석사학위논문

스파이킹 신경망을 위한 심층 신경망 가중치 변환 기법

2022년

한 성 대 학 교 대 학 원

컴 퓨 터 공 학 과

컴 퓨 터 공 학 전 공

이 정 수

석 사 학 위 논 문
지도교수 허준영

스파이킹 신경망을 위한 심층 신경망 가중치 변환 기법

Deep Neural Network Weight Conversion Method for Spiking Neural Network

2022년 6월 일

한 성 대 학 교 대 학 원

컴 퓨 터 공 학 과

컴 퓨 터 공 학 전 공

이 정 수

석 사 학 위 논 문
지도교수 허준영

스파이킹 신경망을 위한 심층 신경망 가중치 변환 기법

Deep Neural Network Weight Conversion Method for Spiking Neural Network

위 논문을 공학 석사학위 논문으로 제출함

2022년 6월 일

한 성 대 학 교 대 학 원

컴 퓨 터 공 학 과

컴 퓨 터 공 학 전 공

이 정 수

이정수의 공학 석사학위 논문을 인준함

2022년 6월 일

심사위원장 정 진 만 (인)

심 사 위 원 민 홍 (인)

심 사 위 원 허 준 영 (인)

국 문 초 록

스파이킹 신경망을 위한 심층 신경망 가중치 변환 기법

한 성 대 학 교 대 학 원
컴 퓨 터 공 학 과
컴 퓨 터 공 학 전 공
이 정 수

딥러닝에서 사용하는 기계학습 모델인 심층 신경망은 인간의 뇌 구조를 모방한 덕분에 기존 모델에 비해 강력한 성능을 보이며 기계학습 방법을 비약적으로 발전시켰다. 두뇌의 뉴런이 시냅스를 통해 연결되어 있듯, 신경망의 노드도 가중치를 통해 서로 연결되어 있으며, 입력되는 데이터에 의해 가중치가 조정되며 신경망이 학습된다. 수백만 개의 노드가 동시에 연산을 하는 신경망의 특성상 학습과 운용에 드는 비용이 매우 크며, 자연어 처리 모델과 같은 대규모 모델이 개발되며 이는 필수적으로 해결해야 하는 문제로 대두되었다. 스파이킹 신경망은 뇌 구조뿐만 아니라, 뉴런의 메커니즘도 함께 차용한 신경망으로, 기존의 신경망보다 향상된 전력 효율성 덕분에 합성곱 신경망과 순환 신경망을 잇는 3세대 신경망으로 각광받고 있다. 그러나 아직 관련 연구가 충분히 이루어지지 않았고, 새로운 모델을 도입하기 위해서는 모델을 다시 학습해야 하는 문제 등 높은 진입장벽으로 상용화에 어려움을 겪고 있다. 본 논문에서는 스파이킹 신경망으로 분류 작업을 진행하여

기존 신경망과 성능을 비교하였고, 생성 작업을 진행하여 생성된 이미지를 확인하였다. 실험 결과, 스파이킹 신경망의 성능이 기존 신경망에 성능에 미치지는 못하였으나, 스파이킹 신경망 하이퍼 파라미터 조정 여부에 따라서 추론 시간과 정확도 등의 수치가 큰 폭으로 변화하는 것을 확인하였다. 또한, 기존 신경망의 기학습된 가중치를 스파이킹 신경망으로 전달하여 학습 없이 신경망을 사용할 수 있게 하는 가중치 변환을 설계하였다. 실험 결과, 10% 내외의 정확도 손실이 있었으나 성공적으로 가중치가 변환되었다. 본 실험들을 통해서, 스파이킹 신경망은 하이퍼 파라미터 조정에 따라서 다양한 도메인에 적용할 수 있으나, 보다 효율적으로 사용하기 위해서는 파라미터 최적화 연구가 필요함을 알 수 있었다. 또한, 가중치 변환을 통해서 효율적으로 모델을 전환하고, 빠르게 프로토타입 모델을 구축할 수 있음을 보였다.

【주요어】 딥러닝, 스파이킹 신경망, 분류, 생성, 가중치 변환

목 차

제 1 장 서 론 ····· 1

제 1 절 딥러닝 발전 동향 ····· 1
제 2 절 연구 내용 ····· 2

제 2 장 관련 연구 ····· 4

제 1 절 스파이킹 신경망 ····· 4
1) 뉴런과 시냅스 ····· 4
2) 스파이킹 신경망 ····· 5
제 2 절 오토인코더 ····· 7
제 3 절 ONNX ····· 9

제 3 장 스파이킹 심층 신경망 추론 ····· 12

제 1 절 분류 ····· 12
제 2 절 생성 ····· 15
제 3 절 가중치 변환 ····· 18

제 4 장 실험 ····· 21

제 1 절 실험 환경 ····· 21
1) 하드웨어, 프레임워크 환경 ····· 21
2) 학습 데이터 ····· 22
제 2 절 분류 ····· 23
제 3 절 생성 ····· 25
제 4 절 가중치 변환 ····· 29
1) 기반 모델 학습 및 ONNX 포맷 배포 ····· 29
2) ONNX 호출 및 전처리 ····· 30

3) 가중치 로드 및 추론 ····· 32

제 5 장 결론 및 향후 연구 방향 ····· 34

참 고 문 헌 ····· 35

ABSTRACT ····· 39

표 목 차

[표 3-1] 분류 실험 신경망 파라미터 ····· 14
[표 3-2] 생성 실험 신경망 파라미터 ····· 17
[표 4-1] 데스크톱 실험 환경 ····· 22
[표 4-2] 라이브러리 버전 ····· 22
[표 4-3] 기타 하이퍼 파라미터 ····· 22
[표 4-4] 전결합 신경망 분류 실험 결과 ····· 24
[표 4-5] 합성곱 신경망 분류 실험 결과 ····· 25
[표 4-6] 케라스 기반 모델 추론 결과 ····· 30
[표 4-7] 스파이킹 모델 추론 결과 ····· 33

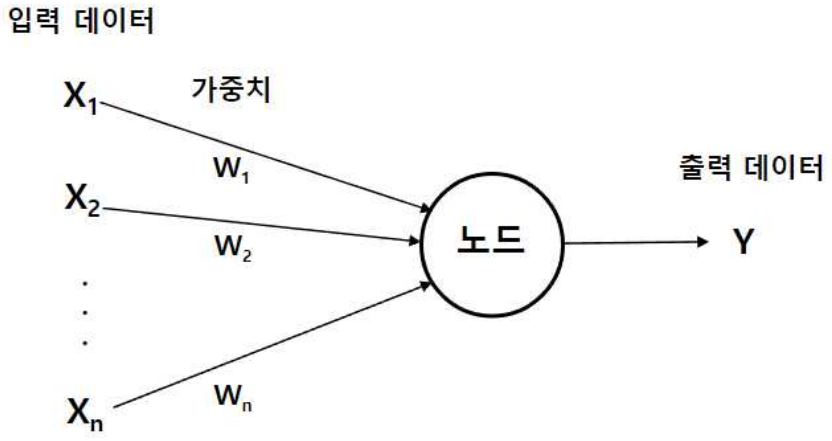
그 림 목 차

[그림 1-1] 신경망 노드 연산 예시 ····· 2
[그림 2-1] 뉴런 모식도 ····· 5
[그림 2-2] 뉴런의 활동 전위 ····· 5
[그림 2-3] 오토인코더 구조 ····· 8
[그림 2-4] Protocol Buffer로 작성된 코드 예시 ····· 10
[그림 2-5] ONNX로 변환된 신경망 모델 ····· 11
[그림 3-1] 시그모이드 함수 그래프 ····· 13
[그림 3-2] 분류 실험 신경망 흐름도 ····· 15
[그림 3-3] 생성 방식에 따른 모델 분류 ····· 16
[그림 3-4] 생성 실험 신경망 흐름도 ····· 17
[그림 3-5] 가중치 변환 과정 ····· 18
[그림 3-6] 변환 전후의 가중치 형태 ····· 19
[그림 4-1] MNIST 데이터 이미지 ····· 23
[그림 4-2] 생성된 MNIST 이미지 ····· 26
[그림 4-3] n_steps에 따른 생성 이미지 SSIM 수치 변화 ····· 27
[그림 4-4] n_steps 별 이미지 생성 결과(n_steps=60, 90, 120) ····· 28
[그림 4-5] Intercepts 별 이미지 생성 결과 ····· 29
[그림 4-6] ONNX 모델 시각화 결과 ····· 30
[그림 4-7] 가중치 관계도 ····· 32

제 1 장 서론

제 1 절 딥러닝 발전 동향

딥러닝은 인간의 두뇌 구조를 모방한 인공신경망(Artificial Neural Network) 모델을 사용하여 데이터를 학습하고 추론한다. 두뇌에서 수많은 뉴런(Neuron)이 서로 얽혀 상호작용하는 것과 유사하게, 신경망도 다수의 노드(Node)가 서로 연결되어 계산 결과를 공유한다. 입력 데이터가 노드에 들어오면 노드의 가중치(Weight)가 곱해져서 다음 노드로 출력되며, 신경망은 이러한 연산을 반복하여 최종 출력값을 결정한다. 최종 출력값의 정오 여부에 따라서 노드의 가중치는 신경망의 최종 출력값이 정답에 가까워지도록 가중치를 조정하여 신경망을 학습시키며, 이러한 가중치가 많아질수록 더욱 복잡하고 비선형적인 데이터도 학습할 수 있게 된다. 최근 딥러닝 모델이 점점 거대해지며 이미지 분류, 소규모 감성 분석 등의 간단한 작업부터 바둑(David et al., 2016)이나 단백질 구조 예측(John et al., 2021), 대규모 문장 번역(Jacob et al., 2018) 등의 복잡한 문제에도 두각을 드러내고 있다. 이러한 작업에 사용되는 거대 모델들은 일반적으로 수억에서 수천억 개의 가중치로 구성되어있으며 다양한 분야에서 기존의 성과를 뛰어넘는 강력한 성능을 발휘한다. 그러나 이와 같은 대규모 모델을 운용하는 데는 매우 높은 비용이 소모된다. 1750억여 개의 가중치를 가지고 있는 자연어 처리 모델인 GPT-3는 한 번의 학습에 약 50억 원에 달하는 비용이 들고(Tom et al., 2020), 이세돌과 대국을 하였던 바둑 인공지능 모델인 알파고(AlphaGo)는 대국 당시 약 1메가와트가량의 전력을 소모하였는데 이는 가정집 100가구가 하루 사용하는 전력 사용량과 맞먹는 수치이다. 이처럼 모델의 대규모화는 운용 전력과 학습 비용 증대 등의 새로운 문제를 야기하였고, 이를 해결하기 위한 연구가 다양한 방면으로 진행되고 있다(이하윤, 신동군, 2019).



[그림 1-1] 신경망 노드 연산 예시

제 2 절 연구 내용

딥러닝 모델이 인간의 두뇌를 모방하였음에도 실제 두뇌와 달리 과도하게 높은 전력 소모량을 보이는 이유는 인공신경망이 뇌의 구조만 모방했을 뿐, 뉴런의 생물학적 메커니즘(Mechanism)이 적용되지 않은 일반 노드를 사용하기 때문이다. 인간의 뉴런은 특유의 메커니즘으로 입출력과 뉴런 상태를 제어하며, 이는 두뇌가 낮은 전력으로도 학습과 기억, 추론을 동시에 할 수 있게 한다. 스파이킹 신경망(Spiking Neural Network)은 이러한 뉴런의 메커니즘을 인공신경망에 적용한 3세대 신경망으로, 합성곱 신경망(Convolution Neural Network)과 순환 신경망(Recurrent Neural Network) 등의 2세대 신경망을 잇는 차세대 모델로 주목받으며 연구가 진행되고 있다(이건명, 2020).

본 논문에서는 스파이킹 심층 신경망의 성능을 검증하기 위해 다양한 작업을 진행하고 결과를 확인한다. 첫 번째로 스파이킹 신경망으로 구축한 분

류 모델을 지도 학습(Supervised Learning)하여 이미지 분류를 진행하고, 동일한 조건으로 실험을 진행한 2세대 분류 모델과 결과를 비교한다. 두 번째로 스파이킹 신경망으로 구축한 이미지 생성 모델을 비지도 학습(Unsupervised Learning)하여 이미지 생성을 진행하고 결과를 확인한다. 분류 실험에서는 검증 데이터 세트에 대한 정확도와 추론 시간 등의 수치를 측정하여 모델의 분류 성능을 비교하며, 생성 실험에서는 실제 생성 이미지를 시각화하고 원본 이미지와 생성 이미지 간 구조적 유사성 지수(Structural Similarity, SSIM)를 측정하여 생성 결과를 비교한다. 또한, 공통으로 모델의 학습 소요 시간과 데이터 추론 및 생성 소요 시간을 측정하여 모델의 연산속도를 비교한다. 세 번째로 학습된 일반 신경망 모델의 가중치를 추출하여 딥러닝 플랫폼들의 중간 표현 포맷인 ONNX(Open Neural Network eXchange) 포맷으로 배포한 뒤, 학습되지 않은 스파이킹 신경망에 삽입하는 실험을 진행하며, 가중치 전달을 통해 스파이킹 신경망 모델의 학습 시간을 단축하고 성능을 향상할 수 있는지 알아본다.

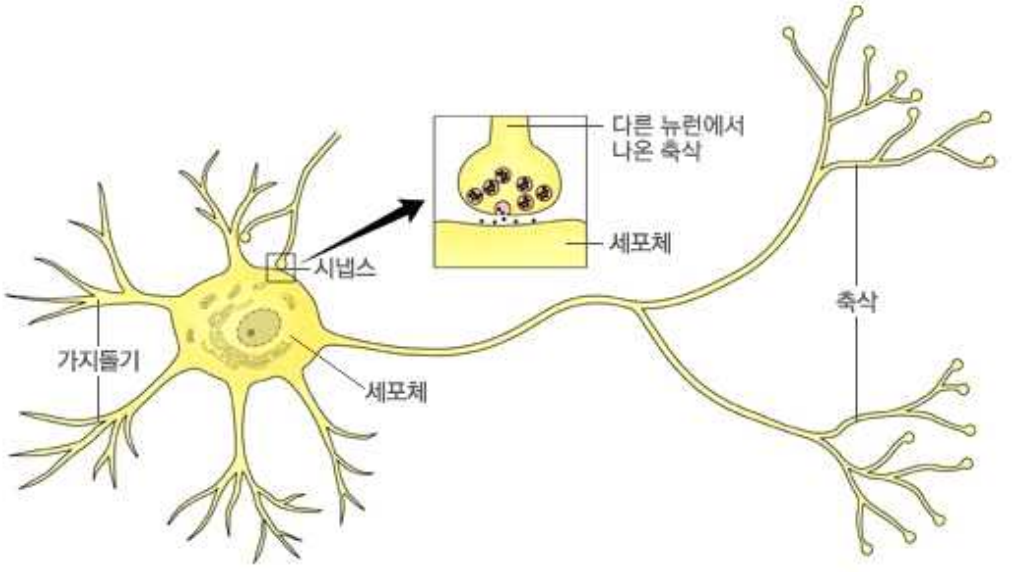
본 논문의 구성은 2장에서 스파이킹 신경망 및 진행할 실험들에 관련된 연구를 소개하며, 3장에서 실험에 사용될 모델의 설계와 가중치 변환 흐름도를 설명한다. 4장에서 제안 기법들에 대한 실험을 진행한 후 결과를 확인하고, 마지막으로 5장에서 결론을 맺고 향후 연구 방향을 제시한다.

제 2 장 관련 연구

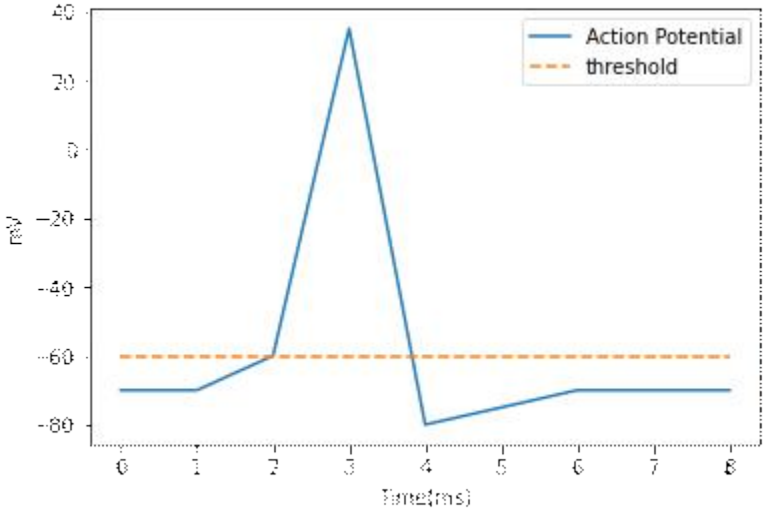
제 1 절 스파이킹 신경망

1) 뉴런과 시냅스

뉴런(Neuron)이란 두뇌의 신경계를 이루고 있는 신경세포를 의미한다. 두뇌에는 무수히 많은 신경세포가 병렬적으로 연결되어 있으며, 이들 간의 상호작용을 바탕으로 의사결정을 내린다. 뉴런에는 다수의 가지돌기(Dendrite)가 뻗어 나가 있으며, 다른 뉴런과의 연결을 담당하는 축삭돌기(Axon)의 말단과 연결되어 있다. 가지돌기와 축삭돌기가 접하는 부분을 시냅스(Synapse)라고 하는데, 이 시냅스를 통해 신경전달물질이 교환되면서 뉴런 간 상호작용이 발생한다(구본웅, 장병탁, 2008). 그림 2-2는 뉴런의 활동 전위 예시를 나타낸다. 뉴런은 세포막 안팎으로 막 전위(Membrane Potential)를 유지하는데, 아무런 자극이 없을 때의 전위를 휴지 전위(Resting Potential)라고 하며 −70mV 내외의 전압을 유지한다. 이때 세포는 활동하지 않는 휴지 상태로 외부의 자극을 대기한다. 휴지 상태인 세포에 시냅스로부터 자극이 오면 탈분극(Depolarization)되어 막 전위가 상승하며, 역치 전위(Threshold Potential)인 −55mV를 넘게 되면 활동 전위(Action Potential)가 발생한다. 활동 전위가 발생하면 세포의 막 전위는 1/1000초 정도의 짧은 시간 동안 급격하게 상승했다가 휴지 전위 이하로 떨어지게 되며, 이후 다시 휴지 상태에 돌입하여 막 전위가 휴지 전위로 상승한다. 이러한 특성으로 활동 전위를 스파이크 전위(Spike Potential)라고도 하며 이는 두뇌가 저전력으로도 고효율, 고성능의 정보 처리를 가능하게 하는 메커니즘이 된다(도우형 등, 2020).



[그림 2-1] 뉴런 모식도



[그림 2-2] 뉴런의 활동 전위

2) 스파이킹 신경망

스파이킹 신경망은 뉴런의 스파이킹 전위 메커니즘을 적용한 인공신경망으로 실제 뉴런의 작동 흐름과 유사하게 데이터 처리에 시간적인 개념을 적용하였다. 노드에 입력 스파이크가 들어오면 시간에 따라 상태가 내부 상태

가 변화하며, 특정 조건을 만족할 때 스파이크가 발화(Firing)하여 다음 노드로 결괏값을 출력한다. 이를 구현한 대표적인 뉴런 모델로 LIF(Leaky Integrate and Fire)가 있다. 넨고에서 구현된 LIF 모델에는 뉴런을 제어하기 위해 다양한 파라미터가 존재한다. 입력이 없을 때 막 전위가 다시 0으로 감쇠하는 시점을 정하는 변수인 tau_rc, 스파이크 이후 막 전위가 0으로 유지되는 시간을 정하는 변수인 tau_ref, 막 전위의 하한값을 결정하는 변수인 min_voltage, 출력 스파이크의 진폭을 결정하는 변수인 amplitude 등이 이에 해당한다. 구체적인 작동방식은, 입력 스파이크가 노드에 전달되면 노드는 입력값과 시냅스 가중치를 곱하여 전위값에 누적한다. 전위값이 역치를 넘으면 다음 노드로 활성 스파이크를 출력하는데, 이때 스파이크는 특정한 크기를 가진 값이 아닌 스파이크 활성 여부를 나타내는 바이너리 값이다. 신경망은 스파이크의 발화 타이밍으로 데이터를 표현하며 최적의 발화 타이밍을 학습하는 것으로 데이터의 표현력을 높인다(이우주 등, 2021). 이처럼 입력 데이터에 따라 선택적으로 발화하는 특징 덕분에 스파이킹 신경망은 기존 신경망에 비해 압도적으로 뛰어난 전력 효율성을 달성하였다.

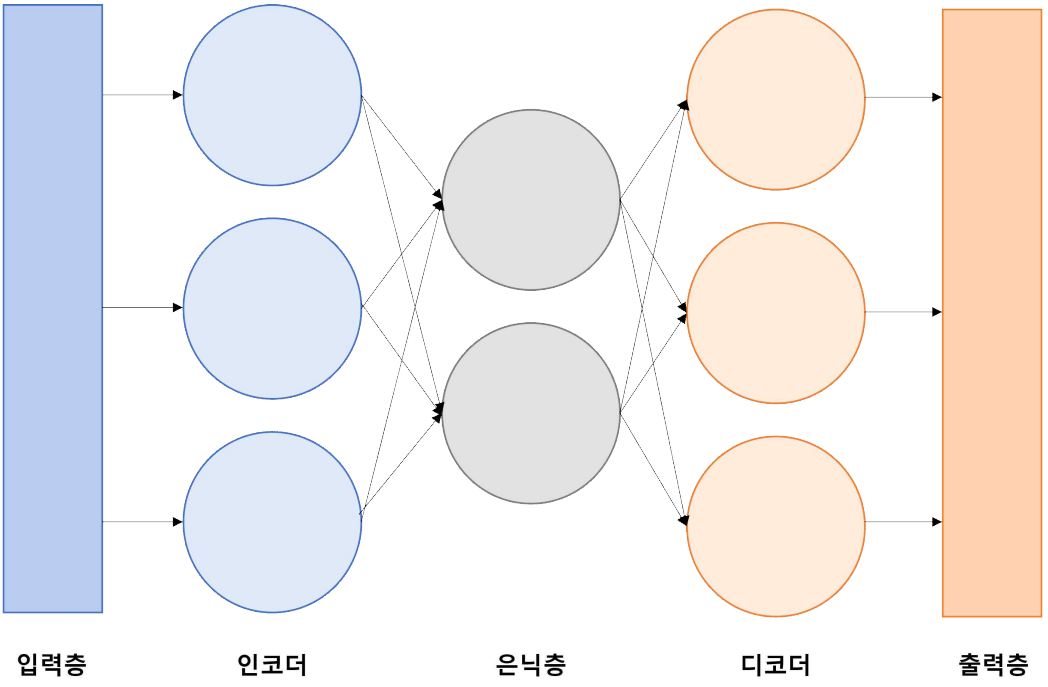
기존의 신경망이 미분을 통한 경사하강법으로 학습하는 데 반해, 스파이킹 신경망은 출력값이 0과 1로 이진화되어있기 때문에 일반적인 방법으로는 미분이 불가능하다. 이러한 스파이킹 신경망을 학습시키는 알고리즘은 대표적으로 STDP(Spike Timing Dependent Plasticity)와 SoftLIF가 있다. STDP는 입력 전의 스파이크와 입력 후의 스파이크 간 시간적 관계를 통해 시냅스 가중치를 학습한다(박성식, 윤성로, 2018). SoftLIF는 스파이킹 신경망 시뮬레이션 프레임워크인 넨고(Nengo)에 적용된 방법으로, 이산적인 LIF의 출력값을 미분 가능한 근사치로 바꾸어 경사하강법으로 학습한 뒤, LIF값으로 추론하는 방식이다(Eric H, Chris E, 2016). 스파이킹 신경망에서 사용되는 뉴런 모델로는 LIF, Hodgkin-Huxley, Izhikevich 등 다양한 모델이 있으며, 현재 가장 많이 사용되는 모델은 LIF 모델이다. LIF 모델의 방정식은

식(1)과 같다. 식에서 v(t)는 막 전위, J(t)는 입력 스파이크 전류, T_{RC}는 막 시간 상수이다.

$$T_{RC}\frac{dv(t)}{dt} = -v(t) + J(t) \qquad (1)$$

제 2 절 오토인코더

오토인코더(AutoEncoder)는 데이터를 압축하는 인코더 구조와 압축된 데이터를 복원하는 디코더 구조로 구성된 모델이며, 비지도 학습에서 널리 사용되는 모델이다. 그림 2-3과 같이 인코더와 디코더 사이에 위치하는 히든 노드는 입력과 출력에 비해 낮은 차원으로 존재한다. 따라서 인코더는 입력 데이터를 보다 낮은 차원의 잠재 벡터로 압축하게 되는데, 디코더가 압축된 데이터를 다시 기존 차원으로 복원한 데이터가 입력 데이터와 동일한 형태를 띠어야 하므로 잠재 벡터에는 데이터를 더욱 잘 표현하는 주성분(Principal Component)들이 남게 된다. 주성분 분석(Principal Component Analysis) 방법과는 데이터를 가장 잘 표현하는 벡터인 주성분을 추출한다는 면에서 유사하지만, 주성분 분석은 선형 연산 기반의 특징 추출 방법이기 때문에 결괏값이 표면적인 특징에 크게 영향을 받는 반면, 오토인코더는 ReLU(Rectified Linear Unit) 등의 비선형 활성화 함수 덕분에 데이터에 내재한 중요한 특징을 추출할 수 있도록 학습되며 이는 곧 데이터의 매니폴드(Manifold)를 추출하도록 학습하는 것과 동일하다고 볼 수 있다.



[그림 2-3] 오토인코더 구조

식 (2)는 오토인코더의 학습 방향이 되는 목적함수(Object Function)이다 (Pierre Baldi, 2012). A는 고차원 데이터를 저차원 표현으로 사상(Mapping)하는 함수이고, B는 반대로 저차원 표현을 고차원 데이터로 사상하는 함수이다. Δ는 두 개의 매개변수 데이터 사이의 거리를 계산하는 재구성 손실함수이다. 결과적으로 x와 $B \circ A(x)$가 같을 때 목적함수가 최솟값을 가지게 되므로 모델은 추출된 주성분으로 입력 데이터의 분포를 재구성하는 방법을 학습하게 된다. 구성이 간단하고 학습이 쉽다는 장점이 있어 다양한 목적으로 사용되는데, 데이터에서 노이즈를 제거하는 목적의 디노이즈(Denoise) 오토인코더(Pascal et al., 2010), 합성곱(Convolution) 노드로 신경망을 구성하여 뛰어난 이미지 처리 성능을 지닌 합성곱 오토인코더(Jonathan et al., 2011), 데이터의 특징 벡터 대신 분포를 추출하도록 학습하는 변분(Variational) 오토인코더(Diederik et al., 2014) 등 다양한 모델이 존재한다.

$$f(x) = \text{argmin}_{A,B} E[\Delta(x, B \circ A(x))] \qquad (2)$$

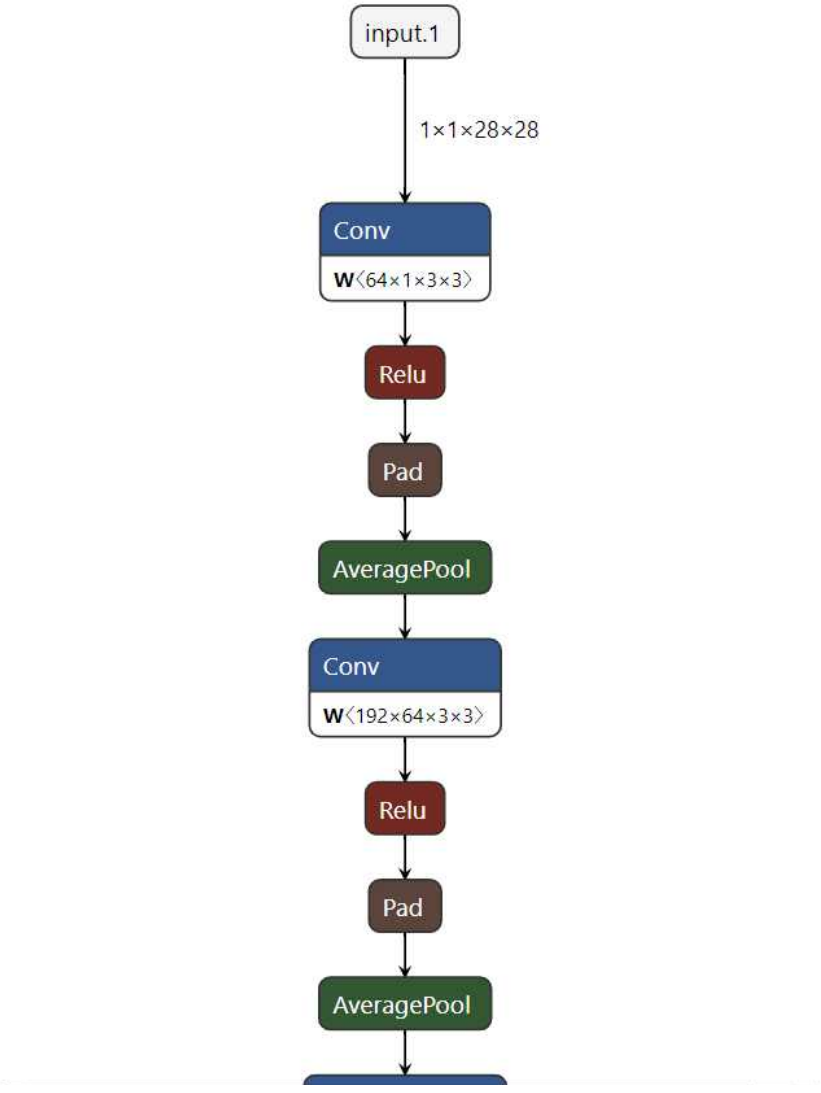
제 3 절 ONNX

ONNX(Open Neural Network eXchange)는 Facebook사와 Microsoft사가 공동 개발한 기계학습 표준 플랫폼이다. 현재 기계학습 분야에서는 Tensorflow, Pytorch, Scikit-Learn 등 다양한 종류의 프레임워크가 사용되고 있다. 이 프레임워크들은 서로 다른 특징을 가지고 있는데, 가령 Facebook사가 개발한 Pytorch는 다양한 종류의 고성능 모델이 사전학습되어 API형식으로 제공되기 때문에 높은 모델 가용성을 자랑하는 반면, Google사가 개발한 Tensorflow는 경량화된 기계학습 라이브러리인 Tensorflow-Lite와 배포 및 추론을 최적화한 형태로 제공하는 Tensorflow Serving 등 유연한 배포 환경을 지원한다. 문제는 이러한 프레임워크들의 모델 그래프 표현 방식이 모두 다르다는 점인데, 이에 따라 모델을 저장하거나 불러오는 확장자가 프레임워크별로 상이하며 이는 통합된 개발 환경 구축을 어렵게 한다.

ONNX는 개별 프레임워크의 모델을 읽어 들여 공통된 중간 표현(Intermediate Representation)으로 변환한다. 중간 표현으로 변환된 모델은 그래프 최적화를 거치고, 최종적으로 생성된 ONNX 그래프는 .onnx 확장자를 가진 ONNX 모델로 저장된다. ONNX의 중간 표현 층은 Google사의 오픈소스 언어인 protocol buffer로 구현되었는데, 구조화된 데이터를 직렬화하여 저장하는 방식을 사용한다. C++, Go, Python, Java 등 다양한 언어를 지원하며 압축률이 높고 직렬화 속도가 빠르기 때문에 모델 저장과 상호 변환이라는 ONNX의 목적과 부합한다(박상민, 허준영, 2020). 또한, 내부적으로 표준 데이터 유형과 기본 연산자가 정의되어 있고, 사용자가 직접 연산자를 정의할 수도 있기 때문에 높은 상호 이식성과 확장성을 지닌다.

```
root@isys313:/home/server2/leejungsoo/proto# cat example.proto
package tutorial;

message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;

  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    required string number = 1;
    optional PhoneType type = 2 [default = HOME];
  }

  repeated PhoneNumber phone = 4;
}

message AddressBook {
  repeated Person person = 1;
}
```

[그림 2-4] Protocol Buffer로 작성된 코드 예시



[그림 2-5] ONNX로 변환된 신경망 모델

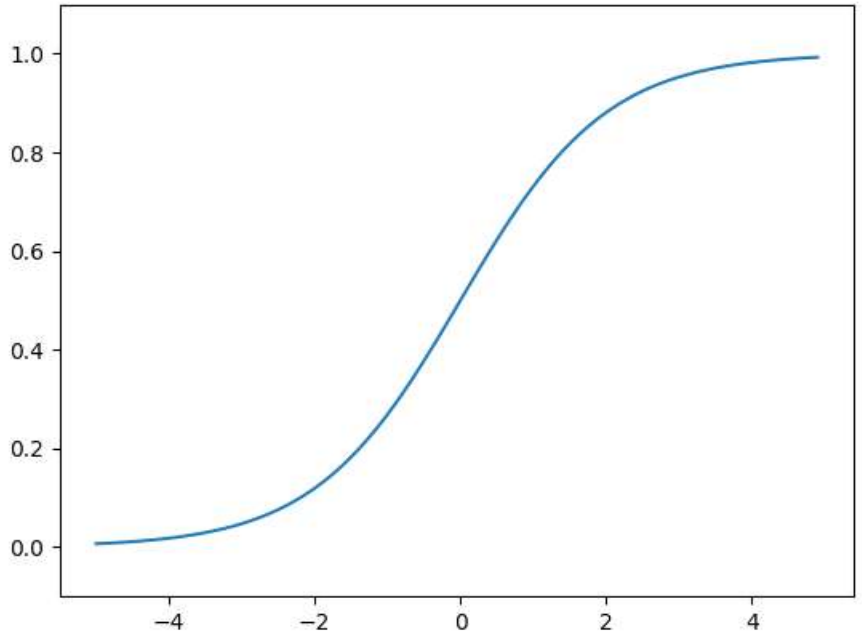
제 3 장 스파이킹 심층 신경망 추론

제 1 절 분류

분류(Classification)란 데이터가 속한 범주를 판단하는 작업으로, 딥러닝의 가장 대표적인 지도학습 방법 중 하나이다. 범주가 있는 레이블(Label)을 가진 데이터로 학습을 진행하고 새로운 데이터의 레이블을 추론하며, 이때 결괏값은 기존에 학습했던 데이터의 레이블 중 하나가 된다. 딥러닝 분야에서 가장 널리 사용되는 방법으로, 음성 인식, 번역, 자율 주행, 의료 이미지 판독 등 다양한 분야에서 기존 방법들을 상회하는 성능을 보이며 활발하게 사용되고 있다.

분류 모델의 말단 층에는 범주 분류를 위한 활성화 함수가 존재하는데, 문제가 이진 분류일 때는 시그모이드 함수를 사용하고 다중 분류일 때는 소프트맥스 함수를 사용한다. 시그모이드 함수는 식 3을 따르며 이를 그래프로 나타내면 그림 3-1과 같다. 함수에 입력되는 오즈(Odds)를 0과 1 사이로 사상시키기 때문에 이상치를 포함하여 어떠한 값이 들어와도 해결이 가능하며, 출력된 값을 0.5와 비교하여 이상이면 양성, 이하이면 음성으로 쉽게 분류할 수 있다. 소프트맥스 함수는 식 4를 따르며, 목표 범주일 확률을 전체 확률로 나누어 계산된다. 따라서 계산 결과가 각각의 범주에 속할 확률들로 나타나며 시그모이드 함수와 동일하게 0과 1 사이의 값으로 사상된다. 이때 결과로 계산된 값을 모두 더했을 때 1이 되기 때문에 확률 계산이 용이하다는 장점이 있다.

$$sigmoid = \frac{1}{1+e^{-x}} = \frac{e^{x}}{e^{x}+1} \qquad (3)$$



[그림 3-1] 시그모이드 함수 그래프

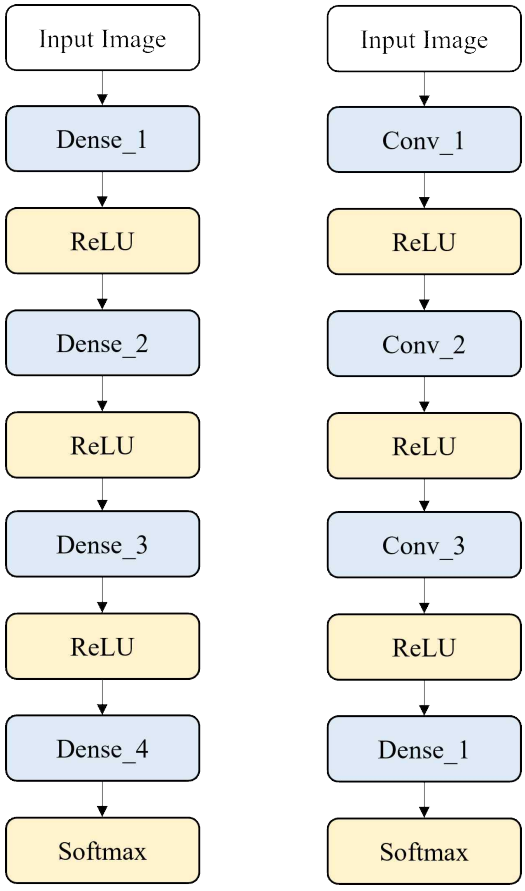
$$y_k = \frac{e^{a_k}}{\sum_{i=1}^{n} e^{a_i}} \qquad (4)$$

본 논문의 분류 실험에서는 스파이킹 신경망과 일반 신경망의 분류 정확도를 비교하여 스파이킹 신경망의 분류 성능을 확인하고, 학습과 추론에 소모되는 시간을 측정하여 신경망의 연산속도를 비교한다. 실험에는 전결합(Fully Connected) 신경망, 합성곱(Convolution) 신경망을 사용하였고, 신경망을 구성하는 하이퍼 파라미터는 표 3-1과 같다. 또한, 신경망의 성능 향상을 위해 모든 층에 활성화 함수를 추가하였다. 각 연산 노드에 이어지는

활성화 함수는 ReLU 함수를 사용하였고, 마지막 전결합 층에는 범주 분류를 위한 소프트맥스 함수를 사용하였다. 스파이킹 신경망도 이와 동일한 구조를 가졌으나 스파이킹 연산을 위해 ReLU 대신 LIF 뉴런 함수를 사용하였다. 합성곱 필터의 크기는 보편적으로 사용되는 3X3을 사용하였고, 첫 번째와 두 번째 층의 스트라이드는 1을 사용하였고, 3번째 층에서 차원 축소를 위해 2를 사용하였다. 패딩은 모든 층에서 0을 사용하였다. 신경망의 흐름도는 그림 3-2와 같다.

[표 3-1] 분류 실험 신경망 파라미터

	전결합 신경망	합성곱 신경망
필터 개수	32, 64, 128, 10	32, 64, 128, 10
필터 크기	X	3X3
스트라이드	X	1, 1, 2
패딩	X	0



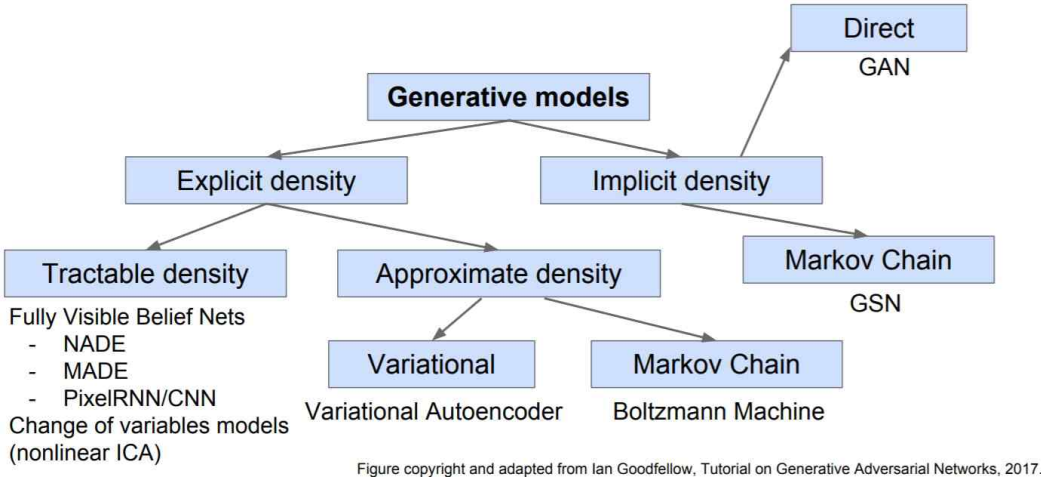
[그림 3-2] 분류 실험 신경망 흐름도

제 2 절 생성

생성(Generation)이란 군집화와 더불어 대표적인 비지도 학습 방법으로, 학습 데이터의 분포와 유사한 데이터를 생성하는 방법이다. 데이터가 가지고 있는 패턴과 분포를 학습하는 것이 목적이기 때문에 학습에 레이블이 필요하지 않으며, 큰 비용이 소모되는 데이터 레이블링 작업이 필요하지 않다는 장점 덕분에 차세대 문제를 해결하기 위한 학습법으로 주목받고 있다. 이미지, 자연어, 음성 등 다양한 데이터를 생성할 수 있기 때문에 이를

사용하여 문맥에 맞는 문장을 생성하거나(Alec et al., 2018), 화가의 화풍을 학습하여 사진을 그림으로 변환하는 등의 응용이 가능하다(Zhu et al., 2017).

그림 3-3(Ian Goodfellow., 2016)과 같이 생성 모델은 크게 두 가지로 분류되는데, 학습 데이터의 분포를 기반으로 데이터를 생성하는 명시적(Explicit) 모델과 학습 데이터를 모르는 상태로 임의의 벡터로 데이터를 생성하는 암시적(Implicit) 모델로 구분된다. 대표적인 명시적 모델은 변분 오토인코더가 있으며, 암시적 모델에는 적대적 생성 신경망(Generative Adversarial Network, GAN)이 있다.



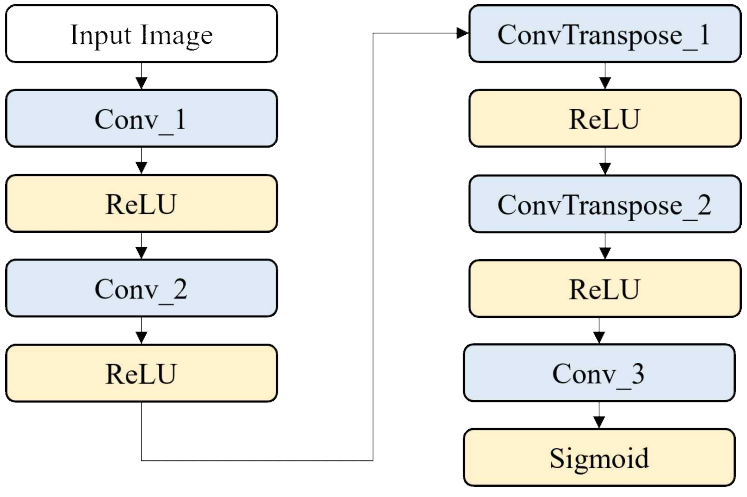
[그림 3-3] 생성 방식에 따른 모델 분류

본 논문의 생성 실험에서는 스파이킹 신경망으로 구축한 오토인코더를 학습하고, 이미지를 생성하여 오토인코더의 성능을 확인한다. 원본 이미지와 생성된 이미지의 차이를 평균제곱오차로 원본 이미지의 복원도를 측정하며, 스파이킹 신경망 파라미터를 조정하여 결과물에 어떤 영향을 미치는지 확인한다. 실험에 사용되는 모델은 총 5개의 합성곱 층으로 구성되었다. 인코더는 데이터 압축을 위한 합성곱 층 2개로 구성하였으며, 디코더는

압축된 표현을 재구축하기 위해 전치 합성곱 층 2개로 구성하였다. 인코더와 디코더을 구성하는 층에는 모두 ReLU 활성화 함수가 포함되어 있다. 말단부에는 픽셀값 표현을 위해 합성곱 층과 시그모이드 함수를 배치하였다. 하이퍼 파라미터는 표 3-2와 같다. 데이터 압축과 복원을 위해 중간층의 차원이 낮도록 필터 개수를 선정하였고, 3X3 크기의 필터를 사용하였다. 효율적으로 피처 맵(Feature Map)을 축소하기 위해 맥스풀링(MaxPooling) 층을 사용하는 대신 합성곱 층의 스트라이드를 2로 설정하였고, 모든 층의 패딩은 1을 사용하였다. 신경망의 흐름도는 그림 3-4와 같다.

[표 3-2] 생성 실험 신경망 파라미터

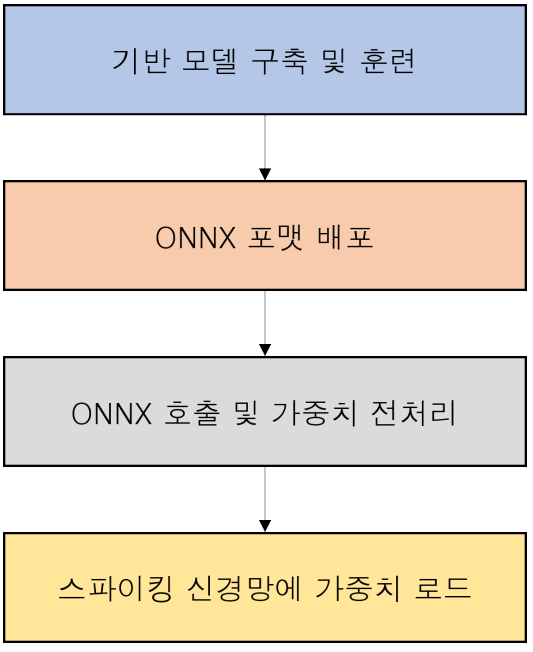
하이퍼 파라미터	값
필터 개수	64, 32, 32, 64, 1
필터 크기	3x3
스트라이드	2, 2, 2, 2, 1
패딩	1



[그림 3-4] 생성 실험 신경망 흐름도

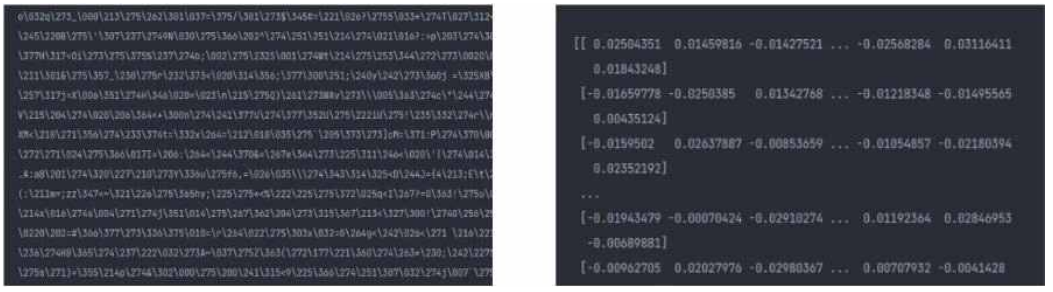
제 3 절 가중치 변환

가중치 변환이란 특정 프레임워크에서 학습된 모델의 가중치를 같은 구조를 가진 다른 프레임워크의 모델로 전달하는 방법이며, 가중치를 전달받은 모델은 추가적인 학습 없이도 기존 모델의 성능을 유지할 수 있다. 딥러닝 모델은 가중치와 연산 종류만 같다면 어느 환경에서도 같은 결과를 출력하는데, 딥러닝 프레임워크의 연산자들은 수학적으로 정의되어 있으므로 연산자 최적화, 부동소수점 정밀도 차이, 난수 등의 특수한 경우를 제외하면 플랫폼 독립적인 연산 결과가 나타난다(이정수, 허준영, 2022). 시아노(Theano), 텐서플로(Tensorflow), 케라스(Keras) 등 다양한 프레임워크가 존재하는 딥러닝 생태계에서 가중치 변환은 특정 프레임워크에 의존하지 않는 환경 구축을 가능케 하고 높은 생산성과 확장성을 제공하며 필수적인 기술이 되었다.



[그림 3-5] 가중치 변환 과정

본 논문의 가중치 변환 실험은 그림 3-5와 같은 과정으로 진행된다. 먼저 가중치 변환 대상이 될 기반 모델을 구축하고 정확도가 어느 정도 수렴할 때까지 학습한다. 이렇게 학습된 모델을 ONNX 포맷으로 배포한다. 이때 케라스 모델은 자체적으로 ONNX 포맷 변환을 지원하지 않기 때문에 먼저 ONNX 모델로 변환하는 과정을 거친다. 모델 변환에는 케라스 모델을 ONNX 모델로 간단하게 변환할 수 있도록 API를 제공하는 tf2onnx 라이브러리를 사용한다. 라이브러리를 사용하여 ONNX 포맷으로 변환된 모델을 배포하는 것으로 모델의 변환과 배포를 마친다. 이후 배포한 ONNX 모델을 스파이킹 신경망이 있는 환경에서 호출하여 가중치를 추출한다. ONNX 모델의 가중치는 모델 그래프 내부의 initializer 컨테이너에 담겨있는데, 데이터가 바이트 형식으로 인코딩되어있기 때문에 ONNX 라이브러리에 포함된 numpy_helper API를 사용하여 numpy.int64 형식의 데이터를 가지고 있는 넘파이(numpy) 배열로 변환한다. 넨고는 모델 가중치로 npz 형식의 데이터를 사용하므로 변환된 넘파이 배열들을 하나의 다차원 리스트로 묶어 npz 파일로 저장함으로써 가중치 전처리를 마친다. 마지막으로 구축된 스파이킹 신경망에 가중치를 로드하여 가중치 변환을 완료한다.



[그림 3-6] 변환 전후의 가중치 형태

가중치 변환이 성공적으로 이루어졌는지 확인하기 위해 가중치를 삽입한 스파이킹 신경망으로 데이터를 추론하여 정확도의 손실률 및 가중치 전달 여

부를 확인한다. 추론 대상은 기존 신경망으로 훈련했던 데이터셋과 동일하며, 본 실험은 MNIST 데이터셋을 대상으로 진행하였고, 모델은 분류 실험에 사용했던 전결합 신경망과 같은 모델을 사용하였다.

제 4 장 실험

제 1 절 실험 환경

1) 하드웨어, 프레임워크 환경

본 논문의 실험을 진행한 하드웨어 환경과 쿠다(CUDA) 버전은 표 4-1과 같다. 구현에 사용된 라이브러리 버전은 표 4-2와 같다. 본 실험에서는 신경망의 학습 가속을 위해 NVIDIA사의 GPU를 사용하였고, 이를 파이썬에서 활용하기 위해 쿠다 라이브러리를 설치하였다. 또한 넨고 프레임워크에서 보다 편리하게 스파이킹 신경망 구축을 지원하는 Nengo_DL 라이브러리를 사용하였다. Nengo_DL은 텐서플로 모듈을 신경망 구축에 사용할 수 있도록 API 형식으로 지원하기 때문에 사용자가 기존에 사용하던 신경망 모듈을 넨고 환경에서도 편리하게 사용할 수 있다는 장점이 있다. 옵티마이저는 Adam(Adaptive Moment Estimation)를 사용하였고 학습률은 기본값인 0.001를 적용하였다. 배치 사이즈는 128을 사용하였고 총 10 에포크로 모델을 학습하였다. n_steps는 스파이킹 신경망에 시계열을 부여하기 위한 변수로 지정된 횟수만큼 데이터를 복사한 뒤 신경망에 주입하게 되며, 횟수가 높을수록 정확도뿐만 아니라 추론 시간도 함께 증가하는 특징이 있다. 본 논문의 분류 실험에서는 1, 15, 30, 45, 60으로 값을 바꿔가며 실험 결과를 비교하였고, 생성 실험에서는 45를 사용하여 실험을 진행하였다.

[표 4-1] 데스크톱 실험 환경

CPU	Intel i7-10700K
GPU	NVIDIA RTX3060
CUDA	11.0
RAM	64G

[표 4-2] 라이브러리 버전

Tensorflow	2.4.0
ONNX	1.7.0
Nengo	3.1.0
Nengo_DL	3.4.0
tf2onnx	1.10.1
Numpy	1.19.5

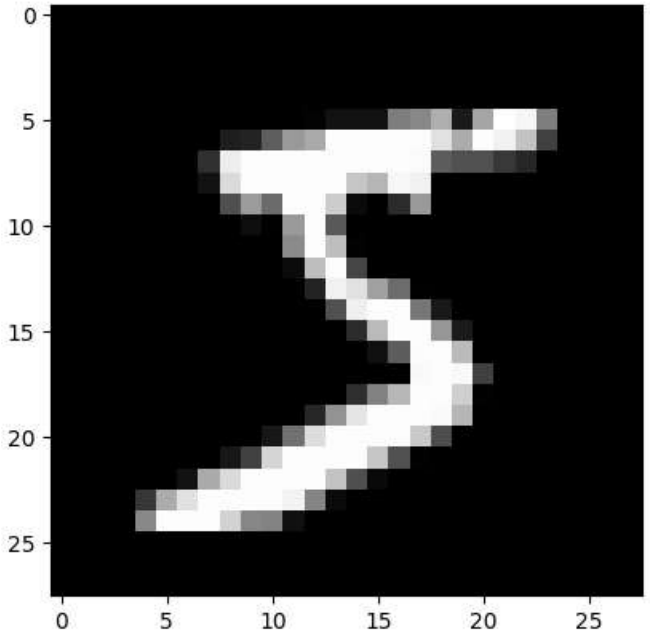
[표 4-3] 기타 하이퍼 파라미터

Optimizer	Adam
Learning Rate	0.001
Batch Size	128
Epoch	10
n_steps	1, 15, 30, 45, 60 / 45

2) 학습 데이터

본 논문에서는 MNIST 데이터셋을 사용하여 실험을 진행하였다. MNIST (Modified National Institute of Standards and Technology database)는 손으로 쓰인 숫자 이미지 데이터베이스로, 기계학습 분야의 학습과 검증에 널리 사용된다. 학습 데이터 60,000장과 검증 데이터 10,000장으로 구성되어 있으며, 각 이미지는 28X28 픽셀 크기를 가지고 있다. 이미지에는 0에서 9까지의 숫자 중, 범주에 해당하는 숫자가 쓰여 있다. 텐서플로와 파이토치 등, 여러 프레임워크에서 API로 호출할 수 있기 때문에 데이터를 저장하고 적재하는 과정을 생략할 수 있으며, 대부분의 모델에서 좋은 성능이 나타나므로 모델이 잘 구현되었고 정상적으로 작동했는지 쉽게 판단하는 데 도움을

준다. 본 실험에서는 원활한 학습을 위해 이미지에 최대-최소 정규화를 적용하였다.



[그림 4-1] MNIST 데이터 이미지

제 2 절 분류

전결합 신경망의 분류 실험 결과는 표 4-4와 같다. 정확도의 경우, 케라스로 구축한 모델이 넨고로 구축한 스파이킹 신경망 모델에 비해 높은 결과를 보였다. 넨고 모델들을 비교하였을 때 n_steps가 증가하며 정확도도 함께 증가하는 추이를 보였다. 특히 n_steps가 1일 때 학습 중 손실값은 지속적으로 감소하였음에도 정확도가 통계적 유의성을 띠지 않으며, 훈련 시간에 영향을 미치지 않는다는 점을 미루어 볼 때, n_steps는 훈련이 끝난 뒤 추론 과정에서만 사용된다는 사실을 알 수 있다. 이러한 결과는 기학습된

스파이킹 신경망 모델을 미세 조정하여 고성능의 하드웨어가 탑재된 서버부터, 낮은 연산량이나 실시간 시스템이 요구되는 임베디드 환경 등 다양한 목적에 맞춰서 사용할 수 있음을 의미한다.

[표 4-4] 전결합 신경망 분류 실험 결과

	Keras	Nengo (1)	Nengo (15)	Nengo (30)	Nengo (45)	Nengo (60)
정확도(%)	97.02	9.80	81.14	91.74	92.49	93.61
훈련 시간(초)	9.89	29.34	30.72	30.83	30.75	29.37
추론 시간(초)	0.50	0.87	3.28	5.98	8.21	11.35

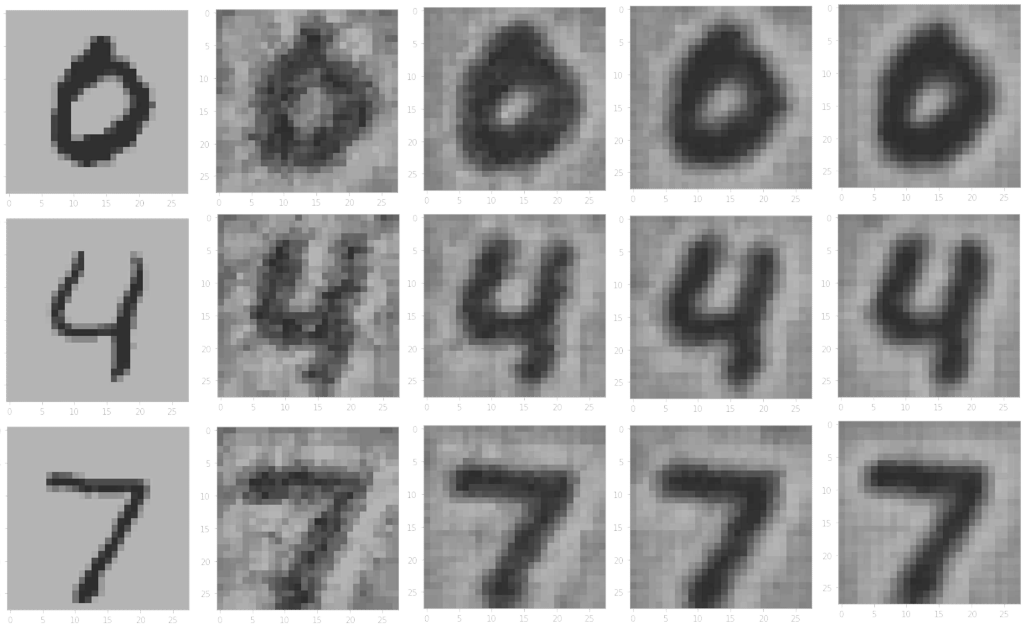
합성곱 신경망의 실험 결과도 이와 유사한 추이를 보인다. 높은 연산량이 요구되는 합성곱 층으로 구성되어있어 전결합 신경망에 비해 약 3~4배의 훈련 시간이 소모되었으며, 약 6배에 달하는 추론 시간이 소모되었다. 정확도의 경우, 최고 정확도는 96.50%로 케라스 모델의 정확도와 유사한 수준의 정확도를 달성하였으며, n_steps가 30을 넘어가면 유의미한 향상을 보이지 않았다. 그러나 추론 시간은 선형적으로 증가하였으며 n_steps가 60일 때의 추론 시간은 1일 때의 추론 시간에 비해 217.7%에 달했는데, 이는 전결합 신경망의 130.4%에 비해 약 87.3% 높은 수치이다. 따라서 n_steps 상승치 당 연산 부하가 높기 때문에 더욱 정밀한 성능 최적점이 요구된다.

[표 4-5] 합성곱 신경망 분류 실험 결과

	Keras	Nengo(1)	Nengo (15)	Nengo (30)	Nengo (45)	Nengo (60)
정확도 (%)	98.65	9.80	92.03	96.50	95.52	96.29
훈련 시간(초)	38.68	108.63	108.04	113.76	113.62	125.60
추론 시간(초)	3.10	3.39	20.26	37.91	56.01	73.83

제 3 절 생성

생성된 이미지는 그림 4-2와 같다. 가장 왼쪽 열은 생성 대상이 된 원본 이미지이고, 두 번째 열부터 n_steps가 5, 15, 30, 45인 경우이다. 같은 행에 있는 이미지들은 모두 같은 원본 이미지를 대상으로 생성한 결과물이다. 시각적으로 판단하였을 때, n_steps가 올라갈수록 점점 이미지가 명확해졌으며, 15 이상인 경우부터는 유의미한 변화가 적은 것으로 확인되었다.



[그림 4-2] 생성된 MNIST 이미지

이러한 차이를 정량적으로 측정하기 위해, 원본 이미지와 생성된 이미지 간 SSIM 수치를 측정하였다. SSIM이란 이미지 간 오차를 인간의 시각적 화질 차이 관점에서 측정하는 방법으로, 휘도(Luminance), 대비(Contrast), 구조(Structural) 측면에서 이미지의 품질을 평가한다. 수식은 식 5와 같으며, 우변을 구성하는 각 항은 각각 식 6, 7, 8과 같다. C_1은 (0.01 * L)을 제곱한 값이며, 이때 L은 대상 이미지가 0~255의 픽셀값을 가질 때 0~255의 값을, 0~1의 픽셀값을 가질 때 1을 사용한다. C_2는 (0.03 * L)을 제곱한 값을, C_3는 $C_2/2$의 값을 가진다.

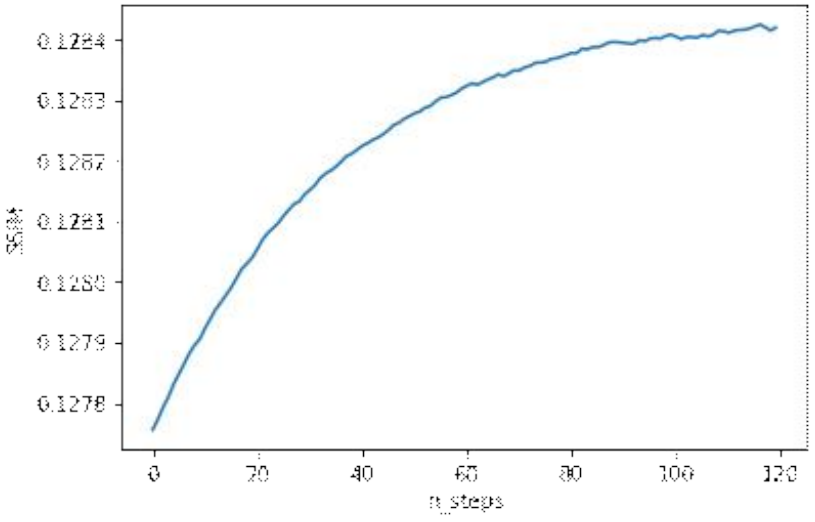
$$SSIM(x,y) = l(x,y)c(x,y)s(x,y) \qquad (5)$$

$$l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^{\,2} + \mu_y^{\,2} + C_1} \qquad (6)$$

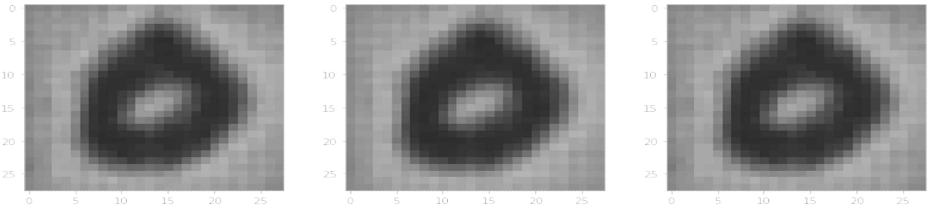
$$c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^{\,2} + \sigma_y^{\,2} + C_2} \qquad (7)$$

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3} \qquad (8)$$

측정 결과는 그림 4-3과 같다. 측정은 생성 실험에 사용하였던 이미지 0을 대상으로 하였고, n_steps가 1인 경우부터 120까지의 생성 결과에 따른 SSIM 수치 변화 그래프로 나타내었다. 측정 결과, n_steps가 상승할수록 SSIM 수치가 명확하게 증가하는 것을 보였으며, n_steps 수치가 낮을 때 급격하게 상승하다가 점차 증가 폭이 감소하는 것을 알 수 있었다. 이는 SSIM 수치 향상이 그림 4-2에서 나타난 생성 이미지 변화와 동일한 추이를 띠며, 그림 4-4와 같이 특정 수치를 초과하면 시각적으로 큰 변화를 인지하기 어렵다는 것을 의미한다.

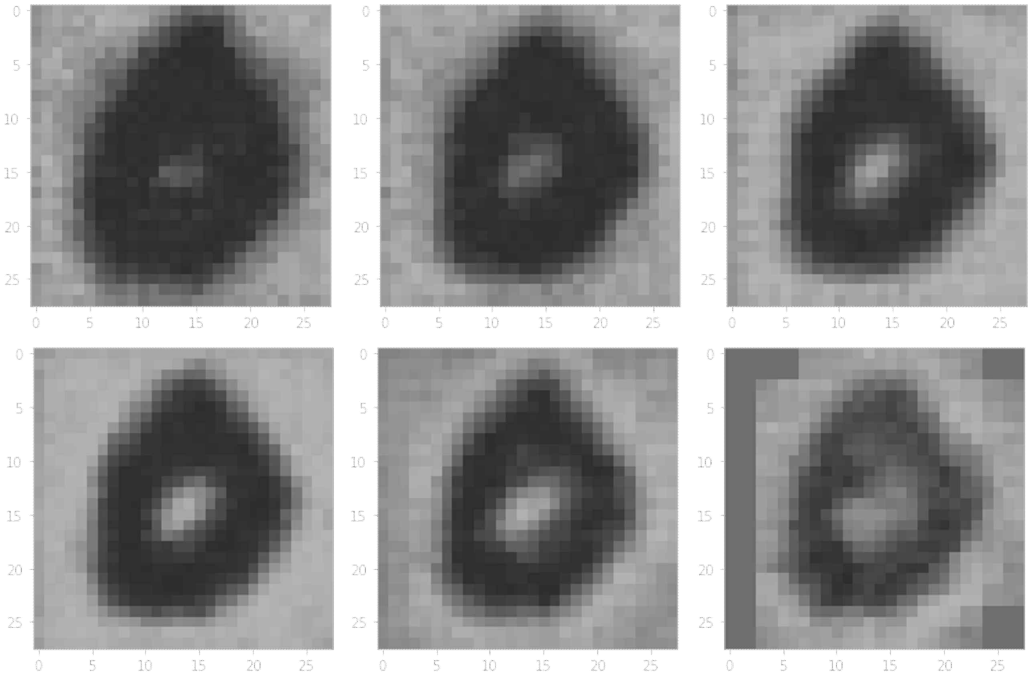


[그림 4-3] n_steps에 따른 생성 이미지 SSIM 수치 변화



[그림 4-4] n_steps 별 이미지 생성 결과 (n_steps=60, 90, 120)

그림 4-5는 스파이킹 신경망의 매개변수인 Intercepts를 조정하며 생성한 결과이다. Intercepts는 스파이킹 뉴런이 발화하기 위한 최솟값을 결정하는 변수이며, 0을 기본값으로 가지고 있다. 좌상단부터 우하단으로 순서대로 Intercepts가 -2, -1.5, -1, -0.5, 0, 0.5일 때의 생성 결과이다. 실험 결과, Intercepts가 너무 맞게 설정되면 정답 픽셀뿐만 아니라 인접한 픽셀도 모두 활성화가 되었고, 반대로 너무 높게 설정되면 전체적으로 이미지가 불명확해지는 양상을 보였다. 본 실험에서는 Intercepts가 -0.5일 때 시각적으로 가장 원본 이미지에 근접한 결과를 보였다. 이러한 결과는 스파이킹 신경망의 하이퍼 파라미터 탐색 공간이 기존 신경망에 비해 훨씬 방대하며, 최적의 결과를 얻기 위해서는 더욱 많은 자원과 연구가 요구된다는 것을 시사한다.



[그림 4-5] Intercepts 별 이미지 생성 결과

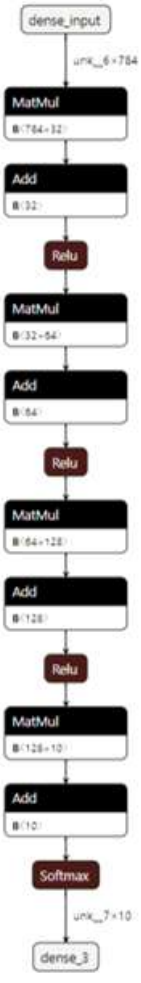
제 4 절 가중치 변환

1) 기반 모델 학습 및 ONNX 포맷 배포

먼저 가중치 전달을 위해서 기반이 되는 모델을 케라스 프레임워크로 구축하고 학습하였다. 학습이 끝난 모델로 검증 데이터를 추론한 결과는 표 4-6과 같다. 훈련이 끝난 모델을 저장하기 위해 tf2onnx 라이브러리를 사용하여 ONNX 모델로 변환한 뒤, ONNX API를 사용하여 모델을 onnx 확장자를 가진 모델로 저장한다. 저장된 모델을 그래프 시각화 도구인 네트론(Netron)으로 시각화한 결과는 그림 4-6과 같다.

[표 4-6] 케라스 기반 모델 추론 결과

측정 항목	측정값
정확도(%)	96.95

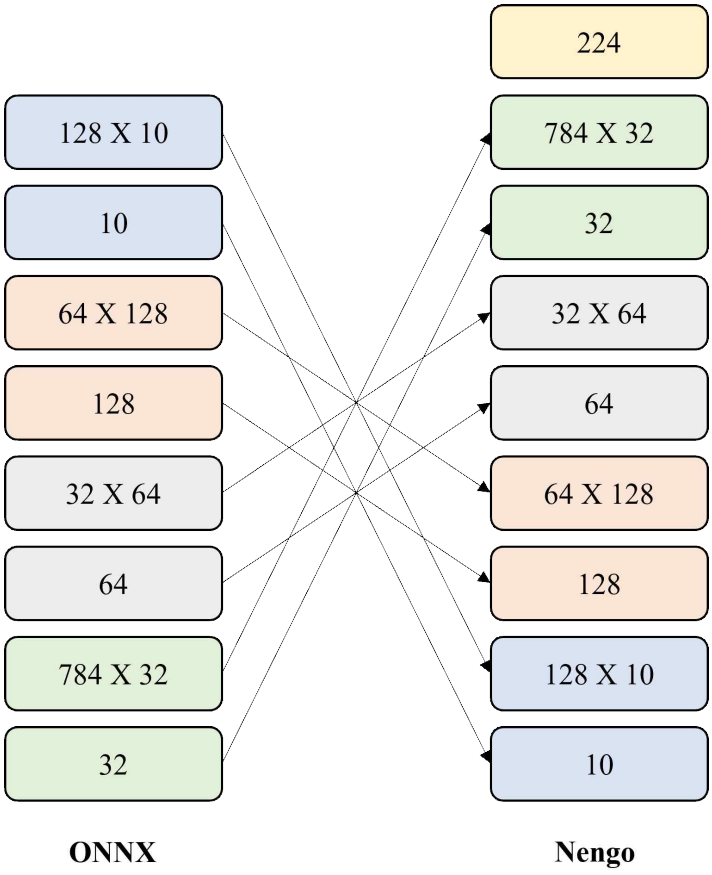


[그림 4-6] ONNX 모델 시각화 결과

2) ONNX 호출 및 전처리

저장된 ONNX 모델은 전처리 과정을 거쳐 넨고에서 사용하는 NPZ 형식의 가중치로 변환된다. ONNX 모델의 가중치는 모델 그래프 내부의 initi

alizer 컨테이너에 키-값 형태로 저장되어있다. 이때 주의할 점은, 그래프 순서와는 반대로 말단 층의 가중치부터 역순으로 저장되어있으므로 순서를 재정렬해야 한다. 본 실험에 사용된 ONNX 모델의 가중치와 넨고에서 요구하는 가중치 관계를 그림으로 표현하면 그림 4-7과 같다. 그림에서 좌측 블록들은 ONNX 모델 가중치 순서이고 오른쪽은 넨고에서 요구하는 가중치 순서이다. n X n 형태로 표현된 블록은 입력 차원 X 출력 차원 형태의 가중치를 요구하는 전결합 층 가중치이고, 바로 이어지는 블록은 연산 결과에 더해지는 편향(Bias)이다. 하나의 층에 존재하는 가중치와 편향치는 함께 연산 되는 요소들이기 때문에 가중치 재정렬 시, 이를 고려하여 순서를 배치하여야 한다. 우측 블록에서 224로 표현된 블록은 스파이킹 신경망에서 사용하는 가중치 중 하나인 neurons이다. 이 가중치는 뉴런의 발화 시점을 조정하는 역할을 하며, 학습을 통해서 데이터에 맞는 최적의 발화 타이밍을 학습하게 된다. neurons는 스파이킹 신경망 고유의 변수이므로 기존 신경망 학습을 통해서는 얻을 수 없으므로 임의의 값으로 대체하였는데, 보편적으로 훈련이 완료된 neurons 가중치의 평균이 1, 오차범위가 0.1 내외였으므로 본 실험에서도 전체 값을 1로 초기화하여 사용하였다. 이렇게 생성한 neurons 가중치를 첫 번째 인덱스에 삽입한 뒤, NPZ 형식으로 압축하여 전처리를 마친다.



[그림 4-7] 가중치 관계도

3) 가중치 로드 및 추론

저장된 가중치를 스파이킹 신경망 모델에 적재하여 검증 데이터를 추론한 결과는 표 4-7과 같다. 실험 결과, n_steps에 따라서 일정 비율 정확도 손실이 있지만 가중치 변환이 성공적으로 이루어졌다는 것을 알 수 있다. n_steps가 40 이상인 경우, 케라스 모델 정확도의 10%~13%에 해당하는 정확도 손실이 발생하는 것을 확인하였다. 이러한 가중치 손실 원인은 미학습된 neurons 가중치, 최적화되지 않은 스파이킹 신경망 하이퍼 파라미터, 상이한 추론 메커니즘 등 다양한 원인이 있을 것으로 판단되며,

이를 해결하기 위해서 다양한 조건에서 최적의 파라미터 조합을 찾는 연구가 필요하다.

[표 4-7] 스파이킹 모델 추론 결과

n_steps	10	40	70	100	130
정확도(%)	69.19	84.96	86.4	87.22	87.26
추론 시간(초)	2.5	7.53	12.8	18.57	23.81

제 5 장 결론 및 향후 연구 방향

본 논문은 스파이킹 신경망 모델의 성능을 확인하기 위해 분류와 생성 실험을 진행하였고, 케라스로 대표되는 기존 신경망 모델의 가중치를 스파이킹 신경망 모델로 전달하는 가중치 변환을 제안하였다. 분류 실험 결과, 넨고로 구축한 스파이킹 신경망 모델은 케라스 모델에 비해 정확도, 훈련 시간, 추론 시간 모두 떨어지는 결과가 나타났으나, 하이퍼 파라미터인 n_steps 수치에 따라서 비등한 결과가 나타나기도 하였다. 생성 실험 결과, 이와 마찬가지로 n_steps와 Intercepts 변수의 수치에 따라서 생성 결과가 유의미하게 변화하는 것을 확인하였다. 이러한 실험 결과를 통해 스파이킹 신경망이 파라미터 변화에 민감하게 반응하며, 이를 응용하여 임베디드 환경부터 고성능 서버 등 다양한 환경에 맞게 조정하여 사용할 수 있지만, 더욱 높은 성능을 발휘하기 위해선 대규모 탐색 공간에서 최적의 파라미터 조합을 찾기 위한 연구가 진행되어야 한다는 점을 알 수 있었다. 또한, 신경망 간 가중치 변환을 설계하여, 스파이킹 신경망을 훈련하지 않아도 기존 신경망의 가중치를 변환하여 높은 성능을 발휘하는 방법을 제안하였다. 가중치 변환 방법을 사용하면 모델의 프로토타입을 더욱 빠르고 쉽게 구현할 수 있게 되며, 자연어 처리 모델과 같은 대규모 모델의 가중치를 훈련하는데 드는 자원을 절약할 수 있게 된다.

향후 개선할 사항으로, 본 논문에서 다루었던 n_steps나 Intercepts를 포함한 다양한 스파이킹 신경망 파라미터의 최적화가 있다. 파라미터에 관한 연구를 통해 신경망의 규모나 작업의 종류에 따른 최적의 파라미터를 찾는 연구를 진행할 계획이다. 또한, 본 논문의 가중치 변환 실험은 전결합 신경망 한 가지에 대해서만 진행을 하였기 때문에, 합성곱 신경망과 순환 신경망 등 다양한 종류의 신경망 모델을 대상으로 하는 가중치 변환을 구현할 예정이다.

참 고 문 헌

1. 국내문헌

이하윤, 신동군. (2019). ARM 기반 IoT 장치에서 효율적인 딥 러닝 수행을 위한 BLAS 및 신경망 라이브러리의 성능 및 에너지 비교. 『정보과학회논문지』, 46(3), 219-227.

이건명. (2020). 『스파이킹 뉴런 모델의 동작과 스파이킹 신경망의 학습』. 한국정보과학회: 정보과학회지

구본웅, 장병탁. (2008). 다중 시냅스가 뉴런의 반응에 미치는 영향에 대한 컴퓨터 시뮬레이션 연구. 『한국컴퓨터종합학술대회』, 35(1), 222-225.

도우형, 박성식, 윤성로. (2020). 깊은 스파이킹 신경망 학습 방법의 최신 연구 동향. 『대한전자공학회 하계학술대회 논문집』, 1829-1834.

이우주, 오광일, 이재진. (2021). Spiking Neural Network 기반 인공지능 반도체: 이것이 답일까? 『[정보통신기획평가원] 주간기술동향』 2010

박성식, 윤성로. (2018). STDP 알고리즘과 스파이크 간의 시간적 상호작용에 따른 SNN의 학습 성능 및 시간 분석. 『정보과학회 컴퓨팅의 실제 논문지 』, 24(9), 482-486.

박상민, 허준영. (2020). ONNX기반 스파이킹 심층 신경망 변환 도구.

『한국인터넷방송통신학회논문지』, 20(2), 165-170.

이정수, 허준영. (2022). 스파이킹 신경망 추론을 위한 심층 신경망 가중치 변환.『스마트미디어저널』, 11(3), 26-30.

2. 국외문헌

David et al. (2016). Matering the game of Go with deep neural networks and tree search. Nature, 529, 484-489

John et al. (2021). Highly accurate protein structure prediction with AlphaFold. Nature, 596, 583-589

Jacob, D., Ming-Wei, C., Kenton, L., Kristina, T. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805

Tom et al. (2020). Language Models are Few-Shot Learners. arXiv:2005.14165

Eric, H., Chris, E. (2016). Training Spiking Deep Networks for Neuromorphic Hardware. arXiv:1611.05141

Pierre, B. (2012). Autoencoders, Unsupervised Learning, and Deep Architectures. JMLR: Workshop and Conference Proceedings, 27, 37-50

Pascal, V., Hugo, L., Isabelle, L. (2010). Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. Journal of Machine Learning Research, 11, 3771-3408

Jonathan, M., Ueli, M., Dan, C. (2011). Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction. ICANN,

6791, 52-59

Diederik, P, K., Max, W. (2013). Auto-Encoder Variational Bayes. arXiv:1312.6114

Alec et al. (2018). Improving Language Understanding by Generative Pre-Training. arXiv:2012.11747v3

Jun-Yan Zhu. (2017). Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. arXiv:1703.10593

Ian Goodfellow. (2016). Generative Adversarial Networks. Nips Tutorial

ABSTRACT

Deep Neural Network Weight Conversion Method for Spiking Neural Network

Lee, Jung-Soo

Major in Computer Engineering

Dept. of Computer Engineering

The Graduate School

Hansung University

The deep neural network, a machine learning model used in deep learning, has improved the field of machine learning by leaps and bounds by mimicking the structure of the human brain, showing powerful performance than existing models. Just as neurons in the brain are connected through synapses, the nodes of a neural network are also connected to each other through weights, and the weights are adjusted according to the input data, and the neural network is trained. Due to the nature of neural networks in which millions of nodes operate simultaneously, the cost of training and operation is very high, and as large scale models such as natural language processing models are developed, this has emerged as an essential problem to be solved. The spiking artificial neural network is a neural network that uses not only the brain structure but also the mechanism of neurons. Thanks to improved power efficiency compared to the second generation neural

network, it is in the spotlight as a third generation neural network that connects Convolutional neural networks and Recurrent neural networks. However, related research has not yet been sufficiently conducted, and commercialization is difficult due to high barriers to entry, such as the need to retrain the model to use a new model. In this paper, we performed classification task with a spiking neural network, compared the performance with the second generation neural network, and confirmed the generated image by proceeding with the generation task. As a result of the experiment, it was confirmed that the performance of the spiking neural network did not reach that of the second generation neural network, but the numerical values such as inference time and accuracy change significantly depending on whether or not the spiking neural network hyperparameter is adjusted. In addition, we designed a weight transformation that allows the neural network to be used without training by transferring the trained weights of the existing neural network to the spiking neural network. As a result of the experiment, there was a loss of accuracy of about 10%, but the weight was successfully converted. Through these experiments, it was found that the spiking neural network can be applied to various domains according to hyperparameter adjustment, but parameter optimization studies are needed to use it more efficiently. In addition, it was shown that it is possible to efficiently convert a model and build a prototype model quickly through weight transformation.

【KEYWORD】 Deep learning, Spiking Neural Network, Classification, Generation, Weight Conversion