MACHINE LEARNING AND PRIVACY

Pei-Yuan Wu

National Taiwan University

Dept. Electrical Engineering

peiyuanwu@ntu.edu.tw

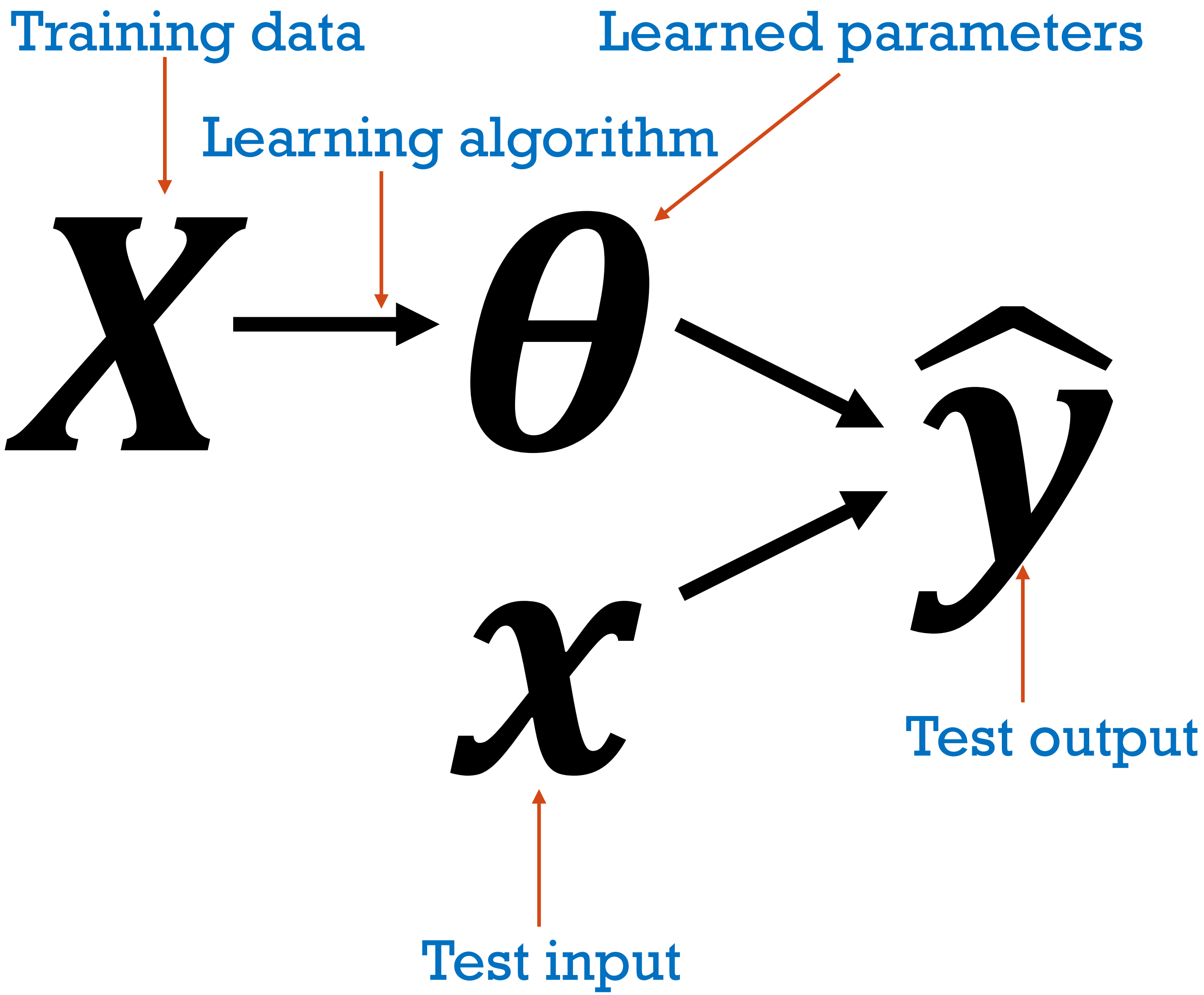
OUTLINE

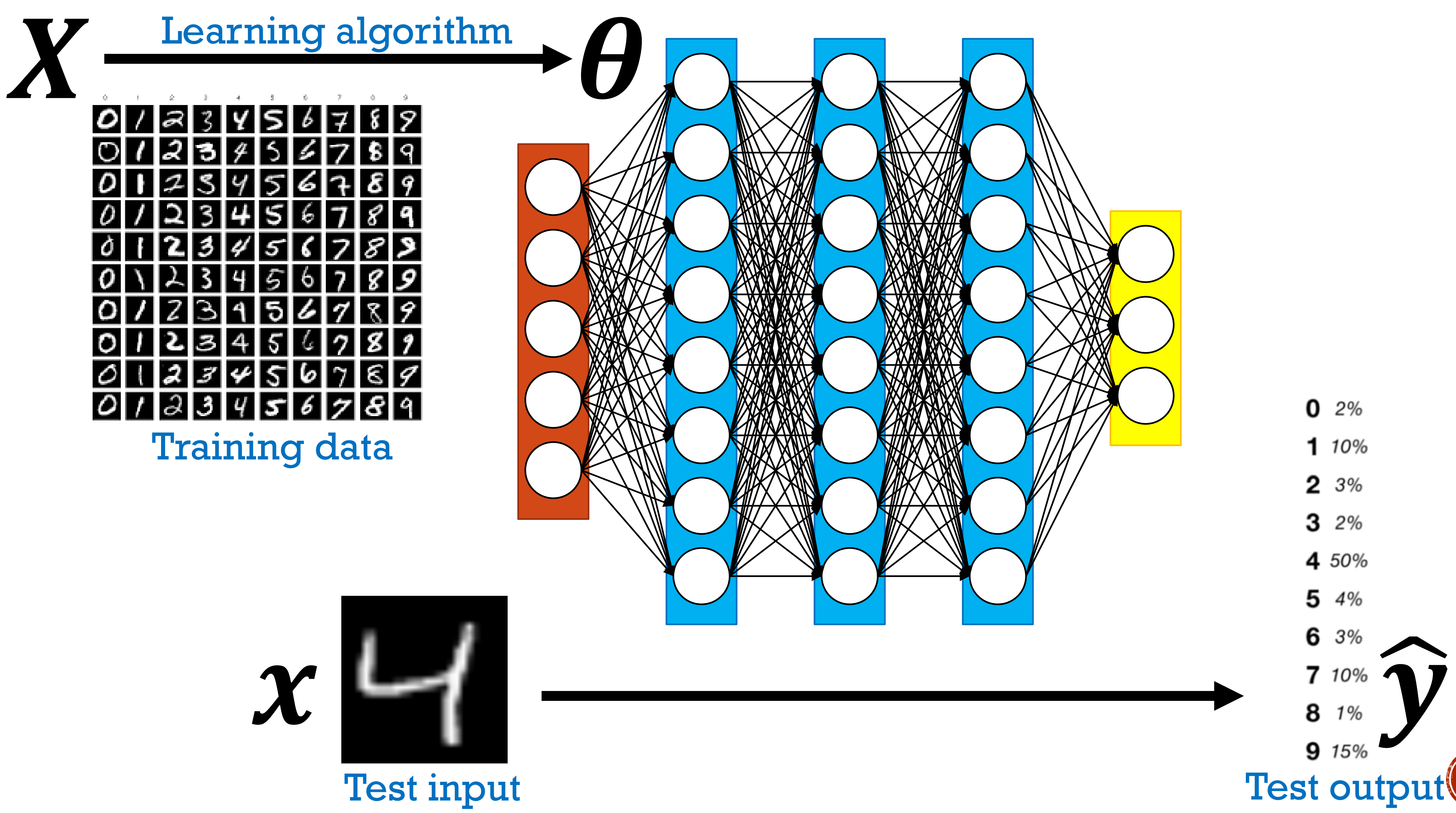
- Security against Attacks that use Machine Learning
  - Model Inversion Attack
  - Membership Inference Attack
  - Adversarial examples attack
  - Differential Privacy
  - Homomorphic Encryption
- Homomorphic encryption example
  - Privacy-Preserving Principle Component Analysis
- Secure Multi Party Computation
- Compressive Privacy Example
- Provable defense against adversarial example attack

3

SECURITY AGAINST ATTACKS THAT USE MACHINE LEARNING

MACHINE LEARNING PIPELINE

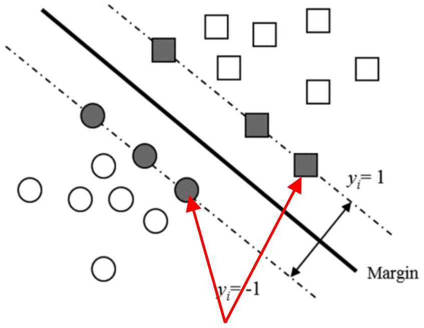



X
Learning algorithm
θ
Training data
x
Test input
0 2%
1 10%
2 3%
3 2%
4 50%
5 4%
6 3%
7 10%
8 1%
9 15%
$\hat{y}$
Test output

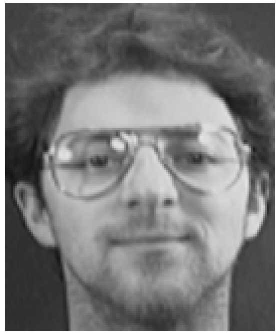
MODEL INVERSION ATTACK

- Adversary Target
  - Create dataset $\hat{X}$ resemble X used to create model θ
- Attack Scheme
  - Support Vector Machine model reveals training data as support vectors.
  - Exploit confidence information:
    - ✓ Many APIs reveal confidence values along with class predictions.
    - ✓ Find training samples for a peculiar class as the input yielding highest confidence on that peculiar class.
- Remedy
  - Only allow black-box access to the model.
  - Confidence values not revealed or rounded.

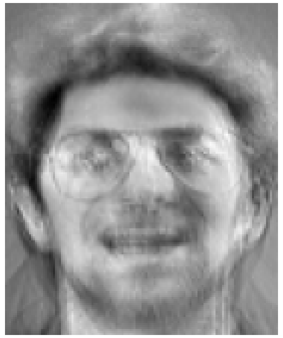
$$X \rightarrow \theta \rightarrow \hat{X}$$



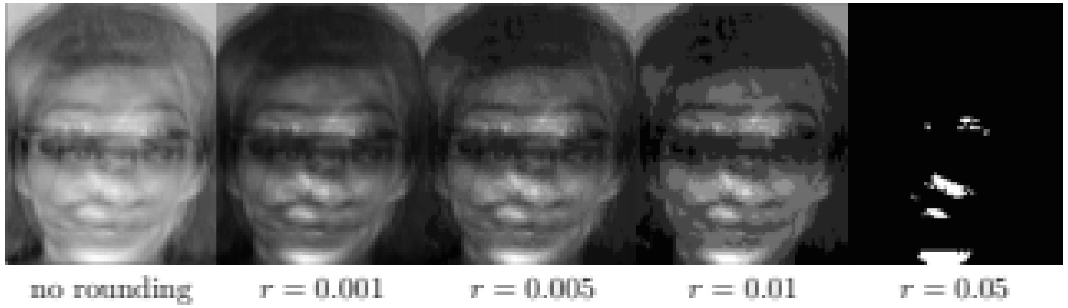
Support vectors



Face in training data



Reconstructed Face by model inversion attack

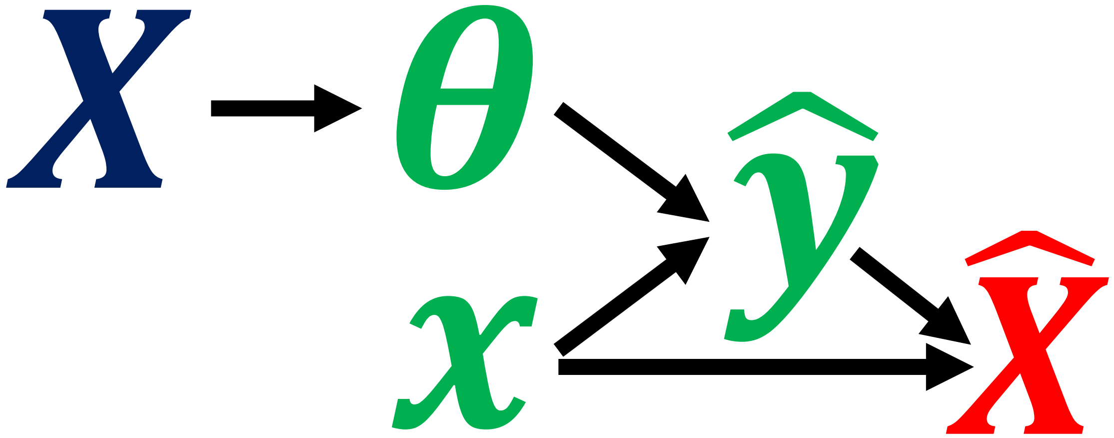


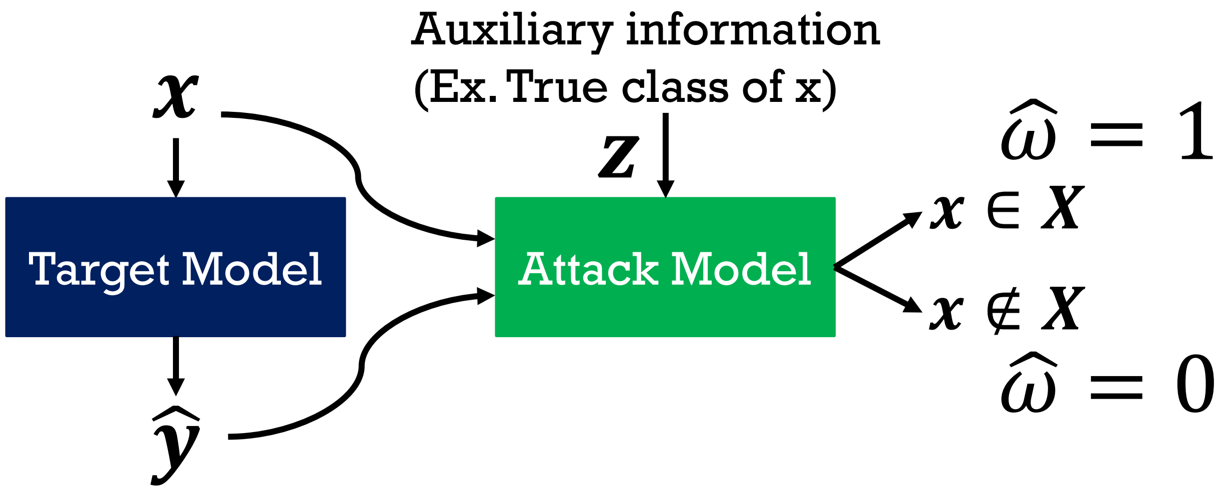
Black-box face reconstruction with rounding confidence

M. Fredrikson, S. Jha, T. Ristenpart, "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures", CCS 2015

MEMBERSHIP INFERENCE ATTACK

- **Adversary Target**
  - Determine if a sample is in X used to create model θ.
    - ✓ Example: Was Bob's record used to train ML model associated with AIDS?
- **Attack Scheme**
  - Exploits the difference between predictions made on training samples versus unseen samples.
- **Remedy**
  - Coarser precision of confidence values, only reveal top k confidence values.
  - Differential privacy.

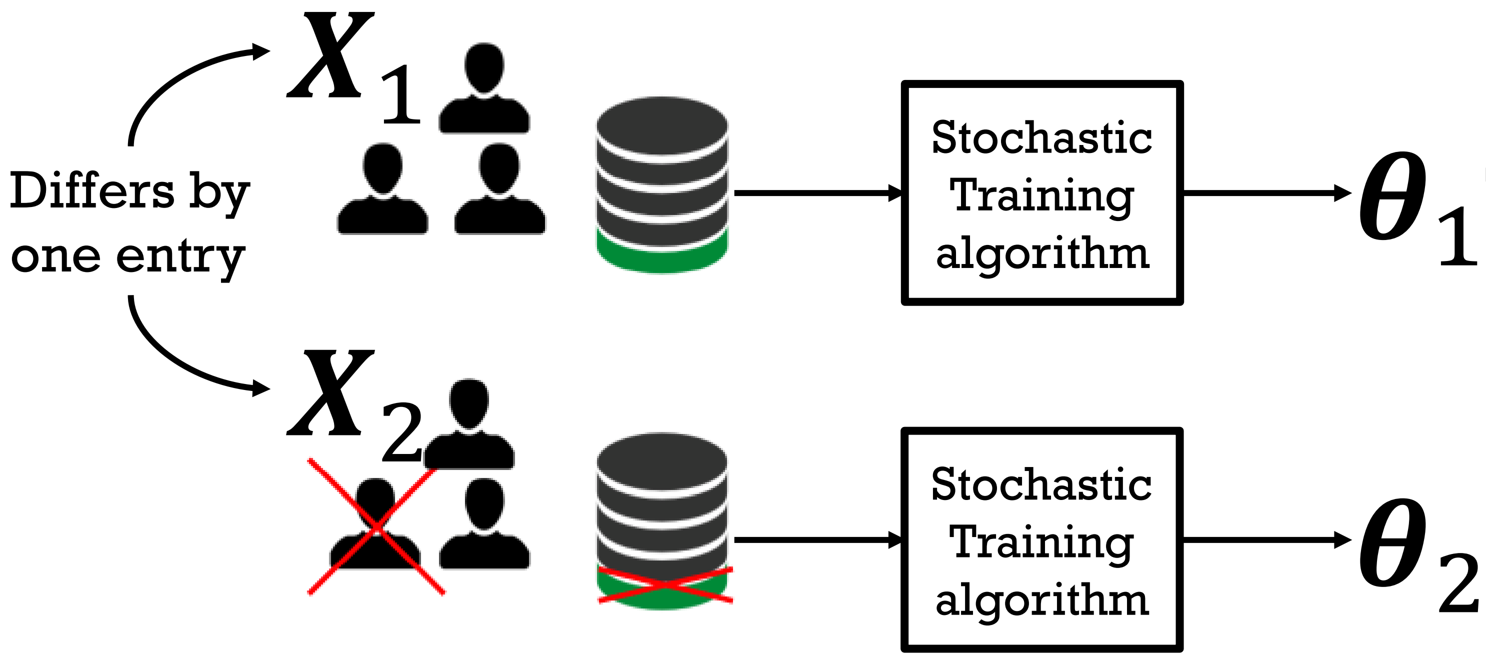




R. Shokri, M. Stronati, C. Song and V. Shmatikov, "Membership Inference Attacks Against Machine Learning Models," *2017 IEEE Symposium on Security and Privacy (SP)*, San Jose, CA, 2017, pp. 3-18.

(ε,δ)-DIFFERENTIAL PRIVACY

- The presence/absence of an entry in the training data has little effect on the trained parameters ➔ Difficult to perform membership inference attack



The (stochastic) training algorithm is (ε,δ)−differentially private if

$$\mathbb{P}[\boldsymbol{\theta}_1 \in S] \leq e^{\varepsilon}\mathbb{P}[\boldsymbol{\theta}_2 \in S] + \delta, \forall S$$

$$\mathbb{P}[\boldsymbol{\theta}_2 \in S] \leq e^{\varepsilon}\mathbb{P}[\boldsymbol{\theta}_1 \in S] + \delta, \forall S$$

If $\delta = 0$ ➔ $e^{-\varepsilon} \leq \frac{\mathbb{P}[\boldsymbol{\theta}_2 \in S]}{\mathbb{P}[\boldsymbol{\theta}_1 \in S]} \leq e^{\varepsilon}, \forall S$

ε is called privacy budget or privacy loss.

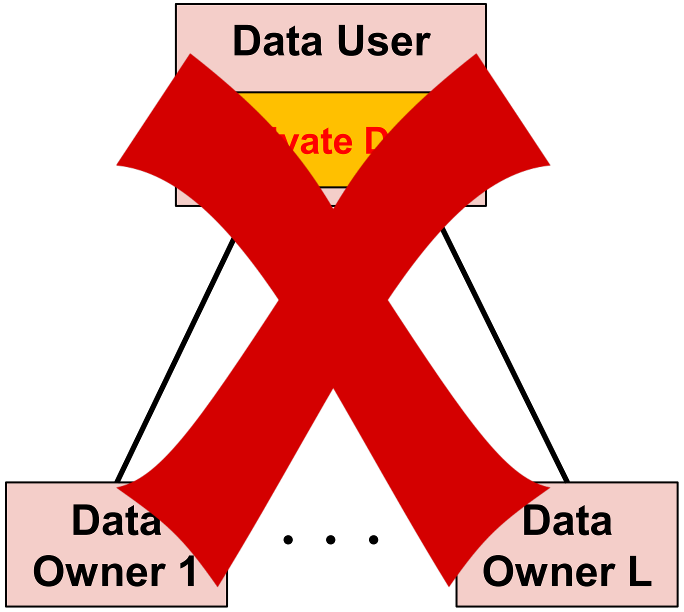
Small ε ➔ $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ has similar probability distributions

➔ Difficult to infer $\mathbf{X}$ from $\boldsymbol{\theta}$

Abadi et al 2016 "Deep Learning with Differential Privacy"
Papernot et al 2016 "Semi-supervised Knowledge Transfer for Deep Learning from Private Training Data"

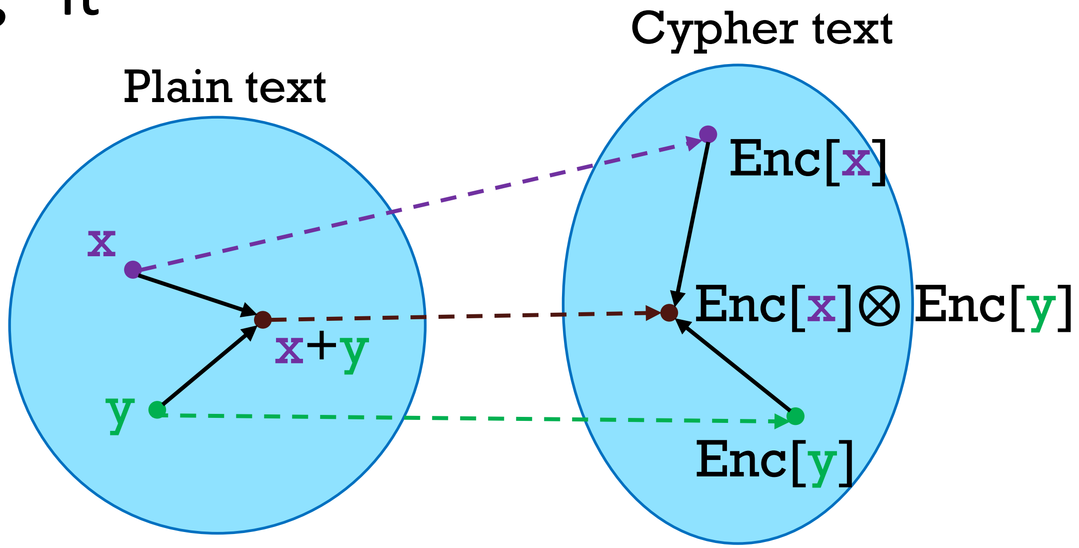
CRYPTOGRAPHIC APPROACHES TO DISTRIBUTED MACHINE LEARNING

- In collaborative learning involving ***multiple data owners***, we need a privacy-aware distributed approach:
  - The data user should build machine learning models, while the data owners keep their data ***private***.



- Special cryptographic approaches allow computing the data without “seeing” it
  - **Additive homomorphic encryption**: Addition on original data = Modular multiplication on encrypted data

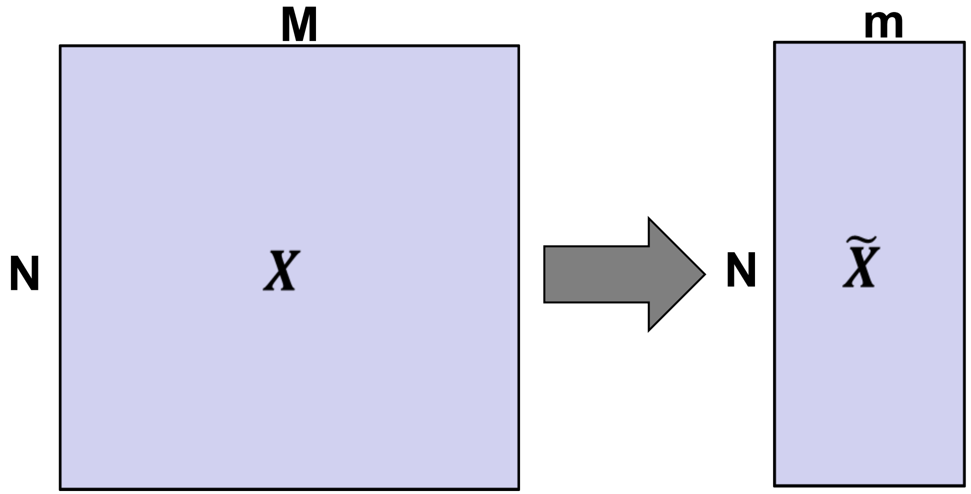
$$Enc[x+y] = Enc[x] \otimes Enc[y]$$



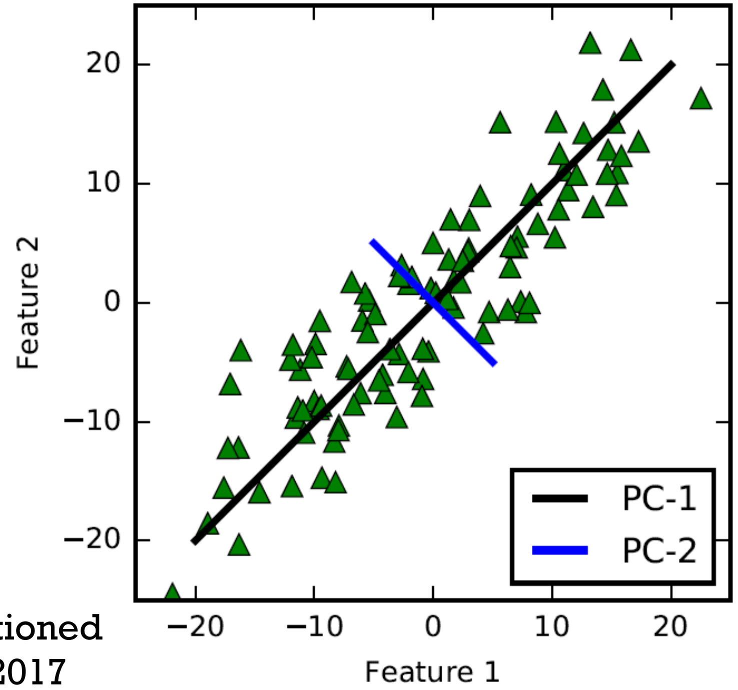
HOMOMORPHIC ENCRYPTION EXAMPLE: PRIVACY-PRESERVING PRINCIPLE COMPONENT ANALYSIS

PRIVACY-PRESERVING PRINCIPLE COMPONENT ANALYSIS

- Dimensionality Reduction:
  - Map the original high-dimensional data onto a lower-dimensional subspace.
  - Reduces the noise, compresses the data, decreases the computational cost, and prevents over-fitting.



- Principle Component Analysis:
  - Target: Find the best sub-space that preserves most of the variance in the original data
  - The principal axes are orthogonal, uncorrelated and ordered by how much variability they retain.



M. Al-Rubaie, P. Y. Wu, J. M. Chang and S. Y. Kung, "Privacy-preserving PCA on horizontally-partitioned data," 2017 IEEE Conference on Dependable and Secure Computing, 280, Taipei, Taiwan, Aug. 2017

CENTRALIZED PCA

- PCA performs Eigen-decomposition of the center-adjusted scatter matrix

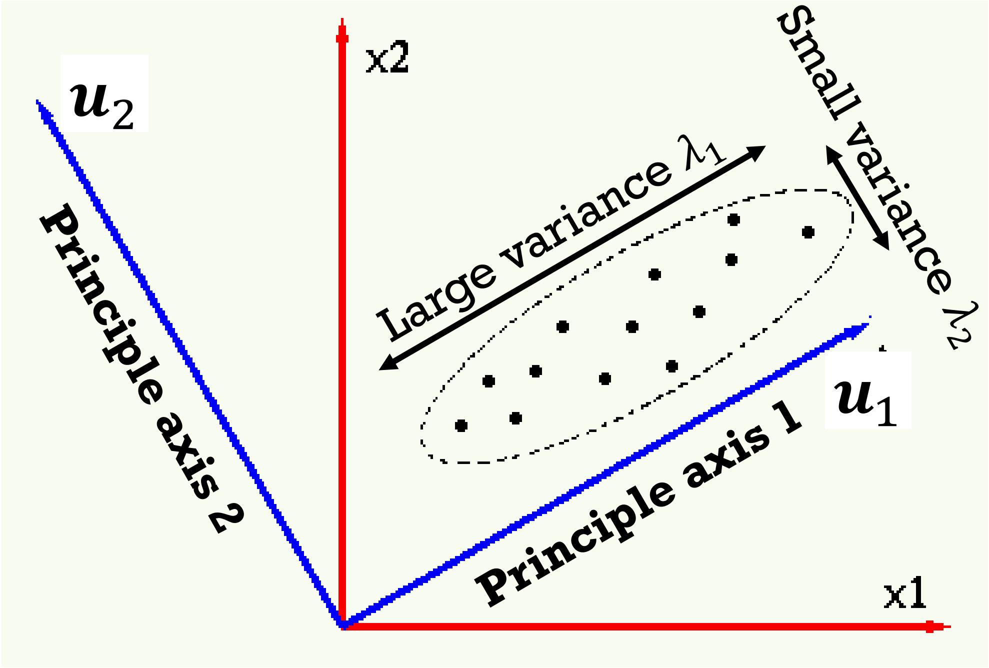
$$\boldsymbol{S} = \sum_{i=1}^{N} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T = \boldsymbol{U\Lambda U}^T$$

$\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_M]$ is a unitary matrix

Principle axes

$$\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & \\ \vdots & & \ddots & \vdots \\ 0 & & \cdots & \lambda_M \end{bmatrix}$$ is a diagonal matrix

Eigen Values



- In centralized PCA, the original data is needed to compute the scatter matrix (suitable only for ***single data owner***)
- Privacy-aware distributed approach is needed for ***multiple data owners.***

DISTRIBUTED SCATTER MATRIX COMPUTATION

- Rewrite scatter matrix as follows:

$$\boldsymbol{S} = \sum_{i=1}^{N} (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^T = \sum_{i=1}^{N} \boldsymbol{x}_i {\boldsymbol{x}_i}^T - N\boldsymbol{\mu}\boldsymbol{\mu}^T = \boldsymbol{R} - \frac{1}{N}\mathbf{v}\mathbf{v}^T$$

where $\boldsymbol{R} = \sum_{i=1}^{N} \boldsymbol{x}_i {\boldsymbol{x}_i}^T, \mathbf{v} = \sum_{i=1}^{N} \boldsymbol{x}_i$

- Each data owner computes the share: $DS^{\ell} = \{\boldsymbol{R}_\ell, \mathbf{v}_\ell, N_\ell\}$

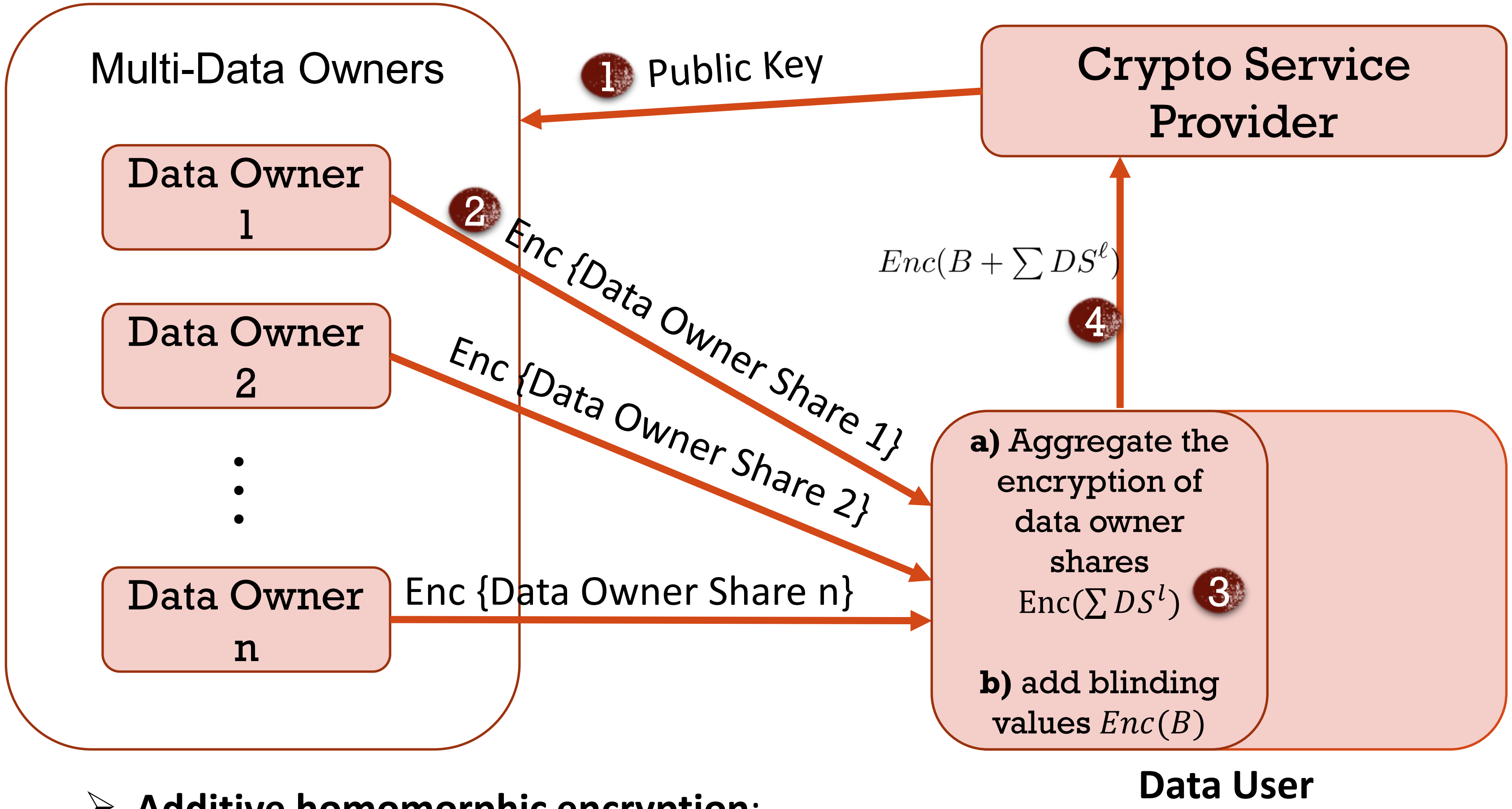
$$\boldsymbol{R}_\ell = \sum_{i \in P_\ell} \boldsymbol{x}_i {\boldsymbol{x}_i}^T, \mathbf{v}_\ell = \sum_{i \in P_\ell} \boldsymbol{x}_i, N_\ell = |P_\ell|$$

P_ℓ is the set of training samples from data owner ℓ

- The scatter matrix is aggregated from all data owners:

$$\boldsymbol{S} = \boldsymbol{R} - \frac{1}{N}\mathbf{v}\mathbf{v}^T, \boldsymbol{R} = \sum_\ell \boldsymbol{R}_\ell, \mathbf{v} = \sum_\ell \mathbf{v}_\ell, N = \sum_\ell N_\ell$$

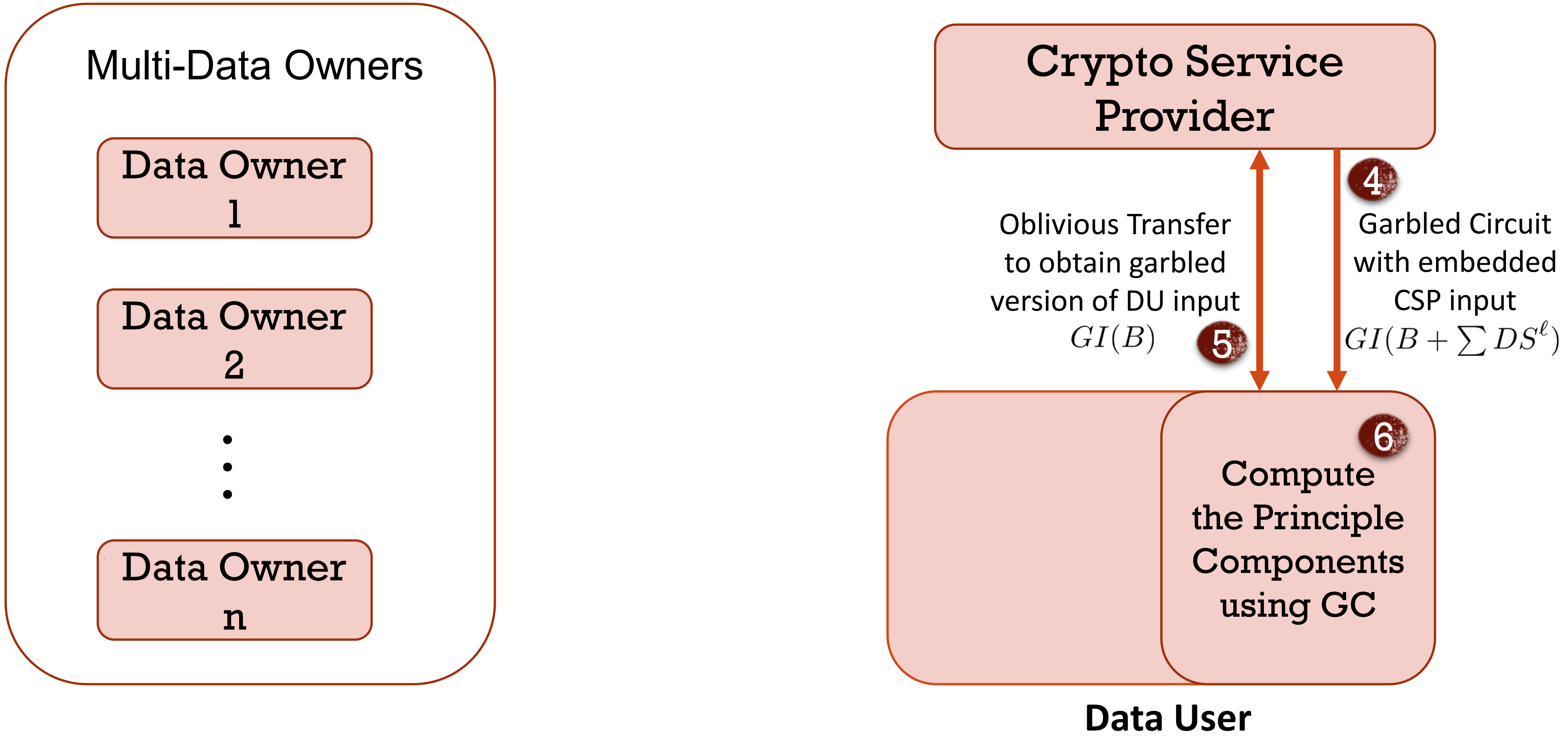
ARCHITECTURE-SCATTER MATRIX COMPUTATION



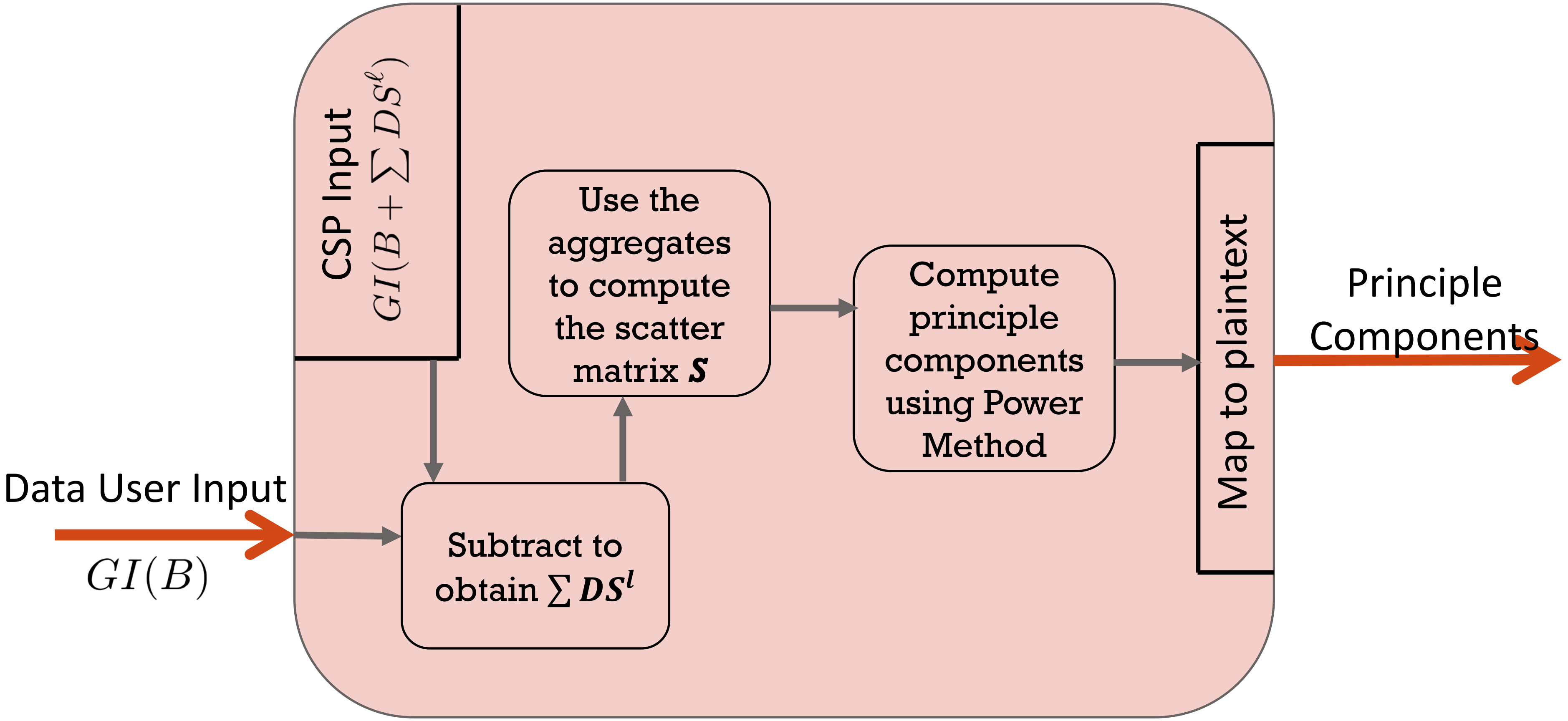
- **Additive homomorphic encryption**:
  Addition on original data = Modular multiplication on encrypted data

$$Enc[x+y] = Enc[x] \otimes Enc[y]$$

ARCHITECTURE-PRINCIPLE AXIS COMPUTATION

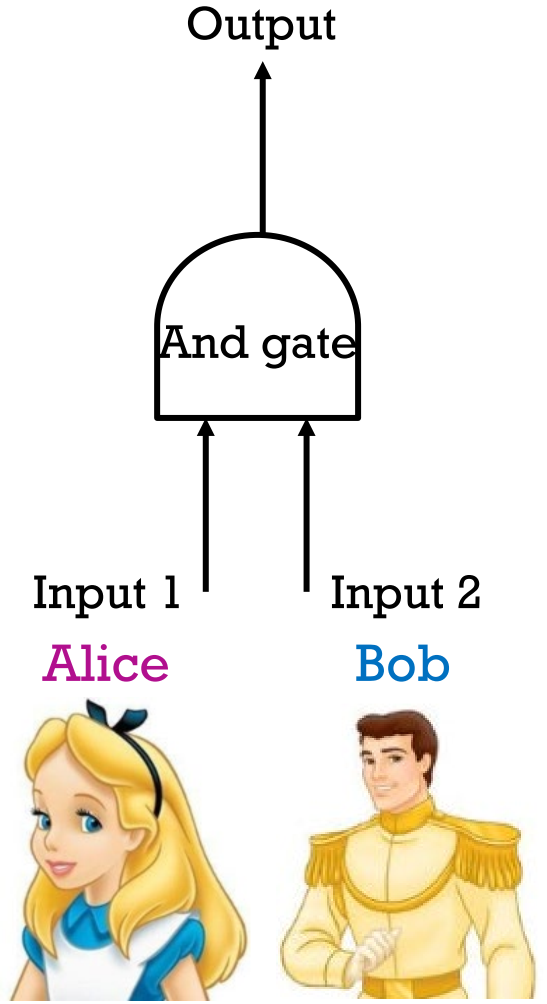


ARCHITECTURE-PRINCIPLE AXIS COMPUTATION (CONT'D)

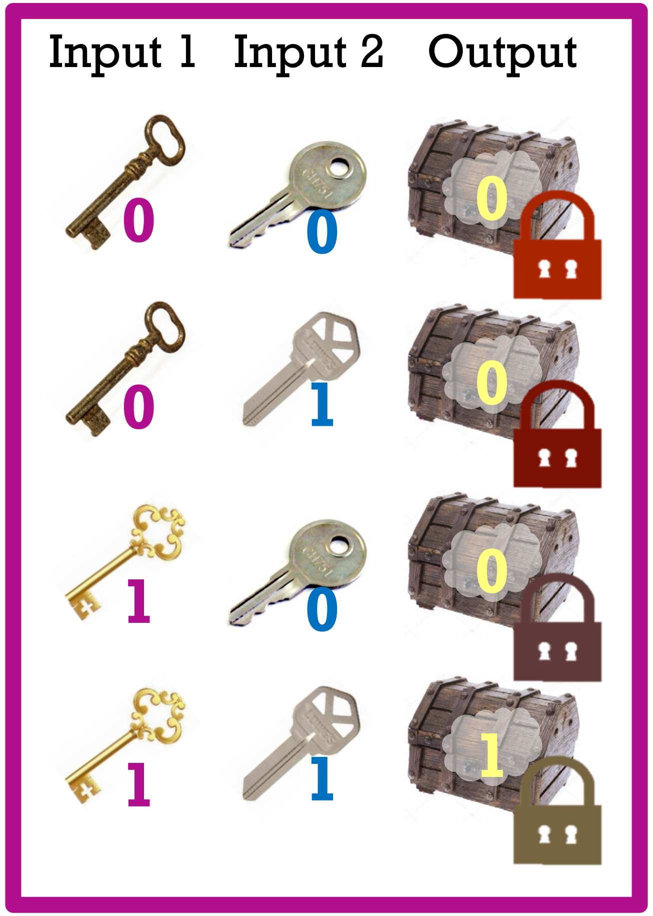


GARBLED CIRCUIT

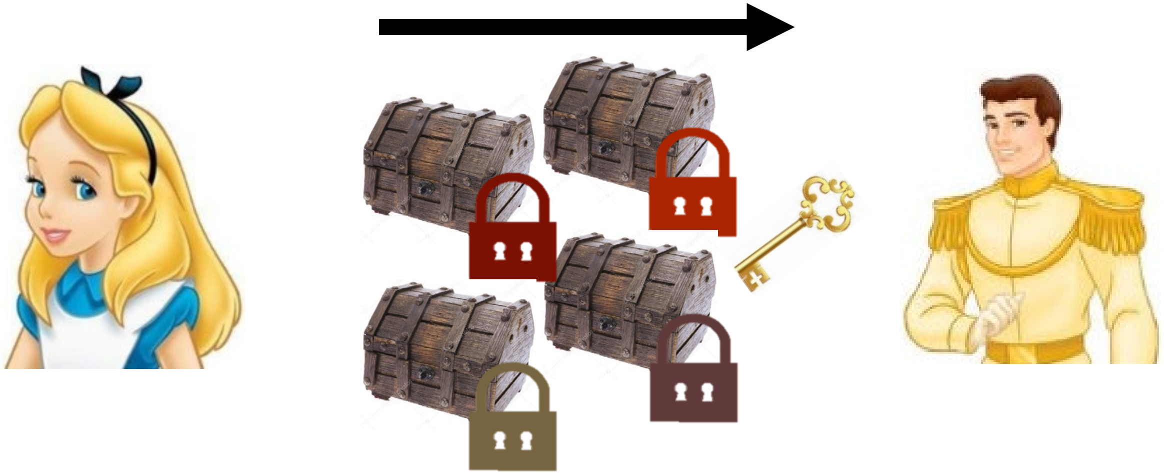
Alice and Bob want to compute AND gate together, but not revealing their own inputs



Alice makes the keys and locks, and lock the cases



Alice shuffles the cases and gives them to Bob the locked cases, as well as her selected key



Bob picks his key from Alice without her knowing which (through **oblivious transfer**)

Bob tries to open all cases with keys he possesses.

Bob sends the only successfully decrypted message to Alice

COMPUTATION COSTS

$Enc[DS^{\ell}]$

PCA Eigen Decomposition with Garbled circuit

Dataset	Features	Classes	Avg. DO time	Avg. DU Coll. / Add time	CSP Dec. time	DU PCA Comp. time
Diabetes	8	2	0.63 sec	10 ms	0.67 sec	28.3 sec (8)
Breast Cancer	10	2	0.93 sec	11 ms	1 sec	49.6 sec (8)
Australian	14	2	1.7 sec	12 ms	1.8 sec	119.1 sec (8)
German	24	2	5 sec	17 ms	5 sec	16.3 min (15)
Ionosphere	34	2	9.8 sec	24 ms	9.9 sec	43.2 min (15)
SensIT Acoustic	50	3	22.5 sec	40 ms	22.7 sec	126.7 min (15)

$$Enc\left[B + \sum_{\ell} DS^{\ell}\right] = Enc[B] \otimes_{\ell} Enc[DS^{\ell}]$$

$$Dec\left[Enc\left[B + \sum_{\ell} DS^{\ell}\right]\right] = B + \sum_{\ell} DS^{\ell}$$

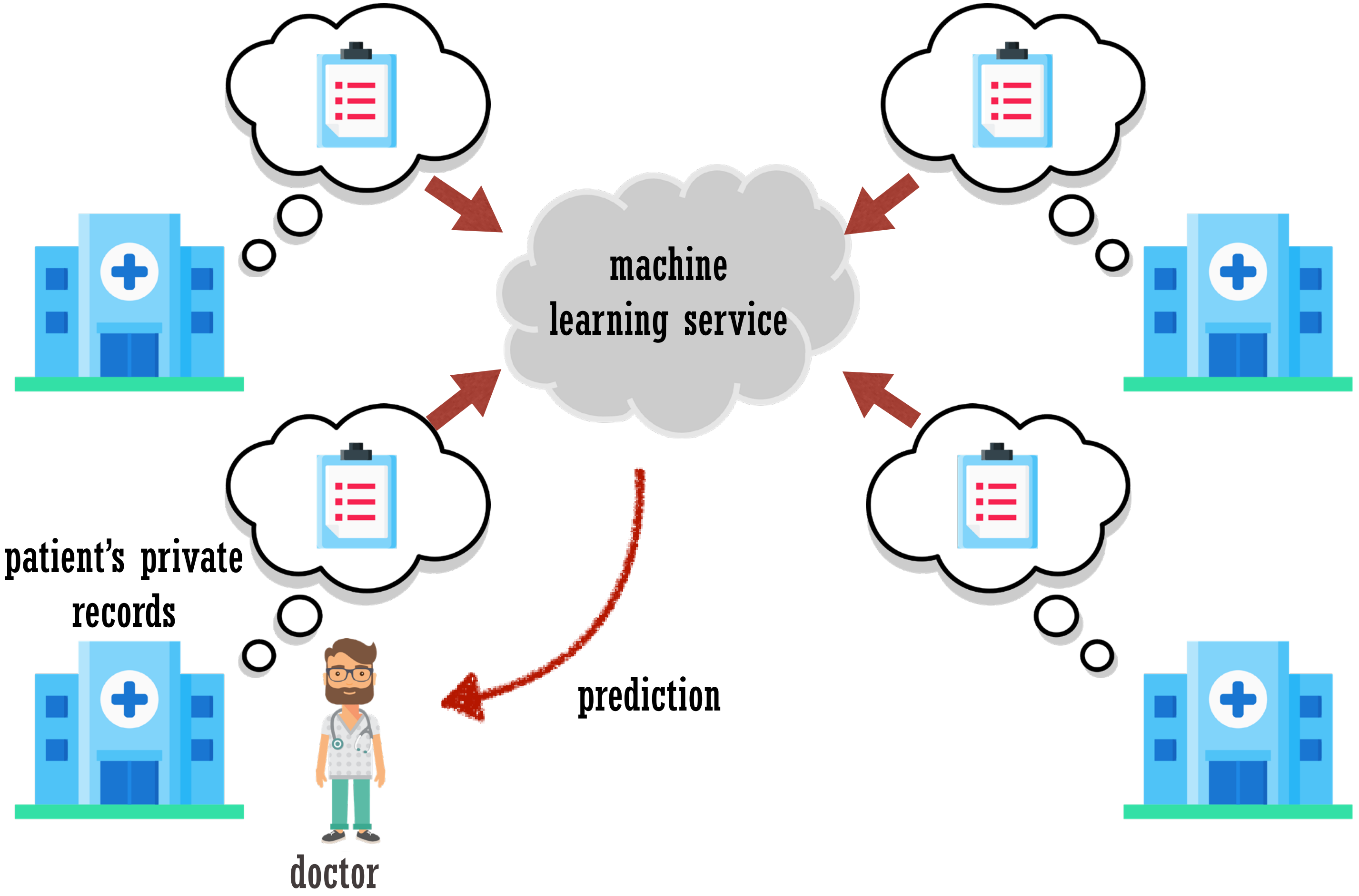
CPU: i5-6600K @ 3.5GHz
RAM: 8 GB
Paillier's Cryptosystem with 1024 bits key length
No multi-threading was used

20

SECURE MULTI-PARTY COMPUTATION

SECURE MULTI-PARTY COMPUTATION (SMPC)

- **Goal:** For multiple parties to jointly compute a function over their inputs, while each party keeping their inputs private from other parties.
- **Example:** Suppose we wish to train a diagnostic model by data from multiple hospitals, while each hospital wishes to keep their own data private.



SMPC EXAMPLE

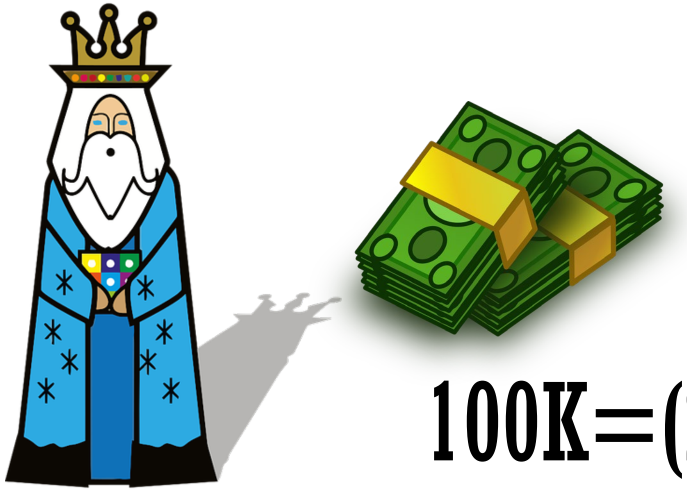
Suppose three people have monthly salary 5K, 100K, 22K, respectively. How can they compute the average salary while keeping their own salary in secret?

SMPC EXAMPLE

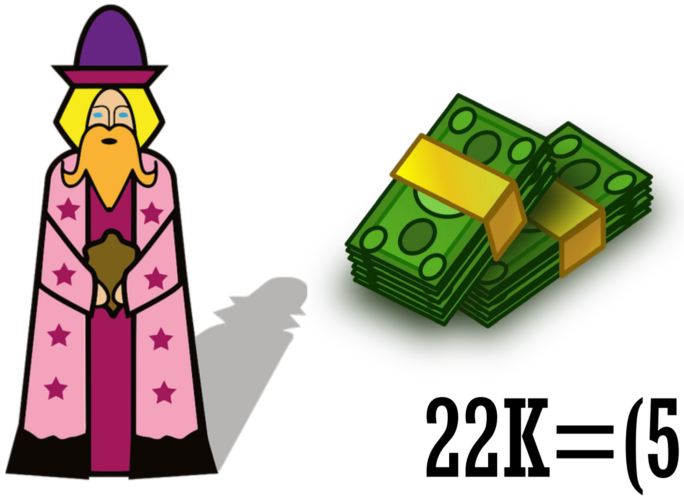
Suppose three people have monthly salary 5K, 100K, 22K, respectively. How can they compute the average salary while keeping their own salary in secret?

Step 1: Every one split their own salary into three parts.

5K=(1K,1K,3K)



100K=(22K,5K,73K)

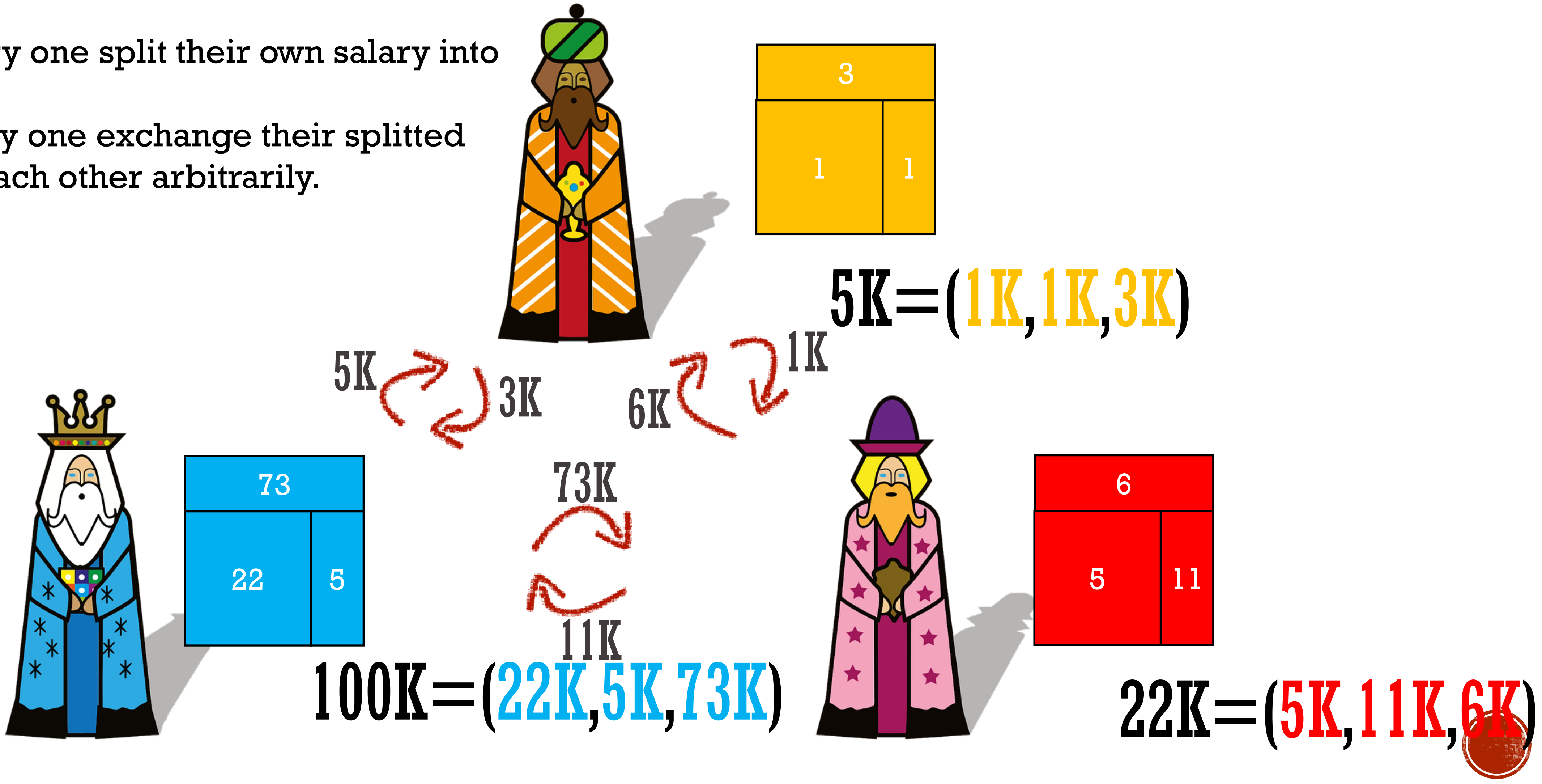


22K=(5K,11K,6K)

SMPC EXAMPLE

Suppose three people have monthly salary 5K, 100K, 22K, respectively. How can they compute the average salary while keeping their own salary in secret?

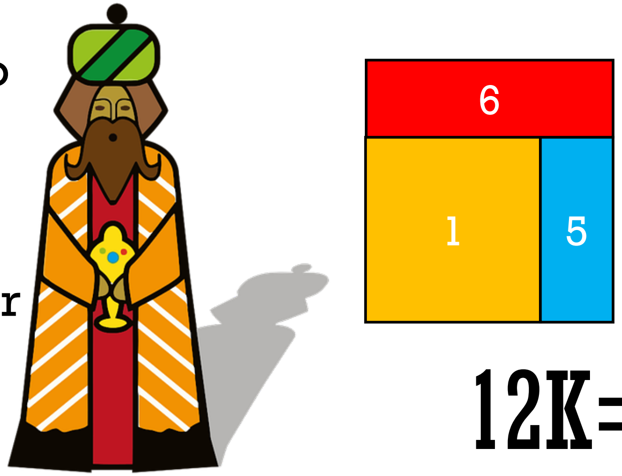
Step 1: Every one split their own salary into three parts.
Step 2: Every one exchange their splitted parts with each other arbitrarily.



SMPC EXAMPLE

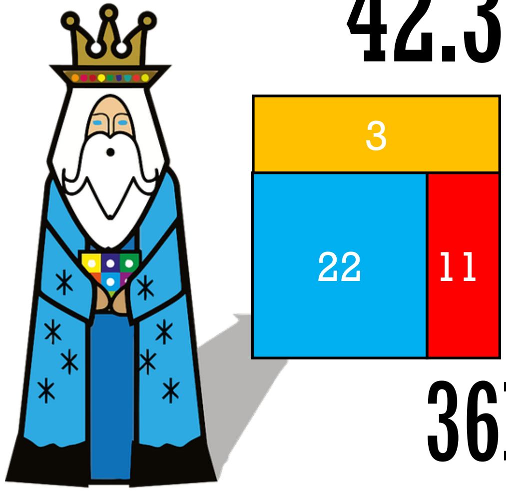
Suppose three people have monthly salary 5K, 100K, 22K, respectively. How can they compute the average salary while keeping their own salary in secret?

Step 1: Every one split their own salary into three parts.
Step 2: Every one exchange their splitted parts with each other arbitrarily.
Step 3: Every one computes the sum of their received part, then compute total average.

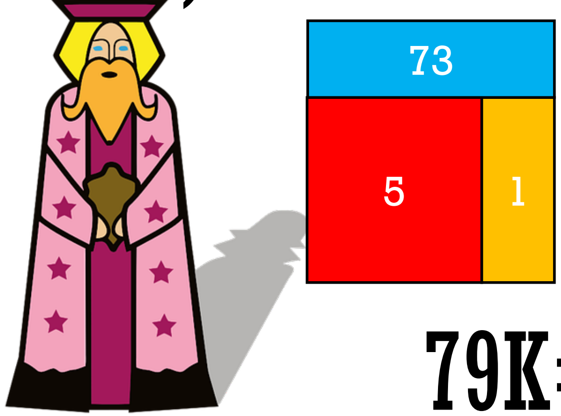


12K=1K+5K+6K

42.3K=(36K+12K+79K)/3



36K=22K+11K+3K



79K=5K+1K+73K

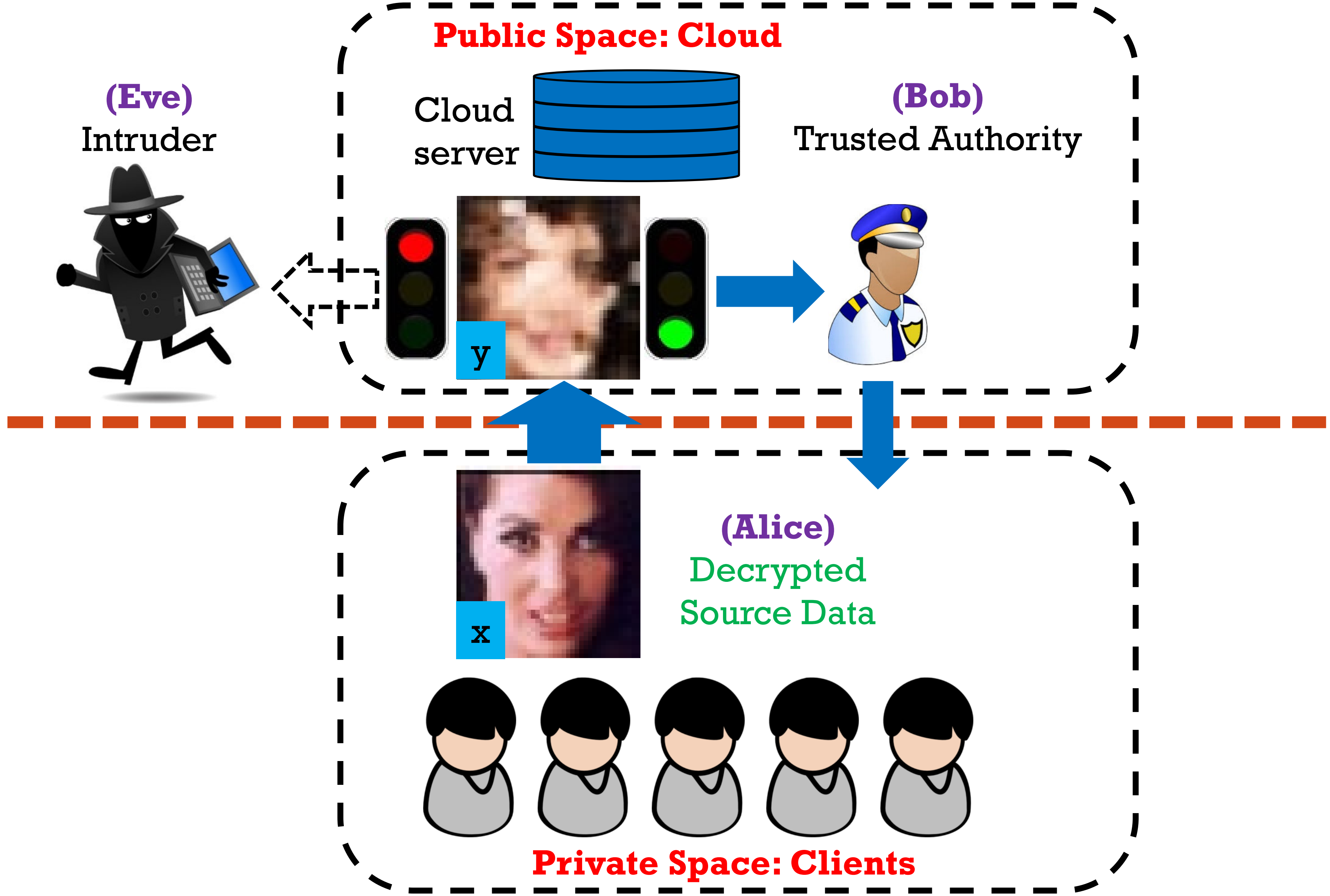
SMPC FOR MACHINE LEARNING

- **SecureML:** Apply SMPC to linear regression, logistic regression and neural network training using the stochastic gradient descent method.
- **CrypTen:** Open source framework built on PyTorch. Developed by Facebook.
- Computation cost is a BIG issue:
  Upon training a 3-layer fully-connected neural network:
  - Plain model: 9 ms per epoch
  - SecureML: 4 mins per epoch
  - CrypTen: 15 mins per epoch

27

COMPRESSIVE PRIVACY EXAMPLE

Compressive Privacy Paradigm

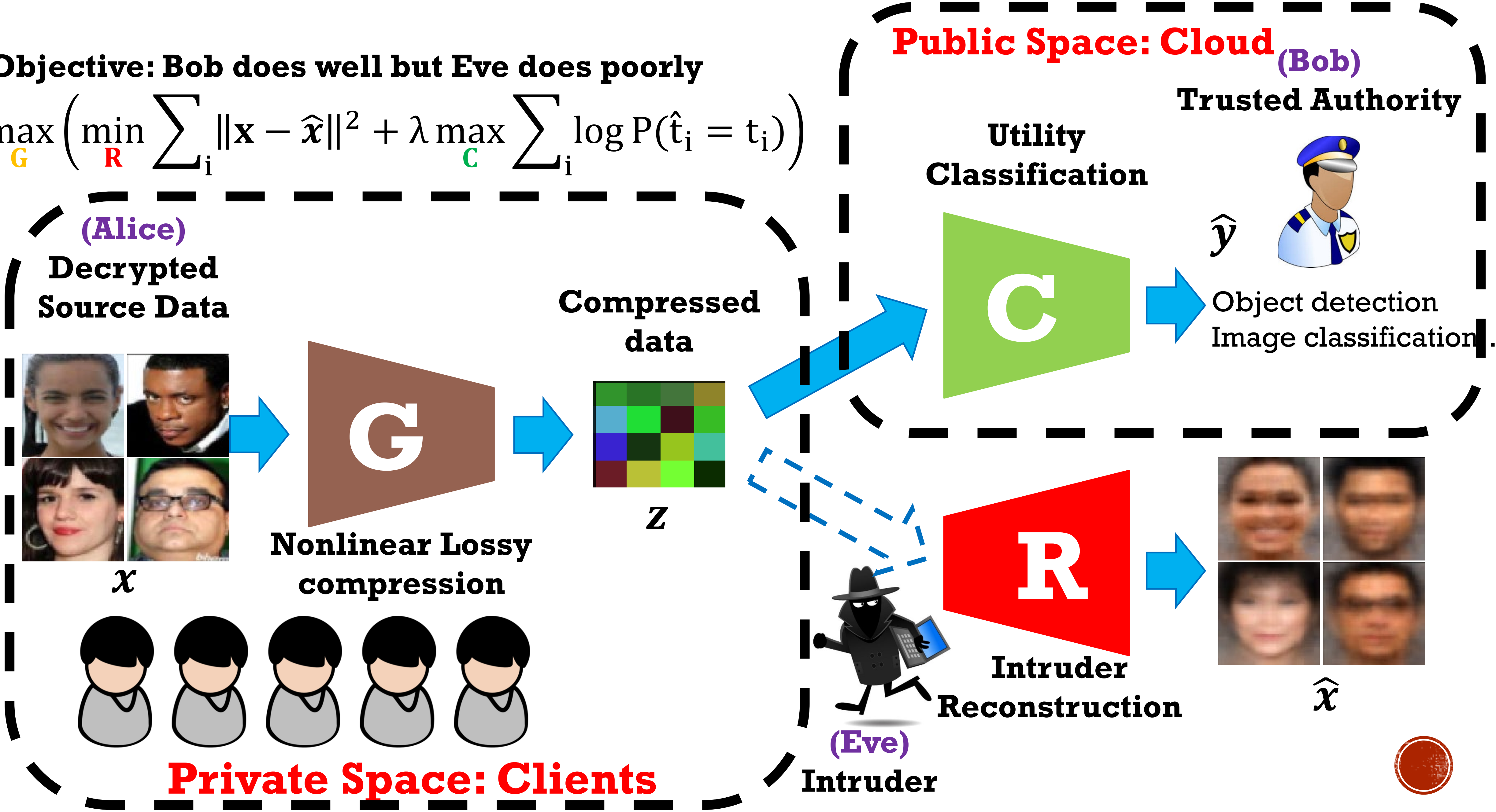


Kung, S. Y. (2018). A Compressive Privacy approach to Generalized Information Bottleneck and Privacy Funnel problems. *Journal of the Franklin Institute*, *355*(4), 1846-1872.

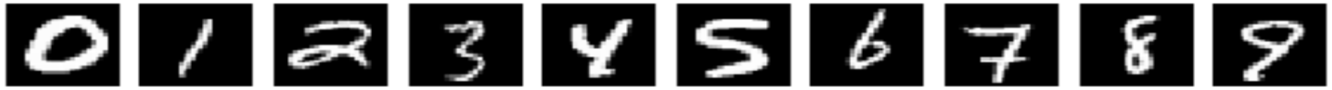
Compressive Privacy Generative Adversarial Network

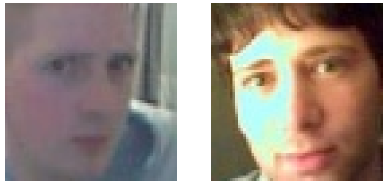
Objective: Bob does well but Eve does poorly

$$\max_{G}\left(\min_{R}\sum_{i}\|\mathbf{x}-\hat{x}\|^{2}+\lambda\max_{C}\sum_{i}\log P(\hat{t}_{i}=t_{i})\right)$$



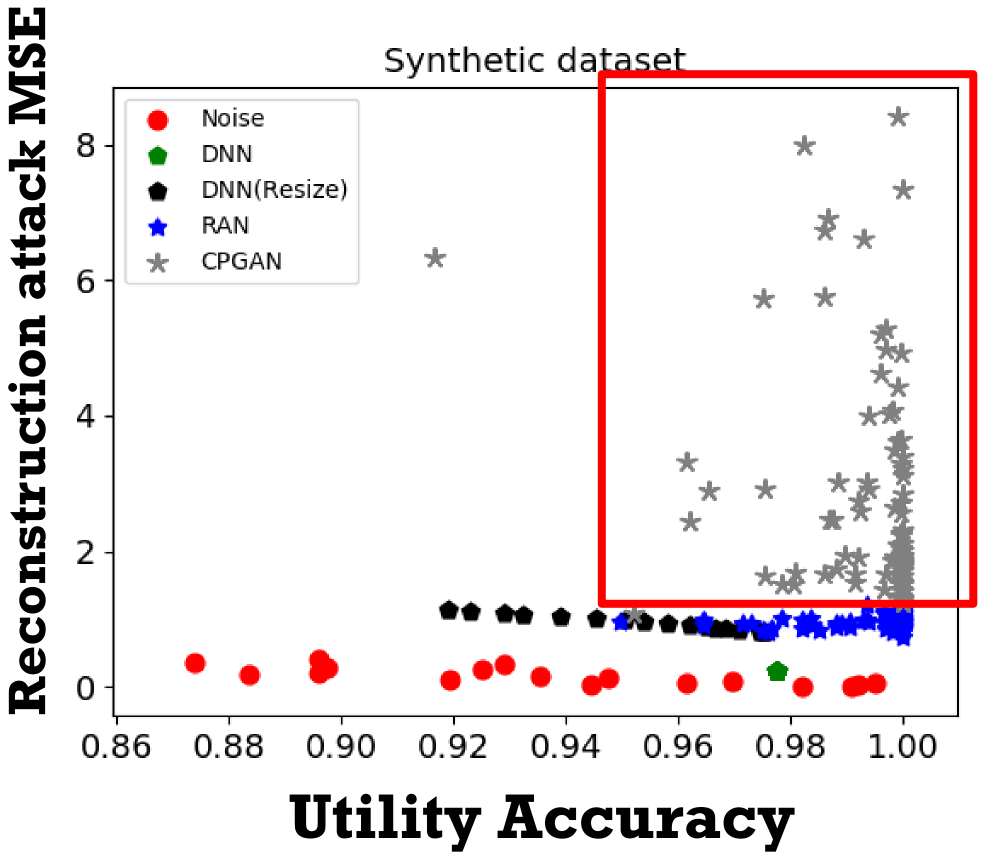
CPGAN FOR BENCHMARK DATASET

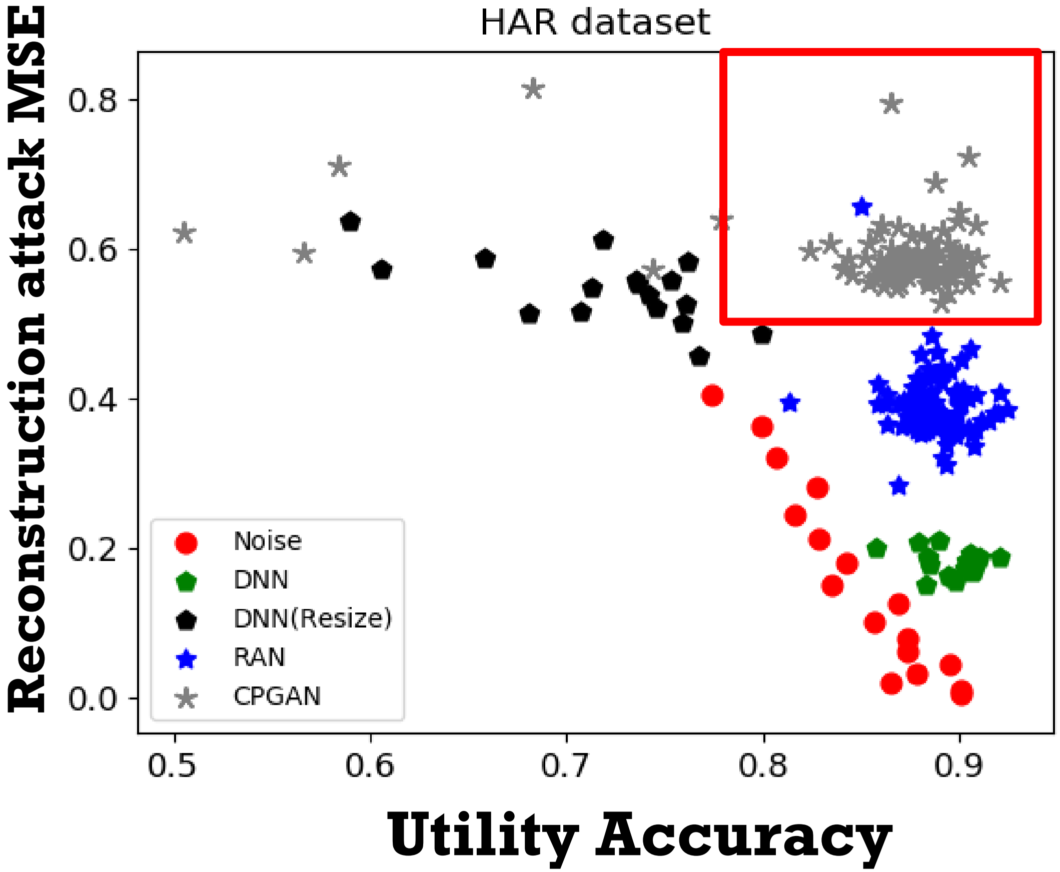
- **Synthetic dataset:**
  - Sampled from Gaussian mixture data model with binary class.
  - Training/testing samples: 20K/2K
- **MNIST:**
  - Training/testing samples: 55000/10000
  - Examples: 
- **UCI Human activity recognition (HAR) dataset**
  - Given the time-series sensor record from ten identities.
  - Six activities: walking, sitting, standing e.t.c.
- **Genki-4K dataset:**
  - Face images with 400 sample. Detect the expression of this image.
  - Example:



COMPARISON WITH PREVIOUS WORKS

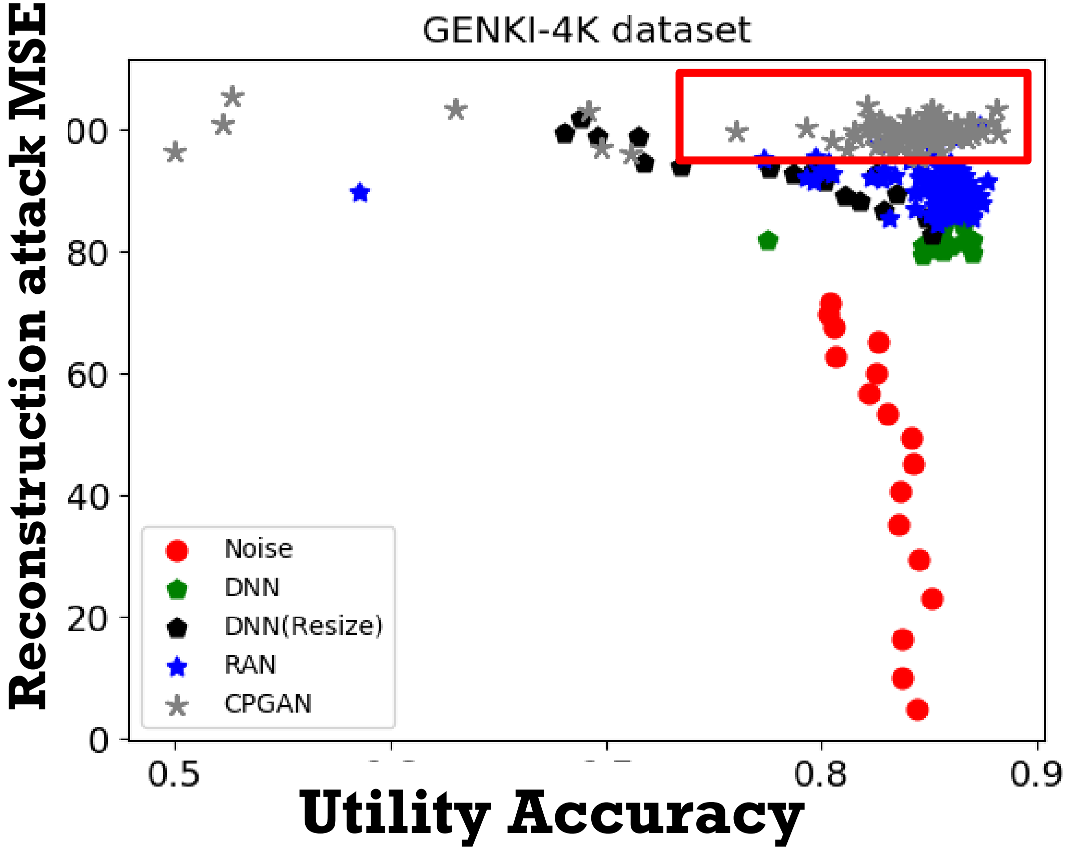
CPGAN (Gray star) outperforms other methods on privacy perspective, but slightly drops (less than 1%) the utility accuracy (trade-off).

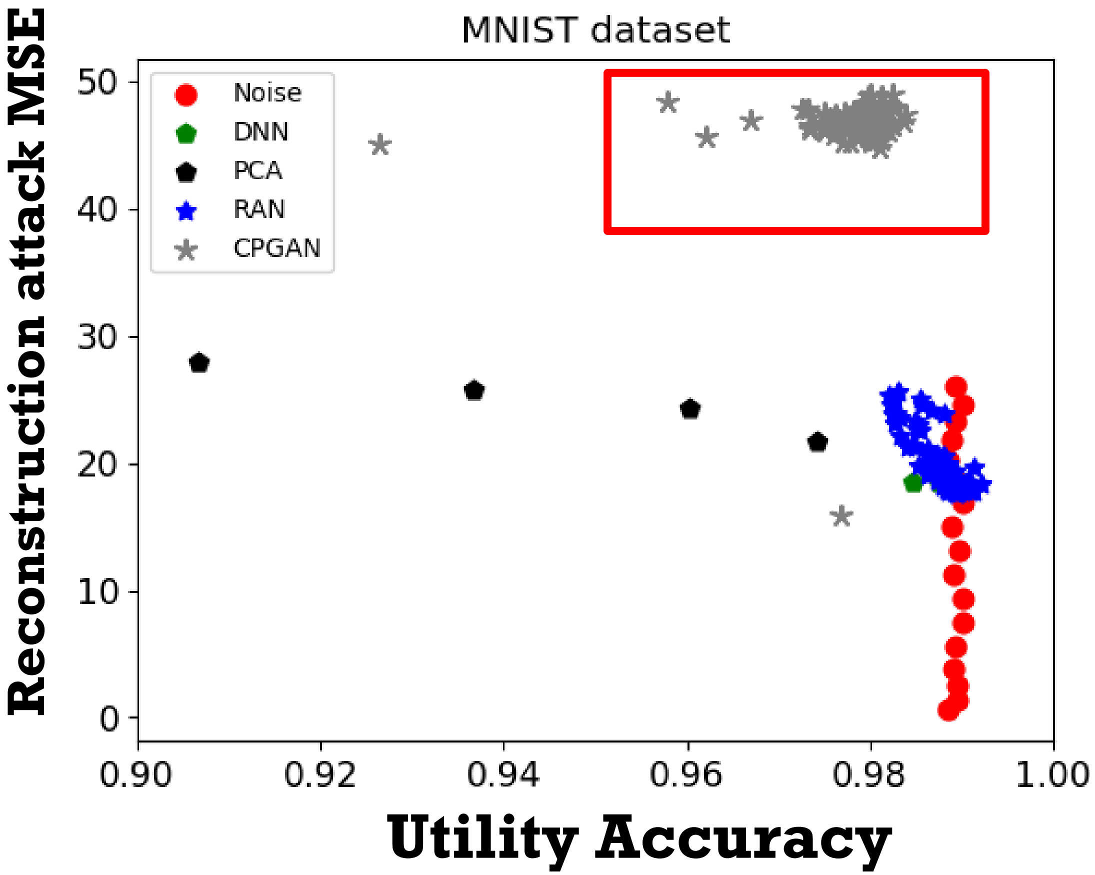




COMPARISON WITH PREVIOUS WORKS (CONT.)

CPGAN (Gray star) outperforms other methods on privacy perspective, but slightly drops (less than 1%) the utility accuracy (trade-off).



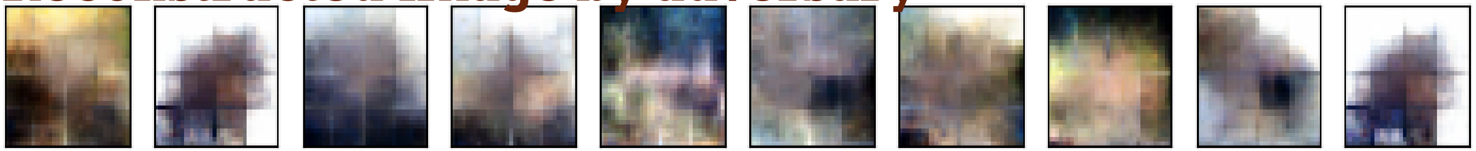


RESULTS ON CIFAR-10 AND SVHN

CIFAR-10: ***Adversary reconstruction***

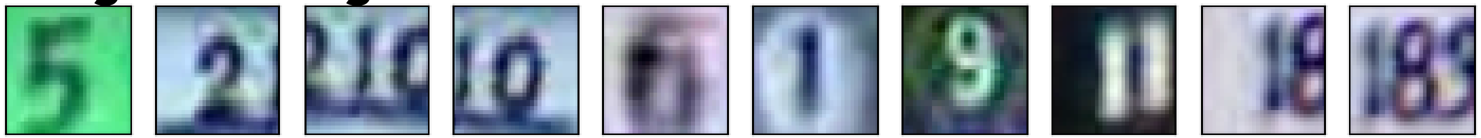
Original images

Reconstructed image by adversary

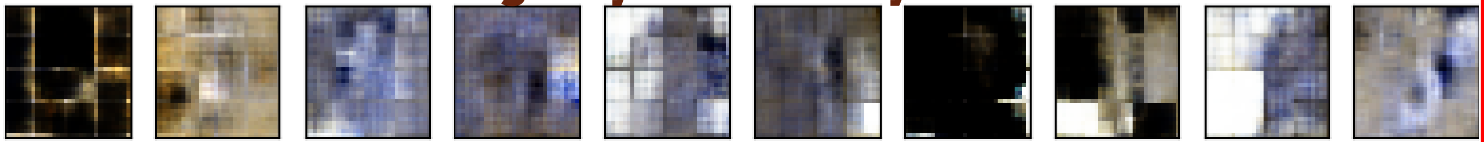


SVHN:

Original images



Reconstructed image by adversary



Classify by *original* image: 96.5%

Classify by *compressed* data: 93.9%

Classify by *original* image: 98.6%

Classify by *compressed* data: 97.7%

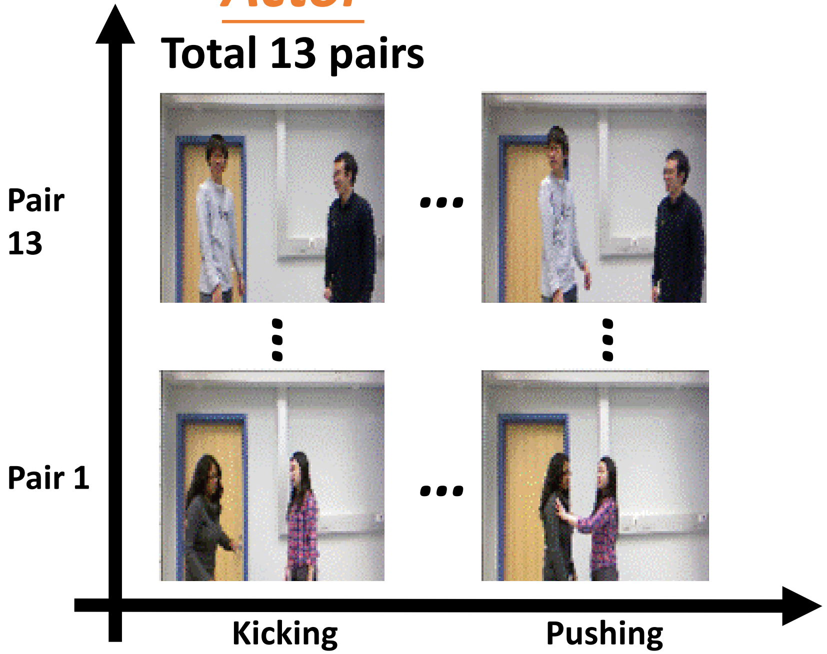
CPGAN defends the reconstruction attack while achieving satisfactory utility performance

Compressive Privacy on Videos

Identity Privacy Preserving

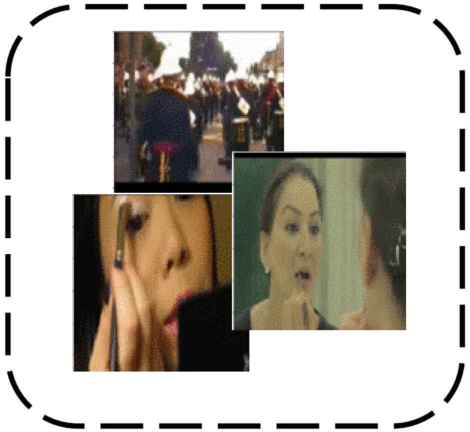
<u>SBU Kinect Interaction Dataset:</u>
Training / Testing: 346 / 36

Actor
Total 13 pairs

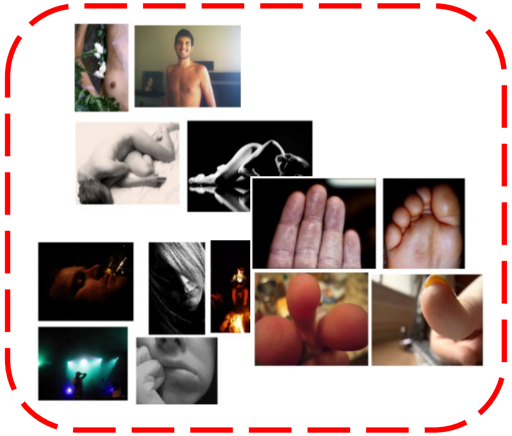


Action
Total 8 actions

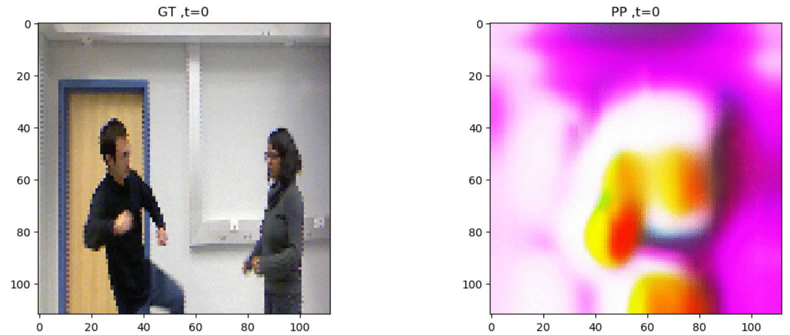
<u>UCF101 Dataset</u>
Training / Testing: 9537 / 3783
Source: YouTube



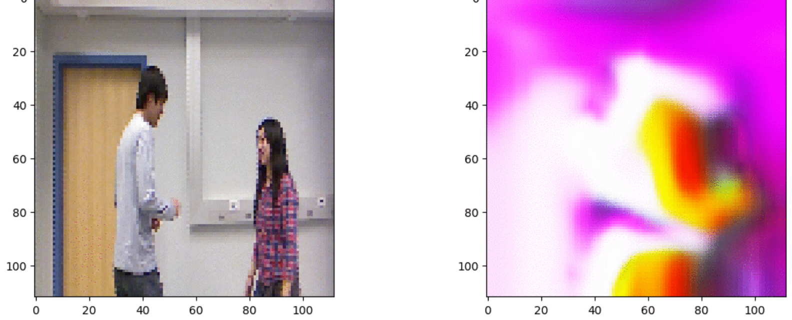
Action
Total 101 actions
Jump, walking, playing violin, …



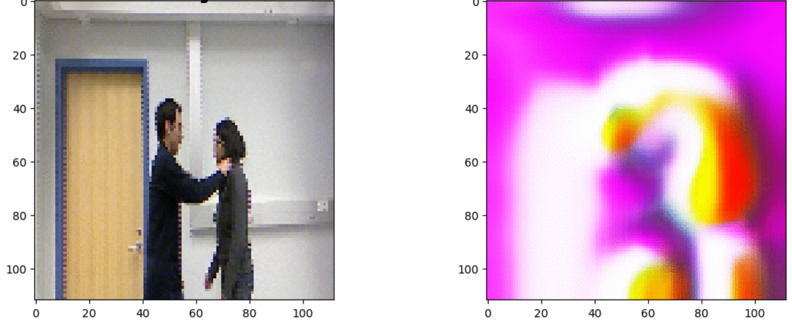
Attribute
Total 7 attributes
face, race, nudity, …



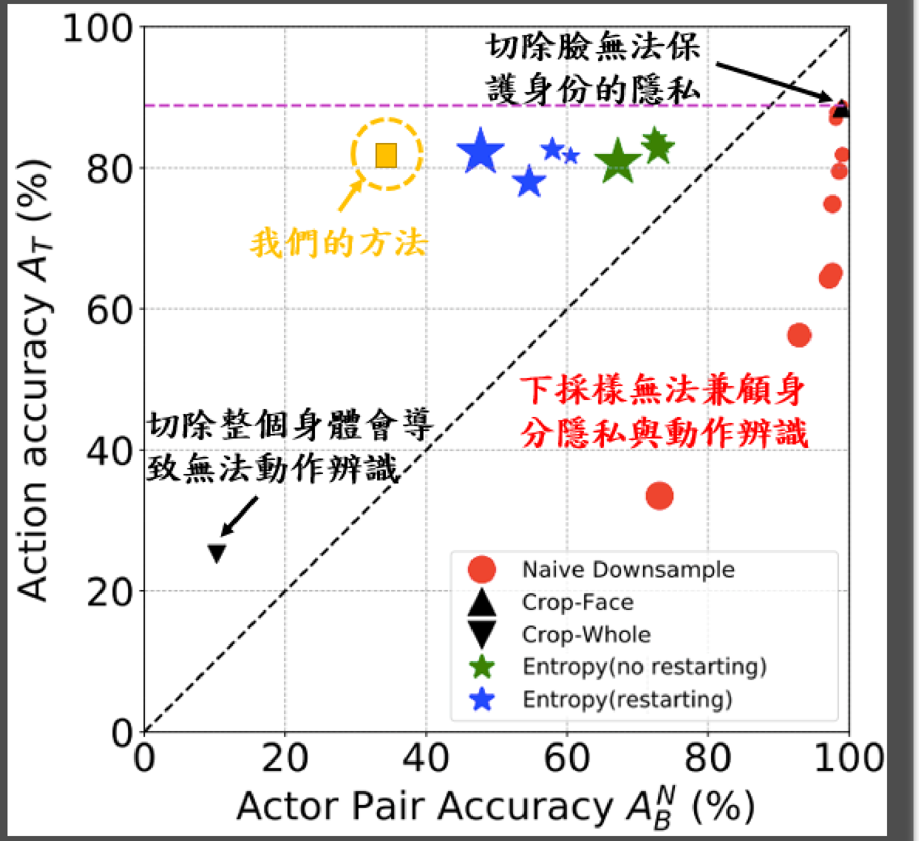
Same Action

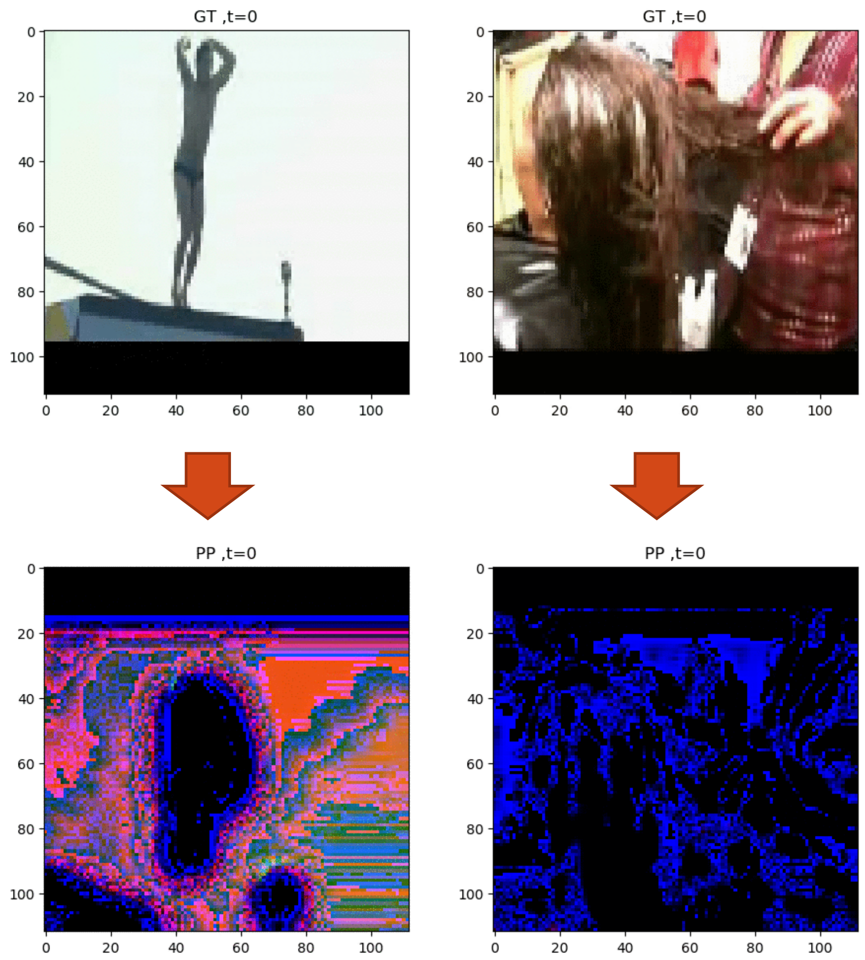


Same Identity

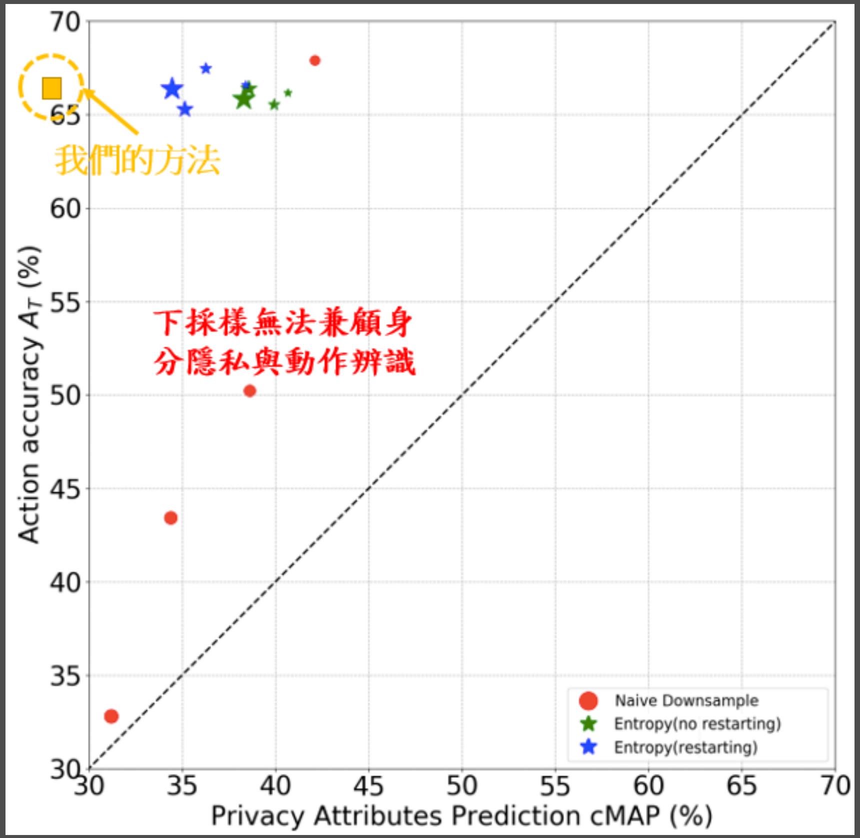


SBU Kinect Interaction Dataset
Result





UCF101 Dataset Result

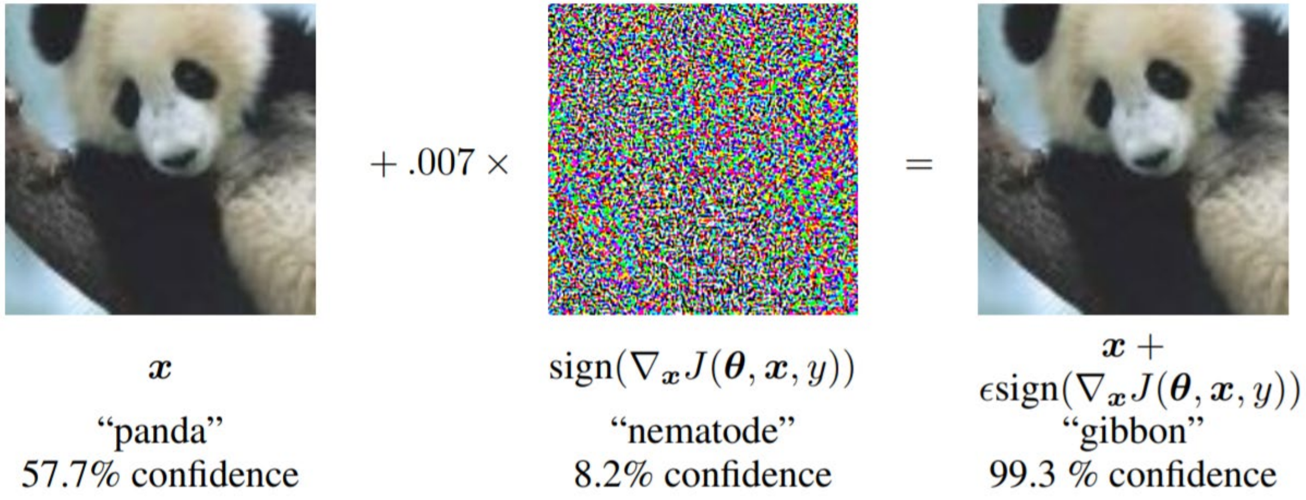


41

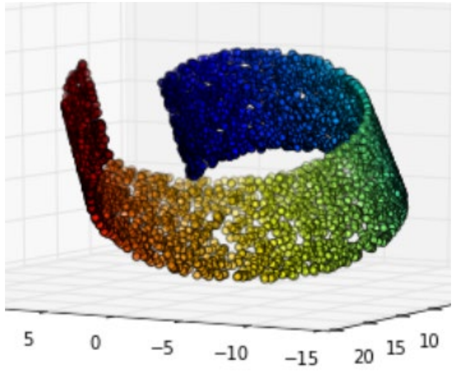
ADVERSARIAL EXAMPLE ATTACK

ADVERSARIAL EXAMPLES

- Adversary Target
  - ➢ Perturbs a sample x to $x + \Delta x$ to fool model $\boldsymbol{\theta}$.
- Possible Causes
  - ➢ Curse of dimensionality
    - ✓ Adversary generates unseen images off the manifold
  - ➢ ReLU activation function
    - ✓ ReLU network is piecewise linear, so carefully designed small noise can aggregate to alter decision.
- Action Items
  - ➢ Detection and removal of adversarial examples
  - ➢ Robust design of neural network.

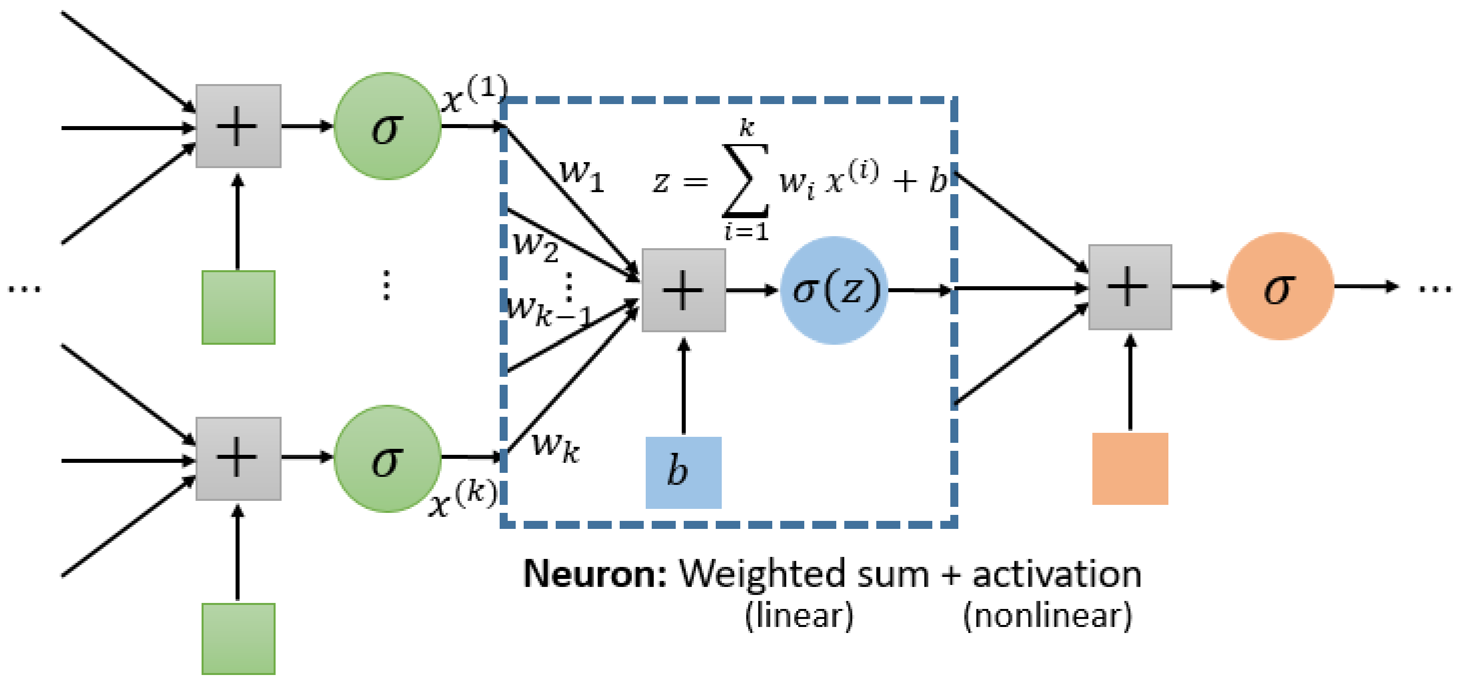


An example of manifold



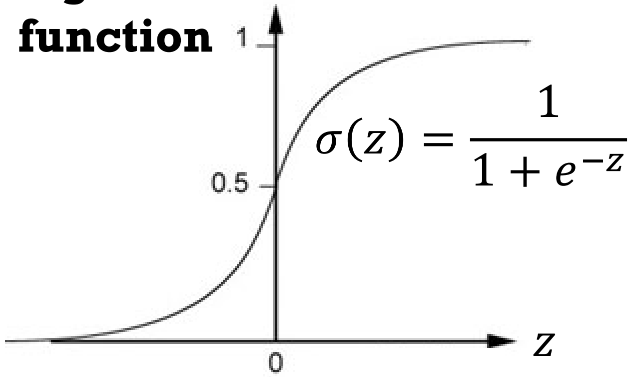
J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," ICLR 2014
https://medium.com/attentive-ai/fooling-cnns-via-adversarial-examples-877a9e0ee84e

RECAP: NEURAL NETWORK

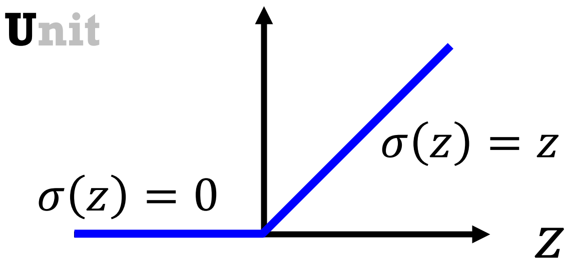


Neuron: Weighted sum + activation
(linear) (nonlinear)

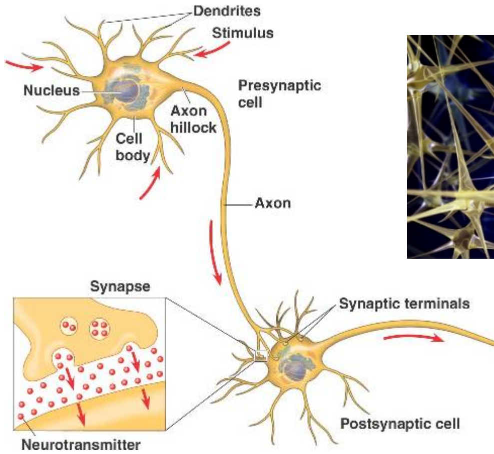
Sigmoid function



Rectified Linear Unit



- Different connection leads to different network structures
- Network parameter θ: all the weights and biases in the "neurons"

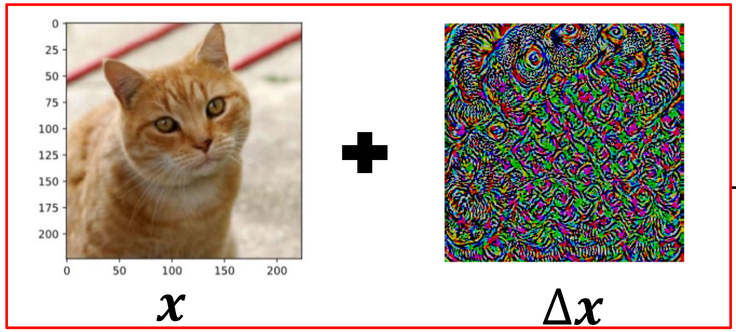


ADVERSARIAL EXAMPLE ATTACK REMEDIES

- **Attacker's goal**
  - Architecture

minimize $\boldsymbol{e}_{y*}{}^{T}\hat{\boldsymbol{z}} - \boldsymbol{e}_{y\text{targ}}{}^{T}\hat{\boldsymbol{z}}$
subject to $\|\Delta \boldsymbol{x}\|_p \leq \epsilon$

$\boldsymbol{x}'$



$\boldsymbol{x}$ + $\Delta \boldsymbol{x}$ → Classifier f_θ → $\hat{\boldsymbol{z}}$

Away from true label

Lure into target label

$e_{y*} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

$e_{y\text{targ}} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$

  - Add a small noise ($\|\Delta \boldsymbol{x}\|_p \leq \epsilon$) ***smartly*** to fool the classifier.

- **Defender methods**
  - Enhance robustness against ***specific*** attack. (Cat-and-mouse game)
  - Pre-processing the adversarial example (May also filter out the important feature)

Rectify original network

Attack img → robustness classifier → Predict label

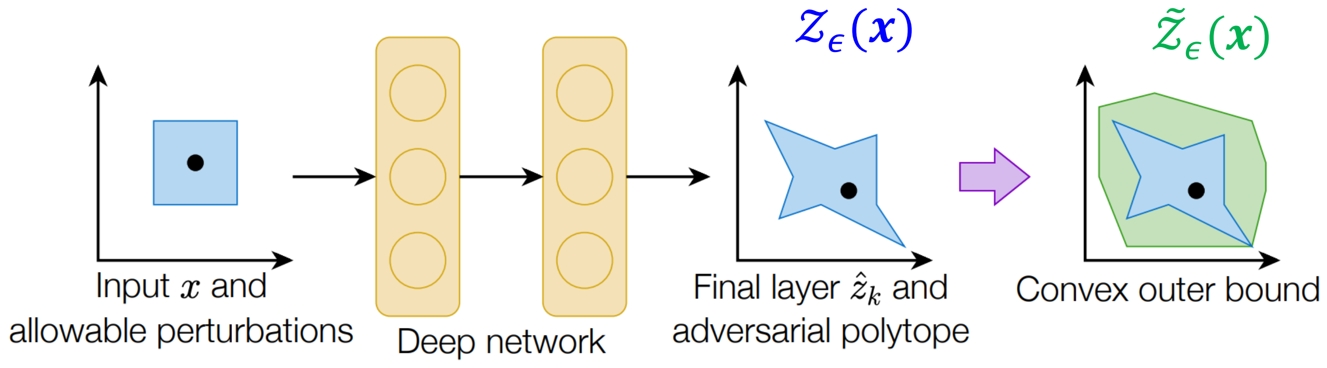
Attack img → preprocess Img block → clean img → Original classifier → Predict label

$\min_{\theta} \sum_{i=1}^{N} L(f_\theta(\boldsymbol{x}_i + \Delta \boldsymbol{x}_i), y_i)$ where $\Delta \boldsymbol{x}_i$ is by some specific attack.

Problem: No theoretical guarantees on the worst case attacking scenario.

PROVABLE DEFENSE VIA CONVEX OUTER ADVERSARIAL POLYTOPE

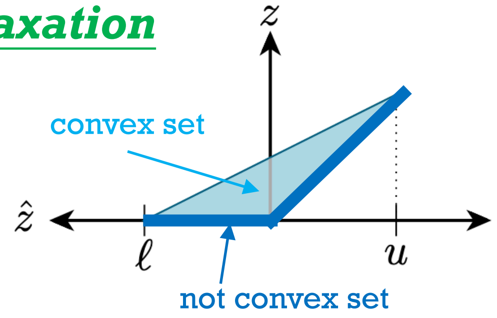
- Base on deep fully-connected network (with ReLU activation)



Each network layer:

- ➢ Linear transform (convex constraint)
- ➢ ReLU (*NOT* convex constraint)

➔*Convex relaxation*



$$\mathcal{Z}_\epsilon(\boldsymbol{x}) = \{f_\theta(\boldsymbol{x}+\Delta\boldsymbol{x}): \|\Delta\boldsymbol{x}\|_\infty \le \epsilon\}$$

Key idea:

- Convex relaxation on cover $\mathcal{Z}_\epsilon(\boldsymbol{x})$ with convex polytope $\tilde{\mathcal{Z}}_\epsilon(\boldsymbol{x})$
- Any solution in dual problem is a lower bound to primal problem.

Difference in confidence value between true label and target label (by adversary)

Primal problem **Dual problem**

dual variables

tunable parameter

$$\min \boxed{\boldsymbol{e}_{y*}{}^T\hat{\boldsymbol{z}}_k - \boldsymbol{e}_{y^{\text{targ}}}{}^T\hat{\boldsymbol{z}}_k}\ \boldsymbol{c}^T\hat{\boldsymbol{z}}_k \quad \text{subject to} \boxed{\begin{array}{c}\hat{\boldsymbol{z}}_k = f_\theta(\boldsymbol{x}+\Delta\boldsymbol{x}) \\ \|\Delta\boldsymbol{x}\|_\infty \le \epsilon\end{array}}\ \mathcal{Z}_\epsilon(\boldsymbol{x}) \ \ge \ \begin{array}{c}\min \boldsymbol{c}^T\hat{\boldsymbol{z}}_k \\ \text{subject to } \hat{\boldsymbol{z}}_k \in \tilde{\mathcal{Z}}_\epsilon(\boldsymbol{x})\end{array} \ = \ \begin{array}{c}\max \theta\boxed{(\boldsymbol{\xi},\boldsymbol{\nu},\boldsymbol{\mu},\boldsymbol{\tau},\boldsymbol{\lambda})} \\ \text{convex constraints}\end{array} \ \ge \ \mathcal{N}_\epsilon(\boldsymbol{x}, y^{\text{targ}}, \boxed{\alpha})$$

(not convex) (convex constraints) (convex constraints)

In neural network form!

Tractable bound on worst case adversarial attack scenario.

Eric Wong and J. Zico Kolter, "Provable defenses against adversarial examples via the convex outer adversarial polytope" ICML 2018

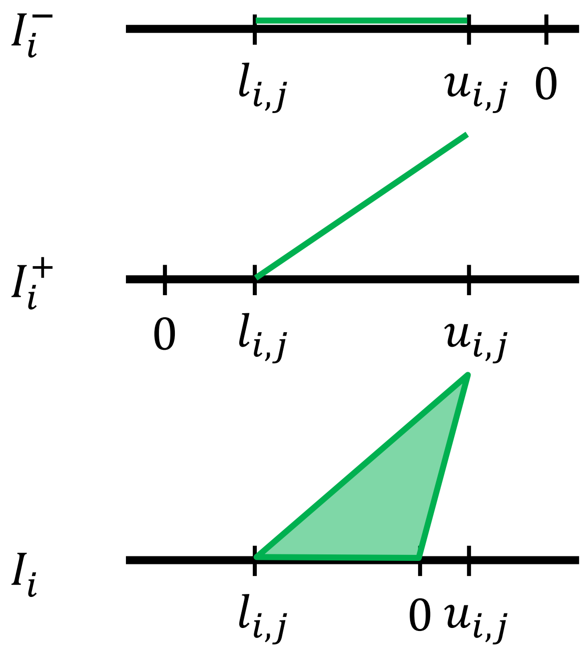
PRIMAL PROBLEM AFTER CONVEX RELAXATION

$$
\begin{array}{ll}
\text{minimize} & \mathbf{c}^T \hat{\mathbf{z}}_k \\
& \boxed{\hat{\mathbf{z}}_{i+1} = \mathbf{W}_i \mathbf{z}_i + \mathbf{b}_i}, \; i = 1, \ldots, k-1 \\
& \boxed{\begin{array}{l} \mathbf{z}_1 \le \mathbf{x} + \boldsymbol{\epsilon} \\ \mathbf{z}_1 \ge \mathbf{x} - \boldsymbol{\epsilon} \end{array}} \\
\text{subject to} & \boxed{\begin{array}{l}
z_{i,j} = 0, \; j \in \mathcal{I}_i^- \\
z_{i,j} = \hat{z}_{i,j}, \; j \in \mathcal{I}_i^+ \\
\left.\begin{array}{l}
z_{i,j} \ge 0 \\
z_{i,j} \ge \hat{z}_{i,j} \\
(u_{i,j} - l_{i,j}) z_{i,j} - u_{i,j} \hat{z}_{i,j} \le -u_{i,j} l_{i,j}
\end{array}\right\} j \in \mathcal{I}_i
\end{array}} \\
\text{variables} & \mathbf{z}_i \in \mathbb{R}^{d_i}, i = 1, \ldots, k-1 \\
& \hat{\mathbf{z}}_i \in \mathbb{R}^{d_i}, i = 2, \ldots, k
\end{array}
$$

weighted sum

$\|\cdot\|_\infty$ constraint on noise

ReLU layer (with convex relaxation)



DERIVE THE DUAL PROBLEM

Introduce dual variables

$$\hat{z}_{i+1} = W_i z_i + b_i \Rightarrow \nu_{i+1} \in \mathbb{R}^{|\hat{z}_{i+1}|}$$
$$z_1 \le x + \epsilon \Rightarrow \xi^+ \in \mathbb{R}^{|x|}$$
$$-z_1 \le -x + \epsilon \Rightarrow \xi^- \in \mathbb{R}^{|x|}$$
$$-z_{i,j} \le 0 \Rightarrow \mu_{i,j} \in \mathbb{R}$$
$$\hat{z}_{i,j} - z_{i,j} \le 0 \Rightarrow \tau_{i,j} \in \mathbb{R}$$
$$-u_{i,j}\hat{z}_{i,j} + (u_{i,j} - \ell_{i,j}) z_{i,j} \le -u_{i,j}\ell_{i,j} \Rightarrow \lambda_{i,j} \in \mathbb{R}$$

Write down the Lagranian

$$L(\mathbf{z},\hat{\mathbf{z}},\boldsymbol{\xi},\boldsymbol{\nu},\boldsymbol{\mu},\boldsymbol{\tau},\lambda)=\mathbf{c}^T\hat{\mathbf{z}}_k+\sum_{i=1}^{k-1}\boldsymbol{\nu}_{i+1}^T(\hat{\mathbf{z}}_{i+1}-(\mathbf{W}_i\mathbf{z}_i+\mathbf{b}_i))+\boldsymbol{\xi}_+^T(\mathbf{z}_1-(\mathbf{x}+\epsilon))+\boldsymbol{\xi}_-^T(-\mathbf{z}_1+(\mathbf{x}-\epsilon))$$
$$+\sum_{i=2}^{k-1}\left(\sum_{j\in\mathcal{I}_i^-\cup\mathcal{I}_i}\mu_{i,j}(-z_{i,j})+\sum_{j\in\mathcal{I}_i^+\cup\mathcal{I}_i}\tau_{i,j}(\hat{z}_{i,j}-z_{i,j})+\sum_{j\in\mathcal{I}_i}\lambda_{i,j}((u_{i,j}-l_{i,j})z_{i,j}-u_{i,j}\hat{z}_{i,j}+u_{i,j}l_{i,j})\right)$$

$$\theta(\boldsymbol{\xi},\boldsymbol{\nu},\boldsymbol{\mu},\boldsymbol{\tau},\boldsymbol{\lambda}) = \inf_{\mathbf{z},\hat{\mathbf{z}}} L(\mathbf{z},\hat{\mathbf{z}},\boldsymbol{\xi},\boldsymbol{\nu},\boldsymbol{\mu},\boldsymbol{\tau},\boldsymbol{\lambda})$$

Dual Problem

$$\begin{aligned}
\text{maximize}\quad & \theta(\boldsymbol{\xi},\boldsymbol{\nu},\boldsymbol{\mu},\boldsymbol{\tau},\lambda) = -\boldsymbol{\xi}_+^T(\mathbf{x}+\epsilon)+\boldsymbol{\xi}_-^T(\mathbf{x}-\epsilon)-\textstyle\sum_{i=1}^{k-1}\boldsymbol{\nu}_{i+1}^T\mathbf{b}_i+\sum_{i=2}^{k-1}\lambda_i^T(\mathbf{u}_i\odot\mathbf{l}_i)\\
\text{subject to}\quad & \boldsymbol{\nu}_k = -\mathbf{c}\\
& \mathbf{W}_1^T\boldsymbol{\nu}_2 = \boldsymbol{\xi}_+ - \boldsymbol{\xi}_-\\
& \left.\begin{aligned}&\boldsymbol{\tau}_i+\boldsymbol{\nu}_i = \mathbf{u}_i\odot\lambda_i\\ &\mathbf{W}_i^T\boldsymbol{\nu}_{i+1}+\boldsymbol{\tau}_i+\boldsymbol{\mu}_i = (\mathbf{u}_i-\mathbf{l}_i)\odot\lambda_i\end{aligned}\right\}\ i=2,\ldots,k-1\\
& \boldsymbol{\xi}_+,\boldsymbol{\xi}_- \in \mathbb{R}^{d_0}_{\ge 0}\\
\text{variables}\quad & \boldsymbol{\nu}_i \in \mathbb{R}^{d_i},\ i=2,\ldots,k\\
& \left.\begin{aligned}&\mu_{i,j}\in\mathbb{R}, && \tau_{i,j}=0, && \lambda_{i,j}=0, && j\in\mathcal{I}_i^-\\ &\mu_{i,j}\ge 0, && \tau_{i,j}\ge 0, && \lambda_{i,j}\ge 0, && j\in\mathcal{I}_i\\ &\mu_{i,j}=0, && \tau_{i,j}\in\mathbb{R}, && \lambda_{i,j}=0, && j\in\mathcal{I}_i^+\end{aligned}\right\}\ i=2,\ldots,k-1
\end{aligned}$$

DUAL PROBLEM IN NEURAL NETWORK FORM

maximize $\quad J_\epsilon(\mathbf{x}, \boldsymbol{\nu}) = -\mathbf{x}^T \hat{\boldsymbol{\nu}}_1 - \epsilon |\hat{\boldsymbol{\nu}}_1| - \sum_{i=1}^{k-1} \mathbf{b}_i^T \boldsymbol{\nu}_{i+1} + \sum_{i=2}^{k-1} \sum_{j \in \mathcal{I}_i} \frac{u_{i,j} l_{i,j}}{u_{i,j} - l_{i,j}} [\nu_{i,j}]_+$

subject to

$$\boxed{\boldsymbol{\nu}_k = -\mathbf{c}} \quad \text{input}$$

$$\left.\begin{aligned} &\boxed{\hat{\nu}_{i,j} = (\mathbf{W}_i^T \boldsymbol{\nu}_{i+1})_j}, \quad i = 1, \dots, k-1 \quad \text{linear transform} \\ &\nu_{i,j} = \left\{ \boxed{\begin{array}{ll} 0 & , \text{ if } j \in \mathcal{I}_i^- \\ \frac{u_{i,j}}{u_{i,j} - l_{i,j}} [\hat{\nu}_{i,j}]_+ - \alpha_{i,j} [\hat{\nu}_{i,j}]_- & , \text{ if } j \in \mathcal{I}_i \\ \hat{\nu}_{i,j} & , \text{ if } j \in \mathcal{I}_i^+ \end{array}} \right. \\ &0 \le \alpha_{i,j} \le 1, \ j \in \mathcal{I}_i \qquad \text{Leaky ReLU} \end{aligned}\right\} i = 2, \dots, k-1$$

variables $\quad \boldsymbol{\nu}, \hat{\boldsymbol{\nu}}$

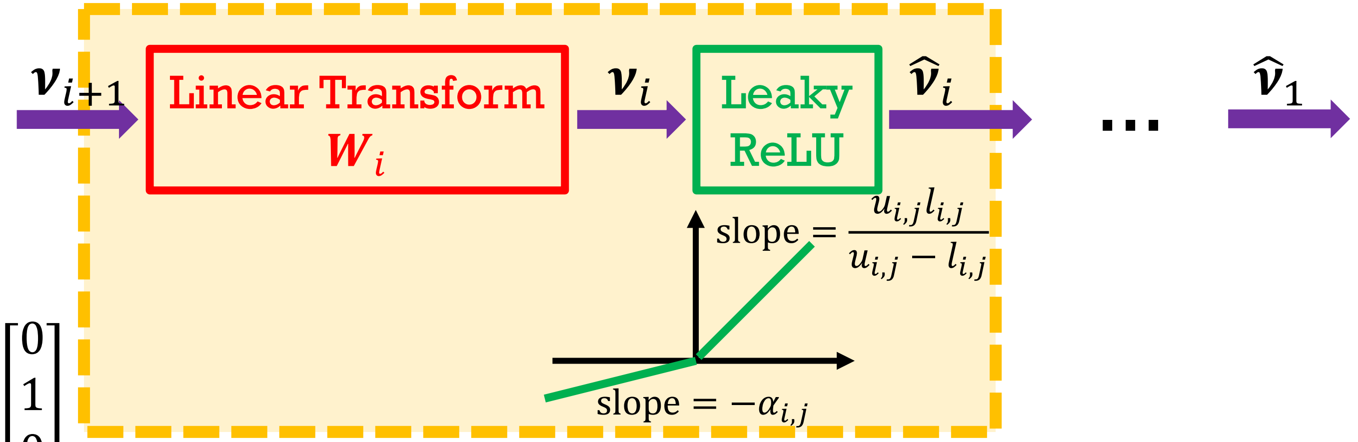
Can be written as neural network $\mathcal{N}_\epsilon(\boldsymbol{x}, y^{\text{targ}}, \alpha)$

$\boldsymbol{\nu}_k = \boldsymbol{c} = \boldsymbol{e}_{y*} - \boldsymbol{e}_{y^{\text{targ}}}$

(depends on y^{targ})

$e_{y*} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

$e_{y^{\text{targ}}} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$



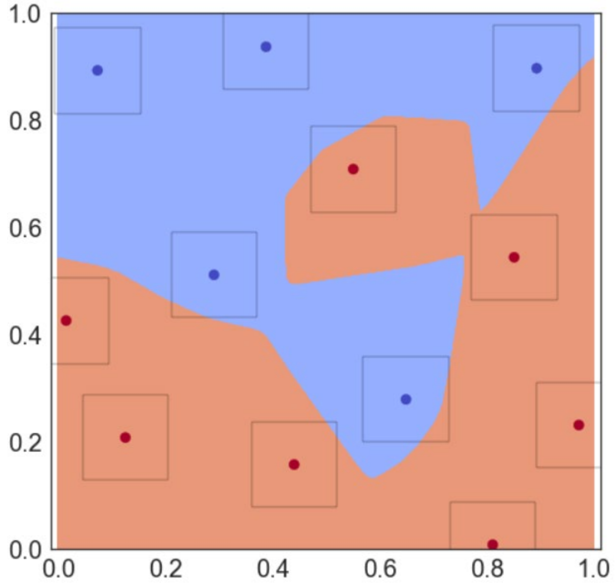
ROBUST LEARNING

$$\begin{matrix} \min \boldsymbol{e}_{y*}{}^T \hat{\boldsymbol{z}}_k - \boldsymbol{e}_{y^{\mathrm{targ}}}{}^T \hat{\boldsymbol{z}}_k \\ \text{subject to } ||\Delta \boldsymbol{x}||_\infty \leq \epsilon \end{matrix} \geq \mathcal{N}_\epsilon(\boldsymbol{x}, y^{\mathrm{targ}}, \alpha)$$

If $\mathcal{N}_\epsilon(\boldsymbol{x}, \alpha)$ is positive for all y^{targ}, then adversary cannot fool the classifier

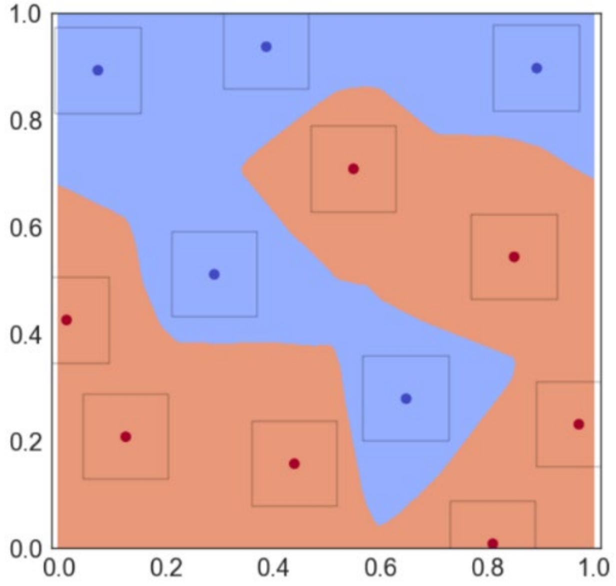
Standard training

$$\min_\theta \sum_{i=1}^{N} L(f_\theta(x_i), y_i)$$



Robust training

$$\max_\theta \min_{y^{\mathrm{targ}}} \sum_{i=1}^{N} \mathcal{N}_\epsilon(\boldsymbol{x}, y^{\mathrm{targ}}, \alpha)$$



TAKE AWAY MESSAGES

- MLaaS raises security issues such as
  - Model inversion attack: Infer training data from model.
  - Membership inference attack: Infer membership from model.
  - Adversarial example attack: Fool the classifier with unperceivable noises.

  and many others…
- Methods for secure machine learning:
  - Differential Privacy: Adding noise to machine learning models.
  - Homomorphic Encryption: Cryptographic approach. Secure but costly.
  - Compressive Privacy: Nonlinear lossy compression
    - ✓ Preserve sufficient information for machine learning service.
    - ✓ Difficult for intruder to reconstruct from compressed data.
    - ✓ CPGAN defends the reconstruction attack while achieving satisfactory utility performance.


Q
&
A