IJHSR
ijhighschoolresearch.org
■ REVIEW ARTICLE

Studies on Environmental Impact and Energy Efficiency of Artificial Intelligence and Machine Learning

Yavuz E. Damkaci

Jamesville-Dewitt High School, Fayetteville, NY 13066, USA.; yavuzdamkaci@gmail.com

ABSTRACT: The consideration of environmental impact and energy efficiency when developing artificial intelligence (AI) and machine learning (ML) models has received some traction recently as ML models get bigger. It has been shown to increase the energy efficiency of models without significantly affecting accuracy. We reviewed five recent studies that focus on investigating and refining the energy efficiency of ML systems through numerous models and methods with either software or hardware variations. The first article proposed using the total number of Floating-Point of Operations (FPO) to better measure efficiency without depending on hardware, location, and AI approaches used in the model. The second article evaluated and compared two deep learning frameworks, Pytorch and Tensorflow, and how each has its strengths and weaknesses regarding its efficiency in the training phase versus the inference phase. The third article analyzed energy-efficient architectures for ML, which includes utilizing more approximation to increase efficiency while having a minimal effect on accuracy significantly. The fourth article focused on holistic architecture solutions for creating efficient run times and energy consumption of neural networks. The last article developed a Java tool to help with increasing energy efficiency on edges with minimal decrease in accuracy. Lastly, we touched on foundation models and their energy efficiency considerations.

KEYWORDS: Artificial Intelligence; Machine Learning; Energy Efficiency; Edge.

■ Introduction

Artificial Intelligence (AI), especially Machine Learning (ML), has significantly increased in the last five years with the introduction of several common applications with widespread use by the general public. With the recent growth, research has been primarily focused on the capabilities and accuracy performance of ML while neglecting its green aspects, such as energy consumption and its efficiency, and increased environmental impact.[1] As an example, for the increased environmental impact of machine learning, an ordinary car produces about 126,000 pounds of carbon dioxide over the course of its lifespan, and a common GPU training neural network model produces about 625,000 pounds of carbon dioxide.[2]

The popularity of Leaderboards[3,4] can be seen as evidence of the strong focus on capabilities and accuracy performance using AI while omitting any mention of cost or efficiency. Even though there are obvious advantages to increasing model accuracy, concentrating solely on this statistic ignores the effects on the economy, environment, and society.

Furthermore, an important class of machine learning applications is mainly defined as "edge computing" devices - in autonomous cars, at the mobile edge, etc., where overall power consumption and energy efficiency are of prime importance due to dependence on rechargeable battery technology. Therefore, system designs must support accelerated computation with improved energy efficiency and minimal compromise on computational accuracy or hardware costs. Currently, only 10% of the enterprise data is processed at the edge, and it is expected that it will rise to 75% by 2025.[5]

Internet of Things (IoT) devices are expected to generate over ten quintillions of data daily by 2025, with Connected and Autonomous Vehicles (CAVs) generating 8TB of data per hour.[6] Edge computing is expected to handle these constraints by handling data locally,[7] but energy-efficient software is crucial for avoiding hardware overheating.[8] Achieving energy efficiency will increase the usage time of the battery. Edge devices like mobile phones and electric vehicles rely on batteries, and their battery life significantly impacts user experience. Improving energy efficiency by 20% can increase an electric car's continuous driving distance from 500 to 600 kilometers. This enhances edge availability and extends the radius of life.[9]

The application of ML has improved several applications (e.g., in the medical, financial, and transportation sectors) that, for example, use speech and image recognition, machine translation, and natural language processing (NLP),[10,11] Clone detection,[12] de-obfuscation,[13] language migration,[14] code summarization,[15] auto-correction,[16] auto-completion,[17] code generation,[18] and program comprehension.[19] these advancements would not have been conceivable without the significant ML advancement. ML models consist of two processing phases: training and inference. Training is a phase to learn the pattern using selected training data. Inference is the phase to put a predicted outcome using a trained ML model.

■ Discussion

This review is concentrated on five selected recent studies[9,20-23] focusing on improving and analyzing the energy efficiency of machine learning systems through various methods and models with either hardware or software modifications.

DOI: 10.36838/v5i4.22

Energy Efficiency and Environmental Impact: Green vs. Red AI:

As recently reported by Schwartz *et al.*,20 DL research increased the computational costs of state-of-the-art AI research as big as 3000,000x between 2012 and 2018, as shown in Figure 1. This surge also increased Green Software Engineering research, which aims to decrease software environmental footprints and support Green AI.[24]

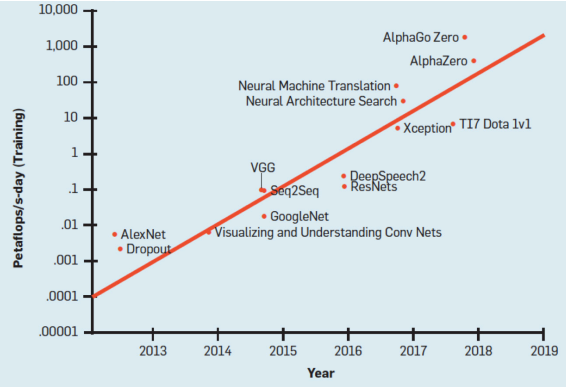

Figure 1: The amount of computing used to train deep learning models between 2012 and 2018, showing the 300,000 times increase, which is correlated to the cost and environmental impact.[20]

In the literature,[20] two terms have been used to describe the current trends: Green AI seeks sustainable and environmentally friendly computing solutions by seeking to treat efficiency as a primary evaluation criterion alongside accuracy, while Red AI seeks to improve accuracy (or related measures) using massive computational power while disregarding the cost—essentially "buying" more robust results.

To achieve a shift from Red AI to Green AI, efficiency must be a more common evaluation criterion for AI products alongside accuracy and related measures. Despite the clear benefits of improving model accuracy, focusing on a single accuracy metric needs to pay attention to the economic, environmental, and social costs of reaching the reported results.

Schwartz *et al.*[20] propose making efficiency a more common evaluation criterion for AI research alongside accuracy and related measures to lead the practice in the field by proposing to reduce Red AI practices while increasing the use of Green AI practices.

Schwartz *et al.*[20] empirical analysis provides evidence that relatively little attention was paid to computational efficiency by the AI research community in comparison to accuracy. They show that Red AI is on the rise despite the well-known diminishing returns of increased cost. As shown in Figure 2, taken from Mahajan *et al.*,[25] accuracy was increased linearly and reached a plateau in some cases. In contrast, training examples increased exponentially, resulting in higher training costs.

To move away from Red AI to Green AI, Schwartz *et al.*[20] identify "key factors that contribute to Red AI and advocate the introduction of a simple, easy-to-compute efficiency metric that could help make some AI research greener, more inclusive, and perhaps more cognitively plausible."

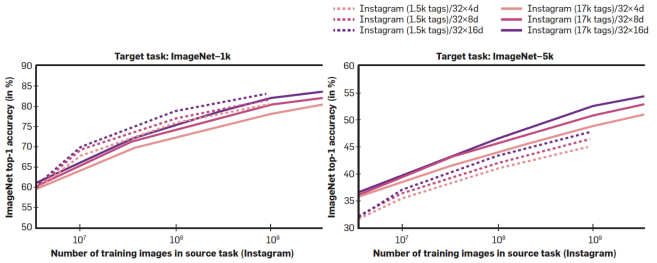

Figure 2: Linear increase of object detection top-1 accuracy in comparison to the exponential growth of training sample size (Instagram).[20]

To move away from Red AI to Green AI, Schwartz *et al.*[20] identify "key factors that contribute to Red AI and advocate the introduction of a simple, easy-to-compute efficiency metric that could help make some AI research greener, more inclusive, and perhaps more cognitively plausible."

To demonstrate the prevalence of Red AI, Schwartz *et al.*[20] randomly sampled 60 papers from top AI conferences (ACL, NeurIPS, and CVPR).d Their results are summarized in Figure 3, and as shown, a large majority of the papers target accuracy (90% of ACL papers, 80% of NeurIPS papers, and 75% of CVPR papers), while only a small portion (10% of AC, and 20% of CVPR) argue for a new efficiency result. Authors[20] state that the AI community focuses on performance measures, such as accuracy, at the expense of efficiency measures, such as speed or model size.

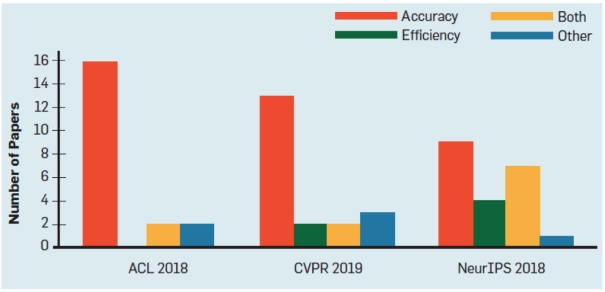

Figure 3: Number of papers, based on 60 randomly selected, presented at top AI conferences targeting accuracy, efficiency, both, or other. ACL: Association for Computational Linguistics, CVPR: Computer Vision and Pattern Recognition, NeurIPS: Neural Information Processing Systems.[20]

Authors[20] proposed to include the computational price tag of developing, training, and running models as a key Green AI practice. The proposed equation is Cost (R) *a* E · D · H where the cost of an AI (R)esult grows linearly with the cost of processing a single (E)xample, the size of the training (D)ataset and the number of (H)yperparameter experiments.

The proposed[20] equation illustrates three major quantities related to the cost of generating a result while ignoring data augmentation. It can be applied for calculating either training or inference costs. According to this equation, models using large parameters will incur high costs. As an example,[20] for such large models which have high costs for processing each example, which leads to large training costs, Google's BERT-large contains roughly 350 million parameters, Grover contains 1.5 billion parameters, NVIDIA's Megatron-LM contains over 8 billion parameters, Google's T5-11B contains 11 billion parameters, and openAI's openGPT-3,4 contains 175 billion parameters. In comparison, open GPT-2 contained only 1.5 billion parameters. This shows the current trend of making

DOI: 10.36838/v5i4.22

models larger, which increases faster than the model performance.

Several measures have been reported to assess the efficiency of different models, such as carbon dioxide emissions, electricity usage, elapsed real time, and the number of parameters. According to Schwartz *et al.*[20] each of these suffers from a deficiency; carbon dioxide emissions highly depend on local electrical structures and location dependent; electricity usage is hardware dependent and does not allow fair comparison between different models and machines, elapsed in real-time is highly hardware dependent, and also influenced by other programs running on the device.

Schwartz *et al.*,[20] suggest using the total number of Floating-Point of Operations (FPO) to better measure efficiency without depending on the model's hardware, location, and AI approach. In addition, FPO directly computes the amount of work done and is tied to the amount of energy consumed. FPO is often correlated with the running time of the model and the amount of work done at each time step.

The authors[20] also discussed the limitations of FPO. FPO does not capture the difference in work based on model implementation, as two different implementations of the same model could result in very different amounts of processing work. Also, FPO ignores the memory used by the model, which has additional energy and monetary costs.

The authors demonstrated the use of FPO compared to various models with different parameters and their top-1 accuracy levels. As shown in Figure 4a (top), the increase in FPO and the number of parameters of ResNet to SENet increases around 90-95%, and top-1 accuracy increases only 3.9%. Similarly, as shown in Figure 4b (bottom), when the number of layers in a model is increased from 50 to 152 (300% increase for different versions of ResNet for single object recognition), the number of parameters and FPO increased significantly. In comparison, top-1 accuracy increased only 2.4%.

The energy efficiency of DL frameworks:

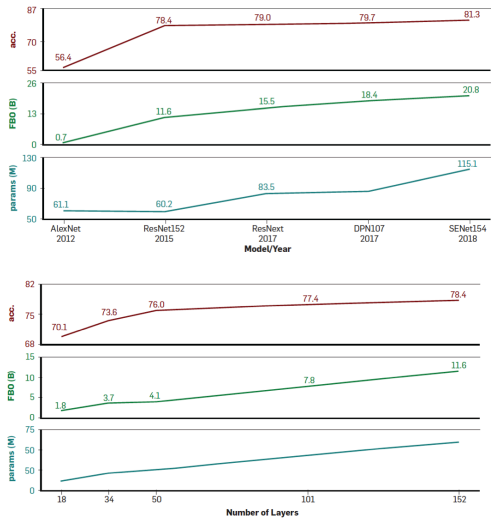

Figure 4: Plots of Parameters (in Millions), FPO (in Billions, and top- 1 accuracy (in percent): 4a-top: Comparison of different leading object recognition models. 4b-bottom: Comparison of various sizes, measured by the number of layers, of the ResNet Model.[20]

Georgiou *et al.*'s[21] paper on Green AI looked into two DL frameworks, in particular, PyTorch[38] and TensorFlow[39], to see whether they have different costs. The authors[21] performed an in-depth empirical analysis to investigate and compare the energy consumption and run-time performance of the frame-works by using six large models for DL from different AI domains, as listed in Table 1.

Table 1: Selected models used in the study of Georgiou *et al.* and their descriptions.[21]

Category	Model (Data Set)	Description
Recommender Systems (RS)	NCF (ml-20m)	It filters and provides feedback based on the NCF specifications
Natural Language Processing (NLP)	Transformer-XL (WikiText-103)	It enables capturing longer-term dependency and resolves the context fragmentation problem
	GNMT (WMT16 EN-DE)	It uses Google's Neural Machine Translation System (GNMT) to translate English to German
Computer Vision (CV)	ResNet-50 (Coco 2014)	It is an image classification algorithm with low complexity
	Mask R-CNN (Coco 2017)	It is a convolution-based neural network for the model of object instance segmentation
	SSD (Coco 2017)	It detects objects in images, using a single deep neural network

The authors[21] studied the most energy and run-time efficient and the most accurate among the two tested DL frameworks. The results of their empirical study showed that there is a significant cost difference between the two DL frameworks in 94% of the cases tested. TensorFlow achieves better energy and run-time performance in the training phase for 100% of the cases, and PyTorch shows better energy and run-time performance in the inference phase for 66% of the cases.

The author's[21] energy and run-time performance test results, both in the training and inference phases, were shown in Tables 2 and 3, respectively.

Authors[21] found that the cheapest DL framework changes, either in the training or inference phase, depending on the model's domain, such as recommender, vision, or NLP. The statistical tests (see Table 2) confirm that the results in the training phase are statistically different, with a large effect size in 24 out of 24 cases (100%), with TensorFlow significantly outperforming PyTorch in 16 out of 24 cases (76%). The statistical test (Table 3) in the inference phase favors PyTorch in 16 out of these 21 cases (76%). Overall, the authors concluded that the cheapest framework for the training phase is TensorFlow, while PyTorch is the least costly for the inference phase.

Table 2: Training Phase: Mean values for the energy consumption (in Joules) and run-time performance (in seconds) of training. The Wilcoxon statistical significance test results (p-value) and effect size ($A^{\wedge}_{12}$) are also reported. PKG is the core and non-core components of the processor, RAM is the main memory, and GPU is the graphics card.[21]

Model	PKG Energy			RAM Energy			GPU Energy			Run-Time		
	PyTorch	TensorFlow	p-value ($\hat{A}_{12}$)	PyTorch	TensorFlow	p-value ($\hat{A}_{12}$)	PyTorch	TensorFlow	p-value ($\hat{A}_{12}$)	PyTorch	TensorFlow	p-value ($\hat{A}_{12}$)
NCF	327,561	**280,428**	< 0.001(1)	51,537	**29,300**	< 0.001(1)	173,505	**85,127**	0.001(0.9)	5,901	**2,710**	< 0.001(1)
Transformer-XL	**144,781**	201,979	< 0.001(1)	**15,151**	24,791	< 0.001(1)	**113,754**	187,333	< 0.001(1)	**1,431**	2,495	< 0.001(1)
GNMT	**195,975**	604,856	< 0.001(1)	**25,164**	75,908	< 0.001(1)	**196,911**	521,321	< 0.001(1)	**2,515**	7,805	< 0.001(1)
ResNet-50	368,245	**258,787**	< 0.001(1)	39,506	**31,752**	< 0.001(1)	236,041	**216,157**	< 0.001(1)	3,472	**2,826**	< 0.001(1)
Mask R-CNN	255,395	**198,027**	< 0.001(1)	27,897	**22,739**	< 0.001(1)	176,012	**146,886**	< 0.001(1)	2,438	**2,028**	< 0.001(1)
SSD	590,050	**185,356**	< 0.001(1)	63,835	**22,678**	< 0.001(1)	412,664	**120,646**	< 0.001(1)	5,667	**2,072**	< 0.001(1)

Table 3: Inference Phase: Mean values for the energy consumption (in Joules) and run-time performance (in seconds) of training. The Wilcoxon statistical significance test results (p-value) and effect size ($A^{\wedge}_{12}$) are also reported. PKG is the core and non-core components of the processor, RAM is the main memory, and GPU is the graphics card.[21]

Model	PKG Energy			RAM Energy			GPU Energy			Run-Time		
	PyTorch	TensorFlow	p-value ($\hat{A}_{12}$)	PyTorch	TensorFlow	p-value ($\hat{A}_{12}$)	PyTorch	TensorFlow	p-value ($\hat{A}_{12}$)	PyTorch	TensorFlow	p-value ($\hat{A}_{12}$)
NCF	16,697	**11,275**	0.48(0.6)	1,780	**1,231**	0.48(0.6)	5,049	**1,829**	< 0.001(1)	159	**113**	0.48(0.6)
Transformer-XL	**21,789**	42,037	< 0.001(1)	**2,333**	4,657	< 0.001(1)	**14,968**	38,769	< 0.001(1)	**208**	509	< 0.001(1)
GNMT	**3,205**	9,638	< 0.001(1)	**346**	992	< 0.001(1)	**1,465**	3,406	< 0.001(1)	**31**	96	< 0.001(1)
ResNet-50	88,653	**36,531**	< 0.001(1)	9,333	**5,356**	< 0.001(1)	53,168	**45,812**	< 0.001(1)	776	**603**	< 0.001(1)
Mask R-CNN	**103,321**	106,301	< 0.001(0.9)	**11,120**	13,529	< 0.001(1)	**70,858**	81,806	< 0.001(1)	**963**	1,337	< 0.001(1)
SSD	**42,906**	67,035	< 0.001(1)	**4,554**	8,048	< 0.001(1)	**17,122**	26,893	< 0.001(1)	**408**	717	< 0.001(1)

The authors[21] used the combined results of training and inference from Tables 2 and 3. They graphed it as shown in Figure 5 because obtaining the final accuracy involves both the training and inference steps. Authors[21] concluded that better energy consumption and run-time performance—in most cases—yield better accuracy results as well. This outcome suggests that Green AI efforts should be increased in AI research and application for better environmental outcomes while obtaining a similar accuracy performance.

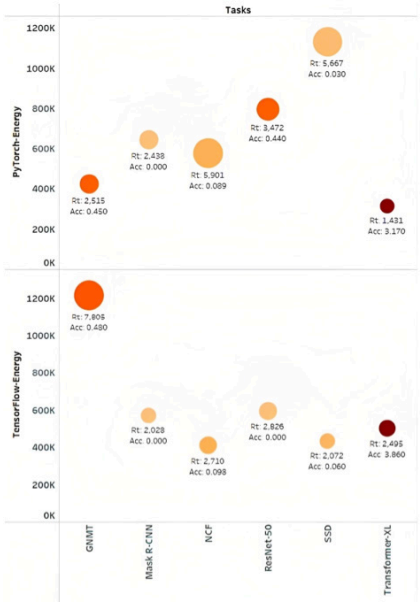

Figure 5: Comparison of energy consumption, run-time performance (Rt), and accuracy (Acc) between TensorFlow and PyTorch. The size of a circle represents the run-time performance (i.e., the bigger the process, the lower the run-time), and color represents the value of the accuracy metric (i.e., the darker the circle, the higher the accuracy); the interpretation depends on the metric of the model. Accuracy has been computed with Hit Rate (NFC), Perplexity (Transformer-XL), BLEU (GNMT), Top-5 error rate (ResNet-50), Avg. Precision (Mask R-CNN) and Precision (SSD).[21]

The authors[21] concluded that DL developers should choose the most appropriate framework for the model at hand while optimizing accuracy, run-time performance, and energy consumption. They also found that the training phase is more expensive than the inference one, thus resulting in a higher carbon footprint impact. Therefore, both researchers and developers should take appropriate steps to reduce its environmental impact.

Several research studies introduced practices on how to use traditional ML efficiently to reduce energy consumption. According to McIntosh *et al.*,[26] empirical studies showed that J48, SMO, and MLP are the algorithms using less energy in training ML models for Android devices, delivering more energy efficiency, better accuracy, and a correlation to algorithms' complexity. Wang *et al.*[27] achieved energy savings ranging from 60% to 90%, with an accuracy loss of 1.2% to 2% by dropping unnecessary computations from Convolutional Neural Network (CNN) models running on FPGAs. In addition, several other studies on energy estimation or measurements for ML studies exist.[28]

Energy efficient architectures for machine learning:

Shafique and his coworkers'[22] study focuses on creating adaptive and energy-efficient machine learning architectures. They showed that building an energy-efficient architecture using approximate computing and multi-objective evolutionary algorithms may result in high accuracy performances (92-94%) while being energy-efficient systems. There are several existing ways of increasing energy efficiency in ML, such as using FPGAs, ASICs, and other specialized compute fabrics like Application-Specific Instruction Set Processors (ASIPs) and Coarse-Grained Reconfigurable Processors (CGRAs) for CNN acceleration. Shafique *et al.*'s[22] study would like to further the energy consumption by "systematic development of energy-efficient accelerators for Machine Learning, specifically in the context of Deep Neural Networks (DNNs), where a high degree of adaptivity can be realized by employing quality-/energy-configurable approximate modules inside the accelerator datapath."

The authors[22] developed a systematic methodology for realizing energy-efficient accelerators, specifically for CNN and DNN-based AI systems, as shown in Figure 6.

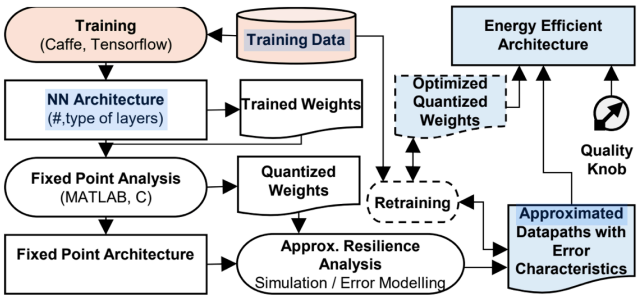

Figure 6: Shafique *et al.* developed the methodology for designing energy-efficient and adaptive neural network accelerator-based architectures for Machine Learning and AI systems.[22]

In the literature, there are several examples of energy-efficient and high-performance hardware accelerators using approximate arithmetic modules, which sometimes result in loss of accuracy and, therefore, require selective approximate arithmetic configuration.[29,30]

In a study[31] where the use of low-power imprecise multipliers using the UCI Machine Learning repository is studied, it has been shown that the loss of accuracy could be overcome by retraining the network by an energy savings of up to 62.49% over accurate face recognition. In a similar study,[32] constrained re-training of the neural network is done to get rid of undesired combinations due to approximation, which resulted in only an accuracy loss of 2.83% for 8-bit and ~0.25% for 12-bit implementations for handwritten digit recognition on the MNIST dataset, as shown in Figure 7.

Shafique *et al.*[22] showed that a careful analysis of data distribution across various data-paths of neural networks can aid in selecting an approximate circuit that provides minimum loss to classification accuracy. As shown in Figure 8, the authors[22] demonstrated that due to the fixed architectural design of DNN, each datapath has a particular data distribution. The highest classification accuracy was achieved when datapaths comprising FM5 and FM6 were approximated compared

DOI: 10.36838/v5i4.22

to the case when FM1 and FM3 were approximated, which showed the lowest accuracy. The authors[22] showed that high accuracy can be obtained via approximation computing, which results in energy-efficient architectures.

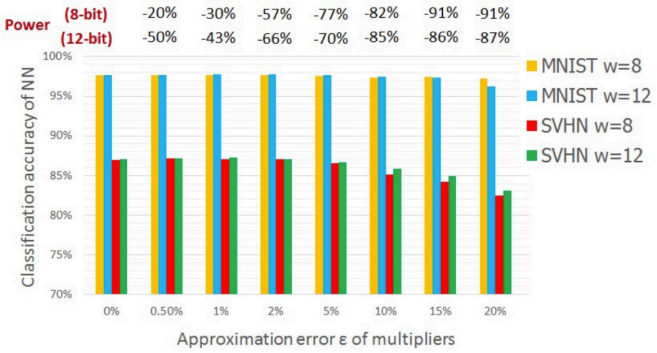

Figure 7: Classification accuracy of Neural Networks (NN) with approximate multipliers for MNIST and SVHN datasets and 8/12 bit precision.[22]

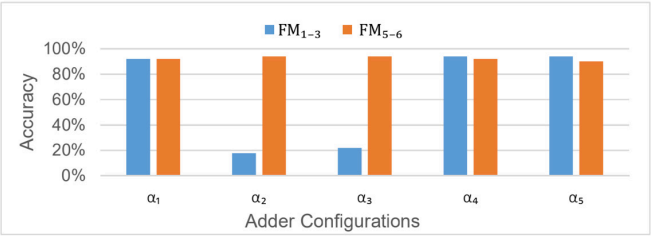

Figure 8: Classification accuracy of Neural Networks (NN) with approximate multipliers for MNIST and SVHN datasets and 8/12 bit precision.[22]

Energy efficiency of DNN implementations:

DNNs have achieved high accuracy in AI tasks, but their computation produces high energy costs.[33] According to Ganguly *et al.*,[23] scaling hardware architectures for higher performance in deep learning can worsen energy costs, increasing dataset sizes and network layers. Neuromorphic workloads, which result from a paradigm shift, change processing from compute-centric to data-centric, requiring emerging memory technologies like 3D-stacked DRAMs,[34] NVMs,[35] and near-data processing (NDP).[36] However, determining the best architecture for specific end-user scenarios and devices is challenging due to different tradeoffs.

Ganguly and coworkers[23] state that the energy efficiency of deep neural networks (DNN) implementations is challenging to evaluate using standard models, benchmarks, frameworks, and tools. Using a large number of GPUs in parallel works, but power consumption is a drawback. Accelerators and ASICs can operate in low-power envelopes but are inflexible. FPGAs are flexible but large, hard to program, and not cost-effective. Runtimes and energy efficiency of CNNs, used in services like image recognition, are critical for users and cloud service providers. Quantifying, measuring, modeling, and predicting energy consumption is crucial for system designers to determine whether to keep DNN computations locally or offload them to a cloud or data center.

DL frameworks like Caffe,[37] Torch,[38] and Tensorflow[39] provide tools for benchmarking applications' performance. Still, there is no support for runtime power/energy measurements across all system components, including CPU, GPU, memory subsystem, fabrics/interconnects, FPGAs, and accelerator/ASIC. Ganguly *et al.*[23] ask the following important questions regarding energy efficiency from a system architecture point of view: How much energy is consumed for an inference made by a neural network, and can we predict the energy consumption of a neural network on a specific implementation for a given set of parameters/constraints? The authors[23] suggest that based on the answers to these questions, a holistic approach is needed to evaluate the energy consumption of neural networks for specific usage scenarios and target devices.

Recently proposed Paleo's model[40] for predicting CNN runtime is limited to real-world performance. NeuralPower[41] developed a predictive framework for the power, runtime, and energy of CNNs without running them on a target platform. The framework provides a breakdown of runtime and power across different components, identifying bottlenecks and analyzing energy precision ratios. NeuralPower models the power and runtime of key layers in CNNs and predicts network performance. However, according to Ganguly and his co-workers,[23] NeuralPower may help evaluate energy-efficient choices for CNN implementations, allowing devices to weigh battery usage constraints and available resources on the cloud if extended beyond GPUs.

Energy Efficiency on Edges:

In their work, Kumar and coworkers[9] studied the JEPO tool as an Eclipse IDE plugin that offers energy-efficient Java suggestions for developers, which can be used in Edge applications. It analyzes Java files and matches them to a pool of suggestions. JEPO1 is an Eclipse plugin that helps developers write energy-efficient machine learning or refactor existing code in real time. It analyzes code and checks for specific patterns, generating suggestions and determining energy-hungry methods in Java projects using the Javassist library.

Kumar *et al.*[9] created Table 4 regarding these patterns, which relate to several Java programming language elements, are displayed along with some suggestions.

Table 4: Java components and energy efficiency suggestions[9]

Java Components	Suggestions
Primitive data types	int is the most energy-efficient primitive data type. Replace if possible.
Scientific notation	Scientific notation results in lower energy consumption of decimal numbers.
Wrapper classes	Integer Wrapper class object is the most energy efficient. Replace if possible.
Static keyword	static keyword consumes up to 17,700% more energy. Avoid if possible.
Arithmetic operators	Modulus arithmetic operator consumes up to 1,620% more energy than other arithmetic operators.
Ternary operator	Ternary operator consumes up to 37% more energy than if-then-else statement.
Short circuit operator	Put most common case first for lower energy consumption.
String concatenation operator	StringBuilder append method consumes much lower energy than String concatenation operator.
String comparison	String compareTo method consumes up to 33% more energy than the String equals method.
Arrays copy	System.arraycopy() is the most energy-efficient way to copy Arrays.
Array traversal	Two-dimensional Array column traversal result in up to 793% more energy.

According to Kumar and co-workers,[9] JEPO also helps measure energy consumption at method granularity, injecting energy and time measurement code at the start and end of each method. According to their initial evaluation showed a 14.46% improvement in package energy consumption, 14.19% in core energy consumption, and 12.93% in execution time, with only a 0.48% drop in classifier accuracy. Based on these results, for data centers, supercomputers, and autonomous vehicles, where vast amounts of data are processed quickly, the energy consumption of software can be significantly reduced with the aid of JEPO. Table 5 is the culmination (adopted from Kumar *et al.*[9]) of hardware and software packages developed specifically for IoT and Edge devices to increase efficiency.

Table 5: Hardware and software packages developed specifically for IoT and Edge devices with energy efficiency in mind.[9]

General Type	Product	Explanation
Hardware: Accelerator Architecture	NVIDIA® DRIVETM PX2 (250W)	embedded energy-efficiency GPU products to support connected and autonomous vehicles
	NVIDIA® JetsonTM platforms (5 – 30W)	embedded energy-efficiency GPU products to support for robots and drones
	DianNao family	energy-efficient hardware accelerators for machine learning algorithms
	Google Tensor Processing Unit (TPU)	Neural networks accelerator in data center
	Edge TPU	embedded version of TPU for edge computing
	IBM TrueNorth and Intel Loihi	neuromorphic processors for complex neural networks algorithms
Hardware: Full-Stack Optimization	EIE	energy efficient inference engine for compressed deep neural network
	ADMM-NN	algorithm-hardware co-design framework of deep neural networks using Alternating Direction Method of Multipliers (ADMM)
Software: Operating System Packages	TinyOS	embedded, component-based operating system for low-power edge devices
	Phi-Stack	co-designed hardware-software stack to natively support web and intelligence applications with minimized cost, footprint, and energy consumption
	OpenEI	a lightweight software platform to equip the edge with intelligent processing capability.
	OpenVDAP	an open vehicular data analysis platform, using EdgeOS-v as operating system
Software: Machine Learning Packages	MXNet	flexible and efficient library for deep learning to support CNN and long short-term memory networks (LSTM).
	Caffe2	lightweight, modular, and scalable deep learning framework,
	PyTorch	python package that provides tensor computation with strong GPU acceleration and deep neural networks built on a tape-based auto-grad system.
	TensorFlow Lite	TensorFlow's lightweight solution which is designed for mobile and edge devices by optimized and quantized kernels to reduce the latency and energy consumption.
	CoreML	deep learning package, by Apple, optimized for on-device performance to minimizes memory footprint and power consumption.
	QNNPACK (Quantized Neural Networks PACKage)	mobile-optimized library, by Facbook, to provide an implementation of common neural network operators on quantized 8-bit tensors to achieve low energy and high-performance.
	Paddle Lite	easy to perform inference on edge and IoT devices and it is compatible with PaddlePaddle and other pre-trained models.

Foundation models:

The term foundation models was first used in 2021 by researchers at Sandford University.[42] Unlike current NLP models, which require extensive training on labeled data which requires extensive time and energy, foundational models require little to no fine-tuning. They are trained on a large amount of unlabeled data that will be employed for various downstream tasks.[43] Foundation models can serve as the basis for several applications of the AI model, as their name implies. The model can transmit knowledge it has learned about one circumstance to another by using self-supervised learning and transfer learning. Foundation models require a vast corpus of training data; however, different models can be built on it for various tasks and applications once developed. The energy and time efficiency provided by the foundational models comes from being a foundation for multiple tasks and application without further training. Since they do not rely on labeled data, it also has time savings at the training phase of the model. Foundation models are a newly defined but fast-growing field of AI, with nearly seventy-five models identified[44] and 200 foundation model startups have emerged, collectively raising $3.5 billion as of October 2022.[45] Foundation models were recently defined as "pervasively influencing society" since unprecedented adoptions of applications such as ChatGPT and Bert.[46]

Conclusion

Real-time processing requirements of machine learning and local battery constraints on Edge make energy efficiency more challenging. Even though both hardware and software are important to increase efficiency, software is a critical bottleneck for IoT and Edge devices, as it can compromise performance and energy efficiency. Extensive research focuses on optimizing software energy consumption through programming languages, programming languages, optimized architectures, and compiling options.

As stated in our introduction, energy efficiency and utilizing the full performance of hardware with reduced heating on the end user devices will extend the battery life, provide a better user experience, and, most importantly, reduce the environmental impact. Since the number of devices is increasing rapidly as their use is also expanding in various scenarios, any improved energy efficiency on the devices using machine learning will substantially impact the environment.

Acknowledgments

The author would like to thank Drs. Bekir Turkkan and Tevfik Koser for their help and feedback in writing this review article. He also would like to thank his high school AP computer science teacher, Jay Lang, for getting him interested in computer science and software engineering and for his excellent guidance since ninth grade.

References

1. A) P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, J. Pineau, "Toward the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning," ArXiv, Computer Science, Computers and Society, 2022. https:// doi.org/10.48550/arXiv.2 002.05651. B) D. Paterson, J. Gonzalez, Q. Le, C. Liang, L. Munguia, D. Rothchild, D. So, M.Tezier, J. Dean, " Carbon Emissions and Large Neural Network Training" ArXiv, Computer Science, Machine Learning, 2021. https://doi.org/10.48550/arXiv.2104.10350 C) A. Gupta, C. Lanteigne, S. Kingsley, "SECure: A Social and Environmental Certificate for AI Systems" ArXiv, Computer Science, Computer and Society, 2020. https://doi.org/10.48550/arXiv.2006.06217 D) A. Lacoste, A. Luccioni, V. Schmidt, T. Dandres, "Quantifying the Carbon Emissions of Machine Learning" ArXiv, Computer Science, Computers and Society, 2019. https://doi.org/10.48550/arX iv.1910.09700
2. E. Strubell, A. Ganesh, and A. McCallum, "Energy and Policy Considerations for Deep Learning in NLP," ArXiv, Computer Science, Computational Language, 2019. https://doi.org/10.48550/arXiv.1906.02243
3. A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S.R. Bowman, "SuperGLUE: A stickier benchmark for general purpose language understanding systems," 2019; arXiv:1905.00537.
4. A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S.R. Bowman GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Proceedings of ICLR, 2019.
5. Edge Computing Solutions. Accessed at https://www.ibm.com/edge-computing
6. T. Stack, "Internet of Things (IoT) data continues to explode exponentially. Who is using that data and how?" URl: https://blogs.cisco.com, 2018.
7. R. Kelly, "Internet of things data to top 1.6 zettabytes by 2022," Campus Technology, 2016, vol. 9, 1536–1233.
8. B. Luo, S. Tan, Z. Yu, and W. Shi, "EdgeBox: Live edge video anal

DOI: 10.36838/v5i4.22

ytics for near real-time event detection," in 2018 IEEE/ACM Sy mposium on Edge Computing (SEC). IEEE, 2018, 347–348.
9. M. Kumar, X. Zhang, L. Liu, Y. Wang, W. Shi, "Energy-Efficient Machine Learning on the Edges" in 2020 IEEE International Par allel and Distributed Systems Symposium Workshops, 2020, 912 -921. 10.1109/IPDSW50202.2020.00153
10. T. N. Sainath, B. Kingsbury, G. Saon, H. Soltau, A. Mohamed, G. Dahl, and B. Ramabhadran. "Deep convolutional neural networks for large-scale speech tasks." in Neural Networks 2015, 64, 39–48. https://doi.org/10.1016/j. neunet.2014.08.005
11. A. Krizhevsky, I. Sutskever, and G. E. Hinton. "ImageNet classif ication with deep convolutional neural networks." in Advances in neural information processing systems. 2012,1097–1105.
12. M. White, M. Tufano, C. Vendome, and D. Poshyvanyk. "Deep learning code fragments for code clone detection. In Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. ACM, 2017,87-98.
13. B. Vasilescu, C. Casalnuovo, and P. Devanbu. "Recovering clear, n atural identifiers from obfuscated JS names." Proceedings of the 2 017 11th Joint Meeting on Foundations of Software Engineering. ACM, 2017, 683–693.
14. A. T. Nguyen, T. T. Nguyen, and T. N. Nguyen. "Lexical statistical machine translation for language migration." Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. ACM, 2013. 651–654.
15. S. Iyer, I. Konstas, A. Cheung, and L. Zettlemoyer. "Summarizin g source code using a neural attention model. In Proceedings of th e 54th Annual Meeting of the Association for Computational Lin guistics, 2016, Vol. 1. 2073–2083.
16. R. Gupta, S. Pal, A. Kanade, and S. Shevade. "DeepFix: Fixing C ommon C Language Errors by Deep Learning." AAAI. 2017, 134 5–1351.
17. S. R. Foster, W. G. Griswold, and S. Lerner. "WitchDoctor: IDE support for real-time auto-completion of refactorings." Software Engineering (ICSE), 2012 34th International Conference on. IE EE, 222–232.
18. W. Ling, P. Blunsom, E. Grefenstette, K. M. Hermann, T. Kočisky, F. Wang, and A. Senior. "Latent Predictor Networks for Code Ge neration." Proceedings of the 54th Annual Meeting of the Associ ation for Computational Linguistics (Volume 1: Long Papers), V ol. 1. 599–609.
19. C. V. Alexandru, S. Panichella, and H. C. Gall. 2017. "Replicatin g parser behavior using neural machine translation." Proceedings of the 25th International Conference on Program Comprehensio n. IEEE Press, 316–319.
20. R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni. "Green AI." Commun. ACM 63, 12 54–63. https://doi. org/10.1145/3381831.
21. S. Georgiou, M. Kechagia, T. Sharma, F. Sarro, Y. Zou, "Green AI: Do Deep Learning Frameworks Have Different Costs?" IEE E/ACM44th International Conference on Software Engineering (ICSE), 2022, 1082-1094. https:// doi.org/10.1145/3510 003.35 10221
22. M. Shafique, R. Hafiz, M.U. Javed, S. Abbas, L. Sekanina, Z. Va sicek, M. Mrazek, "Adaptive and Energy-Efficient Architectures for Machine Learning: Challenges, Opportunities, and Research Roadmap" IEEE Computer Society Annual Symposium on VLS I, 2017, 627-632. https://doi.org/10.1109/ISVLSI.2017.124
23. A. Ganguly, R. Muralidhar, V. Singh, "Towards Energy Efficient non-von Neumann Architectures for Deep Learning" IEE Inter national Symposium on Quality Electronic Design, 2017, 335-34 2. 10.1109/ISQED.2019.8697354
24. Zhang, X., Zhou, X., Lin, M. and Sun, J. "ShuffleNet: Anextreme ly efficient convolutional neural network for mobile devices." http s://doi.org/10.48550/arXiv.1707.01083
25. D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, L. van der Maaten, "Exploring the limits of we akly supervised pretraining" 2018; https:// doi.org/10.48550/arX Xiv.1805.00932
26. A. McIntosh, S. Hassan, and A. Hindle, "What can Android mo bile app developers do about the energy consumption of machine learning?" Empirical Software Engineering, 2018, 1–42.
27. Y. Wang, Z. Jiang, X. Chen, P. Xu, Y. Zhao, Y. Lin, and Z. Wang. 2019. "E2-Train: Training State-of-the-art CNNs with Over 80 % Energy Savings." ArXiv, Computer Science, Machine Learning , 2019. https://doi.org/10.48550/arXiv.1910.13349
28. E. García-Martín, C.F. Rodrigues, G. Riley, and H. Grahn, "Es timation of energy consumption in machine learning." J. Parallel and Distrib. Comput. 2019, 75–88. https://doi.org/10.1016/j.jpdc .2019.07.007
29. W. El-Harouni, S. Rehman, B. S. Prabakaran, A. Kumar, R. Hafiz , M. Shafique, "Embracing Approximate Computing for Energy-Efficient Motion Estimation in High Efficiency Video Coding," IEEE/ACM, 20th DATE Conference, 2017, 10.23919/DATE.2 017.7927209
30. S. Mazahir, O. Hasan, R. Hafiz, M. Shafique, J. Henkel, "An Area -Efficient Consolidated Configurable Error Correction for Appro ximate Hardware Accelerators," ACM/EDAC/IEEE 53rd Desig n Automation Conference (DAC), June 2016. 10.1145/2897 937. 2897981
31. Z. Du, K. Palem, A. Lingamneni, O. Temam, Y. Chen, C. Wu, " Leveraging the Error Resilience of Machine-Learning Applicati ons for Designing Highly Energy Efficient Accelerators," ASP-D AC, 2014, 201–206. 10.1109/ASP DAC.2014.6742890
32. V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasicek, K. Roy, "Design of power-efficient approximate multipliers for approximate artific ial neural networks," ICCAD, 2016, 1-7. https://doi.org/10.1145/ 2966986.2967021
33. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning" Nature, 52 1, 2015, 436-444.
34. R. Hadidi, B. Asgari, B. A. Mudassar, S. Mukhopadhyay, S. Yala manchili, and Hyesoon Kim, "Demystifying the characteristics of 3d-stacked memories: A case study for hybrid memory cube," IE EE Symposium on Worldload Characterization, 2017. 10.1109/I ISWC.2017.8167757
35. Y. Zhang and S. Swanson, "A study of application performance w ith non-volatile main memory." IEEE Symposiumon Mass Stora ge Systems and Technologies (MSST), 2015.10.1109/ MSST.201 5.7208275
36. R. Balasubramonian, J. Chang, T. Manning, J. Moreno, R.Murph y, R. Nair, and S. Swanson, "Near-data processing:Insights from a micro-46 workshop." IEEE Micro, 2014, 34, 36– 42. 10.1109/M M.2014.55
37. C. S. Thakur, J. Molin, G. Cauwenberghs, G. Indiveri, K.Kumar, N. Qiao, J. Schemmel, R. Wang, E. Chicca, J. O. Hasler, "Large-scale neuromorphic spiking array processors: A quest to mimic the brain," arXiv preprint arXiv:1805.08932, 2018.
38. R. Collobert, K. Kavukcuoglu, and C. Farabet. "Torch7: A matla b-like environment for machine learning," In BigLearn,NIPS Workshop, 2011. https://publications.idiap.ch/downloads/papers /2011/Collobert_NIPSWORKSHOP_2011.pdf
39. M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. De vin, S. Ghemawat, G. Irving, M. Isard, *et al*. "Tensorflow: a system for large-scale machine learning," OSDI,2016, 16, 265– 283.http s://www.usenix.org/system/files/conference/osdi16/os di16-abad i.pdf
40. H. Qi, E. R. Sparks, and A. S. Talwalkar, "Paleo: a performance m odel for deep neural networks" ICLR, 2017. https:// github.com/

TalwalkarLab/paleo
41. E. Cai, D. Juan, D. Stamoulis, and D. Marculescu, "Neural powe r: Predict and deploy energy-efficient convolutionaltional neural networks," ArXiv, Computer Science, Machine Learning, 2017. https://doi.org/10.48550/arX iv.1710.05420
42. R. Bommasani, *et. al*. "On the Opportunities and Risks of Foun dational Models" ArXiv, Computer Science, Machine Learning, August 2021, https://doi.org/10.48550/ arXiv.2108.07258
43. M. Murphy, "Foundational Models" IBM Research Blog, May 2022, accessed at https://research.ibm.com/blog/what-are-foun dation-models
44. X. Amatriaian, A. Sankar, J. Bing, P.K. Bodigutla, T. Hazen, M. Kazi. "Transformer Models; an introduction and catalog" ArXiv, Computer Science, Computation and Language, 2023, https:// doi.org/10.48550/arXiv.2302.07730
45. J. Kaufmann, M. Abram, and M. Basta. "Introducing: the scale generative ai index." 2022. https://www.scalevp.com/blog/ intro-ducing-the-scale-generative-ai-index.
46. R. Bommasani, D. Soylu, T.I. Liao, K. A. Creel, P. Liang Ecosys tem Graphs; The Social Footprint of Foundation Models" ArXiv, Computer Science, Machine Learning, 2023. https://doi.org/ 10. 48550/arXiv.2303.15772

Authors

Yavuz Emre Damkaci is a rising senior high school student at Jamesville-Dewitt High School in Upstate New York. He is interested in software engineering and interned at a software start-up to develop an algorithm for their product. He is also been passionate about the environmental, solving related issues, and aiding with its protection.

DOI: 10.36838/v5i4.22