IoT 컴퓨팅 환경을 위한 뉴로모픽 기반 플랫폼의 추론시간 단축

(Reduction of Inference time in Neuromorphic Based Platform for IoT Computing Environments)

김재섭*, 이승연**, 홍지만***
(Jaeseop Kim, Seungyeon Lee, Jiman Hong)

요 약

뉴로모픽 아키텍처는 스파이킹 신경망(SNN, Spiking Neural Network) 모델을 사용하여, 추론 실험을 통해 스파이크 값이 많이 누적될수록 정확한 결과를 도출한다. 추론 결과가 특정 값으로 수렴할 경우, 추론 실험을 더 진행해도 결과의 변화가 작아 소비 전력이 더 커질 수 있다. 특히, 인공지능 기반 IoT 환경에서는 전력 낭비는 큰 문제가 될 수 있다. 따라서 본 논문에서는 뉴로모픽 아키텍처 환경에서 추론 이미지 노출 시간을 조절하여 추론 시간을 단축함으로써 인공지능 기반 IoT의 전력 낭비를 줄이는 기법을 제안한다. 제안한 기법은 추론 정확도의 변화를 반영하여 다음 추론 이미지 노출 시간을 계산한다. 또한, 추론 정확도의 변화량 반영 비율을 계수 값으로 조절할 수 있으며, 다양한 계수 값의 비교 실험을 통해 최적의 계수 값을 찾는다. 제안한 기법은 목표 정확도에 해당하는 추론 이미지 노출 시간은 선형 기법보다 크지만 최종 추론 시간은 선형 기법보다 적다. 제안한 기법의 성능을 측정하고 평가한 결과, 제안한 기법을 적용한 추론 실험이 선형 기법을 적용한 추론 실험보다 최종 노출 시간을 약 90% 단축할 수 있음을 확인한다.

■ 중심어 : 뉴로모픽 아키텍처 ; 스파이킹 신경망 ; 추론 시간 ; IoT 컴퓨팅 환경

Abstract

The neuromorphic architecture uses a spiking neural network (SNN) model to derive more accurate results as more spike values are accumulated through inference experiments. When the inference result converges to a specific value, even if the inference experiment is further performed, the change in the result is smaller and power consumption may increase. In particular, in an AI-based IoT environment, power consumption can be a big problem. Therefore, in this paper, we propose a technique to reduce the power consumption of AI-based IoT by reducing the inference time by adjusting the inference image exposure time in the neuromorphic architecture environment. The proposed technique calculates the next inferred image exposure time by reflecting the change in inference accuracy. In addition, the rate of reflection of the change in inference accuracy can be adjusted with a coefficient value, and an optimal coefficient value is found through a comparison experiment of various coefficient values. In the proposed technique, the inference image exposure time corresponding to the target accuracy is greater than that of the linear technique, but the overall power consumption is less than that of the linear technique. As a result of measuring and evaluating the performance of the proposed method, it is confirmed that the inference experiment applying the proposed method can reduce the final exposure time by about 90% compared to the inference experiment applying the linear method.

■ keywords : Neuromorphic Architecture ; Spiking Neural Networks ; Inference Time ; IoT Computing Environment

* 준회원, 숭실대학교 일반대학원 컴퓨터학과 대학원생
** 준회원, 숭실대학교 일반대학원 컴퓨터학과 대학원생
*** 종신회원, 숭실대학교 컴퓨터학부 교수
이 논문은 2016년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임 (No. NRF-2016R1D1A1B01016073, 가상화 기반 오픈 게이트웨이 플랫폼과 오케스트레이션 보안 서비스 프레임워크 설계 및 구현).

접수일자 : 2022년 03월 08일
수정일자 : 2022년 03월 21일
게재확정일 : 2022년 03월 24일
교신저자 : 홍지만 e-mail : jiman@ssu.ac.kr

Ⅰ. 서 론

중앙처리장치(CPU)와 그래픽 처리 장치(GPU)의 조합으로 구성된 인공지능(AI, Artificial Intelligence) 아키텍처는 병렬 연산을 수행함과 동시에 연산 결과를 빠른 속도로 메모리에 저장해야 하기 때문에 과도한 전력을 소모한다[1-4].

또한, 기존 AI 아키텍처는 유연성이 낮아 디자인된 알고리즘으로만 연산할 수밖에 없는 단점이 있다. 따라서 3세대 AI 아키텍처로 알려진, 인간의 뇌를 모방한 폰 노이만 방식의 뉴로모픽 아키텍처가 제안되었다[5]. 뉴로모픽 아키텍처는 비록 기술 성숙도가 낮고 범용성은 떨어지지만 연산 성능과 소비 전력 효율은 AI 아키텍처 가운데서 사물인터넷(IoT, Internet of Things) 컴퓨팅에 가장 적합한 것으로 알려져 있다[6].

뉴로모픽 아키텍처에서 신경은 뉴런과 뉴런 사이를 연결하는 시냅스로 이루어져 있고, 뉴로모픽 칩에서 뉴런의 역할은 코어가, 시냅스의 역할은 메모리 칩이 담당하고 있다[7]. 시냅스가 반복적으로 뉴런들의 연결을 조절하며 데이터를 병렬로 처리하여 저전력으로 대용량의 데이터를 처리하는데 용이하다[8].

본 논문에서는 효율적인 IoT 컴퓨팅 환경을 구축하기 위하여 뉴로모픽 아키텍처의 추론 시간을 줄이고 소비 전력 측면에서 효율성을 높이는 방법을 제안한다. 이를 위하여 Nengo 프레임워크[9,10]에서 제공하는 NengoLoihi 시뮬레이터를 이용하여 먼저, MNIST(Modified National Institute of Standards and Technology database)[11] 손글씨 숫자 이미지 추론 실험을 진행하여 추론 정확도의 성능 지표를 추출한다. 제안한 기법은 추출된 성능 지표를 기반으로, 다양한 계수 값 조합의 실험 결과 비교를 통해 높은 정확도와 낮은 최종 노출 시간을 보장하는 계수 값을 찾아 사용한다. MNIST 추론 이미지의 적합한 노출 시간을 얻고 이를 통하여, 목표한 정확도를 보장한다.

본 논문의 구성은 다음과 같다. 먼저, 선형 기법을 적용한 추론 실험을 진행하고, 최적에 가까운 추론 이미지 노출 시간을 찾는 데 필요한 시간 한계를 확인한다. 또한, 선형 기법과 비교하여 제안한 기법의 성능을 평가한다. 마지막으로 결론을 서술한다.

Ⅱ. 본 론

본 장에서는 추론 이미지 노출 시간에 따른 추론 정확도를 측정하는 실험을 진행하고, 실험 결과를 바탕으로 추론 시간 단축 기법을 제안한다. 선형 기법은 추론 이미지 노출 시간을 0.001초부터 0.1초까지 100번 동안 선형으로 증가시키는 방법이다. 추론 이미지 노출 시간은 추론에 사용되는 이미지를 뉴로모픽 아키텍처에 노출하는 시간이다. 실험을 통해 선형 기법과 비교하여 제안 기법이 최종 노출 시간을 단축할 수 있음을 확인한다.

1. 추론 시간과 전력 효율성

가. 실험 환경

뉴로모픽 아키텍처에서 SNN을 사용한 추론 실험은 호스트 기기와 뉴로모픽 아키텍처의 통신으로 이루어진다. 호스트 기기는 학습 모델 동작을 위해 입력받은 손글씨 데이터를 뉴로모픽 아키텍처가 지원하는 뉴런 수에 적합하게 크기를 변환하여 뉴로모픽 아키텍처에 전송한다. 그 다음, 뉴로모픽 아키텍처는 적절한 학습 모델을 구동하여 호스트 기기로부터 전달받은 데이터를 학습하고, 인식 결과를 호스트 기기에 전달한다.

추론 실험에서는 총 60,000개의 MNIST 이미지를 학습하고, 테스트 이미지 10,000개 중 500개의 이미지를 대상으로 실험을 진행할 때 0부터 9까지 제대로 분류하는지 확인한다. 추론 실험 시간의 단위는 초(second)이며, 실험 시간은

입력 데이터의 개수와 추론 이미지 노출 시간의 곱으로 정의한다. 제일 작은 추론 이미지 노출 시간인 0.001초부터 100번의 실험을 진행한다. 테스트 데이터는 숫자 구성의 영향을 방지하고자 동일한 개수의 숫자들 집합으로 실험을 진행한다. 인식 결과는 길이가 10인 리스트 형태로 반환되며, 숫자 데이터와 인식한 결과를 비교한다.

나. 실험 결과

그림 1은 선형 기법을 적용한 추론 실험 결과를 보여준다. 추론 이미지 노출 시간이 증가할수록 정확도가 점점 증가하는 것을 확인할 수 있다. 그림 1은 500개의 숫자 이미지를 분류하는 추론 실험을 진행한 결과를 보여준다. 추론 이미지 노출 시간이 0.057초에 도달했을 때, 93.40%의 정확도를 보인다. 추론 이미지 노출 시간을 0.057초부터 0.1초까지 증가시켜 추론 실험을 진행할 때, 추론 정확도는 약 2.6% 증가했으며, 낮은 증가량을 보인다. 추론 정확도의 변화량이 적은 것은 해당 추론 이미지 노출 시간까지 스파이크 값이 누적되어 특정 결과에 수렴하기 때문에 나타난 결과이다.

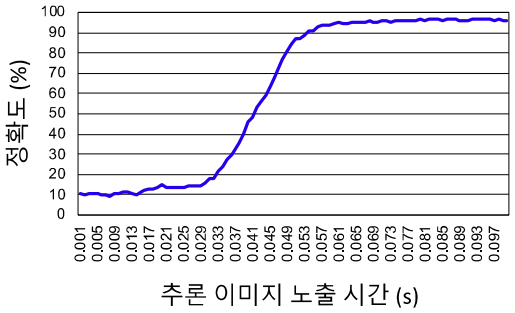


그림 1. 선형 기법의 정확도 변화

추론 이미지 노출 시간은 시뮬레이터가 시뮬레이션하는 시간을 표현한 것으로 실제 실행 시간과는 차이가 있다. 각 실험에서 0.057초 이후의 추론 실험은 추론 정확도 변화에 큰 영향을 주지 않는 것을 알 수 있다. 따라서 0.057초 이후에 진행하는 추론 실험은 전력 낭비를 일으킨다.

선형 기법을 적용한 실험 결과, 각 500개 입력 데이터에 대해 0.057초가 적은 추론 이미지 노출 시간으로 높은 추론 정확도를 얻을 수 있는 가장 효율적인 추론 이미지 노출 시간임을 알 수 있다. 선형 기법으로 실험하였을 때, 500개 입력 데이터에 대해 0.057초에 도달할 때까지 소요된 추론 이미지 노출 시간의 총합은 1.653초이다. 결과적으로 효율적인 추론 이미지 노출 시간을 찾기 위해 소요되는 시간을 줄인다면 추론을 위한 실행 시간이 전체적으로 감소하므로 전력 낭비가 줄어들 수 있다.

2. 추론 시간 단축 기법

그림 2는 그림 1의 정확도 변화량을 보여준다. 그림 1의 정확도 변화량은 세 구간으로 구분할 수 있으며, 이와 마찬가지로 그림 2도 세 구간으로 구분할 수 있다.

그림 2에서 0.001초부터 약 0.028초 사이의 추론 이미지 노출 시간을 나타내는 첫 번째 구간은 스파이크 값이 누적되기 시작하는 구간으로 추론 정확도의 변화량은 많지 않다. 이와 달리, 약 0.029초부터 0.057초 사이의 추론 이미지 노출 시간을 나타내는 두 번째 구간은 스파이크 값으로 결과가 도출되기 시작하는 구간으로 추론 정확도의 변화량이 많다. 마지막으로, 0.058초 이후의 추론 이미지 노출 시간을 나타내는 세 번째 구간은 누적된 스파이크 값이 특정 결과로 수렴하면서 추론 정확도의 변화량은 적다.

따라서 추론 이미지 노출 시간에 따라 변화하는 추론 정확도의 특성을 반영하면 추론 이미지 노출 시간을 찾기 위한 소요 시간을 줄일 수 있다. 본 논문에서는 IoT 컴퓨팅 환경을 위한 뉴로모픽 아키텍처에서 추가적인 추론으로 인한 전력 낭비를 줄이기 위해 추론 이미지 노출 시간을 조절하여 추론 시간을 줄이는 기법을 제안한다.

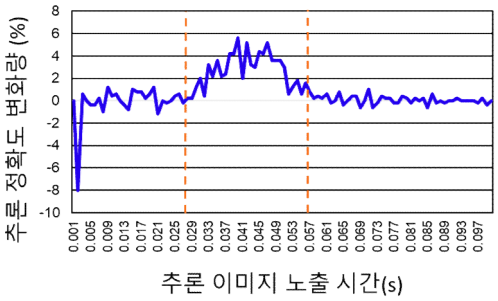


그림 2. 선형 기법의 정확도 변화량

표 1. 수식에 사용되는 표기법

표기법	설명
t_n	n 번째 추론 실험의 추론 이미지 노출 시간
TC_n	n 번째 추론 실험의 추론 정확도
$G(n)$	n 번째 추론의 추론 정확도 변화량을 계산하는 함수
δ	$G(n)$ 에 대한 계수 값
θ	$TC_n + G(n)$ 에 대한 계수 값

표 1은 수식에 사용된 표기법과 설명을 보여준다. n번째 추론 정확도 변화량을 나타내는 $G(n)$은 n번째와 $n-1$번째 추론 정확도의 차에서 n번째와 $n-1$번째 추론 이미지 노출 시간의 차를 나눈 값으로 수식 1과 같이 표현할 수 있다.

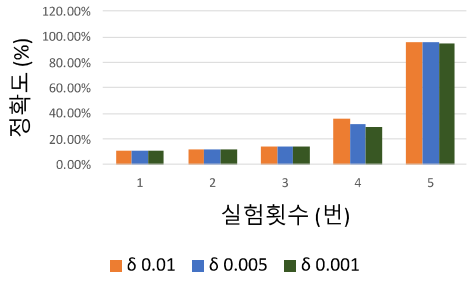
$$G(n) = \frac{TC_n - TC_{n-1}}{t_n - t_{n-1}} \quad (1)$$

$n+1$ 번째 추론 실험의 추론 이미지 노출 시간을 나타내는 t_{n+1}은 n번째 추론 정확도 TC_n와 추론 정확도 변화량 $G(n)$을 사용하여 표현할 수 있다. δ 값으로 추론 정확도 변화량 $G(n)$의 값을 조절하고, θ 값으로 추론 정확도와 추론 정확도 변화량을 더한 $TC_n + G(n)$ 값을 조절하여 수식 2로 표현할 수 있다.

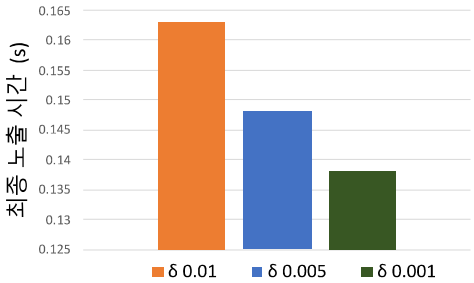
$$t_{n+1} = t_n + (TC_n + G(n) \times \delta) \times \theta \quad (2)$$

그림 3은 θ 값을 0.1로 고정하고 δ 값을 변화시켰을 때, 정확도 변화와 최종 노출 시간 변화를 보여준다. 시험횟수별 정확도는 δ 값 변화에 따라 차이가 크지 않지만, δ 값이 0.01일 때의 최종 노출 시간은 δ 값이 0.005, 0.001일 때에 비해 약 10% 이상이 크다. δ 값이 0.01일 때의 최종 노출 시간이 가장 크고, δ 값이 0.001일 때에는 계수 값이 작아 변화량을 반영하는 비율이 적어 데이터 개수가 많아지면 충분한 정확도에 이르기까지 많은 시간이 소요될 수 있다. 그러므로 본 논문에서는 δ 값을 0.005로 사용한다.

그림 4는 δ 값을 0.005로 고정하고 θ 값을 변화시켰을 때, 정확도 변화와 최종 노출 시간 변화를 보여준다. 시험횟수별 정확도는 θ 값이 0.08일 때를 제외하면, 나머지는 시험횟수 5회 이내에 목표치인 90% 이상에 다다른다. θ 값이 0.1일 때, 0.08을 제외하고 가장 적은 최종 노출 시간을 가지므로, 본 논문에서는 θ 값을 0.1로 사용한다.

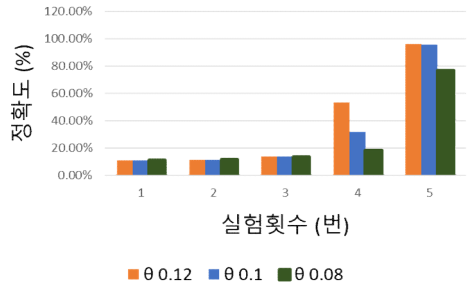


(a) δ 값에 따른 정확도 변화

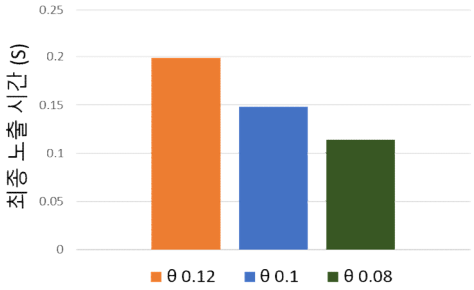


(b) δ 값에 따른 최종 노출 시간 변화

그림 3 δ 값에 따른 정확도 및 최종 노출 시간 변화



(a) θ 값에 따른 정확도 변화



(b) θ 값에 따른 최종 노출 시간 변화

그림 4 θ 값에 따른 정확도 및 최종 노출 시간 변화

2. 실험 및 평가

가. 실험 환경

본 논문에서 추론 실험에 사용한 환경은 표 3과 같다. 실험에서는 x86 기반의 SBC(Single-Board Computer) 개발 보드인 LattePanda Alpha 864s[12]를 호스트 기기로 사용한다. 제안한 기법의 성능을 평가하기 위하여 NengoLoihi 시뮬레이터[13]를 기반으로 MNIST 손글씨 숫자 이미지 응용을 사용하여 목표 정확도에 도달하기 위해 제안 기법과 선형 기법의 최종 노출 시간을 비교하고 분석한다. 제안 기법을 적용한 실험에서 설정한 목표 정확도는 선형 기법을 적용한 실험에서 변화량이 감소하기 시작했던 추론 정확도인 90%로 설정한다. 실험에 사용된 δ, θ 값은 앞에서 다양한 계수 값 비교 실험을 통해 얻은 표2와 같다.

표 2. 추론 실험에 사용하는 제안 기법의 계수 값

계수	값
δ	0.005
θ	0.1

표 3. 추론 실험 환경

카테고리		세부사양
LattePanda Alpha 864s	CPU	1.1-3.4GHz Dual-Core Four-Thread
	Memory	8GB LPDDR3
	Network Interface	1 Gigabit Ethernet
	OS	Ubuntu 16.04.7 LTS

나. 실험 결과

제안 기법을 적용한 실험 결과는 표 4와 같다. 실험 결과는 제안 기법이 선형 기법보다 훨씬 적은 추론 실험으로 비슷한 추론 정확도를 도출할 수 있음을 보여준다. 표 4는 입력 데이터가 500개일 때, 제안한 기법을 적용한 결과를 보여준다. 총 57번의 추론 실험을 진행한 선형 기법과는 달리, 제안 기법을 적용한 실험은 5번의 실험으로 비슷한 실험 결과를 도출할 수 있음을 보인다. 또한, 총 1.653초의 최종 노출 시간이 소요되었던 선형 기법보다, 제안 기법은 총 0.157초로 적은 최종 노출 시간이 소요된다.

표 4. 테스트 이미지 500개로 제안 기법을 적용한 실험 결과
[시간 단위 : second]

실험 반복 횟수	현재 추론 이미지 노출 시간	추론 정확도 (%)	다음 추론 이미지 노출 시간
1	0.001	11.00	0.012
2	0.012	11.40	0.024
3	0.024	13.40	0.038
4	0.038	36.00	0.082
5	0.082	96.40	0.185
합계	0.157		

표 5는 본 논문에서 각각의 기법을 적용하여 호스트 기기에서 목표 정확도로 설정한 90% 이상의 추론 정확도를 도출할 때의 추론 이미지 노출 시간을 보여준다. 설정한 추론 정확도를 얻는 추론 이미지 노출 시간은 제안 기법보다 선형 기법이 더 짧은 것을 확인할 수 있다.

표 5. 추론 이미지 노출 시간 비교

[시간 단위 : second]

입력 데이터 개수	선형 기법으로 측정한 추론 이미지 노출 시간	제안 기법으로 측정한 추론 이미지 노출 시간
500	0.057	0.082

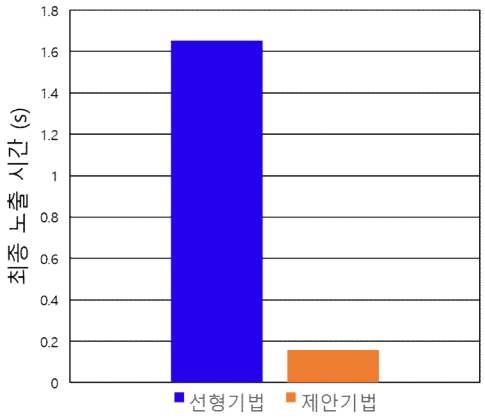


그림 5 기법 별 최종 노출 시간 변화

그림 5은 선형 기법과 제안 기법을 적용하여 진행한 추론 실험에서 Intel Loihi의 최종 노출 시간을 비교한 그래프이다. 선형 기법으로 진행한 추론 실험과 비교하여 제안 기법으로 진행한 추론 실험이 설정한 추론 정확도를 얻기까지 소요된 최종 노출 시간이 90% 이상 적은 것을 확인할 수 있다.

Ⅲ. 결 론

본 논문에서는 추론 정확도의 변화량을 반영하여 소요되는 최종 노출 시간을 단축시키는 기법을 제안하였다. 먼저, 순차적으로 추론 이미지 노출 시간을 증가시키는 선형 기법으로 추론 정확도를 도출하였다. 이 과정에서 추론 정확도가 특정 값으로 수렴하는 구간은 추론 정확도 변화량이 크지 않은 것을 확인하였다. 또한, 목표한 추론 정확도를 도출하는 추론 이미지 노출 시간과 최종 노출 시간을 비교하였다.

제안 기법은 현재 추론 실험의 추론 이미지 노출 시간과 추론 정확도 변화량을 반영하였다. 여러 추론 실험을 통해 최적의 추론 이미지 노출 시간과 추론 정확도 변화 반영 비율을 찾고, 다음 노출 시간을 계산하는 수식의 계수에 적용하였다. 제안 기법을 적용하여 추론 정확도를 측정한 결과, 최종 노출 시간이 선형 기법보다 더 짧은 시간에 비슷한 추론 정확도를 도출할 수 있음을 알 수 있었다. 제안 기법은 추론 정확도를 도출하는 순간의 추론 이미지 노출 시간은 선형 기법보다 상대적으로 소요될 수 있다. 하지만, 추론 정확도에 도달하는 과정을 포함한다면, 제안 기법의 최종 노출 시간은 약 90% 이상 줄일 수 있음을 확인하였다.

References

[1] Younggwan Kim, Jusuk Lee, Ajung Kim, Jiman Hong, "Load Balancing Scheme for Machine Learning Distributed Environment," *Smart Media Journal*, Vol.10, No.1, pp. 25-31, 2021.

[2] Sun Park, Jongwon Kim, "Red Tide Algea Image Classification using Deep Learning based Open Source," *Smart Media Journal*, Vol. 7, No. 2, pp34-39, 2018.

[3] Seo jeong Kim, Jae Su Lee, Hyong Suk Kim, "Deep learning-based Automatic Weed Detection on Onion Field," *Smart Media Journal*, Vol. 7, No.3, pp16-21, 2018.

[4] Hayun Lee, Dongkun Shin, "Performance and Energy Comparison of Different BLAS and Neural Network Libraries for Efficient Deep Learning Inference of ARM-baased IoT Devices," *Journal of KIISE*, Vol. 46, No. 3, pp. 219-227, Mar. 2019.

[5] Boseon Hong, Bongjae Kim, "Hardware Technology Trends for Neuromorphic Computing," *Communications of the Korean Institute of Information Scientists and*

Engineers, Vol 38, No 2, pp. 32-39, Feb. 2020.
[6] Seoyeon Kim, Young-Sun Yun, Jinman Jung, "Design of a Framework for supporting Neuromorphic Hardware in IoT platform," *Communications of the Korean Institute of Information Scientists and Engineers*, Vol 38, No 2, pp. 51-57, Feb. 2020.
[7] Keon Myung Lee, "Behavior of Spiking Neuron Models and Learning of Spiking Neural Networks," *Communications of the Korean Institute of Information Scientists and Engineers*, Vol 38, No 2, pp. 8-19, Feb. 2020.
[8] Jangsaeng Kim, Dongseok Kwon, Sung Yun Woo, Won-Mook Kang, Soochang Lee, Seongbin Oh, Chul-Heung Kim, Jong-Ho Bae, Byung-Gook Park, and Jong-Ho Lee, "Hardware-based spiking neural network architecture using simplified backpropagation algorithm and homeostasis functionality," *Neurocomputing*, Vol. 428, pp. 153-165, Mar. 2021.
[9] Trevor Bekolay, James Bergstra, Eric Hunsberger, Travis DeWolf, Terrence Stewart, Daniel Rasmussen, Xuan Choo, Aaron Voelker, and Chris Eliasmith, "Nengo: a Python tool for building large-scale functional brain models," *Frontiers in Neuroinformatics*, Vol 7, No 48, pp. 1-13, Jan. 2014.
[10] Terrence C Stewart, "A technical overview of the neural engineering framework," *Centre for Theoretical Neuroscience technical report,* University of Waterloo, Canada, Vol 110, 2012.
[11] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, 2278-2323, Nov. 1998.
[12] LattePanda Alpha 864s(2021). https://www.lattepanda.com/products/lattepanda-alpha-864s.html (accessed Mar., 02, 2022).
[13] NengoLoihi(2022). https://www.nengo.ai/nengo-loihi/v1.1.0/index.html (accessed Feb., 25, 2022).

저 자 소 개

김재섭(준회원)

2021년 숭실대학교 컴퓨터학부 학사 졸업.
2021년~현재 숭실대학교 컴퓨터학과 석사 재학.

<주관심분야 : 운영체제, 임베디드 시스템 >

이승연(준회원)

2020년 숭실대학교 컴퓨터학부 학사 졸업.
2022년 숭실대학교 컴퓨터학과 석사 졸업.

<주관심분야 : 운영체제, 임베디드 시스템 >

홍지만(종신회원)

2003년 서울대학교 컴퓨터공학과 박사 졸업.
2004년~2007년 광운대학교 컴퓨터공학과 교수
2007년~현재 숭실대학교 컴퓨터학부 교수.

<주관심분야 : 운영체제, 임베디드 시스템 >