고비용 활성화 함수의 효율적 GPU 추론을 위한 사전 계산 및 룩업 테이블 최적화

김재민[1], 김성균[2], 서지원[3]
[1]한양대학교 인공지능학과 박사과정
[2]한양대학교 컴퓨터소프트웨어학과 박사과정
[3]한양대학교 컴퓨터소프트웨어학과 교수

woals174@hanyang.ac.kr, cheezestick@hanyang.ac.kr, seojiwon@hanyang.ac.kr

Lookup Table Optimization for Efficient GPU Inference of High-Cost Activation Functions

Jaemin Kim, Sungkyun Kim[2], Jiwon Seo[2]
[1]Dept. of Artificial Intelligence, Hanyang University
[2]Dept. of Computer Science, Hanyang University

요 약

본 연구에서는 대규모 언어 모델(LLM)에서 GeLU와 SiLU 활성화 함수의 높은 연산 비용을 해결하기 위해 룩업 테이블(LUT) 기반 최적화 기법을 제안하였다. BERT, GPT2, OLMo 모델을 대상으로 실험을 수행하였으며, 특히 OpenAI GeLU를 사용하는 GPT2 모델에서 최대 9배의 성능 개선을 확인하였다. 또한, 배치 크기 변화에 따른 성능 분석 결과, GPT2는 배치 크기가 클수록 더 큰 성능 향상을 보였고, BERT와 OLMo는 상대적으로 낮은 개선율을 나타냈다. 최적화 기법을 통해 각 활성화 함수의 연산 시간을 크게 줄이면서도 오차율을 낮게 유지할 수 있었다.

1. 서론

딥러닝 모델, 특히 GPT와 같은 대규모 언어 모델(LLM)은 최근 다양한 분야에서 놀라운 성과를 보이며 그 중요성이 커지고 있다. 이러한 성과는 수십억 개 이상의 매개변수를 포함한 복잡한 신경망 구조 덕분에 가능하며, LLM은 자연어 처리, 기계 번역, 챗봇 등에서 뛰어난 성능을 발휘한다. 하지만, 이들 모델은 공통적으로 높은 연산 비용을 요구하는 비선형 활성화 함수를 포함하고 있다는 문제를 가지고 있다.

활성화 함수는 입력 신호를 비선형적으로 변환하여 모델이 더 복잡한 패턴을 학습할 수 있게 하지만, 이로 인해 연산 비용이 크게 증가한다. 특히 LLM과 같은 대규모 모델의 경우 활성화 함수의 연산 효율성은 전체 모델의 성능과 처리 속도에 직접적인 영향을 미친다. 이러한 모델에서 자주 사용되는 활성화 함수로는 GeLU(Gaussian Error Linear Unit)와 SiLU(Sigmoid Linear Unit) 등이 있으며, 이들은 복잡한 수학적 연산을 포함하여 높은 연산 비용을 초래한다.

특히, GeLU 함수는 Gaussian 누적 분포 함수(CDF)를 기반으로 동작하며, 오차 함수(erf)를 계산하는 과정에서 높은 연산 자원을 소모한다. CDF는 입력 값을 정규 분포에 맞춰 변환하는 함수로, 이를 통해 GeLU는 더 부드러운 비선형성을 제공한다.

OpenAI 에서는 더욱 매끄러운 출력을 얻기 위해 Taylor 급수를 활용한 근사 연산을 추가하여 GeLU를 수정하였다. 이러한 근사 방식은 기존의 GeLU보다 정밀한 비선형성을 제공하지만, tanh 함수등 추가적인 연산이 포함되어 연산 비용이 증가하게 된다. 특히 대규모 모델에서 이러한 비선형 활성화 함수는 GPU 자원을 많이 소모하여 성능 병목을 초래할 수 있다.

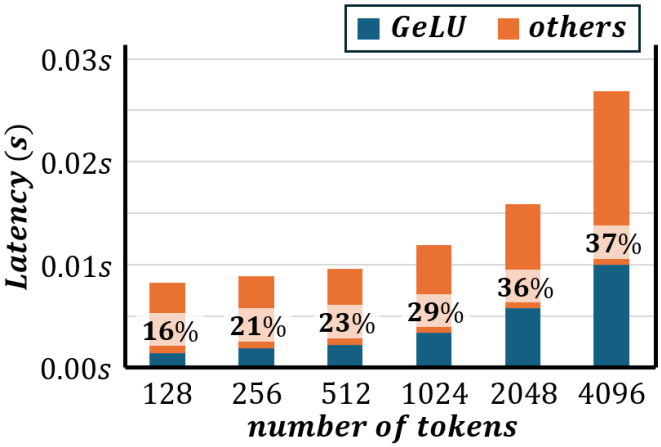
본 논문에서는 이러한 성능 병목 문제를 해결하기 위해 룩업 테이블(LUT)을 이용한 최적화 방법을 제안한다. 이를 통해 복잡한 활성화 함수의 연산을 단순화하고, 대규모 모델의 연산 효율성을 크게 향상시킬 수

있음을 보인다.

2. 배경

2.1 고비용 활성화 함수

신경망의 핵심 요소인 활성화 함수는 모델에 비선형성을 도입하여 각 노드 간의 복잡한 관계를 학습할 수 있도록 하지만, 모든 활성화 함수가 동일한 계산 비용을 요구하는 것은 아니다. 지수 함수, 로그 함수, 시그모이드(sigmoid), 쌍곡탄젠트(tanh)와 같은 함수들은 '고비용 활성화 함수'로 분류되며, 그 계산적 복잡성으로 인해 높은 연산 비용을 초래한다.



(그림 1) GPT-2 모델의 인코딩 연산에서 토큰 수에 따른 총 연산 시간과 GeLU 함수가 차지하는 시간을 나타낸 그래프. 각 막대의 숫자는 총 지연 시간 대비 GeLU 연산의 비율을 의미하며, 토큰 수가 증가함에 따라 GeLU 함수가 차지하는 비중이 어떻게 변화하는지 시각적으로 보여준다.

고비용 활성화 함수들은 ReLU(Rectified Linear Unit)와 같은 단순한 함수와는 달리 복잡한 수학적 연산이 요구된다. 예를 들어, 시그모이드 함수는 지수 함수의 계산을 포함하고, 소프트맥스 함수는 로그 함수 연산을 포함하는데, 이러한 연산은 반복적 알고리즘이나 무한 급수를 통해 구현되기 때문에 상당한 계산 비용을 발생시킨다. 대규모 딥러닝 모델에서는 이와 같은 연산이 수백만 번 반복되어 전체 모델의 성능에 큰 영향을 미친다. 특히 GPT-2 모델의 인코딩 연산에서 1024 개의 토큰을 처리할 경우, (그림 1)과 같이 전체 연산 시간의 29%를 고비용 활성화 함수가 차지하는 것으로 나타났다.

고비용 활성화 함수는 복잡한 계산을 요구하기 때문에 다양한 근사 기법이 사용되지만, 여전히 다 연산이 필요하며 성능 저하 또한 초래할 수 있다. 이에 따라 이러한 함수들의 최적화와 효율적인 근사 기법이 중요한 연구 주제로 떠오르고 있다. 본 연구에서 제안한 룩업 테이블 기법은 복잡한 함수 값을 사전에 계산하여 저장하고, 필요 시 빠르게 참조함으로써 실시간 연산 비용을 크게 줄일 수 있는 가능성을 보여준다.

2.2 LLM 에서 자주 사용하는 고비용 활성화 함수

GeLU (Gaussian Error Linear Unit) [1]:

$$GeLU(x) = x \cdot \frac{1}{2}\left(1 + erf\left(\frac{x}{\sqrt{2}}\right)\right)$$

GeLU 는 입력값이 큰 경우 ReLU 와 유사하게 동작하지만, 작은 음수 입력에 대해 부드러운 곡선을 그리는 특징을 가진다. 이는 모델이 더 부드럽고 연속적인 출력을 생성할 수 있게 해준다. 또한, GeLU 는 확률적 특성을 가지고 있어, 입력값의 중요도를 확률적으로 해석할 수 있다는 장점이 있다. 이러한 특성으로 인해 BERT, GPT 등 많은 최신 LLM 에서 기본 활성화 함수로 채택되고 있다. 그러나 GeLU 는 Gaussian 오차 함수의 계산으로 인해 상대적으로 높은 계산 비용을 요구하며, 이는 대규모 모델에서 중요한 고려 사항이 된다.

OpenAI 는 기존 GeLU 함수의 부드러움을 강화하기 위해 수정된 근사 방식을 도입했다. 이 새로운 GeLU 는 다음과 같은 수식을 따른다:

$$GeLU_{new}(x) = 0x \cdot \left(1 + tanh\left(\sqrt{\frac{2}{\pi}}(x + 0.044715 \cdot x^3)\right)\right)$$

이 방식은 기존 GeLU 보다 더 부드러운 출력을 제공하나, tanh 와 추가 연산으로 인해 계산 비용이 더욱 증가한다. 특히, 대규모 모델에서는 성능 병목이 될 수 있어, 이를 최적화하는 방법이 중요한 과제가 된다.

SiLU (Sigmoid Linear Unit) [2]:

$$SiLU(x) = x \cdot \left(\frac{1}{1 + e^{-x}}\right)$$

SiLU 는 양의 입력에 대해서는 ReLU 와 유사하게 동작하지만, 음수 입력에 대해서는 부드러운 곡선을 그리며 비단조적(non-monotonic) 특성을 가지고 있다. 이로 인해 특정 구간에서 입력값의 복잡한 패턴을 학습하는 데 유리하다. Google 의 연구에서는 SiLU 가 다양한 신경망 구조에서 우수한 성능을 보이는 것으로 확인되었다[참고 문헌 필요]. 그러나 SiLU 역시 시그모이드 함수의 계산 때문에 높은 연산 비용이 발생하여 모델의 효율성에 영향을 미칠 수 있다.

2.3 GPU에서의 고비용 활성화 함수 최적화

고비용 활성화 함수의 계산 효율성 문제는 GPU(Graphics Processing Unit)를 사용한 병렬 처리 환경에서 더욱 중요하다. GPU는 딥러닝 모델의 처리 속도를 크게 향상시킬 수 있지만, 복잡한 활성화 함수는 이러한 이점을 상쇄시킬 수 있다.

GPU에서의 고비용 활성화 함수 최적화는 주로 다음과 같은 방법을 통해 이루어진다:

1. **근사 함수 사용:** 복잡한 활성화 함수를 더 단순한 형태로 근사하여 계산 속도를 향상시킨다.
2. **룩업 테이블 최적화:** GPU 메모리의 특성을 고려한 효율적인 룩업 테이블을 구현하여 빠른 연산을 가능하게 한다.
3. **커스텀 CUDA 커널 구현:** GPU의 특성에 맞는 최적화된 병렬 처리 알고리즘을 개발한다.

이러한 최적화 기법들은 고비용 활성화 함수의 연산 효율성을 크게 향상시킬 수 있지만, 모델 구조, 데이터셋, 하드웨어 환경에 따라 효과가 다르게 나타날 수 있다. 실제 적용 시에는 세심한 벤치마킹과 분석이 필요하며, 모델의 정확도와 속도 간의 균형을 고려해야 한다. 본 연구는 GeLU와 SiLU를 사용하는 LLM 모델에 대해 이러한 최적화 기법을 적용하고 분석한다.

3. 방법론

GeLU 함수는 LLM 에서 중요한 역할을 하지만, Gaussian 누적 분포 함수(CDF)와 OpenAI 의 수정된 GeLU에서 추가된 tanh와 고차항 계산으로 인해 연산 비용이 매우 크다. 이러한 연산이 모델의 모든 계층에서 반복되면서 성능 병목이 발생한다. 본 연구에서는 이러한 비효율적인 연산 문제를 해결하기 위해, 활성화 함수의 값을 미리 계산하고 저장한 룩업 테이블을 활용하는 최적화 방법을 제안한다. 이 방식은 고비용 연산을 실시간으로 수행하는 대신, 미리 계산된 값을 빠르게 조회하여 성능을 크게 향상시킨다. CUDA 기반 shared memory를 이용해 병렬 연산을 최적화함으로써 GPU 상에서의 효율성도 극대화하였다.

3.1 룩업 테이블 기반 최적화

룩업 테이블을 통해 활성화 함수의 고비용 연산을 대체하여 성능을 최적화한다. 단순한 Min-Max 통계를 기반으로 테이블을 생성하는 것만으로는 충분하지 않기 때문에, 두 가지 추가 최적화 기법을 도입하였다.

MIN 및 MAX 값 최적화: 각 활성화 함수의 특성에 따라, 출력 값의 차이가 최소가 되는 최적의 MIN 값과 MAX 값을 찾는다. 이를 위해, 입력 데이터의 분포를 분석한 후, 해당 범위 내에서 출력 값이 안정적으로 변화하는 구간을 선택한다. 이 범위를 벗어나는 입력 값은 MIN 이하일 경우 0으로, MAX 초과 시 y=x로 처리하여 비효율적인 연산을 줄인다.

Calibration 데이터 기반 조정: 실험 데이터에서 추출한 통계 자료를 바탕으로 MIN과 MAX 값을 보정한다. 이는 실제 데이터 분포에 맞는 최적화된 룩업 테이블을 생성하여 더욱 정밀한 추론을 가능하게 한다. 이 과정에서 다양한 데이터셋을 사용하여, 일반화된 성능을 확보한다.

3.2 CUDA 기반 최적화

GPU 상에서의 효율성을 극대화하기 위해, 본 연구는 CUDA를 사용하여 룩업 테이블 기반 최적화를 구현하였다. 특히, shared memory를 활용하여 전역 메모리 접근을 최소화하고, 활성화 함수 계산을 빠르게 처리하도록 설계하였다. Shared Memory는 전역 메모리보다 접근 속도가 빠르며, 여러 스레드가 데이터를 동시에 사용할 수 있어 병렬 처리를 더욱 효율적으로 할 수 있다.

CUDA 커널은 각 스레드 블록에서 활성화 함수의 룩업 테이블을 Shared Memory에 로드 한 후, 이를 바탕으로 입력 값에 대한 조회를 수행한다. 이 방식은 전역 메모리 접근에 따른 성능 병목을 완화하고, 여러 스레드가 동일한 데이터를 빠르게 공유할 수 있도록 한다. 또한, 입력 값이 정확히 테이블 내에 존재하지 않은 경우, 인접한 구간의 값을 선형 보간하여 근사값을 계산함으로써 정밀하게 값을 표현하였다.

결과적으로, CUDA 기반의 병렬화와 Shared Memory 최적화를 통해 GeLU 함수의 연산 성능이 기존 방식에 비해 약 8배 이상 향상되었으며, 이는 대규모 모델에서 더욱 두드러진 성능 개선을 가능하게 한다.

4. 실험 및 결과

본 실험에서는 다양한 딥러닝 모델을 사용하여 룩업 테이블 기반 활성화 함수 최적화 기법의 성능을 평가하였다. 실험에는 BERT[3], GPT2[4], OLMo[5] 총 세개의 모델을 사용하였다. BERT 모델은 트랜스포머 인코더 기반의 모델로, GeLU 활성화 함수를 사용하는 모델이다. GPT2와 OLMo 모델은 트랜스포머 디코더 기반의 모델로, 각각 OpenAI GeLU, SiLU 활성화 함수를 사용한다.

실험은 Ampere 아키텍처 기반의 NVIDIA A6000 GPU에서 수행되었으며, 이에는 48GB GDDR6 메모리와 10,752개의 CUDA 코어가 탑재되어 있다. CUDA 12.1 버전을 사용하였으며, 연산 호환성을 위해 PyTorch 프레임워크를 기반으로 최적화하였다.

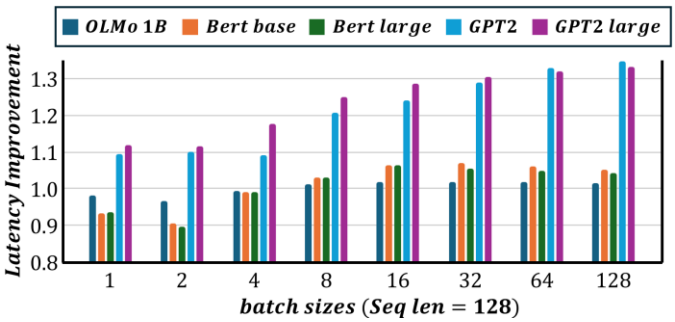
<표 1> 각 활성화 함수에 대해 LUT 테이블 크기가 1024 일 때, 최적의 Min/Max 값, 최대 속도 개선 비율(%), 그리고 오차율을 나타낸다. 여기서 Min/Max 값은 오차율을 최소화하기 위한 최적화 결과이다.

활성화 함수	Min	Max	최대 속도 개선 (%)	오차율 (%)
OpenAI GeLU	-2.567	2.538	892.87	0.1026
GeLU	-2.509	2.712	195.23	0.1148
SiLU	-3.844	3.612	196.89	0.7595

4.1 활성화 함수 별 성능 분석

<표 1>은 LUT 크기가 1024 일 때, 본 기법을 각 활성화 함수에 적용했을 때의 최대 속도 개선율과 오차율을 나타낸다. 각 활성화 함수에 대해 연산을 100 회 수행한 후 평균값을 측정하여 사용하였다. OpenAI GeLU 함수는 LUT 기반 최적화에서 가장 높은 성능 개선을 보였으며, 최대 2048 토큰에서 9 배의 속도 향상이 관찰되었다. 이는 활성화 함수 자체의 높은 연산 비용으로 인해, 실시간 연산 대신 미리 계산된 값을 사용함으로써 극적인 성능 향상이 가능했기 때문이다. 또한, 오차율은 0.1%로 매우 낮게 유지되었다.

반면, GeLU 와 SiLU 는 최대 약 2 배의 개선율을 보였다. 이 두 함수는 OpenAI GeLU 에 비해 연산량이 적기 때문에, LUT 를 통한 최적화 효과가 상대적으로 적었다. SiLU 의 경우 오차율이 0.76%로 다소 높게 나타났다.



(그림 2) 각 LLM 모델에 대해 시퀀스 길이가 128 일 때 배치 크기 변화에 따른 지연시간 개선 비율을 보여준다. 특히 GPT2 모델에서 배치 크기가 커질수록 성능 향상이 두드러진다.

4.2 배치 사이즈에 따른 성능 분석

(그림 2)는 다양한 모델에서 배치 크기의 변화에 따라 본 기법을 적용했을 때 지연시간(Latency) 개선율을 보여준다. GPT2 계열의 모델에서는 모든 조건에서 상대적으로 큰 지연시간 개선을 보였으며, 배치 크기가 커질수록 최대 1.35 배의 성능 향상이 있었다. 이는 GPT2 모델이 OpenAI GeLU 활성화 함수를 채택하고 있으며, 이 연산이 모델 지연 시간 중 높은 비중을 차지하기 때문이다.

반면, BERT 계열의 모델과 OLMo 모델의 경우 전체적으로 성능 개선 정도가 크지 않고, 배치 크기가 작았을 때 성능이 떨어지는 경우를 보였다. 이는 PyTorch 의 Tensor-Core 최적화가 입력 크기가 작을 때 더 빠른 연산을 가능하게 하지만, 메모리 복사 오버헤드로 인해 성능 저하가 발생할 수 있기 때문이다. 배치 크기가 클수록 메모리 접근 비용이 분산되어 성능이 개선되었다.

또한, BERT 와 OLMo 모델은 GPT2 모델에 비해 활성화 함수가 지연시간에 차지하는 비중이 각각 최대 15%, 10% 내외로 작다. 이에 따라 BERT, BERT Large, OLMo 모델은 최대 각각 1.03 배, 1.07 배, 1.06 배의 속도 개선율을 보였다.

5. 결론

룩업 테이블 기반의 최적화는 고비용 활성화 함수의 연산 효율성을 크게 향상시키며, 특히 OpenAI GeLU 와 같은 복잡한 연산에 대해 뛰어난 성능 개선을 보였다. 본 연구의 결과는 대규모 모델에서 활성화 함수의 병목 현상을 완화하고, GPU 리소스를 효율적으로 활용하는 데 기여할 수 있다. 향후 연구에서는 더 다양한 활성화 함수와 데이터셋을 대상으로 추가적인 최적화 및 분석이 필요할 것이다.

이 논문은 2024 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (IITP-2024-2021-0-01817, No.RS-2020-II201373, 인공지능대학원지원(한양대학교))

참고문헌

[1] Hendrycks, Dan, and Kevin Gimpel. "Gaussian error linear units (gelus)." arXiv preprint arXiv:1606.08415 (2016).
[2] Elfwing, Stefan, Eiji Uchibe, and Kenji Doya. "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning." Neural networks 107 (2018): 3-11.
[3] Devlin, Jacob. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
[4] Radford, Alec, et al. "Language models are unsupervised multitask learners." OpenAI blog 1.8 (2019): 9.
[5] Groeneveld, Dirk, et al. "Olmo: Accelerating the science of language models. arXiv preprint, 2024." URL https://api. semanticscholar. org/CorpusID 267365485.