DELPHI: Cryptographic Inference for Neural Networks

Pratyush Mishra  Ryan Lehmkuhl  Akshayaram Srinivasan  Wenting Zheng  Raluca Ada Popa

UC Berkeley
Neural Network Inference

A growing number of applications use neural networks in user interactions

- Home monitoring: detect and recognize visitors
- Baby monitor: motion detection to alert parents

User data is sensitive
Server’s model is proprietary
Client-side inference

Client sees server’s model!

This reveals model weights and leaks information about private training data

\[ \text{[SRS17], [CLEKS18], [MSCS18]} \]

Server-side inference

Server sees client data!
Secure inference goals

Client ($x$) and server ($M$) should learn only prediction $M(x)$.

Server should not learn private client input $x$.

Client should not learn private model weights $M$. 
Prior work on secure inference

<table>
<thead>
<tr>
<th>Protocol type</th>
<th>FHE based</th>
<th>2PC based</th>
<th>Desired</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CryptoNets, CHET, TAPAS</td>
<td>SecureML, Gazelle, MiniONN</td>
<td>Delphi</td>
</tr>
<tr>
<td>Examples</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Functionality/ Accuracy</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Delphi

Cryptographic system for secure inference on convolutional neural networks

Security: achieves semi-honest simulation-based security

Functionality: supports arbitrary CNNs

Efficiency:
- improves bandwidth (9x) and inference latency (22x)
- can utilize GPU/TPU for linear layers
- evaluated on realistic workloads (CIFAR-100, ResNet-32)
Recap: Convolutional Neural Networks

Linear Layers

Non-linear Layers

Convolution

Activation (ReLU)

Convolution

Activation (ReLU)

Fully-connected

Prediction

Input

$\mathbf{f(x)}$

$\mathbf{x}$
Starting point: G

Key insight: use crypto specialized for each layer.

Client

Server

1. Linear layer

\[ c \leftarrow \text{Enc}(Lx + s) \]

Garbled circuits: 2PC protocol for bitwise operations like ReLU

2. Activation

\[ y \leftarrow \text{Dec}(c) \]

\[ y = \text{ReLU}(Lx) \]

\[ y - s \]

\[ Lx \]

\[ \text{ReLU} \]

\[ \text{Enc}(\text{ReLU}(Lx)) \]

Linearly-homomorphic Encryption

\[ \text{Enc}(x) + \text{Enc}(y) = \text{Enc}(x + y) \]
Expensive parts of Gazelle

For ResNet-32, per inference: ~600MB communication, and ~82 sec latency.
Delphi: Optimizing Linear layers

Preprocessing phase

Client
Sample \( r \)
\( y \leftarrow \text{Dec}(c) \)

Server
Sample \( s \)
\( c \leftarrow \text{Enc}(Lr + s) \)

Online phase

Get input \( x \)

\( x + r \)

\( y \leftarrow \text{Dec}(c) \)

\( z := L(x + r) + s \)
\( = Lx + y \)

\( \text{ReLU}(z - y) \)

\( r_2 \)

\( \text{ReLU}(Lx) + r_2 \)
\( = x_2 + r_2 \)

Per inference:
\( >600\text{MB} \sim 350\text{MB} \)
\( \sim 82\text{ s} \sim 13\text{ s} \)

latency

GPU compatible!
Delphi: Optimizing Non-linear Activations

**Problem:** ReLU is cheap for CPUs, but **costly** in 2PC.

**Solution Idea:** Replace ReLUs with quadratic activations, which *are* cheap in 2PC
[ CryptoNets, SecureML ]

**Problem:** Training accurate quad. networks is difficult: algorithms are optimized for all-ReLU networks
Delphi’s Machine Learning Planner

Contains a mixture of ReLU and quadratic activations, and has accuracy > t

Better techniques for training hybrid networks
- Clipping gradients
- Blending in quadratic layers slowly

Specializing Neural Architecture Search to discover hybrid networks
- Adapt PBT algorithm
- Iterative exploration of search space
Delphi’s end-to-end workflow

- **Client**
  - Train initial all-ReLU network
  - Optimize accuracy and efficiency
  - Preprocessing for linear, ReLU, and quadratic layers
  - Online phase for linear, ReLU, and quadratic layers

- **Server**
  - Train all-ReLU CNN
  - Planner
  - Hybrid CNN

- **Input** $x$ connects to **Client Online** and **Server Online**
- **Output** $M(x)$
Implementation

Rust + C++ library with support for GPU acceleration

github.com/mc2-project/delphi

Evaluation

1. Does Delphi’s planner preserve accuracy?
2. Does Delphi’s protocol reduce latency & bandwidth?

Benchmark: ResNet-32 network on CIFAR-100
Planner accuracy

ReLUs are not redundant: accuracy loss > 10%
Most efficient planned network achieves loss of < 2%
Latency and communication

Comparison with Gazelle

- Inference time (s)
  - Delphi
  - Gazelle

- Data transferred (GB)
  - Delphi
  - Gazelle

Comparison:
- Inference time: > 20x
- Data transferred: ~ 9x
Delphi

- Secure inference on convolutional neural networks
- 9-22x more efficient than prior work
- Combines techniques from systems, cryptography, and ML

[ia.cr/2020/050](https://ia.cr/2020/050)

[github.com/mc2-project/delphi](https://github.com/mc2-project/delphi)