Uncovering Energy-Efficient Practices in Deep Learning Training
Preliminary Steps Towards Green AI

Yarally, Tim; Cruz, Luís; Feitosa, Daniel; Sallou, June; Van Deursen, Arie

DOI
10.1109/CAIN58948.2023.00012

Publication date
2023

Document Version
Final published version

Published in
Proceedings - 2023 IEEE/ACM 2nd International Conference on AI Engineering - Software Engineering for AI, CAIN 2023

2023 IEEE/ACM 2nd International Conference on AI Engineering – Software Engineering for AI (CAIN) | 979-8-3503-0113-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/CAIN58948.2023.00012

Uncovering Energy-Efficient Practices in Deep Learning Training: Preliminary Steps Towards Green AI

Tim Yarally*, Luís Cruz*, Daniel Feitosa† June Sallou*, Arie van Deursen*
*Delft University of Technology, The Netherlands - timyarally@hotmail.com, { l.cruz, j.sallou, arie.vandeursen }@tudelft.nl
†University of Groningen, The Netherlands - d.feitosa@rug.nl

***Abstract*— Modern AI practices all strive towards the same goal: better results. In the context of deep learning, the term "results" often refers to the achieved accuracy on a competitive problem set. In this paper, we adopt an idea from the emerging field of Green AI to consider energy consumption as a metric of equal importance to accuracy and to reduce any irrelevant tasks or energy usage. We examine the training stage of the deep learning pipeline from a sustainability perspective, through the study of hyperparameter tuning strategies and the model complexity, two factors vastly impacting the overall pipeline's energy consumption. First, we investigate the effectiveness of grid search, random search and Bayesian optimisation during hyperparameter tuning, and we find that Bayesian optimisation significantly dominates the other strategies. Furthermore, we analyse the architecture of convolutional neural networks with the energy consumption of three prominent layer types: convolutional, linear and ReLU layers. The results show that convolutional layers are the most computationally expensive by a strong margin. Additionally, we observe diminishing returns in accuracy for more energy-hungry models. The overall energy consumption of training can be halved by reducing the network complexity. In conclusion, we highlight innovative and promising energy-efficient practices for training deep learning models. To expand the application of Green AI, we advocate for a shift in the design of deep learning models, by considering the trade-off between energy efficiency and accuracy.**

***Index Terms*—green software, green ai, deep learning, hyperparameter tuning, network architecture**

I. Introduction

AI practices are expensive and can have a significant environmental impact. That is not surprising, since an important challenge within the AI community is improving the accuracy of previously reported systems [30]. Now, a new field is emerging to address this problem: Green AI, with its roots planted deep into the discipline of Sustainable Software Engineering. The software engineering community has increasingly studied the energy efficiency of software systems by developing energy estimation models [6], [25]; developing code analysis and optimisation tools to improve energy efficiency [2], [9], [11], [26]; studying practices that lead to green software [7], [10], [13] and so on. Recently, a new trend is calling for software engineering approaches that consider 'data as the new code', challenging practitioners with new software systems that ship AI-based features. This intersection between Green Software Engineering and AI Engineering is where we find the origin of Green AI. The initial contributions in this field consist of positional papers that are calling for a new research agenda [3], [30], [34]. Since then, the community has developed into studying the energy footprint of AI at different levels [37]. This involves the measurement and reporting of energy consumption [14] next to accuracy, but also the appreciation of research efforts that do not necessarily rely on enterprise-sized data [36] or training budgets.

This study focuses on deep learning, a subset of machine learning and the driver behind many AI applications and services. All experiments are performed with rudimentary neural networks that comprise the building blocks of more complex models. We train these networks on two popular image vision problem sets: FashionMNIST [40] and CIFAR-10 [21]. We adopt the idea of designing neural networks with energy consumption as one of the main considerations. Specifically, we direct our attention to the early phases of the deep learning pipeline and formulate the following research questions:

RQ_1: Between Bayesian optimisation, random optimisation and grid search; which strategy is the most energy-efficient for training a neural network?

RQ_2: Can the complexity of a neural network be reduced such that it consumes less energy while maintaining an acceptable level of accuracy?

First, we analyse Bayesian optimisation, random optimisation and grid search, three popular optimisation strategies, to identify best practices in terms of energy efficiency considerations. Classically, grid search has served as the most popular baseline optimisation strategy in the context of hyperparameter tuning [5]. Nonetheless, there have been studies that present random search as an alternative baseline that competes with or even exceeds grid search in multi-dimensional optimisation problems [4], [5], [24]. Bayesian optimisation is a more powerful strategy that is also more difficult to implement and parallelise. Apart from comparing these three strategies, we demonstrate that further optimisation attempts past a specific point are met with diminishing returns in performance that might not be worth the additional cost of training. Training times can vary greatly depending on the workload and network architecture and there are no rules that state how many optimisation rounds one should perform. This is where the

979-8-3503-0113-7/23/$31.00 ©2023 IEEE
DOI 10.1109/CAIN58948.2023.00012

potential opportunity for energy savings lies.

Secondly, we quantify the effect of a network's architecture in terms of layers, on the actual energy consumption of the GPU. In a similar fashion, we show how more complex models see diminishing returns in their performance, while the energy consumption keeps increasing at a steady rate. By analysing the accuracy of a neural network together with its energy consumption, the perspective of what is currently considered 'the best' model could see a dramatic shift. We believe it is the job of the Green AI community to make such data and observations available to the public so that software engineers can make more informed trade-offs, and more sustainable decisions.

II. Related Work

Given the particularities of different types of software systems, green software contributions span across multiple sub-fields of computer science: mobile computing [6], [10], Web [12], robotics [28], and so on. In our work, we challenge the green software engineering field to expand to AI systems. To the best of our knowledge, related research in Green AI is still preliminary and does not yet follow the scientific method that drives the research in green software. We pinpoint below the most relevant related contributions in Green AI.

Schwartz et al. [30] present an elegant introductory article into this field of research. The authors introduce two novel terms to guide future conversations: Green AI, which refers to AI research that considers computational cost as a primary metric next to accuracy; and Red AI, the most common form of AI research that seeks to improve accuracy without any regards for the computational resources required. Ultimately, Schwartz et al. call for a research agenda that aims to reduce carbon emissions and make the deep learning field more accessible to everyone. Our work takes a preliminary step towards this goal, by presenting empirical results focused on different parts of the training pipeline that can lead to energy-efficiency gains on a larger scale.

Strubell et al. [34] look into the quantity of energy consumption in the domain of Natural Language Processing (NLP). The authors present preliminary results showing that the accuracy of trained NLP models has improved substantially at the expense of a serious amount of energy. Our study aims to provide scientifically proven advice to help design energy-efficient AI systems, including NLP.

Li et al. [23] address a similar problem specifically targeted towards applications of convolutional neural networks (CNNs). In their work, the authors compare a set of well-known CNN models in terms of energy efficiency. They also assess to what degree the different types of layers contribute to the overall energy consumption. As such, they provide percentages for the convolutional layers, fully connected (or linear) layers, pooling layers and ReLU layers. Apart from finding the energy efficiency of these layer types, our study also includes a trade-off analysis where we compare energy consumption to accuracy.

Yang et al. [41] propose a new pruning method, named energy-aware pruning, that removes layers' weights to reduce the energy consumption. They report a reduction in the overall consumption by a factor between 1.6x and 3.7x, with an insignificant loss in accuracy. Our approach to optimise the network architecture is much more coarse-grained compared to the techniques described in this paper. Rather than focusing on fine-tuning the weights inside the layers, we investigate the factor of redundancy of those layers and provide advice that is more relevant from a design-level perspective. Both philosophies could be used together.

Two other related studies examine the trade-off between energy-efficiency and accuracy, either regarding the learning frameworks PyTorch and TensorFlow during training and inference [15], or the solvers used for the training of logistic regression models [16]. For the framework, TensorFlow is more energy and run-time efficient for training, while Pytorch is the best for the inference stage. As for the solver, LBFGS is shown to be more energy-efficient than Newton-CG and SAG. Moreover, these works demonstrate that prioritizing the energy efficiency does not necessarily reflect negatively on the accuracy of the model. In our study, we share the same view that opting for considering energy efficiency while designing the deep learning models does not impair their accuracy in a substantial way. Instead of examining that trade-off regarding the framework or logistic regression models, we inspect the deep learning training practices in more details, focusing on hyperparameter tuning and the network architecture.

III. Background

This section introduces the FashionMNIST and CIFAR-10 datasets; elaborates on grid search, random search and Bayesian optimisation and establishes a basic understanding of linear, convolutional and ReLU layers inside a neural network.

A. Datasets

The original MNIST dataset consists of many grey-scale images of handwritten digits. MNIST has been used excessively as a benchmark to validate many different models. However, with modern technology, the MNIST problem set has become too trivial. Because most networks can achieve near-perfect accuracy on the set, researchers from Zalando have proposed the use of **FashionMNIST** as a direct drop-in-replacement [40]. As such, the dataset is comprised of 28×28 grayscale images of 70,000 different fashion products. Just like in the original MNIST set, the products are separated into 10 categories. Because both datasets are shaped identically, FashionMNIST is immediately compatible with any machine learning package that works with MNIST.

The **CIFAR-10** dataset is a subset of the tiny images dataset [35]. It is composed of 60,000 32×32 RGB images divided into 10 classes [21]. CIFAR-10 presents a challenge that is very similar to FashionMNIST, however, the larger image size and two additional layers increase the complexity of the task significantly.

B. Optimisation Strategies

Grid search is a traditional optimisation strategy that applies an exhaustive search over the hyperparameter space. For

discrete variables, this means that the algorithm considers the Cartesian product of all the values. For continuous variables, it is necessary to select a distribution first. One could for example choose a uniform or log-uniform distribution to map the continuous space to a discrete one. The computational complexity of grid search is exponential in the number of parameters, therefore it quickly becomes impractical to calculate it all the way through. Nevertheless, because the search space is determined at the beginning, the workload can very easily be parallelised, somewhat offsetting this drawback.

In a **random search**, the well-defined structure of the grid is replaced by random selection. Because every drawn sample is completely independent, parallelisation of this algorithm is as trivial as with grid search.

In the context of hyperparameter tuning, the **Bayesian optimisation** algorithm creates and refines a probabilistic regression model of a function $f(x)$ that can be exploited to return the predicted accuracy and corresponding standard deviation. An acquisition function is then used to determine the next most promising set of input variables. For the probabilistic model, Gaussian Processes (GP) are the most popular choice amongst many studies [20], [32], [39]. Bayesian optimisation is especially effective in scenarios where the true value of $f(x)$ is hard to compute, which is the case with neural network training. This is because the probabilistic model needs to evaluate a sufficiently large quantity of samples.

The purpose of the acquisition function is to determine the most promising sample from a set of randomly selected input variables. There are many possible choices that can be considered:

- Probability of Improvement (PI) [22]
- Expected Improvement (EI) [29]
- Upper Confidence Bound (UCB) [33]
- Entropy Search (ES) [17]
- Predictive Entropy Search (PES) [18]
- Knowledge Gradient (KG) [31]

We now elaborate on the PI acquisition function, which is also the function that we use in all of the experiments.

$$PI(x) = P(f(x) \geq f(x_{best})) = \Phi(\frac{\mu(x) - f(x_{best})}{\sigma(x) + \epsilon}) \quad (1)$$

$$PI(x) = P(f(x) \leq f(x_{best})) = \Phi(\frac{f(x_{best}) - \mu(x)}{\sigma(x) + \epsilon}) \quad (2)$$

Equations 1 and 2 are used to calculate the probability of improvement for maximisation and minimisation problems respectively. Since we are interested in maximising the accuracy of a neural network, we will use Equation 1. Here, Φ refers to the cumulative density function of a normal distribution; μ and σ are the predicted value and standard deviation retrieved from the Gaussian regressor, and $f(x_{best})$ is the highest actual accuracy found so far. Algorithm 1 displays an example implementation of a single PI iteration.

Algorithm 1 Bayesian Optimisation - Probability of Improvement

$y = max(Y)$
$Candidates \leftarrow N$ random input samples
x', pi'
for $x \in Candidates$ **do**
  $\mu, \sigma = predict(x)$
  $pi = \Phi(\frac{\mu - y}{\sigma + \epsilon})$
  if $pi > pi'$ **then**
    $x' = x, pi' = pi$
  end if
end for
$X \leftarrow x', Y \leftarrow f(x')$
$fit(X, Y)$

C. Neural Network Layers

Every layer inside a neural network performs some transformation on an input vector x. The obtained output is then passed on to the next layer. **Linear**, or **fully connected layers**, calculate an output by applying a linear transformation through a matrix of weights W [27]. The values of W are optimised and updated during training. The term fully connected comes from the fact that every element of x is mapped to every other element in the output by the matrix multiplication $W^T x$.

Inside a **convolutional layer**, a kernel is used to calculate a weighted summation of the elements of the input layer. The kernel slides across the input layer, considering all elements and their neighbours. A convolutional operation is defined by stride, kernel size and zero padding [1]. The stride determines how many places the kernel slides after each calculation; the kernel size represents the dimensions of the filter and zero padding adds zeros to the outer edges of the input layer. Generally speaking, the output layer is always smaller than the input layer, limiting the maximum number of convolutional layers that can be implemented. However, by applying zero padding, one can prevent this shrinking behaviour if desired. The convolution operation is shown graphically in Figure 1.

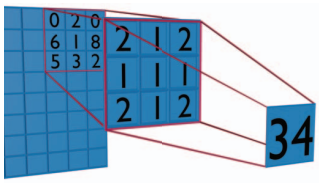


Fig. 1: Convolution operation on an input layer using a 3×3 kernel.

The **Rectified Linear Unit (ReLU) layer** introduces an activation function that applies non-linearity to the input. ReLU is the most common form of non-linearity in CNNs [1]. The function is very simple: An element is deactivated (set to 0) if it is negative; otherwise, the value remains the same.

IV. Research Methods

The goal of this study is to identify trade-off points with respect to the energy consumption during the training phase of the deep learning pipeline.

A. Case Selection

Achieving state-of-the-art accuracy results on challenging data sets is not the main focus. For this reason, we will be working with rudimentary networks architectures that can be trained using consumer-grade hardware. The simplicity of the models facilitates the design of more intricate experimentation and encourages inclusivity. We choose to direct our efforts to image recognition problems. Image recognition is a canonical problem that can be solved with neural networks, and there are a plethora of easily accessible data sets available.

The experiments are performed using three neural networks written with the PyTorch framework[1]. These networks are trained on a single GeForce GTX-1080[2] GPU with images from the FashionMNIST and CIFAR-10 datasets. During every **optimisation round** mentioned in this study, an optimisation algorithm chooses a set of hyperparameter values. An optimisation round lasts for 8 repetitions, during which we use the same set of hyperparameters. A single repetition consists of 25 training epochs[3]. After the 8 repetitions, the highest accuracy and average energy consumption are logged and a new optimisation round starts. We present this experimental design schematically in Figure 2. The structure of the different networks is as follows:

- **DenseLinearNN**: N linear layers, where the number of neurons in each layer scales down linearly towards the number of problem classes.
- **DensePolyNN**: N linear layers, every layer has half the number of neurons as the layer before it.
- **SimpleCNN**: M convolutional layers, each followed by a BatchNorm2d, ReLU and MaxPool2d layer, and N linear layers where every layer has half the number of neurons as the layer before it.

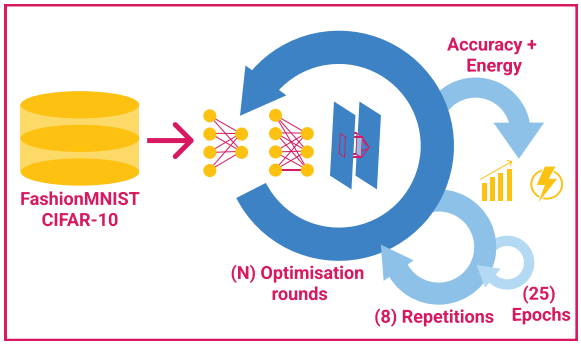


Fig. 2: Methodology process

B. Experimental Tooling

To facilitate and standardise the data collection, we develop a test suite that automates the execution of the experiments. This test suite is available online[4] and contains the implementations of the aforementioned neural networks that can be trained on all the visual problem datasets provided by Pytorch[5]. The test bed is designed to be modular, and we encourage other researchers to add additional neural network designs or different hyperparameter optimisation functions. All the results in this study have been accumulated with this test bed.

C. Data Collection

To answer RQ_1, we compare the convergence rate of three different hyperparameter tuning strategies: Grid search, random optimisation and Bayesian optimisation with the PI acquisition function. Because grid search is an exhaustive method, it quickly becomes infeasible to train a network on every hyperparameter set. To fairly compare grid search to the other strategies, we first generate the complete search space and then proceed to pick random samples from that space until we reach the desired amount of optimisation rounds. Variables with continuous ranges are divided into five uniformly-distributed values. Although this is a partial grid search, the most important difference with random search remains intact: because we select samples from the grid, a limited number of values are considered for every hyperparameter.

For RQ_2, we examine the effect of the neural network's architecture on the absolute energy consumption. To obtain the power usage of the GPU, we query the NVIDIA System Management Interface[6] every 100 milliseconds. We use this to compute the total energy consumption of a training iteration and then factor out the idle energy consumption of the GPU.

D. Data Analysis

To assess the effectiveness of the hyperparameter tuning strategies, we study the convergence rate of the model accuracy according to the number of optimisation rounds. We determine the optimum accuracy as the highest accuracy after which no substantial increase for each additional optimisation round is observed. Similarly, we define the optimum round as the number of the optimisation round when the optimum accuracy is reached. By identifying those values, we can ascertain the most efficient strategy, and establish the optimal number of optimisation rounds sufficient to provide the model with optimum accuracy.

To accurately analyse the effect of the neural network architecture in relation to the energy consumption, first, we would like to show that the hyperparameter set does not contaminate the results. We do so by calculating the coefficient of variance (CV) of the energy consumption and showing that it is very low (< 0.01). The CV is calculated as the standard deviation of a sequence divided by its average.

Prior to any further in-depth analysis, we need to assess whether the energy results obtained in the experiments follow a normal distribution. After a visual inspection of the quantile-quantile (Q-Q) plot, followed by the Shapiro-Wilk test[7], we

[1]https://pytorch.org
[2]https://www.nvidia.com/nl-nl/geforce/10-series/
[3]During one optimisation round with eight repetitions, a network undergoes 25×8 training epochs with the same set of hyperparameters
[4]https://zenodo.org/record/7767313
[5]https://pytorch.org/vision/0.8/datasets.html
[6]https://developer.nvidia.com/nvidia-system-management-interface
[7]https://www.statskingdom.com/shapiro-wilk-test-calculator.html

conclude that our data is not normally distributed. Hence, we opt for a non-parametric analysis and apply the Kruskal-Wallis test to indicate the significance of our independent variables, i.e. whether we may conclude that the layer types have a statistically meaningful impact on the energy consumption. Additionally, we calculate the η^2 as the effect size. We evaluate these effect sizes based on the rules of thumb for Cohen's f [8], which is calculated as $f = \sqrt{\frac{\eta^2}{1-\eta^2}}$. Cohen suggests that the values 0.10, 0.25 and 0.40 convey a small, medium and large effect size respectively. We invert the function to obtain the effect thresholds for η^2: 0.01, 0.06 and 0.14.

V. Experiments

In this section, we present the design of two different experiments, each related to one of the research questions. Section V-A describes the experiment to compare the hyperparameter optimisation strategies. The second experiment, to investigate the relationship between neural network architectures and energy consumption, is described in Section V-B.

A. Hyperparameter Optimisation

Given that the response function of a hyperparameter optimisation problem $f(x_1, ..., x_n)$ has a *low effective dimensionality* [5], meaning that the function can be approximated by another function $g(x_1, ..., x_{n-i})$ with less variables, the hypothesis for RQ_1 is that random search will converge faster than grid search, because it does not consider two identical values more than once. Given enough time, Bayesian optimisation should outperform the other two strategies. However, with a limited run budget, we might observe that the Bayesian strategy performs worse because it chooses to exploit suboptimal solutions rather than explore better ones.

The setup of the experiment, as is depicted in Table I, involves 18 different configurations. Each optimisation strategy is applied twice to the DensePolyNN, DenseLinearNN and SimpleCNN mentioned in section IV. The *hyperparameters* column in Table I shows how many parameters are optimised during a run. The five hyperparameters refer to the learning rate (α), betas (β_1, β_2), epsilon (ϵ) and weight decay (w) of the ADAM optimiser provided by PyTorch[8]. The entire experiment is repeated for both the FashionMNIST and CIFAR-10 datasets.

For every row in Table I, a network is trained on 64 different hyperparameter settings with 8 repetitions for each setting, amounting to 512 training iterations. After each set of repetitions, the optimisation function provides a new set of values for the hyperparameters. A trained model is evaluated and the results are logged. In total, we run 18 configurations $\times$ 64 optimisation rounds $\times$ 8 repetitions $\times$ 2 data sets $=$ 18,432 training iterations.

B. Network Architecture

The second experiment aims to answer RQ_2 by collecting empirical data that shows the relationship between the structure of a neural network and its energy consumption. We present a full factorial design in Table II. The results of this experiment highlight the energy efficiency or lack thereof for the linear, convolutional and ReLU layers. The interesting point for discussion will be whether reducing the network complexity has a significant, positive influence on the energy efficiency, without too heavily compromising on the accuracy.

[8]https://pytorch.org/docs/stable/generated/torch.optim.Adam.html

TABLE I: Comparison of optimisation strategies.

Strategy	Network	hyperparameters
Bayesian	DensePolyNN	α, β_1, β_2
	DenseLinearNN	α, β_1, β_2
	SimpleCNN	α, β_1, β_2
	DensePolyNN	$\alpha, \beta_1, \beta_2, \epsilon, w$
	DenseLinearNN	$\alpha, \beta_1, \beta_2, \epsilon, w$
	SimpleCNN	$\alpha, \beta_1, \beta_2, \epsilon, w$
Random	DensePolyNN	α, β_1, β_2
	DenseLinearNN	α, β_1, β_2
	SimpleCNN	α, β_1, β_2
	DensePolyNN	$\alpha, \beta_1, \beta_2, \epsilon, w$
	DenseLinearNN	$\alpha, \beta_1, \beta_2, \epsilon, w$
	SimpleCNN	$\alpha, \beta_1, \beta_2, \epsilon, w$
Grid	DensePolyNN	α, β_1, β_2
	DenseLinearNN	α, β_1, β_2
	SimpleCNN	α, β_1, β_2
	DensePolyNN	$\alpha, \beta_1, \beta_2, \epsilon, w$
	DenseLinearNN	$\alpha, \beta_1, \beta_2, \epsilon, w$
	SimpleCNN	$\alpha, \beta_1, \beta_2, \epsilon, w$

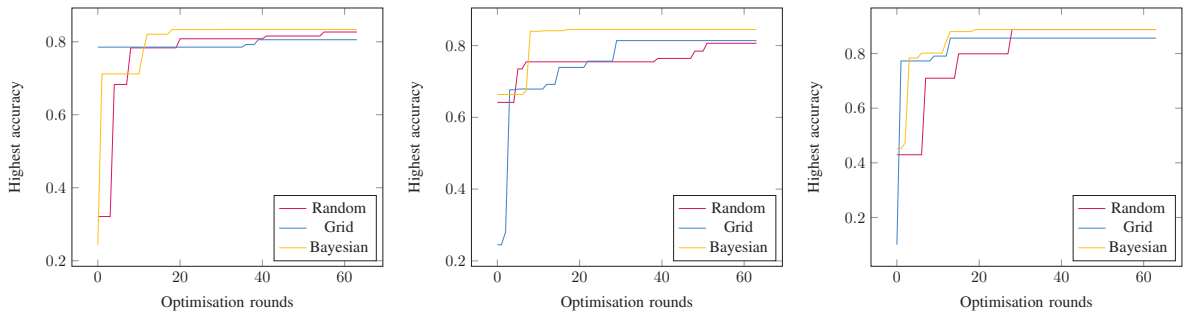
TABLE II: Configurations of the model architecture.

Linear layers	Convolutional layers	ReLU layers
3	1	0
3	1	1
3	4	0
3	4	4
7	1	0
7	1	1
7	4	0
7	4	4

For every row in Table II, the SimpleCNN model from Section IV is trained on 8 different hyperparameter settings with 24 repetitions for each setting, using the random optimisation strategy. Again, the experiment is repeated for both the FashionMNIST and CIFAR-10 datasets. Because accuracy is not the main metric for this experiment, we are less interested in finding different hyperparameter settings as opposed to the first experiment. For this reason, we reduce the number of optimisation rounds and increase the number of repetitions. In total, we run 8 configurations $\times$ 8 optimisation rounds $\times$ 24 repetitions $\times$ 2 data sets $=$ 3072 training iterations.

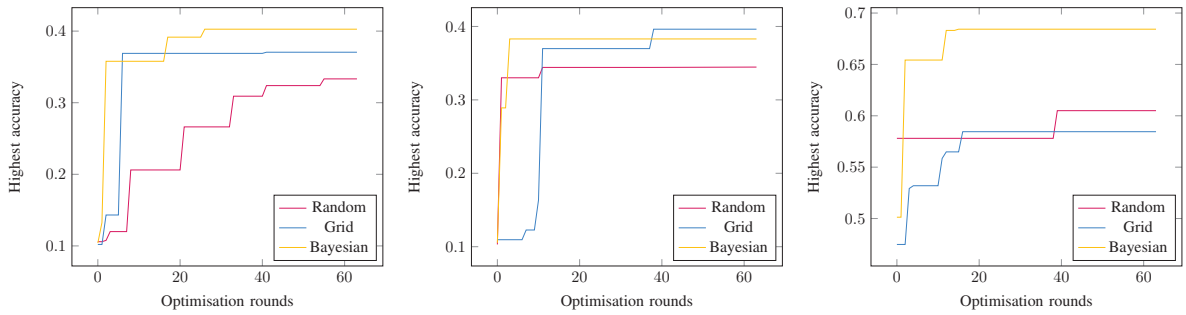
VI. Results

In this section, we report the results of the experiments formulated in Sections V-A and V-B.



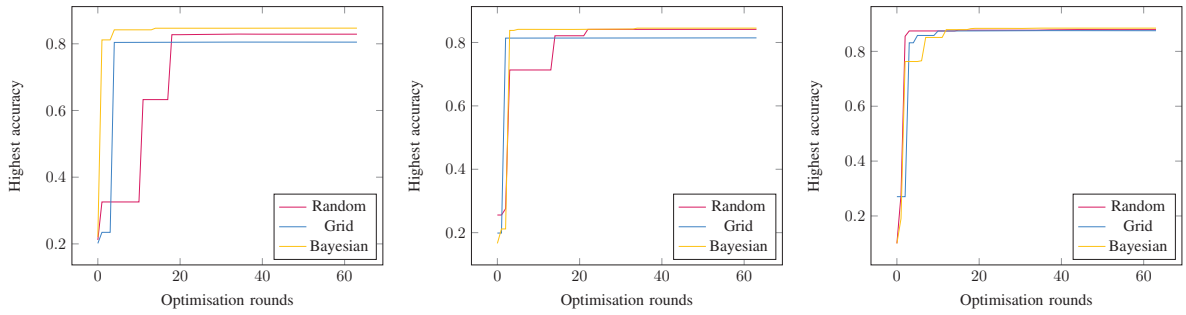
(a) DensePolyNN (b) DenseLinearNN (c) SimpleCNN

Fig. 3: Convergence graphs for the hyperparameter optimisation experiment with 5 parameters on FashionMNIST



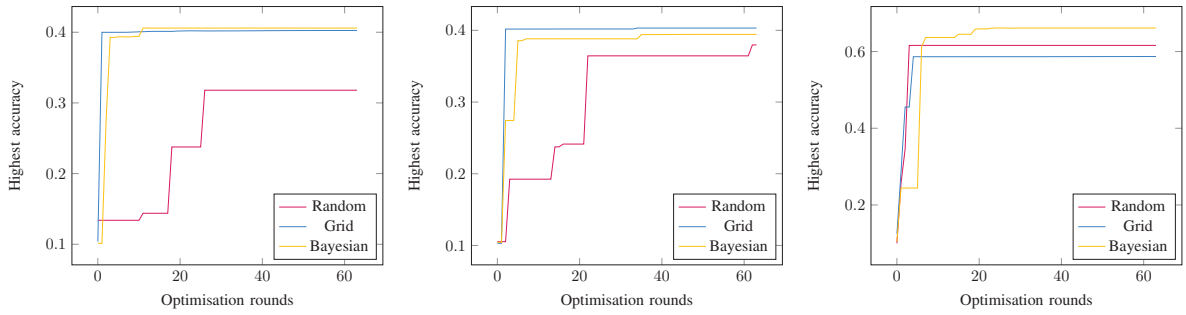
(a) DensePolyNN (b) DenseLinearNN (c) SimpleCNN

Fig. 4: Convergence graphs for the hyperparameter optimisation experiment with 5 parameters on CIFAR-10



(a) DensePolyNN (b) DenseLinearNN (c) SimpleCNN

Fig. 5: Convergence graphs for the hyperparameter optimisation experiment with 3 parameters on FashionMNIST



(a) DensePolyNN (b) DenseLinearNN (c) SimpleCNN

Fig. 6: Convergence graphs for the hyperparameter optimisation experiment with 3 parameters on CIFAR-10

A. Hyperparameter Optimisation

The line graphs in Figures 3 and 4 show the highest achieved accuracy by the number of optimisation rounds for all the settings with 5 hyperparameters (i.e., $\alpha, \beta_1, \beta_2, \epsilon$ and w) on both the FashionMNIST and CIFAR-10 datasets. Figures 5 and 6, display the results for all settings with 3 parameters (i.e., α, β_1 and β_2). The figures are separated into subfigures to distinguish between the results for the DensePolyNN (a), DenseLinearNN (b) and SimpleCNN (c) that were introduced in Section IV. The total runtime of the hyperparameter optimisation experiment (Section V-A) amounts to ± 85 hours.

With an initial visual assessment, a few observations can be made. First, Bayesian optimisation proves to be the most effective strategy when compared with random and grid search. Regardless of the network or workload, it consistently outperforms the other strategies, only being overtaken slightly by grid search twice (4b and 6b) and narrowly matched by random search three times (3c, 5b and 5c). Second, between grid search and random search, there is no definitive winner. Random search performed better than grid search 5 out of 6 times on the FashionMNIST dataset and 2 out of 6 times on CIFAR-10. We have also summarised this data in Table III. This table presents the optimum accuracy (cf. Section IV-D) for every experimental configuration (i.e. network $\times$ optimisation strategy $\times$ dataset $\times$ #hyperparameters) together with the number of optimisation rounds it took to achieve that accuracy.

Finally, notice that the Bayesian optimisation strategy converges to an accuracy optimum within 27 optimisation rounds on average. The two outliers with regard to this rule are marked by an asterisk (*) in Table III. Nonetheless, a quick inspection of the corresponding graphs (5a & 5b) shows that there is only a very slight increase compared to the accuracy that was achieved after 27 optimisation rounds. The same cannot exactly be said for random optimisation. Most of the graphs follow a much more gradual incline with bigger jumps in accuracy. Overall, this strategy takes longer to converge. Grid search, on the other hand, does seem to converge rapidly. The numbers in Table III might suggest otherwise, but similar to what we observe with Bayesian optimisation, the increases in accuracy past 27 optimisation rounds are minimal.

B. Network Architecture

The total runtime for all the different configurations of the network architecture experiment (Section V-B) approximately amounts to 46 hours. The purpose of this experiment is to quantify the relationship between the network architecture and the amount of energy that is being consumed during training.

To reinforce the validity of our results, we first show that the values of the hyperparameters, as chosen by the random optimisation function, do not significantly impact the energy consumption. The coefficient of variance (CV) is a metric that explains the relative size of the standard deviation to the mean. Because we assume that the hyperparameter setting has little to no influence on the energy consumption, we expect a very small CV ($< 1\%$) for all the optimisation rounds of a network. The histogram in Figure 7 depicts the CVs for every row in Table II on both the FashionMNIST and CIFAR-10 datasets. Every data point is a calculation of 8 optimisation rounds, including 24 repetitions. We find an average CV of 0.009 and a maximum value of 0.018.

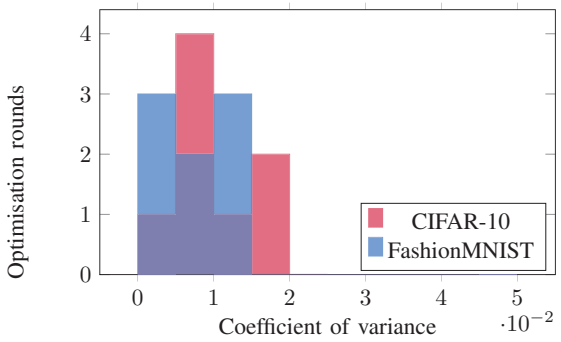


Fig. 7: Histogram of the coefficient of variance for each run on the CIFAR-10 and FasionMNIST datasets

TABLE III: Summary of the results for the optimisation experiment. Values are reported as x | y, where x represents the results with 5 hyperparameters (i.e. $\alpha, \beta_1, \beta_2, \epsilon, w$), and y those with 3 (i.e. α, β_1, β_2). In bold are the values of the highest accuracy or lowest number of optimisation rounds among the three strategies for each network and dataset.

		CIFAR-10		FashionMNIST	
		Accuracy	Optimisation rounds	Accuracy	Optimisation rounds
DensePolyNN	Random	0.33 | 0.32	56 | 27	0.826 | 0.83	56 | 27
	Grid	0.37 | 0.40	42 | 55	0.81 | 0.81	40 | **19**
	Bayesian	**0.40** | **0.41**	**27** | **11**	**0.833** | **0.85**	**29** | 46*
DenseLinearNN	Random	0.35 | 0.38	50 | 63	0.81 | 0.84	52 | **23**
	Grid	**0.40** | **0.40**	53 | **35**	0.81 | 0.81	30 | 54
	Bayesian	0.37 | 0.39	**4** | 38	**0.84** | **0.85**	**18** | 47*
SimpleCNN	Random	0.60 | 0.62	40 | **4**	**0.8879** | 0.879	29 | 39
	Grid	0.58 | 0.59	17 | 55	0.86 | 0.875	**14** | **20**
	Bayesian	**0.68** | **0.66**	**16** | 26	0.8876 | **0.884**	20 | 36

Now that we have shown that the energy consumption of a training iteration is independent of the hyperparameter settings in this experiment, we can analyse the network architecture in isolation. Because the data is not normally distributed, a conclusion made following the procedure described in Section IV-D, we perform the non-parametric Kruskal-Wallis test to identify if the energy consumed to train the network architectures can be distinguished statistically. Table IV presents the corresponding p-values and effect sizes (η^2). Notice that out of the three layer types, convolutional layers and linear layers have a large degree of influence on the energy consumption, while the influence of ReLU layers is small. An additional post hoc comparison shows that all combinations of independent variables are significant as well. To put these statistics into perspective, we compare the increase in average energy consumption by fixing each layer type. We use the notation $x|y$ to distinguish results on the FashionMNIST dataset (x) from those on the CIFAR-10 dataset (y). The presence of ReLU layers contributes an average increase of $2.7\%|2.9\%$. For the linear layers, the jump from 3 to 7 layers accounts for an increase of $4.9\%|6.6\%$. The convolutional layers are the largest sources of energy usage. Introducing 3 additional layers on top of the first one increases the overall consumption by $95.3\%|66.4\%$.

TABLE IV: Kruskal-Wallis test results. From top to bottom, the tables refer to the experiments on the FashionMNIST and CIFAR-10 datasets respectively.

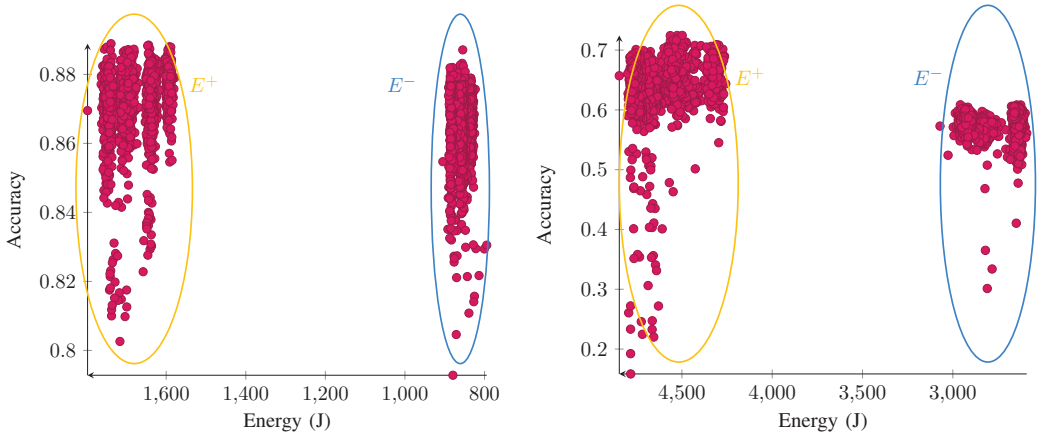
Factor (layer type)	Statistic	p	η^2	magnitude
Linear	481.799	$< .001$	0.155	large
Convolutional	2303.250	$< .001$	0.749	large
ReLU	176.545	$< .001$	0.055	small
Factor (layer type)	**Statistic**	**p**	**η^2**	**magnitude**
Linear	496.807	$< .001$	0.160	large
Convolutional	2303.250	$< .001$	0.749	large
ReLU	106.655	$< .001$	0.033	small

Moreover, we carry out a trade-off analysis with respect to the energy consumption of a neural network and its achieved accuracy on the problem set. This comparison is visualised in Figure 8. The scatter plots in this figure highlight the relationship of the energy consumption in Joules and the achieved accuracy on the test sets of FashionMNIST (a) and CIFAR-10 (b). In both scatter plots, we can discern two clusters; one spread around a higher energy consumption which we will refer to as E^+; the other spread around a lower energy consumption, we call this cluster E^- (notice that the energy axis is reversed). All data points in E^- correspond to network architectures with a single convolutional layer, while the E^+ cluster contains all the networks with four convolutional layers.

Table V summarises the data from the scatter plots into numerical values. The first two columns show the average energy consumption, average accuracy, maximum accuracy and the standard deviation of the accuracy for the E^+ and E^- clusters on the FashionMINST dataset. The latter two columns show the same information on the CIFAR-10 set. Notice that the average and maximum accuracy for both clusters on the FashionMNIST dataset are particularly close together, only varying by less than 1%. For CIFAR-10, which is a more computationally complex set, this difference is more significant. A little over 6% for the average and almost 12% for the maximum accuracy.

TABLE V: Low energy performance compared against high energy performance.

	FashionMNIST		CIFAR-10	
	E^+	E^-	E^+	E^-
Average energy	1674 J	857 J	4588 J	2758 J
Average accuracy	0.872	0.864	0.639	0.572
Max accuracy	0.889	0.887	0.725	0.609
Std accuracy	0.011	0.010	0.056	0.021



(a) FashionMNIST (b) CIFAR-10

Fig. 8: Scatter plots of the energy consumption vs the achieved accuracy

VII. Discussion

This empirical study aims to provide insights into possible improvements for deep learning pipelines out of environmental considerations. In this section, we answer both research questions by analysing the results of the experiments.

A. Hyperparameter Optimisation

The conclusion to RQ_1 is that *Bayesian optimisation* is the most energy-efficient strategy during the training of a machine learning model. Out of all three strategies, Bayesian optimisation consistently finds hyperparameter sets that result in the highest accuracy and it does so within the least amount of optimisation rounds ($\pm$27). Because this strategy requires the storage and constant fitting of a probabilistic model, one downside is the difficulty of parallelisation, but even that is not impossible [32]. Based on our results, there seems to be no good argument to choose one of the other methods.

Nevertheless, we cannot deny the presence of grid search and random search within the deep learning field. Both algorithms are easy to understand and implement, and could serve a purpose during early exploration or calibration. In this context, does one of the algorithms dominate the other? Solely based on our results, we cannot make any decisive claims. We can, however, assess the practicality of both solutions. Grid search is an exhaustive method with a search space that increases exponentially by the number of hyperparameters. Considering every set in that search space is not feasible and goes against our philosophy of energy-efficient training. As a consequence, we can only consider a portion of the complete search space, which defeats the purpose of the grid search. By randomly selecting samples from the search space, grid search devolves into a random search with a finite number of options. Furthermore, as Bergstra and Bengio [5] explain: hyperparameter optimisation problems in high-dimensional spaces have a low effective dimensionality. What this entails in our context is that some parameters will have a much larger influence on the accuracy than others. Figure 9 illustrates how random search exploits this property more effectively than grid search. The cubes in the image represent a three-dimensional problem where only one parameter has a significant influence on the function value. With the grid search (left), although we consider 27 distinct samples, only 3 values of the important parameter are tested. On the contrary, the random search (right) tests a new value for every sample.

Furthermore, random search facilitates the job of AI engineers as it does not require any human guidance apart from selecting the bounds. For these reasons, we recommend the use of random search over grid search for the early stages of the training. Our results show that random search is not worse than grid search for problems with ≤ 5 hyperparameters. Additionally, random search should remain a valid baseline strategy with an increasing number of parameters, while grid search will fall short due to the expanding search space.

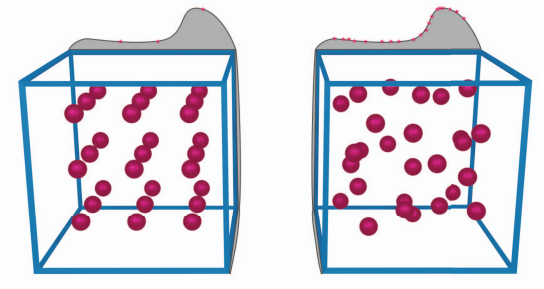


Fig. 9: Grid search (left) vs Random search (right) on a problem with a low effective dimensionality.

B. Network Architecture

To answer RQ_2: reducing the layer complexity of a neural network is a valid option to lower the energy consumption of a deep learning pipeline. Besides computer vision, we believe that our results can be generalised to other fields such as speech recognition or natural language processing as well. The computational complexity of different layer types is a constant that will present similar effects on the energy consumption in a different context. The observation of diminishing performance is also not a very bold claim. However, whether the loss in accuracy resulting from architecture simplifications is acceptable, depends largely on the context. As the visuals in Figure 8a and the corresponding summarised data in Table V make apparent, the accuracy gain for introducing complexity on a relatively simple problem (FashionMINST) is very small. In this case, we would argue that the diminishing return in accuracy is not worth doubling the number of expended Joules. For the CIFAR10 problem set, although there are still diminishing returns, the difference in accuracy we observe is quite significant. Ultimately, what it comes down to is how much error is acceptable for the application in question. To aid this decision, it is important that researchers monitor the performance slope of their model and that they report some metric that relates to the energy efficiency throughout the pipeline, such as Joules or FLOPs. By combining the accuracy and energy trends, we can make more considerate design choices, reduce the layer complexity, and improve the efficiency of the pipeline. As we have shown, these changes could lower the overall energy consumption of a training pipeline by half (i.e., increasing the number of layers from 1 to 4 leads to a 95% energy consumption increase). Many state-of-the-art models could also benefit from this philosophy. Following the current trend, new models are becoming exponentially larger and more costly, while the performance only sees marginal increases. If all these models would also report their energy consumption, it would vastly change the perspective of which one is 'the best' and give rise to new research efforts that focus on energy-efficient design.

C. Extra: optimising GPU load

While collecting energy measurements and analysing results, we were faced with a natural follow-up question: do

we really need to measure energy consumption or could we simply rely on time efficiency? While looking at our data, we noticed that different experiments yield a different GPU load. Hence, a model that trains faster might be using the GPU more efficiently, but one cannot immediately draw conclusions w.r.t. the pipeline's total energy consumption.

Nevertheless, we know from previous research that AI systems that optimise GPU usage can reduce their energy consumption by a factor of 10 [38]. When we look at our experiments, model training resulted in a GPU load that ranged between 40% and 60%. Similar values have been reported by a study conducted at Facebook AI [38]: a large portion of machine learning model experimentation only utilise their GPUs at 30–50%. This shows that an important step for Green AI engineering is to monitor and optimise GPU acceleration in pipelines. One should also consider the embodied carbon from the GPU hardware. This is a serious problem because underutilising models require more GPUs than what should be theoretically sufficient [42].

Hence, we argue that AI frameworks ought to feature GPU-enabled operations out of the box. Tools should be improved to support both AI practitioners and software engineers to monitor and optimise the GPU usage of their AI systems and pipelines. Developing AI systems is already a transdisciplinary field that requires expertise across different domains. Therefore, making energy consumption a first-class citizen for designing these systems will facilitate communication across disciplines and boost Green AI efforts substantially.

VIII. Implications

In this section, we highlight the implications for different stakeholders in AI systems.

Implications to AI Practitioners. Practitioners should be aware of the differences between Green AI and Red AI and the energy-efficient practices that we have laid bare in this study. As such, when developing and tuning new deep learning models, developers should look beyond the realm of baseline optimisation strategies and opt for more advanced techniques such as Bayesian optimisation. Another valid approach is to outsource this part of the training pipeline and implement existing solutions such as the population-based training algorithm from Ray Tune[9].

Implications to Software Engineers. Software engineers are already incorporating transdisciplinary AI teams to enable the productionisation of AI models. We argue that the role of software engineers is quintessential to enable energy-aware AI pipelines. One cannot ask regular AI practitioners to engineer the collection of energy-efficiency metrics – it is important that software engineers have the right knowledge and experience to help include energy as an important factor when developing AI pipelines.

Implications to AI tool developers. Our results show that AI frameworks have to provide green alternatives. For example, there are not many options when selecting hyperparameter strategies. Moreover, there is no information about the energy efficiency of these alternatives. Hence, our results call for more energy-efficient options and better documentation with sustainability tips, in line with previous findings in Green AI [15].

Implications to Researchers. In the past four years, several works have emerged that call for a research agenda that considers energy efficiency in AI [3], [30], [34]. Past these positional papers, the number of hands-on studies is still very limited. Researchers should answer the call by building on our results w.r.t. hyperparameter tuning and efficient network architectures, or explore new areas of energy-efficient practices.

Implications to Tech Organisations. Large corporations are the biggest consumers in the field of AI. In this study, we have shown that the energy consumption of a deep learning model rises at a much faster pace than the performance. Tech organisations should make an effort to measure and report their energy consumption as a metric of equal importance to accuracy. This will change how we evaluate state-of-the-art deep learning models and encourage the development of Green AI.

IX. Threats to Validity

In this section, we go through potential threats to the internal, external and construct validity, as well as the reliability.

Internal validity. It could be argued that our method of measuring energy for RQ_2 does not provide an unbiased value. Different tasks running in the background could introduce noise to our measurements. To reduce the influence of this threat, the experiment was performed on a clean installation of Ubuntu 20.04. The only redundant program that might have had a slight impact on the measurements was a running instance of TeamViewer[10] that was used for periodic monitoring. Every optimisation round included 24 repetitions to drown out this effect. Moreover, when calculating the effect sizes of the layer types, we omit the hyperparameter sets. It is possible that different hyperparameter settings change the overall energy consumption of a neural network, however, in Section VI-B we calculate the coefficient of variance to show that this effect is negligible.

External validity. During the experiments, we did not consider the optimal utilisation of the GPU. This might have a negative impact on the generalizability because the relation between utilisation and power is not necessarily linear. Kistowski et al. [19] find that for CPUs, there is a steep increase in power output starting at around 80% utilisation. Furthermore, we performed the study by using the PyTorch framework only, which is deemed less energy-efficient compared to TensorFlow [15]. Yet, we kept the same framework for all the experiments to limit the impacting factors on the measured variables, and to mitigate the associated threats.

Construct validity. Because we solely consider the power usage of the GPU and ignore the contributions of other components, such as memory access, we do not capture the actual energy consumption for training a model. Nevertheless, we specifically selected GPU-heavy workloads and made sure

[9]https://docs.ray.io/en/latest/tune/tutorials/tune-advanced-tutorial.html

[10]https://www.teamviewer.com/nl/

to factor out the idle energy consumption. The results are therefore still valuable to compare relative to each other.

Reliability. To increase the reliability, we made an effort to assemble a complete replication package. The source code for training the neural networks, along with the results of the experiments and the statistical analysis, are all available online[11]. These components were created by a single developer, but all the involved authors reviewed and approved the entire process. The statistical analysis was replicated by one of the authors to confirm the findings.

X. Conclusion

In this study, we have expanded the horizon of green software to the realm of AI applications. Our empirical study shows that Bayesian optimisation can find the most optimal set of hyperparameters within the least number of iterations, where 27 should be sufficient in most instances (RQ_1). Grid search and random search have their purposes as baseline algorithms. If the parameter bounds are chosen with care, neither of those two strategies significantly dominates the other. Nevertheless, we advocate the use of random optimisation since the exhaustive nature of grid search often implies that one cannot consider the complete search space anyway. Additionally, because the function of hyperparameters has a low effective dimensionality, it is more reasonable to introduce randomness to the search space.

Furthermore, we have investigated the impact of a neural network's architecture on its energy consumption, followed by a trade-off analysis regarding the accuracy (RQ_2). We found that for a substantial increase in energy consumption, the increases in accuracy see diminishing returns. We advise reducing the number of convolutional layers to a point where the accuracy is still within a reasonable margin. This entirely depends on the project in question and should be evaluated case by case.

We hope that our study sheds light on the lopsided relationship between accuracy and energy; that it sparks interest in efficient design practices and helps to shift the evaluation criteria for neural networks to more conservative models.

References

[1] Albawi, S., Mohammed, T.A., Al-Zawi, S.: Understanding of a convolutional neural network. In: 2017 international conference on engineering and technology (ICET). pp. 1–6. Ieee (2017)

[2] Banerjee, A., Roychoudhury, A.: Automated re-factoring of android apps to enhance energy-efficiency. In: Proceedings of the International Conference on Mobile Software Engineering and Systems. pp. 139–150 (2016)

[3] Bender, E.M., Gebru, T., McMillan-Major, A., Shmitchell, S.: On the dangers of stochastic parrots: Can language models be too big? In: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency. pp. 610–623 (2021)

[4] Bergstra, J., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. Advances in neural information processing systems **24** (2011)

[5] Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of machine learning research **13**(2) (2012)

[11] https://zenodo.org/record/7767313

[6] Chowdhury, S., Borle, S., Romansky, S., Hindle, A.: Greenscaler: training software energy models with automatic test generation. Empirical Software Engineering **24**(4), 1649–1692 (2019)

[7] Chowdhury, S., Di Nardo, S., Hindle, A., Jiang, Z.M.J.: An exploratory study on assessing the energy impact of logging on android applications. Empirical Software Engineering **23**(3), 1422–1456 (2018)

[8] Cohen, J.: Statistical power analysis for the behavioral sciences. Routledge (2013)

[9] Cruz, L., Abreu, R.: Performance-based guidelines for energy efficient mobile applications. In: 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft). pp. 46–57. IEEE (2017)

[10] Cruz, L., Abreu, R.: Catalog of energy patterns for mobile applications. Empirical Software Engineering **24**(4), 2209–2235 (2019)

[11] Cruz, L., Abreu, R., Rouvignac, J.N.: Leafactor: Improving energy efficiency of android apps via automatic refactoring. In: 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft). pp. 205–206. IEEE (2017)

[12] De Macedo, J., Abreu, R., Pereira, R., Saraiva, J.: On the runtime and energy performance of webassembly: Is webassembly superior to javascript yet? In: 2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW). pp. 255–262. IEEE (2021)

[13] Feitosa, D., Alders, R., Ampatzoglou, A., Avgeriou, P., Nakagawa, E.Y.: Investigating the effect of design patterns on energy consumption. Journal of Software: Evolution and Process **29**(2) (2017)

[14] García-Martín, E., Rodrigues, C.F., Riley, G., Grahn, H.: Estimation of energy consumption in machine learning. Journal of Parallel and Distributed Computing **134**, 75–88 (2019)

[15] Georgiou, S., Kechagia, M., Sharma, T., Sarro, F., Zou, Y.: Green AI: do deep learning frameworks have different costs? In: ICSE '22: Proceedings of the 44th International Conference on Software Engineering, pp. 1082–1094. Association for Computing Machinery, New York, NY, USA (May 2022). https://doi.org/10.1145/3510003.3510221

[16] Gutiérrez, M., Moraga, M.Á., García, F.: Analysing the energy impact of different optimisations for machine learning models. In: 2022 International Conference on ICT for Sustainability (ICT4S), pp. 13–17. IEEE. https://doi.org/10.1109/ICT4S55073.2022.00016

[17] Hennig, P., Schuler, C.J.: Entropy search for information-efficient global optimization. Journal of Machine Learning Research **13**(6) (2012)

[18] Hernández-Lobato, J.M., Hoffman, M.W., Ghahramani, Z.: Predictive entropy search for efficient global optimization of black-box functions. arXiv preprint arXiv:1406.2541 (2014)

[19] v. Kistowski, J., Block, H., Beckett, J., Lange, K.D., Arnold, J.A., Kounev, S.: Analysis of the influences on server power consumption and energy efficiency for cpu-intensive workloads. In: Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering. pp. 223–234 (2015)

[20] Klein, A., Falkner, S., Bartels, S., Hennig, P., Hutter, F.: Fast bayesian optimization of machine learning hyperparameters on large datasets. In: Artificial Intelligence and Statistics. pp. 528–536. PMLR (2017)

[21] Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)

[22] Kushner, H.J.: A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. Journal of Basic Engineering **86**(1), 97–106 (1964). https://doi.org/10.1115/1.3653121

[23] Li, D., Chen, X., Becchi, M., Zong, Z.: Evaluating the energy efficiency of deep convolutional neural networks on cpus and gpus. In: 2016 IEEE international conferences on big data and cloud computing (BDCloud), social computing and networking (SocialCom), sustainable computing and communications (SustainCom)(BDCloud-SocialCom-SustainCom). pp. 477–484. IEEE (2016)

[24] Liashchynskyi, P., Liashchynskyi, P.: Grid search, random search, genetic algorithm: a big comparison for nas. arXiv preprint arXiv:1912.06059 (2019)

[25] Linares-Vásquez, M., Bavota, G., Bernal-Cárdenas, C., Oliveto, R., Di Penta, M., Poshyvanyk, D.: Mining energy-greedy api usage patterns in android apps: an empirical study. In: Proceedings of the 11th working conference on mining software repositories. pp. 2–11 (2014)

[26] Linares-Vásquez, M., Bernal-Cárdenas, C., Bavota, G., Oliveto, R., Di Penta, M., Poshyvanyk, D.: Gemma: multi-objective optimization of energy consumption of guis in android apps. In: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C). pp. 11–14. IEEE (2017)

[27] Ma, W., Lu, J.: An equivalence of fully connected layer and convolutional layer. arXiv preprint arXiv:1712.01252 (2017)
[28] Malavolta, I., Chinnappan, K., Swanborn, S., Lewis, G.A., Lago, P.: Mining the ros ecosystem for green architectural tactics in robotics and an empirical evaluation. In: 2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR). pp. 300–311. IEEE (2021)
[29] Mockus, J., Tiesis, V., Zilinskas, A.: The application of bayesian methods for seeking the extremum. Towards global optimization **2**(117-129), 2 (1978)
[30] Schwartz, R., Dodge, J., Smith, N.A., Etzioni, O.: Green ai. Communications of the ACM **63**(12), 54–63 (2020)
[31] Scott, W., Frazier, P., Powell, W.: The correlated knowledge gradient for simulation optimization of continuous parameters using gaussian process regression. SIAM Journal on Optimization **21**(3), 996–1026 (2011)
[32] Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems **25** (2012)
[33] Srinivas, N., Krause, A., Kakade, S.M., Seeger, M.: Gaussian process optimization in the bandit setting: No regret and experimental design. arXiv preprint arXiv:0912.3995 (2010)
[34] Strubell, E., Ganesh, A., McCallum, A.: Energy and policy considerations for deep learning in nlp. arXiv preprint arXiv:1906.02243 (2019)
[35] Torralba, A., Fergus, R., Freeman, W.T.: 80 million tiny images: A large data set for nonparametric object and scene recognition. IEEE transactions on pattern analysis and machine intelligence **30**(11), 1958–1970 (2008)
[36] Verdecchia, R., Cruz, L., Sallou, J., Lin, M., Wickenden, J., Hotellier, E.: Data-centric green ai an exploratory empirical study. In: 2022 International Conference on ICT for Sustainability (ICT4S). pp. 35–45 (2022). https://doi.org/10.1109/ICT4S55073.2022.00015
[37] Verdecchia, R., Sallou, J., Cruz, L.: A Systematic Review of Green AI. arXiv (Jan 2023). https://doi.org/10.48550/arXiv.2301.11047
[38] Wu, C.J., Raghavendra, R., Gupta, U., Acun, B., Ardalani, N., Maeng, K., Chang, G., Behram, F.A., Huang, J., Bai, C., et al.: Sustainable ai: Environmental implications, challenges and opportunities. arXiv preprint arXiv:2111.00364 (2021)
[39] Wu, J., Chen, X.Y., Zhang, H., Xiong, L.D., Lei, H., Deng, S.H.: Hyperparameter optimization for machine learning models based on bayesian optimization. Journal of Electronic Science and Technology **17**(1), 26–40 (2019)
[40] Xiao, H., Rasul, K., Vollgraf, R.: Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms (2017)
[41] Yang, T.J., Chen, Y.H., Sze, V.: Designing energy-efficient convolutional neural networks using energy-aware pruning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5687–5695 (2017)
[42] Yeung, G., Borowiec, D., Friday, A., Harper, R., Garraghan, P.: Towards {GPU} utilization prediction for cloud deep learning. In: 12th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 20) (2020)