How Does LLM Compression Affect Weight Exfiltration Attacks?

Davis Brown
University of Pennsylvania

Mantas Mazeika
Center for AI Safety

Abstract

As frontier AIs become more powerful and costly to develop, adversaries have increasing incentives to mount weight exfiltration attacks. In this work, we explore how advanced compression techniques can significantly heighten this risk, particularly for large language models (LLMs). By tailoring compression specifically for exfiltration rather than inference, we demonstrate that attackers could achieve up to $16\times$ compression with minimal trade-offs, reducing the time it would take for an attacker to illicitly transmit model weights from the defender's server from months to days. To study this risk, we propose a quantitative model for exfiltration success and show how compression tactics can greatly reduce exfiltration time and increase attack success rate. With AIs becoming increasingly valuable to industry and government, our findings underscore the urgent need to develop defenses for weight exfiltration and secure model weights.

1 Introduction

The cost of training frontier AI models has skyrocketed, with each new generation of models requiring exponentially more compute to train (Cottier et al., 2024). Beyond their economic significance, advanced AI systems are increasingly viewed as critical assets in national security, given their rapidly improving capabilities. This dual economic and strategic importance has led to increasing interest from industry and government actors in securing model weights from theft. Of particular concern are weight exfiltration attacks, where data centers hosting model weights are compromised, allowing adversaries to surreptitiously extract valuable models. However, the risk of weight exfiltration attacks remains poorly understood, with much uncertainty surrounding their feasibility.

A common tactic in standard data exfiltration attacks is to compress data before transmission over the network (ATT&CK, 2023), reducing the likelihood of detection. However, this tactic has not been extensively studied in the context of weight exfiltration attacks. While prior work on large language model (LLM) compression has focused on optimizing models for efficient inference—achieving high fidelity with up to $4\times$ compression (Liu et al., 2024)—the requirements for inference differ significantly from those in weight exfiltration scenarios. In particular, existing methods are designed to have an efficient forward pass after the initial compression– we refer to this as decompression costs in Figure 1. These differences suggest that more aggressive compression techniques, potentially tailored for exfiltration, could be feasible, leading to a higher risk of successful attacks.

To study how to better defend language model weights from being stolen from servers, we investigate the impact of LLM compression on the feasibility of weight exfiltration attacks. We begin by proposing a simple quantitative model for exfiltration success, which is influenced by factors such as the compression ratio. Next, we show that compression techniques specifically optimized for exfiltration can significantly reduce exfiltration time and improve the likelihood of a successful attack. By relaxing the typical constraints that prioritize efficient inference, we achieve substantially higher compression rates than existing methods, reaching $16\times$ compression with minimal trade-offs. Concerningly, we also find that larger models may be easier to compress, implying that frontier AI models are at even greater risk of exfiltration.

Our results suggest that LLM compression is a significant factor affecting the risk of weight exfiltration attacks. Attacks that would have taken months before may be feasible in days with advanced compression techniques. This motivates further research on the risk of weight exfiltration and fur-

ther investment in securing model weights. In the appendix, we discuss possible defenses that could help mitigate this threat. We provide our experiment code at [anonymized].

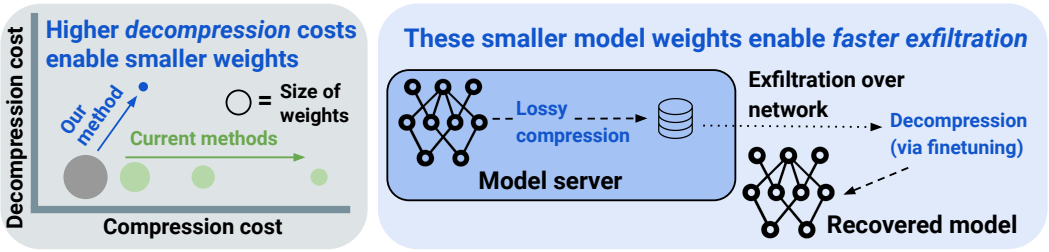


Figure 1: Prior compression methods are optimized for efficient inference, requiring minimal decompression overhead. By relaxing these constraints, we show that significantly smaller model weights can be obtained. This is highly relevant in the setting of weight exfiltration, as it greatly reduces the time to exfiltration and the probability of detection. After being stolen from the server, the model can be decompressed and fine-tuned to recover its performance at a much lower cost than pre-training from scratch.

2 Related Work

Data exfiltration. A long line of work in the security literature has studied data exfiltration threats. Particularly relevant to our settings are advanced persistent threats (APTs) (Alshamrani et al., 2019). These attacks involve targeted infiltration and long dwell time on infiltrated servers to gather and slowly exfiltrate high-value data. In these settings, the amount of data to exfiltrate is a significant consideration, and compression techniques are sometimes used in attacks (ATT&CK, 2023). Several defenses against APTs have been studied in the literature. Moving target defenses periodically modify aspects of the infiltrated system to impede adversaries (Crouse et al., 2015; Sengupta et al., 2020). In the context of LLM exfiltration, Greenblatt (2024) propose physically imposing upload limits on an LLM server to the bare minimum required to meet customer demand.

LLM quantization and compression. A key factor in weight exfiltration attacks is the size of the weights to steal, which can be reduced through compression. Nevo et al. (2024) note that there is much uncertainty around the effective size of model weights, with models commonly served at $2\times$ or even $4\times$ compression. This is enabled by recent work on model quantization for efficient inference, which achieves considerable performance through sophisticated quantization schemes (Zhu et al., 2023; Frantar et al., 2022; Egiazarian et al., 2024; Liu et al., 2024; Tseng et al., 2024). However, research on model quantization has not considered the weight exfiltration setting before. A key contribution of our work is to demonstrate that this setting enables achieving far stronger compression, increasing the risk of weight exfiltration.

3 Weight Exfiltration

Here, we describe the specific threat model that we consider. We base our quantitative model of attacker success on this threat model.

3.1 Threat Model

Attacker capabilities and constraints. We consider a standard Advanced Persistent Threat (APT) with the goal of exfiltrating model weights from an inference server that outputs text, image, video, or audio data to users. The attacker has already compromised the server and gained access to unencrypted model weights but is not an insider. Thus, they cannot upload weights to a USB drive and must extract weights over the server's network. The attacker is restricted in their upload rate because they need to evade detection until the attack is complete. Additionally, the attacker must hide exfiltrated data within legitimate outputs, as the defender uses deep packet inspection to ensure

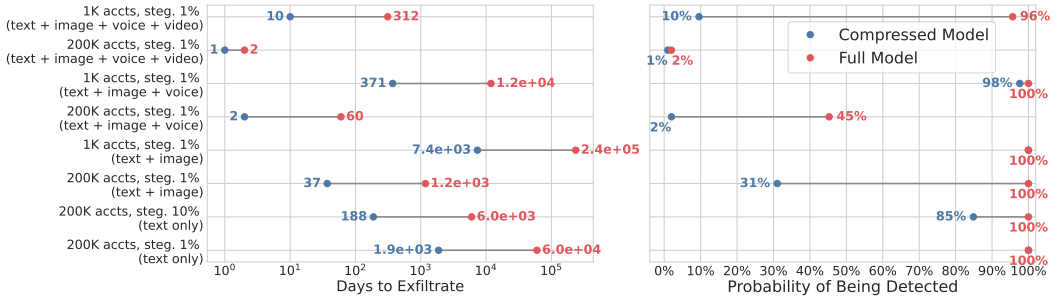


Figure 2: Model compression enables more successful weight exfiltration attacks. We compute the rate of weight exfiltration (left) and the probability of detection (right) for full models and models compressed using our method designed for the weight exfiltration setting. Probability of detection is computed from days to exfiltrate. For more details, see Section 3.2.

that all outgoing traffic conforms to expected formats. Thus, we assume the attacker creates fake accounts and extracts information through responses generated by the model (a form of Sybil attack). We assume that partial exfiltration of the model is far less valuable than full exfiltration. We do not consider hardware security measures, and the specifics of how the server was initially compromised are out of scope.

Defender capabilities and constraints. The defender monitors server activity and network traffic for signs of APT activity, with a fixed probability of detecting malicious behavior per unit time. While detection is not guaranteed, prolonged adversary activity increases the likelihood of detection. The network bandwidth from the server is right-sized to the expected user traffic, and deep packet inspection is employed to ensure that all outgoing traffic matches expected formats. Note that this threat model is only one possible set of assumptions. We discuss limitations of this threat model in Appendix C.

3.2 Quantitative Model of Weight Exfiltration

To better understand the risk of weight exfiltration, we use a simple quantitative model roughly calibrated to the scenario of a weight exfiltration attack on GPT-4 (Achiam et al., 2023). Our model computes the time required for exfiltration and the probability of success given a fixed daily probability of detection. We say attack fails if it is detected before the model is fully exfiltrated, in keeping with our threat model. Our model incorporates various factors and three different usage profiles, allowing us to study a wide variety of attack scenarios. We describe these below.

Factors. Our model starts from an assumption that the exfiltration occurs on an inference server for a public API. The API has many daily users, some number of which are compromised or created by the attacker. The attacker then attempts to exfiltrate the weights over the network before the attack is detected. We describe the factors in our model below.

- **Network egress per user per day** (E): The number of bits sent to each user of the inference API per day.
- **Number of compromised daily users** (N): We assume the attacker mounts a Sybil attack (Douceur, 2002) and can only receive data through steganography on legitimate channels. Note that the attacker may have more accounts than they use each day, swapping out accounts to mimic realistic usage patterns.
- **Steganography ratio** (s): This is the fraction of output bits that the attacker can use for hiding arbitrary data, used for exfiltrating the model weights.
- **Model size** (M): The size of the model to be extracted in bits, before compression is applied.
- **Compression ratio** (c): The multiplicative reduction in model size after applying compression.
- **Probability of detection per day** (p): The probability that the attacker's activity is detected each day. For simplicity, we assume this is a fixed number.

Method	BPP	MMLU	C4 Log Loss
Random Weights	16	25.6	6.07
Original Model (no compression)	16	56.5	3.28
Ours (VQ then train)	0.58	48.4	4.27
	1.15	54.9	3.69
	2.03	55.5	3.33
AQLM	0.58	24.1	8.49
	1.15	23.5	6.39
	2.03	50.7	3.65

Table 1: Comparison of language model compression methods for Qwen2-1.5B (with a budget of 1 billion tokens). MMLU accuracies are percentages. Compared to AQLM, a state-of-the-art compression method, our method obtains much higher performance at given compression levels.

Given these factors, we compute the time until exfiltration in days as

$$T = \frac{M \times c}{E \times N \times s}.$$

The probability of successful exfiltration is computed as $(1-p)^T$.

Usage profiles. We consider four possibilities for the average daily user of an LLM inference API. These correspond to current and anticipated future modes of interaction with LLMs. In all cases, we assume that outputs are converted to standard media formats before being sent over the network to users. These usage profiles are described below.

- **Text Only:** The average user in this scenario generates 10 pages of text per day (500 English words per page). This is in the ballpark of current daily average usage of models such as GPT-4 (Altman, 2024). In reality, some users may generate far more or far less. We assume all users have equal consumption for simplicity.
- **Text + Images:** In addition to the above text usage, users in this scenario generate 1 image per day on average.
- **Text + Images + Voice:** In addition to the usage of the previous scenario, the average user in this scenario generates 30 minutes of voice/audio output per day. This may be used for talking with an LLM as an alternative mode of communication.
- **Text + Images + Voice + Video:** This is a hypothetical future mode of interaction with LLMs involving real-time video output, possibly for interaction with chatbot avatars.

In Appendix D, we derive estimates for the factors in our model and for different usage profiles. In some cases, we consider ranges of values for a given factor.

4 Compression Experiments

Here, we describe our experiments demonstrating that compression techniques can be tailored to the weight exfiltration setting, enabling substantial improvements to the compression ratio. We then evaluate how this affects exfiltration time and success rates under our quantitative model of exfiltration.

4.1 Improved LLM Compression

Method. In our method, we employ a compression technique adapted from additive quantization, a generalization of k-means clustering. Specifically, we begin by splitting the weights of each layer of the large language model (LLM) into consecutive groups by rows and columns. For each group, we

apply k-means clustering to compress the weights into vectors derived from a codebook. Following this initial compression, we subtract the reconstructed weight from the original weight and repeat the compression process with the residual for 1 to 4 iterations, enhancing the compression ratio each time. After compressing the residuals, we apply a one-dimensional k-means clustering to learn a unique scaling factor for each group, further optimizing the storage requirement.

To determine optimal hyperparameters for compression at various bits per parameter values, we perform a hyperparameter sweep using MSE to the original weights as the metric for selection. We then computed the Pareto frontier and used points on this frontier to select hyperparameters for the main experiments. We visualize several such curves in Figure 5.

Decompression involves reconstructing the full parameter vector from its quantized representation. To ensure the functionality of the decompressed model, we perform fine-tuning initially on a pre-training dataset, followed by instruction tuning on a specific dataset if the original model had been instruction-tuned. Unlike previous quantization schemes that require the fine-tuned model to remain quantizable to the same bit depth, our method does not impose such restrictions, allowing more flexibility in restoring the model's performance.

Setup. We compress Qwen2-1.5B (Yang et al., 2024) at three different bits per parameter (BPP) levels, and we use a fixed token budget for decompression. As a baseline, we compare to *Original Model* with no compression and *AQLM* (Egiazarian et al., 2024), a state-of-the-art LLM quantization method for efficient inference. For reference, we also show the performance of randomly-initialized weights after applying the same fine-tuning procedure used in our method (*Random Weights*). We evaluate baselines and our method using three metrics: bits per parameter (BPP), MMLU accuracy and C4 log loss.

For the decompression step of our method, we perform continued pre-training and supervised fine-tuning on RedPajama (Computer, 2023) and Magpie (Xu et al., 2024), respectively. We train for a combined 1 billion tokens on these two datasets, comprising approximately 0.01% of the Qwen2 pre-training data of 7 trillion tokens.

Results. Results are in Table 1. With AQLM, model performance sharply drops off below 2.03 bits per parameter (BPP), while our method has much more graceful degradation. Our method maintains reasonable fidelity at 1.15 BPP, with only a 1.6% reduction in accuracy on MMLU. Thus, our method is able to obtain $16\times$ compression compared to the original model without reducing downstream performance. The Random Weights baseline obtains random chance performance, indicating that the amount of fine-tuning performed during decompression does not introduce new knowledge, and using the compressed weights as a starting point is crucial.

These results demonstrate that when constraints on decompression time are relaxed, models can be compressed to far smaller sizes than was previously known. In Appendix A, we show additional results that suggest larger models may be even easier to compress using our method. In particular, these results suggest that it may be possible to compress Llama-3-70B down to 0.1 BPP using our method, a $160\times$ compression. This is highly concerning for the setting of exfiltration attacks.

4.2 Effect of Compression on Weight Exfiltration

In Figure 2, we show the effect of model compression on time until exfiltration and probability of attack success across various scenarios. Each scenario varies the number of compromised daily users, the steganography ratio, and the usage profile. We explore reasonable ranges for these values, discussed in Section 3.2. We find that model compression significantly affects the feasibility of an attack, reducing exfiltration times by multiple orders of magnitude and meaningfully increasing the probability of attack success.

In Figure 6, we directly visualize the amount of bits extracted as a function of time under our quantitative model. This clarifies that besides model compression, other factors in the quantitative model can have a large impact. Addressing these other factors could be a feasible direction for improving exfiltration defenses. We discuss other possible defenses in Appendix B.

5 Conclusion

In this work, we demonstrated that by relaxing decompression constraints in weight exfiltration scenarios, models can be compressed far more effectively than previously thought. Our results show that such aggressive compression greatly reduces the time required for model exfiltration, making attacks significantly more feasible. These findings emphasize the need for further research on weight exfiltration defenses and further investment into securing model weights as AIs become increasingly critical assets for both industry and national security.

Acknowledgments

We thank the ML Alignment & Theory Scholars program and the Long-Term Future Fund for supporting this work. We also thank the Center for AI Safety for providing compute resources for this project. Finally, we thank Engin Kirda, Aaron Scher, David Archer, and Adam Karvonnen for valuable feedback and guidance.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Adel Alshamrani, Sowmya Myneni, Ankur Chowdhary, and Dijiang Huang. A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Communications Surveys & Tutorials*, 21(2):1851–1877, 2019.

Sam Altman. openai now generates about 100 billion words per day. all people on earth generate about 100 trillion words per day., 2 2024. URL https://twitter.com/sama/status/1756089361609981993. Tweet.

MITRE ATT&CK. T1560.001: Archive via utility, 2023. URL https://attack.mitre.org/techniques/T1560/001/. Accessed: 2024-09-14.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Together Computer. Redpajama: an open dataset for training large language models, 2023. URL https://github.com/togethercomputer/RedPajama-Data.

Ben Cottier, Robi Rahman, Loredana Fattorini, Nestor Maslej, and David Owen. The rising costs of training frontier ai models, 2024.

Michael Crouse, Bryan Prosser, and Errin W Fulp. Probabilistic performance analysis of moving target and deception reconnaissance defenses. In *Proceedings of the Second ACM Workshop on Moving Target Defense*, pp. 21–29, 2015.

John R Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pp. 251–260. Springer, 2002.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Vage Egiazarian, Andrei Panferov, Denis Kuznedelev, Elias Frantar, Artem Babenko, and Dan Alistarh. Extreme compression of large language models via additive quantization. *International Conference on Machine Learning*, 2024. doi: 10.48550/arXiv.2401.06118.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Ryan Greenblatt. Preventing model exfiltration with upload limits, 2 2024. URL https://www.alignmentforum.org/posts/rf66R4YsrCHgWx9RG/preventing-model-exfiltration-with-upload-limits.

Huixian Kang, Hanzhou Wu, and Xinpeng Zhang. Generative text steganography based on lstm network and attention mechanism with keywords. *Electronic Imaging*, 32:1–8, 2020.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. Spinquant–llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406*, 2024.

Vladimir Malinovskii, Denis Mazur, Ivan Ilin, Denis Kuznedelev, Konstantin Burlachenko, Kai Yi, Dan Alistarh, and Peter Richtarik. Pv-tuning: Beyond straight-through estimation for extreme llm compression. *arXiv preprint arXiv:2405.14852*, 2024.

Sella Nevo, Dan Lahav, Ajay Karpur, Yogev Bar-On, Henry Alexander Bradley, and Jeff Alstott. Securing ai model weights. *RAND Research Report*, 2024.

Sailik Sengupta, Ankur Chowdhary, Abdulhakim Sabur, Adel Alshamrani, Dijiang Huang, and Subbarao Kambhampati. A survey of moving target defenses for network security. *IEEE Communications Surveys & Tutorials*, 22(3):1909–1941, 2020.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. Quip#: Even better llm quantization with hadamard incoherence and lattice codebooks. *arXiv preprint arXiv:2402.04396*, 2024.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. *arXiv preprint arXiv:2406.08464*, 2024.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

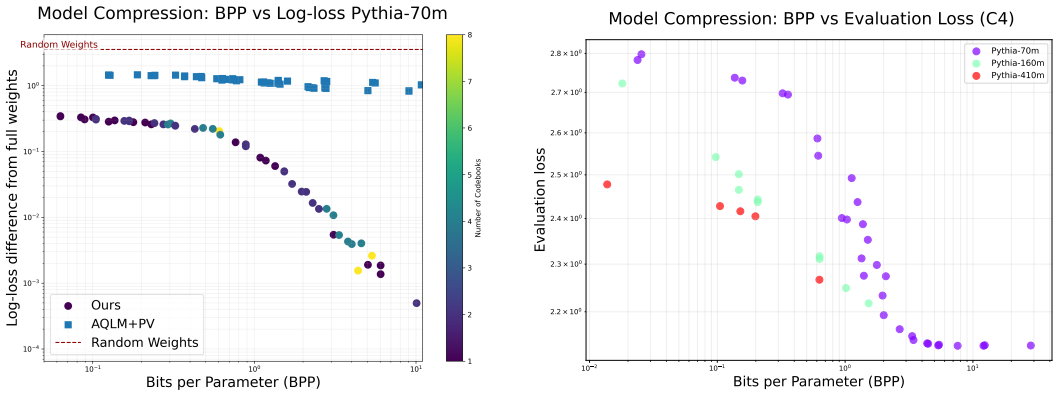
Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *arXiv preprint arXiv:2308.07633*, 2023.

A Rate-distortion in Weight Space

Model weights present a tradeoff between MSE (between the original weights and the compressed weights) and bits-per-parameter. We sample from the Pareto frontier of the compression hyperparameters for our method, showing results in Figure 3, Figure 4, and Figure 5. For these experiments, we use a broader range of models than in the main experiments, including the Pythia suite (Biderman et al., 2023) and Llama 3 models (Dubey et al., 2024). These results reveal several interesting phenomena, which we describe below.

Rate-distortion curves reveal smooth trade-offs. Our main results in Table 1 show that models can be compressed using our method with very little degradation in downstream metrics like log-loss. To obtain a more fine-grained view of how our method performs at different BPP values, we applied it to a range of smaller models in the Pythia suite. On the left side of Figure 3, we show that log-loss for our method starts out much closer to the original model than AQLM enhanced with PV tuning (Malinovskii et al., 2024), a state-of-the-art quantization method. Performance of our method smoothly degrades and changes slope at around 1 BPP.

Larger models are easier to compress. In Figure 4 and Figure 5, we show MSE rate distortion curves for models of various sizes. Interestingly, these reveal that within the Pythia family, the MSE of compressed models gradually improves as model size increases. This is surprising, as MSE is not dependent on the number of parameters. Even more surprising is the striking change in the Pareto frontier between Llama-3-8B and Llama-3-70B, the latter of which maintains very low MSE down to 0.1 BPP with almost no degradation. These results provide evidence that larger models are easier to compress with our method, implying that the relative benefit of compression during exfiltration attacks may increase as models continue scaling up.



(a) Pythia-70m compressed and then trained with a fixed token budget (8k C4 samples).

(b) Three models from the Pythia suite compressed and trained with 150K C4 samples.

Figure 3: Compressing Pythia models to various bits-per-parameter (BPP) and recovering performance with continued training. For Pythia-70m, there is an elbow for both plots at approximately 0.5-0.6 BPP. The trend is less clear for the larger Pythia models. See Figure 4 for BPP vs MSE in weight-space.

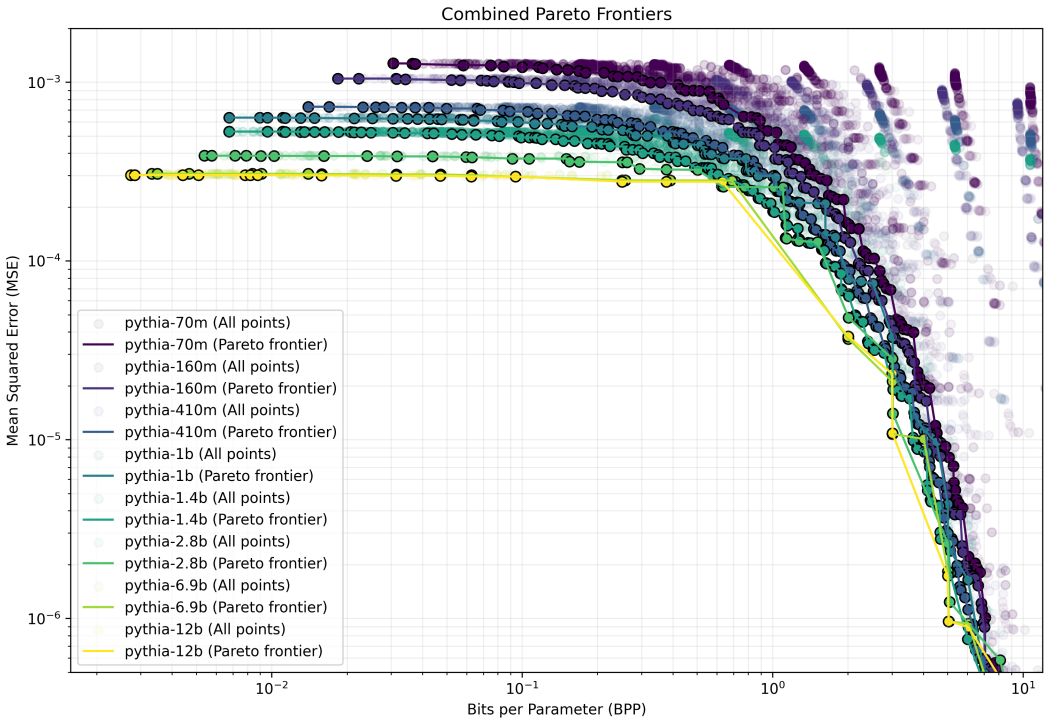
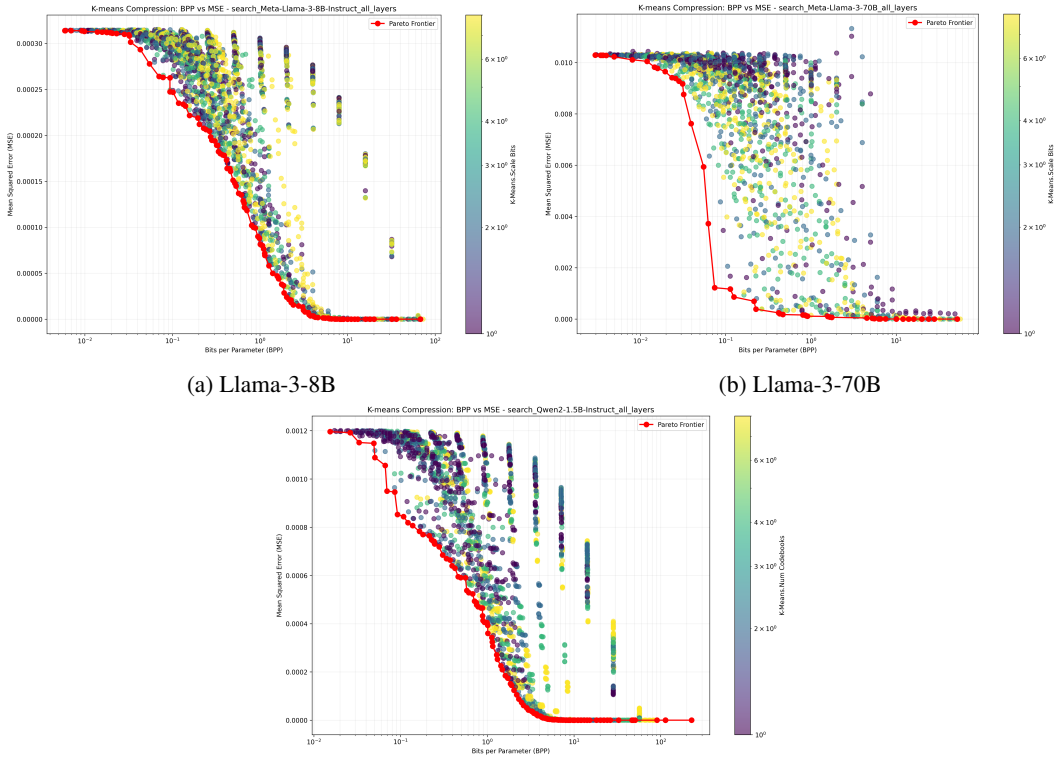


Figure 4: The MSE between the original and compressed weights vs the bits-per-parameter (BPP), along with the Pareto frontier for MSE vs BPP, for each Pythia model. We observe that in weight space, larger models are easier to compress.

B Weight Exfiltration Defenses

In this section, we discuss potential defenses against weight exfiltration attacks at a high level. While many existing defenses designed for Advanced Persistent Threats (APTs) remain highly applicable, weight exfiltration presents unique challenges that could benefit from specialized approaches. Below, we outline several defense strategies that could mitigate the risks of model exfiltration.



(a) Llama-3-8B

(b) Llama-3-70B

(c) Qwen2-1.5B

Figure 5: Comparison of MSE rate-distortion curves on different models. Strikingly, the Llama-3-70B Pareto frontier obtains very high-fidelity compression down to 0.1 bits per parameter (BPP), representing a $160\times$ compression. Note that we do not run full evaluations on Llama 3 models due to resource constraints.

Anomaly detection and monitoring. Traditional APT detection methods typically rely on general anomaly detection, scanning for statistical deviations over large quantities of server activity and network logs. Weight exfiltration, however, occurs in a highly orchestrated environment where server behavior is predictable and narrowly focused. Inference servers handle a single task: processing user inputs and generating outputs in a consistent format. This makes it possible for defenders to apply simple packet inspection techniques to detect anomalies. However, a key challenge lies in the high variance of resource usage in LLM inference, which could allow adversarial activity to be obscured by normal fluctuations in system load.

Hardware-based security measures. Hardware security features embedded in GPUs and CPUs can play a critical role in defending against weight exfiltration. Some NVIDIA GPUs, for instance, implement secure boot mechanisms that use cryptographic verification to ensure that only authorized firmware and software can be executed on the hardware. Although these capabilities are not universally implemented across all inference servers and may have limitations, they offer a promising layer of defense by preventing unauthorized access to model weights at the hardware level.

Upload rate limits. One of the most straightforward and effective defenses is to impose strict upload rate limits on the server, ensuring that data egress is constrained to the minimum required for legitimate user workloads. This was recently proposed by (Greenblatt, 2024), who also suggest compressing model outputs to restrict total data egress. Our quantitative model already accounts for such limits, but if these constraints are absent, exfiltration could become far easier for attackers.

Internal data transfer limits. To further mitigate the risk of weight exfiltration, internal data transfer limits between system components could be enforced. In particular, many LLM-based

systems include additional modules for tasks such as voice or image generation, which can require higher data transfer rates than text-based models. Our results in Section 4.2 suggest that the high network requirements for these additional modalities greatly increase exfiltration success rates. However, this assumes that servers where LLM weights are stored would have high internal bandwidth to servers that integrate different modalities and send outputs to users. By physically decoupling the LLM servers from where final integrated outputs are sent to users, exfiltration data rates could be effectively limited to the core LLM's outputs.

Preventing compression techniques. A potential avenue for future research involves developing defenses that prevent models from being easily compressed for exfiltration. For instance, adding noise to model weights could hinder compression schemes like K-means clustering without significantly impacting the model's performance. This could act as a form of adversarial training specifically targeting the exfiltration method.

C Limitations

In this section, we discuss the limitations of our threat model and analysis. While we present a preliminary investigation of how model compression increases the risk of weight exfiltration, further research is needed to fully understand and model this threat. Some key limitations are described below.

Our threat model does not account for how increased activity by the attacker might increase the rate of detection. We also do not incorporate distributional estimates for the key factors in our quantitative model, although we consider ranges for certain factors. Finally, our model does not factor in the computational costs for the attacker to perform K-means clustering on the infiltrated servers. K-means requires computing and storing cluster centers, which could be detectable by the defender. However, the attacker might mitigate this by running K-means iteratively, uploading chunks of the weights and only storing a small number of cluster centers at a time. By doing so, they could adjust their resource usage to stay below detection thresholds.

D Estimates for Quantitative Model Factors

In Table 2, we show our estimates for the factors in our quantitative model. We calibrate these factors to the setting of an exfiltration attack on a model similar to GPT-4. Below, we describe how we arrive at these estimates.

- **Network egress per user per day:** We estimate this using the different usage profiles. See below for details.
- **Number of compromised daily users:** We start from an assumption of approximately 100 million daily active users for the API and assume that no more than 0.2% of these users could belong to a Sybil attack before suspicious activity would be easily detected. This corresponds to $200,000$ users.
- **Steganography ratio:** For text steganography, values of up to 5 bits per word are achievable (Kang et al., 2020). An average English word taking up 5 ASCII characters requires 40 bits, corresponding to a ratio of roughly 0.1. Steganography may be even more effective for continuous outputs like images, audio, or video. We use a fixed ratio of 0.1 for simplicity.
- **Model size:** We assume a model with 1.8 trillion half-precision floating point parameters, corresponding to 2.88×10^{13} bits.
- **Compression ratio:** We use a compression ratio of $1/32$. This is an extreme level of compression that stresses our method, but we still obtain nontrivial performance at this level.
- **Probability of detection per day:** We assume a value of 0.1%. This value is highly situational, but we assume attackers would adjust their activity to keep it fairly low.

We now describe the estimates of network egress per user per day for each usage profile.

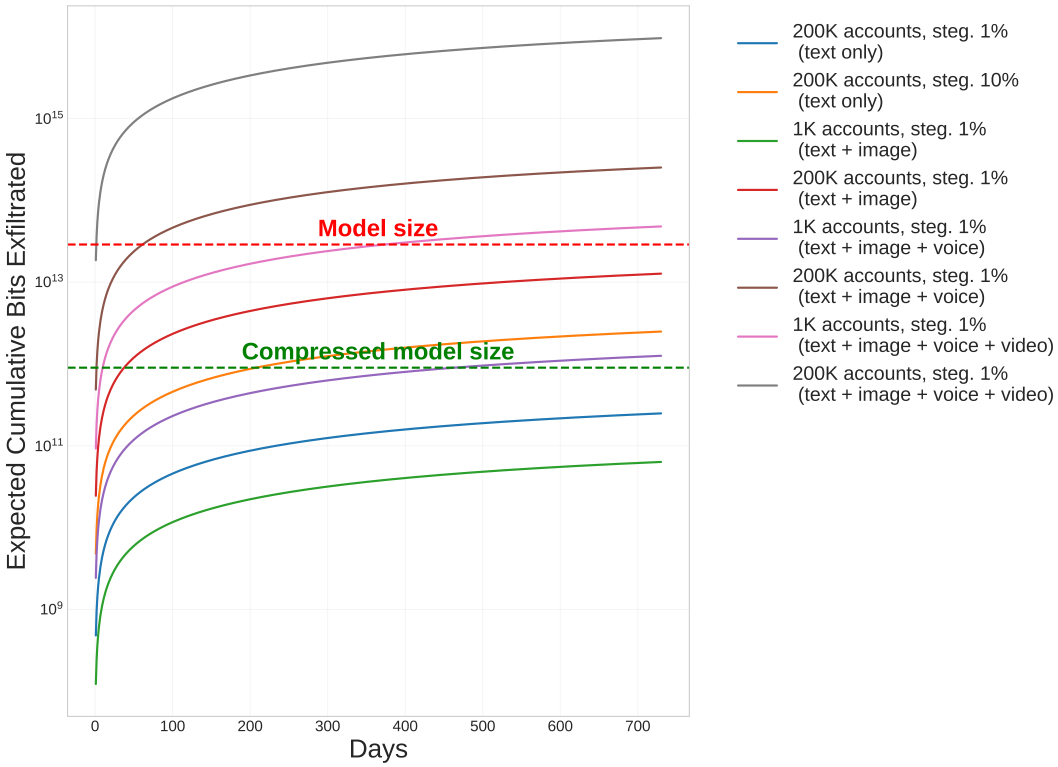


Figure 6: Data exfiltrated over time across different estimates for factors in our quantitative model of exfiltration attacks. The original model size is shown in red, while the model compressed with the assumed compression ratio of $1/32$ is shown in green. While model compression substantially accelerates attacks, other factors have large effects as well, including the usage profile assumed for the average API user. For example, APIs with text, image, voice, and video generation could be susceptible to far quicker exfiltration due to the higher required data rates for serving this content.

Parameter	Assumption	Usage Profile
Network egress per user per day	$240,000$ bits	Text Only
Network egress per user per day	$12,240,000$ bits	Text+Images
Network egress per user per day	$242,640,000$ bits	Text+Images+Voice
Network egress per user per day	$9,242,640,000$ bits	Text+Images+Voice+Video
Number of compromised daily users	$\leq 200,000$	All usage profiles
Steganography ratio	≤ 0.1	All usage profiles
Model size	2.88×10^{13} bits	All usage profiles
Compression ratio	$1/32$	All usage profiles
Probability of detection per day	0.001	All usage profiles

Table 2: Estimates for 1.8 Trillion Parameter Model Exfiltration Analysis

- **Text Only:** We assume each user generates approximately 10 pages of text per day, each taking up 500 English words. Each word is 5 ASCII characters on average with a space, corresponding to $6 \times 500 \times 10 \times 8 = 240,000$ bits.
- **Text + Images:** We assume images are generated at a resolution of 1024×1024 pixels with a 24 bit color depth, corresponding to $1024 \times 1024 \times 24 = 25,165,824$ bits. We assume the image is compressed to 1.5 MB after compression, yielding $1.5 \times 1,000,000 \times 8 = 12,000,000$ bits. This gives a total of $240,000 + 12,000,000 = 12,240,000$ bits.
- **Text + Images + Voice:** We assume 30 minutes of audio per day and that the data is provided as an MP3 with a bitrate of 128 kilobits per second. This gives 128×60 seconds/minute $\times$ 30 minutes $= 230,400$ kilobits, or $230,400,000$ bits. This gives a total of $230,400,000 + 12,240,000 = 242,640,000$ bits.
- **Text + Images + Voice + Video:** We assume 30 minutes of video per day and that the data is provided at 1920×1080 resolution at 30 frames per second with an average bitrate of 5 megabits per second. This gives 5×60 seconds/minute $\times$ 30 minutes $= 9,000$ megabits, or $9,000,000,000$ bits. This gives a total of $9,000,000,000 + 242,640,000 = 9,224,640,000$ bits.