빠르고 정확한 암호화된 신경망 연산을 위한 중간 데이터를 활용한 근사식 생성 기술

남기빈[1], 주유연[1], 하승진[1], 백윤흥[1]
[1]서울대학교 전기정보공학부, 서울대학교 반도체 공동연구소
{kvnam, yyjoo, sjha}@sor.snu.ac.kr, ypaek@snu.ac.kr

Intermediate Data Guided Approximation for Fast and Accurate Encrypted Neural Networks

Kevin Nam[1], Youyeon Joo[1], Seungjin Ha[1], Yunheung Paek[1]
[1]Dept. of Electrical and Computer Enginering and Inter-University Semiconductor Research Center(ISRC), Seoul National University

요 약

동형암호는 프라이버시 보존형 신경망 연산을 가능캐한다. 하지만 동형암호는 비산술연산을 직접 연산하지 못해 근사식을 활용하는데, 신경망 정확도 하락을 일으킨다. 이를 극복하기 위해 재학습, Neural Architecture Search 등 방법들이 등장했지만, 큰 소요시간을 필요로 한다. 본 연구는 이 둘보다 빠르면서도 정확도 하락을 적게 일으키는 중간값 유도 근사식 생성 기술을 제안한다.

1. 서론

동형암호는 데이터를 암호화 상태로 연산할 수 있는 암호체계로, 프라이버시 보존형 클라우드 기반 신경망 연산을 수행할 수 있다. 하지만 동형암호는 연산부하가 평문 대비 매우 큰 것 외에도 일반적으로 산술연산만 지원하여 비산술연산을 직접 지원하지 않는다는 점, 이들을 산술 근사식으로 대체 연산하며 연산오차가 발생하는데, 이는 신경망 정확도 하락으로 이어진다[1].. 여러 연구들이 동형암호 기반 신경망의 정확도를 높이기 위해 노력해왔다. 대표적으로, 보다 정확한 비산술연산의 근사식을 만들어 추론에 활용하는 방법이 있는데[1], 이는 고차식을 활용하지 않는 이상 정확도 하락이 매우 크다는 단점이 있다. 다음 방법으로는, 근사식으로 대체 후, 모델을 재학습 하는 방법인데[2], 이는 학습을 새로 하는 것만큼 많은 시간이 필요하다는 단점이 있다.

이 논문은 재학습과 같은 기존 접근들 대비 소요시간이 필요하지 않으면서 모델의 정확도를 최대한 보존하는 새로운 기술을 제안하며, 이를 기반으로 응용될 수 있는 미래 연구 방향들을 제시한다.

2. 이론적 배경

2.1 동형암호 체계와 지원 연산 종류

<표 1>은 대표적인 동형암호 체계들과 그들의 특징을 나타낸 것이다. CKKS[3]와 같은 RLWE 체계들은 한 암호문에 여러 데이터를 벡터형태로 담아 SIMD 형태로 동시에 연산할 수 있는 패킹 기능을 지원한다. 이를 사용해 단 한번의 곱셈으로 수천, 수만개의 데이터를 연산할 수 있다. 하지만, RLWE 체계들은 태생적으로 산술연산만을 지원한다. 비산술연산은 다항식 형태로 근사하여 연산한다.

종류	체계	연산	패킹
RLWE	BFV, CKKS	산술	O
LWE(GSW)	FHEW, TFHE	산술 & 비산술	X

표 1 대표적인 동형암호 체계

	추론 정확도 (%)	시간 (초)
RLWE(CKKS)	92.02 → 89.42	2,982
LWE(TFHE)	92.02 → 92.02	31,000

표 2 : VGG-16 모델을 활용한 CIFAR-10 추론 성능

TFHE[4]와 같은 LWE 체계들은 비산술 연산을 지원한다. 하지만 LWE 체계들은 패킹을 지원하지 않아 비효율적이라는 단점을 지니고 있다.

2.2 동형암호 기반 신경망 연산 (HENN)

<표 2>는 RLWE와 LWE 체계를 활용한 신경망 연산의 추론 정확도와 연산 시간을 나타낸다. RLWE 체계는 추론 정확도가 학습 정확도 대비 약 2.6% 하락함을 확인할 수 있다. LWE는 정확도 하락은 없지만, RLWE 대비 10.4배 느리다는 치명적인 단점을 지니고 있다. 이에 많은 연구들은 RLWE를 활용하여 연산 효율을 갖추면서 정확도 하락을 줄이

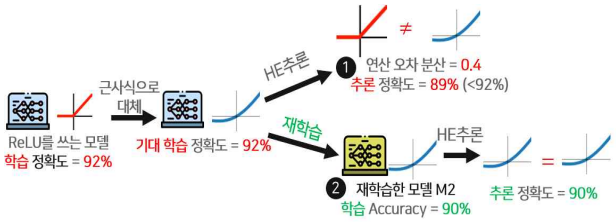

그림 1 RLWE 체계 기반 HENN 연산 방법

근사 차수	정확도 변화 (%)	시간
2	92.18 → 45.67	1.5초
>27	92.18 → 89.42	75.7초

표 3 : IR기법 기반 CIFAR-10 ResNet-20 모델 추론 (GPU)

기 위한 방법들을 모색했다.

[그림 1]은 RLWE를 활용한 HENN 연구 방향들을 보여준다. 학습된 모델의 비산술연산들인 활성화 함수(e.g., ReLU) 근사식 대체 후 추론연산하는 경우[1] 연산오차로 인한 추론 정확도 하락이 발생한다(①, 이하 Inference Replacement, IR기법이라 칭함). 반대로, 근사식 대체 후 재학습 하는 경우[2], 비선형성 부족으로 초기 모델만큼 높은 학습 정확도를 갖추지 못하지만, 추론에서 연산 오차가 발생하지 않으므로 추론 정확도 하락이 발생하지 않는다는 장점이 있다(②, 이하 Re-Training, RT기법이라 칭함). 하지만, 재학습은 초기학습만큼 매우 큰 준비시간이 필요하다는 단점이 있다.

추가적인 방법으로, Neural Architectural Search (NAS)를 활용하는 방법이 있다[5]. 이 방법은 비산술연산마다 대체 근사식 후보군에서 하나씩 적용해 보며 가장 정확도 하락이 낮게 나오는 것을 고르는 방법이다. 재학습과는 다른 방법이지만, search space가 매우 크다는 점에서 재학습만큼 큰 준비시간이 필요한 방법이다.

2.3 기존 접근들의 한계점 분석

IR기법은 소요시간이 짧지만 일반적으로 괜찮은 근사식을 고르는 것으로 **모델과 레이어별 특징 (입력값 등)**을 고려하지 않은 방법이다. 특히, Remez, Taylor와 같은 기법은 전범위가 아닌 특수 범위(e.g., -1<x<1)에 대해 적용할 수 있다는 한계가 있어, 차수를 높여 범위를 넓혀가야 오차를 줄일 수 있다. 따라서, <표 3> 결과와 같이 충분히 고차식(10차 이상)을 선택하지 않으면 추론에 있어 큰 오차를 발생시킨다. 반면 RT 기법과 NAS는 레이어별 특성을 고려하여 각 레이어마다 서로 다른 근사식을 선택할 수 있고 상대적으로 낮은 차수의 근사식을 써서 연산 부하도 줄이며 높은 정확도를 유지할 수

	정확도 변화	재학습/NAS 시간	추론 시간
재학습	92.18 → 90.08	2.1 hours	1.6초
NAS	92.18 → 91.92	44 hours	1.5초

표 4 : RT/NAS 기반 CIFAR-10 ResNet-20 시간 (GPU)

있다. 다만, 재학습, NAS 자체가 추가적인 큰 준비시간을 필요로 한다는 단점이 있다. <표 4>은 RTX A6000 GPU를 사용해서 재학습과 NAS를 수행한 실험 결과인데, 각각 2.1시간, 44시간을 소요했다.

3. 중간값 유도 근사식 생성 기술

3.1 중간값 활용의 효율성

중간값 유도 근사식 생성 기술은 위 언급한 두 접근법들의 장점을 살리고 단점을 극복하는 것을 목표로 만들어졌다. RT/NAS는 레이어별 특성을 분석하여 근사식을 생성하기 위한 과정으로 초기학습과 무관한 추가 연산부하를 많이 일으킨다. 하지만, 레이어별 특성은 초기학습과정을 통해 생성되는 것으로, 초기학습 과정의 정보를 일부 활용한다면, 간단한 방법, 즉 적은 연산부하만으로도 레이어별 특성을 충분히 살릴 수 있을 것이다.

구체적으로, 학습 여러 iteration 중 후반부는 최종 모델 decision boundary에는 민감하게 반응하지만, 각 레이어의 결과, 즉 중간 레이어들의 weight 값들은 크게 변하지 않는다. 따라서, 후반 iteration들에서 최초 입력에 따른 각 ReLU입력은 크게 변하지 않을 것이다. 예를 들어, 최소 입력값이 0에서 255더라도, 첫 convolution의 weight가 0.01이라면, 첫 ReLU의 입력은 0에서 2.55로 그 폭이 매우 좁혀져서 저차식으로도 충분히 정확한 근사식을 만들기 유리해진다. 우리는 이런 신경망 중간값, 즉 각 레이어의 연산 결과를 metadata로 활용해서, 중간값들의 분포를 활용하는 방안을 모색했다.

3.2 K-Means를 활용한 중간값 분포 분석

중간값을 활용하더라도, NAS처럼 넓은 search space를 모두 탐색하는 것은 비효율적일 것이다. 이에 우리는 가장 단순한, 주어진 d개의 점이 있을 때, 이들을 지나는 d-1 차식을 확정적으로 정의할 수 있다는 점을 활용했다. 예를 들어, 근사식의 목표 차수가 2일 때, 3개의 점을, $y = ax^2 + bx + c$에 대입해서 근사식을 구할 수 있고, 이는 단 한번의 3x3 역행렬 곱으로 이루어진다.

이 방법을 사용하기 위해서는 수없이 많은 데이터 중 d개의 점을 우선 선택해야 하는데, 후보군이 학습데이터셋 크기, 즉 수만개가 된다는 점이다. 우리

기법	time complexity
SVM	$O(n^2)$ or $O(n^3)$
Hierarchical Clustering	$O(n^2)$ or $O(n^2 \log n)$
K-Means	$O(n \times k \times i \times m)$

표 4 : Clustering 시간 복잡성 비교 (본문에 요소 설명)

약자	Dataset	Model	학습 정확도	학습 시간
ML5	MNIST	Lenet-5	94%	30초
CR20	CIFAR-10	ResNet-20	92.18	2.1시간
CR32	CIFAR-10	ResNet-32	93.41	3.8시간

표 5 : 실험에 활용한 모델들과 그들의 약자

는 이 과정을 위해 여러 데이터를 k개의 군집으로 분류해주는 K-means 기법을 활용했다. 이를 활용하면 거리를 기준으로 유사한 데이터들을 k 개의 군집으로 묶어, 각 군집 내 평균값으로부터 분산이 최소인 묶음을 구해주게 된다. 따라서, d개의 군집으로 나눈 후, 각 군집의 평균에 해당하는 점들을 지나는 d-1차식을 만들어서, 근사식으로 활용하면, 해당 레이어를 통과하는 값들이 충분히 고려된 근사식을 생성했다고 할 수 있다.

수행 시간 측면에서, K-means는 큰 이점이 있는데, 바로 time complexity가 상대적으로 낮기 때문이다. n, k, i, m는 각각 데이터 수, 군집 수, 반복 횟수, 그리고 차원을 나타낸다. 우리의 경우 k는 다항식의 차수+1, 그리고 m은 x, y를 다루기에 2로 고정되지만, n은 training set 크기로 언급했듯 수만까지 크기가 커질 수 있다. n이 지배하는 상황에서, 다른 기법들보다 낮은 차수의 n을 활용하기에 K-means가 효율적이라 할 수 있다. 또한, K-means가 매우 간단한 머신러닝 기법이라는 점에서 RT/NAS 대비 빠를 것이라 직관적으로 생각할 수 있는 반면, 이들의 복잡도는 활용하는 metric이 다르기 때문에 이론적으로 비교하기 어렵다. 이에 실험적으로 우리 방법의 효율성을 보이고자 한다.

4. 실험 결과

성능 검증을 위해, 우리는 Intel Xeon Gold 6326와 DRAM 1TB, 그리고 RTX A6000 GPU가 탑재된 서버에서 GPU를 사용해 실험을 수행하였다. IR, RT, 그리고 NAS 비교군으로 최신 연구들인 [1], [2], 그리고 [5]를 활용했다. <표 5>는 본 실험에서 활용한 모델들이다. 추후 편의상 이표의 약자들을 활용해서 이 모델들을 언급하겠다. 각 dataset 별 학습에 활용한 batch, epoch 등은 Keras 표준 문서를 따랐다[6].

<표 6>은 기술별 정확도와 하락폭을 표기한 것이고, <표 7>은 우리 기술과 RT/NAS와의 근사식 생

정확도 (하락폭)	IR	RT	NAS	Ours(2차)
ML5	93.93 (-0.07)	93.92 (-0.08)	93.94 (-0.06)	93.94 (-0.06)
CR20	89.42 (-2.76)	90.08 (-2.1)	91.92 (-0.26)	91.88 (-0.3)
CR32	87.61 (-5.8)	88.39 (-5.02)	93.16 -(0.25)	92.42 (-0.99)

표 6 : 기술별 정확도 비교

	RT	NAS	Ours (2차식)
ML5	30 seconds	7 hours	21 seconds
CR20	2.1 hours	44 hours	0.66 hours
CR32	3.8 hours	64 hours	0.84 hours

표 7 : 기술별 추가되는 준비 시간 비교

성 (재학습, 서치) 시간을 비교한 결과이다. RT의 경우, 재학습인만큼 <표 5>의 학습 시간만큼 추가 시간이 필요했고, 정확도 하락은 최대 5.02%까지 발생했다. NAS의 경우 모든 경우의 수를 다 탐색하는 만큼 훨씬 시간이 오래 걸렸지만, 정확도 하락이 RT대비 적게 발생했다 (최대 0.26%). 2차식을 활용한 우리 기술의 경우, 정확도 하락은 최대 0.99%로 NAS보다 크지만 RT보다 적게 발생했다. 하지만 소요시간의 경우 NAS에 비해 57.3-76.1배 적게 걸렸으며, RT보다 1.3-4.5배 적게 걸렸다. 우리 기술이 매우 적은 소요시간만으로 적절한 정확도 하락을 얻을 수 있음을 확인했다.

sec.	IR	RT	NAS	Ours 2차
ML5	0.81	0.33	0.84	0.33
CR20	75.7	1.46	143.2	1.41
CR32	124.3	2.17	228.7	2.21

표 8 : 기술별 추론 시간 비교 (GPU)

<표 8>은 기술별 추론 시간 비교이다. <표 6>과 함께 살펴보면, IR 대비 RT가 빠르면서도, 정확도 하락이 적은 것을 확인할 수 있다. 보다 더 정확도 하락을 줄이기 위해 NAS를 사용할 수 있지만, 추론 시간이 IR보다도 커지는 것을 확인할 수 있다. 우리의 기술을 활용할 경우, 거의 RT와 유사한 수행시간을 유지하면서도 RT보다 매우 적은 정확도 하락을 유발함을 확인할 수 있다.

위 실험 결과들을 통해 우리의 기술이 기존 기술들보다 시간적으로 훨씬 효율적임에도 정확도 하락은 가장 좋은 NAS에 유사한 수준으로 만들어주는 것을 확인할 수 있었다.

5. 토론 및 고찰

실험을 통해 우리 기술이 준비 시간과 추론 시간 측면에서 효율적임을 확인할 수 있다. 정확도 하락의 경우, IR, RT보다 적으며, NAS와 비교했을 시에는 최대 4배까지 크게 발생한다. 하지만 결과적으로

그림 2 기술별 실제 수행 최적화 방법

정확도 하락이 1% 이내로 유지되는 것을 확인할 수 있으며, 특히 4개가 발생하는 CR32의 경우 NAS가 준비 시간으로 64시간이 걸리는 대비 우리 기술이 0.84 시간이 걸리는 것을 고려하면 우리 기술이 효율적임을 확인할 수 있다.

한편, 실험에서 우리 기술은 2차식 탐색만을 활용했다. 이미 IR과 NAS는 고차식을 사용하는데, 우리 기술도 차수를 높이면 대체적으로 정확도 하락이 감소할 것이다. 여기서 우리 기술의 장점은, 지수차승으로 탐색범위가 증가하는 NAS와 다르게, 선형적으로 준비시간이 늘어난다는 점이다. 즉, 준비시간 증가폭이 적게 늘어날 것이고, 차수를 높여도 효율적일 것이다. 하지만 우리 기술의 한계는 어떤 차수를 사용할 것인지 사전에 선택해서 적용해야 한다는 점이다. 즉, 다른 차수를 활용하려면 반복적으로 사용해야하기에, n차와 n+1차식을 모두 고려하려면 실제로는 선형적으로 늘어나는 것보다 더 오래 걸릴 것이라는 점이다. 이런 특징과 관하여, 미래에 우리 기술을 기반으로 하여 효율적으로 다양한 차수에 대한 탐색 연구가 수행되었으면 한다.

[그림 2]와 같이, 기술별로 추가 최적화 방법들을 고려해볼 수 있다. 예를 들어 RT를 사용하지만 처음부터 재학습하는 것이 아니라 transfer learning[7] 등 방법을 활용하여 RT의 준비 시간을 줄일 수 있다는 점을 생각해볼 수 있다. 우리 기술은 초기학습 과정에서 발생하는 중간값들을 사용한다는 점에서, 초기 학습과 동시에 진행할 수 있는데, 이 때 학습 중 idle 한 상태의 thread (GPU의 경우 벡터 프로세서 라인)에 스캐줄링함으로써 후에 별도로 진행하는 시간보다 수행시간을 줄이는 방법을 생각해볼 수 있다. 또한, 학습과정을 근사식 탐색 기술에 맞추어 변화를 꾀할 수도 있을 것이다. 본 연구는 일종의 표준 학습 방법을 따르기 위해 Keras 문서[6]에 나온 학습코드 예시를 따랐으나, 재학습, 그리고 우리 기술의 경우 학습 과정을 수정하여 각 기술에 유리하게 접근해볼 수 있을 여지가 남아있다. 이런 접근 역시 미래 연구 주제 방향으로 남긴다.

6. 결론

이 논문은 동형암호 기반 신경망 연산의 정확도 하락에 대응하기 위해 중간값 유도 근사식 생성 기술을 소개한다. 실험 결과 재학습과 NAS 대비 준비 시간이 크게 줄어들었으며, 정확도 하락도 재학습 대비 매우 작은 등 경쟁력있는 수준으로 유지한다. 이 기술을 기반으로 다양한 차수의 근사식에 적용하는 방법 및 HW 연산 최적화 등 다양한 미래 연구들이 촉진되었으면 한다.

6. ACKNOWLEDGEMENT

이 논문은 연구 수행에 있어 2024년도 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원과 (RS-2023-00277326) 정보통신기획평가원의 지원을 받아 수행된 연구이며 (No.2021-0-00528, No.RS-2023-00277060, IITP-2023-RS-2023-00256081) 2024년도 BK21 FOUR 정보기술 미래인재 교육연구단, 반도체 공동연구소 지원의 결과물이다. 또한, 연구장비를 지원하고 공간을 제공한 서울대학교 컴퓨터연구소에 감사드린다.

참고문헌

[1] Lee, Eunsang, et al. "Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions." International Conference on Machine Learning. PMLR, 2022.

[2] Kim, Donghwan, et al. "HyPHEN: A Hybrid Packing Method and Its Optimizations for Homomorphic Encryption-Based Neural Networks." IEEE Access (2023).

[3] CHEON, Jung Hee, et.al. “Homomorphic encryption for arithmetic of approximate numbers”, ASIACRYPT 2017, Hong Kong, China, December 3-7, 2017, p. 409-437.

[4] CHILLOTTI, et.al. “TFHE: fast fully homomorphic encryption over the torus”, Journal of Cryptology, 2020, 33.1: 34-91.

[5] Ao, Wei, and Vishnu Naresh Boddeti. "{AutoFHE}: Automated Adaption of {CNNs} for Efficient Evaluation over {FHE}." 33rd USENIX Security Symposium (USENIX Security 24). 2024.

[6] François Chollet. Keras. https://keras.io, 2015.

[7] Weiss, Karl, Taghi M. Khoshgoftaar, and DingDing Wang. "A survey of transfer learning." Journal of Big data 3 (2016): 1-40.