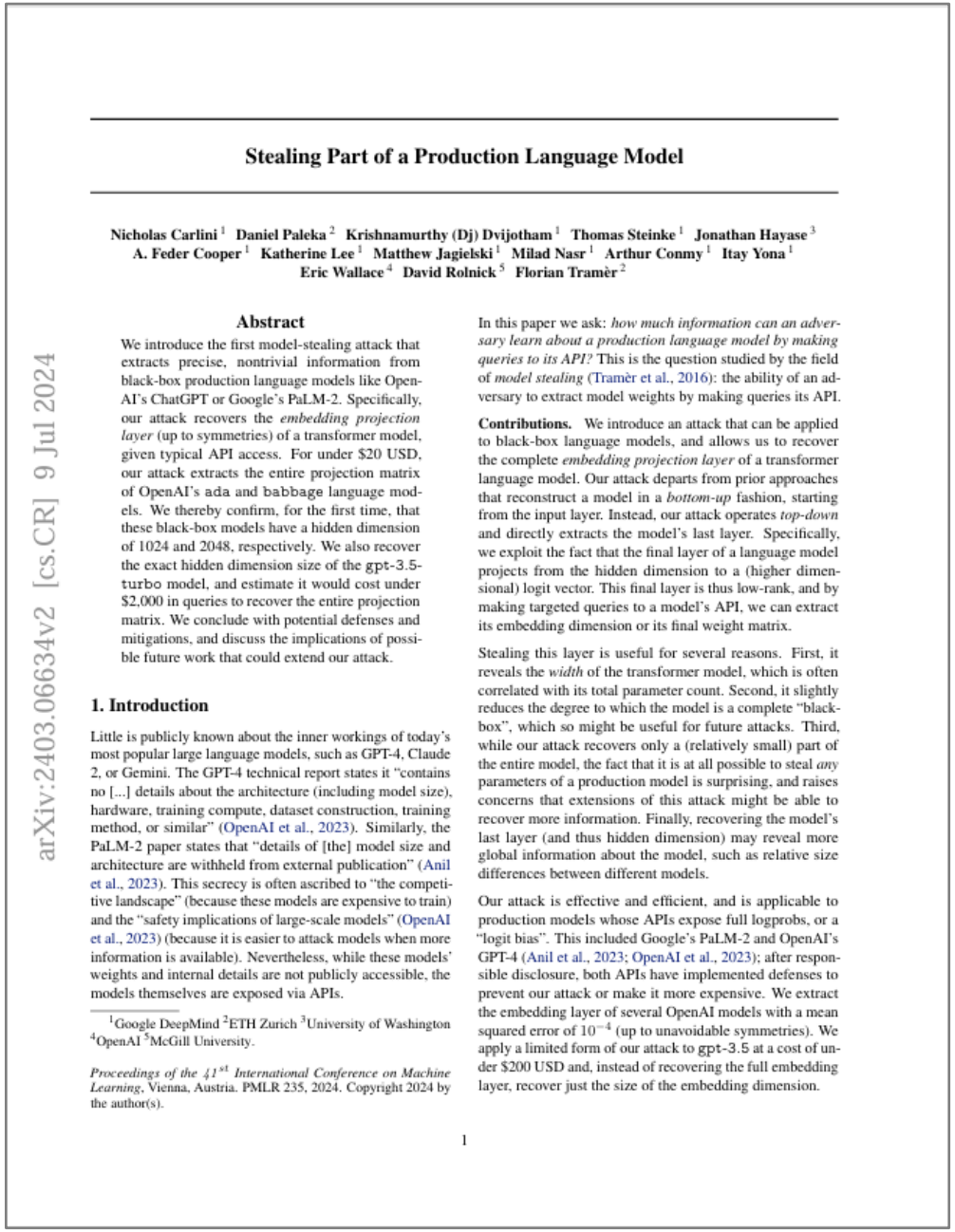
# SDML
# ML Paper Review

August 2024

# Stealing Part of a Production Language Model

Nicholas Carlini et al., Deep Mind+

https://arxiv.org/abs/2403.06634



Stealing Part of a Production Language Model

Nicholas Carlini [1] Daniel Paleka [2] Krishnamurthy (Dj) Dvijotham [1] Thomas Steinke [1] Jonathan Hayase [3]
A. Feder Cooper [1] Katherine Lee [1] Matthew Jagielski [1] Milad Nasr [1] Arthur Conmy [1] Itay Yona [1]
Eric Wallace [4] David Rolnick [5] Florian Tramèr [2]



**Abstract**

We introduce the first model-stealing attack that extracts precise, nontrivial information from black-box production language models like OpenAI's ChatGPT or Google's PaLM-2. Specifically, our attack recovers the *embedding projection layer* (up to symmetries) of a transformer model, given typical API access. For under $20 USD, our attack extracts the entire projection matrix of OpenAI's `ada` and `babbage` language models. We thereby confirm, for the first time, that these black-box models have a hidden dimension of 1024 and 2048, respectively. We also recover the exact hidden dimension size of the `gpt-3.5-turbo` model, and estimate it would cost under $2,000 in queries to recover the entire projection matrix. We conclude with potential defenses and mitigations, and discuss the implications of possible future work that could extend our attack.

## 1. Introduction

Little is publicly known about the inner workings of today's most popular large language models, such as GPT-4, Claude 2, or Gemini. The GPT-4 technical report states it "contains no [...] details about the architecture (including model size), hardware, training compute, dataset construction, training method, or similar" (OpenAI et al., 2023). Similarly, the PaLM-2 paper states that "details of [the] model size and architecture are withheld from external publication" (Anil et al., 2023). This secrecy is often ascribed to "the competitive landscape" (because these models are expensive to train) and the "safety implications of large-scale models" (OpenAI et al., 2023) (because it is easier to attack models when more information is available). Nevertheless, while these models' weights and internal details are not publicly accessible, the models themselves are exposed via APIs.

In this paper we ask: *how much information can an adversary learn about a production language model by making queries to its API?* This is the question studied by the field of *model stealing* (Tramèr et al., 2016): the ability of an adversary to extract model weights by making queries its API.

**Contributions.** We introduce an attack that can be applied to black-box language models, and allows us to recover the complete *embedding projection layer* of a transformer language model. Our attack departs from prior approaches that reconstruct a model in a *bottom-up* fashion, starting from the input layer. Instead, our attack operates *top-down* and directly extracts the model's last layer. Specifically, we exploit the fact that the final layer of a language model projects from the hidden dimension to a (higher dimensional) logit vector. This final layer is thus low-rank, and by making targeted queries to a model's API, we can extract its embedding dimension or its final weight matrix.

Stealing this layer is useful for several reasons. First, it reveals the *width* of the transformer model, which is often correlated with its total parameter count. Second, it slightly reduces the degree to which the model is a complete "black-box", which so might be useful for future attacks. Third, while our attack recovers only a (relatively small) part of the entire model, the fact that it is at all possible to steal *any* parameters of a production model is surprising, and raises concerns that extensions of this attack might be able to recover more information. Finally, recovering the model's last layer (and thus hidden dimension) may reveal more global information about the model, such as relative size differences between different models.

Our attack is effective and efficient, and is applicable to production models whose APIs expose full logprobs, or a "logit bias". This included Google's PaLM-2 and OpenAI's GPT-4 (Anil et al., 2023; OpenAI et al., 2023); after responsible disclosure, both APIs have implemented defenses to prevent our attack or make it more expensive. We extract the embedding layer of several OpenAI models with a mean squared error of $10^{-4}$ (up to unavoidable symmetries). We apply a limited form of our attack to `gpt-3.5` at a cost of under $200 USD and, instead of recovering the full embedding layer, recover just the size of the embedding dimension.

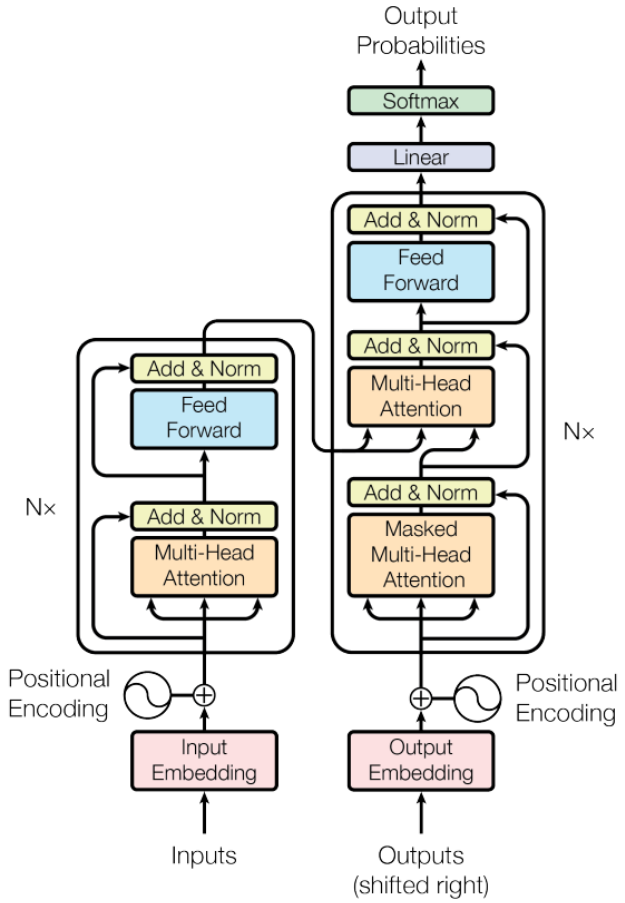
[1]Google DeepMind [2]ETH Zurich [3]University of Washington [4]OpenAI [5]McGill University.

*Proceedings of the 41st International Conference on Machine Learning*, Vienna, Austria. PMLR 235, 2024. 

# Stealing part of a LLM overview

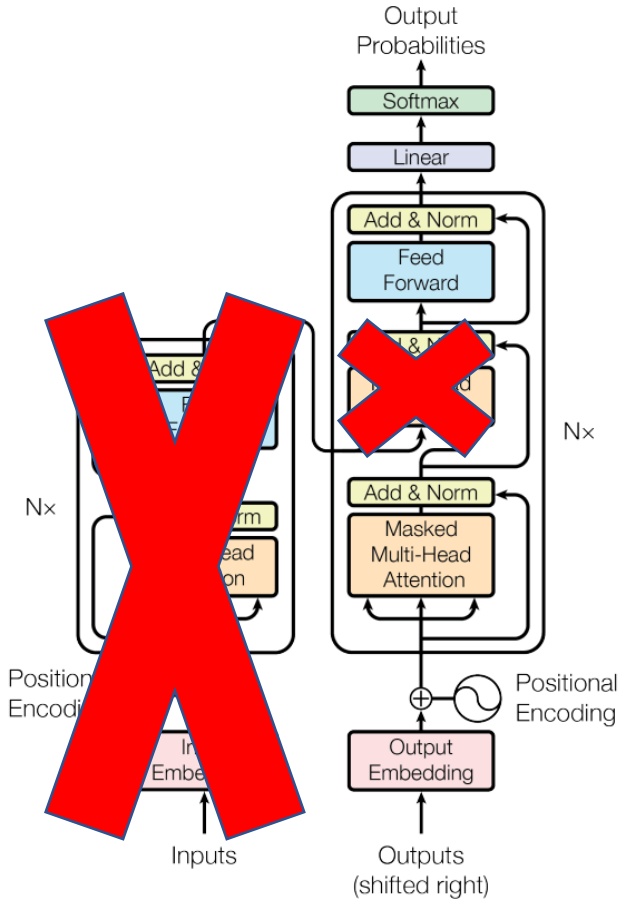
- Common decoder-only LLMs like ChatGPT work on numeric vectors
  - First, they embed text tokens into dense vectors of the hidden model dimension
  - These vectors are modified by the transformer layers
  - Finally, the vectors are projected back into the vocabulary dimension and passed through a softmax to generate output token probabilities
- Developed a set of techniques which can extract the hidden model dimension and the full token projection matrix from black box models
- Techniques use the token log probabilities and the logit bias feature
- Also include a section on mitigation strategies

# Original Transformer

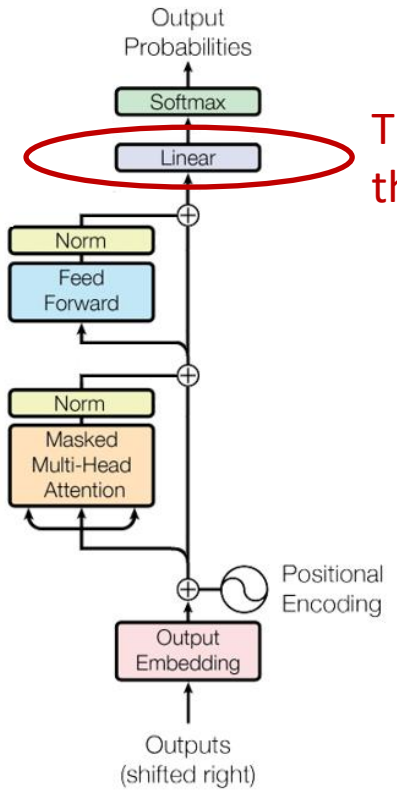



"*Attention Is All You Need*," Vaswani et al., https://arxiv.org/abs/1706.03762

# Decoder-only Transformer

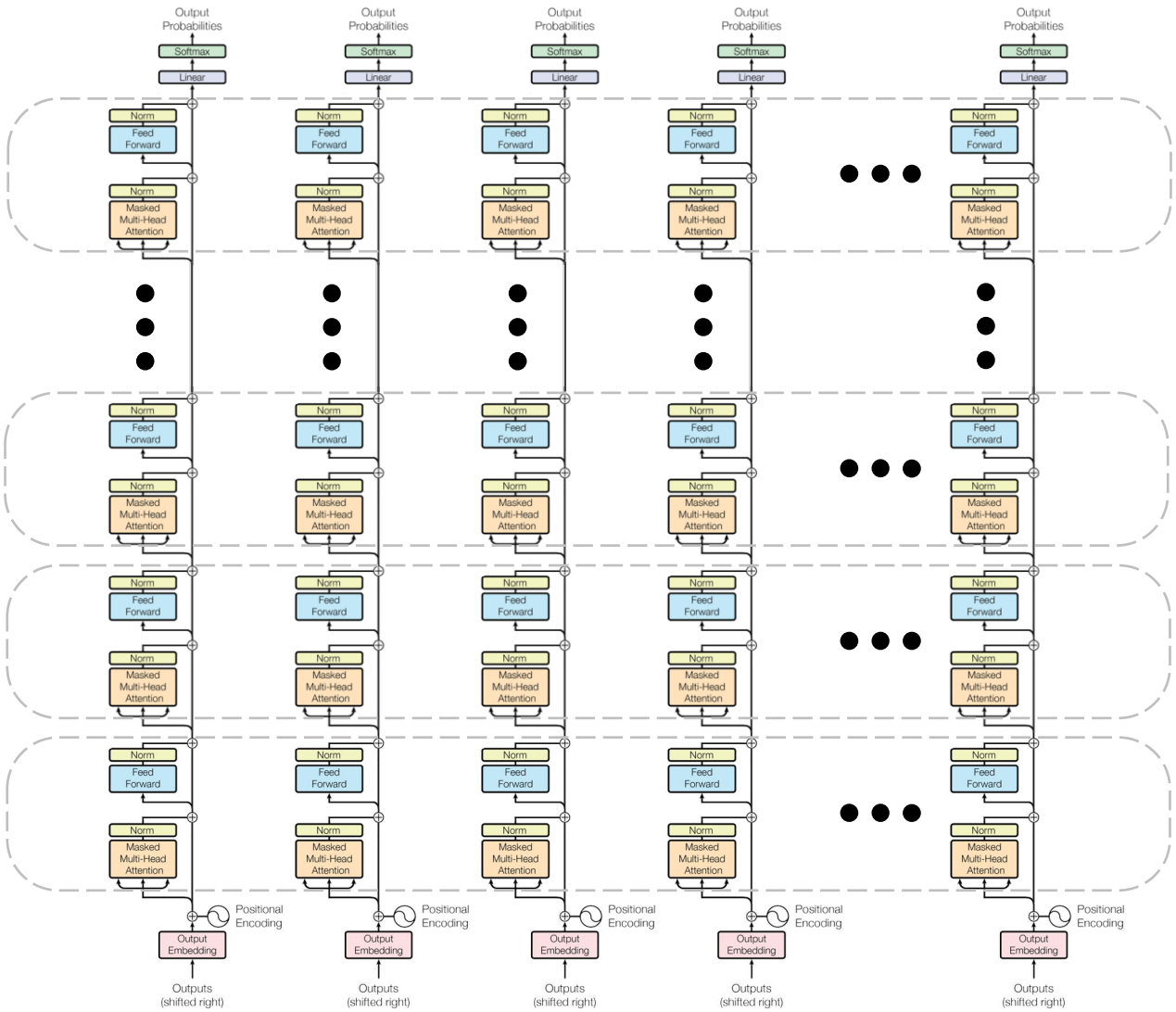


# Decoder-only Transformer, Straight Through

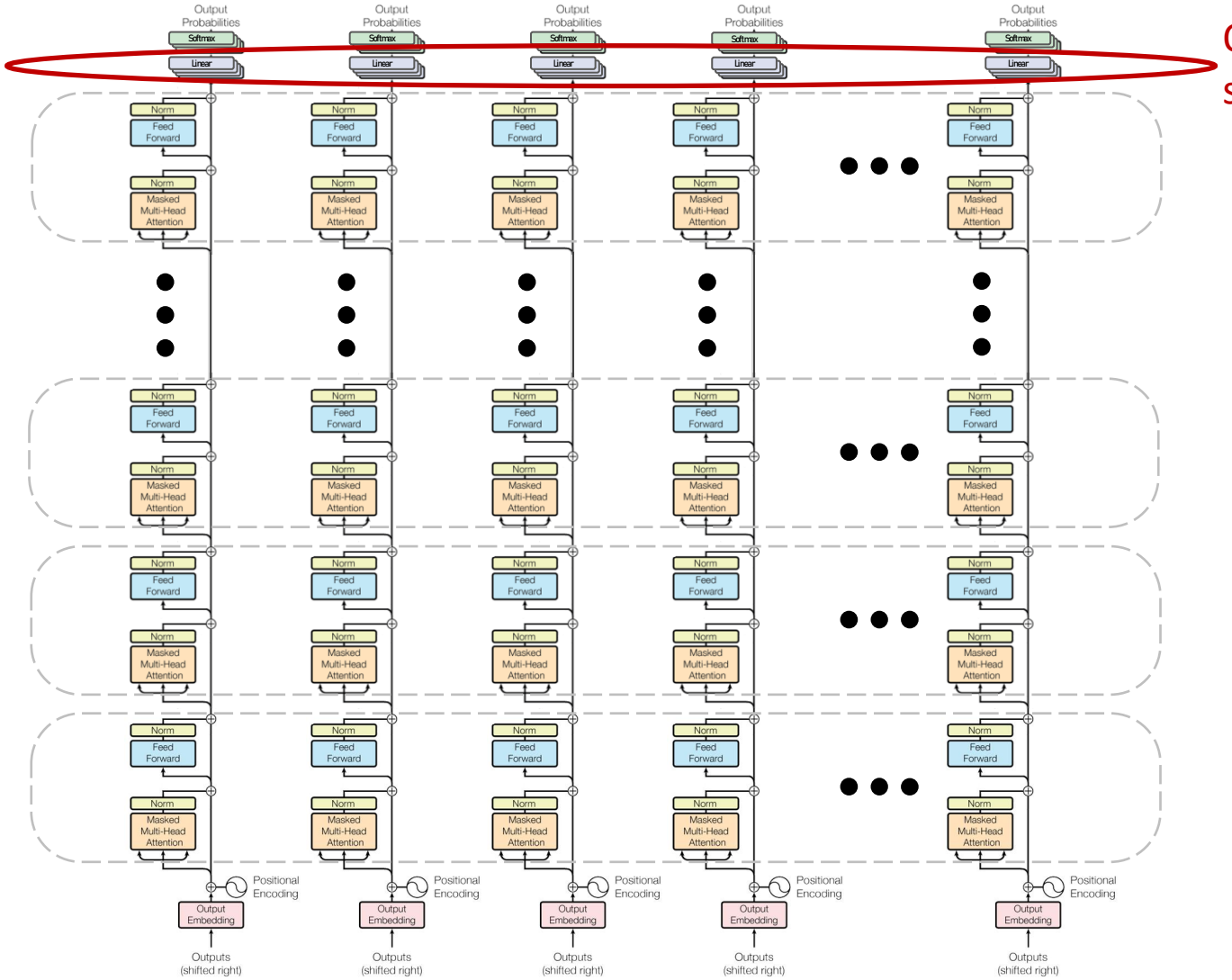



This is where they extracted the size and the weights

# Decoder, Multiple Tokens and Multiple Layers



# Decoder-only LLM




Output projection matrix shown with multiple tokens

# Extracting hidden model dimensionality [1]

- This is an easy to understand attack to start with, though not realistic
- Assume you have access to all of the logits (prior to the softmax)
- For vocabulary size V and model dim D, with D < V:
  - The projection matrix is a V x D matrix, and it can be at most rank D
- Collect N full logit predictions, where N > D, arranging them as columns of a new matrix Q of logits
- This logit matrix is V x N where V > D and N > D, but can be at most rank D because its columns came from the rank D projection matrix
- Use SVD and plot the singular values to see the rank

# Extracting hidden model dimensionality [2]

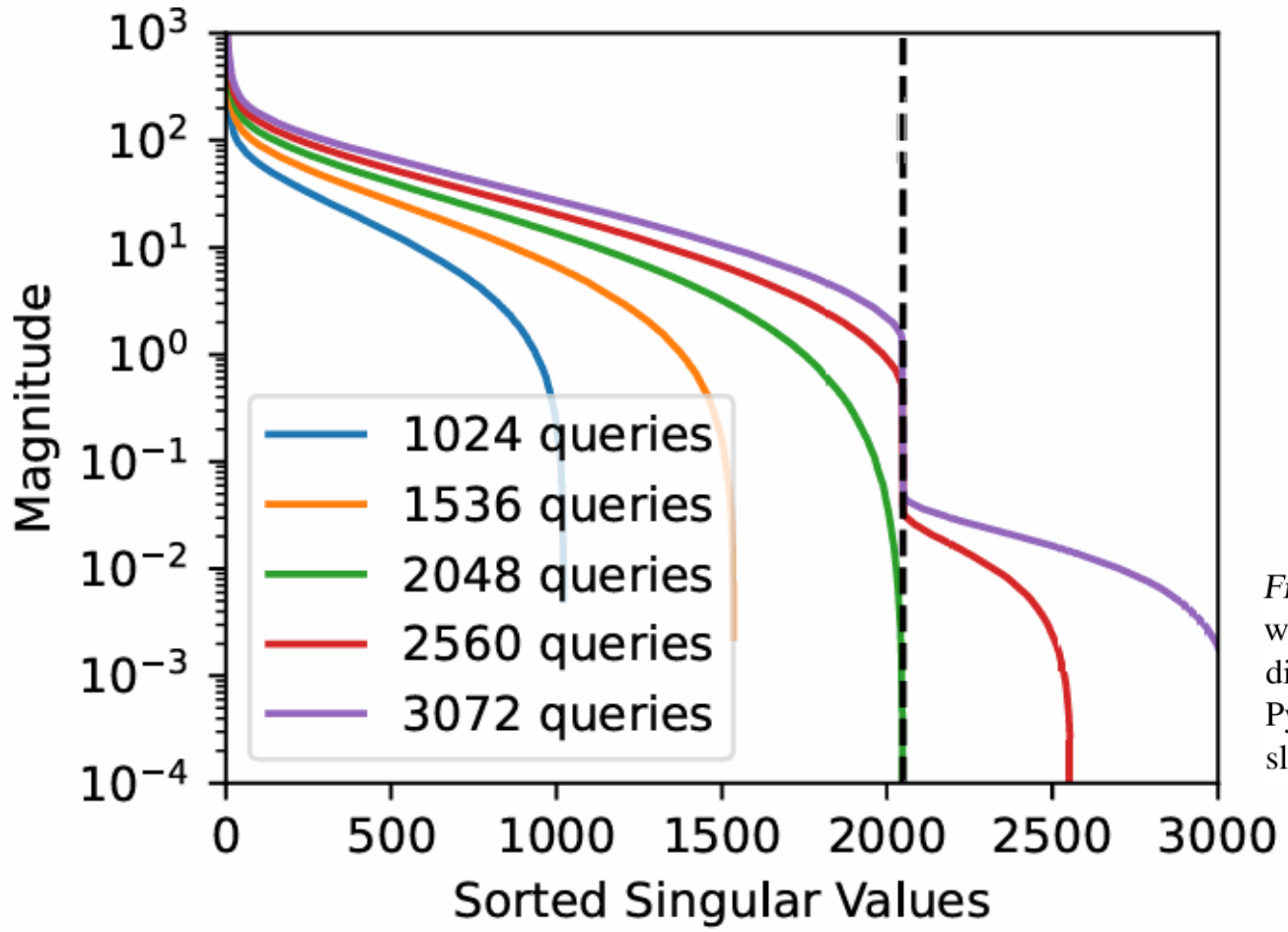



*Figure 1.* SVD can recover the hidden dimensionality of a model when the final output layer dimension is greater than the hidden dimension. Here we extract the hidden dimension (2048) of the Pythia 1.4B model. We can precisely identify the size by obtaining slightly over 2048 full logit vectors.

# Extracting hidden model dimensionality [3]

- Due to floating point limitations, there will be more than D singular values, but the fake ones will be much smaller

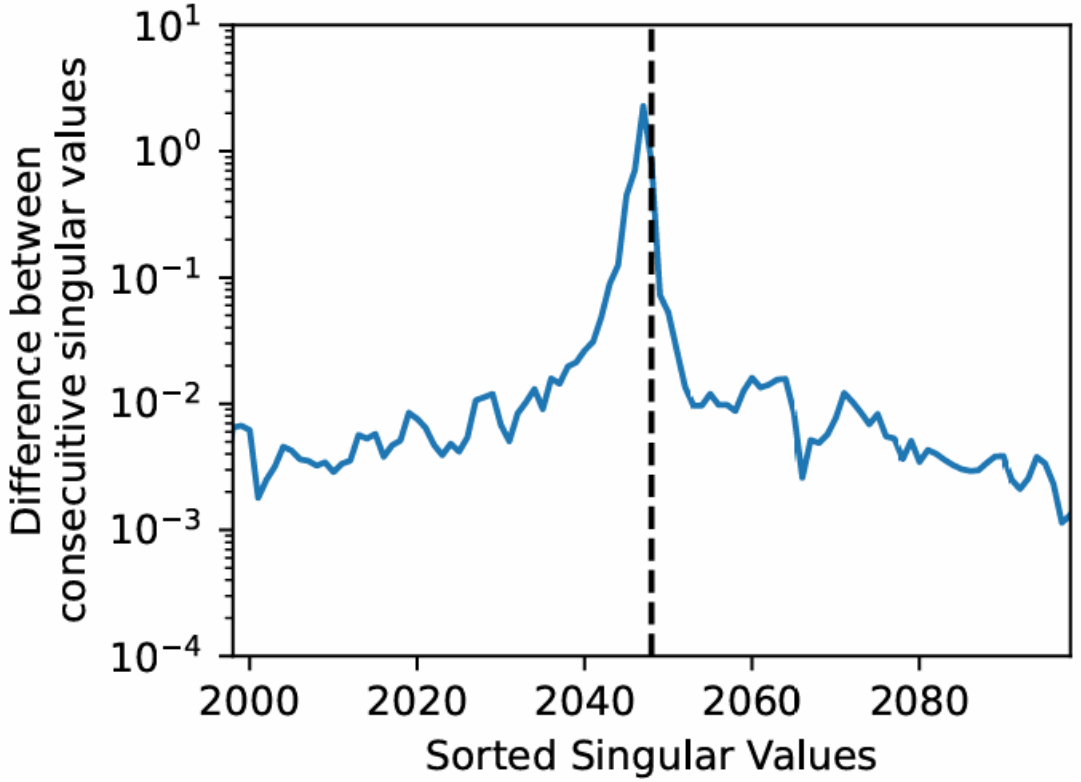



*Figure 2.* Our extraction attack recovers the hidden dimension by identifying a sharp drop in singular values, visualized as a spike in the difference between consecutive singular values. On Pythia-1.4B, a 2048 dimensional model, the spike occurs at 2047 values.

# Full output layer extraction from logits

- Not only does the SVD decomposition allow us to see how many singular values there are (and thereby know the rank), but also if our logit matrix $Q = U \cdot \Sigma \cdot V^T$, then the U matrix directly represents (a linear transformation of) the final layer

*Table 2.* Our attack succeeds across a range of open-source models, at both stealing the model size, and also at reconstructing the output projection matrix (up to invariances; we show the root MSE).

| Model | Hidden Dim | Stolen Size | **W** RMS |
|---|---|---|---|
| GPT-2 Small (fp32) | 768 | $757 \pm 1$ | $4 \cdot 10^{-4}$ |
| GPT-2 XL (fp32) | 1600 | $1599 \pm 1$ | $6 \cdot 10^{-4}$ |
| Pythia-1.4 (fp16) | 2048 | $2047 \pm 1$ | $3 \cdot 10^{-5}$ |
| Pythia-6.9 (fp16) | 4096 | $4096 \pm 1$ | $4 \cdot 10^{-5}$ |
| LLaMA 7B (fp16) | 4096 | $4096 \pm 2$ | $8 \cdot 10^{-5}$ |
| LLaMA 65B (fp16) | 8192 | $8192 \pm 2$ | $5 \cdot 10^{-5}$ |

# Realistic attacks using logit bias feature [1]

- Real APIs offer only the top *K* probabilities, but also offer logit bias
  - Logit bias allows user to specify value to be added to logit for certain tokens
  - If generating a story about farm animals, you could specify -100 to be added to the logit for “cow” to avoid any cows in your story
- If you had the top 5 logits, you could cycle through adding a huge logit bias to different tokens until you recovered all logits
- Production APIs actually return the log probabilities.
  - With bias *B*, the logprob for the *i*th token is:

$$y_i^B = z_i + B - \log\left(\sum_{j \neq i} \exp(z_j) + \exp(z_i + B)\right)$$

  - The extra term goes away for the logprob difference from a reference token:

$$y_R^B - y_i^B - B = z_R - z_i$$

# Realistic attacks using logit bias feature [2]

- If you had the top-5 logprobs and the logit bias feature, you could cycle through adding a huge logit bias to 4 different tokens until you recovered all logits
  - The authors further analyze and explore options related to how many tokens and how many total queries to extract all of the logits
- There is even an algorithm to recover all logits if the API only provides the top-1 logprob and limits the logit bias to a small range {-1, 0}
  - The trick is to query every token twice, once with logit bias 0, and once with -1
  - This requires many queries, possibly more than 2V due to numerical stability issues, but would eventually work

# Realistic attacks using logit bias feature [3]

- Finally it is possible to extract logits using logit bias without logprobs
  - This attack requires a binary search over logit bias values
  - If you want to know the logits within a small value ε
  - Then the cost per token is at most $\log_2( B / \varepsilon )$
- In the appendix, they detail more complex algorithms that track multiple tokens at a time and solve systems of equations, further reducing the number of queries needed

# Efficiency

- The attack algorithms described are reasonably efficient

*Table 4.* Average error at recovering the logit vector for each of the logit-estimation attacks we develop. Our highest precision, and most efficient attack, recovers logits nearly perfectly; other attacks approach this level of precision but at a higher query cost.

| Attack | Logprobs | Bits of precision | Queries per logit |
|---|---|---|---|
| logprob-4 (§5.3) | top-5 | 23.0 | 0.25 |
| logprob-5 (§E) | top-5 | 11.5 | 0.64 |
| logprob-1 (§5.4) | top-1 | 6.1 | 1.0 |
| binary search (§F.1) | ✗ | 7.2 | 10.0 |
| hyperrectangle (§F.2) | ✗ | 15.7 | 5.4 |
| one-of-n (§F.3) | ✗ | 18.0 | 3.7 |

# Results

- The attacks were successful on production models

*Table 3.* Attack success rate on five different black-box models

| | Dimension Extraction | | | Weight Matrix Extraction | | |
|---|---|---|---|---|---|---|
| Model | Size | # Queries | Cost (USD) | RMS | # Queries | Cost (USD) |
| OpenAI `ada` | 1024 ✓ | $< 2 \cdot 10^6$ | $1 | $5 \cdot 10^{-4}$ | $< 2 \cdot 10^7$ | $4 |
| OpenAI `babbage` | 2048 ✓ | $< 4 \cdot 10^6$ | $2 | $7 \cdot 10^{-4}$ | $< 4 \cdot 10^7$ | $12 |
| OpenAI `babbage-002` | 1536 ✓ | $< 4 \cdot 10^6$ | $2 | † | $< 4 \cdot 10^6$ †+ | $12 |
| OpenAI `gpt-3.5-turbo-instruct` | * ✓ | $< 4 \cdot 10^7$ | $200 | † | $< 4 \cdot 10^8$ †+ | $2,000†+ |
| OpenAI `gpt-3.5-turbo-1106` | * ✓ | $< 4 \cdot 10^7$ | $800 | † | $< 4 \cdot 10^8$ †+ | $8,000†+ |

✓ Extracted attack size was exactly correct; confirmed in discussion with OpenAI.
* As part of our responsible disclosure, OpenAI has asked that we do not publish this number.
† Attack not implemented to preserve security of the weights.
+ Estimated cost of attack given the size of the model and estimated scaling ratio.

# Prevention

The attacks described in this paper can be completely prevented using one of the following measures:

- Remove the logit bias feature
- Replace logit bias with a list of blocked tokens
- Change the architecture of the model
  - E.g., split the final layer into two layers, which would be costly
- Post-hoc alteration of the architecture
  - After training is complete, add orthogonal noise that looks realistic

# Mitigation

The attacks described in this paper can be made more costly or less accurate using one of the following measures:

- Don’t allow logit bias and logprobs at the same time
- Add noise
  - Note that logit noise potentially makes the LLM less useful
- Limit the number of logit bias queries for each prompt
  - Serious issues with cost of saving state and privacy risks
- Find a way to detect malicious queries
  - Current defenses insufficient

# Stealing part of a LLM conclusion

- The paper developed a range of techniques able to extract the final layer of black box production LLMs
- Although this work does not appear to be extensible beyond the final layer, and learning just the final layer doesn't seem to have any practical value, the paper shows how architectural decisions and seemingly-innocuous API features such logit bias and logprobs do impact the security of the product
- The authors worked responsibly with vendors, obtaining permission and allowing them time to implement mitigations

# References


- Stealing Machine Learning Models via Prediction APIs
  Florian Tramèr et al. (2016)
  https://arxiv.org/abs/1609.02943