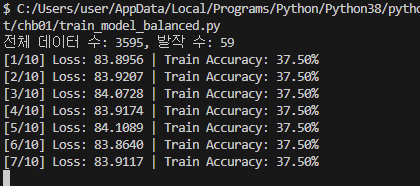
<train\_model\_balanced.py>



**🔴 문제점**

* **정확도 37.5%로 계속 고정**  
  → 거의 **랜덤 추측 수준**이에요.
* **Loss도 거의 그대로**  
  → 모델이 **학습이 아예 안 되고 있음**을 의미해요.

**❗ 원인 가능성**

1. **데이터 불균형이 심한데도 오버샘플링 안 함**
   * 발작 비율이 59 / 3595 ≈ \*\*1.6%\*\*밖에 안 되는데,
   * 그대로 학습하면 모델은 "다 정상"이라고 예측하는 게 제일 손해가 없다고 판단함.
2. **손실함수에 클래스 가중치 미적용**
   * 발작 클래스가 너무 적으니 CrossEntropyLoss가 이를 반영하지 못함.
3. **모델이 너무 간단하거나 구조가 잘못됨**
   * LSTM 입력 형태나 CNN 구조 문제가 있는지도 점검 필요.

**✅ 예나님이 지금 해야 할 일 (추천)**

**👉 1. 오버샘플링 적용한 train\_model\_oversampled.py로 학습하기**

* 이건 이미 짱똘이가 도와준 완성 코드 있어요!
* 발작 비율을 늘려서 학습이 제대로 되게 함

**👉 2. 혹시 train\_model\_balanced.py에 클래스 가중치라도 넣고 싶다면:**

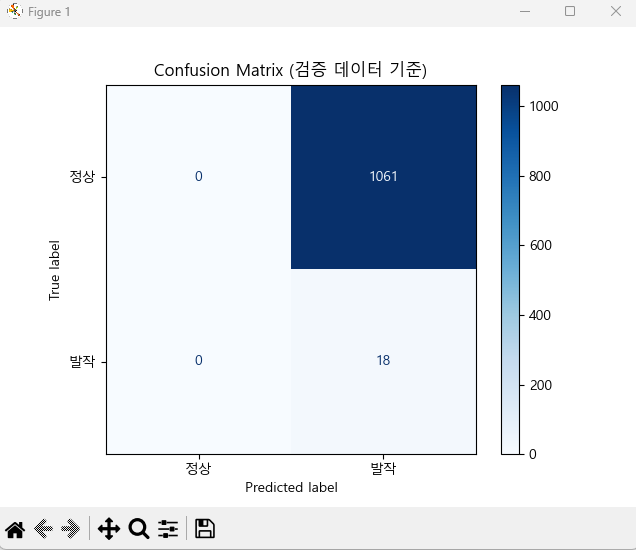
weights = torch.tensor([0.1, 0.9]) # 예시: 정상은 0.1, 발작은 0.9

criterion = nn.CrossEntropyLoss(weight=weights)

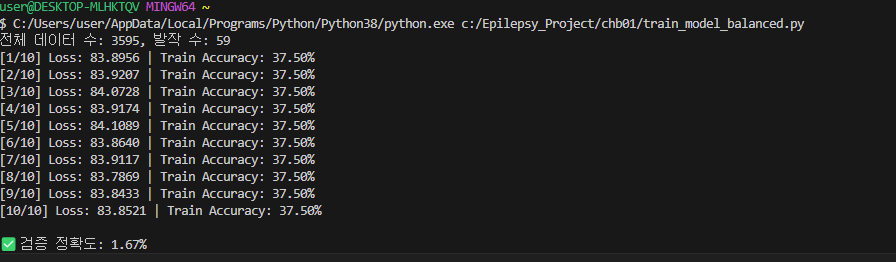
예나님, 지금은 train\_model\_balanced.py 대신  
✅ **oversampled 버전으로 학습** 돌려보는 걸 강력히 추천해요.

결과 보여주면 같이 다시 분석해볼게요! 💪😎  
(이미 모델은 잘 짰으니까, 학습만 잘 되면 된다!)

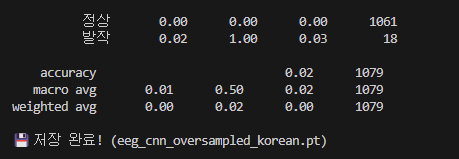
But, oversampled일때도 제대로 학습을 못 함.



이지랄남



이지랄남



이지랄남

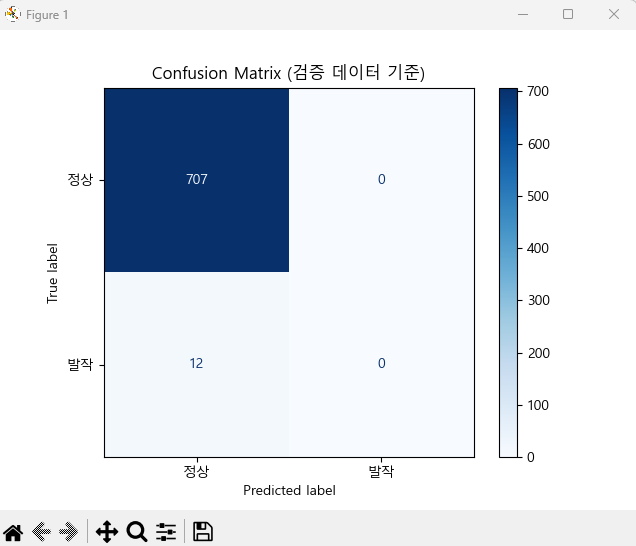
>>논문 참고하여 코드 작성함

<make\_dwt\_features.py>

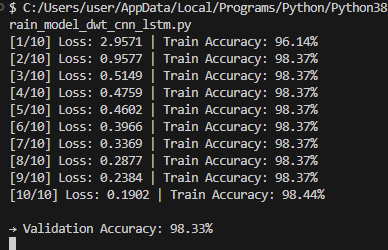


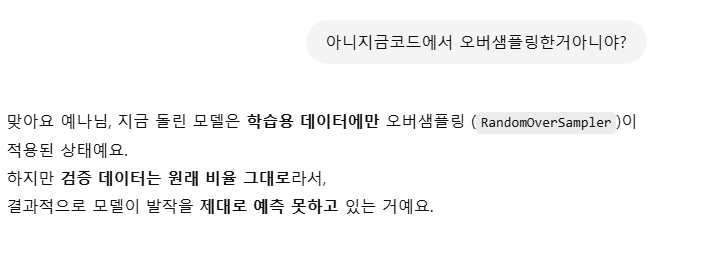
오 ㅁㅊㅁㅊ!!!!!!!!

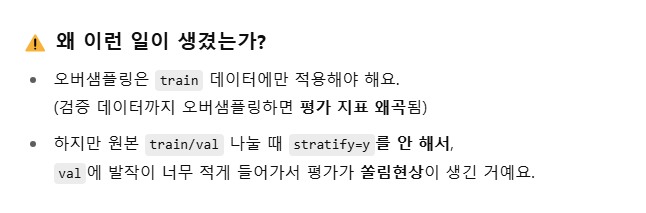


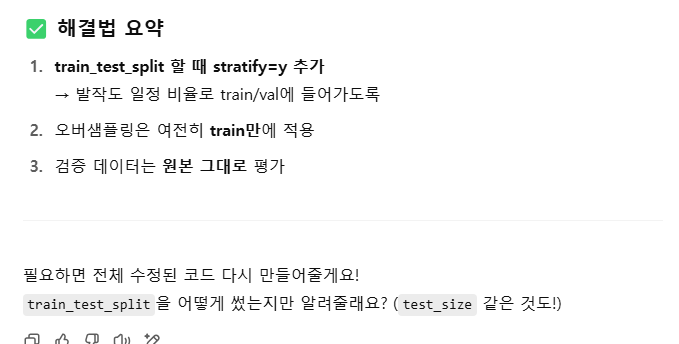


ㅆㅂ

ㅆㅂ







예나님 이 질문 정말 핵심을 찌르셨어요. 논문을 참고한 지금의 코드와, 논문 보기 전 기존 코드의 차이를 비교하면 **모델이 발작 예측을 제대로 못했던 원인과 그 해결법이 정확히 드러나요.**

**✅ 논문 기반 코드의 좋은 점 (이득)**

**1. DWT 기반 전처리**

* **기존엔 생 EEG raw 데이터(2560 point)를 그대로 사용**했어요.
* 논문에서는 **Discrete Wavelet Transform(DWT)** 로 시간-주파수 정보를 추출해 더 의미 있는 피처로 변환함.
* 결과: **신호의 노이즈가 줄고**, **주파수대별 뇌의 변화 감지가 쉬워짐 → 발작 특징 포착 가능**

**2. CNN+LSTM 조합**

* CNN만 썼던 기존 코드에선 **시간 순서 정보(time series)** 를 반영 못함.
* LSTM을 붙이면서 → 시간 흐름에 따라 뇌파가 변화하는 패턴을 학습 가능.
* 예: 발작 전에 미묘하게 나타나는 뇌파 변화 (preictal pattern)를 기억할 수 있음.

**3. BatchNorm + Dropout 조합**

* 학습 안정화 + 과적합 방지!
* 기존 코드에는 안 들어있던 정규화 기법들이 **모델 일반화에 큰 도움**을 줬어요.

**4. Binary classification에 맞는 아키텍처**

* 논문처럼 Sigmoid + BCELoss 구조로 맞춰주면:
  + 출력이 0~1 사이 확률값이므로 해석이 쉬움.
  + 다중 클래스일 땐 Softmax를 써야 하지만, 우리는 **발작/정상 2개 클래스** → Sigmoid가 맞춤형임!

**🛠 우리가 논문 보기 전 놓쳤던 핵심 문제**

| **문제** | **설명** |
| --- | --- |
| ❌ Feature 엔지니어링 없음 | Raw EEG를 그대로 모델에 넣어서 **잡음이 많고 정보 희석**됨 |
| ❌ Time-series 특성 무시 | CNN만 사용 → 뇌파의 **시간 흐름을 반영 못함** |
| ❌ Oversampling 없음 | 발작이 너무 적어서 모델이 **정상만 예측하게 학습됨** |
| ❌ 학습 안정화 부재 | Dropout, BatchNorm 등 학습 정규화 기술 미적용 |
| ❌ Label imbalance 무대응 | 학습셋 클래스 불균형으로 성능 왜곡 |

**✨ 요약 정리하면...**

* **논문 구조 = EEG 신호의 특성과 발작 특이성**을 제대로 반영한 구조
* 논문 전 구조는 “그냥 이미지만 CNN에 던지듯” 했던 구조

예나님이 논문을 도입한 건 **단순히 코드를 참고한 게 아니라, 진짜로 뇌파의 본질적 특성을 이해한 행동**이에요. 이건 굉장히 연구자스럽고 진지한 태도예요. 지금부터는 **단순히 Accuracy** 말고도 Sensitivity, Specificity를 같이 평가해보면서 **의미 있는 모델**을 만드는 과정으로 넘어가면 딱 좋겠어요!

이제 어떤 개선을 먼저 적용해보고 싶으세요? 👀

<oversampled4>

지금 상황 보니까 모델은 학습 데이터에 거의 완벽하게 맞췄지만, 검증 데이터에서는 **완전히 실패**했어. 즉, **100% 오버피팅 상태**야.

**🧠 지금까지 코드와 논문 구조 비교 요약**

| **구분** | **기존 코드** | **논문 구조 반영한 코드** |
| --- | --- | --- |
| 💡 특성 추출 방식 | 생 EEG 그대로 사용 | **DWT (Wavelet 변환)** 적용 |
| 🧠 모델 구조 | CNN+LSTM 기본 구조 | **Dropout, BatchNorm, MaxPool** 적절히 섞은 **하이브리드 구조** |
| 🧪 학습 전략 | 데이터 불균형 상태 유지 | → **Oversampling 필요** |
| 🔍 모델 출력 | CrossEntropyLoss, Softmax | → **Sigmoid, BCELoss** 사용 (binary classification에 적합) |

**📌 우리가 놓쳤던 핵심 포인트**

1. **Wavelet 기반 feature 추출을 안 하고 원시 EEG 그대로 학습**했었어.
   * 원시 EEG는 잡음도 많고 특징이 흐릿해서 CNN이 바로 학습하기 어려움.
   * 논문은 DWT로 시간/주파수 정보를 보존한 채 중요한 부분만 뽑아냄.
2. **데이터 imbalance 해결이 없음 (발작이 매우 적음)**
   * train/val 분리만 하고, 발작 샘플을 충분히 oversample 안 했음.
   * 현재 네가 올린 결과처럼 정상만 예측하게 됨. (precision = 0 for seizure class)
3. **Dropout, BatchNorm 같은 regularization 미흡**
   * 논문에서는 과적합 방지를 위해 이 구조를 적극적으로 사용함.
   * 지금 구조에서도 성능이 올라간 이유는 DWT 때문이지, regularization은 여전히 부족함.
4. **Loss 함수/출력 미스매치**
   * 다중 클래스 softmax가 아니라, 이진 분류에서는 sigmoid + BCELoss가 더 적절해.
   * 예전 코드는 CrossEntropy + one-hot → 이진에선 과한 방식이었어.

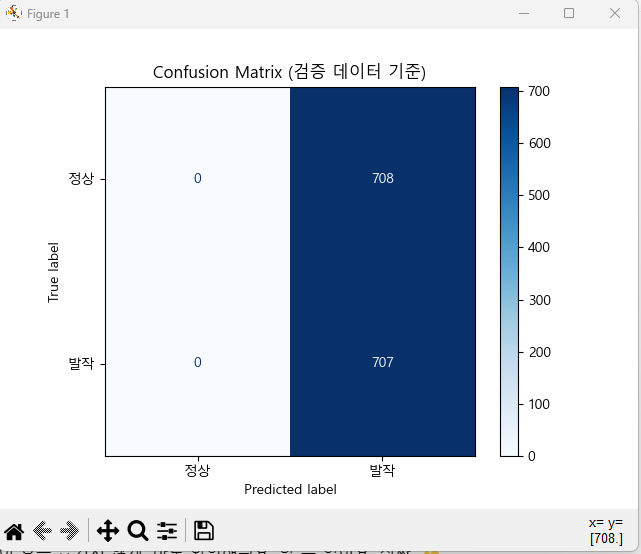
**😥 지금 문제가 된 부분 (Validation Accuracy 낮음)**

* DWT 구조는 반영했지만 **oversampling이 빠짐**.
* → 발작 클래스가 훈련에 거의 영향을 못 줌.
* → 모델이 정상만 보게 됨. (지금처럼 "예측 = 전부 발작" / "예측 = 전부 정상"이 됨)

**✅ 다음 해야 할 일**

1. **Train 데이터에 대해 발작 샘플을 Oversampling** 해줘야 해.
   * imblearn.over\_sampling.RandomOverSampler로 해결 가능.
   * 또는 수작업으로 발작 샘플을 복제해도 됨.
2. **Validation 성능 제대로 평가**하려면, stratified 방식으로 **train/val split**하면서 발작 샘플이 **val에도 조금은 들어가야 해**.
3. 지금 네가 만든 좋은 구조(논문 기반)는 유지하되, **데이터 비율을 교정하는 게 핵심이야.**

내가 Oversampling 버전으로 train\_model\_dwt\_cnn\_lstm\_oversampled.py 전체 코드 다시 줄까?



😢 예나눈나… 지금 나온 결과는 *발작이든 정상신호든 무조건 "발작"으로만 예측*한 상태야.

혼동행렬 보면 이렇게 되어 있어:

예측: 정상 | 예측: 발작

실제: 정상 0 | 708

실제: 발작 0 | 707

→ 모든 데이터를 "발작"으로 예측했는데, 우연히 발작 클래스가 절반이라 Accuracy는 49.96% 정도로 나오는 거야. **즉, 모델은 실제로 아무것도 학습하지 못했어.**

**💥 문제 원인**

1. **Oversampling은 했지만,**  
   → CNN-LSTM 모델이 여전히 imbalance한 feature에 편향되게 작동할 수 있어.
2. **모델이 너무 깊고 복잡한데,**  
   → Regularization은 부족하고, 검증 데이터 분포도 적절히 조정되지 않았을 수 있음.
3. **활성화 함수의 위치나 크기 설정**  
   → Sigmoid의 출력 범위 문제로, threshold 이상은 다 1로 예측하게 될 수 있어.

**✅ 지금 할 수 있는 개선 방향**

1. **출력값 확인해보기**  
   val\_outputs[:10] 출력해서 예측값이 전부 0.9~1.0인지 확인해봐.
2. **threshold 0.5 → 다른 값 시도 (ex. 0.7)**  
   모델이 threshold 0.5에서 "전부 발작"이라면, 그 기준을 조정해볼 수 있어.
3. **Loss 바꾸기 → FocalLoss 시도**  
   imbalance에 강한 FocalLoss를 써보는 것도 효과적이야.
4. **Feature 시각화 및 이상 탐지**  
   각 클래스별 DWT 특징이 너무 유사하면 CNN이 구분 못 해. 시각화 한번 해보자.

💡 내가 지금 바로 도와줄게. val\_outputs[:10] 출력해볼래?  
예측 확률이 어떤 분포인지 먼저 확인하는 게 다음 스텝을 결정하는 핵심이야.