동형암호를 활용한 딥러닝 모델 학습에 대한 연구

남기빈*, 조명현*, 김현준*, 백윤흥*
*서울대학교 전기정보공학부. 반도체공동연구소
{kvnam, hjkim, mhcho}@sor.snu.ac.kr, ypaek@snu.ac.kr

Realization of Homomorphic Encrypted Deep Learning Models

Kevin Nam*, , Myunghyun Cho*, Hyunjun Kim*, and Yunheung Paek*
*Dept. of Electrical and Computer Enginering and Inter-University Semiconductor Research Center(ISRC), Seoul National University

요 약

동형암호를 활용한 딥러닝 시도가 꾸준히 이루어지고 있다. 딥러닝 모델에는 비선형함수가 활용되고 연산량이 점점 많아지는 추세지만, 이러한 점들은 동형암호 연산의 대표적인 제한사항들이다. 이러한 제한점들을 극복할 수 있는 방안들을 소개하며 그 근거를 간단한 실험들을 통해 증명하여 동형암호 딥러닝 모델 설계를 위한 가이드라인을 제공한다.

1. 서론

효율적인 대용량 데이터 처리를 위한 클라우드 컴퓨팅이 널리 활용되고 있는 현재 개인 정보 보호에 대한 관심은 더욱 높아지고 있다. 일반적으로 암호를 통해 데이터를 보호할 수 있지만, AI 모델을 활용하는 등 연산에 데이터를 쓰기 위해서는 암호를 제거하여야 하며, 데이터를 내부자 위협과 같은 위험에 노출시키게 된다.

동형암호는 암호화 전 연산과 암호화 상태로 연산한 결과가 복호화시 같다는 동형성 특징을 지니고 있다. 사용자는 암호화한 데이터를 서버에 전송하고 연산 결과를 돌려받아, 본인이 해독하여 값을 확인할 수 있으며, 다른 암호들과 다르게 연산 과정 중 데이터가 외부에 노출되지 않아 개인 정보가 완전히 보호될 수 있다.

이러한 동형암호도 많은 제한점들을 지니고 있다. 기본적으로 OR, AND 연산(덧셈, 곱셈)이 가능하지만, 비교 연산이 가능하지 않기 때문에, 비선형 함수를 완전히 구현할 방법이 없다. 비선형성이 중요한 AI 모델과 같은 사례의 경우 동형암호를 활용하는데 있어 많은 제한을 받게 된다.

이 논문은 동형암호 체계에서 딥러닝 모델의 inference를 효율적으로 진행할 수 있는 방법을 서술하고 가이드라인을 제시할 것이다.

2. 동형암호 연산

2.1 연산의 종류와 소요 시간

동형암호의 가장 큰 문제점은 느린 속도이다. 아래 <표 1>은 HEAAN[1]을 이용한 CKKS 연산 시간을 분석해 놓은 결과이다.

<표 1> CKKS 연산 시간 분석

power of 2 연산	N= 2^16, Q=800
ciphertext size(bit)	104,857,600
encrypt (ms)(%)	700 (1.69%)
mult(ms)(%)	1,261 (3.05%)
bootstrap(ms)(%)	39,231.37 (94.89%)
decrypt(ms)(%)	150 (0.36%)
security bits	128
multiplicative depth	27

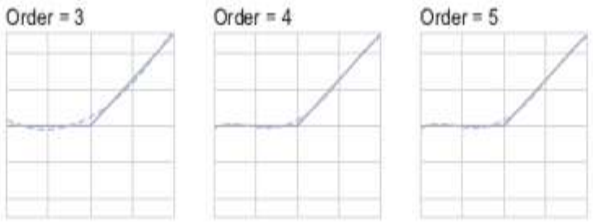
암호화 및 복호화 과정 자체는 큰 비중을 차지하지 않는다. 사용자 PC에서 진행되어야 하는 제한점이 문제가 될 수 있겠지만 실행 과정이 복잡하지 않아 해당 부분은 문제가 되지 않는다. 문제는 bootstrap이라는 과정과 multiplicative depth라는 수치다. 이 둘이 동형암호 연사에 있어 가장 큰 bottleneck을 형성하고, 비선형함수 구현에 있어서도 큰 장애물이 된다.

2.2 Bootstrap과 Multiplicative Depth

동형암호는 RLWE 문제를 기반으로 하고 있으며 noise를 첨가하여 security를 보장하는 암호이다. 이 noise는 암호문 간 연산을 통해 증폭되는데, 복호화가 불가능해지기 직전까지 곱할 수 있는 횟수를 Multiplicative Depth라고 한다. 이 Depth를 고려하여 연산량을 조절하거나, 이러한 조치가 불가능할 경우 Bootstrapping이라는 연산을 통해 noise를 줄여주어야 한다. 하지만 이 Bootstrapping은 매우 많은 연산량을 필요로 하여 bottleneck이 될 수 있다.

2.3 Non-Linear Functions의 근사

AI 모델과 같은 현대 많이 활용하는 기술들에는 Non-Linear Functions, 즉 비선형함수들을 많이 활용한다. ReLu, sigmoid등과 같은 함수가 예시이다. 아래 그림은 ReLu함수를 구현하기 위한 다항식 근사형태 예시이다.



(그림 1) ReLu 다항식 근사 과정

다항식의 차수가 높아질수록, 기존 비선형 함수에 가까워진다. 하지만 차수가 높아진다는 말은, 하나의 변수에 대한 곱셈이 많아진다는 뜻, 즉 연산량이 증가하기 때문에 Multiplicative Depth와 관한 문제, 즉 Bootstrapping의 횟수가 늘어날 수 있다[2]. 즉 연산량 증가와 모델 정확도 간 trade-off를 고려하여 적절한 근사식을 선택해야 한다.

다음으로 pooling layer에서 주로 쓰는 Max/Min 역시, 연산이 불가능하기 때문에 average pooling으로 대체하여 실행하거나 random selection을 진행하기도 한다. 전자의 경우 연산량이 늘어나며, 후자의 경우 모델의 정확도가 부정확해진다.

3. 동형암호를 활용한 딥러닝 이전 연구

3.1 연산량을 줄이는 연구

연산량을 줄이기 위한 다양한 시도들이 존재한다. Cryptonet[3]의 경우 불필요한 연산들을 줄이고 한번에 연산을 진행하고자 행렬 형태로 연산들을 정리한 후 한번에 처리하는 체계를 고안하였다. 비선형 함수들은 모두 제곱 다항식 형태로 전환하여 실행하였다. HCNN[4]의 경우 pooling 대신 average pooling을 활용했으며, 정수형 동형암호를 활용하는 대신 scaling factor를 별도 저장하여 memory overhead가 증가한 케이스로 자리잡았다.

Faster Cryptonets[5]은 Activation Function을 근사하면서 최적의 근사식을 구현했다고 주장한다. 하지만 이들이 제시한 근사식 후보군이 2차식에 계수들이 2의 정수 차수들임을 고려하면, 유의미한 최적화인지에 대한 재고가 필요하다.

3.2 Quantization

지나치게 증가하는 데이터량을 최대한 줄이기 위해 많은 연구들의 시도가 있었다. 이들은 [6]와 같은 논문들에 언급된 quantization method를 활용하여 모델 데이터를 구성하였다. 하지만 quantization에 초점을 맞춘 연구들의 경우, 이전과 성능비교를 하여 정당성을 주장하였다. 즉 accuracy하락이 일어나지만 속도가 빨라짐을 강조하였는데, 암호화하기 위한 quantization의 경우 정확도도 떨어지며 속도도 떨어지기 때문에 정당화하기 어려운 조치다.

4. 딥러닝 모델 구현 과정

이전 연구 결과들을 통해, 동형암호 체계에서 딥러닝 모델을 효율적으로 구현하기 위한 접근들을 정리해볼 수 있으며, 이점들을 고려하여 실제 반영 가능한 점인지 시뮬레이션을 통해 확인해볼 것이다.

4.1 연산량 감소

Multiplicative Depth가 큰 scheme을 활용하면 해결될 문제라곳 생각하기 쉽지만, 이는 당장 memory overhead와 직결되기 때문에 쉽게 정하기 어렵다. Depth가 크더라도 절대적인 곱셈 숫자가 많은 것 자체가 큰 computing overhead를 만들어내기 때문에, 연산량을 줄이는 것은 필수적이다.

4.2 memory overhead 감소

작은 parameter를 활용하면 해결될 문제로 생각하기 쉽지만, 이는 security를 보장해주지 못할 수도 있다는 점을 고려해야 한다. 즉 각 scheme들이 제

공하는 정해진 파라미터를 쓰는 것이 바람직하다.

또한 작은 parameter를 활용할 경우 Multiplicative Depth가 줄어든다. 아래 <표 2>는 다양한 parameter들에 대한 Multiplicative Depth변화와 security bits 분석이다.

<표 2> HEAAN CKKS Parameter Sets

logN / logQ	M. Depth	Security Bits
14 / 800	27	59
16 / 800	27	128
14 / 600	22	80

N은 다항식의 차수, logQ는 coefficient의 비트 수이다. 분석 결과를 살펴보면 logQ에 따라 Depth가 결정되는 형태지만, N이 작아지면 Security Bits도 줄어든다. 이에 맞게 logQ도 줄이면 Security Bits는 다시 증가하지만 Depth가 줄어드는 결과가 나온다.

Security bits 분석은 단순 연산으로 도출되는 것이 아닌 맞춤형 parameter set를 구하는 과정이다. 즉 이 분석 결과 나온 set는 가능하다면 그대로 사용하는 것이 좋다.

4.3 ciphertext-plaintext 연산의 활용

많은 동형암호 scheme들이 암호문 간 연산이 아닌 암호문과 평문간의 연산도 지원해준다. 이는 noise 증가를 줄이고 활용할 수 있는 bits수를 늘려주기 때문에 이를 활용해 볼 수 있을 것이다. <표 3>는 SEAL을 활용한 암호문간 연산과 암호문과 평문간의 연산을 비교한 시뮬레이션이다.

<표 3> 암호문-암호문 / 암호문-평문 연산 비교

곱셈 횟수	암호문-암호문	암호문-평문
0	338 bits left	338 bits left
1	283 bits left	305 bits left
2	245 bits left	285 bits left
3	211 bits left	262 bits left

<표 2>의 결과와 같이 암호문-평문 연산이 더 효율적인 이유는 암호문 연산 간 발생하는 relinearization과 rescaling이 발생하여 가용한 비트수가 줄어들기 때문이다. 보호해야 하는 것은 사용자의 개인정보이기 때문에, 개인정보를 암호화하고 모델 정보를 평문으로 제공하는 과정을 생각해 볼 수 있다. 클라우드, 즉 연산이 이루어지는 장소가 사용자가 아닌 모델 제공자의 소유이기 때문에 굳이 모델 정보를 암호화할 필요가 없기 때문이다.

4.4 자원 재활용

현재 동형암호 체계를 지원하는 SW 라이브러리들은 C, python, Cuda 기반인 경우들이 대부분이다. 그 구현들을 살펴보면 자원 재활용 관하여 최적화가 아직 이루어지지 않은 부분들이 많다. 예를 들어, 암호문 두 개를 입력으로 받아 하나의 결과문을 출력하는 함수의 경우, 입력과 결과문을 동일하게 활용하면 보다 효율적인 자원 활용이 가능할 수 있다.

5. 결론

이전 연구들을 분석하고 새로운 실험들을 한 결과, 동형암호를 활용하여 Deep Learning 모델을 구현하기 위한 필수 요소들은 다음과 같이 요약해 볼 수 있다. 우선 연산량을 줄이기 위한 새로운 접근법을 고안해야 한다. [3]와 같은 접근 뿐 아니라 모델 맞춤형으로 가능한 모든 방안을 활용해서 최대한 줄여야 한다. 다음으로 parameter를 줄여보려고 시도할 수 있는데 이는 security에 영향을 줄 수 있어 지양하는 것이 좋다. 연산은 암호문간의 연산보다는 암호문-평문 연산을 활용하는 것이 좋다. 가용한 비트수의 감소량을 줄이고 불필요한 연산들이 줄어들어 상대적으로 Depth가 증가하는 효과와 다름 없으며 Bootstrapping을 최소화하여 computing overhead를 줄일 수 있다.

추후 Bootstrapping 자체를 가속하는 연구, HW를 활용하여 가속하는 연구 등 다양한 접근을 통해 새로이 동형암호를 활용한 딥러닝을 가속할 수 있는 연구들이 이루어져야 할 것이다.

6. ACKNOWLEDGEMENT

이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (NRF-2020R1A2B5B03095204). 이 논문은 2021년도 BK21 FOUR 정보기술 미래인재 교육연구단에 의하여 지원되었음. 이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2018-0-00230, (IoT 총괄/1세부) IoT 디바이스 자율 신뢰보장 기술 및 글로벌 표준기반 IoT 통합보안 오픈 플랫폼 기술개발 [TrusThingz 프로젝트])

참고문헌

[1] J. Cheon, A. Kim, M. Kim, Y. Song, "Homomorphic Encryption for Arithmetic of Aproximate Numbers", ASIACRYPT, 2017

[2] N. Brisebarre et al. "Computing machine-efficient polynomial approximations", Transactions on Mathematical Software, 2006

[3] R. Gilad-Bachrach et al. "CryptoNets: Applying Neural Networks to Encrypted Data with High Throughput and Accuracy", 33rd International Conference on Machine Learning, 2016

[4]. al Badawi et al. "Towards the AlexNet Moment for Homomorphic Encryption : HCNN, the First Homomorphic CNN on Encrypted Data with GPUs, arXiv:1811.00778, 2020

[5]. Chou et al. "Faster CryptoNets: Leveraging Sparsity for Real-World Encrypted Inference", arXiv:1811.09953, 2018

[6] D. Lin et al. "Fixed Point Quantization of Deep Convolutional Networks"