1. 연구 개요

1.1 문제 설정

이 문서는 시냅스 가중치 업데이트를 **로컬 정책(가중치 정책, weight policy)** 으로 보고 이를 **강화학습(Actor–Critic)** 으로 학습시키는 SNN 실험 설계를 정리한다. 전체 내용은 실제 코드 구현과 1:1로 대응되는 **최종 설계안**만을 포함하며, 실험에 사용하지 않는 옵션이나 후보 아이디어는 모두 제거하였다.

전통적인 **STDP** 는 pre/post 스파이크의 시간차만을 보고 고정된 규칙에 따라 가중치 변화를 결정하는 완전히 하드코딩된 학습 규칙이다.

이 프로젝트에서는 다음과 같이 본다.

- 각 시냅스는 로컬 스파이크 히스토리, 현재 가중치, 레이어 위치 등 구조 정보를 입력으로 받는다.
- 이 로컬 상태를 입력으로 받아 **실수 스칼라 액션(정규화된 가중치 변화량)** 을 출력하는 **Gaussian Actor** 를 둔다.
- 동일한 로컬 상태를 입력으로 받아 **기대 누적 보상(baseline)** 을 추정하는 Critic 을 둔다.
- 즉 시냅스 업데이트는 고정 STDP 규칙이 아니라 **로컬 상태 → 실수 액션** 을 내는 작은 RL 에이전트로 모델링된다.

이 설계를 기반으로 다음 네 가지 실험 시나리오를 비교·분석한다.

1. 완전 비지도 단일 정책 (시나리오 1.1)
2. 완전 비지도 두 정책 (시나리오 1.2)
3. 준지도 단일 정책 분류 (시나리오 2)
4. 완전지도 gradient mimicry (시나리오 3)

네 시나리오는 **동일한 로컬 상태, CNN+MLP 구조, on-policy Actor–Critic 업데이트** 를 공유하되

- 사용하는 SNN 아키텍처
- 정책의 개수(1개 vs 2개)
- 전역 보상(완전 비지도 vs 준지도 분류 vs gradient 기반)

에서 서로 다르게 설계된다. 특히 pre/post 이벤트에 대해서는 **단일 정책**이 둘 다 처리하며, pre/post 구분은 **이벤트 타입 원핫 벡터**로 로컬 상태에만 반영한다. 별도의 pre 전용, post 전용 정책은 사용하지 않는다.

강화학습 관점에서 본 프로젝트의 기본 학습법은 **이미지별 전역 보상을 사용하는 on-policy Monte Carlo Actor–Critic** 이며, REINFORCE with baseline 아이디어를 기반으로 한 **PPO 스타일의 클리핑된 정책경사(MC 기반 PPO)** 를 전 시나리오에서 공통으로 사용한다. 별도의 "순수 REINFORCE 모드"는 두지 않고, 이 문서에서 정의하는 RL 학습은 모두 PPO 로 이해한다.

1.2 실험 시나리오 인덱스

섹션 번호와 실험 시나리오 번호의 대응은 다음과 같다.

- 3장: 실험 1 (시나리오 1.1, 완전 비지도 단일 정책)
- 4장: 실험 2 (시나리오 1.2, 완전 비지도 두 정책)
- 5장: 실험 3 (시나리오 2, 준지도 단일 정책 분류)
- 6장: 실험 4 (시나리오 3, 완전지도 gradient mimicry)

공통 구성 요소는 2장에서 정리한다.

2. 공통 구성 요소

2.1 데이터셋과 입력 인코딩

- 데이터셋은 **MNIST** 를 사용한다.
- **이미지 1장 = 에피소드 1개** 로 정의한다.
- 각 시나리오마다 **시뮬레이션 타임스텝 수** T 를 별도로 두고 모두 CLI 하이퍼파라미터로 노출한다.

픽셀 값 $x_p \in [0,1]$ 에 대해 다음과 같이 **Poisson 인코딩**으로 스파이크를 생성한다.

- 타임스텝 $t = 1, \dots, T$
- 픽셀 p 에 대해 스파이크 $s_p(t) \in 0,1$ 를

$$P\big(s_p(t)=1\big) = \lambda_p \Delta t$$

로 두고 샘플링한다. 여기서 λ_p 는 픽셀 값에 비례하는 발화율이다.

정리하면 한 에피소드 동안

1. 이미지 1장을 일정 시간 T 동안 SNN 에 흘려보내며
2. 발생하는 모든 시냅스 이벤트를 수집하고
3. 에피소드 전체에 대한 전역 보상 R 을 계산한 뒤
4. 그 에피소드에서 수집한 궤적을 포함해 RL 업데이트에 사용한다.

강화학습 관점에서 에피소드는 항상 **이미지 단위**로 정의된다. 실제 구현에서는 여러 이미지를 한 번에 처리하여 **이미지 단위 미니배치**를 구성하고, 이 미니배치 전체를 대상으로 Actor–Critic 업데이트를 수행할 수 있다. 각 에피소드는 자기 이미지에 대한 전역 보상 R 을 따로 가지므로, 서로 다른 이미지를 같은 미니배치에 포함하더라도 보상의 의미가 희석되지 않고 오히려 gradient 분산이 줄어 학습 안정성이 증가한다.

2.2 LIF 뉴런과 스파이크 히스토리

모든 SNN 뉴런은 표준 **Leaky Integrate-and-Fire (LIF)** 모델을 따른다. 막전위 $v(t)$ 에 대해

$$\tau_m \frac{dv(t)}{dt} = -v(t) + R I_{\text{syn}}(t)$$

막전위가 임계값 v_θ 를 넘으면 스파이크를 내고 v 는 reset 된다(soft/hard reset 등 세부 설정은 하이퍼파라미터로 제어).

구현상의 주요 가정은 다음과 같다.

- 뉴런마다 단 하나의 스파이크 배열만 유지한다. 예: $s_j(t) \in 0,1, t = 1, \dots, T$.
- 시냅스 단위 추적 변수는 필요 시 뉴런 스파이크 배열을 참조해 계산하며, 시냅스별 스파이크 배열은 따로 두지 않는다.

각 MNIST 이미지는 독립된 에피소드이므로, **새로운 이미지를 주입하기 전에** 모든 LIF 뉴런과 시냅스의 동역학 상태(막전위 v, 누적 전류, refractory 상태 등)는 초기값으로 리셋한다. 이렇게 해서 서로 다른 이미지 에피소드의 스파이크 활동과 막전위가 섞이지 않도록 하고, RL 관점에서 에피소드 간 독립성을 유지한다.

2.3 로컬 스파이크 히스토리와 CNN 입력

시냅스 i 가 뉴런 $j \to k$ 를 연결한다고 하자.

- pre 뉴런 j 의 스파이크 배열: $s_j(t)$

- post 뉴런 k 의 스파이크 배열: $s_k(t)$

에피소드 내 시각 t 에서 최근 L 개 타임스텝을 잘라

$$x_i^{\text{pre}}(\tau) = s_j(t-\tau), \quad \tau = 0, \ldots, L-1$$

$$x_i^{\text{post}}(\tau) = s_k(t-\tau), \quad \tau = 0, \ldots, L-1$$

를 만든다.

두 배열을 길이 L 의 2채널 1D 시계열로 묶어

$$X_i(t) \in 0,1^{2\times L}$$

로 정의한다.

이 설계에서는 로컬 상태를 두 개의 고정 길이 스파이크 배열과 소수의 추가 feature(현재 가중치, 레이어 인덱스, 이벤트 타입 등)만으로 구성한다. 누적 발화 횟수나 마지막 스파이크 이후 경과 시간 같은 추가 요약 스칼라는 사용하지 않는다.

CNN 전단의 입력은 항상 이 2채널 스파이크 배열 $X_i(t)$ 이며, 이 문서에서 말하는 "스파이크 히스토리"는 곧 이 배열을 의미한다.

2.4 1D CNN 전단 구조

모든 가중치 정책(Actor)과 가치함수(Critic)는 **동일한 1D CNN 전단**을 앞단에 둔다.

- 입력: $X_i(t) \in 0,1^{2\times L}$

구조(구현 기준 예시):

1. Conv1d(in_channels=2, out_channels=16, kernel=5, padding=2, stride=1) + ReLU
2. Conv1d(in_channels=16, out_channels=16, kernel=5, padding=2, stride=1) + ReLU
3. 시간 축 global average pooling → 길이 16 feature 벡터

따라서 CNN 전단의 출력은 항상

$$h_i(t) \in \mathbb{R}^{16}$$

이다.

정책과 Critic 에 대해 공통으로 적용되는 가정은 다음과 같다.

- 정책마다, Critic마다 CNN 전단 파라미터를 공유하지 않는다. 구조는 동일하지만 각 Actor·Critic 이 각각 자신의 CNN+MLP 파라미터를 가진다.
- 실험(시나리오) 간에도 파라미터는 공유하지 않고 각 실험 시작 시 모든 Actor·Critic 을 새로 초기화한다.

2.5 로컬 상태 벡터와 이벤트 타입 원핫 인코딩

로컬 상태 벡터는 **CNN feature + 추가 feature(스칼라 및 저차원 벡터)** 를 concat 하는 late fusion 구조로 만든다.

시냅스 i, 시각 t, 이벤트 $e = (i, t, \text{type})$ 에 대해 항상 포함되는 추가 feature는 다음과 같다.

- 현재 시냅스 가중치 $w_i(t)$

- 이벤트 타입 원핫 벡터 $\mathbf{e}_{\text{type}}(e) \in 0,1^2$, pre/post 구분

이벤트 타입 원핫 벡터는

$$\mathbf{e}_{\text{type}}(e) = \begin{cases} (1,0)^\top & \text{pre 이벤트일 때} \\ (0,1)^\top & \text{post 이벤트일 때} \end{cases}$$

로 정의한다.

완전지도 실험에서는 여기에 레이어 인덱스를 정규화한 스칼라 $l_{\text{norm},i}$ 를 추가로 포함한다.

따라서 입력 벡터는 시나리오별로 다음과 같이 정의한다.

- 완전비지도 및 준지도 시나리오:
$$z_i^{\text{unsup}}(t) = [h_i(t); w_i(t); \mathbf{e}_{\text{type}}(e)]$$
$$z_i^{\text{semi}}(t) = [h_i(t); w_i(t); \mathbf{e}_{\text{type}}(e)]$$
- 완전지도 gradient mimicry 시나리오:
$$z_i^{\text{sup}}(t) = [h_i(t); w_i(t); l_{\text{norm},i}; \mathbf{e}_{\text{type}}(e)]$$

여기서 $[\cdot;\cdot]$ 는 벡터 concat 을 의미한다.

실제 입력 차원 d 는 CNN 출력 차원 16에 추가 feature 차원을 더한 값이다. 예를 들어

- 완전비지도·준지도: $h_i(t)$ 16차원 + $w_i(t)$ 1차원 + $\mathbf{e}_{\text{type}}(e)$ 2차원 → $d_{\text{unsup}} = d_{\text{semi}} = 19$
- 완전지도: $h_i(t)$ 16차원 + $w_i(t)$ 1차원 + $l_{\text{norm},i}$ 1차원 + $\mathbf{e}_{\text{type}}(e)$ 2차원 → $d_{\text{sup}} = 20$

pre/post 구분은 항상 $\mathbf{e}_{\text{type}}(e)$ 로 로컬 상태에 반영되며, 별도의 pre 전용·post 전용 정책은 사용하지 않는다.

2.6 가중치 정책 네트워크 (Actor, Gaussian)

각 가중치 정책 네트워크는 입력 $z_i(t)$ 를 받아 **정규화된 스칼라 가중치 변화량** $\Delta d_i(t)$ 를 출력한다.

MLP head 구조는 모든 정책에서 동일하며 다음과 같다.

1. 입력층: 차원 $d \to 32$ + ReLU
2. 은닉층: $32 \to 32$ + ReLU
3. 출력층: $32 \to 1$ 선형 → Tanh

출력 Tanh 를 통해 평균 $m_i(t) \in [-1,1]$ 을 얻고, 이를 중심으로 하는 Gaussian 정책을 정의한다.

- 시나리오·정책별로 서로 다른 표준편차 σ_{policy} 를 둔다.
    - 예: $\sigma_{\text{unsup1}}, \sigma_{\text{unsup2}}, \sigma_{\text{semi}}, \sigma_{\text{sup}}$
- 모든 σ_{policy} 는 CLI 하이퍼파라미터로 제어한다.

액션은

$$\Delta d_i(t) \sim \mathcal{N}\left(m_i(t), \sigma_{\text{policy}}^2\right)$$

으로 샘플링한다.

샘플링된 $\Delta d_i(t)$ 는 바로 가중치에 더하지 않고, 시나리오별 내부 스케일 및 학습률과 곱해 최종 가중치 변화 Δw_i 에 반영한다. 흥분/억제 시냅스에 대해서는 각각 다른 클리핑 범위를 둔다(3장, 4장 참조).

2.7 가치함수 네트워크 (Critic)

Critic 은 동일한 입력 $z_i(t)$ 를 받아 해당 로컬 상태에서의 **기대 누적 보상 추정값** $V_\phi(z_i(t))$ 를 출력한다.

MLP head 구조는 Actor 와 동일하되 마지막 층에서 Tanh 를 쓰지 않고 선형 출력만 둔다.

1. 입력층: $d \to 32$ + ReLU
2. 은닉층: $32 \to 32$ + ReLU
3. 출력층: $32 \to 1$ 선형 → $V_\phi(z_i(t))$

Actor 와 Critic 은 CNN 전단까지 포함하여 **완전히 다른 파라미터 집합**을 사용하며, 각각 별도의 optimizer 로 학습된다.

2.8 RL 궤적 구조와 에피소드 정의

이 프로젝트에서의 **에피소드**는 MNIST 이미지 1장에 대해 SNN 을 T 스텝 동안 시뮬레이션한 전체 과정을 의미한다.

한 타임스텝 t 에서의 처리 과정은 다음과 같다.

1. 뉴런 j 가 스파이크하면 $s_j(t) = 1$ 이 된다.
2. 뉴런 j 와 연결된 모든 시냅스를 순회하면서
    - j 가 pre 인 시냅스에 대해 **pre 이벤트**
    - j 가 post 인 시냅스에 대해 **post 이벤트**
      를 생성한다.
3. 각 이벤트 $e = (i, t, \text{type})$ 에 대해
    - 로컬 상태 $s_e = z_i(t)$ 를 구성한다. 이때 이벤트 타입 type 은 원핫 벡터 $\mathbf{e}_{\text{type}}(e)$ 로 인코딩되어 $z_i(t)$ 에 포함된다.
    - Actor 평균 m_e 와 샘플 액션 $a_e = \Delta d_e$ 를 계산한다.
    - Critic 출력 $V_e = V_\phi(s_e)$ 를 계산한다.
4. (s_e, a_e, V_e) 를 에피소드 버퍼에 저장한다.

에피소드가 끝난 뒤 해당 이미지에 대한 전역 보상 R 을 계산하고, 버퍼에 들어 있는 모든 이벤트에 대해

- 보상 $r_e = R$
- return G_e

를 채운다. 할인율 γ 는 2.9 절에서 명시한다.

강화학습 관점에서 각 이미지 에피소드는 **전역 보상 R 을 갖는 Monte Carlo 궤적**이며, 전체 프로젝트의 RL 업데이트는 이 MC 리턴과 value baseline 을 사용하는 **MC 기반 PPO Actor–Critic** 으로 정의된다. 구현에서는 여러 에피소드(이미지)를 모아서 이미지 단위 미니배치를 구성하고, 이 미니배치 전체에 대해 PPO 손실을 적용한다.

2.9 Actor–Critic 업데이트와 PPO

에피소드에서 이벤트 인덱스를 $e = 1, \ldots, E$ 로 두면

- 상태: s_e

- 액션: a_e
- Critic 출력: V_e
- 보상: r_e
- return:

$$G_e = \sum_{k \geq e} \gamma^{k-e} r_k$$

이 된다.

이 프로젝트에서는 **모든 시나리오에서 할인율을** $\gamma = 1$ 로 둔다. 이유는 다음과 같다.

1. 전역 보상은 에피소드마다 스칼라 하나로 정의되며 에피소드 내부에서 시간에 따라 바뀌지 않는다. 즉 $r_e = R$ 이고 R 은 에피소드 전체에 대한 요약이다.
2. "에피소드 전체가 좋았는가" 만으로 정책을 학습시키는 설계를 선택했다. 이벤트 간의 시간 순서 정보까지 이용해 세밀한 신뢰할당을 풀기에는 뒷쪽 시간선의 이벤트 또한 앞 레이어(앞 시간)의 영향을 받은 이벤트이기 때문에 시간을 할당하는것이 오히려 의미 왜곡이 될 수 있기 때문이다
3. 이 경우 $\gamma < 1$ 을 두면 같은 상수 보상을 다시 discount 하는 것과 같아서 신호의 크기만 인위적으로 줄일 뿐 의미 있는 시간 구조를 반영하지 못한다.

따라서 모든 이벤트에 대해

$$G_e = R, \quad A_e = G_e - V_e = R - V_e$$

로 두고 advantage A_e 를 정의한다. 이 advantage 를 기반으로 **항상 PPO Actor 손실**을 사용해 정책을 업데이트한다.

2.9.1 PPO Actor 손실

PPO 는 기본적으로 "advantage 로 가중된 log-prob" 를 최대화하는 REINFORCE with baseline 을 기반으로 하되, 한 번의 업데이트에서 정책이 과도하게 바뀌지 않도록 **확률비(ratio)** 를 클리핑하는 방식으로 안정성을 높인다.

에피소드(또는 이미지 미니배치)를 수집할 때 사용한 정책 파라미터를 θ_{old} 라 두고

$$r_e(\theta) = \frac{\pi_\theta(a_e \mid s_e)}{\pi_{\theta_{\text{old}}}(a_e \mid s_e)}$$

를 정의한다. 클리핑 범위 하이퍼파라미터를 $\epsilon > 0$ 이라 할 때, 본 프로젝트에서 사용하는 PPO Actor 손실은

$$L_{\text{actor}}^{\text{PPO}}(\theta) = -\frac{1}{E} \sum_{e=1}^{E} \min\Big(r_e(\theta) A_e, \, \text{clip}\big(r_e(\theta), 1-\epsilon, 1+\epsilon\big) A_e \Big)$$

이다.

- θ_{old} 는 해당 이미지 에피소드(또는 이미지 미니배치)를 수집할 때 고정된 정책 파라미터이다.
- 업데이트 동안 θ 는 gradient 에 따라 변하지만, 분모의 $\pi_{\theta_{\text{old}}}$ 는 항상 rollout 당시 정책으로 고정된다.
- $r_e(\theta)$ 가 $1 \pm \epsilon$ 범위를 벗어나면, 추가적인 정책 변화가 더 이상 목적함수 증가로 이어지지 않으므로 **한 번의 업데이트에서 정책이 과도하게 바뀌는 것을 제한**하는 효과가 있다.

이 프로젝트에서 Actor 업데이트는 항상

$$\theta \leftarrow \theta - \eta_{\text{actor}} \nabla_\theta L_{\text{actor}}^{\text{PPO}}$$

형태로 수행되며, 별도의 "클리핑이 없는 순수 REINFORCE Actor 손실"은 사용하지 않는다. 즉 모든 시나리오에서 RL 학습은 **MC 리턴 기반 PPO Actor–Critic** 으로 통일한다.

2.9.2 Critic 손실

Critic 은 MC 리턴 G_e 를 회귀하는 value function 으로 보고, 전형적인 MSE 손실을 사용한다.

$$L_{\text{critic}}(\phi) = \frac{1}{E}\sum_{e=1}^{E}\big(G_e - V_e\big)^2 = \frac{1}{E}\sum_{e=1}^{E}\big(R - V_e\big)^2$$

업데이트는

$$\phi \leftarrow \phi - \eta_{\text{critic}}\nabla_\phi L_{\text{critic}}$$

로 수행한다. Actor–Critic 구조 전체는 "MC 리턴 기반 advantage Actor–Critic + PPO 클리핑" 으로 해석할 수 있다.

2.9.3 이미지 단위 미니배치 업데이트

이미지별 전역 보상 R 이 정의되어 있으므로, 실제 업데이트에서는

- 이미지 1장당 에피소드 버퍼 1개를 만들고
- 여러 이미지를 모아 에피소드 버퍼들을 이어 붙여
- **이미지 단위 미니배치**를 구성한 뒤

이 미니배치 전체에 대해 $L_{\text{actor}}^{\text{PPO}}$ 와 L_{critic} 을 계산해 한 번 또는 여러 epoch 의 gradient 업데이트를 수행한다.

각 이벤트는 여전히 자기 에피소드(이미지)의 전역 보상 R 에서 온 advantage $A_e = R - V_e$ 를 사용하므로, 다른 이미지와 함께 미니배치에 들어간다고 해서 보상의 의미가 희석되지 않는다. 반대로, 여러 이미지를 평균내어 gradient 를 계산함으로써 Monte Carlo 추정의 분산이 줄어들고 학습 과정이 더 안정화되는 효과가 있다.

2.9.4 전체 PPO 손실 요약

이 프로젝트에서의 **RL 학습 과정은 PPO 기반 정책 경사법 한 가지로 통일**하며,
학습 동안 **최적화하는 전체 손실(미니배치 평균)은 다음과 같이 정의**한다.

$$L_{\text{PPO}}(\theta,\phi) = \mathbb{E}_e\Big[L_e^{\text{CLIP}}(\theta) + c_v\big(G_e - V_\phi(s_e)\big)^2\Big]$$

여기서

- e 는 에피소드에서 뽑은 이벤트 인덱스 (또는 이미지 단위 미니배치 안의 이벤트 인덱스)이며 $\mathbb{E}_e[\cdot]$ 는 해당 이벤트들에 대한 평균이다.
- $L_e^{\text{CLIP}}(\theta)$ 는 2.9.1 절에서 정의한 per-event PPO 클리핑 Actor 손실로 $L_e^{\text{CLIP}}(\theta) = -\min\Big(r_e(\theta)A_e,\ \text{clip}\big(r_e(\theta),1-\epsilon,1+\epsilon\big)A_e\Big)$ 와 같이 정의된다.
- G_e 는 MC 리턴이며, 이 설계에서는 모든 이벤트에 대해 $G_e = R$ 이다.
- $V_\phi(s_e)$ 는 상태 s_e 에 대한 Critic 출력이다.
- $c_v > 0$ 는 value 손실(제곱 오차)의 비중을 조절하는 하이퍼파라미터다.

따라서 **Actor와 Critic을 동시에 학습할 때 사용하는 RL 학습 손실은 위의 L_{PPO} 하나뿐**이며,
모든 파라미터 업데이트는 이 손실의 그래디언트 $\nabla_{\theta,\phi}L_{\text{PPO}}(\theta,\phi)$ 를 기준으로 수행된다.

엔트로피 정규화 항은 기본 설계에서는 사용하지 않고,
정책 표준편차 σ_{policy}, PPO 클리핑 범위 ϵ, 미니배치 크기 및 epoch 수를 직접 하이퍼파라미터로 조절한다.

3. 실험 1 (시나리오 1.1): 완전 비지도 단일 가중치 정책

3.1 목표

실험 1의 목표는 다음과 같다.

- Diehl–Cook 스타일 E/I SNN 에서
- **단 하나의 로컬 가중치 정책** π_{single} 만으로

다음과 같은 현상이 어느 정도까지 자연스럽게 형성되는지 관찰한다.

- 흥분/억제 패턴(양수/음수 가중치 구조)
- 뉴런 역할 분화(특정 뉴런이 특정 숫자에 특화되는지 여부)
- 과도하게 지배적인 뉴런 없이 여러 뉴런이 역할을 나누는지 여부

완전 비지도 학습이므로 학습 시 라벨을 사용하지 않고, 평가 시에만 Diehl & Cook(2015) 에서 사용한 **뉴런 라벨링(neuron labeling)** 으로 분류 정확도를 측정한다.

3.2 SNN 구조 (Diehl–Cook 아키텍처)

기본 구조는 Diehl & Cook(2015)의 주요 아이디어를 따른다.

- 입력층: MNIST 픽셀 784개
- 흥분층(E): N_E 개 LIF 뉴런
- 억제층(I): $N_I = N_E$ 개 LIF 뉴런

연결 구조는 다음과 같다.

1. 입력층 → 흥분층(Input→E): 가변 가중치, 흥분성 시냅스
2. 흥분층 → 억제층(E→I): 각 흥분 뉴런 e_j 와 억제 뉴런 i_j 를 1:1로 묶는다. 흥분 뉴런이 스파이크하면 대응하는 억제 뉴런이 거의 항상 스파이크하도록 고정 회로로 둔다.
3. 억제층 → 흥분층(I→E): 모든 억제 뉴런에서 모든 흥분 뉴런으로 연결, 가변 가중치, 억제성 시냅스

흥분/억제 시냅스의 부호 및 범위는

- 흥분 자리 가중치: $[\text{exc-clip-min}, \text{exc-clip-max}]$
- 억제 자리 가중치: $[\text{inh-clip-min}, \text{inh-clip-max}]$

로 클리핑하며, 두 범위는 공통 CLI 하이퍼파라미터로 제어한다.

3.3 이벤트 생성과 상태 구성

한 타임스텝 t 에서 뉴런 j 가 스파이크하면 다음을 수행한다.

1. j 와 연결된 시냅스를 순회하면서 pre/post 역할에 따라 이벤트를 생성한다.
2. 각 시냅스 i 및 이벤트 $e = (i, t, \text{type})$ 에 대해 2.3–2.5의 방식으로 상태 $z_i(t)$ 를 만든다. 이때 이벤트 타입 type 은 $\mathbf{e}_{\text{type}}(e)$ 로 인코딩된다.
3. Actor 에서 Δd_e 를 샘플링해 가중치를 즉시 업데이트한다.
4. Critic 에서 V_e 를 계산해 (s_e, a_e, V_e) 를 버퍼에 저장한다.

3.4 비지도 전역 보상: 희소성·다양성·안정성

완전 비지도 실험에서는 각 에피소드(이미지 1장)에 대해 흥분층 뉴런들의 발화율 벡터 $(r_1, \ldots, r_{N_E})$, winner 뉴런 정보, 동일 이미지를 여러 번 보여줄 때의 winner 일관성을 이용해 세 가지 항을 정의한다.

공통 표기는 다음과 같다.

- 에피소드 길이: T
- 뉴런 j 의 타임스텝별 스파이크: $s_j(t) \in 0,1$
- 총 스파이크 수와 발화율:

$$S_j = \sum_{t=1}^{T} s_j(t), \quad r_j = \frac{S_j}{T}$$

- 전체 평균 발화율:

$$\bar{r} = \frac{1}{N_E} \sum_{j=1}^{N_E} r_j$$

- 이 에피소드의 winner 뉴런:

$$j^* = \arg\max_j r_j$$

3.4.1 희소성 보상 R_{sparse}

희소성 항은 전체 평균 발화율이 목표 희소성 ρ_{target} 근처에 오도록 유도한다.

$$R_{\text{sparse}} = -\left(\bar{r} - \rho_{\text{target}}\right)^2 = -\left(\frac{1}{N_E}\sum_{j=1}^{N_E} r_j - \rho_{\text{target}}\right)^2$$

$\bar{r}$ 이 ρ_{target} 에 가깝다면 $R_{\text{sparse}} \approx 0$ 에 가깝고, 너무 많이 쏘거나 너무 안 쏘면 큰 음수 페널티를 받는다. 개별 뉴런에 대한 r_j^2 페널티는 두지 않고, 전역 평균 발화율의 음수 제곱만을 사용한다.

3.4.2 다양성 보상 R_{div}: 에피소드 누적 winner 히스토그램

다양성 항은 여러 에피소드에 걸쳐 winner 가 골고루 분포되도록 하는 것을 목표로 한다. 이를 위해 **에피소드 누적 winner 히스토그램**을 사용한다.

에피소드 인덱스를 $k = 1, 2, \ldots$ 라 하고 에피소드 k 의 winner 를

$$j^*(k) = \arg\max_j r_j^{(k)}$$

라 하자. 여기서 $r_j^{(k)}$ 는 에피소드 k 에서 뉴런 j 의 발화율이다.

에피소드 K 시점까지의 winner 카운트와 정규화된 분포는

$$H_j^{(K)} = \sum_{k=1}^{K} \mathbf{1}\left[j^*(k) = j\right], \quad p_j^{(K)} = \frac{H_j^{(K)}}{\sum_{l=1}^{N_E} H_l^{(K)}}$$

이다.

이 분포가 이상적으로는 균등 분포 $1/N_E$ 에 가깝기를 원하므로

$$R_{\text{div}}^{(K)} = -\sum_{j=1}^{N_E} \left(p_j^{(K)} - \frac{1}{N_E}\right)^2$$

로 정의한다. 여러 에피소드에 걸쳐 winner 가 골고루 분포되면 $R_{\text{div}}^{(K)} \approx 0$ 에 가깝고, 몇 개 뉴런만 winner 를 독점하면 큰 음수 페널티를 받는다. 현재 에피소드 K 에서는 이 값을 그대로 다양성 항 R_{div} 로 사용한다.

3.4.3 안정성 보상 R_{stab}: 동일 이미지에서 winner 일관성

안정성 항은 같은 이미지를 반복해서 보여줄 때 winner 뉴런이 매번 뒤집히지 않도록 하는 것을 목표로 한다.

각 training 이미지 x_n 에 대해

- 이전 에피소드에서의 winner: $j_{\text{prev}}(x_n)$
- 현재 에피소드에서의 winner: $j_{\text{curr}}(x_n)$

를 비교해 보상을 정의한다.

$$R_{\text{stab}}(x_n) = \begin{cases} 0 & \text{if } j_{\text{prev}}(x_n) \text{ 가 정의되어 있지 않은 첫 방문} \\ +1 & \text{if } j_{\text{curr}}(x_n) = j_{\text{prev}}(x_n) \\ -1 & \text{if } j_{\text{curr}}(x_n) \neq j_{\text{prev}}(x_n) \end{cases}$$

구현에서는 각 이미지 인덱스 n 에 대해 `prev_winner[n]` 배열을 유지하고, 에피소드 종료 후에 winner 를 비교·갱신한다.

3.4.4 세 항을 합친 전역 보상 R

최종 전역 보상은

$$R = \alpha_{\text{sparse}} R_{\text{sparse}} + \alpha_{\text{div}} R_{\text{div}}^{(K)} + \alpha_{\text{stab}} R_{\text{stab}}(x_n)$$

으로 정의한다. 여기서 $\alpha_{\text{sparse}}, \alpha_{\text{div}}, \alpha_{\text{stab}} > 0$ 는 CLI 로 제어하는 weight이다. RL 은 R 을 최대화하려 하므로

- 전체 평균 발화율을 ρ_{target} 근처로 유지하고
- winner 가 골고루 분포되도록 만들며
- 같은 이미지에 대해 winner 가 너무 자주 바뀌지 않도록 유도한다.

이 보상 R 은 2.9 절의 방식으로 에피소드 내 모든 이벤트에 동일하게 할당되어 Actor–Critic 업데이트에 사용된다.

3.5 RL 학습 절차

시나리오 1.1의 에피소드 처리 절차는 다음과 같다.

1. 순전파 및 이벤트 수집:
    - 입력 이미지를 Poisson 인코딩해 T_{unsup1} 스텝 동안 주입한다.
    - 스파이크 발생 시마다 이벤트를 생성하고 (s_e, a_e, V_e) 를 버퍼에 저장한다.
2. 전역 보상 계산:
    - 에피소드 종료 후 흥분층 발화율 r_j 를 모은다.
    - 3.4의 수식으로 $R_{\text{sparse}}, R_{\text{div}}, R_{\text{stab}}$ 를 계산한다.

- $R = \alpha_{\text{sparse}} R_{\text{sparse}} + \alpha_{\text{div}} R_{\text{div}}^{(K)} + \alpha_{\text{stab}} R_{\text{stab}}(x_n)$ 을 얻는다.

3. Advantage 계산:
   - 해당 에피소드의 모든 이벤트에 대해 $r_e = R, G_e = R, A_e = R - V_e$ 로 채운다.
4. Actor–Critic 업데이트:
   - 단일 에피소드 버퍼만 사용할 수도 있지만, 일반적으로는 여러 이미지에 대한 에피소드 버퍼를 모아 이미지 단위 미니배치를 구성한다.
   - 이 미니배치 전체에 대해 2.9 절의 PPO Actor 손실 $L_{\text{actor}}^{\text{PPO}}$ 와 Critic 손실 L_{critic} 을 계산하고 θ, ϕ 를 한 번 또는 여러 epoch 업데이트한다.

각 이벤트는 자기 이미지의 전역 보상 R 에서 온 advantage $A_e = R - V_e$ 를 사용하므로, 여러 이미지를 섞어 미니배치 업데이트를 수행해도 이미지별 보상의 의미가 희석되지 않는다. 반대로 미니배치 평균을 사용함으로써 Monte Carlo gradient 의 분산을 줄이고 학습 안정성을 높이는 효과가 있다.

3.6 뉴런 라벨링 기반 평가 및 로그·산출물

완전 비지도 실험에서는 학습 단계에서 라벨을 사용하지 않지만 **Diehl–Cook 스타일 뉴런 라벨링**을 통해 분류 정확도를 정의하고 평가한다.

뉴런 라벨링 절차는 다음과 같다.

1. 학습이 충분히 진행된 뒤 training set 전체를 SNN 에 통과시키면서 각 이미지에 대해 정답 라벨 y 와 흥분층 뉴런 j 의 스파이크 수(또는 발화율) $r_j(x)$ 를 기록한다.
2. 각 뉴런 j 와 각 클래스 y 에 대해 해당 클래스에 속하는 모든 예제에 대한 $r_j(x)$ 를 평균 내어 $\bar{r}_{j,y}$ 를 계산한다.
3. 뉴런 j 의 라벨을
$$L_j = \arg\max_y \bar{r}_{j,y}$$
로 정의한다.
4. 어떤 입력 x 에 대해 예측할 때는
   - 흥분층 뉴런들의 반응 $r_j(x)$ 를 모으고
   - 각 클래스 c 에 대해 $L_j = c$ 인 뉴런 집합 $\mathcal{N}_c$ 를 구성한다.
   - 클래스 점수:
$$S_c(x) = \frac{1}{|\mathcal{N}_c|} \sum_{j \in \mathcal{N}_c} r_j(x)$$
   - 최종 예측 라벨:
$$\hat{y} = \arg\max_c S_c(x)$$

이 라벨링을 사용해 train / validation / test accuracy 를 계산하고, 특히 **RL 모델의 train accuracy 변화 곡선**을 기록한다.

실험 1 종료 후 최소한 다음 산출물을 저장한다.

1. Δt–Δd 산점도(STDP 형태 분석): pre–post 시간차 Δt 와 정책 액션 Δd 를 모아 산점도 형태로 저장한다.
2. 학습 진행 곡선: 에피소드 수(또는 사용한 이미지 수)에 따른 $R_{\text{sparse}}, R_{\text{div}}, R_{\text{stab}}$ 및 뉴런 라벨링 기반 train / validation / test accuracy 의 변화를 그래프로 기록한다.
3. 텍스트 로그: 실험 시작/종료 시각, 최종 평가 지표, 사용한 모든 하이퍼파라미터 값을 담은 텍스트 로그 파일을 남긴다. JSON 은 사용하지 않는다.

4. 실험 2 (시나리오 1.2): 완전 비지도 두 가중치 정책

4.1 목표

실험 2의 목표는 다음과 같다.

- 흥분성 시냅스와 억제성 시냅스를 서로 다른 정책으로 분리했을 때
- 동일한 비지도 보상 하에서
    - 두 정책이 어떤 역할을 학습하는지
    - 흥분/억제 가중치 분포와 기능 분화 양상이 어떻게 달라지는지

를 관찰한다.

학습 중에는 라벨을 사용하지 않고, 평가 시에는 실험 1과 동일한 뉴런 라벨링 기반 분류 정확도를 사용한다.

4.2 SNN 구조와 정책 배치

SNN 구조는 3장의 Diehl–Cook 아키텍처와 동일하다.

- Input→E: 흥분성 시냅스
- I→E: 억제성 시냅스
- E→I: 1:1 고정 회로

정책은 다음과 같이 구분한다.

- π_{exc}: Input→E 시냅스에 적용되는 정책
- π_{inh}: I→E 시냅스에 적용되는 정책

두 정책은 구조는 같되 CNN+MLP 파라미터는 서로 독립이다.

가중치 클리핑 범위는 실험 1과 동일하게

- 흥분 자리: $[\text{exc-clip-min}, \text{exc-clip-max}]$
- 억제 자리: $[\text{inh-clip-min}, \text{inh-clip-max}]$

를 사용한다.

로컬 상태 $z_i(t)$ 의 정의는 2.3–2.5와 동일하며, 이벤트 타입은 항상 $\mathbf{e}_{\text{type}}(e)$ 로 원핫 인코딩되어 포함된다. pre/post 여부와 상관없이, 시냅스가 Input→E 또는 I→E 어디에 속하는지에 따라 π_{exc} 또는 π_{inh} 를 선택한다.

4.3 보상과 학습 절차

전역 보상 R 의 정의와 에피소드 처리 절차는 3.4–3.5 와 동일하다. 차이는 Actor 손실에서

- 이벤트가 Input→E 시냅스에 속하면 π_{exc}

- 이벤트가 I→E 시냅스에 속하면 π_{inh}

의 log-prob 를 사용한다는 점뿐이다. Critic 구조와 업데이트는 실험 1과 동일하다.

뉴런 라벨링 방식 역시 3.6에서 정의한 것과 동일하며, 정확도 계산 시 두 실험(1.1, 1.2)을 공통 기준으로 비교할 수 있다.

4.4 로그 및 산출물

실험 2 종료 후에는 다음을 저장한다.

1. 정책별 Δt–Δd 산점도: $\pi_{\text{exc}}, \pi_{\text{inh}}$ 각각에 대해 pre–post 시간차 Δt 와 액션 Δd 의 관계를 나타내는 산점도를 따로 저장한다.
2. 정책별 가중치 분포: 학습 전/후의 흥분성·억제성 가중치 히스토그램을 정책별로 비교한다.
3. 학습 진행 곡선 및 텍스트 로그: 비지도 보상 항들의 변화, 뉴런 라벨링 기반 train / validation / test accuracy, 특히 RL 모델의 train accuracy 변화 곡선을 그래프로 기록하고, 모든 메타데이터와 하이퍼파라미터 및 최종 지표를 텍스트 로그로 저장한다. JSON 은 사용하지 않는다.

5. 실험 3 (시나리오 2): 준지도 단일 정책 분류

5.1 목표

실험 3의 목표는 다음과 같다.

- 앞선 비지도 실험에서 사용한 **동일한 로컬 정책 프레임워크**를 유지하면서
- SNN 자체가 **직접 분류기를 겸하는 구조**에서
- 분류 정확도와 마진을 전역 보상으로 사용했을 때

어느 정도 수준의 성능과 표현력을 얻을 수 있는지 분석한다.

5.2 SNN 구조

준지도 실험에서는 구조를 단순화한다.

- 입력층: 784
- 히든 LIF 층: N_{hidden} (CLI 하이퍼파라미터)
- 출력층: 10 LIF 뉴런

학습 대상 가중치는 Input→Hidden, Hidden→Output 전체이다. 모든 학습 시냅스는 단일 정책 π_{semi} 를 공유한다.

출력층 뉴런 인덱스와 라벨은 1:1로 대응된다. 예를 들어 출력층 뉴런 k 는 숫자 k 를 의미한다.

5.3 순전파와 출력 해석

에피소드 하나는 입력–라벨 쌍 (x, y) 에 대응한다.

1. 입력 x 를 Poisson 인코딩해 T_{semi} 스텝 동안 SNN 에 주입한다.
2. 시뮬레이션 동안 이벤트마다 정책을 호출해 가중치를 업데이트하고 Critic 값을 저장한다.
3. 에피소드 종료 후 출력층 뉴런 k 에 대해 스파이크 수 s_k 와 발화율 $r_k = s_k / T_{\text{semi}}$ 를 계산한다.
4. 예측 라벨은

$$\hat{y} = \arg\max_k r_k$$

로 정의한다.

5.4 보상 설계와 마진 개념

분류 문제를 위해 전역 보상 R 을 다음 두 항의 합으로 설계한다.

1. 기본 분류 보상 R_{cls}:
   - $\hat{y} = y$ 이면 $R_{\text{cls}} = 1$
   - $\hat{y} \neq y$ 이면 $R_{\text{cls}} = -1$
2. 마진 기반 보상 R_{margin}:
   - 정답 뉴런 발화율 r_y 와 오답 중 최대 발화율 $r_{\max,\text{wrong}}$ 의 차이를
     $$M = r_y - r_{\max,\text{wrong}}$$
     로 정의한다.
   - 마진 M 이 클수록 보상을 크게 주기 위해
     $$R_{\text{margin}} = \beta M$$
     으로 둔다. 여기서 β 는 스케일 하이퍼파라미터이다.

최종 전역 보상은

$$R = R_{\text{cls}} + R_{\text{margin}}$$

이다. 마진을 포함하는 이유는 단순히 맞췄는지 여부만으로는 아슬아슬하게 이기는 경우와 압도적으로 이기는 경우를 구분하지 못하기 때문이다. 마진 보상은 출력층 뉴런들이 라벨별로 더 뚜렷하게 분리되도록 유도한다.

5.5 학습 절차와 산출물

에피소드 단위 절차 자체는 3.5와 동일하며, 전역 보상 계산 부분만 $R_{\text{cls}} + R_{\text{margin}}$ 으로 바뀐다. 구현에서는 여러 (x, y) 에피소드를 모아 이미지 단위 미니배치를 구성하고, 이 미니배치 전체에 대해 2.9 절의 **PPO Actor 손실 $L_{\text{actor}}^{\text{PPO}}$ 와 Critic 손실 L_{critic}** 을 적용한다.

실험 3 종료 후에는 다음을 저장한다.

1. 분류 정확도 곡선: train / validation / test accuracy 를 에피소드 수 또는 epoch 에 따라 플롯한다. RL 모델의 train accuracy 변화 곡선을 명시적으로 저장한다.
2. 마진 분포: 마진 $M = r_y - r_{\max,\text{wrong}}$ 의 히스토그램을 학습 전·후로 비교해 출력층 뉴런의 분리 정도를 시각화한다.
3. Δt–Δd 산점도 및 가중치 분포: pre–post 시간차와 액션 Δd 의 관계를 나타내는 산점도와, 학습 전/후 가중치 히스토그램을 저장한다.
4. 텍스트 로그: 전체 실험 설정, 최종 분류 성능, 하이퍼파라미터를 텍스트 로그 파일에 기록한다. JSON 은 사용하지 않는다.

6. 실험 4 (시나리오 3): 완전지도 gradient mimicry

6.1 목표

실험 4의 핵심 질문은 다음과 같다.

> 로컬 가중치 정책이 **전역 supervised gradient 정보를 얼마나 잘 활용해서** 유의미한 업데이트를 만들어 내는가?

이를 위해

- surrogate gradient + BPTT 로 얻은 $\partial \mathcal{L}_{\text{sup}}/\partial w_i$ 를 Teacher gradient 로 보고
- 로컬 정책이 만들어 낸 실제 가중치 변화 $\Delta w_i^{\text{agent}}$ 와
- Teacher 가 제안하는 기준 업데이트 $\Delta w_i^{\text{teacher}}$

의 거리(방향+크기)를 단일 보상 지표로 사용하는 실험을 설계한다.

또한 같은 구조의 SNN 을

- Teacher gradient 를 이용해 직접 supervised 학습시킨 기준선 모델
- 로컬 RL 정책으로 학습시킨 모델

로 나누어 분류 정확도와 loss 수렴 속도를 비교한다.

6.2 SNN 구조

완전지도 실험에서는 다음과 같은 SNN 을 사용한다.

- 입력층: 784
- 여러 층의 LIF 히든층 (예: 256–128–64–32)
- 출력층: 10 LIF 뉴런

입력을 제외한 모든 층 사이의 시냅스가 학습 대상이다. 각 시냅스에는 정규화된 레이어 인덱스 $l_{\text{norm},i} = i/$레이어 개수 를 부여한다(예: 입력 직후 히든층 0.2, 그 다음 0.4, 마지막 히든층 0.8, 출력층 1.0 등).

6.3 로컬 상태와 가중치 정책

각 시냅스에서 2.3–2.5와 동일한 방식으로 스파이크 히스토리 $X_i^{\text{sup}}(t)$ 를 구성하고, 단일 정책 π_{grad} 를 사용한다.

로컬 상태는

$$z_i^{\text{sup}}(t) = [h_i(t); w_i(t); l_{\text{norm},i}; \mathbf{e}_{\text{type}}(e)]$$

로 정의하며, 이벤트 $e = (i, t, \text{type})$ 의 타입은 원핫 벡터 $\mathbf{e}_{\text{type}}(e)$ 로 인코딩된다. pre/post 구분은 이 벡터에만 반영되고, 정책은 항상 단일 π_{grad} 를 사용한다.

이벤트가 발생하면

- π_{grad} 에서 Δd_e 를 샘플링하여 가중치를 업데이트하고
- Critic 에서 V_e 를 계산해 버퍼에 저장한다.

6.4 순전파와 BPTT (Teacher gradient 추출)

에피소드 하나는 입력–라벨 쌍 (x, y) 에 대응한다.

1. 입력 x 를 Poisson 인코딩해 T_{sup} 스텝 동안 SNN 에 주입한다.
2. 시뮬레이션 전체를 기록해 출력층 스파이크 수 s_k 와 발화율 $r_k = s_k/T_{\text{sup}}$ 를 계산한다.
3. 정답 라벨 y 에 대해 softmax(αr_k) 와 cross-entropy 로 supervised 손실 $\mathcal{L}_{\text{sup}}$ 을 정의한다.

4. surrogate gradient 를 사용해 BPTT 로

$$g_i = \frac{\partial \mathcal{L}_{\text{sup}}}{\partial w_i}$$

   를 계산한다.

Teacher 는 이 gradient 값만 제공하며, Teacher 네트워크 자체는 optimizer 로 업데이트하지 않는다.

6.5 gradient 기반 보상 (방향+크기 단일 지표)

각 시냅스에 대해 Teacher 가 제안하는 기준 업데이트를

$$\Delta w_i^{\text{teacher}} = -\eta_{\text{align}} g_i$$

로 정의한다. 여기서 $g_i = \partial \mathcal{L}_{\text{sup}} / \partial w_i$, $\eta_{\text{align}} > 0$ 는 기준 업데이트 스케일(hyperparameter)이다.

에피소드 동안 로컬 정책이 실제로 만든 가중치 변화량을 $\Delta w_i^{\text{agent}}$ 라 할 때, 시냅스별 보상은

$$R_i = -\left(\Delta w_i^{\text{agent}} - \Delta w_i^{\text{teacher}}\right)^2 = -\left(\Delta w_i^{\text{agent}} + \eta_{\text{align}} g_i\right)^2$$

로 둔다. 두 업데이트가 동일하면 $R_i = 0$ (최대 보상) 이고, 방향·크기 차이가 클수록 큰 음수(강한 페널티)를 받는다.

에피소드 동안 실제로 업데이트가 일어난 시냅스 집합 $\mathcal{S}$ 에 대해 전역 보상은

$$R = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} R_i = -\frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \left(\Delta w_i^{\text{agent}} + \eta_{\text{align}} g_i\right)^2$$

으로 정의한다. 즉 **에이전트 업데이트와 Teacher 업데이트 사이의 음의 제곱 오차**가 곧 보상이다.

6.6 학습 절차와 산출물

에피소드 단위 절차는 다음과 같다.

1. SNN 시뮬레이션 및 이벤트 수집(로컬 정책 적용)
2. BPTT 로 Teacher gradient g_i 계산
3. 이벤트 히스토리로부터 각 시냅스의 $\Delta w_i^{\text{agent}}$ 재구성
4. 6.5의 수식으로 전역 보상 R 계산
5. 모든 이벤트에 대해 $r_e = R, G_e = R, A_e = R - V_e$ 를 채운 뒤 2.9 절의 PPO Actor 손실 $L_{\text{actor}}^{\text{PPO}}$ 와 Critic 손실 L_{critic} 을 사용해 Actor–Critic 업데이트를 1회 수행

실험 4 종료 후에는 다음을 기록한다.

1. gradient 정렬 지표 곡선: $-\left(\Delta w_i^{\text{agent}} + \eta_{\text{align}} g_i\right)^2$ 의 평균 및 분포를 에피소드 수에 따라 기록한다.
2. 분류 성능 비교 곡선: 동일 구조를 Teacher gradient 로 직접 supervised 학습시킨 기준선 SNN 과 로컬 RL 정책으로 학습한 SNN 에 대해 train / validation / test accuracy 와 supervised loss 를 epoch 또는 에피소드 수에 따라 비교한다. 특히 RL 모델의 train accuracy 변화 곡선을 명시적으로 저장한다.
3. Δt–Δd 산점도 및 기타 통계: 로컬 정책이 만들어낸 업데이트 패턴을 분석하기 위해 pre–post 시간차 Δt 와 액션 Δd 의 관계를 나타내는 산점도를 저장한다.
4. 텍스트 로그: 실험 시간, 최종 분류 성능, 모든 하이퍼파라미터 등을 하나의 텍스트 로그 파일로 남긴다. JSON 형식은 사용하지 않는다.

7. 기대 기여와 해석

네 가지 실험을 통해 기대하는 기여와 해석 포인트는 다음과 같다.

1. 로컬 가중치 정책의 표현력 평가
   단일 정책, 두 정책, 준지도 분류, gradient mimicry 등 다양한 설정에서 로컬 정책이 기능 분화, 흥분/억제 패턴 형성, 전역 supervised gradient 정보 활용을 얼마나 책임질 수 있는지 정량·정성적으로 평가한다.
2. 신뢰할당 문제와 gradient mimicry 의 역할
   비지도·준지도 시나리오에서는 에피소드 단위 전역 보상만 존재하여 전형적인 신뢰할당 문제가 나타난다. gradient mimicry 시나리오에서는 Teacher gradient 를 통해 좋은 업데이트 방향과 크기에 대한 정보를 직접 제공받아 신뢰할당 문제를 어느 정도 완화할 수 있다.
3. 마진과 뉴런 역할 분화
   준지도 실험에서 마진 기반 보상을 사용할 경우, 정답 뉴런과 오답 뉴런의 발화율 차이가 어떻게 변하는지, 출력층 뉴런이 라벨별 전용 뉴런으로 분화되는지 관찰한다.

8. 공통 CLI 하이퍼파라미터 정리

문서 전반에서 등장하는 주요 하이퍼파라미터들을 CLI 인수 관점에서 한 번에 정리하면 다음과 같다.

8.1 시뮬레이션 관련

- `--T-unsup1`: 시나리오 1.1 의 타임스텝 T_{unsup1}
- `--T-unsup2`: 시나리오 1.2 의 타임스텝 T_{unsup2}
- `--T-semi`: 시나리오 2 의 타임스텝 T_{semi}
- `--T-sup`: 시나리오 3 의 타임스텝 T_{sup}
- `--dt`: 시뮬레이션 시간 간격(필요 시)

8.2 SNN 구조 관련

- `--N-E`: Diehl–Cook 흥분 뉴런 수(시나리오 1.1, 1.2)
- `--N-hidden`: 준지도 실험(시나리오 2)의 히든 뉴런 수
- `--lif-tau-m` 등: LIF 파라미터들
- `--layer-index-scale`: l_{norm} 스케일 조절 계수

8.3 정책·Critic 관련

- `--sigma-unsup1`: 시나리오 1.1 의 Gaussian 정책 표준편차
- `--sigma-unsup2`: 시나리오 1.2 의 Gaussian 정책 표준편차
- `--sigma-semi`: 시나리오 2 의 Gaussian 정책 표준편차
- `--sigma-sup`: 시나리오 3 의 Gaussian 정책 표준편차
- `--lr-actor`: Actor 학습률(필요 시 시나리오별 세분화 가능)
- `--lr-critic`: Critic 학습률
- `--ppo-eps`: PPO 클리핑 범위 ϵ
- `--ppo-epochs`: 한 이미지 배치에 대해 PPO 업데이트를 반복하는 epoch 수
- `--batch-size-images`: 이미지 단위 미니배치 크기(에피소드 수)

8.4 비지도·준지도 보상 관련

- `--rho-target`: 비지도 희소성 목표 발화율 ρ_{target}
- `--alpha-sparse`: R_{sparse} 가중치

- `--alpha-div`: R_{div} 가중치
- `--alpha-stab`: R_{stab} 가중치
- `--beta-margin`: 준지도 마진 보상 스케일 β
- `--exc-clip-min/max`: 흥분 자리 가중치 클리핑 범위
- `--inh-clip-min/max`: 억제 자리 가중치 클리핑 범위

8.5 gradient mimicry 관련

- `--alpha-align`: Teacher gradient 기준 업데이트 스케일 η_{align}
- `--log-gradient-stats`: gradient–Δw 통계 저장 여부(텍스트 기반)

8.6 로깅·실험 관리

- `--seed`: 난수 시드
- `--num-epochs`: epoch 수 또는 에피소드 수
- `--log-interval`: 몇 에피소드마다 로그를 남길지
- `--run-name`: 실험 이름(결과 디렉토리 이름에 사용)

로그 파일 형식은 모두 텍스트 파일을 기본으로 하며, 실험 설계 상 JSON 형식의 로그 파일은 사용하지 않는다.

9. 한계점 및 신뢰할당 문제

마지막으로 이 설계에서 구조적으로 발생하는 한계점과 신뢰할당 문제를 정리한다.

1. 에피소드 단위 전역 보상에 따른 신뢰할당 문제
   모든 시나리오에서 이미지 1장에 대해 전역 보상 R 하나만 계산하고 에피소드 내 모든 이벤트에 동일한 보상을 할당한다. 이 때문에 어떤 이벤트가 보상을 개선 또는 악화시켰는지를 세밀하게 구분하기 어렵고, 전형적인 Monte Carlo 기반 Actor–Critic 과 같은 신뢰할당 문제가 존재한다. MC 기반 PPO 클리핑을 도입해도 전역 보상 자체의 한계는 그대로 남는다.
2. gradient mimicry 의 부분적 완화 효과
   완전지도 시나리오에서 Teacher gradient 를 도입하면 좋은 업데이트 방향과 크기에 대한 정보를 직접 제공받아 신뢰할당 문제를 어느 정도 완화할 수 있다. 다만 보상은 여전히 에피소드 단위로 평균된 형태로 들어가며, surrogate gradient 자체의 한계와 noise 를 그대로 갖고 있기 때문에 신뢰할당 문제가 완전히 사라지지는 않는다.
3. 로컬 상태 표현의 한계
   로컬 상태 $z_i(t)$ 는 두 개의 스파이크 히스토리와 소수의 추가 feature 로만 구성된다. 전역 네트워크 상태나 장기적인 문맥을 관찰할 수 없으므로, 매우 긴 시간 스케일에서만 드러나는 효과나 여러 레이어 간 상호작용을 완벽히 포착하는 데에는 구조적 한계가 있다.
4. 표현력과 생물학적 그럴듯함 간 트레이드오프
   CNN+MLP 기반 로컬 정책은 표현력이 높지만, 생물학적 plausibility 를 위해 입력을 로컬 스파이크 히스토리와 몇 개의 추가 feature 로 제한했다. 이는 더 강력한 non-local 계산을 허용하는 대신 보다 생물학적으로 그럴듯한 구조를 유지하는 쪽을 선택한 결과이며, 순수 성능과 plausibility 사이에서의 트레이드오프를 형성한다.