

Beyond FLOPs: Energy-Aware Knowledge Distillation for Sustainable LLMs on Code-Related Task

Enrique Barba Roque ✉️🏠

Delft University of Technology, Delft, The Netherlands

Luís Cruz ✉️

Delft University of Technology, Delft, The Netherlands

Annibale Panichella ✉️

Delft University of Technology, Delft, The Netherlands

Abstract

Background. Large Language Models (LLMs) are increasingly being applied to Software Engineering (SE) tasks, achieving high accuracy across problems such as clone detection, vulnerability prediction, and code summarization. However, their high computational demands and energy consumption raise sustainability concerns and hinder their use on consumer hardware and resource-constrained platforms. A common way to report the computational cost of an LLM in the literature and industry is to use the number of Floating Point Operations (FLOPs) required to perform a pass over the network.

Aims. This paper investigates the implications of energy-aware knowledge distillation for SE, aiming to improve model efficiency while maintaining performance and to determine whether FLOPs is a reliable energy-aware metric.

Method. We conduct a controlled experiment using MORPH, a Many-Objective Optimization-based distillation methodology, to empirically examine whether FLOPs accurately reflect energy consumption in Clone Detection and Vulnerability Prediction tasks. We extend this methodology to include energy-surrogate models that directly estimate CPU and GPU energy consumption during optimization, and we apply MORPH to generative tasks using CodeT5+ for code summarization.

Results. Our results show that FLOPs is not always a reliable indicator of energy consumption, and better results can be achieved by using energy-surrogate models. Distilled student models can reduce inference energy consumption by up to 90% and memory usage by 86%, with only modest accuracy trade-offs.

Conclusions. Energy-aware knowledge distillation when guided by direct energy surrogates rather than FLOPs can improve the energy consumption, sustainability, and deployability of LLMs for SE applications, enabling efficient models on consumer hardware.

2012 ACM Subject Classification Computing methodologies → Genetic programming; Computing methodologies → Natural language generation; Hardware → Impact on the environment

Keywords and phrases Knowledge distillation, Green AI, Many-objective Optimization, LLMs for Code, FLOPs, AI for SE

Digital Object Identifier 10.4230/LIPIcs.ESEM.2026.22

Category Technical Track Paper

Supplementary Material

Software (Replication Package): <https://doi.org/10.5281/zenodo.20270810>



© Enrique Barba Roque, Luís Cruz, and Annibale Panichella;
licensed under Creative Commons License CC-BY 4.0

20th International Symposium on Empirical Software Engineering and Measurement (ESEM 2026).

Editors: Robert Feldt, Maria Paasivaara, Daniel Mendez, Stefan Wagner, and Marvin Muñoz Barón;

Article No. 22; pp. 22:1–22:21



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

Large language models (LLMs) have recently emerged as a highly useful tool for tasks that require language understanding and generation. Among them, LLMs are capable of supporting an extensive number of Software Engineering (SE) tasks [12], including both classification (e.g., clone detection [20], vulnerability prediction [48]) and generation tasks (e.g., code summarization [1] or code completion [15, 23]). Their capabilities come from massive scales: billions of parameters trained on terabytes of open-source code from platforms such as GitHub [25]. This scale poses significant computational challenges, including the need for large datacenters with dedicated hardware for parallel computation, such as GPUs [44]. Powering these datacenters requires massive amounts of energy: the International Energy Agency projects that datacenters' energy consumption will double over the next 5 years [17], further straining existing electrical infrastructure and increasing greenhouse gas emissions and environmental costs [34].

Optimizing LLM energy consumption requires addressing where and how operational costs accumulate. Because inference dominates lifetime energy usage for high-volume models [10], relying on large, general-purpose models over long sequences incurs substantial computational and energetic overhead [27]. To mitigate this, model compression reduces the compute and memory footprint per query, directly lowering operational energy demands. For SE workflows, such compressed models provide a dual benefit: they minimize runtime power consumption and enable lightweight, local deployment directly inside IDEs, avoiding energy-intensive calls to remote cloud servers [29, 40].

A promising technique that can generate smaller, more efficient models addressing these three aspects is knowledge distillation (KD) [16], where a smaller model (*student*) learns the implicit knowledge of a larger model (*teacher*) for a downstream task. By learning to mimic the teacher's output probability distribution, the student model maintains similar accuracy while using considerably fewer resources. In the context of SE tasks, Panichella [29] proposes MORPH, a method that compresses BERT models to smaller sizes using evolutionary algorithms and predictive models (or surrogates) to improve accuracy and metamorphic robustness. This method achieves state-of-the-art performance on clone detection and vulnerability prediction tasks, surpassing prior work on this topic [35].

While these results are promising, two aspects limit their generalizability. First, MORPH relies on Floating Point Operations (FLOPs) as a proxy for energy consumption. The approach aims to minimize the FLOPs used by a model to reduce energy requirements. FLOPs measure the number of floating-point operations required for a training or inference pass and can be derived directly from a model's architecture and hyperparameters. Their simplicity and hardware-agnostic nature make FLOPs attractive, and they have therefore been widely used in the Machine Learning literature as a stand-in for energy consumption in LLM research [18, 24, 22].

However, the validity of FLOPs as a proxy for energy consumption remains debated in the literature, where their correlation to energy usage is heavily dependent on the context. Most FLOPs estimates – such as those from common profiling tools¹ – capture only the theoretical count of mathematical operations (e.g., matrix multiplications, additions) while ignoring runtime factors that strongly influence energy usage, including memory access patterns, data movement overheads, and hardware-level optimizations on GPUs or TPUs [10, 2, 9]. As a result, FLOPs may not reliably reflect actual energy consumption.

¹ <https://docs.pytorch.org/docs/stable/profiler>

Second, the evaluation for MORPH is limited to BERT-like models compressed from up to 500 MB to 3 MB. Such size reductions do not necessarily yield proportional energy savings, as energy is subject to diminishing returns due to baseline power consumption or memory bandwidth saturation. Furthermore, BERT is now relatively dated, while newer architectures such as CodeT5 [43] and Starcoder 2 [25] have since emerged, raising questions about generalizability. Moreover, the experiments focus only on two binary classification tasks. More complex tasks involving generation not only require students to capture deeper semantic knowledge but also involve multiple forward passes during autoregressive decoding. This substantially increases inference costs and makes energy estimation more challenging.

To address these limitations, we first revisit the efficiency assumptions behind MORPH and then extend the method to optimize for measured energy directly and to handle generation tasks. Concretely, we ask the following research questions:

RQ₁: *Are FLOPs a good proxy of energy consumption in the context of model distillation using Many Objective Optimization?*

To answer this question, we replicate the experiments from the original paper and measure energy during the training and inference phases for different student models with varying FLOPs, using an energy profiler to capture CPU and GPU energy. Statistical testing shows that FLOPs do not correlate well with energy measurements from the profiler, and that other factors, such as specific hyperparameters, may provide stronger correlations with energy.

Based on these results, we modify MORPH to improve its energy estimation. Instead of relying on FLOPs, we introduce measured energy (from the profiler) as a new optimization objective and build a surrogate model to predict energy consumption from hyperparameter combinations. This naturally motivates our next question:

RQ₂: *How energy efficient are MORPH models when using actual energy as an objective?*

This lets MORPH search for configurations that are efficient with respect to observed hardware behavior rather than architectural estimates alone.

Finally, we examine whether Many-Objective Optimized Distillation can be effective beyond classification, in more complex generative tasks that may require deeper semantic knowledge. To investigate this, we extend the MORPH methodology with a new distillation pipeline and investigate:

RQ₃: *How effective is MORPH at reducing LLMs costs for generation tasks, and what is the accuracy tradeoff?*

We evaluate the modified method with CodeT5+ [43], distilling models for code summarization with energy as an additional optimization objective. We compare the resulting students with the teacher model in terms of accuracy (ROUGE-L), size, and energy consumption.

In summary, this paper makes the following contributions:

- Empirical study of FLOPs vs. energy. We replicate MORPH’s experiments and show that FLOPs correlate inconsistently with actual energy consumption in SE tasks.
- Energy-aware extension of MORPH. We introduce measured energy as an optimization objective, with surrogate models to estimate energy from hyperparameters.
- Application of the extended method to CodeT5+ for code summarization, achieving up to 90% lower energy and 86% smaller memory footprint, with modest accuracy trade-offs.

2 Background

Knowledge distillation [16] is a technique for compressing large pretrained models into smaller, more efficient models, ideally with minimal impact on accuracy. The smaller model serves as the student, and the larger model acts as the teacher. Instead of learning directly from

the dataset, the student model learns to predict responses similar to the teacher's. To do this, the training dataset is fed into the teacher and the student, and the student's loss is computed to minimize the distance between the student's and teacher's output probability distribution. Distillation can be done with generic pretraining tasks, to compress general-purpose models [8], or as task-specific distillation, where the student learns from a teacher finetuned to perform a specific task, and can be used for the same task without any additional finetuning.

Knowledge distillation brings multiple advantages. The dataset size required for distilling a student model is smaller, as is the number of epochs needed to reach convergence, which lowers training costs. However, it requires an already pretrained, high-performing teacher. Additionally, for task-specific distillation, depending on the task's complexity, some pretrained weights are required for the student model. For example, in binary classification tasks, the student model's weights can be randomly initialized, whereas more complex tasks, such as generation, may require pretraining or copying weights from the teacher.

On top of this, finding the optimal hyperparameter set for the student model can be challenging. The possible combination of hyperparameters, such as the number of layers or attention heads, makes the search space extremely large. To tackle this, several approaches model this search as an optimization problem with one or more objectives, and the different hyperparameters as variables. The state-of-the-art implementation MORPH [29] employs a Many-Objective optimization approach to compress models for two binary classification SE tasks: Clone Detection and Vulnerability Prediction. It defines 4 objectives for the student model: maximum accuracy, minimum memory footprint, minimum FLOPs, and maximum robustness. For robustness, the paper uses metamorphic testing, which involves creating new data samples by applying transformations to the input data that preserve the code's functionality and semantics, such as replacing English words with synonyms or expanding acronyms. Robustness is measured using prediction flips: given a code sample modified by metamorphic transformations, prediction flips determine whether the model maintains the original prediction.

To solve the Many-objective optimization problem, MORPH employs AGE-MOEA [28], an evolutionary algorithm that evolves an initial population of solutions and converges towards the Pareto front. In Many-objective optimization, the Pareto front is the set of solutions that achieve optimal trade-offs among the objectives. These are solutions in which improving one objective would negatively impact the others. To explore the population space without distilling and evaluating every possible student, MORPH employs surrogate models to improve accuracy and reduce prediction flips. By uniformly sampling the configuration space and training a set of student models, it builds two surrogate models to predict the accuracy and prediction flips of a given configuration, without training that configuration

3 Related Work

Many-Objective Knowledge Distillation for LLMs. Multi- and Many-Objective optimization for Knowledge Distillation is a relatively recent technique that explores the configuration space of student models to optimize for two or more objectives, including maximizing accuracy and minimizing computational cost and size. The first proponent was Shi et al. [36] with COMPRESSOR, which used a single-objective genetic algorithm to distill binary classification models, minimizing a combination of Giga FLOPs and model size. Later, they introduced AVATAR [35], which uses a multi-objective genetic algorithm with 3 objectives: minimize FLOPs and model size, and maximize accuracy.

MORPH [29] builds on top of this previous work, introducing robustness as an additional objective, which is done using metamorphic testing, as explained in Section 2. With its additions – metamorphic testing, surrogate models, and Many-Objective optimization – MORPH achieves state-of-the-art performance compared to AVATAR, but leaves some questions unanswered. First, it uses FLOPs as an optimization objective for energy and sustainability, as does the previous work. However, the relationship between FLOPs and energy consumption is not straightforward and depends on factors such as the target hardware and the potential optimizations it enables. In this work, we aim to determine whether FLOPs is an adequate objective to minimize when reducing the energy consumption of a distilled model.

Another topic these approaches do not address is their validity for generation tasks. Previous work only performs distillation for binary classification tasks, namely, Clone Detection and Vulnerability Prediction. These tasks are much simpler than generation tasks and require less complex models, allowing for more aggressive compression during distillation. In this paper, we will apply this technique to a generation task, Code Summarization, and investigate whether the advantages it offers can be applied to generation tasks.

FLOPs as proxy for energy consumption. The validity of floating-point operations (FLOPs) as a proxy for an AI model’s energy consumption is an ongoing debate in the field. FLOPs were first proposed as an energy proxy in the early stages of Green AI [34, 21] because they can be easily calculated solely from architectural details. Desislavov et al. [10] perform a study on inference energy consumption for a set of AI models for Computer Vision and Natural Language Processing tasks. Here, the authors propose using FLOPs as a proxy for energy consumption, relating the FLOPs used by a model during inference to hardware FLOPs and combining them with GPU hardware specifications, specifically TFLOP/s and power. However, this study fails to address the gap between logical and hardware FLOPs, as it does not account for potential hardware optimizations enabled by GPUs or TPUs.

Asperti et al. [2] investigated the validity of FLOPs as a proxy metric for energy consumption of Convolutional Neural Networks (CNN). They show that FLOPs do not correlate well with energy consumption and propose a metric, α -FLOPS, that accounts for hardware and model architecture to estimate energy more accurately. However, this work focuses solely on CNNs and does not consider other architectures, such as transformers, widely used by LLMs. A similar study by del Rey et al. [9] on multiple Deep Neural Network architectures shows that FLOPs might be a better estimator of GPU usage than energy consumption.

A gap remains regarding whether the correlation between FLOPs and energy consumption observed in previous studies is still applicable to LLMs. Despite this, many LLM-related papers still use it to reflect energy costs of models, as seen in Section 3 with MORPH, but also many others [18, 24, 22]. This paper aims to shed light on this topic and investigate whether FLOPs are a viable proxy for energy consumption in certain LLM tasks.

4 Methodology

This section introduces the changes applied to the original MORPH methodology to investigate our research questions. For **RQ₁**, we maintain the original approach, and empirically measure energy usage of the distilled models in the Pareto front. For **RQ₂**, we extend this approach by adding energy as an additional objective in the function. For **RQ₃**, we take this one step further and use our modified approach for a more complex generation task.

4.1 Adding Energy as Optimization Objective

For research question **RQ₂**, we closely maintain the original methodology steps. The only change we introduce is to redefine the efficiency function, using energy rather than FLOPs to measure a model’s energy efficiency.

To do this, we adapt the surrogate model methodology proposed in the original paper for predicting accuracy and the number of prediction flips. We sample a representative subset of the hyperparameter subset using Latin Hypercube Sampling (LHS), which provides a uniformly distributed set of configurations across the sampling space. For each hyperparameter set, we assess median CPU and GPU energy consumption across multiple runs, following the protocol described in Subsection 5.4, along with accuracy and prediction flips. Then, we use the energy measurements to train two additional regression models to predict GPU and CPU energy consumption for a given set of hyperparameters.

During the optimization phase, instead of computing and minimizing FLOPs for the sampled population, we predict CPU and GPU energy consumption and use the sum as our new minimization objective. If the surrogate models are sufficiently accurate, this serves as an alternative proxy metric for energy.

4.2 Morph for Generation

To investigate **RQ₃**, we adapt MORPH’s objective function to work for a generation task. Concretely, we will test MORPH and CodeT5+ performance for the Code Summarization task. To evaluate the efficacy of a student model for this task, we want the student model to produce summaries as close as possible to those in the dataset, while minimizing energy usage and model size. Therefore, we redefine the problem and objective function as follows:

Definition: Given a LLM teacher model, find a hyperparameter configuration $C = [c_1, \dots, c_m]$ for the student model S that optimizes the following objectives:

$$\begin{cases} \min O_1 = & \text{size}(S) \\ \min O_2 = & \text{efficiency}(S) \\ \max O_3 = & \text{effectiveness}(S, V) \end{cases}$$

where V is the validation set. The main changes to the original formulation lie in the lack of a robustness metric and the measurement of efficiency and effectiveness. To measure effectiveness, we require metrics that measure similarity between generated texts and ground truth. Ideally, we would compare the summaries’ semantic similarity using human judgment or an LLM serving as a judge. However, since the methodology requires training and testing a relatively high number of models, this would significantly increase the time and resources needed. Therefore, we select an ngram-based metric. To partially mitigate the lack of human judgment, we select ROUGE-L as our accuracy metric. ROUGE-L has been shown to align better with human judgment [11] than other state-of-the-art metrics, such as BLEU, METEOR, or CodeBLEU, for code-related tasks.

To assess robustness, the original paper used the number of prediction flips as a metric, since it dealt with binary classification tasks. In the case of summarization, robustness is much more difficult to assess, since there is not a single correct summary for a piece of code. Furthermore, introducing metamorphic code changes, such as synonymizing parameter names, can yield the same changes in the summary, maintaining semantic similarity while using different words. This could also potentially be addressed with human or LLM judgment, but such techniques do not fit the size of these methodologies.

Finally, to measure efficiency, we will use energy measurements of the inference process in Joules, rather than FLOPs, following the process defined in the previous section. Using the same inference dataset, we measure the energy required for each student to complete it.

To create the surrogate models, we follow the technique described in Section 4.1 for the first two research questions and GraphCodeBERT. We sample the hyperparameter space using LHS, distill the models, and measure their ROUGE-L scores on the validation set and their inference energy consumption.

Weight subcloning. Because clone detection and vulnerability prediction are binary classification tasks, the student model weights were randomly initialized, as the number of epochs required for convergence is relatively small. However, for summarization, the implicit knowledge the student model needs to learn is larger, and convergence takes longer with random initialization. Additionally, in our preliminary studies, we observed that randomly initialized students converge to generic responses independently of the input. To solve this, we clone part of the teacher model’s weights into the student model before starting the distillation process.

To copy the most important weights, we apply a weight subcloning technique [33]. This technique runs inference on the teacher model over a small subset of the validation set (around 80,000 tokens) and captures the input and output magnitudes of different neurons and attention heads. Based on these inputs and outputs, it ranks neurons and layers by importance, and copies the most important weights into the student model, respecting the new architecture. According to the study, student models initialized this way achieve 4x faster training times.

Teacher Logit extraction and Distillation Loss. In the original MORPH for binary classification tasks, inference is run on the teacher over the training set, and the two logits per sample (positive probability, negative probability) are saved. These logits are later used during offline distillation, where only the student model needs to be instantiated in memory.

Compared to binary classification, generative tasks can be viewed as classification over the vocabulary size at each output position, making it infeasible to store logits. In our preliminary experiments, the logits for only 64 training samples already required about 1 GB of disk space. Since it is unfeasible to store all the logits, we perform online distillation. We instantiate the teacher and student models on the same GPU and, for each batch, first perform inference on the teacher to obtain the logits, then perform the training step on the student. While this helps avoid storage problems with logits, it increases computational demand, since we need two models instantiated and running inference simultaneously.

To compute the loss between the student output, teacher output, and ground truth, we used the average of the Kullback-Leibler Divergence (KLD) and cross-entropy loss. KLD is a loss function to compute the difference between two probability distributions. In this case, it takes the student and teacher logits and compares the probability distributions for each output token across the entire vocabulary. This loss is the standard used across Knowledge Distillation tasks [47, 3] and it has been shown to work well with LLMs [45].

To ensure the student models remain aligned with the ground truth, we include the cross-entropy loss between the student predictions and the ground-truth probability distribution.

5 Evaluation

The goal of this study is twofold: (1) to validate the use of MORPH for optimizing LLMs with respect to energy consumption, and (2) to assess its applicability to generation tasks.

- **RQ₁**: *Are FLOPs a good proxy of energy consumption in the context of model distillation using Many Objective Optimization?*
- **RQ₂**: *How energy efficient are MORPH models when using actual energy as an objective?*
- **RQ₃**: *How effective is MORPH at reducing LLMs costs for generation tasks, and what is the accuracy tradeoff?*

The objective of **RQ₁** is to empirically assess the validity of FLOPs as a proxy for energy consumption by examining their correlation in models generated by MORPH, thereby identifying potential weaknesses in the existing distillation approaches for code-specific LLMs [29, 35]. **RQ₂** evaluates our proposed modifications to MORPH, which incorporate energy directly into the optimization. The goal is to determine whether these models are more energy-efficient than those distilled using the standard approach.

Finally, the objective of **RQ₃** is to verify the applicability of our modified MORPH method to generation tasks for software engineering. The original MORPH approach is only tested for binary classification tasks. Text generation is a much more complex task, and achieving results similar to those presented in the original paper might not be possible. With this question, we aim to verify whether we can train LLM models suitable for generation tasks that preserve accuracy while being significantly smaller and more energy-efficient.

5.1 Downstream tasks

For **RQ₁**, to keep alignment with the MORPH experiments [29], we evaluate energy consumption of MORPH models on the same software engineering tasks and datasets: Clone Detection and Vulnerability Prediction. With this, we can replicate previous results exactly and draw valid comparisons. For the clone detection task, the dataset used is BigCloneBench [39], a collection of Java code clones obtained by mining 25,000 Java projects from SourceForge and Google Code. For vulnerability prediction, the dataset used is Devign [49], which contains C code from two open-source libraries, Ffmpeg and Qemu. As explained in previous research, this selection avoids data leakage problems [29].

As a generation task, we chose Code Summarization. We chose this task because it is quick to evaluate (does not require running test cases, as in code generation) and provides an objective score quickly via ROUGE-L. However, it is still a relevant task that is used for automated commit messages or pull requests. We use two separate datasets to avoid data leakage problems. For finetuning the teacher, we use the Python split of the CodeXGLUE code summarization dataset [26]. This dataset is extracted from CodeSearchNet and comes from publicly available, open-source GitHub repositories.

However, CodeT5+, the model used for summarization, has been pretrained on CodeSearchNet data, so we cannot use the validation and test splits of CodeXGLUE for evaluation of the model due to data leakage. To solve this problem, we will extract validation data from The Heap [19]. This is a contamination-free dataset extracted from GitHub projects with non-permissive licenses and has been de-duplicated against the most popular training datasets. Using a Python AST parser, we extract functions with docstrings and use them as our evaluation dataset.

5.2 Large Language Models

We employ two pretrained models for these experiments, GraphCodeBERT [14], and CodeT5+ [43]. GraphCodeBERT is a bimodal model pretrained on natural-language and programming-language data, based on BERT and CodeBERT [13], and extends the latter’s capabilities by incorporating information from the code snippet’s data flow graph. In the original MORPH study [29], both CodeBERT and GraphCodeBERT are tested. However, for $\mathbf{RQ}_1$, based on the architecture similarity and the study’s results, we replicate the results only for GraphCodeBERT, since the conclusions from this should be applicable to both models.

For the generation tasks, we use CodeT5+ [43]. This model is based on the Google T5 model and has been pretrained on multiple code-related tasks, including code summarization, using the CodeSearchNet and BigCode datasets. We chose this model because it is more recent than GraphCodeBERT and offers better capabilities. This family also provides small versions, from which we choose the model with 220M parameters. This choice is motivated by our hardware constraints and the need to perform online distillation with 2 models loaded simultaneously. Our experiments also require multiple distillations to create surrogate models and multiple energy measurement runs. Using a smaller model reduces the execution times of the experimental pipeline.

5.3 Empirical Methodology

To answer $\mathbf{RQ}_1$, we sample models with varying FLOPs, measure their energy consumption, and analyze the correlation between the two. To obtain these models, we generate a Pareto front of optimal solutions using the original MORPH approach. In many-objective optimization, the Pareto front consists of solutions for which no objective can be improved without sacrificing at least one other objective. Therefore, sampling the Pareto front will yield a diverse set of models with varying FLOP requirements. After measuring the energy consumption of these models, we use statistical methods, such as ANOVA, to identify correlations among energy, FLOPs, and other hyperparameters.

To answer $\mathbf{RQ}_2$, we compare the models generated by the original MORPH methodology with a modified method that substitutes the FLOPs objective with actual energy consumption predictions, using the surrogate model to predict energy consumption in inference, similar to the surrogate method used for prediction flips. We compare both methodologies using the metrics from the original paper (*size*, *accuracy*, *robustness*, and *FLOPs*) as well as *inference energy consumption* to determine whether models actually consume less energy during inference. We run the original MORPH and the modified version, generating new Pareto fronts and models for both cases. Given the random nature of the MOP optimization algorithm, we distill 20 models and report the median results for the previous metrics, along with the interquartile range. To compare with the original MORPH results, we use the same model selection criteria, choosing the model with size closest to 3MB. We analyze the results using the Wilcoxon signed-rank test [5] to determine whether there is a statistically significant difference in energy consumption between standard and modified MORPH, and the Vargha-Delaney $\hat{A}_{12}$ statistic [42] to determine the effect size.

Finally, to answer $\mathbf{RQ}_3$, we distill a CodeT5+ model using MORPH and the energy objective, and compare the student model with the teacher model. We compare based on three metrics: accuracy (using ROUGE-L), size, and energy efficiency. We also manually inspect some test samples to evaluate the models’ performance based on human judgment.

5.4 Energy Measurement Setup

For our experiments, we measure only energy usage during the inference step. We exclude tokenization and decoding since they depend on the tokenizer, which is not affected by the distillation. Measuring the energy consumption of an AI model is difficult because software cannot run in a vacuum; it depends on its environment: operating system, dependencies, memory, and hardware. Additionally, the hardware can be affected by external constraints, such as room temperature. Consequently, obtaining a precise measurement with a single execution is not guaranteed.

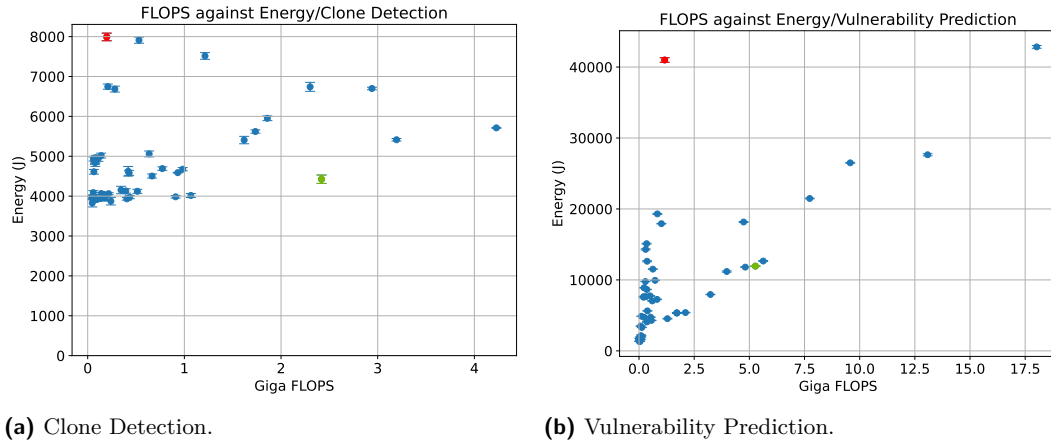
We follow existing guidelines to mitigate external factors [6]. The first is to make execution times long enough to reduce variance in energy measurements. Accordingly, we assess inference energy of the distilled models using an automated script that: (1) loads the model and test dataset, and (2) runs inference on the dataset multiple times. The multiple inferences are run so the energy costs of loading the model do not overshadow the inference costs. Another mitigation is to repeat measurements for each model in a randomized order, with short pauses in between, to reduce variation caused by unexpected system activity. Following the guidelines, we perform 20 executions [6] and report the median energy consumption across runs. Between measurements, we reset the GPU context to ensure no cached results affect the measurements.

The experiments are run on a server equipped with an AMD Ryzen 9 7900X processor, 64 GB of RAM, and an NVIDIA GeForce RTX 4090 GPU. The server runs Ubuntu 22.04.3, with Linux kernel version 6.2.0, and Docker 24.0.5. To measure CPU and GPU energy consumption, we use EnergiBridge [32], a software profiling tool compatible with AMD CPUs and Nvidia GPUs, among others, that uses RAPL and NVML under the hood.

5.5 Parameter settings

For the components of the methodology that remain unchanged from the original implementation, we maintain the parameter configuration from the original paper [29]. This includes the parameters for population creation and mutation, optimization algorithm, and hyperparameter space for GraphCodeBERT. We also replicate the configuration of the original paper for the new energy surrogate models, using Gradient Boosting Regression, and grid search to optimize GBR hyperparameters across the following predefined parameter space: (1) number of decision trees $\in \{50, 100, 150\}$; (2) learning rate $\in \{0.05, 0.10, 0.15, 0.20, 0.25\}$; (3) maximum depth of each decision $3 \in \{2, 3, 4, 5\}$; (4) the minimum number of samples in internal node $\in \{2, 3, 4\}$; (5) minimum number of samples in leaf nodes $\in \{1, 2, 3\}$.

For Code Summarization distillation in CodeT5+ we use the following configuration space: (1) number of hidden layers $\in [1, 12]$; (2) hidden activation $\in \{gelu, relu, silu, gelu-new\}$; (3) number of decoder layers $\in [1, 12]$; (4) hidden size $\in [16, 768]$; (5) number of attention heads $\in [1, 12]$; (6) projection size $\in [1, 64]$; (7) intermediate size $\in [16, 3072]$; (8) relative attention buckets $\in [4, 32]$; (9) relative attention max distance $\in [32, 128]$; (10) dropout rate $\in \{0.1, 0.2, 0.3\}$; (11) feed forward projection $\in \{relu, gated-relu\}$; (12) learning rate $\in \{0.001, 0.0001, 0.00005\}$; (13) batch size $\in \{8, 16\}$. This configuration's maximum values are equivalent in size to the teacher model (CodeT5+ 220M). This way, student models will be no larger than the teacher model. We chose this because, a priori, we do not know the level of compression that can be achieved for this task and model, and it may be that the maximum accuracy lies in the teacher's model size.



(a) Clone Detection.

(b) Vulnerability Prediction.

Figure 1 Inference energy usage against FLOPs.

Additionally, compared to the original paper, we removed hyperparameters for the tokenizer, since modifying it would require retraining the teacher model, since the number of logits equals the vocabulary size for each token.

Finally, for inference during evaluation, we use beam-search multinomial sampling with 5 beams.

6 Results

6.1 RQ1: Correlation of FLOPs and Energy Consumption

We ran MORPH with its default configuration and extracted details of the Pareto front for the Clone Detection and Vulnerability Prediction tasks. For each Pareto front, we ran energy tests on inference, using the same test set for every model and testing each model multiple times in a randomized order, as explained in Section 5.4. Figure 1 shows a scatterplot of total inference energy consumption vs FLOPs for the Pareto Front of the Clone Detection (Figure 1a) and Vulnerability Prediction (Figure 1b) tasks.

For the Clone Detection task, the figure shows no clear correlation between FLOPs and Energy. Most models with fewer Giga FLOPs accumulate around the 4,000 J mark, but there are many models that do not show a straight relationship between FLOPs and energy. For example, the most expensive model uses around 8,001 J while their computational cost is 0.20 Giga FLOPs (highlighted in red), while a model with 2.4 Giga FLOPs uses 4,423 J, around 44 % less energy (highlighted in green).

For the Vulnerability Prediction task (Figure 1b), the results reveal a different behavior and show a much clearer correlation between FLOPs and energy consumption, with a higher number of FLOPs translating into higher inference energy. However, despite a clearer correlation, there is again no straightforward relationship between FLOPs and energy consumption. Similarly to Clone Detection, some models use significantly less energy than others with higher FLOP counts. For example, a model with 1.16 GFLOPs is using 41,122 J (highlighted in red), while a model with 5.27 GFLOPs is using 11,945 J, or 70% less energy (highlighted in green). Relying solely on this correlation may obscure outlier models that are far more energy efficient or, conversely, favor models that are substantially less efficient.

■ **Table 1** ANOVA test results against inference energy for Clone Detection (CD) and Vulnerability Prediction (VP). Grey denotes significance. For CD, all models in the Pareto front have Num Hidden Layers = 1.

Hyperparameter	CD	VP
Tokenizer	0.06	0.82
Vocabulary Size	0.03	0.04
Num Hidden Layers	–	8.26e-11
Hidden Size	2.95e-5	1.89e-4
Hidden Activation	0.01	2.23e-4
Number of Attention Heads	3.69e-14	6.54e-09
Max Sequence Length	6.97e-08	1.71e-05
Position Embedding Type	3.71e-3	0.11
Learning Rate	0.53	0.906290
FLOPS	0.16	6.48e-07

Additionally, during our experiments, we tracked and made surrogate models for both CPU and GPU energy. However, we observed that CPU power usage is mostly constant, and total energy only depends on inference time. This means that the bulk of the processing was happening on the GPU. Higher CPU energy comes only from measuring the same power usage over a longer period. Therefore, for these results, we report total energy consumption (CPU + GPU), since the CPU’s energy consumption is relevant but does not show a clear trend on its own.

Table 1 reports the results of the ANOVA test for inference energy against the different hyperparameters of the model, with p -values < 0.05 indicating a statistical correlation between inference energy and the hyperparameter. For Clone Detection, the ANOVA test results confirm that there is no statistical correlation between FLOPs and energy, with a p -value=0.17. Additionally, the results reveal that other hyperparameters, such as Hidden Size, Number of Attention Heads, and Max Sequence Length, show a high correlation with energy (p -value $\ll 0.05$). However, despite these hyperparameters being part of the FLOPs computation, their correlation with energy does not transfer to the FLOPs metric. From the Figure, it is clear that higher FLOPs do not always translate to higher energy consumption.

For the Vulnerability Prediction task, the ANOVA results confirm the correlation observed in the figure, with a p -value of $6.48e - 07$ for the FLOPs metric against energy consumption. The test also confirms the correlations with the other hyperparameters already observed in Clone Detection.

The results for both tasks suggest that FLOPs can be a somewhat accurate proxy for energy consumption, but other factors that affect energy consumption are not reflected in this metric. In the original MORPH methodology, the same FLOPs computation algorithm taken from Clark et al. [4] is used for both classification tasks. This suggests that FLOPs is not a task-agnostic proxy for energy consumption and fails to capture certain behaviors in the Clone Detection task that strongly influence energy consumption.

Answer to RQ₁. For Clone Detection, FLOPs fail as an energy proxy ($p = 0.17$), while for Vulnerability Prediction FLOPs are statistically associated with energy ($p = 6.48 \times 10^{-7}$). This task-dependent contrast shows that FLOPs are not a reliable task-agnostic proxy, and the observed spread of models with similar FLOPs but different energy confirms that FLOPs are not directly proportional to energy consumption.

6.2 RQ2: Substituting FLOPS with surrogate energy prediction

■ **Table 2** Results of Morph using FLOPs and surrogate energy as optimization objectives. The best result for each metric is highlighted in grey.

Task	Objective Metric	Accuracy (%)		Size (in MB)		# Pred. Flips		Giga FLOPs		Energy (J)	
		Median	Diff.	Median	Diff.	Median	Diff.	Median	Diff.	Median	Diff.
Clone	FLOPS	95.26 (2.42)		3.00 (0.01)		43.00 (14.25)		0.79 (0.36)		5355.83 (1391)	
Detection	Energy	95.95 (0.97)	+6.09%	3.01 (0.06)	+0.01	37.00 (27.00)	-57%	0.98 (0.24)	+21%	3604.50 (824)	-39%
Vulnerability	FLOPS	55.60 (4.77)		2.98 (0.11)		143.00 (257.00)		0.68 (0.47)		10505.50 (6762.75)	
Prediction	Energy	56.20 (4.50)	-0.27%	2.99 (0.13)	+0.01	218.00 (260.00)	-33%	0.98 (0.20)	+36%	14219.00 (8568.75)	+30%

We apply MORPH to produce 20 student models optimized for FLOPs and energy, respectively. Due to the method’s stochastic nature, each model yields different trade-offs. Table 2 reports median and interquartile range for accuracy, model size, prediction flips, GFLOPs, and energy. As per Section 5.3, we select the model with size closest to 3MB.

For accuracy and prediction flips, neither optimization strategy significantly outperforms the other. Energy optimization yields slightly higher accuracy, but the difference is not statistically significant (p -value >0.3) for both Clone Detection and Vulnerability Prediction. Conversely, FLOPs optimization shows fewer prediction flips, but again the difference is not significant (p -value >0.5).

An opposite but similar effect occurs with robustness: the original FLOPs optimization outperforms energy optimization in of prediction flips (e.g., better model robustness). However, this difference is again not statistically significant (p -value >0.5) for both tasks.

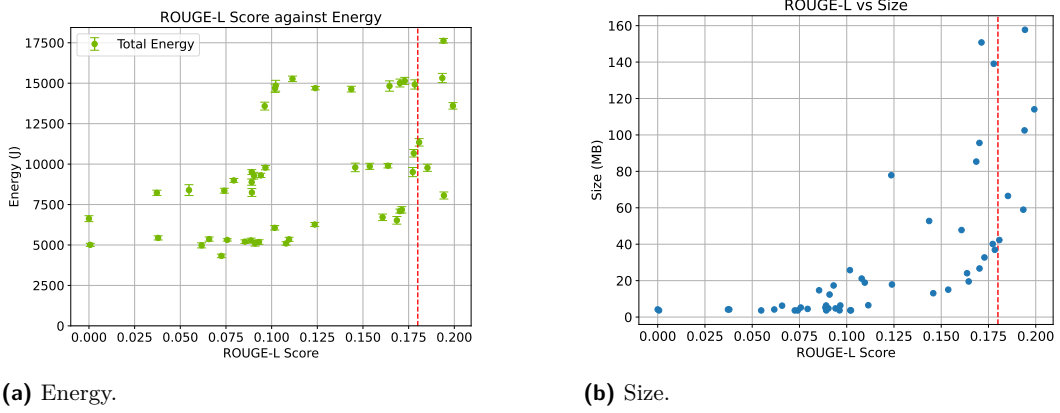
Finally, when looking at sustainability and energy use, improvement results are mixed, depending on the task. For Clone Detection (where FLOPs did not show a statistical correlation with energy), the results show that the energy-optimized approach produces models with higher GigaFLOPs (21%) than the FLOPs-optimized approach. This difference is statistically significant (p -value < 0.01) with a medium effect size ($\hat{A}_{12} = 0.68$). Nevertheless, we observe that this increase in FLOPs does not translate to higher energy consumption, and the energy optimization approach outperforms the original approach with a difference of 39% in median inference energy. This is a statistically significant difference (p -value < 0.01) with a large effect size ($\hat{A}_{12} = 0.925$).

For Vulnerability Prediction – where FLOPs correlated with energy – the energy-based approach yields models with 36% more GFLOPs and about 30% more energy, though the differences are not statistically significant (p -value >0.2).

These results suggest that the energy optimization approach can be useful when FLOPs are unreliable or lack a statistical correlation with energy consumption, as is the case for Clone Detection. For Vulnerability Prediction, the energy approach neither improves nor shows a statistically significant degradation.

Answer to RQ₂. Using surrogate energy instead of FLOPs reduces median inference energy in Clone Detection by 39% (from 5355.83 J to 3604.50 J), and this gain is statistically significant ($p < 0.01$, $\hat{A}_{12} = 0.925$). In Vulnerability Prediction, where FLOPs already correlate with energy, the energy-based optimization changes median energy by +30% and GFLOPs by +36%, but these differences are not statistically significant ($p > 0.2$), and accuracy remains non-significantly different as well ($p > 0.3$).

6.3 RQ3: Application to generation tasks



■ **Figure 2** ROUGE-L of the Pareto front compared to inference energy and model size.

Figure 2 shows the ROUGE-L scores of different generative code summarization models on the Pareto front, along with the other two optimization objectives: energy (Figure 2a) and size (Figure 2b). The red line indicates a threshold for the ROUGE-L score of 0.18. By manually inspecting individual amples and generating summaries, and observing that summaries with approximately 0.15 still fit well to the source code, despite the summary being worded differently and being much shorter. To be conservative with our choice, we selected 0.18 as an acceptable lower bound for our use case. We observe that, for Code Summarization, the Pareto front shows more constraints and much higher trade-offs between the objectives. One of the main observations we can make is that, unlike classification tasks, the 3MB selection rule from the previous study [29] cannot be applied, since models below 10MB achieve ROUGE-L scores below 0.1, which has a significant impact on performance.

■ **Table 3** ROUGE-L, Size and Inference Energy for the teacher model and the most accurate student model. The best result for each metric is highlighted in grey.

Model	ROUGE-L	Size (MB)	Energy (J)
Teacher	0.229	830	82 396
Most Accurate	0.199 (-13%)	114 (-86%)	13 600 (-83%)
Most Efficient	0.194 (-15%)	157 (-81%)	8 059 (-90%)

Table 3 shows the objectives for the teacher model, which obtains an ROUGE-L score of 0.231. Compared with the most accurate student model, the score is 0.199 (a 13% decrease). Therefore, even in the best case, we observe a slight decrease in accuracy. However, by manually inspecting the output of different models, we see that summaries are mostly coherent. Student models struggle more with code that uses specialized nomenclature, such as specific libraries or services. Despite a small decrease in accuracy, this student model achieves an 83% reduction in energy usage while processing the same number of inference samples and a 86% reduction in memory footprint.

Other student models can also achieve a similar level of accuracy to the most accurate student while achieving higher energy savings. To illustrate this, Table 3 also showcases the results for the most efficient model obtained under the constraint that the ROUGE-L score exceeds 0.18. We defined this threshold based on the performance we observed during

manual inspection, since models with this score still perform well with simpler pieces of code. Then, we selected the most efficient model by dividing the ROUGE-L score by the total inference energy, obtaining an Accuracy per unit of Energy metric. We selected the model with the highest value in this metric to prioritize both accuracy and efficiency: choose a model that provides greater accuracy with less energy usage. This kind of selection criterion can be useful for systems with high energy constraints [7, 30]. For this model, we observe an even higher energy reduction, up to 90% compared to the teacher model and 40% less than the most accurate student, albeit with a slightly higher memory footprint.

Answer to RQ₃. For code summarization, many-objective distillation reduces inference energy from 82,396 J to 8,059 J (most efficient student), a 90% reduction, and reduces model size from 830 MB to 114 MB for the most accurate student, an 86% reduction. The performance trade-off is moderate: ROUGE-L drops from 0.229 to 0.199 (−13%) for the most accurate student and to 0.194 (−15%) for the most efficient student.

7 Discussion

7.1 FLOPs as a proxy for energy

Our experiments for RQ₁ show that FLOPs are not a consistent proxy for energy consumption: there is no correlation between FLOPs and inference energy for the Clone Detection task, but a positive correlation for the Vulnerability Prediction task. Furthermore, even for the Vulnerability Prediction task, where statistical correlation is present, we still cannot detect a direct relationship between FLOPs and energy consumption. In this task, we observe some models with similar FLOPs that use vastly different amounts of energy, and models with similar energy that differ widely in FLOPs. By using FLOPs alone to select an energy-efficient model, an AI engineer would miss models that stand out from the correlation and show a reduced energy consumption despite the elevated number of FLOPs.

These results suggest that FLOPs, as a metric, do not capture all the complexities and components involved in a model's total energy consumption. The approach used to compute FLOPs for a model [4] treats FLOPs as mathematical operations rather than machine instructions, treating operations such as multiplications and additions as equally costly. A similar thing happens with profilers from libraries, including PyTorch, which use formulas to estimate mathematical operations and, in some cases, ignore operations such as additions.

This computation also uses only a subset of the model's hyperparameters (Hidden Size, Number of Layers, Sequence Length, Vocabulary Size, Intermediate Size, and Number of Heads). Other factors, such as additional model hyperparameters, dataset characteristics, or inference steps, may influence energy consumption across the whole task. For example, from the ANOVA test, we observe that the Hidden Activation function also has a statistical correlation with energy. Each possible activation function has a different implementation, which can affect FLOPs and energy in different ways. These aspects make FLOPs an unreliable proxy, since the computation understates some aspects and fails to account for others that can affect energy consumption.

The FLOPs computation is also blind to software or hardware optimizations introduced by the libraries or the GPU during execution time. Underlying libraries provided by the operating system and hardware drivers, along with their versions, can affect the energy consumption of any program [31]. An example of this is the omission of operations due to sparsity, where GPUs introduce optimizations to avoid multiplying by 0. Something similar can happen with respect to sequence padding. In this methodology, FLOPs are computed

using a maximum sequence length. However, not all inputs to the model use all this sequence length, with shorter inputs requiring fewer FLOPs. Additionally, when processing a batch through an LLM model with a defined batch size, all inputs get padded to the length of the longest input. The largest sequence sets the length for the entire batch. Other sequences are padded with padding tokens, which are ignored during computation using an attention mask. The FLOPs costs for these padding tokens cannot be assumed to be the same as for normal tokens, since the GPU can introduce optimizations to avoid computations with these tokens. Therefore, the FLOPs count provided by this methodology is an upper bound rather than an exact count of computations per input.

These optimizations might explain the lack of correlation in the Clone Detection task. To perform this task, the maximum sequence length is used when tokenizing each code block. To compare two blocks of code, they are concatenated, giving an effective maximum length of twice the input to the tokenizer. While this is considered for FLOP computation, it also means the average padding length doubles, resulting in the Clone Detection task having more “empty” FLOPs.

In conclusion, FLOPs metrics, as used in this approach and in most of the literature, serve as an upper bound that does not account for all characteristics of the model and task. For simpler tasks or datasets, such as Clone Detection, the actual computation can fall far short of this upper bound, making FLOPs a poor proxy for energy consumption. In comparison, more complex tasks, such as Vulnerability Prediction, are closer to the upper bound on FLOPs and exhibit higher correlation.

A more accurate proxy for energy consumption than FLOPs is inference time, which shows a strong correlation with energy consumption, according to previous studies [2]. Compared to energy, this is a much more accessible metric, since it does not require the use of profilers or energy measurement guidelines [6]. However, to compare with FLOPs, the model still needs to be run. Running and measuring energy for a model can be extremely time-consuming, especially in many-objective optimization problems, where multiple configurations must be evaluated. Using inference time is a reasonable tradeoff, since it still requires running the model, but does not require additional setup for energy measurements. Additionally, inference time provides no insights into the model’s energy impact.

From the results for $\mathbf{RQ}_2$, we learn that, in cases where FLOPs and energy present a correlation (Vulnerability Prediction), using surrogate models instead of FLOPs does not have a statistically significant effect on the final result, for better or worse. However, when FLOPs cannot accurately reflect energy consumption, the surrogate models bridge this gap and produce more efficient models without impact on accuracy. These results show that energy surrogate models are preferable to FLOPs, as they can account for more factors that affect sustainability and perform well regardless of whether a correlation is present.

To build the energy models, we need to measure the inference energy consumption of all the sampled models in the surrogate step of MORPH. This adds an additional step to the sampling, training, and evaluating pipeline from the original methodology. Energy measurements are a costly step, especially in larger student models that tackle more complex tasks, as in the case of CodeT5+ and Code Summarization. Therefore, it is preferable to use FLOPs when we are certain of the correlation, such as in the Vulnerability Prediction task, since these additional costs do not yield any benefits.

In any case, the trade-offs between FLOPs and energy only apply during the optimization phase of this methodology. Once MORPH provides the Pareto front of the optimal models, we observe that, statistical correlations aside, there is no direct relationship between FLOPs and

energy. This is an important consideration when selecting a final student model that meets our constraints. If we wish to select a model with the lowest possible energy consumption, selecting the one with the lowest FLOPs is not guaranteed to yield the lowest energy consumption. FLOPs might be useful for solving the optimization problem, as they are easy to compute (and may correlate with energy). Nevertheless, we propose that developers prioritize actual energy measurements over FLOPs when selecting a final student model, particularly for heavily constrained systems. This recommendation stems from our findings, which indicate that FLOPs are not a reliable proxy for energy consumption in this context, and that related literature should be cautious in its use.

7.2 Many-Objective Optimization for Generation Tasks

Our experiments for **RQ₃** show that Many-Objective distillation can be used for generative SE tasks like Code Summarization, albeit with some caveats. This technique produces student models with a significant reduction in size (up to 86% of the teacher model) and energy consumption during inference (up to 90%). However, none of the student models match the teacher model’s accuracy, resulting in a slight decrease in ROUGE-L score. This drop is similar to results from the literature on other distillation approaches for summarization [37, 38].

Our study also shows how sensitive state-of-the-art models are when applied to datasets other than their original evaluation dataset. To mitigate contamination, we are using The Heap [19] as our evaluation dataset. This dataset follows more strict deduplication techniques than other publicly available datasets to avoid contamination. This affects final ROUGE-L scores for both teachers and students, with both obtaining scores slightly lower than those of other state-of-the-art summarization models. For example, our fine-tuned teacher model has an average ROUGE-L of 0.22 in The Heap. Comparatively, when evaluating the teacher model on CodeXGLUE as in the original CodeT5+ paper [43], we obtain an average ROUGE-L of 0.33, closer to the original results. This indicates that the model is not generalizing well for a different evaluation dataset. This calls for more research on how these models should be evaluated and highlights the need for metrics that better reflect the actual correctness of generative model outputs for software engineering problems, as well as for improved generalization beyond predefined benchmarks.

In our experiments, the student model’s savings come at the cost of a drop in performance. While the observed drop is moderate, it will depend on the use case whether it is acceptable. For example, local models running on a laptop and used for commit message generation can have a strict energy cap, and the accuracy drop is acceptable for such low-critical tasks. On the other hand, a model intended to summarize pull request contributions requires an accurate summary to guide the reviewer who will eventually accept or reject the PR. What is an acceptable drop in accuracy and energy savings will depend on the specific use case, and is hard to define in a research setting.

The Many-objective approach also provides greater flexibility in the student architecture, thanks to weight subcloning. Other approaches in the literature need a student model that has been pretrained and has some general knowledge first [37, 38], or distill into pretrained checkpoints of smaller models. In this case, the available configurations are only those that are pretrained and open source. In comparison, our approach copies the most important weights from the teacher model using weight subcloning. This distillation technique can provide a wider range of hyperparameters while avoiding the expense of pretraining.

8 Threats to Validity

Our approach has some limitations regarding its validity. First is the distillation loss used for the Code Summarization task. In this paper, we used a standard distillation loss function based on the KL-divergence and the cross-entropy function. Model distillation is a rapidly developing subfield of AI. Newer methods are being proposed and bring improved and more complex losses [46], or different distillation paradigms [41]. However, the loss we applied for this paper is widely adopted in state-of-the-art solutions. Moreover, the focus of this paper is the application of Many-Objective Problem optimization towards exploration of the configuration space. We decided to use a standard, well-founded distillation loss and reduce the influence of other variables. Combining newer methods with Many-Objective solvers might yield improved distillation results.

Additionally, for the generation task, we used the ROUGE-L score as the model’s objective, which measures n-gram overlap. This metric is widely adopted in state-of-the-art evaluation, but it comes with its own limitations. ROUGE-L overlooks semantic similarity: there can be different valid summaries paraphrased in different ways. Evaluating semantic similarity would require human judgment or that of a reliable LLM, which incurs significant time and financial costs due to the need for an additional LLM or human participants and does not guarantee the validity of the judgment, especially in the case of LLM-as-a-judge. Given the amount of sampling necessary for the method applied in this paper, this kind of metric would be too costly and not entirely objective for the purpose of Many-Objective optimization. We selected ROUGE-L because, despite its shortcomings, it is widely used in the literature and aligns more closely with human judgment, according to the latest studies [11].

9 Conclusion

This study investigates the validity of FLOPs as an energy proxy for many-objective LLM knowledge distillation. Our findings reveal that the correlation between FLOPs and inference energy of the student model is often unreliable. For Clone Detection, FLOPs do not present a statistical correlation with energy, while Vulnerability Prediction does. Besides this correlation, FLOPs does not show a direct relationship with energy. Selecting the model with lower FLOPs does not always imply selecting the model with lower energy usage.

To address this, we proposed using surrogate energy models to estimate energy consumption for a given hyperparameter set before training. This approach converges to models that are 39% more energy-efficient for Clone Detection tasks, bridging the gap in the correlation between FLOPs and energy efficiency.

We also applied this approach to Code Summarization in CodeT5+, testing its potential for generation tasks. The results showed significant compression levels for students (up to 86% smaller) and reduced energy usage (up to 90%), with a modest impact on accuracy.

Future work will expand the generalizability of these findings by applying them to tasks such as code generation. We also intend to analyze the pitfalls of FLOPs and define metrics that correlate better with energy while requiring less effort than direct energy measurement.

Data Availability Statement

A replication package is available in Zenodo². This package contains the scripts to distill surrogate and final models, as well as a script to measure the energy consumption of the models. It also contains the data collected from our hardware setup, as well as the analysis scripts used to generate the plots and tables in this paper.

References

- 1 Wasi Uddin Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. A transformer-based approach for source code summarization. In *58th Annual Meeting of the Association for Computational Linguistics, ACL*, 2020. doi:10.18653/V1/2020.ACL-MAIN.449.
- 2 Andrea Asperti, Davide Evangelista, and Moreno Marzolla. Dissecting flops along input dimensions for greenai cost estimations. In *Machine Learning, Optimization, and Data Science - 7th International Conference, LOD, Grasmere*, 2021. doi:10.1007/978-3-030-95470-3_7.
- 3 Pengguang Chen, Shu Liu, Hengshuang Zhao, and Jiaya Jia. Distilling knowledge via knowledge review. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021*, 2021. doi:10.1109/CVPR46437.2021.00497.
- 4 Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. ELECTRA: pre-training text encoders as discriminators rather than generators. *CoRR*, 2020. arXiv:2003.10555.
- 5 William Jay Conover. *Practical nonparametric statistics*. Wiley series in probability and statistics. Wiley, New York, NY [u.a.], 3. ed edition, 1999.
- 6 Luís Cruz. Green software engineering done right: a scientific guide to set up energy efficiency experiments, 2021. URL: <http://luiscruz.github.io/2021/10/10/scientific-guide.html>.
- 7 Eduardo Cueto-Mendoza and John Kelleher. A framework for measuring the training efficiency of a neural architecture. *Artificial Intelligence Review*, 57(12):349, 2024. doi:10.1007/s10462-024-10943-8.
- 8 DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. doi:10.48550/arXiv.2501.12948.
- 9 Santiago del Rey, Silverio Martínez-Fernández, Luís Cruz, and Xavier Franch. Do dl models and training environments have an impact on energy consumption? In *49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, September 2023. doi:10.1109/SEAA60479.2023.00031.
- 10 Radosvet Desislavov, Fernando Martínez-Plumed, and José Hernández-Orallo. Trends in ai inference energy consumption: Beyond the performance-vs-parameter laws of deep learning. *Sustainable Computing: Informatics and Systems*, 2023. doi:10.1016/j.suscom.2023.100857.
- 11 Mikhail Evtikhiev, Egor Bogomolov, Yaroslav Sokolov, and Timofey Bryksin. Out of the bleu: How should we assess quality of the code generation models? *Journal of Systems and Software*, 203:111741, 2023. doi:10.1016/j.jss.2023.111741.
- 12 Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, et al. Large language models for software engineering: Survey and open problems. In *2023 International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*. IEEE, 2023. doi:10.1109/ICSE-FoSE59343.2023.00008.
- 13 Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, et al. Codebert: A pre-trained model for programming and natural languages. *CoRR*, abs/2002.08155, 2020. arXiv:2002.08155.
- 14 Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, et al. Graphcodebert: Pre-training code representations with data flow. *CoRR*, abs/2009.08366, 2020. arXiv:2009.08366.

² <https://doi.org/10.5281/zenodo.20270810>

- 15 Daya Guo, Canwen Xu, Nan Duan, Jian Yin, et al. Longcoder: a long-range pre-trained language model for code completion. In *40th International Conference on Machine Learning, ICML'23*, 2023.
- 16 Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. [arXiv:1503.02531](#).
- 17 International Energy Agency. Energy and ai. Special report, International Energy Agency (IEA), Paris, France, April 2025. Published April 2025. CC BY 4.0 licence. URL: <https://www.iea.org/reports/energy-and-ai>.
- 18 Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, et al. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 4163–4174, November 2020. doi:10.18653/v1/2020.findings-emnlp.372.
- 19 Jonathan Katzy, Razvan Mihai Popescu, Arie van Deursen, and Maliheh Izadi. The heap: A contamination-free multilingual code dataset for evaluating large language models. In *IEEE/ACM Second International Conference on AI Foundation Models and Software Engineering, Forge@ICSE 2025, Ottawa*, 2025. doi:10.1109/Forge66646.2025.00025.
- 20 Mohamad Khajezade, Jie Jw Wu, Fatemeh Hendijani Fard, Gema Rodríguez-Pérez, et al. Investigating the efficacy of large language models for code clone detection. In *32nd International Conference on Program Comprehension (ICPC)*, 2024. doi:10.1145/3643916.364503.
- 21 Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *CoRR*, abs/1910.09700, 2019. [arXiv:1910.09700](#).
- 22 Xiaonan Li, Yunfan Shao, Tianxiang Sun, Hang Yan, et al. Accelerating BERT inference for sequence labeling via early-exit. In *59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, August 2021. doi:10.18653/v1/2021.acl-long.16.
- 23 Fang Liu, Ge Li, Yunfei Zhao, and Zhi Jin. Multi-task learning based pre-trained language model for code completion. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering, ASE '20*, pages 473–485, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3324884.3416591.
- 24 Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, et al. FastBERT: a self-distilling BERT with adaptive inference time. In *58th Annual Meeting of the Association for Computational Linguistics*, July 2020. doi:10.18653/v1/2020.acl-main.537.
- 25 Anton Lozhkov, Raymond Li, Loubna Ben Allal, et al. Starcoder 2 and the stack v2: The next generation, 2024. [arXiv:2402.19173](#).
- 26 Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, et al. Codexglue: A machine learning benchmark dataset for code understanding and generation. *CoRR*, 2021. [arXiv:2102.04664](#).
- 27 Sasha Luccioni, Yacine Jernite, and Emma Strubell. Power hungry processing: Watts driving the cost of ai deployment? In *Conference on Fairness, Accountability, and Transparency, FAccT '24*, 2024. doi:10.1145/3630106.3658542.
- 28 Annibale Panichella. An adaptive evolutionary algorithm based on non-euclidean geometry for many-objective optimization. In *Genetic and Evolutionary Computation Conference, GECCO '19*, 2019. doi:10.1145/3321707.3321839.
- 29 Annibale Panichella. Metamorphic-based many-objective distillation of llms for code-related tasks. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*, April 2025. doi:10.1109/ICSE55347.2025.00230.
- 30 Enrique Barba Roque and Luis Cruz. Energy aware development of neuromorphic implantables: From metrics to action. In *11th International Conference on ICT for Sustainability (ICT4S)*, 2025. doi:10.1109/ICT4S68164.2025.00028.
- 31 Enrique Barba Roque, Luis Cruz, and Thomas Durieux. Unveiling the energy vampires: A methodology for debugging software energy consumption. In *47th IEEE/ACM International Conference on Software Engineering, ICSE, Ottawa*, 2025. doi:10.1109/ICSE55347.2025.00118.

- 32 June Sallou, Luís Cruz, and Thomas Durieux. Energibridge: Empowering software sustainability through cross-platform energy measurement, 2023. [arXiv:2312.13897](#).
- 33 Mohammad Samragh, Mehrdad Farajtabar, Sachin Mehta, Raviteja Vemulapalli, et al. Weight subcloning: direct initialization of transformers using larger pretrained ones. *CoRR*, 2023. doi:10.48550/arXiv.2312.09299.
- 34 Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai. *Commun. ACM*, 63(12):54–63, November 2020. doi:10.1145/3381831.
- 35 Jieke Shi, Zhou Yang, Hong Jin Kang, Bowen Xu, et al. Greening large language models of code. In *46th International Conference on Software Engineering: Software Engineering in Society, ICSE-SEIS’24*, 2024. doi:10.1145/3639475.3640097.
- 36 Jieke Shi, Zhou Yang, Bowen Xu, Hong Jin Kang, et al. Compressing pre-trained models of code into 3 mb. In *37th IEEE/ACM International Conference on Automated Software Engineering, ASE ’22*, 2023. doi:10.1145/3551349.3556964.
- 37 Sam Shleifer and Alexander M. Rush. Pre-trained summarization distillation. *CoRR*, abs/2010.13002, 2020. [arXiv:2010.13002](#).
- 38 Chia-Yi Su and Collin McMillan. Distilled GPT for source code summarization. *Autom. Softw. Eng.*, 31(1):22, 2024. doi:10.1007/S10515-024-00421-4.
- 39 Jeffrey Svajlenko, Judith F. Islam, Iman Keivanloo, Chanchal K. Roy, et al. Towards a big data curated benchmark of inter-project code clones. In *2014 IEEE International Conference on Software Maintenance and Evolution*, 2014. doi:10.1109/ICSME.2014.77.
- 40 Alexey Svyatkovskiy, Sebastian Lee, Anna Hadjitofi, Maik Riechert, et al. Fast and memory-efficient neural code completion. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, 2021. doi:10.1109/MSR52588.2021.00045.
- 41 Yijun Tian, Yikun Han, Xiusi Chen, Wei Wang, and Nitesh V. Chawla. Beyond answers: Transferring reasoning capabilities to smaller llms using multi-teacher knowledge distillation. In *18th ACM International Conference on Web Search and Data Mining, WSDM ’25*, 2025. doi:10.1145/3701551.3703577.
- 42 András Vargha and Harold D. Delaney. A critique and improvement of the cl common language effect size statistics of mcgraw and wong. *Journal of Educational and Behavioral Statistics*, 2000. doi:10.3102/10769986025002101.
- 43 Yue Wang, Hung Le, Akhilesh Gotmare, Nghi D. Q. Bui, et al. Codet5+: Open code large language models for code understanding and generation. In *Conference on Empirical Methods in Natural Language Processing*, 2023. doi:10.18653/V1/2023.EMNLP-MAIN.68.
- 44 Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, et al. Sustainable AI: environmental implications, challenges and opportunities. In *5th Conference on Machine Learning and Systems, MLSys 2022, Santa Clara*, 2022.
- 45 Taiqiang Wu, Chaofan Tao, Jiahao Wang, Runming Yang, et al. Rethinking kullback-leibler divergence in knowledge distillation for large language models. In *31st International Conference on Computational Linguistics, COLING, Abu Dhabi*, 2025.
- 46 Runming Yang, Taiqiang Wu, Jiahao Wang, Pengfei Hu, Ngai Wong, and Yujiu Yang. Llm-neo: Parameter efficient knowledge distillation for large language models. *CoRR*, abs/2411.06839, 2024. doi:10.48550/arXiv.2411.06839.
- 47 Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. Decoupled Knowledge Distillation . In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022. doi:10.1109/CVPR52688.2022.01165.
- 48 Xin Zhou, Ting Zhang, and David Lo. Large language model for vulnerability detection: Emerging results and future directions. In *44th International Conference on Software Engineering: New Ideas and Emerging Results, ICSE-NIER’24*, 2024. doi:10.1145/3639476.3639762.
- 49 Yaqin Zhou, Shangqing Liu, Jingkai Siow, Xiaoning Du, et al. Devign: effective vulnerability identification by learning comprehensive program semantics via graph neural networks. In *33rd International Conference on Neural Information Processing Systems*, 2019. URL: <https://dl.acm.org/doi/10.5555/3454287.3455202>.