Privformer: Privacy-preserving Transformer with MPC

Yoshimasa Akimoto U. Tsukuba
Kazuto Fukuchi U. Tsukuba
Yohei Akimoto U. Tsukuba
Jun Sakuma Tokyo Tech./RIKEN AIP
Transformer

- Generic deep learning model for sequence processing [1].
- Used for a variety of applications including machine translation, question answering, and automatic summarization
- Foundational architecture used in modern language generation models such as ChatGPT, BERT, and LLaMA

Privacy concern

- Privacy protection is an issue when using AI services via API
  - E.g., I want to use ChatGPT to solve a question related to my business.
- If prompts contain personal or confidential information, users cannot send them to the server.
- Also, the model is so huge that it is not practical to run it locally.
Privacy Protection

Secure multi-party computation (MPC)

Private information is distributed among multiple parties in the form of secret shares, and the computation is performed securely among multiple parties over the shares.

\[
a = x \oplus y \oplus z
\]

\[
f(a) = u \oplus v \oplus w
\]
Our goal

**Application of MPC to Transformer**

- Evaluation of the Transformer contains numerous number of exponents and inverses, which can be a bottleneck
- Less studied compared to CNN and RNN

**Goal**

- To propose an MPC-friendly Transformer architecture for the Transformer model
- To implement Transformer inference on MPC
Related work

SecureNN[2], Falcon[3]
- DNN and CNN training and inference on MPC
- Supports convolution, ReLU, Maxpool, and Batch Normalization layers

SIRNN[4], SecureNLP[5]
- Enabling RNN inference on MPC
- Can compute, \( \sigma(x) = \frac{1}{1 + e^{-ax}} \) \( \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \)

Related work

**Wang et al. [6]**

- Enabling Transformer inference on MPC
- Supports the embedding process, but does not support softmax in the attention layer
- Proposed a method to speed up the Embedding process
- No improvement proposed for Attention processing

---

Contribution of this study

- Introducing Attention using ReLU, instead of Softmax to reduce the computation and communication cost of the Attention layer on MPC.
- Implemented Transformer inference on FALCOM framework and measured execution time and communication cost by experiments.
Problem setting

**Parties**

- **Model owner**: hold pmodel parameters $W$ for the Transformer
- **Query user**: hold input $X$ used for inference
- **Computing servers**: three servers that perform MPC to compute $Y = \text{Transformer}(W, X)$

- Obtain shares of $W, X$ and compute $Y$
- Never learns $W, X, Y$

- Share $W$ with comp. servers
- Never learns $X, Y$

- Share $X$ with comp servers
- Never learns $W$ and learn $Y$
MPC on FALCON

Falcon framework was used for MPC implementation

- Three party MPC
- Honest-majority
- Support efficient computation of functions frequently used for deep learning

Share representation

- Fixed-point rep.: \( x \in \mathbb{R} \rightarrow [x \cdot 2^{FP}] (\text{mod} \ L) = \hat{x} \in \mathbb{Z}_L \), FP: Precision param.
- Share rep.: share of \( \hat{x} \) \( \langle \hat{x} \rangle = (\hat{x}_1, \hat{x}_2, \hat{x}_3) \) where \( \hat{x} \equiv \hat{x}_1 + \hat{x}_2 + \hat{x}_3 \text{ mod } L \)

Arithmetic operations

- Addition: \( \langle x \rangle + \langle y \rangle = \langle x + y \rangle \)
- Constant magnification: \( c \langle x \rangle = \langle cx \rangle \)
- Multiplication: \( \Pi_{\text{Mult}}(\langle x \rangle, \langle y \rangle) \rightarrow \langle xy \rangle \)

Matrix multiplication: \( \Pi_{\text{MatMul}}(\langle X \rangle, \langle Y \rangle) \rightarrow \langle XY \rangle \)

- \( X \in \mathbb{R}^{l \times m}, Y \in \mathbb{R}^{m \times n} \) Time \( O(lmn) \), Communication \( O(ln) \)
Nonlinear operations

- Computed by combination of addition, constant multiplication, and multiplication
- Relatively costly since containing iterative communication
- Results are approximation when iterative methods are used

<table>
<thead>
<tr>
<th>演算</th>
<th>Time[ms]</th>
<th>Comm[B]</th>
<th>Rounds</th>
<th>Appro x.?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Pi_{\text{Mult}}(\langle x, y \rangle) \to \langle xy \rangle$</td>
<td>1.17</td>
<td>0.02</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>$\Pi_{\text{exp}}(\langle x \rangle) \to \langle e^x \rangle$</td>
<td>9.78</td>
<td>0.13</td>
<td>32</td>
<td>✓</td>
</tr>
<tr>
<td>$\Pi_{\text{Inv}}(\langle x \rangle) \to \langle 1/x \rangle$</td>
<td>43.84</td>
<td>0.85</td>
<td>215</td>
<td>✓</td>
</tr>
<tr>
<td>$\Pi_{\text{InvSqrt}}(\langle x \rangle) \to \langle 1/\sqrt{x} \rangle$</td>
<td>57.47</td>
<td>1.03</td>
<td>260</td>
<td>✓</td>
</tr>
<tr>
<td>$\Pi_{\text{ReLU}}(\langle x \rangle) \to \langle \text{ReLU}(x) \rangle$</td>
<td>6.21</td>
<td>0.13</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>

How can we realize Transformer on MPC using them?
How to realize MPC for Transformer

- Encoder transforms input token sequence $X$ into latent representation
- Decoder transforms $Z$ to $X$
- Embedding, Positional Encoding
- Linear & Softmax layer
- Multi-Head Attention
- Masked Multi-Head Attention
- Feed Forward Network
- Layer Normalization

Pre/post processing
Local computation works

Need to be processed with MPC
Complexity analysis of MPC for Multi-Head Attention

When employing existing primitives naively, which part can be the bottleneck?

- For input $Q, K, V \in \mathbb{R}^{S \times d}$ where $S$ is input sequence length and $d$ is the dimension of embedding vector:

  - Computed with addition, constant mult., $\Pi_{\text{MatMul}}, \Pi_{\text{Exp}}, \Pi_{\text{Div}}$

  $$\text{Attention}(Q, K, V) = \text{Softmax} \left( \frac{Q K^T}{\sqrt{d}} \right) V \text{ where } \text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{k=1}^{d_m} e^{x_k}}$$

  - $\Pi_{\text{Inv}}$: time/comm. complexity $O(S)$
  - $\Pi_{\text{Exp}}$: time/comm. complexity $O(S^2)$
  - $\Pi_{\text{MatMul}}$ for $QK^T$: time/comm. complexity $O(S^2)$
MPC-friendly attention: ReLU Attention

ReLU Attention[7]

- Relacing Softmax with projection with random matrix $\Omega \in \mathbb{R}^{d \times r}$ and ReLU
- By computing $K'^T V$ first, avoid dealing with $S \times S$ matrix, achieving $O(rs)$ complexity
- With sufficiently large $r$, the predictive performance is comparable to Softmax Attention

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

$O(S^2)$ is required for $QK^T$

$$\text{ReLU Attention}(Q, K, V, \Omega) = cQ'(K'^T V)$$

where $Q' = \text{ReLU}(Q\Omega)$ and $K' = \text{ReLU}(K\Omega)$

$K'^T V$ can be computed in $O(rs)$

$Q, K, V \in \mathbb{R}^{S \times d}, \Omega \in \mathbb{R}^{d \times r}, Q', K' \in \mathbb{R}^{S \times r}, r$: dimension after projection

Secure ReLU Attention

• ReLU Attention

ReLU Attention(Q, K, V, Ω) = cQ′(K′T V)
where Q′ = ReLU(QΩ) and K′ = ReLU(KΩ)

• MPC ReLU Attention

⟨ReLUAttention(Q, K, V, Ω)⟩ ← cΠMatMul(⟨Q′⟩, ΠMatMul(⟨(K′)T⟩, ⟨V⟩))

where ⟨Q′⟩ = ΠReLU(ΠMatMul(⟨Q⟩, ⟨Ω⟩)),
⟨K′⟩ = ΠReLU(ΠMatMul(⟨K⟩, ⟨Ω⟩))

Time/space complexity of ΠMatMul, ΠReLU is O(rS)
Round complexity is O(1)
Summary of complexity analysis and outline of experiments

### Summary of complexity analysis

<table>
<thead>
<tr>
<th></th>
<th>Computation</th>
<th>Rounds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\Pi_{\text{MatMul}}$</td>
<td>$\Pi_{\text{Exp}}$</td>
</tr>
<tr>
<td>Softmax Attention</td>
<td>$O(S^2)$</td>
<td>$O(S^2)$</td>
</tr>
<tr>
<td>ReLU Attention</td>
<td>$O(rS)$</td>
<td>-</td>
</tr>
<tr>
<td>Masked Softmax Attention</td>
<td>$O(S^2)$</td>
<td>$O(S^2)$</td>
</tr>
<tr>
<td>Masked ReLU Attention (Naive)</td>
<td>$O(rS)$</td>
<td>-</td>
</tr>
<tr>
<td>Masked ReLU Attention (QK first)</td>
<td>$O(S^2)$</td>
<td>-</td>
</tr>
</tbody>
</table>

### Experiments

- 1) Comparison of Softmax Attention and ReLU Attention in time and communication
- 2) End-to-end evaluation of time and communication for Transformer inference
Experimental environment

Computing servers (Amazon AWS)

- OS: Ubuntu 18.04 LTS, CPU: 2.9 GHz Intel Xeon E5-2666 v3 processor, RAM: 64GB

Network environment

- LAN: All comp servers were located in the same region, average bandwidth during the experiment was 4.93 Gbits/s, average ping response time was 1.17 ms

- WAN: All comp servers were located in Ohio, Tokyo, and London. Average bandwidth during the experiment was 97.4 Mbits/s, average ping response time was 141.67 ms
Experimental settings

Parameter settings
- Parameter follows FALNCON
- Ring on $\mathbb{Z}_L$ where $L = 2^{32}$.
- Fixed-point representation with 13-bit precision
- FALNCON supports honest majority, but for experimental evaluation, we assumed semi-honest adversaries
- Transformer parameters: $d_m = 512, d_{ff} = 2048, h = 8, d = 64, r = 266,$
  Encoder/Decoder layer $N = 6$, batch size : 1

Other settings
- Results are average of ten trials
- Exponentiation was approximated with 4\textsuperscript{th} order Chebyshev polynomial
- For Newton method to compute inverse, the iteration number was set as four
1) Comparison of Softmax Attention and ReLU Attention

Fig. 1. : S=32, ⋯, 1024. Comparison of Softmax Attention and ReLU Attention in comp time [sec], communication [MB], and rounds. $r=266$

- Softmax Attention: time and communication increase quadratically
- ReLU Attention: time and communication increase linearly
- For a sequence length of 1024,
  - 9.41x faster in LAN, 8.25x faster in WAN
  - communication is reduced to 1/12
2) End-to-end evaluation of Transformer inference

Table 2. Time [sec] and comm. [MB] of each layer, $S = 64$

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LAN</td>
<td>WAN</td>
<td></td>
</tr>
<tr>
<td>Multi-Head Attention</td>
<td>0.611</td>
<td>10.944</td>
<td>48.361</td>
</tr>
<tr>
<td>Masked Multi-Head Attention</td>
<td>0.613</td>
<td>12.424</td>
<td>47.231</td>
</tr>
<tr>
<td>Feed Forward Network</td>
<td>1.671</td>
<td>6.669</td>
<td>19.136</td>
</tr>
<tr>
<td>Layer Normalization</td>
<td>0.077</td>
<td>10.753</td>
<td>1.64</td>
</tr>
<tr>
<td>Encoder</td>
<td>14.141</td>
<td>253.164</td>
<td>424.668</td>
</tr>
<tr>
<td>Decoder</td>
<td>17.768</td>
<td>402.85</td>
<td>717.894</td>
</tr>
</tbody>
</table>

- Encoder is executed once and Decoder is executed $S$ times.
- When $s=64$, 19 minutes in LAN, 7.23 hours in WAN.
Conclusion

● Introduced ReLU Attention as MPC-friendly attention
● Proposed an algorithm for Masked ReLU Attention for MPC
● Experimental comparison of Softmax Attention and ReLU Attention shows reduction of computation and communication cost
● Showed that a 64-word sentence can be inferred in about 20 minutes in a LAN environment