System Description Paper
CodeCraft – AIC Phase II Submission

Mohammed A. Metwally

July 24, 2025

Contents

1 Introduction & Problem Statement **2**
1.1 Introduction to the Task 2
1.2 Exploring the Competition's Dataset: First Impressions and Insights 2
1.2.1 General Characteristics 2
1.2.2 EEG Channels and Task-Specific Layout 3
1.2.3 Auxiliary Signals and Signal Quality 3

2 Related Work Challenges **5**

3 Methodology **7**
3.1 High level pipeline Overview 7
3.1.1 MTCformer (V2 , V3) design 9
3.2 Training/inference and Loss Calculation 12
3.2.1 MI training and loss calculation 13
3.2.2 SSVEP training and loss calculation 13
3.2.3 Obtaining Trial-Level Predictions from Windows 13

4 Experimentation Journey **14**

5 Results & Discussion **15**

6 Key Findings & Recommendations **17**

1 Introduction & Problem Statement

1.1 Introduction to the Task

Brain-Computer Interfaces (BCIs) aim to translate human intention into actionable commands by decoding neural activity — a task both inspiring and inherently complex. In this competition, we focus on two foundational paradigms of non-invasive BCI: **Steady-State Visually Evoked Potentials (SSVEP)** and **Motor Imagery (MI)**.

SSVEP is a type of brain response that emerges when a subject gazes at a visual stimulus flickering at a specific frequency. This stimulus evokes electrical oscillations primarily in the **occipital lobe** (back of the brain), matching the flickering frequency. These signals are **strong, periodic, and relatively easy to detect** in controlled settings. However, their strength can vary across users and over time due to fatigue, eye movement, or low visual engagement.

Motor Imagery (MI), on the other hand, involves mentally rehearsing a movement — like imagining moving the left or right hand — without physically executing it. This generates patterns in the **sensorimotor cortex**, particularly detectable over electrodes like C3, C4, and Cz. Unlike SSVEP, MI signals are **subtle, non-rhythmic, and subject-dependent**, relying heavily on the user's concentration and training.

Despite the promise of combining SSVEP and MI to create richer and more flexible BCI systems, **generalizing across users remains a key challenge**. EEG signals vary significantly between individuals due to differences in head anatomy, electrode placement, noise sensitivity, and mental strategies. Many deep learning models tend to **overfit to these idiosyncrasies**, performing well on training subjects but **failing to generalize to unseen users** — especially when test data comes without calibration labels.

This competition pushes us to design a **unified model** that can robustly decode both SSVEP and MI from raw EEG signals — **across subjects, without target labels** — making the challenge one of **transferability and signal disentanglement**. We aim to address this by learning **compact and adaptive representations** of the underlying neural activity, while **disentangling user-invariant and user-specific information**.

1.2 Exploring the Competition's Dataset: First Impressions and Insights

1.2.1 General Characteristics

The dataset used in this competition consists of EEG recordings from 45 healthy male participants with an average age of 20. All data was collected using a sampling rate of 250 Hz. Each subject participated in two different BCI paradigms: Motor Imagery (MI) and Steady-State Visual Evoked Potentials (SSVEP). The data is pre-segmented into trials, with 10 trials per file, and each task is stored in its own directory with clearly labeled `train`, `validation`, and `test` subfolders.

1.2.2 EEG Channels and Task-Specific Layout

Eight scalp electrodes were used for EEG acquisition:

- **Motor Imagery (MI)** focuses on the central and frontal channels: `FZ`, `C3`, `CZ`, `C4`. These are commonly associated with motor cortex activity and are optimal for detecting left/right hand imagery tasks.
- **SSVEP** focuses on posterior and occipital areas: `PZ`, `PO7`, `OZ`, `PO8`. These regions are visually responsive and ideal for capturing steady-state visual evoked potentials.

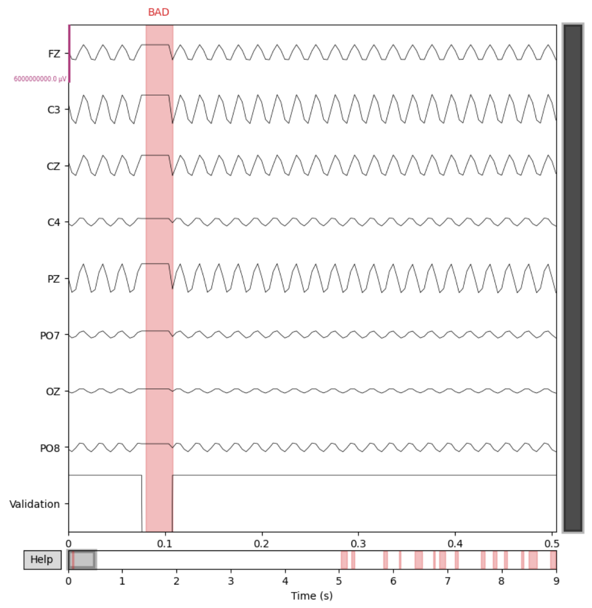
Each EEG file contains concatenated trials, and the labels are separately provided in metadata files like `train.csv` and `validation.csv`. The test set does not include labels. for each task. all of the 8 EEG channels were included.

1.2.3 Auxiliary Signals and Signal Quality

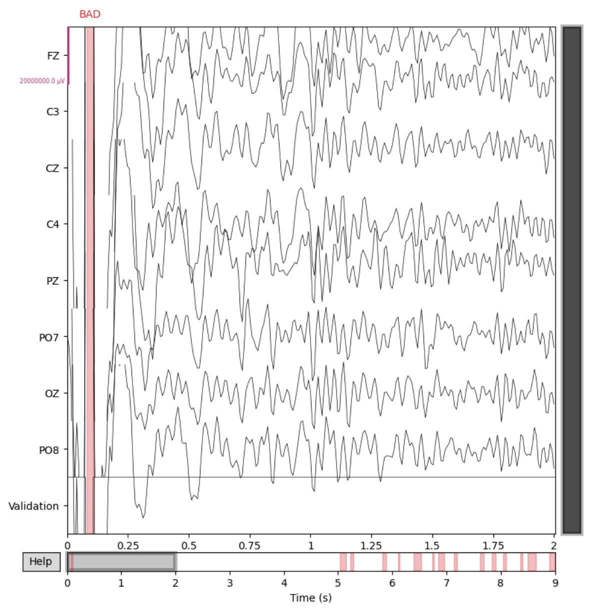
In addition to EEG, the recordings include:

- **Accelerometer (AccX, AccY, AccZ)** and **Gyroscope (Gyro1, Gyro2, Gyro3)** signals — each recorded in 3 axes — useful for detecting head or body movement artifacts during trials.
- **Validation Channel:** A binary signal that marks clean EEG segments. When `Validation = 0`, the corresponding time segment is likely corrupted or of low quality, we treated these with caution.

let's take a quick glance at how the raw data looks. The following visualizations show representative segments from MI (SSVEP data do not differ by much).



(a) Motor Imagery (MI) EEG segment (Raw)



(b) Motor Imagery (MI) EEG segment (After filters)

Figure 1: Raw and filtered EEG signals for Motor Imagery (MI) trials.

On the **left**, we see a segment of a Motor Imagery (MI) trial in its **raw form**. EEG signals from multiple electrodes are displayed, along with **EEG channels**. The **red-marked sections** correspond to intervals flagged as **bad**, which happens when the `Validation` signal drops to 0 — meaning the signal is likely corrupted or unreliable.

These bad intervals appear visually **flattened or smoothed**, we predict that the dataset provider may have applied some form of **linear interpolation**, although the exact method isn't documented.

The EEG signal here looks **very consistent**, almost like a clean, repetitive oscillation — a sign that it might contain low-frequency **power-line interference** (e.g., at 50 Hz), which is quite common in raw EEG recordings.

On the **right**, we show the **same trial** after applying a **bandpass filter (6–24 Hz)** along with a **notch filter (50 Hz)**. we can see how the brain signals are more apparent now.

What Makes This Dataset Challenging?

Working with this dataset proved to be both fascinating and difficult — and for several good reasons.

First: Generalization Across Subjects. We are required to develop models that generalize across subjects. With 35 different individuals performing the tasks — each with their own brainwave patterns, noise levels, and physiological idiosyncrasies — learning robust and consistent representations is far from trivial.

Second: Raw and Unprocessed Signals. The data was provided in a truly **raw**

form, recorded directly from the biosignal amplifier. No preprocessing had been performed: even the signal units (e.g., μV) were unspecified. Making sense of the signals required domain knowledge and hands-on inspection of the waveforms.

Third: Bad or Missing Signal Segments. The dataset includes a `Validation` channel indicating when the signal is unreliable (set to 0). These low-quality regions were surprisingly common — and often appeared in the middle of trials. Many of these segments were either linearly interpolated or outright missing, which made learning temporal patterns more difficult.

Fourth: No Event Markers or Cues. Another limitation is the lack of event markers — we weren't told exactly when the visual cue (to perform the task) appeared. This forced us to either guess or heuristically align the data, increasing the uncertainty in supervision.

Fifth: Relatively Small Dataset. While the dataset covers 35 subjects, the total number of trials per task is just over 2,000 — which is modest by deep learning standards. This makes the risk of overfitting real, especially when designing large architectures or subject-specific pipelines. Even for the fine-tuning task, We were restricted to recording only 20-trials per task to build our personized models. which is out-right far from enough to tune a 400k deep-learning model.

Sixth: Mixed Signal Modalities. On the positive side, the dataset includes motion-related information — relatively clean **accelerometer** and **gyroscope** signals across all three spatial axes. In most cases, these signals were helpful for identifying motion artifacts or improving temporal alignment. Only rarely were these sensors corrupted by excessive motion.

To summarize *These kinds of challenges are exactly what made the competition a genuinely fun and rewarding learning journey. Although it required a fair amount of grappling and experimentation, working through such unique problems was ultimately a rich and memorable experience.*

2 Related Work Challenges

When we first dove into EEG decoding, the landscape was dominated by the classic **signal processing pipelines**. For Motor Imagery (MI), the tried-and-true approach was **Common Spatial Patterns (CSP)** (Ramoser, Muller-Gerking, and Pfurtscheller 2000; Blankertz et al. 2008) followed by *LDA* or *SVM*. CSP can perform remarkably well, but it **often struggles** when noise levels spike or when a new subject shows up.

SSVEP Classics: Canonical Correlation Analysis (CCA) (Yuan et al. 2008) and its Filter Bank variant (FBCCA) (Chen et al. 2015) have long been the go-to for flicker frequency detection in the occipital lobe. Yet, in real-world settings where attention drifts or the signal-to-noise drops, these methods can **quickly lose robustness**.

Artifact Denoising with ICA: Independent Component Analysis (ICA) (Hyvärinen and Oja 2000) is a common go-to for removing artifacts like eye blinks or muscle noise. The catch? It usually requires **manual inspection** of components, which is time-intensive and can accidentally strip out genuine brain activity.

Tree-Based Models: Ensembles like **Random Forests** (Breiman 2001) and **Gradient Boosting Machines** like XGBoost (Friedman 2001), Lighgbm (Friedman 2001) and Catboost (Friedman 2001) can ingest hand-crafted features and handle nonlinear interactions. They shine with clean features but **risk overfitting** when trial counts are low.

Deep Learning from Scratch: EEGNet (Lawhern et al. 2018) proved you could ditch feature engineering entirely, using depthwise and separable convolutions to learn spatial–temporal filters directly from raw EEG. Powerful? Absolutely. But it still needs **large, diverse datasets** to generalize across subjects.

Self-Attention in EEG: Transformer-style models like EEGComformer (Vu et al. 2022) employ self-attention to capture long-range temporal patterns, pushing MI benchmarks further. These architectures are exciting, but they can be **computationally heavy** and **unstable** on smaller EEG datasets.

Domain-Adversarial Training (DANN) (Ganin et al. 2016) tackled the challenge of *domain shift* by encouraging feature representations to be invariant across domains. In EEG, this is especially helpful when training on one subject and testing on another. We incorporate this adversarial training technique in our architecture via a gradient reversal layer.

Adversarial Signal Augmentation (T. Kim et al. 2021) has emerged as a robust method to simulate worst-case perturbations during training. Rather than adding random noise, this technique generates structured perturbations that push the model to learn more resilient features. We use it alongside Gaussian noise to harden our model against real-world disturbances.

Our Key Inspirations:

TabNet's Attentive Transformer (Arik and Pfister 2021) introduced the idea of learnable feature masks with a relaxation-based prior. Although it was built for tabular data, we found its **sparse-mask logic irresistible**. We adopted and adapted this mechanism in a *novel way*, guiding our convolutional branches to focus on diversified components of its inputs.

SSVEPFormer's Convolutional Attention (Morioka et al. 2024) leverages complex-valued convolutions plus a temporal transformer to nail frequency decoding. We borrowed **Much of It's architectural design** and retooled it for both MI and SSVEP tasks, using real-valued inputs for simplicity while preserving the spirit of its **Convolutional Attention** design.

Domain-Adversarial Neural Networks (DANN) (Ganin et al. 2016) gave us a blueprint for **learning features that generalize across domains**—critical in EEG, where subject variability is high. We incorporated a gradient reversal branch into our MI pipeline (MTCFormer-V3), helping the model suppress subject-specific artifacts and learn domain-invariant patterns.

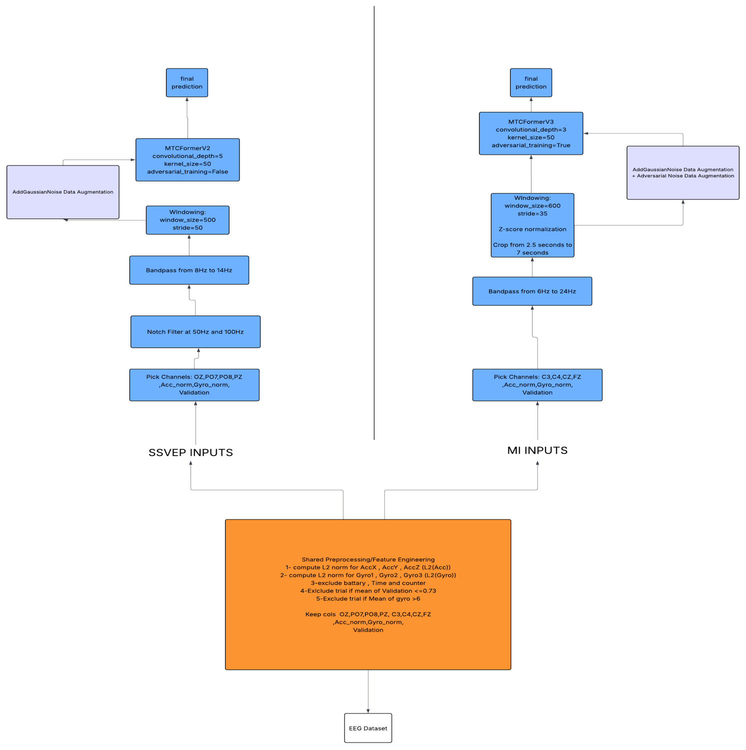
Adversarial Signal Augmentation (T. Kim et al. 2021) inspired us to go beyond classical noise injection. By introducing **targeted perturbations**, we forced our model to sharpen its decision boundaries, improving its robustness under real-world, noisy conditions.

Despite these powerful building blocks, the core challenges remain:

- **Cross-subject generalization**
- **Noisy, raw EEG signal handling**
- **Limited trials per paradigm**
- **Lightweight, interpretable model requirements**

3 Methodology

3.1 High level pipeline Overview



final prediction
MTCFormerV2 convolutional_depth=5 kernel_size=50 adversarial_training=False
AddGaussianNoise Data Augmentation
WIndowing: window_size=500 stride=50
Bandpass from 8Hz to 14Hz
Notch Filter at 50Hz and 100Hz
Pick Channels: OZ,PO7,PO8,PZ ,Acc_norm,Gyro_norm, Validation
SSVEP INPUTS
final prediction
MTCFormerV3 convolutional_depth=3 kernel_size=50 adversarial_training=True
AddGaussianNoise Data Augmentation + Adversarial Noise Data Augmentation
WIndowing: window_size=600 stride=35
Z-score normalization
Crop from 2.5 seconds to 7 seconds
Bandpass from 6Hz to 24Hz
Pick Channels: C3,C4,CZ,FZ ,Acc_norm,Gyro_norm, Validation
MI INPUTS
Shared Preprocessing/Feature Engineering
1- compute L2 norm for AccX , AccY , AccZ (L2(Acc))
2- compute L2 norm for Gyro1 , Gyro2 , Gyro3 (L2(Gyro))
3-exclude battary , Time and counter
4-Exlclude trial if mean of Validation <=0.73
5-Exclude trial if Mean of gyro >6
Keep cols OZ,PO7,PO8,PZ, C3,C4,CZ,FZ ,Acc_norm,Gyro_norm, Validation
EEG Dataset

Figure 2: System full pipeline

The diagram above provides a comprehensive overview of the entire system pipeline—from raw EEG to final prediction. The process is split into two main branches: one for **Motor Imagery (MI)** and another for **Steady-State Visual Evoked Potentials (SSVEP)**. While both pipelines share a set of common preprocessing and feature engineering steps, they diverge in their respective signal processing and model configurations to accommodate the unique characteristics of each paradigm.

At a high level:

- The **MI pipeline** uses our proposed `MTCFormerV3` architecture, which integrates adversarial training and has a convolutional depth of 3.

- The **SSVEP pipeline** uses an earlier version, `MTCFormerV2`, without adversarial training and a convolutional depth of 5.

Shared Preprocessing and Feature Engineering:

Before separating into the MI and SSVEP branches, all data undergoes shared preprocessing to enhance quality and reject noise or invalid samples. These steps are:

1. **Compute L2-norm of accelerometer channels**: For each timepoint, we compute the L2-norm of the 3-axis accelerometer vector:

$$\text{Acc}_{\text{norm}} = \sqrt{\text{Acc}_x^2 + \text{Acc}_y^2 + \text{Acc}_z^2}$$

2. **Compute L2-norm of gyroscope channels**: Similarly, we compute the L2-norm of the gyroscope:

$$\text{Gyro}_{\text{norm}} = \sqrt{\text{Gyro}_x^2 + \text{Gyro}_y^2 + \text{Gyro}_z^2}$$

3. **Exclude trials** based on:
   - Low validation signal (trials with `Validation` ≤ 0.73)
   - Excessive motion (trials where $\text{Mean}(\text{Gyro}_{\text{norm}}) > 6$)

4. **Column selection:** We retain the EEG channels relevant to both tasks:

   Channels kept: $\{\text{OZ}, \text{PO7}, \text{PO8}, \text{PZ}, \text{C3}, \text{C4}, \text{CZ}, \text{FZ}\} \cup \{\text{Acc}_{\text{norm}}, \text{Gyro}_{\text{norm}}, \text{Validation}\}$

SSVEP-Specific Pipeline:

- **Channel Selection:** OZ, PO7, PO8, PZ (posterior-occipital region).
- **Filtering:**
  - Notch filter at 50Hz and 100Hz to remove power-line interference.
  - Bandpass filter from 8Hz to 14Hz to isolate SSVEP frequency content.
- **Windowing:** 2 seconds windows with 0.2 seconds stride.
- **Data Augmentation:** Gaussian noise applied to augment training robustness.
- **Model:** `MTCFormerV2` (convolutional_depth=5, kernel_size=50, no adversarial training).

MI-Specific Pipeline:

- **Channel Selection:** C3, C4, CZ, FZ (motor and frontal regions).
- **Filtering:**

    - Bandpass filter from 6Hz to 24Hz to capture relevant mu and beta rhythms.

- **Trial Cropping:** We crop each trial from 2.5s to 7s to focus on the motor task period.
- **Normalization:** Z-score normalization applied on each trial.
- **Windowing:** roughly 2.3 seconds windows with roughly 0.18 seconds stride.
- **Data Augmentation:** Both Gaussian noise and adversarial noise generation (J. Kim and Ye 2020) are applied.
- **Model:** `MTCFormerV3` (convolutional_depth=3, kernel_size=50, with adversarial training).

3.1.1 MTCformer (V2 , V3) design

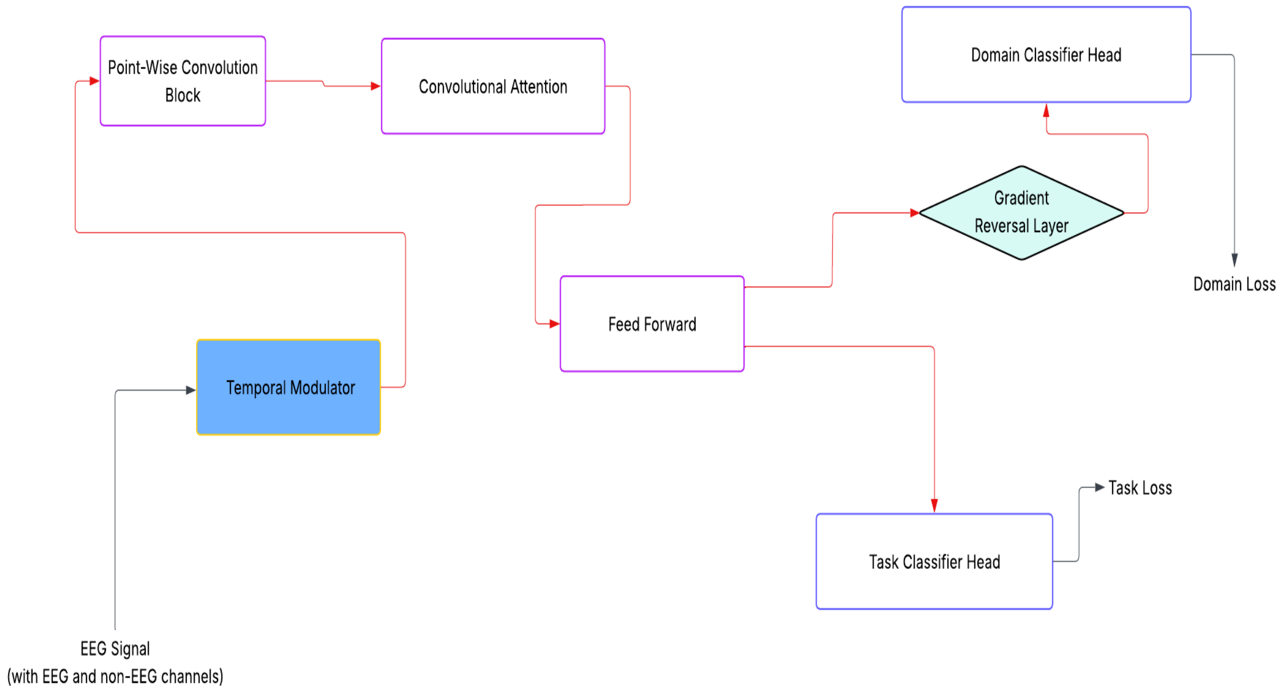
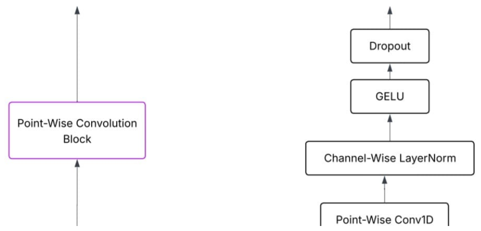


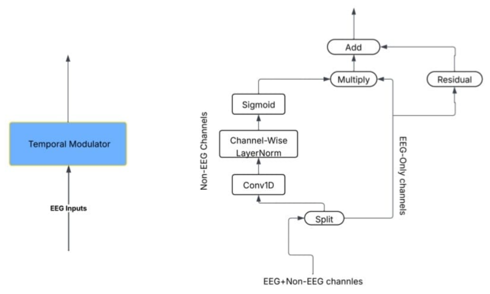
Figure 3: MTCFormer General Design

The above diagram shows the general design of MTCFormer.

Point-wise Convolution: This module performs a 1x1 convolution across the channels, enabling feature mixing across different dimensions without affecting the spatial structure.



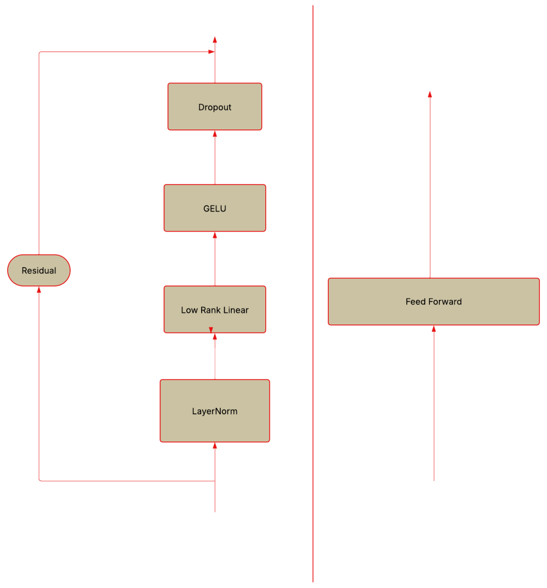
Temporal Modulator: This module modulates the main signal using auxiliary channels, such as accelerometer norms, gyroscope norms, and validation signals, to dynamically adjust the feature representation. A residual connection is added to ensure that the core features of the input are preserved while benefiting from the modulation effect.



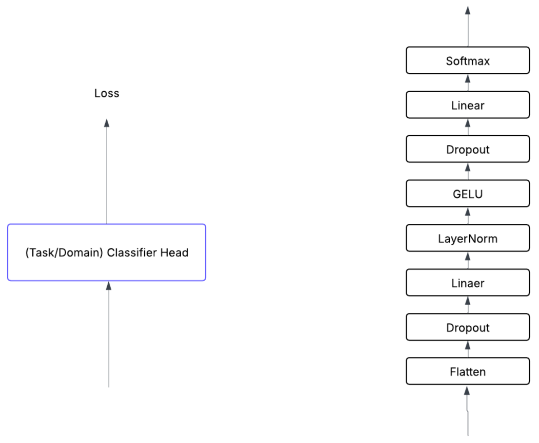
Feed-Forward Block: This block begins with a **Layer Normalization** step to stabilize and normalize the input features. It is followed by a **low-rank linear mapping over time**, where instead of using a single large weight matrix $W \in R^{d \times d}$, we decompose it into two smaller matrices $W_1 \in R^{d \times k}$ and $W_2 \in R^{k \times d}$ such that:

$$\text{Linear}(x) = xW \approx x(W_1 W_2),$$

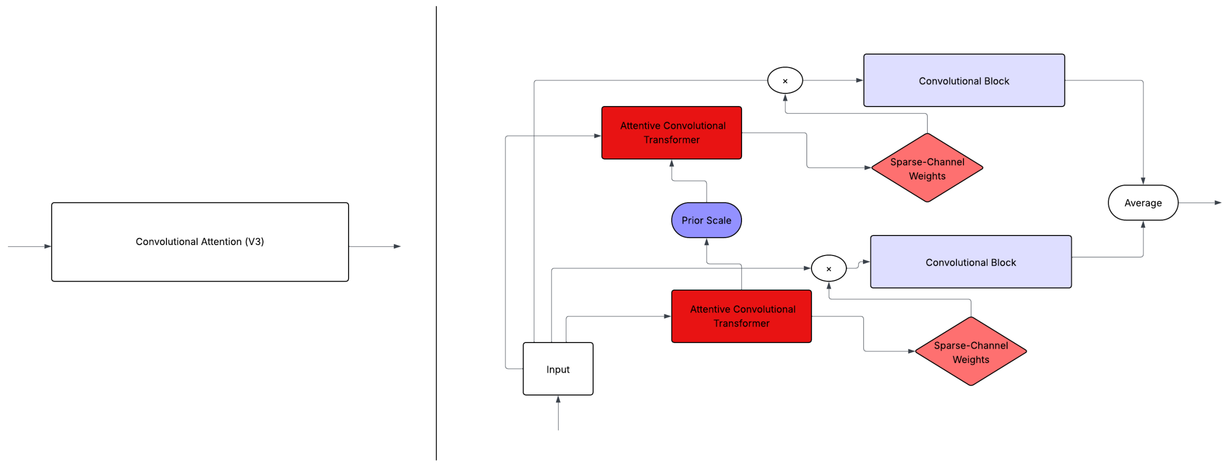
with $k = 100$ in practice. This low-rank approach reduces the parameter count significantly—from approximately $400k$ parameters in the original dense linear layer to just $18k$, with no observed drop in performance.
The linear layer is followed by a **GELU** activation and a **Dropout** layer for regularization. Finally, a **residual connection** is added, enabling deeper networks to train efficiently by preserving the original feature pathways.



Classifier Head: The classifier head consists of two separate classifiers. The first is an *optional subject classifier* that is preceded by a Gradient Reversal Layer (GRL) for domain-adversarial neural network (DANN) training, enabling the model to learn subject-invariant features. The second is the *main task classifier*, which performs the primary prediction task based on the learned representations.



Now we move into the *feature extractor*, which is the Convolutional Attention module. For this part, we designed two separate variants: one for Version 2 (V2) and another for Version 3 (V3) of the model. The rest of the model architecture remains the same across both versions. Let's Start with Version 3



Convolutional Attention V3: This design is inspired by the *attentive transformer* from TabNet(Arik and Pfister 2021), but it has been adapted in a novel way to suit our convolutional feature extractor.

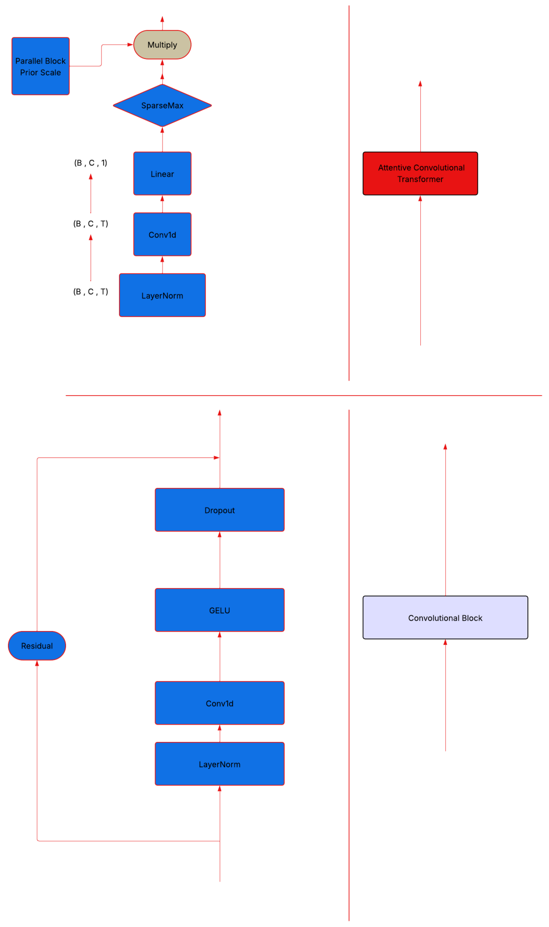
The primary goal of this module is to **encourage the multiple branches of the network to focus on diversified parts of the input, while completely suppressing other irrelevant parts.**

To achieve this, we employ **Sparsemax** (Martins and Astudillo 2016), which produces sparse attention weights by explicitly setting some components to zero, allowing each branch to focus on a distinct subset of features.

We also incorporate the **prior scale** mechanism from TabNet, which recursively updates the prior mask as:

$$P^{(i+1)} = P^{(i)} \odot (1 - M^{(i)}),$$

where $P^{(0)} = 1$ and $M^{(i)}$ is the current branch's mask. This update rule ensures that subsequent branches are encouraged to focus on features not already attended to by earlier branches.



Convolutional Attention V2: The design of this version is intentionally simpler. It does not include the multi-branch sparse attention mechanism introduced in V3. Instead, the **Convolutional Attention V2** is equivalent to using a **single convolutional block** from the V3 design, without the Sparsemax or prior scale mechanism.

3.2 Training/inference and Loss Calculation

For training, we primarily use the **Cross-Entropy Loss** as our base objective. To account for the reliability of different input windows, we weight each sample's loss by a handcrafted weight w_i. The weight w_i for a given window is defined as the mean value of the validation signal over that window, where a higher w_i indicates a more trustworthy window, and a lower w_i indicates a corrupted window that contributes less to the final loss.

3.2.1 MI training and loss calculation

For Motor Imagery (MI) tasks, we additionally employ **Adversarial Domain Adaptation (DANN)** (Ganin et al. 2016) to encourage domain-invariant features. A *domain adaptation loss term* $\mathcal{L}_{\text{DA}}$ is added to the training objective, scaled by a domain factor λ_d that controls the strength of adaptation.

Furthermore, if **adversarial sample generation**(T. Kim et al. 2021) is used (which is the case for MI), the loss from adversarial samples $\mathcal{L}_{\text{ADV}}$ is also added to the total loss with its own scaling factor λ_a.

Thus, the final loss function for MI can be expressed as:

$$\boxed{\mathcal{L}_{\text{MI}} = \sum_{i=1}^{N} w_i \, \mathcal{L}_{\text{CE}}(x_i, y_i) + \lambda_d \, \mathcal{L}_{\text{DA}} + \lambda_a \, \mathcal{L}_{\text{ADV}}}$$

where:

- $\mathcal{L}_{\text{CE}}(x_i, y_i)$ is the cross-entropy loss for the i^{th} sample,
- w_i is the handcrafted weight for the i^{th} sample,
- $\mathcal{L}_{\text{DA}}$ is the adversarial domain adaptation loss,
- $\mathcal{L}_{\text{ADV}}$ is the adversarial sample generation loss,
- λ_d, λ_a are hyperparameters controlling the strength of domain adaptation and adversarial sample contribution, respectively.

3.2.2 SSVEP training and loss calculation

For SSVEP, no adversarial training or domain adaptation is applied. The total loss is simply the weighted sum of cross-entropy losses for each sample:

$$\boxed{\mathcal{L}_{\text{SSVEP}} = \sum_{i=1}^{N} w_i \, \mathcal{L}_{\text{CE}}(x_i, y_i)}$$

3.2.3 Obtaining Trial-Level Predictions from Windows

To aggregate predictions from multiple windows within a single trial, we use a **weighted averaging strategy**. Specifically, each window x_i belonging to a trial is first multiplied by its corresponding handcrafted weight w_i, which is derived from the validation signal (as described above). The final prediction for a trial is computed as:

$$\hat{y}_{\text{trial}} = \frac{\sum_{i=1}^{K} w_i \, \hat{y}_i}{\sum_{i=1}^{K} w_i},$$

where:

- K is the total number of windows in the trial,
- $\hat{y}_i$ is the predicted probability vector or logit output for window i,
- w_i is the validation-derived weight for window i.

This weighted averaging ensures that windows considered more reliable (i.e., with higher validation signal values) contribute more to the final trial-level prediction, while corrupted or low-quality windows have reduced impact.

4 Experimentation Journey

From the very first day of the competition, we stood at a crossroads, staring at two paths. The first path was the "classic" route: handcrafted features and traditional machine learning—methods we were already somewhat familiar with, like gradient-boosted trees or linear models (LDA, SVMs). The second path was a plunge into deep learning, a world we barely understood at the time. Still, we told ourselves: "Let's start with the simpler, more interpretable tools. If we understand the features well, maybe we can control the problem better."

The Traditional ML Phase (Days 1–15). We dedicated the first 15 days to fully exploring the handcrafted feature path. For Motor Imagery (MI), we began with XGBoost, crafting features such as *Common Spatial Patterns (CSP)*, frequency band powers, and power spectral density features. We even leveraged the rich feature set from the `mne-features` library. However, no matter how much we tuned and experimented, performance plateaued at an F1-score of around 0.55.

For SSVEP, we applied classical methods like Canonical Correlation Analysis (CCA) and other handcrafted spectral features, but performance never exceeded 0.35. We attempted leave-K-subject-out cross-validation for more robust evaluation and hyperparameter tuning, but the results remained stubbornly low.

Our SVM "Victory"... That Wasn't. At one point, we tried linear Support Vector Machines (SVMs) and saw an F1-score of 0.63—our first reaction was: *"Wow that easy?"* But inspecting the confusion matrix revealed the truth: the model had simply predicted a single class for all samples. We tried more complex variants, such as SVMs with RBF kernels, and even Linear/Quadratic Discriminant Analysis (LDA/QDA), but the same pattern emerged: they underfit the data. No amount of filtering or feature engineering could push them further. At this point, we concluded that traditional ML alone wouldn't cut it and decided to pivot.

The Hidden Win of Failure Looking back, it's easy to dismiss these 15 days as wasted time. But that would be wrong. Yes, our models didn't work, but those days forged the foundation we desperately needed. We got our first real hands-on experience with signal filtering— windowing stratigies, signal normalization techniques, domain intuition, and the quirks of EEG preprocessing. It was like learning to read the raw "language" of brain signals. Maybe we weren't climbing the leaderboard, but we were gaining something more important:

domain intuition. By the end of those 15 days, we might still have had bad scores, but we were no longer starting from zero.

The Deep Learning Shift. With almost zero prior knowledge in EEG deep learning and very little experience in deeplearning in general, our next move was to dive into existing research. We read about and implemented EEGNet, DeepConvNet, and ShallowConvNet. Immediately, the performance jump was significant: we saw stable F1-scores around 0.6 for MI and 0.5 for SSVEP— a massive improvement compared to the handcrafted methods. This was the moment we realized that the rest of the journey had to focus on deep learning experimentation.

Finding Our Direction. From here, we explored advanced methods like Domain-Adversarial Neural Networks (DANN), EEGConformer, SSVEPFormer, and adversarial sample generation. While transformer-based models like EEGConformer showed promise, they overfit badly on our data. SSVEPFormer, on the other hand, achieved around 0.59 for MI and a stable 0.65 for SSVEP, which was a huge leap. We also found that incorporating adversarial domain adaptation (DANN) stabilized training and improved generalization.

Meanwhile, I recalled prior experience with TabNet from previous tabular competitions, where its attentive transformer inspired the idea of multi-branch attention. We combined these insights to develop our own architecture—**MTCFormer**— tailored specifically for EEG signal representation learning.

Creative Dead Ends That Inspired Us. Along the way, we also experimented with ways to fix corrupted signals where validation signals dropped to zero. We tried an approach we called *"interpolation by reflection"*— a known but rarely used technique in this context. Unfortunately, it didn't yield improvements. This failure led to a new idea: instead of trying to repair the signal manually, why not let the model learn how to fix it? This insight became the basis for our **Temporal Modulator** module in MTCFormer.

Overall, our journey was not linear— it was a mix of trial, error, and unexpected insights. But every failure with traditional ML, every overfitted transformer, and every failed handcrafted "fix" guided us toward building MTCFormerV2 (for SSVEP) and the final MTCFormerV3 (for MI).

5 Results & Discussion

In our experiments, we evaluated the performance of **MTCFormerV2** and **MTCFormerV3** alongside a variety of baseline and competitive methods. These included traditional machine learning models such as **LightGBM**, **XGBoost**, **SVM**, **LDA**, and **QDA**, as well as deep learning architectures like **DeepConvNet**, **ShallowConvNet**, and **EEGConformer**.

To ensure a fair comparison, we used the **official competition split**, which divides the dataset into training, validation, and test sets. All models were trained on the training set, validated on the validation set, and evaluated on the held-out test set. This consistent setup allowed us to directly compare performance across methods similar conditions.

To be able to compare the test set accuracy we fixed the ssvep classifier to the best classifier we had.

booktabs

Table 1: Validation and Test Accuracy for MI Classification Models. Test results are reported for the best-performing SSVEP-MI model pairing.

Model	**Run 1 (Val)**	**Run 2 (Val)**	**Run 3 (Val)**	**Best Test F1**
MTCFormerV2	78.4%	74%	69.2%	**71.2%**
MTCFormerV3	70%	62.5%	68%	**72.3%**
DeepConvNet	55.6%	60.5%	52.8%	62.1%
ShallowConvNet	56.2%	58.3%	54.1%	untested
SVM (Linear)	63.3%	63.3%	63.3%	35.0%
SVM (RBF)	53.2%	53.2%	53.2%	42%
LDA	62.8%	62.8%	62.8%	untested (trivial solution)
QDA	62.8%	62.8%	62.8%	untested (trivial solution)
XGBoost	56.3%	56.3%	56.3%	51%

Interestingly, while **MTCFormerV2** sometimes outperformed **V3** in terms of **training stability**—exhibiting *smoother convergence*—**MTCFormerV3** consistently achieved **higher test accuracies**. In addition to its strong performance, **V3 introduces a pathway for greater interpretability** through its use of **sparse attention gates** (see subsubsection 3.1.1), which produce *meaningful attention masks* capable of highlighting the **spatial relevance**.

An additional benefit of this approach is its **flexibility**: the same sparse attention mechanism can be extended to operate over the **temporal dimension**, allowing the model to *selectively focus on critical time windows* in the EEG signal. Moreover, there is considerable room for design choices within the attention mechanism itself—one can apply **sigmoid**, **softmax**, or even trainable sparsity-inducing mechanisms such as the **Hard Concrete distribution** Louizos, Welling, and Kingma 2018, which approximates L0 regularization using a *soft hard-sigmoid. This was a promising direction we intended to explore but did not have enough time to implement within the competition timeline.*

This makes **MTCFormerV3 a compelling choice** for both **performance-driven** and **explainability-focused** BCI applications. We believe there are so many unexplored yet promising implementation's choices that would make it even more interpretable. It was the best we had for MI. However in SSVEP. It didn't yeild extra gains.

Now moving on to the SSVEP results, where we evaluate the fixed MI classifier (MTC-FormerV3) paired with various SSVEP models:

Table 2: Validation and Test Accuracy for SSVEP Classification Models. Test results are reported for the best-performing MI–SSVEP model pairing (fixed MI: MTCFormerV3).

Model	**Run 1 (Val)**	**Run 2 (Val)**	**Run 3 (Val)**	**Best Test F1**
SSVEPFormer + Complex Repr.	72.1%	–	–	**69.2%**
MTCFormerV2	70.0%	64.3%	66.8%	**72.3%**
SVM (Linear)	33.0%	–	–	(trivial solution)
LDA	34.0%	–	–	(trivial solution)
XGBoost	42.0%	–	–	59.2%

6 Key Findings & Recommendations

Our experiments reveal several important insights about the performance of various models across both MI and SSVEP tasks.

1. MI Models:
MTCFormerV3 showed the strongest test generalization among all MI classifiers, despite having slightly lower average validation accuracy compared to MTCFormerV2.

2. SSVEP Models:
When the MI classifier was fixed to MTCFormerV3, the best results came from MTC-FormerV2. SSVEPformer + Complex representations was far better in Validation Accuracy but lower in test set accuracy.

Recommendations:

- **Investigate Sparse Gating Mechanisms:** Future work should explore the role of sparse gating, particularly the use of L_0 regularization (via the concrete distribution) to create more selective and interpretable pathways within the model. This is especially promising for time-dependent feature selection in the MI branch.
- **Apply Sparse Attention Over Time:** Incorporating attention mechanisms over the temporal axis may enhance the model's ability to focus on informative segments of the signal. However, due to stability issues with `sparsemax` on long sequences, we recommend using L_0-based relaxations for learning time-wise sparsity.
- **Expand the Test Set:** Given the observed variance in test performance across models, we recommend increasing the test set size to better assess generalization and reduce evaluation noise. A larger and more diverse test set would provide stronger evidence of model robustness under realistic deployment scenarios.