# Model Inversion Attacks Against Collaborative Inference

Bertani Alessandro - Corrà Alessandro

# Overview

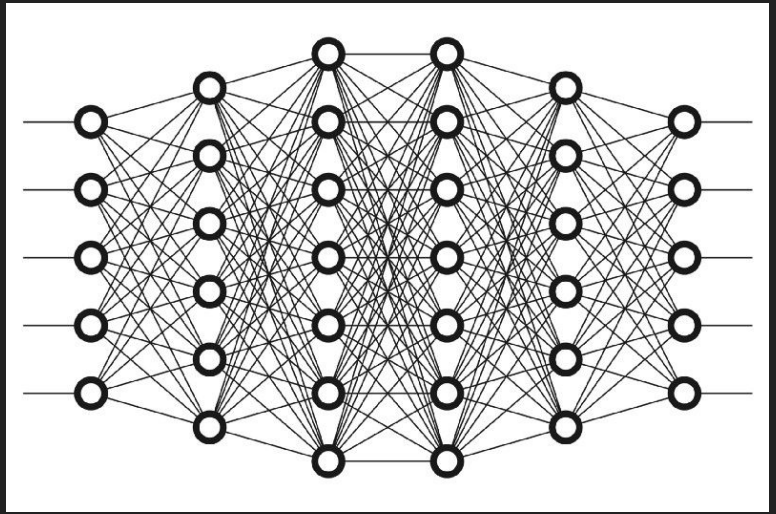
- Introduction
- Content of the paper
- Experiment reproduction
- Considerations

# Introduction

The paper presents a new set of attacks against the **inference** data privacy in **collaborative** Deep Learning systems.

Specifically, the attacks are aimed at Deep Neural Networks (DNNs) which are split and distributed among several participants.

The attacks try to recover user-provided input based on the intermediate results exchanged by the various participants.



# Deep Learning models

Deep Learning (DL) models outperform traditional Machine Learning models on various AI tasks, such as:

- Image classification
- Natural language processing
- Speech recognition
- Anomaly detection

The paper focuses on image classification tasks, addressed by a particular kind of DL model: Deep Neural Networks.

# Deep Neural Networks

A Deep Neural Network is a parameterized non-linear function that maps an input tensor X to an output tensor Y, or $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$.

The basic block of a DNN is a “neuron”, an element that computes a linear or nonlinear function of its inputs.

The neurons are organized in **layers**, and various architectures have different numbers of layers and of neurons per layer.

A layer is a set of neurons with the same depth, and in the simplest possible classification we can distinguish between **input**, **output** and **hidden** layers.

# Deep Neural Networks

DNNs are used for several tasks, such as regression and classification.

Before being able to predict values, a DNN must be trained in order to learn the model parameters that will yield the best prediction possible.

In DNN, the training is performed through **backpropagation** and **stochastic gradient descent** (SGD).

After the model is trained, given an input x the corresponding output can be computed as $y = f_\theta(x)$ (inference process).

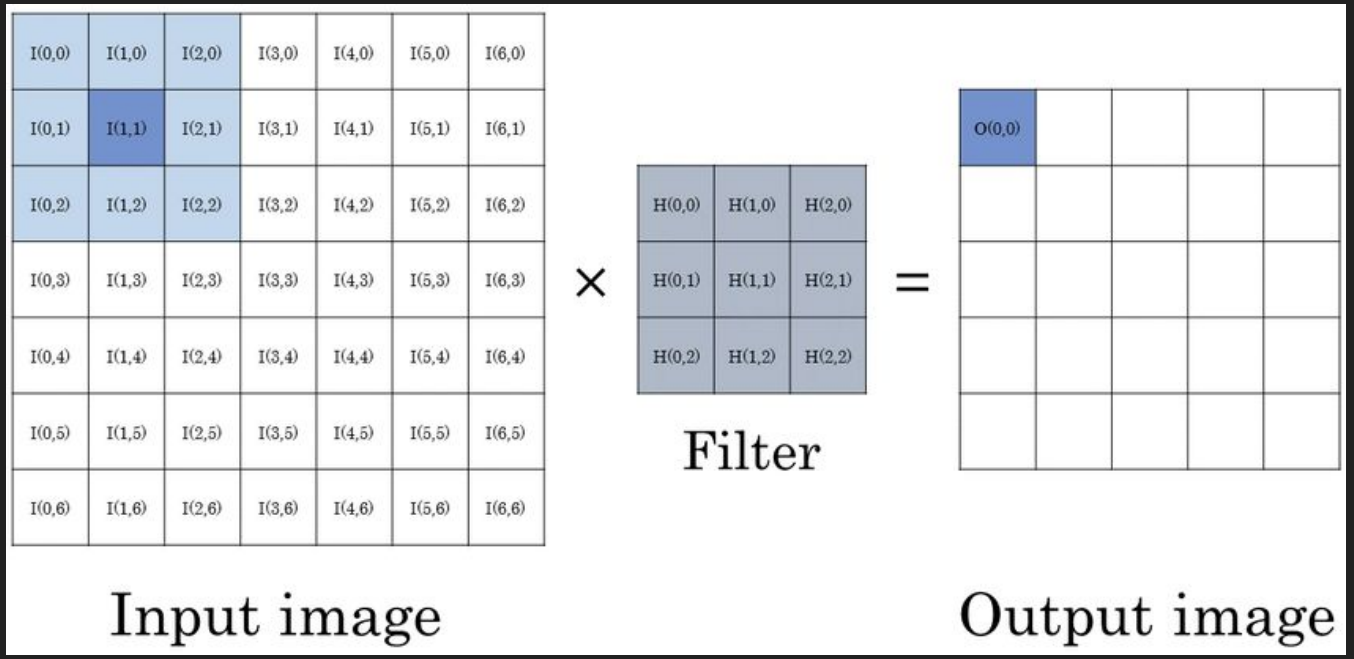
# Convolutional Neural Networks

The authors focused on a special kind of DNN, called “Convolutional Neural Network” (CNN), that is the most used approach when dealing with image-related tasks like:

- Image classification
- Image segmentation
- …

The name comes from the **convolution** operation, that is basically a filter applied pixel-wise to an image in order to map it into some intermediate representation, that is more useful to the network in order to perform the desired task.

# Convolution



# Layers in CNN

In CNNs we have several kinds of layers:

- Convolutional layers: apply the convolution operation on the input volume, producing an output volume which depth depends on the number of filters
- Activation layers: decide whether the output of the previous layer is relevant and whether it should be forwarded. A common activation function is ReLU.
- Pooling layers: perform a downsampling operation along the width and the height of the image, leaving the depth unchanged.
- Dense layers: fully connected layers that form the final part of the CNN.

The first three are combined and used for feature extraction, while fully connected (or dense) layers are used for classification.

# Collaborative environments

Deep Learning models need huge amounts of two things:

- Computational power
- Training data

In order to accelerate the learning and the prediction, **collaborative** systems have been designed, and we can find them in two different forms:

- Collaborative training
- Collaborative inference

This paper focuses on the latter.

# Collaborative Inference

In particular, we focus on the case of **edge-cloud scenario**, in which **edge devices**, such as smartphones, that usually do not have the computational power necessary for a whole DNN, hold the first, simple layers while the other, complex, layers are offloaded to a cloud server.

This leads to better privacy and data protection than the case when the processing is fully offloaded to the server, since it does not receive the whole input data but some intermediate representation of it.

# Overview

- Introduction
- Content of the paper
- Experiment reproduction
- Considerations

# Threat Model

The model is split between two participants:

- $P_1$ is trusted and performs the earlier layers $f_{\theta_1}$
- $P_2$ is not trusted and performs the latter layers $f_{\theta_2}$

$$f_\theta = f_{\theta_2} \circ f_{\theta_1}$$

# Experimental configuration

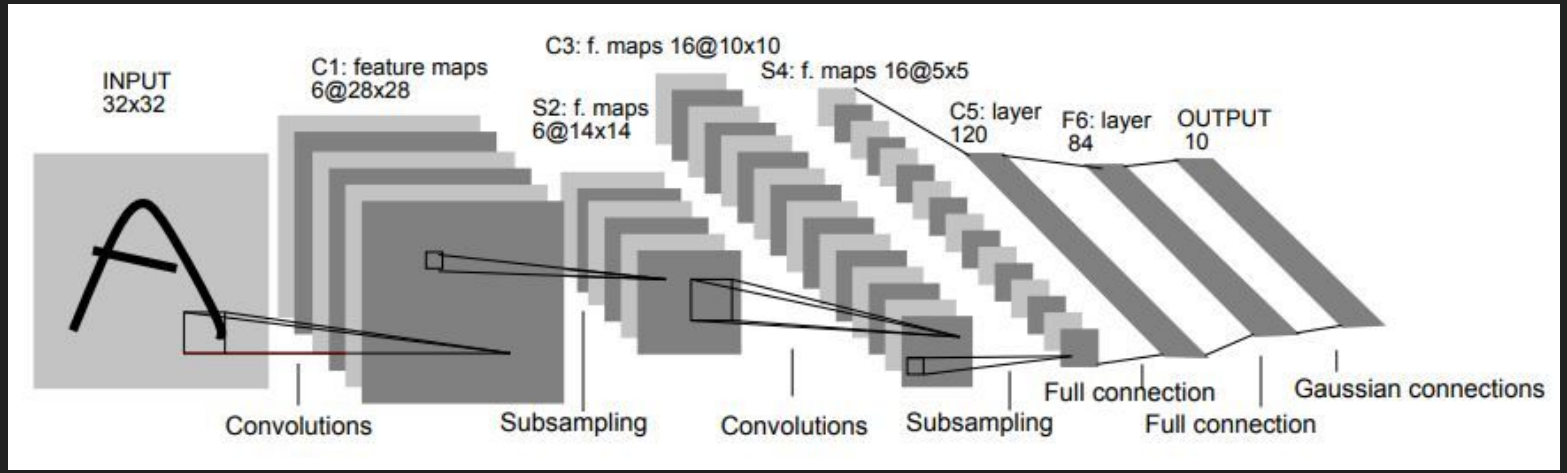
Adversary Capabilities:

| Setting | Target model | | Training Data | | Inference Query |
|---|---|---|---|---|---|
| | Parameters | Structure | Values | Distribution | |
| White-box | ✓ | ✓ | – | – | – |
| Black-box | – | – | ✓ | ✓ | ✓ |
| | – | – | – | ✓ | ✓ |
| | – | – | – | – | ✓ |
| Query-free | – | ✓ | ✓ | ✓ | – |
| | – | – | ✓ | ✓ | – |
| | – | ✓ | – | ✓ | – |
| | – | – | – | ✓ | – |

# Architectures

We'll use different architectures for the two datasets:

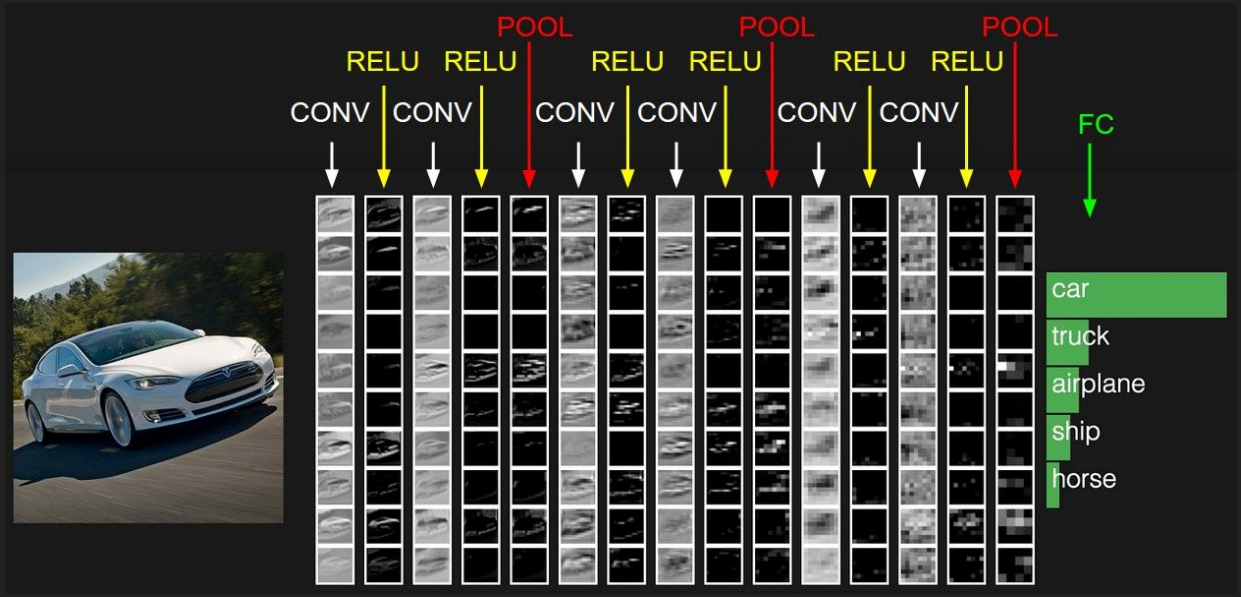
- MNIST: LeNet 5 architecture




http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf

# Architectures

- CIFAR10: custom convolutional architecture




https://cs231n.github.io/convolutional-networks/

# Experimental configuration

DNN Configuration:

| **Dataset** | MNIST | CIFAR10 |
|---|---|---|
| **Target Model** | LeNet-5<br>(2 conv + 3 fc) | 6 conv + 2 fc CNN |
| **Split point** | • 1st conv layer (conv1)<br>• 2nd conv layer after activation (ReLU2) | • 1st conv layer (conv11)<br>• 4th conv layer after activation (ReLU22)<br>• 6th conv layer after activation (ReLU32) |

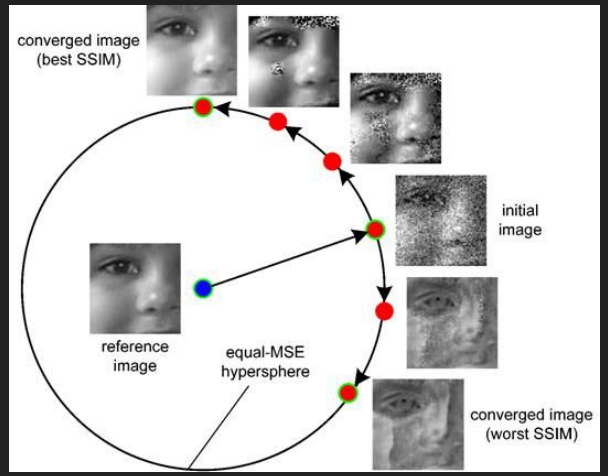
# Evaluation metrics

To evaluate the attack results, two metrics have been used:

PSNR (Peak-Signal-to-Noise-Ratio)

$$PSNR(I, I_0) = 10log(\frac{MAXI^2}{\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}(I(i,j) - I_0(i,j))^2})$$

SSIM (Structural-SIMilarity-index)




https://www.cns.nyu.edu/~lcv/ssim/

# Whitebox Attack

the attacker knows the parameters of the initial layers $f_{\theta_1}$ on the trusted participant

$$ED(x, x_0) = ||f_{\theta_1}(x) - f_{\theta_1}(x_0)||_2^2$$

$$TV(x) = \sum_{i,j}(|x_{i+1,j} - x_{i,j}|^2 + |x_{i,j+1} - x_{i,j}|^2)^{\beta/2}$$

**Algorithm 1** White-box model inversion attack

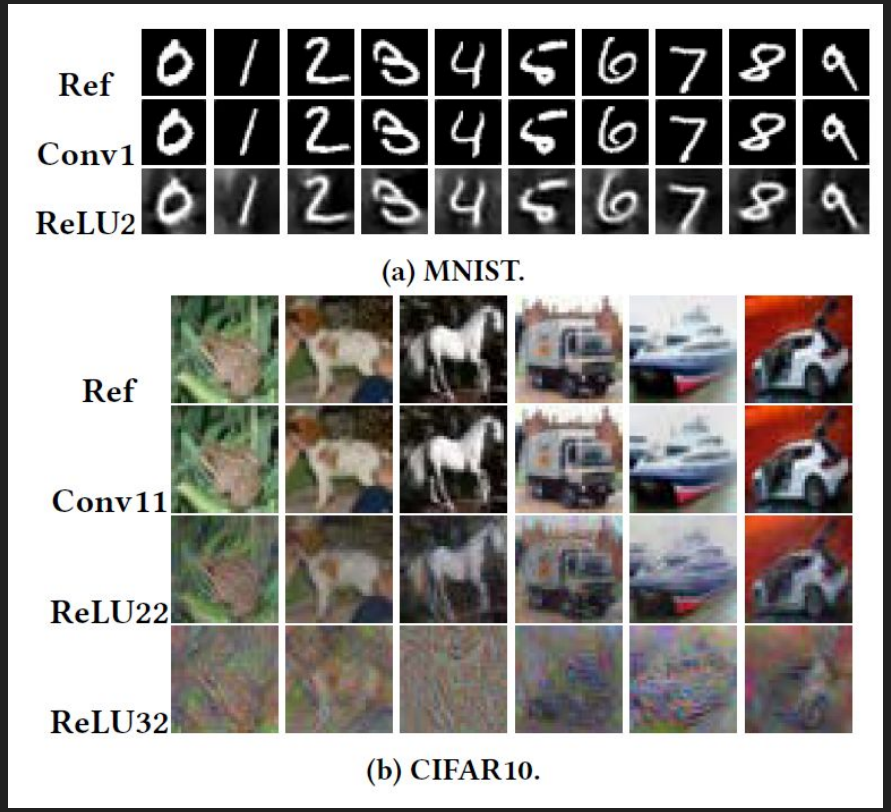
**Function** WhiteboxAttack($f_{\theta 1}$, $f_{\theta 1}(x_0)$, $T$, $\lambda$, $\epsilon$)
/* $f_{\theta 1}$: the target model */
/* $f_{\theta 1}(x_0)$: the intermediate output of sensitive input $x_0$ */
/* $T$: maximum number of iterations */
/* $\lambda$: tradeoff between prior and posteriori information
/* $\epsilon$: step size in GD */

$L(x)=||f_{\theta 1}(x) - f_{\theta 1}(x_0)||_2^2 + \lambda TV(x)$
$t = 0$
$x^{(0)}$= ConstantInit()
**while** ($t < T$) **do**
    $x^{(t+1)} = x^{(t)} - \epsilon * \frac{\partial L(x^{(t)})}{\partial x^{(t)}}$
    $t$+=1
**end for**
**return** $x^{(T)}$

# Whitebox Attack

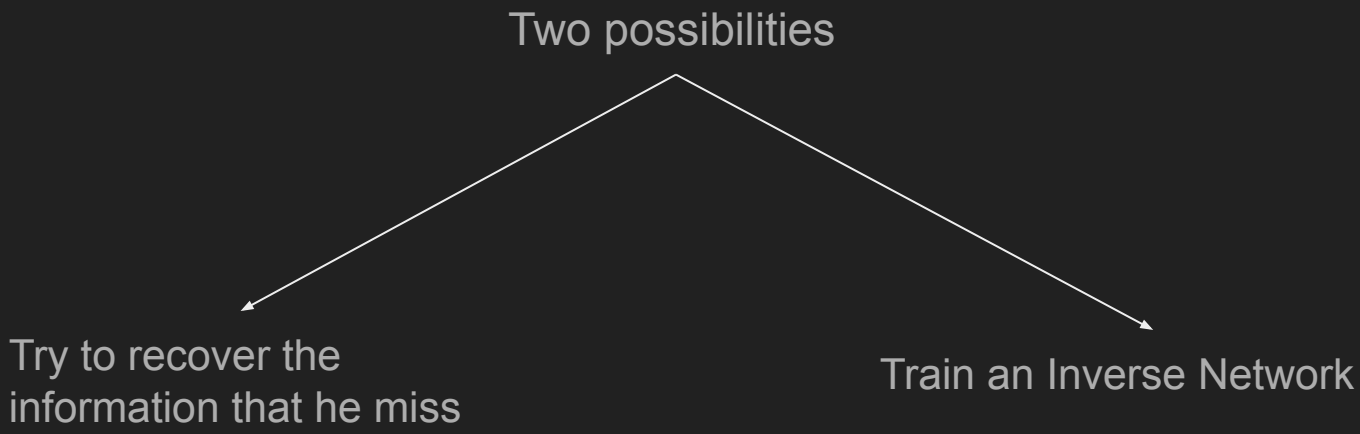
Results:

| | MNIST | | CIFAR10 | | |
|---|---|---|---|---|---|
| | conv1 | ReLU2 | conv11 | ReLU22 | ReLU32 |
| PSNR | 39.69 | 15.10 | 37.59 | 19.47 | 13.38 |
| SSIM | 0.9969 | 0.5998 | 0.9960 | 0.6940 | 0.1625 |



# Blackbox Attack

The attacker does not know the structure or the parameters of $f_{\theta_1}$ but can query the trusted participant



# Blackbox Attack

3 cases:

```
10: Function GenerateTrainingSet()
11: if known training set then
12:     X=data.TrainingSet
13: else if Known training distribution then
14:     X= NewSet ~ data.TrainingSet
15: else
16:     X=GaussianNoise
17: return X
```

1. same training set & distribution
2. same distribution & different set
3. different distribution (random set)

**Algorithm 2** Black-box model inversion attack

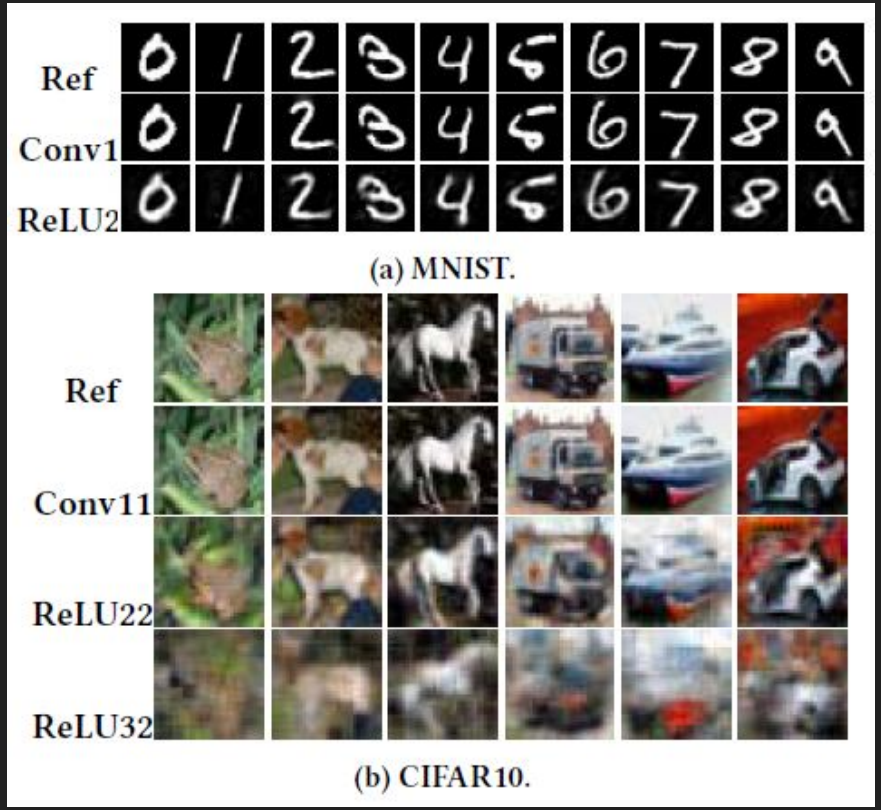
```
1: Function BlackBoxAttack(f_θ1, f_θ1(x_0))
2: /* f_θ1: the target model */
3: /* x_0: the target sensitive input to recover */
4: /* f_θ1(x_0): the intermediate layer output */
5: X=GenerateTrainingSet()
6: g=TrainInverseNet(X, f_θ1)
7: x̂_0 = Inverse(g, f_θ1(x_0))
8: return x̂_0
9:
18:
19: Function TrainInverseNet(X, f_θ1)
20: /* k: BatchSize */
21: /* ϵ: StepSize */
22: g^(0)=Init()
23: while (t < T) do
24:     Randomly sample x_1, x_2...x_k from X
25:     L(g^(t)=1/k Σ_{i=1}^{k} ||g(f_θ1(x_i)) − x_i||_2^2
26:     g^(t+1) = g^(t) − ϵ * ∂L(g^(t))/∂g^(t)
27:     t+=1
28: end while
29: return g^(T)
30:
31: Function Inverse(g, f_θ1(x_0))
32: x̂_0=g(f_θ1(x_0))
33: return x̂_0
```

# Blackbox Attack

Results:

1. Same training data (set)

| | Dataset | MNIST | | CIFAR10 | | |
|---|---|---|---|---|---|---|
| | | conv1 | ReLU2 | conv11 | ReLU22 | ReLU32 |
| PSNR | same set | 39.64 | 20.35 | 49.88 | 19.81 | 15.42 |
| | same dist | 40.72 | 20.81 | 49.02 | 19.36 | 13.95 |
| | rand set | 14.76 | 7.72 | 48.59 | 12.79 | 12.37 |
| SSIM | same set | 0.9887 | 0.7334 | 0.9993 | 0.6939 | 0.3124 |
| | same dist | 0.9950 | 0.8046 | 0.9992 | 0.6802 | 0.2196 |
| | rand set | 0.7188 | 0.4310 | 0.9996 | 0.2930 | 0.0440 |

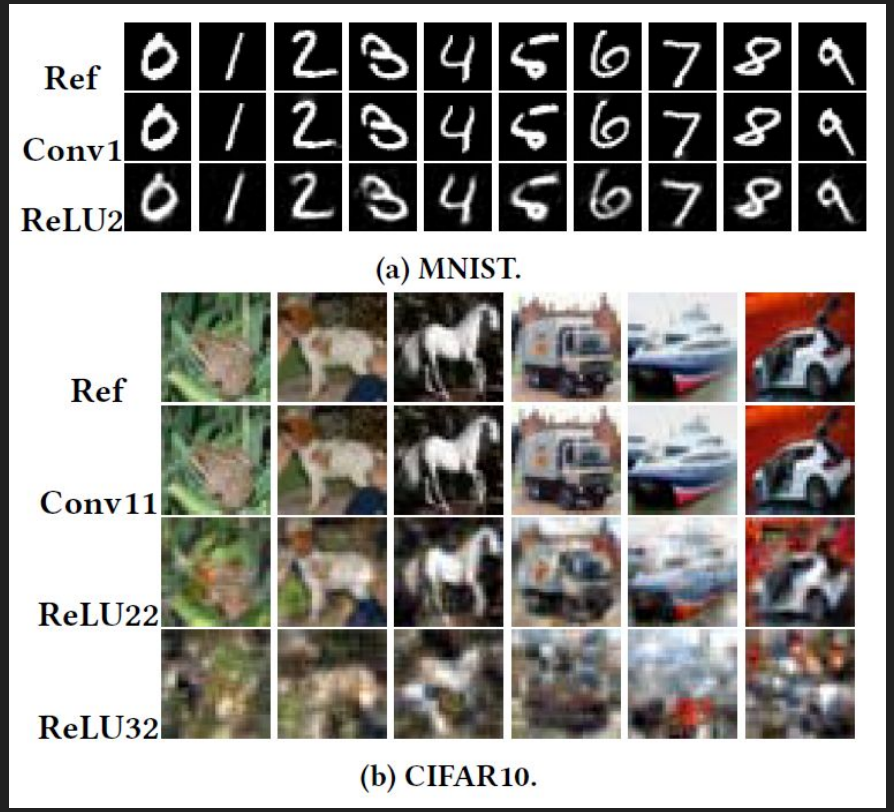


# Blackbox Attack

Results:

2. Same distribution

| | Dataset | MNIST | | CIFAR10 | | |
|---|---|---|---|---|---|---|
| | | conv1 | ReLU2 | conv11 | ReLU22 | ReLU32 |
| PSNR | same set | 39.64 | 20.35 | 49.88 | 19.81 | 15.42 |
| | same dist | 40.72 | 20.81 | 49.02 | 19.36 | 13.95 |
| | rand set | 14.76 | 7.72 | 48.59 | 12.79 | 12.37 |
| SSIM | same set | 0.9887 | 0.7334 | 0.9993 | 0.6939 | 0.3124 |
| | same dist | 0.9950 | 0.8046 | 0.9992 | 0.6802 | 0.2196 |
| | rand set | 0.7188 | 0.4310 | 0.9996 | 0.2930 | 0.0440 |

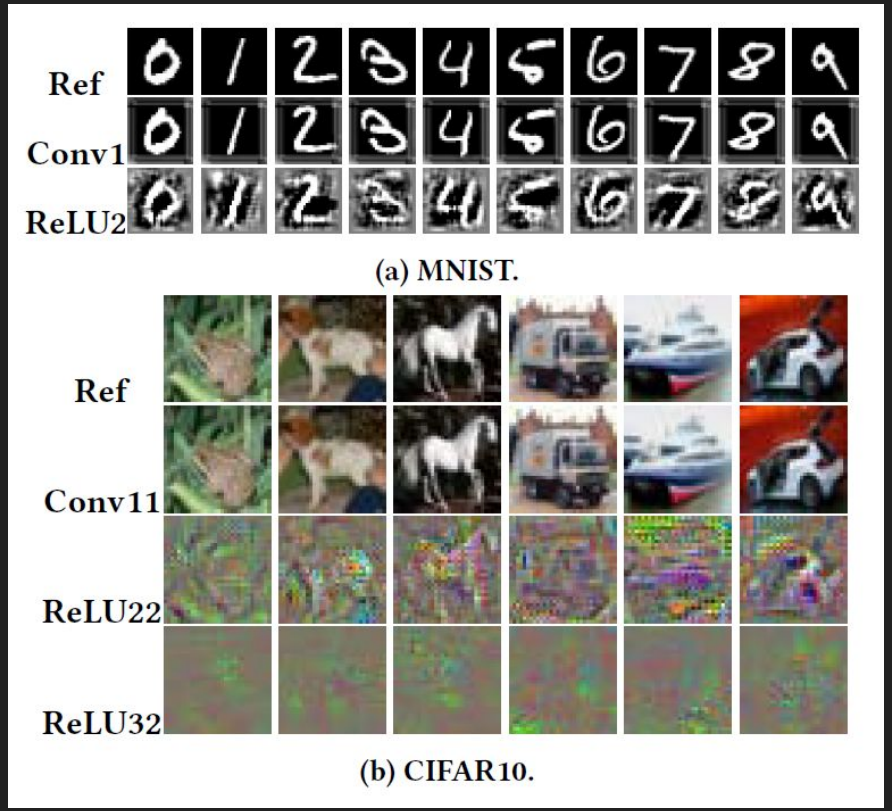


# Blackbox Attack

Results:

3. Same distribution

| | Dataset | MNIST | | CIFAR10 | | |
|---|---|---|---|---|---|---|
| | | conv1 | ReLU2 | conv11 | ReLU22 | ReLU32 |
| PSNR | same set | 39.64 | 20.35 | 49.88 | 19.81 | 15.42 |
| | same dist | 40.72 | 20.81 | 49.02 | 19.36 | 13.95 |
| | rand set | 14.76 | 7.72 | 48.59 | 12.79 | 12.37 |
| SSIM | same set | 0.9887 | 0.7334 | 0.9993 | 0.6939 | 0.3124 |
| | same dist | 0.9950 | 0.8046 | 0.9992 | 0.6802 | 0.2196 |
| | rand set | 0.7188 | 0.4310 | 0.9996 | 0.2930 | 0.0440 |



# Query-free Attack

The attacker cannot query the model on the trusted participant: $f_{\theta_1}$
he does not know the client-side model information

Shadow model reconstruction

# Query-free Attack

Configuration of the training:

1. same dataset, same network structure
2. same distribution, different dataset
3. same dataset, unknown net. structure:

| Dataset | Layer | Target Model | Shadow Model |
|---|---|---|---|
| MNIST | conv1 | One 5X5 conv layer | Two 3X3 conv layers |
| | ReLU2 | Two 5X5 conv layers | Four 3X3 conv layers |
| CIFAR10 | conv11 | One 3X3 64 filters layer | One 3X3 16 filters layer + one 3X3 64 filters layer |
| | ReLU22 | Two 3X3 64 filters layers + two 3X3 128 filters layers | One 5X5 filters layer + one 5X5 128 filters layer |

4. knows nothing

**Algorithm 3** Query-free model inversion attack

**Function** QueryFreeAttack($f_{\theta 1}$, $f_{\theta 2}$, $f_{\theta 1}(x_0)$)
/* $f_{\theta 1}$: the target model */
/* $f_{\theta 2}$: the known model */
/* $x_0$: the target sensitive input to recover */
/* $f_{\theta 1}(x_0)$ the intermediate layer output */
$\hat{f_{\theta 1}}$=ModelReconstruction($S$, $f_{\theta 2}$)
$\hat{x_0}$ = WhiteboxAttack($\hat{f_{\theta 1}}$, $f_{\theta 1}(x_0)$, $T$, $\lambda$, $\epsilon$)
**return** $\hat{x_0}$

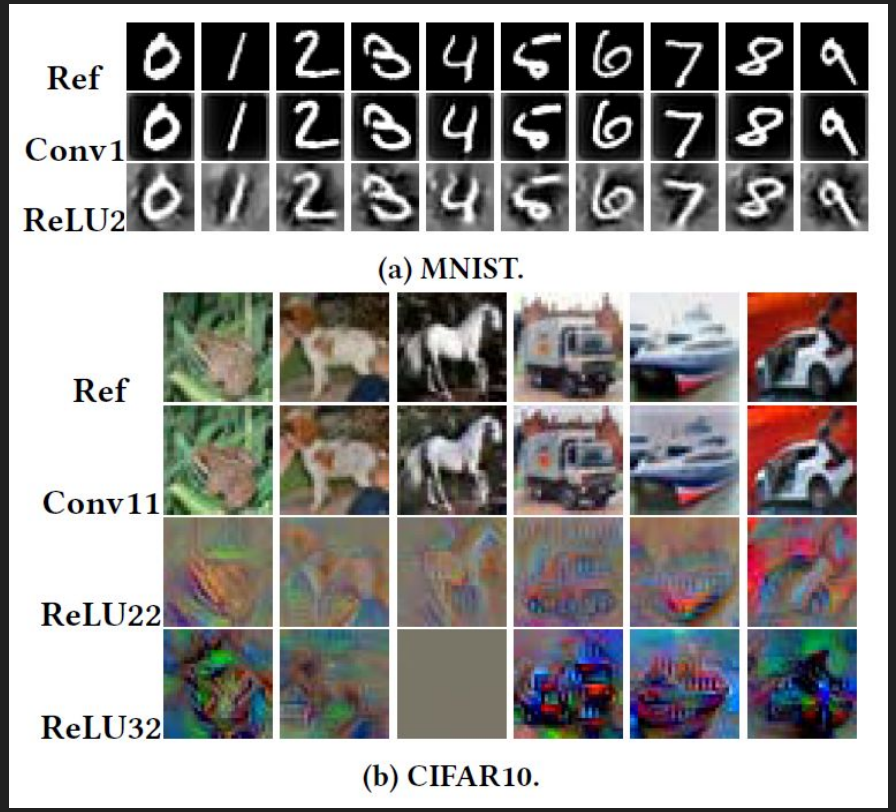
**Function** ModelReconstruction($f_{\theta 2}$)
/* $k$: BatchSize */
/* $\epsilon$: StepSize */
$S$=GenerateTrainingSet()
$g_0$=InitArchitecture()
**while** $(t < T)$ **do**
    Randomly sample $x_1, x_2 ... x_k$ and labels $y_1, y_2 ... y_k$ from $S$
    $L(g^t) = \frac{1}{k} \sum_{i=1}^{k} y_i (f_{\theta 2}(g^t(x_i)) + (1 - y_i)(1 - f_{\theta 2}(g^t(x_i))$
    $g^{(t+1)} = g^t - \epsilon * \frac{\partial L(g^t)}{\partial g^t}$
    $t$+=1
**end while**
**return** $g^{(T)}$

# Query-free Attack

Results:

1. same set, same network structure

| | Dataset / Net Structure | MNIST | | CIFAR10 | | |
|---|---|---|---|---|---|---|
| | | conv1 | ReLU2 | conv11 | ReLU22 | ReLU32 |
| PSNR | same set, same net | 17.60 | 9.61 | 21.16 | 12.74 | 11.09 |
| | diff sets, same net | 21.53 | 9.27 | 21.45 | 11.51 | 11.98 |
| | same sets, diff nets | 12.59 | 8.05 | 17.55 | 12.46 | 10.68 |
| | diff sets, diff nets | 17.86 | 8.03 | 13.06 | 11.30 | 11.47 |
| SSIM | same set, same net | 0.7423 | 0.4981 | 0.9104 | 0.1752 | 0.0419 |
| | diff sets, same net | 0.9121 | 0.4652 | 0.9145 | 0.1723 | 0.0102 |
| | same sets, diff nets | 0.6430 | 0.3790 | 0.6344 | 0.2714 | 0.0247 |
| | diff sets, diff nets | 0.6952 | 0.3226 | 0.1553 | 0.0467 | 0.0793 |

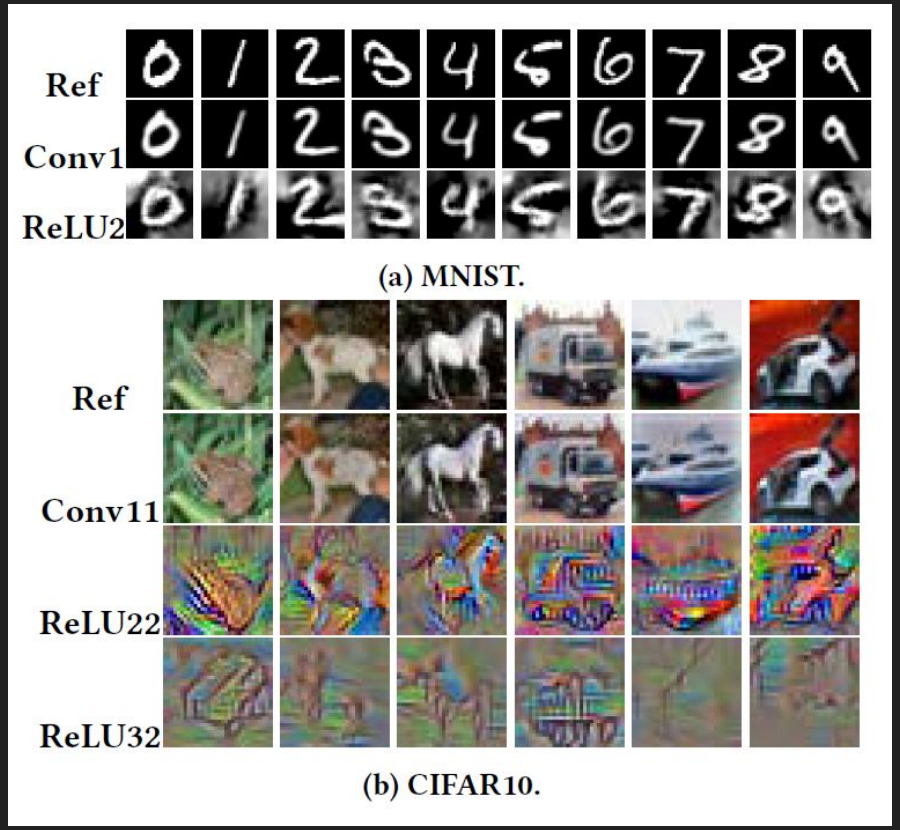


# Query-free Attack

Results:

2. different dataset, same distribution

| | Dataset / Net Structure | MNIST | | CIFAR10 | | |
|---|---|---|---|---|---|---|
| | | conv1 | ReLU2 | conv11 | ReLU22 | ReLU32 |
| PSNR | same set, same net | 17.60 | 9.61 | 21.16 | 12.74 | 11.09 |
| | diff sets, same net | 21.53 | 9.27 | 21.45 | 11.51 | 11.98 |
| | same sets, diff nets | 12.59 | 8.05 | 17.55 | 12.46 | 10.68 |
| | diff sets, diff nets | 17.86 | 8.03 | 13.06 | 11.30 | 11.47 |
| SSIM | same set, same net | 0.7423 | 0.4981 | 0.9104 | 0.1752 | 0.0419 |
| | diff sets, same net | 0.9121 | 0.4652 | 0.9145 | 0.1723 | 0.0102 |
| | same sets, diff nets | 0.6430 | 0.3790 | 0.6344 | 0.2714 | 0.0247 |
| | diff sets, diff nets | 0.6952 | 0.3226 | 0.1553 | 0.0467 | 0.0793 |

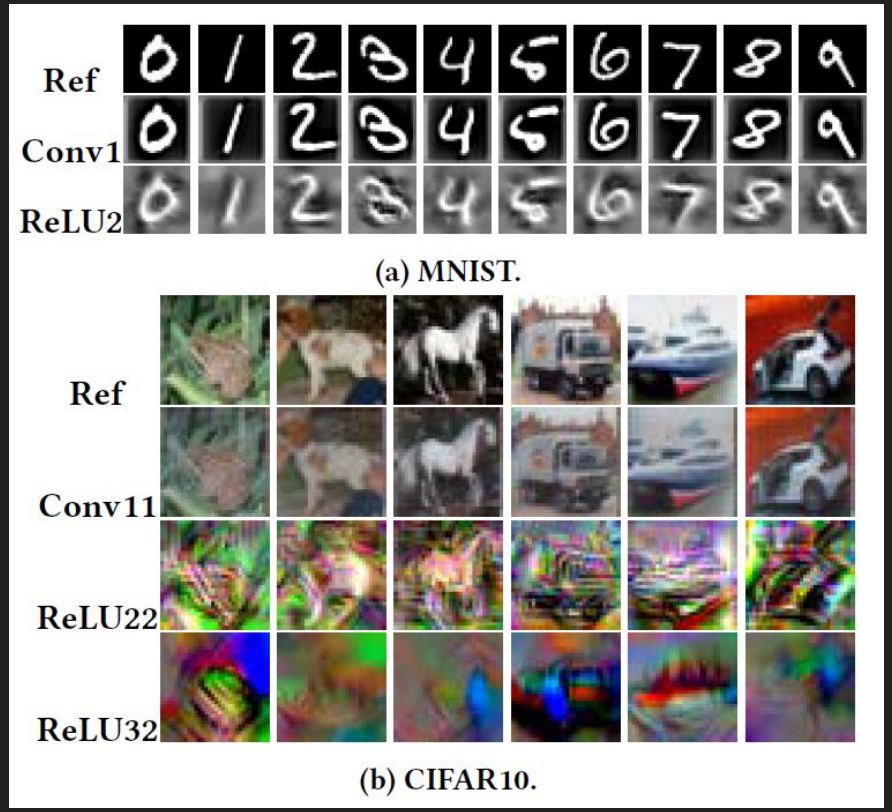


# Query-free Attack

Results:

3. same training, unknown structure

| | Dataset / Net Structure | MNIST | | CIFAR10 | | |
|---|---|---|---|---|---|---|
| | | conv1 | ReLU2 | conv11 | ReLU22 | ReLU32 |
| PSNR | same set, same net | 17.60 | 9.61 | 21.16 | 12.74 | 11.09 |
| | diff sets, same net | 21.53 | 9.27 | 21.45 | 11.51 | 11.98 |
| | same sets, diff nets | 12.59 | 8.05 | 17.55 | 12.46 | 10.68 |
| | diff sets, diff nets | 17.86 | 8.03 | 13.06 | 11.30 | 11.47 |
| SSIM | same set, same net | 0.7423 | 0.4981 | 0.9104 | 0.1752 | 0.0419 |
| | diff sets, same net | 0.9121 | 0.4652 | 0.9145 | 0.1723 | 0.0102 |
| | same sets, diff nets | 0.6430 | 0.3790 | 0.6344 | 0.2714 | 0.0247 |
| | diff sets, diff nets | 0.6952 | 0.3226 | 0.1553 | 0.0467 | 0.0793 |

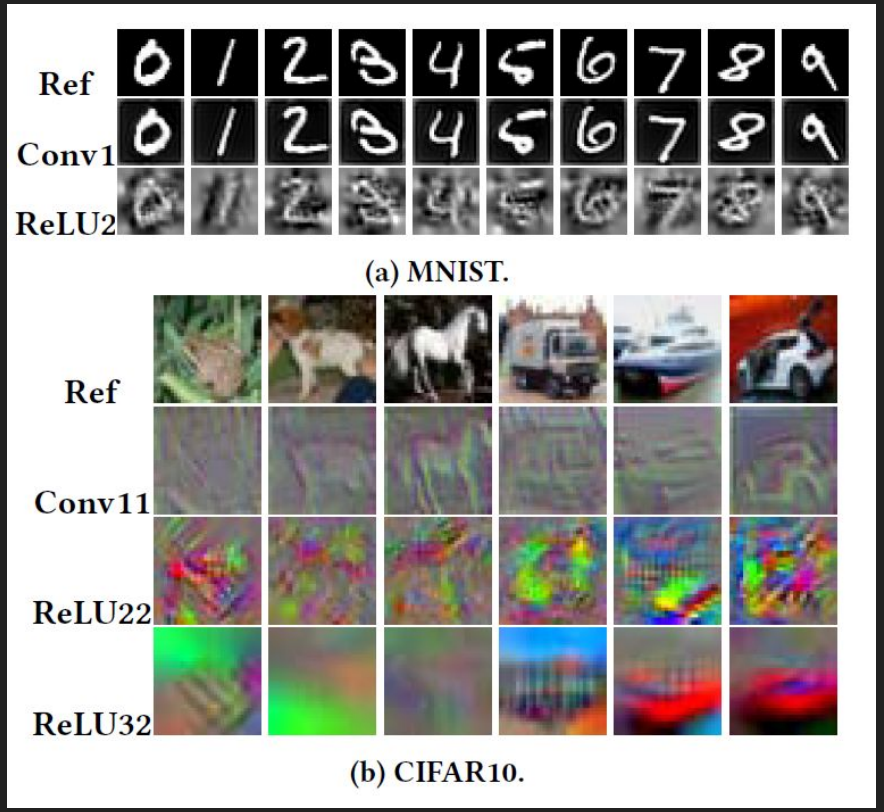


# Query-free Attack

Results:

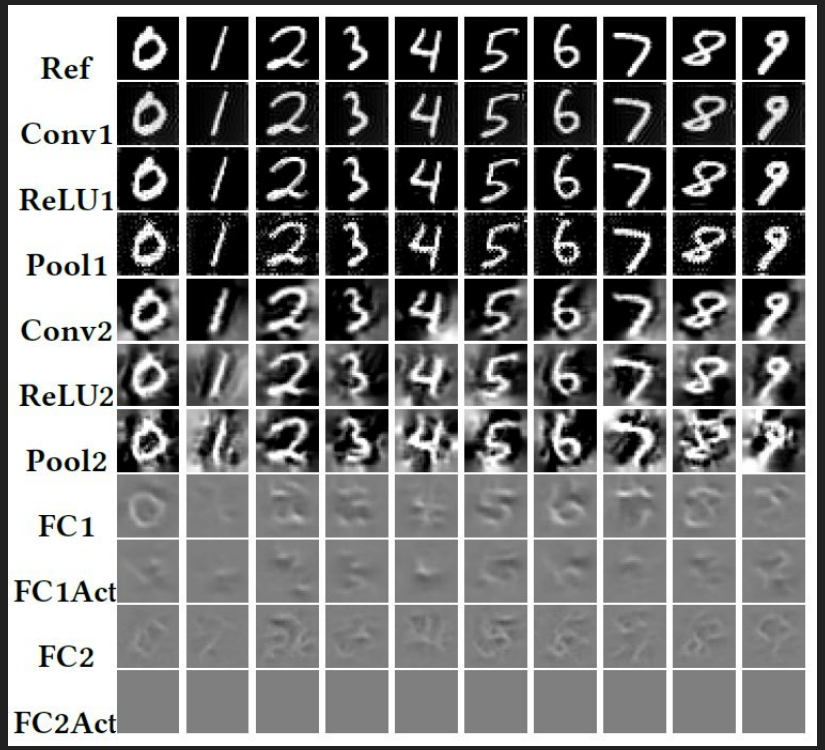
4. know nothing

| | Dataset / Net Structure | MNIST | | CIFAR10 | | |
|---|---|---|---|---|---|---|
| | | conv1 | ReLU2 | conv11 | ReLU22 | ReLU32 |
| PSNR | same set, same net | 17.60 | 9.61 | 21.16 | 12.74 | 11.09 |
| | diff sets, same net | 21.53 | 9.27 | 21.45 | 11.51 | 11.98 |
| | same sets, diff nets | 12.59 | 8.05 | 17.55 | 12.46 | 10.68 |
| | diff sets, diff nets | 17.86 | 8.03 | 13.06 | 11.30 | 11.47 |
| SSIM | same set, same net | 0.7423 | 0.4981 | 0.9104 | 0.1752 | 0.0419 |
| | diff sets, same net | 0.9121 | 0.4652 | 0.9145 | 0.1723 | 0.0102 |
| | same sets, diff nets | 0.6430 | 0.3790 | 0.6344 | 0.2714 | 0.0247 |
| | diff sets, diff nets | 0.6952 | 0.3226 | 0.1553 | 0.0467 | 0.0793 |



# Impacts of the System Configuration

How to split the neural network in the collaborative system to make the inference procedure more secure?

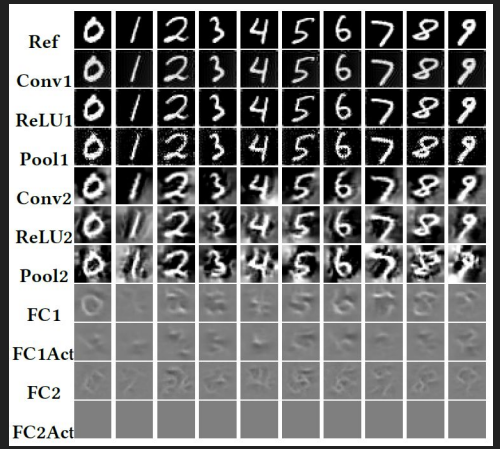


# Impact of Adversary’s Capabilities

- Knowledge of target model
  - as a whole
  - only of the structure of the network
- Knowledge of training dataset
  - its set
  - its distribution
- Capability of model query

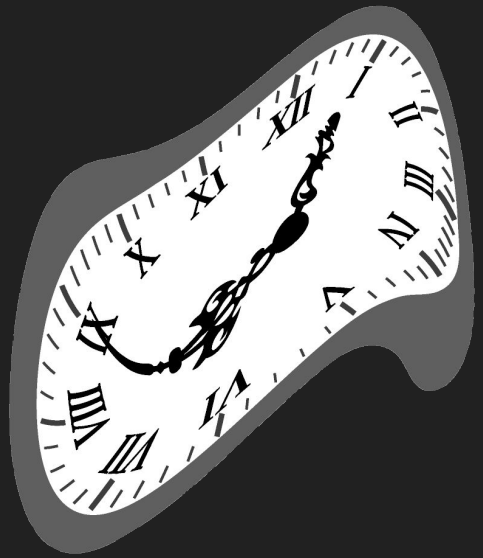
# Defenses

- Net structure:
  - Running fully-connected layers before sending out results
  - Make the client-side network deeper
- Execution environment:
  - Trusted Execution on untrusted participants (TEE)
- Data:
  - Differential privacy
  - Homomorphic encryption



# Overview

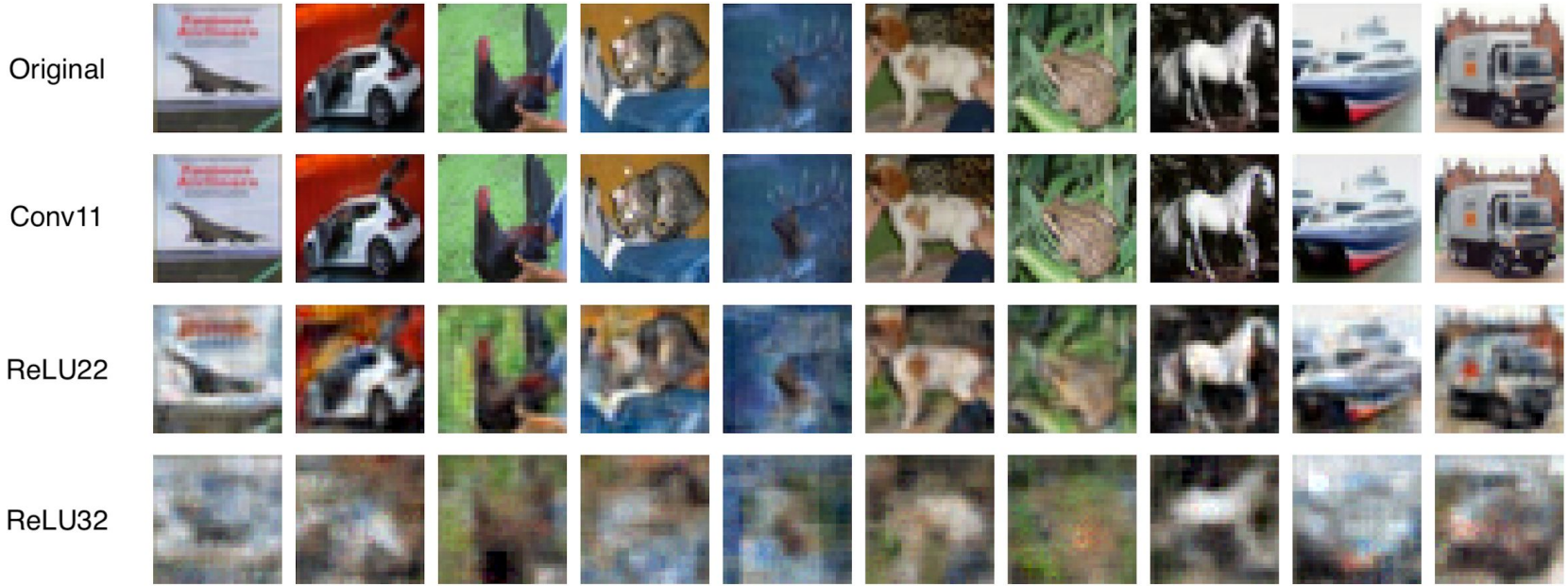
- Introduction
- Content of the paper
- Experiment reproduction
- Considerations



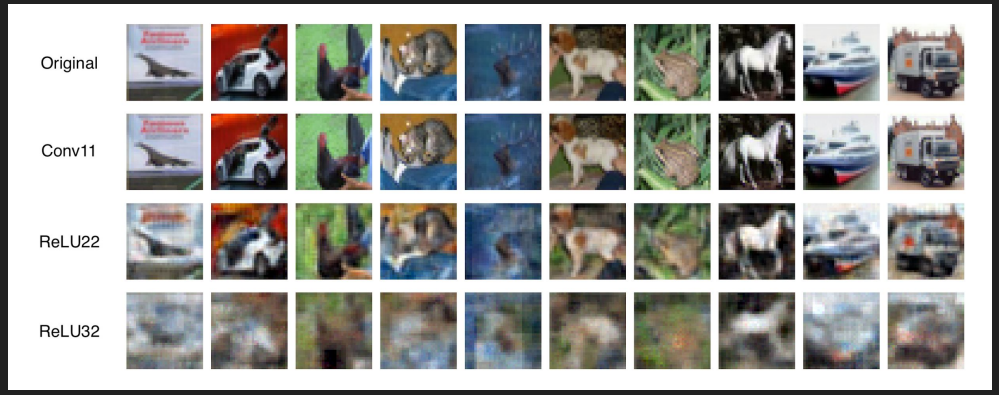
# Whitebox Attack (CIFAR10)

# Whitebox Attack - Comparison

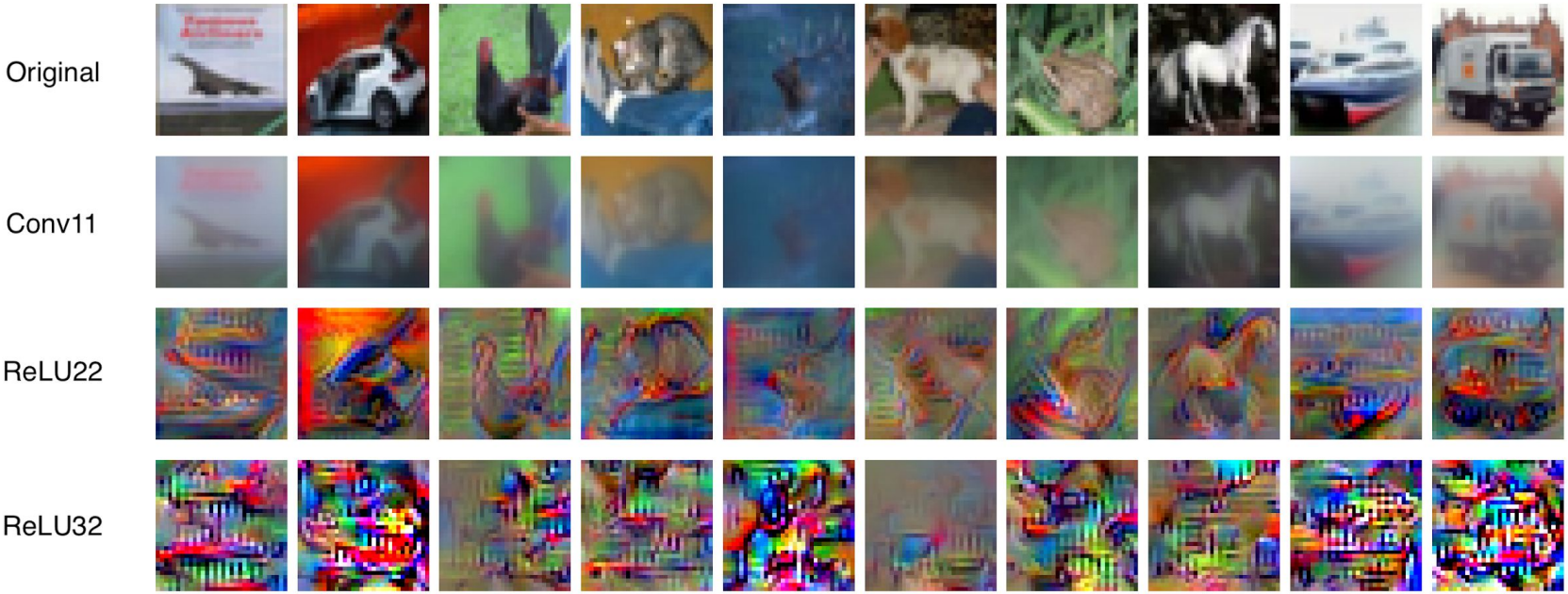

# Blackbox Attack



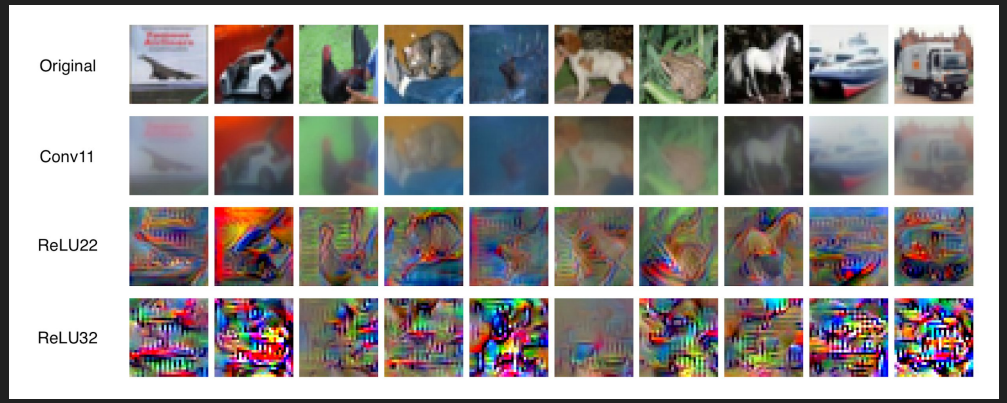
# Blackbox Attack - Comparison




# Query-free Attack



# Query-free attack - Comparison




# Overview

- Introduction
- Content of the paper
- Experiment reproduction
- Considerations
  - Experiments reproduction
  - Paper
  - Suggestion for future research

# Considerations - Experiments reproduction

- Experiments are not reproducible on MNIST (there is no code for that).
- FC1, FC2 layers (even if not so useful) are not available for the reproduction of blackbox and query-free attacks.
- Need for powerful hardware, they could have provided the models directly in order to allow everyone to reproduce the attacks.
    - Trained models: https://github.com/AlessandroCorra/trained-models-for-Model-Inversion-Attack-against-Collaborative-Inference
- Code is not commented at all, it is difficult to understand what they're doing, especially since there are 2000+ lines of code.
- They provide sample commands for the reproduction of the experiment, but don't explain what the parameters are or what they do.

# Considerations - Paper

- The introduction about DNNs is very general, and CNNs, that are used throughout the whole paper, are barely mentioned.
- The rest of the paper is clear, the explanations of the attacks are detailed and both pseudocode and formulas are clarified.
- The authors present a possible problem, they prove that it is a problem and they also suggest possible solutions.

# Suggestions for future research

The authors could investigate the effectiveness of the attacks in real-life scenarios, where the images that are fed to the neural networks are much larger than the ones used in the experiment.

They could also investigate alternative scenarios like different categories of data or different tasks performed by the neural network (e.g. linear regression).

# Questions?