2024년도

학사학위논문

서로게이트 기울기 기반의 스파이크 신경망 구현 및 성능 분석

Implementing and Performance Analyzing a Spiking Neural Network based on Surrogate Gradient

2024년 11월 12일

순천향대학교 공과대학

컴퓨터공학과

이인규

서로게이트 기울기 기반의 스파이크 신경망 구현 및 성능 분석

Implementing and Performance Analyzing a Spiking Neural Network based on Surrogate Gradient

지도교수 이 상 정

이 논문을 공학사 학위 논문으로 제출함

2024년 11월 12일

순천향대학교 공과대학
컴퓨터공학과
이인규

이 인 규의 공학사 학위 논문을 인준함

2024년 11월 12일

심 사 위 원 홍 인 식 인

순천향대학교 공과대학

컴퓨터공학과

초 록

본 논문은 인간 뇌에 존재하는 뉴런의 정보 전달 방식을 모방한 스파이크 신경망을 연구하며, 스파이크 신경망의 개념과 LIF(Leaky Integrate and Fire) 모델에 대해 설명한다. 스파이크 신경망의 이진 특성에 의해 발생하는 기울기 소실 문제를 해결하기 위해 서로게이트 기울기를 사용했으며, LIF 모델을 순환 신경망에 대응시킨다. 그리고 파이토치(PyTorch) 라이브러리를 통해 스파이크 신경망을 구현한다.

구현된 스파이크 신경망을 Fashion-MNIST, Spiking Heidelberg Digits(SHD), Tactile Braille Letters(TBL) 데이터 세트에 적용하고, 각 데이터에 최적화된 파라미터 조합을 도출한다. 파라미터로는 총 네 가지로 은닉층 노드 수, 시간 단계, 서로게이트 기울기 경사, 정규화 손실률을 사용하였고, 파라미터가 스파이크 신경망의 학습 정확도와 실행 시간에 미치는 영향을 분석한다.

분석 결과, 각 데이터마다 최적의 파라미터 조합은 달랐으며, Fashion-MNIST 데이터에서는 은닉층 노드 수와 정규화 손실률이 정확도에 큰 영향을 끼치고, SHD 데이터에서는 시간 단곗값의 증가가 과적합에 큰 영향을 주었다. 또한 TBL 데이터에서는 낮은 시간 단계와 높은 은닉층 노드 수가 학습 시간을 줄이는 데에 영향을 주었다.

본 논문에서 제시한 최적의 파라미터 조합은 데이터 세트별 약 2~4% 정도 정확도 향상을 이뤄내었다. 결론적으로, 스파이크 신경망은 특정 데이터에 최적화된 파라미터 조합의 설정은 성능 향상에 도움을 주며, 더욱 광범위한 파라미터 조합 분석을 통해 더 높은 정확도 향상을 이뤄내는 것이 고려되어야 한다.

키워드: 스파이크 신경망, 서로게이트 기울기, 파라미터 성능 분석

ABSTRACT

This paper studies spiking neural networks, which mimic the information transfer of neurons in the human brain, and explains the concept of Spiking neural networks and the Leaky Integrate and Fire (LIF) model. To solve the gradient vanishing problem caused by the binary nature of Spiking neural networks, we use Surrogate gradients, and the LIF model corresponds to a recurrent neural network. The Spiking neural network is then implemented using the PyTorch library.

We apply the implemented Spiking neural network to the Fashion-MNIST, Spiking Heidelberg Digits (SHD), and Tactile Braille Letters (TBL) datasets and derive the optimal parameter combination for each data set. We use four parameters: number of hidden layer nodes, time step, Surrogate gradient scale, and normalized loss rate, and analyze the impact of the parameters on the learning accuracy and running time of the Spiking neural network.

The results show that the optimal parameter combination is different for each data, and the number of hidden layer nodes and normalized loss rate have a significant impact on the accuracy in Fashion-MNIST data, the increase of time step value has a significant impact on overfitting in SHD data, and the low time step and high number of hidden layer nodes have a significant impact on reducing the learning time in TBL data.

The optimal parameter combinations presented in this paper resulted in accuracy improvements of about 2-4% for each dataset. In conclusion, setting the optimal parameter combinations for specific data helps to improve the performance of spiking neural networks, and more extensive parameter combination analysis should be considered to achieve higher accuracy improvements.

Keywords: Spiking Neural Networks, Surrogate Gradient, Parameter performance analysis

차 례

그 림 차 례

표 차 례

수 식 차 례

제 1 장 서 론

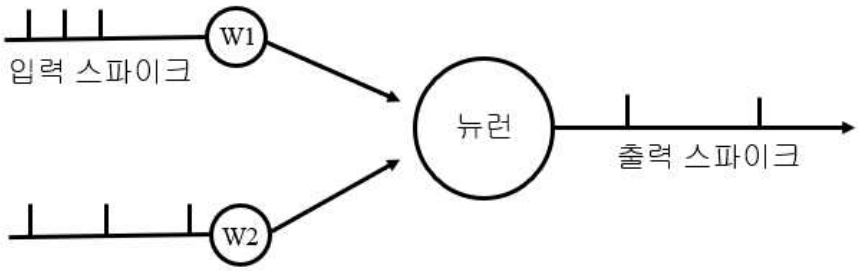
1940년대 이후 컴퓨터 과학자들은 인간 뇌의 생물학적인 특성을 모방하여 인공신경망(Artificial Neural Network) 모델을 연구하였고, 오늘날의 인공신경망 모델들은 수학적 함수 예측 및 분류 분야 등 많은 분야에서 뛰어난 성능을 보이며 사용되고 있다. 이러한 신경망들은 역전파 알고리즘을 통해 학습 과정에서의 계산 속도가 빨라졌고, GPGPU 방식을 통해 컴퓨터의 데이터 연산 속도를 크게 상승시키는 등 인공신경망이 등장한 이후로도 많은 후속 연구가 진행되었다. 그리고 신경망의 에너지 효율을 개선하기 위한 해결책으로 생물학적 뉴런의 정보 전달 방식을 모방한 스파이크 신경망(Spiking Neural Network)이 개발되었다.

본 논문에서는 스파이크 신경망과 신경망의 주요 역할이 되는 LIF(Leaky Integrate and Fire) 모델을 설명한다. 그리고 스파이크 신경망에서 스파이크의 특징에 의해 발생하는 문제인 기울기 소실 문제를 해결하는 방법을 설명하고, 이미지 데이터, 음성 데이터, 센서 데이터를 스파이크 신경망에 적용하는 방법과 각 데이터에 맞는 파라미터 조합을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 생물학적 개념 중 하나인 LIF 모델을 RNN(Recurrent Neural Network)에 대응(Mapping)하는 방법을 설명하며, 3장에서는 스파이크 신경망의 기울기 소실 문제를 해결할 방안 중 하나인 서로게이트 기울기(Surrogate gradient)를 설명한다. 4장에서는 서로게이트 기울기를 다양한 데이터 세트에 적용한 사례를 설명하고, 5장에서는 스파이크 신경망의
파라미터인 은닉층 노드 개수, 시간 단계 개수, 서로게이트 기울기 경사, 정규화 손실률을 그리드 서치(Grid Search) 알고리즘을 통해 찾는다. 그리고 마지막
6장에서는 결론 및 향후 과제를 제시하며 논문의 끝을 맺는다.

제 2 장 이론적 배경

2.1 스파이크 신경망 모델



[그림 1] 스파이크 신경망의 데이터 처리 흐름

스파이크 신경망(Spiking Neural Network)은 인공신경망(Artificial Neural Network)의 한 종류로 인간 뇌의 신경망이 정보를 처리하는 방식을 모방한 네트워크를 의미한다. 기존 RNN(Recurrent Neural Network)이나 CNN(Convolutional Neural Network) 같은 네트워크들은 숫자로 된 값을 주고받지만, 스파이크 신경망은 스파이크 발생 여부(발생 또는 미발생)를 주고받는다는 특징이 있다. 이 특징으로 많은 양의 연산이 필요한 기존 신경망 네트워크들과 달리 스파이크 신경망은 스파이크의 발생 여부만 계산하기 때문에 낮은 전력과 적은 연산량으로 학습이 가능하다는 장점이 있다. 하지만 스파이크 신경망은 다른 신경망에 비해 학습 속도가 느리다는 단점이 있고, 이는 모델의 효율적인 학습을 위해 반드시 개선되어야 한다. 또 스파이크 신경망은 기계학습이 목적이

아닌 실제 인간 또는 생물 뉴런의 행동을 수학적으로 모델링하기 위해 만들어졌으며, 이는 스파이크 신경망이 기계학습 분야뿐만 아니라 실제 인간 뇌의 학습 방법을 연구하는 분야 등으로 확장될 수 있음을 의미한다.

2.2 Hodgkin-Huxley 모델과 LIF 모델

모델이란 “자연현상의 동작 원리를 수학적인 표현으로 추상화하는 과정”이라고 정의한다. 스파이크 신경망을 연구하는 컴퓨터 과학자들은 인간의 뇌가 뉴런과 시냅스를 이용해 학습하는 현상을 수학적으로 가공하고자 하였고, 이산(binary) 스파이크의 발생 시간을 활용해 데이터를 처리하는 스파이크 신경망이 등장했다.

현재는 스파이크 신경망의 다양한 모델들이 존재하는데, 현재 뉴런의 활동 전위(action potential)를 가장 잘 묘사한 모델은 1952년 등장한 Hodgkin-Huxley 모델이다. Hodgkin-Huxley 모델은 뉴런의 활동 전위(action potential)가 어떻게 시작되고 전파되는지 설명하며, 스파이크 신경망을 응용하는 연구 분야에서 주로 Hodgkin-Huxley 모델에서 파생된 LIF(Leaky Integrate and Fire) 모델을 채택하고 있다.

LIF 모델은 뉴런의 네 가지 주요 동작 원리 4가지를 추상화한다.

1. 스파이크 전달: 이전 시냅스 뉴런(pre-synaptic neurons)은 스파이크를 생성하여 다음 시냅스 뉴런(post-synaptic neurons)에 스파이크를 전달한다.

2. 막 전위 전하 저장: 뉴런(neurons)은 이전 시냅스 뉴런(pre-synaptic neurons)에서 전달된 이온(ion)을 뉴런에 저장하여 막 전위(membrane potential)를 형성한다.

3. 임계 전위와 스파이크: 막 전위가 전하를 저장하면 활동 전위(action potential)가 증가하는데, 증가한 이전 시냅스의 막 전위(membrane potential)가 임계 전위를 넘으면 스파이크(spike)를 생성 후 다음 시냅스 뉴런(post-synaptic neurons)에 전달한다. 이후 스파이크 전달을 마친 이전 시냅스는
리셋 전위(reset voltage)로 초기화된다.

4. 누수에 의한 막 전위 감소: 뉴런의 막 전위는 활동 전위에 의해 전압이 높아졌다가도 시간이 지날수록 누수(leaky)로 막(membrane)을 통해 이온이 빠져나가게 되어 감소한다.

$$\tau_m \frac{d}{dt} V(t) = E_L - V(t) + RI(t) \qquad \text{if } V(t) \leq V_{th} \quad V(t) = V_{reset} \tag{1}$$

$V(t)$: 막 전위(membrane potential), 뉴런에 저장된 전위 값

τ_m: 막시간상수(membrane time constrant), 시간에 따른 막 전위 변화 속도

E_L: 누수 전위(leaky potential), 뉴런이 발화하지 않을 때, 누수에 의해 감소한 막 전위가 도달할 수 있는 값

R: 막 저항(membrane resistance), 입력전류가 뉴런의 막 전위에 주는 영향을 조절

$I(t)$: 시냅스 입력전류(input voltage), 시냅스를 통해 뉴런으로 들어오는 입력 신호

V_{th}: 점화 임계치(firing threshold), 뉴런의 막 전위가 이 값에 도달하면 스파이크 발생.

V_{reset}: 리셋전위(reset voltage), 스파이크가 발생한 후 뉴런의 막 전위는 리셋 전위로 초기화

수식 (1)은 LIF 모델 방정식을 상미분 방정식(ordinary differential equation)으로 표현한다. LIF 모델은 이전 시냅스의 입력으로 인한 막 전위 상승과 전하 누수에 따른 막 전위 감소를 표현할 수 있으며, 상미분 방정식으로 표현된다.

상미분 방정식이란 하나의 독립 변수에 대한 미분을 다루는 방정식이며, LIF 모델 방정식에서 뉴런의 막 전위가 시간에 따라 변화하는지 설명하기 위해 상미분 방정식을 사용하였고 막 전위 변화량 및 막 전위 누적 그리고 발화 과정을 효과적으로 표현할 수 있다.

하지만 위에서 소개된 수식(1)은 막 전위의 변화량을 추적하기에는 해석의 어려움이 있다. 따라서 수치적 미분 방정식 방법 중 가장 기본적인 방법인 오일러 방법(Euler's method)을 활용해 막 전위의 변화량을 추적할 수 있는 수식으로 바꾸어 사용하는 것이 일반적이다.

$$\tau_m d\frac{V(t)}{dt} = E_L - V(t) + RI(t) \tag{2}$$

$$\tau_m \frac{V(t+\Delta t) - V(t)}{\Delta t} = E_L - V(t) + RI(t) \tag{3}$$

$$V(t+\Delta t) = V(t) + \frac{\Delta t}{\tau_m}(E_L - V(t) + RI(t)) \tag{4}$$

위 수식처럼 우선 수식(3) 과같이 차분 방정식 형태로 변환하여 시간 간격 Δt로 표현합니다. 그리고 수식(2)을 오일러 방법을 이용해 식을 변화시킨 수식(3)은 Δt의 시간 간격 동안의 막 전위 변화량을 추적할 수 있게 되었고, 막 전위 누적과 발화 과정에서의 막 전위 값을 추적할 수 있게 되었다. 또한 Δt의 값이 작을수록 더 정확한 막 전위 변화량을 얻을 수 있게 된다.

2.3 RNN 모델

순환 신경망(Recurrent Neural Network)은 순차적 또는 시계열 데이터의 정보를 처리하는 신경망이며, 현재 과거 날씨 데이터를 기반으로 미래를 예측하거나, 자연어 처리(NLP)에서는 문장의 앞뒤 맥락을 이해하고 다음 순서에 올 단어를 예측한다.

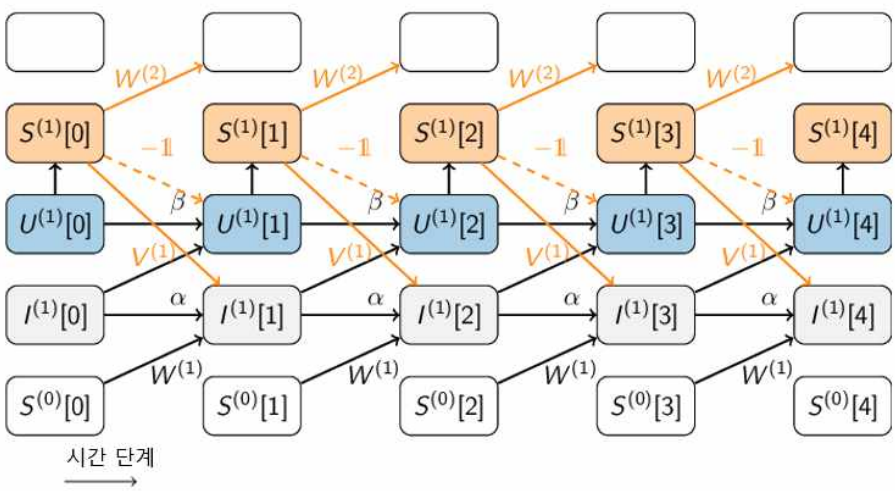
순환 신경망은 전통적인 심층 신경망(Deep Neural Network)과는 다른 특징을 가지고 있는데, 심층 신경망은 입력과 출력이 서로 영향을 주지 않는 독립적인 구조로 구성되어 있다는 것이다. 반면에 순환 신경망은 이전의 입력에서 일부 정보를 가져와 현재의 입력과 출력에 영향을 주는 '메모리' 개념이 있으며, 이를 통해 입력과 출력이 서로 영향을 주는 순환 구조로 구성된다.

또한 순환신경망의 시간 단계는 메모리를 전달하는 횟수를 의미한다. 시간 단계가 많을수록 많은 시간과 데이터의 정밀한 표현이 가능하고, 시간 단계가 적을수록 적은 시간을 표현한다. 이는 스파이크 신경망의 파라미터로써 사용되며, 데이터마다 적합한 값을 찾아야 한다.

이와 같은 순환 신경망의 메모리 개념은 스파이크 신경망에도 유사하게 적용된다. 스파이크 신경망에서는 이전 시냅스 뉴런(pre-synaptic neurons)에서 발생한 스파이크가 다음 시냅스 뉴런(post-synaptic neurons)에 영향을 주기 때문이다. 이러한 특징에 의해 스파이크 신경망의 모델을 구현할 때, 순환 신경망의 구조를 가지며, 본 논문에서도 이러한 적용 방식을 설명한다.

제 3 장 스파이크 신경망의 서로게이트 기울기

3.1 LIF 모델을 RNN에 대응



[그림 2] RNN 모델에 대응된 LIF 모델

이전 장에서 스파이크 신경망에서 순환 신경망의 메모리 특징을 이용해 모델을 구축한다고 설명하였다. 이번 절에서는 스파이크 신경망에서 학습하기 위해 LIF 모델을 순환 신경망에 어떻게 대응하는지를 설명한다. LIF 모델을 순환 신경망에 대응하기 전, 우선 LIF 모델의 기본 동작 4가지인 막 전위 전하 저장 및 변화량, 누수에 의한 막 전위 감소, 스파이크 발생과 저장을 수식으로 표현한다.

막 전위 변화량 및 누수에 의한 막 전위 감소 계산

$$\tau_{mem}\frac{dU_i^{(l)}}{dt} = -(U_i^{(l)} - U_{rest}) + RI_i^{(l)} \quad (5)$$

$U_i^{(l)}$: l 층에 있는 뉴런 i의 막 전위, 해당 값이 임계치에 도달하면 스파이크 발생

U_{rest}: 휴지 전위 또는 리셋 전위, 스파이크 발생 후 또는 입력이 없을 때 막 전위 값은 줄면서 이 값으로 도달

τ_{mem}: 막 시간 상수, 막 전위의 변화 속도를 조절, 값이 클수록 막 전위는 느리게 변화하고 값이 작을수록 막 전위는 빠르게 변화

R: 입력 저항, 입력되는 전하를 제한하며, 값이 클수록 막 전위가 입력값에 대해 둔감해지고 값이 작을수록 막 전위가 입력값에 민감

I_i: 시냅스 입력으로 외부 입력값을 의미한다.

스파이크 발생 총합 계산

$$S_i^{(l)}(t) = \Sigma_{k \in C_i^{(l)}} \delta(t - t_j^k) \quad (6)$$

t_j^k: j 층의 k번째 뉴런의 스파이크 발생 시간을 기록한 값이다.

δ: 디렉 델타(Derac delta) 함수로 입력이 0일 경우 무한대 값을 가지고, 입력이 0이 아닐 때 0의 값을 가진다. 이 함수는 갑작스러운 변화에 대해 계산을 할 때 유용하며, 길고 좁은 스파이크 모양을 하고 있어 이 수식에서는 스파이크를

표현하기 위해 용이하게 사용된다.

스파이크 전달

$$\frac{dI_i}{dt} = -\frac{I_{i(t)}}{\tau_{syn}} + \Sigma_j W_{ij} S_j^{(l-1)}(t) + \Sigma_j V_{ij} S_j^{(l)}(t) \quad (7)$$

W_{ij}: 시냅스 순방향 가중치 행렬, 순환 신경망에서 이전 시점에서 다음 시점으로의 정보 전달 역할

V_{ij}: 시냅스 순환 가중치 행렬, 순환 신경망의 메모리 뉴런으로 사용되며, 정보가 순환되어 현재 시각에 이전 시간 정보를 제공

τ_{syn}: 시냅스 감쇠 시간 상수, 시냅스 입력의 변화 속도를 조절할 때 사용되는데, 값이 증가하면 시냅스 입력의 변화 속도가 감소하고, 반대로 값이 감소하면 시냅스 입력의 변화 속도가 증가

$I_{i(t)}$: 시간 t의 시냅스 입력

$S_j^{(l)(t)}$: 시간 t에서 스파이크 발생 총합, 수식 6을 통해 계산된 값을 의미

$$S_i^{(l)}(t) = \Theta(U_i^{(l)}(t) - \vartheta) \quad (8)$$

Θ: 헤비사이드 계단 함수로 스파이크 같은 갑작스러운 변화의 표현에 유용하게 사용된다.

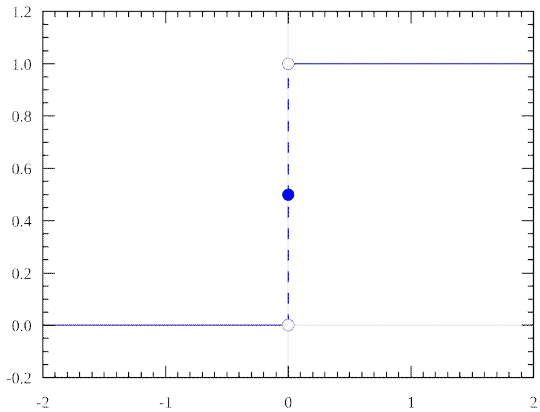
이산 시간 표현을 간결하게 하려면 스파이크의 비선형적인 특징을 더욱 잘 살리기 위해 수식 6에 헤비사이드 계단 함수를 도입해 수식 8을 표현한다. 이는 스파이크의 비선형성 특성을 강조하여 이산 시간 표현을 좀 더 간결하게 할 수 있다. 마지막으로 수식 7을 순환 신경망에 대응시키기 위해서 연속적이지 않은 일정한 간격의 시간인 이산 시간으로 표현한다. 이산 시간으로 표현하는 이유는 LIF 모델을 순환 신경망에 표현할 때 구현에 용이하며 더욱더 효율적이기 때문이다. 이제 이산 시간 표현을 간결하게 바꾸기 위해 $U_{rest}=0$, $R=1$, $V=1$로 값을 설정하여, 수식 8로 표현한다.

$$I_i^{(l)}(t+1)=\alpha I_i^{(l)}(t)+\Sigma_j W_{ij}S_j^{(l-1)}(t)+\Sigma_j V_{ij}S_j^{(l)}(t) \tag{9}$$

$$U_i^{(l)}(t+1)=\beta U_i^{(l)}(t)+I_i^{(l)}(t)-S_i^{(l)}(t) \tag{10}$$

표현된 수식 8과 수식 5, 7을 합쳐 이산 시간 t에 따른 시냅스 뉴런을 계산한다. 표현된 수식 9를 통해 시냅스 뉴런을 계산할 수 있지만, 상수 α를 $e^{(\frac{-\Delta t)}{\tau_{syn}}}$로 전환한다면 수식 10으로 더욱 간단하게 막 전위 변화를 표현할 수 있다. 수식 10은 각각 $\beta U_i^{(l)}(t)$는 누수(leaky) 계산, $I_i^{(l)}(t)$는 시냅스 입력 계산 그리고 $S_i^{(l)}(t)$는 스파이크 발생 후 리셋하는 역할을 표현한다.

3.2 스파이크 신경망의 기울기 소실 문제



[그림 3] 헤비사이드 함수 그래프

지금까지 스파이크 신경망을 포함한 많은 신경망에서 최적화 함수(신경망 모델의 가중치 학습 방식을 의미하며, 손실 함수에 의한 오차(loss)를 줄여 모델을 최적화하는 것을 목표로 하는 함수)로 경사 하강법(Gradient descent, 오차를 최소화하기 위해 기울기의 반대 방향으로 가중치를 업데이트 하는 방식)을 사용하고 있다. 하지만 스파이크 신경망의 분류 모델에서는 스파이크 함수인 헤비사이드 함수의 비선형적인 특징에 의해 좋은 성능을 내기 어렵다. 비선형적인 스파이크 함수는 특정 임곗값 교차점에서의 기울기는 1이 되고, 임곗값 교차점이 아닌 곳에서 기울기는 0이 된다. 이때 기울기가 0인 은닉층의 가중치들은 업데이트되지 않고 유지되는 기울기 소실 문제가 발생한다. 이 문제는 이진 분류 문제에서 모델 대부분은 정확도 약 50%에서 개선되지 않는 문제를 일으킨다. 즉, 스파이크 신경망의 분류 모델의 성능을 개선하려면 은닉층의 가중치들이 학습되어

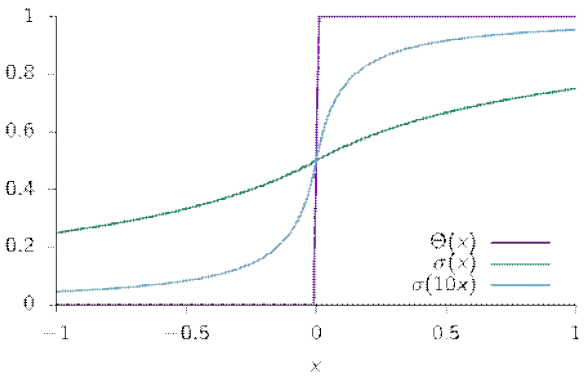
업데이트되도록 수정되어야 한다는 것을 의미한다. 이 논문에서는 은닉층의 가중치들이 학습되고, 기울기 소실 문제를 해결하기 위해서 서로게이트 기울기를 사용한다.

3.3 서로게이트 기울기 도입

$$out(input) = \begin{cases} 1, & \text{if} \quad input > 0 \\ 0, & otherwise \end{cases} \tag{11}$$

$input$: 신경망의 입력 데이터를 의미한다.

수식 11은 서로게이트 기울기의 순전파(foward) 출력 부분으로 입력된 값이 0보다 크면 1을 출력하고 아니면 0을 출력하는데, 이는 스파이크 발생을 구분한다.



[그림 4] 서로게이트 기울기 그래프

$$grad = \frac{grad\_input}{(scale \times abs(input) + 1.0)^2} \tag{12}$$

$scale$: 서로게이트 기울기의 경사를 결정, 값이 크면 경사가 급한 기울기가 되고 반대로 작으면 경사가 완만한 기울기가 된다. 이 값은 학습의 민감도에도 영향을

주기 때문에 상황에 따라 조정해주어야 함

수식 12는 서로게이트 기울기의 역전파(backward) 계산 시 기울기를 근사하는 식을 나타내며, *scale* 값으로 기울기 경사 값을 조정한다. 서로게이트 기울기는 경사 하강법에서 스파이크 함수에 의해 발생하는 비선형 구간을 대체하여 기울기를 근사해 계산할 수 있게 해주는 역할이며, 기울기 소실 문제로 인해 신경망에 있는 은닉층의 가중치들이 업데이트되지 않는 문제를 해결하고 모델의 성능을 개선한다.

3.4 스파이크 신경망 데이터 전처리

스파이크 신경망에서 주로 사용되는 데이터 형식은 시계열 데이터 또는 연속적인 시간 개념을 가진 데이터이며, MNIST 이미지 분류, SHD 데이터 및 TBL 데이터 등 다양한 분야에서 사용된다. 이번 절에서는 이런 데이터들을 학습하기 위한 데이터 전처리 과정을 설명하며, 모든 데이터가 가져야 할 조건인 단위 시간마다 스파이크 발생 여부를 표현하는 방법을 설명한다. 여기서 단위 시간이란 신경망이 데이터를 처리하는 시간 단위를 의미한다. 우선 이 조건을 만족하기 위해서는 데이터를 TTFS(Time-to-first Spike) 알고리즘을 사용해 전 처리한다. TTFS 알고리즘은 뉴런의 입력 강도에 따라 스파이크를 발생시키는 시간을 결정한다. 뉴런의 입력이 크면 빠른 스파이크를 발생시키는 반면에 입력이 작으면 느린 스파이크를 발생시킨다. 또한 TTFS 알고리즘은 입력의 강도가 일정 기준보다 적으면 스파이크를 발생시키지 않으며, 이는 불필요한 데이터 전달을 줄여 효율적이고 빠른 처리 속도를 보인다.

$$T_i = \begin{cases} \max, & \text{if } x_i < thr \\ T_i, & otherwise \end{cases} \tag{13}$$

x_i; i번째 입력 데이터

thr: 스파이크의 임계치이며, 임계치를 넘기면 스파이크가 발생

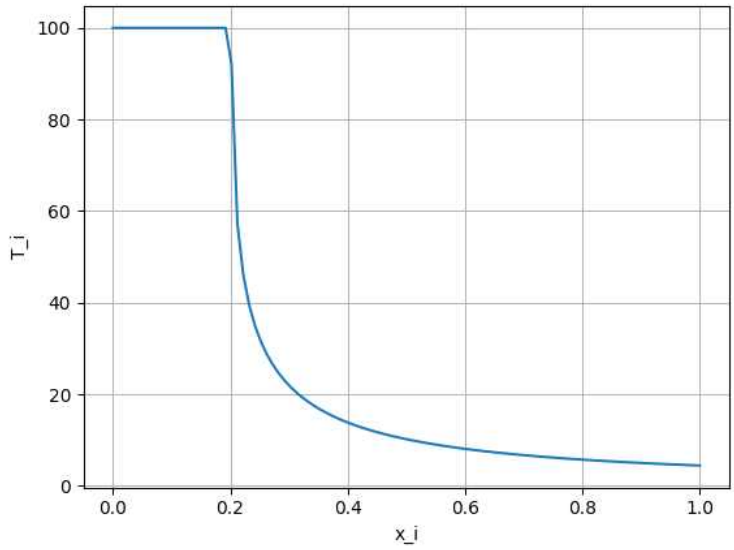
max: 스파이크가 발생하지 않았을 때의 최대 대기 시간

수식 13은 스파이크 발생 조건에 따른 T 값 설정에 대한 수식이다. 입력 데이터(x)가 임계치(thr)를 넘기지 못하면 max 값을 가지고, 임계치를 넘기면 T_i의 값은 스파이크 발생 시간을 가진다.

$$T_i = \tau \times \log\left(\frac{x_i}{(x_{i-thr})}\right) \tag{14}$$

τ: 뉴런의 막 전위 증가 속도를 결정하는 상수, 일반적으로 20 값을 가짐.
T_i: i번째 데이터의 스파이크 발생 시간

여기서 T_i에 대한 계산식을 알아내기 위해서 입력 데이터는 1차원 모양이며, 0~1의 값을 가진다고 가정한다. 수식 14는 T_i값을 계산하는 수식으로, 스파이크 발생 시간을 결정하는 방법을 나타낸다. 수식 14를 계산할 때, $\max = 100$, $thr = 0.2$라 가정했을 때, T_i가 가질 수 있는 데이터는 아래 그림 1과 같다.



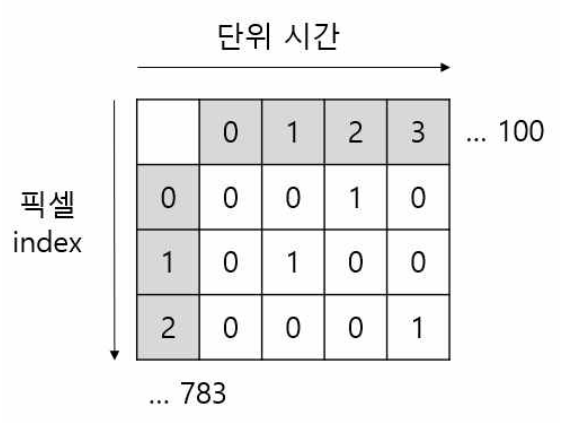
[그림 5] T 값의 입력값에 따른 출력 그래프

그림 5는 입력 데이터(x)가 임곗값을 넘겼을 때, 스파이크가 발생하는 시간을 시각적으로 나타낸다. 또한 x가 thr값에 도달하지 못했을 때 스파이크가 발생하지 않았을 때의 최소 대기 시간인 max 값을 가지고, thr값에 도달하거나 넘겼을 때, 넘긴 정도에 따라 스파이크 발생 시간이 더욱 빨라지는 것을 볼 수 있다.

```
@classmethod
    def current2firing_time(cls, x, tau=20, thr=0.2, tmax=1.0, epsilon=1e-7):
        idx = x < thr
        x = np.clip(x, thr + epsilon, 1e9)
        T = tau * np.log(x / (x - thr))
        T[idx] = tmax # tmax는 수식의 max 값을 의미
        return T
```

[그림 6] 스파이크 신경망의 입력 데이터 전처리 코드

그림 6는 입력 데이터를 전처리하는 코드이다. 이 코드를 통해 학습에 입력되는 데이터는 단위 시간마다 스파이크 발생 여부를 지니며, 스파이크 신경망에서 사용할 수 있는 데이터의 조건을 충족시킨다.



[그림 7] 전처리 된 데이터 행렬 예시

전처리 된 데이터는 그림 7의 형식을 띠며 스파이크 신경망에 사용될 수 있는 조건을 충족했지만, 이는 0 값을 가진 데이터가 너무 많아 공간의 효율이 떨어진다.

픽셀 index	단위 시간	values
0	2	1
1	1	1
2	3	1

... 783

[그림 8] 희소행렬 구조 예시

따라서 그림 8의 행렬을 그림 8처럼 희소행렬로 표현하면, 무수한 0 값으로 인한 메모리 낭비를 막고 공간 효율을 높일 수 있다. 희소행렬은 특히 데이터양이 많을수록 계산 속도가 빨라지고, 더욱 효율적으로 표현할 수 있다. 최종적으로 그림 4로 표현된 데이터가 스파이크 신경망 학습 모델에 사용된다.

3.5 스파이크 신경망 모델의 정규화 손실

우리는 스파이크 신경망(Spiking Neural Network)을 학습시킬 때, 기존 오차함수에 더해 스파이크의 총량을 억제하는 정규화 손실(regularizer loss)을 사용할 필요가 있다. 그 이유는 너무 많은 스파이크의 발생은 계산의 효율을 저하하며, 메모리 낭비로 이어질 수 있어 최소한의 스파이크로 정보를 처리하는 것이 유리하기 때문이다.

$$l(x,y) = \frac{1}{\sum_{n=1}^{N} w_{yn}} l_n \tag{15}$$

x: 학습 데이터의 입력

y: 학습 데이터의 라벨

w_{yn}: 오차함수 가중치, 클래스의 불균형 처리

N: 학습 데이터의 배치크기

스파이크 신경망을 학습시킬 때, 일반적으로 수식 16의 NLL Loss(Negative Log Likelihood Loss) 함수를 사용한다. 따라서 본 논문에서는 NLL Loss에 정규화 손실을 추가한다.

$$regloss = 0.00005 \times \sum S_i \tag{16}$$

S_i: 뉴런 I의 스파이크 발생 여부

수식 16은 스파이크 총량의 합을 $regloss$로 표현한다. 또한 수식 16의 0.00005 값은 파라미터(Parameter)로 쓰일 수 있으며, 모델마다 올바른 값으로 조정하여 사용하는 것이 좋다.

$$loss = l(x,y) + regloss \tag{17}$$

결론적으로 수식 17의 오차함수를 사용하여 학습 과정에서 불필요한 스파이크 억제 및 모델 계산 효율의 향상을 이뤄냈다.

3.6 스파이토치 구현

스파이크 신경망의 구현은 모두 스파이토치(Spytorch)를 참고한다. 스파이토치는 Zenke와 Halvagal이 개발한 소스 코드로 스파이크 신경망과 서로게이트 기울기를 파이썬(Python) 언어의 파이토치 라이브러리를 통해 쉽게 구현한다. 본 논문에서는 해당 코드를 수정하여 하나의 클래스로 간단히 그리드 서치 알고리즘을 적용할 수 있게 확장하였고, 각 데이터의 특성에 맞춘 전처리와 파라미터 분석이 가능하도록 구현한다.

```
import torch
import numpy as np

input_nodes = 784   # 입력 뉴런 개수
output_nodes = 200  # 출력 뉴런 개수

weight = torch.empty((input_nodes, output_nodes), dtype=torch.float, requires_grad=True)
torch.nn.init.normal_(weight, mean=0.0, std=0.2 / np.sqrt(output_nodes))
```

[그림 9] 스파이크 신경망의 초기 가중치 설정 코드

본 논문에서 사용되는 데이터들의 가중치의 초깃값은 모두 같이 He 초깃값(Kaiming He가 만들어낸 방법으로 이미 많은 모델에서 좋은 성능 향상을 증명한 초깃값 설정 방식)과 유사한 방식으로 설정한다. 해당 방식은 과도한 초깃값으로 인해 학습 안전성과 학습 속도의 저하를 방지한다. 그림 9에 제시된 input_nodes, output_nodes 변숫값은 각각 입력 뉴런 개수와 출력 뉴런 개수로 모델과 가중치의 층에 따라 올바르게 변경해야 한다.

```
torch.einsum("abc,cd->abd", (val_1, val_2))
```

[그림 10] 파이토치의 아인슈타인 합산 예시 코드

또한 본 논문의 모든 코드에서의 행렬 곱 계산은 파이토치의 아인슈타인 합산 함수(torch.einsum)를 사용하여 효율적인 계산이 가능하도록 구현하였다. 그림 10의 val_1, val_2는 각각 계산하고자 하는 행렬을 의미하며, ab cd는 각각 해당하는 행렬의 차원을 의미한다.

```python
import torch


class SurrogateGradientSpike(torch.autograd.Function):
    scale = 100.0  # 기울기 정도를 조절

    @staticmethod
    def forward(ctx, input):
        ctx.save_for_backward(input)
        out = torch.zeros_like(input)
        out[input > 0] = 1.0  # 스파이크 여부를 판단

        return out

    @staticmethod
    def backward(ctx, grad_output):
        (input,) = ctx.saved_tensors
        grad_input = grad_output.clone()
        grad = grad_input / (SurrogateGradientSpike.scale * torch.abs(input) + 1.0) ** 2

        return grad

    @classmethod
    def __init__(cls, scale: float):
        cls.scale = scale
```

[그림 11] 서로게이트 기울기 클래스 코드

그림 11은 서로게이트 기울기를 구현한 코드로 파이토치 함수를 상속받아 스파이크 함수로써 사용된다. scale 변수는 서로게이트 기울기의 경사를 정하는 값으로 유효한 범위의 값에서 점차 수정하여 최적의 값을 찾아 신경망 모델의 성능을 높일 수 있다. 또한 서로게이트 기울기 코드는 오차역전파(연쇄 법칙을 이용한 가중치 업데이트 방법으로 효율적인 계산으로 가중치 업데이트가 가능한 방식) 방법을 통해 가중치를 업데이트한다.

제 4 장 스파이크 신경망 파라미터별 성능 분석

4.1 파라미터별 성능 분석 개요

이번장에서는 다양한 스파이크 신경망 데이터 세트들의 파라미터별 성능 비교 분석을 진행한다. 각 데이터 세트에 따른 파라미터 조합과 정확도, 그리고 학습 실행 시간을 기록하여 가장 높은 성능을 내는 파라미터 조합과 그렇지 않은 조합의 특징을 비교한다. 또한 각 파라미터의 관점에서 훈련 데이터 정확도를 비교하며, 훈련 데이터 및 테스트 데이터의 정확도 평균과 실행 시간 평균을 비교한다.

[표 1] 스파이크 신경망 파라미터 목록

파라미터 목록			
번호	**파라미터 변수명**	**설명**	**성능 비교 여부**
1	nb_hidden	은닉층 노드 수	o
2	nb_step	시간 단계	o
3	lr	학습률	
4	nb_epochs	학습 횟수	
5	batch_size	배치 크기	
6	scale	서로게이트 기울기 경사	o
7	regularizer	정규화 손실률	o
8	tau_mem	막 전위 변화율	
9	tau_syn	시냅스 변화율	
10	alpha	시냅스 계산 상수	
11	beta	막 전위 계산 상수	
12	loss_fn	오차함수	

표 1은 스파이크 신경망에서 사용되는 파라미터 목록을 나타낸다. 성능 비교는 전체 파라미터 중 스파이크 신경망 모델의 핵심이 되는 파라미터인 은닉층 뉴런 개수(신경망 모델 복잡도 조절), 시간 단계 개수(단위 시간의 개수 조절), 서로게이트 기울기 경사(스파이크 함수 민감도 조절), 정규화 손실률(스파이크당 오차값 계산 조절)을 사용하였고, 그리드 서치 알고리즘(가능한 모든 파라미터 조합을 탐색하여 정해진 기준에 부합하는 최적의 조합을 찾는 방식으로 시간이 많이 소요된다는 단점이 있지만, 모든 파라미터 조합을 시도해 볼 수 있다는 장점이 있음)을 이용해 성능을 비교 분석한다. 성능 분석을 통해 도출된 가장 높은 성능을 내는 파라미터 조합은 스파이크 신경망에 입문하는 학습자들에게 데이터 세트의 특성에 따른 파라미터 설정 방법을 제시하여 파라미터 설정 시 겪는 어려움을 해소하길 기대한다.

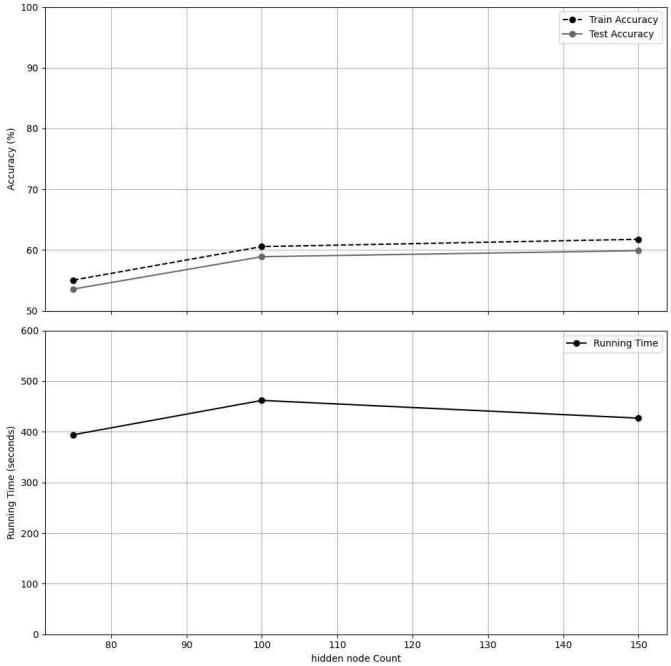
4.2 Fashion-MNIST 데이터 세트의 성능 분석

이미지 분류는 신경망 모델이 실제 문제에서 사용될 수 있는 대표적인 분야 중 하나이다. 특히 MNIST(Modified National Institute of Standards and Technology) 데이터 세트는 이미지 분류 모델의 학습과 성능 평가에 많이 사용된다. MNIST는 주로 손으로 쓴 0~9 숫자, 총 10개의 손 글씨 숫자를 분류하는 데이터 세트로 알려졌지만, 현재는 많은 종류의 MNIST 이미지 데이터 세트가 존재한다. 본 논문에서는 기본적인 숫자 데이터보다 복잡한 TorchVision에서 제공하는 의류 데이터인 Fashion-MNIST를 사용한다.

[그림 12] Fashion-MNIST 라벨별 데이터 시각화

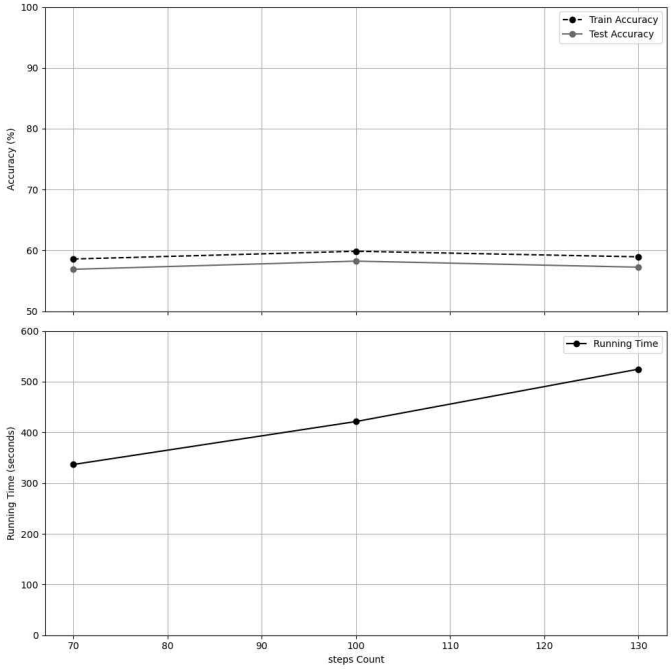
Fashion-MNIST의 각 픽셀은 0~255의 값을 가지며, 가로 28픽셀, 세로 28픽셀 총 784픽셀의 흑백 이미지로 되어 있다. 그리고 티셔츠, 바지, 풀오버, 드레스,

코트, 샌들, 셔츠, 운동화, 가방, 부츠 총 10개의 데이터를 가지고 있고, 훈련 데이터는 총 60,000개이다. 모델 학습 전 데이터를 0~1 사이로 표준화(Standaridze)하여 신경망의 학습 효율을 높인다.



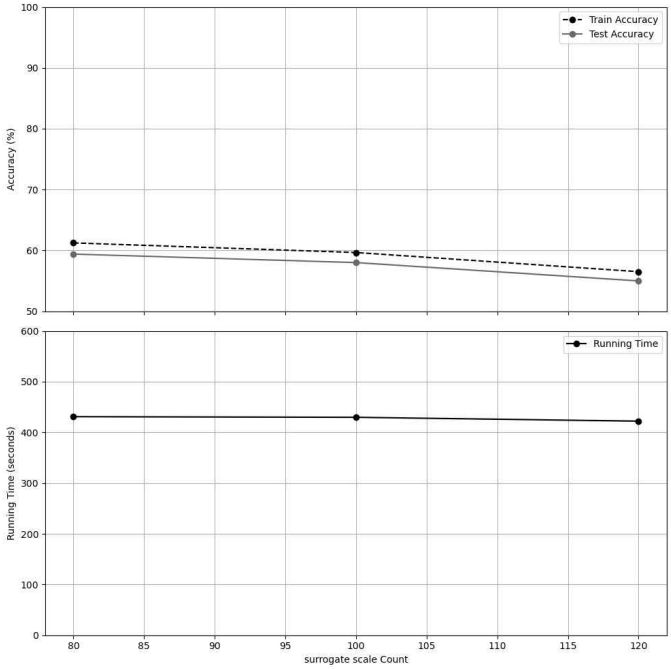
[그림 13] Fashion-MNIST 데이터의 은닉층 노드 수별 평균 정확도 및 평균 실행 시간

그림 13는 은닉층 노드 수 기준 훈련 및 테스트 데이터 정확도와 학습 시간을 보인다. 평균적으로 은닉층 노드의 증가는 정확도 향상에 도움을 주며, 100 이상의 은닉층 설정은 평균 학습 시간을 약 30초 감소시키는 데 도움을 준다. 따라서 Fashion-MNIST 최적의 은닉층 노드 수는 150 이상인 것으로 분석된다.



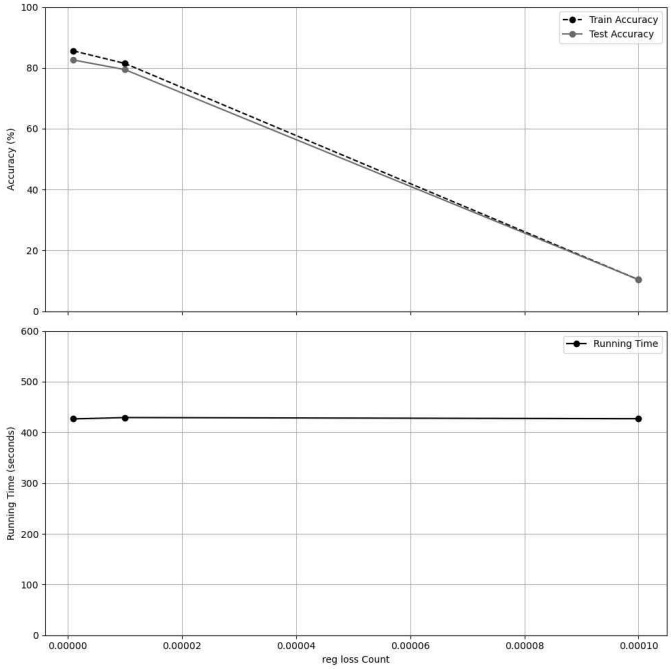
[그림 14] Fashion-MNIST 데이터의 시간 단계별 평균 정확도 및 평균 실행 시간

그림 14의 평균 정확도 그래프를 보면 시간 단계가 100일 때 평균 정확도가 미세하게 높은 것을 알 수 있고, 시간 단곗값이 증가할수록 학습 시간이 증가하는 것을 볼 수 있다. 따라서 Fashion-MNIST의 시간 단곗값은 100이 적절하다고 분석된다.



[그림 15] Fashion-MNIST 데이터의 서로게이트 기울기 경사별 평균 정확도 및 평균 실행 시간

그림 15의 그래프는 학습 및 테스트 데이터의 평균 정확도를 보인다. 경사 값이 작을수록 미세하게 정확도가 높고, 실행 시간 또한 높은 것을 알 수 있다. 하지만 성능 측면에서는 서로게이트 기울기 경사 값이 작을수록 학습 정확도에 긍정적이기 때문에, 최적의 경사 값은 80 이하로 분석된다.



[그림 16] Fashion-MNIST 데이터의 정규화 손실률별 평균 정확도 및 평균 실행 시간

그림 16에서도 정규화 손실률에 따른 학습 시간 변화는 없지만, 학습 정확도에 큰 차이를 보인다. 1e-6의 값일 때 대부분 높은 정확도를 보이지만 1e-4일 때 정확도는 10%에 가까워 학습이 전혀 되지 않는 것이 보인다. 이는 너무 큰 정규화 손실률은 스파이크 신경망 학습에 큰 영향을 준다고 분석된다. 따라서 Fashion-MNIST 데이터에서의 최적의 정규화 손실률은 1e-6 이하이며, 정규화 손실률을 낮게 설정할수록 스파이크 신경망 학습에 유리해질 것으로 분석된다.

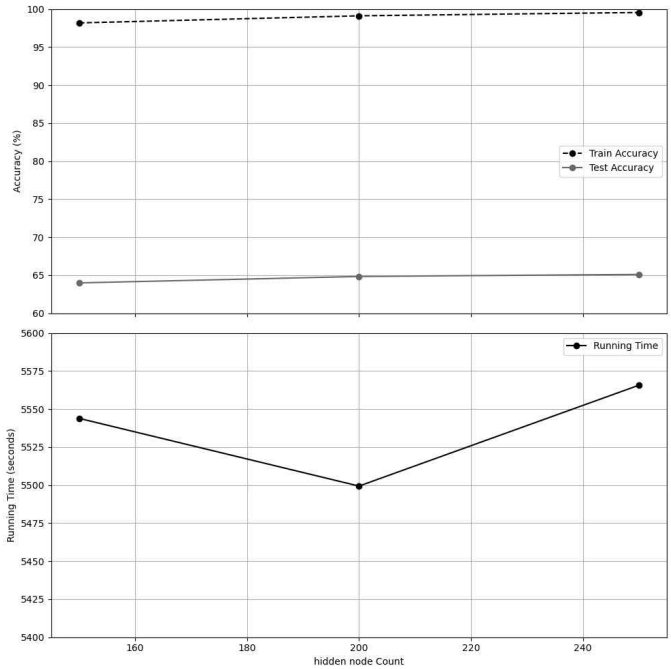
[표 2] Fashion-MNIST 데이터 최적의 파라미터 조합

파라미터 조합				
은닉층 노드 개수	시간 단계	서로게이트 기울기 경사	정규화 손실률	학습 정확도
150+	100	80.0-	0.000001-	89.53%

결론적으로 Fashion-MNIST 같은 이미지 데이터를 스파이크 신경망에 학습시킬 때, 은닉층 노드 개수는 늘리고, 서로게이트 기울기 경사와 정규화 손실률을 낮게 설정하는 것이 이미지 데이터 학습에 유리하며 더 좋은 결과를 낼 수 있을 것으로 분석되며 높은 정규화 손실률은 스파이크 신경망 학습에 큰 영향을 준다.

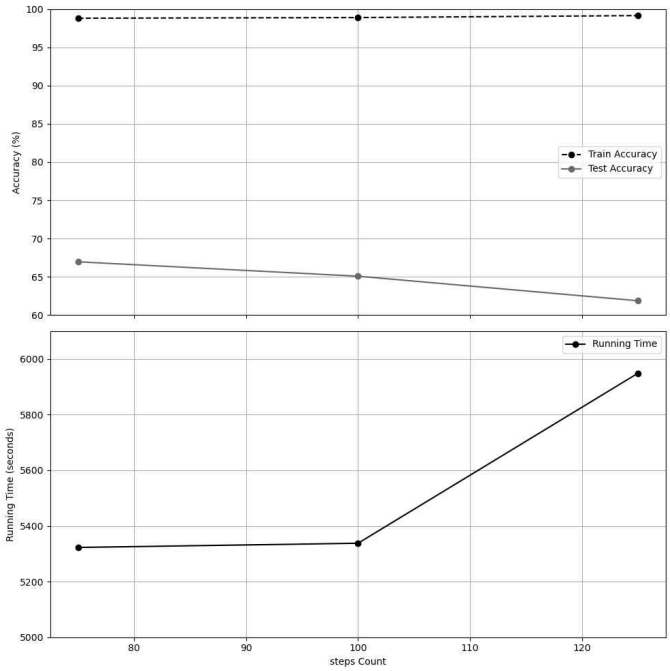
4.3 SHD 데이터 세트의 성능 분석

실제 상황에서 스파이크 신경망이 주로 활용되는 분야는 시계열 데이터 즉, 연속적인 데이터의 분석 및 예측 분야이다. 이번 절에서는 스파이크 신경망에서의 시계열 데이터 적용 사례를 보이며, SHD(Spiking Heidelberg Digits) 시계열 데이터 세트의 적용 방법을 설명한다. SHD 데이터 세트는 시계열 데이터 중 하나로 남성과 여성이 0~9를 발음하는 목소리를 녹음하여 각 발음의 음높이 값을 가지는 데이터 세트이다. 본 논문에서 사용된 SHD 데이터 세트는 이미 스파이크 신경망에 사용될 수 있도록 전처리 되어 스파이크 발생 시간과 그 뉴런 정보를 포함하고 있다.



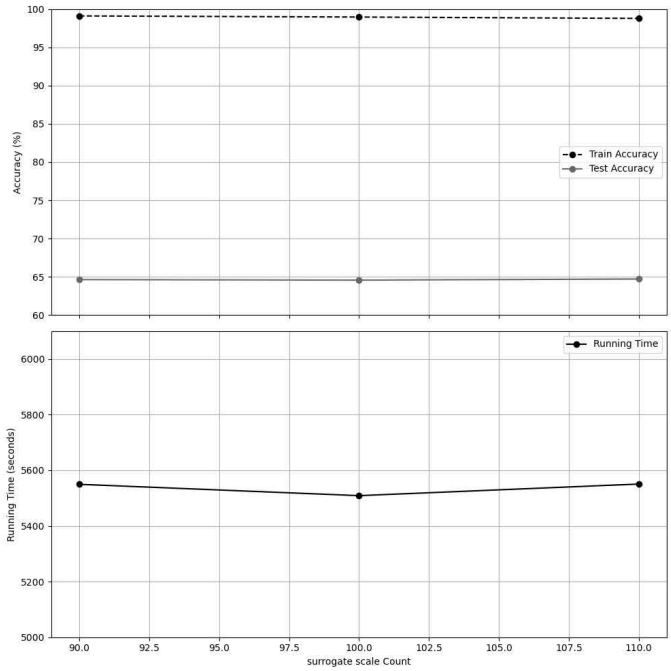
[그림 17] SHD 데이터의 은닉층 노드 수별 평균 정확도 및 평균 실행 시간

그림 17에서도 훈련, 테스트 데이터에서 은닉층 노드 수가 많을수록 높은 정확도를 보인다. 하지만 노드 수 200일 때를 제외하면 실행 시간이 많이 증가한다. 따라서 SHD 데이터 최적의 은닉층 노드 수는 200으로 분석한다.



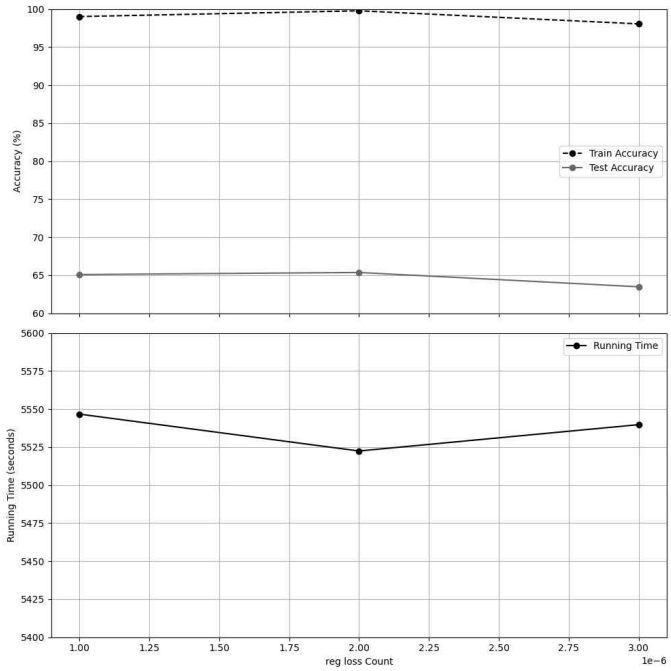
[그림 18] SHD 데이터의 시간 단계별 평균 정확도 및 평균 실행 시간

그림 18에서 시간 단계 값에 따라 확연한 차이를 보이는데, 시간 단계 값이 증가할수록 훈련 데이터의 정확도는 높아지지만 테스트 데이터의 정확도는 낮아진다. 이는 높은 시간 단계 값이 데이터의 과적합(over fitting) 문제를 유발할 수 있다는 것이다. 또한 시간 단계가 80, 100일 때는 평균 5200~5400 범위이던 실행 시간이 시간 단계가 120일 때 급격하게 증가한다. 결론적으로 높은 시간 단계 값을 피하면 과적합과 실행 시간 낭비를 줄일 수 있고, 실험한 값보다 좀 더 낮은 값을 사용해도 될 것으로 분석된다.



[그림 19] SHD 데이터의 서로게이트 기울기 경사별 평균 정확도 및 평균 실행 시간

그림 19는 서로게이트 기울기 경사별 평균 정확도와 평균 실행 시간을 나타낸다. 서로게이트 기울기 경사 값이 100.0일 때, 평균 실행 시간에서 약간 낮은 값을 보인다. 결론적으로 서로게이트 기울기 경사 값은 데이터 학습에 영향은 없지만 실행 시간에는 영향이 있는 것으로 분석되며, 최적의 서로게이트 기울기 경사 값은 100.0으로 분석된다.



[그림 20] SHD 데이터의 정규화 손실률별 평균 정확도 및 평균 실행 시간

그림 20에서도 정규화 손실률이 2e-6일 때 가장 좋은 성능을 내고 있으며, 최적의 정규화 손실률은 2e-6으로 분석된다. 또한 정규화 손실률 설정은 스파이크 신경망의 성능 개선에 중요한 역할을 하는 것으로 분석된다.

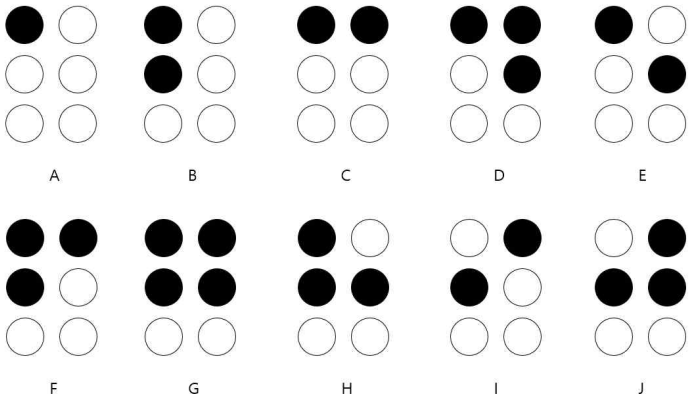
[표 3] SHD 데이터 최적의 파라미터 조합

파라미터 조합				
은닉층 노드 개수	시간 단계	서로게이트 기울기 경사	정규화 손실률	학습 정확도
200+	75−	100.0	0.000002	99.92%

파라미터 조합 총 81개 중, 실행 시간을 고려한 은닉층 노드 개수와 서로게이트 기울기 경사, 실행 시간 및 과적합을 고려한 시간 단계 값 그리고 데이터 정확도와 실행 시간을 고려하여 SHD 데이터는 표 3을 최적의 파라미터 조합으로 제시한다. 이 파라미터 조합은 전체 파라미터 조합 대비 1% 정도 높은 학습 정확도를 보인다.

4.4 TBL 데이터 세트의 성능 분석

연속적인 시계열 데이터를 주로 활용하는 스파이크 신경망은 실제 센서에서 가져온 데이터도 적용할 수 있다. 대표적인 예시로 Tactile Braille Letters(TBL) 데이터 세트를 들 수 있는데, TBL 데이터는 점자를 센서로 읽어온 값을 가진다.



[그림 21] 점자로 표현된 A~J

시각 장애인이나 시력이 낮은 사람들은 그림 21과 같이 6개의 점으로 되어 있는 점자를 손끝의 감각으로 느껴 글을 읽는데, TBL 데이터는 사람 대신 센서가 점자를 읽어 생성된다. 본 논문에서 사용된 TBL 데이터 세트는 띄어쓰기와 A부터 Z까지의 알파벳을 분류하며, 점자의 볼록한 부분에서 센서값이 평평한 부분이 서로 대비되는 점을 이용해 데이터를 생성한다.

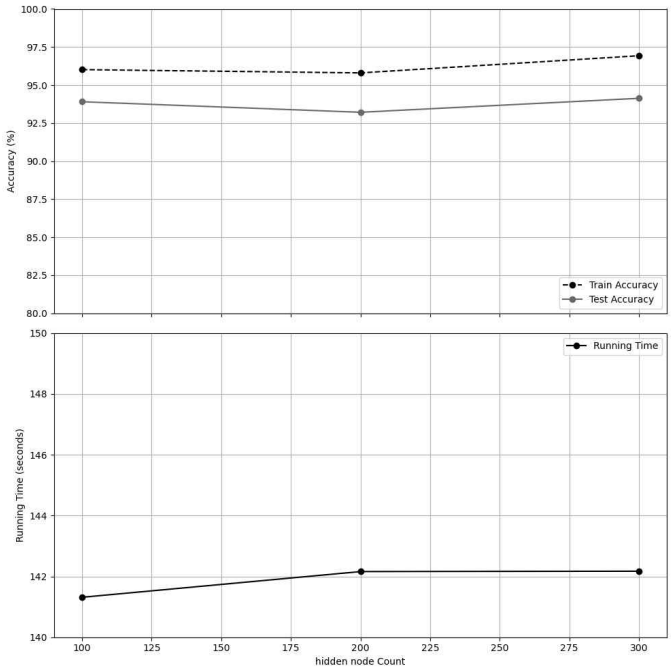
TBL 데이터 세트는 신경망에 바로 사용되지 않고, 크게 두 가지 전처리를 더 진행해야 한다. 첫 번째로 모든 입력 데이터가 고정된 크기로 입력될 수 있도록 입력 데이터의 최소 길이를 기준으로 길이를 고정한다. 이는 신경망 학습에서 입력 데이터는 일관적인 크기여야 한다는 조건을 만족한다.

```
def upsample(data, n=2): # n: 업샘플링할 배수
            shp = data.shape
            tmp = data.reshape(shp + (1,))
            tmp = data.tile((1, 1, 1, n))
            return tmp.reshape((shp[0], n * shp[1], shp[2]))

# data: 입력 데이터
data = upsample(data, n=cls.nb_upsample)
```

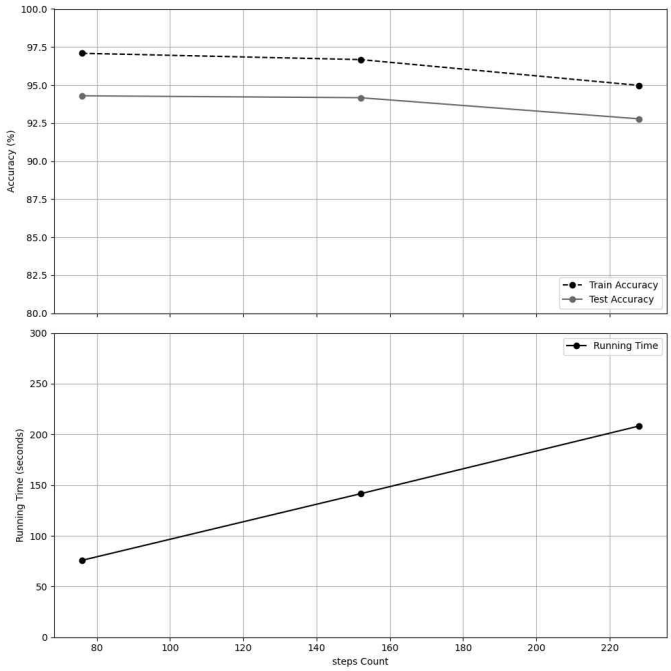
[그림 22] TBL 데이터 업샘플링 함수 코드

두 번째로 최소 길이로 고정된 데이터의 짧아진 길이를 다시 늘여주는 업샘플링(up sampling) 과정을 거쳐야 한다. 그림 22는 실제 코드에서 활용된 업샘플링 함수를 나타낸다. 업샘플링 과정은 입력 데이터의 양을 늘려 데이터의 밀집도를 늘리고 신경망 학습의 효율을 늘릴 수 있다.



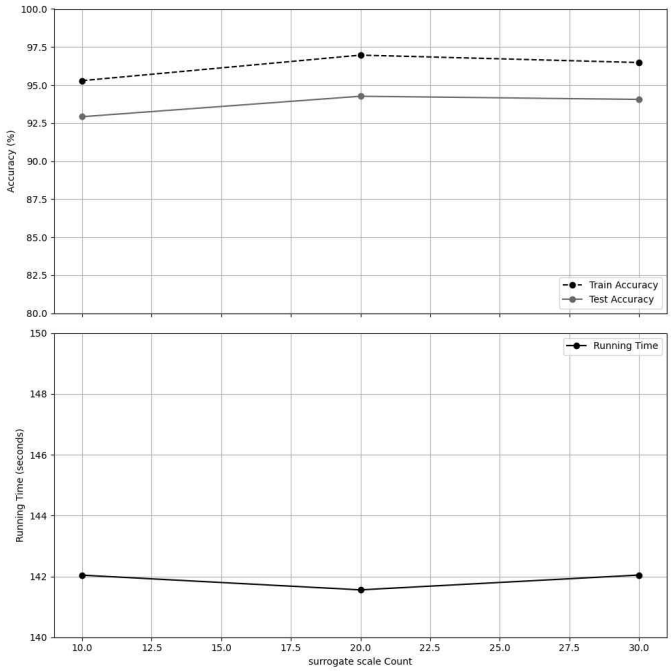
[그림 23] TBL 데이터의 은닉층 노드 수별 평균 정확도 및 평균 실행 시간

그림 23에서도 평균 정확도 및 실행 시간 그래프에서 은닉층 노드 수가 300개일 때 가장 높은 정확도를 보였으며 노드 수가 증가할수록 실행 시간이 감소한다. 따라서 은닉층 노드 수를 늘리는 것은 TBL 데이터에 긍정적인 영향을 줄 것으로 보인다.



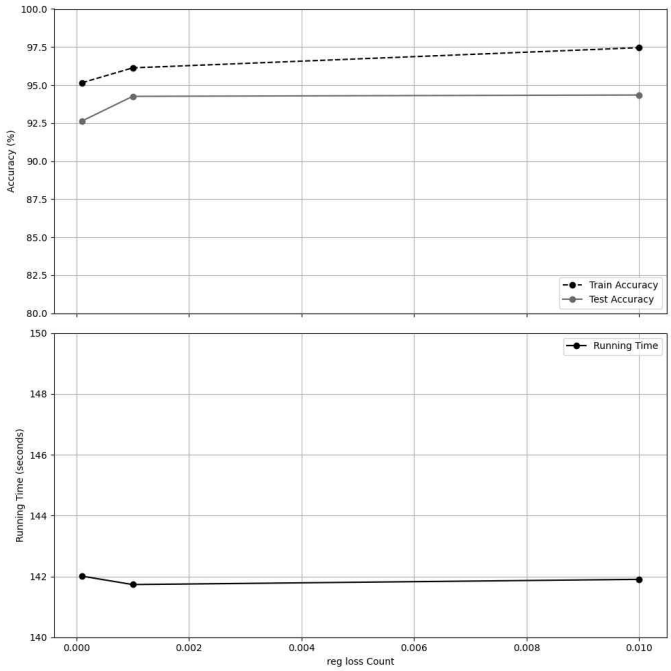
[그림 24] TBL 데이터의 시간 단계별 평균 정확도 및 평균 실행 시간

그림 24는 평균 정확도 및 평균 실행 시간 그래프이며, 시간 단계 값이 낮을수록 높은 정확도와 낮은 실행 시간을 보인다. 이는 시간 단곗값을 낮게 설정할수록 더 높은 정확도와 적은 실행 시간으로 학습할 수 있을 것을 기대할 수 있다.



[그림 25] TBL 데이터의 서로게이트 기울기 경사별 평균 정확도 및 평균 실행 시간

그림 25는 평균 정확도 및 평균 실행 시간 그래프이며, 20.0의 값이 가장 큰 평균 정확도와 적은 실행 시간을 보인다. 결론적으로 TBL 데이터 최적의 서로게이트 기울기 경사 값은 20.0으로 분석된다.



[그림 26] TBL 데이터의 정규화 손실률별 평균 정확도 및 평균 실행 시간

그림 26은 통해 정규화 손실률이 높을수록 높은 정확도를 보이는 것을 알 수 있다. 또한 손실률을 높인다고 해도 실행 시간에 영향을 주지 않기 때문에 정규화 손실률 값은 논문에서 시도 0.01보다 더 높게 설정해도 될 것으로 분석한다.

[표 4] TBL 데이터 최적의 파라미터 조합

파라미터 조합				
은닉층 노드 개수	시간 단계	서로게이트 기울기 경사	정규화 손실률	학습 정확도
300+	76−	20.0	0.01+	99.80%

81가지 파라미터 조합 중 최적의 파라미터 조합으로 표를 제시하며, 이를 통해 약 4%의 정확도 향상을 이뤄내었다. 또한 은닉층 노드 개수는 300보다 더 높게, 시간 단계는 76보다 낮게, 정규화 손실률은 0.01보다 높게 설정하여도 더 좋은 결과를 낼 수 있을 것으로 예상한다.

제 5 장 결론 및 향후 과제

본 논문은 스파이크 신경망을 다양한 데이터 세트에서 활용하고 데이터의 특징에 맞는 최적의 파라미터를 도축하기 위해 작성되었다. 스파이크 신경망은 인간 뇌에 있는 뉴런의 정보 전달 방식을 모방하여 전력 소모가 적어 차세대 신경망 모델로 주목받고 있다. 그러나 스파이크 신경망의 스파이크가 가진 이진 특성으로 인해 발생하는 은닉층에서 기울기 소실 문제와 다른 신경망에 비해 학습 시간이 높다는 한계가 존재한다. 이를 해결하기 위한 최적의 파라미터를 찾는 과정을 제시하는 것이 이 논문의 목표이다.

본 논문에서는 스파이크 신경망을 순환 신경망 구조에 대응시켜 파이토치 라이브러리를 이용해 구현하고, 서로게이트 기울기를 통해 기울기 소실 문제를 해결한다. 또한 스파이크 신경망을 Fashion-MNIST, SHD, TBL 데이터 세트에 적용하여, 각 데이터 세트에 적합한 데이터 전처리 방식을 제공하고 은닉층 노드 수, 시간 단계, 서로게이트 기울기 경사, 정규화 손실률 총 4개의 파라미터별 분석을 통해 파라미터가 스파이크 신경망의 정확도에 미치는 영향을 분석하고, 최적의 파라미터를 도출해낸다.

분석 결과 각 데이터 세트는 스파이크 발생 여부와 그 시간을 표현하도록 전처리해야 하고, 데이터 세트마다 파라미터 조합의 특징이 다르다는 것을 확인하였다. Fashion-MNIST 데이터에서는 은닉층 노드 수와 정규화 손실률이 정확도에 큰 영향을 주었으며, SHD 데이터에서는 시간 단곗값의 증가로 인해 과적합이 발생할 수 있음을 확인하였다. 또한 TBL 데이터에서는 낮은 시간 단계와 높은 은닉층 노드 수가 학습 시간을 줄이는 데에 영향을 주었다. 이렇게 도출해 낸

최적의 파라미터 조합을 통해 데이터 세트별 약 2~4% 정도 정확도 향상을 이뤄내었다.

본 논문은 이미지, 음성, 센서 데이터의 대표적인 예시 하나에 국한되어 분석을 진행했기 때문에 다른 유형의 데이터나 더욱 복잡한 데이터에서 같은 결과를 보장하기 어렵다. 또한 스파이크 신경망의 한계인 학습 속도가 느리다는 것 때문에 많은 파라미터 조합을 시도하지 못했고, 신경망의 기본 파라미터인 에포치(epoch), 학습률(learning rate) 등등 더 많은 파라미터 조합과 파라미터 개수로 추가 분석이 필요하다. 또한 스파이크 신경망의 학습 속도를 개선하는 방향이 마련되어야 하고, 스파이크 신경망의 특징인 전력 소모량을 모델 학습 시 컴퓨터가 소모하는 실제 전력량 비교를 통해 현재 사용하고 있는 딥러닝을 대체할 수 있을지에 대한 분석도 고려되어야 한다.

참 고 문 헌

[1] Emre O. Neftci, Hesham Mostafa, Friedemann Zenke. Surrogate Gradient Learning in Spiking Neural Networks. 2019.

[2] SAMANWOY, GHOSH-DASTIDAR, HOJJAT ADELI, SPIKING NEURAL NETWORKS. International Journal of Neural Systems, Vol.19, No.04, pp. 295-308. 2021.

[3] jinprelude. SNN Basic Tutorial 1 Spiking Neural Network. Jun 21, 2022. https://jinprelude.github.io/posts/SNN-Basic-Tutorial-1-Spiking-Neural-Network%EB%9E%80/. [Accessed: Oct. 18, 2024].

[4] IBM, 순환 신경망(RNN)이란 무엇인가요?.
https://www.ibm.com/kr-ko/topics/recurrent-neural-networks.
[Accessed: Oct, 19, 2024].

[5] fzenke. spytorch github. https://github.com/fzenke/spytorch/tree/main.
[Accessed: Oct, 20, 2024].

[6] Brian authors. Brian2 Introduction.
https://brian2.readthedocs.io/en/stable/introduction/index.html.
[Accessed: Oct, 20, 2024]

[7] Pytorch. Pytorch org. https://pytorch.org/. [Accessed: Oct, 26, 2024].

[8] 사이토 고키. 『밑바닥부터 시작하는 딥러닝』. 한빛미디어. 2017.

[9] Numpy. Numpy Documentation. https://numpy.org/doc/2.1/.
[Accessed: Oct, 26, 2024]

[10] Simon Mueller-Cleve, Lyes Khacef. tactile_braille_reading github. https://github.com/event-driven-robotics/tactile_braille_reading?tab=readme-ov-file. [Accessed: Oct, 27, 2024]

[11] matplotlib. matplotlib docs. https://matplotlib.org/3.5.0/api/_as_gen/matplotlib.pyplot.html#module-matplotlib.pyplot. [Accessed: Oct, 29, 2024]

[12] 그림 2. fzenke. SpyTorchTutorial1. https://github.com/fzenke/spytorch/blob/main/notebooks/figures/snn_graph/snn_graph.png [Accessed: Nov, 11, 2024]

감 사 의 글

처음 대학에 입학한 2020년에는 대학에 다니는 이유를 많은 사람을 만나고, 내 고향에서 벗어나 더 많은 경험을 해보자고 생각하고 있었습니다. 그리고 5년간 다양한 사람을 만나고, 다양한 경험을 할 수 있었습니다. 저는 감사의 글을 통해 5년간 만난 많은 사람에게 감사의 한마디를 전해보고자 합니다. 우선 처음 CS 연구실에 들어온 21년 3월 5일부터 졸업까지 수많은 피드백과 경험을 할 수 있게 도와주신 이상정 교수님, 다양한 과제를 통해 코딩하는 즐거움을 얻게 해주신 천인국 교수님, 항상 잘할 수 있게 독려해 주신 하상호 교수님, 언제나 친절하게 이야기를 들어주시고 많은 것을 알려주신 홍인식 교수님, 학회장으로 지내던 23년에 많은 도움을 주신 이해각 교수님, 컴퓨터에 대해 아무것도 모를 때 재미를 느끼게 해주신 남윤영 교수님, 처음 연구실에 들어왔을 때 랩장으로써 든든했던 송희령 선배님, 개발자로서 많은 인사이트를 얻게 해주신 윤여일 선배님, 연구실 안과 밖에서 모두 많은 도움을 주신 김재진 선배님, 처음 CS 연구실을 소개하고 지금의 내가 있게 해주신 이인재 선배님, 23년 한 해 동안 행복할 수 있게 도와주신 신주용 선배님, 따뜻한 말로 학교생활에 많은 도움을 주신 오민석 선배님, 컴퓨터로 다양한 경험을 할 수 있게 도와주신 김도현 선배님, 깐부로써 21년을 재밌게 보낼 수 있게 해주신 이승하 선배님, 처음 연구실 들어왔을 때부터 계속 도움 주신 민새미 선배님, 멘토로서 1학년 잘 적응할 수 있게 도와주신 이원영 선배님, 처음 만난 날부터 편하게 대할 수 있게 도움 주신 윤준식 선배님, 5년 동안 하나뿐인 동기로서 재밌는 학교생활을 하도록 해준 민서, 솔직한 말로 많은 도움을 준 동균, 차기 랩장으로써 걱정 많지만 항상 잘 해내는 어진, 힘들 때 항상 내 편이 되어준 요원, 어떤 분위기에든 잘 어울려주는 지민, 신입생부터 지금까지 꾸준히 믿음직스러운 은정, 23년 학회장 생활을 잘할 수 있게 도와주신 모든 집행부 선후배와 조교님, 그리고 마지막으로 24년 학회장으로 고생한 순홍 소중한 내 동기와 선후배 모두에게 감사의 글을 전합니다.