Efficiently Computing Similarities to Private Datasets

To appear at ICLR 2024

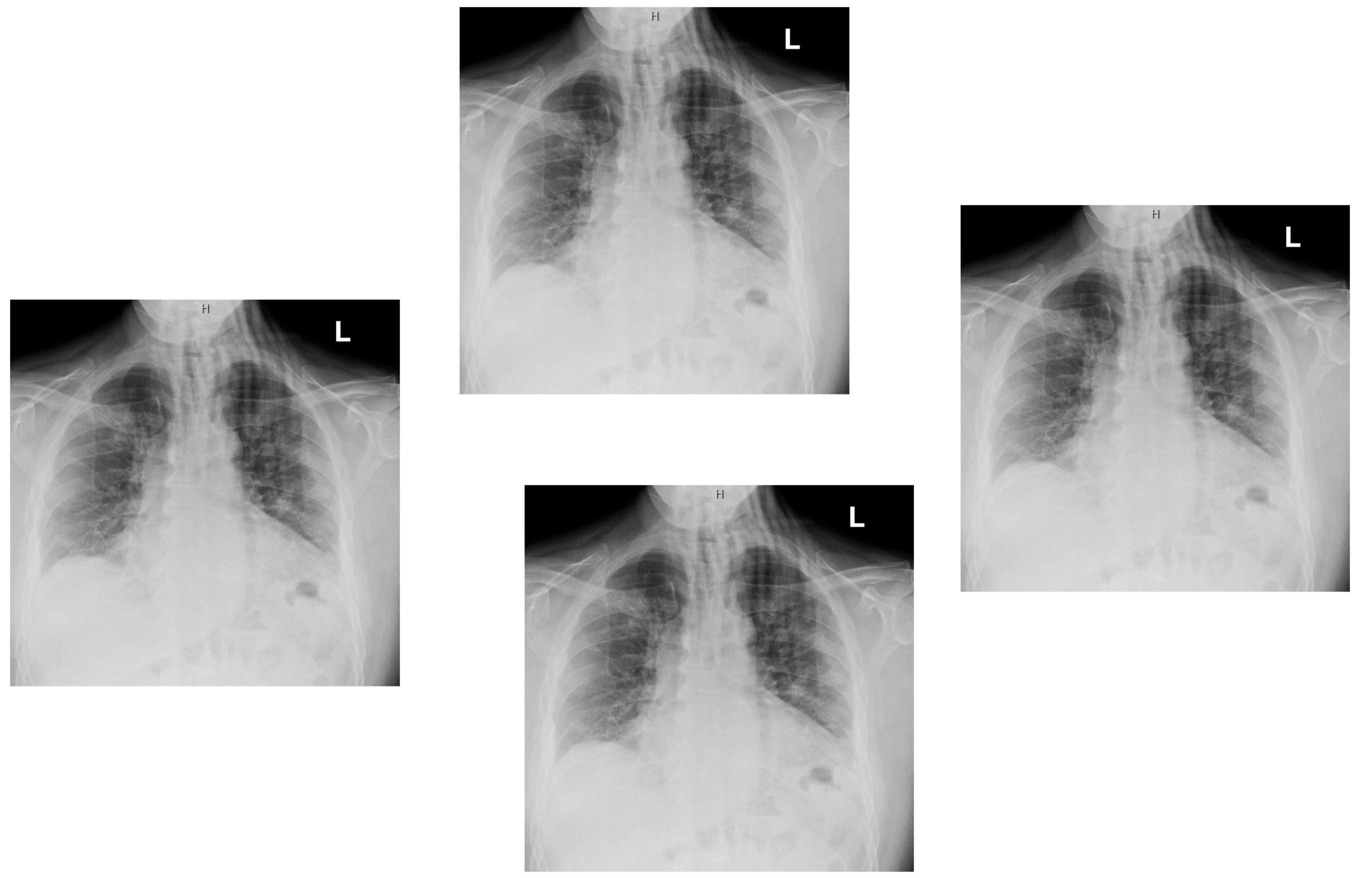
Arturs Backurs*   Zinan Lin*   Sepideh Mahabadi*   Sandeep Silwal^   Jakub Tarnawski*




*Microsoft Research ^MIT

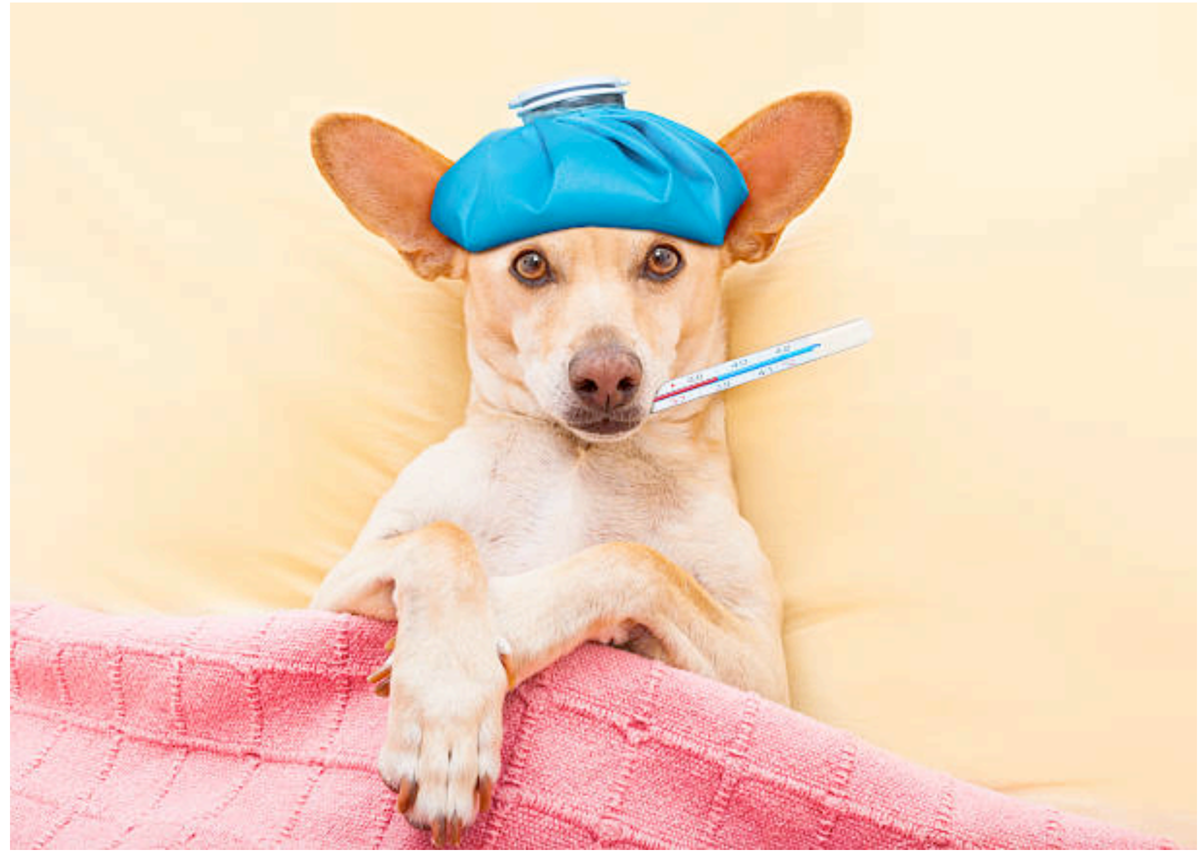

HOSPITAL

Prior Covid
Patients

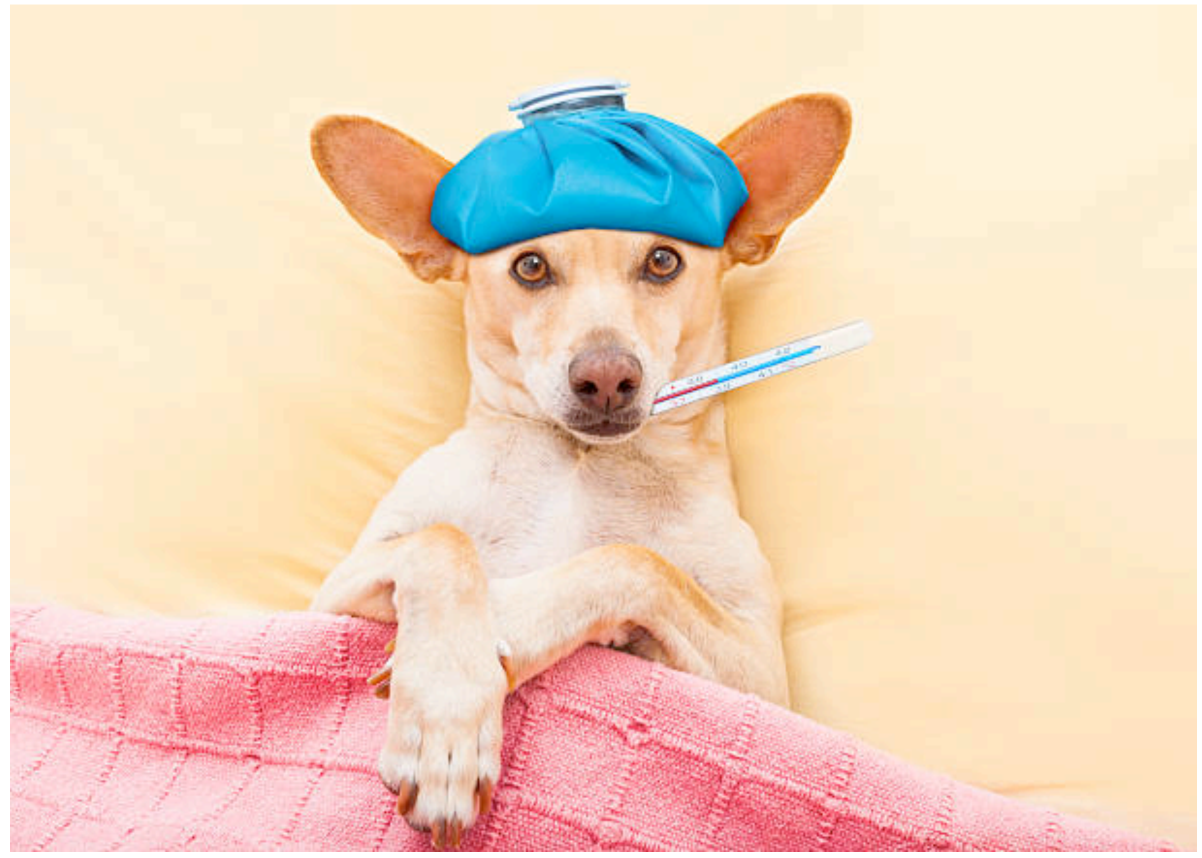


Private!

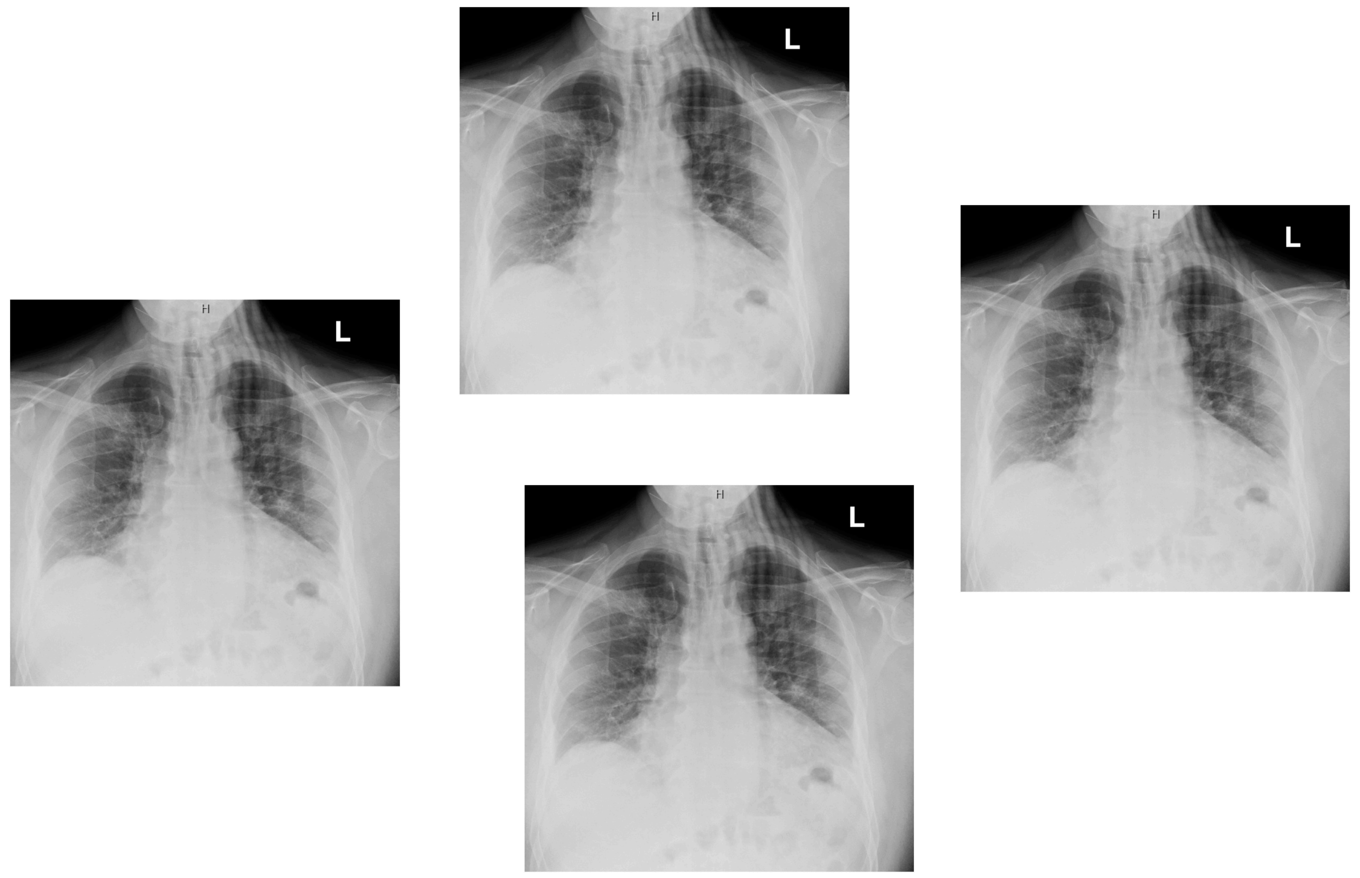
Private!



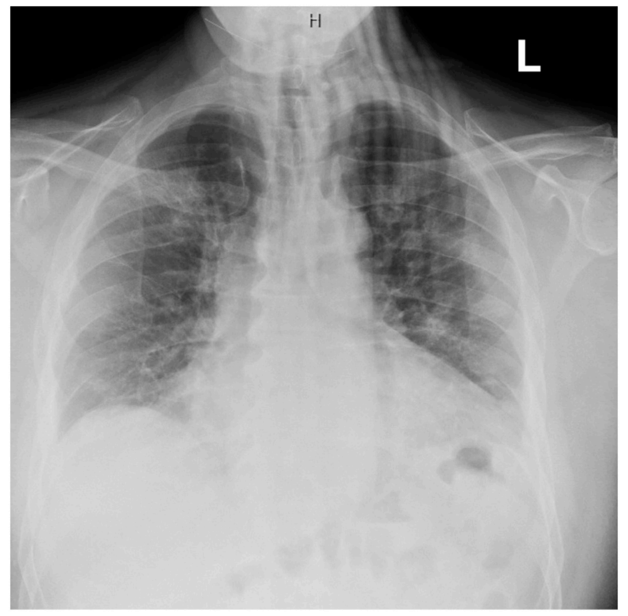
Private!



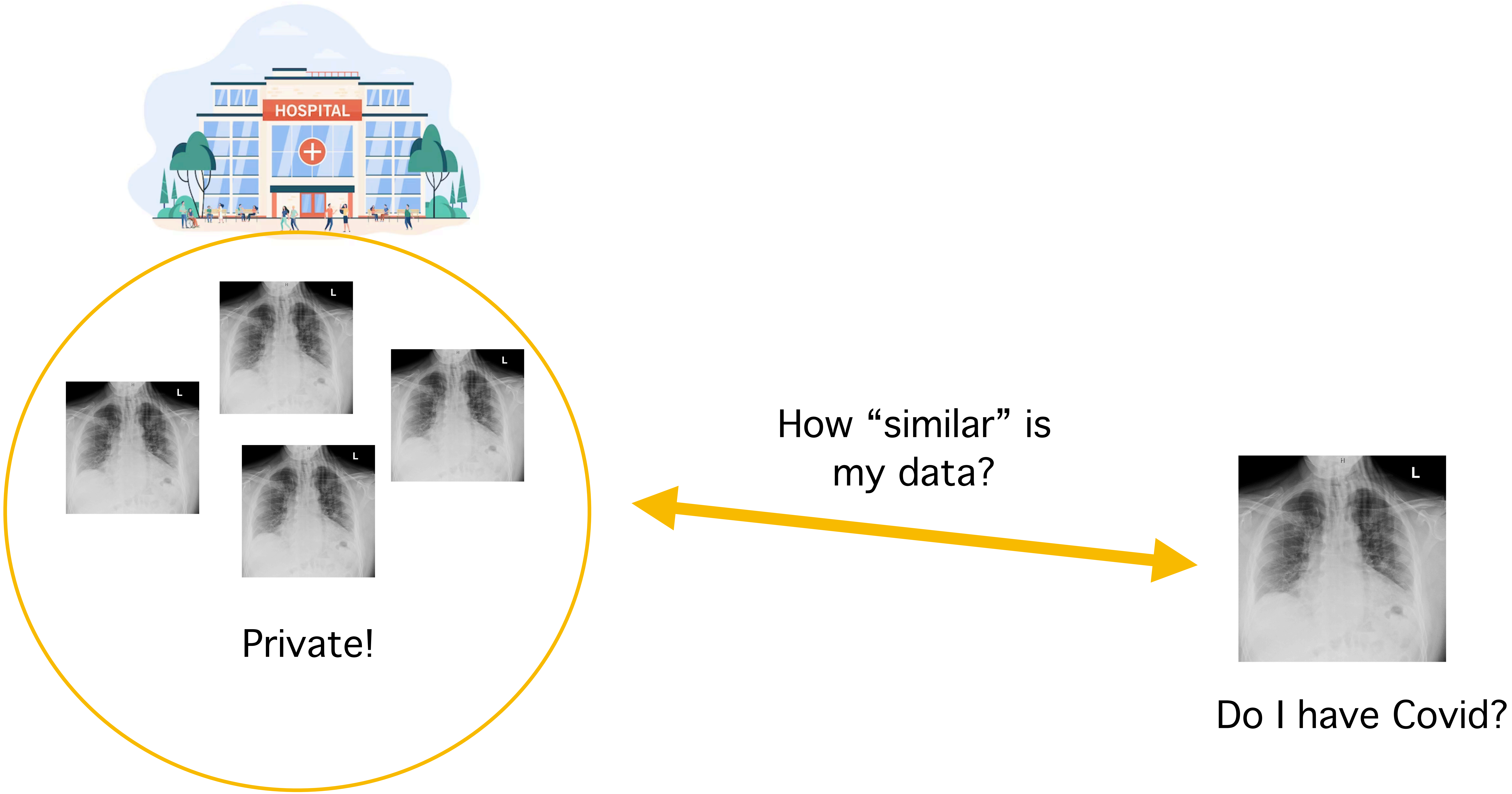
Do I have Covid?



Private!



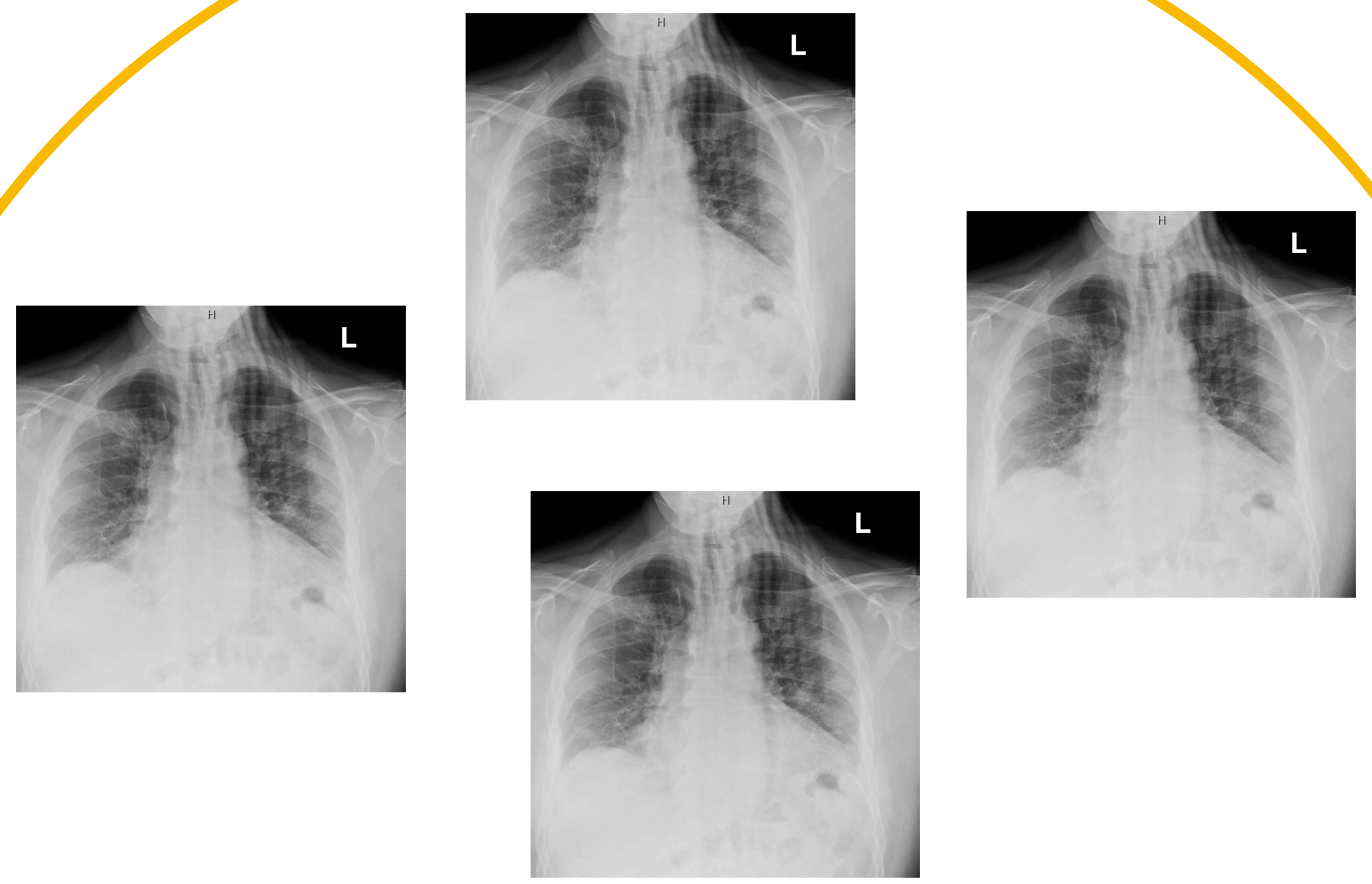
Do I have Covid?


HOSPITAL
Private!
How “similar” is
my data?
Do I have Covid?

Requirements:

- Preserve ‘privacy’ of existing patients
- Output an accurate answer

Lets formalize the problem!

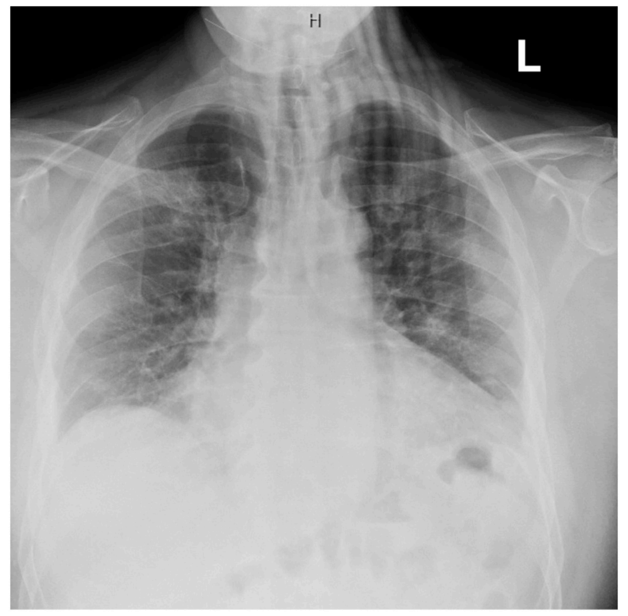


Private!

Requirements:

- Preserve ‘privacy’ of existing patients
- Output an accurate answer

How “similar” is my data?

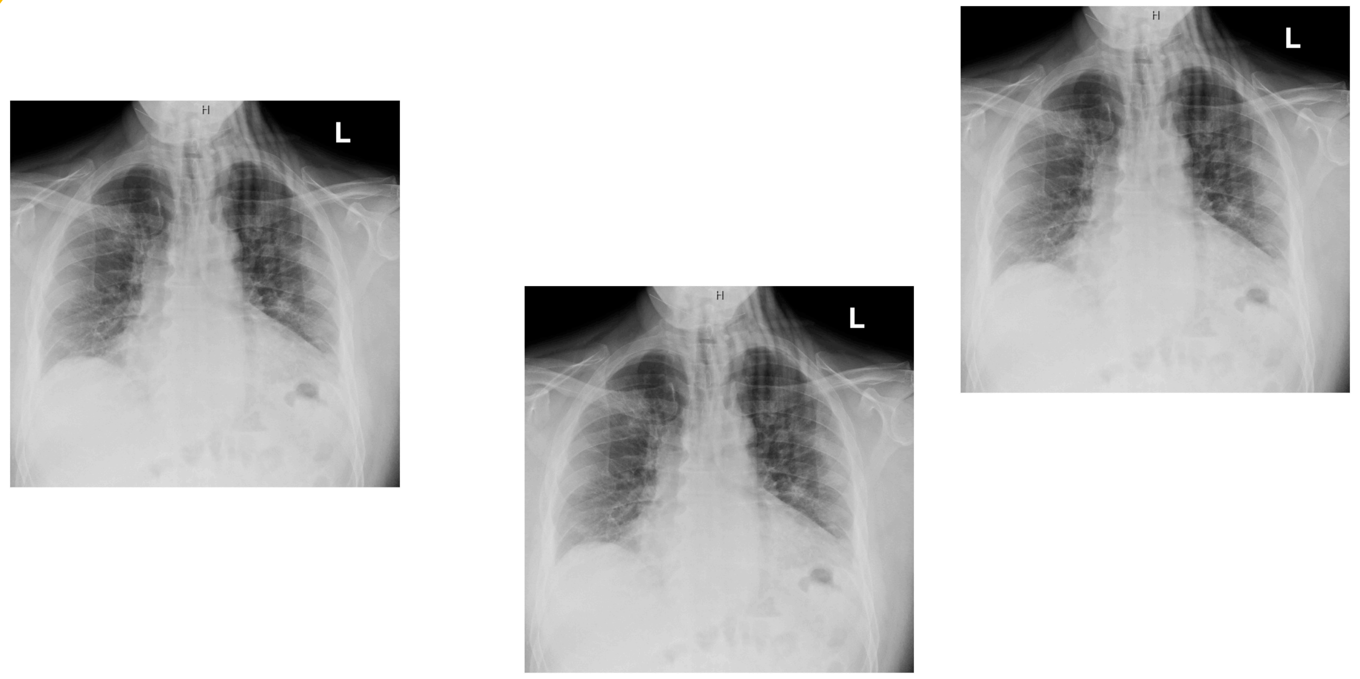


Do I have Covid?

Requirements:

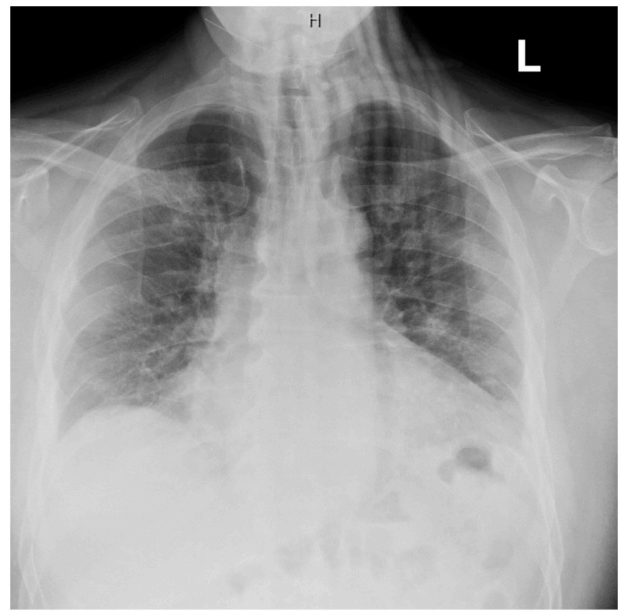
- Preserve ‘privacy’ of existing patients
- Output an accurate answer

$[0.1, -\pi, \log(2)^e, \cdots]$



Private!

How “similar” is my data?



Do I have Covid?

Requirements:

- Preserve ‘privacy’ of existing patients
- Output an accurate answer

Private dataset
$X \subset \mathbb{R}^d$

How “similar” is my data?

Query vector
$y \in \mathbb{R}^d$

Requirements:

- Preserve 'privacy' of existing patients
- Output an accurate answer

Compute $\sum_{x \in X} f(x, y)$ for a desired similarity function $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

Private dataset
$X \subset \mathbb{R}^d$

Query vector
$y \in \mathbb{R}^d$

Requirements:

- Preserve ‘privacy’ of existing patients
- Output an accurate answer

Compute $\sum_{x \in X} f(x, y)$ for a desired similarity function $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

Private dataset
$X \subset \mathbb{R}^d$

Query vector
$y \in \mathbb{R}^d$

Hospital (Algorithm)

Requirements:

- Preserve 'privacy' of existing patients
- Output an accurate answer

Compute $\sum_{x \in X} f(x, y)$ for a desired similarity function $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

Query vector $y \in \mathbb{R}^d$

Hospital (Algorithm)

- Takes in input dataset X
- Alg outputs a data structure $\mathscr{D}$
- $\mathscr{D}$ is differential private wrt X
- On any query y, $\mathscr{D}$ approximates sum

Requirements:

- Preserve 'privacy' of existing patients
- Output an accurate answer

Compute $\sum_{x \in X} f(x, y)$ for a desired similarity function $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

Query vector $y \in \mathbb{R}^d$

Hospital (Algorithm)

- Takes in input dataset X
- Alg outputs a data structure $\mathscr{D}$
- $\mathscr{D}$ is differential private wrt X
- On any query y, $\mathscr{D}$ approximates sum

Note:

- We want the 'summary' $\mathscr{D}$ itself to be private
- Thus we can compute arbitrarily many queries!

Compute $\sum_{x \in X} f(x, y)$ for a desired similarity function $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

Query vector $y \in \mathbb{R}^d$

Hospital (Algorithm)

- Takes in input dataset X
- Alg outputs a data structure $\mathscr{D}$
- $\mathscr{D}$ is differential private wrt X
- On any query y, $\mathscr{D}$ approximates sum

Compute $\sum_{x \in X} f(x, y)$ for a desired similarity function $f : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$

Query vector $y \in \mathbb{R}^d$

Example:

- $\mathscr{D}$ always outputs 42!
- Always private but not very accurate … can we do better?

Input: Data X, privacy parameter ε, similarity function $f(x, y)$

Alg Outputs: An ε-DP (differentially private) data structure $\mathscr{D}$.

Goal:

- On any fixed query y, have small $\mathbb{E}[\,|\,\text{True} - \mathscr{D}(y)\,|\,]$.
- True = $\sum_{x \in X} f(x, y)$
- Expectation over the randomness used to construct $\mathscr{D}$

Input: Data X, privacy parameter ε, similarity function $f(x, y)$

Alg Outputs: An ε-DP (differentially private) data structure $\mathscr{D}$.

Goal:

- On any fixed query y, minimize $\mathbb{E}[|\text{True} - \mathscr{D}(y)|]$.
- Expectation over the randomness used to construct $\mathscr{D}$

Privacy:

- "Alg outputs similar $\mathscr{D}$ even if we change a single data point"
- Smaller ε = more stringent requirements.

What are the tradeoffs between $\mathbb{E}[|\text{True} - \mathscr{D}(y)|]$ and privacy?

Our Results

Study tradeoffs for a wide class of natural (similarity or dissimilarity) functions f.

Let $n = |X|$ denote dataset size, d is dataset dimension

Our Results

Study tradeoffs for a wide class of natural functions f.

Let $n = |X|$ denote dataset size, d is dataset dimension

Error Notation: (M, A) means $\mathbb{E}[\,|\text{True} - \mathcal{D}(y)|\,] \leq M \cdot \text{True} + A$

(Multiplicative error, Additive error)

- We want both to be as small as possible.

n = Dataset Size

d = Dataset dimension

(M, A) means $\mathbb{E}[\,|\,\text{True} - \mathcal{D}(y)\,|\,] \leq M \cdot \text{True} + A$

Hiding ε and log terms

n = Dataset Size

d = Dataset dimension

(M, A) means $\mathbb{E}[\,|\,\text{True} - \mathscr{D}(y)\,|\,] \leq M \cdot \text{True} + A$

Hiding ε and log terms

$f(x, y)$	Our (M, A)	Prior (M, A)	Our Query Time	Prior Query Time	Note/Ref
$\|x - y\|_1$					Bounded data [Huang, Roth '14]
$\|x - y\|_2$					Bounded data [Huang, Roth '14]

n = Dataset Size

d = Dataset dimension

(M, A) means $\mathbb{E}[|\text{True} - \mathcal{D}(y)|] \leq M \cdot \text{True} + A$

Hiding ε and log terms

$f(x, y)$	Our (M, A)	Prior (M, A)	Our Query Time	Prior Query Time	Note/Ref
$\|x - y\|_1$	$\alpha, \frac{d^{1.5}}{\sqrt{\alpha}}$	$0, (nd^7)^{1/3}$	d	d	Bounded data [Huang, Roth '14]
$\|x - y\|_2$					Bounded data [Huang, Roth '14]

n = Dataset Size $\quad (M, A)$ means $\mathbb{E}[\,|\text{True} - \mathscr{D}(y)|\,] \le M \cdot \text{True} + A$

d = Dataset dimension $\quad$ Hiding ε and log terms

$f(x, y)$	Our (M, A)	Prior (M, A)	Our Query Time	Prior Query Time	Note/Ref
$\lVert x - y \rVert_1$	$\alpha, \frac{d^{1.5}}{\sqrt{\alpha}}$	$0, \left(nd^7\right)^{1/3}$	d	d	Bounded data [Huang, Roth '14]
$\lVert x - y \rVert_2$	$\alpha, \frac{1}{\alpha^{1.5}}$	Similar as above	d	$\ge d$	Bounded data [Huang, Roth '14]

n = Dataset Size

d = Dataset dimension

(M, A) means $\mathbb{E}[\,|\text{True} - \mathcal{D}(y)|\,] \leq M \cdot \text{True} + A$

Hiding ε and log terms

$f(x, y)$	Our (M, A)	Prior (M, A)	Our Query Time	Prior Query Time	Note/Ref
$\|x - y\|_1$	$\alpha, \frac{d^{1.5}}{\sqrt{\alpha}}$	$0, \left(nd^7\right)^{1/3}$	d	d	Bounded data [Huang, Roth '14]
$\|x - y\|_2$	$\alpha, \frac{1}{\alpha^{1.5}}$	Similar as above	d	$\geq d$	Bounded data [Huang, Roth '14]
$e^{-\|x-y\|_2}$					[Wagner et al. '23]

n = Dataset Size

d = Dataset dimension

(M, A) means $\mathbb{E}[\,|\text{True} - \mathcal{D}(y)|\,] \leq M \cdot \text{True} + A$

Hiding ε and log terms

$f(x,y)$	Our (M,A)	Prior (M,A)	Our Query Time	Prior Query Time	Note/Ref
$\|x-y\|_1$	$\alpha, \frac{d^{1.5}}{\sqrt{\alpha}}$	$0, \left(nd^7\right)^{1/3}$	d	d	Bounded data [Huang, Roth '14]
$\|x-y\|_2$	$\alpha, \frac{1}{\alpha^{1.5}}$	Similar as above	d	$\geq d$	Bounded data [Huang, Roth '14]
$e^{-\|x-y\|_2}$	$0, \alpha$	$0, \alpha$		$\frac{d}{\alpha^2}$	$n \gg \frac{1}{\alpha}$ [Wagner et al. '23]

n = Dataset Size

d = Dataset dimension

(M, A) means $\mathbb{E}[\,|\text{True} - \mathcal{D}(y)|\,] \leq M \cdot \text{True} + A$

Hiding ε and log terms

$f(x, y)$	Our (M, A)	Prior (M, A)	Our Query Time	Prior Query Time	Note/Ref
$\|x - y\|_1$	$\alpha, \frac{d^{1.5}}{\sqrt{\alpha}}$	$0, \left(nd^7\right)^{1/3}$	d	d	Bounded data [Huang, Roth '14]
$\|x - y\|_2$	$\alpha, \frac{1}{\alpha^{1.5}}$	Similar as above	d	$\geq d$	Bounded data [Huang, Roth '14]
$e^{-\|x-y\|_2}$	$0, \alpha$	$0, \alpha$	$d + \frac{1}{\alpha^4}$	$\frac{d}{\alpha^2}$	$n \gg \frac{1}{\alpha}$ [Wagner et al. '23]

n = Dataset Size
d = Dataset dimension

(M, A) means $\mathbb{E}[\,|\,\text{True} - \mathcal{D}(y)\,|\,] \leq M \cdot \text{True} + A$
Hiding ε and log terms

$f(x, y)$	Our (M, A)	Prior (M, A)	Our Query Time	Prior Query Time	Note/Ref
$\lVert x - y \rVert_1$	$\alpha, \frac{d^{1.5}}{\sqrt{\alpha}}$	$0, \left(nd^7\right)^{1/3}$	d	d	Bounded data [Huang, Roth '14]
$\lVert x - y \rVert_2$	$\alpha, \frac{1}{\alpha^{1.5}}$	Similar as above	d	$\geq d$	Bounded data [Huang, Roth '14]
$e^{-\lVert x-y \rVert_2}$	$0, \alpha$	$0, \alpha$	$d + \frac{1}{\alpha^4}$	$\frac{d}{\alpha^2}$	$n \gg \frac{1}{\alpha}$ [Wagner et al. '23]
$\frac{1}{1 + \lVert x - y \rVert_2}$	$0, \alpha$	None!	$d + \frac{1}{\alpha^4}$	None!	$n \gg \frac{1}{\alpha}$

See paper for other results and lower bounds!

$f(x,y)$	Our (M,A)	Prior (M,A)	Our Query Time	Prior Query Time	Note/Ref
$\lVert x-y \rVert_1$	$\alpha, \frac{d^{1.5}}{\sqrt{\alpha}}$	$0, \left(nd^7\right)^{1/3}$	d	d	Bounded data [Huang, Roth '14]
$\lVert x-y \rVert_2$	$\alpha, \frac{1}{\alpha^{1.5}}$	Similar as above	d	$\geq d$	Bounded data [Huang, Roth '14]
$e^{-\lVert x-y \rVert_2}$	$0, \alpha$	$0, \alpha$	$d + \frac{1}{\alpha^4}$	$\frac{d}{\alpha^2}$	$n \gg \frac{1}{\alpha}$ [Wagner et al. '23]
$\frac{1}{1+\lVert x-y \rVert_2}$	$0, \alpha$	None!	$d + \frac{1}{\alpha^4}$	None!	$n \gg \frac{1}{\alpha}$

What the heck do all of our results have in common??

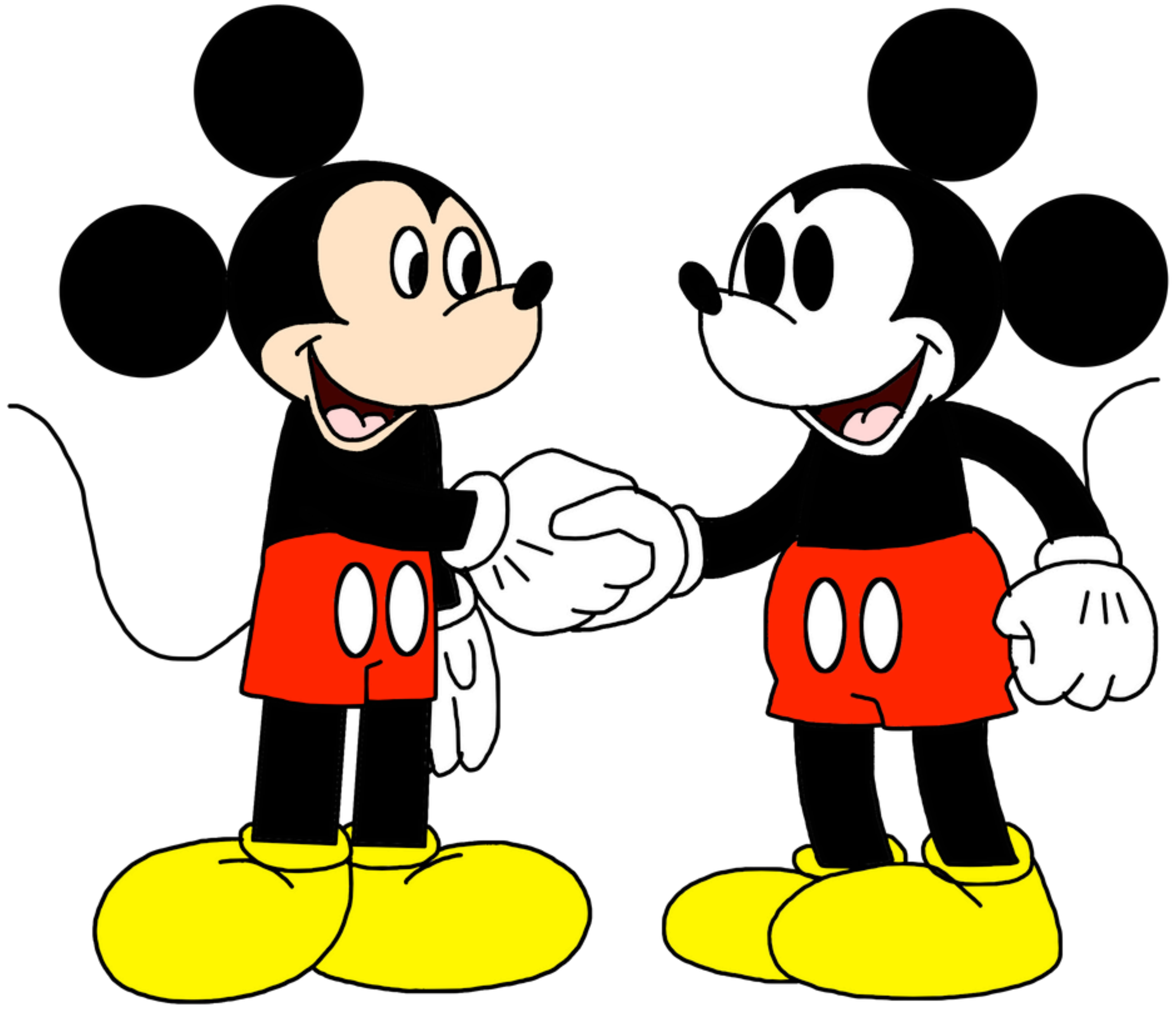
$f(x, y)$	Our (M, A)	Prior (M, A)	Our Query Time	Prior Query Time	Note/Ref
$\|x - y\|_1$	$\alpha, \frac{d^{1.5}}{\sqrt{\alpha}}$	$0, \left(nd^7\right)^{1/3}$	d	d	Bounded data [Huang, Roth '14]
$\|x - y\|_2$	$\alpha, \frac{1}{\alpha^{1.5}}$	Similar as above	d	$\geq d$	Bounded data [Huang, Roth '14]
$e^{-\|x-y\|_2}$	$0, \alpha$	$0, \alpha$	$d + \frac{1}{\alpha^4}$	$\frac{d}{\alpha^2}$	$n \gg \frac{1}{\alpha}$ [Wagner et al. '23]
$\frac{1}{1 + \|x - y\|_2}$	$0, \alpha$	None!	$d + \frac{1}{\alpha^4}$	None!	$n \gg \frac{1}{\alpha}$

All results are unified under one ‘idea’

All results are unified under one 'idea'

Privacy

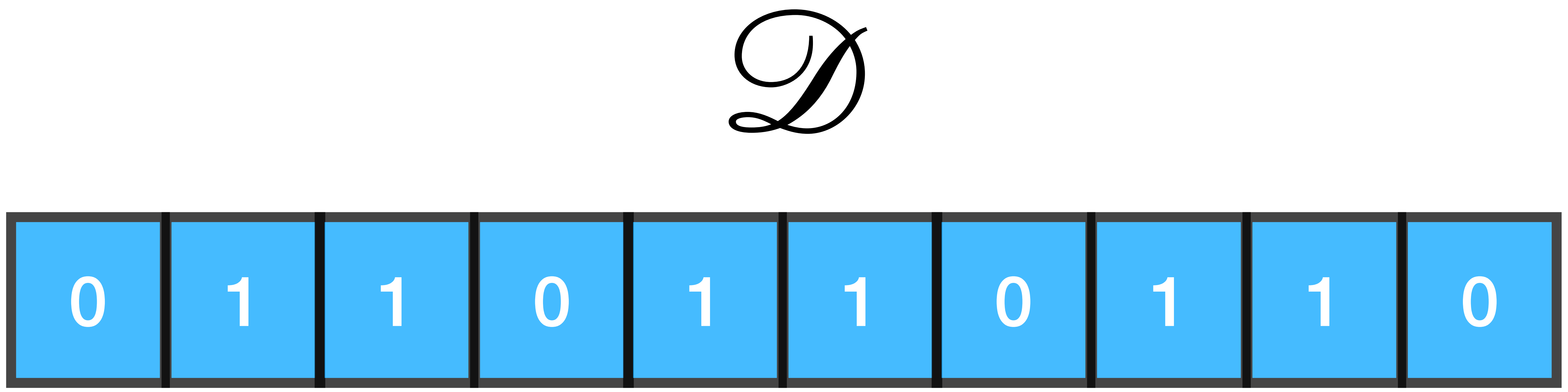
Sublinear Algorithms



*Note Mickey Mouse is in the public domain now I think.... Pls don't sue me

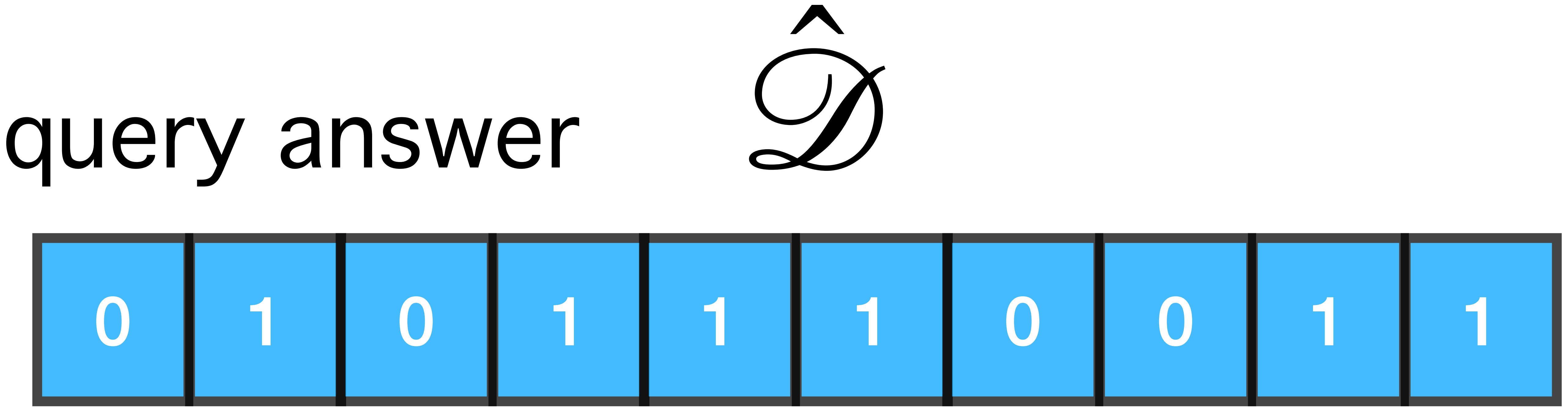
Why? High-level template

- Step 1: Create Data structure $\mathscr{D}$ (first ignoring privacy)
    - $\mathscr{D}$ has sublinear query time



Why? High-level template

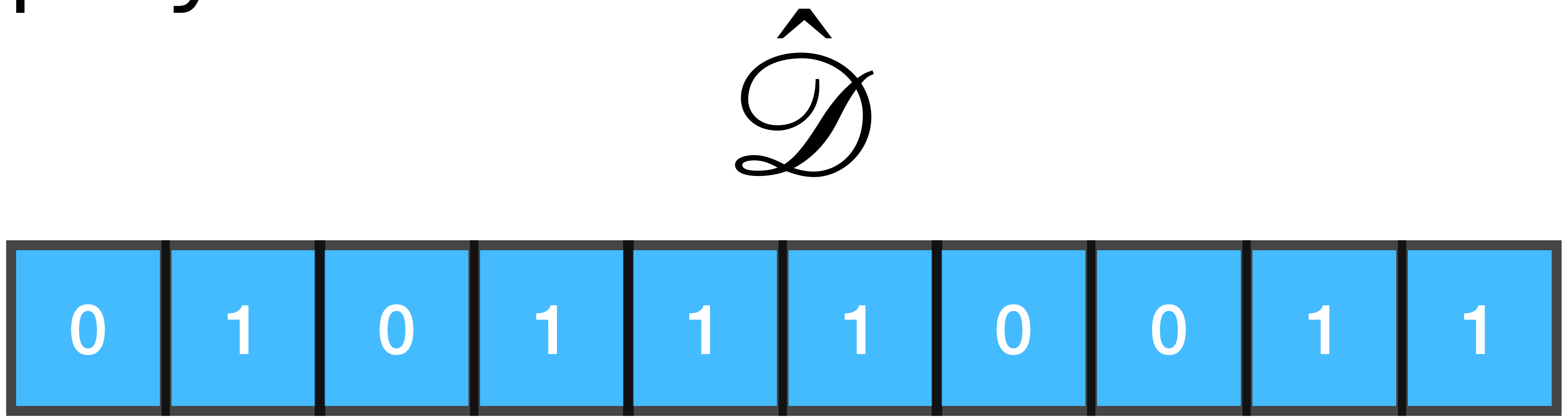
- Step 1: Create Data structure $\mathscr{D}$ (first ignoring privacy)
  - $\mathscr{D}$ has sublinear query time
- Step 2: Add noise to every entry of $\mathscr{D}$ for privacy (common Differential Privacy operation)
- Step 3: Use $\hat{\mathscr{D}}$ to compute query answer



Why? High-level template

Sweeping lots of things under the rug here!

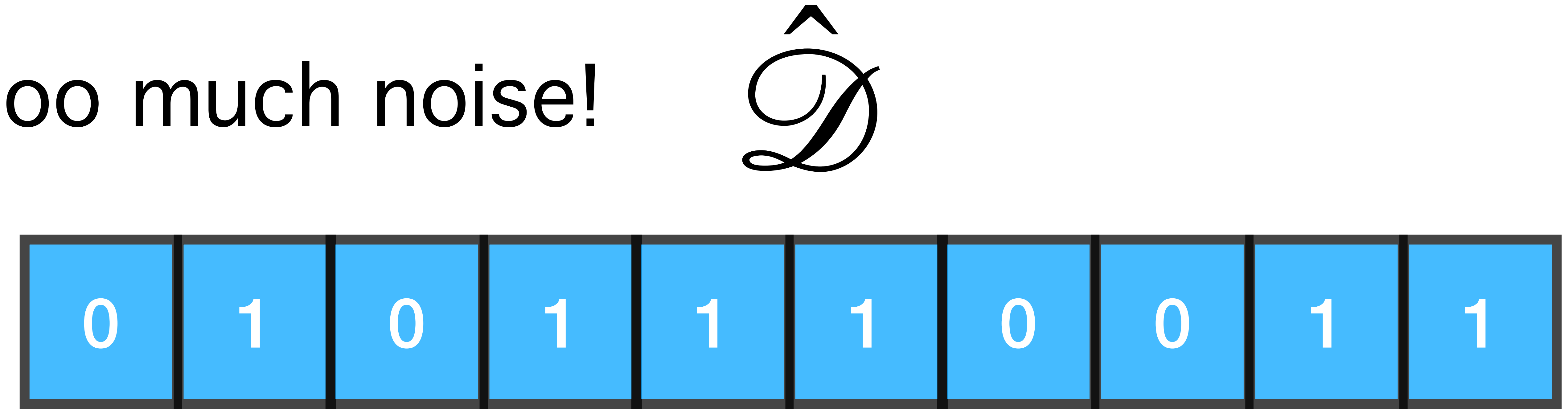
- Step 1: Create Data structure $\mathscr{D}$
    - $\mathscr{D}$ has sublinear query time
- Step 2: Add noise to every entry of $\mathscr{D}$
- Step 3: Use $\hat{\mathscr{D}}$ to compute query answer



- Step 1: Create Data structure $\mathscr{D}$
    - $\mathscr{D}$ has sublinear query time
- Step 2: Add noise to every entry of $\mathscr{D}$
- Step 3: Use $\hat{\mathscr{D}}$ to compute query answer

<u>Crux:</u> Because $\hat{\mathscr{D}}$ has sublinear query time, we only ‘see a small portion’ of $\hat{\mathscr{D}}$ on any query.

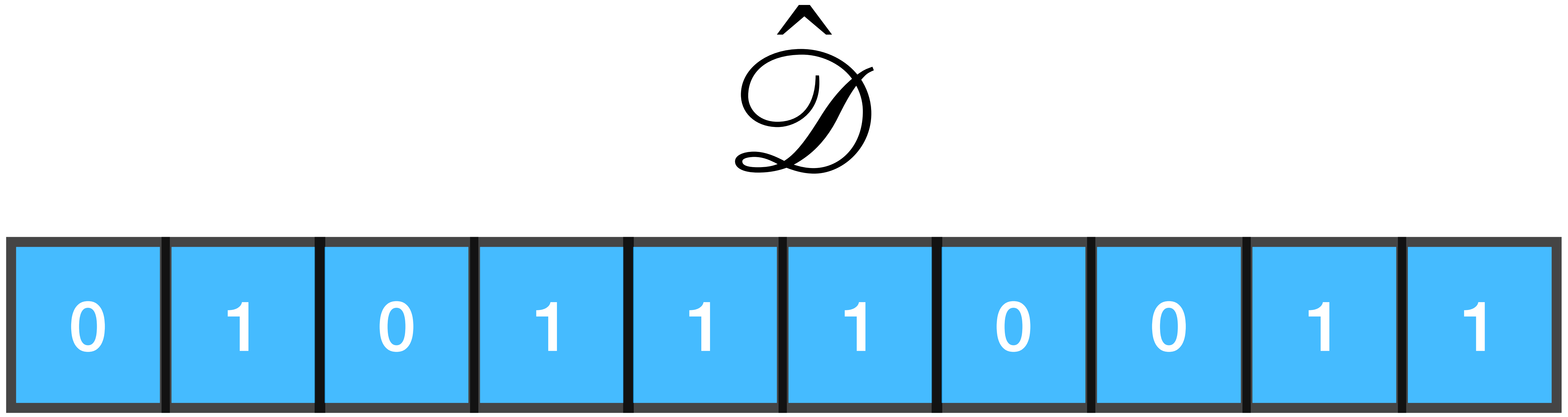
This means we don’t ‘accumulate’ too much noise!



- Step 1: Create Data structure $\mathscr{D}$
    - $\mathscr{D}$ has sublinear query time
- Step 2: Add noise to every entry of $\mathscr{D}$
- Step 3: Use $\hat{\mathscr{D}}$ to compute query answer

<u>Crux:</u> Because $\hat{\mathscr{D}}$ has sublinear query time, we only ‘see a small portion’ of $\hat{\mathscr{D}}$ on any query.

Thus, there is hope for small error!



What sublinear stuff are we using?

Identifying the ‘right’ sublinear structures is the main part of the paper!

Sneak Preview

- $\|x - y\|_1 = \sum_i |x_i - y_i| \implies$ Sufficient to solve 1D problem

Sneak Preview

- $\|x - y\|_1 = \sum_i |x_i - y_i| \implies$ Sufficient to solve 1D problem
- Mantra of computational geometry: every 1D problem can be solved by a tree
- We can design such a tree with sublinear query time

Sneak Preview

- $e^{-\|x-y\|_2}$: We prove a novel dimensionality reduction result

Sneak Preview

- $e^{-\|x-y\|_2}$: We prove a novel dimensionality reduction result
- Can greatly reduce the dimension of data using an oblivious = private map!
- New dimensionality reduction + prior work = faster prior work

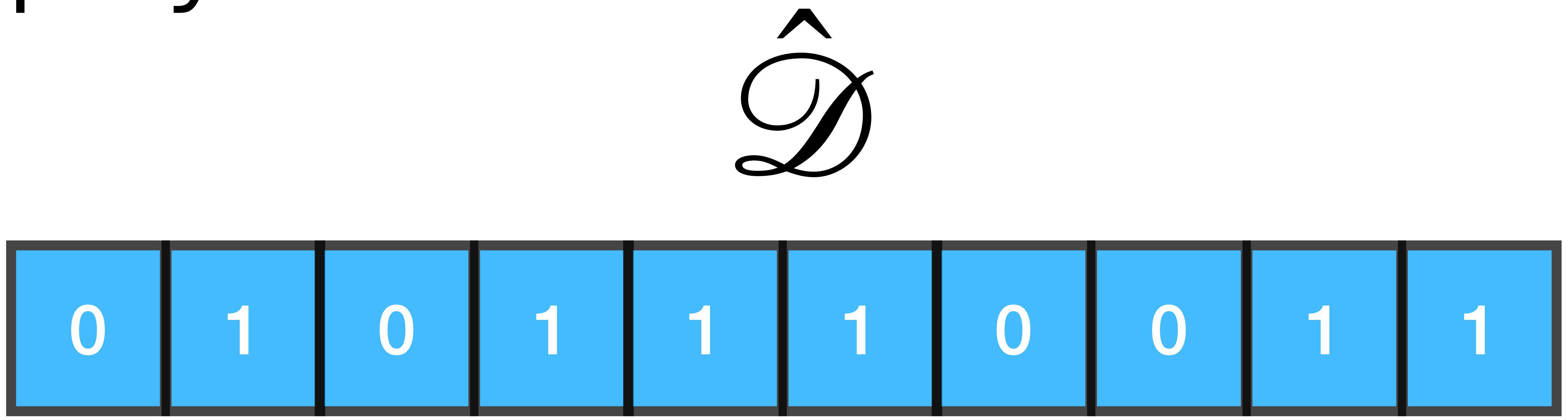
Sneak Preview

- $\frac{1}{1+\|x-y\|_2}$: Reduce this kernel to the case of $e^{-\|x-y\|_2}$
- Use function approx. theory to write $\frac{1}{1+z}$ as a <u>sublinear number</u> of terms that look like e^{-z}

Why? High-level template

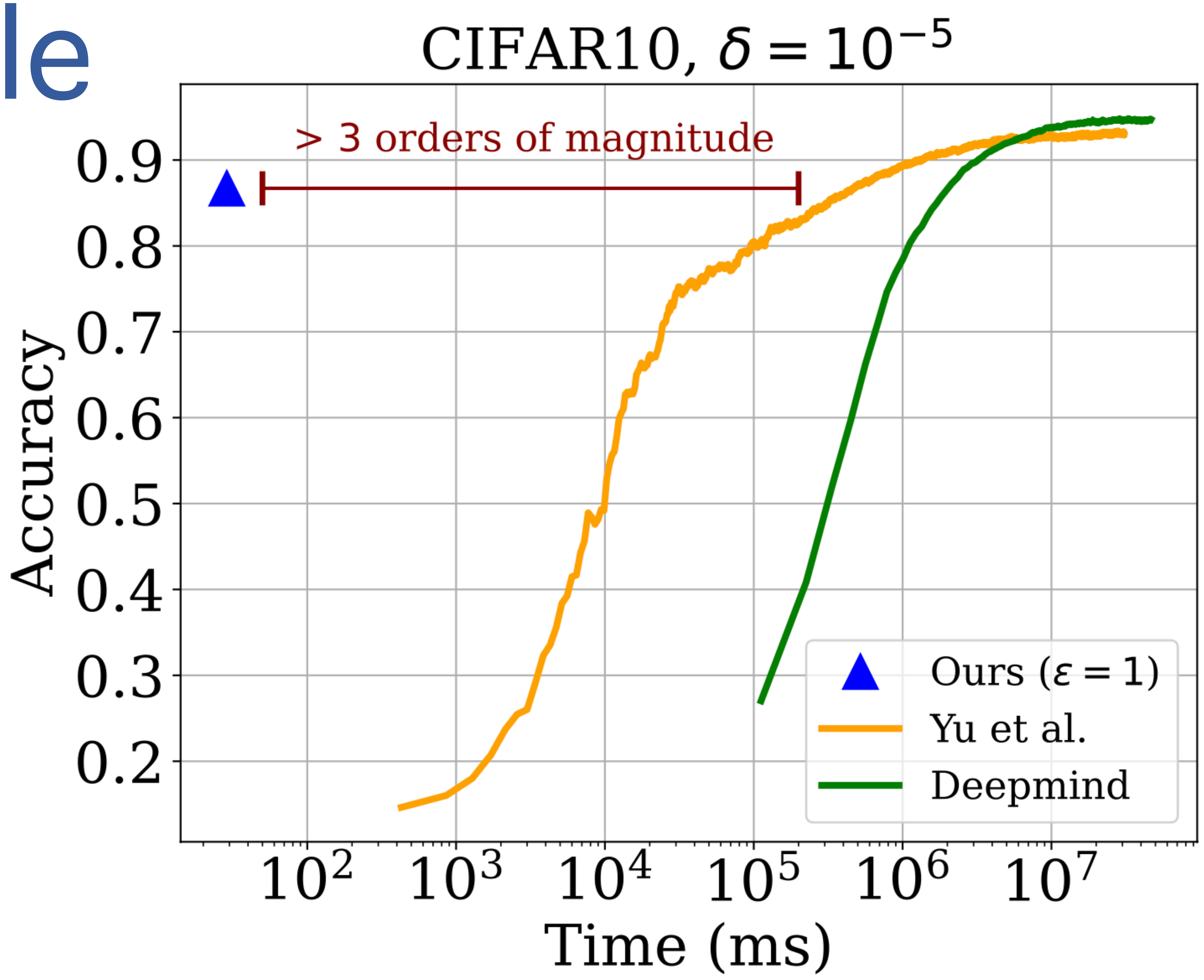
Different tools!

- Step 1: Create Data structure $\mathscr{D}$
  - $\mathscr{D}$ has sublinear query time
- Step 2: Add noise to every entry of $\mathscr{D}$
- Step 3: Use $\hat{\mathscr{D}}$ to compute query answer



Ideas are implementable

- Private image classification task
- Use private similarity data structures to classify
- Similar measured on embeddings of images
- Embeddings curated from a large public model



No deep learning required!

$> 10^3$x faster than SOTA (which uses deep learning magic) for comparable acc.

Thank you and don’t forget to stretch!