Detecting Compute Structuring in AI Governance is likely feasible

Emmanouil Seferis [*1] **Tim Fist** [*2 3]

Abstract

Compute structuring, a technique where AI developers split or modify compute workloads for the purpose of avoiding regulation, poses a challenge for AI governance techniques that rely on the computational properties of AI workloads. This work aims to explore the feasibility of detecting compute structuring and to propose robust detection methods. We do this by first exploring possible forms of compute structuring. Using realistic assumptions about cloud providers' capabilities, we derive a potential detection approach. Further, we perform a comprehensive analysis of possible adversary scenarios and show that our method can detect them efficiently. Finally, we analyze potential future trends in AI compute workloads that could invalidate our proposed detection approach, and discuss possible adaptation and mitigation strategies. Overall, our study indicates that compute structuring detection is probably both feasible and practical to implement.

1. Introduction

1.1. Compute-based AI Governance

As AI models at the frontier of research (Bommasani et al., 2021) like GPT-4 (Achiam et al., 2023), Claude (Anthropic, 2024), and others (Reid et al., 2024; Dubey et al., 2024) continue to demonstrate increasingly sophisticated capabilities, the necessity for robust regulation and assurance of their safe usage has become more pressing. Scaling laws (Villalobos, 2023), which observe that as models grow larger and are trained on more data their performance continues to improve, suggesting that this trend of increasing capabilities is likely to persist. Therefore, it is imperative to establish comprehensive governance frameworks to manage the risks and ensure the beneficial deployment of these powerful AI systems.

Compute resources are pivotal in training frontier AI models, making them a critical focus for regulatory measures, and are at the center of recent legislative efforts, such as the EU AI Act (Commission, 2021) and the White House AI executive order (House, 2023). These laws aim to oversee and control the deployment of computational resources to ensure AI systems are developed and used responsibly. The rationale behind using compute as a regulatory tool is well-founded, as detailed in (Sastry et al., 2024). The study argues that by regulating access to and usage of compute resources, authorities can effectively oversee the development of AI models, preventing misuse and ensuring alignment with safety standards.

One innovative proposal for AI governance is through the utilization of cloud infrastructure, as discussed in a recent paper (Heim et al., 2024). This approach advocates for leveraging the centralized nature of cloud computing to implement regulatory oversight mechanisms. By integrating governance protocols within cloud platforms, regulators can monitor and control the deployment of AI models more efficiently.

1.2. The challenge of Compute Structuring

Most AI regulation proposals mainly rely on identifying relevant workloads (such as frontier model pre-training), and/or specifying a threshold in the total amount of compute operations (OPs, integer or floating point), above which additional measures need to be taken. Such measures could include reporting requirements, or subjecting the model to further red teaming and safety testing (Shevlane et al., 2023; Kinniment et al., 2023; Phuong et al., 2024). However, this approach could be susceptible to "compute structuring", a tactic that undermines regulatory efforts by distributing computational workloads in a way that makes it difficult to classify the workload (e.g. as model training or not), and quantify the amount of compute it consumes (Reuel et al., 2024) [1]. In the context of AI, compute structuring can manifest in several ways; tactics include splitting training

[*]Equal contribution [1]Machine Learning Alignment & Theory (MATS) [2]Institute for Progress (IFP) [3]Center for a New American Security (CNAS). Correspondence to: Tim Fist <tim.fist@ifp.org>.

Workshop on Technical AI Governance (TAIG) at ICML 2025, Vancouver, Canada.

[1]This concept bears a striking resemblance to "transaction structuring" in finance, where individuals break up large transactions into smaller ones to avoid triggering reporting requirements meant to prevent money laundering and tax evasion.

across multiple sequential or parallel jobs, using different accounts or providers, or masking workloads as non-AI tasks. These methods complicate oversight by fragmenting compute usage below regulatory thresholds, hindering enforcement efforts.

1.3. Our Contribution

In this work, **we argue that detecting compute structuring is most likely technically feasible**. To that end, we:

- First, building upon the proposal of (Heim et al., 2024), we make a list of assumptions about cloud providers for frontier AI training and their capabilities in tracking key quantities needed for detecting compute structuring. These include, for example, the ability of cloud providers to estimate OPs, interconnect bandwidth, memory allocation and internet traffic sizes. (Heim et al., 2024) shows that the quantities are already collected by cloud providers in a privacy-preserving way.
- Second, we compile a list of all possible compute structuring methods and group them into categories. Based on the core assumptions, we propose a set of methodologies and algorithms to detect compute structuring, within the same job, and across different jobs and providers.
- We demonstrate that, under the assumptions stated, our methodology can detect the compute structuring methods listed. We perform an extensive analysis of possible scenarios, attempting to capture all different possibilities.

Overall, our argumentation shows that detecting compute structuring is technically feasible. We hope that our work can act as a stepping stone towards a technical implementation of compute structuring detection on cloud providers, as well as informing policy and best practices in AI Governance.

2. Towards detecting Compute Structuring

In this section we present our approach towards detecting compute structuring. First, we list a number of key assumptions about frontier AI cloud providers that can be leveraged for compute structuring detection. Then, we analyze the threat models we anticipate, and group them into three main categories. Subsequently, we develop algorithms for detecting and mitigating these threat models. Finally, we demonstrate that our methodology succeeds against the threat models under the specified assumptions, and under which conditions they might fail.

2.1. Frontier AI cloud providers and key assumptions

Training frontier AI models is a costly and complex endeavor, with expenses reaching around $100 million and expected to rise further (Cottier et al., 2024). As a result, only a few companies can maintain the required infrastructure (Floerecke et al., 2023), while most developers access these resources via cloud-based Infrastructure-as-a-Service (IaaS) platforms.

Recognizing this trend, (Heim et al., 2024) propose using cloud providers as governance intermediaries, enabling them to implement AI governance protocols and support regulators in overseeing advanced AI development. Cloud providers can serve as securers, record keepers, verifiers, and enforcers.

Building on their work, we propose compute structuring detection methods that leverage cloud providers' existing monitoring and record-keeping capabilities. We outline key assumptions about their ability to track critical metrics necessary for detecting such practices.

We present these assumptions below, and defer a detailed discussion and justification in Appendix C.

A1: There are only a few cloud providers (< 10) who have the capability of running frontier model training workloads

A2: Cloud providers are able to record key metrics for each submitted workload (such as OPs, bandwidth, traffic and more), in a privacy-preserving manner

A3: Cloud providers can implement a know your customer (KYC) scheme

A4: Cloud providers may report key information about large enough workloads and their owners with each other, or to a centralized regulating body

A5: Customers will want to train a new frontier model within a reasonable amount of time

The above are the key assumptions we plan to use for developing our compute structuring detection methodologies. Unlike other potentially stronger methods such as hardware verification, which are currently not implemented, we see that most pieces we need are already in place, and the above assumptions can be materialized in a very short amount of time.

2.2. Threat models and analysis

Before proceeding, we need to identify all plausible scenarios of compute structuring, and group them into main categories. We outline the following main categories:

- **Sequential Workloads**: This involves breaking up

a large training task into multiple sequential workloads, each below the reporting threshold, and using partially-trained model weights from previous workloads. The different sequential workloads could further be assigned to different cloud accounts or providers, attempting to additionally harden detection. This is the most straightforward and economically feasible adversary scenario, as the technical details of the workload stay almost the same.

- **Disaggregated Workloads**: This category involves splitting a large training task across multiple cloud accounts/providers, where each serves as a data parallel worker within a larger disaggregated training run, periodically sharing weight updates. This group of threats go into the category of federated / decentralized model training.
- **Masking**: This scenario involves disguising the computational signature of a workload to make it look less like large model training, e.g. by using non-standard bandwidth and memory access patterns. Examples could be making a training job to look like inference / model deployment (by adding "fake" internet traffic), or like a large-scale non-AI related simulation, such as weather or climate forecasting, graphics processing, physics simulations, etc.

An important prerequisite towards detecting compute structuring is the step of workload classification, e.g. recognizing the type of a workload that is submitted / running (Reuel et al., 2024), mostly between *training*, *deployment / inference* or *non AI-related* (Appendix A).

2.3. Detection Algorithms

Based on our assumptions about AI cloud providers and the threat models identified, we now propose detection strategies for compute structuring.

We start with the problem of workload classification, which is generally an important problem in technical AI governance (Reuel et al., 2024), and serves as a foundation for compute structuring detection. For our context the problem can be simplified in the following two ways:

- We are mostly interested in grouping workloads into three categories: "training", "potentially training" and "non-training". As we'll see below, these are the crucial categorizations we need for detecting compute structuring.
- We prefer to "err on the side of caution": it's better to flag a workload as "potentially training" while it's not AI training, rather than the opposite. This is especially the case if we expect potentially severe risks from future AI systems (Phuong et al., 2024). In the case where a workload is flagged as being potentially training, further inspections could be performed, such as for example further regulator checking. Since we're dealing with just a few cloud providers and companies that can finance frontier model training, we assume that the cost of a false positive (assuming there's training when it's not) is not too high.

With that, our approach for workload classification is outlines in alg. 1 (Appendix D) and works as follows: First, we retrieve the customers identifying information, and the characteristics of the intended workload, in terms of OPs (C), the throughput R (OPs per second, or OP/s) and the node-to-node bandwidth B (in bytes per second). Then, we compare these values to some pre-specified thresholds. If all are larger, we have a high degree of certainty that the workload could be performing AI training. Otherwise, if some but not all of the values exceed the threshold, we conservatively flag the workload as "potentially training", to be subjected to further inspection. Finally, if all values are below the thresholds, we can say with high confidence that the workload cannot perform AI training.

Thresholds C, R and B have to be suitably chosen by cloud providers so that they are representative of relatively large training runs - but also few orders of magnitude less than the frontier-level thresholds. For example, according to (House, 2023), any frontier model training using more than $C_{max} = 10^{26}$ OPs has to be declared. Based on that, a reasonable choice for C would be 10^{24}. The reason for this is that in compute structuring, it can be the case that multiple training workloads are aggregated into a larger one, and we need to account for that; this will become more apparent shortly. On the other hand, R and B can be the same as the thresholds we'll select for the overall compute structuring algorithm presented next, as these quantities (in contrary to the overall compute) are not aggregative.

We use alg. 1 as a building block for our main algorithm for compute structuring detection (alg. 2, Appendix D). Alg. 2 works as follows: First, using workload classification alg. 1, we analyze and collect all workloads from a given customer that could be potentially training (have sufficiently high total compute, throughput and node-node bandwidth). Then, for all time intervals between the start of the earliest workload and the finish of the last one, we analyze all sequential workloads, and calculate their combined compute and throughput. Since sequential training workloads could be a split-up training run, if the combined values exceed the thresholds, we flag the workloads for further inspection.

Additionally, alg. 2 also addresses the scenario of data-parallel workloads. Specifically, for all workloads that run in parallel and could be potentially training, we measure the size of data exchange between them. If this exceeds

some threshold, the workloads could be part of a joint data-parallel run; in that case, we sum up their combined compute OPs, and proceed similarly to the sequential case. Finally, we perform a last check combining the above two cases together.

For the thresholds here, C will be dictated by the corresponding legislation (e.g. (House, 2023) or (Commission, 2021)). R will be selected sufficiently small such that a frontier model cannot be trained faster than 3-4 years, as required by assumption A5. B is similar as in the workload classification case, and should be a value corresponding to the bandwidth rate of state-of-the-art GPU connectors. Finally, D should be estimated by the size of frontier models times the expected number of gradient updates, reduced by few orders of magnitude for safety reasons.

2.4. Methodology Efficiency Demonstration / Analysis

In this section, we aim to analyze the effectiveness of the proposed detection strategy against the threat scenarios previously outlined.

Claim 1: Workload classification by alg. 1 is effective To see this, we note two insights: first in the context of compute structuring, we do not need extremely precise workload classification; our aim is to merely flag a workload as performing potentially training. If we expect the capabilities of future models to advance into potentially dangerous levels, it's crucial to not fail in detecting a training run, perhaps at the cost of increasing the false alarms; it's preferable to do some unneeded inspections, rather than failing to detect a malicious training run. The second insight is that any masking attempt on a workload must necessarily be additive: that is, a malicious actor can append fake activity on top of a workload, but: the needed compute, throughput and bandwidth can never decrease by masking. Thus, if a workload exceeds these thresholds, it'll be flagged as potential training in all cases. The thresholds should be selected in such a way that frontier-level training is infeasible below them.

Claim 2: Alg. 2 is effective against sequential workloads Indeed, since we monitor the throughput, it doesn't make any difference if the workloads are split or run as a single one. If the individual workloads can be detected as training, so will be the sequence.

Claim 3: Alg. 2 is effective against parallel workloads, unless they can train almost independently In the context of frontier training, data parallel model instances need to communicate gradient updates with each other, typically after each training step. Therefore, two such nodes will communicate data of the order of the model size after each batch. Normally this amount of data is massive, and this is why AI data centers have to be concentrated in the same location and connected by specialized high-bandwidth links.

A potential challenge in this setup comes from advances in decentralized training. For example, (Douillard et al., 2023; Jaghouar et al., 2024) demonstrated the feasibility of decentralized model training for small LLMs in the range of 100 million - 1 billion parameters. Their setup consists of 4 decentralized servers with 8 AI accelerators each, that perform gradient updates much more infrequent than after each mini-batch. Using this, they manage to reduce the node-node communication bandwidth by $500\times$. However, this approach is still unlikely to work against alg. 2 for the following reasons:

- In the context of frontier training, a communication reduction of $1000\times$ or more is still very large and detectable (normally, a training tun should not exchange any data with the outside world; this enables us to flag any workload that has a training signature but also exchanges data as suspicious. Moreover, the data exchange will happen in regular intervals (after a certain number of training steps), giving us a recognizable pattern.
- The decentralized training setups tested are many orders of magnitude less than the state of the art, so it's yet unclear if such approaches will scale. Finally, for frontier models, every decentralized job will be a significant fraction of the frontier.

The only failure mode is if decentralized training turns out to be possible with almost zero or very low data exchange rates. For example, a scenario of this kind would be training independent experts in the context of MoE and then combining them. (Sukhbaatar et al., 2024) claim that to be possible, but again their setup is not comparable to the frontier; in the case of frontier models, the only tested approach is training the MoE model combined.

Thus, we see that our method can indeed detect all threat models outlined, and we uncover the only possible failure case. In Appendix F we try to address this, by describing how our methodology should adapt to potential future advancements.

3. Conclusion

The aim of this work is to tackle compute structuring, a potentially dangerous scenario in AI governance, where adversaries manage to train frontier AI models while evading regulations, by organizing the computation in specific ways. In order to detect this, we list all possible threat models, and propose an effective detection approach, relying on the capabilities of frontier cloud providers. Further, we manage to identify the possible future potential failure cases, and outline a mitigation approach. We hope that our work can aid towards developing successful AI governance methods to ensure that AI remains safe and beneficial for all.

4. Acknowledgments

We would like to thank the Machine Learning Alignment and Theory Program (MATS) for supporting this research, and especially McKenna Fitzgerald, for her active support and helpful discussions throughout the project. Moreover, we would also like to thank the reviewers of the Technical AI Governance (TAIG) Workshop for their thorough and in-depth feedback.

References

Abramson, J., Adler, J., Dunger, J., Evans, R., Green, T., Pritzel, A., Ronneberger, O., Willmore, L., Ballard, A. J., Bambrick, J., et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, pp. 1–3, 2024.

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Anthropic. The claude 3 model family: Opus, sonnet, haiku. *Anthropic Technical Report*, 2024. URL https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Commission, E. Proposal for a regulation laying down harmonised rules on artificial intelligence (artificial intelligence act), 2021. URL https://artificialintelligenceact.eu/the-act/. Accessed: 2024-08-06.

Cottier, B., Rahman, R., Fattorini, L., Maslej, N., and Owen, D. The rising costs of training frontier ai models, 2024.

Douillard, A., Feng, Q., Rusu, A. A., Chhaparia, R., Donchev, Y., Kuncoro, A., Ranzato, M., Szlam, A., and Shen, J. Diloco: Distributed low-communication training of language models. *arXiv preprint arXiv:2311.08105*, 2023.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Epoch AI. Key trends and figures in machine learning, 2023. URL https://epochai.org/trends. Accessed: 2024-08-16.

Floerecke, S., Ertl, C., and Herzfeldt, A. Major drivers for the rising dominance of the hyperscalers in the infrastructure as a service market segment. *International Journal of Cloud Computing*, 12(1):23–39, 2023.

Gade, P., Lermen, S., Rogers-Smith, C., and Ladish, J. Badllama: cheaply removing safety fine-tuning from llama 2-chat 13b. *arXiv preprint arXiv:2311.00117*, 2023.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.

Heim, L., Fist, T., Egan, J., Huang, S., Zekany, S., Trager, R., Osborne, M. A., and Zilberman, N. Governing through the cloud: The intermediary role of compute providers in ai regulation. *arXiv preprint arXiv:2403.08501*, 2024.

Ho, A., Besiroglu, T., Erdil, E., Owen, D., Rahman, R., Guo, Z. C., Atkinson, D., Thompson, N., and Sevilla, J. Algorithmic progress in language models, 2024.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D., Hendricks, L., Welbl, J., Clark, A., et al. Training compute-optimal large language models. arxiv 2022. *arXiv preprint arXiv:2203.15556*, 10, 2022.

House, T. W. Executive order on the safe, secure, and trustworthy development and use of artificial intelligence, executive order 14110, 2023. URL https://www.whitehouse.gov/briefing-room/statements-releases/2023/10/30/executive-order-on-the-safe-secure-and-trustworthy-development-and-use-of-artificial-intelligence/. Accessed: 2024-08-06.

Jaghouar, S., Ong, J. M., and Hagemann, J. Opendiloco: An open-source framework for globally distributed low-communication training. *arXiv preprint arXiv:2407.07852*, 2024.

Kinniment, M., Sato, L. J. K., Du, H., Goodrich, B., Hasin, M., Chan, L., Miles, L. H., Lin, T. R., Wijk, H., Burget, J., et al. Evaluating language-model agents on realistic autonomous tasks. *arXiv preprint arXiv:2312.11671*, 2023.

McCandlish, S., Kaplan, J., Amodei, D., and Team, O. D. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

NVIDIA. A revolution in the making: How ai and science can mitigate climate change, 2021. URL https://blogs.nvidia.com/blog/ai-science-climate-change/. Accessed: 2024-08-16.

OpenAI. Techniques for training large neural networks. https://openai.com/index/techniques-for-training-large-neural-networks/, 2022. Accessed: 2024-08-10.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Patel, Dylan and Nishball, Daniel. 100,000 h100 clusters: Power, network topology, ethernet vs infiniband, reliability, failures, checkpointing, 2024. URL https://www.semianalysis.com/p/100000-h100-clusters-power-network. Accessed: 2024-09-11.

Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L., Rothchild, D., So, D., Texier, M., and Dean, J. Carbon emissions and large neural network training. arxiv 2021. *arXiv preprint arXiv:2104.10350*.

Phuong, M., Aitchison, M., Catt, E., Cogan, S., Kaskasoli, A., Krakovna, V., Lindner, D., Rahtz, M., Assael, Y., Hodkinson, S., et al. Evaluating frontier models for dangerous capabilities. *arXiv preprint arXiv:2403.13793*, 2024.

Reid, M., Savinov, N., Teplyashin, D., Lepikhin, D., Lillicrap, T., Alayrac, J.-b., Soricut, R., Lazaridou, A., Firat, O., Schrittwieser, J., et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Reuel, A., Bucknall, B., Casper, S., Fist, T., Soder, L., Aarne, O., Hammond, L., Ibrahim, L., Chan, A., Wills, P., et al. Open problems in technical ai governance. *arXiv preprint arXiv:2407.14981*, 2024.

Sastry, G., Heim, L., Belfield, H., Anderljung, M., Brundage, M., Hazell, J., O'Keefe, C., Hadfield, G. K., Ngo, R., Pilz, K., et al. Computing power and the governance of artificial intelligence. *arXiv preprint arXiv:2402.08797*, 2024.

Sevilla, J., Besiroglu, T., Dudney, O., and Ho, A. The longest training run, 2022. URL https://epochai.org/blog/the-longest-training-run. Accessed: 2024-08-13.

Sevilla, J., Besiroglu, T., Cottier, B., You, J., Roldán, E., Villalobos, P., and Erdil, E. Can ai scaling continue through 2030?, 2024. URL https://epoch.ai/blog/can-ai-scaling-continue-through-2030. Accessed: 2025-01-28.

Shevlane, T., Farquhar, S., Garfinkel, B., Phuong, M., Whittlestone, J., Leung, J., Kokotajlo, D., Marchal, N., Anderljung, M., Kolt, N., et al. Model evaluation for extreme risks. *arXiv preprint arXiv:2305.15324*, 2023.

Sukhbaatar, S., Golovneva, O., Sharma, V., Xu, H., Lin, X. V., Rozière, B., Kahn, J., Li, D., Yih, W.-t., Weston, J., et al. Branch-train-mix: Mixing expert llms into a mixture-of-experts llm. *arXiv preprint arXiv:2403.07816*, 2024.

Tamirisa, R., Bharathi, B., Phan, L., Zhou, A., Gatti, A., Suresh, T., Lin, M., Wang, J., Wang, R., Arel, R., et al. Tamper-resistant safeguards for open-weight llms, 2024. *URL https://arxiv. org/abs/2408.00761*, 2024.

Villalobos, P. Scaling laws literature review. *Published online at epochai. org*, 2023.

A. Terminology

Here we provide some background terminology about AI data centers and workloads (Heim et al., 2024).

A **Workload** is a computational task to be performed on the data center, such as for example training an AI model.

An **OP** is a single operation run on computer hardware, typically a multiplication or addition of two numbers. AI workloads typically require a large number of matrix multiplication operations, usually as floating-point (FLOP) or integer operations.

AI accelerators are specialized chips to perform matrix multiplications fast and parallelized, typically used in AI training or interference; for example GPUs and TPUs. These devices are optimized for AI training, featuring trillions of matrix-multiply OPs per second (OP/s or FLOP/s), large and fast memories to fit model parameters, and very fast bandwidth interconnect for inter-device communication.

A **node or server** is a single computer within a data center that contains multiple AI accelerators, as well as its own dedicated CPU, memory and storage.

A **(computing) cluster** is a group of linked nodes that can work together to process a workload.

Finally, a **data center** is a facility housing computing clusters, as well as the equipment and infrastructure needed to operate them.

Generally, AI workloads can belong in one of the following categories (Heim et al., 2024):

- **Design**, in which AI researchers perform experiments with new architectures, hyperparameters etc. These are typically run at multiples in parallel, and at small scales.
- **Training**, where the core model training is performed. This step is also referred to as "pre-training", to distinguish it from further fine-tuning steps.
- **Enhancement**, where a trained model is further fine-tuned on smaller datasets, for example in order to return more coherent answers in the context of LLMs. This step may also involve some form of Reinforcement Learning (Ouyang et al., 2022).
- **Deployment**, in which a trained model is used to make predictions ("inference"), typically served to customers.
- **Non-AI related**, where we classify any other workloads. Those may also use AI accelerators in some cases, but for other purposes (e.g. weather / physics simulations, etc.).

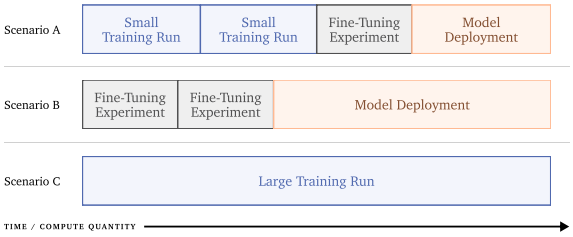

Figure 1. Example illustration of different workload types using the same amount of compute (figure from (Heim et al., 2024)).

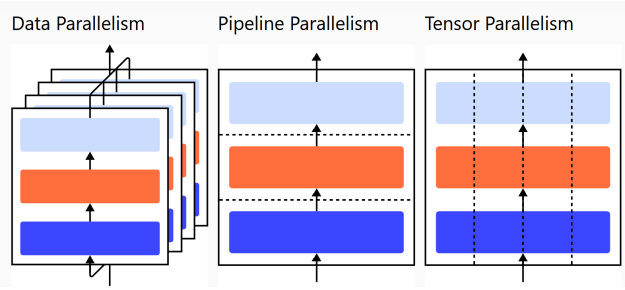

Figure 2. Illustration of different model parallelism types; each color refers to one layer and dashed lines separate different AI accelerators (figure from (OpenAI, 2022)).

Being able to distinguish the type of a workload is necessary in performing compute accounting for a given AI developer. For example, fig. 1 shows an example where the same amount of compute is used in different ways for different scenarios.

B. Basics on large AI training runs

The main challenge of frontier AI model training is to distribute massive computational loads across multiple nodes. The basic setups are as follows (fig. 2):

Single node workload: This is the most basic scenario, where the model can fit within a single AI accelerator. The dataset is split in batches, and the model parameters are updated on each batch, by performing stochastic gradient descent (SGD) (Goodfellow et al., 2016), where we perform a forward pass to get the model's prediction, and a backward pass to update the parameters towards decreasing the task loss. For Transformer models, we can estimate the number of OPs required for the forward and backward pass as 6N, where N is the number of model parameters (Hoffmann et al., 2022).

Data parallelism: In data parallelism, the dataset is split across multiple AI accelerators, each of which holds a copy of the entire model. Each accelerator processes a different mini-batch of data, and the gradients are aggregated and

synchronized across all accelerators after each forward and backward pass. In that way, multiple data batches can be processed in parallel. This method is straightforward but can become bandwidth-intensive as the model size increases, requiring efficient communication between different AI accelerators. For these reasons, AI accelerators within the same server, as well as servers within the same cluster are communicating with specialized high-bandwidth connections. The gradient accumulation across devices is typically performed with a so-called AllReduce operation, where gradients within servers and then across servers are summed in an efficient, hierarchical fashion.

Model parallelism: In model parallelism, the model itself is split across multiple AI accelerators. Each accelerator handles only a part of the model, which allows the training of models that would otherwise be too large to fit into the memory of a single GPU. However, model parallelism introduces complexity in managing dependencies between different parts of the model and often requires sophisticated coordination to minimize idle time during training. In a naive implementation, the accelerator computing the output of the first layers would have to wait idle until the subsequent accelerators compute the remaining forward and backward passes, resulting in an inefficient hardware utilization. To mitigate this, batches are typically further split into micro-batches, so that each device can immediately start computing the forward pass of the next micro-batch while waiting for the backward pass of the previous ones.

Tensor parallelism: Tensor parallelism is another possibility for splitting a large model into multiple AI accelerators: this time, instead of dedicating each layer to a different device, we can split the same layer across devices; each device computes a part of the layer output, and the results are then aggregated. As this needs to happen for each forward and backward pass, efficient communication between accelerators is crucial, even more than in the previous setups.

Pipeline parallelism: Finally, Model, Tensor and Data parallelism can be combined together (especially for very large frontier models) leading to Pipeline parallelism.

Expert parallelism: Additionally, in the case of Mixture of Experts, each token may be computed by a different part of the model. This on the one hand makes splitting a large model across accelerators easier (as each accelerator can implement only a single expert independently), but routing between experts has to be implemented in a communication-efficient manner.

To further optimize the training of large models, additional methodologies have been proposed, such as mixed-precision training (where e.g. we train a model using faster integer operations instead of more expensive floating-point ones, with minimal effect on model accuracy), check-pointing, trading compute for memory and others. We refer the reader to (OpenAI, 2022) for a gentle introduction, as well as (Dubey et al., 2024) for an in-depth description of the hardware setup and process to train the Llama-3 models.

Frontier AI model training and deployment involves multiple challenges, including managing the specialized hardware equipment, handling compute and energy costs, efficient data management, device debugging and monitoring, and more. Frontier AI data centers resolve these challenges for their customers, providing them a simple abstraction to submit their workloads, and resource compartmentalization for each workload.

C. Detailed Assumptions on AI Cloud Providers

In the following, we discuss and elaborate further the assumptions on AI cloud providers stated in the main text.

A1: There are only a few cloud providers (< 10) who have the capability of running frontier model training workloads

This is expected due to the rising cost of training frontier models, the massive upfront cost of the (specialized) equipment and operation / maintenance, as well as future challenges such as energy requirements. The precise number of providers is not significant.

A2: Cloud providers are able to record key metrics for each submitted workload, in a privacy-preserving manner This assumption is analyzed in depth in (Heim et al., 2024), where the authors remark that cloud providers are already tracking most quantities needed; more specifically:

- Hardware configuration requested and time duration: Customers need to declare upfront the hardware configuration they require (number of nodes, usage duration, node and AI accelerator technical specifications) in order to submit a workload. This information is needed by the cloud provider to reserve the required resources and assign them to the workload.
- Cluster-level technical information: network bandwidth between nodes, data ingress/egress, energy consumption: This information is already collected by cloud providers for equipment monitoring purposes.
- Granular node-level data such as AI accelerator core utilization or AI accelerator memory bandwidth utilization: this information can be collected by existing tools, and is collected by some cloud providers.
- Workload-level technical information (code, data, hyperparameters): These are not collected, as it would violate the customer's privacy. However, (Heim et al.,

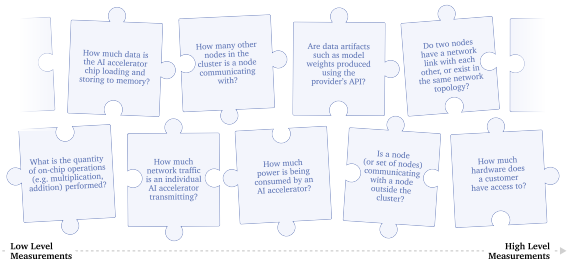

Figure 3. Illustration of possible compute usage metrics for AI workload analysis, organized by granularity level (figure from (Heim et al., 2024)).

2024) argues that this information could potentially be hardware-attested if needed, by means of trusted computation methodologies, in a privacy-preserving way. The authors claim that the required technology already exists, and would take 2-4 years to integrate in AI accelerators and nodes.

The above measurements are depicted in fig. 3, from lower to higher level of granularity.

A3: Cloud providers can implement a know your customer (KYC) scheme

Cloud providers are already required to request and keep track of customer identity verification, including data such as name, billing address, credit card data, IP addresses, date and time of access, device identifiers, language and more. With these, cloud providers can essentially know which customer runs which workloads.

A4: Cloud providers may report key information about large enough workloads and their owners with each other, or to a centralized regulating body

This is currently only partially implemented, and is the main proposal of (Heim et al., 2024). Note that performing this is easy, as cloud providers already collect the required information by assumptions A2 and A3. Moreover, this can be done in a privacy-preserving manner, by e.g. employing customer identifiers instead of legal names. The task can be further facilitated by A1, as there exist only a few frontier cloud providers who need to exchange data with each other and with regulators. The main challenge to achieve this is political / legislative, as different countries / regions (USA, EU, China) with different jurisdiction will need to coordinate with each other, and establish an international regulatory body and a standardized procedure (Heim et al., 2024). However, could providers may be willing to self-regulate and exchange key information with each other prior / without the establishment of a regulation authority.

A5: Customers will want to train a new frontier model within a reasonable amount of time

Developing a frontier AI model is a difficult task, and apart from training, it requires further steps that take time, such as enhancement (fine-tuning), safety-testing and deployment. The aim of our approach is not to make it impossible for a malicious actor to train a frontier model at all (which is infeasible), but to disallow them from doing so within a reasonable amount of time. Following (Sevilla et al., 2022), we assume that an AI developer will want to train their model within a maximum time of around 18 months (the typical duration is much shorter, about 100 days). If our detection method can prevent this and force malicious actors into much larger deployment times (3 years or more), we consider this as "effectively safe": within that time, the state-of-the-art will shift, outpacing the malicious developer, and the approach will give us sufficient time to react.

Algorithm 1 Workload classification for compute structuring

Input: workload information W (including workload initial requirements and runtime measurements)
Constants: AI OPs threshold C, throughput threshold R (in OP/s), inter-node bandwidth threshold B (in GB/s)
Output: workload classification ("training", "potentially training", "non-training")
retrieve customer identifying information $I \leftarrow getCustomerID(W)$
get workload declaration from customer $d \leftarrow getWDeclaration(W)$
if $d =$ "$training$" **then**
    return "training"
end if
retrieve W's theoretical OPs C_W, time duration t_W and node-node bandwidth B_W
if $C_W \geq C$ and $C_W/t_W \geq R$ and $B_W \geq B$ **then**
    return "training"
else if $C_W \geq C$ or $C_W/t_W \geq R$ or $B_W \geq B$ **then**
    return "potentially training"
else
    return "non-training"
end if

D. Complete Compute Structuring Detection Algorithms

E. Analysis of False Positives

A potential challenge with the outlined approach is the possibility of raising too many False Positives, that is flagging workloads as suspicious (frontier training run exceeding regulatory threshold) when in fact they're benign. In that

Algorithm 2 Compute Structuring detection

Input: customer ID ID, list of active workloads W_i (currently running or submitted to be run in the future)
Constants: AI OPs threshold C, throughput threshold R (in OP/s), inter-node bandwidth threshold B (in GB/s), cluster-cluster communication size (GB) D
Output: customer classification ("benign", "to-be-inspected")
get earliest workload starting time t_{start} and latest ending time t_{end}.
for all time intervals $[t_s, t_e]$, $t_s \geq t_{start}$, $t_e \leq t_{end}$ **do**
  gather all workloads W_{all} active within $[t_s, t_e]$ flagged as "potentially training" by alg. 1
  gather all subsets of workloads W_s that are sequential in time, get total time duration t_{comb} and aggregated compute C_s
  if $C_s \geq C$ **or** $C_s/t_{comb} \geq R$ **then**
    return "to-be-inspected"
  end if
  gather all subsets of active workloads W_p running in parallel that exchange data with each other beyond D, and aggregate their compute C_p
  if $C_p \geq C$ **or** $C_p/t_{comb} \geq R$ **then**
    return "to-be-inspected"
  end if
  gather both sequential and parallel workloads exchanging data beyond D, aggregate their compute C_{sp}
  if $C_{sp} \geq C$ **or** $C_{sp}/t_{comb} \geq R$ **then**
    return "to-be-inspected"
  end if
end for

case, an implementation of the proposed methodology may be opposed by frontier labs and other organizations at the forefront of AI research. To investigate this possibility, in this section we perform an analysis of the expected number of False Positives under different scenarios and realistic assumptions.

We identify the following scenarios:

Model serving / inference: This is typically where the largest usage of AI accelerators takes place (Epoch AI, 2023) (60% - 90%), in order to serve models to millions of customers worldwide. Typically, during interference, independent copies of the model are served from different servers, and these workloads are small individually, and don't exchange data with each other. Hence, alg. 1 and 2 will not flag them, resulting in no False Positives in this case.

Model fine-tuning / experiments: Model fine-tuning and other experiments that typically happen in AI labs are well below the compute threshold limits. Moreover, these workloads are isolated, and don't exchange data with each other. Hence, our approach will not flag them, resulting in no False Positives.

Non-AI large workloads: Alg. 1 and 2 will flag any workload that has a signature of a large AI training run, using a large enough amount of compute, specialized accelerators and interconnect bandwidth. Potentially, this could also flag non-AI related workloads, such as weather or physics simulations, or others. However, current non-AI large workloads do not have the typical signature of AI training runs (they do not need data-center GPUs, high interconnect, may also use CPUs, etc), or in the case they use AI accelerators (such as e.g. Neural Network - based weather simulations (NVIDIA, 2021)) the resources used are orders of magnitude below frontier AI training runs. Thus, we expect that our approach will result in only very few False Positives in this case. In the Appendix, we list some of the currently largest High Performance Computing (HPC) runs and inspect their differences with Deep Learning training.

Independent training runs running in parallel: This is another scenario, where an AI organization trains the different relatively large models in parallel, for example one Large Language Model and one Protein Folding Model like AlphaFold (Abramson et al., 2024). These are two independent workloads that run in parallel and do not exchange data with each other. Thus, alg. 1 and 2 will not flag them, except if any individual workload exceeds the thresholds. Hence, we do not expect False Positives in this setting.

Independent training runs happening sequentially: This is the only case where our approach may raise significant False Positives. For example, imagine an AI lab that runs two large workloads sequentially, but each one is independent from the other, e.g. one Large Language Model and one Protein Folding Model. Since alg. 1 and 2 limit the overall throughput of sequential workloads, they may flag this scenario as a potentially single large training run in case the combined workloads exceed the throughput and the compute threshold, even if they're independent in reality. However, this will only happen if the individual workloads are close enough to the threshold. It is unclear how often this scenario happens.

Table 1 summarizes the above findings.

F. Limitations and Mitigation Strategy

As we saw previously, our proposal is effective against all threat models within the current context of ML. However, it may be the case that future advancements shift the landscape and pose further challenges. Therefore, it's important to outline a continuous observation and mitigation approach.

Table 1. expected number of False Positives of alg. 1 and 2 for different scenarios.

Scenario	Expected number of False Positives
Model serving / inference	None or very few
Model fine-tuning / experiments	None or very few
Non-AI large workloads	None or very few
Independent training runs happening in parallel	None or very few
Independent training runs happening sequentially	False Positives in case of large enough independent sequential workloads. We don't expect this to happen often.

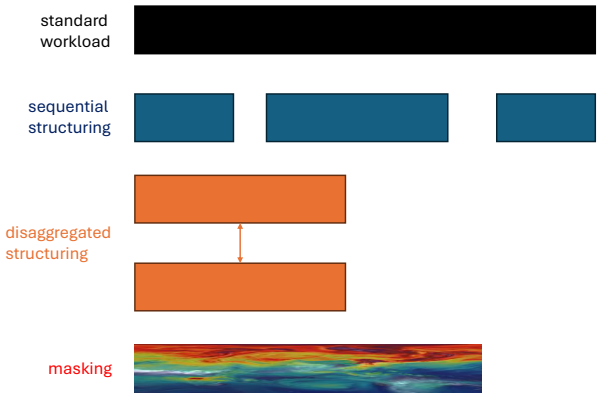

Figure 4. Illustration of the different compute structuring scenarios.

Algorithmic Improvements The compute thresholds currently dictated by regulations such as (House, 2023) are based on the current state of the art in ML training. However, works such as (Ho et al., 2024; Epoch AI, 2023) estimate a trend of around $3\times$ algorithmic improvements in AI training: that is, if training a model with some level of capability requires x OPs today, it may require only $x/3$ next year.

To account for that, it's crucial that our proposed strategy is revisited at regular time intervals. Ideally, subject experts should constantly monitor new developments in ML, estimate trends, and regularly communicate with policy makers, so that mitigation approaches can be adjusted as needed. On the positive side, we see that the detection approach of alg. 2 remains valid in any case, and only the magnitude of the thresholds may need to be adjusted.

Advances in Decentralized Training The only failure case for our approach, as identified before, are future advancements in decentralized frontier model training, where the communication between the different parallel workloads is almost zero. Furthermore, if the individual workloads need not be large (say 10 times a $10\times$ smaller workload, but 100 times a $100\times$ smaller workload) this would make detection even more challenging. To mitigate this, it's crucial that government institutes monitor the state-of-the-art in ML constantly and discuss with policy makers at a frequent and regular basis. Also, such an advancements may make hardware-based attestation methods necessary.

Easiness of safety fine-tuning removal for open-weight models Although our work focuses mostly on the case of detecting frontier model training, this is also a failure scenario that needs to be discussed, in the case of open-weight frontier AI models (Dubey et al., 2024). Namely, researchers have demonstrated (Gade et al., 2023) that it can be easy and cheap to remove the safety fine-tuning (enhancement step) from LLMs. This can pose a significant threat if frontier AI models are open-sourced in the future, as malicious actors could bypass this security measure and then use them for illegal actions. Ideally, the cost of removing the safeguards should be comparable to training the model itself. A very recent work (Tamirisa et al., 2024) proposes an approach towards this direction, but its general efficacy on frontier open-source models remains to be seen [2].

G. Additional Figures

A simple illustration of the identified compute structuring scenarios can be seen in fig. 4:

H. A simple model for estimating the time, cost and energy to train a frontier model

In this section, we aim to develop a simple model for estimating the time, cost and energy requirements of a large transformer model training run, given the model and dataset sizes, as well as some key information about the AI accelerators that are going to be used, such as their computing speed in FLOPs per second, memory, bandwidth interconnect speed and energy consumption.

Our goal is not to calculate the above quantities precisely (as it could be challenging), but to find a reasonable estimate that is correct within a relatively small correction factor (less

[2]Current open and closed-source models are arguably safe; assuming that dangerous capabilities emerge at e.g. an $100\times$ compute increase than current LLMs, the difference when starting from some current pre-trained model vs training from scratch is negligible. Hence, this concern is mostly for future open-weight models.

than an order of magnitude, ideally $\pm 50\%$). To achieve this, we're going to do a series of simplifications:

- Focus on Data Parallelism and ignore Model and Tensor Parallelism.
- Concentrate on Transformers, as they are the predominant architecture for frontier models. According to (Hoffmann et al., 2022), each token requires approximately $6N$ parameters to perform the forward and backward pass, where N is the number of model parameters.
- Use 16-bit (2-byte) floating point format (bfloat16) for all numbers.

Our model will use the following parameters:

- N: Model size (number of parameters)
- D: Dataset size (number of tokens)

Additionally, we'll assume the following parameters for the training cluster:

- n: Number of AI accelerators (GPUs)
- F: GPU FLOPs per second (FLOP/s) in the bfloat16 format
- M: GPU memory in bytes
- B: GPU interconnect speed (in bytes/s). In the simplified version of our model we won't distinguish between inter-server and intra-server bandwidth (which typically differ a bit), and take both to be the same as the GPUs maximum data transfer rate
- W: GPU power consumption (in Watt)

Next, based on that, we present how our model estimates the key quantities required.

Let's start with the time required for training. A lower bound (T_{flop}) of it is the time required to process all tokens; since each token requires $6N$ FLOPs, and we have D tokens in total, the total training FLOPs required are $6ND$; parallelizing this to n AI accelerators, each with F FLOP/s each, gives us a total time of:

$$T_{flop} = \frac{6ND}{nF} \tag{1}$$

Typically, only a fraction of the maximum possible FLOPs of a GPU is utilized; this is accounted for by the Mean FLOP Utilization factor, which for modern data center GPUs is

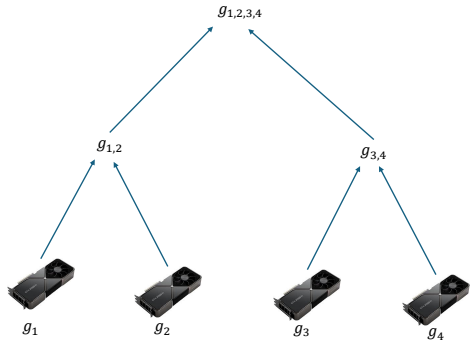

Figure 5. Illustration of gradient communication between GPU devices (4 devices in this example). Initially, each GPU has a local copy of the gradient for its own micro-batch ($g_1, ..., g_4$). At step 1, each accelerator synchronizes (averages) it's gradient with its neighbor, resulting in accumulated gradients $g_{1,2}$, $g_{3,4}$ that are synchronized across "segments" of 2 devices. Then at step 2, devices 1 and 3 (and simultaneously in parallel devices 2 and 4) accumulate gradients $g_{1,2}$ and $g_{3,4}$ to produce the overall gradient $g_{1,2,3,4}$. A the number of synched devices doubles at each round, we see that the overall number of steps needed is $\log_2(n)$, where n is the total number of AI accelerators.

estimated to be about $a_{MFU} = 40\%$ (Patel, Dylan and Nishball, Daniel, 2024). This means that the actual FLOP/s we get can be estimated as $F = a_{MFU} F_{max}$, where F_{max} is the maximum value specified by the manufacturer.

However, this estimate ignores the communication time required after each model update (gradient step), where all GPUs need to synchronize with each other and update model parameters.

At each gradient step, accelerators need to send the content of their memory to each other; this will require a time of M/B seconds. It's reasonable to assume that the memory of each device will be almost fully utilized (as we want to process as many tokens in parallel as possible), thus we estimate here that the gradient parameters in each device will take up almost the full memory.

Additionally, all n accelerators need to communicate their gradients with each other. The most efficient way to do this is with a tree-like structure, where at each step the number of synched devices doubles. Overall, this requires a communication time t_{comm} after each model update that is equal to:

$$t_{comm} = \frac{M}{B} \log_2(n) \tag{2}$$

Assuming that each GPU can perform computations and communicate data in parallel, during a gradient update time t_{comm}, the GPUs can process a batch size b of tokens, where b can be found by the equation: $6Nb = nFt_{comm}$ (since this is the number of FLOPs possible to perform within time t_{comm}). Therefore, this limits the batch size to:

$$b = \frac{nFt_{comm}}{6N} = \frac{nFM\log_2(n)}{6NB} \tag{3}$$

Hence, the time needed to process the entire dataset of D tokens would be: $T_{comm} = \frac{D}{b}t_{comm} = \frac{6ND}{nF} = T_{flop}$; e.g. in theory, communication latencies could be compensated by a large enough batch size (b grows as B decreases, and becomes arbitrarily large when B is close to zero).

However, in practice this is not the case: it turns out that after some "critical batch size" b_{max}, increasing the batch size further yields diminishing and even zero returns (Sevilla et al., 2024; McCandlish et al., 2018). For GPT-4, the batch size is estimated to be around $6 \cdot 10^6$ tokens, and this value is regarded as close to the critical size for language modeling. Therefore, a more reasonable estimate is:

$$b = \min\left(\frac{nFM\log_2(n)}{6NB}, b_{\max}\right) \tag{4}$$

Finally, the training time is estimated as:

$$T = \frac{D}{b}t_{\text{comm}} = \frac{M \cdot D \cdot \log_2(n)}{B \cdot \min\left(\frac{n_{FM}\cdot\log_2(n)}{6NB}, b_{\max}\right)} \tag{5}$$

For the energy consumption, we simply multiply the consumption in Watt per accelerator, by their number and training time. Additionally, we need to account for the fact that the total power consumption of the data center does not come only from the GPUs, but also from the rest of the equipment, cooling etc. We can account for these by the power usage effectiveness (PUE) coefficient a_{PUE}, which is estimated around $a_{PUE} = 1.3$ for modern GPUs such as the NVIDIA H100 (Patterson et al.). Hence, the used energy is:

$$E = n \cdot a_{PUE}W \cdot T \tag{6}$$

The various training costs can then be estimated from these figures. For example, the electricity cost is the energy used divided by the cost per Joule (c_J):

$$C_{energy} = \frac{E}{c_J} \tag{7}$$

For example, for industrial electricity in the US we have $c_J = 0.08\$/kWh = 2.22 \cdot 10^{-8}\$/J$[3].

If the GPUs are rented from the cloud, the cost is calculated by the GPU cost rate per second c_{rent}; for example, for NVIDIA H100s, we have $c_{rent} = 2.5\$/h = 6.9 \cdot 10^{-4}\$/s$ for the cheapest option we found [4]. The total cost is then:

$$C_{rent} = nTc_{rent} \tag{8}$$

Finally, the hardware acquisition cost is

$$C_{hardware} = nP_{GPU}c_{adj} \tag{9}$$

where P_{GPU} is the price per GPU, and c_{adj} is a factor to adjust for the additional data center and infrastructure costs, and can be approximated by $c_{adj} \approx 2$ as a rule of thumb (Patel, Dylan and Nishball, Daniel, 2024).

With that, we can now analyze some simple examples.

Example 1: GPT-4

For GPT-4, researchers estimate a model size $N = 10^{12}$ parameters and $D = 13 \cdot 10^{12}$ tokens [5]. The model was trained on $n = 10^4$ NVIDIA H100 equivalents, for around 90 days. For the H100, we have $F = 1980 \cdot 10^{12}$ FLOP/s, $M = 80 \cdot 10^9$ bytes, $B = 900 \cdot 10^9$ bytes/s from the specs [6]. With that, our model gives:

$$\begin{aligned} T = 114 \text{ days}, \quad E = 2.49 \cdot 10^7 \text{kWh}, \\ C_{energy} = 2 \cdot 10^6\$, C_{rent} = 67.95 \cdot 10^6\$, \\ C_{hardware} = 500 \cdot 10^6\$ \end{aligned} \tag{10}$$

These values are close to the public estimates.

Example 2: Llama 3.1 405b

For the largest Llama model, we have a size of $N = 405 \cdot 10^9$ and $D = 15.6 \cdot 10^{12}$; the model was trained on a cluster of $n = 16 \cdot 10^3$ NVIDIA H100 GPUs. We get:

[3] https://www.statista.com/statistics/1395805/monthly-electricity-price-industrial-sector-united-states/

[4] https://lambdalabs.com/service/gpu-cloud

[5] https://the-decoder.com/gpt-4-architecture-datasets-costs-and-more-leaked/

[6] https://www.nvidia.com/en-us/data-center/h100/

$$T = 35 \text{ days}, \quad E = 1.21 \cdot 10^7 \text{kWh},$$
$$C_{energy} = 0.97 \cdot 10^6\$, C_{rent} = 33.03 \cdot 10^6\$, \quad (11)$$
$$C_{hardware} = 800 \cdot 10^6\$$$

These values are again close to the figures reported in the paper; e.g. the training time was around 60 days. The discrepancy is due to the fact that the training of Llama 3.1 was not as straightforward as described above, but was done in different stages, where the first stage used half of the available GPUs.

Example 3: the cluster of 4090s

An important threat model in AI governance is the possibility of model training using non-specialized or even consumer-grade equipment. To estimate this, we run a hypothetical scenario where one attempts to train a GPT-4 model using a cluster of the most powerful consumer-grade GPUs, the NVIDIA 4090.

For the 4090, we have the following data: $F = 330 \cdot 10^{12}$ FLOP/s (for the bfloat16 format) and $M = 24 \cdot 10^9$ bytes from the specs [7]. For the bandwidth, technologies such as the NVLink work only with data center GPUs. Here, we'll assume a high-end standard Ethernet connection of $B = 100$Gbit/s $= 12.5$GB/s.

First, to get the same FLOP/s as the 10000 H100 cluster, we need the following number of 4090s: $n = 10^4 F_{H100}/F_{4090} = 6 \cdot 10^4$ GPUs. For the training time, assuming the same a_{MFU} coefficient (which is fairly optimistic) we get:

$$T = 114 \text{ days}, \quad E = 9.6 \cdot 10^7 \text{kWh}, \quad (12)$$

For the costs, we have the price $P_{GPU} = 1500\$$, and assume $c_{rent} = 0.3\$/h = 8.3 \cdot 10^{-5}\$/s$. With these, we get:

$$C_{energy} = 7.67 \cdot 10^6\$, C_{rent} = 49.22 \cdot 10^6\$,$$
$$C_{hardware} = 180 \cdot 10^6\$ \quad (13)$$

However, this simple model ignores multiple details that would make a 4090 cluster much harder to use in practice: the difficulty of coordinating that large number of devices without specialized infrastructure and software, the much higher expected failure rates of the devices (which is a substantial challenge in large training runs), the difficulty of building cooling solutions for such a cluster, and many more. Moreover, we see that the overall costs would still be similar as before, even in this simplified setup. Yet on the other hand, the calculation shows the hypothetical feasibility of the setup, if one managed to solve the above technical problems.

[7] https://www.nvidia.com/en-eu/geforce/graphics-cards/40-series/rtx-4090/