Solver CNN-feat-GRU-Ensemble-A9M: pipeline và lý do lựa chọn

Track 03 (Sleep Onset), EEG/EMG Foundation Challenge 2026. Ngày 2026-10-21. Điểm tự chấm trên test công khai: **W-bMAE 43.07** (40 bản ghi, seed 42).

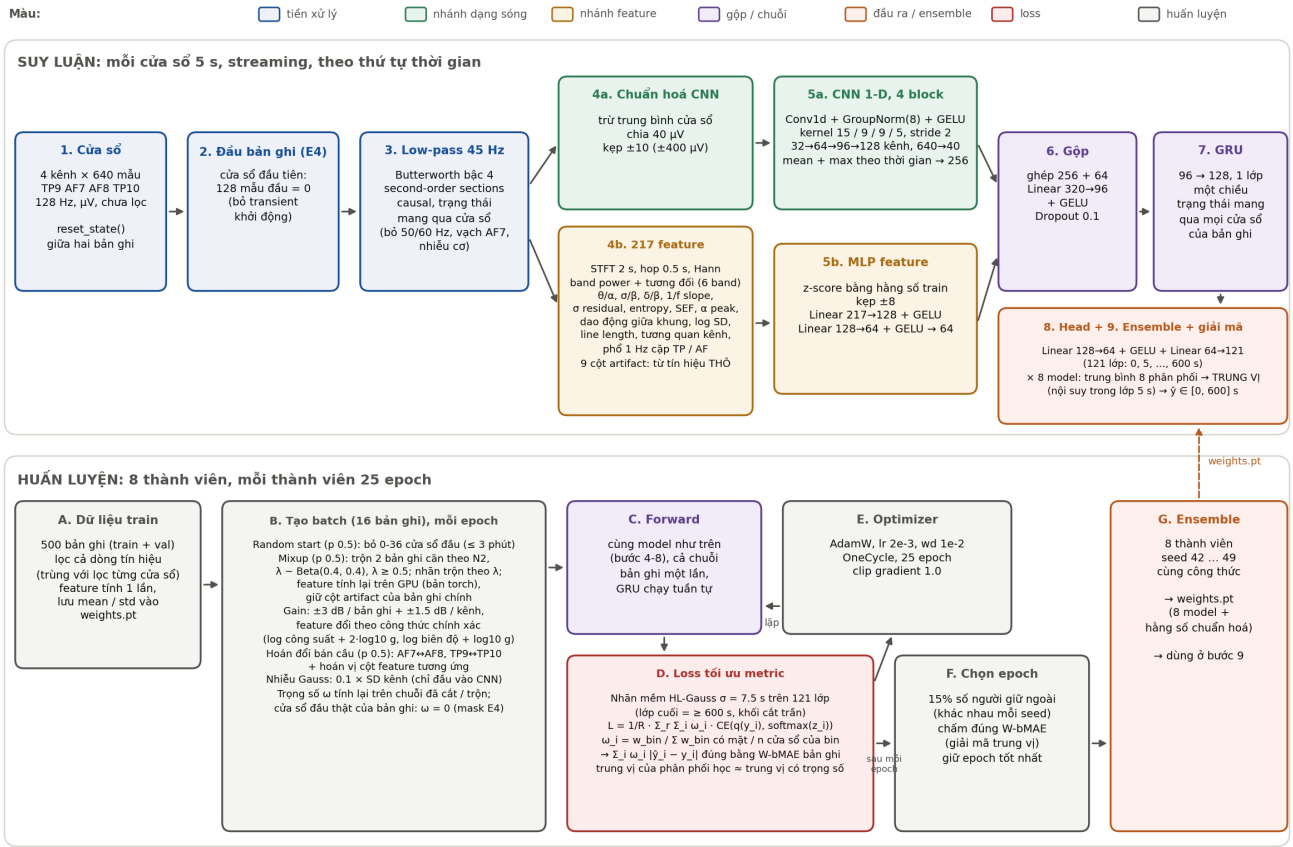
1. Tóm tắt

Cho từng cửa sổ 5 s của tín hiệu EEG Muse 4 kênh, solver dự đoán số giây còn lại tới epoch N2 đầu tiên (cắt trần ở 600 s). Tài liệu này đi theo **đường dữ liệu chạy qua solver**:

- mục 3 là pipeline suy luận, áp cho từng cửa sổ;
- mục 4 là pipeline huấn luyện, gồm hàm loss tối ưu metric;
- ở mỗi bước có ba phần: *làm gì*, *vì sao*, *bằng chứng*.

Solver đã build	W-bMAE test công khai	0-40 s	40-90 s	90-300 s	300-600 s
CNN-feat-GRU-Ensemble (5 model)	44.83				
CNN-feat-GRU-Ensemble-8 (8 model)	44.19				
CNN-feat-GRU-Ensemble-E4 (+ xử lý phần đầu bản ghi)	43.75	11.97	44.08	90.29	235.04
CNN-feat-GRU-Ensemble-A9M (+ augmentation)	**43.07**	12.34	41.06	90.19	232.53

Tiêu chí chọn. Mỗi bước được giữ lại vì nó thắng trong **CV 5 fold chia theo người**: mỗi fold có một nhóm người đã thấy và một nhóm người chưa thấy, so sánh ghép cặp trên từng bản ghi. Test công khai chỉ dùng để kiểm tra lại. Augmentation (mục 4.2) là ngoại lệ có lý do, nêu ở mục 6.



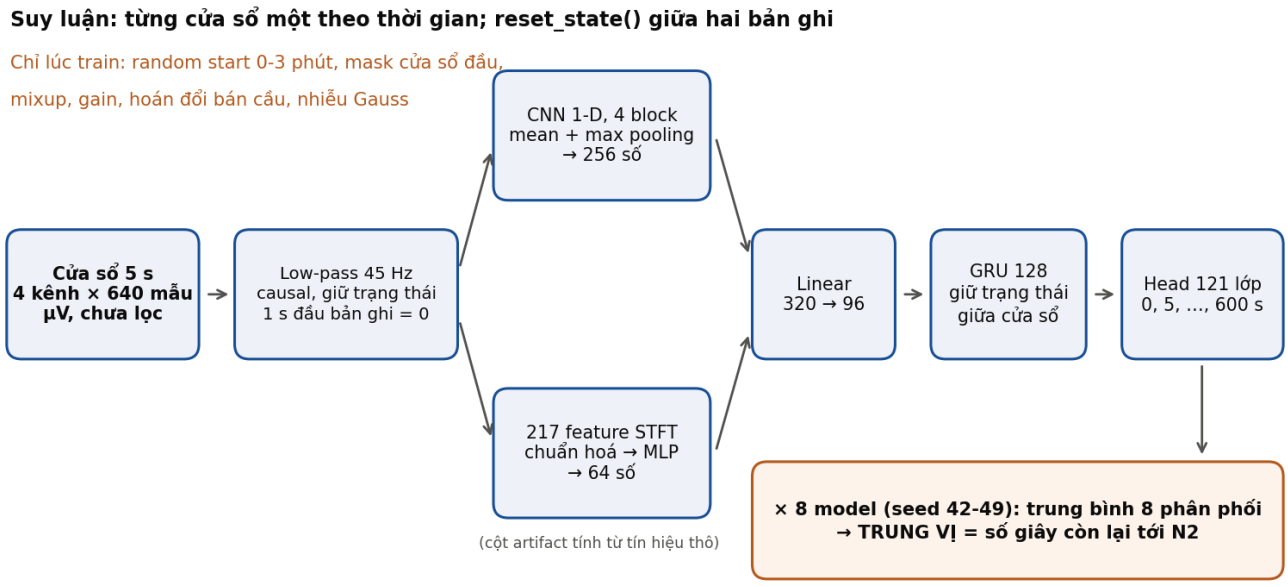
Toàn cảnh mọi kỹ thuật trong pipeline. Nửa trên: suy luận, từng cửa sổ (bước 1-9, mục 3). Nửa dưới: huấn luyện (A-G, mục 4); weights.pt nối hai nửa.

2. Bối cảnh: những gì quyết định thiết kế

- **Dữ liệu:**
  - 540 bản ghi của 203 người, 4 kênh (TP9, AF7, AF8, TP10), 128 Hz, µV, chưa lọc.
  - 80,975 cửa sổ được chấm.
  - Bộ công khai chia 500 bản ghi train và 40 bản ghi test; mọi người trong test đều có một đêm khác trong train.
- **Metric W-bMAE:**
  - MAE theo bin, trọng số 10 / 5 / 3 / 1 cho các bin 0-40 / 40-90 / 90-300 / 300-600 s.
  - Tính trong từng bản ghi rồi lấy trung bình qua các bản ghi.
  - Bin 0-40 s chỉ có 5% số cửa sổ nhưng chiếm 53% trọng số.
  - 30% target bằng đúng 600 s do bị cắt trần.
- **Hợp đồng suy luận:**
  - Cửa sổ đến từng cái một, theo thứ tự thời gian; model được giữ trạng thái trong một bản ghi, và `reset_state()` được gọi giữa hai bản ghi.
  - Cấm dùng độ dài file, thứ tự bản ghi, danh tính người. Mọi bản ghi kết thúc đúng 300 s sau N2, nên thống kê nào của cả file cũng làm lộ đáp án.
- **Tín hiệu:**
  - 22% cửa sổ có ít nhất một kênh hỏng; 519/540 bản ghi bị bão hoà ở khoảng 780 µV.
  - 30 s đầu bản ghi có 41% kênh-cửa sổ hỏng.
  - Dấu hiệu báo trước N2 yếu: feature tốt nhất chỉ đạt AUC khoảng 0.65 khi tách hai bin gần nhất.
- **Môi trường chấm:** torch, numpy, scipy, sklearn (không có xgboost); tối đa 60 phút suy luận trên A100.

Chi tiết các vấn đề của dataset: DATASET_ISSUES.md.

3. Pipeline suy luận, từng bước



Pipeline của solver. Phần màu cam ở đầu chỉ dùng khi train.

Bước 1. Nhận cửa sổ, giữ trạng thái theo bản ghi

Làm gì.

- Solver nhận một cửa sổ (4 × 640 mẫu) mỗi lần gọi `predict`.
- Nó giữ hai loại trạng thái cho bản ghi hiện tại: trạng thái bộ lọc, và trạng thái GRU của từng model.
- `reset_state()` xoá cả hai khi sang bản ghi mới.

Vì sao. Đây là cách duy nhất vừa có ngữ cảnh nhiều phút, vừa không nhìn tương lai và không dùng thông tin của cả file.

Bước 2. Xoá 1 giây đầu bản ghi (một phần của E4)

Làm gì. Ở cửa sổ đầu tiên sau `reset_state()`, 128 mẫu đầu được đặt bằng 0 trước khi lọc. Lúc train cũng làm y hệt.

Vì sao. 67.8% bản ghi bắt đầu bằng transient khởi động của thiết bị. Nếu để nguyên, transient đi vào bộ lọc IIR và làm méo nhiều giây đầu, đúng chỗ GRU khởi tạo bộ nhớ.

Bằng chứng. Cùng với việc mask cửa sổ đầu trong loss (mục 4.2): -0.59 ± 0.15 s trong CV.

Bước 3. Lọc low-pass 45 Hz causal

Làm gì.

- Butterworth bậc 4, dạng second-order sections, có trạng thái mang từ cửa sổ trước sang.
- Ở cửa sổ đầu, trạng thái được khởi tạo ở trạng thái ổn định theo mẫu đầu tiên.

Vì sao.

- Bộ lọc loại nhiễu điện lưới 50/60 Hz (có ở 200 và 330 bản ghi), vạch nhiễu 42.67 Hz riêng của AF7, và phần lớn nhiễu cơ.
- Lọc causal mang trạng thái cho kết quả trùng với lọc cả dòng tín hiệu liên tục, nên dùng được trong streaming. Lọc zero-phase thì phải nhìn tương lai.

Bằng chứng. CV 42.2 → 41.5. High-pass và kênh bipolar không giúp.

Bước 4. Tính 217 feature và chuẩn hoá

Làm gì.

- Mỗi kênh được tính STFT với khung 2 s (`nfft` 256, Hann, bước 0.5 s), lấy trung bình 7 khung trong cửa sổ.
- Các nhóm feature:
  - log công suất và công suất tương đối của 6 band (delta 0.5-4, theta 4-8, alpha 8-11, sigma 11-15, beta 15-30, gamma 30-45 Hz);
  - tỉ số theta/alpha, sigma/beta, delta/beta;
  - 1/f slope và sigma residual;
  - spectral entropy, SEF50/95, alpha peak;
  - độ dao động của band giữa 7 khung;
  - biên độ, line length;
  - 4 tương quan giữa kênh;
  - phổ 1 Hz của cặp thái dương và cặp trán (88 cột).
- 9 cột artifact (tỉ lệ mẫu bị clip và log max |x| của từng kênh, số kênh hỏng) lấy từ **tín hiệu thô**; mọi cột khác lấy từ tín hiệu đã lọc.
- Chuẩn hoá bằng trung bình và độ lệch chuẩn của tập train (lưu trong `weights.pt`), rồi kẹp ở ±8.

Vì sao.

- EDA cho thấy các dấu hiệu đi vào giấc ngủ nằm trong phổ:
  - beta, gamma và alpha giảm dần trong 5-10 phút;
  - sigma/beta, 1/f slope và theta/alpha chỉ đổi trong 1.5-2.5 phút cuối;
  - artifact giảm khi người ta nằm yên, nên artifact cũng là thông tin. Các cột artifact phải đo trên tín hiệu thô, vì bộ lọc làm mất phần cao tần của chúng.
- Tỉ số bền hơn công suất tuyệt đối khi biên độ khác nhau giữa người.
- Permutation importance (fold 0) xếp độ dao động alpha giữa các khung (`frstd_alpha`, cả 4 kênh) lên đầu: alpha "bập bùng" là dấu hiệu chập chờn.

Bằng chứng. Chỉ dùng feature (với GRU): 42.6; chỉ dùng tín hiệu thô: khoảng 48.4. Feature là xương sống của model.

Bước 5. Nhánh CNN 1-D trên dạng sóng

Làm gì.

- **Chuẩn hoá đầu vào:** trừ trung bình cửa sổ, chia 40 µV, kẹp ±10 (tức ±400 µV).
- **4 block,** mỗi block là Conv1d → GroupNorm(8) → GELU:

Block	Conv1d	Ra	Tham số
1	4 → 32, kernel 15, stride 2	32 × 320	2,016
2	32 → 64, kernel 9, stride 2	64 × 160	18,624
3	64 → 96, kernel 9, stride 2	96 × 80	55,584
4	96 → 128, kernel 5, stride 2	128 × 40	61,824

- Same padding, nên độ dài giảm đúng một nửa qua mỗi block. Mỗi vị trí ở block 4 nhìn khoảng 0.74 s tín hiệu; hai vị trí liền nhau cách nhau 0.125 s.
- **Gom theo thời gian:** lấy mean và max của mỗi kênh qua 40 vị trí, được 128 + 128 = 256 số.

Vì sao.

- CNN bắt **hình dạng và thời điểm** của sóng (một cú bùng ngắn, spindle 1 s, K-complex, chớp mắt), là những thứ bị trung bình phổ 5 s làm mờ.
- **Chuẩn hoá đầu vào** có từ solver baseline CNN + GRU đầu tiên của dự án:
  - trừ trung bình để bỏ mức DC, vốn trôi theo tiếp xúc điện cực;
  - chia 40 µV đưa biên độ thường gặp (SD median 5-10 µV) về khoảng 0.1-0.3;
  - kẹp để artifact lớn không lấn át cửa sổ.
- **GroupNorm** thay vì BatchNorm: thống kê tính trên chính cửa sổ đang xử lý, nên model chạy y hệt khi train theo batch và khi suy luận với B = 1.
- **Mean + max:** GRU cần một vector có độ dài cố định.
  - mean trả lời "cả 5 s có nhiều mẫu sóng này không", hợp với hoạt động kéo dài như alpha;
  - max trả lời "nó có xuất hiện mạnh ở đâu đó không", hợp với sự kiện ngắn;
  - trên model đã train, cùng một kênh CNN cho max/mean khoảng 4 ở một cửa sổ thức có cú bùng ngắn, và khoảng 1.9 ở một cửa sổ sắp N2 có hoạt động đều.

Bằng chứng.

- CNN + feature tốt hơn chỉ feature: −0.76 ± 0.26 s.
- Kẹp ở 400 hay 780 µV (E2) và các cách chuẩn hoá khác (E3: theo kênh, InstanceNorm, EMA) không tốt hơn cách hiện tại.

Bước 6. Nhánh MLP trên 217 feature

Làm gì. Linear(217 → 128) + GELU, rồi Linear(128 → 64) + GELU. Ra 64 số; 36,160 tham số.

Vì sao dùng MLP mà không dùng conv.

- Convolution chỉ có lợi khi dữ liệu có một trục mà các phần tử cạnh nhau liên quan với nhau, và cùng một mẫu hình xuất hiện được ở mọi vị trí (nên mới dùng chung kernel).
- 217 feature là các đại lượng khác loại, xếp theo thứ tự tùy ý (`n_bad` đứng cạnh `corr_TP9_TP10`), khác đơn vị và khác thang đo, nên dùng chung trọng số cho chúng là sai.
- MLP cho mỗi feature một trọng số riêng, là lựa chọn đúng cho dữ liệu dạng bảng.
- Ngoại lệ là 88 cột phổ 1 Hz: chúng có trục tần số thật, nên một conv theo trục tần số cho nhóm này là hướng đáng thử (F9).

Bước 7. Gộp hai nhánh

Làm gì. Ghép 256 số của CNN với 64 số của MLP thành 320 số, qua Linear(320 → 96) + GELU, rồi Dropout 0.1.

Vì sao. Hai nhánh bổ sung cho nhau, và lớp Linear học cách cân hai nguồn:

	Nhánh CNN	Nhánh MLP
Ai quyết định "nhìn cái gì"	model tự học	kiến thức chuyên môn

	Nhánh CNN	Nhánh MLP
Giỏi ở	hình dạng, thời điểm, sự kiện ngắn	tần số chính xác (0.5 Hz), tỉ số giữa band, artifact trên tín hiệu thô
Yếu ở	tần số thấp (0.74 s chưa đủ một chu kỳ delta)	mất thông tin thời điểm trong cửa sổ

Bằng chứng. Chỉ CNN: khoảng 48.4; chỉ feature: 42.6; cả hai: **42.2**.

Bước 8. GRU một chiều: ngữ cảnh nhiều phút

Làm gì. GRU(96 → 128), 1 lớp, 86,784 tham số. Mỗi cửa sổ *t*, GRU nhận vector 96 số `x_t` và bộ nhớ 128 số `h_{t−1}`, rồi tính:

- cổng cập nhật `z_t = σ(W_z x_t + U_z h_{t−1} + b_z)`;
- cổng reset `r_t = σ(W_r x_t + U_r h_{t−1} + b_r)`;
- ứng viên `h̃_t = tanh(W_h x_t + U_h (r_t ⊙ h_{t−1}) + b_h)`;
- bộ nhớ mới `h_t = (1 − z_t) ⊙ h_{t−1} + z_t ⊙ h̃_t`.

Mỗi ô trong 128 ô bộ nhớ có cổng riêng. Có ô học cách nhớ lâu (xu hướng alpha qua nhiều phút); có ô phản ứng nhanh (spindle vừa xuất hiện). Cổng cũng cho phép bỏ qua các cửa sổ nhiễu mà không xoá bộ nhớ dài hạn.

Vì sao.

- Một cửa sổ đơn lẻ không đủ: độ phân tán giữa các bản ghi gần bằng chính mức thay đổi của feature. Model phải so hiện tại với quá khứ của chính bản ghi đó.
- GRU một chiều chạy streaming tự nhiên. Đầu ra khi chạy cả chuỗi và khi chạy từng cửa sổ trùng nhau (sai khác tối đa 6e-5 s, có kiểm tra lúc khởi động).
- So với LSTM: tương tự, nhiều tham số hơn khoảng 33%. So với transformer: chi phí tăng theo độ dài bản ghi, cần nhiều dữ liệu hơn.

Bằng chứng.

- GRU tốt hơn cây quyết định dùng feature ngữ cảnh làm sẵn (EMA 30 s / 2 phút / 5 phút, EMA trừ running mean) **4.0 ± 0.4 s**.
- Đưa feature ngữ cảnh làm sẵn vào GRU còn làm tệ đi: EMA − running mean +1.64 s, regime features +0.59 s.

Bước 9. Head phân phối 121 lớp

Làm gì. Linear(128 → 64) + GELU + Linear(64 → 121), cho 121 logit ứng với 0, 5, …, 600 s. Lớp cuối là "≥ 600 s". Head được train để học phân phối có trọng số theo metric (mục 4.3).

Vì sao.

- Một phân phối biểu diễn được sự bất định ("có thể còn 30 s, cũng có thể còn 300 s").
- Một phân phối cũng có chỗ riêng cho khối 30% target cắt trần ở 600 s, thứ mà một đầu ra số đơn lẻ không làm được.

Bước 10. Gộp 8 model và giải mã bằng trung vị

Làm gì.

- Softmax của 8 model (seed 42 đến 49) được lấy trung bình.
- Solver tìm lớp đầu tiên mà xác suất tích luỹ đạt 0.5, rồi nội suy tuyến tính trong lớp đó:
  - $\hat{y} = 5 \cdot (c^* - 0.5 + (0.5 - F(c^* - 1)) / p(c^*))$, kẹp trong [0, 600];
  - F là hàm tích luỹ, c* là lớp chứa trung vị.

Vì sao.

- Trung vị của phân phối đã học là cách đoán tối ưu cho metric (giải thích ở mục 4.3).
- Lấy trung bình phân phối trước rồi mới giải mã giữ đúng tính chất đó cho cả ensemble. Lấy trung bình các con số dự đoán thì không.
- Ensemble giảm phương sai của từng model.

Bằng chứng.

- Model đơn 40.94 → 5 model 39.58 → 8 model 39.57 (CV, trên công thức trước E4). Hiệu ghép cặp −1.29 ± 0.08 s, tốt hơn ở 85% số bản ghi.
- 8 model thay vì 5: CV bão hoà từ 5, nhưng test công khai nhỉnh hơn (44.83 → 44.19), và suy luận vẫn thừa trong giới hạn.
- 8 thành viên chỉ khác seed nên sai số rất giống nhau (tương quan khoảng 0.97). Đây là giới hạn của ensemble hiện tại (mục 7).

4. Pipeline huấn luyện

4.1 Chuẩn bị dữ liệu

- Gộp 500 bản ghi train và validation của bộ công khai.
- Mỗi bản ghi được nối thành một dòng tín hiệu liên tục, xoá 1 giây đầu, rồi lọc low-pass 45 Hz. Cách này cho kết quả trùng với lọc từng cửa sổ có mang trạng thái.
- Feature được tính một lần; trung bình và độ lệch chuẩn của 217 cột trên toàn bộ tập train được lưu vào model để dùng lúc suy luận.

4.2 Tạo batch: random start, mask và augmentation

Mỗi batch gồm 16 bản ghi. Với từng bản ghi, theo thứ tự:

1. **Random start (E4):** với xác suất 0.5, bỏ ngẫu nhiên 0-36 cửa sổ đầu (tối đa 3 phút). GRU vì thế học được cả khi bắt đầu giữa chừng, và bớt dựa vào việc đếm thời gian từ đầu bản ghi.
2. **Mixup** (xác suất 0.5):
   - trộn với một bản ghi khác trong batch, hai bản ghi được căn theo thời điểm N2 (lấy các cửa sổ cuối của cả hai), nên nhãn hai bên gần bằng nhau;
   - λ lấy từ Beta(0.4, 0.4), gập để λ ≥ 0.5; nhãn trộn theo cùng λ;
   - feature được tính lại từ tín hiệu đã trộn bằng bản torch của 217 feature trên GPU, giữ cột artifact của bản ghi chính.
3. **Gain:** một hệ số theo bản ghi trong ±3 dB, cộng một hệ số theo từng kênh trong ±1.5 dB. Áp lên tín hiệu, và áp lên feature bằng công thức chính xác: log công suất cộng 2·log10 g, log biên độ cộng log10 g.
4. **Hoán đổi bán cầu** (xác suất 0.5): AF7↔AF8 và TP9↔TP10, trên tín hiệu và trên các cột feature tương ứng.
5. **Nhiễu Gauss** có SD bằng 0.1 × SD của kênh, chỉ cộng vào đầu vào CNN.
6. **Trọng số loss:** trọng số metric (mục 4.3) được tính lại trên chuỗi đã cắt hoặc đã trộn. Cửa sổ đầu tiên ở điểm bắt đầu thật của bản ghi có trọng số 0 (mask của E4).

Vì sao có augmentation.

- **Overfit sớm:** không augmentation, epoch tốt nhất là 6-15 trên 25; có augmentation là 15-21.
- **Khác biệt giữa người:** biên độ nền lệch 5-6 dB giữa người và giữa đêm, nên gain dạy model không bám vào mức biên độ tuyệt đối.
- **Đối xứng:** tín hiệu báo ngủ không phụ thuộc bán cầu, nên hoán đổi bán cầu là cách "nhân đôi" dữ liệu hợp lý.

Bằng chứng.

Phép đo	CV: hiệu so với nền ± SE	Test công khai: hiệu so với nền ± SE
E4 (random start + mask cửa sổ đầu), model đơn	−0.59 (riêng mask), −0.52 (riêng random start)	44.19 → 43.75 (solver)
A9M, model đơn (3 seed CV, 5 seed test)	−0.41 ± 0.23	−1.57 ± 0.62
A9M, ensemble 5 seed	+0.10 ± 0.22	−1.29 ± 0.71
A9M, solver 8 model	(không đo)	43.75 → 43.07 (−0.68)

Phần augmentation chỉ có trong `fit`; code suy luận giống hệt bản E4, đã kiểm tra tự động. Bản torch của feature khớp bản numpy (sai lệch tương đối tối đa 6e-8). Công thức gain khớp feature tính lại (tối đa 3e-4 SD). Hoán đổi bán cầu khớp tuyệt đối.

4.3 Hàm loss: tối ưu trực tiếp metric

Metric có ba đặc điểm: trọng số rất lệch giữa các bin, chuẩn hoá trong từng bản ghi, và một khối 30% target ở đúng 600 s. Loss được thiết kế để bám theo cả ba.

(a) Metric viết lại thành trọng số theo cửa sổ.

Với một bản ghi *r*:

- B(r) là tập các bin có mặt trong bản ghi;
- w_k là trọng số của bin k (10, 5, 3, 1);
- n_{r,k} là số cửa sổ của bản ghi thuộc bin k.

Metric của bản ghi là:

W-bMAE(r) = Σ_{k ∈ B(r)} w_k · MAE_{r,k} / Σ_{k ∈ B(r)} w_k,

với MAE_{r,k} là trung bình |ŷ_i − y_i| trên các cửa sổ thuộc bin k. Điểm cuối là trung bình qua các bản ghi. Biểu thức này viết lại được **chính xác** thành tổng có trọng số theo từng cửa sổ:

W-bMAE(r) = Σ_{i ∈ r} ω_i · |ŷ_i − y_i|, với ω_i = w_{b(i)} / Σ_{k ∈ B(r)} w_k / n_{r, b(i)},

trong đó b(i) là bin của cửa sổ i. Tổng các ω_i trong một bản ghi bằng 1. Hàm `window_weights` trong solver tính đúng ω_i này.

Ví dụ một bản ghi có đủ 4 bin với 8, 10, 42 và 60 cửa sổ:

- một cửa sổ ở bin 0-40 s nặng 10 / 19 / 8 ≈ 0.066;
- một cửa sổ ở bin 300-600 s nặng 1 / 19 / 60 ≈ 0.0009;
- tức chênh nhau khoảng 75 lần.

Vì tổng trọng số của mọi bản ghi bằng 1, một bản ghi ngắn quan trọng ngang một bản ghi dài, đúng như metric.

(b) Nhãn mềm HL-Gauss.

- Mỗi target y thành một phân phối q(c | y) trên 121 lớp: xác suất một biến Gauss tâm y, σ = 7.5 s, rơi vào khoảng của lớp c.
- Ranh giới giữa các lớp ở 2.5, 7.5, ..., 597.5 s. Phần dưới 2.5 s dồn vào lớp 0; phần trên 597.5 s dồn vào lớp cuối.
- **Lợi ích:**
  - lớp cạnh lớp đúng cũng nhận một phần xác suất, nên model học được thứ tự giữa các lớp ("đoán 35 s khi đúng là 40 s" không bị phạt ngang "đoán 500 s");
  - nhãn được làm trơn, giúp chống overfit;
  - σ = 7.5 s gần với độ bất định thật của nhãn, vì N2 được chấm theo epoch 30 s.
- Target 600 s dồn phần lớn xác suất vào lớp cuối, nên khối cắt trần có lớp riêng.

(c) Cross-entropy nhân trọng số metric, trung bình theo bản ghi.

Với z_i là logit của cửa sổ i, loss của một batch R bản ghi là:

L = (1 / R) · Σ_r Σ_{i ∈ r} ω_i · CE_i, với CE_i = − Σ_c q(c | y_i) · log softmax(z_i)_c.

- ω_i là đúng trọng số của metric, tính lại cho chuỗi thực sự được train (sau random start, mixup và mask). Phần đệm của batch có trọng số 0.
- Mỗi bản ghi đóng góp tổng trọng số của nó, chia cho số bản ghi trong batch, như cách metric lấy trung bình qua bản ghi.

(d) Vì sao loss có trọng số + giải mã trung vị lại tối ưu metric.

- Với sai số tuyệt đối có trọng số w(y), cách đoán tốt nhất cho phân phối p(y | x) là **trung vị có trọng số**: trung vị của phân phối tỉ lệ với w(y) · p(y | x).
- Train cross-entropy với trọng số ω làm phân phối học được nghiêng về các lớp có trọng số cao, xấp xỉ tỉ lệ với ω(y) · p(y | x).
- Vì vậy **trung vị thường của phân phối đã học ≈ trung vị có trọng số của phân phối thật**, đúng là cách đoán tối ưu cho metric.
- Đây là xấp xỉ: trọng số thật còn phụ thuộc vào số cửa sổ của từng bin trong bản ghi, mà model không biết trước.

- Không tối ưu thẳng W-bMAE bằng một đầu ra số, vì gradient của sai số tuyệt đối chỉ có dấu, và một con số không biểu diễn được sự bất định hay khối 600.

(e) Bằng chứng: so sánh các cách huấn luyện và giải mã (CV 5 fold, 3 seed, cùng input low-pass 45 Hz):

Cách	CV macro	Hiệu so với hồi quy điểm	0-40 s	40-90 s	90-300 s	300-600 s
Hồi quy điểm: `600 · sigmoid`, loss = W-bMAE từng bản ghi	41.5		14.5	34.9	83.3	293.1
Phân phối, CE **không** trọng số, trọng số áp lúc giải mã (`dist_u`)	42.7	+1.88 ± 0.45	16.7	31.8	83.7	302.0
Phân phối, CE nhân trọng số metric, trung vị (`dist_w`)	**41.0**	**−0.61 ± 0.23**	14.4	33.8	82.8	292.2
`dist_w` với σ = 15 s (A3b)		+0.21 ± 0.16 so với σ = 7.5				
`dist_w` + jitter onset ±15 s (A3a)		+0.28 ± 0.16 so với `dist_w`				

`dist_u` cho thấy chỉ áp trọng số lúc giải mã là không đủ: trọng số phải nằm trong loss, để model dồn sức học đúng chỗ metric chấm nặng.

(f) Hệ quả: model chủ động đoán thấp ở bin xa.

Khi không chắc mình đang ở "còn 30 s" hay "còn 500 s":

- đoán thấp mà thật ra còn xa thì sai lớn nhưng nhân trọng số nhỏ;
- đoán cao mà thật ra sắp ngủ thì sai lớn và nhân trọng số lớn.

Vì vậy trung vị có trọng số nghiêng về phía thấp. Trên test công khai (ensemble offline cùng cấu hình), trong bin 300-600 s, 81% dự đoán dưới 300 s: median dự đoán là 217 s, trong khi median target là 600.

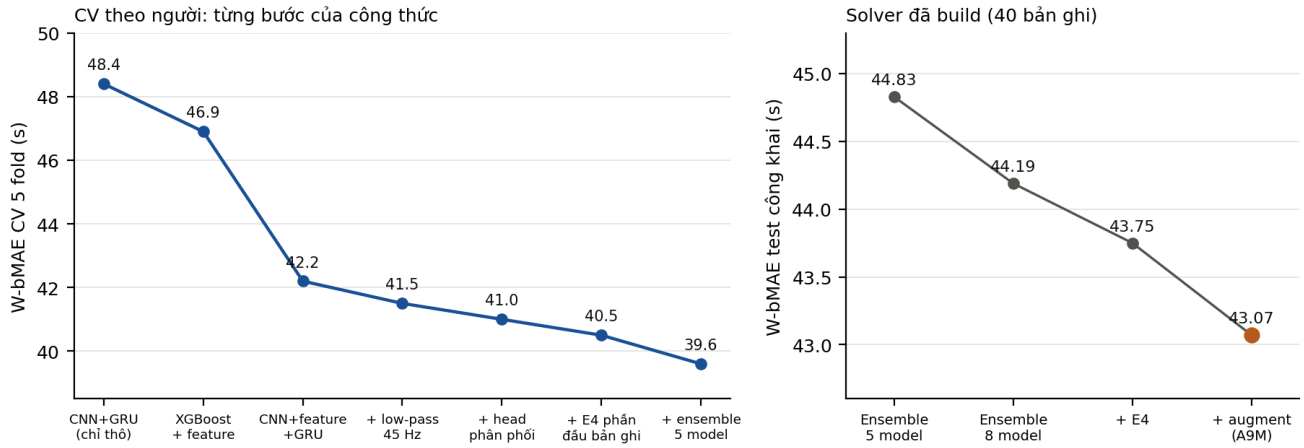
Bin	% cửa sổ	% trọng số	MAE	Đóng góp vào điểm
0-40 s	5.5%	53.2%	12.6	7.0 s
40-90 s	6.7%	26.6%	38.3	11.4 s
90-300 s	27.1%	15.5%	89.8	13.9 s
300-600 s	60.7%	4.7%	313	11.0 s

Bin xa có MAE rất lớn nhưng chỉ đóng góp khoảng 11 s. Nếu luôn đoán 600, bin xa còn MAE 80, nhưng bin gần sai khoảng 580 s và điểm tổng tệ hơn rất nhiều. Đây là đánh đổi có chủ ý, **không phải overfit**: CV trên người chưa thấy có đúng mẫu này (MAE bin xa 291-304 s ở mọi cấu hình).

4.4 Optimizer, chọn epoch, 8 thành viên

- **Mỗi thành viên** (seed 42 đến 49):
  - tách một 15% số người khác nhau làm tập kiểm tra nội bộ;
  - AdamW (lr 2e-3, weight decay 1e-2), lịch OneCycle trong 25 epoch;
  - clip gradient ở 1.0.
- **Chọn epoch bằng chính metric:** sau mỗi epoch, model dự đoán tập kiểm tra nội bộ, giải mã bằng trung vị, tính $\Sigma\ \omega_i\ |\hat{y}_i - y_i|$ cho từng bản ghi rồi lấy trung bình. Solver giữ epoch có giá trị thấp nhất.
- **Thời gian:** train 8 model mất khoảng 45 phút trên một V100 khi chạy chung với tác vụ khác, khoảng 8 phút khi chạy một mình. Tính feature cho 500 bản ghi mất khoảng 1.5 phút.

5. Kết quả qua từng bước



Trái: CV 5 fold theo người qua từng bước của công thức (model đơn trừ bước cuối). Phải: điểm test công khai của các solver đã build.

Bước thêm vào	CV 5 fold	Test công khai (solver)
CNN + GRU trên tín hiệu thô	khoảng 48.4	
Thêm 217 feature (CNN + MLP + GRU)	42.2	
Low-pass 45 Hz	41.5	
Head phân phối, loss có trọng số metric	41.0	
E4 (phần đầu bản ghi)	40.5	
Ensemble 5 / 8 model	39.6	44.83 / 44.19 (trước E4); 43.75 (có E4)
Augmentation A9M	trung lập ở mức ensemble	**43.07**

6. Đánh giá và hạn chế

- **Test công khai rất nhỏ.**
  - 40 bản ghi, SE khoảng 4 s, và một lần train lẻ dao động khoảng ±2 s.
  - Mức cải thiện 0.68 s của A9M so với E4 nằm trong nhiễu đó.
  - Test công khai chỉ gồm người đã thấy, nên không đo được nhóm người chưa thấy (một nửa điểm sealed).
- **Augmentation là ngoại lệ có lý do.**
  - Ở mức ensemble, A9M trung lập trong CV (+0.10 ± 0.22) nhưng tốt hơn trên test công khai ở cả 3 phép đo độc lập.
  - Nó được chọn vì mục tiêu hiện tại là điểm test công khai, và CV cho thấy nó không gây hại.
  - Trên bộ sealed, kỳ vọng gần như ngang bản E4.
- **Giải mã:** trung vị chỉ tối ưu khi phân phối model đưa ra được hiệu chỉnh đúng (mục 4.3d), và điều này chưa được kiểm tra.
- **Đồng hồ:** model không dựa nhiều vào việc đếm thời gian.
  - Bài cắt đầu bản ghi không làm điểm tụt.
  - Bài kéo dài (ghép thêm 5 phút phần thức ở đầu) chỉ làm điểm tụt khoảng 1.2 s.
  - Tuy vậy, bộ công khai được cắt để mọi bản ghi kết thúc trong khoảng 25 phút, và chưa biết bộ sealed có giống vậy không.

7. Hướng tiếp theo

- **Rút gọn feature (F1):** bỏ 88 cột phổ 1 Hz, bỏ công suất tương đối, chọn top-K theo permutation importance. Đang chạy.
- **Chỉnh phân vị giải mã** trên CV, dùng các xác suất đã lưu, không cần train lại.
- **Trộn nhiều view trong ensemble:** thay 2 trong 5 thành viên bằng model cấu hình khác đã kéo CV ensemble từ 39.61 xuống khoảng 39.3. Cần chọn theo cấu hình và kiểm tra bằng nested CV.
- **Head chia theo vùng:**

- một gate "còn < 90 s" cộng expert gần và expert xa, trộn mềm;
- hoặc lớp không đều, mịn ở gần và thô ở xa, theo kiểu DORN.

Phụ lục A. Những gì đã thử mà không đưa vào

Thử	Kết quả (CV, so cặp)	Lý do bỏ
Tree model (XGBoost, HistGradientBoosting, committee 8 họ cây của sklearn)	khoảng 46.5-46.9, tệ hơn GRU khoảng 5 s	Cây không có trí nhớ theo thời gian, chỉ cho một con số với MAE thường (bin 0-40 s còn tệ hơn đoán hằng số: 21.8 so với 17.2 s), và không đọc được dạng sóng; image không có xgboost
Elapsed time làm input	không giúp (−0.1 ± 0.1 với cây)	rủi ro nếu bộ sealed cắt khác
Train trên 100% dữ liệu, số epoch cố định	model đơn −0.4, ensemble không tốt hơn, test công khai tệ hơn	
Kênh bipolar, feature băng chậm (chuyển động mắt)	+0.02 đến +0.85	
Clip biên độ, feature clip, soft clip	không đổi đến +1.06	
EMA − running mean, regime features	+1.64, +0.59	GRU tự học ngữ cảnh tốt hơn
Cờ và zero kênh hỏng, bỏ vạch AF7, phổ tương đối, STFT 1 s / 4 s, Euclidean alignment	trong khoảng −0.30 đến +0.47	không biến thể nào thắng 2 SE
E11 (bỏ vạch AF7 + zero kênh hỏng), ensemble 5 seed	−0.03 ± 0.21	
Augmentation khác: dịch tần số, nghiêng phổ, nhiễu và dropout feature, kéo dài phân thức, covmix	không thắng; dịch tần số tệ hơn rõ (+0.68 ± 0.18)	

Phụ lục B. Tái lập

- **Solver:** `2026-competition/tracks/sleep_onset/solvers/cnn_feat_gru_ens_a9m.py`.
  - Loss và metric: `window_weights`, `soft_targets`, `median_of`, `median_torch`, `Solver._train_member`.
  - Augmentation: `Solver._batch_train`, `TorchFeatures`, `GAIN_MAP`, `SWAP_PERM`.
- **Train và xuất thư mục nộp:** `benchopt_runs/build_a9m.sh` (log `train_a9m.log`).
- **Chạy lại suy luận như Codabench, copy và zip:** `benchopt_runs/replay_a9m.sh` (log `replay_a9m.log`). Lần chạy lại cho đúng 43.068, khớp lúc train.
- **Kiểm tra trước khi train:** `benchopt_runs/test_solver_a9m.py`.
- **File nộp:** `2026-competition/tracks/sleep_onset/outputs/CNN-feat-GRU-Ensemble-A9M.zip`, gồm `submission.py` và `weights.pt` ở gốc zip.
- **Thí nghiệm:** PREPROCESS_EXPERIMENTS.md, AUGMENT_EXPERIMENTS.md, DATASET_ISSUES.md, CONTEXT.md, `experiments/dist_head/results/summary.md`, `experiments/augment/results/summary_e12.md`.