**📘 README 요약안 (Baseline MLP vs DynamicGate-MLP)**

**1. 개념 비교**

| **항목** | **Baseline MLP** | **DynamicGate-MLP** |
| --- | --- | --- |
| 학습 대상 | Weight , Bias | Weight , Bias **+ Gate Logit** |
| 업데이트되는 값 | 가 학습을 통해 변화 | + Gate Logit (→ Gate Prob) |
| 게이트 파라미터 | 없음 |  |
| Forward |  |  |
| Backward |  | + (STE로 학습) |
| 구조 변화 | Dense (고정) | 학습 과정에서 연결이 꺼지거나 켜짐 (희소화) |
| 해석 가능성 | 단순 Weight 크기  기반 | 어떤 연결이 살아남았는지 Gate 행렬(0/1)로 직접 확인 가능 |

**2. 공통점**

* 둘 다 **Weight와 Bias**를 학습하며 분류기를 만든다.
* 둘 다 CrossEntropy Loss를 사용해 MNIST 같은 데이터셋에서 학습 가능하다.
* Forward/Backward 학습 구조는 동일한 PyTorch MLP 기반이다.

**3. 차이점**

* **Baseline MLP**: 모든 연결이 끝까지 유지. → Weight, Bias만 바뀐다.
* **DynamicGate-MLP**:
  + Gate Logit → Sigmoid → Gate Prob () → Threshold → Gate Hard (0/1)
  + 학습하면서 일부 연결이 꺼짐(0) → **실질적으로 네트워크가 희소화됨**
  + Forward에서는 꺼진 연결은 아예 연산 제외.
  + Backward에서는 STE를 통해 Gate Logit에도 gradient 전달.

**4. 왜 필요한가?**

* **추론 효율화**: 학습은 무겁지만, 추론 단계에서는 Gate Hard로 pruning 가능 → 연산량/모델 크기 감소.
* **제어 가능성**: 하이퍼파라미터(β, τ)를 통해 희소화 강도를 조절 → 속도와 정확도 간 트레이드오프 조정.
* **해석 가능성**: 어떤 연결이 중요한지 Gate 행렬(Prob/Hard)로 직접 해석 가능.
* **Dropout/Pruning의 융합**: Dropout처럼 동적, Pruning처럼 영구적 연결 제거.

**5. 구조 비교 다이어그램**

아래 그림을 README에 삽입하면 직관적으로 차이를 보여줄 수 있습니다.

**Baseline MLP vs DynamicGate-MLP**

graph TD

A[입력 x (784)] --> B[fc1: 784→256 W1,b1]

B --> C[ReLU]

C --> D[fc2: 256→10 W2,b2]

D --> E[출력 y]

A2[입력 x (784)] --> B2[fc1: 784→256 W1,b1]

B2 --> G[게이트 행렬 G1 (학습)]

G --> C2[ReLU]

C2 --> D2[fc2: 256→10 W2,b2]

D2 --> H[게이트 행렬 G2 (학습)]

H --> E2[출력 y]

classDef base fill=#CDE,stroke=#333,stroke-width=1px;

classDef gate fill=#FDD,stroke=#933,stroke-width=1px;

class B,D base;

class B2,D2 base;

class G,H gate;

* **왼쪽**: Baseline MLP — Weight와 Bias만 학습
* **오른쪽**: DynamicGate-MLP — Weight, Bias 학습 + Gate 행렬(Prob/Hard) 학습

**6. 한 줄 요약**

Baseline MLP는 단순히 **W, b만 학습**하지만,  
DynamicGate-MLP는 **W, b + Gate를 함께 학습**하여  
“필요한 연결만 살아남는” 희소 구조를 만들어 **추론 효율성과 해석 가능성**을 확보한다.