동적 최댓값 대응 Radix-2 Softmax 설계

정서호, 노수민, 정기석*
한양대학교

iona97@hanyang.ac.kr, smrho@hanyang.ac.kr, *kchung@hanyang.ac.kr

Radix-2 Softmax Design Leveraging Dynamic Max-Value Response

Seo Ho Chung, Soo Min Rho, Ki Seok Chung*
Hanyang Univ., Seoul, Korea

요 약

Softmax 함수는 분류에 유용한 확률 분포를 제공하여 deep learning 네트워크에서 널리 사용되고 있다. 그러나 범용 프로세서에서 사용되는 look-up table (LUT)이나 테일러 급수와 같은 고정밀도 연산을 통한 softmax 구현은 Transformer -based large language model 에서 large area 및 power overhead 가 요구되어 비효율적인 문제가 발생한다. 또한 softmax 내부에는 overflow 문제를 방지하기 위해 입력값에서 최댓값을 빼는 정규화 과정이 필요하므로 추가적인 computation cost 및 memory overhead 문제가 발생하게 된다. 본 논문에서는 이러한 문제점을 해결하고자 동적으로 최댓값에 대응하는 효율적인 radix-2 softmax 를 제안한다. 이는 동적으로 최댓값에 대응하지 못하는 conventional radix-2 softmax 대비 power 는 2.56x, area 는 2.15x, latency 는 1.35x 성능 향상을 보였다.

Ⅰ. 서 론

Transformer neural network 는 자연어 처리 분야에서 state-of-the-art 성능을 기록하면서 중요한 deep learning 모델이 되었다 [1]. 이러한 모델의 구성 요소 중 하나인 softmax 함수는 입력 시퀀스의 각 요소가 얼마나 중요한지를 확률적으로 결정하는 중요한 역할을 하고 있다.

하지만 Transformer neural network 에서의 softmax 함수는 두 가지 문제점이 존재한다 [2], [3]. 첫째, soft-max 연산은 전체 추론 연산 시간에서 적지 않은 부분을 차지하고 있다. 그 이유는 자연 지수 함수를 표현하기 위해 look-up table (LUT)이나 테일러 급수와 같은 large area, power overhead 가 요구되는 방법으로 연산을 수행하기 때문이다. 이러한 문제를 해결하기 위한 방법 중 하나로 자연 지수 함수를 2 의 거듭제곱 연산으로 근사화하는 연구가 존재한다 [2]. 이는 shift 연산을 통해 지수 연산을 구현할 수 있는 장점이 있고, fine-tuning 시 정확도 손실을 1% 미만까지 줄일 수 있다. 둘째, softmax 함수는 지수의 값이 매우 클 때 발생하는 overflow 문제를 방지하기 위해 입력값에서 최댓값을 빼는 정규화 과정이 필요하다. 이로 인해 최댓값을 찾기 위한 추가적인 computation cost 및 memory overhead 문제가 발생하게 된다. 최댓값을 통한 정규화를 반영한 softmax 함수 수식은 다음과 같다.

$$y(x_i) = \frac{e^{x_i - x_{max}}}{\sum_j e^{x_j - x_{max}}}$$

본 논문에서는 이러한 정규화 과정을 효율적으로 처리하기 위해 동적으로 최댓값에 대응하는 radix-2 softmax 구조를 제시한다. 또한 제안하는 동적 최댓값 대응을 적용하지 않은 conventional radix-2 softmax 와 성능 (latency, power, area)를 비교하였다.

Ⅱ. 본론

2.1 Conventional radix-2 softmax

Softmax 함수 구현에서 자연 지수 함수의 복잡성을 해결하기 위해 자연 지수 함수를 2 의 거듭제곱 연산으로 대체하고, 정규화 과정을 수행한 하드웨어 구조는 그림 1 과 같다.

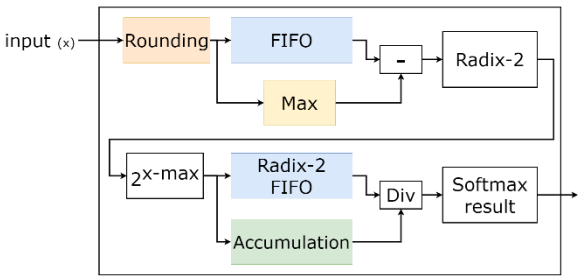

그림 1. Conventional radix-2 softmax

고정 소수점 입력 x 가 정수로 rounding 되고, 최댓값을 찾기 위해 First In First Out (FIFO) 버퍼에 저장된다. Max unit 은 FIFO 의 입력 중 최댓값을 찾는 기능을 수행한다. 입력 데이터가 시퀀스 길이만큼 입력된 이후부터 FIFO 에서 출력이 발생하며, 해당 출력에 최댓값을 뺀 값이 radix-2 unit 에 입력된다. 출력된 2 의 거듭제곱 수는 Radix-2 FIFO 에 저장되면서 누적 연산을 진행하고, 연산이 끝나면 divider 를 통해 최종 softmax 결과를 출력한다.

하지만 그림 1 의 방법은 모든 입력 시퀀스를 FIFO 에 저장하고 난 뒤 최댓값을 구하게 되므로, 정규화 과정에 상당한 지연이 생기는 단점이 존재한다. 또한 최댓값을 찾기 위한 FIFO 와 radix-2 FIFO 가 따로 존재하게

되므로 large area 를 요구하게 되고, 성능 저하를 일으키게 된다.

2.2 동적 최댓값 대응 radix-2 softmax

2.1 의 문제를 해결하기 위해 입력 시퀀스의 최댓값에 동적으로 대응하는 효율적인 radix-2 softmax 를 제안한다. 그림 2 는 conventional radix-2 softmax 연산 알고리즘과 제안하는 동적 최댓값 대응 radix-2 sofmax 알고리즘이다.

Algorithm 1 Conventional softmax

$m_0 = -\infty$
$sum = 0$
for $1 \le i \le S$ **do**
  $m_i \leftarrow max(m_{i-1}, x_i)$
end for
for $1 \le j \le S$ **do**
  $sum \leftarrow sum + 2^{x_j - m_S}$
end for
for $1 \le k \le S$ **do**
  $y_k \leftarrow \frac{2^{x_k - m_S}}{sum}$
end for

Algorithm 2 Proposed softmax

$m_0 = -\infty$
$sum = 0$
// Dynamic max-value response
for $1 \le i \le S$ **do**
  $m_i \leftarrow max(m_{i-1}, x_i)$
  $sum \leftarrow sum \times 2^{m_{i-1} - m_i} + 2^{x_i - m_i}$
end for
for $1 \le j \le S$ **do**
  $y_j \leftarrow \frac{2^{x_j - m_S}}{sum}$
end for
// Reduced 3 loops to 2

그림 2. (a) 기존 방식 알고리즘과
(b) 최댓값에 동적으로 대응하는 알고리즘

예시는 다음과 같다. [3, 1, 5]가 순차적으로 입력이 된다고 가정하자. 첫 번째 입력으로 3 이 들어오면, 현재 입력된 값들의 최댓값이 3 이므로 $sum = 2^{3-3} = 1$ 이다. 다음으로 1 이 입력되면, 현재의 최댓값은 3 으로 유지되어 $sum_{new} = sum_{prev} + 2^{1-3} = 1 * 2^0 + 2^{1-3}$ 으로 연산할 수 있다. 마지막으로 5 가 입력된 경우 최댓값이 5 로 바뀌게 되므로, 기존의 최댓값 3 으로 정규화된 sum 을 최댓값 5 로 재정규화하는 변환 과정이 필요하다. 이는 기존의 sum 에 2^{3-5}를 곱해주는 것으로 수행할 수 있다. 결과적으로 $sum_{new} = sum_{prev} * 2^{3-5} + 2^{5-5} = 2^{-2} + 2^{-4} + 2^0$로 연산된다. 이는 5 가 최댓값임을 알고 누적한 $sum = 2^{3-5} + 2^{1-5} + 2^{5-5}$과 일치하는 것을 알 수 있다. 위의 과정은 그림 2 의 알고리즘 (b) 내부 $sum_{new} = sum_{prev} * 2^{max_{prev} - max_{new}} + 2^{in - max_{new}}$ 부분에 해당한다. 이 경우, 최댓값을 구하기 위해 모든 입력 시퀀스를 FIFO 에 저장할 필요성이 없어져, 정규화 과정에서 발생하는 지연을 제거할 수 있다.

또한 그림 2 의 알고리즘 (b)에 shift 연산을 적용하면 $sum_{new} = sum_{prev} \gg (max_{new} - max_{prev}) + 1 \gg (max_{new} - in)$ 로 표현할 수 있어, 기존의 곱셈 연산을 사용하는 경우보다 구현 비용이 감소한다.

그림 3 은 제안하는 동적 최댓값 대응 radix-2 softmax 하드웨어의 구조이다. 그림 1 과 비교하여 FIFO 가 하나로 줄어든 것을 확인할 수 있고, Max unit 의 경우 최댓값 동적 대응을 위해 Max Difference 및 max-input 을 연산하는 추가 기능이 구현된 것을 확인할 수 있다.

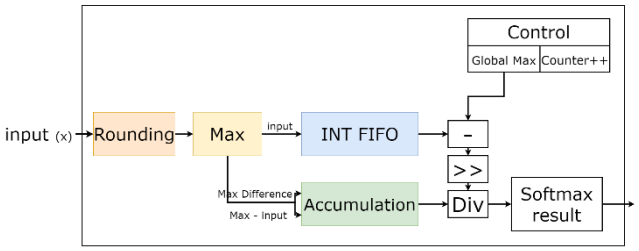

그림 3. Proposed efficient radix-2 softmax

2.3 실험방법 및 결과

제안한 두 softmax 디자인은 Verilog HDL 언어를 사용하여 RTL 설계로 구현하였고, 15nm NangateOpenCell Library 를 사용하여 Synopsys Design Compiler 를 이용하여 합성하였다.

Latency 측정은 GLUE task 데이터를 가정하여 최대 시퀀스 길이를 128 로 설정하였다 [4].

Conventional softmax 구현과 proposed softmax 구현의 전력소모, 회로면적 및 속도 비교 결과는 표 1 과 같다. C 와 P 는 conventional 과 proposed 를 뜻한다.

표 1. C.softmax (6.32GHz)와 P.softmax (5.78GHz)의 성능 비교

	Power (mW)	Area (um^2)	Latency (ns)
C.softmax	102.61	9357.75	62.88
P.softmax	40.08	4345.28	46.54

그 결과 proposed softmax 가 conventional softmax 보다 power 는 2.56x, area 는 2.15x, latency 는 1.35x 의 성능 향상을 보였다. Proposed softmax 는 입력 시퀀스 길이(128 cycles)만큼 기다리지 않아도 되므로 latency 또한 성능이 향상되었다. 제안된 방식이 power, area, latency 에서 모두 성능이 향상되었고, 또 다른 성능 지표인 power delay product (PDP) 및 energy delay product(EDP)에서도 각각 3.45x, 4.67x 의 향상을 보였다.

Ⅲ. 결론

본 논문에서는 softmax 함수 구현 시 발생하는 두가지 문제점을 해결하기 위해 동적으로 입력 시퀀스의 최댓값에 대응하는 효율적인 radix-2 softmax 를 제안한다. 이는 기존의 방식보다 power, area, latency 측면에서 각각 2.56x, 2.15x, 1.35x 향상이 되었으며, 기존의 power overhead, large area, computation cost, memory overhead 문제를 해결할 수 있다. 또한 PDP 및 EDP 에서 각각 3.45x, 4.67x 의 성능 향상이 있음을 확인하였다.

ACKNOWLEDGMENT

본 논문은 2022 년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임(No. 2022-0-00153, 빔포밍 경로 분산을 이용한 AI 네트워크관리 기반 인빌딩용 O-RU 개발)

참 고 문 헌

[1] A. Vaswani, N. Shazeer, N. Parmar et al., “Attention is all you need,”in Proc. of NeurIPS, 2017, pp. 5998-6008.

[2] Stevens, J. R., Venkatesan, R., Dai, S., Khailany, B., & Raghunathan, A. “Softermax: Hardware/software co-design of an efficient softmax for transformers.” In Proc. of 58th ACM/IEEE Design Automation Conference (DAC), 2021 (pp. 469-474). IEEE.

[3] Li, T., Zhang, F., Xie, G., Fan, X., Gao, Y., & Sun, M. “A high-speed reconfigurable architecture for softmax and GELU in vision transformer” in Proc. of Electronics Letters, 2023, 59(5), e12751.

[4] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2018). GLUE: A multi-task benchmark and analysis platform for natural language understanding. arXiv preprint arXiv:1804.07461.