FHE16 CNN

김영준
yjkim@walllnut.com
june0888@hanyang.ac.kr

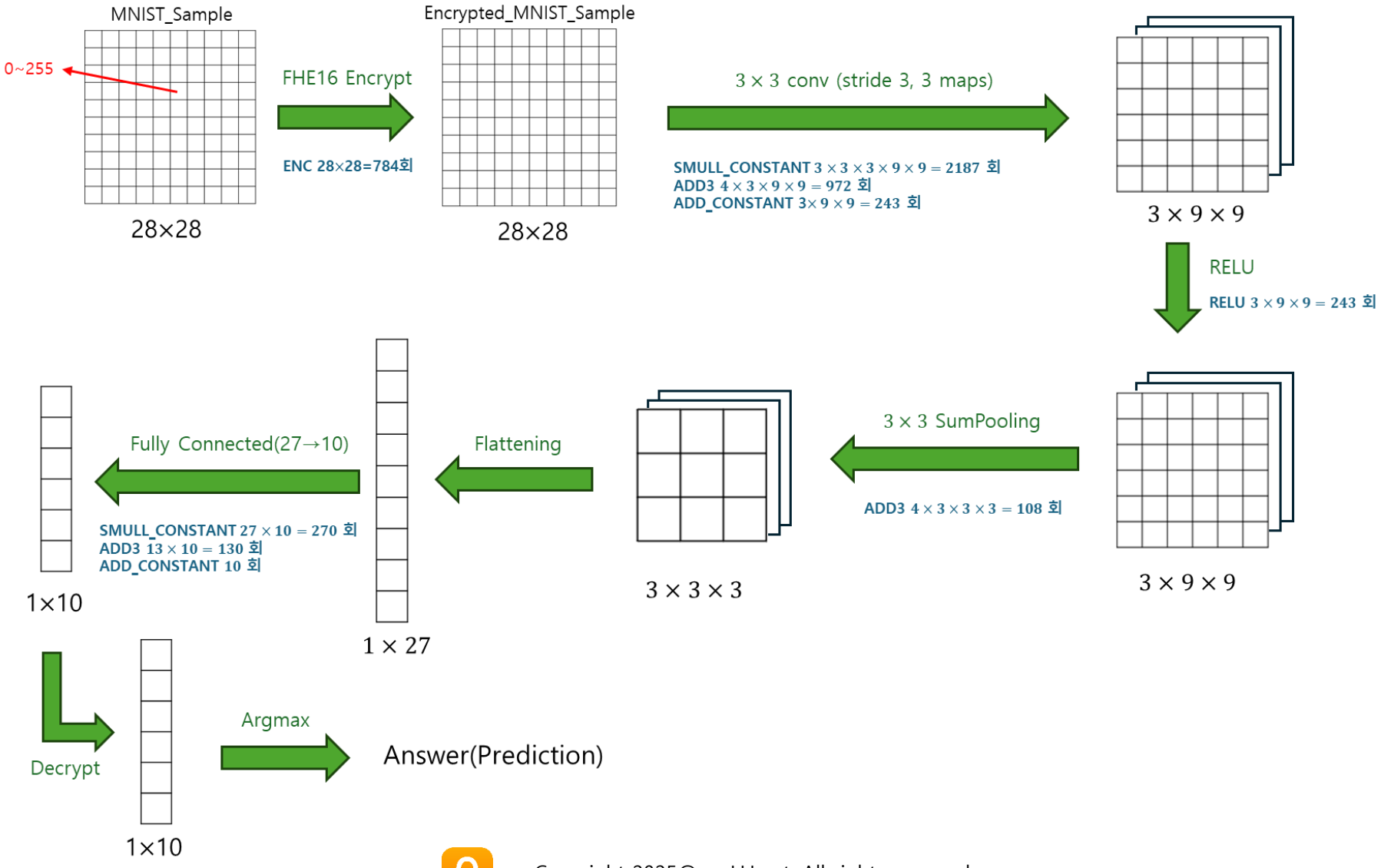
개요

- CNN 모델을 MNIST 데이터셋으로 학습한 뒤, FHE16으로 암호화된 MNIST test set을 입력하여 추론(inference)을 수행하고 성능을 평가하였다.
- FHE16 환경에서는 Softmax나 Sigmoid와 같은 비선형 활성화 함수를 구현할 수 없으며, 정수 연산만을 지원하기 때문에 레이어를 거치며 출력값의 크기가 기하급수적으로 증가한다.
- 이러한 과정에서 overflow를 방지하기 위해 bit 수를 늘릴 경우 연산 시간이 증가하는 한계가 존재한다.
- 이에 따라, 본 모델은 FHE16 환경의 연산 제약을 고려하여

Conv → ReLU → SumPool → FC → argmax

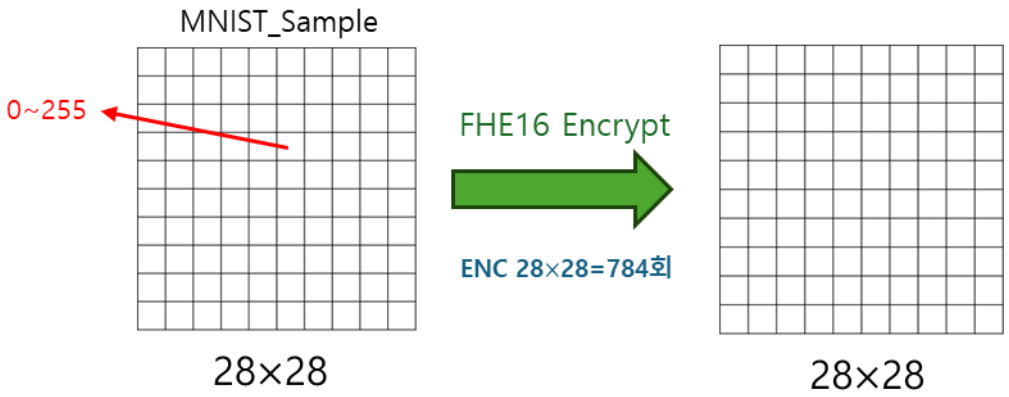
형태의 구조로 설계하였으며, 정수 연산 기반에서도 안정적인 분류 성능을 유지한다.

모델 전체 구조



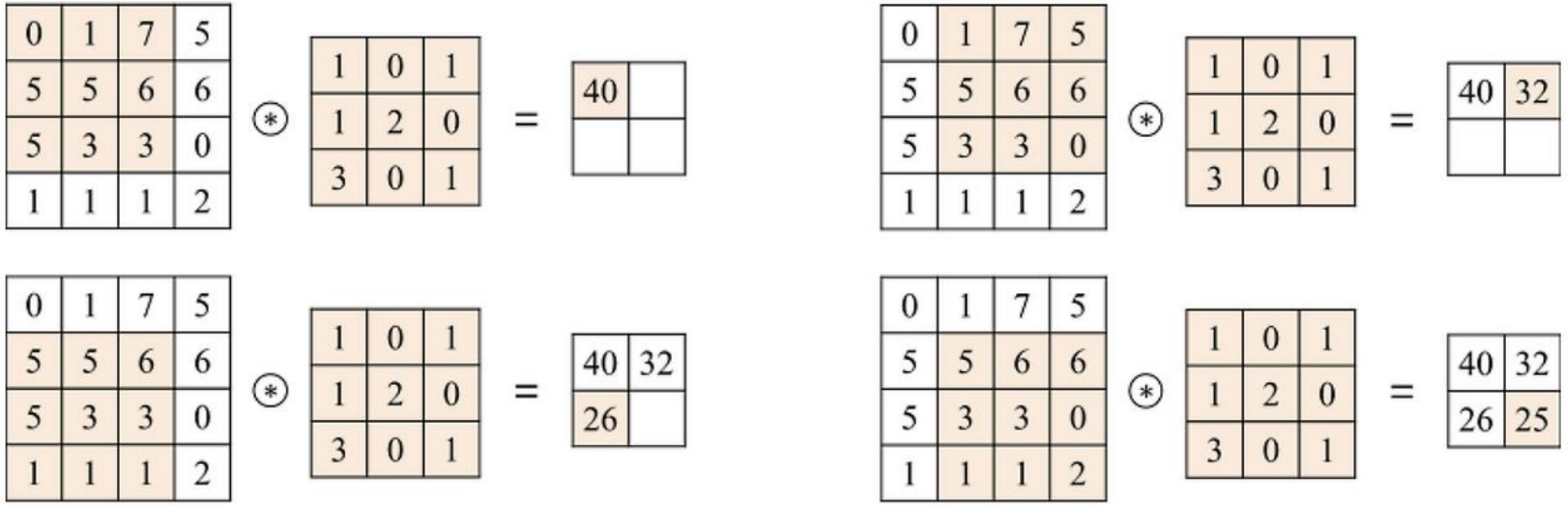
FHE16 Encryption

- MNIST 샘플을 FHE16으로 암호화하는 단계
- MNIST는 28×28 크기의 데이터이므로 하나의 이미지에 대해 총 784회의 encryption 연산이 수행된다.



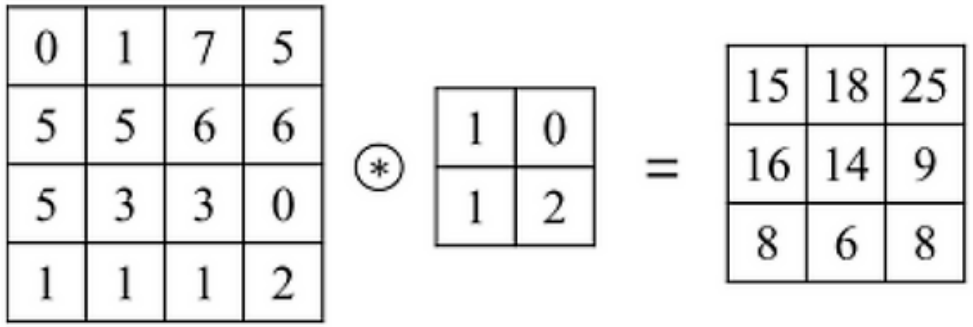
Convolution Layer

- CNN에서 convolution 연산은 다음 그림과 같이 convolution filter를 stride 만큼 행과 열 방향으로 이동시키며, element-wise 하게 곱셈과 덧셈 연산을 수행한다.

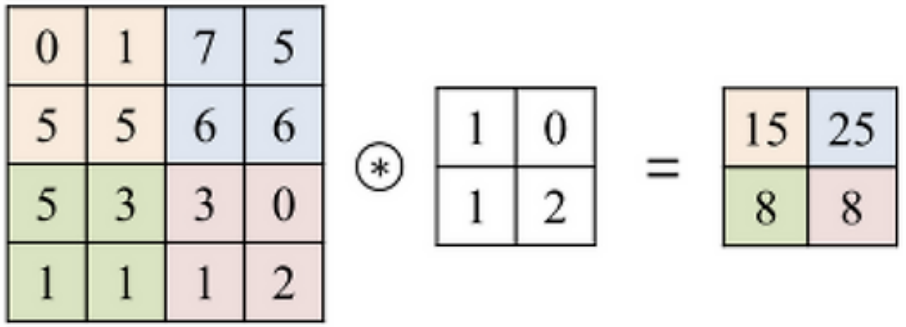


Convolution Layer

- Stride는 Convolution 연산을 수행할 때, 필터를 행과 열 방향으로 몇 칸씩 건너뛰며 적용할지를 결정하는 파라미터이다.



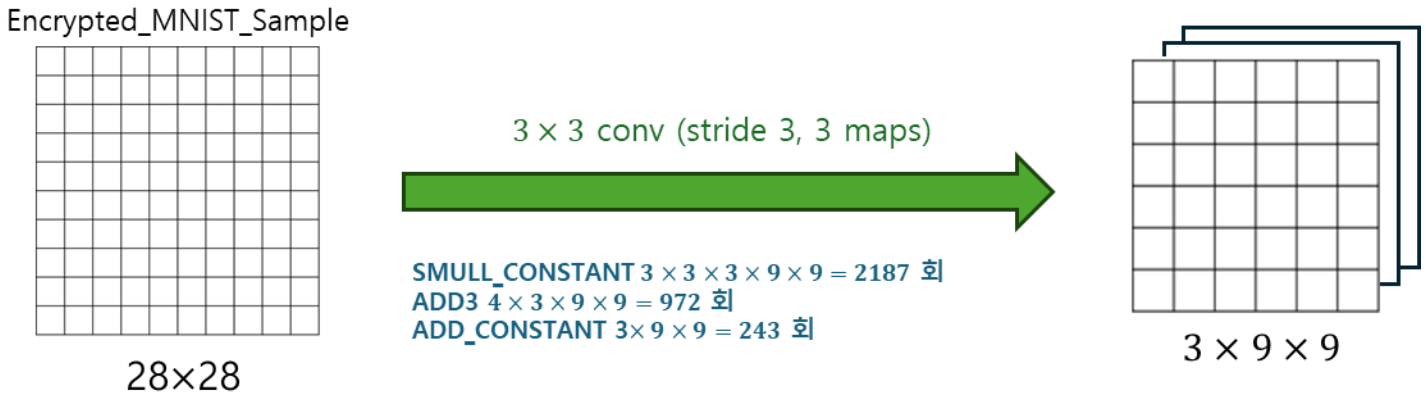
[Stride=1]



[Stride=2]

Convolution Layer

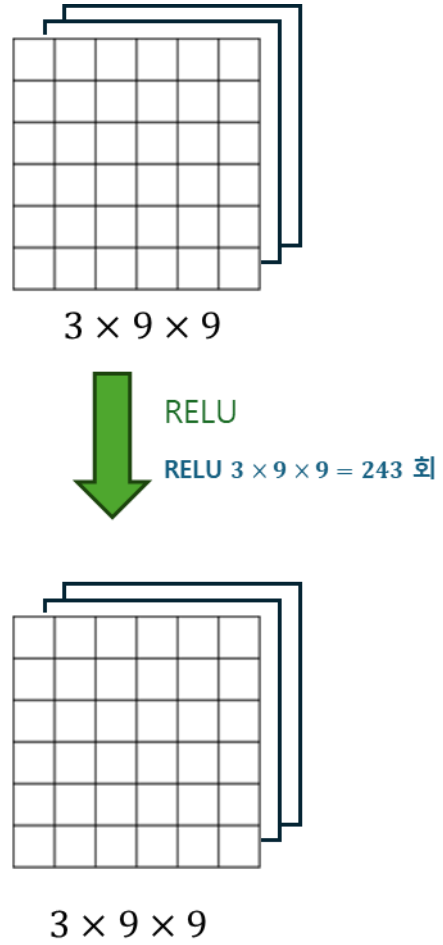
- FHE16 CNN에서는 3×3 filter를 stride 3으로 적용하여 convolution 연산을 수행한다.
  - 필터 내의 각 값은 학습된 weight 값으로 구성되어 있다.
  - 특징(feature)을 더 많이 추출하기 위해 map(channel) 수는 3개로 설정하였다.
  - 3×3 영역의 총 9개 값을 더할 때 ADD3 연산을 4회 사용하여 결과값을 계산하도록 최적화하였다.
  - Scalar 덧셈 연산은 bias를 더하기 위해 사용된다.



ReLU

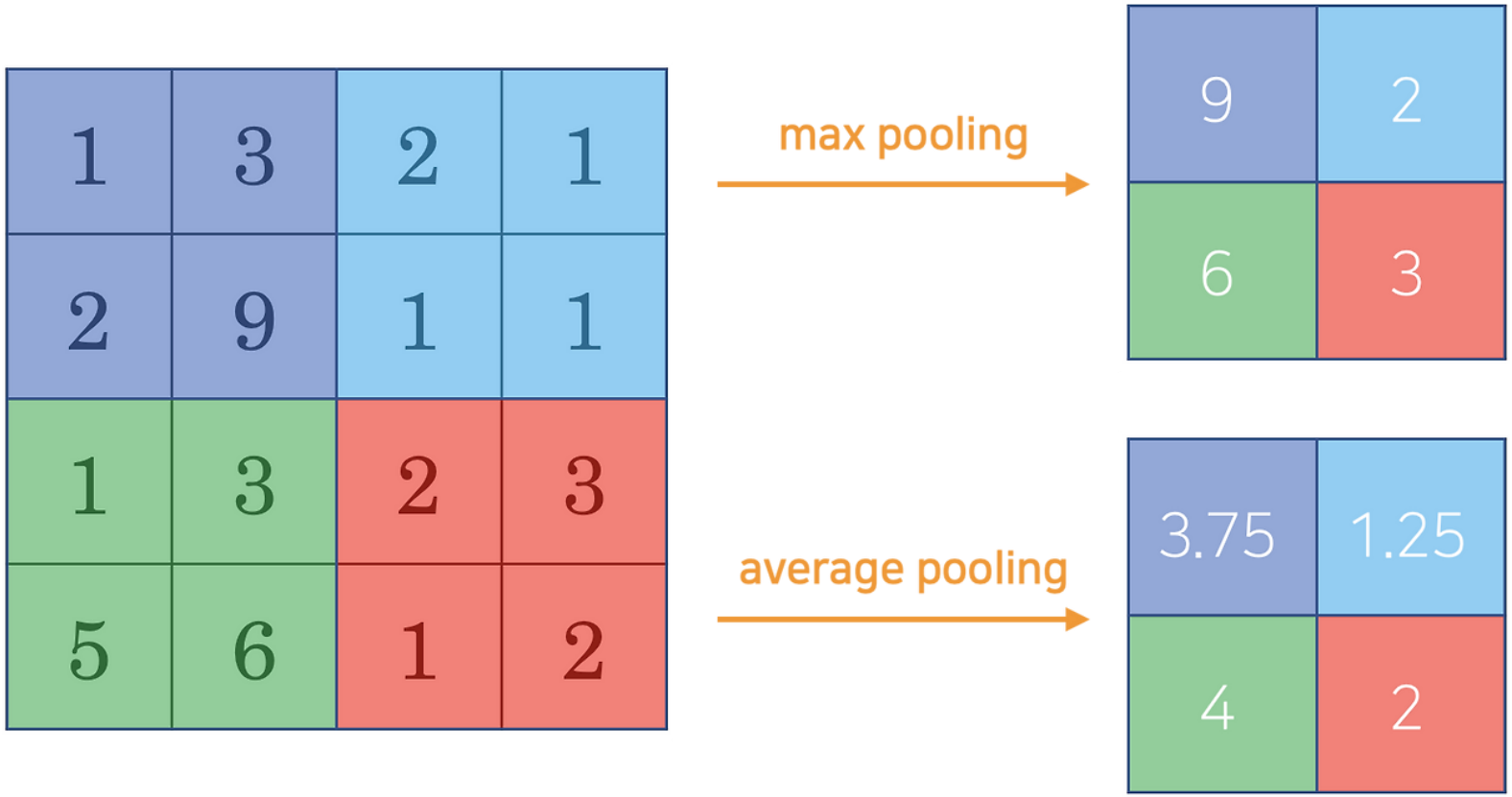
- Gradient 소실을 줄여 학습을 보다 안정적으로 진행하기 위해 사용된다.

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



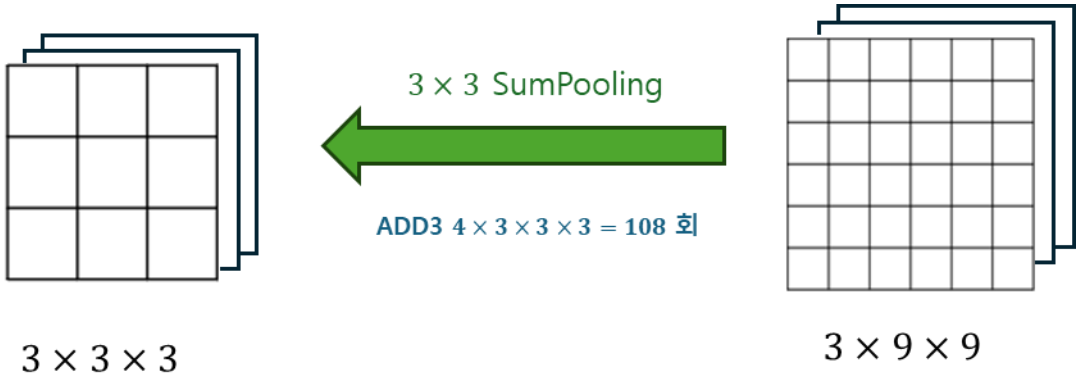
Sumpooling Layer

- Pooling layer는 정보를 요약하기 위해 pool 내의 값들에 대해 최대값(Max)이나 평균(Average) 등을 취하는 레이어이다.



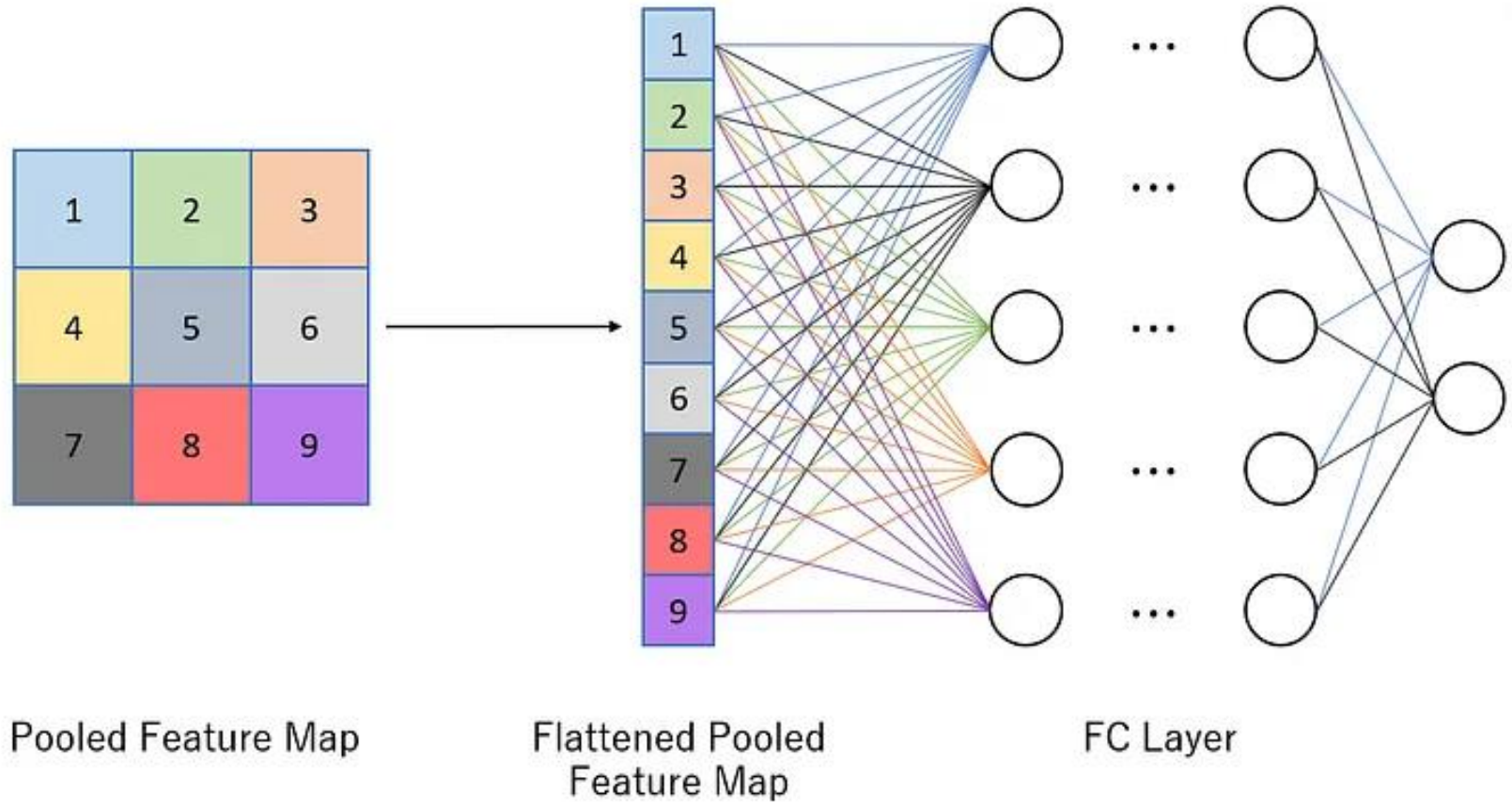
Sumpooling layer

- FHE16 CNN에서는 3 × 3 Sumpooling을 사용하였으며, Convolution layer와 마찬가지로 9개의 값을 더하는 과정에서 ADD3 연산을 4번 사용하는 최적화를 적용하였다.



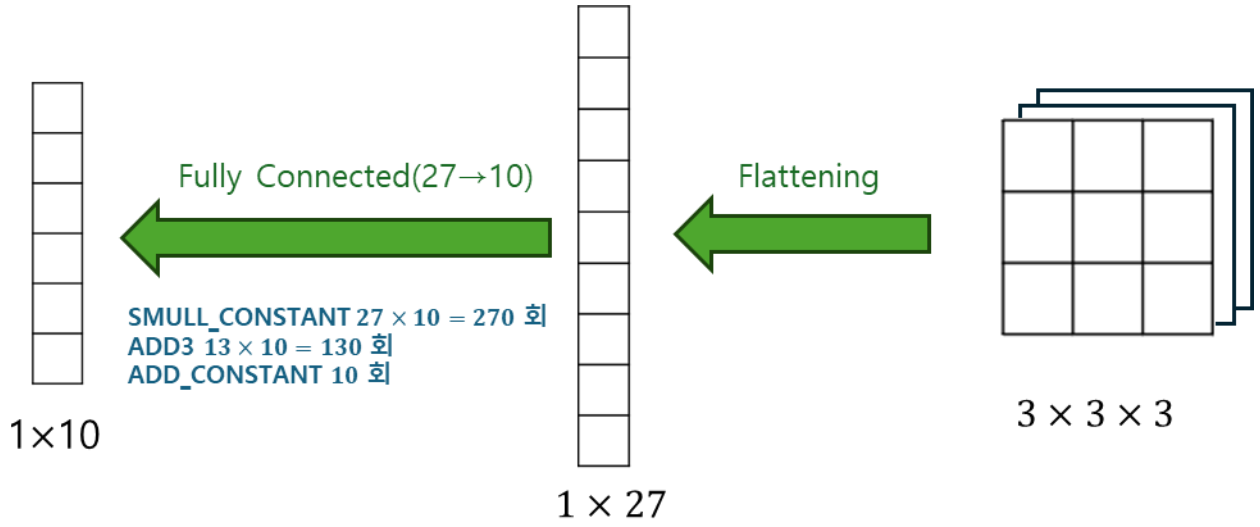
Fully Connected Layer

- Pooling layer를 거친 데이터를 1차원 벡터로 변환(flattening)한 후, 학습된 weight를 사용하여 연산을 수행한다.



Fully Connected layer

- FHE16 CNN에서는 Hidden layer 없이, 입력값이 0~9를 나타내는 10개의 클래스(label) 노드에 직접 연결된다.
- 각 클래스별로 27개의 값을 더해야 하므로 ADD3을 13회(9+3+1) 수행하도록 최적화하였으며, bias를 더하기 위해 1회의 scalar 덧셈 연산이 실행된다.



실험 결과(32-bit)

- FHE16 CNN 모델은 float point weight 기준 81.82%의 정확도를, scale factor=64로 양자화된 int weight 기준으로는 80.47% 정확도를 나타내는 모델을 사용하였다.

```
[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
```

실험 결과(32-bit)

예상 소요 시간

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**

실제 결과

```
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
==========================================
```

각 함수별 연산 시간

```
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
```

실험 결과(32-bit)

- MNIST test sample 1000개에 대한 inference 결과
- 각 배치(100개) 별 Accuracy, 평문 inference 결과와의 차이 수(Diff), 연산 시간을 표기하였다.
  (빠른 실행을 위한 파라미터 세팅 환경에서 실행되었으며, 연산 과정 및 결과는 동일)

```
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%
```

적용된 연산 최적화 기법

1. 자주 사용되는 FHE16_ENCInt(0) 값을 전역 변수 Z로 저장하여 필요할 때마다 해당 전역 변수에서 값을 불러온다.
2. 곱셈 최적화
   - Scalar 곱셈에서 scalar 값이 0이면 변수 Z를 불러오고, 1이면 자기 자신을 그대로 반환한다.
3. 덧셈 최적화
   - 평문으로 공개된 값인 bias 를 더하는 경우에는 scalar 덧셈 연산으로 수행한다.
4. ADD3 최적화
   - 덧셈 연산 시 가능한 한 ADD3 연산을 우선적으로 사용하도록 구현하였다.

Inference 시간을 더 줄일 수 있는 방법

1. 모델 경량화
   - 현재 모델은 이미 충분히 경량화되어 있으며,이보다 정확도를 더 낮출 경우 성능 저하가 과도하게 발생하므로 추가적인 경량화는 사실상 불가능하다고 판단된다.
2. 비트 수 조정
   - 비트 수를 줄일수록 연산 시간이 감소하지만, 최종 결과값의 overflow를 고려한 조절이 필요하다.
     - 현재 세팅에서 이론상 최댓값을 계산하면

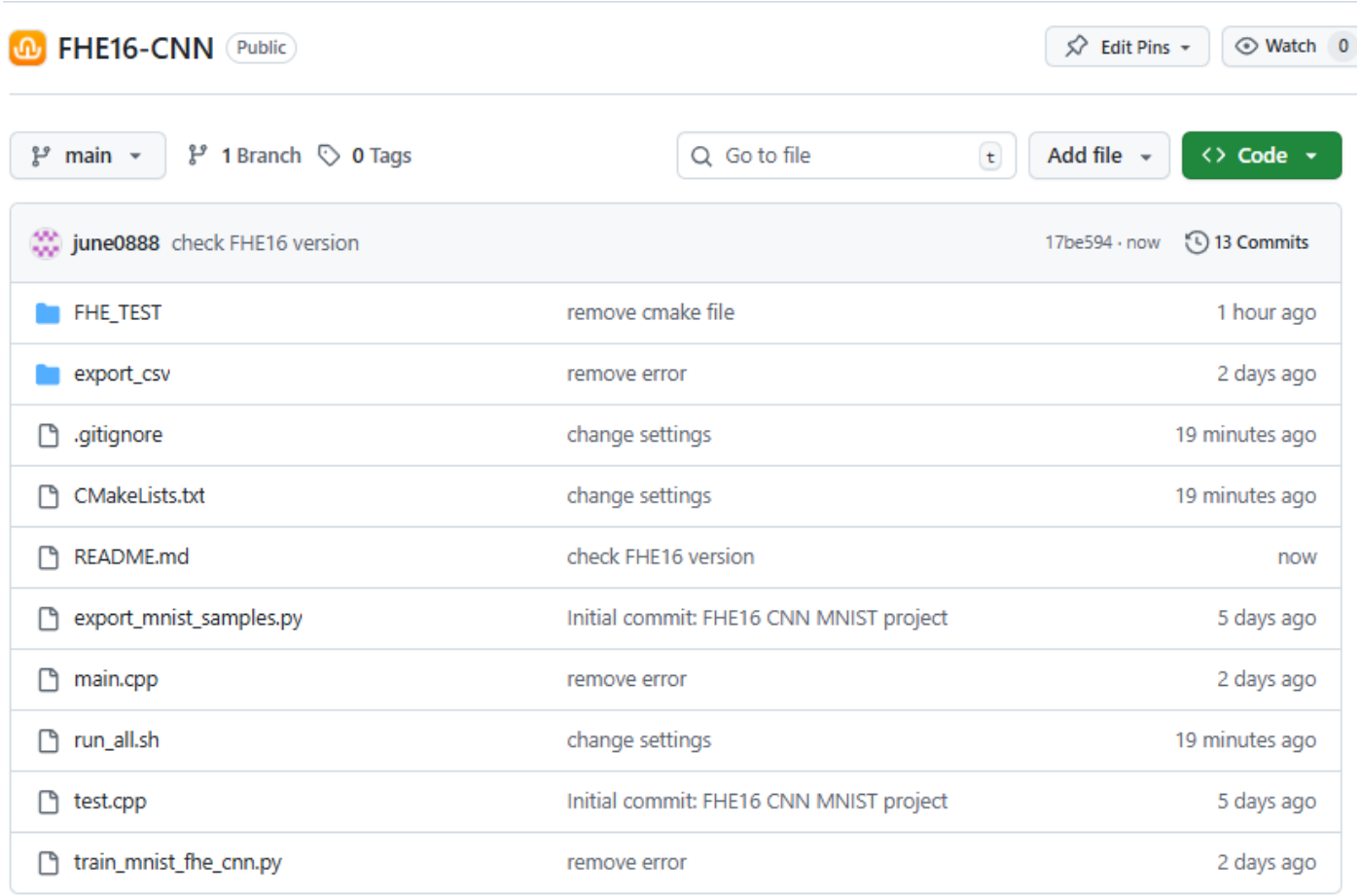
       2^8(MNIST 픽셀 하나의 최댓값)× $\underbrace{2^6(\text{weight최댓값})\times 9}_{\text{Conv layer}}$ × 9(SumPooling)× $\underbrace{2^6(\text{weight최댓값}) \times 27}_{\text{FC layer}}$

       $$< 2^{20} \times 3^7 < 2^{33}$$

       따라서, overflow를 보수적으로 방지하기 위해서는 33비트 이상을 사용해야 한다.

       이를 해결하기 위해 bit shift 또는 divide 연산을 레이어 중간에 삽입하여 결과값의 크기를 줄임으로써, 필요 비트 수를 줄이고 연산 시간을 단축시킬 수 있다.
3. 연산 최적화, 병렬화

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



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

export_mnist_samples.py

- MNIST data set 다운받는 코드

test.cpp

- MNIST test 데이터 1개에 대해 inference를 수행한다.
- 다른 데이터를 테스트하고 싶을 경우, 코드 내에서 파일 이름을 수정하면 된다.

main.cpp

- MNIST test 데이터 100개에 대해 inference를 수행하며, run_all.sh 스크립트를 이용해 10회 반복 실행하여 총 1000개의 데이터에 대한 inference 결과를 accuracy_log.txt에 저장한다.

train_mnist_fhe_cnn.py

- FHE16-CNN 모델을 학습시키는 코드로, 학습이 완료된 weight는 export_csv 폴더에 저장된다.

CPU 성능

```
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
```