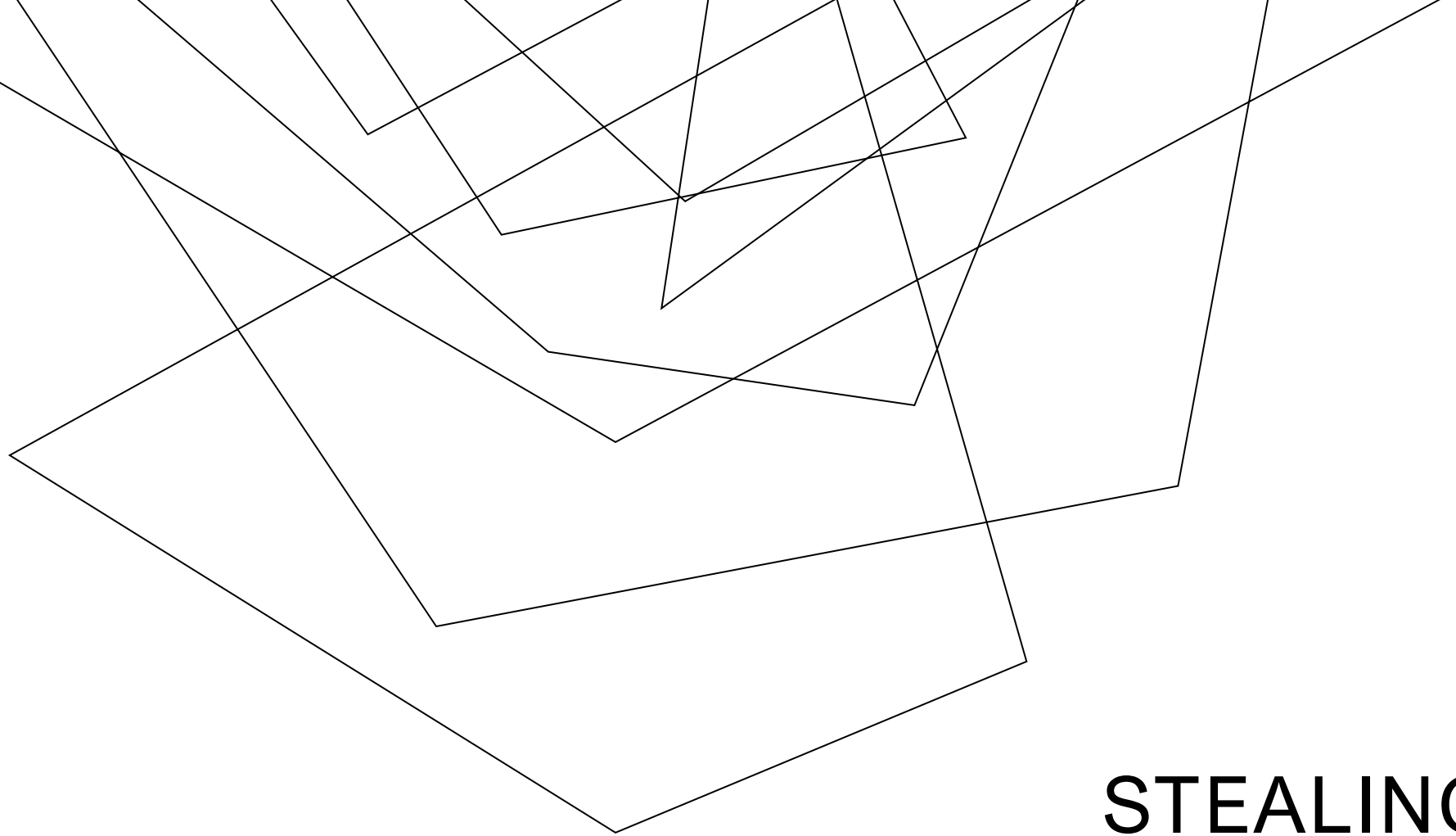


STEALING PART OF A PRODUCTION LANGUAGE MODEL

Presented by: John Atsalakis

WHAT IS THE PAPER ABOUT?

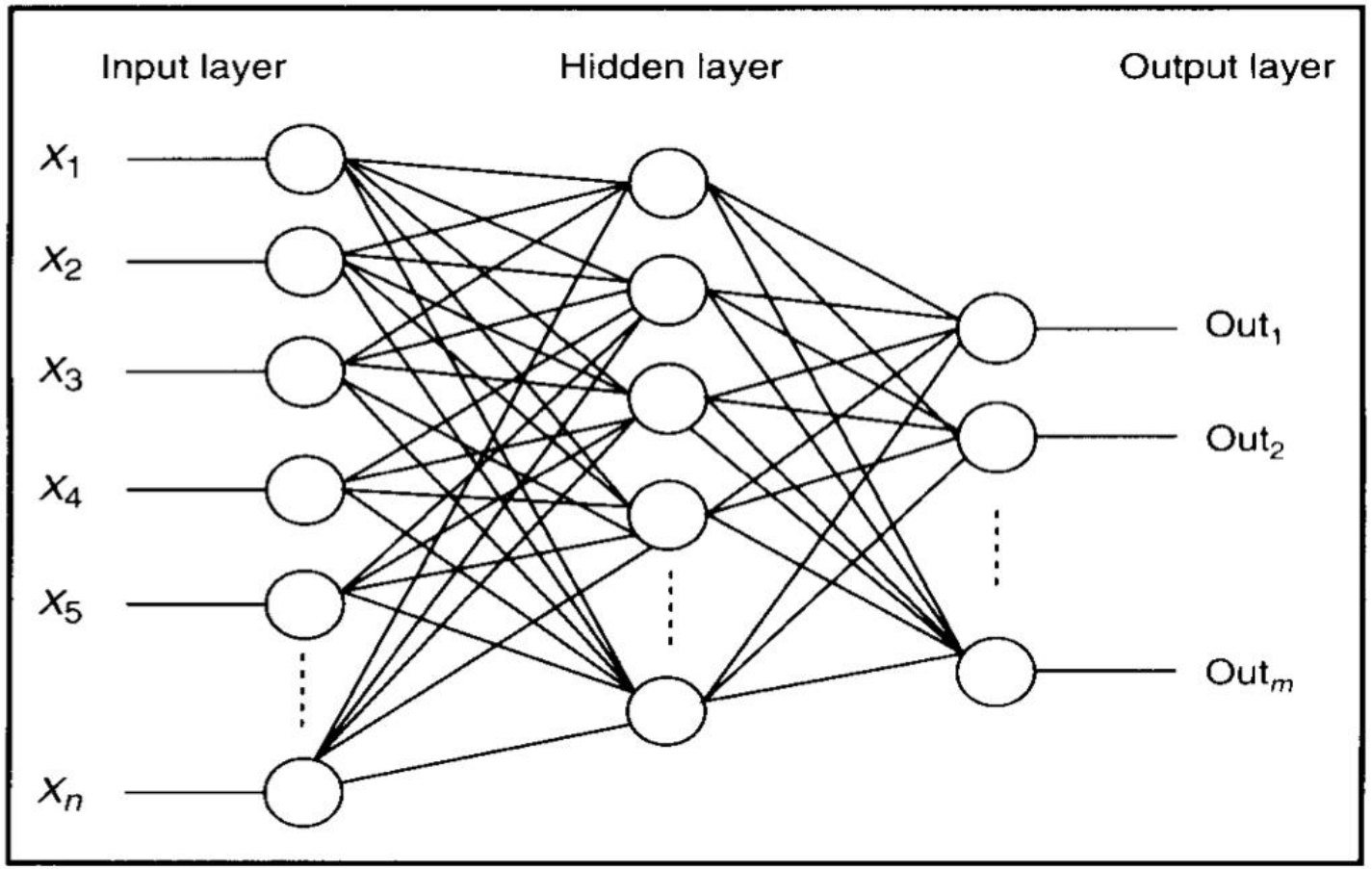
Q: How much information can an adversary learn about a production language model by making queries to its API.

A: This attack recovers the embedding projection layer of a transformer model.

Cost:

- 20$ for the **ada** and **babbage** models.
- 2.000$ (estimated) for the **gpt-3.5-turbo** model.

INTRO TO A NEURAL NETWORK



- Connections: weights + bias
- Neurons: Value

IN MATHEMATICAL TERMS

$y = \text{ReLU}(W_{11} * x_1 + W_{12} * x_2 + \ldots + w_{1n} * x_n + b)$

$$\underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_l \end{bmatrix}}_{\text{output}} = \text{softmax}\left(\underbrace{\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1h} \\ w_{21} & w_{22} & \cdots & w_{2h} \\ \vdots & \vdots & \ddots & \vdots \\ w_{l1} & w_{l2} & \cdots & w_{lh} \end{bmatrix}}_{\text{weights}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_h \end{bmatrix}}_{\text{input}} + \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_l \end{bmatrix}}_{\text{bias}}\right)$$

- **l =** output size
- **h =** hidden size
- **Target:** The weights matrix and the hidden dimension h

WHY IS THIS USEFUL?

1. Reveals the width of the transformer model
2. Reduces the degree to which the model is a black box
3. Raises concerns for more dangerous extensions of this attack
4. May reveal more global information about the model

OTHER APPROACHES

- **Accuracy attacks:** Aim to match the performance
- **Fidelity attacks:** Aim to be functionally equivalent
  - Based on ReLU activations:
    - 2019: If an attacker can compute gradients of a two-layer ReLU model they can steal a nearly bit-for-bit equivalent model
    - 2020: With query access to model outputs an adversary can compute gradients

WHAT IS THE PROBLEM?

The production language models:

1. Accept tokens as inputs
2. Use activations other than ReLUs
3. Contain architectural components that current attacks cannot handle
4. Much larger than the test models
5. Expose only limited outputs

WHAT IS THE DIFFERENCE

- Prior attacks operate bottom-up
  - ➢ Starting from the input layer
- Our attack operates top-down
  - ➢ Starting from the output layer

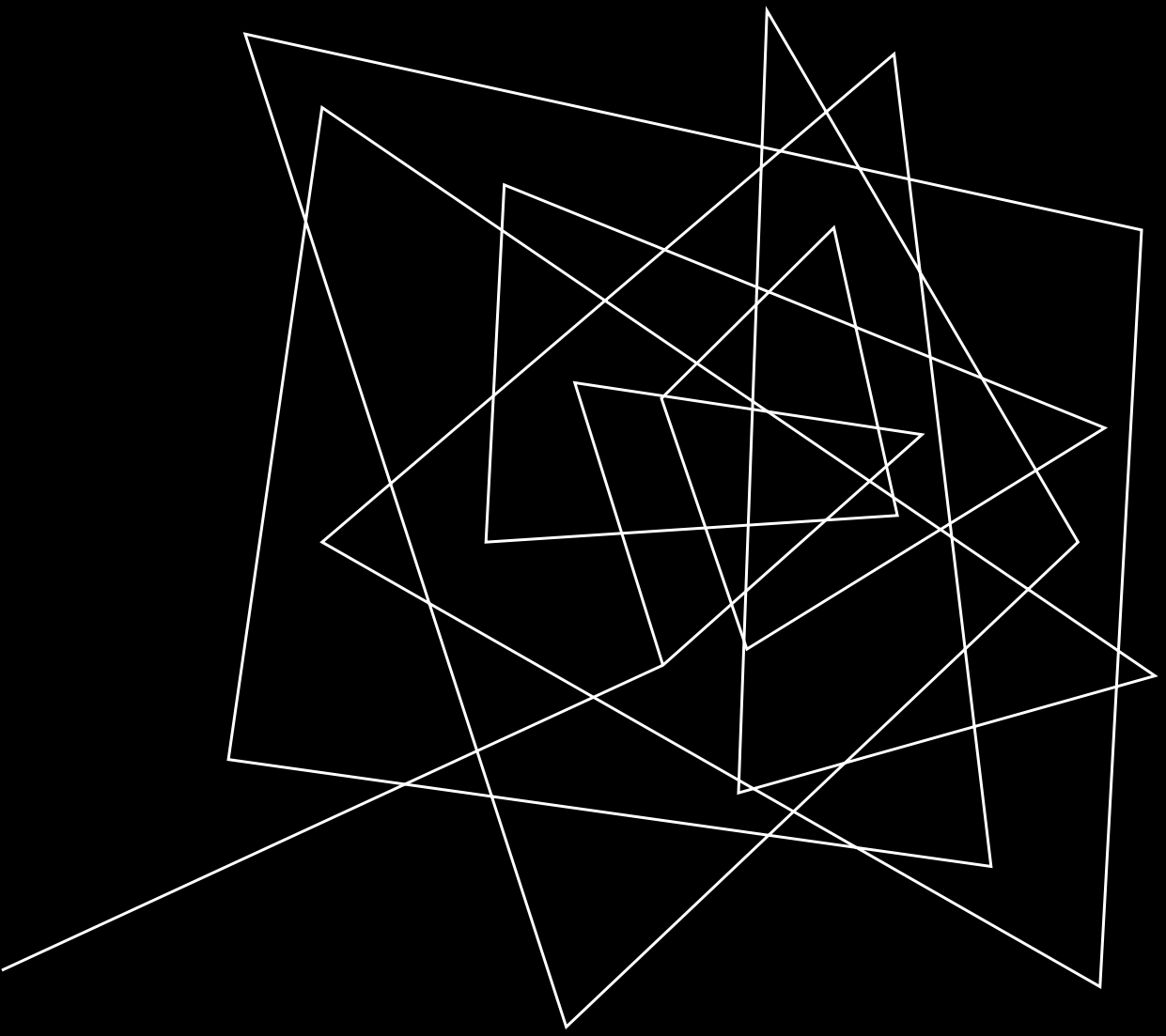
PROBLEM FORMULATION

- Vocabulary: X with P(X) the space of probability distributions over X
- $f_\theta: x^N \rightarrow P(X)$
- $f_\theta(p) = \text{softmax}(W * g_\theta(p))$
- $W = l \times h, \quad h << l$

- $W * g_\theta(p) \rightarrow$ logit
- Softmax(logit) -> logprob

We assume we only have access to a perfect oracle O that does not leak any information.

$y = O(p)$



THE HACK

Steps:

1. **Assume access to logits**
2. Full layer extraction
3. Calculate the logits with the logprobs
4. Calculate the logprobs

ACCESS TO LOGITS

$O(p) <- W^* g_\theta(p)$

Intuition:

- Each output vector is l-dimensional but lives in a h-dimensional subspace.
- By querying the model more than h times we will observe new queries are linearly dependent of past queries.

ALGORITHM

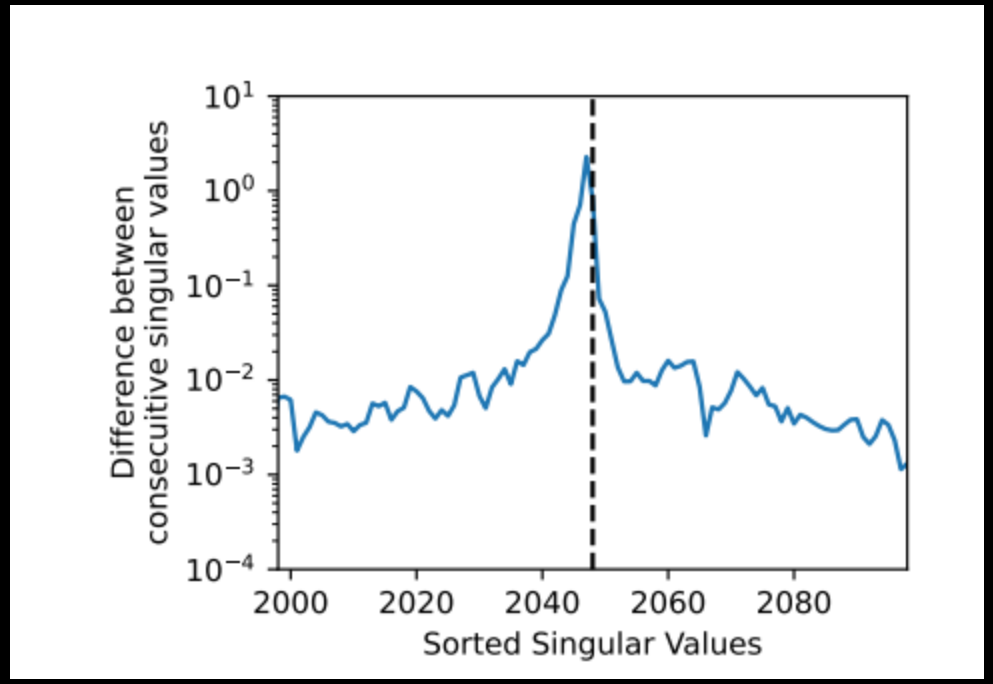
Algorithm 1 Hidden-Dimension Extraction Attack

Require: Oracle LLM $\mathcal{O}$

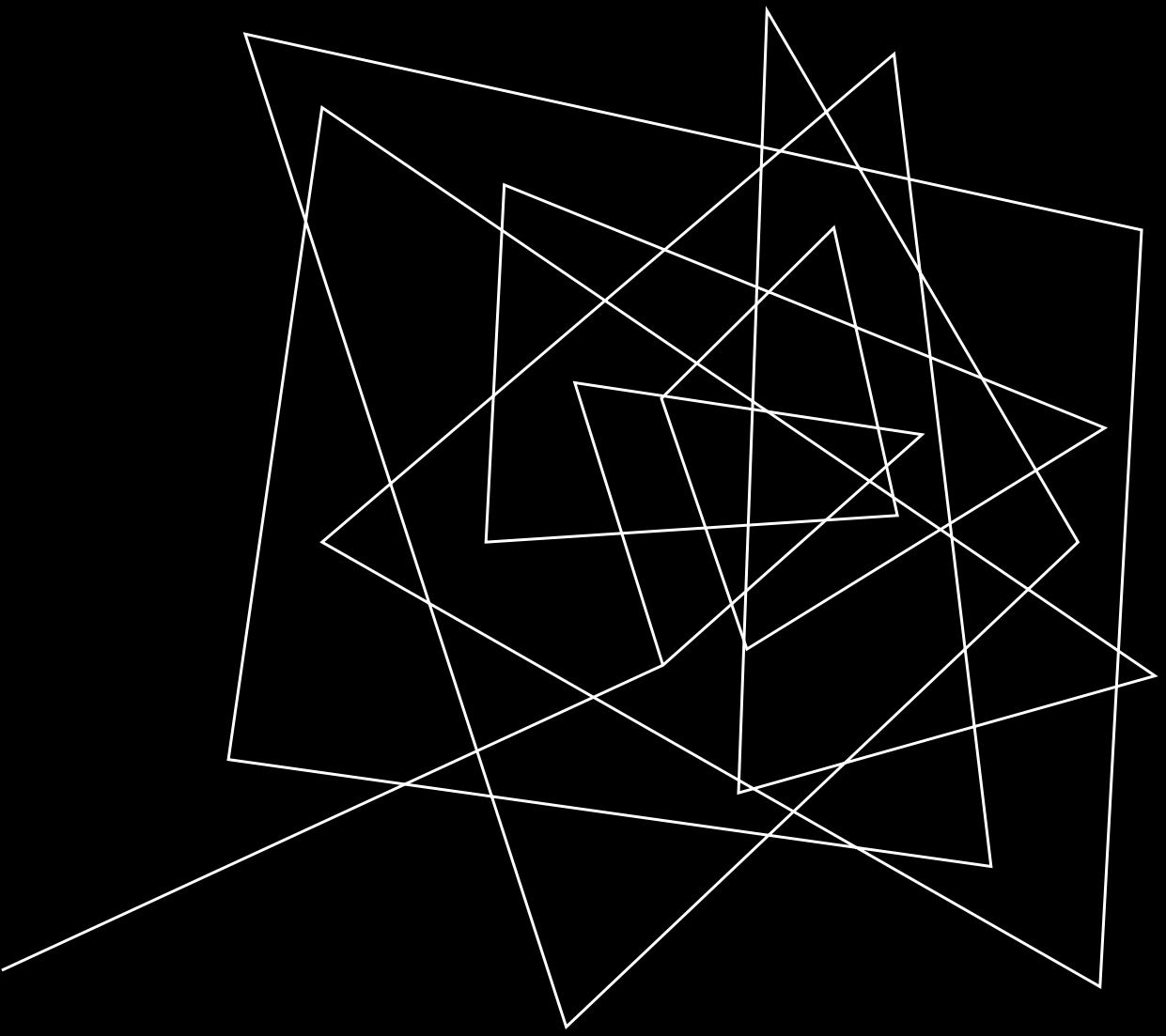
Initialize n to an appropriate value greater than h
Initialize an empty matrix $\mathbf{Q} = \mathbf{0}^{n \times l}$
for $i = 1$ to n **do**
    $p_i \leftarrow \texttt{RandPrefix}()$ ▷ Choose a random prompt
    $\mathbf{Q}_i \leftarrow \mathcal{O}(p_i)$
end for
$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \leftarrow \text{SingularValues}(\mathbf{Q})$
count $\leftarrow \arg\max_i \log\|\lambda_i\| - \log\|\lambda_{i+1}\|$
return count

EARLY RESULTS

- On pythia 1.4B
- The revealed secret size is 2048



- Also worked on gpt 2 and LLaM



NEXT STEP

Steps:

1. Assume access to logits
2. **Full layer extraction**
3. Calculate the logits with the logprobs
4. Calculate the logprobs

FULL LAYER EXTRACTION

Using SVD: $Q = U * \Sigma * V^T$

But: $U * \Sigma = W * G$

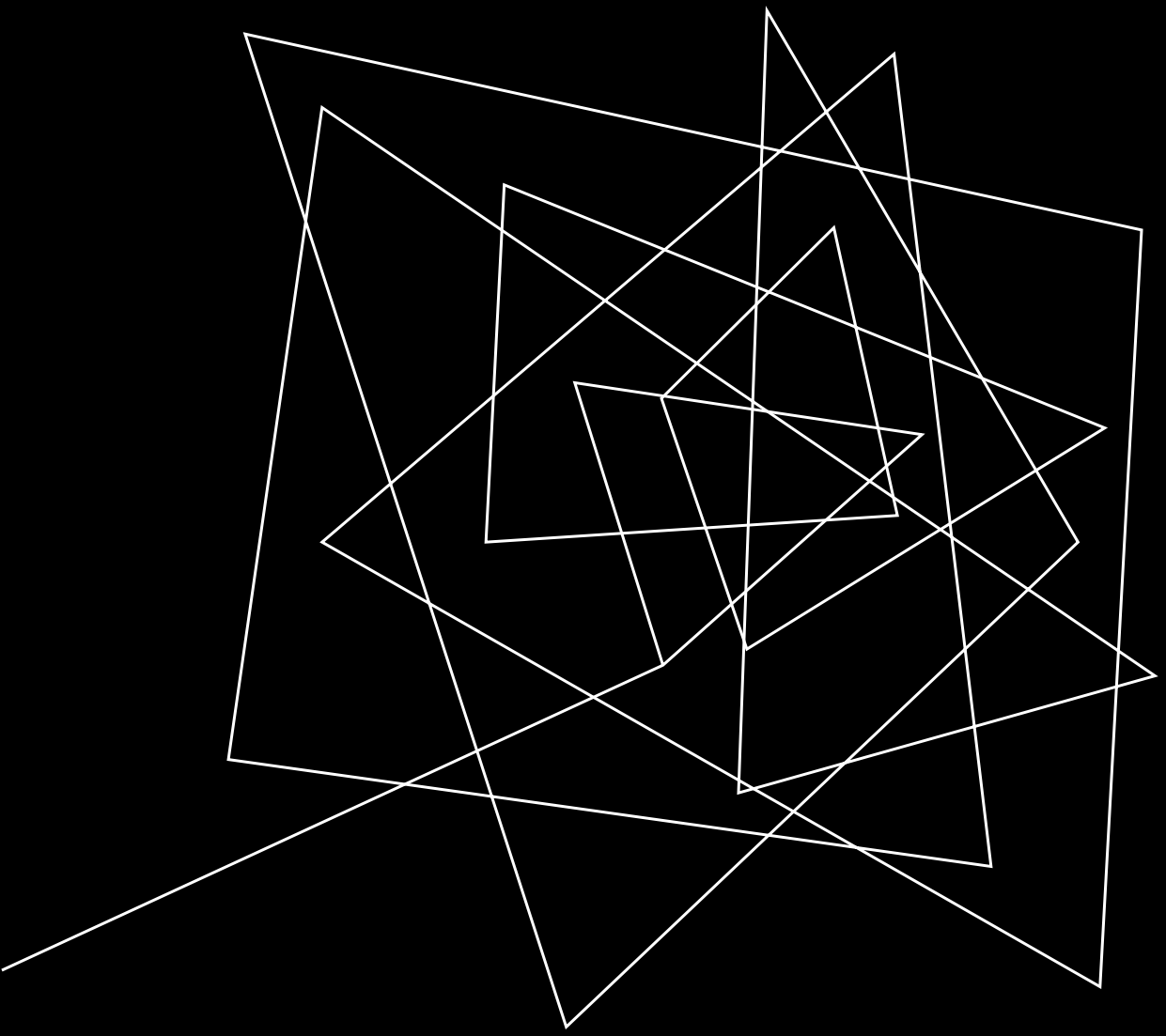
And $W * G$ is really close to W. (up to symmetries)

The attack success rate is evaluated by reporting the

RMS of W and W*G

EARLY RESULTS

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



NEXT STEP

Steps:

1. Assume access to logits
2. Full layer extraction
3. **Calculate the logits with the logprobs**
4. Calculate the logprobs

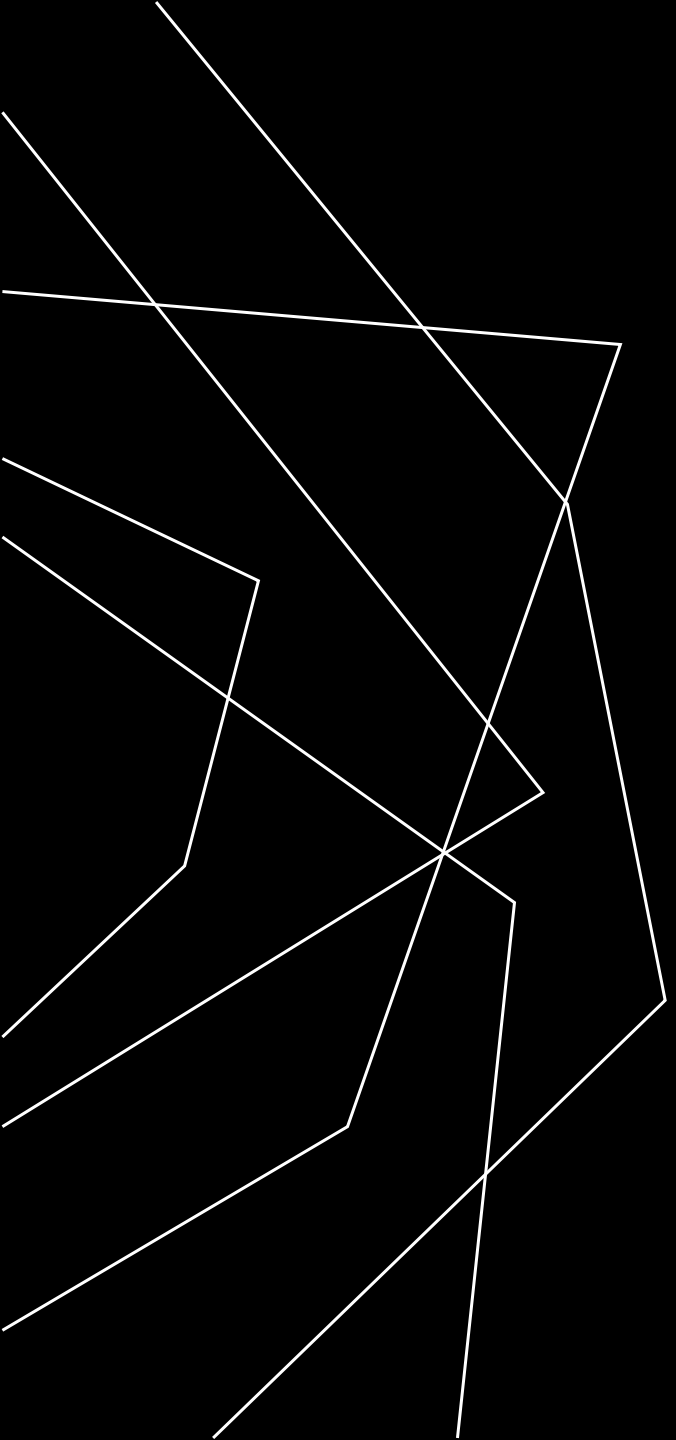
EXTRACT LOGITS USING LOGRPOBS AND BIASES

$O(p,b) <- \text{Top-K}(\text{logsoftmax}(W * g_\theta(p) + b)$

Evaluation methods:

- **Token cost:** Number of tokens sent or received
- **Query cost:** Duration of an attack in number of queries

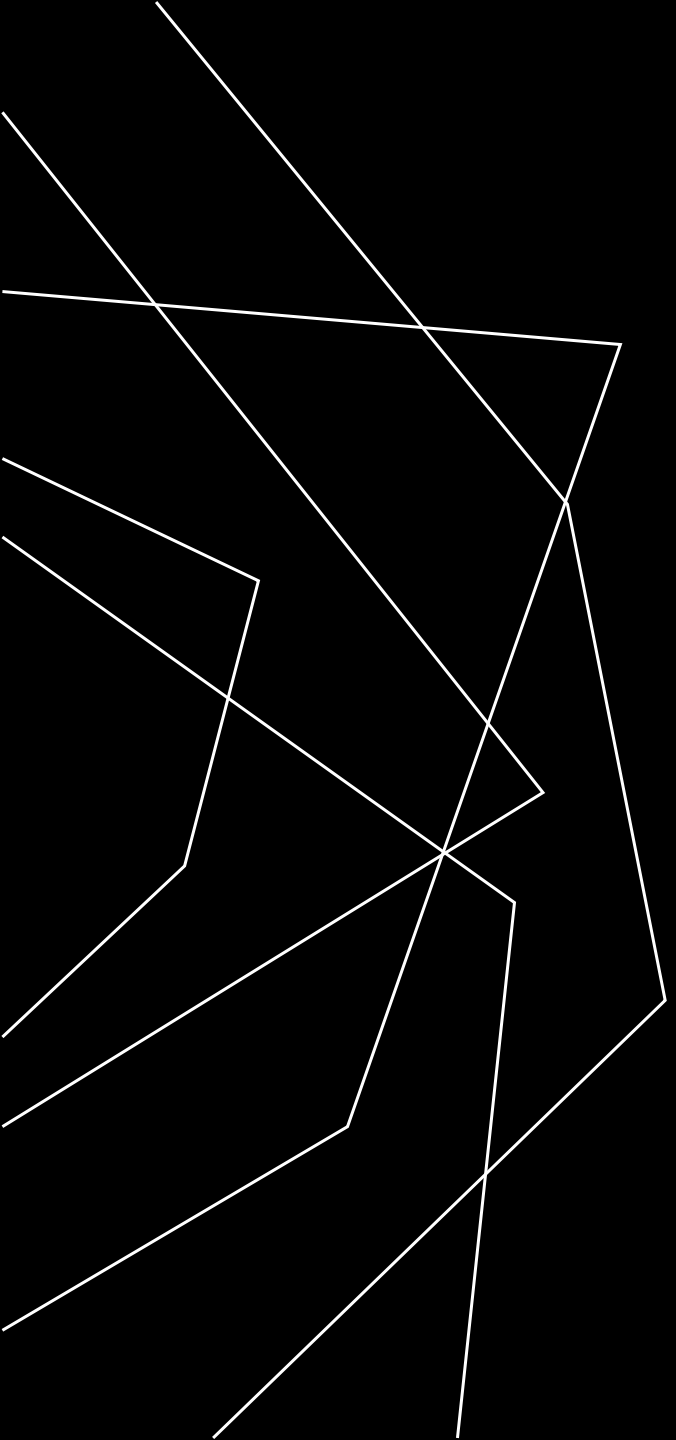
Assume and API that uses Top-5



TOP-5 EXTRACTIONS

An api returns the top-5 logits sorted by their respective logprobs. So we can extract the full logit vector for a prompt p by cycling through different choices for the logit bias and measuring the top-K logits each time.

$$O(p, b_k, b_{k+1}, b_{k+2}, b_{k+3}, b_{k+4} = B)$$



METHOD #1

Result: $y_i = z_i + B + \text{term}(B)$

Each cycle we will keep the top logprob as the reference (y_R) and will calculate the difference to the other four: $y_R - y_i - B = z_R - z_i$

- **Query cost:** 1/(K-1) queries per logit
- **Token cost:** (2+Δ)/4 tokens per query

METHOD #2 COST-OPTIMAL

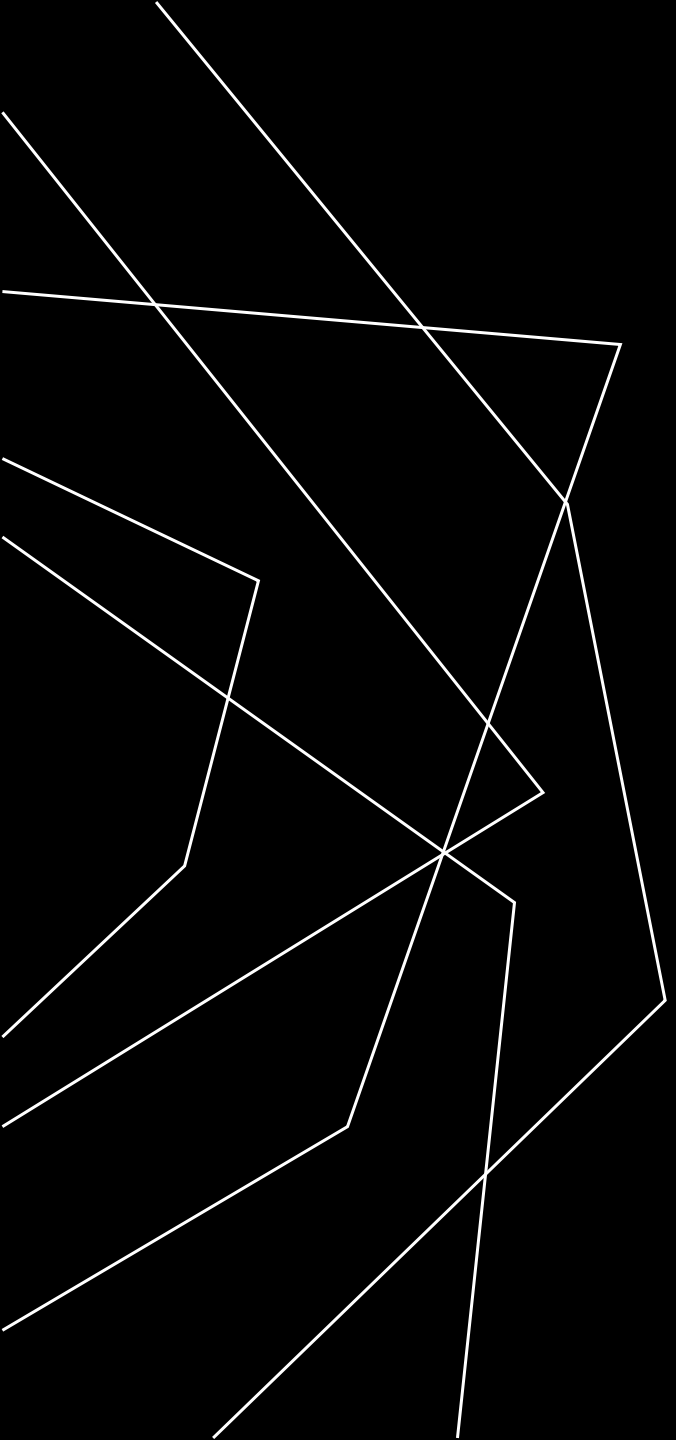
Token X: With Bias B

4 other tokens: With Bias B' < B, but big enough so they will be in the top-5

Ask for m words. (expansion factor)

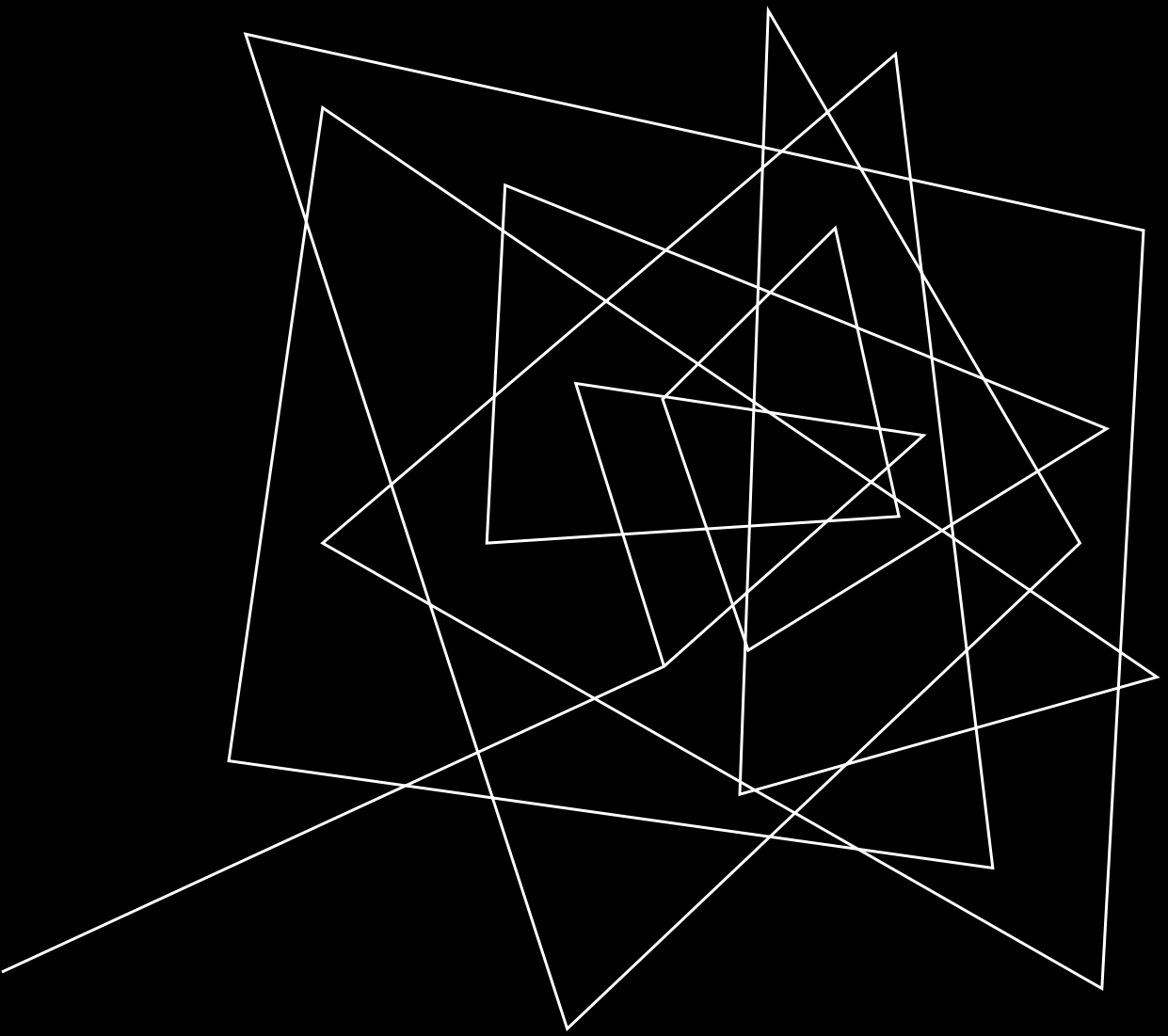
Then: The model will emit [x x x ... x] and we will see the logits of the 4 tokens

- **Query cost:** 1/(4m)
- Token cost: (1 + m + Δ)/(4m)



TOP-1 + BINARY BIAS

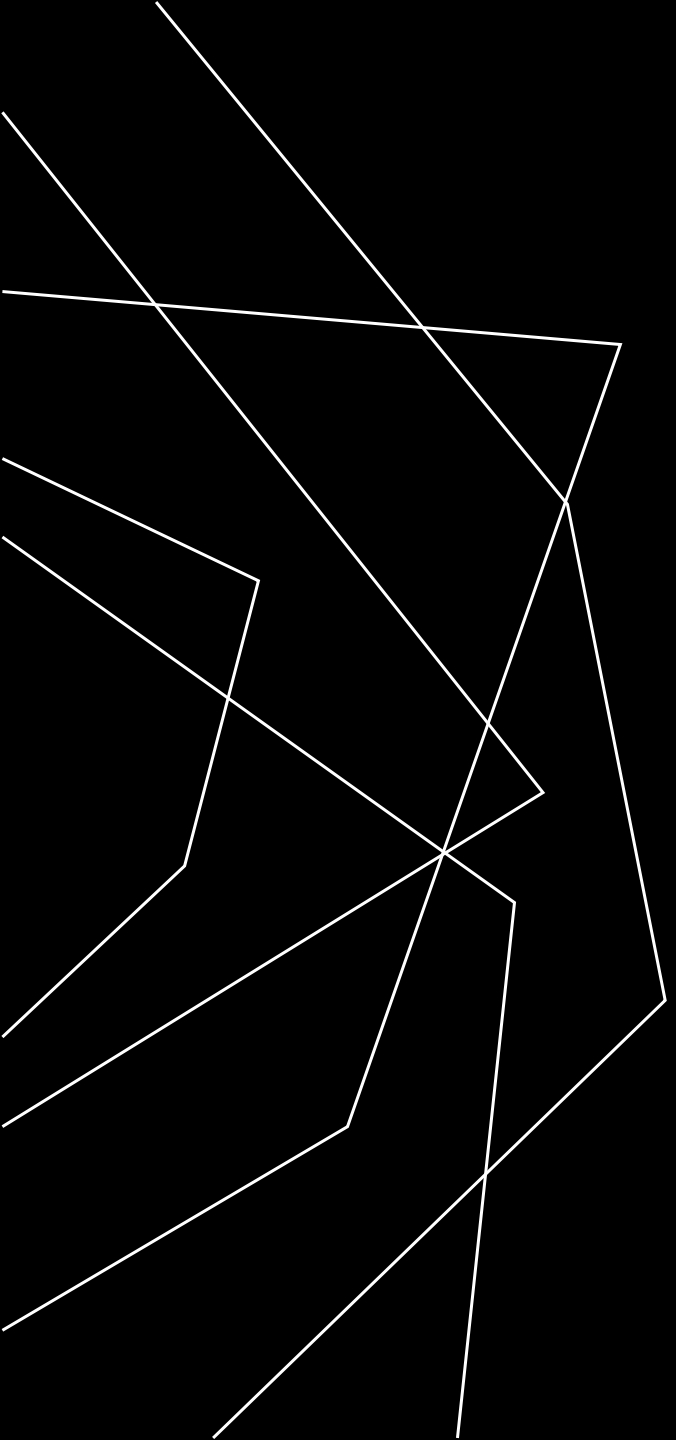
- Query the model twice for a token t
  - Once with no logit bias
  - Once with a bias of −1
- The second time the top token will be slightly more likely
- The logprob of t is the difference between the top logprob in the two executions
- Much less numerically stable – But the constraints are impractical



LAST STEP

Steps:

1. Assume access to logits
2. Full layer extraction
3. Calculate the logits with the logprobs
4. **Calculate the logprobs**



LOGPROB FREE ATTACKS

Intuitively: Assuming a top token A, we binary search the exact bias for a token B which if increased by e will make B the top token.

This bias equals the difference of their logprobs.

- Cost: N log(B/e)
- It is possible to modify multiple tokens at the same time

OTHER ATTACKS

- **Logprob-k:** After the attacker gets the top-K words they burry them with negative bias.
- **Hyperrectangle:** The attacker modifies the biases of multiple tokens at once.
- **One of n:** The attacker isolates a group of tokens by giving them a large bias B.

ATTACK ACCURACY

Attack	Logprobs	Bits of precision	Queries per logit
logprob-4	top-5	23.0	0.25
logprob-5	top-5	11.5	0.64
logprob-1	top-1	6.1	1.0
binary search	✗	7.2	10.0
hyperrectangle	✗	15.7	5.4
one-of-n	✗	18.0	3.7

RESULTS

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†+

HOW FAR ARE THE LOGPROB-FREE ATTACKS AWAY FROM OPTIMAL

- What are the lower bounds on the number of queries needed

Assuming that the logit distribution in [-B,0] is uniform.

The lower bound is proved to be: $\log_2(B/e) / \log_2(N)$

For B = 100, N = 300, e = between 6 and 23 digits of precision. Our attack is just 1 query per logit worse.

AFTERMATH

- Possible defenses
- Possible extensions
- Impact statement

DEFENSES

Prevention

- Remove logit bias
  - There are legitimate use cases for them
- Replace logit bias with block list
- Post-hoc altering the architecture

Mitigations

- Logit bias xor logprobs
- Noise addition
  - ➢ Makes the model less useful
- Detect malicious queries

EXTENSIONS

- Breaking symmetry with quantized weights
- Extending the attack beyond a single layer
- Removing the logit bias assumption
- Exploiting the stolen weights
- Practical stealing of other model information

QUESTIONS THAT COME UP

1. How hazardous are the practical attacks?
2. Do they pose a greater threat to the sequrity of the models than other known attacks?

WHAT CAME AFTER

- The paper got the 'best paper award' in icml 2024.
- OpenAI used some of the defenses mentioned for the future models.

THANK YOU! QUESTIONS?