Improving Carbon Emissions of Federated Large Language Model Inference through Classification of Task-Specificity

Geerd-Dietger Hoffmann
Green Coding Solutions GmbH
Berlin, Germany
didi@green-coding.io

Verena Majuntke
HTW Berlin
Berlin, Germany
verena.majuntke@htw-berlin.de

ABSTRACT

The resource consumption of software and communication infrastructure is an increasing concern, particularly with the emergence of Large Language Models (LLMs). While energy consumption and carbon emission of LLMs during training have been a focus in research, the impact of LLM inference which scales with the number of requests, remains underexplored. In this paper we show that energy efficiency and carbon emission of LLM inference vary depending on the model and on the task category, e.g. math, programming, general knowledge, with which the model is prompted and that smaller specialized models can achieve comparable accuracy while using less resources. We analyze the differences across 8 open-source LLMs when processing prompts of different task categories. Our findings lead to a novel approach: Classifying prompts by using embeddings to route them to the most energy-efficient and least carbon-intensive LLM in a federation of LLMs while keeping accuracy high. We validate the effectiveness of our method through empirical measurements.

CCS CONCEPTS

• **Computing methodologies → Artificial intelligence**; • **Software and its engineering → Extra-functional properties**; • **Computer systems organization → Distributed architectures**.

KEYWORDS

AI, inference, energy consumption, carbon emission, task-specificity

ACM Reference Format:
Geerd-Dietger Hoffmann and Verena Majuntke. 2024. Improving Carbon Emissions of Federated Large Language Model Inference through Classification of Task-Specificity. In *Proceedings of 3rd Workshop on Sustainable Computer Systems (HotCarbon'24).* ACM, New York, NY, USA, 7 pages.

1 INTRODUCTION

Large Language Models (LLM) have recently gained significant attention across industry, research, and civil society sectors. Among the numerous applications, ChatGPT stands out as a particularly prominent example. As of April 2024, ChatGPT reported more than 180 million active users, and its website received more than 1.8 billion visits in March 2024 [11]. Despite the rapid advancement and widespread adoption of artificial intelligence (AI) technologies, there has been little focus on addressing the energy demand and environmental impact of AI. The energy consumption and consequent environmental impact of AI occur across various phases of its lifecycle, *training*, *re-training* and *inference*. A strong research focus has been on the training phase of AI, as the training has proven to be very energy and resource consuming. For instance, the training of GPT-3 - a model with 175B parameters, required 1.287 megawatt-hours (MWh) of energy. Considering the carbon intensity of the electricity grid at the time - 429 grams of CO2 per kilowatt-hour - the training of GPT-3 alone resulted in the emission of approximately 502 tonnes of CO2 [30]. Furthermore, the same authors calculated the energy consumption of Machine Learning at Google for one week of April in 2019, 2020 and 2021. They found out that each year the proportion of Machine Learning had a portion of 10-15% of Google's overall energy consumption with three-fifth being used for inference [29]. However, the cost of LLM inference scales with the number of requests a model has to process. Considering the rapid advancement of AI technology, inference plays a critical role in the total energy usage and carbon emissions of generative artificial intelligence.

This paper focuses on reducing the carbon emission for inference of a set of LLMs. We present an approach that uses machine learning embedding models to classify a prompt by its task category. The prompt is then routed to the most energy-efficient and least carbon-intensive model within a federation of models. The approach is based on the finding that generalized models are more computational intensive and energy-demanding than models that have been trained for a specific task [22].

The contributions of this paper are threefold: (1) A benchmark of the energy consumption and carbon emission during inference of 8 open-source LLMs for different task categories, identifying the most suitable model for each prompt. (2) A system that classifies task categories using embeddings to route prompts to the optimal model for each prompt, based on our benchmarks. (3) Empirical measurements demonstrating that our approach reduces the energy consumption and carbon emission of inference across the federation of LLMs. The results support the use of energy efficient and carbon reduced task-specific models and a routing component as a future direction for sustainable AI. It paves the way to a multi model world in which carbon can be saved through using the optimal model for each prompt.

2 RELATED WORK

Prior research focuses on the energy efficiency and carbon footprint during the training of neural networks [36], [37], [1], and [15]. Strubell et al. [36] show that an increase in model accuracy implies a substantial additional amount of computational resources, escalating financial costs as well as environmental impact. Similarly, Thompson et al. [37] observe that advancements in deep learning models correlate strongly with increases in computing power, suggesting that such progress may become environmentally and economically unsustainable. They also note that energy limitations inherently set an upper bound on the accuracy that can be achieved. Lasse et al. [1] introduce *Carbontracker*, a tool designed to track and predict the carbon footprint during the training phase of deep learning models. The tool aims to foster the development of energy-efficient neural networks. Henderson et al. [15] suggest a framework for real-time tracking of energy consumption and carbon emissions in reinforcement learning algorithms. Based on case studies using the framework, strategies for energy and carbon emission reduction are proposed. In contrast to our approach, the discussed research focuses mainly on the training phase.

Broader approaches have been presented in [20], [6], [12], and [14]. Lenherr et al. [20] show that the focus on enhancing deep learning accuracy comes with unconsidered economic and environmental cost. The work introduces recognition and training efficiency as new metrics aiming at balancing accuracy, complexity and energy consumption of a model. Schwartz et al. [34] argue for resource efficiency as a critical evaluation criterion alongside accuracy. They propose incorporating a "price tag" on the development life cycle of deep learning models to establish baselines for further improvements. Bomasani et al. [6] assess the societal impacts of foundation models, including economic and environmental effects. Another comprehensive approach has been taken by Eilam et al. [12]. In their work they introduce unified AI sustainability metrics as an extension to data center sustainability metrics which have been introduced by Gandhi et al. [14]. They apply their approach to artificial intelligence covering the whole life cycle allowing for a holistic assessment of the sustainability of a model.

Related work in the inference phase has been conducted in [7], [26], [33], [22], [12], [33], and [8]. Canziani et al. [7] identify a gap in the consideration of resource utilization. They analyze fourteen deep neural networks (DNNs) from the ImageNet challenge, focusing on practical metrics such as inference time and power consumption. Their findings reveal a hyperbolic relationship between accuracy and inference time, suggesting that gains in accuracy require disproportionately more computation time. In [33] Samsi et al. analyze the inference performance and inference computational cost of the LLM LLaMA using NVIDIA V100 & A100 as recent GPUs and Alpaca and GSM8K as datasets for their benchmarks. Furthermore, Molom-Ochir and Shenoy [26] evaluate the performance and energy efficiency of popular GPUs used in both embedded and desktop environments. They discover that larger devices do not necessarily equate to greater energy efficiency. Based on their findings, they propose a recommendation algorithm to aid system designers in selecting the most appropriate hardware for specific needs. Luccioni et al. [22] also focus on the inference phase. They show that inference in more generalized models requires order of magnitudes more energy and thus emissions than inference in models which have been tailored for specific tasks even when the number of model parameters is controlled. In [8], the authors examine the carbon emissions and energy use of generative AI inference, using ChatGPT as use case. They develop a workload model for geographic shifting and evaluate different routing strategies like *CarbonMin* directing requests to low-carbon energy models. Their approach reduces emissions while maintaining user experience. The presented approach is similar to our approach as it routes requests to models which run on low-carbon energy. In contrast, to [8] however, we classify requests into task categories and route them to models which have proven to be energy efficient which also implicitly results in less carbon emissions when processing the task category. A combination of both approaches for even more carbon emission optimization should be analyzed.

3 BACKGROUND

In this section we provide the theoretical information for our contribution. We first introduce *LLMs* and *TSLLMs* before we address the topic of energy consumption and carbon emission.

3.1 Large Language Models

A *language model* is a statistical probability distribution over sequences of words in a natural language [35]. These models are essential for natural language tasks such as text understanding and generation complying to linguistic rules. The advent of transformer-based architectures [38] has significantly enhanced the capabilities of language models, leading to large language models (LLMs). A *LLM* is a large-scale, pre-trained statistical language model based on neural networks. LLMs have evolved to perform complex reasoning and mimic human intelligence, establishing them as foundational components for general-purpose AI [25]. A LLM typically consists of hundreds of millions to billions of parameters and is trained on extremely big datasets. The data usually covers various content in order to equip the LLM with some level of generality and versatility. Additionally, an LLM can be tailored for specific tasks through fine-tuning on task-specific training data. In the following we refer to such a LLM as *task-specific LLM (TSLLM).*

To make use of the capabilities of a LLM, a textual input is provided guiding the ouput of the model. The textual input is referred to as a *prompt.* A prompt requires some sort of instructions or questions directed at the model. Moreover, a prompt may comprise more complex structures like a specification in which way the model should structure the output. Carefully crafted prompts enable users to fine-tune the output of a model without adapting the model itself. Engineering a "good" prompt has shown to be a key factor of using the full potential of a LLM [23]. In [32], the authors identified 29 different prompting techniques as of February 2024. One of the earliest forms of prompting techniques is the so called zero-shot prompting [31]. A *zero-shot prompt* is a prompt that causes a model to generate a response without having been specifically trained on the task. The prompt is designed without any examples and the model composes the response based on content and patterns it has learned.

3.2 Energy Consumption and Carbon Emission

Software systems cause emissions when they are executed on respective hardware. The emissions stem from the energy that is used by the hardware (*direct emissions*) and the manufacturing and disposal of the hardware (*embodied emissions*). For the measurements in this paper we use the ISO standard Software Carbon Intensity

(SCI) [18] developed by the Green Software Foundation[1]. In order to calculate and report an SCI score, the specification requires to define 1) The *bounds* of the software system to be measured, 2) the *scale* in terms of a functional unit over which the system scales, and 3) a *definition* of the quantification method.
The SCI score is a rate, carbon emissions per unit R which is calcuated according to the following formula:

$$SCI = ((E * I) + (M))\ per\ R \tag{1}$$

where $E*I$ is the operational energy consumed by a software system consisting of E - the energy (kWh) consumed for a functional unit R , and I the region-specific carbon intensity ($CO2eq/kWh$). Furthermore, M denotes the embodied carbon.

4 APPROACH

As previously stated, the focus of our research is to improve the energy consumption and carbon emission of LLM inference. Our approach has been inspired by the finding that larger generalized models are more energy-demanding than smaller models that have been trained for a specific task [22]. Thus, we introduce a federation of task-specific LLMs (TSLLMs) and employ a classifier component which is placed upstream of the set of models. The classifier assesses each incoming prompt with respect to already known prompts, with their SCI score, to determine the TSLLM within the federation that has been identified to be optimal for energy and carbon efficiency for that specific task, i.e. has the lowest SCI-score. The prompt is then routed to the selected TSLLM. An overview of our approach is depicted in Figure 1. The figure shows the federation consisting of a *Classifier* and a set of TSLLMs ($\{TSLLM_1, TSLLM_2, ..., TSLLM_n\}$). The *Classifier* serves as an interface for using entities. It classifies the task category of a prompt, forwards it to a selected model and returns the response of the model to the using entity. If a new model is added to the federation, the model is evaluated and its information is added to the *Classifier*. Analogously, the *Classifier* deletes the information, if a model is removed from the federation.

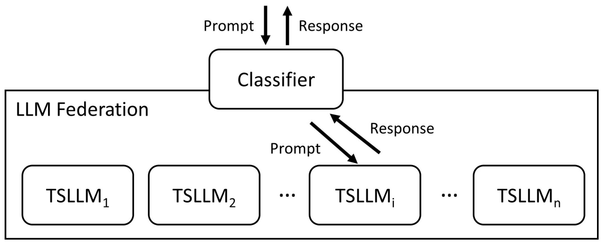

Figure 1: Using a *Classifier* with TSLLM federation

4.1 Model and Data Set Selection

To show the validity of our approach, we conducted a series of measurements described in Section 5. For our evaluation, we selected 8 pre-trained, open-source TSLLMs which are shown in Table 1. The table shows the names of the selected TSLLMs, the architecture it is based on, the number of parameters, the size of the model and its task specialization. We selected the models based on our current memory constraints.

Table 1: Open-source TSLLMs

Name	Arch	Param.	Size	Task spec.
codellama	llama	7B	3.8GB	code
gemma	gemma	9B	5.0GB	general
llama3	llama	8B	4.7GB	general
mistral	llama	7B	4.1GB	general
stablelm2	stablelm	2B	983MB	translation
sqlcoder	llama	7B	4.1GB	sql
tinyllama	llama	1B	638MB	general
wizard-math	llama	7B	4.1GB	math

As specific task categories we identified the following: (1) Python code generation, (2) mathematical understanding, (3) natural language question answering, (4) SQL generation, and (5) language translation. These 5 task categories were chosen because models specifically trained for these task categories were available, ensuring an elaborate and relevant experimental setup. For datasets with respective prompts, we chose *MBPP* [2] for Python code generation, *GSM8K* [10] for mathematical understanding, *WebQuestions* [4] for natural language question answering, *PetSQL*[21] for SQL generation , and *Test Sets* [3] for language translation. The datasets were selected from https://paperswithcode.com. Our selection criteria was (1) the number of papers referencing the dataset, (2) the task category, and (3) the format in which the data is provided.

4.2 Classification and Training

To implement the *Classifier*, we created an embedding. An AI *embedding* is a low-dimensional representation of discrete variables, commonly used in natural language processing (NLP) to capture semantic meanings. Embeddings have shown strong performance in text classification [19]. Foundational studies like [5] and [24] introduced Word2Vec and FastText, which reduce computational requirements while maintaining high accuracy in NLP tasks. For our approach we use the mxbai-embed-large embedding model [27] for embedding creation. All known prompts are precomputed and saved in the chroma [9] database. Chroma is an open source embedding database which implements fast retrieval of closest embeddings using Squared L2 $d = \sum (A_i - B_i)^2$. When a new prompt is encountered, an new embedding is created. Subsequently, the database is queried to get the closest known prompt that has been measured. The *Classifier* then checks the mapping for the TSLLM with the lowest SCI-score and passes the prompt to this TSLLM. In addition to SCI-scores, we manually checked for the correctness of answers and discarded every TSLLM that gave incorrect answers. To train the *Classifier*, the embedding model was fed with the first 50 prompts of each dataset, i.e. category. Moreover, each model was evaluated for those prompts. We then checked for correctness and discarded any incorrect results as we will discuss in Section 6.

5 MEASUREMENTS

We utilized a dedicated measurement machine equipped with an Intel(R) Core(TM) i5-9600K CPU @ 3.70GHz, 32GB of memory,

[1]https://greensoftware.foundation

Table 2: Evaluation model accuracy

	Code					Math					Natural Questions					SQL					Translation				
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
codellama	0.17	0.27	0.17	0.14			0.21							0.01		0.37	0.73	0.30	0.41	0.24	0.01	0.02	0.02	0.03	0.01
gemma	0.59	0.56	0.37	0.72	0.45		0.27				0.04		0.02	0.03	0.03		0.51	0.29	0.28	0.13	0.03	0.04	0.03	0.04	0.03
llama3	0.46	0.38	0.21	0.16	0.17	0.19			0.17		0.27	0.07	0.22	0.01	0.25	0.11	0.29	0.26	0.19	0.28	0.05	0.05	0.04	0.15	0.17
mistral	0.40	0.36	0.40	0.28	0.55	0.33		0.71	0.25		0.10	0.14	0.08	0.06		0.14	0.29	0.20	0.20	0.27	0.02	0.02	0.02	0.03	0.10
stablelm2	0.10	0.10	0.08	0.09	0.10							0.12	0.12	0.14		0.15	0.20	0.08	0.10	0.18	0.05	0.08	0.05	0.04	0.05
sqlcoder	0.09		0.04																	0.06					
tinyllama	0.08	0.03	0.04	0.03								0.02		0.00			0.06	0.05		0.05	0.01			0.01	
wizard	0.38	0.58	0.43	0.54	0.58		0.41	0.31	0.28	0.26	0.32	0.30	0.01	0.17	0.38	0.32	0.35	0.56	0.39	0.30	0.44	0.08	0.36	0.07	0.17

and an MSI GeForce GTX 1080 Armor 8G OC 8 GB. The machine ran a specialized version of Linux, NOP Linux [16] (a modified Ubuntu 22.04), optimized for energy measurement by removing all unnecessary cron jobs and services that could cause interrupts.

The operational software included only a daemon that queried a Redis queue for new jobs, loaded the specified model into memory, executed the query, and stored the results back into Redis. To collect metrics, we implemented a script that periodically read values and logged them to a file in *tmpfs*. We set the sampling period to 99 milliseconds to prevent lockstep sampling while minimizing data collection overhead. The interval was carefully chosen to ensure minimal interference from idle time, with the benchmarking overhead maintained at under 1% [17].

We employed a specialized device to measure the PSU energy consumption of the entire machine. Initially, we attempted to use a PowerSpy2 device manufactured by ALCIOM, however, its reliance on Bluetooth resulted in inaccurate timing and incorrect readings. Additionally, the device frequently crashed when using the logging feature through its official program. Consequently, we switched to using the MCP39F511N chip by Microchip, paired with the Power Monitor Demonstration Board (ADM00706), accessed via USB, to obtain accurate PSU readings. For CPU energy measurements, we utilized Intel RAPL, while GPU energy and temperature data was collected using NVIDIA SMI.

We used the Ollama [28] helper program for model invocation. To ensure reproducibility, we fixed the seed in the models. Our initial findings suggest that further research is required to explore how model parameters such as temperature, context window size, and tail-free sampling impact energy consumption. During measurements, we observed a progressive increase in energy usage per query, attributable to rising GPU temperatures which escalated energy demands. Consequently, we introduced a cooldown period whenever CPU or GPU temperatures exceeded 60 degrees to mitigate this effect.

To determine the SCI-scores across all combinations of models and datasets we define the respective TSLMM as *bounds* and the complete process of answering a prompt excluding the model preparation like loading into memory or building up caches as *scale*. We exclude model loading as this would normally already be in memory in a standard setup. We need to use the same machine with no other components interfering to get comparable readings. As quantification method we use lab-based measurements.
For measuring our approach, we define the *Classifier* plus network traffic plus the inference of the selected TSLMM as *bounds*. As *scale* we specify the complete process of classifying and routing a prompt and its response plus the inference of the selected TSLMM. Similarly, we used lab-based measurements as quantification method.

For both setups we set the location carbon intensity to a fixed value of 385 CO2eq/kWh. The value was retrieved from electricitymaps.com as the average value in Germany for 2022 [13] to get comparable results. Note that the fixed value does not reflect the exact intensity when the measurements were conducted.

We calculated the embodied carbon as described in [39] as public data is not provided by manufacturers. For the other hardware components, data from https://datavizta.boavizta.org/serversimpact was used and multiplied by 2 to account for the GPU creation. We assumed a constant of 574100 for the gCO2 emitted for production. While the value is not correct in absolute terms it has a stable bias. Thus, the impact of manufacturing and disposing the machine is evenly attributed to each model and expresses the embodied carbon in the SCI.

6 EVALUATION

As described in Section 4 we queried each model with 50 prompts per category to obtain SCI-scores for each prompt and each category. Initially we also accounted for total energy consumption of a given prompt. As models do not have a constant draw this resulted in values that do not represent the actual usage of models. Especially larger models showed extensive wait times in execution in which energy consumption would be very low.

Table 3: Average SCI per Category and Model for correct answers

	code	math	questions	sql	translation
codellama	0.1815	0.2260	0.0942	0.2757	0.0328
gemma	0.4726	0.2052	0.0560	0.2846	0.0519
llama3	0.2973	0.1491	0.1830	0.2839	0.0902
mistral	0.3156	0.2573	0.1371	0.2176	0.0420
stablelm2	0.0837	0.0545	0.1164	0.1615	0.0708
sqlcoder	0.1490	0.2469	0.2773	0.1015	0.1910
tinyllama	0.0445	0.0269	0.0163	0.0558	0.0093
wizard	0.5052	0.3218	0.3054	0.4107	0.1720

Because of this, we opted for the SCI approach as the inclusion of execution time adds utilization to the evaluation factor.

The results are shown in Table 3. The values indicate that there are great differences regarding SCI-scores for different categories. tinyllama, for instance, outperforms every other model with respect to its SCI-score. However, in some categories, the accuracy - which we discuss subsequently - of tinyllama is rather low. The results also reveal that models have different SCI-scores when processing prompts of certain task categories. For instance, translation tasks consume far less resources than coding tasks.

Table 4: Accuracy by Category and Model

	code	math	questions	sql	translation
codellama	94%	62%	62%	80%	84%
gemma	98%	72%	64%	72%	88%
llama3	100%	88%	84%	82%	96%
mistral	92%	80%	84%	86%	96%
stablelm2	96%	42%	60%	50%	72%
sqlcoder	44%	48%	24%	60%	20%
tinyllama	92%	56%	36%	28%	54%
wizard	94%	82%	74%	84%	92%

However, the results need to be seen in the context of accuracy. The accuracy of each model for each category is shown in Table 4. To assess, if the models returned the right answer, we manually labeled all 2000 responses as the responses were rather complex. The numbers show some positive and some negative highlights. For instance, llama3 has an accuracy of 100% for code-related prompts. In contrast, sqlcoder has a low accuracy answering prompts regarding natural questions while tinyllama has a low accuracy for sql-related prompts. And while codellama has initially been selected for code related questions, it is outperformed in this category by llama3. The results also suggest that larger, more general models return correct answers in different categories but that smaller models perform equally well in specific domains. tinyllama performs equally well to mistral in the coding category but uses an average SCI of 0.0445 per prompt in comparison to 0.3156.

Table 5: Sum SCI all prompts

	code	math	questions	sql	translation
codellama	9.4944	11.8264	4.4774	14.0644	1.6367
gemma	23.6249	10.3582	2.5469	13.6006	2.5755
llama3	14.8684	7.5321	8.9135	15.0997	4.4717
mistral	16.1504	12.8255	7.0139	12.0887	2.0640
stablelm2	4.2502	2.5880	5.1270	8.1730	3.7212
sqlcoder	5.3165	6.8574	9.3100	5.1250	5.6436
tinyllama	2.1936	1.1683	0.8012	2.6111	0.4889
wizard	25.6002	14.3143	12.9625	21.4763	8.4628

In addition to the 250 training and benchmarking prompts, we also measured 5 validation prompts per category. These were not included in the training set for the classification. The classifier determined the ideal model for the unseen prompt in that the answer is correct and has the lowest SCI. Table 2 shows the results of our measurements. If a model did not return a correct answer, the table shows a blank spot. The results indicate, that models show differences in accuracy depending on the category. tinyllama for example has no correct answer in the math category but performs very well in coding. Thus, it is important to select the correct model to get a correct answer. While the larger general models perform on a wider spectrum the smaller specialized models are very category-specific and thus require a prior classification of prompts.

Table 5 shows the SCI-score sum of inference costs for all 8 models and 5x50 prompts. The results also underpin the difference of the models when processing prompts of a certain category even if the answers are not correct. Selecting a model which is prompted with all categories is the current state of the art. In order to increase the accuracy, models become larger but at the price of energy and carbon emissions.

As the process involves running the embedding model we also measured this step. The embedding model is very energy efficient. Getting the embeddings for 250 prompts took 15 (2+ of 10 runs) seconds and the total training had an SCI of 0.7282 including the addition of all embeddings to the chroma database.

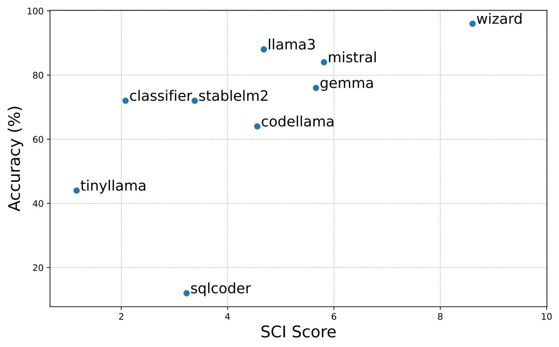

Figure 2: Model SCI-score sum and accuracy

Figure 2 comprises the measurements results for the single model approach and our federated approach with classifier. The x-axis represents the sum of SCI-scores of all 25 evaluation prompts including incorrect answers. The y-axis denotes the accuracy of each model for the 25 prompts. The values show that models like llama3 have a high accuracy of 88% but also a high SCI-score sum of 4.6814. In comparison, our classifier approach, denoted as *classifier* has a SCI-score sum of 2.0807 while accuracy is at 72%. This result is second best with respect to overall SCI-score sums and in the middle with respect to accuracy. However, we assume that the accuracy can be increased when the classifier is being trained on more data.

7 CONCLUSION AND FUTURE WORK

LLM inference significantly impacts the energy consumption and carbon emissions throughout the LLM lifecycle, as costs scale with the number of requests. In this paper, we presented a novel approach incorporating a classifier component upstream of a model federation, classifying prompts to route them to the models with the lowest carbon intensity (SCI) for the task category. While model training is very costly in terms of carbon emissions, inference also has a vast impact and needs optimization. We introduced a framework to measure model inference of known open source models showing that models perform differently when processing prompts from specific task categories. We demonstrated that our approach effectively reduces energy and thus carbon usage, using an embedding model to classify and route unseen prompts with minimal overhead. We found out that smaller models can adequately handle many task categories, and even when a prompt requires reprocessing, there are overall energy savings. For our research we limited the scope to 5 clearly defined task categories with corresponding models. However, the evaluation shows that it is possible to use the classifier component to choose the model which is best suited based on the prompt itself.

In our initial evaluations we utilized simple question-answer data. Currently, we are analyzing prompts that are more complex and lead to the generation of larger content. There is strong indication that models change in their resource usage when the context window becomes larger, which is the case in the chat like experience that is now present. Furthermore, we are researching how changing the model while in a chat can save resources overall. In this approach a more energy efficient model is initially chosen and if the answer is not sufficient the whole conversation is switched to a larger model. Moreover, we are analyzing how energy consumption and carbon emissions are influenced using confidence of models for prompt routing. Further work should also explore how modern chat interactions can be optimized for resource usage, to enhance the efficiency in practical applications and evaluate the approach in context of large dominant LLMs.

REFERENCES

[1] Lasse F. Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. 2020. Carbontracker: Tracking and Predicting the Carbon Footprint of Training Deep Learning Models. arXiv:2007.03051 [cs.CY]

[2] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. *MBPP (Mostly Basic Python Programming)*. https://github.com/google-research/google-research/tree/master/mbpp

[3] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. *MBPP (Mostly Basic Python Programming)*. https://www.statmt.org/wmt18/translation-task.html#download

[4] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. *WebQuestions*. https://worksheets.codalab.org/worksheets/0xba659fe363cb46e7a505c5b6a774dc8a

[5] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics* 5 (2017), 135–146. https://doi.org/10.1162/tacl_a_00051

[6] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. 2022. On the Opportunities and Risks of Foundation Models. arXiv:2108.07258 [cs.LG]

[7] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. 2016. An Analysis of Deep Neural Network Models for Practical Applications. *CoRR* abs/1605.07678 (2016). arXiv:1605.07678 http://arxiv.org/abs/1605.07678

[8] Andrew A Chien, Liuzixuan Lin, Hai Nguyen, Varsha Rao, Tristan Sharma, and Rajini Wijayawardana. 2023. Reducing the Carbon Impact of Generative AI Inference (today and in 2035). In *Proceedings of the 2nd Workshop on Sustainable Computer Systems* (Boston, MA, USA) *(HotCarbon '23)*. Association for Computing Machinery, New York, NY, USA, Article 11, 7 pages. https://doi.org/10.1145/3604930.3605705

[9] Chroma 2023. *chroma*. Chroma. https://www.trychroma.com/

[10] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. *GSM8K*. https://github.com/openai/grade-school-math

[11] Fabio Duarte. 2024. Number of ChatGPT Users. https://explodingtopics.com/blog/chatgpt-users Accessed: 2024-04-29.

[12] Tamar Eilam, Pedro Bello-Maldonado, Bishwaranjan Bhattacharjee, Carlos Costa, Eun Kyung Lee, and Asser Tantawi. 2023. Towards a Methodology and Framework for AI Sustainability Metrics. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems* (Boston, MA, USA) *(HotCarbon '23)*. Association for Computing Machinery, New York, NY, USA, Article 13, 7 pages. https://doi.org/10.1145/3604930.3605715

[13] Electricity Maps 2001. Electricity Maps. Retrieved April 8th, 2024 from https://app.electricitymaps.com/

[14] Anshul Gandhi, Dongyoon Lee, Zhenhua Liu, Shuai Mu, Erez Zadok, Kanad Ghose, Kartik Gopalan, Yu David Liu, Syed Rafiul Hussain, and Patrick Mcdaniel. 2023. Metrics for Sustainability in Data Centers. *SIGENERGY Energy Inform. Rev.* 3, 3 (oct 2023), 40–46. https://doi.org/10.1145/3630614.3630622

[15] Peter Henderson, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, and Joelle Pineau. 2020. Towards the systematic reporting of the energy and carbon footprints of machine learning. *J. Mach. Learn. Res.* 21, 1, Article 248 (jan 2020), 43 pages.

[16] Geerd-Dietger Hoffmann. 2023. *NOP Linux*. Green Coding Solutions GmbH, Berlin, Germany. https://www.green-coding.io/blog/nop-linux/

[17] Geerd-Dietger Hoffmann. 2023. *Overhead of Measurement Providers*. https://docs.green-coding.io/docs/measuring/metric-providers/overhead-of-measurement-providers/

[18] ISO/IEC 21031:2024 2024. Information technology - Software Carbon Intensity (SCI) specification.

[19] Jinhyuk Lee, Zhuyun Dai, Xiaoqi Ren, Blair Chen, Daniel Cer, Jeremy R. Cole, Kai Hui, Michael Boratko, Rajvi Kapadia, Wen Ding, Yi Luan, Sai Meher Karthik Duddu, Gustavo Hernandez Abrego, Weiqiang Shi, Nithi Gupta, Aditya Kusupati, Prateek Jain, Siddhartha Reddy Jonnalagadda, Ming-Wei Chang, and Iftekhar Naim. 2024. Gecko: Versatile Text Embeddings Distilled from Large Language Models. arXiv:2403.20327 [cs.CL] https://arxiv.org/abs/2403.20327

[20] Nicola Lenherr, René Pawlitzek, and Bruno Michel. 2021. New universal sustainability metrics to assess edge intelligence. *Sustainable Computing: Informatics and Systems* 31 (2021), 100580. https://doi.org/10.1016/j.suscom.2021.100580

[21] Zhishuai Li, Xiang Wang, Jingjing Zhao, Sun Yang, Guoqing Du, Xiaoru Hu, Bin Zhang, Yuxiao Ye, Ziyue Li, Rui Zhao, and Hangyu Mao. 2021. *PetSQL*. https://github.com/zhshlii/petsql

[22] Alexandra Sasha Luccioni, Yacine Jernite, and Emma Strubell. 2023. Power Hungry Processing: Watts Driving the Cost of AI Deployment? arXiv:2311.16863 [cs.LG]

[23] Ggaliwango Marvin, Nakayiza Hellen Raudha, Daudi Jjingo, and Joyce Nakatumba-Nabende. 2024. *Prompt Engineering in Large Language Models*. 387–402. https://doi.org/10.1007/978-981-99-7962-2_30

[24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs.CL] https://arxiv.org/abs/1301.3781

[25] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. 2024. Large Language Models: A Survey. arXiv:2402.06196 [cs.CL]

[26] Tergel Molom-Ochir and Rohan Shenoy. 2021. Energy and Cost Considerations for GPU Accelerated AI Inference Workloads. In *2021 IEEE MIT Undergraduate Research Technology Conference (URTC)*. 1–5. https://doi.org/10.1109/URTC54388.2021.9701614

[27] mixedbread.ai 2024. *mxbai-embed-large-v1*. mixedbread.ai. https://www.mixedbread.ai/blog/mxbai-embed-large-v1

[28] Ollama 2024. *Ollama*. https://ollama.com/

[29] David Patterson, Joseph Gonzalez, Urs Hölzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R. So, Maud Texier, and Jeff Dean. 2022. The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink. *Computer* 55, 7 (2022), 18–28. https://doi.org/10.1109/MC.2022.3148714

[30] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon Emissions and Large Neural Network Training. arXiv:2104.10350 [cs.LG]

[31] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[32] Pranab Sahoo, Ayush Kumar Singh, Sriparna Saha, Vinija Jain, Samrat Mondal, and Aman Chadha. 2024. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. arXiv:2402.07927 [cs.AI]

[33] Siddharth Samsi, Dan Zhao, Joseph McDonald, Baolin Li, Adam Michaleas, Michael Jones, William Bergeron, Jeremy Kepner, Devesh Tiwari, and Vijay Gadepally. 2023. From Words to Watts: Benchmarking the Energy Costs of Large Language Model Inference. 1–9. https://doi.org/10.1109/HPEC58863.2023.10363447

[34] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2019. Green AI. arXiv:1907.10597 [cs.CY]

[35] C. E. Shannon. 1951. Prediction and entropy of printed English. *The Bell System Technical Journal* 30, 1 (1951), 50–64. https://doi.org/10.1002/j.1538-7305.1951.tb01366.x

[36] Emma Strubell, Ananya Ganesh, and Andrew Mccallum. 2019. Energy and Policy Considerations for Deep Learning in NLP. 3645–3650. https://doi.org/10.18653/v1/P19-1355

[37] Neil C. Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso. 2022. The Computational Limits of Deep Learning. arXiv:2007.05558 [cs.LG]

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

[39] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga Behram, James Huang, Charles Bai, Michael Gschwind, Anurag Gupta, Myle Ott, Anastasia Melnikov, Salvatore Candido, David Brooks, Geeta Chauhan, Benjamin Lee, Hsien-Hsin S. Lee, Bugra Akyildiz, Maximilian Balandat, Joe Spisak, Ravi Jain, Mike Rabbat, and Kim Hazelwood. 2022. Sustainable AI: Environmental Implications, Challenges and Opportunities. arXiv:2111.00364 [cs.LG]

accepted 13 June 2024