precision ratio M compares models and platforms including accuracy [7]. *M* in Fig. 5, eq. 1 compares inference as in Fig. 4 with $\alpha = 1$, upper graph $\alpha = 4$, lower graph [7]. For large and intermediate models (brown, violet) the difference among platforms is small. For MobileNet *M* is 1000x smaller on TPU than on Cloud, showing that small models improve more on small systems. M for small models like HRV pars (blue, lower graph) is larger since their accuracy is lower due to lower label quality, but the inter-platform variability is small. For ESC30 *M* is lower because 1D recognition of time-based data is efficient using MFCCs. *M* for ESC30 recognition on JNA is $3.03*10^{-6}$ as compared to HRV 0.001. *M* varies by ~6 orders of magnitude between $3*10^{-6}$ and 1.6 (VGG-16 on CPU) where 1.6 is a bad model and platform and a small *M* shows a good model. From an accuracy standpoint ESC30 shows up low (good accuracy) and VGG-16 high (bad accuracy) on the *M* graph (Fig. 5 lower panel). But VGG-16 can classify 1000 while ESC only 30 classes.

M with ($\alpha = 1$, Fig. 5 upper graph) looks very similar like the inference energy comparison in Fig. 4 and is thus not very helpful to provide a more balanced comparison between accuracy and energy. While *M* with ($\alpha = 4$, Fig. 5 lower graph) provides a better viewpoint that well compares models trained on the same dataset. However, it fails on different datasets with different *Cl*: 1000 (image), 30 (audio), and 4 (RR). Deciding among 4 classes is simpler leading to low *M* than among 1000 classes. Due to the low E_{inf}, the HRV pars (blue) reaches low *M*. The lowest is reached by the ESC30 model on all platforms where it could be computed because the accuracy is high. *M* is more than 3 orders of magnitude lower on the same platform than VGG-16 and two orders of magnitude lower than MobileNet. Generally, *M* is high on ARM and server CPU/GPU. A limitation of *M* is that low values show good models while high values show inefficient models that are not very accurate. Discussions based on energy-precision graphs do not lead to a much

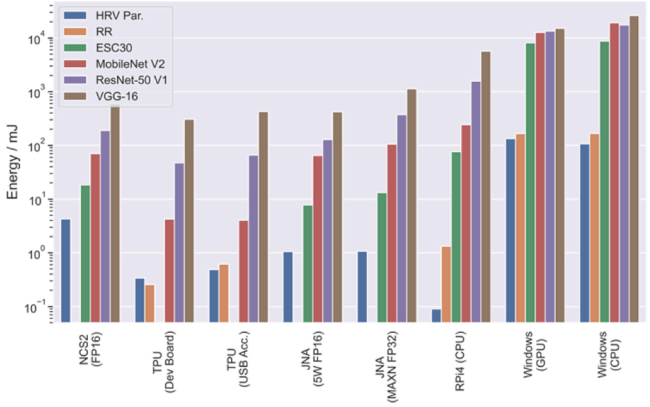


Fig. 4. Inference energy in mJ for stress recognition from HRV pars (blue), or raw HRV (orange), ESC-30 sounds (green), and image recognition for MobileNet model (red), ResNet-50 (violet) and VGG-16 (brown). The left two columns are from the NCS2 (16/32 bit), the next two from TPU (Board/Stick), the third two from the JNA at 5W (16/32 bit), the fourth two from the JNA running at full power, the fifth from the RPi 4 platform, and the final two from a Windows PC (GPU/CPU) (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.).

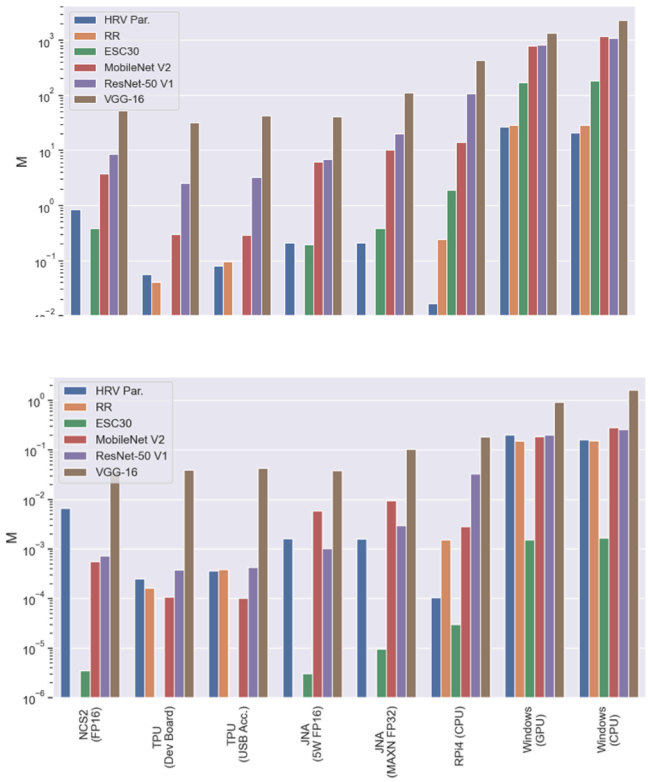


Fig. 5. Energy precision ratio M for α=1 (upper graph) which emphasizes power consumption over accuracy and M for α=4 (lower graph) which emphasizes accuracy over power consumption shown for same models and platforms as in Fig. 4.

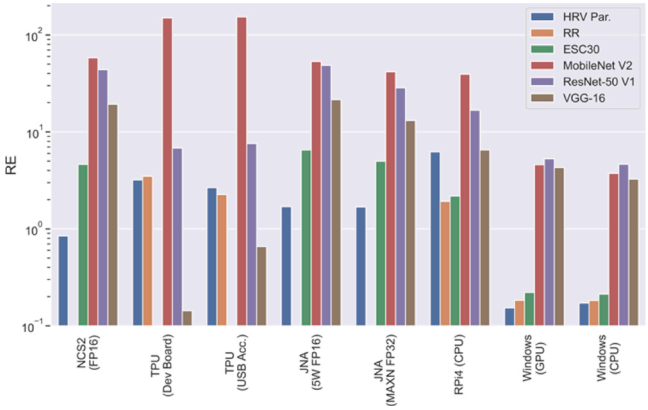


Fig. 6. Recognition efficiency shown for the same models and platforms with the same colors as Fig. 4. *RE* larger than 1 (10°) are considered good while models and platforms with *RE*<1 are considered inefficient.

better understanding on model, library, and platform efficiency.

4.6. Recognition efficiency

Fig. 6 shows a superior *RE* of MobileNet (red) on TPU followed by JNA low power, NCS2, JNA MAXN, the ARM on RPi4. *RE* is much lower for Cloud. ResNet-50 model (violet) is more on par with the accelerated platforms while Cloud performs 10x worse. For ResNet-50 the TPU performs less than JNA and NCS2. For the VGG-16 model (brown) the TPU board performs 100x worse than NCS2 and Jetson. Still JNA and NCS2 are 3−6x more efficient than Cloud. The audio (green) and time-based models, are ranked as follows: *RE* is even among accelerated platforms except for Cloud. ESC 30 *RE* is 10x better on RPI4 than on Cloud which also performs 30x worse than the best accelerated platform. For the HRV Par model (blue) with 4 classes the TPU is performing 4x better than NCS2. Astonishingly *RE* of HRV Par is very high on the RPi4. For this model the Cloud performs two orders of magnitude worse. While image recognition models were carefully optimized, we show that *RE* is helpful to compare the maturity/efficiency on best recognition for the least amount of energy spent in inference. For search results Cl corresponds to the inverse of the predicted positive condition rate. For both cases the accuracy gradient diminishes for increasing Cl and selectivity.

4.7. Cloud-centric and federated learning

E_{train} is calculated like E_{inf}, but power consumptions are 4–6 orders of magnitude larger. Still E_{inf} is more important since it is used much more often. Training large and medium size models (VGG-16 and ResNet-50) takes weeks on Cloud consuming thousands of kWh - too big for edge systems. The only model accessible for smaller systems is FL, distributed training with several edge systems and an integrating server for ESC50, RR, and HRV. Beneficial is that the training is on systems that generate data (camera, Mobile Hub) and receive labels, reducing the energy wasting movement of data to Cloud and leaving personal data with the owner. *Tab. IV* shows E_{train} on Cloud (VGG-16, ResNet-50, and Mobile Net) and on edge systems (MobileNet, ESC 30 and HRV).

Table 4 shows that E_{train} is 1000−10'000x larger than E_{inf}. This is reflected as inverse and square root in training efficiency (*TE*). *TE* in eq. 3 rewards a steep REC_{grad} and large *Cl*. The reason is to fairly distinguish different models because data volume for training and model size increase more than linearly with *Cl*. REC_{grad} decreases with *Cl*. E_{train} is only for the last training round and not for the model development which needs hundreds of rounds to get the hyperparameters optimized for sophisticated models.

E_{train} for HRV pars, RR raw, and ESC50 was measured while E_{train} for MobileNet, ResNet-50, and VGG-16 is from literature [51,52] or calculated via training times. *Tab. IV* shows a higher *RE* on JNA compared to Cloud. For the RPi4 only the HRV pars model is more efficient while the RR and ESC50 models perform worse. Current FL with many epochs leads to higher E_{train} since we have not reduced the model communication overhead like one- or few-shot learning [18]. A disadvantage is that FL requires more epochs for the same accuracy. Federated Averaging, Federated Stochastic Variance Reduced Gradient, and CO-OP on the MNIST dataset show a similar accuracy with the same training epochs with IID data [53]. Trainings were not possible on NCS2 and TPU since these devices are too much limited.

E_{train} of VGG-16 and ResNet-50 are large, requiring weeks of training on multi-GPU clusters. This is better with MobileNet that is optimized for low network complexity so that *RE* becomes high. E_{train} are 392000, 604800, and 14578 kJ for VGG-16, ResNet-50 and MobileNet, respectively. E_{train} for FL is slightly higher than for centralized approaches leading to a 10x shift of the HRV pars curve on RPi4 (blue, solid) towards higher *F*. E_{train} of HRV pars, for example is 0.461 kJ with a centralized approach and 2.73 for a federated approach, about 6x larger! The factor is even larger on JNA here FL consumes 23x more energy which also shifts the *DLLCE* curve (blue, dashed) to almost 50x higher *F*. Unfortunately, we cannot give values for E_{train} *FL* on Cloud (*Tab. IV* right columns). These values are not published because of lack of efficiency and sustainability transparency demands.

Table 4
Training/Life Cycle Energy and Efficiency.

Platform	Model	E_{train} (kJ)	TE	E_{train} FL (kJ)	TE FL
RPi 4	HRV pars	0.461	5.06	2.73	1.89
	RR raw	15.84	1.23	18.74	1.27
	ESC 30	83.25	1.7	104.9	1.66
JNA	HRV pars	0.12	8.88	2.79	1.87
	RR raw	0.606	7.06	10.34	1.71
	ESC 30	10.45	5.26	42.9	2.60
Cloud / PC / Workstation	HRV pars	1.58	2.48		
	RR raw	3.71	2.86		
	ESC 30	31.1	3.05		
	MobileNet V2	14578	95.9		
	ResNet-50-V1	604800	21.4		
	VGG-16	392000	18.6		

4.8. Life cycle aspects

Fig. 7 shows the life cycle efficiency of image, sound, and stress recognition as function of *F*: With >100′000x uses E_{train} is well amortized and reduces *RE* by <10 %. Uses <100x can expand the energy 100x, however. For training in the Cloud (Fig. 7, *left*), E_{train} for an ESC30 model is 3.1 million x larger than E_{inf}. This is expressed as a reduced *DLLCE* for 1000 uses of 0.03. For HRV pars this is 0.4. The ESC30 system needs more than one million uses so that training adds <5% overhead. For ESC30 on JNA (green, dotted line) E_{train} is 1000x larger than E_{inf}. If the model is used 1000x the E_{train} overhead is 20 %. For direct use of RR-intervals E_{train} is low and fully amortized to <1% overhead with 100 uses. For HRV pars on JNA E_{train} needs 1000 uses for a full amortization. FL is not more efficient despite the move of training data to the Cloud because models are as large as datasets. The lifecycle energy consumed for a DL application is calculated by E_{inf}. (Fig. 4) multiplied by *F* and divided by *DLLCE*. Curves in Figs. 7 and 8 are drawn using DLLCE (eq. 4) and *F* sweeping from 100 to 10,000,000.

The lifecycle recognition efficiency (RE_{LC}) shows efficient models ($RE_{LC} > 1$) like HRV pars (blue) or RR intervals (yellow) on RPi4 (solid) or JNA (dashed). MobileNet on Cloud (red dotted) and ResNet on Cloud (violet dotted) reach this threshold but only at $F > 10'000'000$. Most models have to be used more than 100′000x before the training energy is

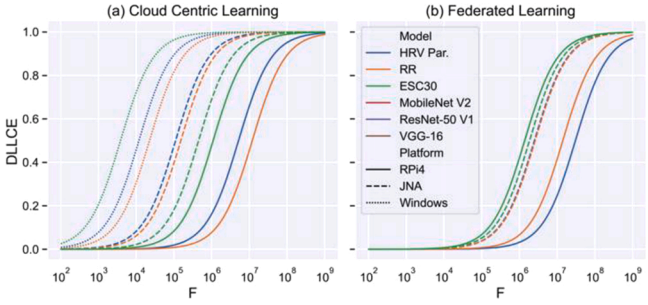


Fig. 7. Deep learning lifecycle efficiency (a) for centralized and (b) FL approaches as function of number of uses (*F*). Training on Cloud (dotted) need longer amortization than on JNA or RPi4 (solid, dashed). Small models (HRV pars, blue and RR, brown) are quickly amortized. ESC30 models trained on Cloud (green, solid) need much more to amortize training (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.).

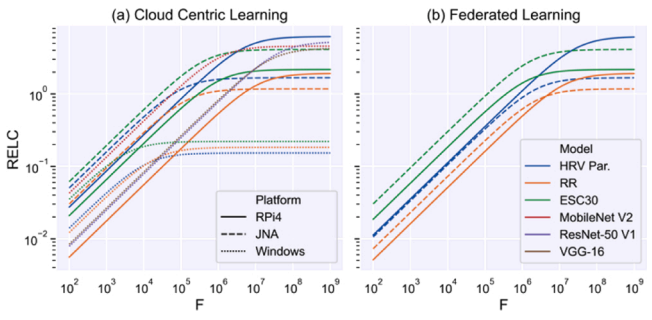


Fig. 8. Recognition efficiency over the entire lifecycle (a) for centralized and (b) federated learning approaches as function of number of uses (F). with models and platforms as in Fig7.

deprecated. Below that threshold most models are E_{train} dominated. Models with intermediate RE_{LC} (0.1–1) are ESC 30 on JNA (green dashed) and VGG-16 on Cloud (brown dotted). In the worst category (RE_{LC} <0.1) we find the ESC30 model on windows as well as image analysis models (VGG-16 and ResNet50) that have not been used enough ($F < 1$ Million) to depreciate E_{train}. The situation does not vary considerably between centralized (a) or FL trained models (b) because data for inference remain on the edge and E_{inf} dominates. The reason for the selection of F ranging from 100 to 10′000′000 in Figs. 7 and 8 is that for the vast majority of models the transition from full training energy domination (linear slope) to full inference energy domination (horizontal line) happens in this range. The curves have been reproduced using Eq. 4 and Eq. 5 with F sweeping the range in 10,000 steps using training and inference energies resulting from previous figures or tables. The reason for the choice of 10′000 and 100 million in Table 5 used to determine total energy is again motivated to characterize a regime that is mainly training energy dominated and a regime that is mainly inference energy dominated.

Table 5 shows the lifecycle energy for 10′000 uses in kJ and for 100 million uses in MJ. Life cycle energy is very small for HRV pars on RPi4 and JNA and 10x larger on Cloud. For ESC-30 the ratio is: JNA 8x smaller compared to RPi4 and 10x smaller than on Cloud. For image recognition on a movie, 100 million inferences are reached in 46 days. Baselining does not include static power while idling (neither Edge nor Cloud). The ratio of active to passive power is 2.5. To include all power (for single task) we have to enlarge RE_{LC} by 2.5. When other tasks run on the system, the factor may vary from 1.5−2. *DLLCE* includes the last training and not the model development which may take 100–1000 such training cycles for highly optimized models needing vast hyper-parameter tuning with grid searches. In such cases the total AI model lifecycle energy exceeds 200′000 MJ, several times larger than the energy a car uses in its lifetime.

It is difficult to find detailed information about hyper-parameter tuning in literature; for this reason, we have not included number of development and tuning cycles in the E_{train} and total energy values. Use of deep learning less than 100 times makes no sense from the point of view of efficiency. Usage numbers of 10^9 are reached when all images are analyzed in a video sequence lasting a few years or when a model was installed on ten million devices each analyzing 1000 still images. Extended use of models (10^{10}, Table 5) is needed to completely write off E_{train}. For readers without experience in energy: We can compare 1 kJ with a wearable being activated for 1 h and 1 MJ with a television set or personal computer running for one hour, or a one-minute shower.

Table 5
Life Cycle Energy For 10′000 And 100 Mio Uses.

Platform	Model	DLLCE 10′000	DLLCE 100′000′000	Energy 10′000 (kJ)	Energy 100′000′000 (MJ)
RPi4	HRV pars	0.044	0.975	0.462	0.0093
	RR raw	0.029	0.945	15.85	0.148
	ESC 30	0.950	0.995	84.01	7.690
JNA	HRV pars	0.288	1	0.135	0.112
	RR raw	0.248	1	0.646	0.399
	ESC 30	0.248	1	10.69	2.331
Cloud / PC / Workstation	HRV pars	0.677	1	2.914	13.36
	RR raw	0.554	1	5.352	16.45
	ESC 30	0.850	1	112.0	809.4
	MobileNet	0.509	1	486.0	1261
	ResNet	0.014	0.830	604934	1947
	VGG-16	0.019	0.891	392150	1894

Fig. 8 shows efficient models ($RE_{LC} > 1$) like HRV pars (blue) or RR intervals (yellow) on RPi4 (solid) or JNA (dashed). MobileNet on Cloud (red dotted) and ResNet on Cloud (violet dotted) reach this threshold but only at $F > 10'000'000$. Most models have to be used more than 100′000x before the training energy is deprecated. Below that threshold most models are E_{train} dominated. Models with intermediate RE_{LC} (0.1–1) are ESC 30 on JNA (green dashed) and VGG-16 on Cloud (brown dotted). In the worst category (RE_{LC} <0.1) we find the ESC30 model on windows as well as image analysis models (VGG-16 and ResNet50) that have not been used enough ($F < 1$ Million) to depreciate E_{train}. The situation does not vary considerably between centralized (a) or FL trained models (b) because data for inference remain on the edge and E_{inf} dominates. The reason for the selection of F ranging from 100 to 10′000′000 in Figs. 7 and 8 is that for the vast majority of models the transition from full training energy domination (linear slope) to full inference energy domination (horizontal line) happens in this range. The curves have been reproduced using eq. 4 and eq. 5 with F sweeping the range in 10,000 steps using training and inference energies resulting from previous figures or tables. The reason for the choice of 10′000 and 100 million in Table 5 used to determine total energy is again motivated to characterize a regime that is mainly training energy dominated and a regime that is mainly inference energy dominated.

5. Discussion, summary, and outlook

5.1. Discussion

Edge platforms were subject to an astonishingly fast recent progress with a speed that far exceeds further development in Cloud systems. For this reason, DL or EI tasks that were inaccessible for these systems two years ago can now easily be handled. One of the main messages of our paper is that edge accelerators are a major component of this development. These accelerators make edge systems strong enough to handle the vast majority of AI applications once they are reasonably optimized and Cloud is only needed in a few exceptional cases. With the focus on Red AI in Cloud the model development neglected this new interesting player and thus has prevented harvesting of low hanging efficiency opportunities. A lot of Cloud focused players underestimate the overall benefits of edge optimized Green AI. The current state is not the end of a trend but only the beginning since the edge AI development will even accelerate with analog neuromorphic accelerators.

Accelerated edge systems are currently the major driver for Green AI and for the trend to render DL and EI sustainable. Models on edge systems recently closely approach accuracies of Cloud-based models but spend 1000−10′000x less energy. During the past decade most models have been optimized exclusively for performance or accuracy which pushed the field towards non-sustainable Red AI. Energy consumption can neither be the sole driver to enforce development towards Green AI, but there is a need for new universal metrics that establish a fair balance between accuracy, complexity, and energy. It is clear that a linear comparison between accuracy and energy is not fair, but we have introduced the energy term as a square root for inference and with 0.25 power for training. To reach universality we also introduced the number of classes a model can distinguish. *RE* now includes accuracy, *Cl* and the square root of energy consumption for a better balance between performance and efficiency or carbon footprint. With *RE* the most efficient

models, libraries, and platforms that improve efficiency of DL can be selected.

RE universally compares among models, platforms, and even humans. Introduction of the universal *RE* and *TE* metrics is crucial since ML efficiency evaluation lacked accurate, universal tools. [24,54]. CNNs have considerably improved accuracy at the cost of increased energy consumption leading to VGG-16 but MobileNet started to partially revert this trend [55]. While VGG-16 shows comparable performance like humans, it has orders of magnitude lower efficiency. MobileNet on TPU is >1000x more efficient exceeding the efficiency of humans in distinguishing 1000 objects. Since, however, humans distinguish more than 30′000 objects within 500 mS and due to the quadratic dependence of recognition energy on *Cl,* humans are still more efficient. Additionally, human reflexes make simple energy-efficient decisions without involvement of our brain. Reflexes are trained by repetitions - the same as training edge systems to react with low latency without Cloud. Biology uses a hierarchy of models! The lower level is fast, energy-efficient, and used very often. The higher-level is for special, rare cases and justifies a higher energy consumption without draining energy resources.

A hierarchy of models is thus also important for efficient DL since inference energy demand scales with the 2^{nd} power of performance and training energy scales with the 4^{th} power of performance. Obviously, humans recognize images while having other activities which allows to do image recognition with less than 1 J. Sound recognition is faster consuming less than 10 mW. Human training may last a year, consuming 631 MJ or 175 kW h, about the same as the last training round of a VVG-16 model, while model development may use 100x more. However, AI models can be copied to hundreds of systems unlike for humans that need to train individually. Stressed or tired humans showed much lower cognitive performance for more complex tasks [38,39,56,57].

DL is a flexible, but inefficient model which needs efficient, specialized hardware with optimal design choices to be sustainable. This inverts the previous software – hardware dogma: A specialized, efficient software runs on a general, inefficient processor. DL is a less efficient general concept that needs specialized, efficient processors or accelerators to be viable. We need to select networks that meet optimizing needs on energy and satisficing needs on accuracy. We also need to carefully select and test the appropriate functional units of accelerators with the right flexibility and specialization as well as the right balance of functional units. Accuracy is the optimizing metric, because it needs to correctly detect a cat. The running time is the satisficing metric which means it has to meet expectation [58].

On edge devices, Once-for-All (OFA) outperforms state-of-the-art methods (up to 4.0 % ImageNet top1 accuracy improvement over MobileNetV3, or same accuracy but 1.5x faster than MobileNetV3, 2.6x faster than EfficientNet w.r.t measured latency) while reducing many orders of magnitude GPU hours and CO_2 emission [20,59]. Argumentation is difficult since always two comparisons have to be done (1) accuracy and (2) time which influences energy – that becomes simpler with *RE*. The focus of OFA is on training while the effect on inference is smaller and hence on the overall lifecycle carbon footprint.

5.2. Summary and outlook

RE universally compares DL model efficiency across compute platforms, libraries, and models. An important step is to measure energy consumption for inference and DL. Lower precision ASIC accelerators reach higher values for *RE* and *TE* compared to Cloud. The key contributors are the accelerator ASICS design choices and matching lean libraries. It is clear that MACS or MADDS are not optimal to compare efficiency, but new values are needed: **Recognition efficiency *RE*, training efficiency *TE*, deep learning lifecycle efficiency *DLLCE,* and life cycle recognition efficiency RE_{LC}.** For a full assessment in the fashion of a life cycle analysis we have defined *DLLCE and* RE_{LC}. These parameters assess how close the overall efficiency is compared to the inference or *RE*. It therefore rewards lean models that can be used many times between re-training.

Public discourse avoids AI's environmental costs but on a path to Green AI transparency needs to be enforced by only allowing results in leaderboards that convincingly show energy demand and overall efficiency. Measuring the carbon footprint of computing and disclosing this information is an important first step: A "Carbon Footprint Label" could raise awareness about the implications of AI adoption [60]. Curbing black box decision making is essential - Black Boxes are neither socially responsible nor Green! Explainability in AI should therefore also include transparency on usefulness based on our recognition efficiency metric.

Humans are not optimizers but satisficers - or corner cutters who are satisfied with good enough as opposed to perfect. Humans use cognitive heuristics to save time and effort. Working out the perfect running path to get away from a lion would get us eaten before we started the calculation. Satisficing also reduces the energy cost of living, which is important since our brains and analogous our computers are power-hungry devices. **This means as soon as efficiency and speed become important, performance maximization has to be abandoned. This is the same that has to happen in the transition from Red AI to Green AI to reach the overall goal of Sustainable Edge Intelligence!**

Red AI optimizes the performance irrespective of the energy needed. This is the optimizer approach which when put in relation to human development would not have allowed survival. Thus, we need to switch to the satisficer strategy to direct Red AI towards a Green AI strategy to help saving the climate and by that our survival. To develop towards that goal, we need parameters that support satisficer decisions in a wide number of cases and these are the recognition efficiency *RE*, training efficiency *TE*, deep learning lifecycle efficiency *DLLCE,* and Life cycle recognition efficiency RE_{LC}.

The key statement showing the need for a universal metric is: "You only can improve when you measure, and you only can standardize when the measurement is universal". For this reason, our work is crucial to drive Red AI towards Green AI. In fact, based on our metrics, we can create a similar standard like ENERGY STAR for appliances and computers: *RE* above 100 is class A, *RE* between 10 and 100 is class B, *RE* from 1 to 10 is class C and *RE* below 1 is class D. This is valid for all models, libraries, and platforms (see Fig. 6). As technology develops the thresholds for class A will be gradually increased potentially reaching 1000 in a few years from now. In order to support large number of uses for a given model the classification should be done on the lifecycle recognition efficiency RE_{LC}.

Author statement

Nicola Lenherr: Software, Formal Analysis, Investigation, Visualization

Rene Pawlitzek: Conceptualization, Editing, Project Administration, Supervision

Bruno Michel: Conceptualization, Methodology, Writing Original Draft, Funding Acquisition, Supervision, Resources.

Declaration of Competing Interest

The authors report no declarations of interest.

Acknowledgment

We acknowledge support from IBM Research Science & Technology.

References

[1] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, J. Kepner, Survey and benchmarking of machine learning accelerators, IEEE High Performance Extreme Computing Conf. (HPEC) (2019) arXiv:1908.11348v1.

[2] M. Horowitz, Computing's energy problem (and what We can Do about It), in: 2014 IEEE Int'l Solid-State Circuits Conf. Digest of Technical Papers (ISSCC), IEEE, 2014, pp. 10–14 [Online]. Available: http://ieeexplore.ieee.org/document/6757323/.

[3] J.L. Hennessy, D.A. Patterson, A new golden age for computer architecture, Comm. ACM 62 (2) (2019) 48–60. http://dl.acm.org/citation.cfm?doid=3310134.3282307.

[4] R. Schwartz, J. Dodge, N.A. Smith, O. Etzioni, Green AI, 2019 arXiv:1907.10597v3.

[5] Y. Shoham, et al., The AI index 2018 annual report, in: AI Index Steering Committee, Human-Centered AI Initiative, Stanford Univ, 2018. http://cdn.aiindex.org/2018/AI%20Index%202018%20Annual%20Report.pdf.

[6] E. Strubell, A. Ganesh, A. McCallu, Energy and Policy Considerations for Deep Learning in NLP, 2019. https://arxiv.org/abs/1906.02243v1.

[7] E. Cai, D.C. Juan, D. Stamoulis, D. Marculescu, NeuralPower: predict and deploy energy-efficient convolutional neural networks, Proc. Mac. Learning Res. 77 (2017) 622–637. ACML.

[8] Edge TPU, 2019 [Online], https://cloud.google.com/edge-tpu/.

[9] J. Hruska, New Movidius Myriad X VPU Packs a Custom NeuralCompute Engine, 2017.

[10] N.C. Thompson, K. Greenewald, K. Lee, G.F. Manso, The Computational Limits of Deep Learning, 2020 arXiv:2007.05558v1.

[11] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, J. Zhang, Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing, 2019 arXiv:1905.10083v1.

[12] X. Zhang, Y. Wang, S. Lu, L. Liu, L. Xu, W. Shi, OpenEI: An Open Framework for Edge Intelligence, 2019 arXiv:1906.01864v1.

[13] J. Konecny, H.B. McMahan, F.X. Yu, A.T. Suresh, D. Bacon, Federated Learning: Strategies for Improving Communication Efficiency, 2017 arXiv:1610.05492v2.

[14] H.B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. Arcas, Communication-efficient learning of deep networks from decentralized data, in: Proc. 20th Int'l Conf. Artificial Intelligence and Statistics (AISTATS), Fort Lauderdale, FL, 2017.

[15] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, A.Y. Zomaya, Edge Intelligence: The Confluence of Edge Computing and Artificial Intelligence, 2020 arXiv:1909.00560v2.

[16] A. Das, T. Brunschwiler, Privacy is what we care about: experimental investigation of federated learning on Edge devices, Proc. First Int'l Workshop Challenges in Artificial Intelligence and Machine Learning for IoT (2019) 39–42. Nov.

[17] D. Hernandez, Measuring the Algorithmic Efficiency of Neural Networks, 2020 and https://github.com/openai/ai-and-efficiency, https://arxiv.org/pdf/2005.04305.pdf.

[18] N. Guha, A. Talwalkar, V. Smith, One-Shot Federated Learning, 2017 arXiv:1902.11175v2.

[19] P.S. Chandakkar, Y. Li, Pak Lun, K. Ding, B. Li, Strategies for Re-training a pruned neural network in an Edge computing paradigm, IEEE 1st Int'l Conf. Edge Computing (2017).

[20] Y. Cheng, D. Wang, P. Zhou, T. Zhang, A survey of model compression and acceleration for deep neural networks, IEEE Signal Processing Magazine, Special Issue on Deep Learning for Image Understanding (2020).

[21] Z. Liu, M. Sun, T. Zhou, G. Huang, T. Darrell, Rethinking the value of network pruning, in: Int'l Conf. Learning Representations, New Orleans, Louisiana, US, 2019. May 6-9, 2019, ICLR.

[22] H.I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.A. Muller, Deep learning for time series classification: a review, Data Min. Knowl. Discov. 33 (2019) 917–963.

[23] F. Abramovich, M. Pensky, Classification With Many Classes: Challenges and Pluses, arXiv:1506.01567v4, 2021.

[24] E. Garcıa-Martin, N. Lavesson, H. Grahn, V. Boeva, Energy efficiency in machine learning: a position paper, in: JMLR: W&CP30th Annual Workshop of the Swedish Artificial Intelligence Soc, Volume. 54, 2017 arXiv:1602.05629v3.

[25] V. Smith, C.K. Chiang, M. Sanjabi, A. Talwalkar, Federated multi-task learning, in: 31st Conf. on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 2017.

[26] Z. Yang, M. Chen, W. Saad, C.S. Hong, M. Shikh-Bahaei, Energy Efficient Federated Learning Over Wireless Communication Networks, 2019 arXiv:1911.02417v1.

[27] A. Kasturi, A.R. Ellore, C. Hota, Fusion learning: a one-shot federated learning, in: ICCS 2020, LNCS 12139, 2020, pp. 424–436.

[28] Z. Xu, L. Li, W. Zou, Exploring Federated Learning on Battery-powered Devices, ACM TURC, Chengdu, China, 2019. May 17–19.

[29] D. Li, X. Chen, M. Becchi, Z. Zong, Evaluating the energy efficiency of deep convolutional neural networks on CPUs and GPUs, IEEE Int'l Conf. Sustainable Computing and Comm. (SustainCom) (2016). https://hpcn.exeter.ac.uk/sustainacom2020/?utm_source=researchbib.

[30] V. Sze, Y.H. Chen, J. Emer, A. Suleiman, Z. Zhang, Hardware for Machine Learning: Challenges and Opportunities, 978-1-5090-5191-5/17/$31.00@2017 IEEE, 2021.

[31] R. Xie, X. Jia, L. Wang, K. Wu, Energy efficiency enhancement for CNN-based deep mobile sensing, IEEE Wirel. Commun. (2019).

[32] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M.A. Horowitz, W.J. Dally, EIE: efficient inference engine on compressed deep neural network, ACM/IEEE 43rd Annual Int'l Symp. Computer Architecture (2016).

[33] RPi4: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/.

[34] Edge TPU: Coral DEV: https://coral.ai/docs/dev-board/datasheet/.

[35] NCS2: https://ark.intel.com/content/www/de/de/ark/products/140109/intel-neural-compute-stick-2.html.

[36] Jetson Nano: https://www.nvidia.com/de-de/autonomous-machines/embedded-systems/jetson-nano/.

[37] A. Oskooei, S. Mai-Chau, J. Weiss, A. Sridhar, M.R. Martınez, B. Michel, De-stress: Deep Learning for Unsupervised Identification of Mental Stress in Firefightersfrom Heart-rate Variability (hrv) Data, 2019 arXiv preprint arXiv:1911.13213.

[38] U. Pluntke, S. Gerke, A. Sridhar, J. Weiss, B. Michel, Evaluation and classifica-tion of physical and psychological stress in firefighters using heart rate variability, 41st Annual Int'l Conf. IEEE Engineering in Medicineand Biology Soc. (EMBC) (2019) 2207–2212.

[39] S. Klingner, Z. Han, Y. Liu, F. Fan, B. Altakrouri, B. Michel, J. Weiss, A. Sridhar, S. Mai-Chau, Firefighter Virtual Reality Simulation for Personalized Stress Detection, in: U. Schmid, F. Klügl, D. Wolter (Eds.), KI 2020: Advances in Artificial Intelligence. KI 2020. Lecture Notes in Computer Science, 12325, Springer, Cham, 2020, https://doi.org/10.1007/978-3-030-58285-2_32.

[40] S. Bianco, R. Cadene, L. Celona, P. Napoletano, Benchmark analysis of representative deep neural network architectures, IEEE Access (2018). https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8506339.

[41] T. Inoue, P. Vinayavekhin, S. Wang, D. Wood, N. Greco, R. Tachibana, Domestic activities classification based on CNN using shuffling and mixing data augmentation. Detection and Classification of Acoustic Scenes and Events, 2018.

[42] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.C. Chen, MobileNetV2: Inverted Residuals and Linear Bottlenecks, 2019. https://arxiv.org/pdf/1801.04381.pdf.

[43] K. He, X. Zhang, S. Ren, J. Sun, Deep Residual Learning for Image Recognition, 2015. https://arxiv.org/pdf/1512.03385.pdf.

[44] K. Simonyan, A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR, 2015. https://arxiv.org/pdf/1409.1556.pdf.

[45] D.P. Kingma, J. Ba, A Method for Stochastic Optimization. https://arxiv.org/abs/1412.6980.

[46] Joy-IT, Germany, https://joy-it.net/de/products/JT-UM25C).

[47] Rotkreuz, Switzerland, https://web.smart-me.com/en/project/smart-me-plug-2/.

[48] M. Hossin, M.N. Sulaiman, A review on evaluation metrics for data classification evaluations, Int'l. J. Data Mining Knowleged Manage. Proc. (IJDKP) 5 (2) (2015) 1–11.

[49] V.C. Coroama, L.M. Hilty, Assessing internet energy intensity: a review of methods and results, Environ. Impact Assess. Rev. 45 (2014) 63–68.

[50] H. Pihkola, M. Hongisto, O. Apilo, M. Lasanen, Evaluating the Energy Consumption of Mobile Data Transfer—From Technology Development to Consumer Behaviour and Life Cycle Thinking, Sustainability 10 (2018) 2494.

[51] M. Hodak, M. Gorvenko, A. Dholakia, Towards power efficiency in deep learning on data center hardware, IEEE Big Data (2019) conf.

[52] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, K. Kreutzer, ImageNet Training in Minutes, 2018. https://arxiv.org/abs/1709.05011.

[53] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, M. Jirstrand, A Performance Evaluation of Federated Learning Algorithms, DIDL, Rennes, France, 2018, 18, Dec. 10–11.

[54] M.J. Walker, S. Diestelhorst, A. Hansson, A.K. Das, S. Yang, B.M. Al-Hashimi, G. V. Merrett, Accurate and stable run-time power modeling for mobile and embedded CPUs, IEEE Trans. Comp.-Aided Des. Integr. Circuits Syst. 36 (1) (2017) 106–119.

[55] V. Sze, Y. Hsin, T.J. Yang, J.S. Emer, Efficient processing of deep neural networks: a tutorial and survey, Proc. IEEE 105 (12) (2017) 2295–2329.

[56] C. Kozyrkov, Artificial Intelligence: Do Stupid Things Faster With More Energy!, 2019. https://towardsdatascience.com/artificial-intelligence-do-stupid-things-faster-with-more-energy-379aa6bac220.

[57] C. Kozyrkov, Introduction to Decision Intelligence, 2019. https://towardsdatascience.com/introduction-to-decision-intelligence-5d147ddab767.

[58] Shankar, Structuring Your Machine Learning Projects, 2020. https://medium.com/structuring-your-machine-learning-projects/satisficing-and-optimizing-metric-24372e0a73c.

[59] H. Cai, C. Gan, T. Wang, Z. Zhang, S. Han, Once-for-all: Train One Network and Specialize It for Efficient Deployment, ICLR, 2020. https://arxiv.org/abs/1908.09791.

[60] B. Brevini, Black boxes, not green: mythologizing artificial intelligence and omitting the environment. Big Data & Society, 2020, pp. 1–5. July–December.