Energy Efficiency of Python Machine Learning Frameworks

Salwa Ajel[1](✉), Francisco Ribeiro[2], Ridha Ejbali[3], and João Saraiva[2]

[1] Faculty of Sciences of Gabes, University of Gabes, Gabes, Tunisia
ajelsaloua@gmail.com

[2] HASLab/INESC TEC, Universidade do Minho, Braga, Portugal
francisco.j.ribeiro@inesctec.pt, saraiva@di.uminho.pt

[3] Research Team in Intelligent Machines, Engineering School of Gabes, University of Gabes, Gabes, Tunisia
ridha_ejbali@ieee.org

Abstract. Although machine learning (ML) is a field that has been the subject of research for decades, a large number of applications with high computational power have recently emerged. Usually, we only focus on solving machine learning problems without considering how much energy has been consumed by the different frameworks used for such applications. This study aims to provide a comparison among four widely used frameworks such as Tensorflow, Keras, Pytorch, and Scikit-learn in terms of many aspects, including energy efficiency, memory usage, execution time, and accuracy. We monitor the performance of such frameworks using different well-known machine learning benchmark problems. Our results show interesting findings, such as slower and faster frameworks consuming less or more energy, higher or lower memory usage, etc. We show how to use our results to provide machine learning developers with information to decide which framework to use for their applications when energy efficiency is a concern.

Keywords: Machine Learning · Keras · Energy-Efficient · Deep Learning · Tensorflow · Memory usage · Pytorch · Execution time

1 Introduction

Computer architecture researchers have been investigating energy efficiency for decades, especially to develop the most advanced, energy-efficient processors. Machine learning researchers, on the other hand, have mostly concentrated on producing highly accurate models without considering energy consumption as a crucial aspect. This is the case for deep learning, where the objective has been to produce deeper and more accurate models without any constraints in terms of computation. These models have grown in computation and memory requirements. These algorithms require high levels of computing power during training as they must be trained on large amounts of data, while during deployment they may be used multiple times. Therefore, we believe that efforts towards estimating energy efficiency and developing tools for researchers to advance their research in energy consumption are necessary for a more scalable and sustainable future.

We believe that the reason why the machine learning community has not shown more interest in energy efficient is because of their lack of familiarity with the current methods to estimating energy and the lack of power models in existing machine learning frameworks, for example, in Tensorflow [11] Caffe2 [10], PyTorch [14], and others to support energy evaluations. Developers have constantly improved these frameworks by adding more features and speed improvements to attract more users and foster research. Recently, the efficacy of several deep learning frameworks has been evaluated in [6]. However, the comparison is only focused on the speed of the convolutional frameworks. Hence, this paper expands a comparative study of four machine learning frameworks, namely: Tensorflow, Keras, Pytorch, and Scikit-learn in terms of energy efficient, memory usage, and runtime metrics. To ensure that our study is as comprehensive as possible, we consider multiple benchmark datasets from different fields (digit recognition, twitter sentiment analysis, malaria cell detection etc.) and measure the performance of the frameworks' implementations of different deep learning algorithms.

The rest of the paper is organized as follows: Section 2 provides the background for the work by defining the various tools used for this study and provides a brief overview of the various ML frameworks we focus on in this paper. Section 3 discusses related work. Section 4 describes the benchmarking setup, which presents the evaluation metrics and the system setup used for the various implementations. Section 5 presents the methodology implementations, which contains the experimental datasets and models. Lastly, Sect. 6 presents the results of the comparative study of the four frameworks in terms of energy efficient, memory usage, and execution time.

2 Background

In this section, we define machine learning, deep learning, their methods, and present an overview of machine learning frameworks.

2.1 Machine Learning

Machine learning is a multidisciplinary process that combines a variety of scientific domains and allows computers to automatically learn from data. Machine learning systems are classified according to the type and amount of human supervision they get during their training.

2.2 Support Vector Machine

A support vector machine is a popular supervised learning model developed by Vladimir Vapnik and used for both data classification and regression. It is typically leveraged for classification problems by constructing a hyperplane where the distance between two data point classes is at its maximum [2]. This hyperplane is known as the decision boundary, separating the data point classes on either side of the plane.

2.3 Deep Learning

Deep learning (DL) is a subfield of machine learning concerned with algorithms inspired by the structure and function of the human brain called artificial neural network. Deep learning algorithms build knowledge through a cascade of layers. The output layer from the preceding layer is given as input to the following layer. In general, deep learning architecture is made up of numerous layers of input and output, as well as parameterized non-linear modules. The parameters are the topic of study. Each layer provides a more detailed depiction than the one before it [18].

2.4 Convolutional Neural Networks

These networks are a specific type of neural network used in the field of deep learning. Being one of the best learning algorithms for understanding image content, it has excelled at image segmentation, classification [3], recognition [15], detection, and retrieval related tasks. The fundamental advantage of CNNs over their antecedents is that they require no human supervision and automatically identify the pertinent features. Neurons in human and animal brains were used to stimulate the construction of CNNs, just like in a traditional neural network [4]. The CNN architecture consists of several layers, such as the convolution layer, the pooling layer, the activation function, and a fully connected layer, with each layer including its functions.

2.5 Recurrent Neural Network

These networks are crucial for evaluating sequential data because they give us a way to incorporate memory into the neural networks. An example of sequential data may be text generation, stock prediction, voice recognition, or simply a network that predicts what to cook today based on the weather and yesterday's dishes. RNNs [8] are frequently used for text processing and text generation because of the way sentences are structured as a sequence of words.

2.6 Overview of Machine Learning Frameworks

In this section, we go through each framework and highlight the difference between them.

– **Tensorflow**—an open-source framework employed for high-performance numerical computation. It was created by Google researchers and engineers and has excellent support for deep learning and machine learning. It allows the user to train their models on both the CPU and the GPU [11].

- **Keras**—an open-source framework used for high-level building blocks for developing almost any kind of deep learning model. It can also be used with Tensorflow, CNTK, and Theano frameworks [1]. It was developed with the goal of facilitating rapid experimentation and is available under the MIT license. Keras runs on Python 2.7 to 3.6 and, depending on the underlying frameworks, can run on both GPUs and CPUs.
- **Pytorch**—a python framework used for GPU-accelerated deep learning. It is a Python interface to the same C libraries that Torch uses that have been optimized. Since 2016, Facebook's AI research team has been working on it [14]. PyTorch offers DNNs constructed on a tape-based autograd scheme and tensor computing with significant GPU acceleration. It has gained popularity because it makes it simple to build complicated architectures.
- **Scikit-learn**—an open-source library for analyzing data mining supported by INRIA, Google, and Telecom Paristech. It uses Python to analyze and build models from several machine learning algorithms, including classification, regression, and clustering [13]. In addition, Scikit-learn can be used for preprocessing data in several ways: standardization, normalization, and cleaning missing or outlier data. It extends the functionality of the NumPy and SciPy packages with various ML algorithms.

Table 1 presents a comparison of different ML libraries regarding different factors.

Table 1. Comparison of Machine Learning frameworks.

	Tensorflow	**Keras**	**Pytorch**	**Scikit-learn**
API Level	High & low level	High level	Low level	High
Speed	For high performance	Slow	For high performance	Fast, high-performance
Architecture	Complex	Simple	Complex	Simple to use
Coding	Reduce size of model with high accuracy	Single line code	Complex	Simple
Debugging	Difficult	Not frequently needed	Better debugging capabilities	Difficult to conduct debugging
Support	Backed by community tech companies	Small community support	Strong community support	Strong community support
Datasets	High performance models	Small	Large datasets	Few small standard datasets
Popularity	Highest popularity	High due to its simplicity	High	High

3 Related Works

Only a few studies have been conducted to compare machine learning and deep learning frameworks in terms of various metrics. These studies highlight the benefits and drawbacks of each framework and help programmers make an informed decision about the best ML framework that suits their needs and resources. A recent comparative study of ML frameworks aims to compare three deep learning frameworks, namely CNTK, TensorFlow, and Theano. *Shatnawi et al.* [16] used multiple benchmark datasets from MNIST and CIFAR-10 in their study, which were used to train a model based on a CNN architecture on two different machines. They found that for the GPU utilization metric in MNIST and CIFAR-10 datasets, TensorFlow had the lowest utilization, followed by Theano and CNTK. For the CPU utilization metric, Theano had the lowest utilization, followed by TensorFlow and CNTK. For memory utilization, while using CPU and GPU, the results were close to each other.

Furthermore, *Gevorkyan et al.* [9] present a comparison of five libraries (Keras, Tensorflow, Pytorch, Theano, and Scikit-learn) carried out on the example of a multilayer perceptron applied to the problem of handwritten digit recognition. They compared the training time depending on the number of epochs and the accuracy of the classifier. As a result, they found that almost all libraries, except PyTorch, show approximately the same learning time. In the case of PyTorch, the longer learning time can be explained by the support of a dynamic computational graph, which appears to impose additional computational costs. In turn, the TensorFlow library showed an average accuracy result, trailing PyTorch and Theano.

In [17], *Shi et al.* conduct a comparison of numerous DL frameworks, including Caffe, MXNet, CNTK, TensorFlow, and Torch. The authors consider three types of neural networks such as fully connected neural networks (FCN), convolutional neural networks (CNN) and recurrent neural networks (RNN). Additionally, they employed various hardware environments, including two CPU platforms and three GPU platforms. They evaluated the selected frameworks using running time and convergence rate. In their trials, they measured the convergence rate using real-world datasets and synthetic datasets, respectively, to assess running time performance.

Another related paper to the same task, *Bahrampour et al.* [5], compares five DL frameworks: TensorFlow, Theano, Torch, Caffe, and Neon, in terms of speed, hardware utilization, and extensibility after applying various convolutional algorithms to the aforementioned frameworks. They carried out their experiments on a single machine for both CPU (multi-threaded) and GPU (Nvidia Titan X) environments. On the MNIST dataset and the ImageNet dataset, convolutional and stacked autoencoder networks were trained in order to compare the two frameworks. On the IMDB dataset, the authors also trained an LSTM network.

In these works, the comparison goal was limited to running time, average accuracy, and CPU and GPU utilization. None of those comparative studies dealt with CPU energy consumption and memory usage. We consider it important to take care of these metrics for machine learning frameworks, which can help ML developers decide which framework to use for their applications when energy efficiency is an issue.

4 Benchmarking Setup

4.1 Evaluation Metrics

We use the four following evaluation metrics to obtain the comparison of the four machine learning frameworks under various ML datasets.

– **Energy Consumption** usually refers to a hardware approach to reduce the power consumption of processors such as CPUs, GPUs, etc., or ways to make processors handle more operations using the same amount of power. The amount of energy consumed is measured in joules (J) and is defined by the following formula $\boldsymbol{E} = \boldsymbol{P} \times \boldsymbol{T}$. Power is the rate at which energy is being consumed. The *average power* during a time interval T is defined as P. It is measured in watts (W), and *time* T is measured in seconds (s).
– **Memory Usage** is defined as the peak memory usage during the training process using the different models.
– **Execution Time** is a critical factor we have measured, for which we can choose the fastest framework for such defined problems (MNIST, CIFAR-10, Malaria Cell, and Twitter emotion classification). It represents the time needed for the training and testing process of the selected model.
– **Accuracy** it is the ratio of the number of correct predictions over the total number of input samples and measures how often the classifier makes correct predictions.

4.2 System Setup

All the experiments are performed on a single machine running on Ubuntu 20.04.2 LTS with an Intel Core(TM) CPU i7-8550U, 16 GB (2 modules: 8 GB + 8 GB; 2400 MHz) of RAM and a 512 GB SSD. We used Python 3.7 as a programming language as it is widely used for implementing Machine Learning frameworks and models. Regarding frameworks, we tested Keras 2.9.0, Tensorflow 2.9.1, Pytorch 0.13.0 and Scikit-learn 1.0.2.

5 Methodology Implementations

We chose four of the most popular and classic datasets: MNIST, CIFAR-10, Malaria Cell, and the Twitter Emotion dataset. The first dataset is MNIST which contains 60.000 grayscale images of ten handwritten digits each. Each image is 28×28 in size [7]. The second dataset is CIFAR-10 which consists of 60.000 colored images of 10 classes, each of which is 32×32 in size [19]. The third dataset is Malaria Cell, which includes 27,558 images of infected and healthy cells. The last dataset is Twitter Emotion which contains six basic emotions:

anger, fear, joy, love, sadness, and surprise. We implemented three CNN models with the three DL frameworks Keras, Tensorflow, and Pytorch using three different datasets. For the Malaria Cell dataset, we implemented a binary CNN model to classify images into infected and uninfected. For the MNIST and CIFAR-10 datasets, we implemented two multiclass models: for MNIST to classify the ten handwritten digits and for CIFAR-10 to classify the ten colored images. However, we implemented an RNN model with the Twitter Emotion dataset and always with the same three deep learning frameworks. For Scikit-learn, we used an SVM model for the MNIST and Malaria Cell datasets with an SVC classifier.

6 Experimental Comparison and Analysis

This section outlines the results obtained by our experiments. To measure the energy efficient, memory usage, execution time, and accuracy, we have to evaluate it using different datasets such as MNIST, CIFAR-10, Malaria Cell, and Emotion Dataset for each implemented framework, including Keras, Tensorflow, Pytorch, and Scikit-learn. In addition, for all benchmarks, we are focusing on the training phase. For measuring the energy consumption, we used the RAPL tool which is capable of providing accurate energy estimates of a piece of code at a very fine-grained level [12]. Also, the current version of RAPL allows it to be invoked from any program written in Python, C, or Java. Table 2 presents the results, and through it we make four interesting observations.

Table 2. Energy consumption, memory usage, runtime and accuracy results by the four framework

Dataset	Framework	Energy	Memory usage	Runtime	Accuracy
MNIST	Keras	14456.66	1596734.5	967086.67	98%
	Pytorch	2882.93	370162.5	192856.3	99%
	Tensorflow	1720.15	1255334	115324.6	99%
	Scikit-learn	10.96	88767.5	597.07	79%
CIFAR-10	Keras	9300.18	3457869.5	622175.71	81%
	Pytorch	1891.4	575526	126522	51%
	Tensorflow	60464.3	3037790	225907.43	79%
Malaria Cell	Keras	3032.7	2726318.5	202881	73%
	Pytorch	5801.91	15726834.5	3121898	50%
	Tensorflow	1110.08	1325864	74266.72	95%
	Scikit-learn	50.4	86092.5	2804.59	72%
Emotion	Keras	4763.8	1020269	319332.14	99%
	Pytorch	4871.83	15709695	332615	50%
	Tensorflow	2543.71	828118.5	169704.5	88%

First, regarding the MNIST dataset and the three DL frameworks, Tensorflow was the most energy efficient, the fastest one, and achieved the highest accuracy as Pytorch, but used less memory than Keras. Keras was the least energy efficient among the 3 frameworks, as well as having the worst memory intensity, runtime, and accuracy.

Second, For CIFAR-10, Pytorch was the most energy-efficient, the fastest, and the least memory-intensive, but it was the one that presented the worst accuracy. Tensorflow was the least energy efficient among the 3 frameworks, followed by Keras, but used less memory than Keras.

Third, For Malaria Cell, Tensorflow was the most energy efficient, followed by Keras, which was the fastest, achieved the highest accuracy, and used less memory than Keras. Pytorch was the least energy efficient, memory intensive, runtime, and accurate of the 3 frameworks.

Fourth, Tensorflow was the most energy-efficient on the Emotion dataset and achieved a higher accuracy than Pytorch while also being faster. However, Pytorch was the least energy efficient and memory intensive. Keras came in second place regarding all metrics except accuracy, for which it presented the best result.

For both of its datasets, MNIST and Malaria Cell, the Scikit-learn framework always consumes much less energy, memory, and executes faster. We think that is due to the simplicity of the SVM algorithm, unlike other frameworks that use the CNN and RNN algorithms. However, it comes at the expense of being less accurate.

The following four charts - Figs. 1, 2, 3, 4 - explain the previous observations of the comparison of energy efficient, memory usage, and running time as well as the accuracy of the four frameworks.

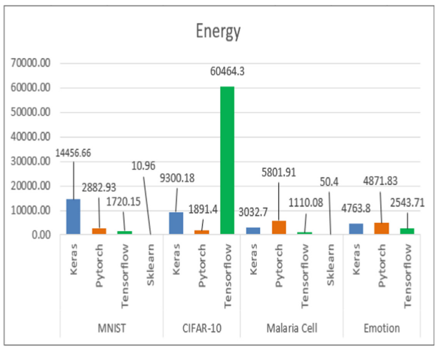

Fig. 1. Energy efficient comparison.

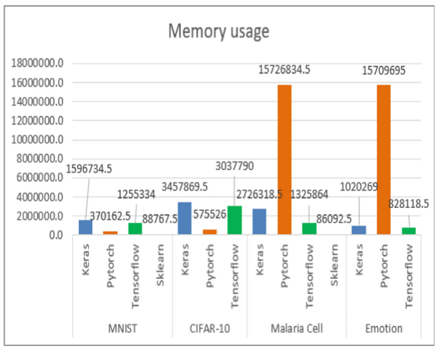

Fig. 2. Memory usage comparison.

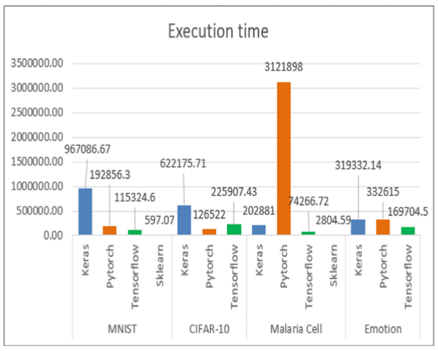

Fig. 3. Running time comparison.

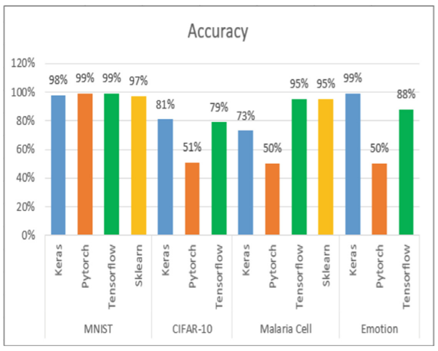

Fig. 4. Accuracy comparison.

7 Conclusion and Future Work

This research aims to help machine learning developers and engineers to produce faster applications and to allow them to become more energy-aware when programming by producing a comparative study based on the four most commonly used frameworks: Tensorflow, Keras, Pytorch, and Scikit-learn. To validate this comparison, we implemented different benchmarks to monitor the performance of such frameworks in terms of many important aspects including energy consumption, memory usage, running time and accuracy.

As for future work, we are looking forward to ameliorating our solution by adding energy efficient based on other platforms' hardware such as GPUs and FPGAs.

Acknowledgement. We want to thank the Ministry of Higher Education and Gabes University for facilitating the travel of Salwa Ajel to Portugal, the HASLab/INESC TEC, Universidade do Minho (Portugal) for the technical support of the work, and the Erasmus Jamies for accepting Salwa Ajel's application.

References

1. Aakash, N., Sayak, P.M.M.R.: Keras: high-level neural networks API. https://keras.io/ (2018). Accessed 20 Oct 2018
2. Ahmed, F.S., et al.: A hybrid machine learning framework to predict mortality in paralytic ileus patients using electronic health records (EHRs). J. Ambient Intell. Human Comput. **12**, 685–693 (2021)
3. Ali, R.B., Ejbali, R., Zaied, M.: A deep convolutional neural wavelet network for classification of medical images, vol. 14, pp. 1488–1498. Science Publications (2018). https://doi.org/10.3844/jcssp.2018.1488.1498
4. Alzubaidi, L., et al.: Review of deep learning: concepts, CNN architectures, challenges, applications, future directions, vol. 8 (2021)
5. Bahrampour, S., Ramakrishnan, N., Schott, L., Shah, M.: Comparative study of deep learning software frameworks (2015)

6. Chintala, S.: Convnet-benchmarks. https://mse238blog.stanford.edu/2017/07/gnakhare/hardware-options-for-machinedeep-learning. https://github.com/soumith/convnet-benchmarks/ (2015). Accessed 30 Oct 2015
7. Deng, L.: The MNIST database of handwritten digit images for machine learning research, vol. 29, pp. 141–142 (2012). https://doi.org/10.1109/MSP.2012.2211477
8. Elman, J.: Distributed representations, simple recurrent networks, and grammatical structure, vol. 7 (1993)
9. Gevorkyan, M.N., Demidova, A.V., Demidova, T.S., Sobolev, A.A.: Review and comparative analysis of machine learning libraries for machine learning, vol. 27, pp. 305–315 (2019)
10. Jia, Y., et al.: Caffe: convolutional architecture for fast feature embedding, vol. abs/1408.5093 (2014). http://arxiv.org/abs/1408.5093
11. Nguyen, G., et al.: Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey. Artif. Intell. Rev. **52**(1), 77–124 (2019). https://doi.org/10.1007/s10462-018-09679-z
12. Pereira, R., et al.: Energy efficiency across programming languages: how do energy, time, and memory related. In: Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering, pp. 256–267 (2017)
13. Phaladisailoed, T., Numnonda, T.: Machine learning models comparison for bitcoin price prediction. In: 2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE), pp. 506–511 (2018)
14. PyTorch: PyTorch deep learning framework that puts python first. http://pytorch.org/ (2018). Accessed 20 Oct 2018
15. Said, S., Jemai, O., Hassairi, S., Ejbali, R., Zaied, M., Ben Amar, C.: Deep wavelet network for image classification. In: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 922–927 (2016)
16. Shatnawi, A., Al-Bdour, G., Al-Qurran, R., Al-Ayyoub, M.: A comparative study of open source deep learning frameworks. In: 2018 9th International Conference on Information and Communication Systems (ICICS), pp. 72–77 (2018). https://doi.org/10.1109/IACS.2018.8355444
17. Shi, S., Wang, Q., Xu, P., Chu, X.: Benchmarking state-of-the-art deep learning software tools. In: 2016 7th International Conference on Cloud Computing and Big Data (CCBD), pp. 99–104 (2016). https://doi.org/10.1109/CCBD.2016.029
18. Shorten, C., Khoshgoftaar, T.M.: A survey on image data augmentation for deep learning. J. Big Data **6**(1), 1–48 (2019). https://doi.org/10.1186/s40537-019-0197-0
19. Wu, Y., et al.: A comparative measurement study of deep learning as a service framework. IEEE Trans. Services Comput. **15**, 551–566 (2022). https://doi.org/10.1109/TSC.2019.2928551