Fast Homomorphic Evaluation of Deep Discretized Neural Networks

Florian Bourse Michele Minelli Matthias Minihold Pascal Paillier

ENS, CNRS, PSL Research University, INRIA
(Work done while visiting CryptoExperts)

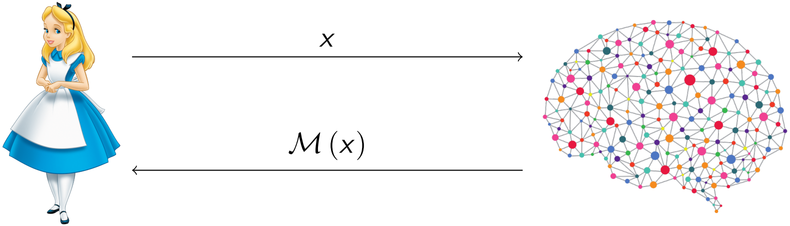
CRYPTO 2018 – UCSB, Santa Barbara

Machine Learning as a Service (MLaaS)

Machine Learning as a Service (MLaaS)

$\xrightarrow{\quad x \quad}$

Machine Learning as a Service (MLaaS)



Machine Learning as a Service (MLaaS)

Alice's privacy!

?

Possible solution: FHE.

Machine Learning as a Service (MLaaS)

$$\xrightarrow{\mathsf{Enc}(x)}$$

Possible solution: FHE.

Machine Learning as a Service (MLaaS)

$$\xrightarrow{\mathrm{Enc}\,(x)}$$

$$\xleftarrow{\mathrm{Enc}\,(\mathcal{M}\,(x))}$$

Possible solution: FHE.

Machine Learning as a Service (MLaaS)

$$\xrightarrow{\mathsf{Enc}\left(x\right)}$$

$$\xleftarrow{\mathsf{Enc}\left(\mathcal{M}\left(x\right)\right)}$$

Possible solution: FHE.

- ✓ Privacy — data is encrypted (both input and output)
- ✗ Efficiency — main issue with FHE-based solutions

Machine Learning as a Service (MLaaS)

$\xrightarrow{\mathsf{Enc}(x)}$

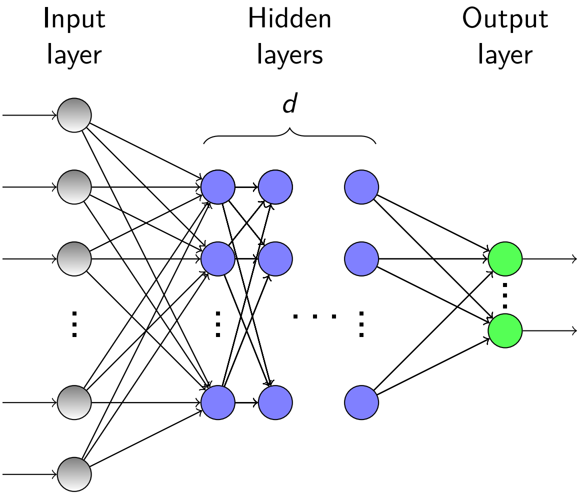
$\xleftarrow{\mathsf{Enc}(\mathcal{M}(x))}$

Possible solution: FHE.

✓	Privacy	data is encrypted (both input and output)
✗	Efficiency	main issue with FHE-based solutions

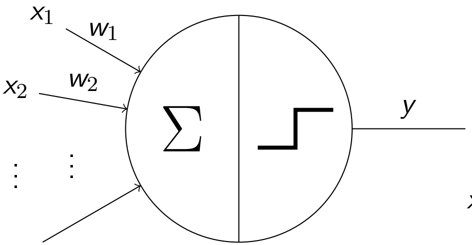
Goal of this work: homomorphic *evaluation* of trained networks.

(Very quick) refresher on neural networks



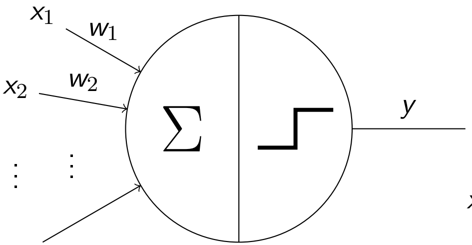
(Very quick) refresher on neural networks

Computation for every neuron:



$x_i, w_i, y \in \mathbb{R}$

Computation for every neuron:



$x_i, w_i, y \in \mathbb{R}$

$$y = f\left(\sum_i w_i x_i\right),$$

where f is an *activation function*.

A specific use case

We consider the problem of *digit recognition*.

A specific use case

We consider the problem of *digit recognition*.

 $\longrightarrow$  $\longrightarrow$ 7

A specific use case

We consider the problem of *digit recognition*.

A specific use case

We consider the problem of *digit recognition*.

Dataset: MNIST ($60\,000$ training img + $10\,000$ test img).

Cryptonets [DGBL+16]

Cryptonets [DGBL+16]

✓ Achieves blind, non-interactive classification

Cryptonets [DGBL$^+$16]

- ✓ Achieves blind, non-interactive classification
- ✓ Near state-of-the-art accuracy (98.95%)

Cryptonets [DGBL$^+$16]

- ✓ Achieves blind, non-interactive classification
- ✓ Near state-of-the-art accuracy (98.95%)
- ✗ Replaces sigmoidal activ. functions with low-degree $f(x) = x^2$

Cryptonets [DGBL[+]16]

- ✓ Achieves blind, non-interactive classification
- ✓ Near state-of-the-art accuracy (98.95%)
- ✗ Replaces sigmoidal activ. functions with low-degree $f(x) = x^2$
- ✗ Uses SHE $\implies$ parameters have to be chosen at setup time

Cryptonets [DGBL$^+$16]

✓ Achieves blind, non-interactive classification

✓ Near state-of-the-art accuracy (98.95%)

✗ Replaces sigmoidal activ. functions with low-degree $f(x) = x^2$

✗ Uses SHE $\implies$ parameters have to be chosen at setup time

Main limitation

The computation at neuron level depends on the total multiplicative depth of the network $\implies$ bad for deep networks!

Cryptonets [DGBL$^+$16]

✓ Achieves blind, non-interactive classification

✓ Near state-of-the-art accuracy (98.95%)

✗ Replaces sigmoidal activ. functions with low-degree $f(x) = x^2$

✗ Uses SHE $\implies$ parameters have to be chosen at setup time

Main limitation

The computation at neuron level depends on the total multiplicative depth of the network $\implies$ bad for deep networks!

Goal: make the computation scale-invariant $\implies$ bootstrapping.

A restriction on the model

We want to homomorphically compute the multisum

$$\sum_i w_i x_i$$

A restriction on the model

We want to homomorphically compute the multisum

$$\sum_i w_i x_i$$

Given $w_1, \ldots, w_p$ and $\mathsf{Enc}(x_1), \ldots, \mathsf{Enc}(x_p)$, do

$$\sum_i w_i \cdot \mathsf{Enc}(x_i)$$

A restriction on the model

We want to homomorphically compute the multisum

$$\sum_i w_i x_i$$

Given $w_1, \ldots, w_p$ and $\mathsf{Enc}(x_1), \ldots, \mathsf{Enc}(x_p)$, do

$$\sum_i w_i \cdot \mathsf{Enc}(x_i)$$

Proceed with caution

In order to maintain correctness, we need $w_i \in \mathbb{Z}$

A restriction on the model

We want to homomorphically compute the multisum

$$\sum_i w_i x_i$$

Given $w_1, \ldots, w_p$ and $\mathsf{Enc}(x_1), \ldots, \mathsf{Enc}(x_p)$, do

$$\sum_i w_i \cdot \mathsf{Enc}(x_i)$$

Proceed with caution

In order to maintain correctness, we need $w_i \in \mathbb{Z} \implies$ trade-off efficiency vs. accuracy!

Discretized neural networks (DiNNs)

Goal: *FHE-friendly* model of neural network.

Discretized neural networks (DiNNs)

Goal: *FHE-friendly* model of neural network.

Definition

A DiNN is a neural network whose inputs are integer values in $\{-I, \ldots, I\}$, and whose weights are integer values in $\{-W, \ldots, W\}$, for some $I, W \in \mathbb{N}$.

For every activated neuron of the network, the activation function maps the multisum to integer values in $\{-I, \ldots, I\}$.

Discretized neural networks (DiNNs)

Goal: *FHE-friendly* model of neural network.

Definition

A DiNN is a neural network whose inputs are integer values in $\{-I, \dots, I\}$, and whose weights are integer values in $\{-W, \dots, W\}$, for some $I, W \in \mathbb{N}$.

For every activated neuron of the network, the activation function maps the multisum to integer values in $\{-I, \dots, I\}$.

- Not as restrictive as it seems: e.g., binarized NNs;

Discretized neural networks (DiNNs)

Goal: *FHE-friendly* model of neural network.

Definition

A DiNN is a neural network whose inputs are integer values in $\{-I, \ldots, I\}$, and whose weights are integer values in $\{-W, \ldots, W\}$, for some $I, W \in \mathbb{N}$.

For every activated neuron of the network, the activation function maps the multisum to integer values in $\{-I, \ldots, I\}$.

- Not as restrictive as it seems: e.g., binarized NNs;
- Trade-off between size and performance;

Discretized neural networks (DiNNs)

Goal: *FHE-friendly* model of neural network.

Definition

A DiNN is a neural network whose inputs are integer values in $\{-I, \ldots, I\}$, and whose weights are integer values in $\{-W, \ldots, W\}$, for some $I, W \in \mathbb{N}$.

For every activated neuron of the network, the activation function maps the multisum to integer values in $\{-I, \ldots, I\}$.

- Not as restrictive as it seems: e.g., binarized NNs;
- Trade-off between size and performance;
- (A basic) conversion is extremely easy.

Homomorphic evaluation of a DiNN

1. **Evaluate the multisum:** easy – just need a linearly hom. scheme

$$\sum_i w_i \cdot \mathrm{Enc}\left(x_i\right) = \mathrm{Enc}\left(\sum_i w_i x_i\right)$$

Homomorphic evaluation of a DiNN

1. **Evaluate the multisum:** easy – just need a linearly hom. scheme
2. **Apply the activation function:** depends on the function

$$\mathrm{Enc}\left(f\left(\sum_i w_i x_i\right)\right)$$

Homomorphic evaluation of a DiNN

1. **Evaluate the multisum:** easy – just need a linearly hom. scheme
2. **Apply the activation function:** depends on the function
3. **Bootstrap:** can be costly

$$\mathrm{Enc}^*\left(f\left(\sum_i w_i x_i\right)\right)$$

Homomorphic evaluation of a DiNN

1. **Evaluate the multisum:** easy – just need a linearly hom. scheme
2. **Apply the activation function:** depends on the function
3. **Bootstrap:** can be costly
4. **Repeat for all the layers**

$$\mathsf{Enc}^*\left(f\left(\sum_i w_i x_i\right)\right)$$

Homomorphic evaluation of a DiNN

1. **Evaluate the multisum:** easy – just need a linearly hom. scheme
2. **Apply the activation function:** depends on the function
3. **Bootstrap:** can be costly
4. **Repeat for all the layers**

Issues:

- **Choose the message space:** guess, statistics, or worst-case

Homomorphic evaluation of a DiNN

1. **Evaluate the multisum:** easy – just need a linearly hom. scheme
2. **Apply the activation function:** depends on the function
3. **Bootstrap:** can be costly
4. **Repeat for all the layers**

Issues:

- **Choose the message space:** guess, statistics, or worst-case
- **The noise grows:** need to start from a very small noise

Homomorphic evaluation of a DiNN

1. **Evaluate the multisum:** easy – just need a linearly hom. scheme
2. **Apply the activation function:** depends on the function
3. **Bootstrap:** can be costly
4. **Repeat for all the layers**

Issues:

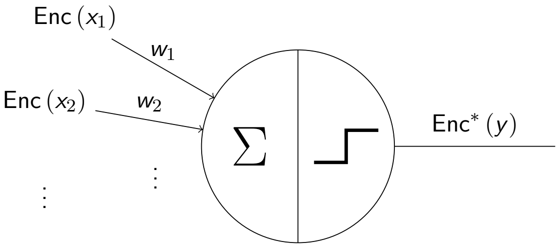
- **Choose the message space:** guess, statistics, or worst-case
- **The noise grows:** need to start from a very small noise
- **How do we apply the activation function homomorphically?**

Basic idea: activate during bootstrapping

Combine bootstrapping & activation function:

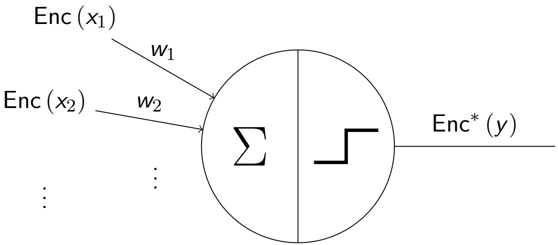
$$\mathsf{Enc}\left(x\right) \rightarrow \mathsf{Enc}^{*}\left(f\left(x\right)\right)$$

Basic idea: activate during bootstrapping



$$y = f\left(\sum_i w_i x_i\right)$$

Basic idea: activate during bootstrapping

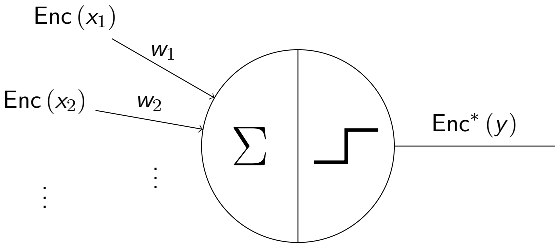


$$y = f\left(\sum_i w_i x_i\right)$$

Two steps:

1. Compute the multisum $\sum_i w_i x_i$

Basic idea: activate during bootstrapping



$$y = f\left(\sum_i w_i x_i\right)$$

Two steps:

1. Compute the multisum $\sum_i w_i x_i$
2. Bootstrap to the activated value

TFHE: a framework for faster bootstrapping [CGGI16,CGGI17]

$\boxed{\mathbb{T} := \mathbb{R}/\mathbb{Z}}$

Basic assumption: learning with errors (LWE) over the torus

$$(\mathbf{a},\ b = \langle \mathbf{s}, \mathbf{a} \rangle + e \mod 1) \overset{c}{\approx} (\mathbf{a},\ \mathbf{u}), \qquad e \leftarrow \chi_\alpha,\ \mathbf{s} \leftarrow_\$ \{0,1\}^n,\ \mathbf{a}, \mathbf{u} \leftarrow_\$ \mathbb{T}^n.$$

TFHE: a framework for faster bootstrapping [CGGI16,CGGI17]

$\mathbb{T} := \mathbb{R}/\mathbb{Z}$

Basic assumption: learning with errors (LWE) over the torus

$$(\mathbf{a},\ b = \langle \mathbf{s}, \mathbf{a} \rangle + e \mod 1) \overset{c}{\approx} (\mathbf{a},\ \mathbf{u}), \qquad e \leftarrow \chi_\alpha,\ \mathbf{s} \leftarrow_\$ \{0,1\}^n,\ \mathbf{a}, \mathbf{u} \leftarrow_\$ \mathbb{T}^n.$$

Scheme	Message	Ciphertext
LWE	scalar	$(n+1)$ scalars
TLWE	polynomial	$(k+1)$ polynomials

TFHE: a framework for faster bootstrapping [CGGI16,CGGI17]

$\mathbb{T} := \mathbb{R}/\mathbb{Z}$

Basic assumption: learning with errors (LWE) over the torus

$$(\mathbf{a},\ b = \langle \mathbf{s}, \mathbf{a} \rangle + e \mod 1) \overset{c}{\approx} (\mathbf{a},\ \mathbf{u}), \qquad e \leftarrow \chi_\alpha,\ \mathbf{s} \leftarrow_\$ \{0,1\}^n,\ \mathbf{a}, \mathbf{u} \leftarrow_\$ \mathbb{T}^n.$$

Scheme	Message	Ciphertext
LWE	scalar	$(n+1)$ scalars
TLWE	polynomial	$(k+1)$ polynomials

Overview of the bootstrapping procedure:

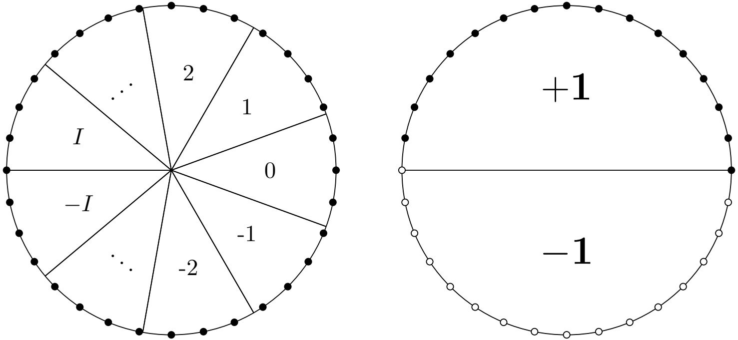
1. Hom. compute $X^{b-\langle \mathbf{s}, \mathbf{a} \rangle}$: spin the wheel
2. Pick the ciphertext pointed to by the arrow
3. Switch back to the original key

Our activation function

We focus on $f(x) = \text{sign}(x)$.

Our activation function

We focus on $f(x) = \text{sign}(x)$.



Refining TFHE

1. Reducing bandwidth usage
2. Dynamically changing the message space

Refining TFHE

1. Reducing bandwidth usage
2. Dynamically changing the message space

Standard packing technique: encrypt a polynomial instead of a scalar.

$$ct = \mathrm{TLWE.Encrypt}\left(\sum_i p_i X^i\right)$$

Refining TFHE

1. Reducing bandwidth usage
2. Dynamically changing the message space

Standard packing technique: encrypt a polynomial instead of a scalar.

$$ct = \text{TLWE.Encrypt}\left(\sum_i p_i X^i\right)$$

Same thing for weights (in the clear) **in the first hidden layer**: $w_{pol} := \sum_i w_i X^{-i}$.

1. Reducing bandwidth usage
2. Dynamically changing the message space

Standard packing technique: encrypt a polynomial instead of a scalar.

$$ct = \text{TLWE.Encrypt}\left(\sum_i p_i X^i\right)$$

Same thing for weights (in the clear) **in the first hidden layer**: $w_{pol} := \sum_i w_i X^{-i}$.

The constant term of $ct \cdot w_{pol}$ is then $\text{Enc}\left(\sum_i w_i x_i\right)$.

Refining TFHE

1. Reducing bandwidth usage
2. Dynamically changing the message space

Refining TFHE

1. Reducing bandwidth usage
2. Dynamically changing the message space

Fact We can keep the msg space constant (bound on all multisums).

Refining TFHE

1. Reducing bandwidth usage
2. Dynamically changing the message space

Fact We can keep the msg space constant (bound on all multisums).

Better idea Change the msg space to reduce errors. Intuition: less slices when we do not need them.

Refining TFHE

1. Reducing bandwidth usage
2. Dynamically changing the message space

Fact We can keep the msg space constant (bound on all multisums).

Better idea Change the msg space to reduce errors. Intuition: less slices when we do not need them.

How Details in the paper. Quick intuition: change what we put in the wheel.

Refining TFHE

1. Reducing bandwidth usage
2. Dynamically changing the message space

Fact We can keep the msg space constant (bound on all multisums).

Better idea Change the msg space to reduce errors. Intuition: less slices when we do not need them.

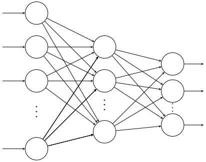
How Details in the paper. Quick intuition: change what we put in the wheel.

Bottom line

We can start with any message space at encryption time, and change it dynamically during the bootstrapping.

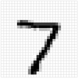
Overview of the process

Evaluation of a DiNN with 30 neurons in the hidden layer:



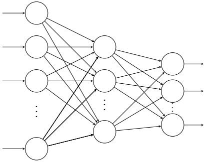
User

Server



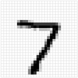
Overview of the process

Evaluation of a DiNN with 30 neurons in the hidden layer:



User

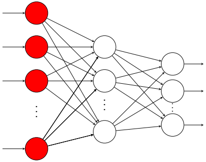
Server



$\xrightarrow{\text{Enc}\left(\sum_i p_i X^i\right)}$

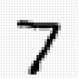
Overview of the process

Evaluation of a DiNN with 30 neurons in the hidden layer:



User

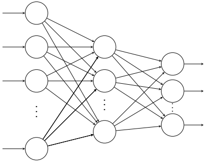
Server

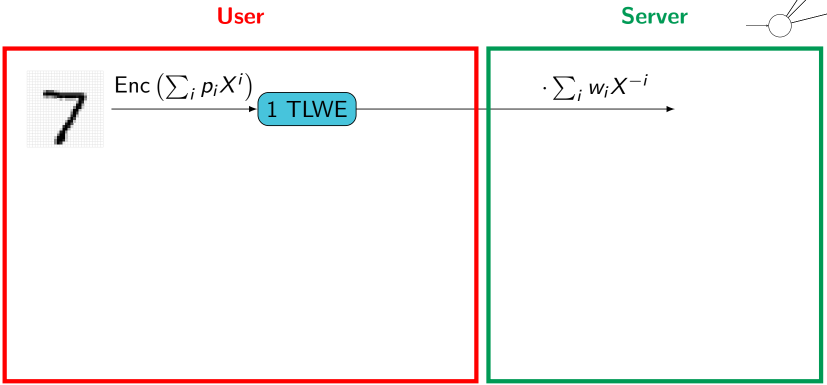


$\xrightarrow{\mathrm{Enc}\left(\sum_i p_i X^i\right)}$ 1 TLWE

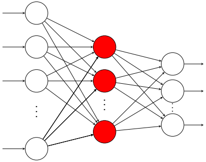
Overview of the process

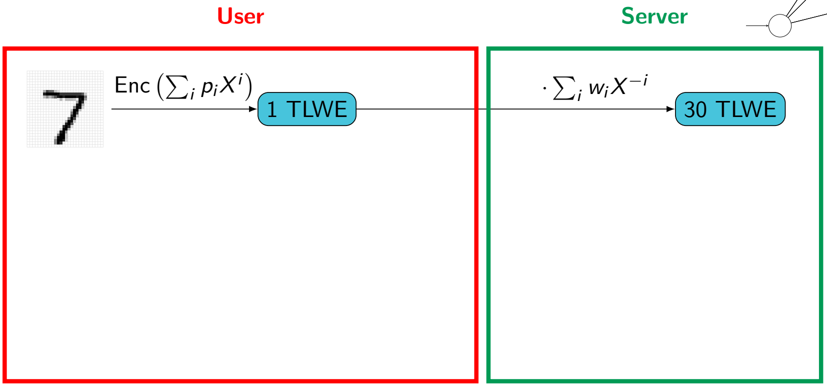
Evaluation of a DiNN with 30 neurons in the hidden layer:





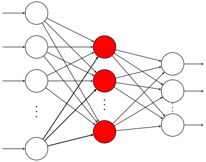
Evaluation of a DiNN with 30 neurons in the hidden layer:



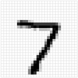


Overview of the process

Evaluation of a DiNN with 30 neurons in the hidden layer:



User



$\text{Enc}\left(\sum_i p_i X^i\right)$ → 1 TLWE

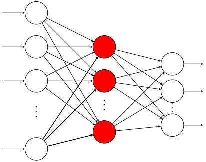
Server

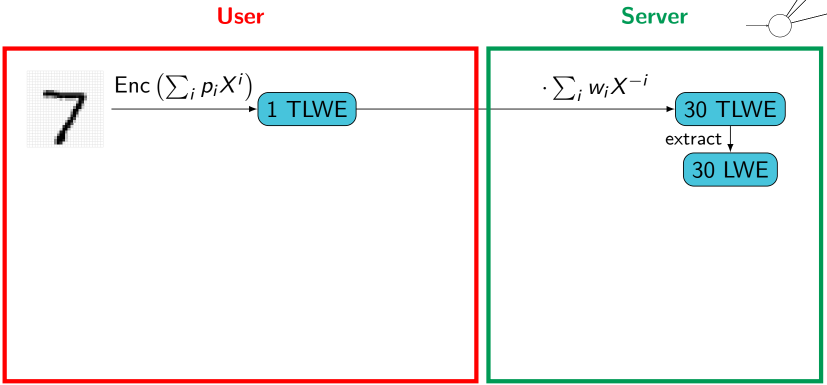
$\cdot \sum_i w_i X^{-i}$ → 30 TLWE

extract

Overview of the process

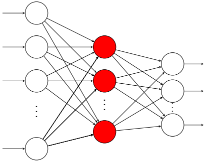
Evaluation of a DiNN with 30 neurons in the hidden layer:

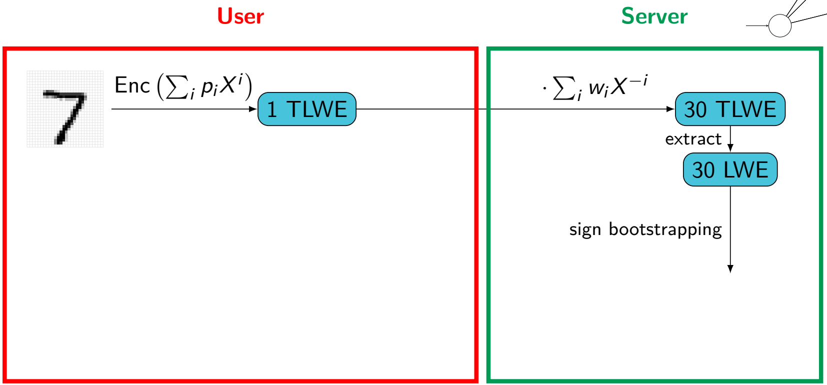




Overview of the process

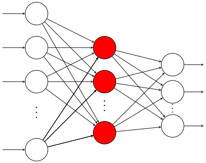
Evaluation of a DiNN with 30 neurons in the hidden layer:

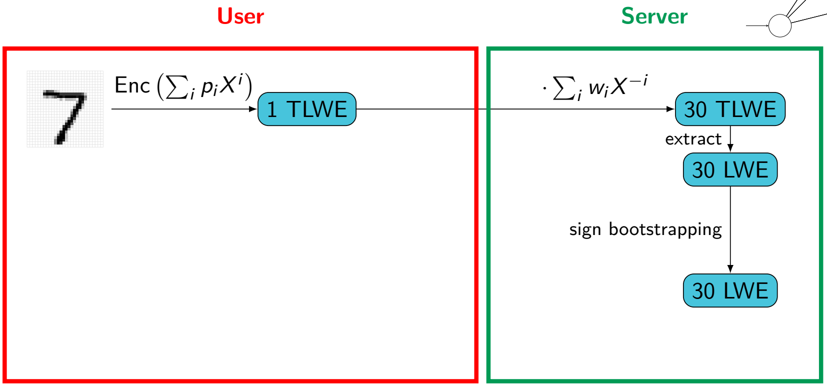




Overview of the process

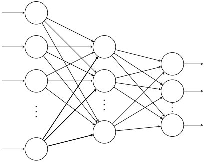
Evaluation of a DiNN with 30 neurons in the hidden layer:

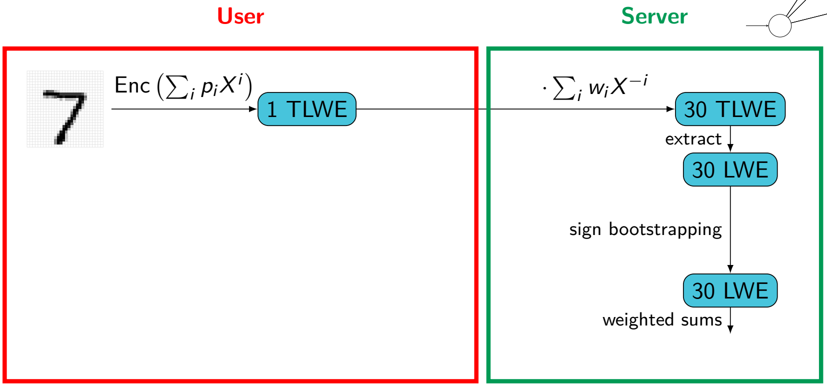




Overview of the process

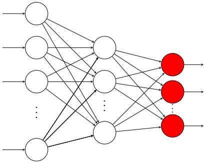
Evaluation of a DiNN with 30 neurons in the hidden layer:

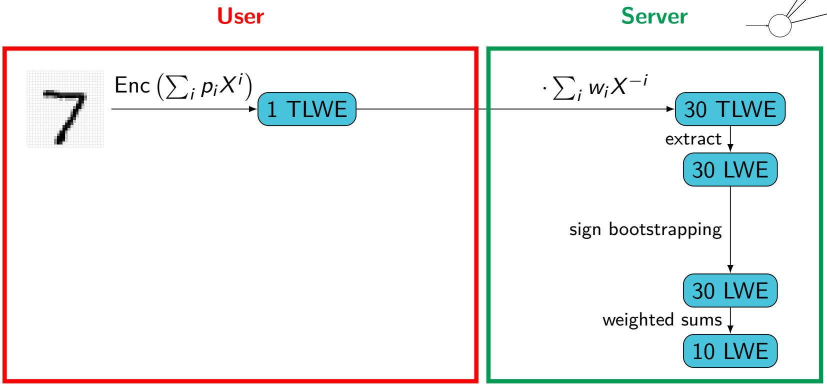




Overview of the process

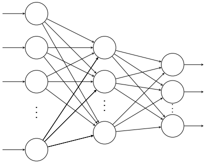
Evaluation of a DiNN with 30 neurons in the hidden layer:

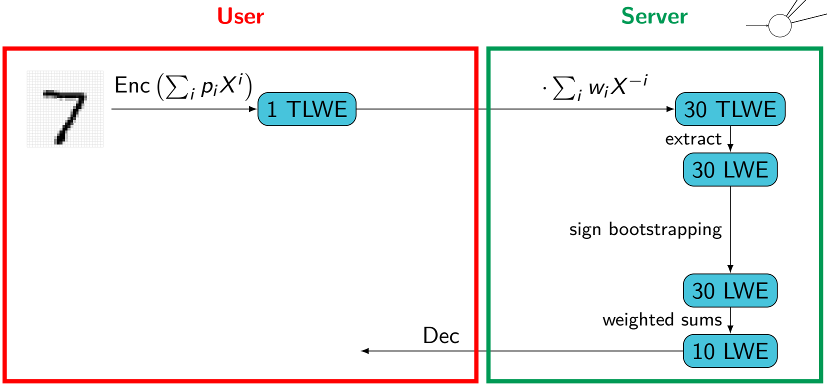




Overview of the process

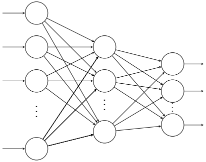
Evaluation of a DiNN with 30 neurons in the hidden layer:

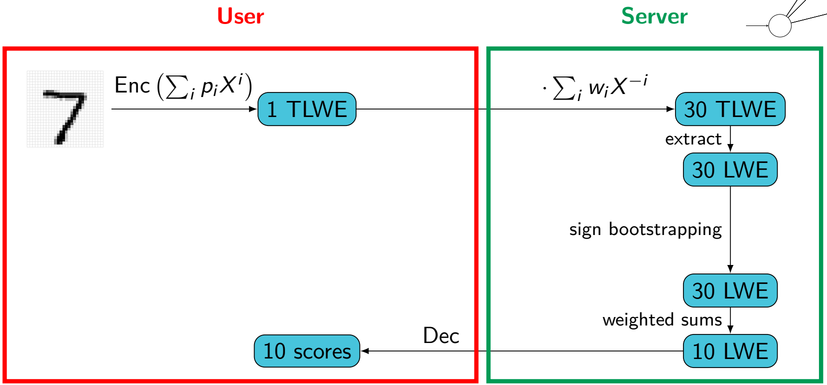




Overview of the process

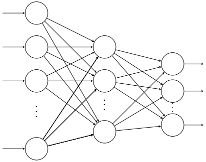
Evaluation of a DiNN with 30 neurons in the hidden layer:

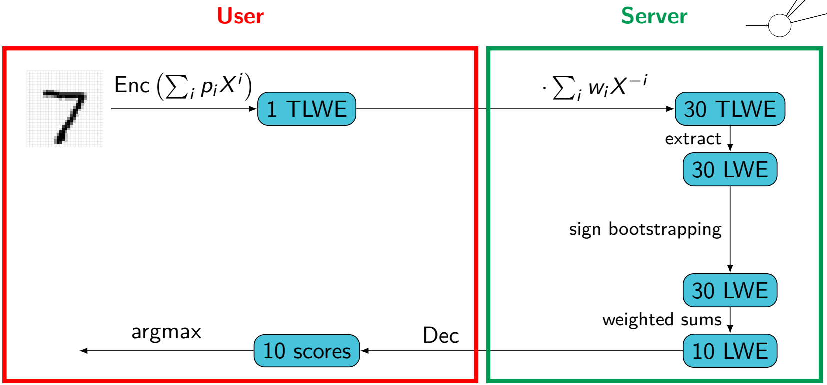




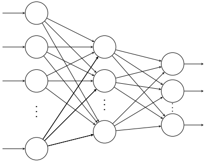
Overview of the process

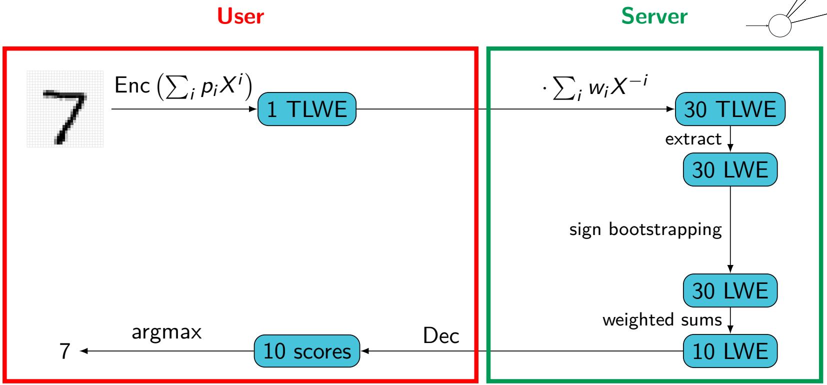
Evaluation of a DiNN with 30 neurons in the hidden layer:





Evaluation of a DiNN with 30 neurons in the hidden layer:





Experimental results

On inputs in the clear

	Original NN ($\mathbb{R}$)	DiNN + hard_sigmoid	DiNN + sign
30 neurons	94.76%	93.76% (-1%)	93.55% (-1.21%)
100 neurons	96.75%	96.62% (-0.13%)	96.43% (-0.32%)

On encrypted inputs

	Accur.	Disag.	Wrong BS	Disag. (wrong BS)	Time
30 or	93.71%	273 (105–121)	3383/300000	196/273	0.515 s
30 un	93.46%	270 (119–110)	2912/300000	164/270	0.491 s
100 or	96.26%	127 (61–44)	9088/1000000	105/127	1.679 s
100 un	96.35%	150 (66–58)	7452/1000000	99/150	1.64 s

or = original un = unfolded

Experimental results

On inputs in the clear

	Original NN ($\mathbb{R}$)	**DiNN + hard_sigmoid**	**DiNN + sign**
30 neurons	94.76%	93.76% (-1%)	93.55% (-1.21%)
100 neurons	96.75%	96.62% (-0.13%)	**96.43%** (-0.32%)

On encrypted inputs

	Accur.	**Disag.**	**Wrong BS**	**Disag. (wrong BS)**	**Time**
30 or	93.71%	273 (105–121)	3383/300000	196/273	0.515 s
30 un	93.46%	270 (119–110)	2912/300000	164/270	0.491 s
100 or	**96.26%**	127 (61–44)	9088/1000000	105/127	1.679 s
100 un	**96.35%**	150 (66–58)	7452/1000000	99/150	1.64 s

or = original un = unfolded

Benchmarks

	Neurons	Size of ct.	Accuracy	Time enc	Time eval	Time dec
FHE-DiNN 30	30	8.0 kB	93.71%	0.000168 s	0.49 s	0.0000106 s
FHE-DiNN 100	100	8.0 kB	96.35%	0.000168 s	1.65 s	0.0000106 s

Benchmarks

	Neurons	Size of ct.	Accuracy	Time enc	Time eval	Time dec
FHE-DiNN 30	30	8.0 kB	93.71%	0.000168 s	0.49 s	0.0000106 s
FHE-DiNN 100	100	8.0 kB	96.35%	0.000168 s	1.65 s	0.0000106 s

independent of the network

Benchmarks

	Neurons	Size of ct.	Accuracy	Time enc	Time eval	Time dec
FHE-DiNN 30	30	8.0 kB	93.71%	0.000168 s	0.49 s	0.0000106 s
FHE-DiNN 100	100	8.0 kB	96.35%	0.000168 s	1.65 s	0.0000106 s

scales linearly

Open problems and future directions

- Build better DiNNs: more attention to the conversion (+ retraining)

Open problems and future directions

- Build better DiNNs: more attention to the conversion (+ retraining)
- Implement on GPU to have realistic timings

Open problems and future directions

- Build better DiNNs: more attention to the conversion (+ retraining)
- Implement on GPU to have realistic timings
- More models (e.g., convolutional NNs) and machine learning problems

Open problems and future directions

- Build better DiNNs: more attention to the conversion ($+$ retraining)
- Implement on GPU to have realistic timings
- More models (e.g., convolutional NNs) and machine learning problems

Research needed

We need a fast way to evaluate other, more complex, functions (e.g., max or ReLU[a]).

[a] $\mathsf{ReLU}\left(x\right) = \max\left(0, x\right)$

Open problems and future directions

- Build better DiNNs: more attention to the conversion (+ retraining)
- Implement on GPU to have realistic timings
- More models (e.g., convolutional NNs) and machine learning problems

Research needed

We need a fast way to evaluate other, more complex, functions (e.g., max or ReLU[a]).

[a] $\text{ReLU}(x) = \max(0, x)$

Thank you for your attention!

Questions?