MPCDiff: Testing and Repairing MPC-Hardened Deep Learning Models

Qi Pang[1], Yuanyuan Yuan[2], Shuai Wang[2]

[1]Carnegie Mellon University

[2]The Hong Kong University of Science and Technology



ChatGPT was temporarily banned in Italy

BBC

Home News Sport Business Innovation Culture Travel Earth Video Live

ChatGPT banned in Italy over privacy concerns

1 April 2023

Share

CNN BUSINESS

Markets Tech Media Calculators Videos

Italy blocks ChatGPT over privacy concerns

By Livvy Doherty and Sharon Braithwaite, CNN

2 minute read · Updated 12:19 PM EDT, Fri March 31, 2023

CNBC

MARKETS BUSINESS INVESTING TECH POLITICS CNBC TV INVESTING CLUB PRO

TECH

Italy became the first Western country to ban ChatGPT. Here's what other countries are doing

PUBLISHED TUE, APR 4 2023·4:48 AM EDT | UPDATED MON, APR 17 2023·1:24 AM EDT

Ryan Browne
@RYAN_BROWNE_

SHARE

THE WALL STREET JOURNAL.

World Business U.S. Politics Economy Tech Finance Opinion Arts & Culture Lifestyle

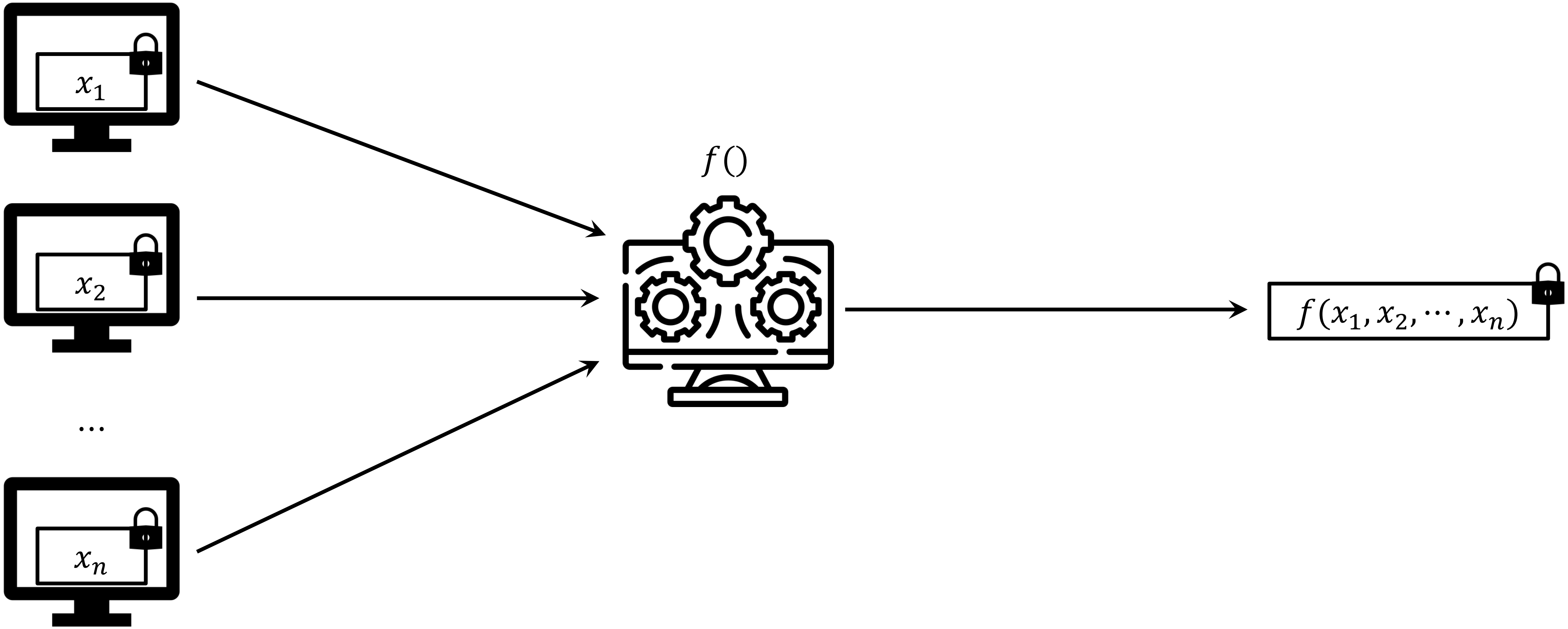
ChatGPT Banned in Italy Over Data-Privacy Concerns

Privacy order comes as regulatory scrutiny over artificial-intelligence tools grows

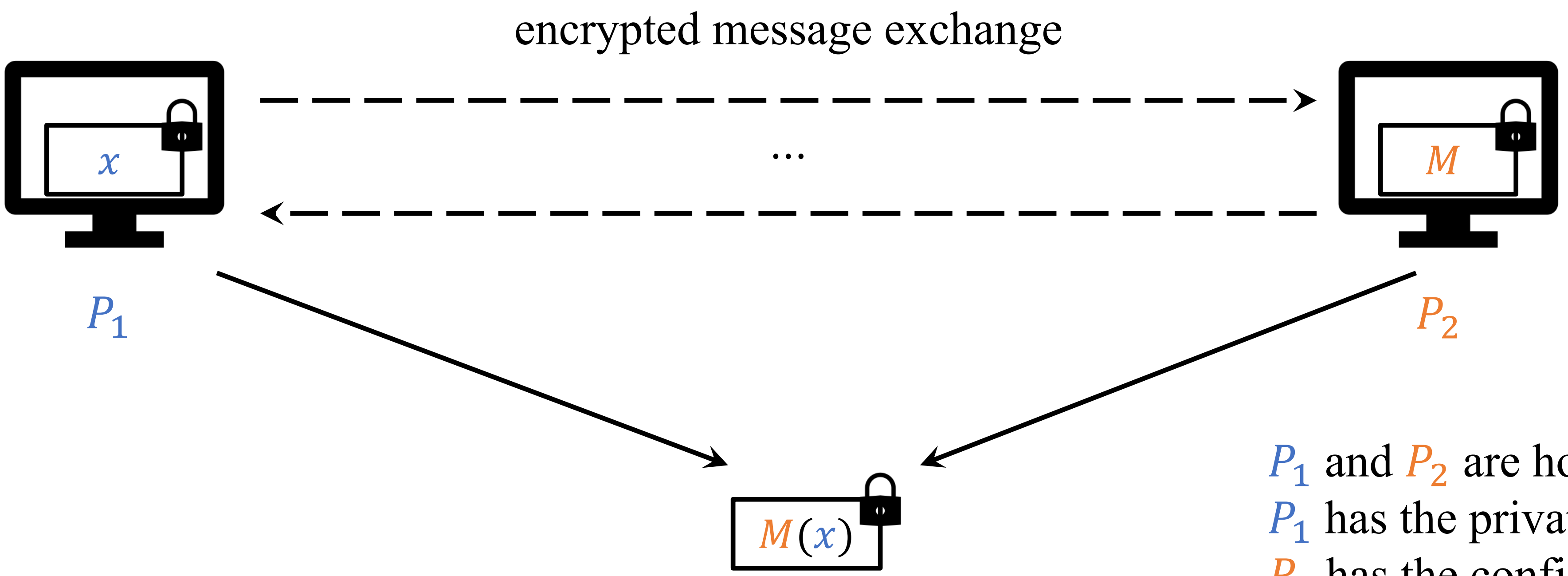
By Margherita Stancati Follow *and Sam Schechner* Follow

Updated March 31, 2023 5:06 pm ET

Secure multi-party computation (MPC)[YAO86, GMW87, BGW88]



Secure two-party deep learning inference



P_1 and P_2 are honest but curious.
P_1 has the private input data x.
P_2 has the confidential model M.

Examples of secure inference for DL models

SecureML: A System for Scalable Privacy-Preserving Machine Learning[MZ17]

GAZELLE: A Low Latency Framework for Secure Neural Network Inference[JVC18]

Iron: Private Inference on Transformers[HLC+22]

BumbleBee: Secure Two-party Inference Framework for Large Transformers[LHG+23]

Delphi: A Cryptographic Inference Service for Neural Networks[MLS+20]

Cheetah: Lean and Fast Secure Two-Party Deep Neural Network Inference[HLH+22]

BOLT: Privacy-Preserving, Accurate and Efficient Inference for Transformers[PZM+23]

TFEncrypted

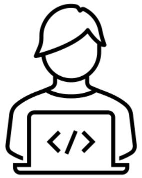
Syft

CrypTen

MPC-Hardened DL models

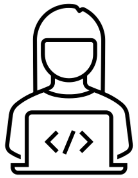
- Addition

$$x = [x]_1 + [x]_2 \bmod p$$
$$y = [y]_1 + [y]_2 \bmod p$$
$$z = x + y \bmod p$$



$[x]_1, [y]_1$

$$[z]_1 = [x]_1 + [y]_1 \bmod p$$



$[x]_2, [y]_2$

$$[z]_2 = [x]_2 + [y]_2 \bmod p$$

$$z = [z]_1 + [z]_2 \bmod p = x + y \bmod p$$

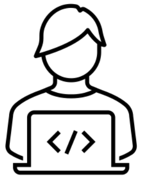
MPC-Hardened DL models

- Multiplication

$$x = [x]_1 + [x]_2 \bmod p$$
$$y = [y]_1 + [y]_2 \bmod p$$
$$z = x \times y \bmod p$$

Beaver triples:

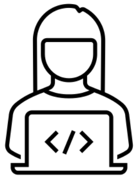
$$c = a \times b$$



$[x]_1, [y]_1$
$[a]_1, [b]_1, [c]_1$

$[\alpha]_1 = [x]_1 - [a]_1$
$[\beta]_1 = [y]_1 - [b]_1$

$[\alpha]_2 = [x]_2 - [a]_2$
$[\beta]_2 = [y]_2 - [b]_2$



$[x]_2, [y]_2$
$[a]_2, [b]_2, [c]_2$

$$[z]_1 = [c]_1 + \alpha[b]_1 + \beta[a]_1 + \alpha\beta$$

$$[z]_2 = [c]_2 + \alpha[b]_2 + \beta[a]_2 + \alpha\beta$$

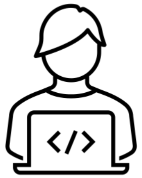
$$z = [z]_1 + [z]_2 \bmod p = x \times y \bmod p$$

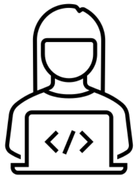
MPC-Hardened DL models

- Non-linear functions

$$x = [x]_1 + [x]_2 \bmod p$$

$$y = \exp(x) = \lim_{n \to \infty} \left(1 + \frac{x}{n}\right)^n$$





encrypted message exchange

$[x]_1$ ⟶ ⟵ $[x]_2$

$$[z]_1 = \left[\left(1 + \frac{x}{n}\right)^n\right]_1 \qquad [z]_2 = \left[\left(1 + \frac{x}{n}\right)^n\right]_2$$

$$z = [z]_1 + [z]_2 \bmod p = \exp(x)$$

Observation#1: Fixed-point representation

- Use **fixed-point arithmetic** to represent a floating-point value $\tilde{\boldsymbol{x}} \in R$:
  - $x = \lfloor \tilde{x} 2^m \rfloor$, m is the precision bit.

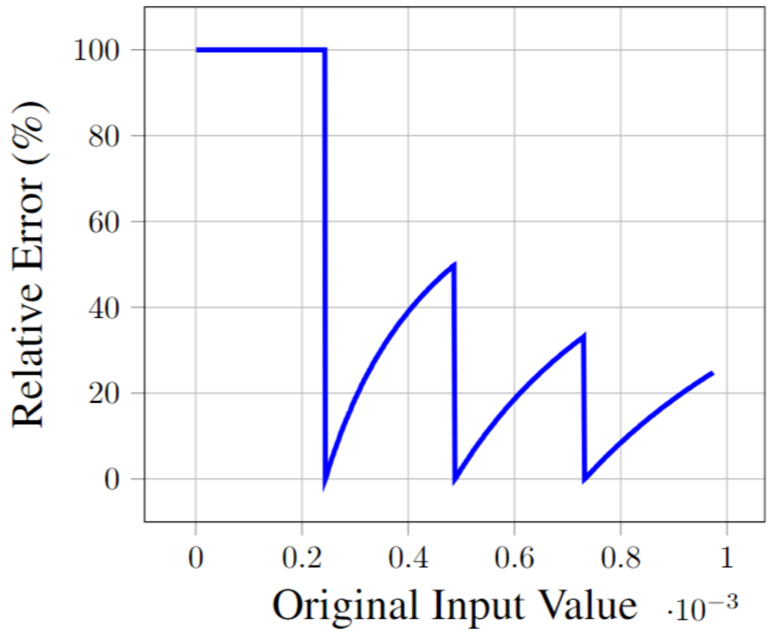


Figure 1. Relative error of fixed-point representation.

Observation#1: Fixed-point representation

- Use fixed-point arithmetic to represent a floating-point value $\tilde{\boldsymbol{x}} \in$ R:
  - $x = \lfloor \tilde{x} 2^{m} \rfloor$, m is the precision bit.
- The multiplication results is truncated by m bits for subsequent computation.
  - $z = x \times y = \lfloor \tilde{x} 2^{m} \rfloor \times \lfloor \tilde{y} 2^{m} \rfloor$, has 2m bits scale.
  - Local truncation drops the last m bits of $[z]_1$ and $[z]_2$ locally, resulting in a **1-bit random error** in the last bit (w.h.p.).

Observation#2: Non-linear function approximation

- There are many non-linear functions in DL models like Sigmoid, Tanh, and GELU.
  - These functions are usually **approximated** in MPC.

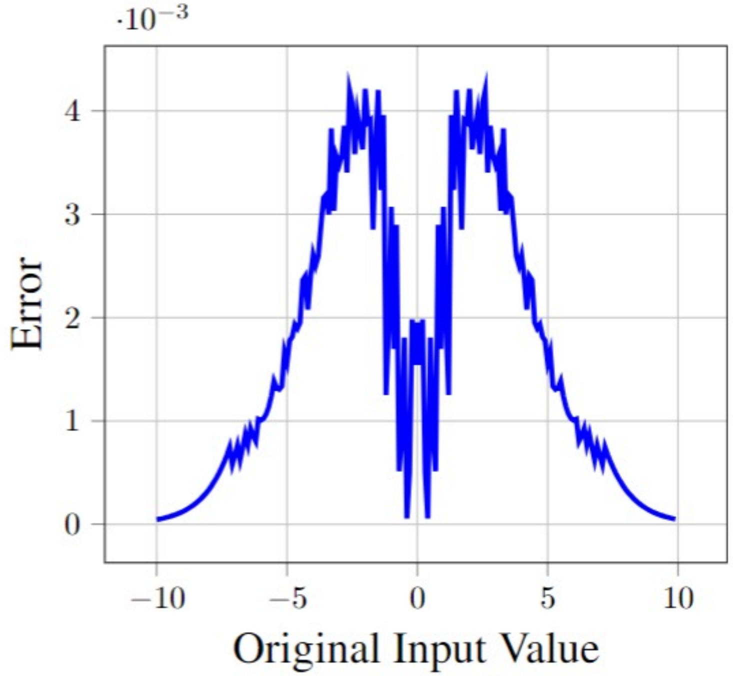
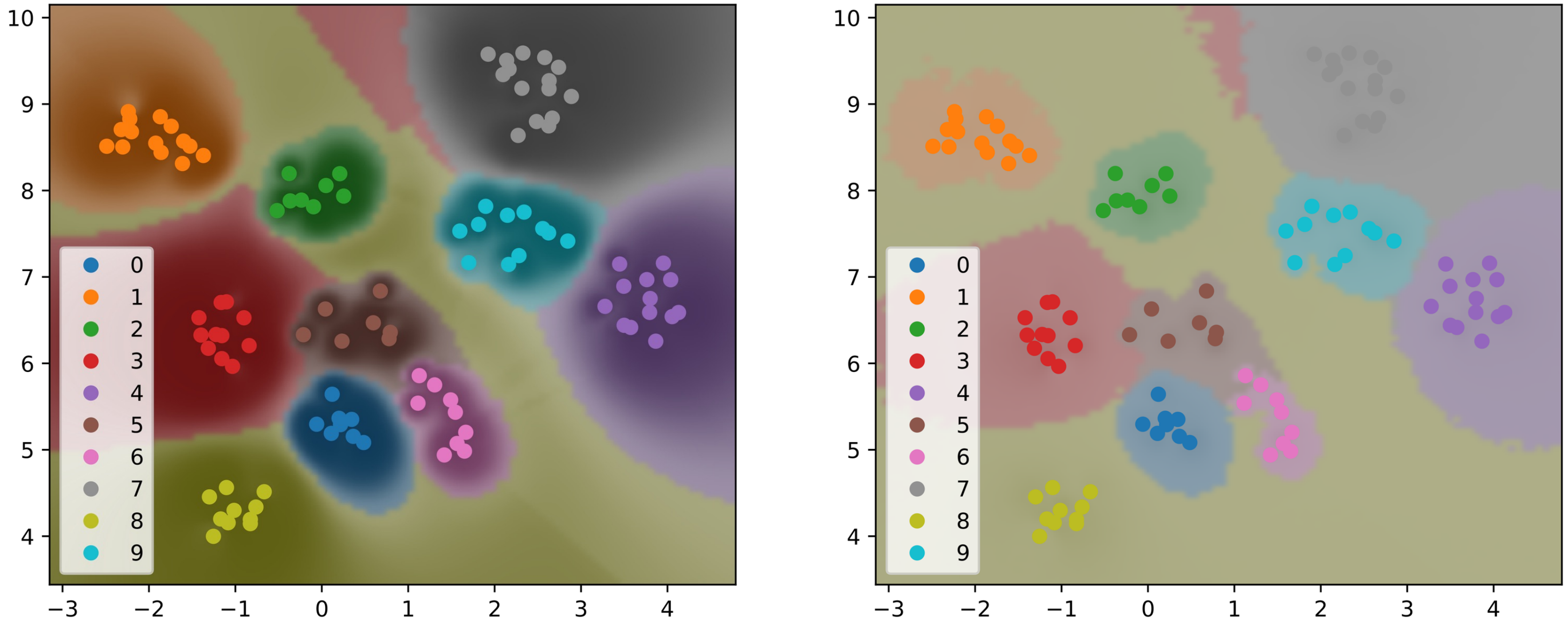


Figure 2. Relative error of Sigmoid approximation.

Observation#3: Decision boundary shifting

- The errors will result in a **decision boundary shifting** in MPC-hardened DL models.



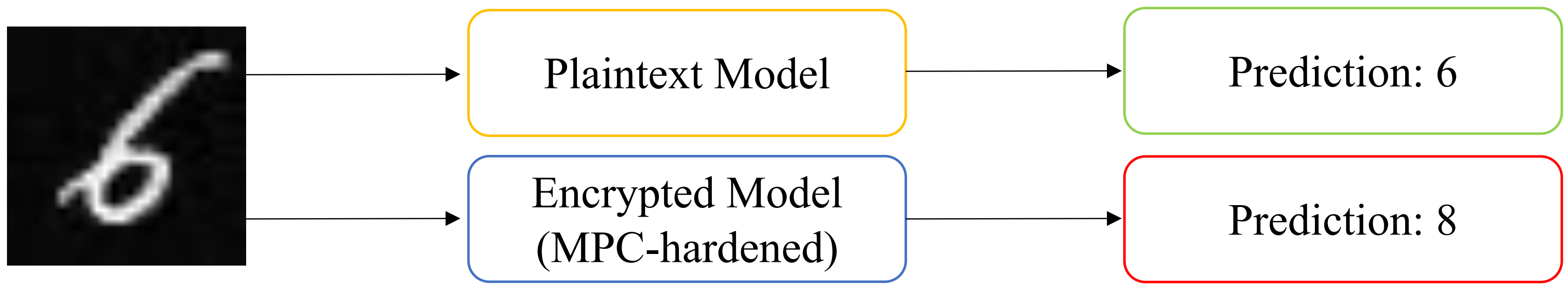
(a). Decision boundaries of the **original** model.

(b). Decision boundaries of the **MPC-hardened** model.

Figure 3. Decision boundaries of LeNet and its MPC-hardened version for classifying MNIST images.

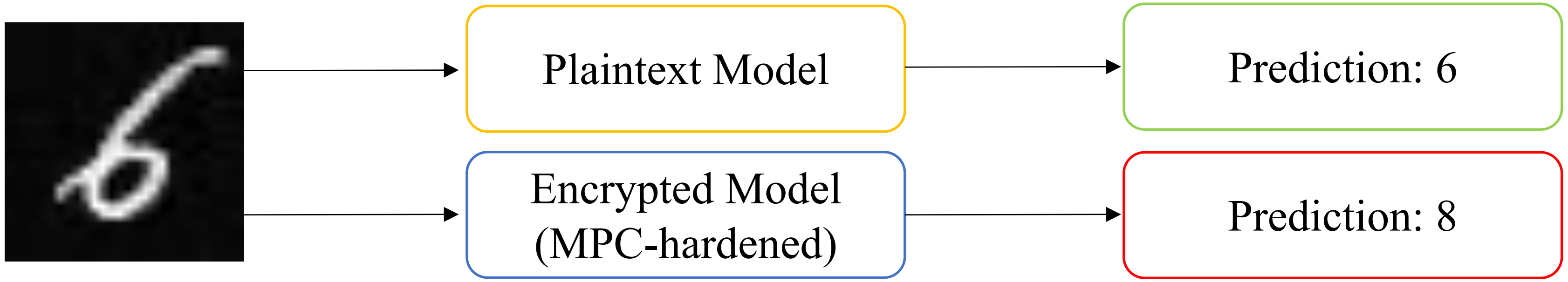
Observation#3: Decision boundary shifting

- The errors will result in a **decision boundary shifting** in MPC-hardened DL models.



Observation#3: Decision boundary shifting

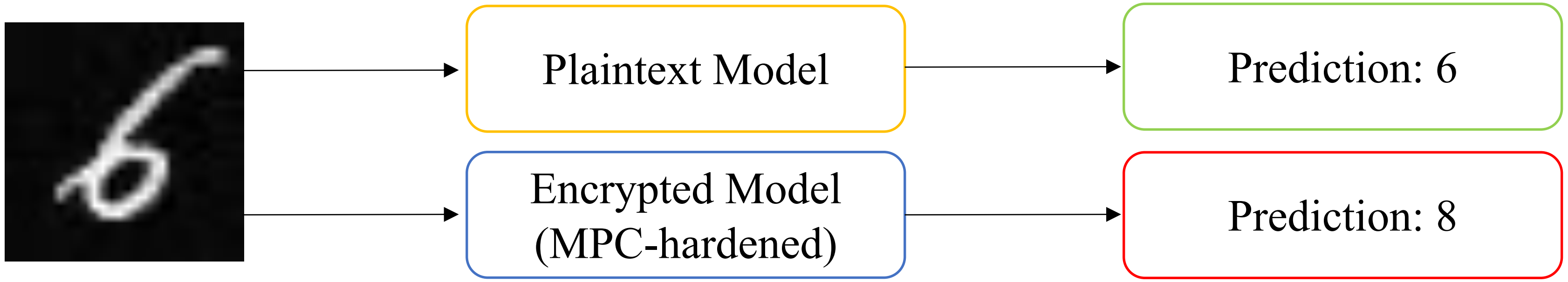
- The errors will result in a **decision boundary shifting** in MPC-hardened DL models.



Can we find such deviation-triggering inputs efficiently?
Can we mitigate this issue by repairing the MPC-hardened models?

Observation#3: Decision boundary shifting

- The errors will result in a **decision boundary shifting** in MPC-hardened DL models.



Can we find such deviation-triggering inputs efficiently?
Can we mitigate this issue by repairing the MPC-hardened models?

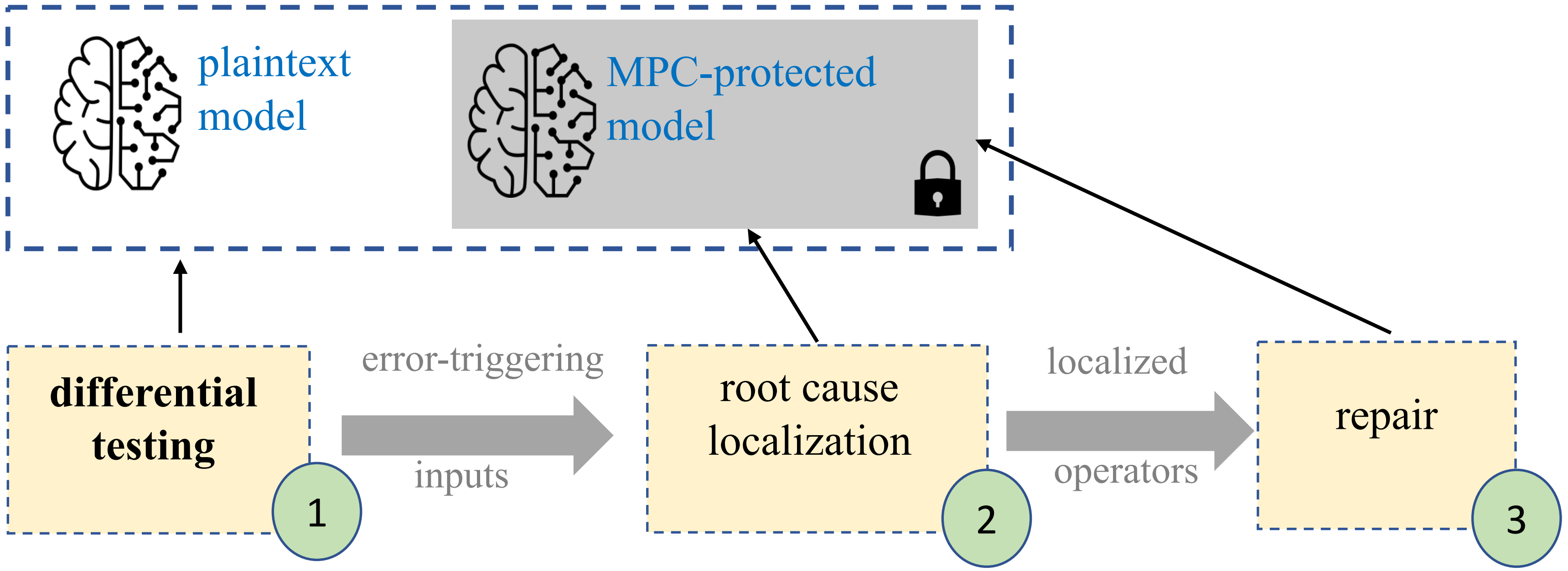
MPCDiff

Application scenario

- MPCDiff is designed for model maintainers and developers to **assess and improve the robustness** of MPC-hardened DL models.
- MPCDiff aims to **bridge the gap** between the MPC-hardened and the plaintext models.
- The testing and repairing in MPCDiff are launched in the **localhost** network settings by the developer.

Our approach: MPCDiff

- MPCDiff automatically generates/uncovers these deviation-triggering inputs by **feedback-driven differential testing**.



MPCDiff : Differential testing

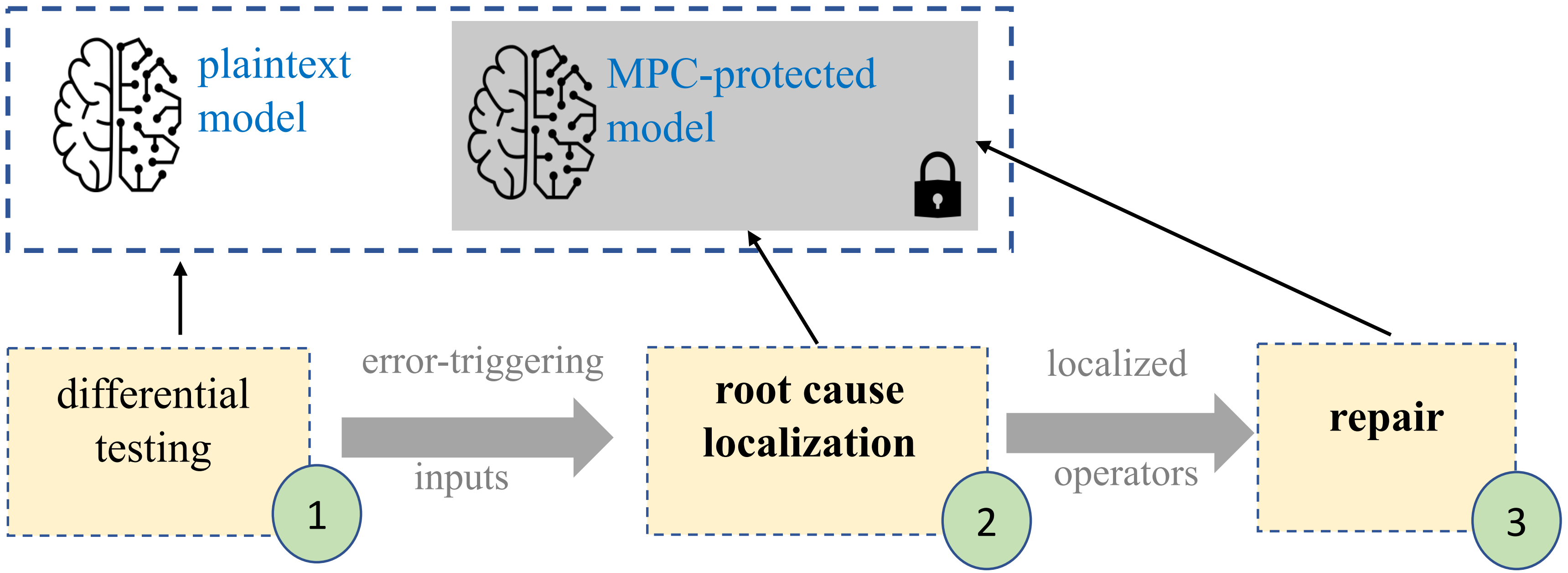
MPCDiff employs feedback-driven differential testing to explore inputs that result in deviant outputs of MPC-protected models and their plaintext models.

$$maximize_{x'}: \delta = \left|M_p(x') - M_m(x')\right|, \qquad s.t.\ |x' - x| \leq v$$

plaintext model MPC-hardened model mutated input original input

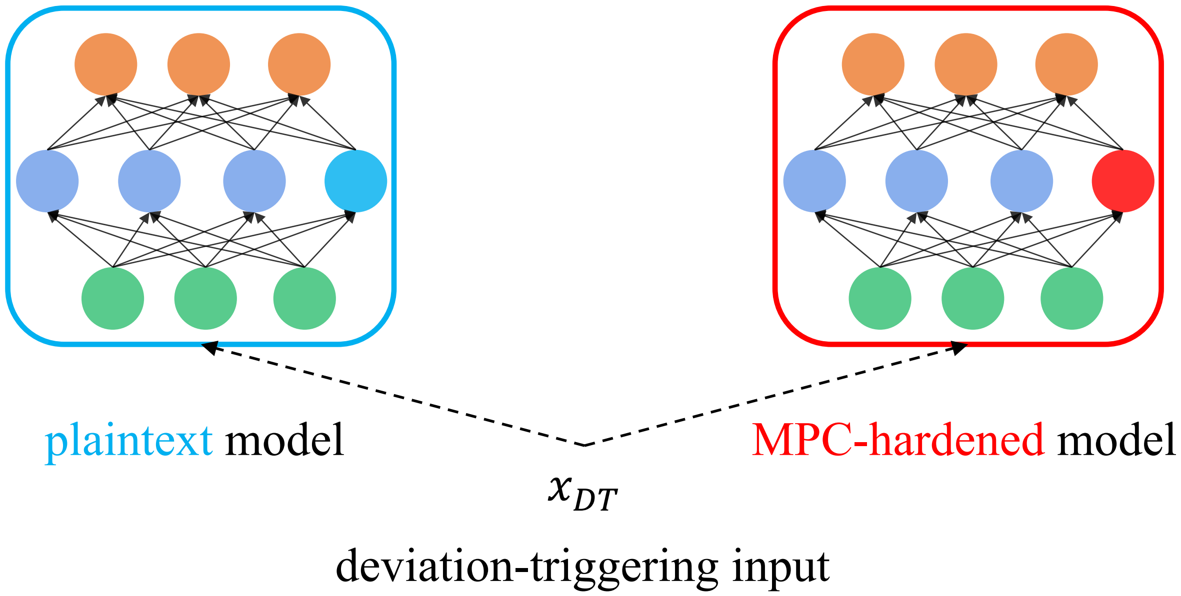
Our approach: MPCDiff

- MPCDiff automatically generates/uncovers these deviation-triggering inputs.
- With these inputs, MPCDiff localizes the root causes and repairs the model with the localized operators.



MPCDiff : Root cause localization

- MPCDiff employs a **voting-based method** to localize neurons that **primarily contribute to the deviation**.



$$\delta_i(x_{DT}) = |n_i^p - n_i^m|$$

If $\delta_i(x_{DT}) \geq \tau_+$:

$$w_i \rightarrow w_i + 1$$

If $\delta_i(x_{DT}) \leq \tau_-$:

$$w_i \rightarrow w_i - 1$$

n_i^p: The i^{th} neuron of the plaintext model.
n_i^m: The i^{th} neuron of the MPC-hardened model.
τ_+, τ_-: Thresholds.
w_i: importance weight of the i^{th} neuron.

MPCDiff : Repairing

- MPCDiff **increases the approximation level of the non-linear functions** that produce the neurons with high importance weights.

☑ The nonlinear functions on the neurons contribute more to the deviation are evaluated more accurately.

☑ Precision bit tuning achieves an optimal balance between preventing overflow and enhancing robustness.

Evaluation setup

Datasets, models, and MPC protocols.

Framework	Model	Datasets	Plaintext Accuracy	Encrypted Accuracy
CrypTen	LeNet MLP-Sigmoid MLP-GELU	MNIST Credit Bank	98.65% 82.93% 90.00%	97.25% 80.70% 89.90%
TF-Encrypted	LeNet MLP-Sigmoid MLP-GELU	MNIST Credit Bank	98.20% 82.93% 90.10%	96.90% 80.10% 90.10%
PySyft	LeNet MLP-Sigmoid MLP-GELU	MNIST Credit Bank	97.95% 82.93% 90.10%	97.35% 80.70% 89.40%

Both plaintext and encrypted models achieve good accuracy.

Findings: Testing

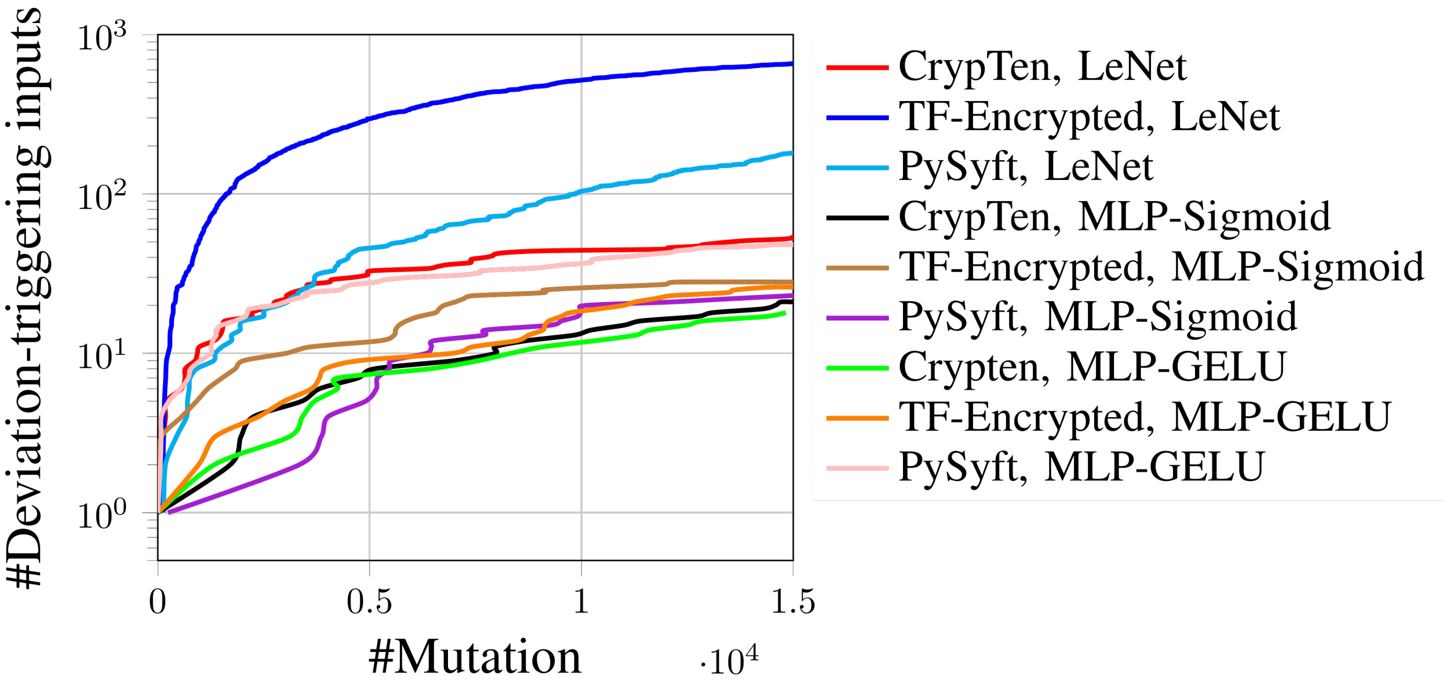


Figure 4. #Deviation-triggering inputs found by MPCDiff.

MPCDiff can effectively detect a great number of deviation-triggering inputs on various datasets and models.

Findings: Testing

MPC Framework	Datasets	Error-Inducing Inputs	Avg. L2-Distance
CrypTen	MNIST	 ···	0.0018
	Credit	[0.000, 1.000, ..., 0.014, 0.039] ···	0.019
	Bank	[0.494, 0.454, ..., 0.957, 0.860] ···	0.018
TF-Encrypted	MNIST	 ···	0.0029
	Credit	[0.010, 0.000, ..., 0.276, 0.009] ···	0.0022
	Bank	[0.197, 0.636, ..., 0.000, 0.170] ···	0.032
PySyft	MNIST	 ···	0.0034
	Credit	[0.802, 0.000, ..., 0.846, 0.297] ···	0.023
	Bank	[0.049, 0.727, ..., 0.060, 0.106] ···	0.015

Figure 5. Examples of deviation-triggering inputs found by MPCDiff.

The deviation-triggering inputs have high quality, with close distance to normal data and hard to distinguish.

Findings: Repairing

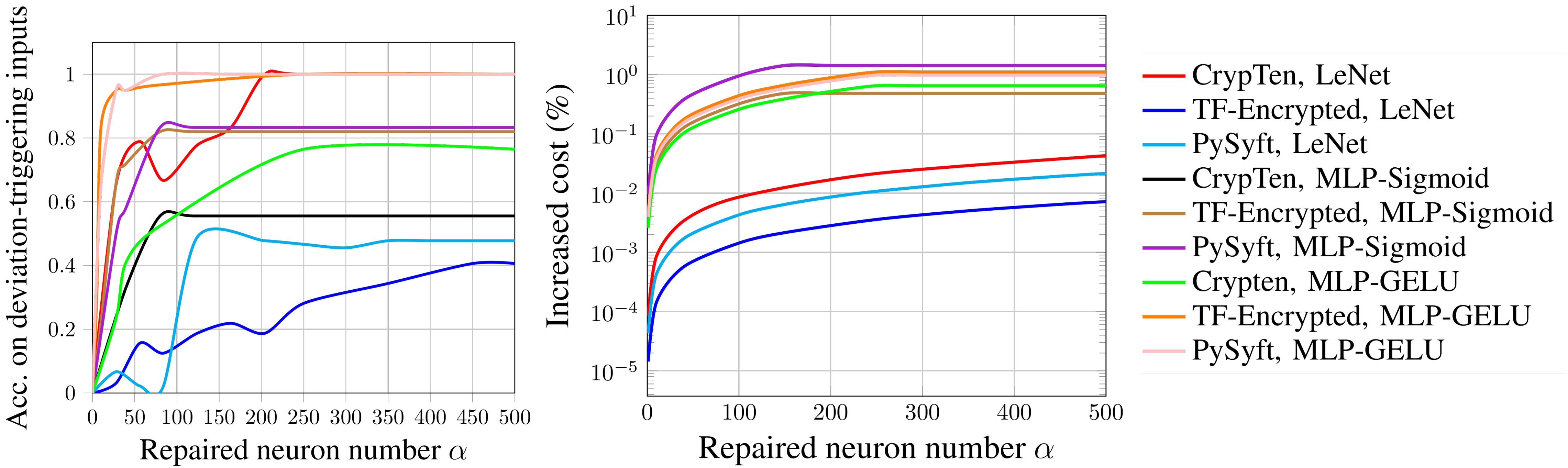
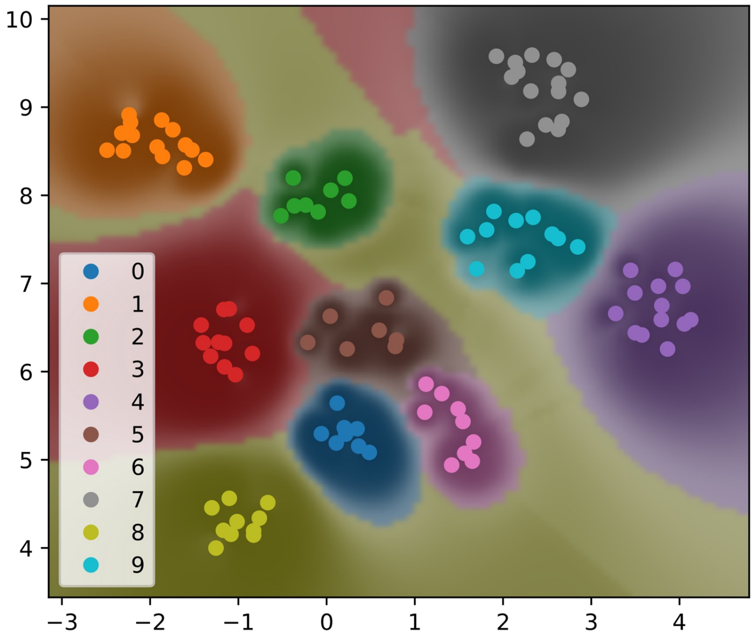
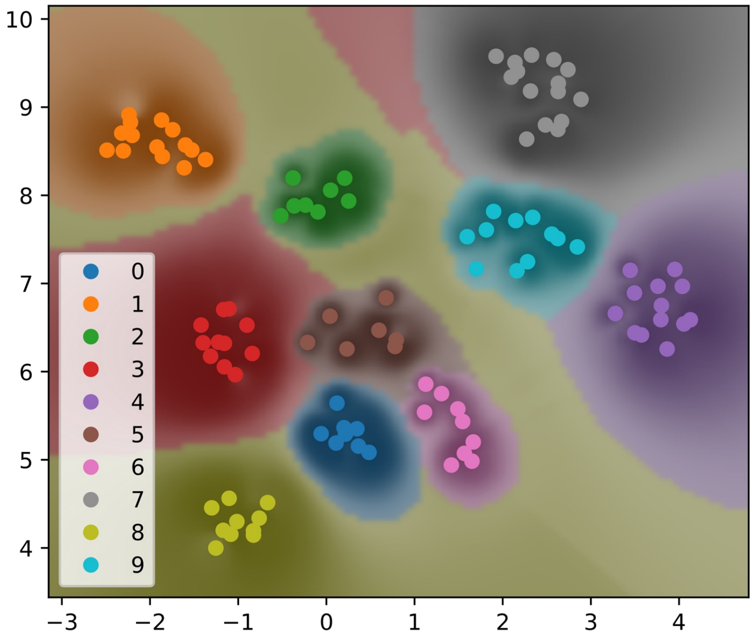


Figure 6. Repaired neuron number vs accuracy and increased cost.

Findings: Repairing



(a). Decision boundaries of the **original** model.



(b). Decision boundaries of the **repaired MPC-hardened** model.

Figure 7. Decision boundaries of LeNet and its repaired MPC-hardened version for classifying MNIST images.

The repaired MPC-hardened models have better accuracy than the original MPC-hardened models on test data. The repaired MPC-hardened models are significantly more robust than the original MPC-hardened models.

Take away

Email: qipang@cmu.edu Code:

- Conceptually
  - Reveal deviation-triggering inputs particularly exist in MPC-hardened DL models.
- Technically
  - MPCDiff incorporates a set of simple but effective designs to uncover deviation-triggering inputs and repair MPC-hardened models.
- Empirically
  - MPCDiff finds a large number of deviation-triggering inputs across different popular MPC platforms, models, and datasets.
  - Repairing significantly improves the MPC-hardened models' robustness.