Energy Measurements of the Life Cycle of Foundation Models

Abstract

Foundation models (FM) have become the main target of machine learning (ML) research, as they have been shown to produce human-level responses in conversational AI and image generation. Models like OpenAI's ChatGPT or Google's Bard are now mainstream tools used on a daily basis for generating code, writing essays, and many other NLP tasks. Training and deploying (i.e., running inference queries) such models requires large amounts of compute power over extended periods of time. This translates to high-energy usage, that without proper management can lead to significant amounts of carbon emissions being released to the atmosphere. This study aims to establish best practices as it relates to making FM a sustainable tool, by collecting and analyzing energy measurements of FM jobs run on cloud computing infrastructure with OpenShift.

1 Introduction

In training an AI system, the choice of hyperparameters used (i.e., learning rate, batch size, etc.) impacts the convergence of the model as well as the energy used during training. In many cases, these are competing objectives as an ideal scenario is that of a highly accurate model trained with the least amount of energy. Achieving high model accuracy is done by training over longer periods of time, increasing the number of training samples, tuning hyperparameters over several training rounds, all of which result in higher energy usage. It is very difficult to estimate how a particular hyperparameter choice affects the convergence of the model. However, a few hyperparameter choices are known to, generally, improve the convergence at the cost of more computational resources. Here we focus on the **batch size** which defines the number of training samples run *together* at a time during the forward and backward pass of the training phase. The assumption here is that, a larger batch size improves the convergence of the method as the *context* of the training is better understood with larger sizes, leading to faster convergence. In turn, using large batch sizes requires more memory limiting the number of samples one can use for a given hardware, and increasing the energy consumption of the system used (e.g., more GPU compute cores are activated due to increased utilization).

2 Methodology

The foundation model stack (FMS) at IBM Research provides off-the-shelf tools to train foundation models on OpenShift cloud infrastructure. These are cloud-ready Python scripts to train FM models (such as the RoBERTa model used in NLP) that expose hyperparameters, such as the batch size, so users can tune the training to their liking. We treat these training scripts as *black box* programs to which we attach power measuring profilers that are agnostic to the type of training being performed (see Figure 1).

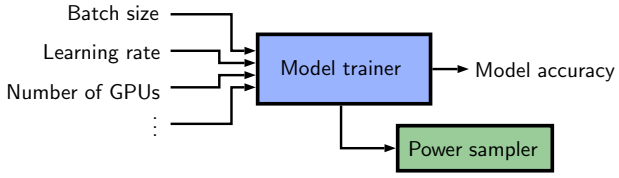
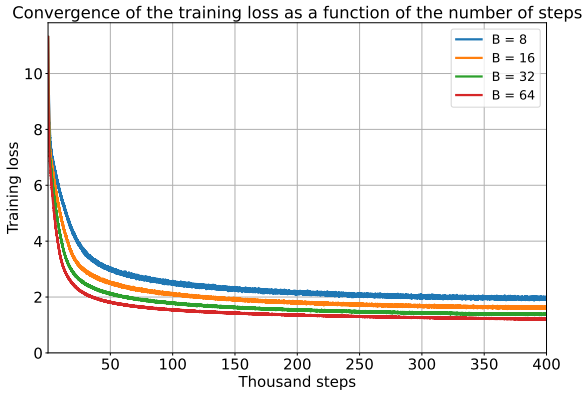


Figure 1: Model training and power measuring architecture

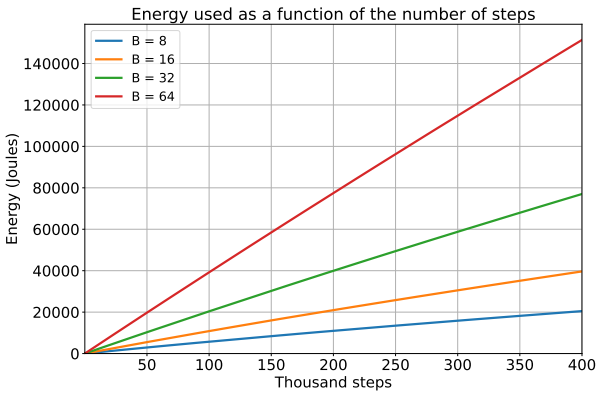
3 Results

For our evaluation in this report, we focus only on the batch size and explore its impact on the model convergence and the energy (power integrated over time) consumption. The setup for this training is as follows:

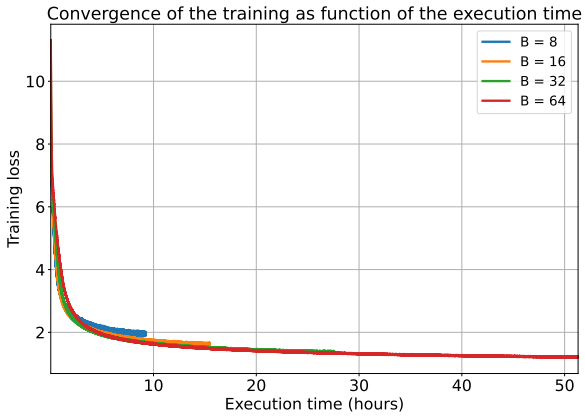
- 1 compute node with: 8 GPUs, 16 CPUs, and 64 GB of memory
- Single POD with one PyTorch container



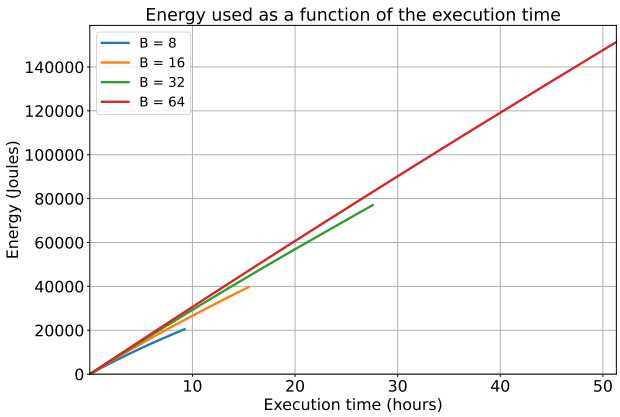
(a) Convergence as a function of the number of steps



(b) Energy used as a function of the number of steps



(c) Convergence as a function of the execution time



(d) Energy used as a function of the execution time

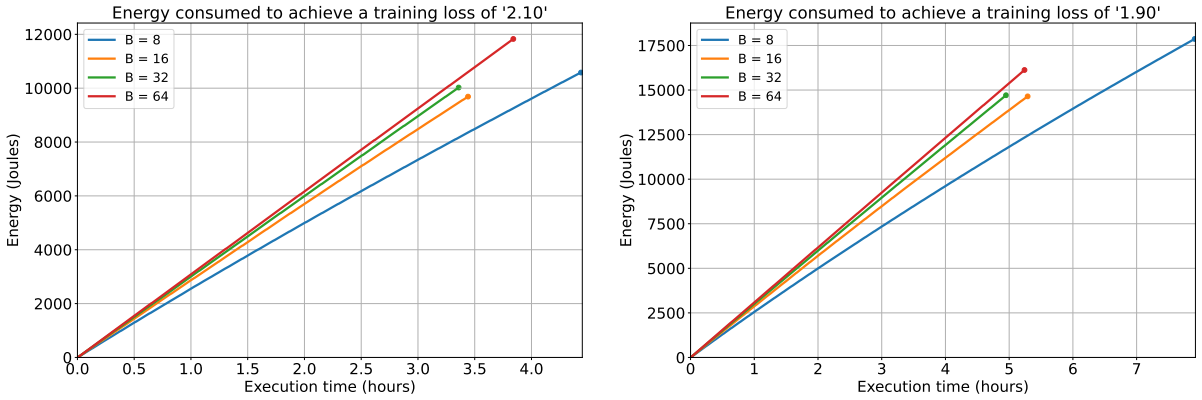
Figure 2: Convergence and energy measurements over the complete unstopped execution (i.e., all $N = 400000$ training steps)

- NCCL communication enabled

The model is trained for exactly $N = 400000$ steps while reporting the training loss every 10 steps and the validation loss every 5000 steps. We first look at the results using the training loss as the accuracy metric. Figure 2 shows the overall convergence over the duration of the training as well as the energy consumed during that training. The top row shows the horizontal axis as the number of steps, while the bottom row shows the horizontal axis as time. As expected, for the same number of steps, larger batch sizes take longer to compute, resulting in longer training jobs and higher energy consumption.

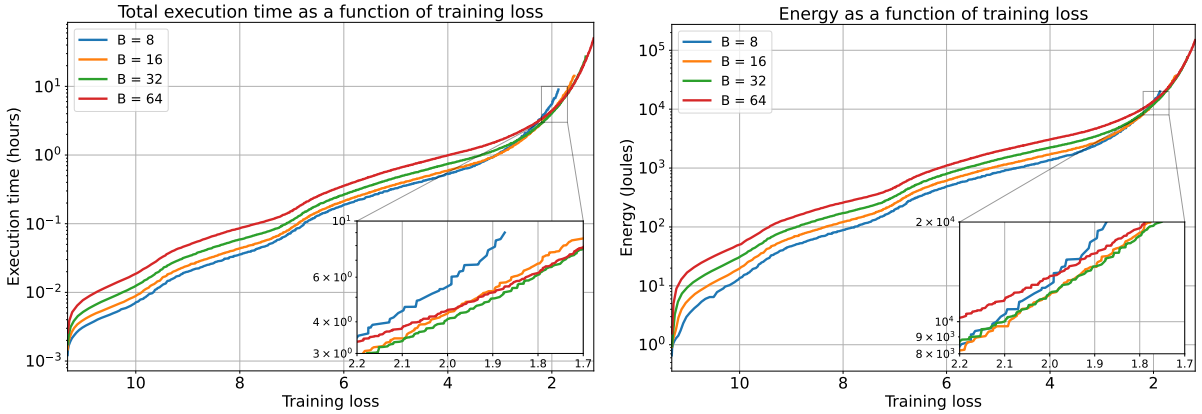
In practice, a training job is stopped once an accuracy *threshold* is reached thus saving time and energy. As shown in Figure 2a, a larger batch size leads to faster convergence as shown by the larger batch sizes reaching lower training losses with fewer number of steps. However, the energy and computational cost-per-iteration (i.e., runtime) is higher with larger batch sizes making the choosing of batch size unclear. In Figure 3 we look at the energy consumed by the training once the algorithm has reached a prespecified threshold.

General wisdom suggests to train AI jobs with the largest batch size possible to improve convergence and resource utilization. This assumption comes at the cost of higher energy consumption as the cost-per-iteration is higher due to the increased computational load. However, as we can observe from Figure 3, it possible to achieve the same levels of accuracy with smaller batch sizes while minimizing the energy consumption. Moreover, general wisdom dictates as well that using a larger batch size could lead to faster execution times as the model converges faster. From the results shown we see that, not only is it energy wasteful to use larger batch sizes, but it can also slow down the training. Our results show it took more time to converge to the desired accuracy for the $B = 64$ results when compared to $B = 16$ and $B = 32$ for threshold 2.1, and $B = 16$ for threshold 1.9. Figure 4 presents the total time and energy



(a) Energy for converging to training loss of 2.1 (b) Energy for converging to training loss of 1.9

Figure 3: Energy measurements for prespecified accuracy thresholds based on the training loss



(a) Execution time as a functions of the training loss (b) Energy as a functions of the training loss

Figure 4: Execution time and energy measurements for a range of accuracy thresholds based on the training loss

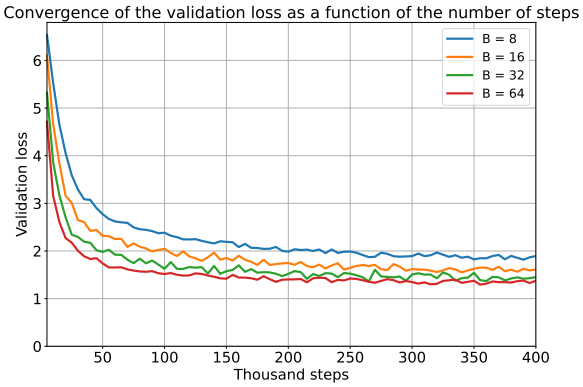
spent for a range of accuracy thresholds. These plots are meant to guide decision choices in terms of what batch size to use for a given threshold so that execution time or energy consumption are minimized. For a desired training loss, these plots can be used to find what batch size gives the best execution time and/or best energy consumption.

4 Conclusions and Future Work

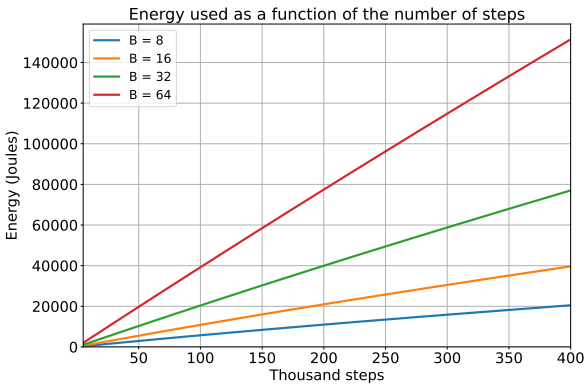
Training and deploying FM tools has become the target of machine learning as incredible results have been achieved with the potential to be improved as larger and larger models are trained. However, it is important to enable this technology in a sustainable way through efficient resource utilization and minimizing energy waste. This study aims to provide guidelines to ML engineers to help them choose how to approach a training job with sustainability metrics in mind, as opposed to just focusing purely on the model accuracy. We saw that, common wisdom may not apply if energy is to be considered as saturating the GPU devices is not necessarily the most efficient way to run a training job. To fully understand the impact of GPU saturation and hyperparameter choices in a more comprehensive way, more studies with different models need to be performed. Furthermore, exploiting hardware constraints such as GPU frequency, can compensate for the increase in computational cost while preserving the model convergence and thus it is another aspect of the training that we aim to look at in the future.

A Accuracy in terms of validation loss

In the results section, we present our analysis based on the *training* loss. A more realistic metric is to use the *validation* loss, which evaluates a partially trained model with a validation dataset for the purpose of understanding how well the training is performing. Contrary to the training loss, the validation loss is not provided to the training algorithm and is meant to provide a more "realistic" view of how well the model does in a real-world deployment. The following plots are the same as the ones before with the validation loss used as the accuracy metric.

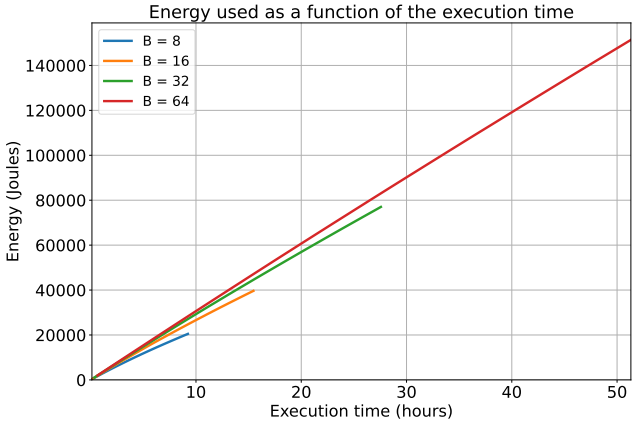


(a) Convergence as a function of the number of steps



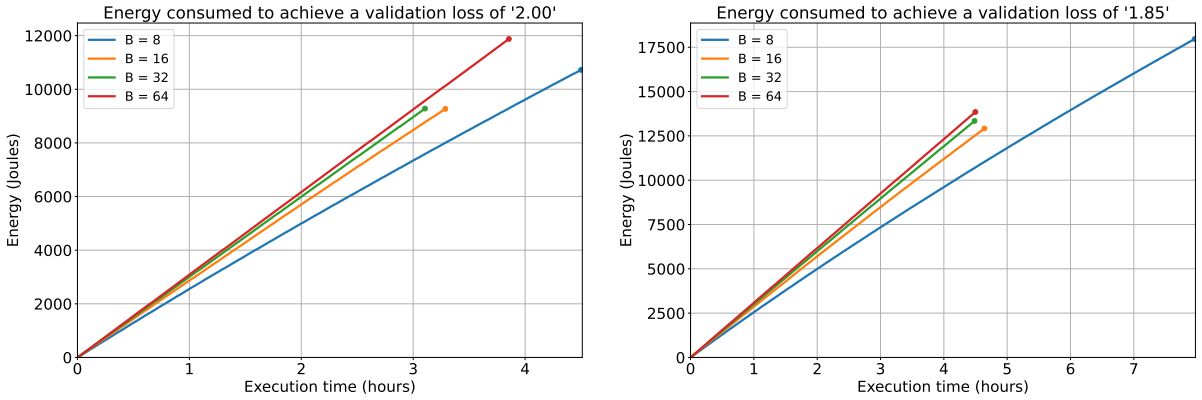
(b) Energy used as a function of the number of steps

(c) Convergence as a function of the execution time



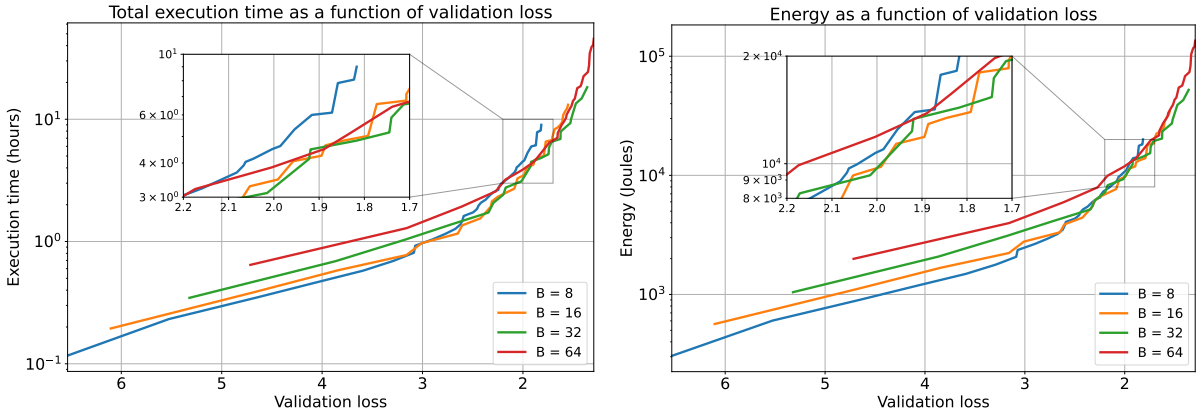
(d) Energy used as a function of the execution time

Figure 5: Convergence and energy measurements over the complete unstopped execution (i.e., all $N = 400000$ training steps)



(a) Energy for converging to validation loss of 2.1 (b) Energy for converging to validation loss of 1.85

Figure 6: Energy measurements for prespecified accuracy thresholds based on the validation loss



(a) Execution time as a functions of the validation loss (b) Energy as a functions of the validation loss

Figure 7: Execution time and energy measurements for a range of accuracy thresholds based on the validation loss