Journal of KIIT. Vol. 19, No. 11, pp. 55-62, Nov. 30, 2021. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2021.19.11.55

마이크로컨트롤러의 메모리 구조를 고려한 다층 신경망 최적화

최 권 택

Optimization of Multi-layer Neural Network Considering Memory Structure of Microcontroller

Kwontaeg Choi

요 약

인공지능 산업의 발달로 AI 기술과 IoT 기술을 융합한 사물지능(AIoT)에 관한 다양한 연구가 진행되고 있다. 이를 위해 주로 사용하는 마이크로컨트롤러에서는 동적으로 할당 가능한 메모리 크기가 작아 고차원 특징을 사용하는 신경망 알고리즘을 적용하기 어렵다. 본 논문에서는 FLASH 메모리에 학습 파라미터를 저장하고, 메모리 용량이 작은 SRAM에서 입출력 메모리 주소 교환을 통해 메모리 사용을 최적화하는 신경망 구조를 제안한다. UCI HAR 데이터셋 실험에서 TensorFlow Lite는 메모리 부족으로 컴파일이 불가능한 반면, 제안하는 방법은 동일한 크기의 바이너리 파일을 생성해 FLASH 55%, SRAM 16%만 사용해서 다층 신경망을 수행시킬 수 있었다. 수행시간은 두 배정도 증가했지만, UCI HAR 데이터셋에서 43ms로 실시간 수행이 가능했다.

Abstract

With the development of the artificial intelligence industry, various researches are being conducted on AIoT, a fusion of AI technology and IoT technology. For this purpose, it is difficult to apply a neural network algorithm using high-dimensional features due to the small size of dynamically allocable memory in microcontrollers mainly used for this purpose. In this paper, we propose a neural network structure that stores learning parameters in FLASH memory and optimizes memory usage by exchanging input/output memory addresses in SRAM with a small memory capacity. In the UCI HAR dataset experiment, TensorFlow Lite was unable to compile due to lack of memory, whereas the proposed method was able to create a binary file of the same size and perform a multilayer neural network using only 55% FLASH and 16% SRAM. Although the execution time was doubled, real-time execution was possible in 43 ms in the UCI HAR dataset.

Keywords
artifial neural network, tensorFlow lite, edge computing, arduino

* 강남대학교 소프트웨어응용학부 교수
- ORCID: https://orcid.org/0000-0001-5331-321X

· Received: Oct. 25, 2021, Revised: Nov. 14, 2021, Accepted: Nov. 17, 2021
· Corresponding Author: Kwontaeg Choi
Dept. of Software Application, Kangnam University, Changwondaehak-ro, Uichang-gu, Changwon-si, Gyeongsangnam-do, 51140, Korea
Tel.: +82-31-213-3098, Email: kwontaeg.choi@kangnam.ac.kr

Ⅰ. 서 론

인공지능 관련 연구 및 산업의 발전에 힘입어 주요 IT 기업들은 플랫폼 시장을 선점하기 위해서 클라우드 기반 인공지능 플랫폼을 앞다투어 제공하고 있다. 이러한 플랫폼은 다양한 기능과 편리성, 그리고 시스템 확대에 따른 스케일 업, 스케일 아웃 지원을 통해 서비스의 성능과 용량을 조정할 수 있다.

이러한 클라우드 기반 인공지능 플랫폼은 많은 장점이 있지만, 다양한 환경에서 서비스 확장을 위해 해결할 문제점 또한 가지고 있다. 대표적인 문제점 중의 하나는 서비스가 항상 인터넷 연결을 요구한다는 점이다. 그리고 인터넷 환경, 특히 무선 연결 상황에 따라 네트워크 지연 문제로 안정적인 서비스가 어려울 수 있다. 게다가 음성 및 카메라를 이용하는 서비스의 경우, 개인 사생활 정보 침해 우려도 있다.

이러한 문제점들을 보완하기 위해 에지 컴퓨팅(Edge computing)에 관한 연구가 활발히 이루어지고 있다[1]-[3]. 특히 IoT와 인공지능 기술을 융합한 사물지능(AIoT)에 관한 연구가 관심을 받고 있다[4][5]. 모바일과 임베디드 시스템에서 성능 벤치마킹에 관한 포괄적인 연구도 수행되었다[6]. 이들 연구에서는 복잡한 신경망 알고리즘을 수행하기 위해서 Nvidia의 Jetson, Rasberry Pi 같은 리눅스 기반 에지 컴퓨팅 디바이스를 사용하고 있다. 관련 연구로 Jetson TX2, Jetson Nano, Rasberry Pi의 벤치마킹[7], Jetson Nano에서 자율주행 시스템을 위한 기계학습 알고리즘 비교[8], 침입 탐지 시스템을[9] 들 수 있다.

그동안 IoT 관련 연구에서는 주로 Arduino 계열의 마이크로컨트롤러를 사용했는데, 이 장치는 센서 입출력 및 간단한 신호처리만 가능하다. 따라서 CPU 성능과 충분한 메모리가 필요한 다층 신경망 알고리즘을 사용할 수 없다.

이러한 한계를 극복하고자 최근 Arduino 계열 장치의 CPU 성능과 메모리 용량이 크게 개선되었고, 이제는 음성, 비디오, 제스처 데이터를 처리할 수 있어 다양한 분야에 응용되고 있다.

본 논문에서는 Arduino 계열 마이크로컨트롤러의 메모리 용량과 구조에 적합한 다층 신경망 알고리즘을 제안한다. 2장에서 마이크로컨트롤러의 메모리 구조에 대한 문제점을 살펴보고, 3장에서 핵심 알고리즘을 설명한다. 4장에서는 공용 데이터셋을 사용한 실험 결과를 보이고, 5장에서 결론 및 향후 과제를 논한다.

Ⅱ. 관련 연구

에지 컴퓨팅 연구를 위해 주로 사용하는 하드웨어 장치는 Raspberry Pi와 Nvidia Jetson Nano, AGX XAVIER이다. 이러한 장치는 리눅스 OS를 기반으로 수십 GB 크기의 마이크로SD가 하드디스크 역할을 담당하고, 4GB~8GB 크기의 RAM에서 프로그램이 실행된다. PC에서 사용하는 이러한 구조를 von Neumann 아키텍처라하고, 많은 메모리를 사용하는 인공지능 알고리즘을 운영할 수 있다.

반면 Arduino 계열 장치의 경우, 저 사양 CPU와 수십KB 메모리 한계로 인공지능 알고리즘을 실행시키기에는 많은 한계가 있어, 주로 간단한 센서 데이터 처리에 응용되고 있다. 그러나 관련 산업이 빠르게 발전하면서 이제는 음성인식, 스트리밍 데이터 처리, 영상인식, 지능형 카메라 분야에 응용되고 있다[10]-[12]. 특히 ESP32 시리즈와 Arduino Nano 계열 장치는 다양한 멀티미디어 및 복잡한 센서 데이터를 처리할 목적으로 개발되었다. 특히 예를 들어 ESP32-CAM 장치는 2M 수준의 OV2640 카메라 모듈을 가지고 있어 영상을 이용해 손으로 쓴 문자를 인식하는 컴퓨터 비전 분야에 응용되고 있다[12]. ESP32-CAM은 가격도 7$~20$로 저렴해 향후 다양한 분야에 활용될 수 있을 것으로 기대된다.

표 1에 대표적인 Arduino 계열 제품에 대한 메모리 용량을 정리하였다. Arduino 계열 마이크로컨트롤러는 von Neuman 아키텍처가 아닌 Harvard 아키텍처로 인해 메모리 구조도 다양하고, 프로그램 메모리와 데이터 메모리도 분리되어 있다. 따라서 메모리에 접근하는 방식이 다소 다르다[13]. Harvard 아키텍처에서 주로 사용하는 FLASH와 EEPROM은 프로그램과 데이터를 저장하는 용도로 사용되고, SRAM과 PSRAM은 데이터를 읽고 쓸 수 있는 동적 메모리 용도로 사용된다.

표 1. Arduino 제품별 메모리 비교
Table 1. Memory comparison by Arduino products

	FLASH	SRAM	PSRAM	EEPROM
Uno	32KB	2KB	-	1KB
MEGA	256KB	8KB	-	1KB
Node MCU	4MB	520KB	-	-
ESP32	1MB	320KB	4MB	-
Arduino Nano	1MB	256KB	-	-
TTGO T-Camera	4MB	520KB	8MB	-

표 1에서 볼 수 있듯 마이크로컨트롤러는 리눅스 기반 에지 컴퓨팅 하드웨어와 달리 메모리가 매우 제한되어 있다. 메모리 용량도 문제지만 읽기 전용 FLASH 메모리에 비해 읽고 쓰기가 가능한 SRAM 용량이 상대적으로 적다.

이렇게 작은 메모리를 가지고 있는 마이크로컨트롤러에서도 딥러닝 알고리즘을 응용할 수 있도록, 최근 TensorFlow Lite API가 발표되었다. 특히 Arm Cortex M3 프로세서에서 16KB 메모리만을 사용해 핵심 알고리즘을 구현했고, Cortex-M 시리즈 아키텍처를 기반으로 하는 여러 프로세서에서 광범위하게 검증되고 다양한 연구가 진행되고 있다[14][15].

그러나 TensorFlow Lite가 FLASH와 SRAM을 사용해 학습 파라미터를 저장하기 때문에 SRAM이 상대적으로 작은 마이크로컨트롤러에서 규모가 큰 다층 신경망을 구현하기에는 한계가 있다.

본 논문에서는 TensorFlow를 이용해 학습한 신경망의 구조와 학습 파라미터를 FLASH 메모리에 저장하고, SRAM에서 입출력 메모리 주소 교환 방식을 사용해, 마이크로컨트롤러의 메모리 구조에 최적화된 다층 신경망 알고리즘을 제안한다.

III. 본 론

3.1 FLASH 메모리 기반 학습 파라미터 저장

TensorFlow Lite를 사용해 Arduino 계열 마이크로컨트롤러에서 신경망 구현을 위한 과정은 그림 1과 같다. 우선 PC에서 TensorFlow를 이용해 신경망을 학습하고(step 1), TF convertor를 사용해 신경망 구조와 학습 데이터를 바이너리 파일로 저장한다(step 2). 이 바이너리 파일은 xxd 유틸리티를 사용해 16진수 형식의 배열로 이루어진 c 코드로 변환된다(step 3). 이때 16진수 배열은 그림 1처럼 const unsigned char* 타입으로 정의된다. 이렇게 생성된 코드와 API 라이브러리 코드는 컴파일/업로드 과정을 거쳐서 장치에서 실행된다(step 4). 이러한 방식은 컴파일 과정에서 전역 변수가 FLASH가 아닌 SRAM에 저장되기 때문에 SRAM 메모리가 작은 Arduino 계열의 마이크로컨트롤러에는 적합하지 않다.

FLASH에 학습 파라미터를 저장하기 위해서는 PROG 매크로를 사용해야 한다. 따라서 xxd 툴과 TF Convertor를 수정할 필요가 있다. 이를 위해 제안하는 xxd 유틸리티는 데이터 타입만을 변경해 (const PROG unsigned char*) C 코드가 생성되도록 했다.

제안하는 TF convertor에서는 메모리 액세스 속도가 느린 FLASH 메모리에 대한 반복적인 접근 횟수를 최소화한다. 이를 위해 그림 2처럼 신경망 구조와 학습 파라미터를 구조화해서 바이너리 파일을 생성한다. 우선 앞부분에 레이어 개수가 오고, 다음 각 레이어의 구조 정보(노드 수, 활성화 함수 종류)가 위치한다. 이후 개별 레이어의 학습 파라미터(가중치와 바이어스)가 위치한다.

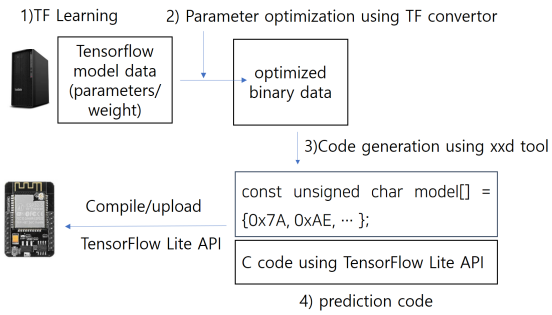

그림 1. TensorFlow Lite를 사용한 신경망 구현 과정
Fig. 1. Neural network implementation process using TensorFlow Lite

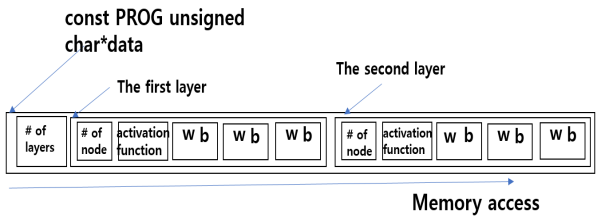

그림 2. 제안하는 방법에서 학습 파라미터 저장 구조
Fig. 2. Learning parameter storage structure in the proposed method

이를 통해 주소값을 1 씩 증가시켜 메모리 주소를 앞뒤로 이동하지 않고 1번의 스캔으로 레이어의 출력값을 계산할 수 있다. 이렇게 생성된 바이너리 파일은 제안하는 xxd 툴을 사용해 const PROG unsigned char* 타입으로 정의된 16진수 형식의 배열로 이루어진 C 코드로 변환된다.

3.2 입출력 메모리 주소 변경을 사용한 최적화

대개 바이너리 데이터를 FLASH 메모리에 적재한 후 신경망의 출력을 계산하기 위해 SRAM을 사용한다. 즉 그림 3(a)처럼 레이어 별로 메모리 공간을 사용한다. 따라서 두 개의 은닉층으로 이루어진 신경망은 (입력 노드 수 + 첫 번째 은닉층 노드 수 + 두 번째 은닉층 노드 수 + 출력층 노드 수) × sizeof(type) 만큼 SRAM 메모리 공간이 필요하다. 이러한 방식은 은닉층이 많아질수록 많은 메모리를 요구한다.

제안하는 방법은 그림 3(b)처럼 고정된 크기의 1차원의 메모리 공간을 할당하고 입출력 메모리 주소를 교환하는 방식을 사용해 신경망의 출력값을 계산한다. 필요한 메모리 공간의 크기는 레이어의 노드 수가 신경망의 구조를 설계할 때 결정되기 때문에 식 (1)처럼 각 레이어의 입력 노드 수와 출력 노드 수의 최댓값으로 계산할 수 있다.

$$\text{Memory size} = \max(Ii + Oi) \times \text{sizeof(type)} \quad (1)$$

여기서 Ii와 Oi는 i 번째 레이어의 입력 노드 수와 출력 노드 수를 의미한다.

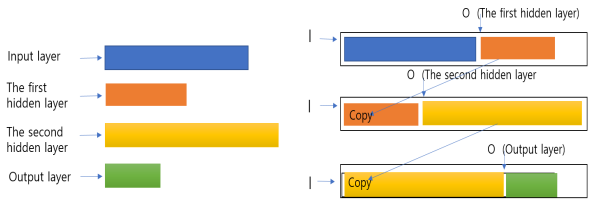

그림 3. 메모리 입출력 주소 교환을 통한 메모리 공간 최적화, (a) 일반적인 메모리 할당 방법 (b) 제안하는 방법을 이용한 메모리 할당 방법

Fig. 3. Memory space optimization through input/output address exchange, (a) Conventional memory layout, (b) Proposed memory layout

신경망의 출력을 계산하기 전에 메모리 할당이 필요하기 때문에 이 정보는 학습 후 계산되어 그림 2에서처럼 레이어 수 다음에 놓여진다.

제안하는 방법에서 각 레이어의 출력값을 계산하기 위해 현재 계산 중인 레이어의 입력 데이터 주소 I와 출력 데이터 주소 O를 사용해 현재 레이어의 출력을 계산한다(step 1). 다음으로 출력 영역을 I 영역에 복사한다(step 2). 그리고 출력 주소 O값을 I 영역의 다음 주소로 변경한다. 즉 현재 레이어의 출력값을 다음 레이어의 입력값으로 변경한다(step 3). 이러한 과정은 모든 레이어에 대해서 적용되어 신경망의 마지막 출력값을 계산한다. 그림 3(b)에서는 이 과정을 3번 반복해서 신경망의 출력값(녹색)이 계산되는 과정을 보여준다.

제안하는 방법에서는 2개의 은닉층이 있는 신경망 출력값을 계산하기 위해 3번의 출력값 계산과, 두 번의 메모리 복사가 필요하다. 이로 인해 기존 방법 대비 추가적인 메모리 복사 비용이 발생한다. 그러나 적은 메모리를 사용해 다층 신경망을 구성함으로써 마이크로컨트롤러에서 메모리 사용량을 최소화할 수 있다.

제안하는 방법은 신경망의 은닉층 수에 상관없이 1차원 메모리 공간을 할당하고, 입출력 메모리 주소를 변경하는 방식을 사용해 적은 메모리 공간을 효율적으로 사용한 다층 신경망을 구현했다.

IV. 실험 및 분석

4.1 실험 데이터셋

제안하는 방법의 효율성을 검증하기 위해서 3개의 공용 데이터셋을 사용해 메모리 사용량과 실행 속도를 비교, 분석한다. ESP32 계열과 Arduino Nano의 경우 카메라, 음성, 제스처 인식에 응용할 수 있어, 특징 수가 큰 데이터셋을 포함시켰다. 이를 표 2에 정리하였다.

4.2 가용 메모리 분석

다양한 종류의 마이크로컨트롤러에서 다층 신경망 적용을 위한 메모리 특성을 파악하기 위해 국내

외에서 널리 사용 중인 Arduino Uno, ESP8266, ESP32, Arduino Nano BLE 보드를 사용해 실험했다. Arduino 계열 하드웨어는 Raspberry Pi, Nvidia Jetson Nano처럼 OS가 존재하지 않지만 부트 코드와 하드웨어 구조를 지원하기 위한 필수 코드가 필요하다. 따라서 모든 FLASH 및 SRAM 메모리 공간을 알고리즘 구현에 사용할 수는 없다.

첫 번째 실험에서는 사용 가능한 메모리 공간을 측정했다. 구현 코드가 없는 setup() 함수와 loop() 함수로 이루어진 코드를 업로드 후, 메모리 사용량(사용 가능한 메모리 용량, 비율, 사용 메모리/전체 메모리)을 측정해 표 3에 정리하였다.

측정 결과를 보면 UNO는 부트 코드가 매우 작아 대부분의 메모리 영역을 응용프로그램에서 사용할 수 있다. 다만 가용 메모리가 31KB, 2KB로 매우 작다. 다양한 센서 기능이 포함된 ESP8266의 경우 필수 코드가 많아 응용프로그램에서 사용 가능한 메모리 공간이 ESP32, BLE 보드보다 상대적으로 적다. ESP32 보드의 경우, 응용프로그램에서 사용할 수 있는 FLASH와 SRAM 크기는 각각 2826KB, 291KB이다. 따라서 ESP32는 학습 데이터를 FLASH 메모리에 저장함으로써 데이터의 특징 수가 큰 멀티미디어 데이터를 분류할 수 있는 다층 신경망 적용이 가능함을 알 수 있다.

표 2. 실험에 사용된 공용 데이터셋
Table 2. Public datasets used in the experiement

		feature	#samples	#class
Digits[16]	handwritten numbers	64	1797	10
Fashion([17]	fashion item	784	60000	10
UCI HAR[18]	IMU data	561	10299	6

표 3. 기본 메모리 사용량 비교
Table 3. Basic memory usage comparison

	FLASH	SRAM
UNO	31KB 99%(444/32256)	2KB 100%(9/2048)
ESP8266	780KB 77%(246103/1044464)	49KB 61%(32244/82920)
ESP32	2826KB 92%(252179/3145728)	291KB 96%(14620/313060),
BLE	884KB 93%(77828/983040)	215KB 84%(42192/ 262144)

4.3 변환된 파라미터 바이너리 크기 비교

TensorFlow Lite와 제안하는 방법 모두 TensorFlow 프레임워크를 기반으로 PC에서 학습한 후, 신경망 구조와 학습 파라미터는 컨버터 툴을 사용해 바이너리 파일로 변환한다. 그리고 Arduino에서 사용할 수 있도록 xxd 툴로 c 코드로 변환한다. 이때 바이너리 파일의 크기는 학습 파라미터 수에 의해 결정되고, 학습 파라미터 수는 특징 수, 은닉층의 노드 수, 분류할 클래스 수에 따라 달라진다.

다양한 크기의 신경망에 따른 메모리 사용량 비교 실험을 위해 Digits 셋에는, 은닉층 16개 노드, 출력층 10개 노드로 신경망을 구성했다. Fashion 셋에 대해서는 두 개의 은닉층을 사용하고, 각각 32, 10개 노드로 신경망을 구성했다. UCI HAR 셋의 경우는 서로 다른 세 개의 신경망을 생성했다. UCI HAR-1은 한 개의 은닉층(노드 수 100개)과 출력층(노드 수 6개)으로 구성되었다, UCI HAR-2와 UCI HAR-3은 두 개의 은닉층과 하나의 출력층으로 구성되어 있고, 은닉층 노드의 수는 각각 150, 100, 6 그리고 150, 150, 6으로 충분히 크게 신경망을 설계했다.

바이너리 파일의 크기에 영향을 주는 또 다른 요소는 학습 파라미터 타입이다. TensorFlow Lite의 경우 변환과정 중 최적화 옵션을 사용할 수 있는데, 학습 파라미터를 int8 혹은 float16 타입으로 양자화하여 메모리 사용량과 속도를 개선할 수 있다. 제안하는 방법은 양자화 없이 float32 타입을 사용했다. 위에서 설명한 5개의 다층 신경망 모델로 학습했을 때 변환된 바이너리 파일의 크기를(byte) 표 4에 정리하였다.

표 4의 실험 결과를 보면, 양자화 없이 float32 타입을 사용할 때 TensorFlow Lite와 제안하는 방법의 바이너리 파일의 크기는 비슷하다. TensorFlow Lite의 경우 float16 타입으로 양자화할 때 바이너리 파일의 크기는 50% 수준으로 줄어들고, int8 타입으로 양자화할 경우 25% 수준으로 줄어든 것을 확인할 수 있다. int8 타입의 경우 float32 타입에 비해 25% 수준으로 바이너리 파일의 크기가 줄기 때문에 마이크로컨트롤러에서 다층 신경망을 구현할 때 매우 적합하지만, 4바이트 실수를 1바이트 실수로 변환할 경우 인식률이 크게 저하될 수 있다.

표 4. 바이너리 파일 크기 비교
Table 4. Comparison of binary file size

	TensorFlow lite			Proposed
	int8	float16	float32	float32
Digits (16,10)	3,296	4,400	6,180	4,861
Fashion (32,10)	27,968	52,816	103,056	101,821
UCI HAR-1 (100,6)	119,520	233,792	464,936	463,645
UCI HAR-2 (150,100,6)	193,184	379,952	756,360	754,653
UCI HAR-3 (150,150,6)	202,080	395,664	787,760	786,053

4.4 계산 속도 비교

제안하는 방법은 SRAM이 아닌 FLASH 메모리에 학습 파라미터를 저장한다. 두 메모리는 전자기적 특성과 메모리 주소 계산 방식이 달라 메모리 액세스 속도가 다르다. Arduino 계열 보드에서 FLASH 메모리에 접근할 때 전용 매크로를 사용해 주소를 계산하기 때문에 SRAM에 비해 메모리 접근 속도가 느릴 수 있다. 신경망의 출력값을 계산하는 과정을 고려해 SRAM과 SRAM 사이의 연산 시간과 SRAM과 FLASH 사이의 연산 시간의 차이를 측정했다. 두 메모리 변수를 곱해서 SRAM에 1,000,000번 누적하는 연산의 수행 속도를 측정해 여러 장치(CPU 스펙 표기)에서의 결과를 표 5에 정리하였다.

실험 결과를 보면 SRAM-SRAM이 SRAM-FLASH 연산보다 조금 더 빠르지만 큰 차이는 없어 보인다. 이는 FLASH 메모리에 학습 파라미터를 저장해도 신경망 출력 계산을 위한 비용은 증가하지 않음을 의미한다. ESP32 보드가 BLE 보드보다 5.6배 빠르기 때문에 신경망 알고리즘을 활용하기에 적합하다는 것도 확인할 수 있다.

계산 속도 비교에 관한 다음 실험으로 3개의 데이터 셋에서 1개 샘플 데이터에 대한 신경망 출력 속도를 비교했다. 현재까지 TensorFlow Lite를 공식적으로 지원하는 디바이스는 Nano BLE와 최근 지원을 시작한 ESP32이다. 본 논문에서는 ESP32에 대한 자료가 충분하지 않아 Nano BLE에서 비교 실험하고 결과를 표 6에 정리하였다.

실험 결과를 비교해 보면 제안하는 방법이 TensorFlow Lite를 사용한 방법에 비해 2배 정도 느리다.

표 5. FLASH 메모리와 SRAM 사이의 계산 속도 비교
Table 5. Comparison of speed for FLASH and SRAM memory

	SRAM-SRAM	SRAM-FLASH
UNO(16MHz) ATmega328P	1834ms	1865ms
ESP8266(80MHz) Xtensa single-core	193ms	200ms
ESP32(160MHz) Xtensa dual-core	34ms	34ms
BLE(64MHz) Arm Cortex-M4	193ms	195ms

표 6. 예측 시간 비교
Table 6. Comparison of prediction time

	TensorFlow lite	proposed
Digits	0.5ms	1ms
Fashion	6ms	11ms
UCI HAR	x	51ms/43ms

그러나 Fashion 셋에서 수행시간이 11ms로 실시간 응용이 가능하다. UCI HAR 셋은 특징 수가 561개로 TensorFlow를 사용해 학습할 경우, 학습 파라미터가 5000개 수준으로, Nano BLE 모델에서 SRAM 메모리 부족 문제로 컴파일이 불가능했다. 반면 제안하는 방법은 컴파일이 가능했고, 수행 시간은 51ms로 실시간 응용이 가능하다. 추가적으로 UCI HAR 셋에 대해서 ESP32에서 수행 시간을 측정했다. 43ms로 마이크로컨트롤러에서 다소 큰 데이터에 대해서도 실시간 예측이 가능함을 알 수 있다. 제안하는 방법의 수행 속도가 느린 이유는 신경망 알고리즘을 구현하는 과정에서 메모리 접근에 대한 최적화를 고려하지 않았기 때문으로 보인다. 이는 향후 개선이 필요하다.

4.5 메모리 사용량 비교

마지막 실험은 TensorFlow를 사용해 학습한 후, 마이크로컨트롤러에서 신경망 실행을 위해 TensorFlow Lite를 이용한 방법과 제안하는 방법의 FLASH와 SRAM의 메모리 사용량을 측정했다. ESP8266, ESP32, NanoBLE 보드에서 3개의 데이터셋에 대한 메모리 사용량을 측정하였다. 이전 실험에서와 같은 이유로 TensorFlow Lite는 Nano BLE에서만 실험하고, 제안하는 방법은 3개의 보드에서 모두 실험하고 표 7에 정리하였다.

표 7. 메모리 사용량 비교
Table 7. Comparison of memory usage

Board	Datasets	TensorFlow lite		proposed	
		FLASH	SRAM	FLASH	SRAM
ESP 8266	Digits			256,552 24%	32,436 39%
	Fashion			353,528 33%	32,436 39%
	UCI HAR-1			715,352 68%	32,436 39%
ESP 32	Digits			266,345 20%	1,542 4%
	Fashion			836,3305 27%	1,542 4%
	UCI HAR-1			725,129 55%	15,428 4%
Nano BLE	Digits	193,256 19%	69,312 26%	84,788 8%	42,192 16%
	Fashion	290,248 29%	166,304 63%	181,652 18%	42,192 16%
	UCI HAR-1	380,288 36%^{float16}	256,344 97%^{float16}	543,476 55%	42,192 16%

실험 결과를 보면 Nano BLE 보드에서 모든 데이터셋에 대해서 제안하는 방법이 FLASH와 SRAM을 더 적게 사용한다. 그리고 제안하는 방법이 FLASH 메모리에 학습 파라미터를 저장하기 때문에 사용하는 SRAM의 크기는 데이터셋의 크기에 영향을 받지 않음을 알 수 있다.

TensorFlow Lite의 경우 데이터셋이 클수록 SRAM 사용량이 많아져 Fashion 데이터셋에서 63%의 SRAM을 사용하고, UCI HAR-1에서는 SRAM 부족으로 컴파일이 불가능했다. float32 타입이 아닌 float16 타입으로 양자화했을 경우 컴파일이 가능했다. 그러나 FLASH 사용량은 36%로 적당했지만, SRAM 사용량은 97%로 매우 높아, 여러 기능이 복합적으로 실행되는 경우 적용이 어려울 수 있다.

반면 제안하는 방법은 UCI HAR-1에서 FLASH 55%, SRAM 16%만 사용해서 알고리즘 실행이 가능했다. 마이크로컨트롤러에서는 SRAM이 FLASH에 비해 메모리 용량이 작아, SRAM의 크기가 알고리즘의 사용 가능 여부를 결정하게 된다. 제안하는 방법은 SRAM이 아닌 FLASH 메모리를 사용해 학습 파라미터를 저장하기 때문에 특징 수가 다소 큰 데이터를 포함하고 있는 3개의 데이터셋에 대해서 모두 적용이 가능했다.

V. 결론 및 향후 과제

본 논문에서는 마이크로컨트롤러 기반의 AIoT를 구현하기 위해 SRAM이 아닌 FLASH 메모리에 학습 파라미터를 저장하는 다층 신경망 구조를 제안했다. 이를 통해 제스처 인식처럼 데이터의 특징이 큰 응용 분야에서도 실시간 적용이 가능한 것을 확인했다. 향후 CNN, RNN 같은 신경망으로의 확장과 마이크로컨트롤러 메모리 구조에 최적화된 신경망 알고리즘에 대한 연구가 필요해 보인다.

References

[1] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge Computing for the Internet of Things: A Case Study", in IEEE Internet of Things Journal, Vol. 5, No. 2, pp. 1275-1284, Apr. 2018. https://doi.org/10.1109/JIOT.2018.2805263.

[2] F. Wang, M. Zhang, X. Wang, X. Ma, and J. Liu, "Deep Learning for Edge Computing Applications: A State-of-the-Art Survey", in IEEE Access, Vol. 8, pp. 58322-58336, Mar. 2020. https://doi.org/10.1109/ACCESS.2020.2982411.

[3] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of Edge Computing and Deep Learning: A Comprehensive Survey", in IEEE Communications Surveys & Tutorials, Vol. 22, No. 2, pp. 869-904, Jan. 2020. https://doi.org/10.1109/COMST.2020.2970550.

[4] Luo, Chunjie, et al., "AIoT bench: towards comprehensive benchmarking mobile and embedded device intelligence", International Symposium on Benchmarking, Measuring and Optimization. Springer, Cham, pp 31-35, Dec. 2018.

[5] Dong, Bowei, et al., "Technology evolution from self-powered sensors to AIoT enabled smart homes", Nano Energy, Vol. 79, Jan. 2021. https://

doi.org/10.1016/j.nanoen.2020.105414.

[6] Zhang, Xinqian, et al., "Efficient Federated Learning for Cloud-Based AIoT Applications", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 40, No. 11, pp. 2211-2223, Nov. 2021. https://doi.org/10.1109/TCAD.2020.3046665.

[7] Benchmark Analysis of Jetson TX2, Jetson Nano and Raspberry PI using Deep-CNN Ahmet Ali Süzen, Burhan Duman, Betül Şen, 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)

[8] Wuttichai Vijitkunsawat and Peerasak Chantngarm, "Comparison of Machine Learning Algorithm's on Self-Driving Car Navigation using Nvidia Jetson Nano", 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology(ECTI-CON), Phuket, Thailand, Jun. 2021. https://doi.org/10.1109/ECTI-CON49241.2020.9158311.

[9] Vimal, S., et al., "Edge computing-based intrusion detection system for smart cities development using IoT in urban areas", Internet of things in smart Technologies for Sustainable Urban Development (2020), pp. 219-237. Apr. 2020. http://dx.doi.org/10.1007/978-3-030-34328-6_14.

[10] Kristian Dokic, Marko Martinovic, and Bojan Radisic, "Neural Networks with ESP32 - Are Two Heads Faster than One?", 6th Conference on Data Science and Machine Learning Applications (CDMA), Riyadh, Saudi Arabia, Mar. 2020. https://doi.org/10.1109/CDMA47397.2020.00030.

[11] Dokic, Kristian, Dubravka Mandusic, and Bojan Radisic, "Analysis of ESP32 SoC for Feed-Forward Neural Network Applications", International Conference Europe Middle East & North Africa Information Systems and Technologies to Support Learning. Springer, Cham, Marrakech, Morocco, pp. 165-175, Nov. 2019.

[12] Dokic, Kristian, "Microcontrollers on the Edge–Is ESP32 with Camera Ready for Machine Learning?", International Conference on Image and Signal Processing. Springer, Cham, Marrakesh, Morocco, pp. 213-220, Jun. 2020.

[13] Khadse, Rajratna, Nitin Gawai, and Bagwan M. Faruk, "Overview and comparative study of different microcontrollers", International Journal for Research in Applied Science & Engineering Technology (IJRASET) (2014): 311-315.

[14] David, Robert, et al., "Tensorflow lite micro: Embedded machine learning on tinyml systems", arXiv preprint arXiv:2010.08678, Oct. 2020.

[15] Singh Anubhav and Rimjhim Bhadani, "Mobile Deep Learning with TensorFlow Lite, ML Kit and Flutter: Build scalable real-world projects to implement end-to-end neural networks on Android and iOS", Packt Publishing Ltd, Apr. 2020.

[16] Breukelen, M., et al., "Handwritten digit recognition by combined classifiers", Kybernetika, Vol. 34, No. 4, pp. 381-386, 1998.

[17] Han Xiao, Kashif Rasul, and Roland Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms", CoRR abs/1708.07747, Aug. 2017.

[18] Anguita, Davide, et al., "Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine", International workshop on ambient assisted living. Springer, Berlin, Heidelberg, pp. 216-223, Dec. 2012.

저자소개

최 권 택 (Kwontaeg Choi)

2006년 2월 : 연세대학교 컴퓨터공학과(공학석사)
2011년 2월 : 연세대학교 컴퓨터공학과(공학박사)
2016년 3월 ~ 현재 : 강남대학교 소프트웨어응용학부 교수
관심분야 : 가상현실, 증강현실, 기계학습, 인공신경망, 컴퓨터비