Detecting Compute Structuring in AI Governance is Likely Feasible

Emmanouil Seferis[*1], **Tim Fist**[*2, 3]
[1]Machine Learning Alignment & Theory (MATS)
[2]Institute for Progress (IFP)
[3]Center for a New American Security (CNAS)
tim.fist@ifp.org

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.

Introduction

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. These advanced models have shown remarkable prowess in natural language understanding, generation, and other complex tasks, raising both opportunities and challenges. 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. Recent legislative efforts, such as the EU AI Act (Commission 2021) and the White House AI executive order (House 2023), have recognized the importance of compute in the AI development pipeline. 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. This method offers a scalable and flexible solution, enabling real-time oversight and the enforcement of compliance standards across diverse AI applications. The proposal highlights the potential of cloud-based governance to enhance transparency, accountability, and security in the rapidly evolving AI landscape.

The challenge of Compute Structuring

These proposals and regulations 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). 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.

In the context of AI, compute structuring can manifest in several ways. One plausible method involves dividing a large

[*]These authors contributed equally.

training workload into multiple sequential workloads, each taking as input partially-trained weights from the previous workload, and each executed using a different cloud account or provider. Each segment of the workload falls below the reporting threshold, yet together they cumulatively achieve the same result as a single, larger workload. Another approach is distributing a large workload across several cloud accounts or providers, where each account functions as a part of a larger data-parallel training run, periodically sharing updates to maintain synchronization. Finally, yet another possibility is masking an AI workload to appear that it's non-AI related. These practices can significantly hinder regulatory efforts to monitor and control the use of computational resources in AI development.

Our Contribution

In this work, **we make the case 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 upload/download internet traffic sizes. Heim et al.15 shows that the quantities are already collected by cloud providers in a privacy-preserving way. Moreover, we require providers to implement a Know-Your-Customer (KYC) scheme, and report large-enough training jobs to a central regulating body in an anonymized 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.
- Finally, we discuss possible future trends in terms of potential algorithmic improvements, and how our methodology can adapt for them.

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.

Background

In this section, we present some background material that is later required in the main sections.

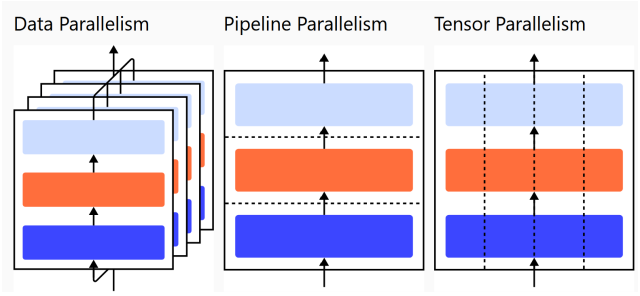

Figure 1: Illustration of different model parallelism types; each color refers to one layer and dashed lines separate different AI accelerators (figure from (OpenAI 2022)).

Basics on large AI training runs

In this section we provide some background information on large scale AI data centers and frontier model training.

The main challenge of frontier AI model training is to distribute massive computational loads across multiple nodes. The basic setups are as follows (fig. 1):

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, Bengio, and Courville 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.

In order 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), checkpointing, 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.

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 will indeed be successful against the threat models under the specified assumptions, and under which conditions they might fail.

Frontier AI cloud providers and key assumptions

As discussed in the previous section, training a frontier AI model and maintaining the hardware and infrastructure to do so is an expensive and difficult process. Further, the cost of training frontier models is currently in around 100 million dollars, and is expected to rise significantly more in the future (Cottier et al. 2024). Due to these factors, only a handful of companies are able to afford and maintain the infrastructure needed for frontier AI model training (Floerecke, Ertl, and Herzfeldt 2023). Other AI developers typically access these resources in an Infrastructure-as-a-Service (IaaS) model, through cloud computing.

Recognizing this trend, (Heim et al. 2024) propose to utilize frontier AI cloud providers as a governance intermediary: cloud providers can implement and integrate AI governance protocols, allowing or helping regulators to monitor and control the development and deployment of advanced AI models. Providers can have multiple roles, acting as securers, record keepers, verifiers and in some cases enforcers.

In this work, we build upon the proposal of (Heim et al. 2024), and propose compute structuring detection methodologies that are integrated to the workflow of frontier AI cloud providers, and utilize their record keeping and monitoring capabilities. More specifically, we list a series of key assumptions about cloud providers and their capabilities in tracking key quantities needed for detecting compute structuring. In the following, we present them, along with a description for each one.

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 in order 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. 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

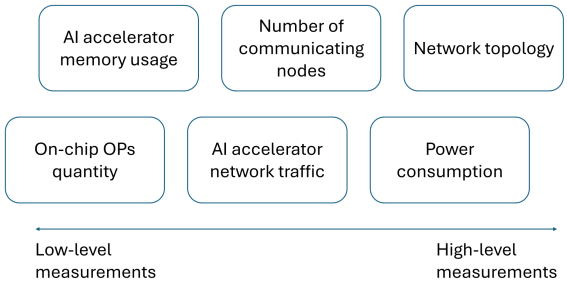

Figure 2: Illustration of possible compute usage metrics for AI workload analysis, organized by granularity level (figure from (Heim et al. 2024)).

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. 2, 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.

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.

Threat models and analysis

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

We identify the following main threads:

- **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, and only the logistics change. This also has several similarities to use-cases in finance, where malicious actors attempt to break up large transactions into smaller ones that are below the required regulatory thresholds, in order to avoid detection (Sanction Scanner 2024).
- **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. Although standard, synchronized training is currently the dominant approach for frontier model training (where decentralized approaches are though to be less learning efficient, very recent work has demonstrated the feasibility of decentralized model training for small LLMs in the range of 100 million - 1 billion parameters (Douillard et al. 2023; Jaghouar, Ong, and Hagemann 2024), achieving similar performance to standard training, while using around 500x less communication (using less frequent gradient updates along the decentralized nodes). Moreover, in the Mixture-of-Experts (MoE) case, another possibility is to train the experts independently and then combine them via some final fine-tuning (Sukhbaatar et al. 2024). It's currently not known if such approaches would scale for frontier AI models, and how efficient that would be. For example, the experiments of (Douillard et al. 2023; Jaghouar, Ong, and Hagemann

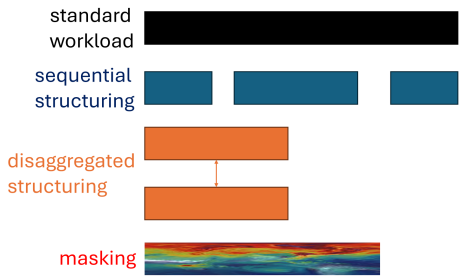

Figure 3: Illustration of the different compute structuring scenarios.

2024) used only 4 decentralized servers with 8 AI accelerators (NVIDIA H100 GPUs) each, which is orders of magnitude lower than frontier 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 interference / 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.

A simple illustration of the above scenarios can be seen in fig. 3:

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*.

Detection Algorithms

Based on our assumptions about AI cloud providers and the threat models identified, we now propose detection strategies for compute structuring. We present our results outlined as algorithms, and discuss the motivations behind them. Then, in the next section, we'll analyze the performance of our method against possible different threat scenarios, under which conditions they're effective, and importantly, we identify the precise conditions when they aren't.

We start with the problem of workload classification, as it serves as a foundation for compute structuring detection. The goal is to classify a given workload into one of the categories outlined in the previous section, given information about the resources allocated, and additional measurements that cloud providers can collect. In the following, we assume that we have access to the measurements listed in Assumption A2.

In general, workload classification is an important problem in technical AI governance (Reuel et al. 2024). Some cloud providers have made available datasets for their workloads (Google 2015; Tang et al. 2022), and researchers have attempted classifying them using various methods (Weiss et al. 2022); however, these datasets are not representative of frontier AI model training, so further research needs to be done.

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

For the context of compute structuring detection, we can relax the general problem of workload classification, 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 inspection. 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 negative (assuming there's training when it's not) is not too high.

With that, our approach for workload classification is outlines in alg. 1:

Alg. 1 works as follows. First, we retrieve the customers identifying information, and the characteristics of the intended workload, in terms of OPs, the throughput (OPs per second, or OP/s) and the node-to-node bandwidth (in bytes per second). We may also request customers to self-declare

the objective of their workloads, for regulation perspectives. 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.

Now we're ready to present our main algorithm for compute structuring detection (alg. 2):

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.

Methodology Efficiency Demonstration / Analysis

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

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

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, no matter what a malicious actor may further do. 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, Ong, and Hagemann 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 to the outside world; this enables us to flag any workload that has a training signature but also exchanges data as suspicious.
- 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 3 orders of magnitude 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 the next section, we try to address this, by describing how our methodology should adapt to potential future advancements.

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.

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.

In order 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. In order 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[1].

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 successfully AI governance methods for ensuring that AI remains safe and beneficial for all.

[1]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.

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 all anonymous reviewers for their thorough and in-depth feedback.

References

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

Anthropic. 2024. The Claude 3 Model Family: Opus, Sonnet, Haiku. *Anthropic Technical Report*.

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. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Commission, E. 2021. Proposal for a Regulation laying down harmonised rules on artificial intelligence (Artificial Intelligence Act). Accessed: 2024-08-06.

Cottier, B.; Rahman, R.; Fattorini, L.; Maslej, N.; and Owen, D. 2024. The rising costs of training frontier AI models. arXiv:2405.21015.

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

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.

Epoch AI. 2023. Key Trends and Figures in Machine Learning. https://epochai.org/trends (accessed 2024-08-16).

Floerecke, S.; Ertl, C.; and Herzfeldt, A. 2023. 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.

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

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

Google. 2015. google/cluster-data. https://github.com/google/cluster-data/. Accessed: 2024-08-14.

Heim, L.; Fist, T.; Egan, J.; Huang, S.; Zekany, S.; Trager, R.; Osborne, M. A.; and Zilberman, N. 2024. Governing Through the Cloud: The Intermediary Role of Compute Providers in AI Regulation. *arXiv preprint arXiv:2403.08501*.

Ho, A.; Besiroglu, T.; Erdil, E.; Owen, D.; Rahman, R.; Guo, Z. C.; Atkinson, D.; Thompson, N.; and Sevilla, J. 2024. Algorithmic progress in language models. arXiv:2403.05812.

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

House, T. W. 2023. Executive Order on the Safe, Secure, and Trustworthy Development and Use of Artificial Intelligence, Executive Order 14110. Accessed: 2024-08-06.

Jaghouar, S.; Ong, J. M.; and Hagemann, J. 2024. OpenDiLoCo: An Open-Source Framework for Globally Distributed Low-Communication Training. *arXiv preprint arXiv:2407.07852*.

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. 2023. Evaluating language-model agents on realistic autonomous tasks. *arXiv preprint arXiv:2312.11671*.

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

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

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

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

Sanction Scanner. 2024. What is the Difference Between Smurfing and Structuring? https://sanctionscanner.com/blog/what-is-the-difference-between-smurfing-andstructuring-594/. Accessed: 2024-08-14.

Sastry, G.; Heim, L.; Belfield, H.; Anderljung, M.; Brundage, M.; Hazell, J.; O'Keefe, C.; Hadfield, G. K.; Ngo, R.; Pilz, K.; et al. 2024. Computing Power and the Governance of Artificial Intelligence. *arXiv preprint arXiv:2402.08797*.

Sevilla, J.; Besiroglu, T.; Dudney, O.; and Ho, A. 2022. The Longest Training Run. Accessed: 2024-08-13.

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

Sukhbaatar, S.; Golovneva, O.; Sharma, V.; Xu, H.; Lin, X. V.; Rozière, B.; Kahn, J.; Li, D.; Yih, W.-t.; Weston, J.; et al. 2024. Branch-Train-MiX: Mixing Expert LLMs into a Mixture-of-Experts LLM. *arXiv preprint arXiv:2403.07816*.

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

Tang, B. J.; Chen, Q.; Weiss, M. L.; Frey, N. C.; McDonald, J.; Bestor, D.; Yee, C.; Arcand, W.; Bergeron, W.; Byun, C.; et al. 2022. The mit supercloud workload classification challenge. In *2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 708–714. IEEE.

Villalobos, P. 2023. Scaling Laws Literature Review. *Published online at epochai. org*.

Weiss, M. L.; McDonald, J.; Bestor, D.; Yee, C.; Edelman, D.; Jones, M.; Prout, A.; Bowne, A.; McEvoy, L.; Gadepally, V.; et al. 2022. An Evaluation of Low Overhead Time Series Preprocessing Techniques for Downstream Machine Learning. In *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, 1–6. IEEE.