다층퍼셉트론과 합성곱 신경망 및 AI 의 에너지 소비

임문선 백승우(멘토)

대전대신고등학교(Daejeon Daeshin High.) A.C.T.(KE)

Energy consumption of multilayer perceptrons, convolutional neural networks, and AI

ABSTRACT: XNOR 게이트를 추론하는 다층 퍼셉트론 신경망을 직접 설계하고 영상 분류를 위한 합성곱 신경망의 주요 연산 과정을 탐구하여 AI 추론의 핵심 연산을 파악하고, AI 학습과 추론 과정의 컴퓨터 시스템의 전력 소모와 관련하여 XNOR 게이트 추론기의 언어별 구현에 따른 메모리 사용량과 절감 방법을 탐구하고 최신 AI 용 반도체의 저정밀도 연산 지원 추세를 탐구하였다.

Ⅰ. 서론

오랜 침체기에 있던 AI 기술이 딥러닝 기술에 의해 급격히 발전하여 각 분야에서 본격적으로 사용되면서 직장인의 고용, 학생의 진로 차원의 문제부터 기술 패권주의에 대비하여 독자 기술 확보 및 안정적 운영을 해야 한다는 국가차원의 문제까지 제기되고 있다.

초기 신경망의 문제를 해결한 다층 퍼셉트론 신경망(Multi-Layer Perceptron Neural Network)과 영상 분류 문제를 다층 퍼셉트론 신경망보다 더 효율적으로 해결한 합성곱 신경망(Convolutional Neural Network)은 딥러닝 기반 AI 학습 기술을 이해하는 중요한 관문이며, 이를 통해서 GPU 보다 전력을 효율적으로 사용하는 AI 추론기의 핵심 기능을 이해할 수 있다.

본 탐구는 XNOR 게이트를 추론하는 MLP 와 두 가지 영상을 구분하는 CNN 의 연산 과정을 가시화하여 AI 추론의 핵심 연산을 파악하고, AI 학습 및 활용 단계에서 전력 소모를 줄이는 방법의 하나로 AI 추론기의 메모리 사용량 절감 방법에 대해 알아본다.

Ⅱ. 연구 방법

2. 신경망 **탐구**

2.1 XNOR Gate 추론기의 MLP 구현

XNOR(Exclusive NOR) 게이트는 두 입력이 동일하면 1, 서로 다르면 0 을 출력하는 논리 게이트로서 XOR(Exclusive OR) 게이트와 반대로 동작하며, XOR 게이트와 마찬가지로 비선형 결정 경계를 갖기 때문에 단층 퍼셉트론으로는 구현할 수 없다.

표 1. XOR, XNOR 진리표

x1	x2	XOR(x1, x2)	XNOR(x1, x2)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

XNOR 게이트는 XOR 게이트의 정의를 이용하여 식(1)과 같이 정의할 수 있다.

$$\begin{aligned} &\text{XNOR}(x1, x2) \\ &= \text{NOT}(\mathbf{XOR(x1, x2)}) \\ &= \text{NOT}(\mathbf{AND}(\text{NAND}(x1, x2), \mathbf{OR}(x1, x2))) \\ &= \text{NAND}(\text{NAND}(x1, x2), \text{OR}(x1, x2)) \end{aligned} \quad (1)$$

식(1)을 노드 2 개로 구성된 1 개의 은닉층을 갖는 다층 퍼셉트론(MLP) 구조로 [그림 1]과 같이 나타낼 수 있다.

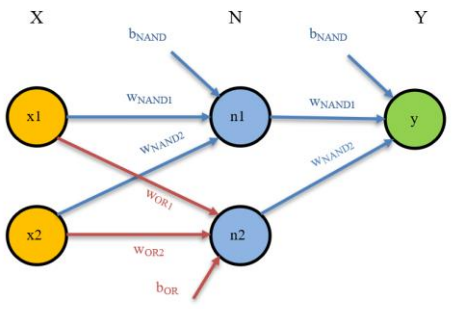

그림 1. XNOR MLP 구현

표 2. XNOR MLP 구현의 노드별 의미

X		N		Y
x1	x2	n1=NAND	n2=OR	y=XNOR
0	0	1	0	1
0	1	1	1	0
1	0	1	1	0
1	1	0	1	1

이 다층 퍼셉트론 구조에서, 청색 가중치와 편향을 이용해서 NAND 연산을 수행하고, 적색 가중치와 편향을 이용해서 OR 연산을 수행할 수 있으면, 은닉층의 n1 은 NAND(x1, x2)에 해당하고 n2 는 OR(x1, x2)에 해당되어 출력층의 y 는 구하고자 하는 XNOR(x1, x2)로 계산된다.

이를 위해서 은닉층의 값은 0 또는 1 이 되어야 하므로 치역이 0 과 1 사이로 제한되는 logistic 함수 S 를 활성화 함수로서 사용한다.

$$Sx = \frac{1}{1+e^{-x}} \quad (2)$$

식(2)를 활성화함수로 사용할 때, W_{NAND1} , W_{NAND2}, b_{NAND}, W_{OR1}, W_{OR2}, b_{OR} 은 각각 -4, -4, 6, 4, 4, -2 일 수 있다. 이 가중치와 편향값을 이용해서 은닉층 N 과 출력층 Y 의 다층 퍼셉트론 행렬 계산을 하고 활성화함수를 적용하면 [그림 2]와 같은 계산 단계를 거친다.

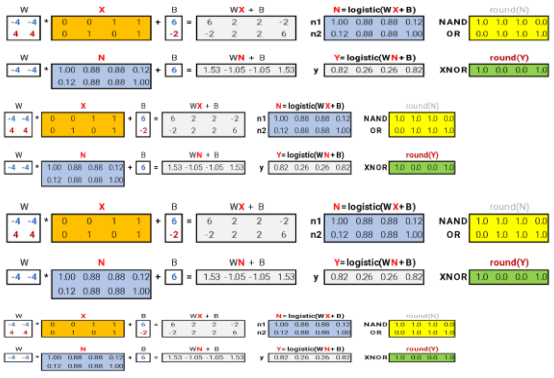

그림 2. XNOR MLP 계산 과정

은닉층 N, 최종값 Y 를 반올림하면 입력값 [(0,0), (0,1), (1,0), (1,1)]에 대한 n1=NAND, n2=OR, y=XNOR 을 각각 얻게 된다.

2.2 영상분류를 위한 CNN 의 작동 원리

가로 픽셀 W 개, 세로 픽셀 H 개인 흑백영상은 총 W*H 개의 픽셀로 구성된다. 이 영상을 예를 들어 두 가지의 영상으로 분류하기 위해 다층 퍼셉트론(MLP)을 사용한다면 중간의 은닉층은 [그림 3]과 같이 입력층의 W*H 개의 픽셀에 대한 모든 연결의 가중치와 편향값을 가져야 한다.

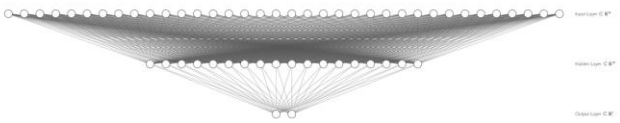

그림 3. 6x6 해상도 단색영상의 이진 분류를 위한 MLP

앞서의 XNOR 다층 퍼셉트론 구현에서, 은닉층의 각 노드는 입력층의 각 노드에 대해 특정한 의미를 갖는다는 것을 확인하였지만, 은닉층의 한 노드가 영상내에서 서로 멀리 떨어진 픽셀들에 대해서 동시에 유의미한 값을 가질 확률은 별로 없다. 따라서 영상에 대해서는 다층 퍼셉트론의 가중치와 편향값의 대부분이 효용이 없을 것이라는 점에 착안하여, 인근 픽셀에 대해서만 의미를 찾는 합성곱 네트워크(Convolutional Neural Network)가 등장하게 되었다. 이는 작은 영상 조각(커널, kernel)들과 같은 크기의 영상 조각을 픽셀 단위로 각각 곱해서 합하는 합성곱 연산 결과로 이루어진 특성맵들을 구하고 마지막에 완전연결 레이어를 이용하는 것으로 MLP 에 비해서 파라미터 수를 크게 줄이면서도 영상 인식 분야에서 좋은 성능을 내었다.

6x6 해상도 단색 영상에 대해 "/" 패턴 또는 "\" 패턴의 두 경우로 판별하기 위해, 3x3 합성곱 커널 2 개로 6x6 해상도의 특성맵 2 개를 구한 후, 2x2 Max-Pooling 을 적용해서 3x3 특성맵 2 개를 구하고, 18 개의 노드로 구성된 선형층으로 변환하여 2 개의 출력층으로 출력하는 CNN 을 다이어그램으로 나타내면 [그림 4]와 같다.

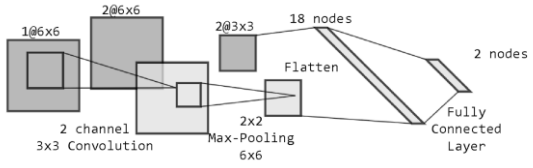

그림 4. 6x6 해상도 단색영상의 이진 구분을 위한 CNN

[그림 5]는 6x6 크기 단색 영상에 대해 "/" 패턴 또는 "\" 패턴의 두 경우로 판별하는 CNN 에 대하여, 구분하고자

하는 패턴과 유사한 3x3 커널 2 개를 가정하여 일부 구간의 연산을 엑셀로 가시화한 결과이다.

그림 5. CNN 의 합성곱 레이어와 풀링 레이어 연산 과정

2.3 AI 추론의 핵심 연산

MLP 추론기와 마지막 단계에서 MLP 를 포함하는 CNN 추론기에는 공통적으로 단계마다 가중치 행렬과 노드값의 행렬곱에 편향값 벡터를 합하는 행렬 연산이 수행되며, 이후 다양한 활성화 함수를 적용한다. CNN 은 2 차원 단색 영상 또는 3 차원 컬러 영상에 대한 반복적인 합성곱 연산 및 활성화 함수 적용, 풀링 기능이 추가로 필요하다.

3. AI 의 에너지 소비 문제 탐구

3.1 컴퓨터 시스템의 에너지 소비 원인

컴퓨터 시스템을 구성하는 장치들은 대부분 CMOS 소자로 구성되어 있으며, 전력 소모는 식(3)과 같이 정적 전력 소모 $P_{(static)}$와 동적 전력 소모 $P_{(dynamic)}$으로 구분할 수 있으며 소자의 발열을 유발한다.

$$P_{tot} = P_{(static)} + P_{(dynamic)} \quad (3)$$

동적 전력 소모 $P_{(dynamic)}$는, 트랜지스터의 상태가 변할 때의 **과도 전력 소모 P_T**와 출력 노드나 배선에 존재하는 정전용량을 충전하거나 방전하기 위한 **부하 정전용량 전력 소모 P_L**로 나눌 수 있으며, N_{SW}개의 비트가 바뀔 때에 다음과 같이 모델링 된다.

$$P_{(dynamic)} = P_T + P_L = \left[C_{pd} \bullet f_I \bullet N_{SW} + \sum_{n=1}^{N_{SW}} \left(C_{Ln} \bullet f_{On} \right) \right] \times V_{CC}^2 \quad (4)$$

식(4)에서 C_{pd} 는 전력 소모 정전 용량, f_I 는 입력 신호 주파수, C_{Ln} 은 n 번째 출력의 부하 정전용량, f_{On} 은 n 번째 출력의 출력 주파수, V_{CC} 는 공급 전압을 뜻하며, 연산뿐만 아니라 Cache 메모리나 DRAM 액세스 작업 등에도 공통적으로 적용될 수 있다. 따라서 식(4)로부터 저전압 프로세서의 사용뿐만 아니라 프로그램이 사용하는 메모리를 줄이는 것도 소자의 에너지 소비 절감과 컴퓨터 시스템 냉각에 도움이 된다는 것을 확인할 수 있다.

3.2 XNOR MLP 의 언어별 구현

앞 절의 XNOR MLP 추론기는 행렬곱과 합 연산과 행렬 요소에 대한 개별적인 logistic 함수로 구성되어 있다. 이는 행렬 연산과 vectorization 을 지원하는 MATLAB/Octave 언어로 다음처럼 매우 간결하게 계산할 수 있지만 숫자 데이터 기본타입은 double 로서 8 바이트씩 사용된다.

```
L = @(x) 1 ./(1+exp(-x));
W1 = [ -4, -4; 4, 4];
B1 = [ 6; -2];
W2 = [-4, -4];
B2 = 6;
X = [0, 0; 0, 1; 1, 0; 1, 1]'
N = L(W1*X+B1);
XNOR = round(L(W2*N+B2))
```

Python 언어에서도 NumPy 패키지를 이용하여 비슷한 수준으로 표현할 수 있으며, 숫자 데이터형을 np.int64, np.int32, np.int16, np.int8 등으로 쉽게 지정할 수 있다.

```
import numpy as np
```

```
T = np.int8    # np.int16, np.int32, np.int64


L = lambda x: 1 / (1 + np.exp(-x))

W1 = np.array([[-4, -4], [4, 4]], dtype=T)

B1 = np.array([[6], [-2]], dtype=T)

W2 = np.array([[-4, -4]], dtype=T)

B2 = T(6)


X = np.array([[0, 0, 1, 1], [0, 1, 0, 1]], dtype=T)

N = L(W1 @ X + B1)

XNOR = np.int32(np.round(L(W2 @ N + B2)))


print("X:")

print(X)

print("XNOR:\n", XNOR)
```

C 언어의 경우도 숫자 데이터형을 8 비트를 사용하는 signed char 까지 자유롭게 사용할 수 있지만, 행렬연산을 직접 지원하지 않으므로 MATLAB/Octave 나 Python NumPy 처럼 간결하지는 않다.

```
#include <stdio.h>


typedef signed int T;


T s(T x) { return x > 0; }


int main() {

    T W1[2][2] = {{-4,-4},{4,4}};

    T B1[2] = {6,-2};

    T W2[2] = {-4,-4};

    T B2 = 6;

    T X[4][2] = {{0,0},{0,1},{1,0},{1,1}};


    for(int i=0;i<4;i++) {

        T N[2] = {0};

        for(int j=0;j<2;j++) {

            N[j] = s(W1[j][0]*X[i][0]

 + W1[j][1]*X[i][1] + B1[j]);

        }

        T out = s(W2[0]*N[0] + W2[1]*N[1] + B2);

        printf("%d ", out);

    }

    printf("\n");


    int total_size =

sizeof(W1)+sizeof(B1)+sizeof(W2)+sizeof(B2);

    printf("Total weights size: %d bytes\n",

 total_size);

}
```

3.3 메모리 절약을 위한 8 비트 미만 정밀도 정수

C 언어는 bit-field 기능을 이용해서 정수형의 길이를 임의로 설정할 수 있고, 구조체 packing 기능과 함께 사용하면 변수들을 위한 메모리 공간을 낭비하지 않고 사용할 수 있다. 이를 이용하여 앞 절의 XNOR MLP 추론기의 가중치와 편향값을 8 비트 미만의 길이로 사용할 수 있도록 아래와 같이 변경하였다.

```
#include <stdio.h>


#ifndef BW

#define BW 4

#endif
```

```
typedef struct __attribute__((__packed__)) {
   signed int w11:BW, w12:BW, w21:BW, w22:BW,
   signed int b1:BW, b2:BW, w2_1:BW, w2_2:BW, b3:BW;
} W4;

int s(int x) { return x > 0; }

int xnor(W4 W, int x1, int x2) {
   W4 N;
   N.w11 = s(W.w11*x1 + W.w12*x2 + W.b1);
   N.w21 = s(W.w21*x1 + W.w22*x2 + W.b2);
   return s(W.w2_1*N.w11 + W.w2_2*N.w21 + W.b3);
}

int main() {
   W4 W = {-4,-4,4,4,6,-2,-4,-4,6};
   int X[4][2] = {{0,0},{0,1},{1,0},{1,1}};

   for(int i=0;i<4;i++)
      printf("%d ", xnor(W,X[i][0],X[i][1]));

   printf("\n Packed weights W4 size: %zu bytes (BW=%d bits)\n", sizeof(W), BW);
}
```

이 방법으로 [표 3]과 같이 총 9 Byte 가 필요하던 9 개 가중치와 편향값을 각각 4 비트, 총 36 비트로 압축하여, 총 5 Byte 에 저장하고 정상적으로 XNOR 추론을 수행했다.

표 3. Bit-field 와 Structure packing 을 이용한 XNOR 파라미터 실험 결과

BW	Packed W4 size	XNOR Output	Notes
8 bits	9 Bytes	1 0 0 1	Pass
7 bits	8 Bytes	1 0 0 1	Pass
6 bits	7 Bytes	1 0 0 1	Pass
5 bits	6 Bytes	1 0 0 1	Pass
4 bits	5 Bytes	1 0 0 1	Pass
3 bits	4 Bytes	0 0 0 0	Fail

3.4 AI 분야의 저정밀도 연산 사용 추세

GPU(Graphics Processing Unit)가 과학기술용으로 처음 쓰일 때에는 정밀한 연산을 위해서 32 비트 부동소수점(FP32) 성능 위주로 개발되었지만, GPU 가 AI 학습 및 추론을 위해서 많이 쓰이면서 점차 16 비트 데이터형(FP16, BF16), 8 비트 데이터형(FP8, cFP8), 4 비트 데이터형(INT4, FP4), 바이너리 데이터형 등의 저정밀도 연산을 지원하고 있으며, 저정밀도 연산을 사용할수록 높은 연산능력을 보인다.

특히 AI 기능이 실생활에서 본격적으로 사용되면서 AI 추론에 주로 쓰이는 행렬곱과 합 및 활성화 함수 등 한정된 기능에 특화되어 소모전력 대비 높은 추론 성능을 갖춘 NPU(Neural Processing Unit)들이 개발되어 AWS EC2 등의 추론 전용 클러스터부터 각종 데스크탑 또는 모바일 CPU 와 Rockchip 등의 모바일 프로세서에 내장되어 AI edge 디바이스로서 사용되고 있다[표 4].

표 4. NPU 용도 및 저정밀도 연산용 데이터 타입

제품	용도	주요 데이터 타입
Amazon Inferentia	추론용 클라우드	FP16, BF16, INT8
Amazon Inferentia2	추론용 클라우드	FP16, BF16, INT8 FP32, TF32, cFP8
AMD Ryzen AI	PC, 랩탑	FP32 INT8(CNN)

		BF16(CNN, NLP)
Intel NPU	PC, 랩탑	INT8, FP16
Hailo-8	PC, 랩탑, SBC	4,8,16-bits
Rockchip NPU	모바일, SBC	INT4,INT8,INT16,FP16

IV. 결론

XNOR 게이트 추론기와 CNN 의 주요 연산 과정을 탐구하여 AI 학습 및 추론의 핵심 연산이 행렬 연산임을 확인하였고, 컴퓨터 시스템의 전력 소모 모델을 통하여 메모리 사용량을 줄이는 것 또한 AI 학습 및 추론 과정의 전력 소모에 도움이 된다는 것을 확인하여 XNOR 추론기의 메모리 사용량을 줄이는 방법을 다양한 언어와 기법으로 확인하였다.

V. 논의

PyTorch 등으로 학습한 실제 AI 모델의 데이터타입을 변경하여 메모리 사용량을 줄이고 전력 절감 효과를 측정하는 것은 추후에 탐구할 과제이다.

참고 문헌

[1] 와쿠이 요시우키. 2020. 엑셀로 배우는 딥러닝. 성안당. 서울

[2] 조태호. 2025. 모두의 딥러닝. 개정 4 판, 길벗, 서울

[3] 김성필. 2016. 딥러닝 첫걸음. 한빛미디어, 서울

[4] Abul Sarwar. 1997. CMOS Power Consumption and C_{pd} Calculation. Texas Instruments

[5] Mark Horowitz, 2014. Computing's Energy Problem(and what we can do about it). ISSCC 2014

[6] AMD. Ryzen AI Software 1.6.1 documentation. https://ryzenai.docs.amd.com/en/latest/relnotes.html

[7] Intel. Intel Neural Processing Unit(Intel NPU). https://edc.intel.com/content/www/us/en/design/products/platforms/details/arrow-lake-s/core-ultra-200s-series-processors-datasheet-volume-1-of-2/intel-neural-processing-unit-intel-npu/

[8] Hailo. 2024. Hailo Dataflow Compiler User Guide Release 3.27.0

[9] Rockchip. 2023. Rockchip RK3588 Datasheet