High-precision RNS-CKKS on small word-size architecture

Duhyeong Kim, Intel Labs

FHE.org Meetup

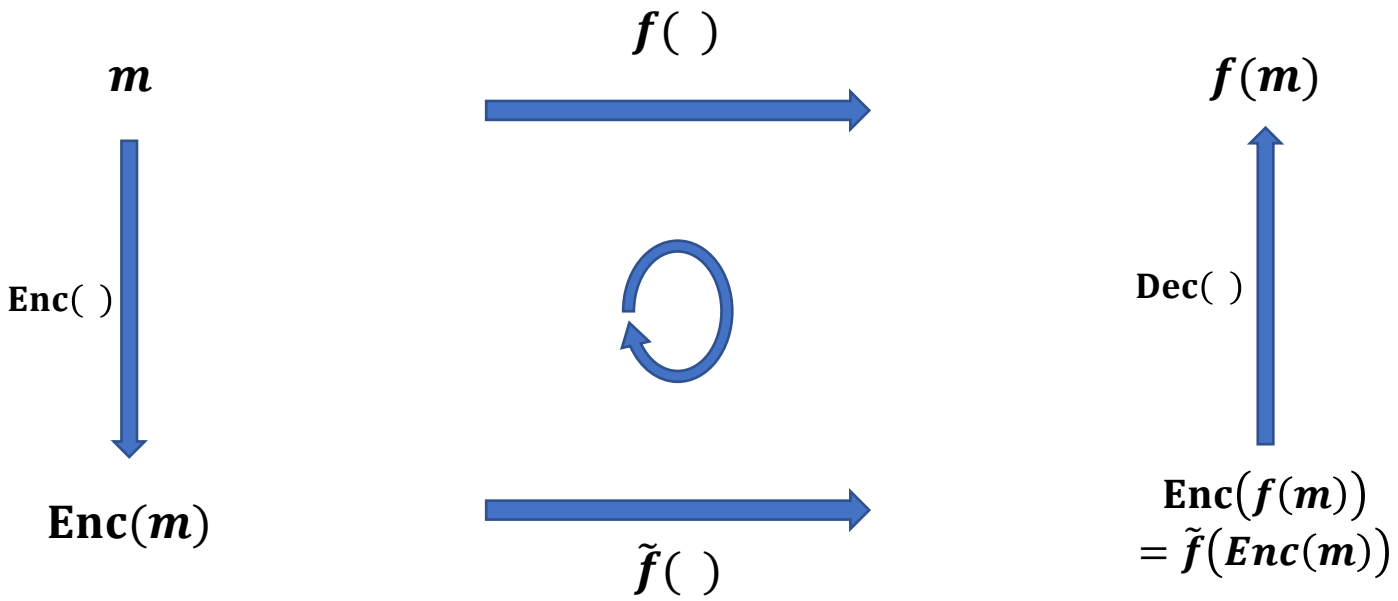
Jan 11th, 2024

Overview

- Enable high-precision RNS-CKKS on fixed but smaller word-size architectures
  - Single scaling → **Composite scaling**
- Enable functionally correct CKKS composite scaling in two open-source libraries
  - OpenFHE: C++, enabled by Intel labs
  - Lattigo: Go, enabled by Seoul National University (SNU)
- Demonstrate with secure parameters the **equivalence** between single and composite scaling
  - **7-layer CNN Inference** with longitudinal packing in OpenFHE-CKKS with composite scaling
  - **7-layer CNN Inference** with multiplexed packing in Lattigo-CKKS with composite scaling
  - **Logistic Regression Training** in OpenFHE-CKKS with composite scaling

Fully Homomorphic Encryption (FHE)

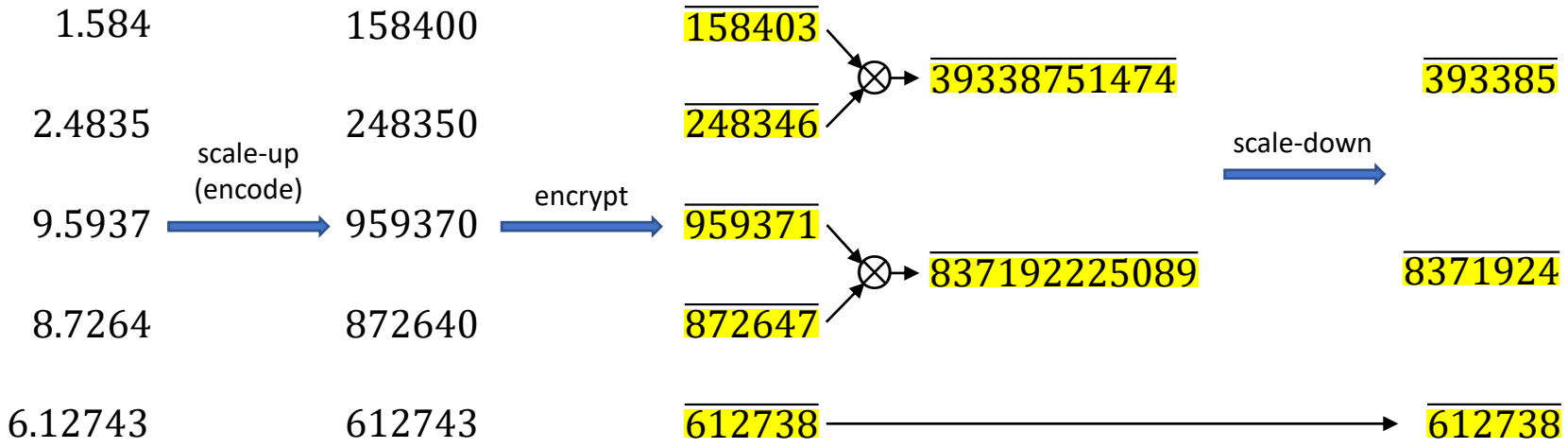
- Any computation on encrypted data “without decryption process”



CKKS: FHE for real-number arithmetic

How can we think of the “approximate” computation in CKKS?

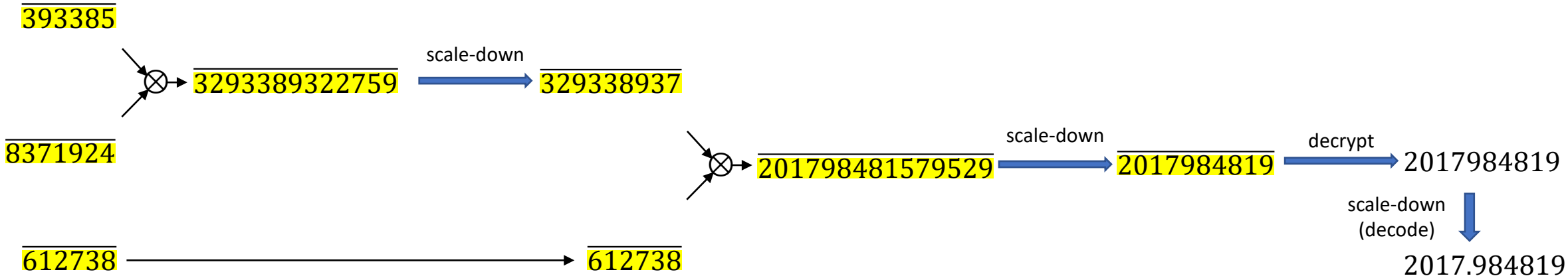
- **Imitation of “fixed-point” arithmetic in cleartext version**
- Example: computation of $1.584 \times 2.4835 \times 9.5937 \times 8.7264 \times 6.12743 \ (\approx 2017.9897)$



CKKS: FHE for real-number arithmetic

How can we think of the “approximate” computation in CKKS?

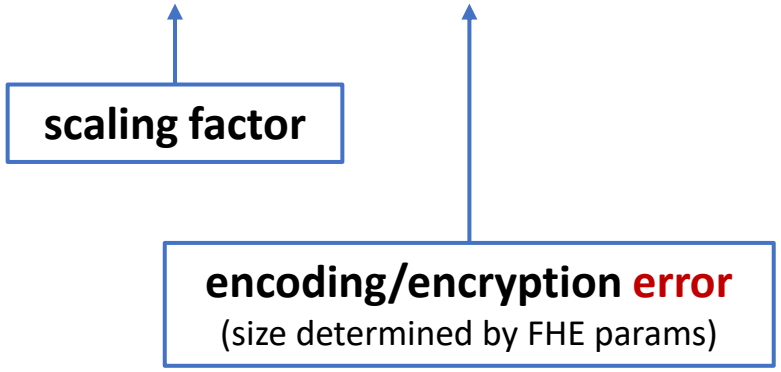
- **Imitation of “fixed-point” arithmetic in cleartext version**
- Example: computation of $1.584 \times 2.4835 \times 9.5937 \times 8.7264 \times 6.12743 \ (\approx 2017.9897)$



Scaling Factor in CKKS

- Determine the **"initial precision bits"** under the decimal point
- CKKS Encoding/Encryption results in

$$m \mapsto \boldsymbol{\Delta} \cdot m + e$$



- Larger $\boldsymbol{\Delta}$, start with higher precision
- Smaller $\boldsymbol{\Delta}$, start with lower precision

Scaling Factor in CKKS

- Exponential growth of Scaling Factor
  - $(\boldsymbol{\Delta} \cdot m) \cdot (\boldsymbol{\Delta} \cdot m') = \boldsymbol{\Delta^2} \cdot mm'$
  - $\left(\boldsymbol{\Delta^{2^k}} \cdot m\right) \cdot \left(\boldsymbol{\Delta^{2^k}} \cdot m'\right) = \boldsymbol{\Delta^{2^{k+1}}} \cdot mm'$
- How to control the growth of scaling factor?

"rescale"

- Rescale(ct): $ct \bmod \Delta^{\ell} \mapsto \left\lfloor \frac{ct}{\Delta} \right\rceil \bmod \Delta^{\ell-1}$ (from the context of "original" CKKS)
  - $(\boldsymbol{\Delta} \cdot m) \cdot (\boldsymbol{\Delta} \cdot m') = \boldsymbol{\Delta^2} \cdot mm' \xrightarrow{\text{Rescale } (1/\Delta)} \boldsymbol{\Delta} \cdot mm'$

Rescale in RNS-CKKS

- RNS-CKKS
  - An efficient way to implement CKKS w/o big-number arithmetic
  - Ctxt moduli $Q_\ell = q_0 q_1 \cdots q_\ell$ for level ℓ (instead of modulo Δ^ℓ)

$$\mathbf{RNS}_{\boldsymbol{Q_\ell}}(\boldsymbol{x}) \coloneqq (\boldsymbol{x} \bmod \boldsymbol{q_0}, \boldsymbol{x} \bmod \boldsymbol{q_1}, \dots, \boldsymbol{x} \bmod \boldsymbol{q_\ell})$$

- Rescale modulo Q_ℓ in RNS?
  - **No efficient** way to compute $\mathbf{x} \mapsto \left\lfloor \frac{\mathbf{1}}{\boldsymbol{\Delta}} \cdot \boldsymbol{x} \right\rceil$
  - Instead, we can **efficiently** compute $\mathbf{x} \mapsto \left\lfloor \frac{\mathbf{1}}{\boldsymbol{q_\ell}} \cdot \boldsymbol{x} \right\rceil$
    - $\left\lfloor \frac{1}{q_\ell} \cdot x \right\rceil = q_\ell^{-1} \cdot (x - x \bmod q_\ell)$
    - Easy to obtain the RNS representation of $x \bmod q_\ell$
      - $\mathrm{RNS}_{Q_{\ell-1}}(x \bmod q_\ell) = (x \bmod q_\ell, x \bmod q_\ell, \dots, x \bmod q_\ell)$

Rescale in RNS-CKKS

- **Case 1: $\log \Delta$ < word-size**
  - Set each prime q_ℓ to be $\log \Delta$ bits
  - Perform the **"single scaling"**

$$\mathbf{x} \bmod Q_\ell \mapsto \left\lfloor \frac{\mathbf{1}}{\boldsymbol{q}_\ell} \cdot \boldsymbol{x} \right\rceil \bmod Q_{\ell-1}$$

- **Case 2: $\log \Delta$ > word-size**
  - Set each product of q_ℓ's to be $\log \Delta$ bits
  - Perform the **"composite scaling"**

Rescale in RNS-CKKS

- **Case 1: $\log \Delta <$ word-size**
  - Set each prime q_ℓ to be $\log \Delta$ bits
  - Perform the **"single scaling"**

$$\mathbf{x} \bmod Q_\ell \mapsto \left\lfloor \frac{\mathbf{1}}{\boldsymbol{q}_\ell} \cdot \boldsymbol{x} \right\rceil \bmod Q_{\ell-1}$$

- **Case 2: $\log \Delta >$ word-size**
  - Set each product of q_ℓ's to be $\log \Delta$ bits
  - Perform the **"composite scaling"** (degree = 2)

$$\mathbf{x} \bmod Q_\ell \mapsto \left\lfloor \frac{\mathbf{1}}{\boldsymbol{q}_\ell \boldsymbol{q}_{\ell-1}} \cdot \boldsymbol{x} \right\rceil \bmod Q_{\ell-2}$$

Rescale in RNS-CKKS

- **Case 1: $\log \Delta <$ word-size**
  - Set each prime q_ℓ to be $\log \Delta$ bits
  - Perform the **"single scaling"**

$$\mathbf{x} \bmod Q_\ell \mapsto \left\lfloor \frac{\mathbf{1}}{\boldsymbol{q}_\ell} \cdot \boldsymbol{x} \right\rceil \bmod Q_{\ell-1}$$

- **Case 2: $\log \Delta >$ word-size**
  - Set each product of q_ℓ's to be $\log \Delta$ bits
  - Perform the **"composite scaling"** (degree = 3)

$$\mathbf{x} \bmod Q_\ell \mapsto \left\lfloor \frac{\mathbf{1}}{\boldsymbol{q}_\ell \boldsymbol{q}_{\ell-1} \boldsymbol{q}_{\ell-2}} \cdot \boldsymbol{x} \right\rceil \bmod Q_{\ell-3}$$

Rescale in RNS-CKKS

- **Case 1: $\log \Delta <$ word-size**
  - Set each prime q_ℓ to be $\log \Delta$ bits
  - Perform the **"single scaling"**

$$\mathbf{x} \bmod Q_\ell \mapsto \left\lfloor \frac{\mathbf{1}}{\boldsymbol{q}_\ell} \cdot \boldsymbol{x} \right\rceil \bmod Q_{\ell-1}$$

- **Case 2: $\log \Delta >$ word-size**
  - Set each product of q_ℓ's to be $\log \Delta$ bits
  - Perform the **"composite scaling"** (degree = t)

$$\mathbf{x} \bmod Q_\ell \mapsto \left\lfloor \frac{\mathbf{1}}{\boldsymbol{q}_\ell \cdots \boldsymbol{q}_{\ell-t+1}} \cdot \boldsymbol{x} \right\rceil \bmod Q_{\ell-t}$$

Rescale in RNS-CKKS

- **Examples**
  - $\log \Delta = \mathbf{30}$, word-size $= \mathbf{64}$: single scaling
  - $\log \Delta = \mathbf{30}$, word-size $= \mathbf{32}$: single scaling
  - $\log \Delta = \mathbf{50}$, word-size $= \mathbf{64}$: single scaling
  - $\log \Delta = \mathbf{50}$, word-size $= \mathbf{32}$: composite scaling (double-prime)
  - $\log \Delta = \mathbf{70}$, word-size $= \mathbf{64}$: composite scaling (double-prime)
  - $\log \Delta = \mathbf{70}$, word-size $= \mathbf{32}$: composite scaling (triple-prime)

$\vdots$ $\vdots$ $\vdots$

Precision Issue due to Rescale

- Original CKKS: **NO** precision issue
  - ➢ Scaling factor is **ALWAYS** preserved as $\boldsymbol{\Delta}$
- RNS-CKKS: **YES** precision issue
  - ➢ Scaling factor is **NOT** be preserved as $\boldsymbol{\Delta}$
    - ○ Division by q_ℓ's, instead of Δ
    - ○ $\boldsymbol{\Delta^2/q_\ell \neq \Delta}$
  - ➢ Critical Impact to **Homomorphic Addition**
    - ○ $\mathrm{Enc}(\Delta \cdot m) + \mathrm{Enc}(\Delta' \cdot m') = \mathrm{Enc}(\Delta \cdot (m + \boldsymbol{\Delta'/\Delta} \cdot m')) \neq \mathrm{Enc}(\Delta \cdot (m + m'))$
    - ○ The ratio $\boldsymbol{\Delta'/\Delta}$ $(\neq 1)$ directly harms the precision

Precision Issue due to Rescale

- **Solution 1: Choose the primes properly**
  - To keep the scaling factors (not equal but) **very close to Δ**
  - Single Scaling
    - Requirement: $\boldsymbol{q_\ell \simeq \Delta}$ (proposed in original RNS-CKKS)
    - $\boldsymbol{\Delta^2/q_\ell \simeq \Delta}$
  - Composite Scaling
    - Requirement: $\boldsymbol{q_\ell q_{\ell-1} \simeq \Delta}$
    - $\boldsymbol{\Delta^2/q_\ell q_{\ell-1} \simeq \Delta}$
  - **Precision** (Single Scaling v.s. Composite Scaling)
    - **NO Difference** in Mult + Relin + Rescale
    - **Closeness** of q_ℓ (resp. $q_\ell q_{\ell-1}$) and Δ affects the **Add Precision**

Precision Issue due to Rescale

- **Solution 2: Exact Scaling**
  - Differences v.s. Solution 1
    - Scaling factor $\boldsymbol{\Delta_i}$ for each level $\boldsymbol{i}$
    - $\boldsymbol{\Delta_i}$'s are **NOT** required to be **very close to $\boldsymbol{\Delta}$**
    - Adjust the ciphertext scaling factors to $\boldsymbol{\Delta_i}$ before Add and Mult
    - As a result, we "always" add two ciphertexts with "same" scaling factors
  - **Precision** (Single Scaling v.s. Composite Scaling)
    - **NO Difference** in Mult + Relin + Rescale
    - **NO Difference** in Add
  - We implemented 32-bit RNS-CKKS in OpenFHE and Lattigo with **Solution 2**
    - "FLEXIBLEAUTO" mode in OpenFHE
    - Bootstrapping enabled in both libraries

Theoretical Analysis on Precision

$\mathrm{Rescale}(ct)$: $ct \bmod Q_i \mapsto \left\lfloor \frac{1}{q_i} \cdot ct \right\rceil \bmod Q_{i-1}$ (single scaling)

$\mathrm{Rescale}^{(t)}(ct)$: $ct \bmod Q_i \mapsto \left\lfloor \frac{1}{q_i q_{i-1} \cdots q_{i-t+1}} \cdot ct \right\rceil \bmod Q_{i-t}$ (composite scaling)

Theorem. *Let $B_{rs}, B_{comp-rs}$ be the upper bounds of the error induced by $\mathrm{Rescale}(\cdot)$ and $\mathrm{Rescale}^{(t)}(\cdot)$, respectively. Then, it holds that*

$$B_{comp-rs} \leq \left(\frac{1}{q_i q_{i-1} \cdots q_{i-t+1}} + \frac{1}{q_i q_{i-1} \cdots q_{i-t+2}} + \cdots + \frac{1}{q_i} + 1 \right) B_{rs} \approx \left(\frac{1}{q_i} + 1 \right) B_{rs}$$

*Hence, composite scaling results in less than $\log\left(\frac{1}{q_i} + 1\right) \approx \frac{3.322}{q_i}$ bit **precision loss**, which is **negligible**, compared to single scaling.*

Experimental Results

7-layer CNN Inference (CIFAR-10)

- Implementation in OpenFHE with longitudinal packing
  - **Unit tests** with **Same Precision**

Unit tests	Precision bits 64-bit single scaling	Precision bits 32-bit composite scaling
Fully connected	39	39
ReLU	40	40
Mean pool	41	41
Convolution	39	39
Bootstrapping	12	12

========== **Parameters** ==========

Ring dimension : **65536**

Scaling factor : 2^{58}

- **Same** for both cases

Primes

- **58-bit** primes for 64-bit case
- **(29, 29)-bit** primes for 32-bit case
  - Double-prime scaling
  - 58 = 29 + 29

Security

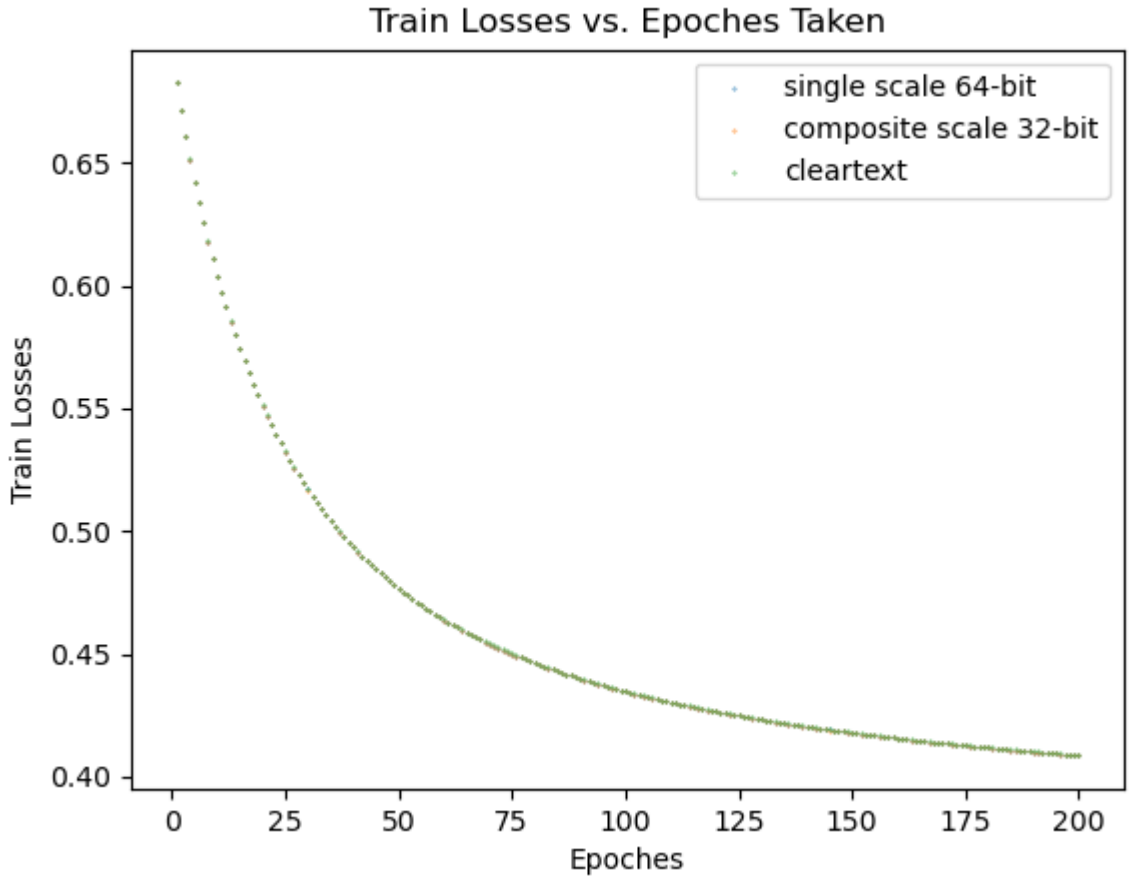
- **Same** for both cases

- Implementation in Lattigo with multiplexed packing
  - The **end-to-end CNN Inference results match** up to **5** digits after the decimal point
  - 14 consecutive bootstrapping (2 per layer, before and after ReLU)

Experimental Results

Logistic Regression Training

- Reference code: https://github.com/openfheorg/openfhe-logreg-training-examples
- 1 bootstrapping per epoch



========== Parameters ==========

Ring dimension : 32768

Scaling factor : 2^{58}

- **Same** for both cases

Primes

- **58-bit** primes for 64-bit case
- **(29, 29)-bit** primes for 32-bit case
  - Double-prime scaling
  - 58 = 29 + 29

Security

- **Same** for both cases

 # Bootstrapping

- **Same** for both cases

Wrap-up

- **Result:** Enable high-precision RNS-CKKS on small word-size architectures without multi-precision arithmetic
  - Use of small word-size: GPU, FPGA, Embedded devices, etc.
  - Arbitrary precision for bootstrapping combined with Meta-BTS
- **Limitation:** Choice of scaling factor
  - Lower bound exists on each prime (NTT condition)
  - E.g., $\Delta = 2^{40} \rightarrow$ two 20-bit primes for double-prime scaling
    - How many 20-bit "NTT-friendly" primes exist for the dimension $N = 2^{16}$?
  - Several small intervals that are not usable as scaling factor
- **Implementation:** Not public yet but planning for open-sourcing composite-scaling variant of OpenFHE-CKKS

thank you!

https://eprint.iacr.org/2023/1462