Miguel Breault

CAUSAL TRANSFORMER CHALLENGE

for EEG Quality Classification (EEGMMIDB)

Presented to

Usef Faghihi

UQTR

Department of Mathematics and Computer Science

January 30, 2026

Preface

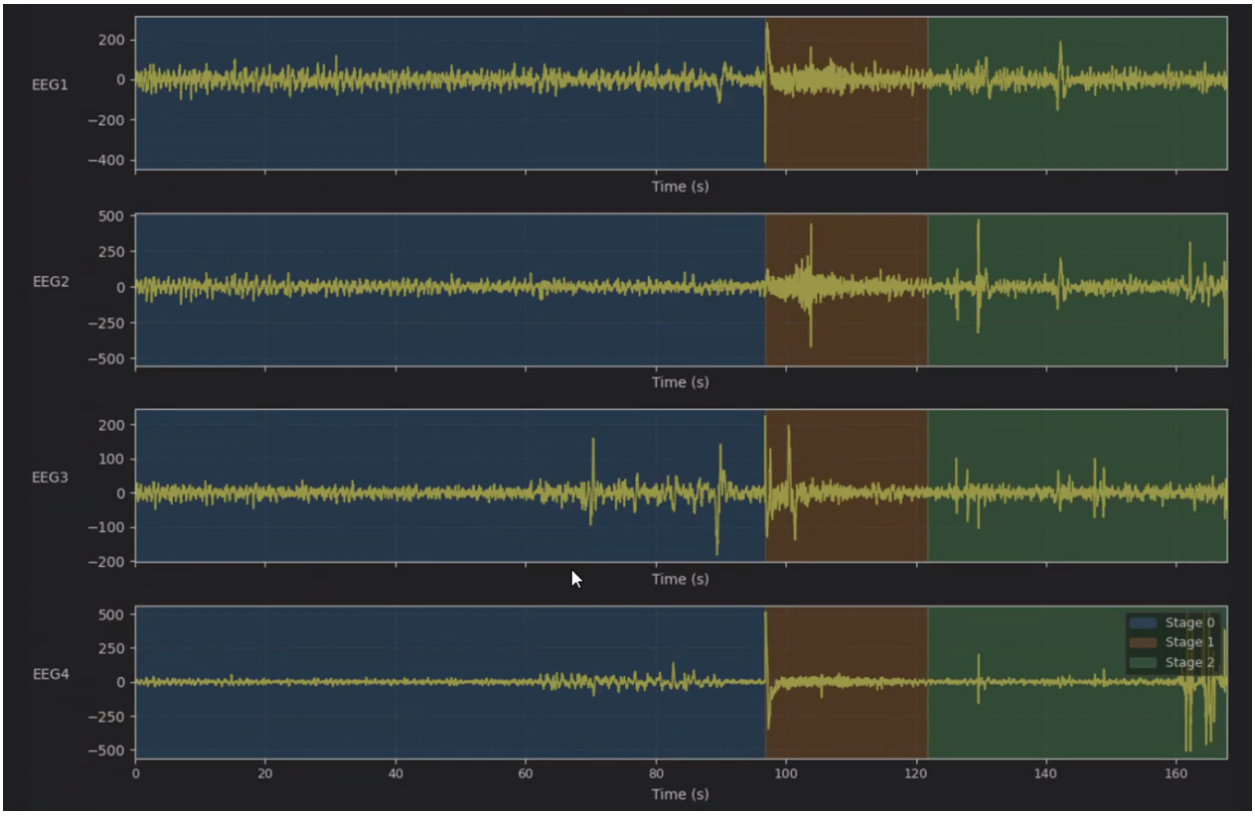


Figure 1: EEG: raw signal and quality control (illustration).

This project originated from a desire to apply the rigor of the *CausalTransformer* [4] to a concrete case of EEG analysis. As the initial repository was incomplete for this task, I developed the necessary pipeline to go from raw data to classification, taking care to properly disable "medical" causality in favor of a temporal approach.

In this report, I present the implementation on EEGMMIDB [1] and place particular emphasis on the anti-leakage mechanisms. I detail how I secured the training process (isolated normalization, subject-based separation) and validated the results through strict checks (permutations, inter-batch tests), all tracked via MLflow.

What I retain most from this experience is that a high score is worthless if it relies on a data leak: the method must be unassailable. Faced with the project's scale, I used *OpenAI Codex CLI* [6] as a global development and learning accelerator, while constantly exercising critical validation on the code and results to ensure mastery.

1 Executive Summary

Section summary — This section presents the project's objective, the approach taken to transform the *CausalTransformer* into a 3-class EEG classification pipeline, the quantitative results from MLflow exports, and the main limitations (heuristic labels, non-clinical scope).

Objective. The *CausalTransformer* repository (initially designed for causal inference and counterfactual estimation) was adapted into a **3-class EEG quality classification** pipeline {bad, medium, excellent} on EEGMMIDB/EEGBCI (PhysioNet), with a strong constraint of **zero data leakage** and traceability via MLflow.

Approach. The adaptation is based on (i) an EEG data pipeline (EDF reading, windowing, feature extraction, normalization) and (ii) the use of the Causal Transformer in *classification* mode (disabling the treatment branch: `treatment_mode=none`, `dim_treatments=0`) while retaining the causal masking mechanisms described in the reference architecture [5].

Results (MLflow exports). On the candidate configuration (`raw8` + `manualcw`, seed=600, 5 folds), the CT model achieves a test **Balanced Accuracy (subject)** of **0.906 ± 0.056** and a **Macro-F1 (record)** of **0.840 ± 0.026** (see Tables 1 and 4). The internal baselines (EEGNet, ShallowConvNet, SimpleCNN, CSP+LDA) are summarized in Table 4.

Validity Checks. Two key checks are presented: (i) *label permutation* (expected performance close to random chance) and (ii) *input/label decoupling* via inter-batch permutation (`CT_SHUFFLE_TRAIN_INPUTS_MODE=inter_batch_decouple`), intended as a "strong" anti-autopilot test. A summary is provided in Table 9.

Limitations. The {bad, medium, excellent} labels are **heuristic** (non-clinical) and derived from signal quality metrics (Section 3.3). The dataset citation information (DOI **10.13026/C28G6P**, license **ODC-By 1.0**) is included in the bibliography.

Contents

2 Context and Objectives

Section summary — This section outlines the problem (variability in EEG signal quality), describes the objective of three-class classification, and explains the choice of adapting the *Causal Transformer* (originally intended for counterfactual outcomes) to a temporal classification task.

2.1 The EEG Problem

The quality of EEG signals varies greatly depending on the subjects, sessions, and experimental conditions. Artifacts (line noise, saturation, flat segments, amplitude variations) can degrade downstream analysis (e.g., task classification, event detection). In this project, quality is formulated as a **3-class classification** task {bad, medium, excellent} in order to provide a reproducible quality control (QC) module.

Key constraint: EEGMMIDB does not provide clinical quality labels. The labels used here are therefore **heuristic** and should be interpreted as a statistical approximation, not as medical ground truth (see Section 3.3).

2.2 The Causal Transformer Approach

The *Causal Transformer* [5] is designed to model time series with causal masking (no access to the future) and structured inputs of covariates and treatments, in order to estimate outcomes (potentially counterfactual). The repository under study implements this family of models (notably via Transformer modules and their attention masks).

Adaptation to EEG classification. The adaptation consists of using the Transformer backbone to predict a quality class from sequences of EEG windows, while deactivating the "treatment" logic when the task does not require it:

- **Deactivation of treatments**: EEG configuration with `treatment_mode=none` and `dim_treatments=0` (confirmed by the configuration exported in `params.json`).
- **Classification head**: output in 3 classes (`dim_outcomes=3`) and training via a cross-entropy type loss (see `config/dataset/eegmmidb.yaml` and `src/models/time_varying_model.py`).
- **Evaluation aggregations**: metrics calculated at the window, record, and subject levels, exported as `metrics.json`, `predictions_*.csv`, confusion matrices, and classification reports.

Technical contributions: Based on inspection of the code, Hydra configurations, and exported MLflow artifacts (without git diff):

- addition of an EEGMMIDB dataset (EDF reading, windowing, feature extraction, labeling);
- removal/neutralization of the "shock target" logic and focus on classification;

- instrumentation of validity checks (shuffle labels and "strong" inter-batch shuffle inputs);
- addition of EEG baselines (EEGNet, ShallowConvNet, SimpleCNN, CSP+LDA) and structured MLflow exports.

3 MLOps Engineering and Scientific Rigor (Pipeline)

Section summary — This section describes the EEGMMIDB data pipeline (EDF reading, windowing, feature extraction), the subject-disjoint split strategy, and the anti-leakage mechanisms (train-only normalization, train-only labeling, validity checks). It also explains how the {bad, medium, excellent} classes are constructed.

3.1 System Diagrams

This subsection provides a compact overview (faithful to the implementation) of the Causal Transformer adaptation used for EEG quality classification on EEGMMIDB, as well as the zero-leakage data pipeline implemented in this repository.

Figure 2 summarizes how the multi-input CT is used here with the treatment branch disabled (no treatments, EEG covariates only) to produce a 3-class quality prediction.

Figure 3 highlights the essential anti-leakage guarantees: subject-disjoint split and fitting on the training set only, for both normalization and quantile-based labeling.

Finally, Figure 4 provides a simplified view of the main classes involved (PyTorch Lightning, CT/EDCT components, and the EEGMMIDB dataset collection).

3.2 "Zero-Leakage" Data Pipeline

EEGMMIDB / EEGBCI Dataset (PhysioNet): Protocol and Format

Source and format. The EEG data comes from EEGMMIDB/EEGBCI (PhysioNet) [1] and is stored locally as `.edf` (EDF+) files. In this repository, the expected/accepted paths for discovering EDF files are implemented in `src/data/physionet_eegmmidb/dataset.py` (layout detection + recursive scan).

Version, DOI, and license. EEGMMIDB/EEGBCI (PhysioNet) is referenced as version **1.0.0** (Sept. 9, 2009), DOI **10.13026/C28G6P**, under the **Open Data Commons Attribution License v1.0 (ODC-By)** [1].

Recommended citations. The same documentation requests citing (i) BCI2000 [7] and (ii) the standard PhysioNet reference [2].

Subjects and recordings. The local copy used in this project contains **109** subject folders (`S001` to `S109`) in `data/eegmmidb/MNE-eegbci-data/files/`. EEGMMIDB is described as containing **over 1500** 1- to 2-minute EEG recordings from **109 volunteers** [1].

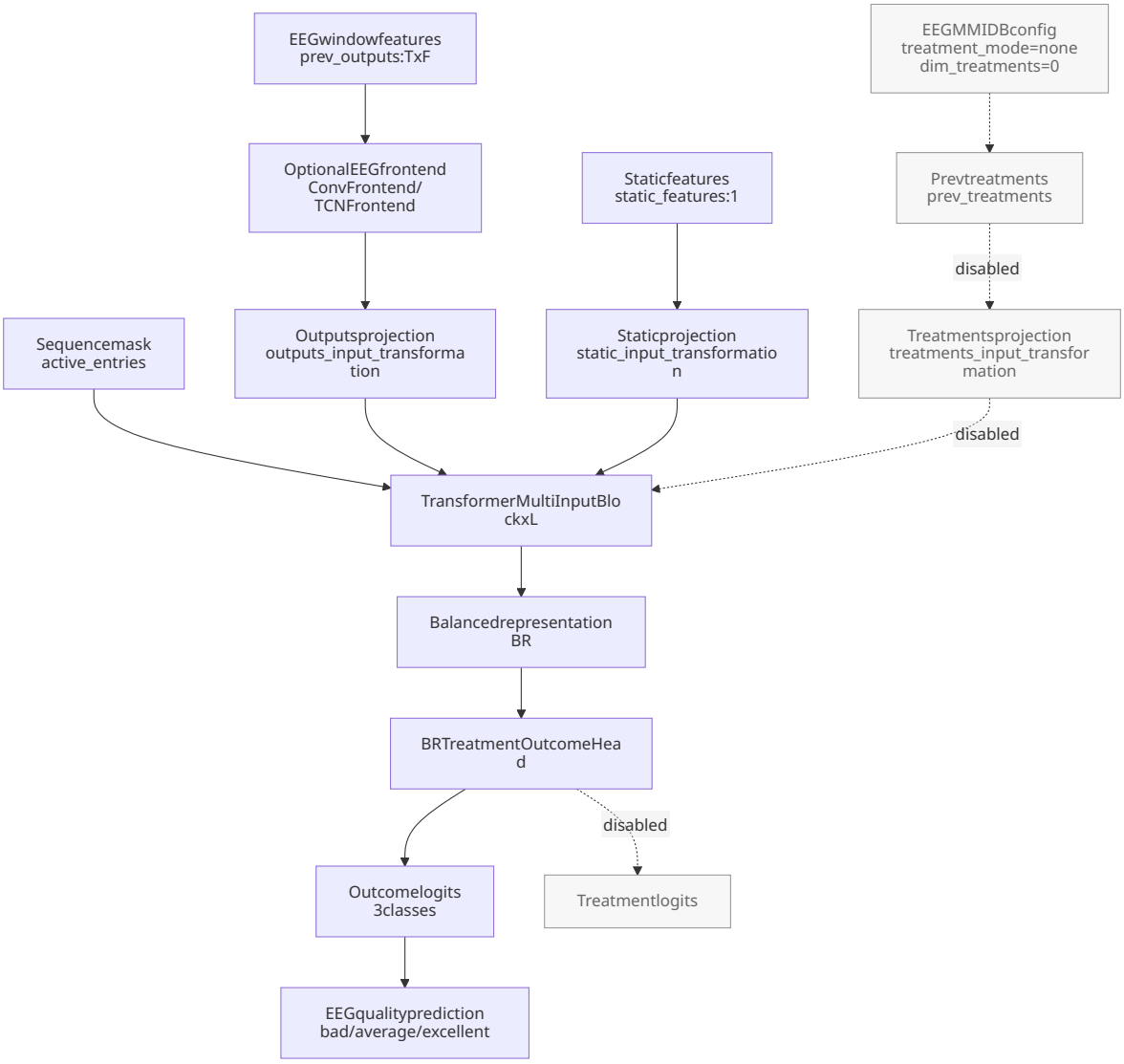


Figure 2: Causal Transformer architecture adapted for EEG quality classification. The treatment branch is disabled (treatment_mode=none, dim_treatments=0) to use only EEG covariates.

Annotations. `.edf.event` files are present alongside the EDFs (e.g., `data/eegmmidb/MNE-eegbci-data/files/S001/S001R04.edf.event`) and contain `T0/T1/T2` markers. EEGMMIDB specifies that these annotations are identical to those in the EDF+ annotation channel and describes the codes [1]:

- `T0`: rest;
- `T1`: start of movement (real or imagined) of the **left** hand (runs 3, 4, 7, 8, 11, 12) or **both fists** (runs 5, 6, 9, 10, 13, 14);
- `T2`: start of movement (real or imagined) of the **right** hand (runs 3, 4, 7, 8, 11, 12) or **both feet** (runs 5, 6, 9, 10, 13, 14).

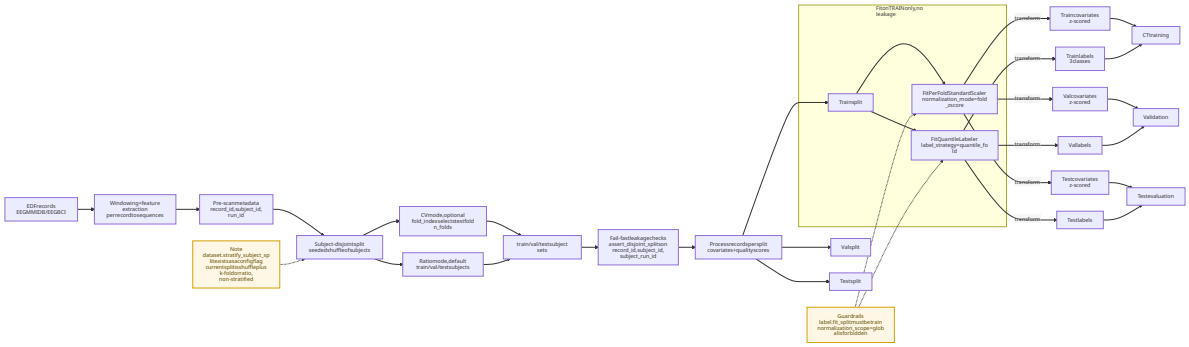


Figure 3: Zero-leakage cross-validation pipeline. Normalization statistics and quantile labeling thresholds are fitted on the training subjects only, then applied to the validation/test subjects.

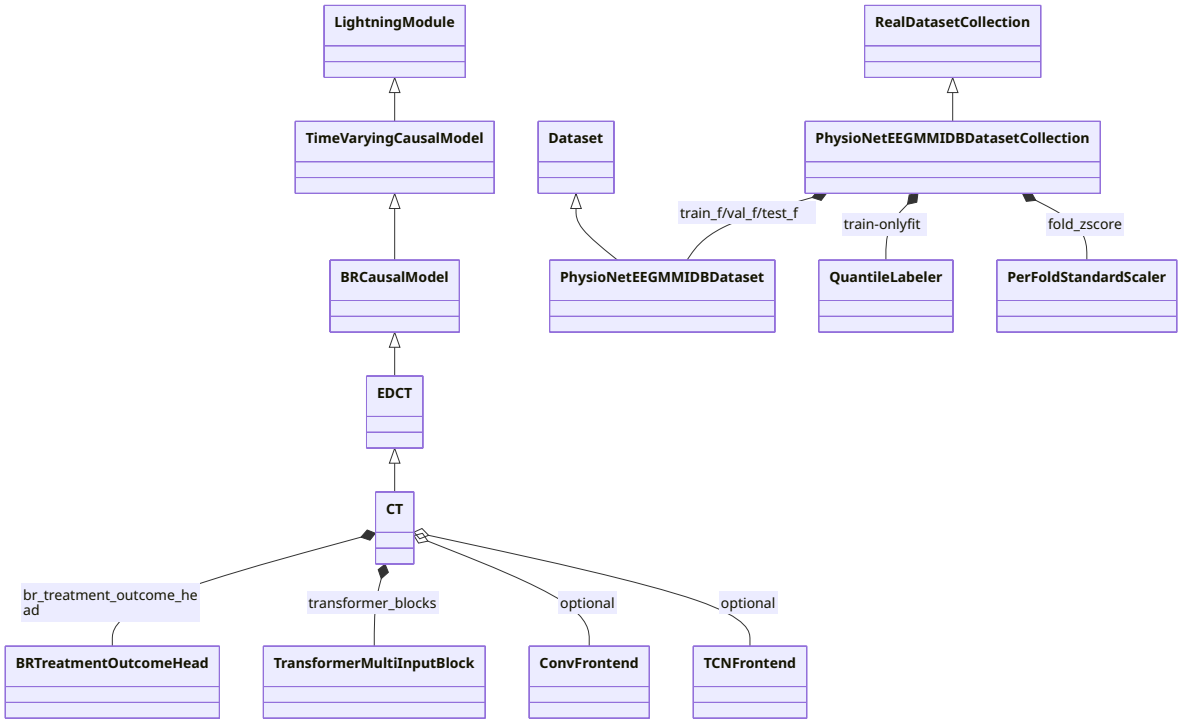


Figure 4: Simplified class structure of the implementation (PyTorch Lightning + CausalTransformer components + EEGMMIDB dataset collection).

Important: these markers describe the *motor tasks* of the EEGMMIDB protocol and are not used for defining the *quality* labels (Section 3.3).

Protocol (runs). Recordings are named `SXXXRYY.edf` (subject `SXXX`, run `RYY`). The EEGMMIDB documentation indicates that each subject performs **14 runs**: 2 baselines (1 minute) then 4 tasks repeated 3 times (2 minutes each). The sequence of runs is:

Run	Description
1	Baseline, eyes open
2	Baseline, eyes closed
3, 7, 11	Task 1: open/close left or right fist (real movement)
4, 8, 12	Task 2: imagine opening/closing left or right fist
5, 9, 13	Task 3: open/close both fists or both feet (real movement)
6, 10, 14	Task 4: imagine opening/closing both fists or both feet

Sampling frequency and channels. Reading the EDF header via MNE (`mne.io.read_raw_edf(...,preload=False)`) on a local recording (`S001R04.edf`) indicates:

- sampling frequency: **160 Hz**;
- number of EEG channels: **64**;
- example channel names: `Fc5.`, `Fc3.`, `Fc1.`, `Fcz.`, `Fc2.`, `Fc4.`, `Fc6.`, `C5..`, ...

Note. The EEGMMIDB documentation describes EDF+ files containing 64 EEG signals (160 Hz) and an annotation channel; the `.event` files contain the same annotations.

Montage (10-10 system)

The EEGMMIDB documentation indicates a 64-electrode montage according to the international 10-10 system, excluding: `Nz`, `F9`, `F10`, `FT9`, `FT10`, `A1`, `A2`, `TP9`, `TP10`, `P9`, `P10`. A reference figure is provided with the local copy of the dataset.

Windowing and sequences

Windowing. The dataset converts each EDF recording into a sequence of time windows via:

- `dataset.window_seconds = 2.0`;
- `dataset.stride_seconds = 1.0`;
- `dataset.max_seq_length = 60` (truncation if necessary).

These values are visible in the `params.json` of the final runs (e.g., fold 0, seed=600).

EEG Features ("raw8" config)

The final runs presented use `dataset.feature_set=raw8`. In the code (`src/data/physionet_eegmmidb/dataset.py`), `raw8` corresponds to 8 features calculated per window (aggregated over channels):

- `rms_mean`, `rms_std`;
- `ptp_mean`, `ptp_std`;
- `line_ratio_mean` (55-65 Hz, relative);

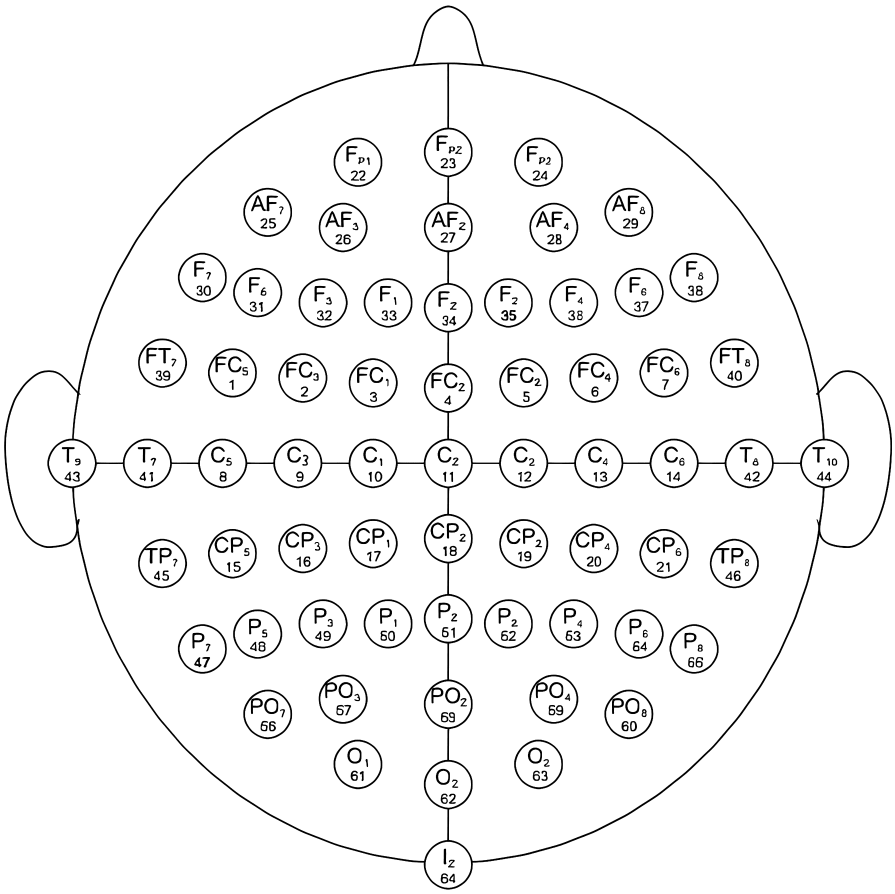


Figure 5: 64-channel montage provided in the local EEGMMIDB copy (`data/eegmmidb/MNE-eegbci-data/files/64_channel_sharbrough.png`).

- `alpha_ratio_mean`, `beta_ratio_mean`, `theta_ratio_mean`.

Subject-disjoint splits and cross-validation

Principle. The objective is to avoid "subject leakage": no window from a subject should be seen in both training and testing. The pipeline uses `dataset.split_by_subject=True` and `dataset.fold_index` to produce **subject-disjoint** splits.

CV Mechanism. In the current implementation (`src/data/physionet_eegmmidb/dataset.py::_split_subjects`), the CV:

- splits the subjects into `k = dataset.n_folds` blocks (here `k=5`);
- chooses the `fold_index` block as **test**;
- splits the remaining subjects into **train/val** according to `train_ratio` and `val_ratio`.

Important note. The `dataset.stratify_subject_split` parameter is present and exported in the `params.json` files, but it is not explicitly used by `_split_subjects` in the current state of the code; this is discussed as a limitation and area for improvement (Section 7).

Anti-leakage safeguards (assertions). After constructing the splits, the code applies disjointness checks (`assert_disjoint_splits`) on several identifiers (`record_id`, `subject_id`, `subject_run_id`) and raises a `RuntimeError` in case of overlap (`src/data/physionet_eegmmidb/dataset.py`).

"Train-only" normalization (anti-leakage)

Normalization is configured as `dataset.normalization_mode=fold_zscore` with `dataset.normalization_scope=train`. The code explicitly forbids `normalization_scope=global` (fatal error) because this would leak statistics from the val/test sets into the training set (`src/data/physionet_eegmmidb/dataset.py::_apply_normalization`).

3.3 Heuristic Labeling (Ground Truth)

Why a heuristic? EEGMMIDB is a dataset of motor tasks (execution/imagery) and does not provide "clinical quality" labels directly usable for a QC task. The {bad, medium, excellent} classes are therefore constructed automatically from objective signal metrics.

Quality score per window. A scalar score is calculated per window from metrics (RMS/PTP amplitude, frequency components, line noise, etc.). The "composite" version (in `src/data/utils/quality.py`) combines an SNR term and penalties for line noise/saturation/flatline.

QuantileLabeler (3 classes). The class thresholds come from a quantile labeler (`src/data/utils/labeling.py::QuantileLabeler`):

- low threshold = `q_low` quantile (default 0.33);
- high threshold = `q_high` quantile (default 0.66);
- mapping: `score` $\leq$ `t_low` $\rightarrow$ **bad** (0), `t_low < score < t_high` $\rightarrow$ **medium** (1), `score` $\geq$ `t_high` $\rightarrow$ **excellent** (2).

Train-only (anti-leakage). In the `label_strategy=quantile_fold` strategy, the quantiles are fitted (`fit`) **only on the training scores** of the current fold, then applied (`transform`) to val/test. The MLflow exports include labeling artifacts (e.g., `artifacts/labelers/quantile_labeler.json`).

4 Performance Results (Benchmark)

Section summary — This section presents the test metrics from MLflow exports: (i) the 5-fold cross-validation of the CT model on the candidate configuration, (ii) a quantitative comparison with several EEG baselines, and (iii) a per-class analysis at the window/record/subject levels.

4.1 Comparative Performance (Internal SOTA)

Cross-validation (CT, seed=600). Table 1 summarizes the **test** scores per fold for the candidate configuration (CT, `raw8` + `manualcw`). The metrics are calculated at the *window*, *record*, and *subject* levels. To situate the epoch selection and validation stability, Table 2 reports the same metrics on the **validation** set.

Table 1: Cross-validation (test) — metrics per fold (CT, raw8, manualcw, seed=600).

Fold	Run ID	Run name	epoch	BalAcc (window)	Macro-F1 (window)	Acc (window)	BalAcc (record)	Macro-F1 (record)	Acc (record)	BalAcc (subject)	Macro-F1 (subject)
0	1ebc9c06	dazzling-pig-862	55	0.823	0.825	0.825	0.870	0.867	0.867	0.900	0.900
1	16521b4c	bold-fawn-723	67	0.793	0.777	0.763	0.807	0.801	0.789	0.833	0.822
2	a8f5bd3a	masked-foal-764	60	0.802	0.806	0.798	0.823	0.830	0.818	0.944	0.950
3	b7f7e83a	fearless-shad-563	38	0.822	0.800	0.835	0.893	0.853	0.890	0.976	0.921
4	d5599a49	dapper-hound-665	45	0.784	0.790	0.798	0.842	0.851	0.854	0.878	0.875
Mean ± Std				**0.805 ± 0.017**	**0.799 ± 0.018**	**0.804 ± 0.028**	**0.847 ± 0.035**	**0.840 ± 0.026**	**0.843 ± 0.040**	**0.906 ± 0.056**	**0.894 ± 0.049**

Table 2: Cross-validation (val) — metrics per fold (CT, raw8, manualcw, seed=600).

Fold	Run ID	Run name	epoch	BalAcc (window)	Macro-F1 (window)	Acc (window)	BalAcc (record)	Macro-F1 (record)	Acc (record)	BalAcc (subject)	Macro-F1 (subject)
0	1ebc9c06	dazzling-pig-862	55	0.792	0.790	0.805	0.871	0.848	0.871	0.847	0.763
1	16521b4c	bold-fawn-723	67	0.784	0.791	0.804	0.769	0.785	0.833	0.833	0.871
2	a8f5bd3a	masked-foal-764	60	0.784	0.786	0.795	0.818	0.826	0.843	1.000	1.000
3	b7f7e83a	fearless-shad-563	38	0.755	0.764	0.789	0.800	0.806	0.829	0.822	0.806
4	d5599a49	dapper-hound-665	45	0.842	0.830	0.870	0.885	0.858	0.902	0.970	0.873
Mean ± Std				**0.791 ± 0.032**	**0.792 ± 0.024**	**0.813 ± 0.033**	**0.829 ± 0.049**	**0.824 ± 0.030**	**0.856 ± 0.031**	**0.894 ± 0.084**	**0.863 ± 0.090**

Baselines (detail). To isolate the reference models, Table 3 presents only the EEG baselines (mean ± standard deviation over 5 folds).

Table 3: EEG Baselines (test) — mean ± std over 5 folds.

Model	Balanced Acc (subject)	Macro-F1 (record)
EEGNet	0.706 ± 0.070	0.697 ± 0.043
ShallowConvNet	0.557 ± 0.098	0.542 ± 0.068
SimpleCNN	0.757 ± 0.089	0.735 ± 0.054
CSP+LDA	0.503 ± 0.072	0.370 ± 0.070

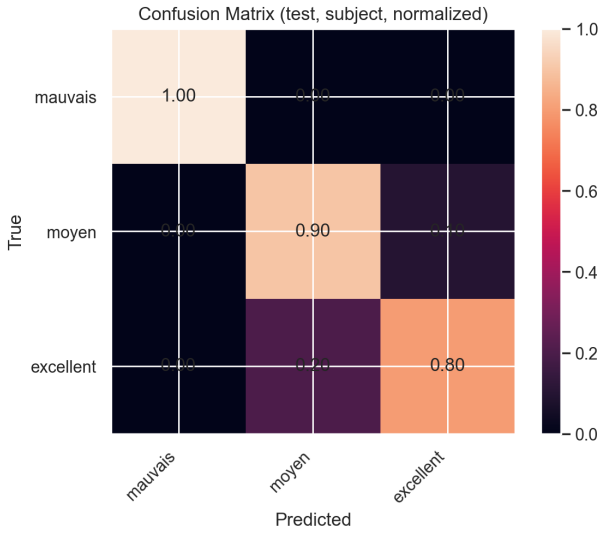
Baselines. The internal comparison of CT vs. baselines (EEGNet, ShallowConvNet, SimpleCNN, CSP+LDA) is provided in Table 4. The baseline scores come from the MLflow export of the baselines (`runs_summary.csv` file; deterministic selection of the most recent run per {model, fold}).

Table 4: Internal Benchmark — CT vs. baselines (test).

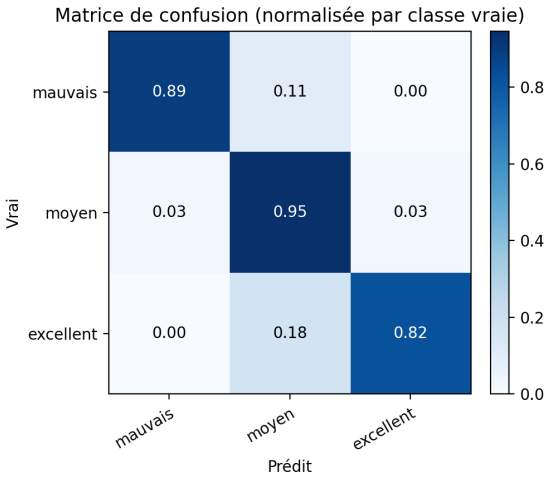
Model	BalAcc (subject)	Macro-F1 (record)	Observations
CT (Raw8, manualcw)	**0.906 ± 0.056**	**0.840 ± 0.026**	Candidate config (seed=600, 5 folds, train-only).
EEGNet	0.706 ± 0.070	0.697 ± 0.043	
ShallowConvNet	0.557 ± 0.098	0.542 ± 0.068	
SimpleCNN	0.757 ± 0.089	0.735 ± 0.054	
CSP+LDA	0.503 ± 0.072	0.370 ± 0.070	

4.2 Confusion Matrices (Example vs. Aggregated)

***Subject* level.** Figure 6 compares the normalized confusion matrix of fold 0 (seed=600) to the matrix aggregated over 5 folds (seed=600). Additional figures (record/window levels and training curves) are provided in the appendix (Section 9.4).



(a) Fold 0, seed=600



(b) Aggregated 5 folds, seed=600

Figure 6: Normalized confusion matrices at the *subject* level (test).

4.3 Stability Analysis (Multi-Seed Robustness)

The multi-seed robustness (**train-only** pipeline) is summarized in Table 5 for two complete campaigns (seeds 600 and 700, 5 folds each).

Table 5: Multi-seed robustness (test) — train-only pipeline (CT, raw8, manualcw, 5 folds).

Seed	Balanced Acc (subject)	Macro-F1 (record)
600	0.906 ± 0.056	0.840 ± 0.026
700	0.884 ± 0.066	0.830 ± 0.046

Per-Class Analysis (Aggregated over 5 folds, seed=600)

To understand difficult classes, the test predictions were concatenated over the 5 folds (window, record, subject level) to produce per-class reports.

Table 6: Class report (test, aggregated over 5 folds) — window level (seed=600).

Class	Support	Precision	Recall	F1
bad	32425	0.910	0.865	0.887
medium	29851	0.673	0.804	0.733
excellent	29255	0.860	0.736	0.793

Table 7: Class report (test, aggregated over 5 folds) — record level (seed=600).

Class	Support	Precision	Recall	F1
bad	532	0.944	0.891	0.917
medium	534	0.740	0.865	0.798
excellent	460	0.877	0.763	0.816

5 Causality and Model Validation

Section summary — This section covers the tests aimed at verifying that the pipeline learns a non-trivial input-label relationship and does not exploit obvious leaks. It discusses (i) input shuffles, (ii) validity checks (label permutation, inter-batch decoupling), and (iii) error analysis via confusion matrices.

5.1 Proof of Temporal Dependence (Input Shuffle)

The repository implements two families of input shuffles:

- **Intra-sequence shuffle**: permutation of the order of windows **within** a sequence (train-only), while maintaining the alignment between covariates and labels. This test primarily evaluates the model's sensitivity to the *order* of the windows.

Table 8: Class report (test, aggregated over 5 folds) — subject level (seed=600).

Class	Support	Precision	Recall	F1
bad	38	0.971	0.895	0.932
medium	37	0.778	0.946	0.854
excellent	34	0.966	0.824	0.889

- **Inter-batch shuffle (decoupling)**: permutation of inputs **between** batch elements without permuting the labels. This test explicitly breaks the input-label correspondence and should cause metrics to drop to random chance if the pipeline is sound.

Why intra-sequence shuffle might show little degradation at the *subject* level. In the EEGMMIDB code (`src/data/physionet_eegmmidb/dataset.py`), the `CT_SHUFFLE_TRAIN_INPUTS=1` flag permutes the time axis **in each training sequence** in *lock-step* across covariates/treatments/quality: the input-label alignment is therefore not broken. For this task, (i) the supervision is largely *window-dependent* (quality calculated window by window) and (ii) the record/subject aggregations (averages/votes) are **invariant** to the permutation of window order. Consequence: this test is informative about sensitivity to order, but it does not constitute a "strong" anti-autopilot check.

5.2 Validity Checks (Quality Control)

The following validity checks are considered the most informative under the "max safe" constraint:

- **Label permutation**: performance should drop to random chance (3 classes).
- **Inter-batch decoupling**: permutation of inputs only, with fixed labels, which should also bring scores close to random chance.

Table 9: Validity Checks — test results from MLflow exports.

Test	Status	BalAcc (test)	Macro-F1 (test)	Run ID(s)
Label permutation (folds 0–4)	PASS	BalAcc subject (mean ± std) = 0.3333 ± 0 (identical over 5 folds)	Macro-F1 subject (mean ± std) = 0.1633 ± 0.0564	`691ce7bf`, `ae0afe45`, `2bd040d2`, `c6c94f0d`, `9cea104f`
Input-shuffle inter-batch-decouple (fold 0)	PASS	Δ BalAcc (window/record/subject) = -0.3778 / -0.4527 / -0.5524	Δ Macro-F1 (window/record/subject) = -0.5386 / -0.5885 / -0.6611	baseline: `d45bb37f` shuffle: `47eda505`

Interpretation. Table 9 summarizes the exported validity check runs. For a 3-class task, a Balanced Accuracy close to 1/3 is consistent with "random chance" behavior.

Additional tests. The repository also contains a test script (`runnables/quality_tests.py`) including subject ID prediction and channel ablations. No corresponding exported runs are included in the MLflow export used for this report; these tests are therefore listed as minimal future actions (Section 7).

5.3 Error Analysis (Confusion Matrix)

Confusion matrix. A *fold* vs. aggregated comparison at the *subject* level is presented in Section 4.2 (Figure 6). Complementary matrices at the *record* and *window* levels are provided in the appendix (Section 9.4).

Prediction confidence. For illustrative purposes, Figure 7 shows the confidence histogram (max of probabilities) at the window level for fold 0 (seed=600), as exported by the post-processing.

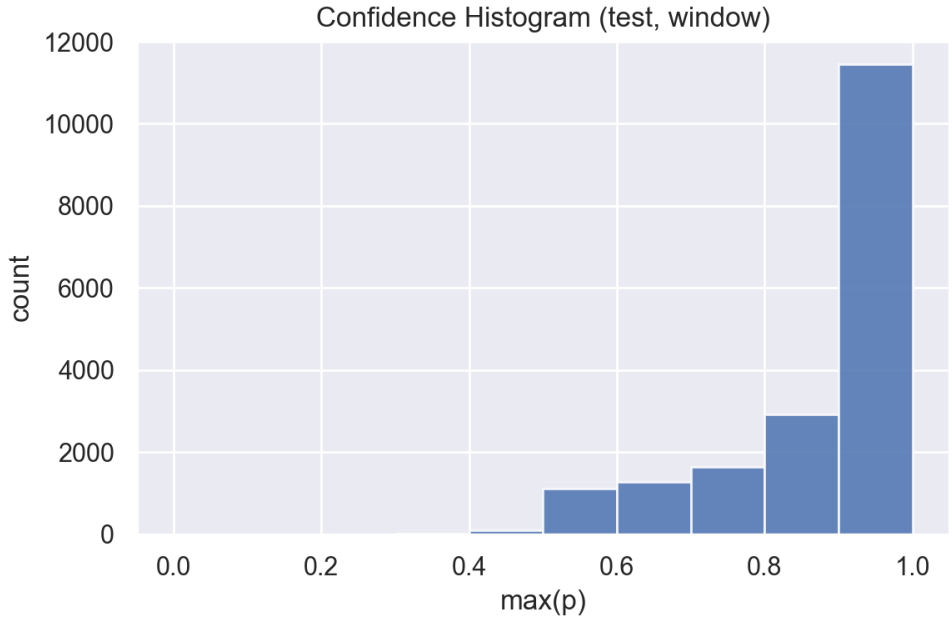


Figure 7: Confidence histogram (*window* level, test, fold 0, seed=600) from the exported artifacts.

6 Challenges Encountered and Resolutions

Section summary — This section summarizes the technical difficulties encountered and the fixes applied during the project. Each point is linked to an exported proof

(`params.json` configuration, MLflow metrics/artifacts, and/or code inspection). If an item is *reported* but not directly re-verifiable from the exports, it is explicitly indicated.

6.1 Label ↔ Feature Alignment

- **Problem (reported).** QC labels are based on amplitude statistics (RMS/PTP/*line ratio*). Early iterations based solely on normalized bandpower features could suppress some of the amplitude information, creating a mismatch between supervision and inputs.
- **Resolution.** Two complementary approaches: (i) use a `feature_set` that includes amplitude information (`raw8`, used in the final runs; Section 3.3); (ii) for bandpower presets, explicitly concatenate amplitude statistics via `dataset.bandpower_include_quality=True`. Any dimension change requires regenerating the cache (`.npz`).
- **Proofs (exports).** The final configuration uses `dataset.feature_set=raw8` and `dataset.bandpower_include_quality=False` (Appendix: Listing 1). The performance obtained on the seed=600campaign is reported in Table 1.

6.2 More Robust Record/Subject Aggregation

- **Problem (reported).** Aggregation at the record level by mean of probabilities (`mean_prob`) can be unstable in the face of outlier windows.
- **Resolution.** Switched to `record_level_agg=mean_logit` (mean of logits) and added robust aggregation options (e.g., `trimmed_mean_prob` with `record_level_trim`).
- **Proofs (exports + code).** The final runs are exported with `exp.record_level_agg=mean_logit` (Listing 1) and the `predictions_test_record.csv` files contain the `agg_used_record` column (artifacts exported by `src/models/time_varying_model.py`). The associated record-level scores are reported in Table 1 (Macro-F1 record = 0.840 ± 0.026).

6.3 Attention Masking (Padding, Causality) and fp16 Stability

- **Problem (reported).** Risk of leakage through padding tokens if the attention mask is not applied symmetrically (queries and keys). In fp16, using large negative constants (e.g., -10^9) can cause overflows.
- **Resolution.** Consistent masking of queries *and* keys (padding) and a *dtype-safe* fill value. In practice, EEG runs are executed with `precision=32` on Windows.
- **Proofs (exports + code).** Implementation in `src/models/utils_transformer.py` (functions `_build_attn_mask` and `_get_mask_fill_value`). The exported configuration uses `exp.precision=32` (Listing 1).

6.4 Optimization / Regularization (Stability and Inter-Fold Variance)

- **Problem (reported).** Early runs were sensitive to overfitting (fixed LR, little regularization, and instabilities in fp16).
- **Resolution.** Regularization via `weight_decay=1e-4`, `cosine_warmup` scheduler (warmup `warmup_pct=0.05`), and `label_smoothing=0.05`. These choices are integrated into the CT EEG presets.
- **Proofs (exports).** These hyperparameters appear in the condensed configuration (Listing 1). The associated CV metrics are reported in Tables 2 and 1.

6.5 Distribution of Subject Splits

- **Problem (reported).** Variations in class distribution between train/val/test at the subject level (inter-split shift) likely to increase inter-fold variance.
- **Resolution (reported).** Introduction/activation of `dataset.stratify_subject_split=True` to stabilize class proportions during the construction of subject folds.
- **Current state (verified in code).** The parameter is present and exported (`params.json`), but the currently used CV implementation (`src/data/physionet_eegmmidb/dataset.py::_split_subjects`) does not implement explicit stratification. However, a technical basis exists (`_make_subject_folds(...,strata=...)`, `_compute_subject_strata`) and constitutes a minimal improvement (see Section 7).

6.6 Validity Checks and Anti-"Autopilot"

- **Problem.** Risk of "autopilot" models (learning artifacts, implicit leaks) giving artificially high scores.
- **Resolution.** Two minimal checks archived in the exports: (i) label permutation, (ii) input/label decoupling via inter-batch permutation (`CT_SHUFFLE_TRAIN_INPUTS_MODE=inter_batch_decouple`). A key point is that the intra-sequence shuffle does not break the input↔label alignment and may therefore have little impact at the *subject* level (discussion in Section 7 and Section 4.2).
- **Proofs (exports).** Table 9 lists the runs used (IDs + summary). For a 3-class task, a Balanced Accuracy close to 1/3 is consistent with "random chance" behavior.
- **Split check only (reported).** A `CT_ONLY_SPLIT_CHECK=1` mode is used to verify the disjointness of splits without training. The available exports contain the flag, but not the full text logs of the disjointness check; detailed textual verification is therefore *partial* from the exports alone.

6.7 Practical Obstacles (Windows, Hydra, Cache)

- **Windows / DataLoader (reported).** Using multiple workers in PyTorch DataLoaders can cause OpenMP conflicts (`libiomp5md.dll` already initialized). For

stability, the EEG presets fix `dataset.num_workers=0`, `exp.num_workers=0`, `exp.persistent_workers=False`, and `exp.pin_memory=False` (values visible in the exported `params.json`).

- **Hydra inspection (reported).** To display the resolved configuration without executing: `--cfgjob`. This option avoids interpolation errors (e.g., `UnsupportedInterpolationType`) during a quick inspection.
- **Feature cache.** Changes in feature dimensions require rebuilding the cache (`.npz`) to avoid dimension collisions.

7 Discussion and Limitations

Section summary — This section interprets the results, considering the choice of labels and the evaluation protocol, identifies verifiable limitations of the pipeline (unused parameters, missing information, methodological risks), and proposes minimal, actionable improvements.

7.1 Model Strengths

- **Traceability.** The presented results come from structured MLflow exports (`mlflow_exports`) including `metrics.json`, `params.json`, predictions, and confusion matrices.
- **Realistic subject-based evaluation.** The evaluation is conducted in a subject-disjoint mode (objective: to avoid memorizing individual signatures), with metrics reported at the *subject* level.
- **"Train-only" rigor.** The code applies normalization fitted on the training set and prohibits `normalization_scope=global`. Quantile-based labeling is also fitted on the training scores of the fold (`label_strategy=quantile_fold`).
- **Validity checks.** The *label permutation* and *inter-batch decoupling* tests provide a safeguard against "autopilot" learning (Table 9).

7.2 Known Limitations

- **Non-clinical labels.** The {bad, medium, excellent} classes are derived from a quality score calculated on the signal (Section 3.3). They do not constitute a medical annotation; therefore, the results should not be interpreted as clinical performance.
- **QC objective vs. task dataset.** EEGMMIDB is a dataset of motor tasks (execution/imagery). The `T0/T1/T2` markers describe *rest* and *start of movement* events (hands/feet) according to the protocol; however, they are not used to define the QC labels, which remain derived from a signal quality score.
- **"stratify_subject_split" parameter.** The `dataset.stratify_subject_split` parameter is present and exported, but the current CV split implementation (`_split_subjects`) does not perform explicit stratification. A minimal improvement would be to use the `_make_subject_folds(..., strata=...)` function already present in the dataset.

- **Risk of label "tautology".** In the `raw8` configuration, some features (RMS/PTP/line ratio) are close to the metrics used to construct the quality score. This is not a train-test leak per se, but it can make the task close to a direct ranking of the same family of statistics, limiting the "semantic" scope of the label.
- **Cost and deployment.** No latency/memory benchmarks were exported in `mlflow_exports`. Any comparison of inference cost would require a dedicated micro-benchmark (direct measurement on the target machine).

8 Conclusion and Deployment Recommendation

Section summary — This section concludes on the feasibility of transforming the *CausalTransformer* repository into a quality EEG classification pipeline, summarizes the results obtained on EEGMMIDB, and proposes minimal future steps to strengthen validity and reproducibility.

Conclusion. The *CausalTransformer* repository has been converted into a complete 3-class EEG classification pipeline, including a dedicated dataset (EEGMMIDB), train-only heuristic labeling, subject-disjoint evaluation, and auditable MLflow exports. On the final **seed=600** campaign (Table 1), the CT model (`raw8` + `manualcw`) outperforms the internal baselines on the critical *subject-level* metric (Table 4) and shows consistent multi-seed stability (Table 5).

Recommendation (at proof-of-concept stage).

- For academic/demonstrator use: keep CT as the main model to leverage the existing multi-level evaluation and validity check infrastructure.
- For a real-world deployment scenario: explicitly compare CT to a lighter baseline (e.g., EEGNet) on the target machine, with latency/memory measurements (not available in the current exports).

Future work (minimal and verifiable).

- **Subject-stratified CV (implementation).** Utilize `_make_subject_folds(...,stratа=...)` and `_compute_subject_strata` so that `dataset.stratify_subject_split=True` has a measurable effect (then export the per-split distributions via MLflow).
- **Calibration.** Add calibration metrics and figures (e.g., ECE/Brier + reliability diagram) at the window/record/subject levels in the post-analysis, and archive them in `mlflow_exports`.
- **Feature/label ablations.** Quantify the impact of the label-feature alignment via controlled runs (`raw8` vs. `quality3` vs. bandpower, with/without `bandpower_include_quality`) exported and compared at the *subject level*.
- **Additional QC validations.** Add/archive MLflow runs for *subject ID prediction* and *channel ablation* (`runnables/quality_tests.py`) to make these tests auditable (artifacts not included in the export used here).

9 Appendices

9.1 Final Configuration (`hydra`)

Objective. To provide a *minimal* extract of the key hyperparameters of the reference run (fold 0, seed=600), as exported in `params.json`, to allow for a reproducible execution without untraceable copying. **Source.** MLflow export (run `dazzling-pig-862`, file `params.json`).

Listing 1: Condensed configuration (excerpt).

```yaml
dataset:
  name: eegmmidb
  feature_set: raw8
  window_seconds: 2.0
  stride_seconds: 1.0
  max_seq_length: 60
  split_by_subject: True
  n_folds: 5
  fold_index: 0
  normalization_mode: fold_zscore
  normalization_scope: train
  label_strategy: quantile_fold
  quantile_range: [0.33, 0.66]
  label_feature: composite

model:
  name: CT
  multi:
    max_seq_length: 65
    optimizer:
      optimizer_cls: adam
      learning_rate: 0.0003
      weight_decay: 0.0001
      lr_scheduler:
        type: cosine_warmup
        warmup_pct: 0.05
        t_max: 100

exp:
  seed: 600
  max_epochs: 100
  precision: 32
  class_weights_mode: manual
  class_weights: [1.0, 1.5, 1.0]
  label_smoothing: 0.05
  record_level_agg: mean_logit
  subject_level_agg: mean_prob
```

The dataset truncates/packs sequences to 60 time steps; the model is configured with a maximum capacity of 65 (upper bound), which leaves a margin without impacting the observed sequences.

9.2 MLflow Experiment Index

> **Section summary** — This appendix provides an index of the experiments and runs used in the report (stable IDs) as well as the local locations of the exports (reusable without the MLflow UI).

- **CT EEGMMIDB (export)**: `665075068361836799`
  Folder: `mlflow_exports/experiment_665075068361836799_CT_eegmmidb/`
- **Baselines EEGMMIDB (export)**: `886282226387861497`
  Folder: `mlflow_exports/experiment_886282226387861497_baselines_eegmmidb/`

Presented CT runs (seed=600, 5 folds).

- fold 0: `1ebc9c06368b41c9831524d285701843` (`dazzling-pig-862`)

- fold 1: `16521b4c34b140e9aade8717f099261d` (`bold-fawn-723`)
- fold 2: `a8f5bd3a58a94786bf71d821026eacf4` (`masked-foal-764`)
- fold 3: `b7f7e83ab61940f1b99e9213d6c3dc6e` (`fearless-shad-563`)
- fold 4: `d5599a491aed4c30beb8dd6981b410d9` (`dapper-hound-665`)

Option B (seed=700, 5 folds).

- fold 0: `54c985de34014f7daff3ab6378c61cb4` (`likeable-owl-318`)
- fold 1: `b6694b46fa7345aa8157f6be47ae14aa` (`unleashed-bee-731`)
- fold 2: `7da77fac36064dc0bbba4bd40640d70f` (`exultant-bird-298`)
- fold 3: `a974048a273242279bee17753f4de6e4` (`fun-roo-177`)
- fold 4: `07df9328143b46e8b00e8e3357fbc216` (`nebulous-wren-944`)

9.3 Reproducibility (Windows, commands, exports)

Section summary — This appendix records the minimal elements for reproducibility on Windows: OS/Conda, environment variables, cross-validation commands, and MLflow run exports.

OS / environment. The referenced executions come from a Windows Conda environment (`(causaltransformer)`) and CMD commands from `C:\Projects\CausalTransformer`.

Environment variables ("normal run" state).

Listing 2: Variables (excerpt).

```
set "PYTHONPATH=."
set HYDRA_FULL_ERROR=1
set CT_ONLY_SPLIT_CHECK=0
set CT_SHUFFLE_TRAIN_LABELS=0
set CT_SHUFFLE_TRAIN_INPUTS=0
set CT_SHUFFLE_TRAIN_INPUTS_MODE=
```

CV campaign (seed=600, folds 0–4) — Windows CMD commands (faithful excerpt).

Listing 3: CV command (seed=600).

```
set "PYTHONPATH=."
set HYDRA_FULL_ERROR=1

set CT_ONLY_SPLIT_CHECK=0
set CT_SHUFFLE_TRAIN_LABELS=0
set CT_SHUFFLE_TRAIN_INPUTS=0
set CT_SHUFFLE_TRAIN_INPUTS_MODE=

for %i in (0 1 2 3 4) do python runnables\train_multi.py ^
  +experiment=eegmmidb_ct_raw8_seed200_manualcw ^
  dataset.data_dir="C:\Projects\CausalTransformer\data\eegmmidb" ^
  dataset.stratify_subject_split=true ^
  dataset.cache=false ^
  fold_index=%i ^
  exp.seed=600 ^
  dataset.num_workers=0 exp.num_workers=0 ^
  exp.persistent_workers=false
```

Note on experiment names. The name `+experiment=...` is an alias to a Hydra preset and may contain a seed number that does not correspond to the seed actually executed. The effective seed is always the one passed via the command line (e.g., `exp.seed=600`).

Export of MLflow results.

Listing 4: Command (excerpt).

```
set "PYTHONPATH=."
python scripts\export_results.py --experiment-name "CT/eegmmidb" --last-n 20
    --post-analyze True
```

9.4 Additional Figures

Section summary — This appendix gathers useful but non-essential figures for the main body of the report: learning curves and additional confusion matrices (record/window levels). The normalized versions are included below; when available, raw (count) versions also exist in `report_latex/figures/`. All these figures come from `report_latex/figures/` and are either copied from `mlflow_exports` or generated from the exports (script `scripts/make_report_assets.py`).

9.4.1 Training Curves (example, fold 0, seed=600)

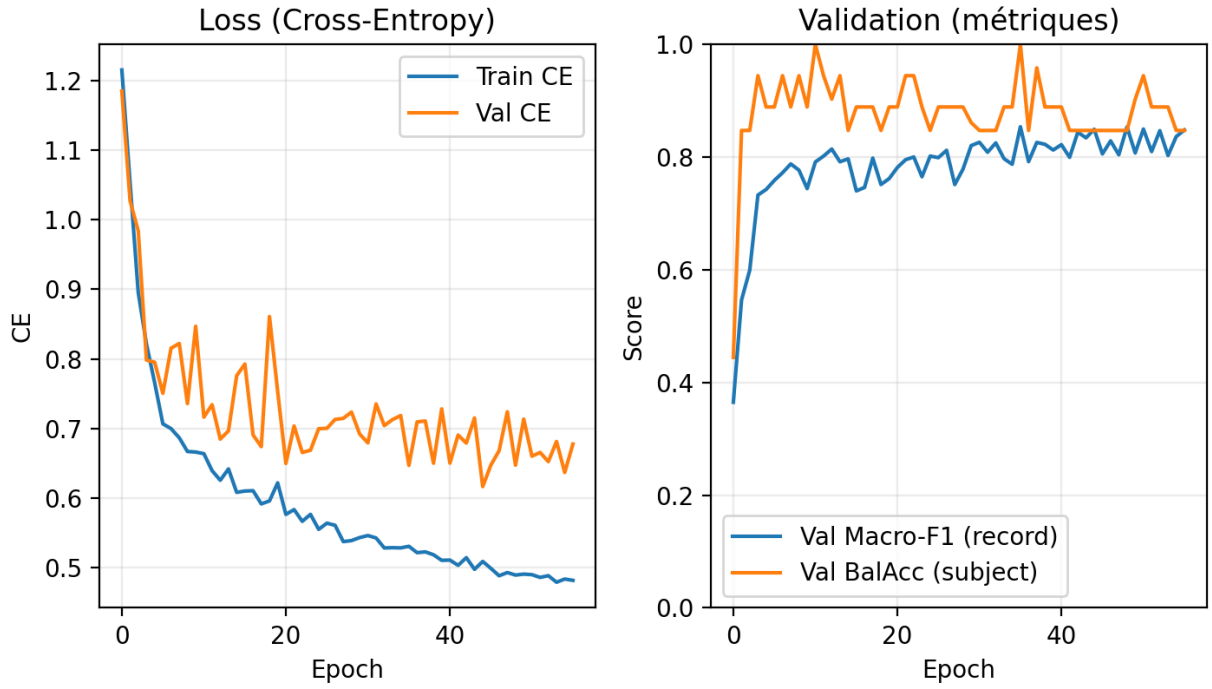
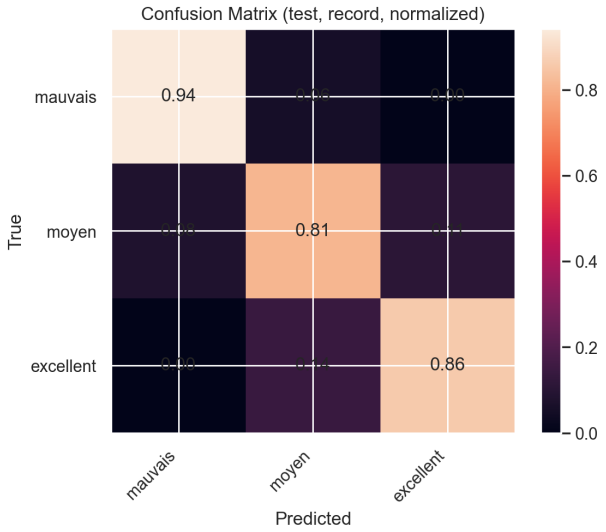
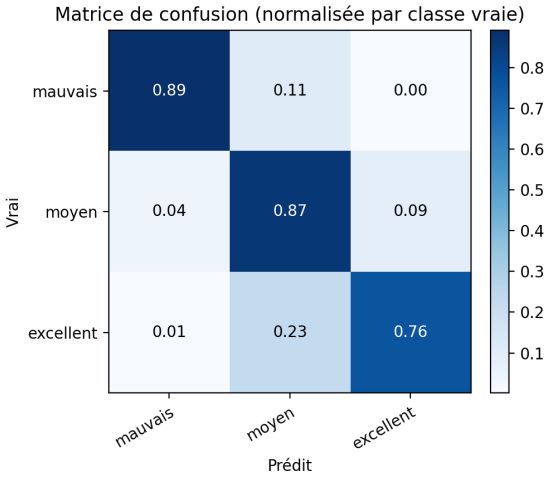


Figure 8: Training/validation curves (fold 0, seed=600) generated from local MLflow logs (`mlruns`).

9.4.2 Additional Confusion Matrices (seed=600)



(a) Record, fold 0, normalized



(b) Record, aggregated 5 folds, normalized

Figure 9: Confusion matrices (*record* level, test, seed=600).

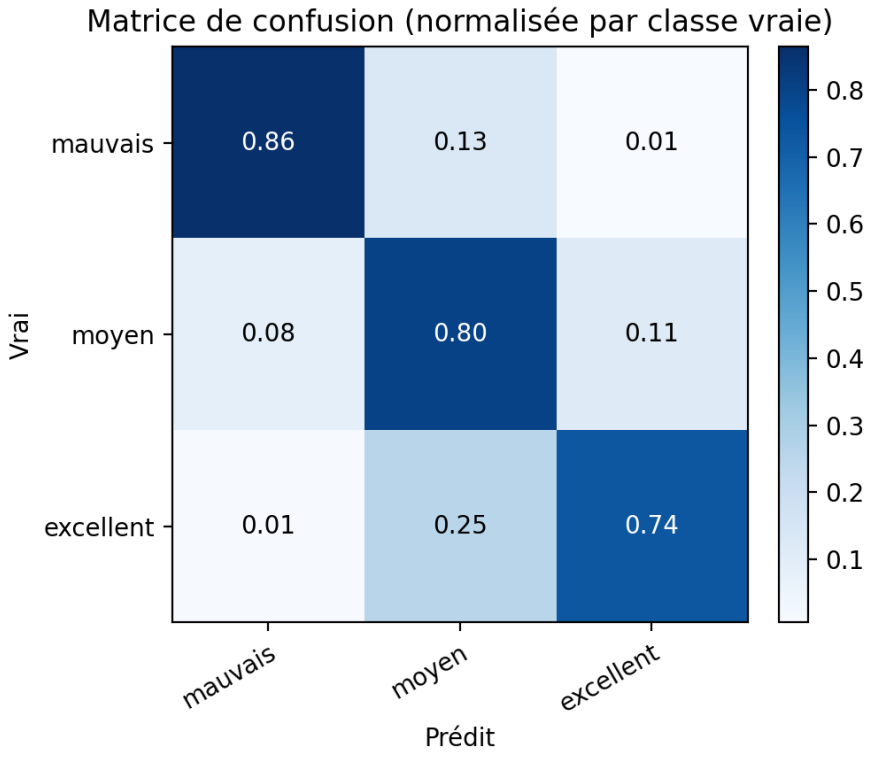


Figure 10: Normalized confusion matrix (*window* level, test, aggregated 5 folds, seed=600).

9.5 Tools & Environment

Section summary — This appendix lists the software environment as documented in the repository (pinned dependencies) and as exported from the Windows Conda environment used to train the final runs. The objective is to make explicit any potential discrepancy between "expected" versions and actually used versions.

- **Pinned dependencies (repository reference)**: e.g., `torch==1.13.1`, `pytorch-lightning==1.4.5`, `hydra-core==1.3.2`, `mlflow==2.12.2`, `mne==1.4.2`.
- **Effective versions (Windows Conda env)**: excerpt below.

Listing 5: Software versions exported from the training environment.

```
python 3.8.20 (default, Oct 3 2024, 15:19:54) [MSC v.1929 64 bit (AMD64)]
platform Windows-10-10.0.26100-SP0
numpy 1.24.3
pandas 2.0.3
mne 1.6.1
mlflow 2.17.2
hydra 1.3.2
torch 2.4.1
lightning 1.4.5
sklearn 1.3.2
```

9.6 Differences vs. Original Repo

Section summary — This appendix summarizes the major differences observed through inspection of the code, configurations, and exports (without git diff).

- **Addition of an EEGMMIDB dataset**: `src/data/physionet_eegmmidb/` (EDF scan, windowing, features, heuristic labels).
- **Switch to classification**: configuration `dim_outcomes=3`, logit outputs + cross-entropy (`src/models/time_varying_model.py`).
- **Neutralization of "shock target" / treatment logic**: adaptation to a task without treatments (`treatment_mode=none`, `dim_treatments=0`).
- **EEG Baselines**: `src/models/baselines.py` and `runnables/train_baselines.py`.
- **Enriched MLflow exports**: multi-level metrics, predictions, confusion matrices, and reports exported (`mlflow_exports`).

9.7 Compliance Checklist (final report)

Section summary — This checklist replaces dependencies on external documents and makes explicit the points covered and remaining limitations, without requiring separate files to be opened.

- **Self-contained (reading).** The reproduction commands (Windows), key hyperparameters, results, and validity checks necessary for understanding are included in this PDF.
- **Dataset citations.** EEGMMIDB/EEGBCI (PhysioNet) is cited with DOI **10.13026/C28G6P** and license **ODC-By 1.0** (Bibliography).
- **Validity checks (anti-leak).** Label permutation and inter-batch decoupling are archived and summarized in Table 9.
- **Bibliography.** Key references (Causal Transformer, EEGMMIDB, BCI2000, PhysioNet, EEGNet) are present without stubs.
- **Remaining limitations (optional).** Additional QC tests (subject ID prediction, channel ablation) are not included in the MLflow export used in this report.

References

[1] Eeg motor movement/imagery dataset (eegmmidb/eegbci). PhysioNet, 2009. URL https://physionet.org/content/eegmmidb/1.0.0/. Version 1.0.0 (9 sept. 2009). Licence : Open Data Commons Attribution License v1.0 (ODC-By).

[2] A. Goldberger, L. Amaral, L. Glass, J. Hausdorff, P. C. Ivanov, R. Mark, J. E. Mietus, G. B. Moody, C. K. Peng, and H. E. Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101 (23):e215–e220, 2000. Citation standard PhysioNet telle que fournie dans la documentation EEGMMIDB locale; RRID:SCR_007345.

[3] Vernon J. Lawhern, Amelia J. Solon, Nicholas R. Waytowich, Stephen M. Gordon, Chou P. Hung, and Brent J. Lance. Eegnet: a compact convolutional neural network for eeg-based brain–computer interfaces. *Journal of Neural Engineering*, 15(5):056013, 2018. doi: 10.1088/1741-2552/aace8c. URL https://doi.org/10.1088/1741-2552/aace8c.

[4] Valentyn Melnychuk. Causaltransformer. Dépôt GitHub (code source), 2022. URL https://github.com/Valentyn1997/CausalTransformer. Consulté le 2026-01-29.

[5] Valentyn Melnychuk, Dennis Frauen, and Stefan Feuerriegel. Causal transformer for estimating counterfactual outcomes. arXiv, 2022. URL https://arxiv.org/abs/2204.07258.

[6] OpenAI. Codex cli. Software and documentation. URL https://developers.openai.com/codex/cli/. Consulté le 2026-01-29.

[7] G. Schalk, D. J. McFarland, T. Hinterberger, N. Birbaumer, and J. R. Wolpaw. Bci2000: A general-purpose brain-computer interface (bci) system. *IEEE Transactions on Biomedical Engineering*, 51(6):1034–1043, 2004. doi: 10.1109/TBME.2004.827072.