FHE16 CNN

Youngjun Kim
yjkim@walllnut.com
june0888@hanyang.ac.kr

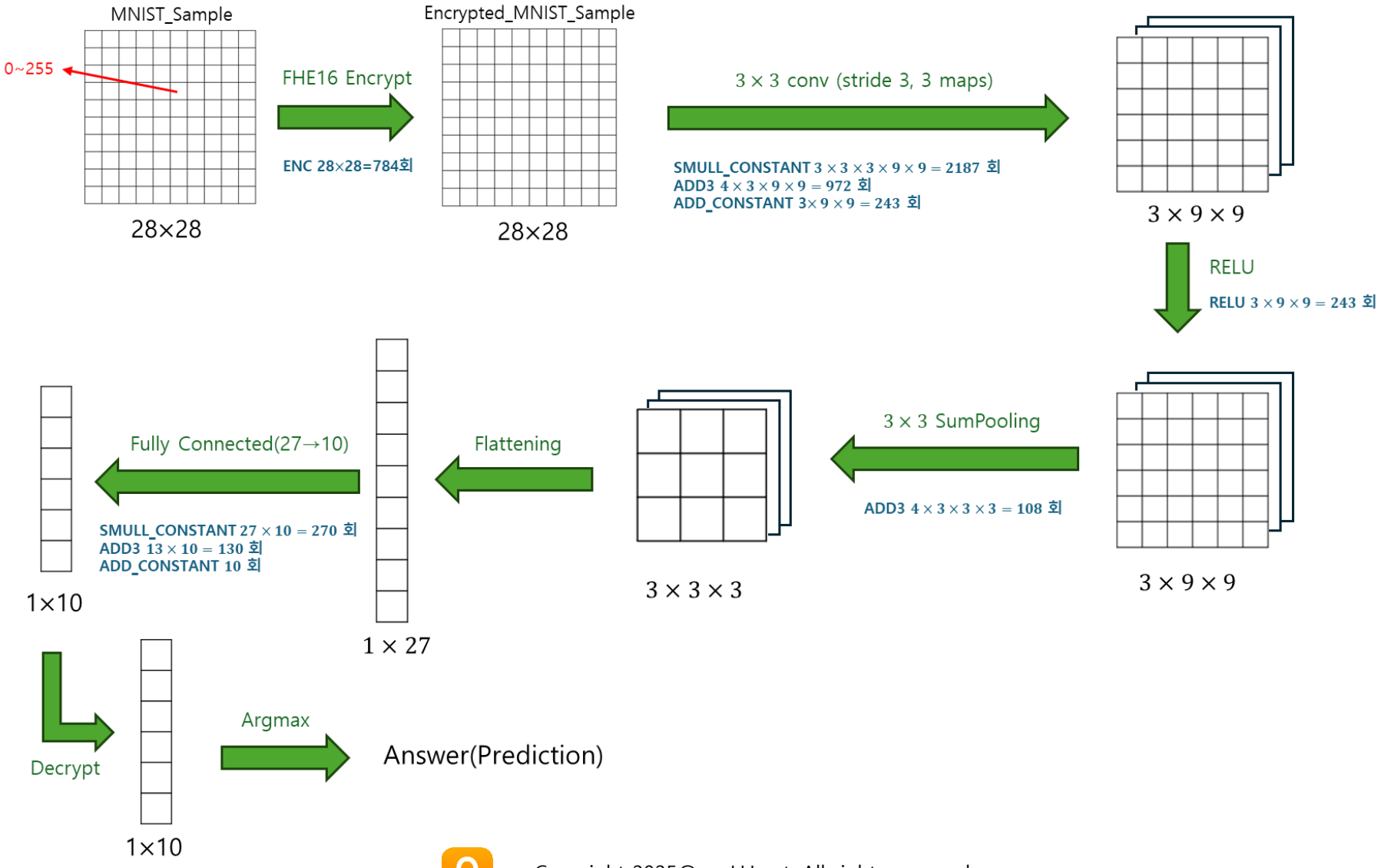
Overview

- The CNN model was trained on the MNIST dataset, and inference was performed using an FHE16-encrypted MNIST test set to evaluate its performance.
- In the FHE16 environment, nonlinear activation functions such as Softmax and Sigmoid cannot be implemented, and only integer arithmetic is supported.
- As a result, the magnitude of output values grows exponentially through layers, and increasing the bit width to prevent overflow leads to higher computation time.
- Considering these constraints, the FHE16 CNN was designed with a streamlined structure

Conv → ReLU → SumPool → FC → argmax

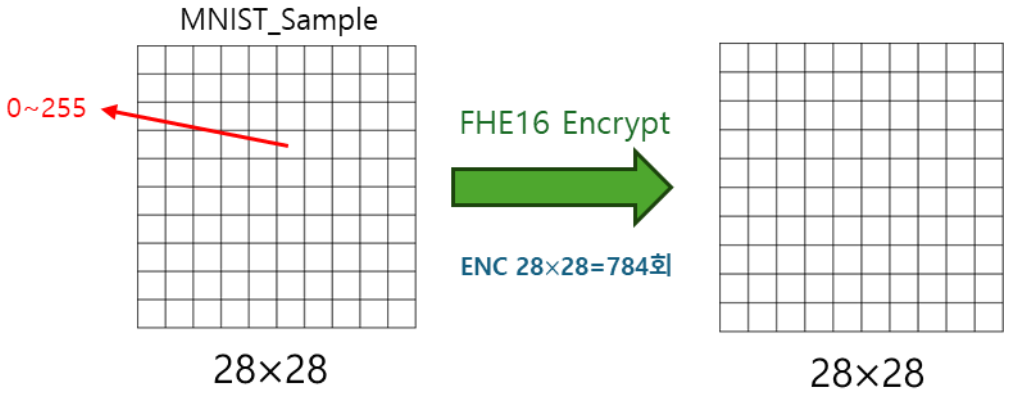
which maintains stable classification performance under integer-only computation.

Model Architecture



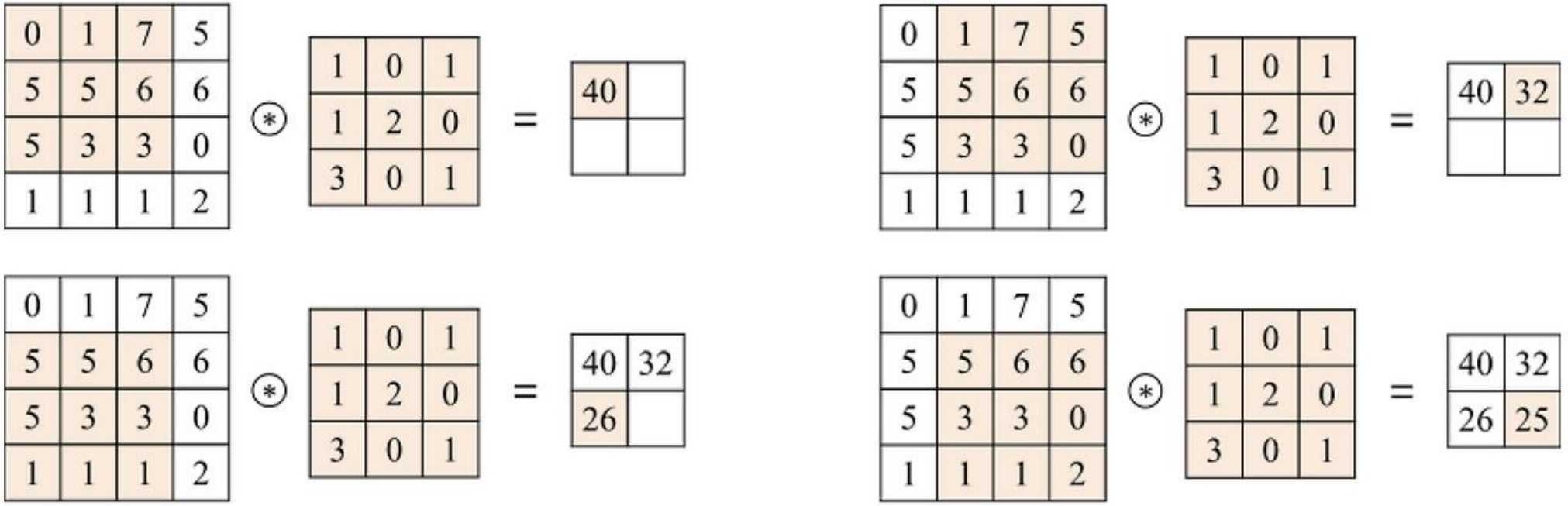
FHE16 Encryption

- The process of encrypting MNIST samples with FHE16.
- Since each MNIST image has a size of 28×28, a total of 784 encryption operations are performed per image.



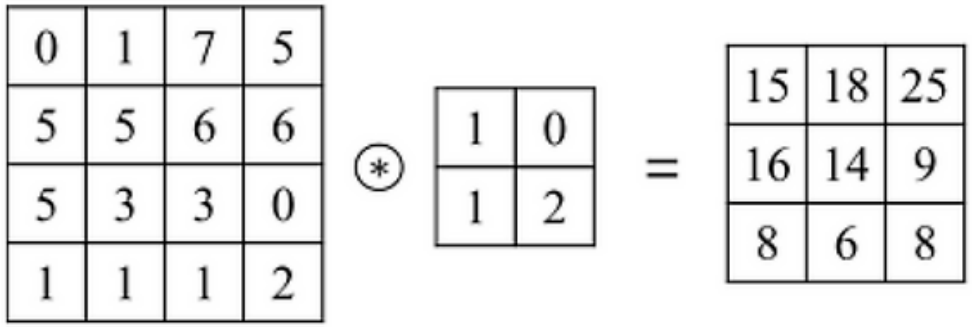
Convolution Layer

- In a CNN, the convolution operation applies a filter, moving it across the input along the rows and columns by the given stride, and performs element-wise multiplication and addition as shown in the figure below.

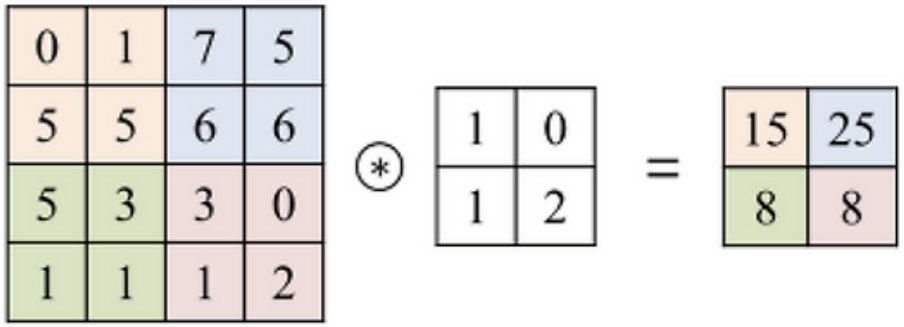


Convolution Layer

- Stride is a parameter that determines how many steps the filter moves along the rows and columns during the convolution operation.



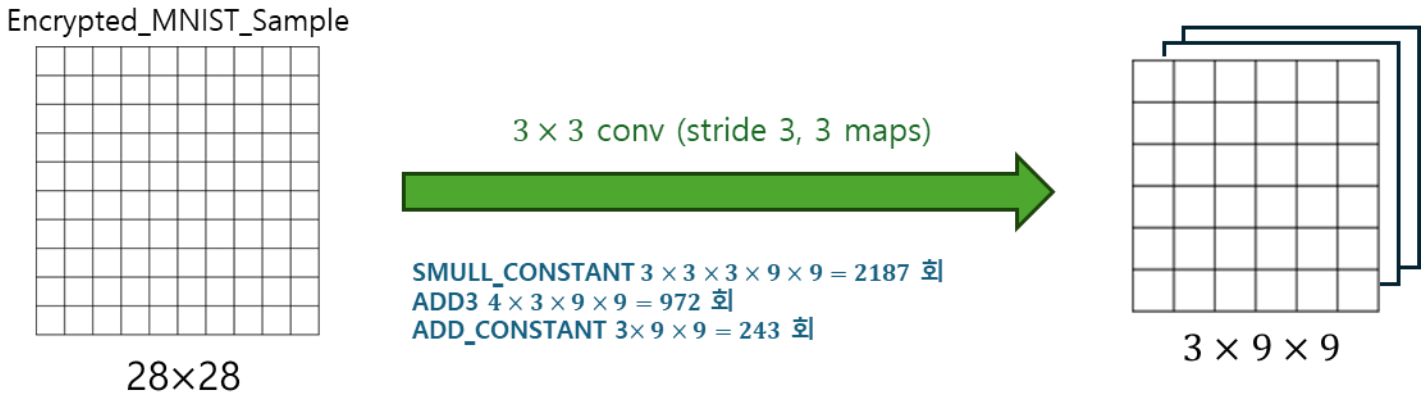
[Stride=1]



[Stride=2]

Convolution Layer

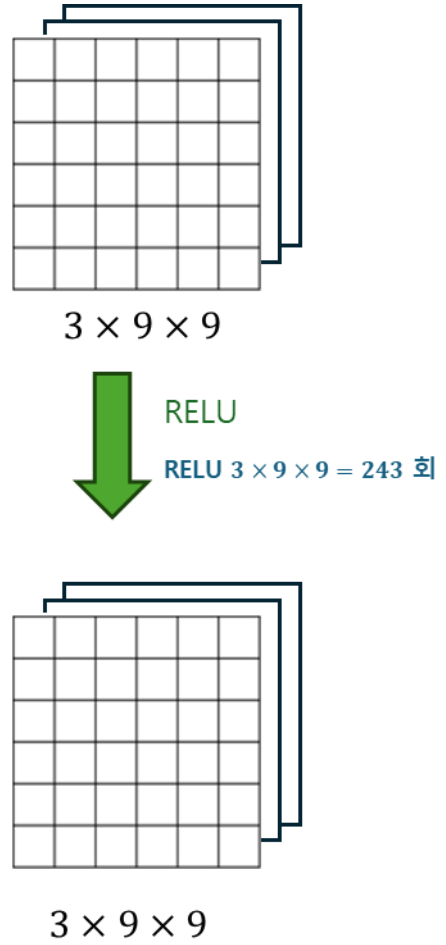
- In the FHE16 CNN, the convolution operation is performed using a 3×3 filter with a stride of 3.
  - Each value within the filter corresponds to a trained weight.
  - To extract more features, the number of maps (channels) is set to 3.
  - When summing the 9 values within each 3×3 region, the computation is optimized by using the ADD3 operation four times to obtain the result.
  - The scalar addition is used to add the bias term.



ReLU

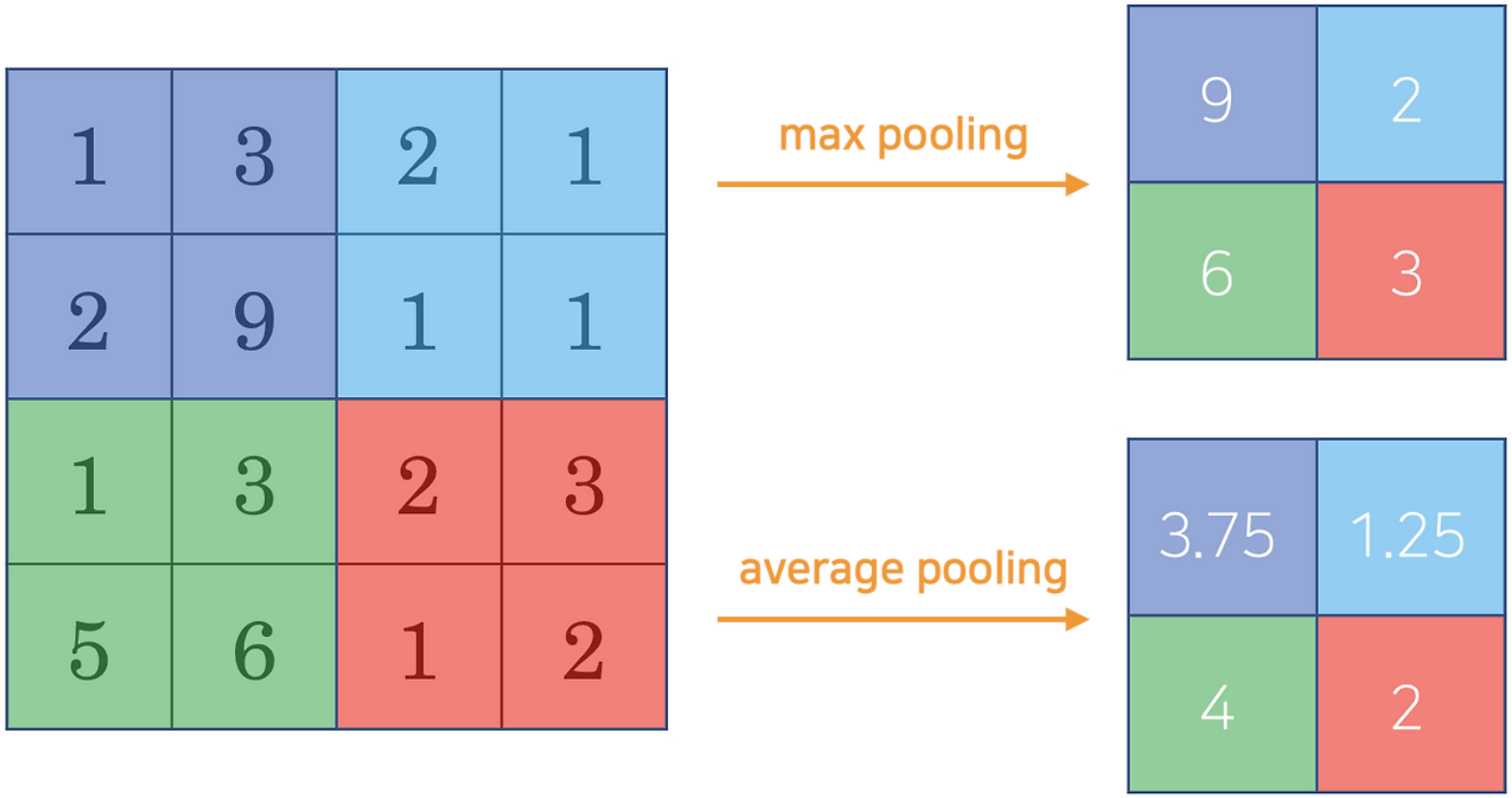
- It is used to reduce gradient vanishing and make the training process more stable.

$$ReLU(x) = \max(0, x)$$



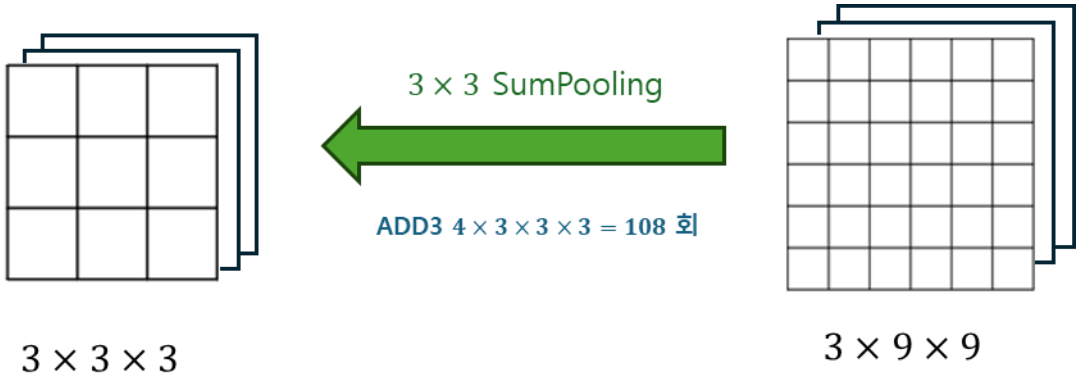
Sumpooling Layer

- A pooling layer is used to summarize information by taking operations such as maximum or average over the values within each pool.



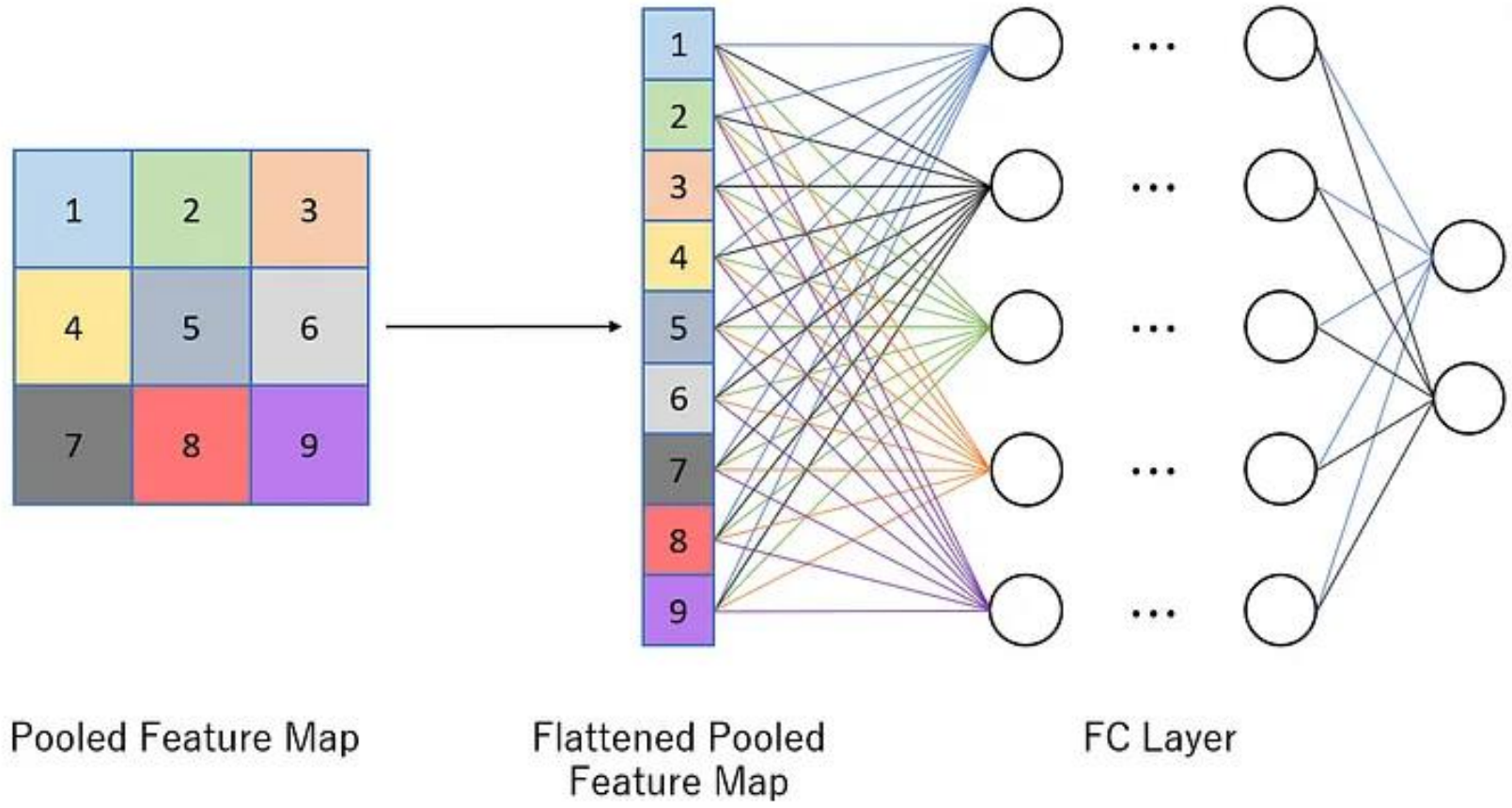
Sumpooling Layer

- In the FHE16 CNN, 3×3 SumPooling is used, and similar to the convolution layer, the process of summing 9 values is optimized by applying the ADD3 operation 4 times.



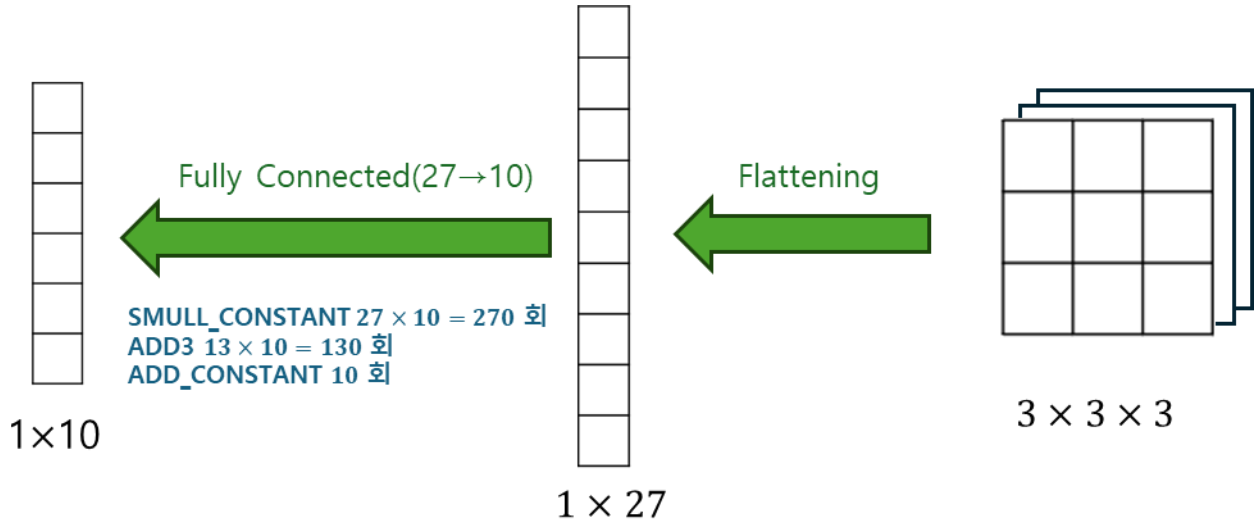
Fully Connected Layer

- After passing through the pooling layer, the data is converted into a one-dimensional vector (flattening), and computations are performed using the trained weights.



Fully Connected Layer

- In the FHE16 CNN, there is no hidden layer, and the input is directly connected to 10 class (label) nodes representing digits 0 through 9.
- For each class, 27 values are summed, optimized to perform the ADD3 operation 13 times (9 + 3 + 1), and an scalar addition is executed to add the bias term.



Experimental Results (32-bit)

- The FHE16 CNN model used in this study achieved 81.82% accuracy with floating-point weights, and 80.47% accuracy with integer weights quantized using a scale factor of 64.

```
[Epoch 20] test acc (float) = 81.82%
[Eval] quantized acc (scale=64) = 80.47%
[Stats] max|conv|=0.051110, max|fc|=0.496301
[Quant] exporting with scale = 64
```

Experimental Results (32-bit)

Estimated Time

Encryption: 784 × 150ms(ENCInt)≈**118s**
Convolution: 2187 × 102ms(SMULL_CONSTANT) + 972 × 118ms(ADD3) + 243 × 100ms (ADD_CONSTANT)≈ **358s**
ReLU: 243 × 14ms(RELU)≈**3s**
SumPooling: 108 × 118ms(ADD3)≈ **13s**
FC: 270 × 101ms(SMULL_CONSTANT) + 130 × 100ms(ADD3) + 10 × 82ms(ADD_CONSTANT) ≈ **41s**
Total: **533s**

Measured Time

```
Encryption: 117.895 s
Conv: 259.830 s
ReLU: 3.512 s
SumPool: 13.023 s
Flatten: 0.000 s
FC: 39.944 s
Argmax(decrypt): 0.001 s
========================================
True label : 7
Predicted  : 7 (logit=81173)
Total time : 434.204 s
========================================
```

Execution Time per Function

```
FHE16_ENCInt         : 150.758 ms
FHE16_ADD            : 100.944 ms
FHE16_SUB            : 100.563 ms
FHE16_ADD3           : 118.522 ms
FHE16_SMULL_CONSTANT: 101.707 ms
FHE16_ADD_CONSTANT   : 82.095 ms
FHE16_RELU           : 14.345 ms
FHE16_MAX           : 101.890 ms
```

Experimental Results (32-bit)

- Results of inference on 1,000 MNIST test samples.
- For each batch of 100 samples, accuracy, the difference (Diff) from the plaintext inference result, and the execution time are reported. (The experiment was run with parameters optimized for faster execution, but the computation process and results remain identical.)

```
Batch,Correct,Accuracy(%),Diff,Time(s)
[Batch 0] Accuracy=82/100 (82%) Diff=0/100 Time=1690.32 s
Batch 0 [Batch 0] Accuracy=82/100 (82.000%) Diff=0/100 Time=1690.317 s
[Batch 1] Accuracy=90/100 (90%) Diff=0/100 Time=1689.29 s
Batch 1 [Batch 1] Accuracy=90/100 (90.000%) Diff=0/100 Time=1689.291 s
[Batch 2] Accuracy=84/100 (84%) Diff=0/100 Time=1695.83 s
Batch 2 [Batch 2] Accuracy=84/100 (84.000%) Diff=0/100 Time=1695.826 s
[Batch 3] Accuracy=80/100 (80%) Diff=0/100 Time=1684.01 s
Batch 3 [Batch 3] Accuracy=80/100 (80.000%) Diff=0/100 Time=1684.014 s
[Batch 4] Accuracy=74/100 (74%) Diff=0/100 Time=1689.19 s
Batch 4 [Batch 4] Accuracy=74/100 (74.000%) Diff=0/100 Time=1689.188 s
[Batch 5] Accuracy=72/100 (72%) Diff=0/100 Time=1688.08 s
Batch 5 [Batch 5] Accuracy=72/100 (72.000%) Diff=0/100 Time=1688.078 s
[Batch 6] Accuracy=72/100 (72%) Diff=0/100 Time=1683.91 s
Batch 6 [Batch 6] Accuracy=72/100 (72.000%) Diff=0/100 Time=1683.911 s
[Batch 7] Accuracy=78/100 (78%) Diff=0/100 Time=1694.37 s
Batch 7 [Batch 7] Accuracy=78/100 (78.000%) Diff=0/100 Time=1694.375 s
[Batch 8] Accuracy=78/100 (78%) Diff=0/100 Time=1696.88 s
Batch 8 [Batch 8] Accuracy=78/100 (78.000%) Diff=0/100 Time=1696.884 s
[Batch 9] Accuracy=70/100 (70%) Diff=0/100 Time=1702.77 s
Batch 9 [Batch 9] Accuracy=70/100 (70.000%) Diff=0/100 Time=1702.772 s
================================
Total correct: 780 / 1000
Total diff: 0 / 1000
Total accuracy: 78.00%
Total diff rate: 0.00%
```

Applied Optimization Techniques

1. The frequently used value FHE16_ENCInt(0) is stored in a global variable Z, which is referenced whenever needed.
2. Multiplication Optimization
    - In scalar multiplication, if the scalar value is 0, the variable Z is returned; if it is 1, the operand itself is returned without modification.
3. Addition Optimization
    - When adding a plaintext bias value, the operation is performed as a scalar addition.
4. ADD3 Optimization
    - For addition operations, ADD3 is used preferentially whenever possible to improve efficiency.

Methods to Further Reduce Inference Time

1. Model Simplification
   - The current model is already sufficiently lightweight, and further reducing the accuracy would cause excessive performance degradation. Therefore, additional simplification is considered practically infeasible.
2. Bit Precision Adjustment
   - Reducing the bit precision decreases computation time, but the risk of overflow in the final output must be considered. Under the current configuration, the theoretical maximum value is calculated as follows:

$$2^8(\text{maximum MNIST pixel value}) \times \underbrace{2^6(\text{maximum weight value})}_{\text{Conv layer}} \times 9 \times 9(\text{SumPooling})$$

$$\times \underbrace{2^6(\text{maximum weight } \textit{value}) \times 27}_{\text{FC layer}} < 2^{20} \times 3^7 < 2^{33}$$

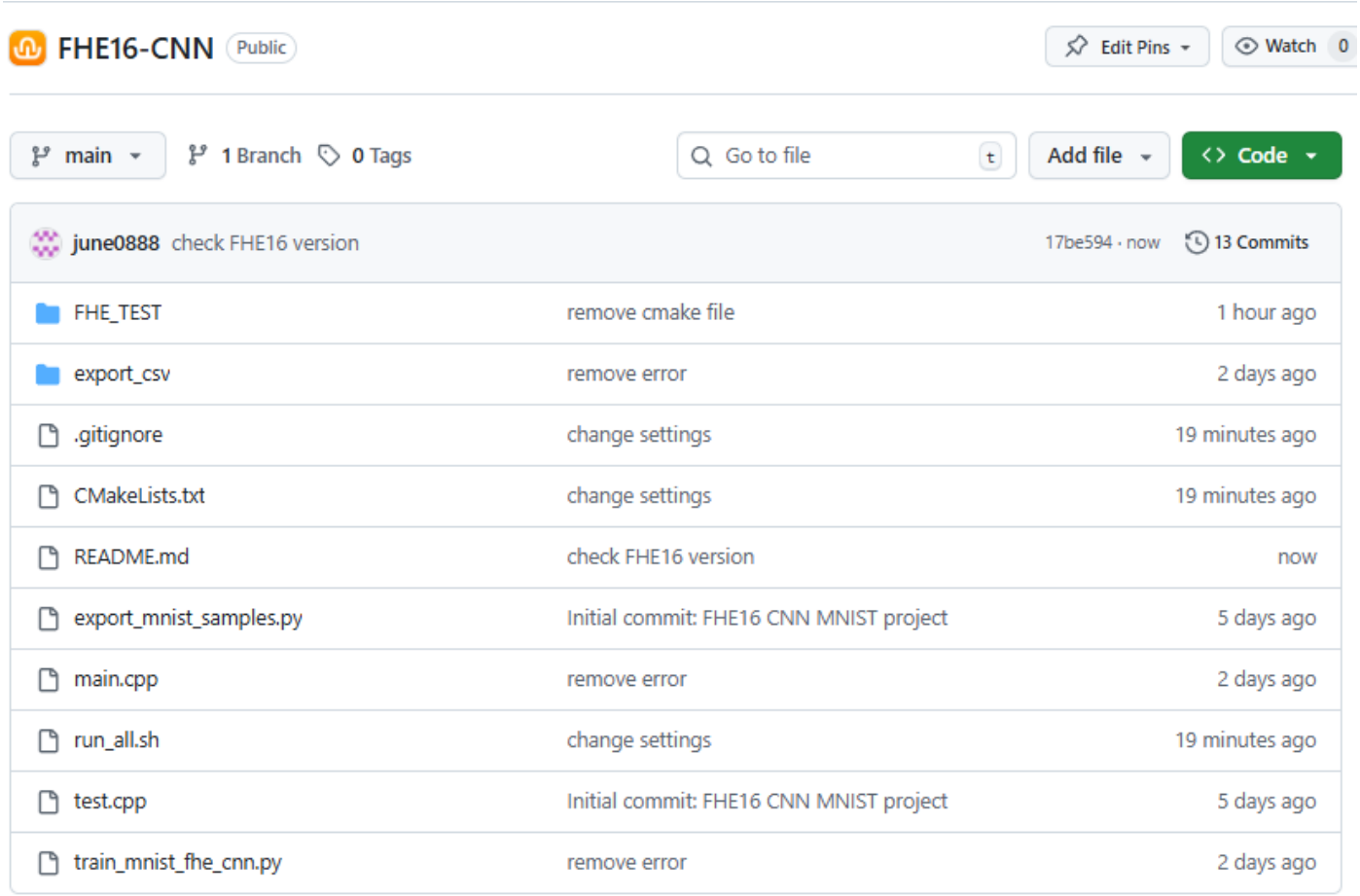
Methods to Further Reduce Inference Time

Thus, at least 33 bits are required to conservatively prevent overflow.

To address this, bit-shift or divide operations can be inserted between layers to reduce intermediate values, lowering the required bit width and shortening the computation time.

3. Operation Optimization and Parallelization

Git code (https://github.com/waLLLnut/FHE16-CNN)



Git code (https://github.com/waLLLnut/FHE16-CNN)

export_mnist_samples.py

- Code for downloading the MNIST dataset.

test.cpp

- Performs inference on a single MNIST test sample.
- To test a different sample, modify the file name inside the code.

main.cpp

- Performs inference on 100 MNIST test samples.
- Using the run_all.sh script, the program is executed 10 times to process a total of 1,000 samples, and the results are saved to accuracy_log.txt.

train_mnist_fhe_cnn.py

- Trains the FHE16-CNN model.
- After training is completed, the weights are saved in the export_csv folder.

CPU Performance

```
Architecture:                    x86_64
  CPU op-mode(s):                32-bit, 64-bit
  Address sizes:                 46 bits physical, 48 bits virtual
  Byte Order:                    Little Endian
CPU(s):                          48
  On-line CPU(s) list:           0-47
Vendor ID:                       GenuineIntel
  Model name:                    Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz
    CPU family:                  6
    Model:                       85
    Thread(s) per core:          2
    Core(s) per socket:          24
    Socket(s):                   1
    Stepping:                    7
    CPU(s) scaling MHz:          25%
    CPU max MHz:                 4000.0000
    CPU min MHz:                 1000.0000
    BogoMIPS:                    4800.00
```